Bachelor's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Mathematical Sciences

Sunniva Engan

Provable Security of Authenticated Encryption Schemes

Bachelor's thesis in Mathematical Sciences
Supervisor: Jiaxin Pan
June 2022

NTNU
Norwegian University of
Science and Technology

Sunniva Engan

Provable Security of Authenticated Encryption Schemes

Bachelor's thesis in Mathematical Sciences
Supervisor: Jiaxin Pan
June 2022

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Mathematical Sciences

NTNU

Kunnskap for en bedre verden

Abstract

This bachelor thesis studies the provable security of the randomized counter mode (RCM) and the Galois counter mode (GCM). This is done by making use of sequences of games to structure our proofs. We will prove that the randomized counter mode is CPA-secure under the PRF assumption, and we will prove that the GCM is nonce-based AEAD-secure under the assumption that the underlying block cipher is a secure PRF and that the keyed hash function GHASH is an XOR-DUF.

Contents

Chapter 1

Introduction

Authenticated encryption (AE) schemes are symmetric key encryption schemes that provide both confidentiality and authenticity for the message that is to be encrypted. The need for AE security emerged as one saw that safely combining an encryption only scheme with a message authentication code (MAC) did not always work as intended.[1] One of the paradigms for AE schemes is generic construction, where two examples of ways to combine confidentiality and integrity are Encrypt-then-MAC, and MAC-then-Encrypt. However, only Encrypt-then-MAC guarantees AE security when constructed from a combination of a CPA-secure scheme and a secure MAC. An example of a MAC-then-Encrypt that was shown to be insecure is SSL 3.0 with the POODLE attack.[2]

Counter mode is a block cipher mode of operation. A block cipher is in itself not an encryption scheme, but we use it as a building block for other schemes. A mode of operation is a way of making schemes from a block cipher. Counter mode is a mode that has become a preferable choice for high-speed encryption.[3] However, it does not provide authenticity for data on its own. One of the popular AE schemes that makes use of the counter mode is the Galois counter mode (GCM). It uses a nonce-based counter mode for encryption, and also a keyed hash function to provide authenticity. The counter mode used for encryption and decryption in the GCM is essentially the same as in the randomized counter mode, however it requires the use of unique nonces instead of uniformly distributed elements. GCM was designed to fit the need for an efficient authenticated encryption mode that also was free of patents.[3]

GCM is a mode of operation for block ciphers that supports associated data. That is, in addition to encrypting and authenticating the message that is to be sent, it gives the opportunity to include information that is to be authenticated, but not encrypted. This can for example be packet headers.[3] Furthermore, it can be used as a stand-alone MAC for the associated data if the message string that is to be encrypted is empty. Another good property of the GCM is that it can input nonces of arbitrary length, with the requirement that the nonces must be unique for each encryption query. In this thesis, we focus on the case where the nonce length is 96 bits, and we will look at the GCM for 128-bit block ciphers. GCM has been standardized by NIST, the U.S. National Institute of Standards and Technology.[4]

The goal of this thesis is to prove that the GCM is AEAD-secure. To do so, we first introduce some helpful definitions. Then we prove the difference lemma, which is a central lemma in provable security. After that we introduce and prove the security of the randomized counter mode. The RCM is included because its security proof is similar to one of the proofs we need for the security of the GCM, and also to get used to the proof framework used throughout this thesis. We finish with the security proof of GCM, showing that it is AEAD-secure under the assumption that the underlying block cipher is a secure PRF and that GHASH is an XOR-DUF.

Chapter 2

Definitions

In this thesis, we use techniques and code-based games from [5] to write our schemes, security definitions and proofs. We also use techniques from [6] for the sequences of games. This way of writing and structuring the proofs is chosen because it organizes the definitions and transitions for our game-based proofs well.

In order to talk about the security of different cryptographic schemes, we need to establish some various forms of security a scheme can possess. We begin our discussion by establishing what terms such as "negligible", "poly-bounded" and "super-poly" mean. For our scheme to be secure, we would like the probability that an adversary trying to break our scheme succeeds to be negligible. The negligibility of the success of an adversary is connected to the security parameter of the scheme, however there will not be a detailed discussion regarding this connection in this thesis.

Definition 2.0.1 (Negligible function)**.** A function $f : \mathbb{N} \to \mathbb{R}$ is said to be *negligible* if for any polynomial p, we have that

$$\lim_{x \to \infty} p(x)f(x) = 0$$

■

Definition 2.0.2 (Poly-bounded)**.** We say that a function $f : \mathbb{N}_{\geq 1} \to \mathbb{R}$ is *poly-bounded* if there exists two constants $c, d, \in \mathbb{R}^+$ such that for all integers $n \geq 0$, $|f(n)| \leq n^c + d$. ■

Definition 2.0.3 (Super-poly)**.** A function $f : \mathbb{Z} \to \mathbb{R}$ is said to be *super-poly* if $\frac{1}{f}$ is negligible. ■

Remark 2.0.4. We will use the following facts about negligible, poly-bounded and super-poly values. If ϵ and ϵ' are negligible values and Q and Q' are poly-bounded values, then the following holds:

- $\epsilon \cdot Q$ is a negligible value
- $\epsilon + \epsilon'$ is a negligible value
- $Q + Q'$ and $Q \cdot Q'$ are poly-bounded values

We will now give some other basic definitions that will be used to continue our discussion into the security proofs of different cryptographic schemes.

Definition 2.0.5 (Security parameter)**.** The *security parameter* is a parameter for a cryptographic scheme that decides the level of security. ■

Definition 2.0.6 (Polynomial time)**.** An algorithm is said to run in *polynomial time* if for each input λ, the algorithm is $\mathcal{O}(\lambda^k)$ for some fixed k. ■

Definition 2.0.7 (PPT algorithm)**.** An algorithm is a *probabilistic polynomial time algorithm* (PPT algorithm) if it runs in polynomial time and uses randomness. ■

Definition 2.0.8 (Flag)**.** A *flag* is a boolean variable in a game that changes its values at most once from its initial value as false. Once a flag becomes true, it never turns false again. ■

Definition 2.0.9 (Identical-until-bad games)**.** Two games G and H are called *identical-until-bad games* if they are syntactically equivalent except for what follows the setting of a flag to true. ■

Definition 2.0.10 (Correctness property)**.** The correctness property for a cryptographic scheme π = (Gen, Enc, Dec) states that for all keys $k \in \mathcal{K}$ and for all messages $m \in \mathcal{M}$, the following computation

$$m' \leftarrow \mathsf{Dec}(k, \mathsf{Enc}(k, m))$$

yields

$$\Pr[m' = m] = 1.$$

The correctness property ensures that a given scheme correctly relates the decryption of the ciphertexts to their corresponding encrypted messages. ■

The following security definitions are the ones we will make use of for the rest of this thesis.

Definition 2.0.11 (CPA-security, bit-guessing version)**.** Let $\pi = (\mathsf{Enc}, \mathsf{Dec})$ be a cipher defined over $(\mathcal{K}, \mathcal{M}, \mathcal{C})$, where $\mathcal{K}$ is the key space, $\mathcal{M}$ is the message space and $\mathcal{C}$ is the ciphertext space. Let $\mathcal{A}$ be a PPT adversary, and let Q be the number of encryption queries made by adversary $\mathcal{A}$. We define the following game:

Game IND-CPA:	**Oracle** Enc(m_0, m_1): // Q queries
1 : $b \leftarrow\$ \{0,1\}$	1 : $c \leftarrow\$ \mathsf{Enc}(k, m_b)$
2 : $k \leftarrow\$ \mathcal{K}$	2 : **return** c
3 : $b' \leftarrow\$ \mathcal{A}^{\mathrm{Enc}(\cdot,\cdot)}$	
4 : **return** $b = b'$	

We define $\mathcal{A}$'s advantage with respect to π as

$$\mathsf{Adv}^{\text{ind-cpa}}_{\pi,\mathcal{A}}(n) = |\Pr[\text{IND-CPA}^{\mathcal{A}} \Rightarrow 1] - \frac{1}{2}|.$$

The cipher π is said to be *semantically secure against chosen plaintext attacks* (CPA-secure), if $\mathsf{Adv}^{\text{ind-cpa}}_{\pi,\mathcal{A}}(n)$ is negligible for all PPT adversaries $\mathcal{A}$ in the security parameter n. ■

Remark 2.0.12. When using the arrow " $\leftarrow\$$ ", it carries two different meanings. When used to the left of an algorithm such as the adversary, the arrow indicates that the algorithm that outputs some value is probabilistic. However, if it is used to the left of some set it indicates that one samples an element from that set uniformly.

Definition 2.0.13 (CI-security)**.** Let $\pi = (\mathsf{Enc}, \mathsf{Dec})$ be a cipher defined over $(\mathcal{K}, \mathcal{M}, \mathcal{C})$, where $\mathcal{K}$ is the key space, $\mathcal{M}$ is the message space and $\mathcal{C}$ is the ciphertext space. Let $\mathcal{A}$ be a PPT adversary, and let Q be the number of encryption queries made by adversary $\mathcal{A}$. We define the following game:

Game CI:	**Oracle** Enc(m): // Q queries
1 : $k \leftarrow\$ \mathcal{K}$	1 : $c \leftarrow\$ \mathsf{Enc}(k, m)$
2 : $c \leftarrow\$ \mathcal{A}^{\mathrm{Enc}(\cdot)}$	2 : $L_C = L_C \cup \{c\}$
3 : **return** $(c \notin L_C) \wedge (\mathsf{Dec}(k, c) \neq \perp)$	3 : **return** c

We define $\mathcal{A}$'s advantage with respect to π as

$$\mathsf{Adv}^{\mathrm{ci}}_{\pi,\mathcal{A}}(n) = \Pr[\mathrm{CI}^{\mathcal{A}} \Rightarrow 1].$$

The cipher π is said to provide *ciphertext integrity* (or to be CI-secure), if $\mathsf{Adv}^{\mathrm{ci}}_{\pi,\mathcal{A}}(n)$ is negligible for all PPT adversaries $\mathcal{A}$ in the security parameter n. ■

Definition 2.0.14 (AE-secure)**.** A scheme π is said to provide authenticated encryption or to be *AE-secure* if the cipher is both CPA-secure and CI-secure. ■

