

BACHELOR THESIS



RADBOD UNIVERSITY

TBD

Author:
Stijn Vandenput

Supervisors:
Martijn Stam
Bart Mennink

April 2, 2024

Abstract

Contents

1	Introduction	1
2	Preliminaries	1
2.1	General Notation	2
2.2	Authenticated Encryption	2
2.3	Message Authentication	2
2.4	Nonces and Locks	2
2.5	Security Notions	3
2.6	Security Proofs of Generic Composition	4
3	NRS and GKP in Detail	4
3.1	NRS	4
3.1.1	Primitives	5
3.1.2	Composition	6
3.2	GKP	6
3.2.1	Primitives	7
3.2.2	Composition	8
3.3	Comparison of GKP and NRS	9
4	Lock-based Authenticated Encryption	9
4.1	IAE	10
4.2	IAE Security Model	10
5	Composition	11
5.1	Used Primitives	11
5.2	Composition	12
5.3	Security Bounds	12
6	Use Cases	14
6.1	PKE Schemes	15
7	Related Work	15
8	Open Problems	15
9	Conclusion	15
	References	16
	Appendix A	16
10	N1	16
11	N2	21
12	N3	26
	Appendix B	32

1 Introduction

To implement public-key encryption, a hybrid paradigm is typically followed: To encrypt a message, a ephemeral key is generated using a randomized key encapsulation mechanism (KEM). This key is then used to encrypt the message using a deterministic data encapsulation mechanism (DEM). Both the KEM and DEM output their own ciphertexts, which are concatenated to form the public-key encryption ciphertext. Classical analysis of DEMs uses a single user. Because of this, the good security bounds of these analysis do not always transfer to the real worlds where you can have millions of users. To make sure the security bounds are also good if you have many users, you can use larger ephemeral keys. Although using larger keys for DEMs is generally safer, expanding the size of the key is not always a viable option. This might be due to limitations in computing power or memory, or due to security primitives having fixed key sizes. Gligorijević, Kiltz and Poettering, henceforth GKP, propose augmentation using locks (originally called tags, but renamed to locks here to avoid overloaded terms) as an alternative solution to key expansion in a multi user setting [1]. Augmentation with locks works by giving the security primitive an additional input field, called the lock, to distinguish multiple users using the same key. The augmentation can improve security in a multi-user setting, without the need to expand the key.

After defining this augmentation, GKP apply the augmentation to a DEM and a MAC function, to create an augmented DEM (ADEM) and an augmented MAC (AMAC). These two are combined to construct an authenticated encryption primitive following the generic encrypt-then-MAC composition from Bellare and Nampreppe [2]. This composition is proven secure whenever the underlying ADEM and AMAC are secure. The generic composition of authenticated encryption has been revised in a paper by Nampreppe, Rogaway and Shrimpton [3], which we henceforth call NRS. In this revision, the generic composition using a MAC function and a deterministic encryption primitive is more thoroughly investigated. They shine light on the fact that the kind of encryption primitive used has a impact on which kinds of generic compositions are secure. When using a nonce-based authenticated encryption primitive and a MAC function, the encrypt-and-MAC, the encrypt-then-MAC and the MAC-then-encrypt are proven to be secure whenever the underlying primitives are secure.

Although the ADEM+AMAC composition given by GKP is secure, it is not clearly defined as authenticated encryption primitive. As a result, its security is evaluated as a DEM, not as an authenticated encryption primitive. Additionally, only the encrypt-then-MAC composition is considered by GKP and it is not shown if other composition methods could be secure. In this thesis, generic composition of authenticated encryption using locks is more thoroughly investigated. We formally compare the constructions and the notation from NRS and GKP (Chapter 3). To evaluate generic composition using locks, we define a new cryptographic primitive which we analyze in a multi-user setting (Chapter 4). Using the knowledge from NRS, three generic compositions (encrypt-and-MAC, encrypt-then-MAC and MAC-then-encrypt) of this primitive are considered, all using a lock-based encryption primitive and a lock-based MAC function (Chapter 5). These compositions are then proven to be secure, whenever the underlying primitives are secure (Appendix A).

2 Preliminaries

In this section we will explain several concepts important to the rest of our work, as well as some general notation.

2.1 General Notation

Strings are binary and bit-wise, the set of all strings is $\{0,1\}^*$. The length of x is written as $|x|$, the concatenation of x and y as $x \parallel y$, a being the result of b as $a \leftarrow b$, and taking a uniform random sampling from set z and assigning it to x as $x \xleftarrow{\$} z$. We write N for the number of users and allow a single type of error message written as \perp . Any tuple containing \perp will be \perp as well. We define the following spaces, all of them being subsets of the set of all strings: nonempty key space \mathcal{K} , lock space \mathcal{L} , nonce space \mathcal{N} , message space \mathcal{M} , ciphertext space \mathcal{C} , tag space \mathcal{T} , and associated data space \mathcal{A} . Unless stated otherwise, \mathcal{M} contains at least two strings, and if \mathcal{M} or \mathcal{A} contains a string of length x , it must contain all strings of length x . There are no further constraints on these spaces.

2.2 Authenticated Encryption

Two different security requirements are data privacy, the insurance that data cannot be viewed by an unauthorized party, and data integrity, the insurance that data has not been modified by an unauthorized party. Authenticated encryption combines both of these security requirements into one and ensures both data privacy and integrity. A basic authenticated encryption scheme consists of an encryption call and a decryption call. The encryption call takes a message and a key to a self-authenticating ciphertext. The decryption call takes a self-authenticating ciphertext and a key to a message. Some authenticated encryption schemes allow an additional input AD, short for associated data. The associated data is specifically required to not have data privacy but does require data integrity. Authenticated encryption schemes that support AD are called AEAD schemes.

2.3 Message Authentication

Message authentication can be done using a message authentication code, MAC for short. A basic MAC function takes a message and key and outputs a tag which authenticates the message. Some MAC function also have a verification call that takes a message, key and tag and outputs either *true* or *false*. A mac function can have different security requirements, it is said to be PRF secure when it is infeasible to distinguish the tag it outputs from the result of a pseudo-random function from all message-key pairs to all tags. A MAC is said to be unforgeable when it is infeasible to create a valid message-tag pair without knowledge of the secret key. A PRF secure MAC is also unforgeable, given the tag space is big enough, while a unforgeable MAC is not necessarily PRF secure.

2.4 Nonces and Locks

A basic deterministic encryption scheme takes a message and key as input, and outputs a ciphertext. Using this encryption scheme, a message encrypted under the same key leads to the same ciphertext. Both GPK and NRS resolve this by giving the encryption scheme an additional argument. GPK uses locks while NRS uses nonces. Although nonces and locks look similar, their purpose and exact working differ leading to different use cases. Most notably, nonces are useful when one user is allowed to encrypt multiple messages and locks are only useful when there are multiple users.

Nonces Using a basic deterministic encryption scheme, a message encrypted twice by the same user results in the same ciphertext. This can leak information about the message, and can be

prevented using a nonce. A nonce is a number that is assumed to only be used once per user to encrypt a message. Whenever a message is encrypted twice with the same key, but with two different nonces, the resulting ciphertexts should be indistinguishable from two ciphertexts corresponding to two different messages. As a result, it is infeasible for an adversary to guess if a message has been sent multiple times. The adversary is usually allowed to let a user decrypt multiple messages with one nonce. Nonces are only used when a user uses its key multiple times, as otherwise a message will never be encrypted by the same user twice.

Locks Using a basic deterministic encryption scheme, a message encrypted by two users that have the same key results in the same ciphertext. This can leak information about the secret keys used, and can be prevented using locks. Whereas nonces are bound to the message, locks are bound to the user. Each user has one lock, provided the users have one key each, and will encrypt all their messages using that lock. Whenever a message is encrypted twice with the same key, but with two different locks, the resulting ciphertexts should be indistinguishable from two ciphertexts corresponding to two different messages. As a result, it is infeasible for an adversary to see when two users have a key collision unless locks collide as well. To prevent collisions in locks, we assume locks to be globally unique. The adversary is usually only allowed to let a user decrypt messages with the correct lock. Locks are only used in a multi-user setting, as key collision is impossible when there is one user.

2.5 Security Notions

The security of a cryptographic construction can be modeled as a distinguishing advantage. There are different things we can distinguish on, leading to different security notions. To understand NRS and GKP it is important to see which security notions they use, all the relevant notions are written below.

IND-CPA vs IND-CCA You can model against an passive or an active attacker. A passive attacker can only read the messages while an active attacker can also alter the messages. A passive attacker can be modelled using a chosen plaintext attack, CPA for short. In this model, the adversary can choose the plaintext that is encrypted, but not the ciphertext that is decrypted. An active attacker can be modelled using a chosen ciphertext attack, CCA for short. In this model, the adversary can choose the plaintext that is encrypted, as well as the ciphertext that is decrypted. Shorthand notations for the two is IND-CPA and IND-CCA, respectively. IND-CCA implies IND-CPA, but not the other way around. Both GKP and NRS model the authenticated encryption primitive using IND-CPA and the underlying encryption primitive using IND-CCA.

IND-\$ vs IND-LOR Left-or-right-indistinguishability refers to a situation where the adversary gives two messages, and is given a ciphertext. The adversary has to guess which of the two messages corresponds to the ciphertext. \$-indistinguishability refers to a situation where the adversary is given access to either the real construction, or to a lazily sampled random function \$. This random function returns a random string with the same length as the ciphertext would have. The adversary has to guess which of these two it has access to. As long as the length of the ciphertext only depends on the length of the message, not the content, IND-\$ implies IND-LOR, but not the other way around(**todo: add citation**). IND-\$ is used by GPK and IND-LOR is used by NRS

Both of these are separate dimensions and they can be combined into 4 different notions. For example IND-\$-CCA refers to a situation where the adversary has to distinguish between the

real construction, or a random function while being able to choose both the plaintext that is encrypted and the ciphertext that is decrypted.

Game Based Security Notions These security notions can be written in a game-based format, using pseudocode instead of text. As an example, the IND-\$-CPA game of a nonce-based encryption scheme can be found in Figure 1. A challenge bit b is given to the game, in this case b signals whether we are in the real or the ideal world. The adversary guesses this bit and returns b' , signaling its guess for b . In addition, the adversary can have access to oracles. In our example there is only one oracle that takes a nonce and a message. Using game based notation, you can clearly write out all the limitations. For example, the limitation that nonces cannot be reused is modeled by lines 0, 5 and 6. Lines 8 and 9 model how the random function $\$$ behaves. These limitations could be written out in text based format as well but when there are multiple limitations, writing it out in a game based format can make both the security notion, as well as security proofs, more comprehensible and precise.

2.6 Security Proofs of Generic Composition

To prove the security of a generic composition we use a security reduction. To define the reduction we bind the advantage of the generic composition by terms of the advantages of the underlying primitives. To do this we show that an advantage on the generic composition can be leveraged to gain an advantage on the underlying primitives. In other words, we prove that if we can break the security of the generic composition, then we can break the security of one of the underlying primitives. After proving this, we can conclude the composition is secure as long as the underlying primitives are secure. A reduction is said to be tight when its security does not depend on the attack and lose when the security does depend on the adversary's attack.

3 NRS and GKP in Detail

In this section we explain the parts from GKP and NRS important to our work. Afterwards, a comparison is made between the two. Some notations will be different from the original papers for improved consistency. What are called tags by GKP, we will call locks instead to avoid confusion with the output of MAC functions and we call the output of the AMAC the tag instead of the ciphertext. The security notions from NRS are converted to a game-based format using insights from **(todo add citation for Automated Analysis of Protocols that use Authenticated Encryption: How Subtle AEAD)** in order to better match the notation from GKP and be more adaptable to a multi-user setting. The security games are only explained briefly in this section, a more in-depth explanation of the relevant constructs can be found in section 4.2.

3.1 NRS

Three generic ways to compose an authenticated encryption scheme are discussed in a paper written by Bellare and Namprempre [2]: encrypt-then-MAC, encrypt-and-MAC and MAC-then-encrypt. In this paper, encrypt-then-MAC is considered the only secure composition when using probabilistic encryption as a building block. NRS notes that the type of encryption scheme used influences which one of these compositions are secure. The constructions are generalized to using nonce-based encryption, nA for short, and a PRF secure MAC function as a building blocks to create nonce-based authenticated encryption schemes, nAEs for short. Using these blocks, all three constructions are proven secure.

Game nE-IND-\$-CPA _A ^b	Oracle Oenc(n, m)
0 : $U \leftarrow \emptyset$	5 : if $n \in U$: return \perp
1 : $k \xleftarrow{\$} \mathcal{K}$	6 : $U \leftarrow U \cup \{n\}$
2 : $b' \leftarrow A$	7 : $c \leftarrow E(k, n, m)$
3 : return b'	8 : if $b = 1 \wedge c \neq \perp$:
	9 : $c \xleftarrow{\$} \{0, 1\}^{ c }$
	10 : return c

Figure 1: nE-IND-\$-CPA game, A has access to oracle Oenc.

