Boosting Authenticated Encryption Robustness with Minimal Modifications

Tomer Ashur[1](✉), Orr Dunkelman[2], and Atul Luykx[1,3]

[1] imec-COSIC, KU Leuven, Leuven, Belgium
{tashur,atul.luykx}@esat.kuleuven.be
[2] University of Haifa, Haifa, Israel
orrd@cs.haifa.ac.il
[3] Department of Computer Science, University of California, Davis, One Shields Ave, Davis, CA 95616, USA

Abstract. Secure and highly efficient authenticated encryption (AE) algorithms which achieve data confidentiality and authenticity in the symmetric-key setting have existed for well over a decade. By all conventional measures, AES-OCB seems to be the AE algorithm of choice on any platform with AES-NI: it has a proof showing it is secure assuming AES is, and it is one of the fastest out of all such algorithms. However, algorithms such as AES-GCM and ChaCha20+Poly1305 have seen more widespread adoption, even though they will likely never outperform AES-OCB on platforms with AES-NI. Given the fact that changing algorithms is a long and costly process, some have set out to maximize the security that can be achieved with the already deployed algorithms, without sacrificing efficiency: ChaCha20+Poly1305 already improves over GCM in how it authenticates, GCM-SIV uses GCM's underlying components to provide nonce misuse resistance, and TLS1.3 introduces a randomized nonce in order to improve GCM's multi-user security. We continue this line of work by looking more closely at GCM and ChaCha20+Poly1305 to see what robustness they already provide over algorithms such as OCB, and whether minor variants of the algorithms can be used for applications where defense in depth is critical. We formalize and illustrate how GCM and ChaCha20+Poly1305 offer varying degrees of resilience to nonce misuse, as they can recover quickly from repeated nonces, as opposed to OCB, which loses all security. More surprisingly, by introducing minor tweaks such as an additional XOR, we can create a GCM variant which provides security even when unverified plaintext is released.

Keywords: Authenticated encryption · Robust · AES · OCB · ChaCha20 · Poly1305 GCM · RUP

1 Introduction

Authenticated encryption (AE) is well established within the research community as the method to achieve confidentiality and authenticity using symmetric keys. Initially introduced as a response to a need in practice [6,7], it has caught on in recent years. As a result, AE is used in many different environments, each with their own security and efficiency requirements. For this reason, the ongoing CAESAR competition [20], which aims to identify the next generation of authenticated encryption schemes, drafted three use cases as guides to what AE schemes should target: lightweight, high-performance, and defense in depth.

Within the high-performance category, OCB [43,64,65] is one of the most competitive AE schemes. Over ten years old, it is well known for its speed, and theoretically achieves the best performance when measured in block cipher calls. Although OCB has been standardized [1,44], adoption has remained limited, for which its patents are usually assumed to be the main cause.

Instead, GCM [49] was chosen as the baseline algorithm with which to compare in the CAESAR competition. GCM is widely adopted and standardized [1,24], and although it remains slower than OCB due to the additional universal hash on the output, it is getting more competitive as a result of improved hardware support [30]. ChaCha20+Poly1305 [12,13,56] is a popular alternative for settings where AES-NI is not implemented.

OCB, GCM, and ChaCha20+Poly1305 all target the high-performance category. Other than the fact that GCM and ChaCha20+Poly1305 are already widely adopted, and setting aside differences between using AES versus ChaCha20, from a conventional point of view there seems to be little reason to prefer them over OCB.

1.1 Robust Algorithms

The increased adoption of AE has been accompanied by an improved understanding of the limits of AE security within the research community. Even though OCB, GCM, and ChaCha20+Poly1305 are secure as proved in the conventional models (relative to their underlying primitives), questions often arise as to how robust they are once one of the assumptions in those models no longer holds. Already in 2002, Ferguson [25] pointed out that with a birthday bound attack on OCB one can mount forgeries, and Joux [42] illustrated with his “forbidden attack” how one can similarly construct forgeries for GCM after a repeated nonce. Furthermore, many have expressed concerns with the improved effectiveness of multi-key brute-force attacks [14,16,17,21,28] when applied to widely deployed algorithms.

Given the fact that modifying and deploying algorithms requires significant effort, and that the longer algorithms are used, the more their components are optimized, there has been interest in finding minimal modifications to deployed algorithms so that they are robust to settings which break one of the assumptions of the conventional security definitions. For example, TLS added extra nonce randomization to combat easier key-recovery attacks in the multi-key setting, which was later analyzed by Bellare and Tackmann [11]. The combination of ChaCha20+Poly1305 is neither a direct application of Encrypt-then-MAC [6] nor a copy of GCM: the authentication key used for Poly1305 is changed for *every* message, thereby preventing attacks which make GCM fragile. Going a step further, worries about nonce misuse in GCM have led Gueron and Lindell to use the components underlying GCM in order to create GCM-SIV [31], an algorithm that provides best possible security even when nonces are repeated. The common theme among these modifications is to squeeze as much security out of the schemes without sacrificing efficiency.

1.2 Release of Unverified Plaintext

Previous modifications have focused on providing additional security in the multi-key setting, or when nonces are repeated. However, other robust security properties, such as security with variable-length tags [63], under distinguishable decryption failures [19], or under release of unverified plaintext [3] are equally desirable. The CAESAR competition's use case describing defense in depth lists authenticity and limited confidentiality damage from release of unverified plaintexts (RUP) as desirable properties [15].

One of the advantages of schemes secure under release of unverified plaintext is that they provide another line of defense with faulty implementations: if an implementation for whatever reason fails to check authenticity, then RUP-confidentiality guarantees that if the ciphertext did not originate from the sender or was modified en route, the resulting decrypted plaintext will look like garbage. Furthermore, there are settings where a RUP-secure AE scheme provides desirable properties beyond confidentiality and authenticity; in Appendix C we explain informally how our construction can be used to efficiently prevent the crypto-tagging attack in Tor, which is an attack on user anonymity.

State-of-the-art research might give the impression that achieving RUP security by minimally modifying existing schemes is out of reach: all designs providing such security either require significant changes, a completely new design, or an additional pass, making the schemes slower and adding design complexity. This is because so far the only solutions provided are essentially variable-input-length (VIL) ciphers [8], which can be viewed as block ciphers that can process arbitrarily long messages. However, VIL ciphers are "heavy" constructions, requiring often three or more passes over the plaintext in order to ensure sufficient mixing, or relying on subtle design choices to achieve security.

1.3 Contributions

We continue the line of research on robust AE design by exploring properties and variants of OCB, GCM, and ChaCha20+Poly1305 which go beyond the conventional view of AE.

Our first contribution focuses on analyzing the difference in nonce robustness provided by OCB, GCM, and ChaCha20+Poly1305, to provide a framework complementing the work of others [18,25,42,53]. The conventional nonce misuse models are very black and white about security: GCM and ChaCha20+Poly1305 do not provide security under nonce misuse since an adversary can determine the XOR of two plaintexts when both are encrypted under the same nonce.

However, what the conventional security models do not capture is that this insecurity affects only the involved plaintexts and does not "spill" onto others. If, for example, a faulty implementation repeats a nonce for a pair of plaintexts and then changes it correctly, confidentiality is only compromised for the plaintexts in the pair, and not for future plaintexts. In some sense, GCM (with 96 bit nonces) and ChaCha20+Poly1305 allow one to gracefully recover from re-used nonces by making them unique again, leading us to formalize such a definition, *nonce-misuse resilience*: plaintexts encrypted under unique nonces remain compartmentalized even when other plaintexts are compromised.

Within this model we establish that OCB is not resilient to nonce misuse, confirm that GCM with 96 bit nonces only provides confidentiality resilience, and that ChaCha20+Poly1305 provides both authenticity and confidentiality resilience, thereby formally showing that ChaCha20+Poly1305's choice to depart from both the Encrypt-then-MAC and GCM designs boosts robustness to nonce misuse. Inspired by this result, one can also tweak GCM to achieve the same level of nonce misuse resilience by applying Minematsu and Iwata's composition [53].

Our second, more surprising contribution is a minor modification to GCM which achieves both RUP confidentiality and authenticity, which neither OCB, GCM, nor ChaCha20+Poly1305 currently provide. Our design approach is generic, meaning it can add RUP security to a general class of encryption schemes. The core idea is to use a digest of the ciphertext to "hide" the nonce in such a way that recovering it properly requires that no change was made to the ciphertext. As a result, if a change *did* occur, it would affect the nonce, which, when used by the decryption algorithm, would decrypt the ciphertext into meaningless data.

2 Related Work

Our approach to analyzing nonce misuse differs from the line of research on *online* nonce misuse resistance [4,27,36], which seeks to analyze schemes which are not able to provide the best possible robustness to nonce misuse [66], but are able to guarantee more than nonce misuse resilience. Böck, Zauner, Devlin, Somorovsky, and Jovanovic [18] investigate the practical applicability of nonce-misusing attacks in TLS by searching for servers which repeat nonces with GCM.

Besides nonce misuse, another extension to the basic AE security model considers what happens when decryption algorithms may output multiple decryption errors [19]. Further research explored the security of known AE schemes when their decryption algorithms release partially decrypted plaintext before verification is complete [3], also known as the *release of unverified plaintext* (RUP) model. Both the multiple decryption error and RUP models were unified by Barwell, Page, and Stam [5] in the *subtle AE* framework, by using a "leakage" function which captures information leaked via a side channel. The "leakage" function represents any information that can be received through additional channels. Hoang, Krovetz, and Rogaway introduce the concept of "Robust AE" (RAE) [35] which formalizes one of the strongest types of security that an AE scheme can satisfy. Our use of the term "robust" describes a gradient, in which RAE represents the most robust form of AE, and conventional definitions the most basic form.

Imamura, Minematsu, and Iwata [37] show that ChaCha20+Poly1305 maintains authenticity in the RUP setting.

We follow Shrimpton and Terashima [71] in taking a modular approach to the problem of adding RUP security to encryption schemes, by first providing a solution in the most general form possible, and then providing an instantiation. Furthermore, our construction is similar to the lower half of Shrimpton and Terashima's PIV construction. However, their goal is to achieve something similar to a VIL cipher, which we argue might be overkill in some scenarios. Note that combining SIV [66] with our construction would result in a solution very similar to PIV. RIV [2] is another construction which takes a modular approach in designing a robust AE scheme.

For a survey on ways to construct VIL ciphers, see Shrimpton and Terashima's paper [71]. All the previous methods are generic approaches to designing VIL ciphers, although there are dedicated approaches as well, such as AEZ [35], which in fact aims for RAE.

Hirose, Sasaki, and Yasuda [33] presented a construction similar to ours. However, their construction only accounts for changes over the tag, rather than the entire ciphertext, hence their solution only provides limited robustness and would, for example, not prevent the Tor crypto-tagging attack described in Appendix C. In recent work, Hirose, Sasaki, and Yasuda [34] introduce constructions which do account for changes over the entire ciphertext, and focus on formalizing how such AE constructions make verification unskippable.

3 Preliminaries

3.1 Notation

The set of strings of length not greater than x bits is $\{0,1\}^{\leq x}$, and the set of strings of arbitrary length is $\{0,1\}^*$. Unless specified otherwise, all sets are subsets of $\{0,1\}^*$. If $X, Y \in \{0,1\}^*$, then $|X|$ is the length of X, and $X \parallel Y$ and XY denote the concatenation of X and Y.

