AE2: Upgrading AEAD Privacy

MIHIR BELLARE University of California San Diego

Based on: M. Bellare, R. Ng and B. Tackmann. Nonces are Noticed: AEAD Revisited. Crypto 2019. Cryptology ePrint Archive 2019/624.

TLDR:

TLDR:

AE1 / AEAD : (Nonce-based) Authenticated Encryption [RBBK01, R02]
In current standards
Schemes: GCM, OCB, …

TLDR:

AE1 / AEAD : (Nonce-based) Authenticated Encryption [RBBK01, R02]
In current standards
Schemes: GCM, OCB, …

This talk is concerned with PRIVACY. Not authenticity.

TLDR:

AE1 / AEAD : (Nonce-based) Authenticated Encryption [RBBK01, R02]
In current standards
Schemes: GCM, OCB, …

This talk is concerned with PRIVACY. Not authenticity.

AE1 is supposed to provide privacy for ANY choice of nonce.

TLDR:

AE1 / AEAD : (Nonce-based) Authenticated Encryption [RBBK01, R02]
In current standards
Schemes: GCM, OCB, …

This talk is concerned with PRIVACY. Not authenticity.

AE1 is supposed to provide privacy for ANY choice of nonce.

We explain that it doesn't. AE1 has two privacy weaknesses:

- It can fail to provide message privacy
- It can fail to provide meta-data privacy

TLDR:

AE1 / AEAD : (Nonce-based) Authenticated Encryption [RBBK01, R02]
In current standards
Schemes: GCM, OCB, …

This talk is concerned with PRIVACY. Not authenticity.

AE1 is supposed to provide privacy for ANY choice of nonce.

We explain that it doesn't. AE1 has two privacy weaknesses:

- It can fail to provide message privacy
- It can fail to provide meta-data privacy

These weaknesses arise due to the way nonces are treated.
These weaknesses are present in ALL AE1 schemes.

TLDR:

AE1 / AEAD : (Nonce-based) Authenticated Encryption [RBBK01, R02]
In current standards
Schemes: GCM, OCB, …

This talk is concerned with PRIVACY. Not authenticity.

AE1 is supposed to provide privacy for ANY choice of nonce.

We explain that it doesn't. AE1 has two privacy weaknesses:

- It can fail to provide message privacy
- It can fail to provide meta-data privacy

These weaknesses arise due to the way nonces are treated.
These weaknesses are present in ALL AE1 schemes.

There are two solutions:

- Specify and mandate "SAFE" nonce choices (hard and error-prone)
- Switch to **AE2** (pretty easy for new schemes)

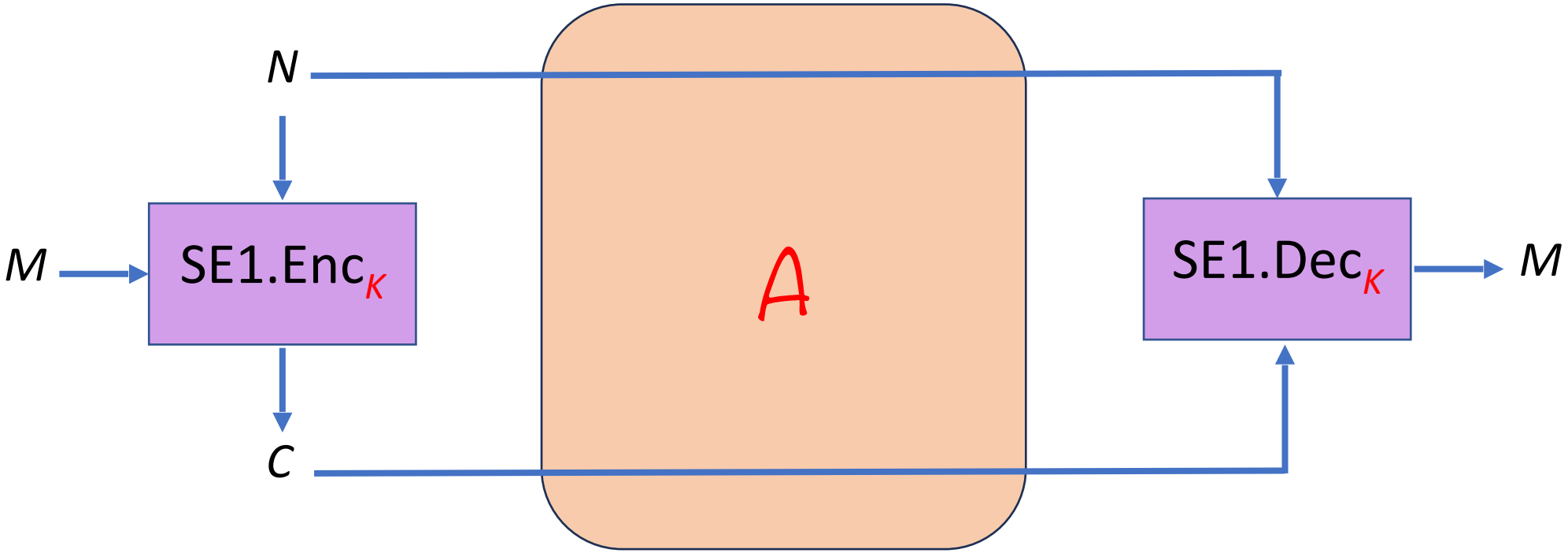
AE1 / AEAD : Authenticated Encryption Today

M : Message
N : Nonce
C : Ciphertext
K : Key

Scheme SE1 specifies encryption algorithm SE1.Enc and decryption algorithm SE1.Dec.
Note that the decryption algorithm needs and gets the nonce *N* as input as per the AE1 syntax.