Definition 2.0.15 (Pseudo-random function)**.** A *pseudo-random function (PRF)*

$$F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$$

is a deterministic algorithm defined over $(\mathcal{K}, \mathcal{X}, \mathcal{Y})$ where $\mathcal{K}$ is the key space, $\mathcal{X}$ is the input space, and $\mathcal{Y}$ is the output space. All these sets are finite. Let $\mathrm{Funs}[\mathcal{X}, \mathcal{Y}]$ denote the set of all functions $f : \mathcal{X} \to \mathcal{Y}$. ■

Definition 2.0.16 (Secure PRF, left-or-right style)**.** Let F be a PRF defined over $(\mathcal{K}, \mathcal{X}, \mathcal{Y})$, where $\mathcal{K}$ is the key space, $\mathcal{X}$ is the input space and $\mathcal{Y}$ is the output space. Let $\mathcal{A}$ be a PPT adversary, and let Q be the number of queries the adversary $\mathcal{A}$ makes. We define the following game:

Game LR-PRF$_0$/LR-PRF$_1$:	**Oracle** Eval(x): // Q queries
1 : $k \leftarrow\$ \mathcal{K}$	1 : $y \leftarrow f(x)$
2 : $f \leftarrow F(k, \cdot)$	2 : **return** y
3 : $f \leftarrow \mathrm{Funs}[\mathcal{X}, \mathcal{Y}]$	
4 : $b' \leftarrow\$ \mathcal{A}^{\mathrm{Eval}(\cdot)}$	
5 : **return** b'	

We define $\mathcal{A}$'s advantage with respect to F as

$$\mathsf{Adv}^{\text{lr-prf}}_{F,\mathcal{A}}(n) = |\Pr[\text{LR-PRF}^{\mathcal{A}}_0 \Rightarrow 1] - \Pr[\text{LR-PRF}^{\mathcal{A}}_1 \Rightarrow 1]|.$$

We say that a F is *PRF secure* if $\mathsf{Adv}^{\text{lr-prf}}_{F,\mathcal{A}}(n)$ is negligible for all PPT adversaries $\mathcal{A}$ in the security parameter n. ■

Definition 2.0.17 (Secure block cipher)**.** A block cipher $\mathrm{B} = (E, D)$ is a deterministic cipher defined over $(\mathcal{K}, \mathcal{X})$ where $\mathcal{K}$ denotes the key space and $\mathcal{X}$ is called the data block space. An element of $\mathcal{X}$ is called a data block. For each key $k \in \mathcal{K}$, we can define a function $f_k = \mathsf{Enc}(k, \cdot)$ which is a bijection by the correctness property of the deterministic encryption algorithm and that $\mathcal{X}$ is a finite set. Let $\mathrm{Perm}[\mathcal{X}]$ Denote the set of all permutations on the data block space.

Let $\mathcal{A}$ be a PPT adversary, and let Q be the number of queries made by adversary $\mathcal{A}$. We define the following game:

Game LR-BC$_0$/LR-BC$_1$:	**Oracle** Eval(x): // Q queries
1 : $k \leftarrow\$ \mathcal{K}$	1 : $y \leftarrow f(x)$
2 : $f \leftarrow E(k, \cdot)$	2 : **return** y
3 : $f \leftarrow \mathrm{Perm}[\mathcal{X}]$	
4 : $b' \leftarrow\$ \mathcal{A}^{\mathrm{Eval}(\cdot)}$	
5 : **return** b'	

We define $\mathcal{A}$'s advantage with respect to E as

$$\mathsf{Adv}^{\text{lr-bc}}_{E,\mathcal{A}}(n) = |\Pr[\text{LR-BC}_0^{\mathcal{A}} \Rightarrow 1] - \Pr[\text{LR-BC}_1^{\mathcal{A}} \Rightarrow 1]|.$$

We say that a block cipher B is *BC secure* if $\mathsf{Adv}^{\text{lr-bc}}_{E,\mathcal{A}}(n)$ is negligible for all PPT adversaries $\mathcal{A}$ in the security parameter n. ∎

We will now state the security definitions we need for AEAD security. Rogaway formalized the AEAD problem in 2002 [7], where the goal was to have the opportunity of providing both privacy and authenticity for some information, and only authenticity for other data that is to be sent in clear text.

Definition 2.0.18 (Nonce-based CPA-security with associated data)**.** Let $\pi = (\mathsf{Enc}, \mathsf{Dec})$ be a nonce-based AD cipher defined over $(\mathcal{K}, \mathcal{M}, \mathcal{C}, \mathcal{N}, \mathcal{D})$, where $\mathcal{K}$ is the key space, $\mathcal{M}$ is the message space, $\mathcal{C}$ is the ciphertext space, $\mathcal{N}$ is the nonce space and $\mathcal{D}$ is the associated data space. Let $\mathcal{A}$ be a PPT adversary, and let Q be the number of encryption queries made by adversary $\mathcal{A}$. We define the following game:

Game nCPA_{ad}:	**Oracle** $\text{Enc}(m_0, m_1, n, d)$: // Q queries
1 : $b \leftarrow\$ \{0,1\}$	1 : **if** $n \in L_n$:
2 : $k \leftarrow\$ \mathcal{K}$	2 : **return** $\perp$
3 : $b' \leftarrow\$ \mathcal{A}^{\text{Enc}(\cdot,\cdot,\cdot,\cdot)}$	3 : $L_n = L_n \cup \{n\}$
4 : **return** $b = b'$	4 : $c \leftarrow \mathsf{Enc}(k, m_b, n, d)$
	5 : **return** c

We define $\mathcal{A}$'s advantage with respect to π as

$$\mathsf{Adv}^{\text{ncpa}_{\text{ad}}}_{\pi,\mathcal{A}}(n) = |\Pr[\text{nCPA}_{\text{ad}}{}^{\mathcal{A}} \Rightarrow 1] - \frac{1}{2}|.$$

The nonce-based AD cipher π is said to be *nonce-based semantically secure against chosen plaintext attack* (or to be nCPA_{ad}-secure), if $\mathsf{Adv}^{\text{ncpa}_{\text{ad}}}_{\pi,\mathcal{A}}(n)$ is negligible for all PPT adversaries $\mathcal{A}$ in the security parameter n. ∎

Definition 2.0.19 (Nonce-based CI-security with associated data)**.** Let $\pi = (\mathsf{Enc}, \mathsf{Dec})$ be a nonce-based AD-cipher defined over $(\mathcal{K}, \mathcal{M}, \mathcal{C}, \mathcal{N}, \mathcal{D})$, where $\mathcal{K}$ is the key space, $\mathcal{M}$ is the message space, $\mathcal{C}$ is the ciphertext space, $\mathcal{N}$ is the nonce space and $\mathcal{D}$ is the associated data space. Let $\mathcal{A}$ be a PPT adversary, and let Q be the number of encryption queries made by adversary $\mathcal{A}$. We define the following game:

Game nCI_{ad}:	**Oracle** $\text{Enc}(m, n, d)$: // Q queries
1 : $k \leftarrow\$ \mathcal{K}$	1 : **if** $n \in L_N$:
2 : $(c^*, n^*, d^*) \leftarrow\$ \mathcal{A}^{\text{Enc}(\cdot,\cdot,\cdot)}$	2 : **return** $\perp$
3 : **return** $((c^*, n^*, d^*) \notin \text{L}_{\text{CND}}) \wedge (\mathsf{Dec}(k, c) \neq \perp)$	3 : $L_N = L_N \cup \{n\}$
	4 : $c \leftarrow \mathsf{Enc}(k, m, n, d)$
	5 : $\text{L}_{\text{CND}} = \text{L}_{\text{CND}} \cup (c, n, d)$
	6 : **return** c

We define $\mathcal{A}$'s advantage with respect to π as

$$\mathsf{Adv}^{\text{nci}_{\text{ad}}}_{\pi,\mathcal{A}}(n) = \Pr[\text{nCI}_{\text{ad}}{}^{\mathcal{A}} \Rightarrow 1].$$

The nonce-based AD-cipher π is said to provide *nonce-based ciphertext integrity with associated data* (or to be nCI_{ad}-secure), if $\mathsf{Adv}^{\text{nci}_{\text{ad}}}_{\pi,\mathcal{A}}(n)$ is negligible for all PPT adversaries $\mathcal{A}$ in the security parameter n. ∎

Remark 2.0.20. The nonce-based encryption algorithm is deterministic. Instead of having a probabilistic algorithm and a larger ciphertext space, we require that each encryption query provides a unique nonce.

Definition 2.0.21 (AEAD-secure)**.** A scheme π is said to provide authenticated encryption with associated data or to be *nonce-based AEAD-secure* if the cipher is both nCPA_{ad} secure and nCI_{ad} secure. ∎

Definition 2.0.22 (Computational DUF)**.** Let H be a keyed hash function defined over $(\mathcal{K}, \mathcal{M}, \mathcal{T})$ Where $\mathcal{K}$ is the key space, $\mathcal{M}$ is the message space and $\mathcal{T} = \mathbb{Z}_N$ is the digest space, and let $\mathcal{A}$ be a PPT adversary. We define the following game:

Game DUF:

$k \leftarrow\$ \mathcal{K}$
$(m_0, m_1, \delta) \leftarrow\$ \mathcal{A}$
if $H(k, m_1) - H(k, m_0) = \delta$:
  return $= 1$
else :
  return $= 0$

We define $\mathcal{A}$'s advantage with respect to H as

$$\mathsf{Adv}^{\text{duf}}_{H,\mathcal{A}}(n) = \Pr[\text{DUF}^{\mathcal{A}} \Rightarrow 1].$$

We say that H is a *computational difference unpredictability function* (or to be a computational DUF) if $\mathsf{Adv}^{\text{duf}}_{H,\mathcal{A}}(n)$ is negligible for all PPT adversaries $\mathcal{A}$ in the security parameter n. ∎

Remark 2.0.23. One can also have the digest space $\mathcal{T} = \{0,1\}^n$, where we instead use XOR as the difference operator. In that case, δ denotes the bit string one gets from using the XOR operation on the hash values of the input messages. The adversary then has a valid forgery if it can output two messages and the valid delta corresponding to them.

