

BACHELOR THESIS



RADBOD HONOURS ACADEMY

TBD

Author:
Stijn Vandenput

Supervisors:
Martijn Stam
Bart Mennink

20/05/2022

Abstract

Contents

1	Introduction	1
2	Preliminaries	1
2.1	notation table	1
3	Existing AE/DEM notations in more detail	2
3.1	Existing notation from GKP	2
3.1.1	notation	3
3.1.2	used primitives	3
3.1.3	goal	4
3.1.4	Security model	4
3.1.5	construction	4
3.2	Existing notation from NRS	5
3.2.1	notation	5
3.2.2	used primitives	5
3.2.3	goal	6
3.2.4	Security model	6
3.2.5	construction	7
4	New Definition	7
4.1	notation	7
4.2	goal	7
4.3	Security model	8
5	Constructions	8
5.1	used primitives	8
5.2	construction	10
6	Use cases	12
7	Related Work	12
8	Conclusion	12
	References	13
9	Appendix	13

1 Introduction

should consist of:

- explaining the challenge
- my contribution

Within the field of cryptography there are two different fields, symmetric and asymmetric crypto. The former is about situations where the users have a shared secret already, the latter is about situations where this is not the case. Sometimes work in asymmetric crypto uses constructions that are more common in symmetric crypto, and in this fashion, a paper by Gligou, 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, to be better applicable to common use cases. The aim of this thesis is to apply the knowledge of NRS to the setting of GKP and in doing so, create a primitive for authenticated encryption in an asymmetric crypto setting.

2 Preliminaries

should consist of:

- general notation
- AE
- PKE schemes?
- nonces vs IVs (and different IVs)
- game based security notions
- ind-CPA/ind-CPA/ind-CCA

2.1 notation 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 space	\mathcal{M}	\mathcal{M}	\mathcal{M}	set of all possible message options
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 space	-	\mathcal{A}	\mathcal{A}	set of all possible associated data options
associated data	-	A	a	data you want to authenticate but not encrypt
tag space	\mathcal{C}	-	\mathcal{T}	set of all possible tag options
tag	c	T	t	output of mac function
key space	\mathcal{K}	\mathcal{K}	\mathcal{K}	set of all possible key options
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
users	N	-	N	number of users
adversary	A	\mathcal{A}	A	the bad guy
random sampling	$\leftarrow^{\$}$	\leftarrow	$\leftarrow^{\$}$	get a random element from the set
result of randomised function	$\leftarrow^{\$}$	-	\leftarrow	get the result of a randomised function with given inputs
result of function	\leftarrow	\leftarrow	\leftarrow	get the result of a function with given inputs
concatination	$a b$	$a b$	$a b$	concatanation of two strings
true	<i>true</i>	-	<i>true</i>	boolean true
false	<i>false</i>	-	<i>false</i>	boolean false
invalid	\perp	\perp	\perp	operation is invalid

strings are binary and bit-wise

3 Existing AE/DEM notations in more detail

should consist of:

- the definition of the two paper, if possible already brought more toward one notation standard
- explain how both relate to the preliminaries and overall story

3.1 Existing notation from GKP

Below are the security notations from GKP that are relevant for this thesis. Some of the variable names are rewritten towards a common notation with NRS.

3.1.1 notation

\mathcal{M} is a message space, \mathcal{K} is a finite key space, \mathcal{L} is a lock space and \mathcal{C} is a ciphertext space. N is the number of users.

3.1.2 used primitives

- ADEM: the ADEM exists of tuple $(A.\text{enc}, A.\text{dec})$, $A.\text{enc}$ is a deterministic algorithm that 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} . $A.\text{dec}$ is a deterministic algorithm that 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 $A.\text{dec}(k, l, A.\text{enc}(k, l, m)) = m$. The security of the ADEM is defined with $\text{Adv}_{\text{ADEM}, A, N}^{\text{l-miot-ind}} = |\Pr[\text{L-IND-CPA}_{A, N}^0 = 1] - \Pr[\text{L-IND-CPA}_{A, N}^1 = 1]|$, defined by the following game:

Game $\text{L-IND-CPA}_{A, N}^b$	Oracle $\text{Oenc}(j, l, m_0, m_1)$
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 : $k_j \leftarrow^{\$} \mathcal{K}$	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 1: L-IND-CPA game, A has access to oracle Oenc . The corresponding game can be found in GKP figure 9 (note that this has a decryption oracle the ADEM is not allowed to use).