Example Schemes: GCM, OCB, …

AE1 / AEAD : Authenticated Encryption Today



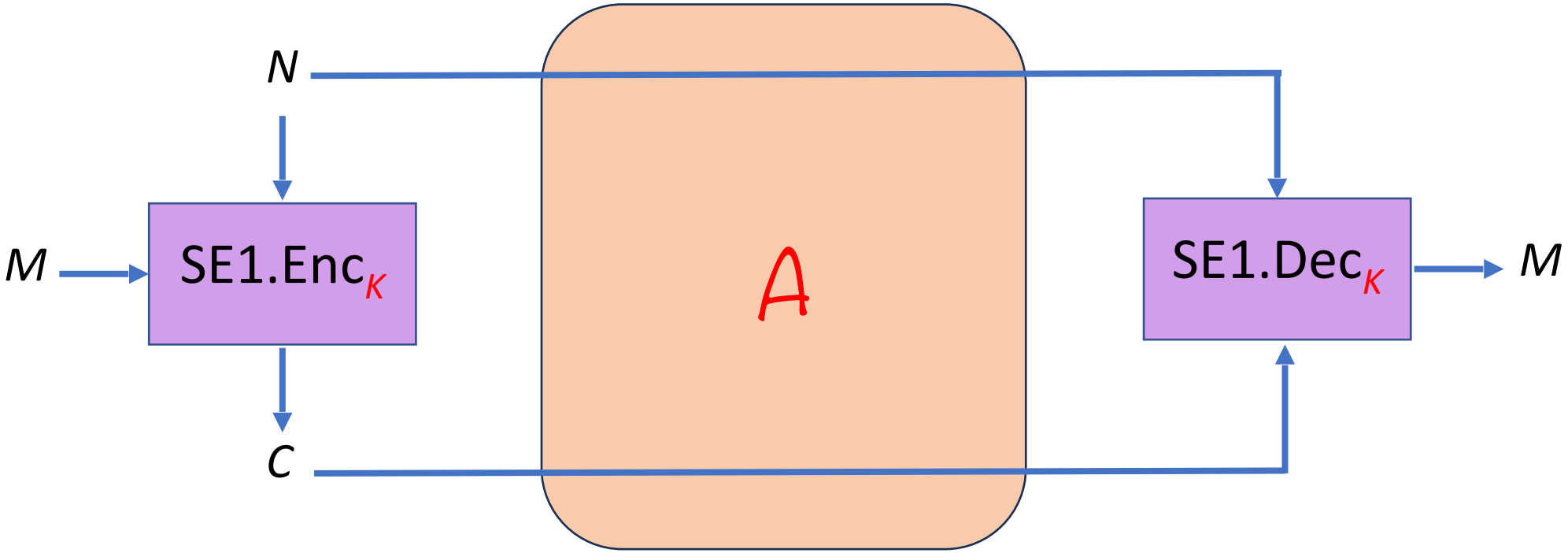
M : Message
N : Nonce
C : Ciphertext
K : Key

Scheme SE1 specifies encryption algorithm SE1.Enc and decryption algorithm SE1.Dec.
Note that the decryption algorithm needs and gets the nonce *N* as input as per the AE1 syntax.

Example Schemes: GCM, OCB, …

Security goal: Privacy of message *M* and authenticity of *C*

AE1 / AEAD : Authenticated Encryption Today



M : Message
N : Nonce
C : Ciphertext
K : Key

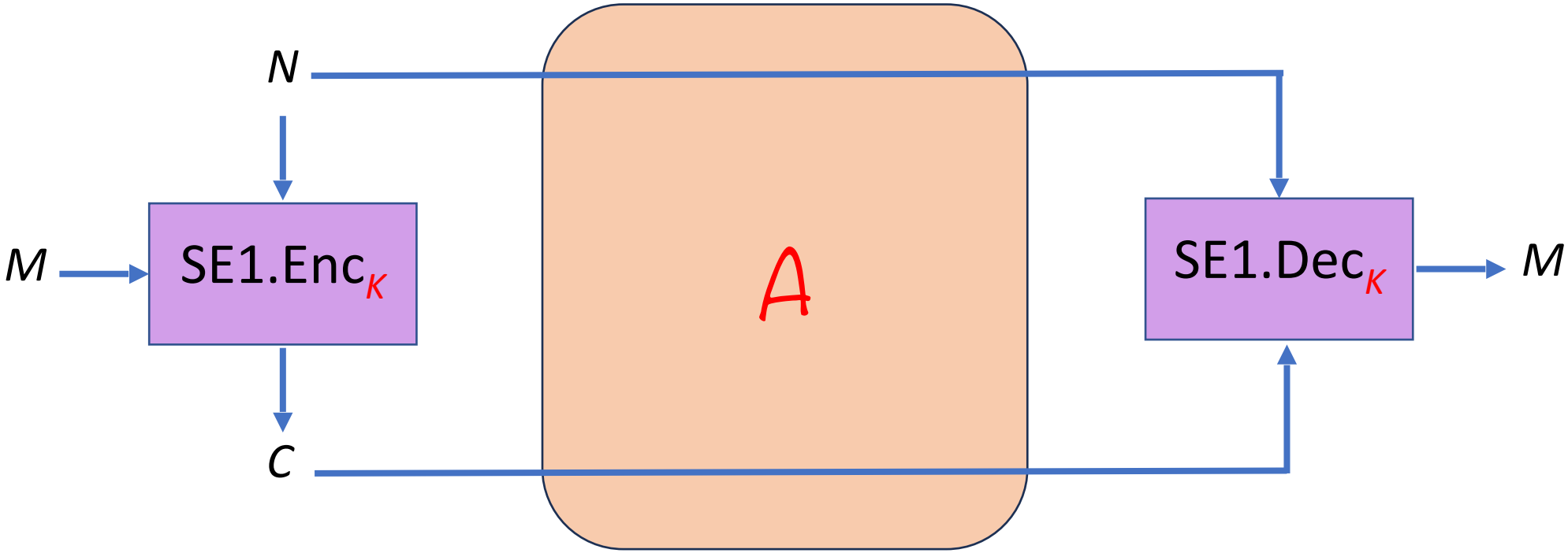
Scheme SE1 specifies encryption algorithm SE1.Enc and decryption algorithm SE1.Dec.
Note that the decryption algorithm needs and gets the nonce *N* as input as per the AE1 syntax.