3.1.1 Primitives

nE A nonce-based encryption scheme is defined by a triple $\Pi = (\mathcal{K}, E, D)$. Deterministic encryption algorithm E takes three inputs (k, n, m) and outputs a value c , the length of c only depends the length of k , n and m . If, and only if, (k, n, m) is not in $\mathcal{K} \times \mathcal{N} \times \mathcal{M}$, c will be \perp . Decryption algorithm D takes three inputs (k, n, c) and outputs a value m . Both E and D are required to satisfy correctness (if $E(k, n, m) = c \neq \perp$, then $D(k, n, c) = m$) and tidiness (if $D(k, n, c) = m \neq \perp$, then $E(k, n, m) = c$).

nE security The security of a nE is defined as

$$\text{Adv}_{\Pi, A}^{\text{nE}} = \Pr[\text{nE-IND-}\$-\text{CPA}_A^0 = 0] - \Pr[\text{nE-IND-}\$-\text{CPA}_A^1 = 0]$$

where nE-IND-\$-CPA is in Figure 1. Set U keeps track of all used nonces as the adversary is not allowed to repeat nonces.

MAC A MAC is defined by an algorithm F that takes a key k in \mathcal{K} and a string m and outputs either a n -bit tag t or \perp . The domain of F is the set X of all m such that $F(k, m) \neq \perp$ is in X , this domain may not depend on k .

MAC security NRS require the MAC to be PRF secure. The security is defined as

$$\text{Adv}_{F, A}^{\text{MAC}} = \Pr[\text{MAC-PRF}_A^0 = 0] - \Pr[\text{MAC-PRF}_A^1 = 0]$$

where MAC-PRF is in Figure 2. In this game the set U keeps track of the used messages to prevent trivial distinctions.

nAE A nonce-based authenticated encryption scheme is defined by a triple $\Pi = (\mathcal{K}, E, D)$. Deterministic encryption algorithm E takes four inputs (k, n, a, m) and outputs a value c , the length of c only depends the length of k , n , a and m . If, and only if, (k, n, a, m) is not in $\mathcal{K} \times \mathcal{N} \times \mathcal{A} \times \mathcal{M}$, c will be \perp . Decryption algorithm D takes four inputs (k, n, a, c) and outputs a value m . both E and D are required to satisfy correctness (if $E(k, n, a, m) = c \neq \perp$, then $D(k, n, a, c) = m$) and tidiness (if $D(k, n, a, c) = m \neq \perp$, then $E(k, n, a, m) = c$).

nAE security The security of a nAE is defined as

$$\text{Adv}_{\Pi, A}^{\text{nAE}} = \Pr[\text{nAE-IND-}\$-\text{CCA}_A^0 = 0] - \Pr[\text{nAE-IND-}\$-\text{CCA}_A^1 = 0]$$

Game MAC-PRF _A ^b	Oracle Omac(<i>m</i>)
0: $U \leftarrow \emptyset$	4: if $m \in U$: return \perp
1: $k \xleftarrow{\$} \mathcal{K}$	5: $U \leftarrow U \cup \{m\}$
2: $b' \leftarrow A$	6: $t \leftarrow F(k, m)$
3: return b'	7: if $b = 1 \wedge t \neq \perp$:
	8: $t \xleftarrow{\$} \{0, 1\}^{ t }$
	9: return t

Figure 2: MAC-PRF, A has access to oracle Omac and U is the set of used messages.

Game nAE-IND- \mathcal{S} -CCA _A ^b	Oracle Oenc(n, a, m)	Oracle Odec(n, a, c)
0: $U \leftarrow \emptyset$	6: if $n \in U$: return \perp	14: if $b = 1$: return \perp
1: $Q \leftarrow \emptyset$	7: $U \leftarrow U \cup \{n\}$	15: if $(n, a, _, c) \in Q$: return \perp
2: $k \xleftarrow{\$} \mathcal{K}$	8: if $(n, a, m, _) \in Q$: return \perp	16: $m \leftarrow D(k, n, a, c)$
3: $b' \leftarrow A$	9: $c \leftarrow E(k, n, a, m)$	17: $Q \leftarrow Q \cup \{(n, a, m, c)\}$
4: return b'	10: if $b = 1 \wedge c \neq \perp$:	18: return m
	11: $c \xleftarrow{\$} \{0, 1\}^{ c }$	
	12: $Q \leftarrow Q \cup \{(n, a, m, c)\}$	
	13: return c	

Figure 3: nAE-IND- \mathcal{S} -CCA game, A has access to oracles Oenc and Odec.

where nAE-IND- \mathcal{S} -CCA is in Figure 3. The adversary is not allowed to repeat nonces on encryption, set U keeps track of all used nonces. Following the translation of IND- \mathcal{S} -CCA to a security game for AE from (**todo add citation for Automated Analysis of Protocols that use Authenticated Encryption: How Subtle AEAD**), $_$ denotes a variable that is irrelevant and set Q keeps track of all query results in order to prevent trivial distinctions.

3.1.2 Composition

NRS define several different schemes that compose an nAE from an nE and a MAC, they define the composition to be secure if there is a tight reduction from breaking the nAE-security of the scheme to breaking the nE-security and the PRF security of the underlying primitives. Three different schemes, named N1, N2 and N3 were proven to be secure they can be viewed in Figure 6 of NRS. Noteworthy is that these relate to encrypt-and-MAC, encrypt-then-MAC and MAC-then-encrypt, respectively.

3.2 GKP

GKP discusses the concept of augmentation using locks. They start by showing how some data encapsulation mechanisms are vulnerable to a passive multi-instance distinguishing- and key recovery and how this can lead to problems when used in public key encryption. They define the augmented data encapsulation mechanisms, ADEM for short, that uses locks to negate these insecurities. Additionally, they show how an ADEM that is secure against passive attacks can

Game L-IND-LOR-CPA $_{A,N}^b$	Oracle Oenc(j, l, m_0, m_1)
0 : $L \leftarrow \emptyset$	6 : if $C_j \neq \emptyset$: return \perp
1 : for $j \in [1..N]$:	7 : if $l \in L$: return \perp
2 : $k_j \xleftarrow{\$} \mathcal{K}$	8 : $L \leftarrow L \cup \{l\}$
3 : $C_j \leftarrow \emptyset$	9 : $l_j \leftarrow l$
4 : $b' \leftarrow A$	10 : $c \leftarrow A.\text{enc}(k_j, l_j, m_b)$
5 : return b'	11 : $C_j \leftarrow C_j \cup \{c\}$
	12 : return c

Figure 4: L-IND-LOR-CPA game, A has access to oracle Oenc.

be combined with a MAC that is augmented in a similar fashion, called an AMAC, to construct an ADEM that is safe against active attackers. This composition is similar to composition N2 from NRS.

3.2.1 Primitives

ADEM An ADEM scheme is defined by a tuple $(A.\text{enc}, A.\text{dec})$. Deterministic algorithm $A.\text{enc}$ takes a key k in \mathcal{K} , a lock l in \mathcal{L} and a message m in \mathcal{M} and outputs a ciphertext c in \mathcal{C} . Deterministic algorithm $A.\text{dec}$ takes a k in \mathcal{K} , a lock l in \mathcal{L} and a ciphertext c in \mathcal{C} and outputs a message m in \mathcal{M} or \perp to indicate rejection. The correctness requirement is that for every combination of k, l and m we have $A.\text{dec}(k, l, A.\text{enc}(k, l, m)) = m$. We will consider both CPA and CCA security separately for this scheme.

ADEM-CPA security The security of a ADEM-CPA, just called ADEM by GKP, is defined as

$$\text{Adv}_{\text{ADEM}, A, N}^{\text{l-ind-lor-cpa}} = \Pr[\text{L-IND-LOR-CPA}_{A, N}^0 = 0] - \Pr[\text{L-IND-LOR-CPA}_{A, N}^1 = 0]$$

where L-IND-LOR-CPA is in Figure 4. Every user is only allowed one encryption as enforced by lines 3, 6 and 11. Locks may not repeat between users as enforced by lines 0, 7, 8 and 9. The corresponding game can be found in Figure 9 from GKP. Note that this figure also includes a decryption oracle, but the adversary is not allowed to use this oracle considering CPA security.

ADEM-CCA security The security of an ADEM-CCA, called ADEM' by GKP, is defined as

$$\text{Adv}_{\text{ADEM}', A, N}^{\text{l-ind-lor-cca}} = \Pr[\text{L-IND-LOR-CCA}_{A, N}^0 = 0] - \Pr[\text{L-IND-LOR-CCA}_{A, N}^1 = 0]$$

where L-IND-LOR-CCA is in figure 5. Every user is only allowed one encryption query as enforced by lines 3, 6 and 11. Locks may not repeat between users as enforced by lines 0, 7, 8 and 9. Decryption queries are only allowed after the given user made an encryption as enforced by lines 3, 11 and 13. Line 14 prevents trivial distinctions. The corresponding game can be found in Figure 9 of GKP.

AMAC An AMAC scheme is defined by a tuple $(M.\text{mac}, M.\text{vrf})$. Deterministic algorithm $M.\text{mac}$ takes a key k in \mathcal{K} , a lock l in \mathcal{L} , and a message m in \mathcal{M} and outputs a tag t in \mathcal{T} .

Game L-IND-LOR-CCA _{A,N} ^b	Oracle Oenc(j, l, m_0, m_1)	Oracle Odec(j, c)
0 : $L \leftarrow \emptyset$	6 : if $C_j \neq \emptyset$: return \perp	13 : if $C_j = \emptyset$: return \perp
1 : for $j \in [1..N]$:	7 : if $l \in L$: return \perp	14 : if $c \in C_j$: return \perp
2 : $k_j \xleftarrow{\$} \mathcal{K}$	8 : $L \leftarrow L \cup \{l\}$	15 : $m \leftarrow A.\text{dec}'(k_j, l_j, c)$
3 : $C_j \leftarrow \emptyset$	9 : $l_j \leftarrow l$	16 : return m
4 : $b' \leftarrow A$	10 : $c \leftarrow A.\text{enc}'(k_j, l_j, m_b)$	
5 : return b'	11 : $C_j \leftarrow C_j \cup \{c\}$	
	12 : return c	

Figure 5: L-IND-LOR-CCA game, A has access to oracles Oenc and Odec and the locks in line 10 and 15 are the same.

Game L-MIOT-UF _{A,N}	Oracle Omac(j, l, m)	Oracle Ovrif(j, m, t)
0 : $\text{forged} \leftarrow 0$	7 : if $T_j \neq \emptyset$: return \perp	14 : if $T_j = \emptyset$: return \perp
1 : $L \leftarrow \emptyset$	8 : if $l \in L$: return \perp	15 : if $(m, t) \in T_j$: return \perp
2 : for $j \in [1..N]$:	9 : $L \leftarrow L \cup \{l\}$	16 : if $M.\text{vrf}(k_j, l_j, m, t)$:
3 : $k_j \xleftarrow{\$} \mathcal{K}$	10 : $l_j \leftarrow l$	17 : $\text{forged} \leftarrow 1$
4 : $T_j \leftarrow \emptyset$	11 : $t \leftarrow M.\text{mac}(k_j, l_j, m)$	18 : return <i>true</i>
5 : run A	12 : $T_j \leftarrow T_j \cup \{(m, t)\}$	19 : else : return <i>false</i>
6 : return forged	13 : return t	

Figure 6: L-MIOT-UF game, A has access to oracles Omac and Ovrif and the locks in line 11 and 16 are the same.

Deterministic algorithm $M.\text{vrf}$ takes a key k in \mathcal{K} , a lock l in \mathcal{L} , a message m in \mathcal{M} and a tag t in \mathcal{T} and returns either *true* or *false*. The correctness requirement is that for every combination of k , l and m , all corresponding $t \leftarrow M.\text{mac}(k, l, m)$ gives $M.\text{vrf}(k, l, m, t) = \text{true}$.

AMAC security The security of a AMAC is defined as

$$\text{Adv}_{\text{AMAC}, A, N}^{\text{L-MIOT-UF}} = \Pr[\text{L-MIOT-UF}_{A, N} = 1]$$

where L-MIOT-UF is in figure 6. Every user is only allowed one MAC query as enforced by lines 4, 7 and 12. Locks may not repeat between users as enforced by lines 1, 8, 9 and 10. Verification queries are only allowed after the user made an mac query as enforced by lines 4, 12 and 14. Line 15 prevents trivial distinctions. The corresponding game can be found in Figure 15 of GKP.

Notational Differences GKP do not require \mathcal{M} to contain at least two strings, and to contain all strings of length x if it contains a string of length x . Additionally, \mathcal{K} is required to be finite but not required to be non-empty.

3.2.2 Composition

GKP construct an ADEM scheme that is CCA secure, using an ADEM scheme that is CPA secure and an AMAC scheme. The composition follows the the encrypt-then-MAC method from [2].

Proc A.enc'(k, l, m)	Proc A.dec'(k, l, c)
0 : $(k_{dem}, k_{mac}) \leftarrow k$	5 : $(k_{dem}, k_{mac}) \leftarrow k$
1 : $c' \leftarrow A.enc(k_{dem}, l, m)$	6 : $(c', t) \leftarrow c$
2 : $t \leftarrow M.mac(k_{mac}, l, c')$	7 : if M.vrf(k_{mac}, l, c', t) :
3 : $c \leftarrow (c', t)$	8 : $m \leftarrow A.dec(k_{dem}, l, c')$
4 : return c	9 : return m
	10 : else : return \perp

Figure 7: A.enc and A.dec calls, The corresponding calls can be found in Figure 16 of GKP.