- AMAC: the AMAC exists of tuple $(M.\text{mac}, M.\text{vrf})$. $M.\text{mac}$ is a deterministic algorithm that takes a key k in \mathcal{K} , a lock l in \mathcal{L} , and a message m in \mathcal{M} and outputs a ciphertext t in \mathcal{T} . $M.\text{vrf}$ 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 $t \leftarrow M.\text{mac}(k, l, m)$ gives $M.\text{vrf}(k, l, m, t) = \text{true}$. The security of the AMAC is defined with $\text{Adv}_{\text{AMAC}, A, N}^{\text{L-MIOT-UF}} = \Pr[\text{L-MIOT-UF}_{A, N} = 1]$, defined by the following game:

Game L-MIOT-UF _{A,N}	Oracle Omac(j, l, m)	Oracle Ovr(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 : $k_j \leftarrow^{\$} \mathcal{K}$	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 2: L-MIOT-UF game, A has access to oracles Omac and Ovr and the locks in line 11 and 16 are the same. The corresponding game can be found in GKP figure 15.

3.1.3 goal

The goal is to create a ADEM' the ADEM' exists of tuple $(A.\text{enc}', A.\text{dec}')$, $A.\text{enc}'$ is a deterministic algorithm that 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} . $A.\text{dec}'$ is a deterministic algorithm that 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 $A.\text{dec}'(k, l, A.\text{enc}'(k, l, m)) = m$.

3.1.4 Security model

The security of the ADEM' is defined with $\text{Adv}_{\text{ADEM}', A, N}^{\text{l-miot-ind}} = |\Pr[\text{L-IND-CCA}_{A, N}^0 = 1] - \Pr[\text{L-IND-CCA}_{A, N}^1 = 1]|$, defined by the following game:

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 : $k_j \leftarrow^{\$} \mathcal{K}$	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 3: L-IND-CCA game, A has access to oracles Oenc and Odec and the locks in line 10 and 15 are the same. The corresponding game can be found in GKP figure 9.

3.1.5 construction

The goal is met by creating $A.\text{enc}'$ and $A.\text{dec}'$ calls which are build using the calls the primitives provide us:

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 4: A.enc' and A.dec' calls, The corresponding calls can be found in GKP figure 16.

The construction is deemed secure as for any N and a A that makes Q_d many Odec queries, there exist B and C such that $\mathbf{Adv}_{ADEM', A, N}^{l\text{-miot-ind}} \leq 2\mathbf{Adv}_{AMAC, B, N}^{l\text{-miot-uf}} + \mathbf{Adv}_{ADAM, C, N}^{l\text{-miot-ind}}$ 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 OvrF queries, and C poses no Odec query.

3.2 Existing notation from NRS

Below are the security notations from NRS that are relevant for this thesis. Some of the variable names are rewritten towards a common notation with GKP.

3.2.1 notation

\mathcal{K} is a nonempty key space, \mathcal{N} is a non-empty nonce space, \mathcal{M} is a message space and \mathcal{A} is the associated-data space. \mathcal{M} contains at least two strings, and if \mathcal{M} and \mathcal{A} contain a string of length x , they must contain all strings of length x .

3.2.2 used primitives

- **nE**: A nonce-based encryption scheme is defined by triple $\Pi = (\mathcal{K}, E, D)$. E is a deterministic encryption algorithm that takes three inputs (k, n, m) to a value c , the length of c only depends on 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 . D is the decryption algorithm that takes three inputs (k, n, c) to a value m . E and D are inverse of each other implying 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 $E(k, n, m) = c$). The security is defined as $\mathbf{Adv}_{\Pi, A}^{nE} = |\Pr[\text{nE-IND}_A^0 = 1] - \Pr[\text{nE-IND}_A^1 = 1]|$, where nE-IND\$ is defined as follows:

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

Figure 5: nE-IND\$ game, A has access to oracle Oenc and U is the set of used nonces