Example Schemes: GCM, OCB, …

Security goal: Privacy of message *M*

Our concern is privacy so we drop the associated data

AE1 / AEAD : Authenticated Encryption Today



M : Message
N : Nonce
C : Ciphertext
K : Key

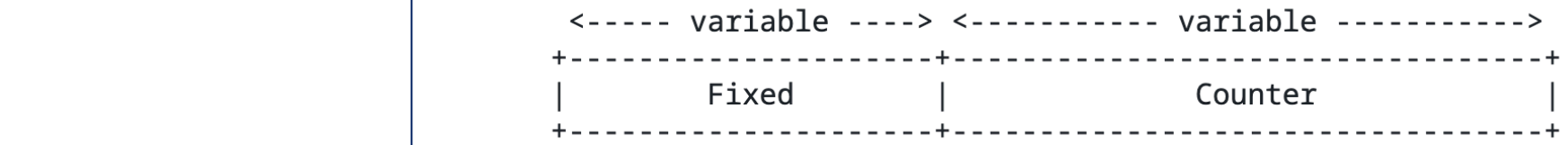
AE1 allows ANY choice of nonce.
The only restriction is that a nonce should not be reused across different encryptions.

3.1. Requirements on Nonce Generation

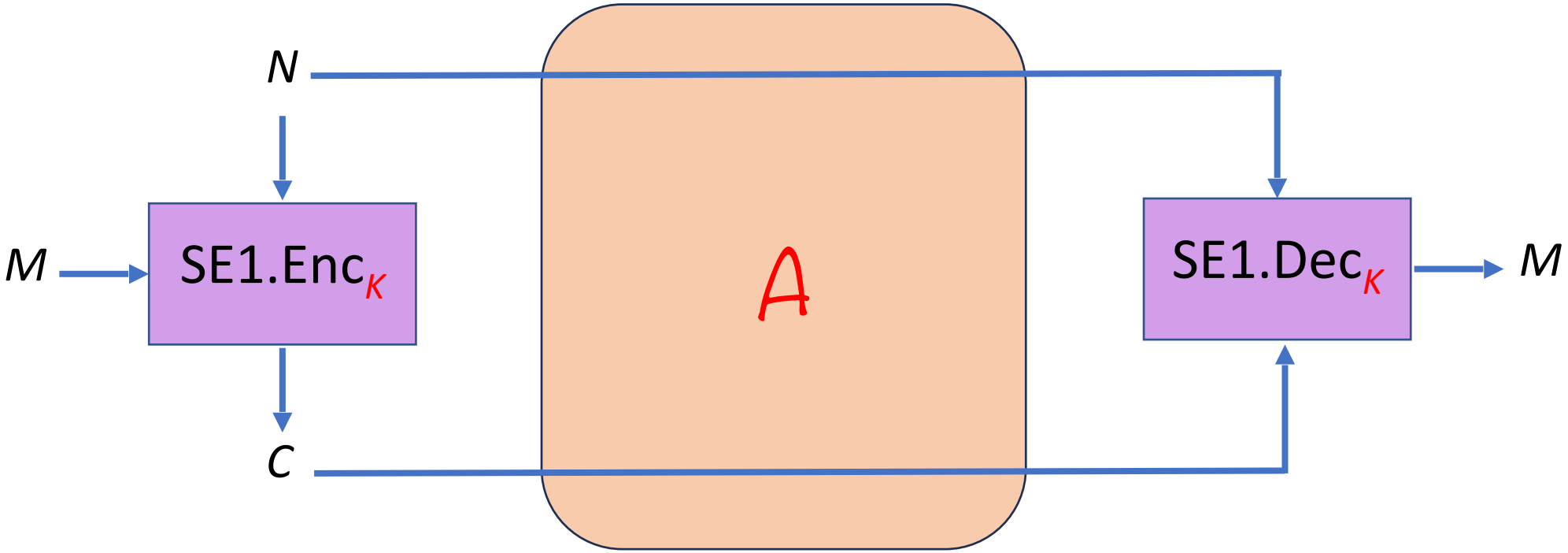
It is essential for security that the nonces be constructed in a manner that respects the requirement that each nonce value be distinct for each invocation of the authenticated encryption operation, for any fixed value of the key. In this section, we call

3.2. Recommended Nonce Formation

The following method to construct nonces is RECOMMENDED. The nonce is formatted as illustrated in Figure 1, with the initial octets consisting of a Fixed field, and the final octets consisting of a Counter field. For each fixed key, the length of each of these fields, and thus the length of the nonce, is fixed. Implementations SHOULD support 12-octet nonces in which the Counter field is four octets long.



