Bachelor Thesis

Radboud Honours Academy

TBD

Author:
Stijn Vandenput

Supervisor:
Martijn Stam
Bart Mennink

20/05/2022

Abstract

Contents

1 Introduction **1**

2 Preliminaries **1**
2.1 notation table . . . 2

3 Existing AE/DEM notations in more detail **2**
3.1 Existing notation from Hybrid Encryption in a Multi-user Setting, revised . . . 2
3.1.1 notation . . . 2
3.1.2 used primitives . . . 3
3.1.3 goal . . . 3
3.1.4 Security model . . . 4
3.1.5 construction . . . 4
3.2 Existing notation from Generic Composition Reconsidered . . . 4
3.2.1 notation . . . 4
3.2.2 used primitives . . . 4
3.2.3 goal . . . 5
3.2.4 Security model . . . 5
3.2.5 construction . . . 6

4 New Definition **6**
4.1 notation . . . 6
4.2 goal . . . 6
4.3 Security model . . . 7
4.4 choices made . . . 7

5 Constructions **7**
5.1 used primitives . . . 8
5.2 construction . . . 9

6 Use cases **10**

7 Related Work **11**

8 Conclusion **11**

References **12**

9 Appendix **12**

1 Introduction

should consist of:

- explaining the challenge
- my contribution

Within the field of cryptography there are a 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, Hybrid Encryption in a Multi-user Setting, revised uses a construction that is very similar to Authenticated Encryption following the generic encrypt-then-MAC construction. This construction has since been revised in Generic Composition Reconsidered to be better applicable to common use cases. The aim of this thesis is to apply the knowledge of Generic Composition Reconsidered to the setting of Hybrid Encryption in a Multi-user Setting, revised and in doing so, create a better primitive for authenticated encryption in a asymmetric crypto setting.

2 Preliminaries

should consist of:

- general notation
- generic AE construction
- generic PKE schemes
- nonces vs locks

2.1 notation table

Name	pkc	GCrec	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	$\mathcal{C}$	$\mathcal{T}$	$\mathcal{T}$	set off 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	$\leftarrow^{\$}$	$\twoheadleftarrow$	$\leftarrow^{\$}$	get a random ellermetn form the set
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	*true*	-	*true*	boolean true
false	*false*	-	*false*	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 Hybrid Encryption in a Multi-user Setting, revised

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.enc,A.dec), A.enc take 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.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.dec(k,l,A.enc(k,l,m)) = m. The security of the ADEM is defined with $\mathbf{Adv}^{\text{l-miot-ind}}_{\text{ADEM},A,N}$ = |Pr[L-MIOT-IND$^0_{A,N}$ = 1] − Pr[L-MIOT-IND$^1_{A,N}$ = 1]|, defined by the following game:

Game L-MIOT-IND$^b_{A,N}$	**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.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.enc(k_j, l_j, m_b)	
5 : **return** b'	11 : $C_j \leftarrow C_j \cup \{c\}$	
	12 : **return** c	

Figure 1: L-MIOT-IND game, A has access to oracles Oenc and Odec and the locks in line 10 and 15 are the same.

- AMAC: the AMAC exists of tuple (M.mac,M.vrf). 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 ciphertext c in $\mathcal{C}$. M.vrf takes a key k in $\mathcal{K}$, a lock l in $\mathcal{L}$, a message m in $\mathcal{M}$ and a ciphertext c in $\mathcal{C}$ and returns either *true* or *false*. The correctness requirement is that for every combination of k, l and m, all corresponding $c \leftarrow$M.mac(k,l,m) gives M.vrf(k,l,m,c) = *true*. The security of the AMAC is defined with $\mathbf{Adv}^{\text{L-MIOT-UF}}_{\text{AMAC},A,N}$ = Pr[L-MIOT-UF$_{A,N}$], defined by the following game:

Game L-MIOT-UF$_{A,N}$	**Oracle** Omac(j,l,m)	**Oracle** Ovrf(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 : $k_j \leftarrow^{\$} \mathcal{K}$	10 : $l_j \leftarrow l$	17 : forged $\leftarrow 1$
4 : $T_j \leftarrow \emptyset$	11 : $t \leftarrow$ M.mac(k_j, l_j, m)	18 : **return** *true*
5 : $b' \leftarrow A$	12 : $T_j \leftarrow T_j \cup \{(m,t)\}$	19 : **else** : **return** *false*
6 : **return** b'	13 : **return** t	

Figure 2: L-MIOT-UF game, A has access to oracles Omac and Ovrf and the locks in line 11 and 16 are the same.

3.1.3 goal

The goal is to make a scheme ADEM' exists of tuple (A.enc',A.dec') which has the same security of the ADEM, but is also secure against active attacks.