- MAC: The MAC is a deterministic algorithm F that takes in a k in \mathcal{K} and a string m and outputs either a n -bit length t or \perp . The domain of F is the set X such that $F(k, m) \neq \perp$. This domain may not depend on k . The security is defined as $\text{Adv}_{F,A}^{\text{MAC}} = |\Pr[\text{MAC-PRF}_A^0 = 1] - \Pr[\text{MAC-PRF}_A^1 = 1]|$, where MAC-PRF is defined as follows:

Game MAC-PRF_A^b	Oracle $\text{Omac}(m)$
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 6: MAC-PRF, A has access to oracle Omac and U is the set of used messages

3.2.3 goal

The goal is a nonce-based authenticated encryption scheme defined by triple $\Pi = (\mathcal{K}, E, D)$. E is a deterministic encryption algorithm that takes four inputs (k, n, a, m) to 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 . D is the decryption algorithm that takes four inputs (k, n, a, c) to a value m . E and D are inverse of each other implying 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 $E(k, n, a, m) = c$)

3.2.4 Security model

The security 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 defined as follows:

Game nAE-IND_A^b	Oracle $\text{Oenc}(n, a, m)$	Oracle $\text{Odec}(n, a, c)$
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 : $m \leftarrow \text{D}(k, n, a, c)$
3 : $b' \leftarrow A$	9 : $c \leftarrow \text{E}(k, n, a, m)$	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 7: nAE-IND\$ game, A has access to oracles Oenc and Odec , U is the set of used nonces and Q is the set of query results. $_$ denotes a variable that is irrelevant

3.2.5 construction

The goal is met by creating 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 the can be viewed in NRS figure 6.

4 New Definition

should consist of:

- syntax of the primitive (input, output, correctness, tidiness, expected bounds)
- game based code
- explanation of the choices made
- formal comparison with other choices

4.1 notation

\mathcal{K} is a non-empty key space, \mathcal{L} is a non-empty lock space and \mathcal{M} is a message space. \mathcal{M} contains at least two strings, and if it contains a string of length x , it must contain all strings of length x . N is the number of users.

4.2 goal

The end goal is to build a lock-based one time use Authenticated Encryption scheme (loAE). The AE scheme is defined by tuple $(\text{AE.enc}, \text{AE.dec})$. AE.enc is a deterministic encryption algorithm that takes three inputs (k, l, m) to 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 . AE.dec is the decryption algorithm that takes three inputs (k, n, c) to a value m . AE.enc and EA.dec are required to have 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 $\text{AE.enc}(k, l, m) = c$).

4.3 Security model

The security is defined as $\mathbf{Adv}_{A,N}^{\text{loAE}} = |\Pr[\text{loAE-IND\$}_{A,N}^0 = 1] - \Pr[\text{loAE-IND\$}_{A,N}^1 = 1]|$, where loAE-IND\$ is defined as follows:

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 $b = 1$: return \perp
1 : for $j \in [1..N]$:	7 : if $l \in L$: return \perp	16 : if $C_j = \perp$: return \perp
2 : $k_j \xleftarrow{\$} \mathcal{K}$	8 : $L \leftarrow L \cup \{l\}$	17 : if $c = C_j$: return \perp
3 : $C_j \leftarrow \perp$	9 : $l_j \leftarrow l$	18 : $m \leftarrow \text{AE.dec}(k_j, l_j, c)$
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.

5 Constructions

should consist of:

- what old primitives to use
- the old primitives
- how to construct the new primitive from old primitives
- security bounds + proof
- comparison with existing alternatives

5.1 used primitives

\mathcal{K} is a nonempty key space, \mathcal{N} is a non-empty nonce space and \mathcal{M} is a message space. \mathcal{M} contain at least two strings, and if it contain a string of length x , it must contain all strings of length x . N is the number of users.

