

BACHELOR THESIS



RADBOD UNIVERSITY

TBD

Author:
Stijn Vandenput

Supervisors:
Martijn Stam
Bart Mennink

20/05/2022

Abstract

Contents

1	Introduction	1
2	Preliminaries	1
2.1	General Notation	1
2.2	AE	1
2.3	PKE Schemes?	1
2.4	Nonces vs Locks (and different iv's)	1
2.5	Game Based Security Notions	1
2.6	Security Notions	1
2.7	Security Proofs	1
3	NRS and GKP in Detail	1
3.1	NRS	2
3.1.1	Used Primitives	2
3.1.2	Nonce-Based Authenticated Encryption	3
3.1.3	Construction	3
3.2	GKP	4
3.2.1	Used Primitives	4
3.2.2	ADEM'	5
3.2.3	Construction	5
3.3	Comparison of GKP and NRS	6
4	Defining the New Primitive	7
4.1	loAE	7
4.2	Security Model	7
4.3	Explanation of the Security Model	8
5	Constructions	8
5.1	Used Primitives	8
5.2	Construction	9
5.3	Security Bounds	11
5.4	Comparison with Existing Alternatives	11
6	Use Cases	11
6.1	PKE Schemes	11
7	Related Work	11
8	Conclusion	11
	References	12
9	Appendix A	12

1 Introduction

Although symmetric and asymmetric cryptography are both subfields of cryptography, their research area's can be quite separated. This can lead to knowledge gaps between the two when work in asymmetric crypto uses constructions that are more common in symmetric crypto or the other way around. In this fashion, a paper by Giacon, Kiltz and Poettering [1], which we henceforth call GKP, uses a construction that is very similar to Authenticated Encryption following the generic encrypt-then-MAC construction from Bellare and Namprempre [2]. This construction has since been revised in a paper by Namprempre, Rogaway and Thomas Shrimpton [3], which we henceforth call NRS. In this revision, a new set of constructions is given that be better applicable to common use cases. The aim of this thesis is to apply the knowledge from NRS to the setting of GKP and while doing so, create a new primitive for authenticated encryption suited for asymmetric settings.

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 random sampling from y and assigning it to x as $x \xleftarrow{\$} y$. We allow a single type of error message written as \perp . \mathcal{K} is a nonempty key space, \mathcal{L} is a lock space, \mathcal{N} is a nonce space, \mathcal{M} is a message space and \mathcal{A} is the associated-data space. \mathcal{M} contain at least two strings, and if \mathcal{M} or \mathcal{A} contain a string of length x , it must contain all strings of length x . N is the number of users.

2.2 AE

2.3 PKE Schemes?

2.4 Nonces vs Locks (and different iv's)

2.5 Game Based Security Notions

2.6 Security Notions

ind-\$/ind-lor/ind-cpa/ind-cca also note active and passive attackers

2.7 Security Proofs

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 papers. Some notations will be different from the original papers for improved consistency. What are called tags in 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

Game nE-IND_A^b	Oracle $\text{Oenc}(n, m)$
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 : $c \leftarrow \text{E}(k, n, m)$
3 : return b'	8 : if $b = 1 \wedge c \neq \perp$:
	9 : $c \xleftarrow{\$} \{0, 1\}^{ c }$
	10 : return c

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

Authenticated Encryption: How Subtle AEAD) in order to better match the notation from GKP and be more adaptable to a multi-user setting.

3.1 NRS

Three generic ways to construct a 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 one when using probabilistic encryption as a building block. Within NRS these constructions are generalized to using nonce- or iv-based encryption as a building block to create nonce-based authenticated encryption schemes, nAEs for short. We will look at the constructions using a nonce-based encryption, nE for short, and a PRF secure MAC function.

3.1.1 Used Primitives

nE A nonce-based encryption scheme is defined by triple $\Pi = (\mathcal{K}, \text{E}, \text{D})$. 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 $\text{E}(k, n, m) = c \neq \perp$, then $\text{D}(k, n, c) = m$) and tidiness (if $\text{D}(k, n, c) = m \neq \perp$, then $\text{E}(k, n, m) = c$).