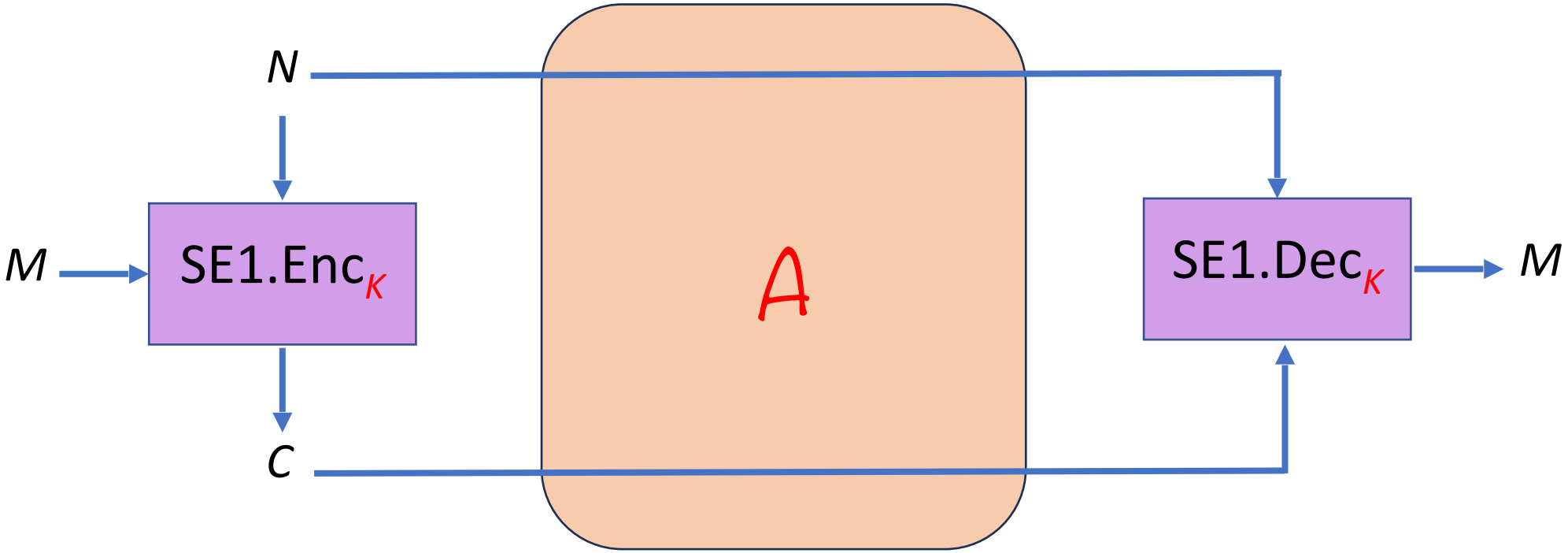
AE1 / AEAD : Authenticated Encryption Today



M : Message
N : Nonce
C : Ciphertext
K : Key

The AE1 claim: Privacy of message *M* is provided REGARDLESS of the choice of the nonce.

AE1 / AEAD : Authenticated Encryption Today

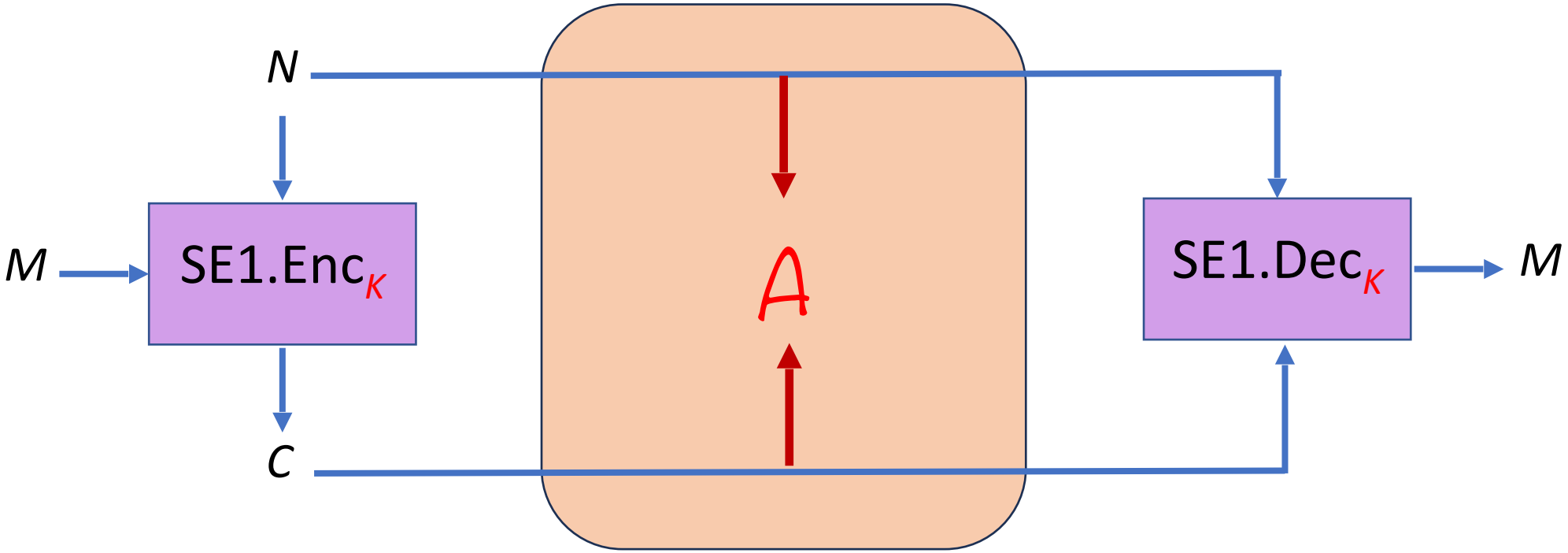


M : Message
N : Nonce
C : Ciphertext
K : Key

The AE1 claim: Privacy of message *M* is provided REGARDLESS of the choice of the nonce.

Yet it should be obvious from the above picture that this claim is not true.

AE1 / AEAD : Authenticated Encryption Today



M : Message
N : Nonce
C : Ciphertext
K : Key

The AE1 claim: Privacy of message M is provided REGARDLESS of the choice of the nonce.

Yet it should be obvious from the above picture that this claim is not true.

Why? Because there exist ``UNSAFE'' choices of nonces.

Namely, nonces that carry information about M. For example:

- N = SHA256(M), or even
- $N = M$

An example

Suppose we want to encrypt distinct messages M_1, M_2, …

An example

Suppose we want to encrypt distinct messages $M_1, M_2, \ldots$

A convenient choice of nonces is N_1 = SHA256(M_1), N_2 = SHA256(M_2), …

Let C_1 = SE1.Enc(K,N_1,M_1), C_2 = SE1.Enc(K,N_2,M_2), … be the corresponding ciphertexts, where SE1 is AE1-secure

These nonces will be distinct, hence are allowed.

So our understanding of AE1 is that privacy of $M_1, M_2, \ldots$ should be provided

An example

Suppose we want to encrypt distinct messages M_1, M_2, …

A convenient choice of nonces is N_1 = SHA256(M_1), N_2 = SHA256(M_2), …

Let C_1 = SE1.Enc(K,N_1,M_1), C_2 = SE1.Enc(K,N_2,M_2), … be the corresponding ciphertexts, where SE1 is AE1-secure

These nonces will be distinct, hence are allowed.

