Introduction to Cryptography

F. Koeune – O. Pereira

MAT2450 – Lecture 4

Message authenticity

So far, we have only considered the confidentiality aspect of communication
Message authenticity is an important aspect as well

- Does this message really originate from Alice?
- Am I sure it has not been modified?

Useful for

- Financial transactions
- Critical orders
- ...

But do we need something new?

If the message is encrypted, the attacker cannot get a clue about its content

- Is this not sufficient to prevent him from tampering with it?

Consider the CPA-secure scheme we discussed last week:

- Encryption $\langle r, s \rangle := \langle r, F_k(r) \oplus m \rangle$
- Decryption: $m := s \oplus F_k(r)$

What happens if an attacker flips one bit of s?

- Attacker can induce arbitrary modifications on the message
- Similar things happen with all the encryption schemes/modes that we described!

Authenticity

Two flavors to authenticity:

1. Integrity: is this message authentic? ($\perp$ confidentiality)
2. Non malleability: could this plaintext result from a controlled manipulation of a ciphertext?

What can we hope to achieve?

Impossible to prevent an adversary from tampering with the message

But we can at least make modifications detectable

- (stronger than error detection codes!)

Message authentication codes (MAC): principle

To perform authenticated communication

- Alice and Bob agree (once and for all) on a common key k
- To send message m, Alice
  - Uses a generation algorithm to compute a tag t from m and k
  - Sends m together with t
- On reception, Bob
  - Uses a verification algorithm to check whether t is a valid tag for m and k
  - Accepts m iff the answer is positive

Message authentication code (MAC): definition

A message authentication code is a triple $\Pi := \langle \mathrm{Gen}, \mathrm{Mac}, \mathrm{Vrfy} \rangle$

- Gen: probabilistically selects a key $k \in \mathcal{K}$
- Mac: on input $m \in \mathcal{M}, k \in \mathcal{K}$ computes a tag $t \leftarrow \mathrm{Mac}_k(m)$
- Vrfy: on input m, t, k, outputs a bit $b := \mathrm{Vrfy}_k(m, t)$, with $b = 1$ meaning valid and $b = 0$ meaning invalid

Remarks:

- Mac may be probabilistic
- But we can assume without loss of generality that Vrfy is deterministic

MAC correctness

For every valid m, k, it must hold that

$$\mathrm{Vrfy}_k(m, \mathrm{Mac}_k(m)) = 1$$

i.e. "a legitimately produced MAC is always considered valid"

Defining MAC security

Intuition: no PPT adversary should be able to generate a valid tag on any "new" message, even if he can

- observe valid MAC tags
- obtain valid MAC tags for messages that he chooses

As before, this will be modeled by providing the adversary with an oracle

MAC security: definition

Given $\Pi := \langle \mathrm{Gen}, \mathrm{Mac}, \mathrm{Vrfy} \rangle$, and adversary $\mathcal{A}$, define the following experiment $\mathsf{MacForge}_{\mathcal{A},\Pi}(n)$:

1. Choose $k \leftarrow \mathrm{Gen}(1^n)$
2. $\mathcal{A}$ is given oracle access to $\mathrm{Mac}_k(\cdot)$. Let $\mathcal{Q}$ denote the set of queries made to $\mathrm{Mac}_k(\cdot)$
3. $\mathcal{A}$ outputs a pair (m, t)
4. Define $\mathsf{MacForge}_{\mathcal{A},\Pi}(n) := 1$ iff $\mathrm{Vrfy}_k(m, t) = 1$ and $m \notin \mathcal{Q}$

MAC security: definition

$\Pi := \langle \mathrm{Gen}, \mathrm{Mac}, \mathrm{Vrfy} \rangle$ is *existentially unforgeable under an adaptive chosen-message attack* (EUF-CMA) if $\forall$ PPT $\mathcal{A}$, $\exists\ \epsilon$:

$$\Pr[\mathsf{MacForge}_{\mathcal{A},\Pi}(n) = 1] \leq \epsilon(n)$$

Quizz

Why

$$\Pr[\mathsf{MacForge}_{\mathcal{A},\Pi}(n) = 1] \leq \epsilon(n)$$

and not

$$\Pr[\mathsf{MacForge}_{\mathcal{A},\Pi}(n) = 1] \leq \frac{1}{2} + \epsilon(n) \qquad ?$$

1. Because of the condition $m \notin \mathcal{Q}$
2. Because there is no "easy guess" for the adversary

Answer: 2

- Here the adversary does not have a set of two options among which to choose his final answer

More on EUF-CMA security

Are we asking for too much?