3.1.4 Security model

The security of the ADEM is defined with $\mathbf{Adv}^{\text{l-miot-ind}}_{\text{ADEM'},A,N} = |\Pr[\text{L-MIOT-IND}^0_{A,N} = 1] - \Pr[\text{L-MIOT-IND}^1_{A,N} = 1]|$ where game L-MIOT-IND uses (A.enc',A.dec') instead of the former (A.enc,A.dec)

3.1.5 construction

The goal is met by creating new A.enc' and A.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 : $(k_{dem}, k_{mac}) \leftarrow k$	6 : $(k_{dem}, k_{mac}) \leftarrow k$
1 : $c' \leftarrow \text{A.enc}(k_{dem}, l, m)$	7 : $(c', t) \leftarrow c$
2 : $t \leftarrow \text{M.mac}(k_{mac}, l, c')$	8 : **if** $\text{M.mac}(k_{mac}, l, c', t)$:
3 : $c \leftarrow (c', t)$	9 : $m \leftarrow \text{A.dec}(k_{dem}, l, c')$
4 : **return** c	10 : **return** m
	11 : **else** : **return** $\perp$

Figure 3: A.enc' and A.dec' calls

The advantage is $\mathbf{Adv}^{\text{l-miot-ind}}_{\text{ADEM'},A,N} \leq 2\mathbf{Adv}^{\text{l-miot-uf}}_{\text{AMAC},B,N} + \mathbf{Adv}^{\text{l-miot-ind}}_{\text{ADAM},C,N}$. Where the running time of B is at most that of A plus the time required to run N-many ADEM encapsulations and Qd-many ADEM decapsulations and the running time of C is the same as the running time of A. Additionally, B poses at most Qd-many Ovrf queries, and C poses no Odec query.

3.2 Existing notation from Generic Composition Reconsidered

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. $\mathcal{M}$ contain 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 $\Pi = (\mathcal{K},\text{E},\text{D})$. E is a deterministic encryption algorithm that takes three inputs (k,n,m) to a value c, the length of c only depends the length of k, n and m. When (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 $\text{E}(k,n,m) = c \neq \perp$, then $\text{D}(k,n,c) = m$) and tidiness (if $\text{D}(k,n,c) = m \neq \perp$, then $\text{E}(k,n,m) = c$). The security is defined as $\mathbf{Adv}^{\text{nE}}_{\Pi,A} = |\Pr[\text{nE-IND\$}^0_A = 1] - \Pr[\text{nE-IND\$}^1_A = 1]|$, where nE-IND\$ is defined as follows:

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

Figure 4: 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 $\mathrm{F}(k,m) \neq \perp$. The security is defined as $\mathbf{Adv}^{\mathrm{MAC}}_{\mathrm{F},A} = |\Pr[\text{MAC-PRF}^0_A = 1] - \Pr[\text{MAC-PRF}^1_A = 1]|$, where MAC-PRF is defined as follows:

Game MAC-PRF^b_A	**Oracle** Omac(m)
0 : $U = \emptyset$	5 : **if** $m \in U$: **return** $\perp$
1 : $k \leftarrow^{\$} \mathcal{K}$	6 : $U = U \cup n$
2 : $b' \leftarrow A$	7 : $t \leftarrow \mathrm{F}(k, m)$
3 : **return** b'	8 : **if** $b = 1 \wedge t \neq \perp$:
	9 : $t \leftarrow^{\$} \{0,1\}^{\lvert t \rvert}$
	10 : **return** t

Figure 5: MAC-PRF, A has access to oracle Omac and U is the set of used messages

3.2.3 goal

The end goal is a nonce-based authenticated encryption scheme defined by triple $\Pi = (\mathcal{K},\mathrm{E},\mathrm{D})$. 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. When (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 $\mathrm{E}(k,n,a,m) = c \neq \perp$, then $\mathrm{D}(k,n,a,c) = m$) and tidiness (if $\mathrm{D}(k,n,a,c) = m \neq \perp$, then $\mathrm{E}(k,n,a,m) = c$)

3.2.4 Security model

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

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

Figure 6: 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 figure 6 from [1] on page 270.

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 nonempty key space, $\mathcal{L}$ is a non-empty lock 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.

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 (AE.enc,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. When (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 E.dec are inverse of each other implying correctness (if AE.enc(k,l,m) = $c \neq \perp$, then AE.dec(k,l,c) = m) and tidiness (if AE.dec(k,l,c) = $m \neq \perp$, then AE.enc(k,l,m) = c).

4.3 Security model

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

Game loAE-IND$\$^b_{A,N}$	**Oracle** Oenc(j,l,m)	**Oracle** 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 : $\quad k_j \leftarrow^{\$} \mathcal{K}$	8 : $L \leftarrow L \cup \{l\}$	17 : **if** $c \in C_j$: **return** $\perp$
3 : $\quad C_j \leftarrow \perp$	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 : $\quad c \leftarrow^{\$} \{0,1\}^{\lvert c \rvert}$	
	13 : $C_j \leftarrow c$	
	14 : **return** c	

Figure 7: loAE-IND\$ game, adversary has access to oracles Oenc and Odec.