Definition 2.0.24 (XOR-DUF)**.** Let H be a keyed hash function defined over $(\mathcal{K}, \mathcal{M}, \mathcal{T})$ Where $\mathcal{K}$ is the key space, $\mathcal{M}$ is the message space and $\mathcal{T} = \{0,1\}^n$ is the digest space, and let $\mathcal{A}$ be a PPT adversary. We define the following game:

Game XOR-DUF:

$k \leftarrow\$ \mathcal{K}$
$(m_0, m_1, \delta) \leftarrow\$ \mathcal{A}$
if $H(k, m_1) \oplus H(k, m_0) = \delta$:
  return $= 1$
else :
  return $= 0$

We define $\mathcal{A}$'s advantage with respect to H as

$$\mathsf{Adv}^{\text{xor-duf}}_{H,\mathcal{A}}(n) = \Pr[\text{XOR-DUF}^{\mathcal{A}} \Rightarrow 1].$$

We say that H is an XOR-DUF if $\mathsf{Adv}^{\text{xor-duf}}_{H,\mathcal{A}}(n)$ is negligible for all PPT adversaries $\mathcal{A}$ in the security parameter n. ∎

We end this part of the thesis by stating and proving the difference lemma, which gives us an important tool for bounding the difference of two identical-until-bad games.

Lemma 2.0.25 (**Difference lemma**). *Let G and H be identical-until-bad games, and let $\mathcal{A}$ be an adversary. Let $Bool_G$ be the event that the flag Bool turns true during the execution of $\mathcal{A}$ in game G, and let $Bool_H$ be the event that the flag turns true during the execution of $\mathcal{A}$ in game H. Then the following inequality holds:*

$$\mathsf{Adv}(G^{\mathcal{A}}, H^{\mathcal{A}}) = |Pr[G^{\mathcal{A}} \Rightarrow 1] - Pr[H^{\mathcal{A}} \Rightarrow 1]| \leq Pr[Bool_G] = Pr[Bool_H].$$

Proof. Since G and H are identical-until-bad games, we know that they are syntactically equivalent until the flag Bool. Also, if the flag never changes value, the games are identical. Therefore

$$\Pr[\mathrm{Bool_G}] = \Pr[\mathrm{Bool_H}].$$

Also, since the games are identical if the flag does not change its truth value, we have that

$$\Pr[\mathrm{G}^{\mathcal{A}} \Rightarrow 1 | \neg \mathrm{Bool_G}] = \Pr[\mathrm{H}^{\mathcal{A}} \Rightarrow 1 | \neg \mathrm{Bool_H}].$$

Now calculations give us that

$$\begin{aligned}
\mathsf{Adv}(\mathrm{G}^{\mathcal{A}}, \mathrm{H}^{\mathcal{A}}) &= |\Pr[\mathrm{G}^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathrm{H}^{\mathcal{A}} \Rightarrow 1]| \\
&= |\Pr[\mathrm{G}^{\mathcal{A}} \Rightarrow 1 | \neg \mathrm{Bool_G}] + \Pr[\mathrm{G}^{\mathcal{A}} \Rightarrow 1 | \mathrm{Bool_G}] - (\Pr[\mathrm{H}^{\mathcal{A}} \Rightarrow 1 | \neg \mathrm{Bool_H}] + \Pr[\mathrm{H}^{\mathcal{A}} \Rightarrow 1 | \mathrm{Bool_H}])| \\
&= |\Pr[\mathrm{G}^{\mathcal{A}} \Rightarrow 1 | \mathrm{Bool_G}] - \Pr[\mathrm{H}^{\mathcal{A}} \Rightarrow 1 | \mathrm{Bool_H}]| \\
&\leq \Pr[\mathrm{Bool_G}] = \Pr[\mathrm{Bool_H}].
\end{aligned}$$

Thus the inequality holds. □

Chapter 3

Randomized Counter Mode (RCM)

In this section we look at the randomized counter mode. We will show that it is CPA-secure under the assumption that the function we build the scheme from is a secure PRF. The proof follows the one in "A Graduate course in Applied Cryptography" in section 5.4.2.[2] We formalize the proof from the book using code-based sequences of games.

Construction 3.0.1 (Randomized counter mode). Let F be a secure PRF defined over $(\mathcal{K}, \mathcal{X}, \mathcal{Y})$, where $\mathcal{K}$ is the key space, $\mathcal{X} = \{0, \ldots, N-1\}$ for $N \in \mathbb{N}$ is the input space and $\mathcal{Y} = \{0,1\}^h$ for $h \in \mathbb{N}$ is the output space. For some poly-bounded $l \geq 1$, we define the scheme RCM $= (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ over $(\mathcal{K}, \mathcal{M}, \mathcal{C}) = (\mathcal{K}, \mathcal{Y}^{\leq l}, \mathcal{X} \times \mathcal{Y}^{\leq l})$ as follows:

$\mathsf{Gen}(1^n)$:	$\mathsf{Enc}(k, m)$:	$\mathsf{Dec}(k, \mathsf{ct})$:
1 : $k \leftarrow\$ \mathcal{K}$	1 : $v = \lvert m \rvert$	1 : **parse** $\mathsf{ct} = (x, c)$
2 : **return** k	2 : $x \leftarrow\$ \mathcal{X}$	2 : $v = \lvert c \rvert$
	3 : **for** $j = 0 \ldots v-1$ **do** :	3 : **for** $j = 0 \ldots v-1$ **do** :
	4 : $\quad c[j] \leftarrow F(k, x+j \mod N) \oplus m[j]$	4 : $\quad m[j] \leftarrow F(k, x+j \mod N) \oplus c[j]$
	5 : $\mathsf{ct} = (x, c)$	5 : **return** m
	6 : **return** ct	

For notation, the function $|\cdot|$ outputs a non-negative integer which is the number of blocks in the input.
Correctness: We now show the correctness of the RCM scheme. Let $m' \in \mathcal{M}$ and $k \in \mathcal{K}$. We compute the encryption and decryption using the chosen message and key as follows:

$\mathsf{Enc}(k, m')$:	$\mathsf{Dec}(k, \mathsf{ct}')$:
$v = \lvert m' \rvert$	**parse** $\mathsf{ct}' = (x, c')$
$x \leftarrow\$ \mathcal{X}$	$v = \lvert c' \rvert$
for $j = 0 \ldots v-1$ **do** :	**for** $j = 0 \ldots v-1$ **do** :
$\quad c'[j] \leftarrow F(k, x+j \mod N) \oplus m'[j]$	$\quad m[j] \leftarrow F(k, x+j \mod N) \oplus c'[j]$
$\mathsf{ct}' := (x, c')$	$\quad m[j] \leftarrow F(k, x+j \mod N) \oplus (F(k, x+j \mod N) \oplus m'[j])$
return ct'	$\quad m[j] \leftarrow 0^h \oplus m'[j]$
	$\quad m[j] \leftarrow m'[j]$
	return m

Since $m[j] = m'[j]$ for each $j = 0, \ldots, v-1$, we have that $m = m'$. By this, correctness holds.

Theorem 3.0.2. *If F is a secure PRF and N is super-poly, then for any poly-bounded $l \geq 1$ and poly-bounded Q denoting the number of encryption queries, the cipher RCM described above is a CPA-secure cipher.*

In particular, for every PPT adversary $\mathcal{A}$ playing the Q-query CPA security game against the RCM, there exists a PPT adversary $\mathcal{B}$ that plays the PRF security game against F such that

$$\mathsf{Adv}^{\textit{ind-cpa}}_{RCM,\mathcal{A}}(n) \leq \frac{Q^2 l}{N} + \mathsf{Adv}^{\textit{lr-prf}}_{F,\mathcal{B}}(n).$$

Proof. We give four games. Game 0 is exactly the CPA-security game between the adversary $\mathcal{A}$ and the scheme RCM. Game 1 is the same as Game 0 except that we use the assumption that the function F is a secure PRF, so we switch F out with a truly random function. In Game 2 we switch F out by drawing uniformly distributed values directly from the output space. Game 2 and Game 3 are identical-until-bad games, where Game 3 keeps track of earlier values for its input, while Game 2 do not. The description of these games are as follows:

Game $G_0/G_1/G_2/G_3$:		`Oracle` $\mathtt{Enc}(m_0, m_1)$: `//`Q `queries`	
1 :	$b \leftarrow\$ \{0,1\}$	1 :	$v = \lvert m_0 \rvert = \lvert m_1 \rvert$
2 :	$k \leftarrow\$ \mathcal{K}$	2 :	$x \leftarrow\$ \mathcal{X}$
3 :	$f \leftarrow F(k, \cdot)$	3 :	**for** $j = 0 \ldots l-1$ **do** :
4 :	$f \leftarrow\$ \mathrm{Funs}[\mathcal{X}, \mathcal{Y}]$	4 :	$x_j \leftarrow x + j \mod N$
5 :	$b' \leftarrow\$ \mathcal{A}^{\mathrm{Enc}(\cdot,\cdot)}$	5 :	$y_j \leftarrow f(x_j)$ $y_j \leftarrow\$ \mathcal{Y}$
6 :	**return** $b = b'$	6 :	**if** $x_j \in L_x$: bad $\leftarrow$ true;
		7 :	$y_j \leftarrow \Sigma[x_j]$
		8 :	$L_x \leftarrow L_x \cup \{x_j\}$
		9 :	$\Sigma[x_j] := y_j$
		10 :	**for** $j = 0 \ldots v-1$ **do** :
		11 :	$c[j] \leftarrow y_j \oplus m_b[j]$
		12 :	**return** (x, c)

We follow the convention that for the sequence of games, each of the coloured statements are executed in their respective game and so on. That is, the red statement is executed in Game 1 and so on, and similarly for the other colours/games. The non-coloured pseudo-code is executed in all games.