So our understanding of AE1 is that privacy of M_1, M_2, … should be provided

But privacy of M_1, M_2, … clearly isn't provided.

The adversary gets N_1 and C_1.

If M_1 has low entropy, a brute-force attack recovers it from N_1 = SHA256(M_1)

An example

Suppose we want to encrypt distinct messages M_1, M_2, …

A convenient choice of nonces is N_1 = SHA256(M_1), N_2 = SHA256(M_2), …

Let C_1 = SE1.Enc(K,N_1,M_1), C_2 = SE1.Enc(K,N_2,M_2), … be the corresponding ciphertexts, where SE1 is AE1-secure

These nonces will be distinct, hence are allowed.

So our understanding of AE1 is that privacy of M_1, M_2, … should be provided

But privacy of M_1, M_2, … clearly isn't provided.

The adversary gets N_1 and C_1.

If M_1 has low entropy, a brute-force attack recovers it from N_1 = SHA256(M_1)

But doesn't this contradict the security guarantee of the AE1 definition?

No. Because in the latter the adversary is given ONLY C_1, C_2, …

Nonces are assumed to be magically communicated to the recipient.

In reality however, and as per the RFCs, nonces will be sent with the ciphertexts, allowing the attack.

An example

Suppose we want to encrypt distinct messages M_1, M_2, …

A convenient choice of nonces is N_1 = SHA256(M_1), N_2 = SHA256(M_2), …

Let C_1 = SE1.Enc(K,N_1,M_1), C_2 = SE1.Enc(K,N_2,M_2), … be the corresponding ciphertexts, where SE1 is AE1-secure

These nonces will be distinct, hence are allowed.

So our understanding of AE1 is that privacy of M_1, M_2, … should be provided

But privacy of M_1, M_2, … clearly isn't provided.

The adversary gets N_1 and C_1.

If M_1 has low entropy, a brute-force attack recovers it from N_1 = SHA256(M_1)

But doesn't this contradict the security guarantee of the AE1 definition?

No. Because in the latter the adversary is given ONLY C_1, C_2, …

Nonces are assumed to be magically communicated to the recipient.

In reality however, and as per the RFCs, nonces will be sent with the ciphertexts, allowing the attack.

So should we not use this choice of nonces?

They are nice, convenient choices. They SHOULD work.

And with AE2, they WILL work.

More unsafe nonces?

Nonces are meta-data. Recommended and used choices such as counters, device identities, disk-sector numbers, packet headers, ... can reveal information about the system and identity of the sender.

More unsafe nonces?

Nonces are meta-data. Recommended and used choices such as counters, device identities, disk-sector numbers, packet headers, ... can reveal information about the system and identity of the sender.

RFC 5116

When there are multiple devices performing encryption ... use a nonce format that contains a field that is distinct for each one of the devices.

More unsafe nonces?

Nonces are meta-data. Recommended and used choices such as counters, device identities, disk-sector numbers, packet headers, ... can reveal information about the system and identity of the sender.

RFC 5116

When there are multiple devices performing encryption ... use a nonce format that contains a field that is distinct for each one of the devices.

But this nonce will reveal the device identity

More unsafe nonces?

Nonces are meta-data. Recommended and used choices such as counters, device identities, disk-sector numbers, packet headers, ... can reveal information about the system and identity of the sender.

RFC 5116

When there are multiple devices performing encryption ... use a nonce format that contains a field that is distinct for each one of the devices.

But this nonce will reveal the device identity

[Ro13, Real World Crypto]

AE1-secure NBE1 provides anonymity because the ciphertext is indistinguishable from random.

More unsafe nonces?

Nonces are meta-data. Recommended and used choices such as counters, device identities, disk-sector numbers, packet headers, ... can reveal information about the system and identity of the sender.

RFC 5116

When there are multiple devices performing encryption ... use a nonce format that contains a field that is distinct for each one of the devices.

But this nonce will reveal the device identity

[Ro13, Real World Crypto]

AE1-secure NBE1 provides anonymity because the ciphertext is indistinguishable from random.

But the nonce can violate anonymity

Choices

Choice 1: Standards should mandate SAFE nonce choices.

- Applications and implementations burdened by having to ensure their choices are safe
- No clear definition of, or agreement about, what is ``SAFE''
- Error-prone

Choice 2: Standardize and use AE2.

- AE2 hides the nonce, so now **ALL** nonce choices are SAFE!
- Applications and implementors can TRULY use ANY (non-repeating) choices.
- AE2 schemes are cheap

FMO : Frequently Made Objections

Some answers

This is SILLY. There really isn’t a problem.

FMO : Frequently Made Objections

This is SILLY. There really isn't a problem.

AE1 schemes have PROOFs showing they meet formal DEFINITIONS of privacy. So how can they not provide privacy?

Some answers

FMO : Frequently Made Objections

This is SILLY. There really isn't a problem.

AE1 schemes have PROOFs showing they meet formal DEFINITIONS of privacy. So how can they not provide privacy?

Some answers

The problem is the DEFINITION.

FMO : Frequently Made Objections

This is SILLY. There really isn't a problem.

AE1 schemes have PROOFs showing they meet formal DEFINITIONS of privacy. So how can they not provide privacy?

I do not use unsafe nonces.

Some answers

The problem is the DEFINITION.

FMO : Frequently Made Objections

This is SILLY. There really isn't a problem.

AE1 schemes have PROOFs showing they meet formal DEFINITIONS of privacy. So how can they not provide privacy?

I do not use unsafe nonces.

Some answers

The problem is the DEFINITION.

Great!
But they are ALLOWED.
By research papers and standards.

FMO : Frequently Made Objections

This is SILLY. There really isn't a problem.

AE1 schemes have PROOFs showing they meet formal DEFINITIONS of privacy. So how can they not provide privacy?