Let ε denote the empty string, and let 0^n denote the n-bit string consisting of only zeros. Given a block size n, the function $\mathsf{len}_n(X)$ represents the length of X modulo 2^n as an n-bit string, and $X0^{*n}$ is X padded on the right with 0-bits to get a string of length a multiple of n. If $X \in \{0,1\}^*$, then $|X|_n = \lceil |X| / n \rceil$ is X's length in n-bit blocks. The operation

$$X[1]X[2]\cdots X[x] \xleftarrow{n} X \tag{1}$$

denotes splitting X into substrings such that $|X[i]| = n$ for $i = 1, \ldots, x-1$, $0 < |X[x]| \leq n$, and $X[1] \| X[2] \| \cdots \| X[x] = X$.

The set of n-bit strings is also viewed as the finite field $GF(2^n)$, by mapping $a_{n-1} \ldots a_1 a_0$ to the polynomial $a(\mathtt{x}) = a_{n-1} + a_{n-2}\mathtt{x} + \cdots + a_1\mathtt{x}^{n-1} + a_0\mathtt{x}^{n-1} \in$

$GF(2)[\mathrm{x}]$, and fixing an irreducible polynomial which defines multiplication in the field. For $n = 128$, the irreducible polynomial is $1 + \mathtt{x} + \mathtt{x}^2 + \mathtt{x}^7 + \mathtt{x}^{128}$, the one used for GCM.

The function $\mathsf{int}(Y)$ maps the j bit string $Y = a_{j-1} \dots a_1 a_0$ to the integer $i = a_{j-1}2^{j-1} + \dots + a_1 2 + a_0$, and $\mathsf{str}_j(i)$ maps the integer $i = a_{j-1}2^{j-1} + \dots + a_1 2 + a_0 < 2^j$ to the j bit string $a_{j-1} \dots a_1 a_0$. Let $\mathsf{inc}_m(X)$ denote the function which adds one modulo 2^m to X when viewed as an integer:

$$\mathsf{inc}_m(X) := \mathsf{str}_m(\mathsf{int}(X) + 1 \bmod 2^m) .$$

Define $\mathsf{msb}_j(X)$ to be the function that returns the j most significant bits of X, and $\mathsf{lsb}_j(X)$ the j least significant bits.

For a keyed function defined on a domain $\mathsf{K} \times \mathsf{X}$, we write $F(K, X)$ and $F_K(X)$ interchangeably. If the function has three inputs, $\mathsf{K} \times \mathsf{N} \times \mathsf{X}$, then the second input will often be written as a superscript, $F(K, N, X) = F_K^N(X)$. If $E : \{0,1\}^n \to \{0,1\}^m$ is a function, then the notation

$$F \leftarrow E(C \parallel \cdot) \tag{2}$$

defines F to be the function from $\{0,1\}^{n-|C|}$ to $\{0,1\}^m$ which maps an element $X \in \{0,1\}^{n-|C|}$ to $E(C \parallel X)$.

The expression $a \overset{?}{=} b$ evaluates to $\top$ if a equals b, and $\perp$ otherwise.

3.2 Adversaries and Advantages

An adversary $\mathbf{A}$ is an algorithm which interacts with an oracle O. Let $\mathbf{A}^O = 1$ be the event that $\mathbf{A}$ outputs 1 when interacting with O, then define

$$\underset{\mathbf{A}}{\Delta}(f\,;\,g) := \left| \mathbf{P}\left[\mathbf{A}^f = 1\right] - \mathbf{P}\left[\mathbf{A}^g = 1\right] \right| , \tag{3}$$

which is the advantage of $\mathbf{A}$ in distinguishing f from g, where f and g are viewed as random variables. The notation can be extended to multiple oracles by setting $O = (O_1, \dots, O_\ell)$.

We assume that all keyed functions do not change their output length under different keys, that is, $|F_K(X)|$ is the same for all $K \in \mathsf{K}$. Given a keyed function F, define $\$_F$ to be the algorithm which, given X as input, outputs a string chosen uniformly at random from the set of strings of length $|F_K(X)|$ for any key K. When given the same input, $\$_F$ returns the same output. Often $\$_F$ is called a *random oracle.*

3.3 Authenticated Encryption Schemes

The syntax for conventional authenticated encryption (AE) schemes specifies an encryption and decryption algorithm, where the decryption algorithm may output either plaintext or a single, pre-defined error symbol. Formally, an AE scheme is a tuple of functions — encryption Enc and decryption Dec — where

$$\mathsf{Enc} : \mathsf{K} \times \mathsf{N} \times \mathsf{M} \to \mathsf{C}\,, \tag{4}$$

$$\mathsf{Dec} : \mathsf{K} \times \mathsf{N} \times \mathsf{C} \to \mathsf{M} \cup \{\bot\}\ , \tag{5}$$

with K the keys, N the nonces, M the messages, C the ciphertexts, and $\bot$ an error symbol not contained in M, which represents verification failure. It must be the case that for all $K \in \mathsf{K}$, $N \in \mathsf{N}$, and $M \in \mathsf{M}$,

$$\mathsf{Dec}_K^N(\mathsf{Enc}_K^N(M)) = M \quad . \tag{6}$$

AE schemes must provide both chosen-ciphertext confidentiality and authenticity. The AE advantage of adversary $\mathbf{A}$ against $\Pi = (\mathsf{Enc}, \mathsf{Dec})$ is

$$\mathsf{AE}_{\Pi}(\mathbf{A}) := \underset{\mathbf{A}}{\Delta}\left(\mathsf{Enc}_K, \mathsf{Dec}_K\,;\,\$_{\mathsf{Enc}}, \bot\right)\ , \tag{7}$$

where $\mathbf{A}$ is *nonce-respecting*, meaning the same nonce is never queried twice to Enc. Nonces may be repeated with Dec. Furthermore, $\mathbf{A}$ cannot use the output of an O_1^N query as the input to an O_2^N with the same nonce N, otherwise such queries result in trivial wins.

4 Resilience to Nonce Misuse

Rogaway and Shrimpton [66,67] formalize the best possible security when adversaries may re-use nonces. They illustrate how such *nonce misuse resistance* can be achieved using the construction SIV, which was later the inspiration for GCM-SIV [31].

Finding attacks against OCB, GCM, and ChaCha20+Poly1305 which exploit repeated nonces is relatively straightforward. When nonces are repeated, OCB is not much better than ECB mode [57] since one can easily identify when plaintext blocks are repeated across messages in the same block position. In GCM, keystreams are tied to nonces, hence all messages encrypted with the same nonce will use the same keystream, allowing one to recover the XOR of plaintexts; furthermore, authenticity is broken using Joux's forbidden attack [42]. ChaCha20+Poly1305 suffers from similar attacks as GCM. However, looking more closely at the nonce misusing attacks, one can see that the three algorithms behave very differently.

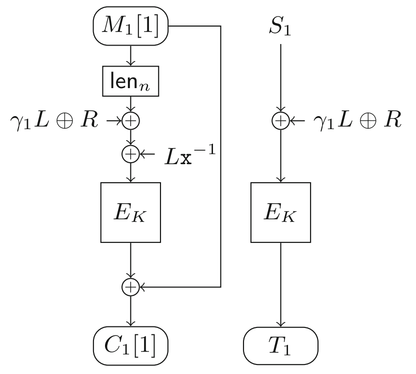
For a description of OCB, GCM, and ChaCha20+Poly1305, and the notation we use see Appendices A.1, A.2, and A.3, respectively.

4.1 OCB Attacks

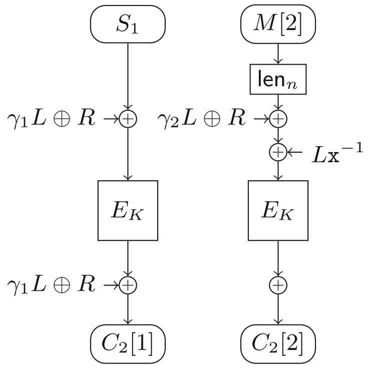
OCB computes two intermediate keys L and R, which it uses to mask the block cipher inputs and outputs. The value L is computed as the output of the block cipher when given 0^n as input, $L := E_K(0^n)$, and remains fixed per key.

The value R changes per nonce, and is computed by encrypting $L \oplus N$ under the block cipher. Finally, the masks are computed as $\gamma_i \cdot L \oplus R$.

Ferguson [25] illustrates how to recover the intermediate key L by finding a collision using a birthday-bound attack, and subsequently shows how to perform forgeries with L for any nonce. In fact, a chosen-plaintext confidentiality attack can be performed as well, by XORing the sequence $(\gamma_1 \cdot L, \gamma_2 \cdot L, \ldots, \gamma_m \cdot L)$ to the plaintext and ciphertext in order to remove dependence on L. This compromises OCB's confidentiality under any nonce N since repeated plaintext blocks in the same message will encrypt to the same ciphertext block. Below we show how to recover L using a nonce-repeating attack.



(a) First query with message M_1 of length one block, which can be chosen arbitrarily.



(b) Second query with message $M_2 = S_1 \| M_2[2]$, where $M_2[2]$ can be arbitrarily chosen. Computation of the tag is not included in the figure.

Fig. 1. An illustration of two queries which would form the first step of the OCB attack. In both cases $R = E_K(L \oplus N)$.

Our attack needs to repeat a particular nonce four times, and works best when $\tau = n$. First, encrypt an arbitrary full-block message M_1 of block length m under nonce N. Receive the corresponding tag T_1 and let S_1 denote the checksum used to generate T_1, so that $T_1 = E_K(S_1 \oplus Z[m])$, where $Z[m] = \gamma_m L \oplus R$. Encrypt another message M_2 of length greater than m blocks under the same nonce N, with the mth block of M_2 equal to $M_2[m] = S_1$. The two queries are depicted in Fig. 1. Note that the corresponding ciphertext block $C_2[m]$ equals

$$E_K(S_1 \oplus Z[m]) \oplus Z[m] \,, \tag{8}$$

and so

$$C_2[m] \oplus T_1 = Z[m] = \gamma_m L \oplus R \,. \tag{9}$$

Encrypt another two messages M_1' and M_2' under nonce N where M_1' has length $m' \neq m$, performing the same steps as above to receive T_1' and $C_2[m']$ such that

$$C_2[m'] \oplus T_1' = Z[m'] = \gamma_{m'} L \oplus R \,. \tag{10}$$

Then L can be recovered from

$$C_2[m] \oplus T_1 \oplus C_2[m'] \oplus T_1' = (\gamma_m \oplus \gamma_{m'})L\,. \tag{11}$$

4.2 Chosen-Plaintext Confidentiality

Although the above attack against OCB requires a nonce to be repeated four times, once those repetitions have occurred, OCB can no longer guarantee security. As already observed by Joux [42], one cannot apply a similar confidentiality attack to GCM, since every new nonce generates a new, roughly independent keystream, and no information can be determined from the plaintext without knowing anything about the keystream. The intuition that no information about the plaintext can be determined from other keystreams can be formalized with the following definition.