We now analyze the above games in more detail. We assume that the underlying probability space is the same in all four games. This is so that the flags in Game 2 and Game 3 turn identically. By our construction of Game 0, we have that

$$\mathsf{Adv}^{\text{ind-cpa}}_{\text{RCM},\mathcal{A}}(n) = \lvert \Pr[\mathrm{G}_0^{\mathcal{A}} \Rightarrow 1] - \frac{1}{2} \rvert.$$

We can make a reduction $\mathcal{B}$ against the PRF security of F that simulates Game 0 and Game 1 for $\mathcal{A}$. By constructing $\mathcal{B}$, we can connect Game 0 and Game 1 to the PRF security. It inputs the function f and has to distinguish if f is a truly random function or if it is the real function F. Let LR-PRF$_0$ denote the game where where f is the real function F with the correct key as its first input, and let LR-PRF$_1$ denote the game where f is a truly random function. Adversary $\mathcal{B}$ is defined as follows:

Adversary $\mathcal{B}(f)$:	**Oracle** $\text{Enc}(m_0, m_1)$: //Q queries
1 : $b \leftarrow\$ \{0,1\}$	1 : $v = \lvert m_0\rvert = \lvert m_1\rvert$
2 : $b' \leftarrow\$ \mathcal{A}^{\text{Enc}(\cdot,\cdot)}$	2 : $x \leftarrow\$ \mathcal{X}$
3 : **return** $b = b'$	3 : **for** $j = 0 \ldots l-1$ **do** :
	4 : $\quad x_j \leftarrow x + j \mod N$
	5 : $\quad y_j \leftarrow f(x_j)$
	6 : **for** $j = 0 \ldots v-1$ **do** :
	7 : $\quad c[j] \leftarrow y_j \oplus m_b[j]$
	8 : **return** (x, c)

If $\mathcal{B}$ is in Game 0 of the PRF games, then $f = F(k, \cdot)$, and therefore $\Pr[\text{LR-PRF}_0^{\mathcal{B}} \Rightarrow 1] = \Pr[\text{G}_0^{\mathcal{A}} \Rightarrow 1]$. If $\mathcal{B}$ is in Game 1 of the PRF games, then $f \leftarrow\$ \text{Funs}[\mathcal{X}, \mathcal{Y}]$, and we have $\Pr[\text{LR-PRF}_1^{\mathcal{B}} \Rightarrow 1] = \Pr[\text{G}_1^{\mathcal{A}} \Rightarrow 1]$. Therefore

$$\begin{aligned}\mathsf{Adv}^{\text{lr-prf}}_{F,\mathcal{B}}(n) &= \lvert\Pr[\text{LR-PRF}_0^{\mathcal{B}} \Rightarrow 1] - \Pr[\text{LR-PRF}_1^{\mathcal{B}} \Rightarrow 1]\rvert \\ &= \lvert\Pr[\text{G}_0^{\mathcal{A}} \Rightarrow 1] - \Pr[\text{G}_1^{\mathcal{A}} \Rightarrow 1]\rvert.\end{aligned}$$

Now we move on to Game 2 and Game 3. We begin discussing Game 3 since it is connected to Game 1. This game can be viewed as having a direct implementation of the function f from Game 1. It draws uniformly distributed elements directly from the output space $\mathcal{Y}$, and assigns them to x_j-values in the same manner as a random function would do. That is, it keeps track of previously assigned values. We have that

$$\Pr[\text{G}_1^{\mathcal{A}} \Rightarrow 1] = \Pr[\text{G}_3^{\mathcal{A}} \Rightarrow 1].$$

In Game 2, the adversary's output bit b' is independent of b since the y_j-values are independently distributed over the output space. Therefore $\Pr[\text{G}_2^{\mathcal{A}} \Rightarrow 1] = \frac{1}{2}$.

Since Game 2 and Game 3 are identical-until-bad games, we have by the difference lemma that

$$\lvert\Pr[\text{G}_2^{\mathcal{A}} \Rightarrow 1] - \Pr[\text{G}_3^{\mathcal{A}} \Rightarrow 1]\rvert \leq \Pr[\mathsf{bad}].$$

We want to analyze $\Pr[\mathsf{bad}]$ to find an upper bound for it. The flag turns true if two x_j-values from two different encryption queries are the same. So we want to find the probability of two x_j-values having a collision.

For two values $a \leftarrow\$ \mathcal{X}$ and $b \leftarrow\$ \mathcal{X}$ used as the initial counting value in two different encryption queries, let $S = \{a \mod N, \ldots a + l - 1 \mod N\}$ and $T = \{b \mod N, \ldots b + l - 1 \mod N\}$ for notation. We have a collision if

$$S \cap T \neq \emptyset.$$

T intersects S if $b \in \{a + j \mid -l + 1 \leq j \leq l + 1\}$. Assuming that $N \geq 2l$ (which is reasonable, since N is super-poly and l is poly-bounded), this happens with a probability of $\frac{2l-1}{N}$. This is because the number of elements in $\{a + j \mid -l + 1 \leq j \leq l + 1\}$ is $2l - 1$, and the number of elements in $\mathcal{X}$ is N.

By the union bound, for a countable set of events $A_1, A_2, \ldots$ we have that

$$\Pr[\bigcup_{i=1}^{\infty} A_i] \leq \sum_{i=1}^{\infty} \Pr[A_i].$$

We use this for our bound on Pr[bad]. As we saw above, the probability of two encryption queries to have a collision of counter values is $\frac{2l-1}{N}$, and there are Q encryption queries. Therefore there are $\binom{Q}{2}$ ways to have a collision with the given probability. By this, we have that

$$\Pr[\mathsf{bad}] \leq \frac{Q(Q-1)}{2}\frac{2l-1}{N} \leq \frac{2Q^2l}{2N} = \frac{Q^2l}{N}.$$

This gives us that

$$\Pr[\mathsf{bad}] \leq \frac{Q^2l}{N}.$$

Since N is super-poly, $\frac{1}{N}$ is by definition a negligible value. Therefore by properties of negligible values, $l \cdot \frac{1}{N} = \frac{l}{N}$ is also a negligible value since l is poly-bounded. In addition, Q^2 is poly-bounded since Q is poly-bounded. Therefore the bound $\frac{Q^2l}{N}$ is still negligible and so is Pr[bad].

Connecting our games we have by the triangle inequality that

$$\begin{aligned}
\mathsf{Adv}^{\text{ind-cpa}}_{\text{RCM},\mathcal{A}}(n) &= |\Pr[\mathrm{G}_0^{\mathcal{A}} \Rightarrow 1]| - \frac{1}{2}| \\
&\leq |\Pr[\mathrm{G}_0^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathrm{G}_1^{\mathcal{A}} \Rightarrow 1]| + |\Pr[\mathrm{G}_1^{\mathcal{A}} \Rightarrow 1] - \frac{1}{2}| \\
&= |\Pr[\mathrm{G}_0^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathrm{G}_1^{\mathcal{A}} \Rightarrow 1]| + |\Pr[\mathrm{G}_3^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathrm{G}_2^{\mathcal{A}} \Rightarrow 1]| \\
&\leq \mathsf{Adv}^{\text{lr-prf}}_{F,\mathcal{B}}(n) + \Pr[\mathsf{bad}] \\
&\leq \frac{Q^2l}{N} + \mathsf{Adv}^{\text{lr-prf}}_{F,\mathcal{B}}(n).
\end{aligned}$$

We have shown that $\frac{Q^2l}{N}$ is a negligible value, and $\mathsf{Adv}^{\text{lr-prf}}_{F,\mathcal{B}}(n)$ is negligible by our assumption that F is a secure PRF. Thus by the properties of negligible values, $\mathsf{Adv}^{\text{ind-cpa}}_{\text{RCM},\mathcal{A}}(n)$ is negligible as well. This concludes our proof of the CPA-security of the randomized counter mode. □

Remark 3.0.3. In the theorem we assumed N to be super-poly, because we wanted $\frac{1}{N}$ to be negligible. One could also assume N to be exponential and we would have the same result, however super-poly is a weaker assumption, and an exponentially large N makes the algorithm less efficient.

Chapter 4

Galois Counter Mode (GCM)

In this section we will look at the Galois counter mode, which was designed by David A. McGrew and John Viega. It shares similarities with the randomized counter mode, and its $nCPA_{ad}$ security proof is essentially the same as the CPA proof of RCM. Instead of using a uniformly chosen value for its counter, GCM use a unique nonce to make its counting values for each encryption query. In this thesis will look at GCM constructed from the 128-bit AES. We assume that the length of the nonces are fixed to 96 bits for simplicity, and we also assume that each message to be encrypted has a length which is a multiple of 128, so we are always encrypting complete blocks. Normally the only assumption on the length of the plaintext, associated data and the nonce is that is has to be a multiple of 8.[4] Other than the specifications given above, we follow the length requirements of input as the ones stated in [3].

Construction 4.0.1 (Galois counter mode). Let $\mathrm{B} = (E, D)$ be a block cipher defined over $(\mathcal{K}, \mathcal{X})$ where $\mathcal{K} = \{0,1\}^{128}$ is the key space and $\mathcal{X} = \{0,1\}^{128}$ is the data block space. Let GHASH be a keyed hash function defined over $(\mathcal{K}, \mathcal{M}, \mathcal{T})$ where $\mathcal{K} = \{0,1\}^{128}$ is the key space, $\mathcal{M} = \mathcal{X}^{\leq l}$ for some positive integer l is the message space, and $\mathcal{T} = \{0,1\}^{128}$ is the digest space. For 96-bit nonces, we define $\mathrm{GCM} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ over $(\mathcal{K}, \mathcal{M}, \mathcal{C}, \mathcal{D}, \mathcal{N})$ where $\mathcal{K} = \{0,1\}^{128}$ is the key space, $\mathcal{M} = \mathcal{X}^{\leq 2^{32}-2}$ is the message space, $\mathcal{C} = \mathcal{X}^{\leq 2^{32}-2}$ is the ciphertext space, $\mathcal{D} = \{0,1\}^{2^{64}}$ is the associated data space and $\mathcal{N} = \{0,1\}^{96}$ is the nonce space as follows:

$\mathsf{Gen}(1^n)$:

1 : $k \leftarrow\$ \mathcal{K}$
2 : **return** k

$\mathsf{Enc}(k, m, n, d)$:

1 : $k_m = E(k, 0^{128})$
2 : $x \leftarrow (n||0^{31}1) \in \{0,1\}^{128}$
3 : $x' \leftarrow x + 1$
4 : $v := |m|$
5 : **for** $j = 0 \ldots v-1$ **do** :
6 : $\quad c[j] \leftarrow E(k, x') \oplus m[j]$
7 : $\quad x' \leftarrow x' + 1$
8 : dpad $:= -\mathrm{len}(d) \mod 128$
9 : cpad $:= -\mathrm{len}(c) \mod 128$
10 : $d' \leftarrow d||0^{\text{dpad}}$
11 : $c' \leftarrow c||0^{\text{cpad}}$
12 : input $\leftarrow (d'||c'||[\mathrm{len}(d)]_{64}||[\mathrm{len}(c)]_{64})$
13 : $h \leftarrow \mathrm{GHASH}(k_m, \text{input})$
14 : $t \leftarrow h \oplus E(k, x)$
15 : ct $:= (c, t)$
16 : **return** ct