- Definition includes *existential* attacks:
  $\mathcal{A}$ wins if he forges anything, even something meaningless

But we want to be application independent (see encryption)

And is it enough?

- Does not protect from replay attacks:
  Resubmit same \$100 money order 100 times

But hard to solve in a non-interactive way:
need challenge-response or timestamping or ...

Quizz

Suppose $\mathcal{A}$ submits m, gets t, and then outputs (m', t) such that $m \neq m'$ and $\mathrm{Vrfy}_k(m', t) = 1$.

1. Does he win the security game?
2. Is this something we want to capture?

$\Rightarrow$ Yes, it would be considered as an attack

- And would have actual consequences "in real life"

Quizz

Suppose $\mathcal{A}$ submits m, gets t, and then outputs (m, t') such that $t \neq t'$ and $\mathrm{Vrfy}_k(m, t') = 1$.

1. Does he win the security game?
2. Is this a problem?

Notion of *strong* unforgeability:

- A stronger notion than EUF-CMA
- $\mathcal{A}$ wins as soon as he produces new valid (m, t) pair

Constructing secure MACs

We will use pseudorandom functions

Define $\Pi := \langle \mathrm{Gen}, \mathrm{Mac}, \mathrm{Vrfy} \rangle$ as:

- Gen: choose random $k \leftarrow \{0,1\}^n$
- Mac: on input $m, k \in \{0,1\}^n$, output $t := F_k(m)$
- Vrfy: on input $k, m, t \in \{0,1\}^n$ output 1 iff $t = F_k(m)$

Security of this PRF-based MAC

Theorem: if F is a PRF, this MAC is EUF-CMA secure

Proof: in two steps:

- Prove that the scheme is secure if F_k is replaced by a truly random function f
- Prove that if the scheme (with F_k) were insecure, we could distinguish F_k from a truly random function

$\Rightarrow$ Very similar to the CPA security proof

$\Rightarrow$ Will be proposed as part of next exercise session

Extension to variable-length messages

As F_k is a length-preserving function, our construction only works for messages such that $|m| = |k|$

How can we extend this to variable-length messages?

Consider $m = m_1 || \ldots || m_d$ such that $|m_i| = |k|$

Let us try some simple constructions

Idea 1: XOR blocks

Define $t := \mathrm{Mac}_k(\bigoplus_i m_i)$

Does this yield a secure MAC?

No: $\mathcal{A}$ could easily forge a valid tag for m', where

- $m'_1 := m_1 \oplus 1$
- $m'_2 := m_2 \oplus 1$
- $m'_i := m_i$ for $3 \leq i \leq d$

$\mathrm{Vrfy}_k(m', t) = 1$

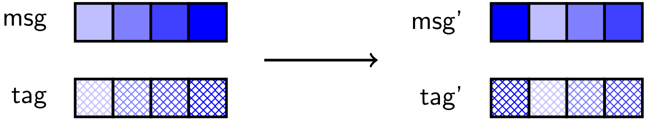
Idea 2: Authenticate blocks separately

Define $t := \langle t_1, \ldots, t_d \rangle$ with $t_i := \mathrm{Mac}_k(m_i)$

Quizz: Does this yield a secure MAC?

- OK, too easy: can you show an example?

- $\mathcal{A}$ can easily build
  - $t' := \langle t_d, t_1, \ldots, t_{d-1} \rangle$ as a valid tag for $m' := m_d || m_1 \ldots || m_{d-1}$
  - $t'' := \langle t_1, t_3 \rangle$ as a valid tag for $m'' := m_1 || m_3$
  - ...



Idea 3: Add a sequence number to idea 2

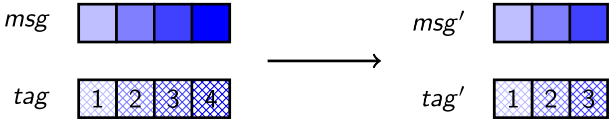
To prevent block reordering, add a sequence number to each block

Define $t := \langle t_1 \ldots t_d \rangle$ with $t_i := \mathrm{Mac}_k(i||m_i)$

