Bachelor Thesis

Radboud Honours Academy

TBD

Author:
Stijn Vandenput

Supervisor:
Martijn Stam
Bart Mennink

20/05/2022

Abstract

test [1]

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 def 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

3.1.2 used primitives

- ADEM: the ADEM exist of tuple (A.enc,A.dec), A.enc take a key
- AMAC: input fixed length tag, fixed length key and variable length message lead to a fixed length cythertext. It should be improbable to make a forgery (a pair (key, tag, message, cythertext) that verifies without begin generated by calling O.mac(key, tag, message) first). The AMAC gives us access to a mac and a ver call.

3.1.3 goal

Each user is provided with two keys, a message and a tag that is bound to the user and does not repeat between users. The message is encrypted using the tag and two keys to generate a cythertext consisting of two parts. First part is Cdem which is the message encrypted with the nonce and the first key while the second part is Cmac that is the mac computed over Cdem, the tag and the second key. Given only one queries to Oenc per user and multiple queries to Odec which always occurs after the Oenc queries, the message should be protected against active adversaries as long as DEM and MAC are secure.

3.1.4 Sec model

the security is purely based on the games for the AMAC and ADEM that are visible below where the ADAM calls are replaced with the ADEM' calls. All variables are elaborated in the paper

Game $\mathsf{N\text{-}MIOT\text{-}UF}_{\mathsf{A},N}$
$\mathsf{forged} \leftarrow 0$
$T \leftarrow \emptyset$
for all $j \in [1\,..\,N]$:
  $K_j \xleftarrow{\$} \mathcal{K}$
  $C_j \leftarrow \emptyset$
run A
return forged

Oracle $\mathsf{Omac}(j, t, m)$
if $C_j \neq \emptyset$: return $\perp$
if $t \in T$: return $\perp$
$T \leftarrow T \cup \{t\}$; $t_j \leftarrow t$
$c \leftarrow \mathsf{M.mac}(K_j, t_j, m)$
$C_j \leftarrow C_j \cup \{(m, c)\}$
return c

Oracle $\mathsf{Ovrf}(j, m, c)$
if $C_j = \emptyset$: return $\perp$
if $(m, c) \in C_j$: return $\perp$
if $\mathsf{M.vrf}(K_j, t_j, m, c)$:
  $\mathsf{forged} \leftarrow 1$
  return *true*
return *false*

Figure 1: AMAC game

Game $\mathsf{N\text{-}MIOT\text{-}IND}^{b}_{\mathsf{A},N}$
$T \leftarrow \emptyset$
for all $j \in [1\,..\,N]$:
  $K_j \xleftarrow{\$} \mathcal{K}$
  $C_j \leftarrow \emptyset$
$b' \xleftarrow{\$} \mathsf{A}$
return b'

Oracle $\mathsf{Oenc}(j, t, m_0, m_1)$
if $C_j \neq \emptyset$: return $\perp$
if $t \in T$: return $\perp$
$T \leftarrow T \cup \{t\}$; $t_j \leftarrow t$
$c \leftarrow \mathsf{A.enc}(K_j, t_j, m_b)$
$C_j \leftarrow C_j \cup \{c\}$
return c

Oracle $\mathsf{Odec}(j, c)$
if $C_j = \emptyset$: return $\perp$
if $c \in C_j$: return $\perp$
$m \leftarrow \mathsf{A.dec}(K_j, t_j, c)$
return m

Figure 2: ADEM game

with

Proc $\mathsf{A.enc}'(K, t, m)$
$(K_{\mathsf{dem}}, K_{\mathsf{mac}}) \leftarrow K$
$c_{\mathsf{dem}} \leftarrow \mathsf{A.enc}(K_{\mathsf{dem}}, t, m)$
$c_{\mathsf{mac}} \leftarrow \mathsf{M.mac}(K_{\mathsf{mac}}, t, c_{\mathsf{dem}})$
$c \leftarrow (c_{\mathsf{dem}}, c_{\mathsf{mac}})$
return c

Proc $\mathsf{A.dec}'(K, t, c)$
$(K_{\mathsf{dem}}, K_{\mathsf{mac}}) \leftarrow K$
$(c_{\mathsf{dem}}, c_{\mathsf{mac}}) \leftarrow c$
if $\mathsf{M.vrf}(K_{\mathsf{mac}}, t, c_{\mathsf{dem}}, c_{\mathsf{mac}})$:
  $m \leftarrow \mathsf{A.dec}(K_{\mathsf{dem}}, t, c_{\mathsf{dem}})$
  return m
return $\perp$