Definition 1. *Let* $\mathbf{A}$ *be an adversary and* $(\mathsf{Enc}, \mathsf{Dec})$ *an AE scheme, then the CPA resilience advantage of* $\mathbf{A}$ *against* $(\mathsf{Enc}, \mathsf{Dec})$ *is defined as*

$$\underset{\mathbf{A}}{\Delta}\left(\mathsf{Enc}_K, \mathsf{Enc}_K\,;\, \$_{\mathsf{Enc}}, \mathsf{Enc}_K\right)\,, \tag{12}$$

where $\mathbf{A}$ *may re-use nonces with* O_2*, but it may not re-use nonces with* O_1*, nor may it use a nonce already queried to* O_2 *for an* O_1*-query and vice versa.*

The above definition allows adversaries to perform nonce-reusing attacks with Enc_K, but forces the adversary to win by distinguishing Enc_K from $\$_{\mathsf{Enc}}$ using a nonce-respecting attack, thereby capturing the intuition that a scheme which provides confidentiality resilience to nonce misuse must maintain confidentiality for properly generated nonces, even if the attacker is given the power to re-use other nonces. Note that the form of our definition follows the framework of Barwell, Page, and Stam [5], by separately providing oracles representing the adversary's goal (Enc_K versus $\$_{\mathsf{Enc}}$), as well as oracles representing the adversary's power (the second Enc_K).

In order for schemes to be secure according to the above definition, they must ensure that encryption under one nonce is roughly independent from encryption under another, even if adversaries may gain information by re-using nonces with the encryption oracle. Proving that GCM with 96 bit nonces satisfies this definition up to the birthday bound is straightforward. First note that adversaries which only have access to GCM encryption are essentially interacting with a stream cipher, CTR mode, since unless a nonce is repeated, no two block cipher calls are ever the same. This fact holds even if $E_K(0^n)$ is released to the adversary, since this value is never output by the underlying CTR mode. Then, after applying a PRP-PRF switch, the keystreams generated by the underlying CTR mode under different nonces are independent of each other and uniformly distributed. Therefore interacting with $(\mathsf{Enc}_K, \mathsf{Enc}_K)$ is indistinguishable from interacting with $(\$_{\mathsf{Enc}}, \mathsf{Enc}_K)$. Similar reasoning applies to ChaCha20, assuming the underlying ChaCha20 block function is a good PRF.

Furthermore, OCB does not provide security according to the above definition, because an adversary can make nonce-repeating queries to Enc_K via its O_2 oracle to recover L, and can then perform a confidentiality attack with its other oracle. Similarly, GCM with non-96 bit nonces does not provide nonce resilient confidentiality: since adversaries can recover $E_K(0^n) = L$ (e.g. using Joux's forbidden attack [42]), they can manipulate the counters used in the underlying CTR mode to perform a confidentiality attack, since GHASH_L is applied to the nonce before using it in CTR mode (see e.g. Fig. 5).

4.3 Authenticity

Unlike confidentiality, if a nonce is repeated with GCM, then attackers can recover the authentication key, allowing one to construct forgeries for arbitrary nonces, as illustrated by Joux [42]. Therefore, even though 96-bit-nonce GCM is resilient to nonce misuse when considering chosen plaintext confidentiality attacks, it is not resilient with respect to authenticity. Similarly, OCB is not resilient to nonce misuse with respect to authenticity.

With ChaCha20+Poly1305, authentication keys are changed with every nonce, hence even if a nonce is repeated and the authentication key recovered, an adversary will only be able to forge ciphertexts under the compromised nonce. Such authentication resilience can be formalized as follows.

Definition 2. *Let* $\mathbf{A}$ *be an adversary and* $(\mathsf{Enc}, \mathsf{Dec})$ *an AE scheme, then the authenticity resilience advantage of* $\mathbf{A}$ *against* $(\mathsf{Enc}, \mathsf{Dec})$ *is*

$$\underset{\mathbf{A}}{\Delta}\left(\mathsf{Enc}_K, \mathsf{Dec}_K\,;\, \mathsf{Enc}_K, \perp\right)\,, \tag{13}$$

where if a nonce is used twice with O_1*, then it cannot be used in an* O_2 *query, and adversaries may not query* $O_1^N(M) = C$ *followed by* $O_2^N(C)$*.*

Here the Enc_K oracle is the adversary's power, since it may repeat nonces with that oracle. The challenge of the adversary is to distinguish Dec_K and $\perp$, by constructing a forgery with a nonce which has not been repeated to Enc_K.

The only difference between the above definition and the conventional definition of authenticity is in the restrictions on the adversary: in the conventional definition adversaries must be nonce-respecting, whereas in this definition they may repeat nonces, but may not use repeated nonces to construct forgeries.

One way for schemes to provide authenticity resilience is to ensure that tags verified during decryption under one nonce are independent of verification under another. For example, assuming that the ChaCha20 block function behaves as a PRF, each keystream generated by ChaCha20 under one nonce is independent of the keystreams generated under different nonces, since, as was the case with 96-bit-nonce GCM, no two block function calls are the same. Furthermore, Poly1305 is keyed using the output of the keystream. This means that, after replacing the ChaCha20 block function by a uniformly random function, each nonce picks a different, independently distributed instance of ChaCha20+Poly1305. In particular, tag production and verification under one nonce is independent of other nonces. Say that an adversary submits a decryption query (N, C). If N was never queried to any previous Enc_K query, then tag verification is independent of all previous Enc_K queries, and it is unlikely that a forgery will be successful. Even if N was queried previously to Enc_K, then it could have only been queried once to Enc_K, and tag verification will be independent of all other Enc_K queries, meaning the adversary will have no better chance of attacking the scheme than if it had been nonce-respecting.

OCB and GCM do not satisfy the above definition because an adversary can use the Enc_K oracle to recover intermediate keys, and perform forgeries. However, there is an easy way for 96-bit-nonce GCM to mimic ChaCha20+Poly1305 such that it does become resilient to nonce re-use: produce an additional keystream block with its underlying CTR mode, and use the output of that block as the authentication key for each nonce. Minematsu and Iwata [53] consider a general version of this construction written in terms of a variable-output-length PRF and a MAC, and by replacing the PRF with CTR mode and the MAC with GHASH, one can construct a variant of GCM which provides authenticity resilience under nonce misuse, with security justification following along the lines of ChaCha20+Poly1305.

4.4 Chosen-Ciphertext Confidentiality

Much like in the conventional settings, schemes which achieve both chosen-plaintext confidentiality and authenticity resilience, achieve chosen-ciphertext confidentiality resilience, as defined below.

Definition 3. *The CCA confidentiality resilience advantage of* $\mathbf{A}$ *against* $(\mathsf{Enc}, \mathsf{Dec})$ *is*

$$\underset{\mathbf{A}}{\Delta}\left(\mathsf{Enc}_K, \mathsf{Enc}_K, \mathsf{Dec}_K\,;\,\$, \mathsf{Enc}_K, \bot\right)\,, \tag{14}$$

where nonces may not be repeated with queries to O_1*, a nonce used twice with* O_2 *cannot be used for an* O_3 *query, a query* $O_1^N(M) = C$ *or* $O_2^N(M) = C$ *may not be followed by* $O_3^N(C)$*, and finally a nonce* N *used to query* O_1^N *may not be re-used to query* O_2^N*, and vice versa.*

The fact that CPA confidentiality and authenticity resilience imply the above definition follows from a straightforward application of the triangle inequality:

$$\underset{\mathbf{A}}{\Delta}\left(\mathsf{Enc}_K, \mathsf{Enc}_K, \mathsf{Dec}_K\,;\,\$, \mathsf{Enc}_K, \bot\right) \le \underset{\mathbf{A}}{\Delta}\left(\mathsf{Enc}_K, \mathsf{Enc}_K, \mathsf{Dec}_K\,;\,\mathsf{Enc}_K, \mathsf{Enc}_K, \bot\right) \tag{15}$$

$$+ \underset{\mathbf{A}}{\Delta}\left(\mathsf{Enc}_K, \mathsf{Enc}_K, \bot\,;\,\$, \mathsf{Enc}_K, \bot\right) \tag{16}$$

The first term on the right hand side can be bounded above by authenticity of $(\mathsf{Enc}, \mathsf{Dec})$, and the second term by confidentiality.

5 Adding RUP Security to Encryption Schemes

In this section we introduce our generic method of adding RUP security to a class of encryption schemes. Following Shrimpton and Terashima [71], we take a modular approach in designing our construction. We start by describing the generic components from which the construction will be made, namely tweakable block ciphers and encryption schemes, and the security requirements they must satisfy, SPRP and SRND [32], respectively. The advantage of this approach is that the sufficient conditions to achieve security under release of unverified plaintext are made explicit, and then, depending upon the available primitives, different instantiations of the construction can be considered without resorting to new proofs.

Following a discussion of the components, we describe the construction, and discuss informally why it enhances the security of the underlying encryption scheme. The generic construction achieves RUPAE, meaning it provides both authenticity and confidentiality even if unverified plaintext is released. A formal security argument for the construction is given in Appendix B. Finally we complete the section by discussing an instantiation, GCM-RUP, which uses GCM's components to create a scheme which provides RUP-security.

5.1 Definitions

Following the RUP-model [3], we focus on designing **separated AE schemes**, where the decryption algorithm is split into plaintext computation and verification algorithms, to ensure that the decryption algorithm does not "hide" weaknesses behind the error symbol. Furthermore, our construction will communicate nonces in-band, meaning it will encrypt them and consider them as part of the ciphertext. As a result, the nonce no longer needs to be synchronized or communicated explicitly, as sufficient information is contained in the value S. This changes the syntax slightly, since now the decryption and verification algorithms no longer accept an explicit nonce input.

Formally, a separated AE scheme which communicates nonces in-band is a triplet of functions — encryption SEnc, decryption SDec, and verification SVer — where

$$\mathsf{SEnc} : \mathsf{K} \times \mathsf{N} \times \mathsf{M} \to \mathsf{C}\,, \tag{17}$$

$$\mathsf{SDec} : \mathsf{K} \times \mathsf{C} \to \mathsf{M}\,, \tag{18}$$

$$\mathsf{SVer} : \mathsf{K} \times \mathsf{C} \to \{\bot, \top\}\ . \tag{19}$$

with K the keys, N the nonces, M the messages, and C the ciphertexts. Recall that the symbols $\top$ and $\bot$ represent success and failure of verification, respectively, and we assume that neither are elements of M. It must be the case that for all $K \in \mathsf{K}$, $N \in \mathsf{N}$, and $M \in \mathsf{M}$,

$$\mathsf{SDec}_K(\mathsf{SEnc}_K^N(M)) = M \quad \text{and} \quad \mathsf{SVer}_K(\mathsf{SEnc}_K^N(M)) = \top\,. \tag{20}$$

From a separated AE scheme $(\mathsf{SEnc}, \mathsf{SDec}, \mathsf{SVer})$ one can reconstruct the following conventional AE scheme $(\mathsf{AEnc}, \mathsf{ADec})$:

$$\mathsf{AEnc}_K^N(M) := \mathsf{SEnc}_K^N(M) \tag{21}$$

$$\mathsf{ADec}_K(C) := \begin{cases} \mathsf{SDec}_K(C) & \text{if } \mathsf{SVer}_K(C) = \top \\ \bot & \text{otherwise} \end{cases} , \tag{22}$$

where we assume that the AE scheme communicates nonces in-band as well.

Separated AE schemes must provide both chosen-ciphertext confidentiality and authenticity. Both of these security aspects are captured in the **RUPAE** measure of Barwell, Page, and Stam [5]. We adopt a stronger version of their definition, by requiring the decryption algorithm to look "random" as well. Let Π denote a separated AE scheme $(\mathsf{SEnc}, \mathsf{SDec}, \mathsf{SVer})$, then the *RUPAE*-advantage of adversary $\mathbf{A}$ against Π is

$$\mathsf{RUPAE}_{\Pi}(\mathbf{A}) := \underset{\mathbf{A}}{\Delta} \left(\mathsf{SEnc}_K, \mathsf{SDec}_K, \mathsf{SVer}_K \,;\, \$_{\mathsf{SEnc}}, \$_{\mathsf{SDec}}, \bot\right) , \tag{23}$$

where $\mathbf{A}$ is *nonce-respecting*, meaning the same nonce is never queried twice to SEnc. Nonces may be repeated with SDec and SVer. Furthermore, $\mathbf{A}$ cannot use the output of an O_1^N query as the input to an O_2^N or O_3^N query with the same nonce N, otherwise such queries result in trivial wins.