$\mathsf{Dec}(k, \mathrm{ct}, n, d)$:

1 : **parse** ct $= (c, t)$
2 : $k_m = E(k, 0^{128})$
3 : dpad $:= -\mathrm{len}(d) \mod 128$
4 : cpad $:= -\mathrm{len}(c) \mod 128$
5 : $d' \leftarrow d||0^{\text{dpad}}$
6 : $c' \leftarrow c||0^{\text{cpad}}$
7 : input $\leftarrow (d'||c'||[\mathrm{len}(d)]_{64}||[\mathrm{len}(c)]_{64})$
8 : $h \leftarrow \mathrm{GHASH}(k_m, \text{input})$
9 : $x \leftarrow (n||0^{31}1) \in \{0,1\}^{128}$
10 : $x' \leftarrow x + 1$
11 : $t' \leftarrow h \oplus E(k, x)$
12 : **if** $t' \neq t$:
13 : $\quad$ **return** $:= \perp$
14 : **else** :
15 : $\quad v := |c|$
16 : $\quad$ **for** $j = 0 \ldots v-1$ **do** :
17 : $\quad\quad m[j] \leftarrow E(k, x') \oplus c[j]$
18 : $\quad\quad x' \leftarrow x' + 1$
19 : $\quad$ **return** m

To give a description of the different notation used in the scheme: the expression $A||B$ denotes the concatenation of two bit strings. 0^l for some l denotes the zero string of length l. $|\cdot|$ of some value outputs a non-negative integer which is the number of blocks in the argument. $\mathrm{len}(\cdot)$ of some value outputs a non-negative integer which is the total number of bits in the argument. The function $[\mathrm{len}(\cdot)]_{64}$ outputs a 64-bit string, which is a bit-representation of the number of bits in the argument. The values dpad and cpad are used for padding the input for GHASH. Both dpad and cpad denote the number of zeros that need to be added to d and c respectively, such that the number of bits in the string becomes a multiple of 128.

Correctness: We now show the correctness of the GCM scheme. Let $m^* \in \mathcal{M}$, $n \in \mathcal{N}$, $d \in \mathcal{D}$ and $k \in \mathcal{K}$. We compute the encryption and decryption using the chosen key, message, nonce and associated data as follows:

$\mathsf{Enc}(k, m^*, n, d)$:	$\mathsf{Dec}(k, \mathsf{ct}, n, d)$:
$k_m = E(k, 0^{128})$	**parse** $\mathsf{ct} = (c, t)$
$x \leftarrow (n \| 0^{31}1) \in \{0,1\}^{128}$	$k_m = E(k, 0^{128})$
$x' \leftarrow x + 1$	$\text{dpad} := -\text{len}(d) \mod 128$
$v := \|m^*\|$	$\text{cpad} := -\text{len}(c) \mod 128$
for $j = 0 \dots v-1$ **do** :	$d' \leftarrow d \| 0^{\text{dpad}}$
$\quad c[j] \leftarrow E(k, x') \oplus m^*[j]$	$c' \leftarrow c \| 0^{\text{cpad}}$
$\quad x' \leftarrow x' + 1$	$h \leftarrow \text{GHASH}(k_m, d' \| c' \| [\text{len}(d)]_{64} \| [\text{len}(c)]_{64})$
$\text{dpad} := -\text{len}(d) \mod 128$	$x \leftarrow (n \| 0^{31}1) \in \{0,1\}^{128}$
$\text{cpad} := -\text{len}(c) \mod 128$	$x' \leftarrow x + 1$
$d' \leftarrow d \| 0^{\text{dpad}}$	$t' \leftarrow h \oplus E(k, x)$
$c' \leftarrow c \| 0^{\text{cpad}}$	**if** $t' \neq t$:
$h \leftarrow \text{GHASH}(k_m, d' \| c' \| [\text{len}(d)]_{64} \| [\text{len}(c)]_{64})$	$\quad$ **return** $:= \perp$
$t \leftarrow h \oplus E(k, x)$	**else** :
$\text{ct} := (c, t)$	$\quad v := \|c\|$
return ct	$\quad$ **for** $j = 0 \dots v-1$ **do** :
	$\quad\quad m[j] \leftarrow E(k, x') \oplus c[j]$
	$\quad\quad m[j] \leftarrow E(k, x') \oplus (E(k, x') \oplus m^*[j])$
	$\quad\quad m[j] \leftarrow 0^{128} \oplus m^*[j]$
	$\quad\quad m[j] \leftarrow m^*[j]$
	$\quad\quad x' \leftarrow x' + 1$
	$\quad$ **return** m

Since we input the same key, nonce and associated data into both the encryption and decryption algorithm, all the values k_m, x, x', dpad, cpad, d', c', $[\text{len}(d)]_{64}$, $[\text{len}(c)]_{64}$ and $E(k, x)$ in both algorithms will all be equal. This implies that h also will be the same. We then have that $t = t'$, which means the decryption algorithm will not abort. From this point on, the computation of m is essentially the same as the one in the randomized counter mode. Since the term $E(k, x')$ cancels out, $m[j] = m^*[j]$ for each $j = 0, \dots, v-1$ and we have that $m = m^*$. By this, correctness holds.

Theorem 4.0.2 (Nonce-based AEAD security of GCM)**.** *The GCM is nonce-based AEAD-secure under the assumption that the underlying block cipher is a secure PRF, and that GHASH is an XOR-DUF.*

In particular, for every PPT adversary $\mathcal{A}$ playing the Q-query nCPA$_{ad}$ game against GCM, there exists a PPT adversary $\mathcal{B}$ that plays the PRF security game against F such that

$$\mathsf{Adv}^{ncpa_{ad}}_{GCM,\mathcal{A}}(n) = \mathsf{Adv}^{lr\text{-}prf}_{E,\mathcal{B}}(n).$$

For every PPT adversary $\mathcal{A}$ playing the Q-query nCI$_{ad}$ game against GCM, there exists a PPT adversary $\mathcal{B}$ that plays the PRF security game against F and a PPT adversary $\mathcal{C}$ that plays the XOR-DUF security game against GHASH such that

$$\mathsf{Adv}^{nci_{ad}}_{GCM,\mathcal{A}}(n) \leq \mathsf{Adv}^{lr\text{-}prf}_{E,\mathcal{B}}(n) + \mathsf{Adv}^{xor\text{-}duf}_{GHASH,\mathcal{C}}(n).$$

Proof. To show that GCM is nonce-based AEAD-secure, we must show that it is both nCPA_{ad} secure and nCI_{ad} secure. We structure this by proving each of these results as separate lemmas. Theorem 4.0.2 then follows.

Lemma 4.0.3. *For every PPT adversary $\mathcal{A}$ playing the Q-query nCPA_{ad} game against GCM, there exists a PPT adversary $\mathcal{B}$ that plays the PRF security game against F such that*

$$\mathsf{Adv}^{ncpa_{ad}}_{GCM,\mathcal{A}}(n) = \mathsf{Adv}^{lr\text{-}prf}_{E,\mathcal{B}}(n).$$

Proof. An important aspect of the proof is that adversary $\mathcal{A}$ do not have direct access to the underlying block cipher. That enables us to switch it out with a truly random function and then make use of the PRF assumption. Because of this, the main idea of this proof is the same as in the randomized counter mode. By switching out the underlying block cipher with a truly random function, we are able to make a reduction that bounds the nCPA_{ad} advantage. However, since the nonce is fixed to 96 bits and the number of message blocks is less than or equal to $2^{32}-2$, we will see that we are guaranteed unique input for each use of the underlying block cipher. We will look at this in more detail. We define the following games:

Game $G_0/G_1/G_2/G_3$:	`Oracle` $\texttt{Enc}(m_0, m_1, n, d)$: `//`Q `queries`
$k \leftarrow\$ \mathcal{K}$	**if** $n \in L_n$:
$b \leftarrow\$ \{0,1\}$	**return** $\perp$
$k_{\text{rand}} \leftarrow\$ \mathcal{X}$	$L_n \leftarrow L_n \cup \{n\}$
$f \leftarrow E(k,\cdot)$	$k_m = f(0^{128})$ $k_m \leftarrow k_{\text{rand}}$
$f \leftarrow\$ \text{Funs}[\mathcal{X},\mathcal{X}]$	$x \leftarrow (n \Vert 0^{31}1)$
$b' \leftarrow\$ \mathcal{A}^{\text{Enc}(\cdot,\cdot,\cdot,\cdot)}$	$x' \leftarrow x+1$
return $b = b'$	$v := \lvert m \rvert$
	for $j = 0 \ldots 2^{32}-3$ **do** :
	$y_j \leftarrow f(x')$ $y_j \leftarrow\$ \mathcal{X}$
	$x_j = x'$
	$x' \leftarrow x'+1$
	if $x' \in L_x$: bad $\leftarrow$ True;
	$y_j \leftarrow \Sigma[x_j]$
	$L_x \leftarrow L_x \cup \{x_j\}$
	$\Sigma[x_j] := y_j$
	for $j = 0 \ldots v-1$ **do** :
	$c[j] \leftarrow y_j \oplus m[j]$
	$\texttt{dpad} := -\texttt{len}(d) \mod 128$
	$\texttt{cpad} := -\texttt{len}(c) \mod 128$
	$d' \leftarrow d \Vert 0^{\texttt{dpad}}$
	$c' \leftarrow c \Vert 0^{\texttt{cpad}}$
	$h \leftarrow \texttt{GHASH}(k_m, d' \Vert c' \Vert [\texttt{len}(d)]_{64} \Vert [\texttt{len}(c)]_{64})$
	$y \leftarrow\$ \mathcal{X}$
	if $x \in L_x$: bad $\leftarrow$ True;
	$y \leftarrow \Sigma[x]$
	$L_x \leftarrow L_x \cup \{x\}$
	$\Sigma[x] := y$
	$t \leftarrow h \oplus f(x)$ $t \leftarrow h \oplus y$
	$\texttt{ct} := (c,t)$
	return ct

Each of the coloured statements are executed in their respective games and so on. That is, the red statement is executed in game 1 and so on, and similarly for the other colours. We assume that the underlying probability space is the same in all four games.

