1 current sec model to transcribe ADEM + AMAC to a more symmetric style

1.1 used primitives

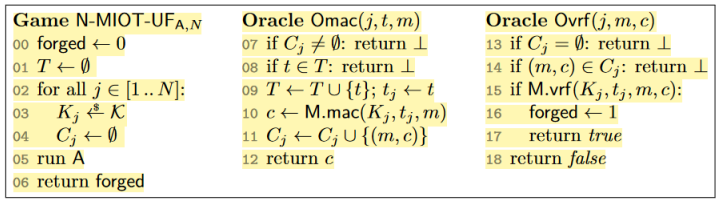
- ADEM: input fixed length tag, fixed length key and variable length message lead to a variable length cythertext. It should be improbable distinguish the cythertexts of two messages. (adversary may choose two cythertexts and has to guess which one of the two is encrypted). The ADEM gives us access to a enc and a dec call.
- 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.

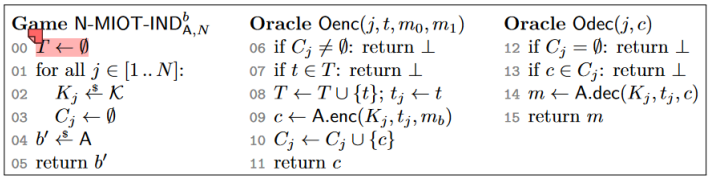
1.2 goal

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.

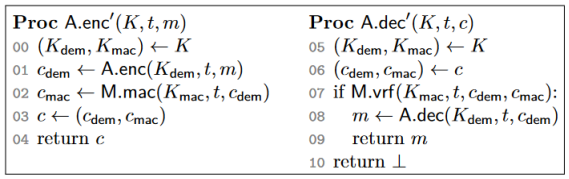
1.3 Sec model

the security is purely based on the games for the AMAC and ADEM that are visible below. All variables are elaborated in the paper





with



2 needed sec model to transcribe ADEM + AMAC to a more symmetric style

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

2.1 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 RO (adversary has to guess if he is talking to RO 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 random oracle (adversary has to guess if he is talking to RO or MAC). The MAC gives us access to a mac call.

2.2 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.

2.3 Sec model

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

Game $\text{MAC}^M_{A,N}$	**Oracle** Omac(j,l,m)
0 : $L \leftarrow \emptyset$	5 : **if** $T_j \neq \emptyset$: **return** $\perp$
1 : **for** $j \in [1..N]$:	6 : **if** $l \in L$: **return** $\perp$
2 : $K_j \leftarrow^{\$} K$	7 : $L \leftarrow L \cup \{l\}$
3 : $b' \leftarrow A$	8 : $l_j = l$
4 : **return** b'	9 : $t \leftarrow M.mac(K_j, l_j, m)$
	10 : **return** t

Figure 1: MAC game, M is either the MAC or Random Oracle

Game $\text{DEM}^E_{A,N}$	**Oracle** Omac(j,l,m)
0 : $L \leftarrow \emptyset$	5 : **if** $T_j \neq \emptyset$: **return** $\perp$
1 : **for** $j \in [1..N]$:	6 : **if** $l \in L$: **return** $\perp$
2 : $K_j \leftarrow^{\$} K$	7 : $L \leftarrow L \cup \{l\}$
3 : $b' \leftarrow A$	8 : $l_j = l$
4 : **return** b'	9 : $c \leftarrow E.enc(K_j, l_j, m)$
	10 : **return** c

Figure 2: DEM game, E is either the DEM or Random Oracle

Game $\text{AE}^{AE}_{A,N}$	**Oracle** Oenc(j,l,m)	**Oracle** Odec(j,m)
0 : $L \leftarrow \emptyset$	6 : **if** $T_j \neq \emptyset$: **return** $\perp$	13 : **if** $c_j \neq \emptyset$: **return** $\perp$
1 : **for** $j \in [1..N]$:	7 : **if** $l \in L$: **return** $\perp$	14 : **if** $c \in C_j$: **return** $\perp$
2 : $K_j \leftarrow^{\$} K$	8 : $L \leftarrow L \cup \{l\}$	15 : $m \leftarrow EA.dec(K_j, L_j, c)$
3 : $C_j \leftarrow \emptyset$	9 : $l_j = l$	16 : **return** m
4 : $b' \leftarrow A$	10 : $c \leftarrow EA.enc(K_j, l_j, m)$	
5 : **return** b'	11 : $C_j \leftarrow C_j \cup c$	
	12 : **return** t	

Figure 3: AE game, where AE is either the AE scheme having access to the MAC and DEM or RO

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

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

The AE schemes should be constructed from the DEM and the MAC. Following General Composition reconsidered, three ways to construct this EA 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.

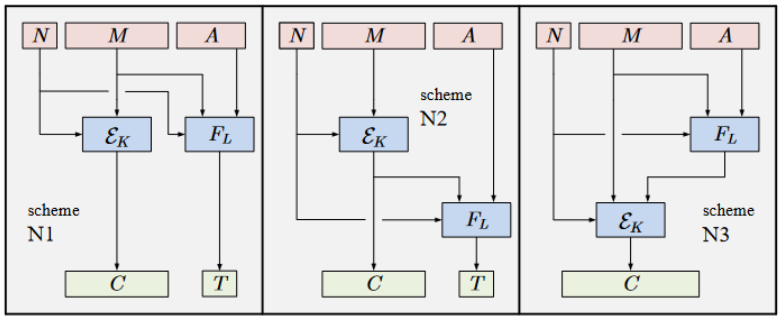


Figure 4: Original N schemes from Generic Composition reconsidered

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

Figure 5: Calls based on N1

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

Figure 6: Calls based on N2

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

Figure 7: Calls based on N3

3 burning questions

- Q1: Is het oke om de games gebazeerd te hebben op RO
  Q2:(klopt het dat RO stickt sterker is dan left-right)
- Q: in GCrec, waarom zijn er bij de n-schemes geen N4 en N5 schemas?
  A: die voegen niets toe Q2: klopt dat?
- Q: sommige dingen staan twee keer in crypto.bib, is er een voorkeur in welke je cite?
  A: journal -¿ procedings -¿ eprint (make sure that these could differ)

4 current todo's

- meer structuur aan brengen in main
- verplaatsen naar main
- CEASAR GCren people (paper)
- add crypto.bib as submodule
- change to ind-$
- introduction to modern cryptography katz lindell?
- in meer detail opschrijven wat de variables zijn van de sec games en wat de aannamens zijn
- mayhabs beginnen met schrijven in main

5 main idea

The PKC paper ends with a ADEM + AMAC construction as "solution". The original paper from ENC -¿ MAC has been revised, so this should prob be revised as well. In general its nice to write down thing in a more "sym crypto" style as we use symmetric primitives. It would probably also be nice to revise it more in general and see what other ways there are to reach the end-goal expected in the PKC paper.