Bachelor Thesis

Radboud University

TBD

Author:
Stijn Vandenput

Supervisors:
Martijn Stam
Bart Mennink

20/05/2022

Abstract

Contents

1 Introduction

Although symmetric and asymmetric cryptography are both subfields of cryptography, their research area's can be quite separated. This can lead to knowledge gaps between the two when work in asymmetric crypto uses constructions that are more common in symmetric crypto or the other way around. In this fashion, a paper by Giacon, 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. In this revision, a new set of constructions is given that be better applicable to common use cases. The aim of this thesis is to apply the knowledge from NRS to the setting of GKP and while doing so, create a new primitive for authenticated encryption suited for asymmetric settings.

2 Preliminaries

In this section we will explain several concepts important to the rest of our work, as well as some general notation.

2.1 General Notation

Strings are binary and bit-wise, the set of all strings is $\{0,1\}^*$. The length of x is written as |x|, the concatenation of x and y as x $\|$ y, a being the result of b as a $\leftarrow$ b and taking a random sampling from y and assigning it to x as x $\xleftarrow{\$}$ y. We allow a single type of error message written as $\perp$. $\mathcal{K}$ is a nonempty key space, $\mathcal{L}$ is a lock space, $\mathcal{N}$ is a 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 $\mathcal{M}$ or $\mathcal{A}$ contain a string of length x, it must contain all strings of length x. N is the number of users.

2.2 Authenticated Encryption

Two different security requirements are data privacy, the insurance that data cannot be viewed by a unauthorized party, and data integrity, the insurance that data has not been modified by a unauthorized party. Data privacy can be achieved by using encryption primitives while data integrity is often achieved using a MAC function. Authenticated encryption combines both of these security requirements into one and ensures both data privacy and integrity. In addition some authenticated encryption schemes allow you to provide additional data, AD. For this data, only data integrity is required. Authenticated encryption schemes that allow AD are called AEAD schemes.

2.3 Nonces and Locks

The birthday bound tells us that whenever the length of a key has n bits, a collision between is likely after sampling $2^{n/2}$ different keys.(**Question: citation needed?**) As a result, key collisions can become a problem within when considering multiple users. Most commonly, this problem is evaded by supplying a additional, non-secret, argument to the security primitive. The idea behind this is that, even when the keys of different users collide, the encryption of the same message will result in different ciphertexts when this x is different. Both the key and the new argument would need to collide for the same problems to arise. Different approaches of using this additional argument have been developed, of which nonces and locks are of interest to us.

Nonces Nonce is short for number once used. As the name suggests, this number is assumed to only be used once per user to encrypt a message. Decrypting multiple messages with one nonce is allowed. Generally, nonces are allowed to repeat between users but, when specifically stated, they can be globally unique.

Locks Locks work a little bit different, where nonces are bound to the message, locks are bound to the user. When first encrypting the lock value is provided and bound to the user. On decryption, no lock value is provided. Instead any decryptions will be made with the lock value bound to the user. Locks are assumed to be globally unique.

Between the two, locks are more restrictive as you cannot decrypt multiple messages with different locks using the same key. You can also not decrypt a messages before encrypting one as the user would not have a lock bound to it yet. When keys are used only once, both of these restrictions are irrelevant as you would never want to decrypt messages with different locks, or decrypt before encrypting. As a result, locks are more suited in this setting. (**Question: can I to say this?**)

2.4 Security Notions

The advantage of the adversary can me modeled as a distinguishing advantage. There are many different things we can distinguish, leading to many different security notions. The relevant ones are written below.

IND-CPA vs IND-CCA You can model against an passive or a active attacker. A passive attacker can only read the send messages while a active attacker can also alter the send messages. A passive attacker can be modelled using a chosen plaintext attack model, CPA for short. In this model, the adversary can choose the plaintext that it want to encrypt, but has no influence over the ciphertext. A active attacker can be modelled using a chosen ciphertext attack model, CCA for short. In this model, the adversary can choose the plaintext that it want to encrypt, as well as the ciphertext it wants to decrypt. IND-CPA refers to CPA indistinguishablility while IND-CCA refers to CCA indistinguishablility. IND-CPA implies IND-CCA,but not the other way around.