Game G_0 is the real nCPA$_{ad}$game, so we have that

$$\mathsf{Adv}^{\text{ncpa}_{\text{ad}}}_{\text{GCM},\mathcal{A}}(n) = |\Pr[G_0^{\mathcal{A}} \Rightarrow 1] - \frac{1}{2}|.$$

Game G_1 replaces the underlying block cipher $E(k,\cdot)$ with a truly random function. We will use this to construct an adversary $\mathcal{B}$ against the PRF security of the underlying block cipher E, simulating G_0 and G_1 for $\mathcal{A}$. That is, we will construct adversary $\mathcal{B}$ such that

$$\mathsf{Adv}^{\text{lr-prf}}_{E,\mathcal{B}}(n) = |\Pr[G_0^{\mathcal{A}} \Rightarrow 1] - \Pr[G_1^{\mathcal{A}} \Rightarrow 1]|.$$

$\mathcal{B}$ inputs the function f and has to distinguish if f is the real function, or if it is a random function $f \leftarrow\$ \text{Funs}[\mathcal{X},\mathcal{X}]$. We construct the reduction as follows:

Adversary $\mathcal{B}(f)$:	**Oracle** $\text{Enc}(m_0,m_1,n,d)$: //Q queries
1 : $b \leftarrow\$ \{0,1\}$	1 : **if** $n \in L_n$:
2 : $b' \leftarrow\$ \mathcal{A}^{\text{Enc}(\cdot,\cdot,\cdot,\cdot)}$	2 : **return** $\perp$
3 : **return** $b = b'$	3 : $L_n \leftarrow L_n \cup \{n\}$
	4 : $k_m = f(0^{128})$
	5 : $x \leftarrow (n \| 0^{31}1)$
	6 : $x' \leftarrow x + 1$
	7 : $v := \|m\|$
	8 : **for** $j = 0 \ldots v-1$ **do** :
	9 : $c[j] \leftarrow f(x') \oplus m[j]$
	10 : $x' \leftarrow x' + 1$
	11 : dpad $:= -\text{len}(d) \mod 128$
	12 : cpad $:= -\text{len}(c) \mod 128$
	13 : $d' \leftarrow d \| 0^{\text{dpad}}$
	14 : $c' \leftarrow c \| 0^{\text{cpad}}$
	15 : $h \leftarrow \text{GHASH}(k_m, d' \| c' \| [\text{len}(d)]_{64} \| [\text{len}(c)]_{64})$
	16 : $t \leftarrow h \oplus f(x)$
	17 : ct $:= (c,t)$
	18 : **return** ct

We see that if $b = 0$, $\mathcal{B}$ perfectly simulates G_0, and if $b = 1$, $\mathcal{B}$ perfectly simulates G_1. Therefore, we have that

$$\begin{aligned}\mathsf{Adv}^{\text{lr-prf}}_{E,\mathcal{B}}(n) &= |\Pr[\text{LR-PRF}_0^{\mathcal{B}} \Rightarrow 1] - \Pr[\text{LR-PRF}_1^{\mathcal{B}} \Rightarrow 1]| \\ &= |\Pr[G_0^{\mathcal{A}} \Rightarrow 1] - \Pr[G_1^{\mathcal{A}} \Rightarrow 1]|.\end{aligned}$$

We now look at G_2 and G_3. We begin with G_3, since it is connected to G_1. G_3 switches to a direct implementation of a truly random function f. Namely, it draws uniformly distributed elements directly

from $\mathcal{X}$, and assigns them to x_j-values by use of the array Σ, keeping track of them. It also generates k_m by drawing k_{rand} uniformly from $\mathcal{X}$, and assigning its value to k_m. Therefore we have

$$\Pr[\mathrm{G}_1^{\mathcal{A}} \Rightarrow 1] = \Pr[\mathrm{G}_3^{\mathcal{A}} \Rightarrow 1].$$

In G_2, we do not keep track of previous x-values. If the game is in some encryption query asked to use f on the same x-value twice, the game will with high probability give two different values assigned to x. G_2 also raises a secret flag bad, but the event bad is independent of G_2, since it does not influence anything in this game. This is essentially the same situation as in Game 2 and Game 3 in the security proof of the RCM. Since all the y_j-values are chosen independently and are uniformly distributed over $\mathcal{X}$, we are basically using independent one-time pads for encryption of each message block. Therefore the adversary $\mathcal{A}$ cannot do any better than guessing randomly which of the messages m_0 and m_1 that has been encrypted. Therefore we have that

$$\Pr[\mathrm{G}_2^{\mathcal{A}} \Rightarrow 1] = \frac{1}{2}.$$

G_2 and G_3 are identical-until-bad games, so we can use the difference lemma to bound their difference by

$$|\Pr[\mathrm{G}_2^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathrm{G}_3^{\mathcal{A}} \Rightarrow 1]| \leq \Pr[\mathsf{bad}].$$

We want to calculate Pr[bad], which is the probability that at least two of the counting values from two different encryption queries are the same. The counter consists of some unique nonce and 32 subsequent bits used for the counting mechanism. Therefore, for two counter values to be the same, we need to flip the last bit of the nonce. The counter value used for encryption starts at $n||0^{30}10$, so if this were to happen, we would have to increment the counter $2^{32} - 1$ times. However, since we have defined the message space to be $\mathcal{M} = \{0,1\}^{\leq 2^{32}-2}$, we are guaranteed that this will never happen. The maximal number of increments on the counter is the same as the number of message blocks. Therefore, we have that

$$\Pr[\mathsf{bad}] = 0 \implies \Pr[\mathrm{G}_3^{\mathcal{A}} \Rightarrow 1] = \Pr[\mathrm{G}_2^{\mathcal{A}} \Rightarrow 1] = \frac{1}{2}.$$

From an earlier argument, we know that $\Pr[\mathrm{G}_3^{\mathcal{A}} \Rightarrow 1] = \Pr[\mathrm{G}_1^{\mathcal{A}} \Rightarrow 1]$, and therefore $\Pr[\mathrm{G}_1^{\mathcal{A}} \Rightarrow 1] = \frac{1}{2}$. By connecting our games, it then follows that

$$\begin{aligned}
\mathsf{Adv}^{\mathrm{ncpa_{ad}}}_{\mathrm{GCM},\mathcal{A}}(n) &= |\Pr[\mathrm{G}_0^{\mathcal{A}} \Rightarrow 1] - \frac{1}{2}| \\
&= |\Pr[\mathrm{G}_0^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathrm{G}_2^{\mathcal{A}} \Rightarrow 1]| \\
&= |\Pr[\mathrm{G}_0^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathrm{G}_3^{\mathcal{A}} \Rightarrow 1]| \\
&= |\Pr[\mathrm{G}_0^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathrm{G}_1^{\mathcal{A}} \Rightarrow 1]| \\
&= \mathsf{Adv}^{\text{lr-prf}}_{E,\mathcal{B}}(n).
\end{aligned}$$

Since we assumed that the underlying block cipher E is a secure PRF, this shows that $\mathsf{Adv}^{\mathrm{ncpa_{ad}}}_{\mathrm{GCM},\mathcal{A}}(n)$ is negligible, meaning GCM is $\mathrm{nCPA_{ad}}$-secure. This concludes the first part of our proof. □

Lemma 4.0.4. *For every PPT adversary $\mathcal{A}$ playing the Q-query $\mathrm{nCI_{ad}}$ game against GCM, there exists a PPT adversary $\mathcal{B}$ that plays the PRF security game against F and a PPT adversary $\mathcal{C}$ that plays the XOR-DUF security game against GHASH such that*

$$\mathsf{Adv}^{nci_{ad}}_{GCM,\mathcal{A}}(n) \leq \mathsf{Adv}^{lr\text{-}prf}_{E,\mathcal{B}}(n) + \mathsf{Adv}^{xor\text{-}duf}_{GHASH,\mathcal{C}}(n).$$

Proof. In this proof as well, adversary $\mathcal{A}$ do not have direct access to the underlying block cipher. Therefore we can switch out the underlying block cipher with a truly random function. We use this to construct an adversary $\mathcal{B}$ against the PRF assumption. Then we proceed by making a direct implementation of a truly random function. In the last game, we pick the tag itself uniformly from $\{0,1\}^{128}$. That means the corresponding y-value used for the tag is instead implicitly defined, and also we do not need GHASH to compute the tag. We will use this to construct an adversary $\mathcal{C}$ against the XOR-DUF assumption. We define the following games:

Game $G_0/G_1/G_2/G_3$:

$k \leftarrow\$ \mathcal{K}$
$k_{\text{rand}} \leftarrow\$ \mathcal{X}$
$f \leftarrow E(k,\cdot)$
$f \leftarrow\$ \text{Funs}[\mathcal{X},\mathcal{X}]$
$(\text{ct}^*, n^*, d^*) \leftarrow\$ \mathcal{A}^{\text{Enc}(\cdot,\cdot,\cdot)}$
if $((\text{ct}^*, n^*, d^*) \notin \text{L}_{\text{CND}})$:
parse $\text{ct}^* = (c^*, t^*)$
  $k'_m := f(0^{128})$ $k'_m \leftarrow k_{\text{rand}}$
  $\text{dpad}' := -\text{len}(d^*) \mod 128$
  $\text{cpad}' := -\text{len}(c^*) \mod 128$
  $d^{*\prime} \leftarrow d^* || 0^{\text{dpad}'}$
  $c^{*\prime} \leftarrow c^* || 0^{\text{cpad}'}$
  $h' \leftarrow \text{GHASH}(k'_m, d^{*\prime} || c^{*\prime} || [\text{len}(d^*)]_{64} || [\text{len}(c^*)]_{64})$
  $x \leftarrow (n^* || 0^{31} 1)$
  $t' = h' \oplus f(x)$ $t' \leftarrow h' \oplus \Sigma[x]$ $t' \leftarrow \Sigma_t[x]$
  if $t' = t^*$:
    return 1
  else :
    return 0
return 0

Oracle $\text{Enc}(m,n,d)$: //Q queries

if $n \in L_n$:
  return $\perp$
$L_n \leftarrow L_n \cup \{n\}$
$k_m = f(0^{128})$ $k_m \leftarrow k_{\text{rand}}$
$x \leftarrow (n || 0^{31} 1)$
$x' \leftarrow x + 1$
$v := |m|$
for $j = 0 \ldots 2^{32} - 3$ **do** :
  $y_j \leftarrow f(x')$ $y_j \leftarrow\$ \mathcal{X}$
  $x_j = x'$
  if $x' \in L_x$:
    $y_j \leftarrow \Sigma[x_j]$
  $L_x \leftarrow L_x \cup \{x_j\}$
  $\Sigma[x_j] := y_j$
  $x' \leftarrow x' + 1$
for $j = 0 \ldots v - 1$ **do** :
  $c[j] \leftarrow y_j \oplus m[j]$
$\text{dpad} := -\text{len}(d) \mod 128$
$\text{cpad} := -\text{len}(c) \mod 128$
$d' \leftarrow d || 0^{\text{dpad}}$
$c' \leftarrow c || 0^{\text{cpad}}$
$h \leftarrow \text{GHASH}(k_m, d' || c' || [\text{len}(d)]_{64} || [\text{len}(c)]_{64})$
$y \leftarrow\$ \mathcal{X}$
if $x \in L_x$:
  $y \leftarrow \Sigma[x]$
$\Sigma[x] := y$
$t \leftarrow h \oplus f(x)$ $t \leftarrow h \oplus y$
$t \leftarrow\$ \{0,1\}^{128}$
if $x \in L_x$:
  $t \leftarrow \Sigma_t[x]$
$\Sigma_t[x] := t$
$L_x \leftarrow L_x \cup \{x\}$
$\text{ct} := (c,t)$
$\text{L}_{\text{CND}} \leftarrow \text{L}_{\text{CND}} \cup \{(\text{ct}, n, d)\}$
return ct

In this sequence of games, we write out the verification part in detail to show how it can be simulated in G_1, G_2 and G_3, when we do not make use of the secret key k.

