Authenticated Encryption

Elena Kirshanova

Course “Information and Network Security”
Lecture 7
21 апреля 2020 г.

Agenda

Up until now:

- Confidentiality (using Symmetric Encryption)
- Integrity (MAC, HMAC)

These were protections against eavesdropping (**passive** adversary)

Agenda

Up until now:

- Confidentiality (using Symmetric Encryption)
- Integrity (MAC, HMAC)

These were protections against eavesdropping (**passive** adversary)

Today:
Protect data against (tampering) (**active** adversary):
Authenticated Encryption

Authenticated Encryption: definition

An Authenticated Encryption (AE) system consists of three ppt algorithms

- Key generation: KeyGen$(1^\lambda) : k \leftarrow \mathcal{K}$
- Encryption: Enc $: \mathcal{K} \times \mathcal{M} \times \mathcal{N} \rightarrow \mathcal{C}$
- Decryption: Dec $: \mathcal{K} \times \mathcal{C} \times \mathcal{N} \rightarrow \mathcal{M} \cup \{\perp\}$

$\mathcal{K}$ - key space, $\mathcal{M}$ - message space, $\mathcal{C}$ - ciphertext space, $\mathcal{N}$ - nonce space.

Authenticated Encryption: definition

An Authenticated Encryption (AE) system consists of three ppt algorithms

- Key generation: $\text{KeyGen}(1^\lambda) : k \leftarrow \mathcal{K}$
- Encryption: $\text{Enc} : \mathcal{K} \times \mathcal{M} \times \mathcal{N} \rightarrow \mathcal{C}$
- Decryption: $\text{Dec} : \mathcal{K} \times \mathcal{C} \times \mathcal{N} \rightarrow \mathcal{M} \cup \{\perp\}$

$\mathcal{K}$ - key space, $\mathcal{M}$ - message space, $\mathcal{C}$ - ciphertext space, $\mathcal{N}$ - nonce space.

NEW: $\{\perp\}$ - ciphertext is rejected

Authenticated Encryption: definition

An Authenticated Encryption (AE) system consists of three ppt algorithms

- Key generation: $\mathsf{KeyGen}(1^\lambda) : k \leftarrow \mathcal{K}$
- Encryption: $\mathsf{Enc} : \mathcal{K} \times \mathcal{M} \times \mathcal{N} \rightarrow \mathcal{C}$
- Decryption: $\mathsf{Dec} : \mathcal{K} \times \mathcal{C} \times \mathcal{N} \rightarrow \mathcal{M} \cup \{\perp\}$

$\mathcal{K}$ - key space, $\mathcal{M}$ - message space, $\mathcal{C}$ - ciphertext space, $\mathcal{N}$ - nonce space.

NEW: $\{\perp\}$ - ciphertext is rejected
Nonce = "number that can only be used once"
It can be predictable, but should never be used twice for the same key.
Example: values derived from IV in various modes of encryption.

Security of AE

An Authenticated Encryption (AE) system consists of three ppt algorithms

- Key generation: KeyGen$(1^\lambda) : k \leftarrow \mathcal{K}$
- Encryption: Enc $: \mathcal{K} \times \mathcal{M} \times \mathcal{N} \rightarrow \mathcal{C}$
- Decryption: Dec $: \mathcal{K} \times \mathcal{C} \times \mathcal{M} \times \mathcal{N} \rightarrow \mathcal{M} \cup \{\perp\}$

Correctness: $\forall m, \forall k, \forall n :$ Dec$(k,$ Enc$(k, m, n), n) = m$

Security of AE

An Authenticated Encryption (AE) system consists of three ppt algorithms

- Key generation: $\mathsf{KeyGen}(1^\lambda) : k \leftarrow \mathcal{K}$
- Encryption: $\mathsf{Enc} : \mathcal{K} \times \mathcal{M} \times \mathcal{N} \rightarrow \mathcal{C}$
- Decryption: $\mathsf{Dec} : \mathcal{K} \times \mathcal{C} \times \mathcal{M} \times \mathcal{N} \rightarrow \mathcal{M} \cup \{\perp\}$

Correctness: $\forall m, \forall k, \forall n$: $\mathsf{Dec}(k, \mathsf{Enc}(k, m, n), n) = m$ Security:

- $\mathsf{Enc}(k, m_0, n)$ is indistinguishable from $\mathsf{Enc}(k, m_1, n)$ $\forall m_0 ! = m_1$ (without knowledge of k)
- No ppt adversary can create a new ciphertext that does not decrypt to $\{\perp\}$.