The resulting algorithms A.enc' and A.dec' are in Figure 7. They define the composition to be secure as there is a tight reduction from breaking the ADEM-CCA of the scheme to breaking the ADEM-CPA or the AMAC security of the underlying primitives

3.3 Comparison of GKP and NRS

In this section we will highlight how GKP and NRS differ, as well as why.

Context and Aim Historically, a single-user that reuses a single key is considered in a symmetric context, NRS follows this trend as they wrote in a this context. In contrast, GKP wrote in the context of hybrid encryption, a context that considers multiple users that use their encryption key once. Apart from this difference in context, there is also a different aim. While NRS aims to generalize the generic nAE composition, GKP aims to find a single composition that can be used for hybrid encryption. Most notably, this results in NRS evaluating 20 possible compositions while GKP evaluates one. Additionally, NRS incorporates AD while GKP does not.

Security Notion The security notions from both papers also reflect the differences in contexts. NRS writes the security notions in a IND-\$ fashion, common in symmetric cryptography. Conversely, GKP writes them in a IND-LOR fashion, common in Hybrid encryption. In other words, NRS requires the valid ciphertext to be indistinguishable from random strings while GKP requires them to be indistinguishable from each other. As a result, the MAC primitives of the two papers have different security requirements. NRS requires the tag to be indistinguishable from a random string while GKP requires the tag to be unforgeable. Furthermore, NRS considers nonces while GKP considers locks to match their respective settings.

4 Lock-based Authenticated Encryption

To evaluate the security of generic composition using locks, we define a new security primitive: the lock-based Authenticated Encryption scheme, LAE scheme for short. This LAE is similar to a the nAE from NRS, but it uses locks instead of nonces. Additionally it does not use associated data (AD). We will evaluate the security in a multi user setting where encryption keys are used once.

Game $\text{IAE-IND-}\$ \text{-CCA}_{A,N}^b$	Oracle $\text{Oenc}(j, l, m)$	Oracle $\text{Odec}(j, c)$
0: $L \leftarrow \emptyset$	6: if $C_j \neq \perp$: return \perp	15: if $C_j = \perp$: return \perp
1: for $j \in [1..N]$:	7: if $l \in L$: return \perp	16: if $c = C_j$: return \perp
2: $k_j \xleftarrow{\$} \mathcal{K}$	8: $L \leftarrow L \cup \{l\}$	17: $m \leftarrow \text{AE.dec}(k_j, l_j, c)$
3: $C_j \leftarrow \perp$	9: $l_j \leftarrow l$	18: if $b = 1$: $m \leftarrow \perp$
4: $b' \leftarrow A$	10: $c \leftarrow \text{AE.enc}(k_j, l_j, m)$	19: return m
5: return b'	11: if $b = 1 \wedge c \neq \perp$:	
	12: $c \xleftarrow{\$} \{0, 1\}^{ c }$	
	13: $C_j \leftarrow c$	
	14: return c	

Figure 8: IAE-IND- $\$$ -CCA game, adversary has access to oracles Oenc and Odec.

4.1 IAE

A IAE scheme is defined by a tuple $(\text{AE.enc}, \text{AE.dec})$. Deterministic algorithm AE.enc takes three inputs (k, l, m) and outputs a value c , where the length of c only depends on the length of k , l and m . If, and only if, (k, l, m) is not in $\mathcal{K} \times \mathcal{L} \times \mathcal{M}$, c will be \perp . Deterministic algorithm AE.dec takes three inputs (k, l, c) and outputs a value m . Both AE.enc and EA.dec are required to satisfy correctness (if $\text{AE.enc}(k, l, m) = c \neq \perp$, then $\text{AE.dec}(k, l, c) = m$) and tidiness (if $\text{AE.dec}(k, l, c) = m \neq \perp$, then $\text{AE.enc}(k, l, m) = c$).

4.2 IAE Security Model

The security is defined as

$$\text{Adv}_{A,N}^{\text{IAE}} = \Pr[\text{IAE-IND-}\$ \text{-CCA}_{A,N}^0 = 0] - \Pr[\text{IAE-IND-}\$ \text{-CCA}_{A,N}^1 = 0]$$

where IAE-IND- $\$$ -CCA is in Figure 8. Because we consider multiple users who use their keys once, decryption queries of a user are only allowed after an encryption has been made and the user is only allowed one encryption query. On decryption, we use a function that always returns \perp to ensure the adversary cannot guess which ciphertexts would be valid ciphertexts. The idea behind the resulting security game is explained below.

Multiple users Line 1 loops over all the users to initialize with a random key in line 2 and an invalid ciphertext in line 3. Whenever the adversary calls one of the oracles Oenc or Odec, it has to specify user j .

Locks Line 0 initializes the set of all used locks to the empty set. Locks are not allowed to repeat, if the lock is in the set of used locks we return \perp on line 7. If this check passes, we add the lock to the sets of used locks in line 8 and bind it to the user in line 9. Note that locks may be added to the set of used locks even if they are never used to encrypt a valid message. (**todo: see if this needs to be altered**)

One-time use keys The variable C_j is used to prevent multiple encryptions per user. In contrast to GKP, we do not use set notation, as we can never have multiple ciphertexts related

to one user. In line 3, we set C_j to be undefined, if the ciphertext is defined in line 6, we return \perp . In line 13, the newly computed ciphertext is bound to C_j . If the encryption was invalid, C_j will stay undefined. This leads to the adversary being able to call Oenc twice on a single user, but will not give the adversary an advantage as the values for which AE.enc returns \perp are known. If the user has made no valid encryption yet, decryption is not allowed and we return \perp on line 15 as C_j will be undefined.

Preventing trivial distinctions Line 16 prevents trivial distinctions. If the ciphertext given to Odec is allowed to be the same as the ciphertext returned by Oenc , it would be trivial to distinguish the real and ideal world. In this case, the ideal world would return \perp while the real world would not. For this reason the real world should return \perp as well.

Encryption and decryption If the given arguments are valid, and we are in the real world, line 10 encrypts the message and line 17 decrypts the message.

Implementation of $\$$ On encryption, whenever AE returns \perp , the random function should return \perp as well. Therefore, the random function is only called if $b = 1$ and AE.enc does not return \perp . This is checked in line 11. If the check passes, the random function lazily samples a string uniformly at random from the set of all strings with the length of the ciphertext. This random string is bound to the ciphertext in line 12. On decryption, the ideal world always returns \perp . (**todo: add part about ideal vs attainable**)

5 Composition

In this section we discuss how we can construct a safe IAE . Similarly to GKP and NRS we will look at compositions combining a deterministic encryption primitive and MAC primitive. First, we write down the definitions of these two primitives, then we will look at how we can combine the two and which security bounds we can expect. Lastly we compare our choices with existing alternatives.

5.1 Used Primitives

IE A lock-based use encryption scheme, IE for short, is defined by a tuple $(\text{E.enc}, \text{E.dec})$. Deterministic algorithm E.enc takes three inputs (k, l, m) and outputs a value c , the length of c only depends on the length of k , l and m . If, and only if, (k, l, m) is not in $\mathcal{K} \times \mathcal{L} \times \mathcal{M}$, c will be \perp . Deterministic algorithm E.dec takes three inputs (k, l, c) and outputs a value m . Both E.enc and E.dec are required to satisfy correctness (if $\text{E.enc}(k, l, m) = c \neq \perp$, then $\text{E.dec}(k, l, c) = m$) and tidiness (if $\text{E.dec}(k, l, c) = m \neq \perp$, then $\text{E.enc}(k, l, m) = c$). Ciphertext space \mathcal{C} consists of all valid ciphertexts.

IE security The security of a IE is defined as

$$\text{Adv}_{A,N}^{\text{IE}} = \Pr[\text{IE-IND-}\$-\text{CPA}_{A,N}^0 = 0] - \Pr[\text{IE-IND-}\$-\text{CPA}_{A,N}^1 = 0]$$

where $\text{IE-IND-}\$-\text{CPA}$ is in Figure 9. The user is only allowed one encryption query and decryption queries are only allowed after the encryption. Locks may not repeat between users.

Game $\text{IE-IND-}\$ \text{-CPA}_{A,N}^b$	Oracle $\text{Oenc}(j, l, m)$
0: $L \leftarrow \emptyset$	6: if $C_j \neq \perp$: return \perp
1: for $j \in [1..N]$:	7: if $l \in L$: return \perp
2: $k_j \xleftarrow{\$} \mathcal{K}$	8: $L \leftarrow L \cup \{l\}$
3: $C_j \leftarrow \perp$	9: $l_j \leftarrow l$
4: $b' \leftarrow A$	10: $c \leftarrow \text{E.enc}(k_j, l_j, m)$
5: return b'	11: if $b = 1 \wedge c \neq \perp$:
	12: $c \xleftarrow{\$} \{0, 1\}^{ c }$
	13: $C_j \leftarrow c$
	14: return c

Figure 9: IE-IND- $\$$ -CPA game, A has access to oracle Oenc .

IMAC A lock-based MAC is defined by a deterministic algorithm M.mac that takes a fixed length k in \mathcal{K} , a fixed length l in \mathcal{L} and a variable length message m in \mathcal{M} and outputs either a n -bit length string we call tag t , or \perp . If, and only if, (k, l, m) is not in $\mathcal{K} \times \mathcal{L} \times \mathcal{M}$, t will be \perp . Tag space \mathcal{T} consists of all valid tags.

IMAC security The security of a lock bases, one-time use PRF secure MAC is defined as

$$\text{Adv}_{\text{F},A,N}^{\text{IMAC}} = \Pr[\text{IMAC-PRF}_{A,N}^0 = 0] - \Pr[\text{IMAC-PRF}_{A,N}^1 = 0]$$

where IMAC-PRF is in Figure 10. Every user is only allowed one MAC query and verification queries are only allowed after the MAC query. Locks may not repeat between users. In contrast to the MAC-PRF from NRS, a verification oracle is needed as we only allow one Omac query per user. In the real world Ovrif will check similar constraints as the Odec from Figure 8. If a Omac query has been made for the given user, and the given message-tag pair is not the result of this query, then the pair is verified. In the ideal world, uniformly random function tag is used instead of the IMAC. To define this function we write $\text{Func}(\mathcal{K} \times \mathcal{L} \times \mathcal{M}, \mathcal{T})$ to denote the set of all functions from key space \mathcal{K} , lock space \mathcal{L} and message space \mathcal{M} to tag space \mathcal{T} . We need to define this function specifically as we want the tags resulting from computations in oracle Ovrif to match with those in oracle Omac. When the input of tag is outside its domain, it will return \perp .

5.2 Composition

Following NRS, three ways to construct this IAE are of interest, namely the ones following from the N1, N2 and N3 scheme. The schemes, adjusted to our setting, are in Figure 11. NRS considers 17 more schemes but as none of them has proven to be secure we will not consider those. The AE.enc and AE.dec calls corresponding to N1, N2 and N3 are in Figure 12, 13 and 14 respectively.

5.3 Security Bounds

We define the composition secure if there is a tight (**todo: look at tight sec reductions**) reduction from breaking the IAE-security of the scheme to breaking the IE-security or the IMAC security of the underlying primitives. More specifically, we proof the following theorem:

Game $\text{IMAC-PRF}_{A,N}^b$	Oracle $\text{Omac}(j, l, m)$	Oracle $\text{Ovrf}(j, m, t)$
0 : $L \leftarrow \emptyset$	8 : if $T_j \neq \perp$: return \perp	15 : if $T_j = \perp$: return \perp
1 : if $b = 1$:	9 : if $l \in L$: return \perp	16 : if $(m, t) = T_j$: return \perp
2 : $\text{tag} \xleftarrow{\$} \text{Func}(\mathcal{K} \times \mathcal{L} \times \mathcal{M})$	10 : $L \leftarrow L \cup \{l\}$	17 : $t' \leftarrow \text{M.mac}(k_j, l_j, m)$
3 : for $j \in [1..N]$:	11 : $l_j \leftarrow l$	18 : if $b = 1$:
4 : $k_j \xleftarrow{\$} \mathcal{K}$	12 : $t \leftarrow \text{M.mac}(k_j, l_j, m)$	19 : $t' \leftarrow \text{tag}(k_j, l_j, m)$
5 : $T_j \leftarrow \perp$	13 : if $b = 1 \wedge t \neq \perp$:	20 : if $t = t'$
6 : $b' \leftarrow A$	14 : $t \leftarrow \text{tag}(k_j, l_j, m)$	21 : return <i>true</i>
7 : return b'	15 : $T_j \leftarrow (m, t)$	22 : return <i>false</i>
	16 : return t	

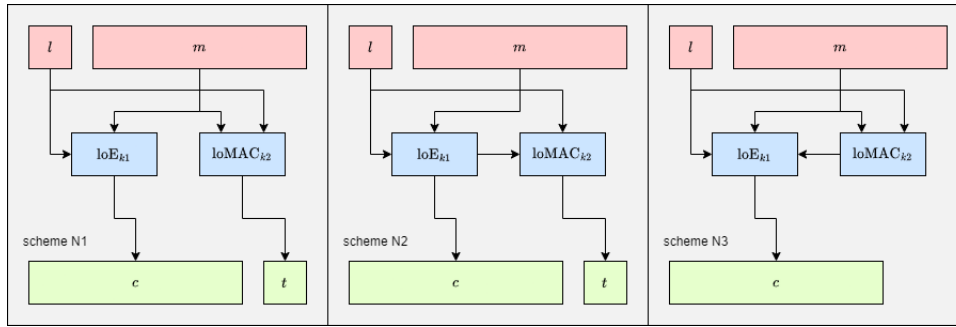
Figure 10: IMAC-PRF game, A has access to oracle Omac .