Figure 3: ADEM' calls

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 respectively. $\mathcal{M}$ and $\mathcal{A}$ contain at least two strings, and if they 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 (K,E,D). E is a deterministic encryption algorithm that takes three inputs (k,n,m) to a value c, the length of c value only depends the length of k, n and m. when (k,n,m) is not in $K \times N \times M$, 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:

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

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 K and a value x. The domain of F is the set X such that F(k,x) $\neq \perp$. The security of F is defined by $Adv_F^{prf} = Pr[A^F \Rightarrow 1] - [A^p \Rightarrow 1]$. the game on the left selects a random k from 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 (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 $K \times N \times A \times M$, 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) $/eqc \neq \perp$, then D(k,n,a,c) = m) and tidiness (if D(k,n,a,c) $/eqm \neq \perp$, then E(k,n,a,m) = c)

3.2.4 Sec model

Security is defined as follows:

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

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

for now we only look at the nonce based options as the pkc paper does that too.

4.1 used primitives

- DEM: input fixed length nonce, fixed length key and variable length message lead to a variable length cythertext which should be improbable to distinguish from a random function

$ (adversary has to guess if he is talking to $ or DEM). The dem gives us access to a enc and dec call.

- MAC: input fixed length nonce, fix length key and variable length message lead to a fixed length tag that should be improbable to distinguishable from a random function $ (adversary has to guess if he is talking to $ or MAC). The MAC gives us access to a mac call.

4.2 goal

Each user is provided with two keys, a message and a lock that does not repeat between users. The message is encrypted using the lock and two keys. Given only one queries to Oenc per user and multiple queries to Odec which always occurs after the Oenc queries, the message should be protected against active adversaries as long as DEM and MAC are secure.

4.3 Sec model

We define the following sec games for the MAC, the DEM and the AE (names will be improved later):

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

Game $\mathrm{DEM}^{E}_{A,N}$	**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'	$c \leftarrow E.enc(K_j, l_j, m)$
	return c

Figure 5: DEM game where E is either the DEM or a random function $, adversary A has access to Oenc

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** $\perp$
$\quad K_j \xleftarrow{\$} K$	$L \leftarrow L \cup \{l\}$	$m \leftarrow AE.dec(K_j, L_j, c)$
$\quad C_j \leftarrow \emptyset$	$l_j = l$	**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 6: 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

We should consider what the \$ calls should do, there are several cases to consider:

- \$ replacing M: \$.mac(k,l,m) calls t = M.mac(k,l,m) then outputs $\perp$ if t is $\perp$ or $|\, t \,|$ random bits otherwise.
- \$ replacing E: \$.enc(k,l,m) calls c = E.enc(k,l,m) then outputs $\perp$ if c is $\perp$ or $|\, c \,|$ random bits otherwise.
- \$ replacing AE: \$.enc(k,l,m) calls c = AE.enc(k,l,m) then outputs $\perp$ if c is $\perp$ or $|\, c \,|$ random bits otherwise. \$.dec(k,l,c) always returns $\perp$.

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.

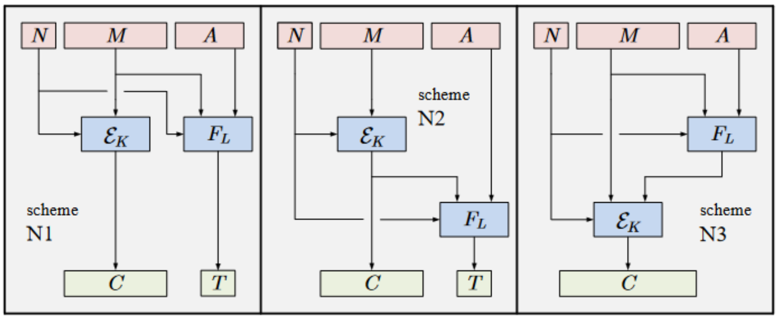

Figure 7: Original N schemes from Generic Composition reconsidered

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 8: 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 9: 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 10: Calls based on N3

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