Bachelor Thesis

Radboud Honours Academy

TBD

Author:
Stijn Vandenput

Supervisor:
Martijn Stam
Bart Mennink

20/05/2022

Abstract

Contents

1 Introduction

should consist of:

- explaining the challenge
- my contribution

2 Preliminaries

should consist of:

- recapping known definitions
- primitive definitions

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

3.1 Existing notation from pkc

3.1.1 notation

$\mathcal{M}$ is a message space, $\mathcal{K}$ is a finite key space, $\mathcal{T}$ is a tag 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 $\mathcal{K}$, a tag t in $\mathcal{T}$ and a message m in $\mathcal{M}$ and outputs a ciphertext c in $\mathcal{C}$. A.dec takes a K $\mathcal{K}$, a tag t in $\mathcal{T}$ 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 all K in $\mathcal{K}$ and t in $\mathcal{T}$ and m in $\mathcal{M}$ we have A.dec(K,t,A.enc(K,t,m)) = m. The security of the ADEM is defined with $\mathbf{Adv}^{\text{n-moit-ind}}_{\text{ADEM,A,N}} = |\Pr[\text{N-MIOT-IND}^0_{\text{A,N}}] - \Pr[\text{N-MIOT-IND}^1_{\text{A,N}}]|$, defined by the following game:

Game $\text{N-MIOT-IND}^b_{\mathsf{A},N}$	**Oracle** $\mathsf{Oenc}(j,t,m_0,m_1)$	**Oracle** $\mathsf{Odec}(j,c)$
00 $T \leftarrow \emptyset$	06 if $C_j \neq \emptyset$: return $\perp$	12 if $C_j = \emptyset$: return $\perp$
01 for all $j \in [1..N]$:	07 if $t \in T$: return $\perp$	13 if $c \in C_j$: return $\perp$
02 $\quad K_j \xleftarrow{\$} \mathcal{K}$	08 $T \leftarrow T \cup \{t\}$; $t_j \leftarrow t$	14 $m \leftarrow \mathsf{A.dec}(K_j, t_j, c)$
03 $\quad C_j \leftarrow \emptyset$	09 $c \leftarrow \mathsf{A.enc}(K_j, t_j, m_b)$	15 return m
04 $b' \xleftarrow{\$} \mathsf{A}$	10 $C_j \leftarrow C_j \cup \{c\}$	
05 return b'	11 return c	

  Where adversary A has access to oracles Oenc and Odec and the tags in line 9 and 14 are the same.

- AMAC: the AMAC exists of tuple (M.mac,M.vrf). M.mac takes a key K in $\mathcal{K}$, a tag t in $\mathcal{T}$, 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 tag t in $\mathcal{T}$, a message m in $\mathcal{M}$ and a ciphertext c in $\mathcal{C}$ and returns either true of false. The correctness requirement is that for all K in $\mathcal{K}$, t in $\mathcal{T}$ and message in $\mathcal{M}$ and c in

[M.mac(K,t,m)] we have M.vrf(K,t,m,c)=true. The security of the AMAC is defined with $\textbf{Adv}^{\text{N-moit-uf}}_{\text{AMAC,A,N}} = \Pr[\text{N-MIOT-UF}_{\text{A,N}}]$, defined by the following game:

```
Game N-MIOT-UF_{A,N}
00 forged ← 0
01 T ← ∅
02 for all j ∈ [1..N]:
03     K_j ←$ 𝒦
04     C_j ← ∅
05 run A
06 return forged
```

```
Oracle Omac(j, t, m)
07 if C_j ≠ ∅: return ⊥
08 if t ∈ T: return ⊥
09 T ← T ∪ {t}; t_j ← t
10 c ← M.mac(K_j, t_j, m)
11 C_j ← C_j ∪ {(m, c)}
12 return c
```

```
Oracle Ovrf(j, m, c)
13 if C_j = ∅: return ⊥
14 if (m, c) ∈ C_j: return ⊥
15 if M.vrf(K_j, t_j, m, c):
16     forged ← 1
17     return true
18 return false
```

Where Adversary A can access oracles Omac and Ovrf and the tags in line 10 and 15 are the same.