- loDEM: a lock-based one time use DEM scheme is defined by tuple $(\text{E.enc}, \text{E.dec})$. E.enc is a deterministic encryption algorithm that takes three inputs (k, l, m) to 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 . E.dec is the decryption algorithm that takes three inputs (k, l, c) to a value m . E.enc and E.dec are required to have 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 $\text{E.enc}(k, l, m) = c$). The security is defined as $\mathbf{Adv}_{A,N}^{\text{loDEM}} = |\Pr[\text{loDEM-IND\$}_{A,N}^0 = 1] - \Pr[\text{loDEM-IND\$}_{A,N}^1 = 1]|$, where loDEM-IND\$ is defined as follows:

Game $\text{loDEM-IND}^b_{A,N}$	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: loDEM-IND\$

- loMAC: The lock-based one time use MAC is a deterministic algorithm M.mac that takes in 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 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 . the security is defined as $\text{Adv}^{\text{loMAC}}_{F,A,N} = |\Pr[\text{loMAC-PRF}^0_{A,N} = 1] - \Pr[\text{loMAC-PRF}^1_{A,N} = 1]|$, where loMAC-PRF is defined as follows:

Game $\text{loMAC-PRF}^b_{A,N}$	Oracle $\text{Omac}(j, l, m)$	Oracle $\text{Ovrf}(j, m, t)$
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: $T_j \leftarrow t$	
	14: return t	

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

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

Figure 11: loMAC-PRF v2, A has access to oracle Omac **init** selects a uniformly random function from $\mathcal{K} \times \mathcal{L} \times \mathcal{M}$ to the set of all n -bit long strings

I am not sure which of these figures is better, figure 9 is closes to the loAE game, but get the feeling it might not be fully accurate as a prf is likely to return *true* when enough guesses are made as the output is always n -bits. If you can make n^2 queries, you can distinguish between $b = 0$ and $b = 1$ as a message will for sure verify correctly once if you try to verify it will all possible tags, where this strategy would not work when distinguishing with a prf as a prf will also verify correctly once if you try to verify it will all possible tags. However, I think we can disregard this as this problem would otherwise also arise when defining the loAE security. The second one feels more accurate to me as it also matches better with the mac requirement from NRSon pdf page 8. I think figure 9 is good for this setting but figure 10 might be a bit more complete so I am not sure which to use

5.2 construction

The loAE schemes we will look at are constructed from the loDEM and the loMAC. Following NRS, three ways to construct this loAE are of interest, namely the ones following from the N1, N2 and N3 scheme. One thing to keep in mind with this that these schemes would originally use associated data. For now we can discard this but it is not proven that the same security results would also follow from this case without associated data. Down here the schemes, adjusted to our setting, can be found, followed by the AE.enc and AE.dec calls that can we construct following these schemes. These calls can be plugged into game loAE-IND\$ to find their respective security games.

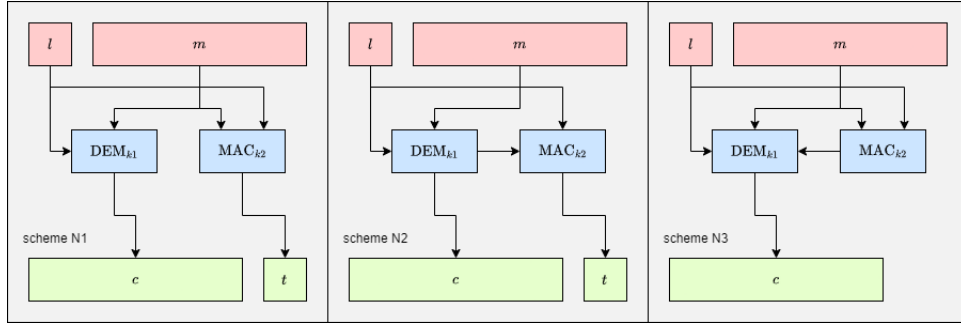


Figure 12: Adjusted N schemes from NRS

AE.enc(k, l, m)	AE.dec(k, l, c)
0 : $(k1, k2) \leftarrow k$	5 : $(k1, k2) \leftarrow k$
1 : $c' \leftarrow \text{E.enc}(k1, l, m)$	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 $t = t'$: return m
	10 : else : return \perp

Figure 13: 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 $t = t'$: return m
	10 : else : return \perp

Figure 14: 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 : $m' \leftarrow m \parallel t$	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 $t = t'$: return m
	10 : else : return \perp

Figure 15: Calls based on N3

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

6 Use cases

should consist of:

- possible use cases

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