Figure 11: Adjusted N schemes from NRS

$\text{AE.enc}(k, l, m)$	$\text{AE.dec}(k, l, c)$
0 : $(k1, k2) \leftarrow k$	5 : $(k1, k2) \leftarrow k$
1 : $c' \leftarrow \text{E.enc}(k1, l, m)$	6 : $(c', t) \leftarrow c$
2 : $t \leftarrow \text{M.mac}(k2, l, m)$	7 : $m \leftarrow \text{E.dec}(k1, l, c')$
3 : $c \leftarrow (c', t)$	8 : $t' \leftarrow \text{M.mac}(k2, l, m)$
4 : return c	9 : if $t \neq t'$: $m \leftarrow \perp$
	10 : return m

Figure 12: Calls based on N1

AE.enc(k, l, m)	AE.dec(k, l, c)
0 : $(k1, k2) \leftarrow k$	5 : $(k1, k2) \leftarrow k$
1 : $c' \leftarrow \text{E.enc}(k1, l, m)$	6 : $(c', t) \leftarrow c$
2 : $t \leftarrow \text{M.mac}(k2, l, c')$	7 : $m \leftarrow \text{E.dec}(k1, l, c')$
3 : $c \leftarrow (c', t)$	8 : $t' \leftarrow \text{M.mac}(k2, l, c')$
4 : return c	9 : if $t \neq t' : m \leftarrow \perp$
	10 : return m

Figure 13: Calls based on N2

AE.enc(k, l, m)	AE.dec(k, l, c)
0 : $(k1, k2) \leftarrow k$	5 : $(k1, k2) \leftarrow k$
1 : $t \leftarrow \text{M.mac}(k2, l, m)$	6 : $m' \leftarrow \text{E.dec}(k1, l, c)$
2 : $m' \leftarrow m t$	7 : $(m, t) \leftarrow m'$
3 : $c \leftarrow \text{E.enc}(k1, l, m')$	8 : $t' \leftarrow \text{M.mac}(k2, l, m)$
4 : return c	9 : if $t \neq t' : m \leftarrow \perp$
	10 : return m

Figure 14: Calls based on N3

Theorem 1. *Let LAE be constructed from lMAC and lE as described in Figure 12, 13 or 14. Let ciphertext space \mathcal{C} from the lE be a subset of message space \mathcal{M} from the lMAC and let lMAC and lE have a shared lock space. Then, for any number of users N and any LAE adversary A that poses at most Q_e many Oenc queries, and at most Q_d many Odec queries, there exist a lMAC adversary B and a lE adversary C such that:*

$$\text{Adv}_{A,N}^{\text{LAE}} \leq \text{Adv}_{B,N}^{\text{lMAC}} + \frac{Q_d}{2^n} + \text{Adv}_{C,N}^{\text{lE}},$$

where n is the output length of the lMAC in bits. The running time of B is at most that of A plus the time required to run Q_e many E.enc encapsulations and Q_d many E.dec decapsulations. The running time of C is at most that of A . Additionally, B makes at most Q_e many Omac queries and at most Q_d many Ovrq queries and C makes at most Q_e many Oenc queries.

Within this theorem, both Q_e and Q_d refer to the total queries the adversary is allowed to make, not the queries per user. As a result Q_e is limited by N .

To prove this we prove the cases N1, N2 and N3 separately, the full proof can be found in appendix B.

6 Use Cases

should consist of:

- possible use cases

6.1 PKE Schemes

7 Related Work

(Note: I just have a general structure for now)

- multi-user setting: GKP, possible other problem to fix the degradation of bounds in multi user setting?
- Generic composition: "Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm" by Mihir Bellare and Chanathip Namprempre, NRS and the follow up for NRS which proofs N4
- non-generic AEAD constructions where locks might also work: "GCM" and "OCB"

(question: do you think this is enough?)

8 Open Problems

(Note: I just have a general structure and some questions for now)

- Other possible N schemes that are not proven secure or insecure
- Addition of AD
- adding locks to non-generic AE constructions

(question: do you think this is enough?)

9 Conclusion

References

- [1] F. Giacon, E. Kiltz, and B. Poettering, “Hybrid encryption in a multi-user setting, revisited,” 2018, pp. 159–189. DOI: [10.1007/978-3-319-76578-5_6](https://doi.org/10.1007/978-3-319-76578-5_6).
- [2] M. Bellare and C. Namprempre, “Authenticated encryption: Relations among notions and analysis of the generic composition paradigm,” 2000, pp. 531–545. DOI: [10.1007/3-540-44448-3_41](https://doi.org/10.1007/3-540-44448-3_41).
- [3] C. Namprempre, P. Rogaway, and T. Shrimpton, “Reconsidering generic composition,” 2014, pp. 257–274. DOI: [10.1007/978-3-642-55220-5_15](https://doi.org/10.1007/978-3-642-55220-5_15).
- [4] M. Bellare and P. Rogaway, “The security of triple encryption and a framework for code-based game-playing proofs,” 2006, pp. 409–426. DOI: [10.1007/11761679_25](https://doi.org/10.1007/11761679_25).

Appendix A

To proof Theorem 1, we proof the cases N1, N2 and N3 separately, after proving all three we can conclude Theorem 1 holds.

10 N1

First, we define our theorem:

Theorem 2. *Let lAE be constructed from lMAC and lE as described in Figure 12 Let ciphertext space \mathcal{C} from the lE be a subset of message space \mathcal{M} from the lMAC and let lMAC and lE have a shared lock space. Then, for any number of users N and any lAE adversary A that poses at most Q_e many Oenc queries, and at most Q_d many Odec queries, there exist a lMAC adversary B and a lE adversary C such that:*

$$\mathbf{Adv}_{A,N}^{\text{lAE}} \leq \mathbf{Adv}_{B,N}^{\text{lMAC}} + \frac{Q_d}{2^n} + \mathbf{Adv}_{C,N}^{\text{lE}} ,$$

where n is the output length of the lMAC in bits. The running time of B is at most that of A plus the time required to run Q_e many E.enc encapsulations and Q_d many E.dec decapsulations. The running time of C is at most that of A . Additionally, B makes at most Q_e many Omac queries and at most Q_d many Ovrif queries and C makes at most Q_e many Oenc queries.

Within this theorem, both Q_e and Q_d refer to the total queries the adversary is allowed to make, not the queries per user. As a result Q_e is limited by N .

Proof. To prove this theorem we start by defining game lAE-N1 in Figure 15. This game is the game lAE-IND-\$-CCA (Figure 8), with AE.enc and AE.dec substituted with the N1 algorithms from Figure 12.

Game $\text{lAE-N1}_{A,N}^b$	Oracle $\text{Oenc}(j, l, m)$	Oracle $\text{Odec}(j, c)$
0 : $L \leftarrow \emptyset$	6 : if $C_j \neq \perp$: return \perp	18 : if $C_j = \perp$: return \perp
1 : for $j \in [1..N]$:	7 : if $l \in L$: return \perp	19 : if $c = C_j$: return \perp
2 : $k_j \xleftarrow{\$} \mathcal{K}$	8 : $L \leftarrow L \cup \{l\}$	20 : $(k1, k2) \leftarrow k_j$
3 : $C_j \leftarrow \perp$	9 : $l_j \leftarrow l$	21 : $(c', t) \leftarrow c$
4 : $b' \leftarrow A$	10 : $(k1, k2) \leftarrow k_j$	22 : $m \leftarrow \text{E.dec}(k1, l_j, c')$
5 : return b'	11 : $c' \leftarrow \text{E.enc}(k1, l_j, m)$	23 : $t' \leftarrow \text{M.mac}(k2, l_j, m)$
	12 : $t \leftarrow \text{M.mac}(k2, l_j, m)$	24 : if $t \neq t' : m \leftarrow \perp$
	13 : $c \leftarrow (c', t)$	25 : if $b = 1 : m \leftarrow \perp$
	14 : if $b = 1 \wedge c \neq \perp$:	26 : return m
	15 : $c \xleftarrow{\$} \{0, 1\}^{ c }$	
	16 : $C_j \leftarrow c$	
	17 : return c	

Figure 15: lAE-N1 game, adversary has access to oracles Oenc and Odec.

By definition, this gives us

$$\text{Adv}_{A,N}^{\text{lAE}} = \Pr[\text{lAE-N1}_{A,N}^0 = 0] - \Pr[\text{lAE-N1}_{A,N}^1 = 0].$$

Next we define game N1-switch-1 in Figure 16. The only difference between this game and game lAE-N1^0 is the fact that N1-switch-1 uses the uniformly random function tag , instead of the IMAC. To define this function we write $\text{Func}(\mathcal{K} \times \mathcal{L} \times \mathcal{M}, \mathcal{T})$ to denote the set of all functions from the key space of the MAC \mathcal{K} , the shared lock space \mathcal{L} and message space \mathcal{M} to the tag space \mathcal{T} . We define this function specifically as we want the tags resulting from computations in oracle Oenc to match with those in oracle Odec. When the input of tag is outside its domain, it will return \perp .

Game $\text{N1-switch-1}_{A,N}$	Oracle $\text{Oenc}(j, l, m)$	Oracle $\text{Odec}(j, c)$
0 : $L \leftarrow \emptyset$	7 : if $C_j \neq \perp$: return \perp	17 : if $C_j = \perp$: return \perp
1 : $\text{tag} \xleftarrow{\$} \text{Func}(\mathcal{K}_{\text{mac}} \times \mathcal{L} \times \mathcal{M})$	8 : if $l \in L$: return \perp	18 : if $c = C_j$: return \perp
2 : for $j \in [1..N]$:	9 : $L \leftarrow L \cup \{l\}$	19 : $(k1, k2) \leftarrow k_j$
3 : $k_j \xleftarrow{\$} \mathcal{K}$	10 : $l_j \leftarrow l$	20 : $(c', t) \leftarrow c$
4 : $C_j \leftarrow \perp$	11 : $(k1, k2) \leftarrow k_j$	21 : $m \leftarrow \text{E.dec}(k1, l_j, c')$
5 : $b' \leftarrow A$	12 : $c' \leftarrow \text{E.enc}(k1, l_j, m)$	22 : $t' \leftarrow \text{tag}(k2, l_j, m)$
6 : return b'	13 : $t \leftarrow \text{tag}(k2, l_j, m)$	23 : if $t \neq t' : m \leftarrow \perp$
	14 : $c \leftarrow (c', t)$	24 : return m
	15 : $C_j \leftarrow c$	
	16 : return c	

Figure 16: N1-switch-1, adversary has access to oracles Oenc and Odec. Key space \mathcal{K}_{mac} is the key space from M.mac. Lines 13 and 22 are different compared to lAE-N1^0 , additionally lines 14, 15 and 25 from lAE-N1 are removed.

Using this game, we expand the probability:

$$\begin{aligned} \mathbf{Adv}_{A,N}^{\text{IAE}} &= \Pr[\text{IAE-N1}_{A,N}^0 = 0] - \Pr[\text{N1-switch-1}_{A,N} = 0] \\ &\quad + \Pr[\text{N1-switch-1}_{A,N} = 0] - \Pr[\text{IAE-N1}_{A,N}^1 = 0]. \end{aligned}$$

Next, we can rewrite $\Pr[\text{IAE-N1}_{A,N} = 0] - \Pr[\text{N1-switch-1}_{A,N} = 0]$ into a IMAC advantage. To do so, we define adversary B against IMAC in Figure 17. This adversary is playing game IMAC-PRF (Figure 10), and has access to A .

Adversary B	if A calls Oracle Oenc(j, l, m)	if A calls Oracle Odec(j, c)
0 : $L \leftarrow \emptyset$	6 : if $C_j \neq \perp$: return \perp	15 : if $C_j = \perp$: return \perp
1 : for $j \in [1..N]$:	7 : if $l \in L$: return \perp	16 : if $c = C_j$: return \perp
2 : $k_j \xleftarrow{\$} \mathcal{K}_{\text{enc}}$	8 : $L \leftarrow L \cup \{l\}$	17 : $(c', t) \leftarrow c$
3 : $C_j \leftarrow \perp$	9 : $l_j \leftarrow l$	18 : $m \leftarrow \text{E.dec}(k_j, l_j, c')$
4 : $b' \leftarrow \text{run } A$	10 : $c' \leftarrow \text{E.enc}(k_j, l_j, m)$	19 : $\text{passed} \leftarrow \text{Ovrf}(j, m, t')$
5 : return b'	11 : $t \leftarrow \text{Omac}(j, l_j, m)$	20 : if $\neg \text{passed}$: $m \leftarrow \perp$
	12 : $c \leftarrow (c', t)$	21 : return m
	13 : $C_j \leftarrow c$	
	14 : return c	

Figure 17: Adversary B , has access to A and oracles Omac and Ovrf. Key space \mathcal{K}_{enc} is the key space from E.enc.