IND-$ vs IND-LOR Left or right indistinguishablility refers to a situation where the adversary can give two messages, and has to guess which of the two is encrypted. $ indistinguishablility refers to a situation where the adversary is given access to either the real construct, or to a random function $. This random function returns a random string with the same length as the ciphertext would have. As long as the length of the ciphertext is not randomized, IND-$ implies IND-LOR, but not the other way around. (**todo: add citation**)

Both of these are separate dimension and can be combined into 4 different notations. For example IND-CCA-$ refers to a situation where the adversary has to distinguish between the real construct, or a random function while being able to choose both the plaintext and the ciphertext.

PRF-MAC vs unforgeable MAC Mac functions is said to be PRF secure when the tag it outputs is not be distinguishable from a PRF taking the input space to the tag space. A mac is said to be unforgeable when it is infeasible to create a valid message-tag pair without using the secret key. A PRF secure mac is also unforgeable while a unforgeable mac is not necessarily PRF secure.

2.5 Game Based Security Notions

Security notions can be written in a game based format, using pseudocode instead of text. As a example, a security game for the IND-CPA-$ game of a nonce based encryption block can be found in figure 1. A bit that models the challenge is given to the game, in this case the challenge bit b signals wether we are in the real of the ideal world. The adversary has to return his guess for this bit as b'. In addition, the adversary can have access to oracles. In our example there is only one oracle that takes a nonce and a message. Using game based notation, you can clearly write out all the limitations. For example, the limitation that nonces can not be reused is modeled by line 0, 5 and 6. Line 8 and 9 model how the random function $ behaves. These limitations could be written out in text based format as well but when there are multiple limitations, writing it out in a game based format can be more comprehensible.

2.6 Security Proofs

general introduction to how we proof security of generic composition.

3 NRS and GKP in Detail

In this section we explain the parts from GKP and NRS important to our work. Afterwards, a comparison is made between the two papers. Some notations will be different from the original papers for improved consistency. What are called tags in GKP, we will call locks instead to avoid confusion with the output of mac functions and we call the output of the AMAC the tag instead of the ciphertext. The security notions from NRS are converted to a game-based format using insights from (**todo add citation for Automated Analysis of Protocols that use Authenticated Encryption: How Subtle AEAD**) in order to better match the notation from GKP and be more adaptable to a multi-user setting.

3.1 NRS

Three generic ways to construct a authenticated encryption scheme are discussed in a paper written by Bellare and Namprempre [2]: encrypt-then-mac, encrypt-and-mac and mac-then-encrypt. In this paper, encrypt-then-mac is considered the only secure one when using probabilistic encryption as a building block. Within NRS these constructions are generalized to using nonce- or iv-based encryption as a building block to create nonce-based authenticated encryption schemes, nAEs for short. We will look at the constructions using a nonce-based encryption, nE for short, and a PRF secure MAC function.

3.1.1 Used Primitives

nE A nonce-based encryption scheme is defined by triple $\Pi = (\mathcal{K}, \mathrm{E}, \mathrm{D})$. Deterministic encryption algorithm E takes three inputs (k, n, m) and outputs a value c, the length of c only depends 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 $\perp$. Decryption algorithm D takes three inputs (k, n, c) and outputs a value m. Both E and D are required to satisfy correctness (if $\mathrm{E}(k, n, m) = c \neq \perp$, then $\mathrm{D}(k, n, c) = m$) and tidiness (if $\mathrm{D}(k, n, c) = m \neq \perp$, then $\mathrm{E}(k, n, m) = c$).

Game nE-IND-CPA-$\$^b_A$	**Oracle** $\mathrm{Oenc}(n, m)$
$U \leftarrow \emptyset$	**if** $n \in U$: **return** $\perp$
$k \xleftarrow{\$} \mathcal{K}$	$U \leftarrow U \cup \{n\}$
$b' \leftarrow A$	$c \leftarrow \mathrm{E}(k, n, m)$
return b'	**if** $b = 1 \wedge c \neq \perp$:
	$\quad c \xleftarrow{\$} \{0,1\}^{\lvert c \rvert}$
	return c

Figure 1: nE-IND-CPA-$ game, A has access to oracle Oenc.