I do not use unsafe nonces.

Have you found any attack in the wild?
No reason to worry until then.

Some answers

The problem is the DEFINITION.

Great!
But they are ALLOWED.
By research papers and standards.

FMO : Frequently Made Objections

This is SILLY. There really isn't a problem.

AE1 schemes have PROOFs showing they meet formal DEFINITIONS of privacy. So how can they not provide privacy?

I do not use unsafe nonces.

Have you found any attack in the wild?
No reason to worry until then.

Some answers

The problem is the DEFINITION.

Great!
But they are ALLOWED.
By research papers and standards.

Be proactive, not reactive.
Prevention is better than cure?

FMO : Frequently Made Objections

This is SILLY. There really isn't a problem.

AE1 schemes have PROOFs showing they meet formal DEFINITIONS of privacy. So how can they not provide privacy?

I do not use unsafe nonces.

Have you found any attack in the wild?
No reason to worry until then.

We can trust applications, libraries and implementors to make safe nonce choices.

Some answers

The problem is the DEFINITION.

Great!
But they are ALLOWED.
By research papers and standards.

Be proactive, not reactive.
Prevention is better than cure?

FMO : Frequently Made Objections

This is SILLY. There really isn't a problem.

AE1 schemes have PROOFs showing they meet formal DEFINITIONS of privacy. So how can they not provide privacy?

I do not use unsafe nonces.

Have you found any attack in the wild?
No reason to worry until then.

We can trust applications, libraries and implementors to make safe nonce choices.

Some answers

The problem is the DEFINITION.

Great!
But they are ALLOWED.
By research papers and standards.

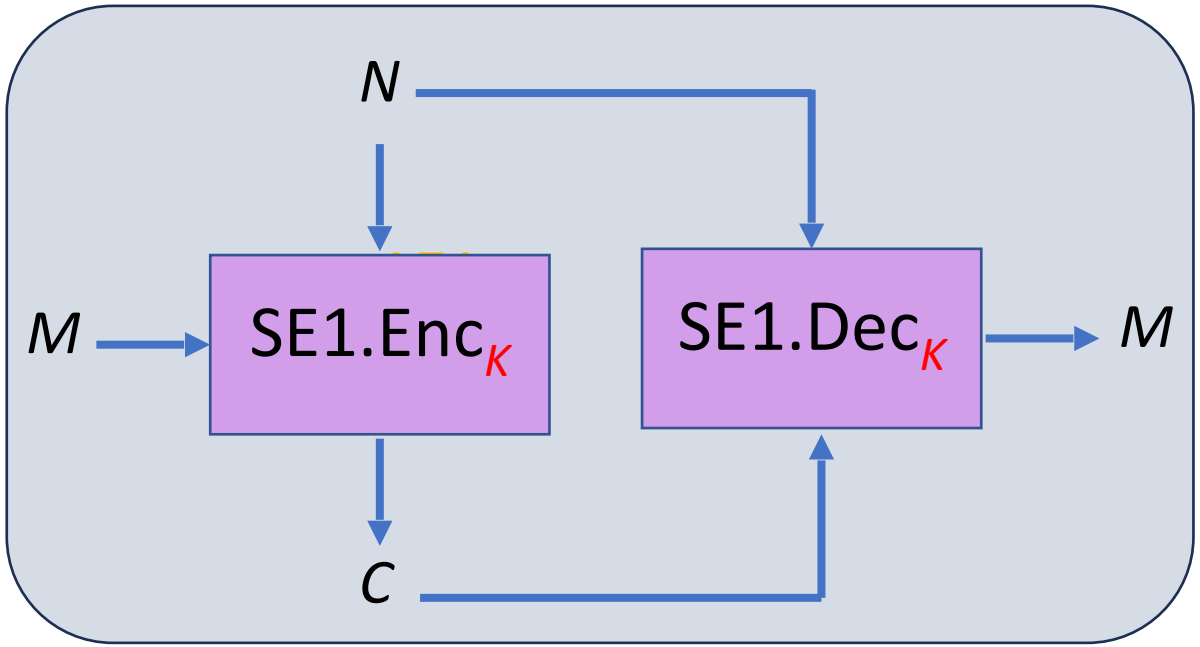
Be proactive, not reactive.
Prevention is better than cure?