Idea 3: Add a sequence number to idea 2

Prevents block reordering, but

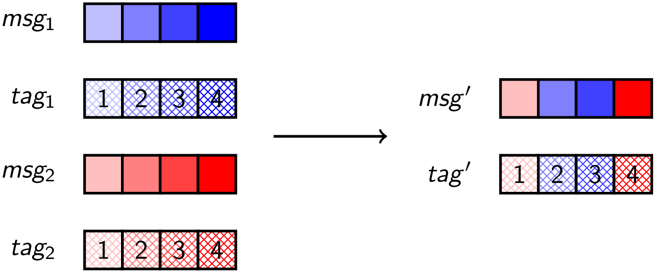
- Message truncation still possible



- Message combination still possible

Idea 3: Add a sequence number to idea 2

- Message combination still possible



A secure solution

To prevent previous attacks, we would need partial tags to depend on

$$t_i := \mathrm{Mac}_k(r||l||i||m_i)$$

- Message block (of course)
- Block number (to prevent block reordering)
- Full message length (to prevent message truncation)
- Unique, random identifier (to prevent message combination)

The full solution

Suppose we have a secure fixed-length MAC

$$\Pi' := \langle \mathrm{Gen}', \mathrm{Mac}', \mathrm{Vrfy}' \rangle$$

We will define a variable-length MAC

$$\Pi := \langle \mathrm{Gen}, \mathrm{Mac}, \mathrm{Vrfy} \rangle$$

as follows

The full solution

- Gen: choose random $k \leftarrow \{0,1\}^n$
- Mac: on input $k \in \{0,1\}^n$ and $m \in \{0,1\}^*$ of length $l < 2^{\frac{n}{4}}$
  - Parse m into blocks $m_1, \ldots, m_d$ of length $\frac{n}{4}$ each (pad with 0's if necessary)
  - Choose random $r \leftarrow \{0,1\}^{\frac{n}{4}}$
  - Compute $t_i := \mathrm{Mac}_k(r||l||i||m_i)$ for $1 \leq i \leq d$, with $|r| = |l| = |i| = \frac{n}{4}$
  - Output $t := \langle r, t_1, \ldots, t_d \rangle$
- Vrfy: on input $k, m, t = \langle r, t_1, \ldots, t_{d'} \rangle$,
  - Parse m into blocks $m_1, \ldots, m_d$ of length $\frac{n}{4}$ each
  - Output 1 iff $d = d'$ and, $\forall \quad 1 \leq i \leq d$, $\mathrm{Vrfy}'_k(r||l||i||m_i, t_i)$

A secure solution

Theorem: If Π' is a secure fixed-length MAC for messages of length n, then Π is a MAC that is existentially unforgeable under an adaptive chosen-message attack

Proof: By reduction:

- Show that any adversary breaking Π can be turned into an adversary breaking Π'

Efficiency

The above construction is provably secure but pretty inefficient

For a message of length $l = n.v$

- Pseudorandom function has to be applied $4.v$ times
- Tag is $4.l + \frac{n}{4}$ bit long

Can we do better?

Yes

- Some solutions based on block/stream ciphers and modes of operation
- Other based on hash functions (next chapter)

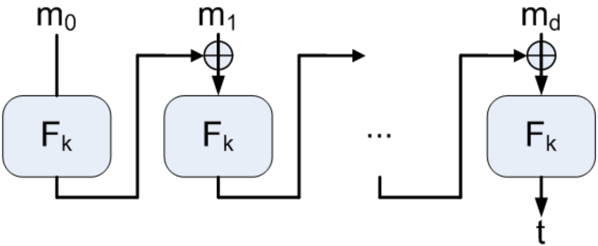
CBC-MAC

A construction similar to the CBC mode of encryption
Provably secure
Efficient

- For a d-block message, d executions of the PRF
- Tag is 1 block long

$\Rightarrow$ Quite common in practice (including in TLS ≥ 1.2)

CBC-MAC for fixed-length messages



Remark: if intermediary values are output, scheme is not secure any more!

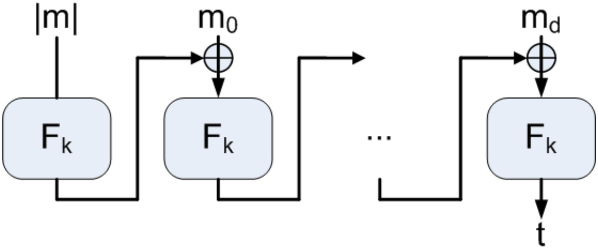
CBC-MAC for fixed-length messages

Theorem: If F is a PRF, then CBC-MAC is a fixed-length MAC that is existentially unforgeable under an adaptive chosen-message attack

This scheme is not secure for variable-length messages

But can be extended for this purpose in several ways

CBC-MAC for variable-length messages



This construction is provably secure (but proof is complex)

Summary: What can and can't be done with a MAC?