Game MAC-PRF^b_A	**Oracle** $\mathrm{Omac}(m)$
$U \leftarrow \emptyset$	**if** $m \in U$: **return** $\perp$
$k \xleftarrow{\$} \mathcal{K}$	$U \leftarrow U \cup \{m\}$
$b' \leftarrow A$	$t \leftarrow \mathrm{F}(k, m)$
return b'	**if** $b = 1 \wedge t \neq \perp$:
	$\quad t \xleftarrow{\$} \{0,1\}^{\lvert t \rvert}$
	return t

Figure 2: MAC-PRF, A has access to oracle Omac and U is the set of used messages.

nE security The security of a nE is defined as $\mathbf{Adv}^{\mathrm{nE}}_{\Pi,A} = |\Pr[\text{nE-IND-CPA-}\$^0_A = 1] - \Pr[\text{nE-IND-CPA-}\$^1_A = 1]|$, where nE-IND-CPA-$ is in figure 1. In this game, set U keeps track of all used nonces as the adversary is not allowed to repeat those.

MAC A MAC scheme is defined by algorithm F that takes a key k in $\mathcal{K}$ and a string m and outputs either a n-bit tag t or $\perp$. The domain of F is the set X off al m such that $\mathrm{F}(k, m) \neq \perp$ is in X, this domain may not depend on k.

MAC security The security of a MAC 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 in figure 2. In this game the set U keeps track of the used messages to prevent trivial wins.

3.1.2 Nonce-Based Authenticated Encryption

A nonce-based authenticated encryption scheme is defined by triple $\Pi = (\mathcal{K}, \mathrm{E}, \mathrm{D})$. Deterministic encryption algorithm E takes four inputs (k, n, a, m) and outputs 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 $\perp$. Decryption algorithm D takes four inputs (k, n, a, c) and outputs a value m. both E and D are required to satisfy 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 $\mathrm{E}(k, n, a, m) = c$).

nAE security The security of a nAE is defined as $\mathbf{Adv}^{\mathrm{nAE}}_{\Pi,A} = |\Pr[\text{nAE-IND-CCA-}\$^0_A = 1] - \Pr[\text{nAE-IND-CCA-}\$^1_A = 1]|$, where nAE-IND-CCA-$ is in figure 3. In this game, set U

Game nAE-IND-CCA-$\$^b_A$	**Oracle** $\mathrm{Oenc}(n, a, m)$	**Oracle** $\mathrm{Odec}(n, a, c)$
$U \leftarrow \emptyset$	**if** $n \in U$: **return** $\perp$	**if** $b = 1$: **return** $\perp$
$Q \leftarrow \emptyset$	$U \leftarrow U \cup \{n\}$	**if** $(n, a, \_, c) \in Q$: **return** $\perp$
$k \xleftarrow{\$} \mathcal{K}$	**if** $(n, a, m, \_) \in Q$: **return** $\perp$	$m \leftarrow \mathrm{D}(k, n, a, c)$
$b' \leftarrow A$	$c \leftarrow \mathrm{E}(k, n, a, m)$	$Q \leftarrow Q \cup \{(n, a, m, c)\}$
return b'	**if** $b = 1 \wedge c \neq \perp$:	**return** m
	$\quad c \xleftarrow{\$} \{0,1\}^{\lvert c \rvert}$	
	$Q \leftarrow Q \cup \{(n, a, m, c)\}$	
	return c	

Figure 3: nAE-IND-CCA-\$ game, A has access to oracles Oenc and Odec.

keeps track of all used nonces as the adversary is not allowed to repeat those on encryption. Following the translation of IND-CCA-\$ to a security game for AE from (**todo add citation for Automated Analysis of Protocols that use Authenticated Encryption: How Subtle AEAD**), $\_$ denotes a variable that is irrelevant and set Q keeps tack of all query results in order to prevent trivial wins.

3.1.3 Construction

A nAE scheme is constructed by several 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 they can be viewed in figure 6 of NRS. Noteworthy is that these relate to encrypt-and-mac, encrypt-then-mac,and mac-then-encrypt respectively, showing the general believe that encrypt-then-mac is the only safe construction does not transfer to this setting.