The runtime of B is that of A . For every Oenc query A makes, B computes E.enc once, and calls Omac once. For every Odec query A makes, B computes E.dec once and calls Ovrf once. Note that, alternatively, B could return 0 if passed is *true* to avoid having to do E.dec computations. To increase consistency with the other two cases, these computations are still made. We can see that $\Pr[\text{IMAC-PRF}_{B,N}^0 = 0] = \Pr[\text{IAE-N1}_{A,N}^0 = 0]$ as B perfectly simulates game IAE-N1 with $b = 0$ when its own b is 0. In addition, $\Pr[\text{IMAC-PRF}_{B,N}^1 = 0] = \Pr[\text{N1-switch-1}_{A,N} = 0]$ as B perfectly simulates game N1-switch-1 whenever its own b is 1. As a result we can rewrite our advantage to:

$$\begin{aligned} \mathbf{Adv}_{A,N}^{\text{IAE}} &= \Pr[\text{IAE-N1}_{A,N}^0 = 0] - \Pr[\text{N1-switch-1}_{A,N} = 0] \\ &\quad + \Pr[\text{N1-switch-1}_{A,N} = 0] - \Pr[\text{IAE-N1}_{A,N}^1 = 0] \\ &= \Pr[\text{IMAC-PRF}_{B,N}^0 = 0] - \Pr[\text{IMAC-PRF}_{B,N}^1 = 0] \\ &\quad + \Pr[\text{N1-switch-1}_{A,N} = 0] - \Pr[\text{IAE-N1}_{A,N}^1 = 0] \\ &= \mathbf{Adv}_{B,N}^{\text{IMAC}} + \Pr[\text{N1-switch-1}_{A,N} = 0] - \Pr[\text{IAE-N1}_{A,N}^1 = 0]. \end{aligned}$$

To expand our probability again, we define game N1-switch-2 in Figure 18. The Odec oracle from this game always returns \perp , apart from this difference, it is equivalent to the first switch game. Although the Odec always returns \perp , it is written down more elaborately to include the event *bad*, this supports a well-known proof tactic [4]. We use this game to expand our probability:

$$\begin{aligned} \mathbf{Adv}_{A,N}^{\text{IAE}} &= \mathbf{Adv}_{B,N}^{\text{IMAC}} + \Pr[\text{N1-switch-1}_{A,N} = 0] - \Pr[\text{N1-switch-2}_{A,N} = 0] \\ &\quad + \Pr[\text{N1-switch-2}_{A,N} = 0] - \Pr[\text{IAE-N1}_{A,N}^1 = 0]. \end{aligned}$$

Game N1-switch-2 _{A,N}	Oracle Oenc(<i>j, l, m</i>)	Oracle Odec(<i>j, c</i>)
0 : $L \leftarrow \emptyset$	7 : if $C_j \neq \perp$: return \perp	17 : if $C_j = \perp$: return \perp
1 : $tag \xleftarrow{\$} Func(\mathcal{K}_{mac} \times \mathcal{L} \times \mathcal{M})$	8 : if $l \in L$: return \perp	18 : if $c = C_j$: return \perp
2 : for $j \in [1..N]$:	9 : $L \leftarrow L \cup \{l\}$	19 : $(k1, k2) \leftarrow k_j$
3 : $k_j \xleftarrow{\$} \mathcal{K}$	10 : $l_j \leftarrow l$	20 : $(c', t) \leftarrow c$
4 : $C_j \leftarrow \perp$	11 : $(k1, k2) \leftarrow k_j$	21 : $m \leftarrow E.dec(k1, l_j, c')$
5 : $b' \leftarrow A$	12 : $c' \leftarrow E.enc(k1, l_j, m)$	22 : $t' \leftarrow tag(k2, l_j, m)$
6 : return b'	13 : $t \leftarrow tag(k2, l_j, m)$	23 : if $t \neq t' : m \leftarrow \perp$
	14 : $c \leftarrow (c', t)$	24 : else :
	15 : $C_j \leftarrow c$	25 : $bad \leftarrow true$
	16 : return c	26 : $m \leftarrow \perp$
		27 : return m

Figure 18: N1-switch-2 game, adversary has access to oracles Oenc and Odec. Key space \mathcal{K}_{mac} is the key space from M.mac. Line 24-26 are different compared to N1-switch-1.

As N1-switch-1 and N1-switch-2 are so called identical-until-*bad* [4], meaning they are equivalent as long as the event *bad* is not set to *true*, we know:

$$\Pr[\text{N1-switch-1}_{A,N} = 0] - \Pr[\text{N1-switch-2}_{A,N} = 0] \leq \Pr[bad = true].$$

As *bad* is set to *true* if, and only if, $t=t'$, we can state $\Pr[bad = true] = \Pr[t = t']$. The adversary needs to provide tag t and ciphertext c' , where ciphertext c' leads to a message m that is used as input to the *tag* function. The provided tag-ciphertext pair may not be the result of the encryption query corresponding to the provided user. Combined with the tidiness of the encryption, it is ensured that, for any sensible adversarial query, the message m cannot be the message which is encrypted for the provided user. This is because if m would be the encrypted message, the correct tag cannot be provided and thus no information can be gained with the query. Consequently, t and t' are only equal when the adversary is able to guess the output of *tag* for a message that is not encrypted for the provided user. The function *tag* is uniformly random so, with every fresh ciphertext, the probability that t and t' are equal is $\frac{1}{2^n}$. Combined with at most Q_d Odec queries we get $\Pr[t = t'] = \Pr[bad = true] \leq \frac{Q_d}{2^n}$ and thus, we can fill in $\Pr[\text{N2-switch-1}_{A,N} = 0] - \Pr[\text{N1-switch-2}_{A,N} = 0] \leq \Pr[bad = true] \leq \frac{Q_d}{2^n}$ to obtain:

$$\mathbf{Adv}_{A,N}^{\text{IAE}} \leq \mathbf{Adv}_{B,N}^{\text{IMAC}} + \frac{Q_d}{2^n} + \Pr[\text{N1-switch-2}_{A,N} = 0] - \Pr[\text{IAE-N2}_{A,N}^1 = 0].$$

We define game N1-switch-3 in Figure 19 to expand our probability one last time. Switch game 3 is equivalent to switch game 2 but always returns lazily sampled random bits when the outcome of E.enc is valid. It might seem like there is a difference as t can no longer become \perp . This is not the case as, due to the chosen input spaces of tag, tag can never return \perp when the input to E.enc is valid. In other words, tag can only return \perp when c' is already \perp , and thus tag returning \perp will never influence c on line 12. We also simplify Odec as we no longer need the event *bad*. We use this game to expand our probability to:

$$\begin{aligned} \mathbf{Adv}_{A,N}^{\text{IAE}} &\leq \mathbf{Adv}_{B,N}^{\text{IMAC}} + \frac{Q_d}{2^n} + \Pr[\text{N1-switch-2}_{A,N} = 0] - \Pr[\text{N1-switch-3}_{A,N} = 0] \\ &\quad + \Pr[\text{N1-switch-3}_{A,N} = 0] - \Pr[\text{IAE-N1}_{A,N}^1 = 0]. \end{aligned}$$

Game N1-switch-3 _{A,N}	Oracle Oenc(j, l, m)	Oracle Odec(j, c)
0 : $L \leftarrow \emptyset$	7 : if $C_j \neq \perp$: return \perp	19 : return \perp
1 : for $j \in [1..N]$:	8 : if $l \in L$: return \perp	
2 : $k_j \xleftarrow{\$} \mathcal{K}_{enc}$	9 : $L \leftarrow L \cup \{l\}$	
3 : $C_j \leftarrow \perp$	10 : $l_j \leftarrow l$	
4 : $b' \leftarrow A$	11 : $c' \leftarrow \text{E.enc}(k_j, l_j, m)$	
5 : return b'	12 : $t \xleftarrow{\$} \{0, 1\}^n$	
	13 : $c \leftarrow (c', t)$	
	14 : if $c \neq \perp$:	
	15 : $c \xleftarrow{\$} \{0, 1\}^{ c }$	
	16 : $C_j \leftarrow c$	
	17 : return c	

Figure 19: N1-switch-3 game, adversary has access to oracles Oenc and Odec. Key space \mathcal{K}_{enc} is the key space from E.enc. Line 14 and 15 are different compared to N1-switch-2, and Odec is simplified.

Now we can rewrite $\Pr[\text{N1-switch-2}_{A,N} = 0] - \Pr[\text{N1-switch-3}_{A,N} = 0]$ into a IE advantage. To do so, we define adversary C against IE in Figure 20. This adversary is playing game IE-IND- $\$$ -CPA (Figure 9), and has access to A .

Adversary C	if A calls Oracle Oenc(j, l, m)	if A calls Oracle Odec(j, c)
0 : $L \leftarrow \emptyset$	5 : if $C_j \neq \perp$: return \perp	14 : return \perp
1 : for $j \in [1..N]$:	6 : if $l \in L$: return \perp	
2 : $C_j \leftarrow \perp$	7 : $L \leftarrow L \cup \{l\}$	
3 : $b' \leftarrow \text{run } A$	8 : $l_j \leftarrow l$	
4 : return b'	9 : $c' \leftarrow \text{Oenc}(j, l_j, m)$	
	10 : $t \xleftarrow{\$} \{0, 1\}^n$	
	11 : $c \leftarrow (c', t)$	
	12 : $C_j \leftarrow c$	
	13 : return c	

Figure 20: Adversary C , has access to A and oracle Oenc. Note the Oenc in line 9 refers to the encryption oracle Oenc that C has access to, not the oracle Oenc A has access to.

The runtime of C is that of A . For every Oenc query A makes, C makes Q_e one Oenc query. We can see that $\Pr[\text{N1-switch-2}_{A,N} = 0] = \Pr[\text{IE-IND-}\$ \text{-CPA}_{C,N}^0 = 0]$ as C perfectly simulates N1-switch-2 when its own b is 0. In addition, $\Pr[\text{N1-switch-3}_{A,N} = 0] = \Pr[\text{IE-IND-}\$ \text{-CPA}_{C,N}^1 = 0]$

as C perfectly simulates N1-switch-3 when its own b is 1. This gives us:

$$\begin{aligned}
\mathbf{Adv}_{A,N}^{\text{lAE}} &\leq \mathbf{Adv}_{B,N}^{\text{lMAC}} + \frac{Q_d}{2^n} + \Pr[\text{N1-switch-2}_{A,N} = 0] - \Pr[\text{N1-switch-3}_{A,N} = 0] \\
&\quad + \Pr[\text{N1-switch-3}_{A,N} = 0] - \Pr[\text{lAE-N1}_{A,N}^1 = 0]. \\
&\leq \mathbf{Adv}_{B,N}^{\text{lMAC}} + \frac{Q_d}{2^n} + \Pr[\text{lE-IND-}\$-\text{CPA}_{C,N}^0 = 0] - \Pr[\text{lE-IND-}\$-\text{CPA}_{C,N}^1 = 0] \\
&\quad + \Pr[\text{N1-switch-3}_{A,N} = 0] - \Pr[\text{lAE-N1}_{A,N}^1 = 0]. \\
&\leq \mathbf{Adv}_{B,N}^{\text{lMAC}} + \frac{Q_d}{2^n} + \mathbf{Adv}_{C,N}^{\text{lE}} \\
&\quad + \Pr[\text{N1-switch-3}_{A,N} = 0] - \Pr[\text{lAE-N1}_{A,N}^1 = 0].
\end{aligned}$$

Lastly, $\Pr[\text{N1-switch-3}_{A,N} = 0]$ and $\Pr[\text{lAE-N1}_{A,N}^1 = 0]$ are equivalent by definition, giving the result:

$$\mathbf{Adv}_{A,N}^{\text{lAE}} \leq \mathbf{Adv}_{B,N}^{\text{lMAC}} + \frac{Q_d}{2^n} + \mathbf{Adv}_{C,N}^{\text{lE}}.$$

□

11 N2

First, we define our theorem:

Theorem 3. *Let lAE be constructed from lMAC and lE as described in Figure 13. Let ciphertext space \mathcal{C} from the lE be a subset of message space \mathcal{M} from the lMAC and let lMAC and lE have a shared lock space. Then, for any number of users N and any lAE adversary A that poses at most Q_e many Oenc queries, and at most Q_d many Odec queries, there exist a lMAC adversary B and a lE adversary C such that:*

$$\mathbf{Adv}_{A,N}^{\text{lAE}} \leq \mathbf{Adv}_{B,N}^{\text{lMAC}} + \frac{Q_d}{2^n} + \mathbf{Adv}_{C,N}^{\text{lE}},$$

where n is the output length of the lMAC in bits. The running time of B is at most that of A plus the time required to run Q_e many E.enc encapsulations and Q_d many E.dec decapsulations. The running time of C is at most that of A . Additionally, B makes at most Q_e many Omac queries and at most Q_d many Ovrq queries and C makes at most Q_e many Oenc queries.