5.2 Generic Construction

Components. A *tweakable block cipher* [47] is a pair of functions (E, D), with

$$\mathsf{E} : \mathsf{K} \times \mathsf{T} \times \mathsf{X} \to \mathsf{X} \tag{24}$$

$$\mathsf{D} : \mathsf{K} \times \mathsf{T} \times \mathsf{X} \to \mathsf{X} , \tag{25}$$

where K is the key space, T the tweak space, and X the domain, where $\mathsf{X} = \{0,1\}^x$ is a set of strings of a particular length. For all $K \in \mathsf{K}$ and $T \in \mathsf{T}$ it must be the case that E_K^T is a permutation with D_K^T as inverse. We will need to measure the SPRP quality of the tweakable block cipher, which is defined as

$$\mathsf{SPRP}(\mathbf{A}) := \underset{\mathbf{A}}{\Delta} \left(\mathsf{E}_K, \mathsf{D}_K \,;\, \pi, \pi^{-1}\right) , \tag{26}$$

where K is chosen uniformly at random from K, and (π, π^{-1}) is a family of independent, uniformly distributed random permutations over X indexed by T.

Although Liskov, Rivest, and Wagner [47] introduced the concept of finite-tweak-length (FTL) block ciphers, for our construction we need tweakable block ciphers that can process variable tweak lengths (VTL). Starting from an FTL block cipher, one can construct a VTL block cipher by compressing the tweak using a universal hash function, and using the resulting output as the tweak for the FTL block cipher, as explained by Coron et al. [22]. Minematsu and Iwata [54] introduce the XTX construction which extends tweak length while minimizing security loss.

There are a few dedicated constructions of FTL block ciphers: the hash function SKEIN [26] contains an underlying tweakable block cipher, the CAESAR competition candidates Joltik [41] and Deoxys [40] also developed new tweakable block ciphers, and the TWEAKEY framework [39] tackles the problem of designing tweakable block ciphers in general. Besides dedicated constructions, there are also constructions of tweakable block ciphers using regular block ciphers; see for example Rogaway's XE and XEX constructions [64], Minematsu's beyond-birthday bound construction [52], Landecker, Shrimpton, and Terashima's CLRW2 construction [45], and Mennink's beyond-birthday bound constructions [51].

An *encryption scheme* $(\mathsf{Enc}, \mathsf{Dec})$ is a separated AE scheme without SVer. The basic security requirement for encryption schemes is chosen-plaintext confidentiality, but this is not sufficient for our purpose. In particular, a mode like CBC [55] will not work, since during decryption a change in the nonce will only affect the first decrypted plaintext block. We need encryption schemes where during decryption a change in the nonce will result in the entire plaintext changing. Modes such as CTR [55], OFB [55], and the encryption of OCB [43,64,65] suffice. In particular, it is necessary that both encryption and decryption algorithms give uniform random output when distinct nonces are input across both encryption *and* decryption. For example, with CTR mode, decryption is the same as encryption, and if nonces are never repeated across both algorithms then its output will always look uniformly random.

We use Shrimpton and Terashima's [71] SRND measure for encryption schemes, which was introduced by Halevi and Rogaway [32]:

$$\mathsf{SRND}(\mathbf{A}) := \underset{\mathbf{A}}{\Delta}\left(\mathsf{Enc}_K, \mathsf{Dec}_K\, ;\, \$_{\mathsf{Enc}}, \$_{\mathsf{Dec}}\right), \tag{27}$$

where K is chosen uniformly at random from K, and $\mathbf{A}$ must use a different nonce for *every* query it makes, to both of its oracles.

Description. Let $(\mathsf{Enc}, \mathsf{Dec})$ be an encryption scheme with key space K, nonce space N, message space M, and ciphertext space C. Let (E, D) be a tweakable block cipher with $\mathsf{T} = \mathsf{N} \times \mathsf{C}$, $\mathsf{X} = \mathsf{N}$, and key space K. Let $\alpha \in \{0,1\}^\tau$ be some pre-defined constant. Then define the separated AE scheme $(\mathsf{SEnc}, \mathsf{SDec}, \mathsf{SVer})$ as follows. The key space is K^2, with keys denoted by (K, L), the nonce space is N, the message space is M, and the ciphertext space is $\mathsf{N} \times \mathsf{C}$:

$$\mathsf{SEnc}^N_{K,L}(M) := \left(\mathsf{E}^C_L(N), C\right) \tag{28}$$

$$\text{with } C := \mathsf{Enc}^N_K(\alpha \parallel M) \tag{29}$$

$$\mathsf{SDec}_{K,L}(S, C) := \mathsf{lsb}_{|C|-\tau}\left(\mathsf{Dec}^{N'}_K(C)\right) \tag{30}$$

$$\text{with } N' := \mathsf{D}^C_L(S) \tag{31}$$

$$\mathsf{SVer}_{K,L}(S, C) := \left(\mathsf{msb}_\tau\left(\mathsf{Dec}^{N'}_K(C)\right) \overset{?}{=} \alpha\right). \tag{32}$$

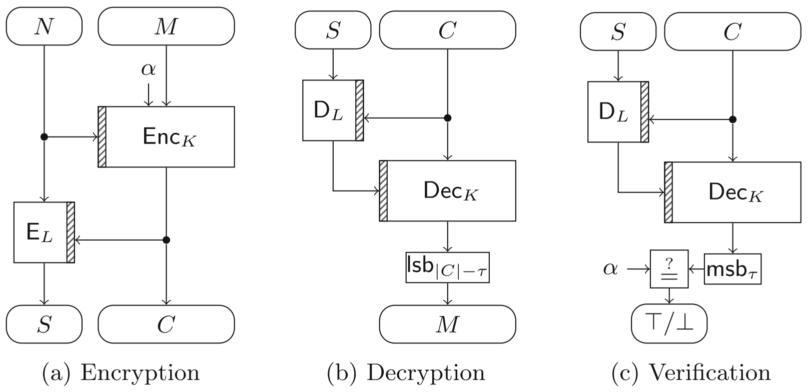


Fig. 2. Adding RUP security to an existing encryption scheme. The circles indicate duplication of the value.

The construction is depicted in Fig. 2.

The construction adds robustness to the encryption scheme $(\mathsf{Enc}, \mathsf{Dec})$ by compressing the ciphertext via the tweak of the tweakable block cipher, and using that information to encrypt the nonce. As a result, during decryption, if any bit of the ciphertext is modified, then the ciphertext will result in a different tweak with essentially probability one, and the tweakable block cipher will decrypt the nonce into some random value, which is used as the new nonce for Dec. By assumption, Dec will output garbage, or more precisely, plaintext which is unrelated to any other plaintext queried.

Similarly, if the ciphertext is kept the same, and the encrypted nonce, S, is modified, then the tweakable block cipher will be queried on an input for which it has not been queried on before with the given tweak computed from the ciphertext. As a result, the decryption of S will be random, and Dec's output will look random as well.

With respect to authenticity, our construction follows the encode-then-encipher paradigm [9], which uses redundancy in the plaintext in order to guarantee authenticity. The level of authenticity is determined by the length of the constant α: if verification can be removed, then α's length is set to zero. However, the only requirement from α is to be known to both sides, and users may use any predictable bits already present in the plaintext.

5.3 GCM-RUP

We illustrate an instantiation of the construction using familiar primitives, namely those used to construct GCM [49,50]. The resulting instantiation uses three independent keys, but only makes three minor modifications to AES-GCM in order to achieve RUP security:

1. the plaintext is prepended by a string of zero bits of length τ,
2. the nonce N instead of $\mathsf{GHASH}(\varepsilon, N)$ is used to generate the mask for the polynomial hash, and
3. the output of GHASH is XORed with the nonce before it is encrypted.

See Fig. 3 for an illustration.

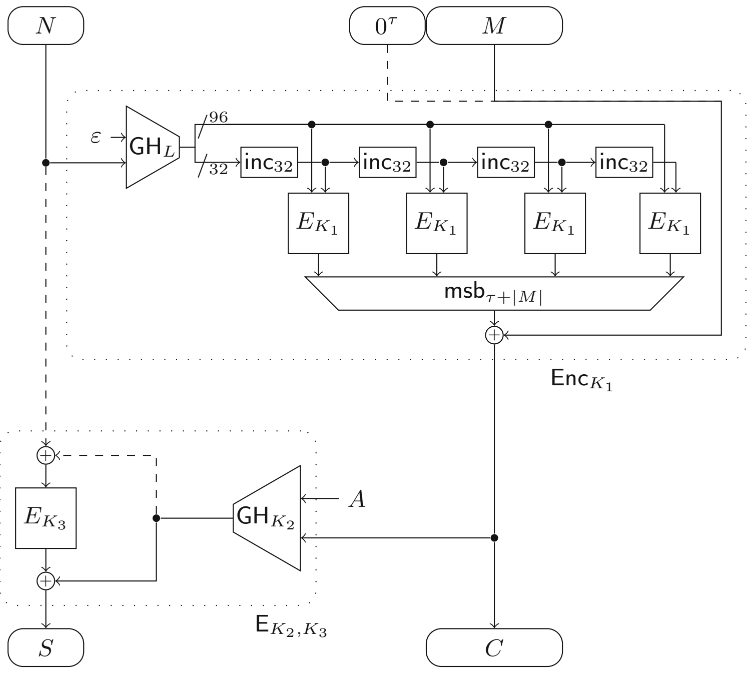


Fig. 3. Instantiation of our construction using GCM's components. Changes from GCM are indicated using a dashed pattern, and the dotted boxes point out the underlying encryption scheme and tweakable block cipher. Filled circles indicate duplication of the values. GH is GHASH, and $/_m$ indicates the number of bits on a wire. The value L is $E_{K_1}(0^n)$, and A represents associated data.

Appendix A.2 contains a description of the GCM components that we use to describe the instantiation, including the function GHASH, defined in Algorithm 3, and CTR mode, defined in Algorithm 4. Note that our formalization above did not include associated data, whereas GCM-RUP does, however it is straightforward to extend the definitions and generic construction to include it.