3.1.3 goal

The goal is to make a scheme ADEM' from the ADEM and AMAC which has the same security of the ADEM, but is also secure against active attacks.

3.1.4 Security model

The security is defined by creating new A.enc' and A.dec' calls which are build using the calls the primitives provide us, and placing those in the ADEM game defined earlier:

```
Proc A.enc'(K, t, m)
(K_dem, K_mac) ← K
c_dem ← A.enc(K_dem, t, m)
c_mac ← M.mac(K_mac, t, c_dem)
c ← (c_dem, c_mac)
return c
```

```
Proc A.dec'(K, t, c)
(K_dem, K_mac) ← K
(c_dem, c_mac) ← c
if M.vrf(K_mac, t, c_dem, c_mac):
    m ← A.dec(K_dem, t, c_dem)
    return m
return ⊥
```

The new advantage is $\textbf{Adv}^{\text{n-miot}}_{\text{ADEM',A,N}} \leq 2\textbf{Adv}^{\text{n-miot-uf}}_{\text{AMAC,B,N}} + \textbf{Adv}^{\text{n-moit-ind}}_{\text{ADAM,C,N}}$. 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. $\mathcal{M}$ (and $\mathcal{A}$? it's a bit ambiguous in the paper but I assume this part only applies to $\mathcal{M}$) 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 E scheme is defined by triple ($\mathcal{K}$,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 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 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 follows:

$$\mathbf{Adv}^{\mathrm{nE}}_{\Pi}(\mathcal{A}) = \Pr\left[\mathcal{A}^{\mathcal{E}(\cdot,\cdot)} \Rightarrow 1\right] - \Pr\left[\mathcal{A}^{\$(\cdot,\cdot)} \Rightarrow 1\right]$$

where $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is a nE scheme; $K \twoheadleftarrow \mathcal{K}$ at the beginning of each game; $\mathcal{E}(N, M)$ returns $\mathcal{E}_K(N, M)$; $\$(N, M)$ computes $C \leftarrow \mathcal{E}_K(N, M)$, returns $\perp$ if $C = \perp$, and otherwise returns $|C|$ random bits; and $\mathcal{A}$ may not repeat the first component of an oracle query.

- MAC: The MAC is a deterministic algorithm F that takes in a K in $\mathcal{K}$ and a string x and outputs either a n-bit length T or $\perp$. The domain of F is the set X such that F(K,x) $\neq \perp$. The security of F is defined by $\mathbf{Adv}^{\mathrm{pfr}}_{\mathrm{F}}$ =|Pr[A^{F}] − Pr[A^{P}]|. the game on the left selects a random K from $\mathcal{K}$ and provides oracle access to F(K,.) the game on the right selects a uniformly random function p from X to $\{1,0\}^n$ and provide oracle access to it. With each oracle, queries outside X return $\perp$

3.2.3 goal

The end goal is a nonce-based authenticated encryption scheme ($\mathcal{K}$,E,D). E is a deterministic encryption algorithm that takes four inputs (K,N,A,M) to a value C, the length of C value 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 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

Security is defined as follows:

$$\mathbf{Adv}^{\mathrm{nAE}}_{\Pi}(\mathcal{A}) = \Pr\left[\mathcal{A}^{\mathcal{E}(\cdot,\cdot,\cdot),\,\mathcal{D}(\cdot,\cdot,\cdot)} \Rightarrow 1\right] - \Pr\left[\mathcal{A}^{\$(\cdot,\cdot,\cdot),\,\perp(\cdot,\cdot,\cdot)} \Rightarrow 1\right]$$