MACs allow

- Reliable exchange of information
- Between parties having agreed on a key
- And trusting each other

But anyone who can check a MAC can also forge one

- Suitable only for "closed communities"

Authenticity

Two flavors to authenticity:

1. Integrity: is this message authentic? ($\perp$ confidentiality)
2. Non malleability: could this plaintext result from a controlled manipulation of a ciphertext?

. . . Let us revisit confidentiality and encryption

Is CPA security a sufficient criterion?

We have given the adversary the possibility to

- Observe ciphertexts
- Encrypt plaintexts of his choice

What else could an attacker do?

- *Decrypt* ciphertexts of his choice

We will formalize this by giving $\mathcal{A}$ access to a *decryption oracle*

Chosen-ciphertext attacks

Given $\Pi := \langle \mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec} \rangle$, and adversary $\mathcal{A}$, define the following experiment $\mathsf{PrivK}^{\mathsf{cca}}_{\mathcal{A},\Pi}(n)$:

1. Choose $k \leftarrow \mathrm{Gen}(1^n)$
2. $\mathcal{A}$ is given oracle access to $\mathrm{Enc}_k(\cdot)$ and $\mathrm{Dec}_k(\cdot)$
3. $\mathcal{A}$ outputs $m_0, m_1 \in \mathcal{M}$
4. Choose $b \leftarrow \{0, 1\}$ and send $c := \mathrm{Enc}_k(m_b)$ to $\mathcal{A}$
5. $\mathcal{A}$ is again given oracle access to $\mathrm{Enc}_k(\cdot)$ and $\mathrm{Dec}_k(\cdot)$, but cannot ask $\mathrm{Dec}_k(c)$
6. $\mathcal{A}$ outputs b'
7. Define $\mathsf{PrivK}^{\mathsf{cca}}_{\mathcal{A},\Pi}(n) := 1$ iff $b = b'$

Security against CCA

$\Pi := \langle \mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec} \rangle$ has *indistinguishable encryption under a chosen-ciphertext attack* if $\forall$ PPT $\mathcal{A}$, $\exists\ \epsilon$:

$$\Pr[\mathsf{PrivK}^{\mathsf{cca}}_{\mathcal{A},\Pi}(n) = 1] \leq \frac{1}{2} + \epsilon(n)$$

Security against CCA

Which of the schemes we saw so far is CCA-secure?

None!

Example: attacking the scheme $c := \langle r, F_k(r) \oplus m \rangle$

- Send $m_0 = 0^n, m_1 = 1^n$ and receive $c := \mathrm{Enc}_k(m_b)$
- Flip the last bit of c
- Ask decryption of flipped message
  - (It is a different message $\Rightarrow$ allowed)
- See whether you get $0 \ldots 01$ or $1 \ldots 10$

As long as $\mathcal{A}$ can *manipulate* ciphertexts, we cannot prevent this kind of attack

Security against CCA

Is it what we want?

Good enough in many cases:

- Encrypt a key:
  any ciphertext manipulation will result in an unrelated key, that will not work and can be leaked without risk
- Encrypt a vote:
  any vote manipulation will result in an unrelated vote, that cannot leak about the original one

But we may want more:
unforgeable ciphertexts

- If decryption succeeds, then I have the right key
- If vote is valid, then it contains the expected intent

Unforgeable encryption

Given $\Pi := \langle \mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec} \rangle$, and adversary $\mathcal{A}$, define the following experiment $\mathsf{EncForge}_{\mathcal{A},\Pi}(n)$:

1. Choose $k \leftarrow \mathrm{Gen}(1^n)$
2. $\mathcal{A}$ is given oracle access to $\mathrm{Enc}_k(\cdot)$
   all queries are stored in $\mathcal{Q}$
3. $\mathcal{A}$ outputs c
4. Define $\mathsf{EncForge}_{\mathcal{A},\Pi}(n) := 1$ iff

$$\mathrm{Dec}_k(c) \neq \perp \text{ and } \mathrm{Dec}_k(c) \notin \mathcal{Q}$$