We should reduce the chance of error
We should make implementors' lives easier

SYNTAX of an authenticated encryption scheme

SYNTAX of an authenticated encryption scheme

AE1

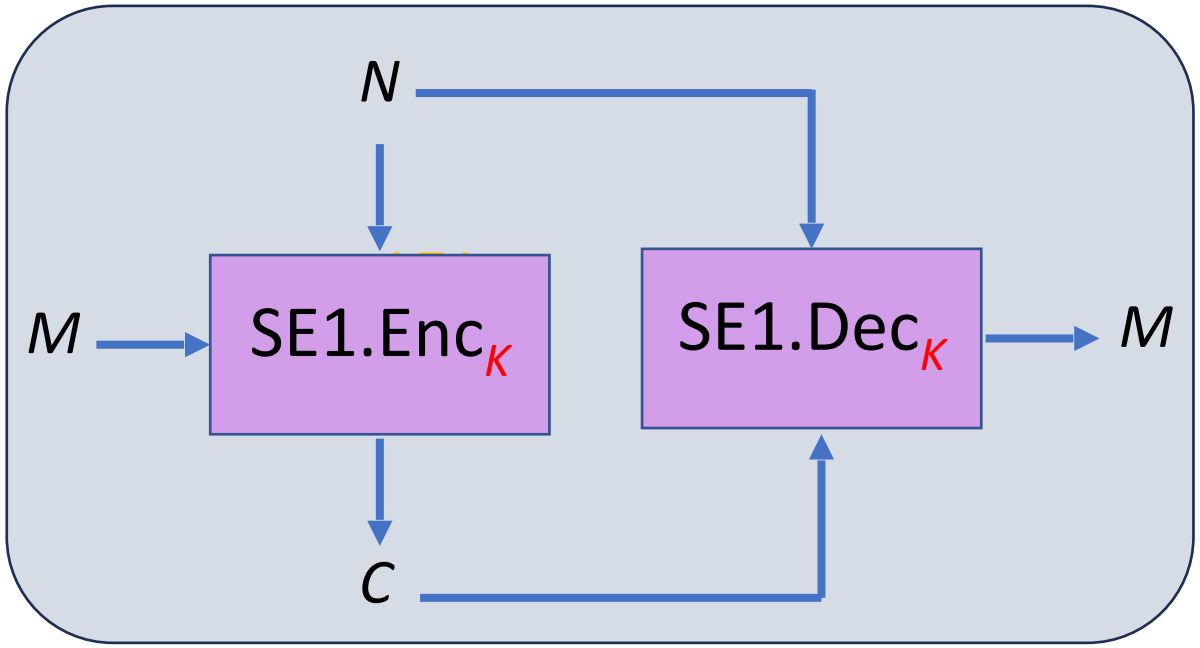


Nonce *N* is **INPUT** to **BOTH** SE1.Enc and SE1.Dec

Enc and Dec are DETERMINISTIC algorithms

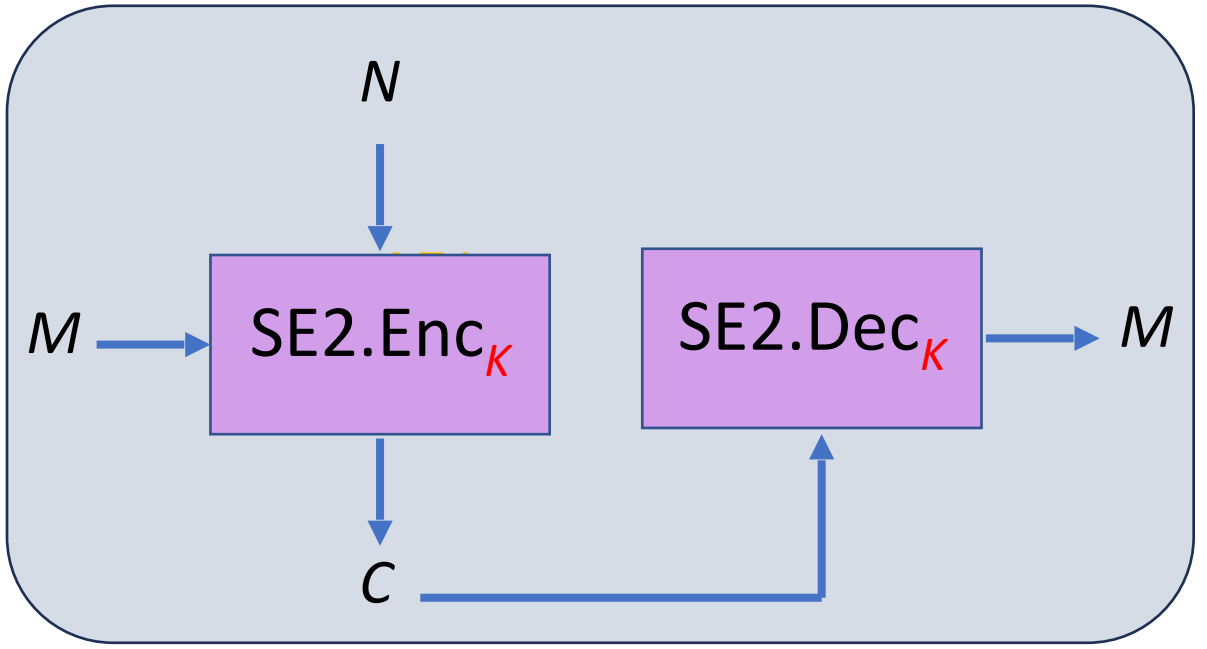
SYNTAX of an authenticated encryption scheme

AE1



Nonce *N* is **INPUT** to **BOTH** SE1.Enc and SE1.Dec

AE2



Nonce *N* is **INPUT** to SE2.Enc but **NOT** to SE2.Dec

Enc and Dec are DETERMINISTIC algorithms

Recall AE1 security [RBBK01,Ro02]

Adversary has black-box access to Enc and Dec oracles. But it **cannot repeat a nonce to Enc**

Note that the Dec oracle gets a nonce as explicit input from the adversary

AE1 definitions assume the nonce is sent securely and out-of-band to the receiver
But in practice, it is sent in the clear along with the ciphertext, unless the receiver already has it.

This is the notion of security that GCM, OCB and some CAESAR candidates have been proven secure under.

AE2 security [BNT19]

Adversary has black-box access to Enc and Dec oracles. But it **cannot repeat a nonce to Enc**

Note that the Dec oracle DOES NOT GET a nonce input from the adversary

AE2 do not assume nonces are sent out of band. Decryption must be possible given ONLY the ciphertext *C*.

[BNT19] give schemes meeting this notion.

Remarks

The key change in moving from AE1 to AE2 is in the syntax: Decryption no longer gets the nonce as input.

The AE2 security definition then ensures that both the message AND the nonce are hidden.

Remarks

The key change in moving from AE1 to AE2 is in the syntax: Decryption no longer gets the nonce as input.

The AE2 security definition then ensures that both the message AND the nonce are hidden.

Q: Is GCM AE2 secure?

A: No.

More precisely, the question does not make sense since GCM does not have the AE2 syntax.

Example revisited

Suppose we want to encrypt distinct messages M_1, M_2, …

Example revisited

Suppose we want to encrypt distinct messages $M_1, M_2, \ldots$

A convenient choice of nonces is N_1 = SHA256(M_1), N_2 = SHA256(M_2), …

Let C_1 = SE2.Enc(K,N_1,M_1), C_2 = SE2.Enc(K,N_2,M_2), … be the corresponding ciphertexts, where SE2 is AE2-secure.

These nonces will be distinct, hence are allowed.