Within this theorem, both Q_e and Q_d refer to the total queries the adversary is allowed to make, not the queries per user. As a result Q_e is limited by N .

Proof. To prove this theorem we start by defining game lAE-N2 in Figure 21. This game is the game lAE-IND- $\$$ -CCA (Figure 8), with AE.enc and AE.dec substituted with the N2 algorithms from Figure 13.

Game $\text{lAE-N2}_{A,N}^b$	Oracle $\text{Oenc}(j, l, m)$	Oracle $\text{Odec}(j, c)$
0 : $L \leftarrow \emptyset$	6 : if $C_j \neq \perp$: return \perp	18 : if $C_j = \perp$: return \perp
1 : for $j \in [1..N]$:	7 : if $l \in L$: return \perp	19 : if $c = C_j$: return \perp
2 : $k_j \xleftarrow{\$} \mathcal{K}$	8 : $L \leftarrow L \cup \{l\}$	20 : $(k1, k2) \leftarrow k_j$
3 : $C_j \leftarrow \perp$	9 : $l_j \leftarrow l$	21 : $(c', t) \leftarrow c$
4 : $b' \leftarrow A$	10 : $(k1, k2) \leftarrow k_j$	22 : $m \leftarrow \text{E.dec}(k1, l_j, c')$
5 : return b'	11 : $c' \leftarrow \text{E.enc}(k1, l_j, m)$	23 : $t' \leftarrow \text{M.mac}(k2, l_j, c')$
	12 : $t \leftarrow \text{M.mac}(k2, l_j, c')$	24 : if $t \neq t'$: $m \leftarrow \perp$
	13 : $c \leftarrow (c', t)$	25 : if $b = 1$: $m \leftarrow \perp$
	14 : if $b = 1 \wedge c \neq \perp$:	26 : return m
	15 : $c \xleftarrow{\$} \{0, 1\}^{ c }$	
	16 : $C_j \leftarrow c$	
	17 : return c	

Figure 21: lAE-N2 game, adversary has access to oracles Oenc and Odec.

By definition, this gives us

$$\text{Adv}_{A,N}^{\text{lAE}} = \Pr[\text{lAE-N2}_{A,N}^0 = 0] - \Pr[\text{lAE-N2}_{A,N}^1 = 0].$$

Next we define game N2-switch-1 in Figure 22. The only difference between this game and game lAE-N2⁰ is the fact that N2-switch-1 uses the uniformly random function tag , instead of the IMAC. To define this function we write $\text{Func}(\mathcal{K} \times \mathcal{L} \times \mathcal{C}, \mathcal{T})$ to denote the set of all functions from the key space of the MAC \mathcal{K} , the shared lock space \mathcal{L} and ciphertext space from E.enc \mathcal{C} to the tag space \mathcal{T} . We define this function specifically as we want the tags resulting from computations in oracle Oenc to match with those in oracle Odec. When the input of tag is outside its domain, it will return \perp .

Game $\text{N2-switch-1}_{A,N}$	Oracle $\text{Oenc}(j, l, m)$	Oracle $\text{Odec}(j, c)$
0 : $L \leftarrow \emptyset$	7 : if $C_j \neq \perp$: return \perp	17 : if $C_j = \perp$: return \perp
1 : $\text{tag} \xleftarrow{\$} \text{Func}(\mathcal{K}_{\text{mac}} \times \mathcal{L} \times \mathcal{C})$	8 : if $l \in L$: return \perp	18 : if $c = C_j$: return \perp
2 : for $j \in [1..N]$:	9 : $L \leftarrow L \cup \{l\}$	19 : $(k1, k2) \leftarrow k_j$
3 : $k_j \xleftarrow{\$} \mathcal{K}$	10 : $l_j \leftarrow l$	20 : $(c', t) \leftarrow c$
4 : $C_j \leftarrow \perp$	11 : $(k1, k2) \leftarrow k_j$	21 : $m \leftarrow \text{E.dec}(k1, l_j, c')$
5 : $b' \leftarrow A$	12 : $c' \leftarrow \text{E.enc}(k1, l_j, m)$	22 : $t' \leftarrow \text{tag}(k2, l_j, c')$
6 : return b'	13 : $t \leftarrow \text{tag}(k2, l_j, c')$	23 : if $t \neq t'$: $m \leftarrow \perp$
	14 : $c \leftarrow (c', t)$	24 : return m
	15 : $C_j \leftarrow c$	
	16 : return c	

Figure 22: N2-switch-1, adversary has access to oracles Oenc and Odec. Key space \mathcal{K}_{mac} is the key space from M.mac. Lines 13 and 22 are different compared to lAE-N2⁰, additionally lines 14, 15 and 25 from lAE-N2 are removed.

Using this game, we expand the probability:

$$\begin{aligned} \mathbf{Adv}_{A,N}^{\text{IAE}} &= \Pr[\text{IAE-N2}_{A,N}^0 = 0] - \Pr[\text{N2-switch-1}_{A,N} = 0] \\ &\quad + \Pr[\text{N2-switch-1}_{A,N} = 0] - \Pr[\text{IAE-N2}_{A,N}^1 = 0]. \end{aligned}$$

Next, we can rewrite $\Pr[\text{IAE-N2}_{A,N} = 0] - \Pr[\text{N2-switch-1}_{A,N} = 0]$ into a IMAC advantage. To do so, we define adversary B against IMAC in Figure 23. This adversary is playing game IMAC-PRF (Figure 10), and has access to A .

Adversary B	if A calls Oracle Oenc(j, l, m)	if A calls Oracle Odec(j, c)
0 : $L \leftarrow \emptyset$	6 : if $C_j \neq \perp$: return \perp	15 : if $C_j = \perp$: return \perp
1 : for $j \in [1..N]$:	7 : if $l \in L$: return \perp	16 : if $c = C_j$: return \perp
2 : $k_j \xleftarrow{\$} \mathcal{K}_{\text{enc}}$	8 : $L \leftarrow L \cup \{l\}$	17 : $(c', t) \leftarrow c$
3 : $C_j \leftarrow \perp$	9 : $l_j \leftarrow l$	18 : $m \leftarrow \text{E.dec}(k_j, l_j, c')$
4 : $b' \leftarrow \text{run } A$	10 : $c' \leftarrow \text{E.enc}(k_j, l_j, m)$	19 : $\text{passed} \leftarrow \text{Ovrf}(j, c', t')$
5 : return b'	11 : $t \leftarrow \text{Omac}(j, l_j, c')$	20 : if $\neg \text{passed}$: $m \leftarrow \perp$
	12 : $c \leftarrow (c', t)$	21 : return m
	13 : $C_j \leftarrow c$	
	14 : return c	

Figure 23: Adversary B , has access to A and oracles Omac and Ovrf. Key space \mathcal{K}_{enc} is the key space from E.enc.

The runtime of B is that of A . For every Oenc query A makes, B computes E.enc once, and calls Omac once. For every Odec query A makes, B computes E.dec once and calls Ovrf once. Note that, alternatively, B could return 0 if passed is *true* to avoid having to do E.dec computations. To increase consistency with the other two cases, these computations are still made. We can see that $\Pr[\text{IMAC-PRF}_{B,N}^0 = 0] = \Pr[\text{IAE-N2}_{A,N}^0 = 0]$ as B perfectly simulates game IAE-N2 with $b = 0$ when its own b is 0. In addition, $\Pr[\text{IMAC-PRF}_{B,N}^1 = 0] = \Pr[\text{N2-switch-1}_{A,N} = 0]$ as B perfectly simulates game N2-switch-1 whenever its own b is 1. As a result we can rewrite our advantage to:

$$\begin{aligned} \mathbf{Adv}_{A,N}^{\text{IAE}} &= \Pr[\text{IAE-N2}_{A,N}^0 = 0] - \Pr[\text{N2-switch-1}_{A,N} = 0] \\ &\quad + \Pr[\text{N2-switch-1}_{A,N} = 0] - \Pr[\text{IAE-N2}_{A,N}^1 = 0] \\ &= \Pr[\text{IMAC-PRF}_{B,N}^0 = 0] - \Pr[\text{IMAC-PRF}_{B,N}^1 = 0] \\ &\quad + \Pr[\text{N2-switch-1}_{A,N} = 0] - \Pr[\text{IAE-N2}_{A,N}^1 = 0] \\ &= \mathbf{Adv}_{B,N}^{\text{IMAC}} + \Pr[\text{N2-switch-1}_{A,N} = 0] - \Pr[\text{IAE-N2}_{A,N}^1 = 0]. \end{aligned}$$

To expand our probability again, we define game N2-switch-2 in Figure 24. The Odec oracle from this game always returns \perp , apart from this difference, it is equivalent to the first switch game. Although the Odec always returns \perp , it is written down more elaborately to include the event *bad*, this supports a well-known proof tactic [4]. We use this game to expand our probability:

$$\begin{aligned} \mathbf{Adv}_{A,N}^{\text{IAE}} &= \mathbf{Adv}_{B,N}^{\text{IMAC}} + \Pr[\text{N2-switch-1}_{A,N} = 0] - \Pr[\text{N2-switch-2}_{A,N} = 0] \\ &\quad + \Pr[\text{N2-switch-2}_{A,N} = 0] - \Pr[\text{IAE-N2}_{A,N}^1 = 0]. \end{aligned}$$

Game N2-switch-2 _{A,N}	Oracle Oenc(<i>j, l, m</i>)	Oracle Odec(<i>j, c</i>)
0 : $L \leftarrow \emptyset$	7 : if $C_j \neq \perp$: return \perp	17 : if $C_j = \perp$: return \perp
1 : $tag \xleftarrow{\$} Func(\mathcal{K}_{mac} \times \mathcal{L} \times \mathcal{C})$	8 : if $l \in L$: return \perp	18 : if $c = C_j$: return \perp
2 : for $j \in [1..N]$:	9 : $L \leftarrow L \cup \{l\}$	19 : $(k1, k2) \leftarrow k_j$
3 : $k_j \xleftarrow{\$} \mathcal{K}$	10 : $l_j \leftarrow l$	20 : $(c', t) \leftarrow c$
4 : $C_j \leftarrow \perp$	11 : $(k1, k2) \leftarrow k_j$	21 : $m \leftarrow E.dec(k1, l_j, c')$
5 : $b' \leftarrow A$	12 : $c' \leftarrow E.enc(k1, l_j, m)$	22 : $t' \leftarrow tag(k2, l_j, c')$
6 : return b'	13 : $t \leftarrow tag(k2, l_j, c')$	23 : if $t \neq t'$: $m \leftarrow \perp$
	14 : $c \leftarrow (c', t)$	24 : else :
	15 : $C_j \leftarrow c$	25 : $bad \leftarrow true$
	16 : return c	26 : $m \leftarrow \perp$
		27 : return m

Figure 24: N2-switch-2 game, adversary has access to oracles Oenc and Odec. Key space \mathcal{K}_{mac} is the key space from M.mac. Line 24-26 are different compared to N2-switch-1.

As N2-switch-1 and N2-switch-2 are so called identical-until-*bad* [4], meaning they are equivalent as long as the event *bad* is not set to *true*, we know:

$$\Pr[\text{N2-switch-1}_{A,N} = 0] - \Pr[\text{N2-switch-2}_{A,N} = 0] \leq \Pr[bad = true].$$

As *bad* is set to *true* if, and only if, $t=t'$, we can state $\Pr[bad = true] = \Pr[t = t']$. The adversary needs to provide tag t and ciphertext c' for the *tag* function, where the provided tag-ciphertext pair may not be the result of the encryption query corresponding to the provided user. Consequently, t and t' are only equal when the adversary is able to guess the output of *tag* for a ciphertext that is not encrypted for the provided user. The function *tag* is uniformly random so, with every fresh ciphertext, the probability that t and t' are equal is $\frac{1}{2^n}$. Combined with at most Q_d Odec queries we get $\Pr[t = t'] = \Pr[bad = true] \leq \frac{Q_d}{2^n}$ and thus, we can fill in $\Pr[\text{N2-switch-1}_{A,N} = 0] - \Pr[\text{N2-switch-2}_{A,N} = 0] \leq \Pr[bad = true] \leq \frac{Q_d}{2^n}$ to obtain:

$$\mathbf{Adv}_{A,N}^{\text{IAE}} \leq \mathbf{Adv}_{B,N}^{\text{IMAC}} + \frac{Q_d}{2^n} + \Pr[\text{N2-switch-2}_{A,N} = 0] - \Pr[\text{IAE-N2}_{A,N}^1 = 0].$$

We define game N2-switch-3 in Figure 25 to expand our probability one last time. Switch game 3 is equivalent to switch game 2 but always returns lazily sampled random bits when the outcome of E.enc is valid. It might seem like there is a difference as t can no longer become \perp . This is not the case as, due to the chosen input spaces of tag, tag can never return \perp when the input to E.enc is valid. In other words, tag can only return \perp when c' is already \perp , and thus tag returning \perp will never influence c on line 12. We also simplify Odec as we no longer need the event *bad*. We use this game to expand our probability to:

$$\begin{aligned} \mathbf{Adv}_{A,N}^{\text{IAE}} &\leq \mathbf{Adv}_{B,N}^{\text{IMAC}} + \frac{Q_d}{2^n} + \Pr[\text{N2-switch-2}_{A,N} = 0] - \Pr[\text{N2-switch-3}_{A,N} = 0] \\ &\quad + \Pr[\text{N2-switch-3}_{A,N} = 0] - \Pr[\text{IAE-N2}_{A,N}^1 = 0]. \end{aligned}$$

Game N2-switch-3 _{A,N}	Oracle Oenc(j, l, m)	Oracle Odec(j, c)
0 : $L \leftarrow \emptyset$	7 : if $C_j \neq \perp$: return \perp	19 : return \perp
1 : for $j \in [1..N]$:	8 : if $l \in L$: return \perp	
2 : $k_j \xleftarrow{\$} \mathcal{K}_{enc}$	9 : $L \leftarrow L \cup \{l\}$	
3 : $C_j \leftarrow \perp$	10 : $l_j \leftarrow l$	
4 : $b' \leftarrow A$	11 : $c' \leftarrow \text{E.enc}(k_j, l_j, m)$	
5 : return b'	12 : $t \xleftarrow{\$} \{0, 1\}^n$	
	13 : $c \leftarrow (c', t)$	
	14 : if $c \neq \perp$:	
	15 : $c \xleftarrow{\$} \{0, 1\}^{ c }$	
	16 : $C_j \leftarrow c$	
	17 : return c	

Figure 25: N2-switch-3 game, adversary has access to oracles Oenc and Odec. Key space \mathcal{K}_{enc} is the key space from E.enc. Line 14 and 15 are different compared to N2-switch-2, and Odec is simplified.