3.2 GKP

In GKP, the concept of augmentation using locks is discussed. The authors start by showing some data encapsulation mechanisms are vulnerable to a passive multi-instance distinguishing- and key recovery and how this can lead to problems when used in public key encryption. They define the augmented data encapsulation mechanisms, ADEM for short, that uses locks to negate these insecurities. Additionally, they show how a ADEM that is secure against passive attacks can be combined with a MAC that is augmented in a similar fashion, called a AMAC, to construct ADEM' that is safe against active attackers. This construction is similar to construction N2 from NRS.

3.2.1 Used Primitives

In GKP, $\mathcal{M}$ is not required to contain at least two strings, and to contain all strings of length x if it contains a string of length x. Additionally, $\mathcal{K}$ is required to be finite but not required to be non-empty.

ADEM A ADEM scheme is defined by tuple $(\mathrm{A.enc}, \mathrm{A.dec})$. Deterministic algorithm A.enc 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 $\mathcal{C}$.

Game L-IND-CPA-LOR$^b_{A,N}$	**Oracle** Oenc(j, l, m_0, m_1)
$L \leftarrow \emptyset$	**if** $C_j \neq \emptyset$: **return** $\perp$
for $j \in [1..N]$:	**if** $l \in L$: **return** $\perp$
$\quad k_j \xleftarrow{\$} \mathcal{K}$	$L \leftarrow L \cup \{l\}$
$\quad C_j \leftarrow \emptyset$	$l_j \leftarrow l$
$b' \leftarrow A$	$c \leftarrow$ A.enc(k_j, l_j, m_b)
return b'	$C_j \leftarrow C_j \cup \{c\}$
	return c

Figure 4: L-IND-CPA-LOR game, A has access to oracle Oenc.

Deterministic algorithm 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$(k, l, \text{A.enc}(k, l, m)) = m$.

ADEM security The security of a ADEM is defined as $\mathbf{Adv}^{\text{l-ind-cpa-lor}}_{\text{ADEM},A,N} = |\Pr[\text{L-IND-CPA-LOR}^0_{A,N} = 1] - \Pr[\text{L-IND-CPA-LOR}^1_{A,N} = 1]|$, where L-IND-CPA-LOR is in figure 4. Every user is only allowed one encryption query and locks may not repeat between users. Decryption queries are only allowed after the encryption. The corresponding game can be found in figure 9 from GKP, note that this figure also included a decryption oracle the adversary is not allowed to use. (**Question: I do not really know if I should elaborate on how this translates to the game, as the game is based on the game of gkp**)

AMAC A AMAC scheme is defined by tuple (M.mac, M.vrf). Deterministic algorithm 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 tag t in $\mathcal{T}$. Deterministic algorithm M.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.mac(k, l, m) gives M.vrf$(k, l, m, t) = $ *true*.

AMAC security The security of a AMAC is defined as $\mathbf{Adv}^{\text{L-MIOT-UF}}_{\text{AMAC},A,N} = \Pr[\text{L-MIOT-UF}_{A,N} = 1]$, where L-MIOT-UF is in 5. Every user is only allowed one mac query and locks may not repeat between users. Verification queries are only allowed after the encryption. The corresponding game can be found in figure 15 of GKP.

3.2.2 ADEM'

A ADEM' scheme is defined by tuple (A.enc', A.dec'). Deterministic algorithm A.enc' 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 $\mathcal{C}$. Deterministic algorithm 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'$(k, l, \text{A.enc'}(k, l, m)) = m$.

ADEM' security The security of a ADEM' is defined as $\mathbf{Adv}^{\text{l-ind-cca-lor}}_{\text{ADEM'},A,N} = |\Pr[\text{L-IND-CCA-LOR}^0_{A,N} = 1] - \Pr[\text{L-IND-CCA-LOR}^1_{A,N} = 1]|$, where L-IND-CCA-LOR is in 6. Every user is only allowed one encryption query and locks may not repeat between users. Decryption queries are only allowed after the encryption. The corresponding game can be found in figure 9 of GKP.