where $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is an nAE scheme; $K \twoheadleftarrow \mathcal{K}$ at the beginning of each game; $\mathcal{E}(N, A, M)$ returns $\mathcal{E}_K(N, A, M)$ and $\mathcal{D}(N, A, C)$ returns $\mathcal{D}_K(N, A, C)$; and $\$(N, A, M)$ computes $C \leftarrow \mathcal{E}_K(N, A, M)$, returns $\perp$ if $C = \perp$, and $|C|$ random bits otherwise, and $\perp(N, A, M)$ returns $\perp$; and $\mathcal{A}$ may not repeat the first component of an encryption (=left) query, nor make a decryption (=right) query (N, A, C) after C was obtained from a prior encryption (=left) query (N, A, M).

We define the scheme 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.

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, $\mathcal{N}$ 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.

4.2 used primitives

- DEM: a 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 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,n,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,N=l,c) $= m \neq \perp$, then E.enc(k,l,m) = c). The security is defined with the following game where \$.enc(k,l,m) calls c = E.enc(k,l,m) then outputs $\perp$ if c is $\perp$ or $\mid c \mid$ random bits otherwise and the advantage is calculated with $\mathbf{Adv}^{\mathrm{DEM}}_{\mathrm{A,N}} = |\Pr[\mathrm{DEM}^{\mathrm{DEM}}_{\mathrm{A,N}}] - \Pr[\mathrm{DEM}^{\$}_{\mathrm{A,N}}]|$:

Game $\mathrm{DEM}^{E}_{A,N}$	**Oracle** Oenc(j,l,m)
$L \leftarrow \emptyset$	**if** $T_j \neq \emptyset$: **return** $\perp$
for $j \in [1..N]$:	**if** $l \in L$: **return** $\perp$
$\quad K_j \xleftarrow{\$} K$	$L \leftarrow L \cup \{l\}$
$b' \leftarrow A$	$l_j = l$
return b'	$c \leftarrow E.enc(K_j, l_j, m)$
	return c

Figure 1: DEM game where E is either the DEM or a random function \$, adversary A has access to Oenc

- MAC: The 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 of F is defined by the following security game where \$.mac(k,l,m) calls t = M.mac(k,l,m) then outputs $\perp$ if t is $\perp$ or $\mid t \mid$ random bits otherwise and the advantage is calculated with $\mathbf{Adv}^{\mathrm{MAC}}_{\mathrm{A,N}} = |\Pr[\mathrm{MAC}^{\mathrm{MAC}}_{\mathrm{A,N}}] - \Pr[\mathrm{MAC}^{\$}_{\mathrm{A,N}}]|$:

Game $\mathrm{MAC}^{M}_{A,N}$	**Oracle** Omac(j,l,m)
$L \leftarrow \emptyset$	**if** $T_j \neq \emptyset$: **return** $\perp$
for $j \in [1..N]$:	**if** $l \in L$: **return** $\perp$
$\quad K_j \xleftarrow{\$} K$	$L \leftarrow L \cup \{l\}$
$b' \leftarrow A$	$l_j = l$
return b'	$t \leftarrow M.mac(K_j, l_j, m)$
	return t

Figure 2: MAC game where M is either the MAC or a random function \$, adversary A has access to Omac

4.3 goal

The end goal is to build a Authenticated Encryption scheme (EA) secure against active attacks from the underlying primitives. 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 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,N=l,c) $= m \neq \perp$, then AE.enc(k,l,m) = c).