Security The security of a nE is defined as $\text{Adv}_{\Pi, A}^{\text{nE}} = |\Pr[\text{nE-IND}_A^0 = 1] - \Pr[\text{nE-IND}_A^1 = 1]|$, where nE-IND\$ is in figure 1. In this game, set U keeps track of all used nonces as the adversary is not allowed to repeat those.

MAC A MAC scheme is defined by 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 $\text{F}(k, m) \neq \perp$ is in X , this domain may not depend on k .

Security The security of a MAC is defined as $\text{Adv}_{\text{F}, A}^{\text{MAC}} = |\Pr[\text{MAC-PRF}_A^0 = 1] - \Pr[\text{MAC-PRF}_A^1 = 1]|$, where MAC-PRF is in figure 2. In this game the set U keeps track of the used messages to prevent trivial wins.

Game MAC-PRF $_A^b$	Oracle Omac(m)
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 $_A^b$	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\$ game, A has access to oracles Oenc and Odec.

3.1.2 Nonce-Based Authenticated Encryption

A nonce-based authenticated encryption scheme is defined by 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 $E(k, n, a, m) = c$).

Security The security of a nAE is defined as $\text{Adv}_{\Pi, A}^{\text{nAE}} = |\Pr[\text{nAE-IND}_A^0 = 1] - \Pr[\text{nAE-IND}_A^1 = 1]|$, where nAE-IND\$ is in figure 3. In this game, set U keeps track of all used nonces as the adversary is not allowed to repeat those. Following the translation of IND\$ 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 tack of all query results in order to prevent trivial wins.

3.1.3 Construction

A nAE scheme is constructed by several different schemes that combine the mac and nE into a nAE. We define the constructions secure as 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

Game L-IND-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-CPA game, A has access to oracle Oenc.

be viewed in figure 6 of NRS. Noteworthy is that these relate to encrypt-and-mac, encrypt-then-mac, and mac-then-encrypt respectively, showing the general believe that encrypt-then-mac is the only safe construction does not transfer to this setting.

3.2 GKP

In GKP, the concept of augmentation using locks is discussed. The authors start by showing 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 a ADEM that is secure against passive attacks can be combined with a MAC that is augmented in a similar fashion, called a AMAC, to construct ADEM' that is safe against active attackers. This construction is similar to construction N2 from NRS.

3.2.1 Used Primitives

In GKP, \mathcal{M} is not required 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.

ADEM A ADEM scheme is defined by 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 $A.\text{dec}(k, l, A.\text{enc}(k, l, m)) = m$.

Security The security of a ADEM is defined as $\text{Adv}_{\text{ADEM}, A, N}^{\text{l-ind-cpa}} = |\Pr[\text{L-IND-CPA}_{A, N}^0 = 1] - \Pr[\text{L-IND-CPA}_{A, N}^1 = 1]|$, where L-IND-CPA is in figure 4. Every user is only allowed one encryption query and locks may not repeat between users. Decryption queries are only allowed after the encryption. The corresponding game can be found in figure 9 from GKP, note that this figure also included a decryption oracle the adversary is not allowed to use. (**Question: I do not really know if I should elaborate on how this translates to the game, as the game is based on the game of gkp**)

Game L-MIOT-UF _{A,N}	Oracle Omac(j, l, m)	Oracle Ovr(j, m, t)
0 : 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.vrf(k_j, l_j, m, t) :
3 : $k_j \xleftarrow{\$} \mathcal{K}$	10 : $l_j \leftarrow l$	17 : forged $\leftarrow 1$
4 : $T_j \leftarrow \emptyset$	11 : $t \leftarrow \text{M.mac}(k_j, l_j, m)$	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 5: L-MIOT-UF game, A has access to oracles Omac and Ovr and the locks in line 11 and 16 are the same.