Since the generic construction's security relies on generating random nonce input during decryption, in order to maintain security up to the birthday bound on the block size, as is the case with GCM, the nonce size in the instantiation is fixed to be the same as the block size. The encryption scheme underlying the

Algorithm 1. $\mathsf{GCM\text{-}RUP}_{K_1K_2K_3}(A, M)$

```
    Input: $K_1K_2K_3 \in \{0,1\}^{3n}$, $A \in \{0,1\}^{\leq n2^{32}}$, $M \in \{0,1\}^{\leq n2^{32}}$
    Output: $(S, C) \in \{0,1\}^n \times \{0,1\}^{\tau+|M|}$
1   $M \leftarrow 0^\tau \parallel M$
2   $L \leftarrow E_{K_1}(0^n)$
3   $I \leftarrow \mathsf{GHASH}_L(\varepsilon, N)$
4   $m \leftarrow |M|_n$
5   $F \leftarrow E_{K_1}(\mathsf{msb}_{96}(I) \parallel \cdot)$
6   $S \leftarrow \mathsf{CTR}[F](\mathsf{inc}_{32}(\mathsf{lsb}_{32}(I)), m)$
7   $C \leftarrow M \oplus \mathsf{msb}_{|M|}(S)$
8   $T \leftarrow \mathsf{GHASH}_{K_2}(N \| A, C)$
9   $S \leftarrow E_{K_3}(I \oplus T) \oplus T$
10  return $(S, C)$
```

instantiation, $(\mathsf{Enc}, \mathsf{Dec})$, is the same as GCM without authentication, or in other words CTR mode, therefore Enc and Dec are identical, and so the SRND quality of $(\mathsf{Enc}, \mathsf{Dec})$ can be measured by looking only at Enc-queries. This allows us to use the GCM confidentiality result of Niwa et al. [58,59], which gives $(\mathsf{Enc}, \mathsf{Dec})$ an SRND-bound of

$$\frac{0.5(\sigma + q + d + 1)^2}{2^n} + \frac{64 \cdot q(\sigma + q + d)}{2^n}, \tag{33}$$

where σ is the total number of blocks queried, q the number of Enc queries, d the number of Dec queries, and the nonce length is n bits, which is the block size as well.

Security of the underlying tweakable block cipher follows from the XTX construction of Minematsu and Iwata [54], where we extend the tweak space of a block cipher to arbitrary tweak size by XORing GHASH to both the input and output of the block cipher. Hence the SPRP-quality of the underlying tweakable block cipher is

$$\frac{q^2(\ell + 1)}{2^n}, \tag{34}$$

where q is the total number of queries made to the tweakable block cipher, and ℓ is the maximal tweak length, or in other words, the maximal ciphertext and associated data length in blocks.

Putting together the results along with the result of Appendix B, we get the following bound for the instantiation.

Theorem 1. *Let* $\mathbf{A}$ *be a RUPAE-adversary against the instantiation making at most* q SEnc *queries, and* v SDec *and* SVer *queries. Say that at most* σ *blocks are queried, with* ℓ *the maximum ciphertext and associated data block length of any query, then* $\mathbf{A}$*'s advantage is at most*

$$\frac{0.5(\sigma + q + v + 1)^2}{2^n} + \frac{64 \cdot q(\sigma + q + v)}{2^n} + \frac{(q + v)^2(\ell + 1)}{2^n} + 2\frac{v(q + v + 1)}{2^n - q - v}. \tag{35}$$

If $q + v \leq 2^{n-1}$, then since $q + v \leq \sigma$, the bound can be simplified to

$$\frac{3 \cdot 64 \cdot \sigma^2}{2^n} + \frac{\sigma^2(\ell + 1)}{2^n}, \tag{36}$$

which is similar to GCM's security bounds [38,58].

Acknowledgments. The authors would like to thank Günes Acar, Roger Dingledine, Ian Goldberg, Mark Juarez, Bart Mennink, and Vincent Rijmen, as well as the anonymous reviewers. This work was supported in part by the Research Council KU Leuven: GOA TENSE (GOA/11/007). Tomer Ashur was supported in part by the Research Fund KU Leuven, OT/13/071. Orr Dunkelman was supported in part by the Israeli Science Foundation through grant No. 827/12 and by the Commission of the European Communities through the Horizon 2020 program under project number 645622 PQCRYPTO. Atul Luykx is supported by a Fellowship from the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen). This article is based upon work from COST Action IC1403 CRYPTACUS, supported by COST (European Cooperation in Science and Technology).

A Algorithm Descriptions

In this section we provide descriptions of OCB, GCM, and ChaCha20+Poly1305. The descriptions are only given to the level of detail sufficient for the paper. The notation is borrowed various sources: the description of OCB by Rogaway, Bellare, and Black [65], the description of GCM by Iwata, Ohashi, and Minematsu [38], and the documents by Procter analyzing ChaCha20+Poly1305 [61,62].

A.1 OCB

In this section we describe the OCB mode of operation [43,64,65]. We focus on OCB version 1 [65], however our results extend to all versions of OCB. We do not include associated data as we do not need it for the OCB attacks. The reference used for the figure, pseudocode, and notation below is from [65]. Let $E : \mathsf{K} \times \{0,1\}^n \to \{0,1\}^n$ be a block cipher and let τ denote the tag length, which is an integer between 0 and n. Let $\gamma_1, \gamma_2, \ldots$ be constants, whose values depend on the version of OCB used; for example, in OCB1 [65] these are Gray codes. Then Algorithm 2 gives pseudocode describing OCB encryption, and Fig. 4 provides an accompanying diagram.

A.2 GCM

In this section we describe the GCM mode of operation [49,50] with nonces of 128 bit length. We let $E : \{0,1\}^{128} \times \{0,1\}^{128} \to \{0,1\}^{128}$ denote a block cipher. The function GHASH is defined in Algorithm 3. Algorithm 5 provides pseudocode for GCM encryption, which also uses the keystream generator CTR mode in Algorithm 4. See Fig. 5 for an illustration.

Algorithm 2. $\mathsf{OCB}_K(N, M)$

```
   Input: $K \in \{0,1\}^n$, $M \in \{0,1\}^*$
   Output: $C \in \{0,1\}^*$
1  $M[1]M[2]\cdots M[m] \xleftarrow{n} M$
2  $L \leftarrow E_K(0^n)$
3  $R \leftarrow E_K(N \oplus L)$
4  for $i = 1$ to $m$ do
5      $Z[i] = \gamma_i \cdot L \oplus R$
6  end
7  for $i = 1$ to $m$ do
8      $C[i] \leftarrow E_K(M[i] \oplus Z[i]) \oplus Z[i]$
9  end
10 $X[m] \leftarrow \mathsf{len}_n(M[m]) \oplus L \cdot \mathtt{x}^{-1} \oplus Z[m]$
11 $Y[m] \leftarrow E_K(X[m])$
12 $C[m] \leftarrow Y[m] \oplus M[m]$
13 Checksum$\leftarrow M[1] \oplus \cdots \oplus M[m-1] \oplus C[m]0^{*n} \oplus Y[m]$
14 $T \leftarrow \mathsf{msb}_\tau\Big(E_K(\text{Checksum} \oplus Z[m])\Big)$
15 return $C[1]\cdots C[m]T$
```

Algorithm 3. $\mathsf{GHASH}_L(A, C)$

```
   Input: $L \in \{0,1\}^n$, $A \in \{0,1\}^{\leq n(2^{n/2}-1)}$, $C \in \{0,1\}^{\leq n(2^{n/2}-1)}$
   Output: $Y \in \{0,1\}^n$
1  $X \leftarrow A0^{*n} \parallel C0^{*n} \parallel \mathsf{str}_{n/2}(|A|) \parallel \mathsf{str}_{n/2}(|C|)$
2  $X[1]X[2]\cdots X[x] \xleftarrow{n} X$
3  $Y \leftarrow 0^n$
4  for $j = 1$ to $x$ do
5      $Y \leftarrow L \cdot (Y \oplus X[j])$
6  end
7  return $Y$
```

Algorithm 4. $\mathsf{CTR}[F](X, m)$

```
   Input: $F : \{0,1\}^x \rightarrow \{0,1\}^n$ , $X \in \{0,1\}^x$, $m \in \mathbb{N}$
   Output: $S \in \{0,1\}^{mn}$
1  $I \leftarrow X$
2  for $j = 1$ to $m$ do
3      $S[j] \leftarrow F(I)$
4      $I \leftarrow \mathsf{inc}_x(I)$
5  end
6  $S \leftarrow S[1]S[2]\cdots S[m]$
7  return $S$
```

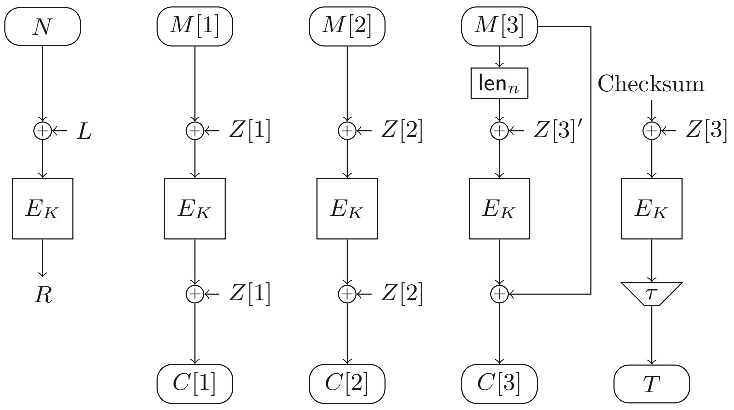


Fig. 4. The OCB mode of operation applied to a plaintext of length at most three blocks. The value L is $E_K(0^n)$ and $Z[3]' = Z[3] \oplus L \cdot \mathtt{x}^{-1}$.

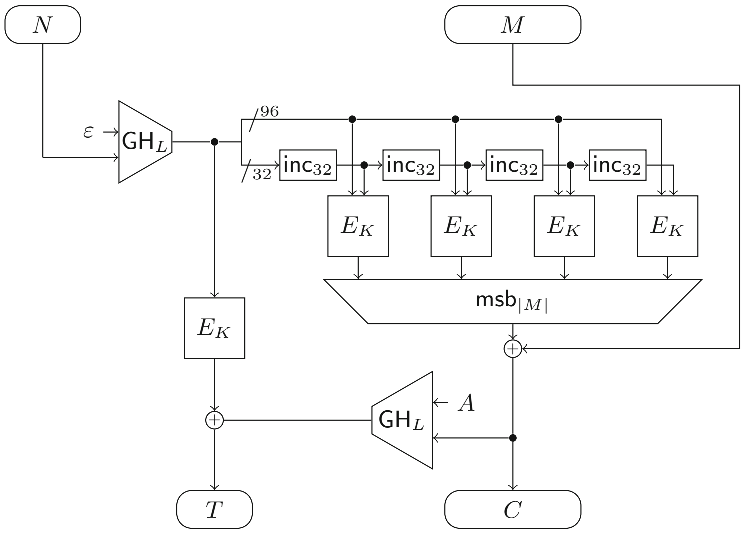


Fig. 5. The GCM mode of operation with 128 bit nonces. GH is GHASH and $/_m$ indicates the number of bits on a wire. The value L is $E_K(\mathsf{str}_n(0))$.

Algorithm 5. $\mathsf{GCM}_K(N, A, M)$

```
Input: $K \in \{0,1\}^{128}$, $N \in \{0,1\}^{128}$, $A \in \{0,1\}^{\leq 128\cdot 2^{32}}$, $M \in \{0,1\}^{\leq 128\cdot 2^{32}}$
Output: $(C,T) \in \{0,1\}^{\leq 128\cdot 2^{32}} \times \{0,1\}^{128}$
1 $L \leftarrow E_K(\mathsf{str}_{128}(0))$
2 $I \leftarrow \mathsf{GHASH}_L(\varepsilon, N)$
3 $m \leftarrow |M|_{128}$
4 $F \leftarrow E_K\left(\mathsf{msb}_{96}(I) \parallel \cdot\right)$
5 $C \leftarrow M \oplus \mathsf{msb}_{|M|}\left(\mathsf{CTR}[F]\left(\mathsf{inc}_{32}(\mathsf{lsb}_{32}(I)), m\right)\right)$
6 $T \leftarrow E_K(I) \oplus \mathsf{GHASH}_L(A, C)$
7 return $(C,T)$
```

A.3 ChaCha20+Poly1305

Our description of ChaCha20+Poly1305 is taken from the RFC [56] describing it, as well as Procter's analysis [61,62]. The combination of ChaCha20 [13] and Poly1305 [12] is similar to that of GCM, with the main differences being the fact that block cipher calls are replaced by ChaCha20's block function calls, and the key for Poly1305 is generated differently.