4.4 choices made

In this security model, there are a lot of choices made, in this section I will elaborate on these. Firstly, the used MAC primitive is required to be indistinguishable from RO. This choice was made as it is required for AE constructions from Generic Composition Reconsidered. I also choose to have "locks" instead of nonces. In this setting, this results in the adversary not being able to make decryption queries for any incorrect lock values. I choose this as a setting with locks will suffice the setting of hybrid encryption while probably being easier to prove. For this reason I also choose to not allow multiple encryption calls for one user, as well as not allowing decryption calls to a user which encryption call is not yet made. In my security game for the DEM, there is no decryption oracle considered. When distinguishing from a random function, this is equivalent to a situation in which a decryption oracle is considered (I am not 100% sure about his part). Lastly, we choose to indistinguishably from a random function as a security model, instead of indistinguishably of two encrypted messages. This is a stronger security notion in the current setting as the length of the ciphertext only depends on the length of the inputs. Indistinguishably of two encrypted messages probably suffices in the current setting so I might still change to this if proving indistinguishably from a random function seems infeasible.

5 Constructions

should consist of:

- 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 (E.enc,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. When (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 inverse of each other implying correctness (if E.enc(k,l,m) $= c \neq \perp$, then E.dec(k,l,c) $= m$) and tidiness (if E.dec(k,l,c) $= m \neq \perp$, then E.enc(k,l,m) $= c$). The security is defined as $\mathbf{Adv}^{\text{loDEM}}_{A,N} = |\Pr[\text{loDEM-IND\$}^0_{A,N} = 1] - \Pr[\text{loDEM-IND\$}^1_{A,N} = 1]|$, where loDEM-IND\$ is defined as follows:

Game loDEM-IND$\$^b_{A,N}$	**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 : $\quad k_j \leftarrow^{\$} \mathcal{K}$	8 : $L \leftarrow L \cup \{l\}$
3 : $\quad C_j \leftarrow \perp$	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 : $\quad c \leftarrow^{\$} \{0,1\}^{\lvert c \rvert}$
	13 : $C_j \leftarrow c$
	14 : **return** c

Figure 8: 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 tag or $\perp$. The domain of M.mac is the set X such that M.mac(k,l,m) $\neq \perp$. the security is defined as $\mathbf{Adv}^{\text{loMAC}}_{\text{F},A,N} = |\Pr[\text{loMAC-PRF}^0_{A,N} = 1] - \Pr[\text{loMAC-PRF}^1_{A,N} = 1]|$, where loMAC-PRF is defined as follows:

Game loMAC-PRF$^b_{A,N}$	**Oracle** Omac(j,l,m)
0 : $L \leftarrow \emptyset$	6 : **if** $T_j \neq \perp$: **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 : $T_j \leftarrow \perp$	9 : $l_j \leftarrow l$
4 : $b' \leftarrow A$	10 : $t \leftarrow \text{M.mac}(k_j, l_j, m)$
5 : **return** b'	11 : **if** $b = 1 \wedge t \neq \perp$:
	12 : $t \leftarrow^{\$} \{0,1\}^{\lvert t \rvert}$
	13 : $T_j \leftarrow t$
	14 : **return** t

Figure 9: loMAC-PRF, A has access to oracle Omac

5.2 construction

The loAE schemes we will look at are constructed from the loDEM and the loMAC. Following Generic Composition Reconsidered, 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.

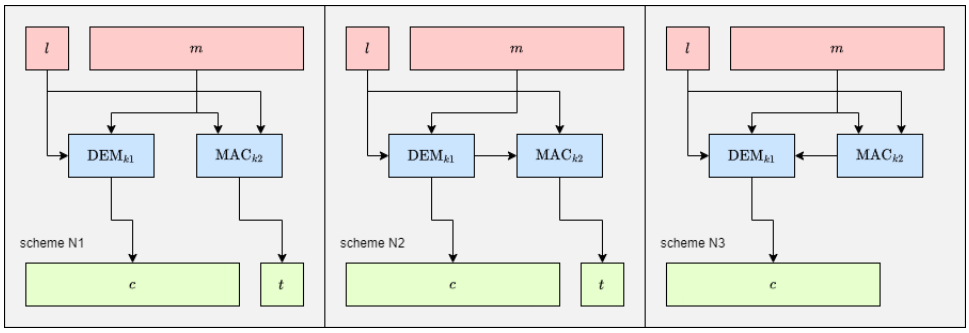

Figure 10: Adjusted N schemes from Generic Composition Reconsidered

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 11: 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 12: 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 : $m' \leftarrow m \Vert t$	7 : $(m, t) \leftarrow m'$
3 : $c \leftarrow E.enc(k1, l, m')$	8 : $t' \leftarrow \text{M.mac}(k2, l, m)$
4 : **return** c	9 : **if** $t = t'$: **return** m
	10 : **else** : **return** $\perp$

Figure 13: Calls based on N3

The scheme is considered secure when there is a tight reduction from breaking the AE-security of the scheme to breaking the defined 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] C. Namprempre, P. Rogaway, and T. Shrimpton, "Reconsidering generic composition," 2014, pp. 257–274. DOI: 10.1007/978-3-642-55220-5_15.

9 Appendix