AMAC A AMAC scheme is defined by tuple $(\text{M.mac}, \text{M.vrf})$. Deterministic algorithm M.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} . Deterministic algorithm M.vrf takes a key k in \mathcal{K} , a lock l in \mathcal{L} , a message m in \mathcal{M} and a ciphertext 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 $t \leftarrow \text{M.mac}(k, l, m)$ gives $\text{M.vrf}(k, l, m, t) = \text{true}$.

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 5. Every user is only allowed one mac query and locks may not repeat between users. Verification queries are only allowed after the encryption. The corresponding game can be found in figure 15 of GKP.

3.2.2 ADEM'

A ADEM' scheme is defined by tuple $(\text{A.enc}', \text{A.dec}')$. Deterministic algorithm $\text{A.enc}'$ 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 $\text{A.dec}'$ 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 $\text{A.dec}'(k, l, \text{A.enc}'(k, l, m)) = m$.

Security The security of a ADEM' is defined as $\text{Adv}_{\text{ADEM}', A, N}^{\text{l-ind-cca}} = |\Pr[\text{L-IND-CCA}_{A, N}^0 = 1] - \Pr[\text{L-IND-CCA}_{A, N}^1 = 1]|$, where L-IND-CCA is in 6. Every user is only allowed one encryption query and locks may not repeat between users. Decryption queries are only allowed after the encryption. The corresponding game can be found in figure 9 of GKP.

3.2.3 Construction

The ADEM' scheme considered is made by creating $\text{A.enc}'$ and $\text{A.dec}'$ calls using the calls the primitives provide us as seen in figure 7. The construction is deemed secure as for any N and a A that makes Q_d many Odec queries, the exist B and C such that $\text{Adv}_{\text{ADEM}', A, N}^{\text{l-ind-cca}} \leq 2\text{Adv}_{\text{AMAC}, B, N}^{\text{l-miot-uf}} + \text{Adv}_{\text{ADAM}, C, N}^{\text{l-ind-cpa}}$ holds. Where the running time of B is at most that of A plus the time required to run N -many ADEM encapsulations and Q_d -many ADEM decapsulations and the running time of C is the same as the running time of A . Additionally, B poses at most Q_d -many Ovr queries, and C poses no Odec query.

Game L-IND-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 6: L-IND-CCA game, A has access to oracles Oenc and Odec and the locks in line 10 and 15 are the same.

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 : $c' \leftarrow A.\text{enc}(k_{dem}, l, m)$	6 : $(c', t) \leftarrow c$
2 : $t \leftarrow M.\text{mac}(k_{mac}, l, c')$	7 : if $M.\text{vrf}(k_{mac}, l, c', t)$:
3 : $c \leftarrow (c', t)$	8 : $m \leftarrow A.\text{dec}(k_{dem}, l, c')$
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.

3.3 Comparison of GKP and NRS

In this section we will highlight the important differences between GKP and NRS. (**Question: I feel like this section is very factual right now, while it might also be worthwhile to elaborate on some things a bit more.**)

Setting NRS is written is single-user, multiple-use key setting while GKP is written is a multi-user, single-use key setting. As a result, GKP uses locks while NRS uses nonces. (**Question: is it necessary to elaborate on how these settings lead to lock, or nonce usage, or would it suffice to explain this in section 2.4.**)

Aim While NRS is aimed at generalizing the AE constructions, GKP is aimed aimed at finding a single construction that is safe when used in public key encryption. Most notably, this results in NRS evaluating 20 possible constructions while GKP evaluates one. Additionally, the constructions from NRS are able to use AD while the construction form GKP cannot.

Security Notion The security notions of NRS are written in a IND\$ fashion while the security notions of GKP are written in a lor fashion. In other words, NRS requires the valid ciphertext to be indistinguishable from random strings, GKP only requires them to be indistinguishable from each other. As a result, the MAC primitives of the two papers have different security requirements. In NRS, the tag is required to be indistinguishable from a random string while in GKP the tag is only required to be unforgeable.