The ChaCha20 block function is denoted by

$$\mathsf{CC} : \{0,1\}^{256} \times \{0,1\}^{32} \times \{0,1\}^{96} \rightarrow \{0,1\}^{512}\ , \tag{37}$$

which operates on keys of length 256 bits, a block number of length 32 bits, a nonce of length 96 bits, and with an output of size 512 bits. The Poly1305 universal hash function is denoted by

$$\mathsf{Poly} : \{0,1\}^{128} \times \{0,1\}^{*} \rightarrow \{0,1\}^{128}\ . \tag{38}$$

The description of ChaCha20+Poly1305 encryption is given in Algorithm 6.

Algorithm 6. $\mathsf{CC\&Poly}_K(N, A, M)$

```
Input: $K \in \{0,1\}^{256}$, $N \in \{0,1\}^{96}$, $A \in \{0,1\}^{\leq 8\cdot(2^{64}-1)}$, $M \in \{0,1\}^{\leq 512\cdot(2^{32}-1)}$
Output: $(C,T) \in \{0,1\}^{|M|} \times \{0,1\}^{128}$
1 $F \leftarrow \mathsf{CC}_K(\cdot, N)$
2 $C \leftarrow M \oplus \mathsf{msb}_{|M|}\left(\mathsf{CTR}[F]\left(\mathsf{str}_{32}(1), |M|_{512}\right)\right)$
3 $L \leftarrow \mathsf{msb}_{256}\left(F(\mathsf{str}_{32}(0))\right)$
4 $T \leftarrow \mathsf{lsb}_{128}(L) \oplus \mathsf{Poly}_{\mathsf{msb}_{128}(L)}\left(A0^{*128} \| C0^{*128} \| \mathsf{str}_{64}(|A|_8) \| \mathsf{str}_{64}(|C|_8)\right)$
5 return $(C,T)$
```

B Formal Security Argument For The Generic Construction

We start by defining two reductions which use an adversary $\mathbf{A}$ playing the RUPAE game against the construction $\mathbb{S} = (\mathsf{SEnc}, \mathsf{SDec}, \mathsf{SVer})$. Let $\mathbb{B} = (\mathsf{E}, \mathsf{D})$ denote the tweakable block cipher and $\mathbb{E} = (\mathsf{Enc}, \mathsf{Dec})$ the encryption scheme. Furthermore, let $\$_{\mathbb{S}} = (\$_{\mathsf{SEnc}}, \$_{\mathsf{SDec}}, \bot)$, $\$_{\mathbb{B}} := (\pi, \pi^{-1})$, where (π, π^{-1}) is from the definition of SPRP security in Eq. (26), and $\$_{\mathbb{E}} := (\$_{\mathsf{Enc}}, \$_{\mathsf{Dec}})$. Then we define the following two reductions:

1. A reduction $\mathbf{B}\langle\mathbf{A}\rangle$ to the SPRP quality of the tweakable block cipher $\mathbb{B}$, meaning $\mathbf{B}\langle\mathbf{A}\rangle$ will attempt to distinguish $\mathbb{B}$ from $\$_{\mathbb{B}}$, using $\mathbf{A}$, an algorithm which is expecting either $\mathbb{S}$ or $\$_{\mathbb{S}}$. The reduction $\mathbf{B}$ generates a key K independently, and uses K to simulate the encryption scheme $\mathbb{E}$. Then, $\mathbf{B}$ runs $\mathbf{A}$, and responds to $\mathbf{A}$'s queries by reconstructing $\mathbb{S}$ using its own oracles, either $\mathbb{B}$ or $\$_{\mathbb{B}}$, and the simulated $\mathbb{E}$.
2. A reduction $\mathbf{C}\langle\mathbf{A}\rangle$ to the SRND quality of the encryption scheme $\mathbb{E}$. In contrast with $\mathbf{B}$, the reduction $\mathbf{C}$ simulates $\$_{\mathbb{B}}$ instead of $\mathbb{B}$. Then using its own oracles, either $\mathbb{E}$ or $\$_{\mathbb{E}}$, and $\$_{\mathbb{B}}$, $\mathbf{C}$ reconstructs $\mathbb{S}$.

Theorem 2. *The advantage of any nonce-respecting RUPAE adversary* $\mathbf{A}$ *attempting to distinguish* $\mathbb{S}$ *from* $\$_{\mathbb{S}}$*, making at most* q SEnc *queries, and at most* v SDec *and* SVer *queries, is bounded above by*

$$2\frac{v(q+v)}{|\mathsf{N}| - q - v} + \frac{v}{2^{\tau}} + \mathsf{SPRP}_{\mathbb{B}}(\mathbf{B}\langle\mathbf{A}\rangle) + \mathsf{SRND}_{\mathbb{E}}(\mathbf{C}\langle\mathbf{A}\rangle)\,. \tag{39}$$

Proof. Let $\mathbb{S}[\Pi, \Sigma]$ denote $\mathbb{S}$ using Π as tweakable block cipher and Σ as encryption scheme. By definition, we seek to bound

$$\mathsf{RUPAE}(\mathbf{A}) = \underset{\mathbf{A}}{\Delta}\left(\mathbb{S}[\mathbb{B}, \mathbb{E}]\,;\,\$_{\mathbb{S}}\right)\,. \tag{40}$$

Applying the triangle inequality, we get

$$\underset{\mathbf{A}}{\Delta}\left(\mathbb{S}[\mathbb{B}, \mathbb{E}]\,;\,\$_{\mathbb{S}}\right) \le \underset{\mathbf{A}}{\Delta}\left(\mathbb{S}[\mathbb{B}, \mathbb{E}]\,;\,\mathbb{S}[\$_{\mathbb{B}}, \mathbb{E}]\right) + \underset{\mathbf{A}}{\Delta}\left(\mathbb{S}[\$_{\mathbb{B}}, \mathbb{E}]\,;\,\$_{\mathbb{S}}\right) \tag{41}$$

Using reduction $\mathbf{B}\langle\mathbf{A}\rangle$, we know that

$$\underset{\mathbf{A}}{\Delta}\left(\mathbb{S}[\mathbb{B}, \mathbb{E}]\,;\,\mathbb{S}[\$_{\mathbb{B}}, \mathbb{E}]\right) \le \underset{\mathbf{B}\langle\mathbf{A}\rangle}{\Delta}\left(\mathbb{B}\,;\,\$_{\mathbb{B}}\right)\,. \tag{42}$$

Therefore we can focus on

$$\underset{\mathbf{A}}{\Delta}\left(\mathbb{S}[\$_{\mathbb{B}}, \mathbb{E}]\,;\,\$_{\mathbb{S}}\right) \tag{43}$$

which in turn is bounded above by

$$\underset{\mathbf{A}}{\Delta}\left(\mathbb{S}[\$_{\mathbb{B}}, \mathbb{E}]\,;\,\mathbb{S}[\$_{\mathbb{B}}, \$_{\mathbb{E}}]\right) + \underset{\mathbf{A}}{\Delta}\left(\mathbb{S}[\$_{\mathbb{B}}, \$_{\mathbb{E}}]\,;\,\$_{\mathbb{S}}\right)\,. \tag{44}$$

The analysis of these two remaining terms relies on computing the probability that **A** makes a query which results in a nonce collision during a decryption query, thereby violating the SRND game's requirement. In the analysis below, we find that the probability of such an occurence is at most

$$\varepsilon := \frac{v(q+v)}{|\mathsf{N}| - q - v} \,. \tag{45}$$

Therefore, using the reduction $\mathbf{C}\langle\mathbf{A}\rangle$ we know that the first term of Eq. (44) is bounded by

$$\varepsilon + \underset{\mathbf{C}\langle\mathbf{A}\rangle}{\Delta} (\mathbb{E}\,;\,\$_{\mathbb{E}}) \,, \tag{46}$$

and the bound for

$$\underset{\mathbf{A}}{\Delta} (\mathbb{S}[\$_{\mathbb{B}}, \$_{\mathbb{E}}]\,;\,\$_{\mathbb{S}}) \tag{47}$$

is given below.

Say that **A** generates SEnc inputs (N_1, M_1), (N_2, M_2), $\ldots$, (N_q, M_q), and SDec and SVer inputs (S_1, C_1), (S_2, C_2), $\ldots$, (S_v, C_v), where (S_i, C_i) could be the input to either an SDec or SVer query. Let N_i^* denote the nonce input to $\$_{\mathsf{Dec}}$ resulting from the query (S_i, C_i), that is

$$N_i^* = \pi^{-1, C_i}(S_i) \,. \tag{48}$$

Similarly, define M_i^* and α_i^* such that

$$\alpha_i^* \| M_i^* = \$_{\mathsf{Dec}}^{N_i^*}(C_i) \,. \tag{49}$$

We call N_i^*, M_i^*, and α_i^* the "decrypted" nonces, plaintexts, and constants, respectively.

If the nonces N_i and N_j^* are distinct from each other then the SRND game's requirement is respected, hence $\$_{\mathbb{E}}$ will always give uniformly distributed and independent output. Let event denote the event that either $N_i = N_j$ for $1 \le i < j \le q$, or $N_i^* = N_j^*$ for $1 \le i < j \le v$, or $N_i = N_j^*$ for $1 \le i \le q$ and $1 \le j \le v$. Then, by the fundamental lemma of game playing [10], Eq. (47) can be bounded by

$$\mathbf{P}\left[\mathsf{event}\right] + \mathbf{P}\left[\exists\, i \text{ s.t. } \alpha_i^* = \alpha \;\middle|\; \overline{\mathsf{event}}\right] \,, \tag{50}$$

where $\overline{\mathsf{event}}$ is the negation of event. Given $\overline{\mathsf{event}}$, the nonce input to $\$_{\mathsf{Dec}}$ will always be distinct, hence the α_i^* are independent and uniformly distributed, which means the quantity on the right is bounded above by $v/2^\tau$.

Therefore we focus on the probability of event, i.e. that there is a collision in the N_i and N_j^*. By hypothesis, **A** is nonce-respecting, hence we know that $N_i \neq N_j$ for $1 \le i < j \le q$. Therefore we focus on the case that a decrypted nonce collides with some N_i, or another decrypted nonce.

Consider the query (S_i, C_i) associated to the ith decrypted nonce N_i^*, and say that event has not yet been triggered. Let (N_j, M_j) be a previous SEnc query, and (S_j, C_j) its corresponding output. By hypothesis, $(S_j, C_j) \neq (S_i, C_i)$.

If $C_j \neq C_i$, then the tweak input to (π, π^{-1}) will be different for the SEnc and SDec or SVer queries, hence the probability that N_i^* collides with N_j is at most $1/|\mathsf{N}|$. If $C_j = C_i$, then $S_j \neq S_i$, which means that (π, π^{-1}) is queried under the same tweak for both the SEnc and SDec or SVer queries. However, the probability that

$$N_j = \pi^{-1,C_j}(S_j) = \pi^{-1,C_i}(S_i) = N_i^* \tag{51}$$

is at most $1/(|\mathsf{N}| - q - v)$.