Now we can rewrite $\Pr[\text{N2-switch-2}_{A,N} = 0] - \Pr[\text{N2-switch-3}_{A,N} = 0]$ into a IE advantage. To do so, we define adversary C against IE in Figure 26. This adversary is playing game IE-IND- $\$$ -CPA (Figure 9), and has access to A .

Adversary C	if A calls Oracle Oenc(j, l, m)	if A calls Oracle Odec(j, c)
0 : $L \leftarrow \emptyset$	5 : if $C_j \neq \perp$: return \perp	14 : return \perp
1 : for $j \in [1..N]$:	6 : if $l \in L$: return \perp	
2 : $C_j \leftarrow \perp$	7 : $L \leftarrow L \cup \{l\}$	
3 : $b' \leftarrow \text{run } A$	8 : $l_j \leftarrow l$	
4 : return b'	9 : $c' \leftarrow \text{Oenc}(j, l_j, m)$	
	10 : $t \xleftarrow{\$} \{0, 1\}^n$	
	11 : $c \leftarrow (c', t)$	
	12 : $C_j \leftarrow c$	
	13 : return c	

Figure 26: Adversary C , has access to A and oracle Oenc. Note the Oenc in line 9 refers to the encryption oracle Oenc that C has access to, not the oracle Oenc A has access to.

The runtime of C is that of A . For every Oenc query A makes, C makes Q_e one Oenc query. We can see that $\Pr[\text{N2-switch-2}_{A,N} = 0] = \Pr[\text{IE-IND-}\$ \text{-CPA}_{C,N}^0 = 0]$ as C perfectly simulates N2-switch-2 when its own b is 0. In addition, $\Pr[\text{N2-switch-3}_{A,N} = 0] = \Pr[\text{IE-IND-}\$ \text{-CPA}_{C,N}^1 = 0]$

as C perfectly simulates N2-switch-3 when its own b is 1. This gives us:

$$\begin{aligned}
\mathbf{Adv}_{A,N}^{\text{lAE}} &\leq \mathbf{Adv}_{B,N}^{\text{lMAC}} + \frac{Q_d}{2^n} + \Pr[\text{N2-switch-2}_{A,N} = 0] - \Pr[\text{N2-switch-3}_{A,N} = 0] \\
&\quad + \Pr[\text{N2-switch-3}_{A,N} = 0] - \Pr[\text{lAE-N2}^1_{A,N} = 0]. \\
&\leq \mathbf{Adv}_{B,N}^{\text{lMAC}} + \frac{Q_d}{2^n} + \Pr[\text{lE-IND-}\$-\text{CPA}^0_{C,N} = 0] - \Pr[\text{lE-IND-}\$-\text{CPA}^1_{C,N} = 0] \\
&\quad + \Pr[\text{N2-switch-3}_{A,N} = 0] - \Pr[\text{lAE-N2}^1_{A,N} = 0]. \\
&\leq \mathbf{Adv}_{B,N}^{\text{lMAC}} + \frac{Q_d}{2^n} + \mathbf{Adv}_{C,N}^{\text{lE}} \\
&\quad + \Pr[\text{N2-switch-3}_{A,N} = 0] - \Pr[\text{lAE-N2}^1_{A,N} = 0].
\end{aligned}$$

Lastly, $\Pr[\text{N2-switch-3}_{A,N} = 0]$ and $\Pr[\text{lAE-N2}^1_{A,N} = 0]$ are equivalent by definition, giving the result:

$$\mathbf{Adv}_{A,N}^{\text{lAE}} \leq \mathbf{Adv}_{B,N}^{\text{lMAC}} + \frac{Q_d}{2^n} + \mathbf{Adv}_{C,N}^{\text{lE}}.$$

□

12 N3

First, we define our theorem:

Theorem 4. *Let lAE be constructed from lMAC and lE as described in Figure 14. Let ciphertext space \mathcal{C} from the lE be a subset of message space \mathcal{M} from the lMAC and let lMAC and lE have a shared lock space. Then, for any number of users N and any lAE adversary A that poses at most Q_e many Oenc queries, and at most Q_d many Odec queries, there exist a lMAC adversary B and a lE adversary C such that:*

$$\mathbf{Adv}_{A,N}^{\text{lAE}} \leq \mathbf{Adv}_{B,N}^{\text{lMAC}} + \frac{Q_d}{2^n} + \mathbf{Adv}_{C,N}^{\text{lE}},$$

where n is the output length of the lMAC in bits. The running time of B is at most that of A plus the time required to run Q_e many E.enc encapsulations and Q_d many E.dec decapsulations. The running time of C is at most that of A . Additionally, B makes at most Q_e many Omac queries and at most Q_d many Ovrq queries and C makes at most Q_e many Oenc queries.

Within this theorem, both Q_e and Q_d refer to the total queries the adversary is allowed to make, not the queries per user. As a result Q_e is limited by N .

Proof. To prove this theorem we start by defining game lAE-N3 in Figure 27. This game is the game lAE-IND- $\$$ -CCA (Figure 8), with AE.enc and AE.dec substituted with the N3 algorithms from Figure 14.

Game $\text{lAE-N3}_{A,N}^b$	Oracle $\text{Oenc}(j, l, m)$	Oracle $\text{Odec}(j, c)$
0 : $L \leftarrow \emptyset$	6 : if $C_j \neq \perp$: return \perp	18 : if $C_j = \perp$: return \perp
1 : for $j \in [1..N]$:	7 : if $l \in L$: return \perp	19 : if $c = C_j$: return \perp
2 : $k_j \xleftarrow{\$} \mathcal{K}$	8 : $L \leftarrow L \cup \{l\}$	20 : $(k1, k2) \leftarrow k_j$
3 : $C_j \leftarrow \perp$	9 : $l_j \leftarrow l$	21 : $m' \leftarrow \text{E.dec}(k1, l_j, c)$
4 : $b' \leftarrow A$	10 : $(k1, k2) \leftarrow k_j$	22 : $(m, t) \leftarrow m'$
5 : return b'	11 : $t \leftarrow \text{M.mac}(k2, l_j, m)$	23 : $t' \leftarrow \text{M.mac}(k2, l_j, m)$
	12 : $m' \leftarrow m \parallel t$	24 : if $t \neq t' : m \leftarrow \perp$
	13 : $c \leftarrow \text{E.enc}(k1, l_j, m')$	25 : if $b = 1 : m \leftarrow \perp$
	14 : if $b = 1 \wedge c \neq \perp$:	26 : return m
	15 : $c \xleftarrow{\$} \{0, 1\}^{ c }$	
	16 : $C_j \leftarrow c$	
	17 : return c	

Figure 27: lAE-N3 game, adversary has access to oracles Oenc and Odec.

By definition, this gives us

$$\text{Adv}_{A,N}^{\text{lAE}} = \Pr[\text{lAE-N3}_{A,N}^0 = 0] - \Pr[\text{lAE-N3}_{A,N}^1 = 0].$$

Next we define game N3-switch-1 in Figure 28. The only difference between this game and game lAE-N3^0 is the fact that N3-switch-1 uses the uniformly random function tag , instead of the IMAC. To define this function we write $\text{Func}(\mathcal{K} \times \mathcal{L} \times \mathcal{M}, \mathcal{T})$ to denote the set of all functions from the key space of the MAC \mathcal{K} , the shared lock space \mathcal{L} and the message space \mathcal{M} to the tag space \mathcal{T} . We define this function specifically as we want the tags resulting from computations in oracle Oenc to match with those in oracle Odec. When the input of tag is outside its domain, it will return \perp .

Game $\text{N3-switch-1}_{A,N}$	Oracle $\text{Oenc}(j, l, m)$	Oracle $\text{Odec}(j, c)$
0 : $L \leftarrow \emptyset$	7 : if $C_j \neq \perp$: return \perp	17 : if $C_j = \perp$: return \perp
1 : $\text{tag} \xleftarrow{\$} \text{Func}(\mathcal{K}_{\text{mac}} \times \mathcal{L} \times \mathcal{M})$	8 : if $l \in L$: return \perp	18 : if $c = C_j$: return \perp
2 : for $j \in [1..N]$:	9 : $L \leftarrow L \cup \{l\}$	19 : $(k1, k2) \leftarrow k_j$
3 : $k_j \xleftarrow{\$} \mathcal{K}$	10 : $l_j \leftarrow l$	20 : $m' \leftarrow \text{E.dec}(k1, l_j, c)$
4 : $C_j \leftarrow \perp$	11 : $(k1, k2) \leftarrow k_j$	21 : $(m, t) \leftarrow m'$
5 : $b' \leftarrow A$	12 : $t \leftarrow \text{tag}(k2, l_j, m)$	22 : $t' \leftarrow \text{tag}(k2, l_j, m)$
6 : return b'	13 : $m' \leftarrow m \parallel t$	23 : if $t \neq t' : m \leftarrow \perp$
	14 : $c \leftarrow \text{E.enc}(k1, l_j, m')$	24 : return m
	15 : $C_j \leftarrow c$	
	16 : return c	

Figure 28: N3-switch-1, adversary has access to oracles Oenc and Odec. Key space \mathcal{K}_{mac} is the key space from M.mac . Lines 12 and 22 are different compared to lAE-N3^0 , additionally lines 14, 15 and 25 from lAE-N3 are removed.

Using this game, we expand the probability:

$$\begin{aligned} \mathbf{Adv}_{A,N}^{\text{IAE}} &= \Pr[\text{IAE-N3}_{A,N}^0 = 0] - \Pr[\text{N3-switch-1}_{A,N} = 0] \\ &\quad + \Pr[\text{N3-switch-1}_{A,N} = 0] - \Pr[\text{IAE-N3}_{A,N}^1 = 0]. \end{aligned}$$

Next, we can rewrite $\Pr[\text{IAE-N3}_{A,N} = 0] - \Pr[\text{N3-switch-1}_{A,N} = 0]$ into a IMAC advantage. To do so, we define adversary B against IMAC in Figure 29. This adversary is playing game IMAC-PRF (Figure 10), and has access to A .

Adversary B	if A calls Oracle $\text{Oenc}(j, l, m)$	if A calls Oracle $\text{Odec}(j, c)$
0 : $L \leftarrow \emptyset$	6 : if $C_j \neq \perp$: return \perp	15 : if $C_j = \perp$: return \perp
1 : for $j \in [1..N]$:	7 : if $l \in L$: return \perp	16 : if $c = C_j$: return \perp
2 : $k_j \xleftarrow{\$} \mathcal{K}_{\text{enc}}$	8 : $L \leftarrow L \cup \{l\}$	17 : $m' \leftarrow \text{E.dec}(k, l_j, c)$
3 : $C_j \leftarrow \perp$	9 : $l_j \leftarrow l$	18 : $(m, t) \leftarrow m'$
4 : $b' \leftarrow \text{run } A$	10 : $t \leftarrow \text{Omac}(k_j, l_j, m)$	19 : $\text{passed} \leftarrow \text{Ovrf}(j, m, t)$
5 : return b'	11 : $m' \leftarrow m \parallel t$	20 : if $\neg \text{passed}$: $m \leftarrow \perp$
	12 : $c \leftarrow \text{E.enc}(k_j, l_j, m')$	21 : return m
	13 : $C_j \leftarrow c$	
	14 : return c	

Figure 29: Adversary B , has access to A and oracles Omac and Ovrf . Key space \mathcal{K}_{enc} is the key space from E.enc .