Game $\text{loAE-IND\$}_{A,N}^b$	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 $b = 1$: $m = \perp$
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: loAE-IND\$ game, adversary has access to oracles Oenc and Odec.

4 Defining the New Primitive

In this section we will discuss a new security primitive, the lock-based one time use Authenticated Encryption scheme, loAE scheme for short. As the name suggests, this primitive is used in a setting where a key is used only once to encrypt and authenticate a single message. We use locks instead of nonces, as you will never have to decrypt messages with multiple nonces for a single user. Below, we discuss the notation of the loAE.

4.1 loAE

A loAE scheme is defined by tuple $(\text{AE.enc}, \text{AE.dec})$. Deterministic algorithm AE.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 AE.dec takes three inputs (k, n, 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 Security Model

To define the security, we use a ind\$ security notion instead of left-or-right one as, in our setting, ind\$ is the stronger security notion. (**Question: is it necessary to elaborate on how this here, or would it suffice to explain this in section 2.6**) We use a function that always returns \perp on decryption calls to ensure the adversary can not guess which ciphertexts would be valid ciphertexts. To match our setting, the user is only allowed one encryption query, locks may not repeat between users and decryption queries are only allowed after the encryption. The security is defined as $\text{Adv}_{A,N}^{\text{loAE}} = |\Pr[\text{loAE-IND\$}_{A,N}^0 = 1] - \Pr[\text{loAE-IND\$}_{A,N}^1 = 1]|$, where loAE-IND\$ is in figure 3.

4.3 Explanation of the Security Model

In this subsection we will discuss the security game loAE-IND\$ line by line.

Draft:

line 0: We start by initializing the set of all used locks to the empty set. *line 1:* Next we loop over each user to initialize them. *line 2:* For each user, we sample a uniformly random key from the key space and assign it to that user-key. *line 3:* for each user, we initialize the user-ciphertext to be undefined. We do not use set notation for this as done in GKP, as we can never have multiple ciphertexts related to one user. *line 4:* After initializing all users, we run the algorithm of the adversary and save the output. *line 5:* We end by returning this output. *line 6:* We do not allow multiple encryptions per user. Therefore, if the ciphertexts is defined we return \perp . *line 7:* Locks are not allowed to repeat, if the lock is in the set of used sets we return \perp . *line 8:* If these checks pass, we start the encryption by adding the lock to the sets of used locks *line 9:* We bind the lock to the user-lock. *line 10:* We encrypt the message using the user-key and the user-lock. *line 11:* The random function should return \perp whenever AE returns \perp . Therefore the random function is only called if $b = 1$ and AE does not return \perp . *line 12:* The random function samples a string with the uniformly at random from the set of all strings with the length of the ciphertext. **(todo: add part about ideal vs attainable)** *line 13:* The computed ciphertext is bound to the user-ciphertext. **(todo: add part about the fact that this might be \perp)** *line 14:* Return the ciphertext. *line 15:* If the user-ciphertext is not defined, decryption is not allowed and we return \perp . *line 16:* If the input ciphertext of Odec is allowed to be the same as the user-ciphertext, then it is trivial to distinguish as in this case the ideal words will return \perp while the real world does not. For this reason the real world should return \perp as well. *line 17:* If these checks pass, we decrypt the ciphertext. *line 18:* The ideal world always returns \perp . *line 19:* We return the decrypted message.

5 Constructions

In this section we discuss how we can construct a safe loAE. Similarly to GKP and NRS we will look at constructions combining a deterministic encryption primitive and mac primitive. First 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

loE A lock-based one time use encryption scheme, loE for short, is defined by tuple $(E.\text{enc}, E.\text{dec})$. Deterministic algorithm $E.\text{enc}$ 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 $E.\text{dec}$ takes three inputs (k, l, c) and outputs a value m . Both $E.\text{enc}$ and $E.\text{dec}$ are required to satisfy correctness (if $E.\text{enc}(k, l, m) = c \neq \perp$, then $E.\text{dec}(k, l, c) = m$) and tidiness (if $E.\text{dec}(k, l, c) = m \neq \perp$, then $E.\text{enc}(k, l, m) = c$).