Now consider the probability that an SEnc query (N_i, M_i) is such that N_i equals N_j^* for some previous SDec or SVer query. Since the adversary's view is independent of N_j^*, it can guess N_j^* with probability at most $1/(|\mathsf{N}| - q - v)$. Therefore, the probability that a decrypted nonce collides with some nonce N_j is at most

$$\frac{qv}{|\mathsf{N}| - q - v} \,. \tag{52}$$

Given that no decrypted nonces collide with any nonce N_j, we are left with the event that two decrypted nonces collide with each other. However, similar reasoning as above shows that this probability is bounded above by

$$\frac{v^2}{|\mathsf{N}| - q - v} \,, \tag{53}$$

Putting the above computations together, if $\mathbf{A}$ makes q SEnc queries, and v SDec and SVer queries, then Eq. (47) is bounded above by

$$\frac{v(q+v)}{|\mathsf{N}| - q - v} + \frac{v}{2^\tau} \,. \tag{54}$$

□

C Application to Tor

The advantage in coming up with new, robust AE schemes is that they can then be used for applications which go beyond the traditional goals of ensuring data confidentiality and authenticity between two communicating parties. Consider for example Tor [23], which uses CTR mode [55] to ensure anonymity. CTR mode is a basic encryption scheme which provides data confidentiality, and no authenticity. In particular, its decryption algorithm provides no robustness to changes in its ciphertext: a change in the ith bit of ciphertext will result in the same change to the ith bit of the resulting plaintext. This property enables the crypto-tagging attack [73] against Tor, which breaches anonymity. Using an RAE [35] or encode-then-encipher [9,71] scheme prevents the crypto-tagging attack, and potentially introduces a new level of robustness to Tor's anonymity. Hence, the Tor community has initiated a search for replacements for CTR mode [48].

However, replacing CTR mode with known robust solutions not only comes at an efficiency cost, but also increased design, and hence implementation, complexity. This is because so far the only solutions provided for full robustness are essentially variable-input-length (VIL) ciphers [8], which can be viewed as block ciphers that can process arbitrarily long messages. However, VIL ciphers are "heavy" constructions, requiring often three or more passes over the plaintext in order to ensure sufficient mixing, or relying on subtle design choices to achieve security.

We now outline how our construction can be used in Tor to avoid the crypto-tagging attack [73]. Our intention is not to provide a detailed description, but to give a high-level overview.

C.1 Tor

Tor [23] is a circuit-based low-latency anonymous communication service. The core idea underlying Tor is *onion routing*, a distributed overlay network designed to anonymize TCP-based applications, presented by Syverson, Reed and Goldschlag in [72].

Generally speaking, Tor communication is encrypted and relayed by nodes in the Tor-network via *circuits*. When building circuits, clients exchange keys with several nodes, usually 3, where each node only knows its predecessor and successor.

Clients prepare messages using multiple layers of encryption. First, the message is encrypted using the key and nonce shared with the circuit's last node. The resulting ciphertext is then encrypted again with the keys and nonce of the one-before-last node. This process is repeated for each node, until the first node's key is used.

The output of the multi-layered encryption is then sent from the client to the first node, which decrypts one layer, and forwards the result to the next node. In every step, another layer of encryption is removed, and the message is passed forward, until it reaches the last node. The last node authenticates and forwards the message to the intended recipient outside of the Tor network.

C.2 The Crypto-tagging Attack

By design, the Tor protocol offers an end-to-end integrity check, which the exit node does by computing a SHA-1 digest of the decrypted message. Such a check prevents e.g., attacks by rogue nodes which "tag" the encrypted message, and then search outbound communication for the corresponding corrupted traffic.

In 2012, an anonymous email was sent to the Tor developers mailing list describing the *crypto-tagging attack* [73]. In this attack, two nodes, the entry and exit nodes, collude by tagging and untagging messages upon entry and exit to the network, respectively, thereby making the changes transparent to all other parties.[1] Due to the mode of operation used for encryption, CTR mode, the location of corrupt bits introduced at the entry to the network are maintained

[1] Tagging can be done in several ways. We mention here only one: the entry node XORs an identifing string to the message they are passing. Untagging is done by XORing the same identifier by the exit node.

through all decryptions, and can be removed by the exit node by just knowing their location. Furthermore, since the integrity check is only performed by the exit node, the corruption cannot be detected by intermediate nodes in the circuit. Moreover, the attack is amplified by the fact that if only one of the nodes (i.e., either the entry node or the exit node) is malicious, the tagged message cannot be verified, and the circuit is destroyed. Any new circuit where only one of the nodes is malicious will also be destroyed, thus biasing the set of possible circuits towards compromised ones.

An obvious solution to this problem is to add an authentication tag to each layer of the encryption, allowing intermediate nodes to verify the data passed through them and act according to some policy. However, in the discussion following the attack, such a solution was ruled out due to two main problems: (i) by adding authentication tags, the available bandwidth for sending messages is reduced, and (ii) the circuit's length could be revealed, an undesirable property in such systems.

C.3 Avoiding the Attack

We propose a different approach allowing intermediate nodes to release unverified plaintext, using the generic construction proposed in Sect. 5. The only change from the above procedure for preparing the message is how the nonces are chosen.

As before, clients start by encrypting the plaintext with the key and nonce of the last node using CTR mode. Then, the ciphertext is compressed and used as a tweak for the encryption of the nonce as per Fig. 2. Afterwards, the encrypted nonce, S, is used as the nonce for the next layer of encryption, i.e., with the keys of the one-before last node. This is repeated for each node of the circuit all the way to the first one. The result is a multi-layered application of our construction where the first layer receives the nonce and the plaintext as input, and each subsequent layer receives the previous layer's output. The new RUP secure layered encryption mode of operation is presented in Fig. 6, where each layer can be realized using e.g., the robust version of GCM presented in Sect. 5.3 with $|\alpha| = 0$.

When the message is ready, the client sends the ciphertext, along with the 3-times encrypted nonce to the first node. The first node uses the decryption algorithm as per Fig. 2 to remove the outermost encryption, and forwards the result, as well as the now 2-times encrypted nonce, to the next node. After the last layer of encryption has been removed by the last node, it authenticates the message and sends it to the intended recipient.

The security against an adversary trying to mount the crypto-tagging attack comes from the fact that any change to the ciphertext will affect the entire message, effectively decrypting it to garbage. In other words, once decrypted by a non-colluding node, the crypto-tag corrupts the nonce, which will then be used to decrypt the message into garbage. Using the Tor terminology, by the time the message reaches the exit node, the crypto-tag can no longer be removed and the message is unrecognizable and should thus be dropped and the circuit is torn down.

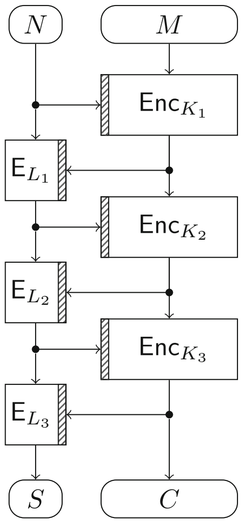


Fig. 6. A RUP secure 3-node layered encryption. The three layers are distinguished by their keys: $(K_1, L_1), (K_2, L_2)$, and (K_3, L_3).

For example, consider a circuit with three nodes, and say that (S_1, C_1), (S_2, C_2), and (S_3, C_3) are the outputs of the first, second, and third layers of encryption, respectively. In particular, the client uses (N, P) to produce (S_1, C_1), then (S_1, C_1) to produce (S_2, C_2), and (S_2, C_2) to produce (S_3, C_3). Finally, (S_3, C_3) is sent to the first node. Say that the first node is malicious, namely it decrypts (S_3, C_3) and obtains (S_2, C_2), then proceeds to tag (S_2, C_2) and passes (S_2', C_2') instead of (S_2, C_2) as it is supposed to do. Then, assuming the second node is honest, it will follow the protocol and decrypt (S_2', C_2'). However, by the properties of our construction, we know that the decryption will be random since $(S_2, C_2) \neq (S_2', C_2')$, and in particular, the first node will not be able to predict anything about (S_1', C_1'), i.e., the decryption of (S_2', C_2'). As a result, the second node will pass (S_1', C_1') to the third node, and the third node will not be able to conclude anything, regardless of whether it shares information with the first node or not. In particular, it would not be able to conclude the source and the destination of the message.

The disadvantage to our approach is that 16 extra bytes must be expropriated to send the encrypted nonce S. However, unlike adding per-hop authentication tags, the reduction in available bandwidth to send messages is fixed, and does not change according to the circuit length. Furthermore, the solution can be built efficiently using familiar components, and is simple enough to allow for fast deployment.

References

1. ISO/IEC JTC 1/SC 27 19772:2009 Information technology – Security techniques – Authenticated encryption. International Organization for Standardization, Geneva, Switzerland
2. Abed, F., Forler, C., List, E., Lucks, S., Wenzel, J.: RIV for robust authenticated encryption. In: Peyrin, T. (ed.) FSE 2016. LNCS, vol. 9783, pp. 23–42. Springer, Heidelberg (2016). doi:10.1007/978-3-662-52993-5_2
3. Andreeva, E., Bogdanov, A., Luykx, A., Mennink, B., Mouha, N., Yasuda K.: How to securely release unverified plaintext in authenticated encryption. In: Sarkar, P., Iwata, T. (eds.) [70], pp. 105–125
4. Andreeva, E., Bogdanov, A., Luykx, A., Mennink, B., Tischhauser, E., Yasuda, K.: Parallelizable and authenticated online ciphers. In: Sako, K., Sarkar, P. (eds.) [69], pp. 424–443
5. Barwell, G., Page, D., Stam, M.: Rogue decryption failures: reconciling AE robustness notions. In: Groth, J. (ed.) [29], pp. 94–111
6. Bellare, M., Namprempre, C.: Authenticated encryption: relations among notions and analysis of the generic composition paradigm. In: Okamoto, T. (ed.) [60], pp. 531–545
7. Bellare, M., Namprempre, C.: Authenticated encryption: relations among notions and analysis of the generic composition paradigm. J. Cryptol. **21**(4), 469–491 (2008)
8. Bellare, M., Rogaway, P.: On the construction of variable-input-length ciphers. In: Knudsen, L.R. (ed.) FSE 1999. LNCS, vol. 1636, pp. 231–244. Springer, Heidelberg (1999). doi:10.1007/3-540-48519-8_17
9. Bellare, M., Rogaway, P.: Encode-then-encipher encryption: how to exploit nonces or redundancy in plaintexts for efficient cryptography. In: Okamoto, T. (ed.) [60], pp. 317–330
10. Bellare, M., Rogaway, P.: The security of triple encryption and a framework for code-based game-playing proofs. In: Vaudenay, S. (ed.) [74], pp. 409–426
11. Bellare, M., Tackmann, B.: The multi-user security of authenticated encryption: AES-GCM in TLS 1.3. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9814, pp. 247–276. Springer, Heidelberg (2016). doi:10.1007/978-3-662-53018-4_10
12. Bernstein, D.J.: The Poly1305-AES message-authentication code. In: Gilbert, H., Handschuh, H. (eds.) FSE 2005. LNCS, vol. 3557, pp. 32–49. Springer, Heidelberg (2005). doi:10.1007/11502760_3
13. Bernstein, D.J.: ChaCha, a variant of Salsa20 (2008). http://cr.yp.to/papers.html#chacha
14. Bernstein, D.J.: 2015.11.20: break a dozen secret keys, get a million more for free. The cr.yp.to blog (2015). https://blog.cr.yp.to/20151120-batchattacks.html
15. Bernstein, D.J.: CAESAR use cases. In: Google Groups: Cryptographic Competitions (2016). https://groups.google.com/forum/#!topic/crypto-competitions/DLv193SPSDc
16. Biham, E.: How to decrypt or even substitute des-encrypted messages in 2^{28} steps. Inf. Process. Lett. **84**(3), 117–124 (2002)
17. Biryukov, A., Mukhopadhyay, S., Sarkar, P.: Improved time-memory trade-offs with multiple data. In: Preneel, B., Tavares, S.E. (eds.) SAC 2005. LNCS, vol. 3897, pp. 110–127. Springer, Heidelberg (2006). doi:10.1007/11693383_8