Unforgeable encryption

The encryption scheme $\Pi := \langle \mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec} \rangle$ is unforgeable if, $\forall$ PPT adversary $\mathcal{A}$, $\exists$ negl. ϵ s.t.:

$$\Pr[\mathsf{EncForge}_{\mathcal{A},\Pi}(n) = 1] \leq \epsilon(n)$$

Observe:

- Enc behaves like a MAC with message recovery (through Dec) Prevent forgery of encryption of new messages
- Unforgeability gives authenticity, not confidentiality

Authenticated Encryption

$\Pi := \langle \mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec} \rangle$ is an *authenticated encryption scheme* (AE) if it is CCA-secure and unforgeable

Note:

- Today's gold standard for confidentiality and authenticity in the private key setting!
- All schemes used for data transport in new TLS 1.3 are AE
- Often *AEAD* rather than just AE:
  Associated Data are authenticated with ciphertext but available in cleartext

CAESAR competition

A public competition aiming to choose a portfolio of authenticated encryption schemes

- Started in 2012
- Final round candidates announced in March 2018
- Final portfolio announced in February 2019
    - 6 algorithms retained (3 use cases)

Cryptographic competitions

Introduction
Secret-key cryptography
Disasters
Features

Focused competitions:
AES
eSTREAM
SHA-3
PHC
CAESAR

Broader evaluations:
CRYPTREC
NESSIE

CAESAR details:
Submissions

Introduction

A new competition, CAESAR, is now calling for submissions of **authenticated ciphers**. This competition follows a long tradition of focused competitions in secret-key cryptography:

- In 1997 the United States National Institute of Standards and Technology (NIST) announced an open competition for a new Advanced Encryption Standard. This competition attracted 15 block-cipher submissions from 50 cryptographers around the world, and then public security evaluations from an even larger pool of cryptanalysts, along with performance evaluations. Eventually NIST chose Rijndael as AES.
- In 2004 ECRYPT, a Network of Excellence funded by the European Union, announced eSTREAM, the ECRYPT Stream Cipher Project. This project called for submissions of "new stream ciphers suitable for widespread adoption". This call attracted 34 stream-cipher submissions from 100 cryptographers around the world, and then hundreds of security evaluations and performance evaluations, following the same pattern as AES but on a larger scale. Eventually the eSTREAM committee selected a portfolio containing several stream ciphers.
- In 2007 NIST announced an open competition for a new hash standard, SHA-3. This competition attracted 64 hash-function submissions from 200 cryptographers around the world, and then a tremendous volume of security evaluations and performance evaluations. Eventually NIST chose Keccak as SHA-3.

NIST competition

A similar competition focusing on lightweight solutions

> *"Each submission package* **shall** *describe a single algorithm, or a collection of algorithms, that implements the authenticated encryption with associated data (AEAD) functionality"*

- Call launched in August 2018
  - 56 accepted submissions
- Finalists for last round announced in March 2021
  - 10 retained candidates
  - Last round should last 1 year

NIST Issues First Call for 'Lightweight Cryptography' to Protect Small Electronics

April 18, 2018

Credit: N. Hanacek/NIST

Cryptography experts at the National Institute of Standards and Technology (NIST) are kicking off an effort to protect the data created by innumerable tiny networked devices such as those in the "internet of things" (IoT), which will need a new class of cryptographic defenses against cyberattacks.

Creating these defenses is the goal of NIST's lightweight cryptography initiative, which aims to develop cryptographic algorithm standards that can work within

MEDIA CONTACT

Chad Boutin
charles.boutin@nist.gov
(301) 975-4261

ORGANIZATIONS

Information Technology Laboratory
Computer Security Division
Cryptographic Technology Group

Combining MAC and encryption

Suppose:

- A and B want to have a confidential and authentic conversation
- They have:
  - CPA-secure encryption $\langle \mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec} \rangle$
  - Strongly unforgeable MAC $\langle \mathrm{Gen}', \mathrm{Mac}, \mathrm{Vrfy} \rangle$

  and share the appropriate secret keys.
- When receiving an invalid message, an error signal is sent

Should they transmit message m as:

1. $\mathrm{Enc}_{k_1}(m) \| \mathrm{Mac}_{k_2}(m)$ – MAC and Enc? (SSH)
2. $\mathrm{Enc}_{k_1}(m \| \mathrm{Mac}_{k_2}(m))$ – MAC then Enc? (SSL)
3. $\mathrm{Enc}_{k_1}(m) \| \mathrm{Mac}_{k_2}(\mathrm{Enc}_{k_1}(m))$ – Enc then MAC? (IPSec)

MAC and Enc

Should they transmit message m as:

1. $\mathrm{Enc}_{k_1}(m) \| \mathrm{Mac}_{k_2}(m)$ – MAC and Enc?