Security The security of a loE is defined as $\text{Adv}_{A,N}^{\text{loE}} = |\Pr[\text{loE-IND}_{A,N}^0 = 1] - \Pr[\text{loE-IND}_{A,N}^1 = 1]|$, where loE-IND\$ is in figure 9. The user is only allowed one encryption query and locks may not repeat between users. Decryption queries are only allowed after the encryption.

Game loE-IND $_{A,N}^b$	Oracle 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: loE-IND\$ game, A has access to oracle Oenc.

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

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

loMAC A lock-based one time use MAC is 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 .

Security The security of a PRF-MAC is defined as $\text{Adv}_{F,A,N}^{\text{loMAC}} = |\Pr[\text{loMAC-PRF}_{A,N}^0 = 1] - \Pr[\text{loMAC-PRF}_{A,N}^1 = 1]|$, where loMAC-PRF is in figure 10. The user is only allowed one mac query and locks may not repeat between users. Verification queries are only allowed after the encryption.

5.2 Construction

Following NRS, three ways to construct this loAE 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 no one 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.

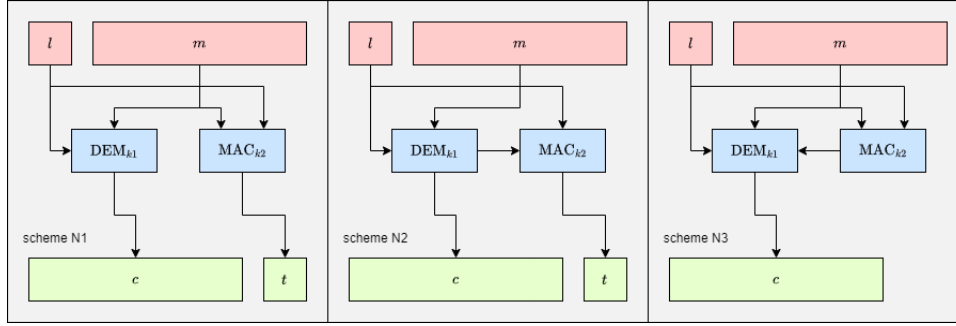


Figure 11: Adjusted N schemes from NRS

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

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

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

Figure 14: Calls based on N3

5.3 Security Bounds

We define the constructions secure if there is a tight reduction from breaking the loAE-security of the scheme to breaking the loE-security and the loMAC security of the underlying primitives.

5.4 Comparison with Existing Alternatives

6 Use Cases

should consist of:

- possible use cases

6.1 PKE Schemes

7 Related Work

Location not final yet

8 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).

9 Appendix A

(**todo: elaborate more on this table**)

Below is a table which highlights the differences in notation between GKP and NRS, as well as give the notation I will be using.

Name	GKP	NRS	my notation	rough meaning
message	m	M	m	message the user sends
ciphertext space	\mathcal{C}	-	\mathcal{C}	set of all possible ciphertext options
ciphertext	c	C	c	encrypted message
associated data	-	A	a	data you want to authenticate but not encrypt
tag space	\mathcal{C}	-	\mathcal{T}	set off all possible tag options
tag	c	T	t	output of mac function
key	k	K	k	user key
nonce space	-	\mathcal{N}	\mathcal{N}	set of all nonce options
nonce	-	n	n	number only used once
lock space	\mathcal{T}	-	\mathcal{L}	set of all possible lock options
lock	t	-	l	nonce that is bound to the user
adversary	A	\mathcal{A}	A	the bad guy
random sampling	$\overset{\$}{\leftarrow}$	\leftarrow	$\overset{\$}{\leftarrow}$	get a random element from the set
result of randomized function	$\overset{\$}{\leftarrow}$	-	\leftarrow	get the result of a randomized function with given inputs