Game L-MIOT-UF$_{A,N}$

0 : forged $\leftarrow 0$
1 : $L \leftarrow \emptyset$
2 : **for** $j \in [1..N]$:
3 : $\quad k_j \xleftarrow{\$} \mathcal{K}$
4 : $\quad T_j \leftarrow \emptyset$
5 : **run** A
6 : **return** forged

Oracle Omac(j, l, m)

7 : **if** $T_j \neq \emptyset$: **return** $\perp$
8 : **if** $l \in L$: **return** $\perp$
9 : $L \leftarrow L \cup \{l\}$
10 : $l_j \leftarrow l$
11 : $t \leftarrow$ M.mac(k_j, l_j, m)
12 : $T_j \leftarrow T_j \cup \{(m, t)\}$
13 : **return** t

Oracle Ovrf(j, m, t)

14 : **if** $T_j = \emptyset$: **return** $\perp$
15 : **if** $(m, t) \in T_j$: **return** $\perp$
16 : **if** M.vrf(k_j, l_j, m, t) :
17 : $\quad$ forged $\leftarrow 1$
18 : $\quad$ **return** *true*
19 : **else** : **return** *false*

Figure 5: L-MIOT-UF game, A has access to oracles Omac and Ovrf and the locks in line 11 and 16 are the same.

Game L-IND-CCA-LOR$^b_{A,N}$

0 : $L \leftarrow \emptyset$
1 : **for** $j \in [1..N]$:
2 : $\quad k_j \xleftarrow{\$} \mathcal{K}$
3 : $\quad C_j \leftarrow \emptyset$
4 : $b' \leftarrow A$
5 : **return** b'

Oracle Oenc(j, l, m_0, m_1)

6 : **if** $C_j \neq \emptyset$: **return** $\perp$
7 : **if** $l \in L$: **return** $\perp$
8 : $L \leftarrow L \cup \{l\}$
9 : $l_j \leftarrow l$
10 : $c \leftarrow$ A.enc'(k_j, l_j, m_b)
11 : $C_j \leftarrow C_j \cup \{c\}$
12 : **return** c

Oracle Odec(j, c)

13 : **if** $C_j = \emptyset$: **return** $\perp$
14 : **if** $c \in C_j$: **return** $\perp$
15 : $m \leftarrow$ A.dec'(k_j, l_j, c)
16 : **return** m

Figure 6: L-IND-CCA-LOR game, A has access to oracles Oenc and Odec and the locks in line 10 and 15 are the same.

Proc A.enc'(k, l, m)	**Proc** A.dec'(k, l, c)
$(k_{dem}, k_{mac}) \leftarrow k$	$(k_{dem}, k_{mac}) \leftarrow k$
$c' \leftarrow \text{A.enc}(k_{dem}, l, m)$	$(c', t) \leftarrow c$
$t \leftarrow \text{M.mac}(k_{mac}, l, c')$	**if** $\text{M.vrf}(k_{mac}, l, c', t)$:
$c \leftarrow (c', t)$	$m \leftarrow \text{A.dec}(k_{dem}, l, c')$
return c	**return** m
	else : **return** $\perp$

Figure 7: A.enc' and A.dec' calls, The corresponding calls can be found in figure 16 of GKP.

3.2.3 Construction

The ADEM' scheme considered is made by creating A.enc' and A.dec' calls using the calls the primitives provide us as seen in figure 7. The construction is deemed secure as for any N and a A that makes Q_d many Odec queries, the exist B and C such that $\mathbf{Adv}^{\text{l-ind-cca-lor}}_{\text{ADEM'},A,N} \leq 2\mathbf{Adv}^{\text{l-miot-uf}}_{\text{AMAC},B,N} + \mathbf{Adv}^{\text{l-ind-cpa-lor}}_{\text{ADAM},C,N}$ 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.

3.3 Comparison of GKP and NRS

In this section we will highlight the important differences between GKP and NRS. (**Question: I feel like this section is very factual right now, while it might also be worthwhile to elaborate on some things a bit more.**)

Setting NRS is written is single-user, multiple-use key setting while GKP is written is a multi-user, one-time use key setting. As a result, GKP uses locks while NRS uses nonces. (**Question: is it necessary to elaborate on how these settings lead to lock, or nonce usage, or would it suffice to explain this in section 2.4**).