Game G_0 is the real nCI_{ad}-game, so we have that

$$\mathsf{Adv}^{\text{nci}_{\text{ad}}}_{\text{GCM},\mathcal{A}}(n) = \Pr[G_0^{\mathcal{A}} \Rightarrow 1].$$

Game G_1 replace the underlying block cipher $E(k,\cdot)$ with a truly random function. We use G_0 and G_1 to construct an adversary $\mathcal{B}$ against the PRF security of the underlying block cipher E. Reduction $\mathcal{B}$ simulate G_0 and G_1 for adversary $\mathcal{A}$ and makes use of $\mathcal{A}$'s forgery in its bit guess for the PRF challenge. It inputs the function f, which is either the underlying block cipher or a truly random function $f \leftarrow\$ \text{Funs}[\mathcal{X},\mathcal{X}]$. We construct the reduction $\mathcal{B}$ as follows:

Adversary $\mathcal{B}(f)$:

$(\text{ct}^*, n^*, d^*) \leftarrow\$ \mathcal{A}^{\text{Enc}(\cdot,\cdot,\cdot)}$
parse $\text{ct}^* = (c^*, t^*)$
if $((\text{ct}^*, n^*, d^*) \notin \text{L}_{\text{CND}})$:
  $k'_m := f(0^{128})$
  $\text{dpad}' := -\text{len}(d^*) \mod 128$
  $\text{cpad}' := -\text{len}(c^*) \mod 128$
  $d^{*\prime} \leftarrow d^* || 0^{\text{dpad}'}$
  $c^{*\prime} \leftarrow c^* || 0^{\text{cpad}'}$
  $h' \leftarrow \text{GHASH}(k'_m, d^{*\prime} || c^{*\prime} || [\text{len}(d^*)]_{64} || [\text{len}(c^*)]_{64})$
  $x \leftarrow (n^* || 0^{31} 1)$
  $t' = h' \oplus f(x)$
  if $t' = t^*$:
    return 1
  else :
    return 0
return 0

Oracle $\text{Enc}(m,n,d)$: //Q queries

if $n \in L_n$:
  return $\perp$
$L_n \leftarrow L_n \cup \{n\}$
$k_m = f(0^{128})$
$x \leftarrow (n || 0^{31} 1)$
$x' \leftarrow x + 1$
$v := |m|$
for $j = 0 \ldots v-1$ **do** :
  $c[j] \leftarrow f(x') \oplus m[j]$
  $x' \leftarrow x' + 1$
$\text{dpad} := -\text{len}(d) \mod 128$
$\text{cpad} := -\text{len}(c) \mod 128$
$d' \leftarrow d || 0^{\text{dpad}}$
$c' \leftarrow c || 0^{\text{cpad}}$
$h \leftarrow \text{GHASH}(k_m, d' || c' || [\text{len}(d)]_{64} || [\text{len}(c)]_{64})$
$t \leftarrow h \oplus f(x)$
$\text{ct} := (c,t)$
$\text{L}_{\text{CND}} \leftarrow \text{L}_{\text{CND}} \cup \{(\text{ct}, n, d)\}$
return ct

The reduction $\mathcal{B}$ checks the validity of $\mathcal{A}$'s forgery by first making sure that the triple has not been previously queried, and then it runs the validity test by the decryption algorithm using f. If $\mathcal{A}$ makes a valid forgery, adversary $\mathcal{B}$ outputs 1, and if not it outputs 0. This gives us that if $f = E(k,\cdot)$, $\mathcal{B}$ perfectly simulates G_0, and if $f \leftarrow\$ \text{Funs}[\mathcal{X},\mathcal{X}]$, $\mathcal{B}$ perfectly simulates G_1. Therefore, we have that

$$\begin{aligned}\mathsf{Adv}^{\text{lr-prf}}_{E,\mathcal{B}}(n) &= |\Pr[\text{LR-PRF}_0^{\mathcal{B}} \Rightarrow 1] - \Pr[\text{LR-PRF}_1^{\mathcal{B}} \Rightarrow 1]| \\ &= |\Pr[G_0^{\mathcal{A}} \Rightarrow 1] - \Pr[G_1^{\mathcal{A}} \Rightarrow 1]|.\end{aligned}$$

We now discuss G_2 and G_3. G_2 in the nCI_{ad} proof is essentially the same as G_3 in the $nCPA_{ad}$ proof. It switches to a direct implementation of a truly random function f. Namely, it draws uniformly distributed elements directly from $\mathcal{X}$, and assigns them to x_j-values and the x-value by use of the array Σ, keeping track of them. It also generates k_m by drawing k_{rand} uniformly from $\mathcal{X}$, and assigning the value to k_m. Therefore we have

$$\Pr[G_1^{\mathcal{A}} \Rightarrow 1] = \Pr[G_2^{\mathcal{A}} \Rightarrow 1].$$

The only difference between G_2 and G_3, is that in G_3 we draw the tag t uniformly from $\{0,1\}^{128}$, instead of drawing the y-value used for the tag uniformly. That means we implicitly define the y-value, since it must satisfy $t = h \oplus y$. Because adversary $\mathcal{A}$ has no direct access to the y_j-values or the y-value used to simulate the random function, G_2 and G_3 are indistinguishable for $\mathcal{A}$. Therefore

$$\Pr[G_2^{\mathcal{A}} \Rightarrow 1] = \Pr[G_3^{\mathcal{A}} \Rightarrow 1].$$

Now we can make a reduction on G_3, since we do not need GHASH to simulate encryption queries for $\mathcal{A}$. We construct adversary $\mathcal{C}$ against the XOR-DUF assumption, simulating G_3 for adversary $\mathcal{A}$ as follows:

Adversary $\mathcal{C}$:

$k_{\text{rand}} \leftarrow\$\, \mathcal{X}$
$(\text{ct}^*, n^*, d^*) \leftarrow\$\, \mathcal{A}^{\text{Enc}(\cdot,\cdot,\cdot)}$
if $((\text{ct}^*, n^*, d^*) \notin \text{L}_{\text{CND}})$:
  parse $\text{ct}^* = (c^*, t^*)$
  $k'_m \leftarrow k_{\text{rand}}$
  $\text{dpad}' := -\text{len}(d^*) \mod 128$
  $\text{cpad}' := -\text{len}(c^*) \mod 128$
  $d^{*\prime} \leftarrow d^* || 0^{\text{dpad}'}$
  $c^{*\prime} \leftarrow c^* || 0^{\text{cpad}'}$
  $h' \leftarrow \text{GHASH}(k'_m, d^{*\prime} || c^{*\prime} || [\text{len}(d^*)]_{64} || [\text{len}(c^*)]_{64})$
  $x \leftarrow (n^* || 0^{31} 1)$
  $t' \leftarrow \Sigma_t[x]$
  if $t' = t^*$:
    Find $(\text{ct}_k, n_k, d_k) \in \text{L}_{\text{CND}}$ such that $n^* = n_k$
    parse $\text{ct}_k = (c_k, t_k)$
    $\delta := t^* \oplus t_k$
    $p_0 := (d^{*\prime} || c^{*\prime} || [\text{len}(d^*)]_{64} || [\text{len}(c^*)]_{64})$
    $p_1 := (d'_k || c'_k || [\text{len}(d_k)]_{64} || [\text{len}(c_k)]_{64})$
    return (p_0, p_1, δ)
  else :
    return $\perp$
return $\perp$

Oracle $\text{Enc}(m, n, d)$: //Q queries

if $n \in L_n$:
  return $\perp$
$L_n \leftarrow L_n \cup \{n\}$
$k_m \leftarrow k_{\text{rand}}$
$x \leftarrow (n || 0^{31} 1)$
$x' \leftarrow x + 1$
$v := |m|$
for $j = 0 \ldots 2^{32} - 3$ **do** :
  $y_j \leftarrow\$\, \mathcal{X}$
  $x_j = x'$
  $x' \leftarrow x' + 1$
  if $x' \in L_x$:
    $y_j \leftarrow \Sigma[x_j]$
  $L_x \leftarrow L_x\{x_j\}$
  $\Sigma[x_j] := y_j$
for $j = 0 \ldots v - 1$ **do** :
  $c[j] \leftarrow y_j \oplus m[j]$
$\text{dpad} := -\text{len}(d) \mod 128$
$\text{cpad} := -\text{len}(c) \mod 128$
$d' \leftarrow d || 0^{\text{dpad}}$
$c' \leftarrow c || 0^{\text{cpad}}$
$t \leftarrow\$\, \{0,1\}^{128}$
if $x \in L_x$:
  $t \leftarrow \Sigma_t[x]$
$\Sigma_t[x] := t$
$L_x \leftarrow L_x \cup \{x\}$
$\text{ct} := (c, t)$
$\text{L}_{\text{CND}} \leftarrow \text{L}_{\text{CND}} \cup \{(\text{ct}, n, d)\}$
return ct

For the forgery (ct^*, n^*, d^*) there are two cases: either $n \in L_n$ or $n \notin L_n$, meaning either adversary $\mathcal{A}$ has asked an encryption query on the nonce used in the forgery or not. In the case of $n \notin L_n$, all the y_j-values and the t-value used for encryption are not yet set for a nonce n specified by $\mathcal{C}$. Since the valid tag t is never set, $\mathcal{A}$ can never guess the correct tag. Therefore the probability of $\mathcal{A}$ outputting a valid tag is 0, so (ct^*, n^*, d^*) is never a valid forgery in this case.