Example revisited

Suppose we want to encrypt distinct messages $M_1, M_2, \ldots$

A convenient choice of nonces is N_1 = SHA256(M_1), N_2 = SHA256(M_2), …

Let C_1 = SE2.Enc(K,N_1,M_1), C_2 = SE2.Enc(K,N_2,M_2), … be the corresponding ciphertexts, where SE2 is AE2-secure.

These nonces will be distinct, hence are allowed.

AE2 provides security even with this choice of nonces.

The brute-force attack no longer works.

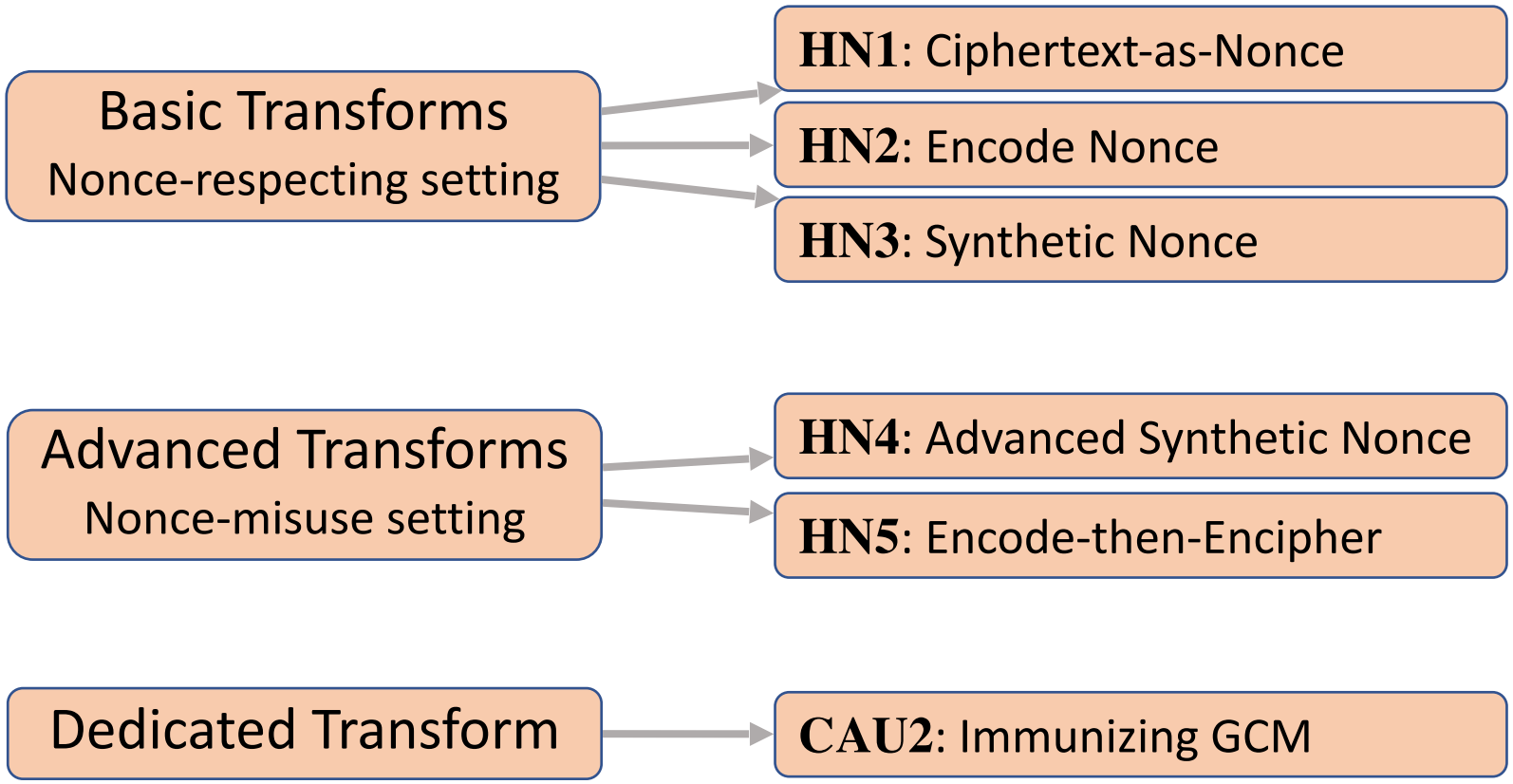
AE2 schemes from [BNT19]

[BNT19] construct AE2 schemes from AE1 schemes, rather than from scratch, because

- There are many efficient, optimized, standardized, deployed AE1 schemes
- Application designers can now easily add nonce-hiding to them

The [BNT19] constructions have minimal (optimal) bandwidth overhead and low (like, one block-cipher call) computational overhead.

AE2 schemes from [BNT19]



Extends AE2-security to the nonce-misuse setting, as [RS06] extended AE1-security.

CAU2 immunizes GCM with lower overhead than generic transforms.

Schemes for AE2 standards?

Dedicated designs can be even cheaper than the [BNT19] transforms.

The upcoming **Flex** scheme is one such.

We hope to see more proposals!

Related work

Bernstein groups.google.com cryptographic competitions forum posting on how communicated nonces can compromise privacy, and constructions to address it, May 2013. Elements reflected in his CAESAR call, PMN and SMN. Formalized by [NRS, ePrint 2013] as AE5.

[ADL17] use the AE2 syntax as a technical step in their RUP designs.

[ChRo19] study anonymous AE, which, like AE2, hides the nonce.

Summary

AE1 (AEAD) has been presented, and understood, as providing message privacy for ANY choice of nonce. But it doesn't.

AE1 schemes like GCM, OCB, CAESAR, in particular, do not provide security for arbitrary nonces.

The issue is that in-the-clear communicated nonces can violate the very message privacy encryption is trying to ensure.

In-the-clear communicated nonces also expose meta-data about the sender.

AE2 addresses this by hiding the nonce in addition to the message.

AE2 can be built quite cheaply.

An AE2 scheme may be a valuable option for future standards.