4.4 Security model

The security is defined by the following security game where \$.enc(k,l,m) calls c = AE.enc(k,l,m) then outputs $\perp$ if c is $\perp$ or $\mid c \mid$ random bits otherwise. \$.dec(k,l,c) always returns $\perp$ and the advantage is calculated with $\mathbf{Adv}^{\mathrm{AE}}_{\mathrm{A,N}} = |\Pr[\mathrm{AE}^{\mathrm{AE}}_{\mathrm{A,N}}] - \Pr[\mathrm{AE}^{\$}_{\mathrm{A,N}}]|$:

Game $\mathrm{AE}^{AE}_{A,N}$	**Oracle** Oenc(j,l,m)	**Oracle** Odec(j,m)
$L \leftarrow \emptyset$	**if** $T_j \neq \emptyset$: **return** $\perp$	**if** $c_j \neq \emptyset$: **return** $\perp$
for $j \in [1..N]$:	**if** $l \in L$: **return** $\perp$	**if** $c \in C_j$: **return** $\perp$
$K_j \xleftarrow{\$} K$	$L \leftarrow L \cup \{l\}$	$m \leftarrow AE.dec(K_j, L_j, c)$
$C_j \leftarrow \emptyset$	$l_j = l$	**return** m
$b' \leftarrow A$	$c \leftarrow AE.enc(K_j, l_j, m)$	
return b'	$C_j \leftarrow C_j \cup c$	
	return t	

Figure 3: AE game, where AE is either the AE scheme build from the MAC and DEM or a random function \$, adversary A has access to Oenc and Odec

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.

4.5 (

choices made)

5 Constructions

should consist of:

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

The AE schemes should be constructed from the DEM and the MAC. Following General Composition reconsidered, three ways to construct this AE 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 initial schemes can be found, followed by the AE.enc and AE.dec calls that can we construct following these schemes.

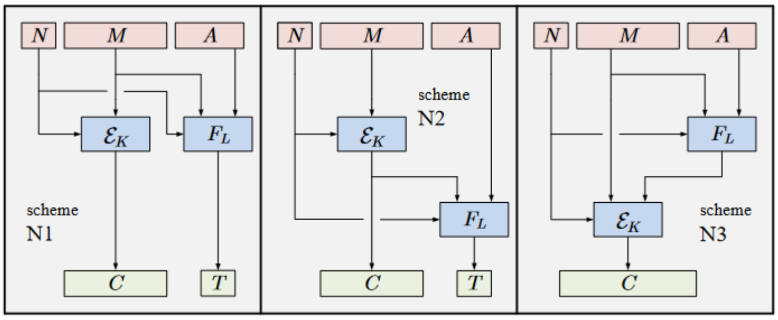

Figure 4: Original N schemes from Generic Composition reconsidered

AE.enc(k,l,m)	AE.dec(k,l,c)
$(k1, k2) \leftarrow k$	$(k1, k2) \leftarrow k$
$c' = E.enc(k1, l, m)$	$(c', t) \leftarrow c$
$t = M.mac(k2, l, m)$	$m = E.dec(k1, l, c')$
$c = (c', t)$	$t' = M.mac(k2, l, m)$
return c	**if** $t = t'$: **return** m
	else : **return** $\perp$

Figure 5: Calls based on N1

AE.enc(k,l,m)	AE.dec(k,l,c)
$(k1, k2) \leftarrow k$	$(k1, k2) \leftarrow k$
$c' = E.enc(k1, l, m)$	$(c', t) \leftarrow c$
$t = M.mac(k2, l, c')$	$m = E.dec(k1, l, c')$
$c = (c', t)$	$t' = M.mac(k2, l, c')$
return c	**if** $t = t'$: **return** m
	else : **return** $\perp$

Figure 6: Calls based on N2

AE.enc(k,l,m)	AE.dec(k,l,c)
$(k1, k2) \leftarrow k$	$(k1, k2) \leftarrow k$
$t = M.mac(k2, l, m)$	$m' = E.dec(k1, l, c)$
$m' = mt$	$(m, t) \leftarrow m'$
$c = E.enc(k1, l, m')$	$t' = M.mac(k2, l, m)$
return c	**if** $t = t'$: **return** m
	else : **return** $\perp$

Figure 7: Calls based on N3

6 Use cases

should consist of:

- possible use cases

7 Related Work

Location not final yet

8 Conclusion

9 Appendix