Aim While NRS is aimed at generalizing the AE constructions, GKP is aimed aimed at finding a single construction that is safe when used in public key encryption. Most notably, this results in NRS evaluating 20 possible constructions while GKP evaluates one. Additionally, the constructions from NRS are able to use AD while the construction form GKP cannot.

Security Notion The security notions of NRS are written in a IND-$ fashion while the security notions of GKP are written in a lor fashion. In other words, NRS requires the valid ciphertext to be indistinguishable from random strings, GKP only requires them to be indistinguishable from each other. As a result, the MAC primitives of the two papers have different security requirements. In NRS, the tag is required to be indistinguishable from a random string while in GKP the tag is only required to be unforgeable.

Game loAE-IND-CCA-$\$^b_{A,N}$

$L \leftarrow \emptyset$
for $j \in [1..N]$:
  $k_j \xleftarrow{\$} \mathcal{K}$
  $C_j \leftarrow \perp$
$b' \leftarrow A$
return b'

Oracle Oenc(j, l, m)

if $C_j \neq \perp$: **return** $\perp$
if $l \in L$: **return** $\perp$
$L \leftarrow L \cup \{l\}$
$l_j \leftarrow l$
$c \leftarrow \text{AE.enc}(k_j, l_j, m)$
if $b = 1 \wedge c \neq \perp$:
  $c \xleftarrow{\$} \{0,1\}^{|c|}$
$C_j \leftarrow c$
return c

Oracle Odec(j, c)

if $C_j = \perp$: **return** $\perp$
if $c = C_j$: **return** $\perp$
$m \leftarrow \text{AE.dec}(k_j, l_j, c)$
if $b = 1$: $m = \perp$
return m

Figure 8: loAE-IND-CCA-$ game, adversary has access to oracles Oenc and Odec.

4 Defining the New Primitive

In this section we will discuss a new security primitive, the lock-based one-time use Authenticated Encryption scheme, loAE scheme for short. As the name suggests, this primitive is used in a setting where a key is used only once to encrypt and authenticate a single message.

4.1 loAE

A loAE scheme is defined by tuple (AE.enc, AE.dec). Deterministic algorithm AE.enc takes three inputs (k, l, m) and outputs 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 $\perp$. Deterministic algorithm AE.dec takes three inputs (k, n, c) and outputs a value m. Both AE.enc and EA.dec are required to satisfy 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$(k, l, m) = c$).

4.2 Security Model

The security is defined as $\mathbf{Adv}^{\text{loAE}}_{A,N} = |\Pr[\text{loAE-IND-CCA-}\$^0_{A,N} = 1] - \Pr[\text{loAE-IND-CCA-}\$^1_{A,N} = 1]|$, where loAE-IND-CCA-$ is in figure 8.

4.3 Explanation of the Security Model

In this section we will elaborate on the security game loAE-IND-CCA-$, as well as why this model was chosen over some alternatives. As you will never have to decrypt messages with multiple nonces for a single user, we uses locks instead of nonces. Locks may not repeat between users. The user is only allowed one encryption query and decryption queries are only allowed after the encryption as we are using one-time use keys.

To define the security, we use a ind-$ security notion instead of left-or-right one as it is the stronger security notion in our setting. (**Question: is it necessary to elaborate on how this here, or would it suffice to explain this in section 2.6**) On decryption, we use a function that always returns $\perp$ to ensure the adversary can not guess which ciphertexts would be valid ciphertexts.

Multiple users Line 1 loops over all the users to initialize them while line 2 assigns a random key to each user. These users are given as an argument to the oracles Oenc and Odec.

Locks Line 0 initializes the set of all used locks to the empty set. Locks are not allowed to repeat, if the lock is in the set of used sets we return $\perp$ on line 7. If this check passes, add the lock to the sets of used locks in line 8 and bind it ot the user in line 9. Note that locks may be added to the set of used locks even if they are never used to encrypt a valid message. (**todo: see if this needs to be altered**)

Encryption and decryption If the given arguments are valid, and we are in the real world, line 10 encrypts the message and line 17 decrypts the message.