No!

- a MAC does not guarantee any privacy!

Example:

- Define $\mathrm{Mac}'_k(m) := \mathrm{Mac}_k(m) \| m$
- If Mac is secure, then so is Mac'
- But breaks CCA security trivially

MAC then Enc

Should they transmit message m as:

2. $\mathrm{Enc}_{k_1}(m \| \mathrm{Mac}_{k_2}(m))$ – MAC then Enc?

No!

- Enc may allow some malleability of ciphertexts

Example:

- Suppose Enc is a stream cipher where m is XORed with a pseudorandom stream (e.g., in CTR mode)
- Define $\mathrm{Enc}'_k(m) := \mathrm{Enc}_k(t(m))$, where $t(m)$ transforms each bit of m as follows:
  - each $'0'$ is replaced with $'00'$
  - each $'1'$ is replaced with $'01'$ or $'10'$ with $\Pr = 1/2$
- Dec' is adapted from Dec accordingly
- $\langle \mathrm{Gen}, \mathrm{Enc}', \mathrm{Dec}' \rangle$ is also CPA-secure

MAC then Enc

Should they transmit message m as:

2. $\mathrm{Enc}_{k_1}(m \| \mathrm{Mac}_{k_2}(m))$ – MAC then Enc?

Example – continued:

- Now, apply Mac-then-Enc using $\langle \mathrm{Gen}, \mathrm{Enc}', \mathrm{Dec}' \rangle$
- Suppose $\mathcal{A}$ wants to distinguish between messages $m_0 := 0$ and $m_1 := 1$
- $\mathcal{A}$ sees $c := \mathrm{Enc}_{k_1}(m_i \| \mathrm{Mac}_{k_2}(m_i))$
- $\mathcal{A}$ flips the first two bits of c and transmits!
  - if $m_i = 0$ then Vrfy fails and an error is sent
  - if $m_i = 1$ then Vrfy succeeds
- message can be recovered...

Can you make this an attack against CCA security? Unforgeability?

Enc then MAC

Should they transmit message m as:

3. $\mathrm{Enc}_{k_1}(m) \| \mathrm{Mac}_{k_2}(\mathrm{Enc}_{k_1}(m))$ – Enc then MAC?

This can be shown to work!

Intuition:

- Mac security guarantees that $\mathcal{A}$ cannot produce any new message not resulting in an error

$\Rightarrow$ $\mathcal{A}$ can just eavesdrop ciphertexts
but this does not help winning a CPA-game

Authenticated encryption modes

Still waiting for the end of the NIST competition

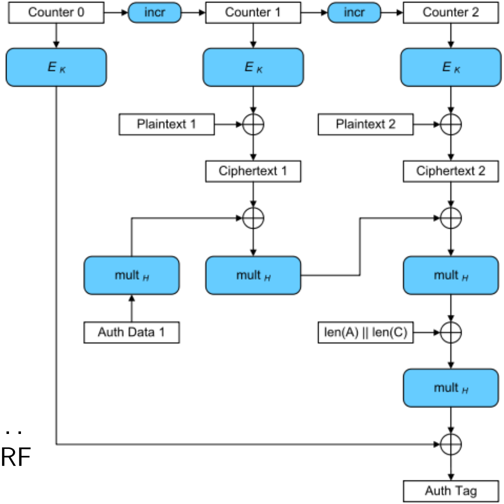
Standardised in TLS 1.3:

1. CCM mode:
   - $t = \text{CBC-MAC}_k(m)$
   - $c = \text{CTR-ENC}_k(m \| t)$

   Just needs a PRF
   Non-generic composition with a single key
   (would fail badly with CBC-MAC and CBC-ENC!)

Authenticated encryption modes

2. GCM mode:



Also used in SSH, . . .
Mult. faster than PRF

Authenticated encryption modes

2. GCM mode:

- CTR mode for ciphertext $\langle c_1, \ldots, c_l \rangle$
- $H = \mathrm{Enc}_k(0^n)$ used as random point on a polynomial
- Tag computed as $(((c_1 \cdot H \oplus c_2) \cdot H \oplus c_3) \cdots)$

Intuition: c_i's define a pseudorandom polynomial
Forgery on given IV $\approx$ means: got a different polynomial with a root in unknown H