In the case of $n \in L_n$, all the y_j-values are already decided. Reduction $\mathcal{C}$ has access to previous (ct, n, d) triples, so it can retrieve the triple such that $n = n^*$. We denote this triple as $((c_k, t_k), n_k, d_k)$, where $n_k = n^*$. For a valid forgery, we must have that $c^* = c_k$, since each nonce is only used once. Because $n^* = n_k$, all counting values are the same, meaning that $y_j^* = y_{jk}$ for all $j \in \{0, \ldots, 2^{32} - 3\}$, and $y^* = y_k$ implicitly. Since we assume that (ct^*, n^*, d^*) is a valid forgery, we have that $d^* \neq d_k$ and $t^* \neq t_k$. If not, then $(\text{ct}^*, n^*, d^*) \in \text{L}_{\text{CND}}$. Therefore since $t^* = y^* \oplus h^*$ and $t_k = y_k \oplus h_k$, and we know that $t^* \neq t_k$ and $y^* = y_k$ implicitly, we must have that $h^* \neq h_k$. If not, then t^* and t_k would be equal.

Since $y^* = y_k$ implicitly, reduction $\mathcal{C}$ can retrieve $h^* \oplus h_k$ by the following computation:

$$\begin{aligned} t^* \oplus t_k &= (h^* \oplus y^*) \oplus (h_k \oplus y_k) \\ &= h^* \oplus h_k \oplus (y^* \oplus y_k) \\ &= h^* \oplus h_k \oplus (0^{128}) \\ &= h^* \oplus h_k. \end{aligned}$$

Now, if we let $\delta = t^* \oplus t_k = h^* \oplus h_k$, we break XOR-DUF if we are able to compute the correct input for GHASH corresponding to h^* and h_k. Reduction $\mathcal{C}$ has access to all the values d^*, c^*, d_k and c_k, so it can compute $p_0 = (d^{*\prime}||c^{*\prime}||\text{len}(d^*)||\text{len}(c^*))$ and $p_1 = (d_k'||c_k'||\text{len}(d_k)||\text{len}(c_k))$ such that $h^* = \text{GHASH}(p_0)$ and $h_k = \text{GHASH}(p_1)$. Since $d^* \neq d_k$, we know that $p_0 \neq p_1$. Therefore (p_0, p_1, δ) is a valid forgery against the XOR-DUF assumption.

We then have that

$$\Pr[\text{G}_3^{\mathcal{A}} \Rightarrow 1] = \Pr[\text{XOR-DUF}^{\mathcal{C}} \Rightarrow 1] = \mathsf{Adv}^{\text{xor-duf}}_{\text{GHASH},\mathcal{C}}(n).$$

Connecting the games above, it follows that

$$\begin{aligned} \mathsf{Adv}^{\text{nci}_{\text{ad}}}_{\text{GCM},\mathcal{A}}(n) &= \Pr[\text{G}_0^{\mathcal{A}} \Rightarrow 1] \\ &\leq |\Pr[\text{G}_0^{\mathcal{A}} \Rightarrow 1 - \text{G}_1^{\mathcal{A}} \Rightarrow 1]| + \Pr[\text{G}_1^{\mathcal{A}} \Rightarrow 1] \\ &= |\Pr[\text{G}_0^{\mathcal{A}} \Rightarrow 1 - \text{G}_1^{\mathcal{A}} \Rightarrow 1]| + \Pr[\text{G}_2^{\mathcal{A}} \Rightarrow 1] \\ &= |\Pr[\text{G}_0^{\mathcal{A}} \Rightarrow 1 - \text{G}_1^{\mathcal{A}} \Rightarrow 1]| + \Pr[\text{G}_3^{\mathcal{A}} \Rightarrow 1] \\ &= \mathsf{Adv}^{\text{lr-prf}}_{E,\mathcal{B}}(n) + \mathsf{Adv}^{\text{xor-duf}}_{\text{GHASH},\mathcal{C}}(n). \end{aligned}$$

Since we assumed that the underlying block cipher is secure as a PRF and that GHASH is an XOR-DUF, it follows that $\mathsf{Adv}^{\text{nci}_{\text{ad}}}_{\text{GCM},\mathcal{A}}(n)$ is negligible. Therefore GCM is nCI_{ad} secure. □

By lemma 4.0.3 and 4.0.4, we have that GCM is both nCPA_{ad} and nCI_{ad} secure, thus nonce-based AEAD-secure. This concludes our proof. □

Bibliography

[1] M. Bellare, P. Rogaway and D. Wagner, *Eax: A conventional authenticated-encryption mode*, Cryptology ePrint Archive, Report 2003/069, https://ia.cr/2003/069, 2003.

[2] D. Boneh and V. Shoup, *A Graduate Course in Applied Cryptograhy*, 0.5. 2020, pp. 27–29, 132–136, 186–190, 353, 364–367, 376–378.

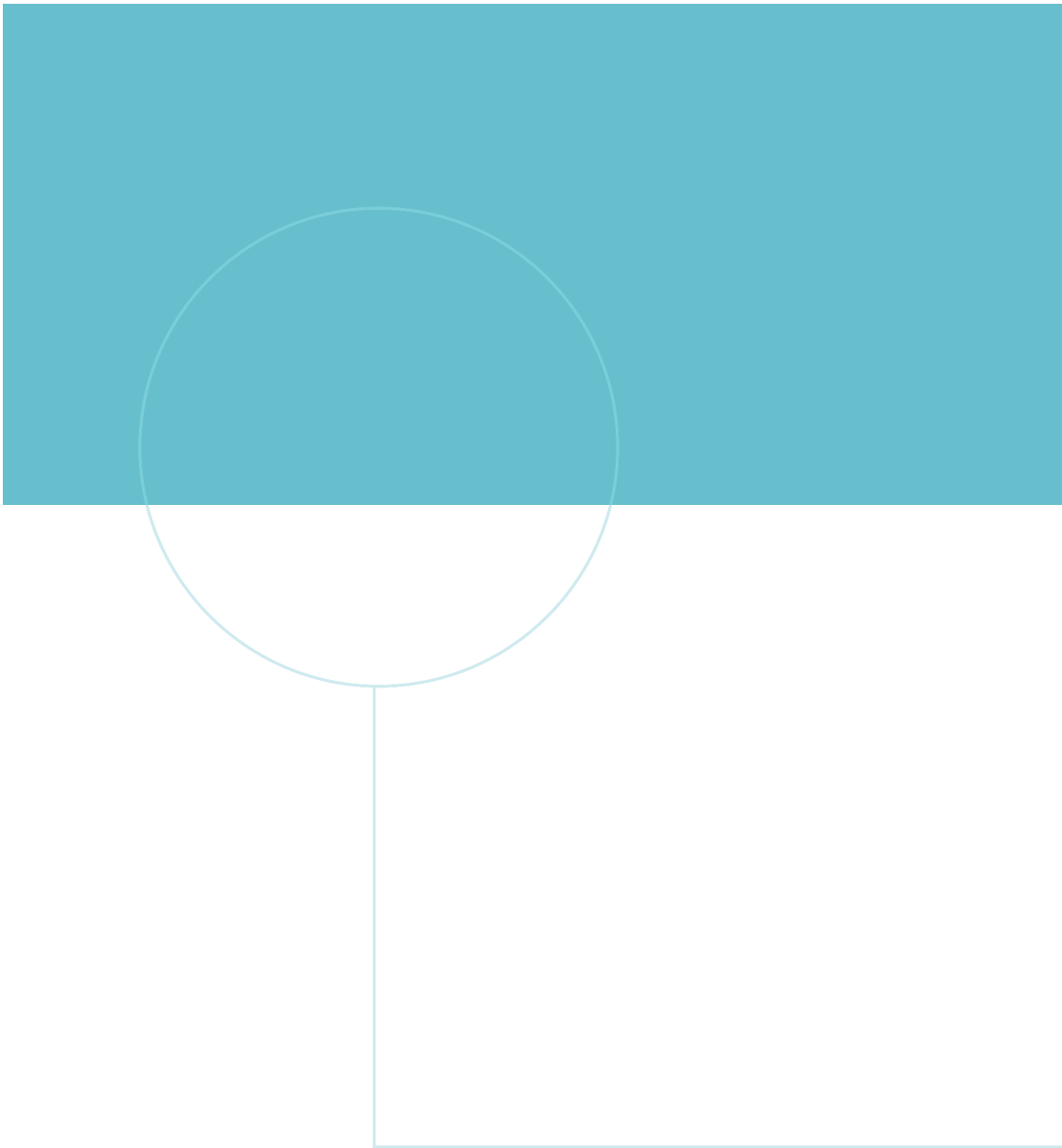
[3] D. A. McGrew and J. Viega, 'The security and performance of the galois/counter mode (gcm) of operation,' in *Progress in Cryptology - INDOCRYPT 2004*, A. Canteaut and K. Viswanathan, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 343–355, ISBN: 978-3-540-30556-9. DOI: 10.1007/978-3-540-30556-9_27.

[4] M. Dworkin, *Recommendation for block cipher modes of operation: Galois/counter mode (gcm) and gmac*, Nov. 2007. [Online]. Available: https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=51288.

[5] M. Bellare and P. Rogaway, *Code-based game-playing proofs and the security of triple encryption*, mihir@cs.ucsd.edu 13498 received 30 Nov 2004, last revised 16 Dec 2006, 2004. [Online]. Available: http://eprint.iacr.org/2004/331.

[6] V. Shoup, *Sequences of games: A tool for taming complexity in security proofs*, Cryptology ePrint Archive, Report 2004/332, https://ia.cr/2004/332, 2004.

[7] P. Rogaway, 'Authenticated-encryption with associated-data,' in *ACM CCS 2002: 9th Conference on Computer and Communications Security*, V. Atluri, Ed., Washington, DC, USA: ACM Press, 2002, pp. 98–107. DOI: 10.1145/586110.586125.



NTNU

Norwegian University of
Science and Technology