One-time use keys The variable C_j is used to prevent multiple encryptions from making multiple encryptions. In contrast to GKP, we do not use set notation, as we can never have multiple ciphertexts related to one user. In line 3, we set C_j to be undefined. We do not allow multiple encryptions per user. Therefore, if the ciphertexts is defined in line 6, we return $\perp$. In line 13, the newly computed ciphertext is bound to C_j. If the encryption was invalid, C_j will stay undefined. Even though this leads to the adversary being able to call Oenc twice on a single user, this will not give the adversary a advantage as the values for which AE.enc returns $\perp$ are known. If the user has made no valid encryption yet, decryption is not allowed and we return $\perp$ on line 15 as C_j will be undefined.

Preventing trivial wins Line 16 prevents a trivial win. If the ciphertext given to Odec is allowed to be the same as the ciphertext returned by Oenc, it would be trivial to distinguish the real and ideal world. The ideal world would return $\perp$ while the real world would not. For this reason the real world should return $\perp$ as well.

Implementation of \$ On encryption, whenever AE returns $\perp$, the random function should return $\perp$ as well. Therefore, the random function is only called if b = 1 and AE.enc does not return $\perp$. This is checked in line 11. If the check passes, the random function samples as string uniformly at random from the set of all strings with the length of the ciphertext and binds it to the ciphertext in line 12. On decryption, the ideal world always returns $\perp$. **(todo: add part about ideal vs attainable)**

5 Constructions

In this section we discuss how we can construct a safe loAE. Similarly to GKP and NRS we will look at constructions combining a deterministic encryption primitive and mac primitive. First write down the definitions of these two primitives, then we will look at how we can combine the two and which security bounds we can expect. Lastly we compare our choices with existing alternatives.

5.1 Used Primitives

loE A lock-based one-time use encryption scheme, loE for short, is defined by tuple (E.enc, E.dec). Deterministic algorithm E.enc takes three inputs (k, l, m) and outputs 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 $\perp$. Deterministic algorithm E.dec takes three inputs (k, l, c) and outputs

Game loE-IND-CPA-$\$^b_{A,N}$	**Oracle** $\mathrm{Oenc}(j, l, m)$
$L \leftarrow \emptyset$	**if** $C_j \neq \perp$: **return** $\perp$
for $j \in [1..N]$:	**if** $l \in L$: **return** $\perp$
$\quad k_j \xleftarrow{\$} \mathcal{K}$	$L \leftarrow L \cup \{l\}$
$\quad C_j \leftarrow \perp$	$l_j \leftarrow l$
$b' \leftarrow A$	$c \leftarrow \mathrm{E.enc}(k_j, l_j, m)$
return b'	**if** $b = 1 \wedge c \neq \perp$:
	$\quad c \xleftarrow{\$} \{0,1\}^{\lvert c \rvert}$
	$C_j \leftarrow c$
	return c

Figure 9: loE-IND-CPA-\$ game, Ahas access to oracle Oenc.

a value m. Both E.enc and E.dec are required to satisfy correctness (if $\mathrm{E.enc}(k, l, m) = c \neq \perp$, then $\mathrm{E.dec}(k, l, c) = m$) and tidiness (if $\mathrm{E.dec}(k, l, c) = m \neq \perp$, then $\mathrm{E.enc}(k, l, m) = c$).

loE security The security of a loE is defined as $\mathbf{Adv}^{\mathrm{loE}}_{A,N} = |\Pr[\text{loE-IND-CPA-}\$^0_{A,N} = 1] - \Pr[\text{loE-IND-CPA-}\$^1_{A,N} = 1]|$, where loE-IND-CPA-\$ is in figure 9. The user is only allowed one encryption query and locks may not repeat between users. Decryption queries are only allowed after the encryption.

loMAC A lock-based one-time use MAC is a deterministic algorithm M.mac that takes 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 tag t, or $\perp$. If, and only if, (k, l, m) is not in $\mathcal{K} \times \mathcal{L} \times \mathcal{M}$, t will be $\perp$.