18. Böck, H., Zauner, A., Devlin, S., Somorovsky, J., Jovanovic, P.: Nonce-disrespecting adversaries: practical forgery attacks on GCM in TLS. In: 10th USENIX Workshop on Offensive Technologies, WOOT 16, Austin, TX, 8–9 August 2016. USENIX Association (2016)
19. Boldyreva, A., Degabriele, J.P., Paterson, K.G., Stam, M.: On symmetric encryption with distinguishable decryption failures. In: Moriai, S. (ed.) FSE 2013. LNCS, vol. 8424, pp. 367–390. Springer, Heidelberg (2014). doi:10.1007/978-3-662-43933-3_19
20. CAESAR: Competition for authenticated encryption: security, applicability, and robustness, May 2014. http://competitions.cr.yp.to/caesar.html
21. Chatterjee, S., Menezes, A., Sarkar, P.: Another look at tightness. In: Miri, A., Vaudenay, S. (eds.) SAC 2011. LNCS, vol. 7118, pp. 293–319. Springer, Heidelberg (2012). doi:10.1007/978-3-642-28496-0_18
22. Coron, J.-S., Dodis, Y., Mandal, A., Seurin, Y.: A domain extender for the ideal cipher. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 273–289. Springer, Heidelberg (2010). doi:10.1007/978-3-642-11799-2_17
23. Dingledine, R., Mathewson, N., Syverson, P.F.: Tor: the second-generation onion router. In: Blaze, M. (ed.) Proceedings of the 13th USENIX Security Symposium, 9–13 August 2004, San Diego, CA, USA, pp. 303–320. USENIX (2004)
24. Dworkin, M.J.: Sp 800–38d. recommendation for block cipher modes of operation: galois/counter mode (gcm) and gmac (2007)
25. Ferguson, N.: Collision attacks on OCB. http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/comments/General_Comments/papers/Ferguson.pdf
26. Ferguson, N., Lucks, S., Schneier, B., Whiting, D., Bellare, M., Kohno, T., Callas, J., Walker, J.: The skein hash function family. http://skein-hash.info/
27. Fleischmann, E., Forler, C., Lucks, S.: McOE: a family of almost foolproof on-line authenticated encryption schemes. In: Canteaut, A. (ed.) FSE 2012. LNCS, vol. 7549, pp. 196–215. Springer, Heidelberg (2012). doi:10.1007/978-3-642-34047-5_12
28. Fouque, P., Joux, A., Mavromati, C.: Multi-user collisions: applications to discrete logarithm, even-mansour and PRINCE. In: Sarkar, P., Iwata, T. (eds.) [70], pp. 420–438
29. Groth, J. (ed.): IMACC 2015. LNCS, vol. 9496. Springer, Cham (2015)
30. Gueron, S.: AES-GCM software performance on the current high end CPUs as a performance baseline for CAESAR competition. Directions in Authenticated Ciphers (DIAC) (2013)
31. Gueron, S., Lindell, Y.: GCM-SIV: full nonce misuse-resistant authenticated encryption at under one cycle per byte. In: Ray, I., Li, N., Kruegel, C. (eds.) Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, 12–16 October 2015, pp. 109–119. ACM (2015)
32. Halevi, S., Rogaway, P.: A parallelizable enciphering mode. In: Okamoto, T. (ed.) CT-RSA 2004. LNCS, vol. 2964, pp. 292–304. Springer, Heidelberg (2004). doi:10.1007/978-3-540-24660-2_23
33. Hirose, S., Sasaki, Y., Yasuda, K.: Iv-fv authenticated encryption and triplet-robust decryption. In: Early Symmetric Crypto, ESC 2015, Clervaux, Luxembourg, 12–16 January 2015
34. Hirose, S., Sasaki, Y., Yasuda, K.: Message-recovery macs and verification-unskippable AE. IACR Cryptol. ePrint Arch. **2017**, 260 (2017)
35. Hoang, V.T., Krovetz, T., Rogaway, P.: Robust authenticated-encryption AEZ and the problem that it solves. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9056, pp. 15–44. Springer, Heidelberg (2015). doi:10.1007/978-3-662-46800-5_2

36. Hoang, V.T., Reyhanitabar, R., Rogaway, P., Vizár, D.: Online authenticated-encryption and its nonce-reuse misuse-resistance. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9215, pp. 493–517. Springer, Heidelberg (2015). doi:10.1007/978-3-662-47989-6_24
37. Imamura, K., Minematsu, K., Iwata, T.: Integrity analysis of authenticated encryption based on stream ciphers. In: Chen, L., Han, J. (eds.) ProvSec 2016. LNCS, vol. 10005, pp. 257–276. Springer, Cham (2016). doi:10.1007/978-3-319-47422-9_15
38. Iwata, T., Ohashi, K., Minematsu, K.: Breaking and repairing GCM security proofs. In: Safavi-Naini, R., Canetti, R. (eds.) [68], pp. 31–49
39. Jean, J., Nikolić, I., Peyrin, T.: Tweaks and keys for block ciphers: the TWEAKEY framework. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014. LNCS, vol. 8874, pp. 274–288. Springer, Heidelberg (2014). doi:10.1007/978-3-662-45608-8_15
40. Jean, J., Nikolić, I., Peyrin, T.: Deoxys v1.3. CAESAR submissions (2015). http://competitions.cr.yp.to/round2/deoxysv13.pdf
41. Jean, J., Nikolić, I., Peyrin, T.: Joltik v1.3. CAESAR submissions (2015). http://competitions.cr.yp.to/round2/joltikv13.pdf
42. Joux, A.: Comments on the draft GCM specification – authentication failuresin NIST version of GCM. http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/comments/800-38_Series-Drafts/GCM/Joux_comments.pdf
43. Krovetz, T., Rogaway, P.: The OCB authenticated-encryption algorithm, June 2013. http://datatracker.ietf.org/doc/draft-irtf-cfrg-ocb
44. Krovetz, T., Rogaway, P.: The OCB authenticated-encryption algorithm. RFC 7253, May 2014
45. Landecker, W., Shrimpton, T., Terashima, R.S.: Tweakable blockciphers with beyond birthday-bound security. In: Safavi-Naini, R., Canetti, R. (ed.) [68], pp. 14–30
46. Leander, G. (ed.): FSE 2015. LNCS, vol. 9054. Springer, Heidelberg (2015)
47. Liskov, M., Rivest, R.L., Wagner, D.: Tweakable block ciphers. J. Cryptol. **24**(3), 588–613 (2011)
48. Mathewson, N.: Cryptographic directions in Tor: past and future. In: Real World Cryptography Conference (2016)
49. McGrew, D.A., Viega, J.: The security and performance of the galois/counter mode (GCM) of operation. In: Canteaut, A., Viswanathan, K. (eds.) INDOCRYPT 2004. LNCS, vol. 3348, pp. 343–355. Springer, Heidelberg (2004). doi:10.1007/978-3-540-30556-9_27
50. McGrew, D.A., Viega, J.: The security and performance of the galois/counter mode of operation (full version). IACR Cryptol. ePrint Arch. **2004**, 193 (2004)
51. Mennink, B.: Optimally secure tweakable blockciphers. In: Leander, G. (ed.) [46], pp. 428–448
52. Minematsu, K.: Beyond-birthday-bound security based on tweakable block cipher. In: Dunkelman, O. (ed.) FSE 2009. LNCS, vol. 5665, pp. 308–326. Springer, Heidelberg (2009). doi:10.1007/978-3-642-03317-9_19
53. Minematsu, K., Iwata, T.: More on generic composition. In: Early Symmetric Crypto (ESC) 2015, pp. 69–71 (2015)
54. Minematsu, K., Iwata, T.: Tweak-length extension for tweakable blockciphers. In: Groth, J. (ed.) [29], pp. 77–93
55. National Institute of Standards and Technology: DES Modes of Operation. FIPS 81, December 1980
56. Nir, Y., Langley, A.: ChaCha20 and Poly1305 for IETF protocols. RFC 7539, May 2015

57. NIST Special Publication 800–38A: Recommendation for block cipher modes of operation - Modes and techniques. National Institute of Standards and Technology (2001)
58. Niwa, Y., Ohashi, K., Minematsu, K., Iwata, T.: GCM security bounds reconsidered. In: Leander, G. (ed.) [46], pp. 385–407
59. Niwa, Y., Ohashi, K., Minematsu, K., Iwata, T.: GCM security bounds reconsidered. IACR Cryptol. ePrint Arch. **2015**, 214 (2015)
60. Okamoto, T. (ed.): ASIACRYPT 2000. LNCS, vol. 1976. Springer, Heidelberg (2000)
61. Procter, G.: A security analysis of the composition of chacha20 and poly1305. IACR Cryptol. ePrint Arch. **2014**, 613 (2014)
62. Procter, G.: The design and analysis of symmetric cryptosystems. Ph.D. thesis (2015)
63. Reyhanitabar, R., Vaudenay, S., Vizár, D.: Authenticated encryption with variable stretch. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016. LNCS, vol. 10031, pp. 396–425. Springer, Heidelberg (2016). doi:10.1007/978-3-662-53887-6_15
64. Rogaway, P.: Efficient instantiations of tweakable blockciphers and refinements to modes OCB and PMAC. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 16–31. Springer, Heidelberg (2004). doi:10.1007/978-3-540-30539-2_2
65. Rogaway, P., Bellare, M., Black, J.: OCB: a block-cipher mode of operation for efficient authenticated encryption. ACM Trans. Inf. Syst. Secur. **6**(3), 365–403 (2003)
66. Rogaway, P., Shrimpton, T.: A provable-security treatment of the key-wrap problem. In: Vaudenay, S. (ed.) [74], pp. 373–390
67. Rogaway, P., Shrimpton, T.: Deterministic authenticated-encryption: a provable-security treatment of the key-wrap problem. IACR Cryptol. ePrint Arch. **2006**, 221 (2006)
68. Safavi-Naini, R., Canetti, R. (eds.): CRYPTO 2012. LNCS, vol. 7417. Springer, Heidelberg (2012)
69. Sako, K., Sarkar, P. (eds.): ASIACRYPT 2013. LNCS, vol. 8269. Springer, Heidelberg (2013)
70. Sarkar, P., Iwata, T. (eds.): ASIACRYPT 2014. LNCS, vol. 8873. Springer, Heidelberg (2014)
71. Shrimpton, T., Terashima, R.S.: A modular framework for building variable-input-length tweakable ciphers. In: Sako, K., Sarkar,P. (eds.) [69], pp. 405–423
72. Syverson, P.F., Goldschlag, D.M., Reed, M.G.: Anonymous connections and onion routing. In: 1997 IEEE Symposium on Security and Privacy, 4–7 May 1997, Oakland, CA, USA, pp. 44–54. IEEE Computer Society (1997)
73. The 23 Raccoons. Analysis of the Relative Severity of Tagging Attacks, March 2012. Email to the Tor developers mailing list https://lists.torproject.org/pipermail/tor-dev/2012-March/003347.html
74. Vaudenay, S. (ed.): EUROCRYPT 2006. LNCS, vol. 4004. Springer, Heidelberg (2006)