The runtime of B is that of A . For every Oenc query A makes, B computes E.enc once, and calls Omac once. For every Odec query A makes, B computes E.dec once and calls Ovrf once. Note that, alternatively, B could return 0 if passed is *true* to avoid having to do E.dec computations. To increase consistency with the other two cases, these computations are still made. We can see that $\Pr[\text{IMAC-PRF}_{B,N}^0 = 0] = \Pr[\text{IAE-N3}_{A,N}^0 = 0]$ as B perfectly simulates game IAE-N3 with $b = 0$ when its own b is 0. In addition, $\Pr[\text{IMAC-PRF}_{B,N}^1 = 0] = \Pr[\text{N3-switch-1}_{A,N} = 0]$ as B perfectly simulates game N3-switch-1 whenever its own b is 1. As a result we can rewrite our advantage to:

$$\begin{aligned} \mathbf{Adv}_{A,N}^{\text{IAE}} &= \Pr[\text{IAE-N3}_{A,N}^0 = 0] - \Pr[\text{N3-switch-1}_{A,N} = 0] \\ &\quad + \Pr[\text{N3-switch-1}_{A,N} = 0] - \Pr[\text{IAE-N3}_{A,N}^1 = 0] \\ &= \Pr[\text{IMAC-PRF}_{B,N}^0 = 0] - \Pr[\text{IMAC-PRF}_{B,N}^1 = 0] \\ &\quad + \Pr[\text{N3-switch-1}_{A,N} = 0] - \Pr[\text{IAE-N3}_{A,N}^1 = 0] \\ &= \mathbf{Adv}_{B,N}^{\text{IMAC}} + \Pr[\text{N3-switch-1}_{A,N} = 0] - \Pr[\text{IAE-N3}_{A,N}^1 = 0]. \end{aligned}$$

To expand our probability again, we define game N3-switch-2 in Figure 30. The Odec oracle from this game always returns \perp , apart from this difference, it is equivalent to the first switch game. Although the Odec always returns \perp , it is written down more elaborately to include the event *bad*, this supports a well-known proof tactic [4]. We use this game to expand our probability:

$$\begin{aligned} \mathbf{Adv}_{A,N}^{\text{IAE}} &= \mathbf{Adv}_{B,N}^{\text{IMAC}} + \Pr[\text{N3-switch-1}_{A,N} = 0] - \Pr[\text{N3-switch-2}_{A,N} = 0] \\ &\quad + \Pr[\text{N3-switch-2}_{A,N} = 0] - \Pr[\text{IAE-N3}_{A,N}^1 = 0]. \end{aligned}$$

Game N3-switch-2 _{A,N}	Oracle Oenc(<i>j, l, m</i>)	Oracle Odec(<i>j, c</i>)
0 : $L \leftarrow \emptyset$	7 : if $C_j \neq \perp$: return \perp	17 : if $C_j = \perp$: return \perp
1 : $tag \xleftarrow{\$} Func(\mathcal{K}_{mac} \times \mathcal{L} \times \mathcal{M})$	8 : if $l \in L$: return \perp	18 : if $c = C_j$: return \perp
2 : for $j \in [1..N]$:	9 : $L \leftarrow L \cup \{l\}$	19 : $(k1, k2) \leftarrow k_j$
3 : $k_j \xleftarrow{\$} \mathcal{K}$	10 : $l_j \leftarrow l$	20 : $m' \leftarrow E.dec(k1, l_j, c)$
4 : $C_j \leftarrow \perp$	11 : $(k1, k2) \leftarrow k_j$	21 : $(m, t) \leftarrow m'$
5 : $b' \leftarrow A$	12 : $t \leftarrow tag(k2, l_j, m)$	22 : $t' \leftarrow tag(k2, l_j, m)$
6 : return b'	13 : $m' \leftarrow m t$	23 : if $t \neq t' : m \leftarrow \perp$
	14 : $c \leftarrow E.enc(k1, l_j, m')$	24 : else :
	15 : $C_j \leftarrow c$	25 : $bad \leftarrow true$
	16 : return c	26 : $m \leftarrow \perp$
		27 : return m

Figure 30: N3-switch-2 game, adversary has access to oracles Oenc and Odec. Key space \mathcal{K}_{mac} is the key space from M.mac. Line 24-26 are different compared to N3-switch-1.

As N3-switch-1 and N3-switch-2 are so called identical-until-*bad* [4], meaning they are equivalent as long as the event *bad* is not set to *true*, we know:

$$\Pr[\text{N3-switch-1}_{A,N} = 0] - \Pr[\text{N3-switch-2}_{A,N} = 0] \leq \Pr[bad = true].$$

As *bad* is set to *true* if, and only if, $t=t'$, we can state $\Pr[bad = true] = \Pr[t = t']$.

The adversary needs to provide a ciphertext c' that leads to a message m that is used as input to the *tag* function and a tag t . The provided ciphertext may not be the result of the encryption query corresponding to the provided user, which also ensures the message-tag pair derived from this ciphertext cannot be the message-tag pair which is encrypted for the provided user. Because the function *tag* is uniformly random and the output need to match with the newly obtained message-tag pair, the probability that t and t' are equal is $\frac{1}{2^n}$ with every fresh Odec query. Combined with at most Q_d Odec queries we get $\Pr[t = t'] = \Pr[bad = true] \leq \frac{Q_d}{2^n}$ and thus, we can fill in $\Pr[\text{N3-switch-1}_{A,N} = 0] - \Pr[\text{N3-switch-2}_{A,N} = 0] \leq \Pr[bad = true] \leq \frac{Q_d}{2^n}$ to obtain:

$$\mathbf{Adv}_{A,N}^{\text{IAE}} \leq \mathbf{Adv}_{B,N}^{\text{IMAC}} + \frac{Q_d}{2^n} + \Pr[\text{N3-switch-2}_{A,N} = 0] - \Pr[\text{IAE-N3}_{A,N}^1 = 0].$$

We define game N3-switch-3 in Figure 31 to expand our probability one last time. Switch game 3 is equivalent to switch game 2 but always returns lazily sampled random bits when the outcome of E.enc is valid. It might seem like there is a difference as t can no longer become \perp . This is not the case as, due to the chosen input spaces of tag, tag can never return \perp when the input to E.enc is valid. In other words, tag can only return \perp when c' is already \perp , and thus tag returning \perp will never influence con line 13. We also simplify Odec as we no longer need the event *bad*. We use this game to expand our probability to:

$$\begin{aligned} \mathbf{Adv}_{A,N}^{\text{IAE}} &\leq \mathbf{Adv}_{B,N}^{\text{IMAC}} + \frac{Q_d}{2^n} + \Pr[\text{N3-switch-2}_{A,N} = 0] - \Pr[\text{N3-switch-3}_{A,N} = 0] \\ &\quad + \Pr[\text{N3-switch-3}_{A,N} = 0] - \Pr[\text{IAE-N3}_{A,N}^1 = 0]. \end{aligned}$$

Game N3-switch-3 _{A,N}	Oracle Oenc(j, l, m)	Oracle Odec(j, c)
0 : $L \leftarrow \emptyset$	7 : if $C_j \neq \perp$: return \perp	19 : return \perp
1 : for $j \in [1..N]$:	8 : if $l \in L$: return \perp	
2 : $k_j \xleftarrow{\$} \mathcal{K}_{enc}$	9 : $L \leftarrow L \cup \{l\}$	
3 : $C_j \leftarrow \perp$	10 : $l_j \leftarrow l$	
4 : $b' \leftarrow A$	11 : $t \xleftarrow{\$} \{0, 1\}^n$	
5 : return b'	12 : $m' \leftarrow m \ t$	
	13 : $c \leftarrow \text{E.enc}(k_j, l, m')$	
	14 : if $c \neq \perp$:	
	15 : $c \xleftarrow{\$} \{0, 1\}^{ c }$	
	16 : $C_j \leftarrow c$	
	17 : return c	

Figure 31: N3-switch-3 game, adversary has access to oracles Oenc and Odec. Key space \mathcal{K}_{enc} is the key space from E.enc. Line 14 and 15 are different compared to N3-switch-2, and Odec is simplified.

Now we can rewrite $\Pr[\text{N3-switch-2}_{A,N} = 0] - \Pr[\text{N3-switch-3}_{A,N} = 0]$ into a IE advantage. To do so, we define adversary C against IE in Figure 32. This adversary is playing game IE-IND- $\$$ -CPA (Figure 9), and has access to A .

Adversary C	if A calls Oracle Oenc(j, l, m)	if A calls Oracle Odec(j, c)
0 : $L \leftarrow \emptyset$	5 : if $C_j \neq \perp$: return \perp	14 : return \perp
1 : for $j \in [1..N]$:	6 : if $l \in L$: return \perp	
2 : $C_j \leftarrow \perp$	7 : $L \leftarrow L \cup \{l\}$	
3 : $b' \leftarrow \text{run } A$	8 : $l_j \leftarrow l$	
4 : return b'	9 : $t \xleftarrow{\$} \{0, 1\}^n$	
	10 : $m' \leftarrow m \ t$	
	11 : $c \leftarrow \text{Oenc}(j, l, m')$	
	12 : $C_j \leftarrow c$	
	13 : return c	

Figure 32: Adversary C , has access to A and oracle Oenc. Note the Oenc in line 11 refers to the encryption oracle Oenc that C has access to, not the oracle Oenc A has access to.

The runtime of C is that of A . For every Oenc query A makes, C makes Q_e one Oenc query. We can see that $\Pr[\text{N3-switch-2}_{A,N} = 0] = \Pr[\text{IE-IND-}\$ \text{-CPA}_{C,N}^0 = 0]$ as C perfectly simulates N3-switch-2 when its own b is 0. In addition, $\Pr[\text{N3-switch-3}_{A,N} = 0] = \Pr[\text{IE-IND-}\$ \text{-CPA}_{C,N}^1 = 0]$

as C perfectly simulates N3-switch-3 when its own b is 1. This gives us:

$$\begin{aligned}
\mathbf{Adv}_{A,N}^{\text{IAE}} &\leq \mathbf{Adv}_{B,N}^{\text{IMAC}} + \frac{Q_d}{2^n} + \Pr[\text{N3-switch-2}_{A,N} = 0] - \Pr[\text{N3-switch-3}_{A,N} = 0] \\
&\quad + \Pr[\text{N3-switch-3}_{A,N} = 0] - \Pr[\text{IAE-N3}_{A,N}^1 = 0]. \\
&\leq \mathbf{Adv}_{B,N}^{\text{IMAC}} + \frac{Q_d}{2^n} + \Pr[\text{IE-IND-}\$-\text{CPA}_{C,N}^0 = 0] - \Pr[\text{IE-IND-}\$-\text{CPA}_{C,N}^1 = 0] \\
&\quad + \Pr[\text{N3-switch-3}_{A,N} = 0] - \Pr[\text{IAE-N3}_{A,N}^1 = 0]. \\
&\leq \mathbf{Adv}_{B,N}^{\text{IMAC}} + \frac{Q_d}{2^n} + \mathbf{Adv}_{C,N}^{\text{IE}} \\
&\quad + \Pr[\text{N3-switch-3}_{A,N} = 0] - \Pr[\text{IAE-N3}_{A,N}^1 = 0].
\end{aligned}$$

Lastly, $\Pr[\text{N3-switch-3}_{A,N} = 0]$ and $\Pr[\text{IAE-N3}_{A,N}^1 = 0]$ are equivalent by definition, giving the result:

$$\mathbf{Adv}_{A,N}^{\text{IAE}} \leq \mathbf{Adv}_{B,N}^{\text{IMAC}} + \frac{Q_d}{2^n} + \mathbf{Adv}_{C,N}^{\text{IE}} .$$

□

Appendix B

Below is a short version of the story I wanna tell in this thesis, and some todos. These may both be ignored for now as they will be removed later on anyway.

Story The story I would like to tell starts with locks as a augmentation in a multi-user setting. GKP introduces locks as a augmentation to a AE scheme in a hybrid encryption context as an alternative to longer keys. Locks might be preferred in due to technical constraints (lack in computing power or fixed key size), longer keys are not feasible. The AE composition that is considered in GKP follows the generic Encrypt-then-MAC composition that originated in a symmetric crypto setting. This generic Encrypt-then-MAC composition has been reconsidered in a symmetric crypto setting by NRS. In my thesis, I reconsider the lock-based AE in a symmetric crypto setting using the knowledge from NRS. NRS gives me a good example of how a generic AE composition should be considered in a symmetric setting. The result will be lock-based, one time use AE primitive, considered in a symmetric setting. This primitive will be useful for hybrid encryption, possibly it has some more use cases.

Todo I put the todos in a list and explain them shortly when needed

- security proofs of N1 and N3 (only after N2 is done)
- learning more about hybrid encryption
- looking for possible other use cases
- compare the LAE with existing alternatives
- look one time pad + information theoretical mac component (I think you made a note about this a few meetings back)

papers I put the potential papers in a list and explain them shortly when needed. I am not sure which of these I need or which actually exist, it is just a list I wrote down as a starting point.

- paper on why $\text{ind\$}$ implies ind lor
- paper with other lock implementations (will prob cite gkp)
- papers for other use cases (if I want to include that)
- papers for other alternatives (if I want to include that)
- paper on the information theoretical mac component (if I want to include that)