loMAC security The security of a lock bases, one-time use PRF secure MAC is defined as $\mathbf{Adv}^{\mathrm{loMAC}}_{\mathrm{F},A,N} = |\Pr[\text{loMAC-PRF}^0_{A,N} = 1] - \Pr[\text{loMAC-PRF}^1_{A,N} = 1]|$, where loMAC-PRF is in figure 10. The user is only allowed one mac query and locks may not repeat between users. Verification queries are only allowed after the encryption. in contrast to the MAC-PRF form NRS, we need a verification oracle as we only allow one Omac query per user. The PRF will always return $\perp$ to match the loEA. (**Question: Is this enough explanation when the ideal vs attainable dilemma has been explained in section 4.3 already**)

5.2 Construction

Following NRS, three ways to construct this loAE are of interest, namely the ones following from the N1, N2 and N3 scheme. The schemes, adjusted to our setting, are in figure 11. NRS considers 17 more schemes but as no one of them has proven to be secure we will not consider those. The AE.enc and AE.dec calls corresponding to N1, N2 and N3 are in figure 12, 13 and 14 respectively.

5.3 Security Bounds

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

Game $\text{loMAC-PRF}^{b}_{A,N}$

$L \leftarrow \emptyset$
for $j \in [1..N]$:
  $k_j \xleftarrow{\$} \mathcal{K}$
  $T_j \leftarrow \perp$
$b' \leftarrow A$
return b'

Oracle $\text{Omac}(j, l, m)$

if $T_j \neq \perp$: **return** $\perp$
if $l \in L$: **return** $\perp$
$L \leftarrow L \cup \{l\}$
$l_j \leftarrow l$
$t \leftarrow \text{M.mac}(k_j, l_j, m)$
if $b = 1 \wedge t \neq \perp$:
  $t \xleftarrow{\$} \{0,1\}^{|t|}$
$T_j \leftarrow (m, t)$
return t

Oracle $\text{Ovrf}(j, m, t)$

if $T_j = \perp$: **return** $\perp$
if $(m, t) = T_j$: **return** $\perp$
if $b = 1$: **return** *false*
$t' \leftarrow \text{M.mac}(k_j, l_j, m)$
if $t = t'$
  return *true*
return *false*

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

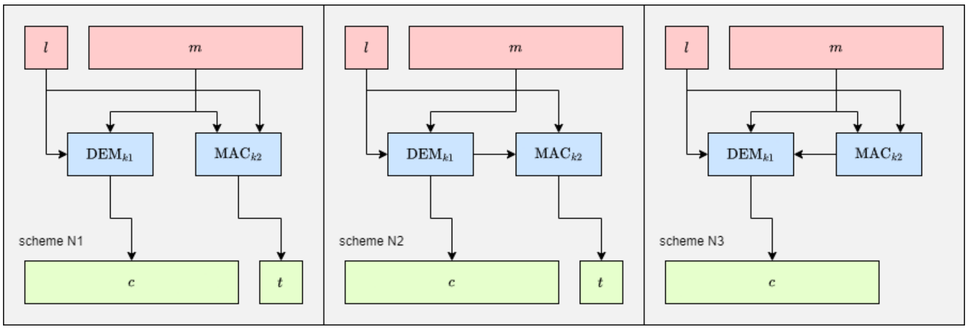

Figure 11: Adjusted N schemes from NRS

$\text{AE.enc}(k, l, m)$

$(k1, k2) \leftarrow k$
$c' \leftarrow \text{E.enc}(k1, l, m)$
$t \leftarrow \text{M.mac}(k2, l, m)$
$c \leftarrow (c', t)$
return c

$\text{AE.dec}(k, l, c)$

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

Figure 12: Calls based on N1

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

Figure 13: Calls based on N2

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

Figure 14: Calls based on N3

5.4 Comparison with Existing Alternatives

6 Use Cases

should consist of:

- possible use cases

6.1 PKE Schemes

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.

[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.

[3] 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 A

(**todo: elaborate more on this 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	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	-	A	a	data you want to authenticate but not encrypt
tag space	$\mathcal{C}$	-	$\mathcal{T}$	set off all possible tag options
tag	c	T	t	output of mac function
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
adversary	A	$\mathcal{A}$	A	the bad guy
random sampling	$\xleftarrow{\$}$	$\twoheadleftarrow$	$\xleftarrow{\$}$	get a random ellermetn form the set
result of randomized function	$\xleftarrow{\$}$	-	$\leftarrow$	get the result of a randomized function with given inputs