Security of AE

Authenticated Encryption provides

- Authenticity: If $\text{Dec}(k, c, n)! = \{\perp\}$, then the receiver is ensured that the message comes from someone who knows k

Security of AE

Authenticated Encryption provides

- Authenticity: If $\text{Dec}(k, c, n)! = \{\bot\}$, then the receiver is ensured that the message comes from someone who knows k
- AE $\implies$ Chosen Ciphertext Security

Security of AE

Authenticated Encryption provides

- Authenticity: If $\text{Dec}(k, c, n)! = \{\perp\}$, then the receiver is ensured that the message comes from someone who knows k
- AE $\implies$ Chosen Ciphertext Security

In Chosen Ciphertext Attack (CCA) an adversary can

- obtain encryptions of messages of his choice
- ask for decryption of any ciphertext of his choice except one specific “challenge” c

A CCA adversary is a very powerful adversary.
Why does it capture real life attacks?

Example of CCA attack (IPSec, simplified)

Let Enc be a block cipher in CTR mode
The message m consists of a header "to Bob"+ the rest

Alice		Mail server		Bob
k	$\xrightarrow{c = \text{Enc}(k, m = \text{To Bob}\|\|\ldots)}$	k	$\xrightarrow{m}$	

Example of CCA attack (IPSec, simplified)

Let Enc be a block cipher in CTR mode
The message m consists of a header "to Bob"+ the rest

Alice		Mail server		Bob
k	$\xrightarrow{c\,=\text{Enc}(k,\, m=\text{To Bob}\|\|\ldots)}$	k	$\xrightarrow{m}$	

$\xrightarrow{c}$ Eve $\xrightarrow{\hat{c}}$

Example of CCA attack (IPSec, simplified)

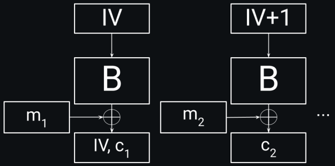
Let Enc be a block cipher in CTR mode
The message m consists of a header "to Bob"+ the rest

Alice | Mail server | Bob

$k \quad \xrightarrow{c = \text{Enc}(k, m = \text{To Bob}||\ldots)} \quad k \quad \xrightarrow{m}$

$\xrightarrow{c}$ Eve $\xrightarrow{\hat{c}}$

Assume `len`("to Bob") == `len`("to Eve") == block-size.



$\hat{c}_1 = c_1 \oplus$["to Bob"] $\oplus$ ["to Eve"]

The rest blocks of $\hat{c}$ are equal to c.

Eve knows m by querying Dec$(\hat{c})$.

Construction of AE

AE = Secure Encryption + Secure Mac

Two keys: Encryption key k_E, MAC key K_M

Construction of AE

AE = Secure Encryption + Secure Mac

Two keys: Encryption key k_E, MAC key K_M

Two main paradigms:

I. Encrypt-then-MAC

1. $c = \mathsf{Enc}(k_E, m)$
2. $t = \mathsf{MAC}(k_M, c)$
3. return (c, t)

Example: IPSec

Construction of AE

$$\text{AE} = \text{Secure Encryption} + \text{Secure Mac}$$

Two keys: Encryption key k_E, MAC key K_M

Two main paradigms:

I. Encrypt-then-MAC

1. $c = \text{Enc}(k_E, m)$
2. $t = \text{MAC}(k_M, c)$
3. return (c, t)

Example: IPSec

II. MAC-then-Encrypt

1. $t = \text{MAC}(k_M, n)$
2. $c = \text{Enc}(k_E, m||t)$
3. return c

Example: SSL

Construction of AE

AE = Secure Encryption + Secure Mac

Two keys: Encryption key k_E, MAC key K_M

Two main paradigms:

I. Encrypt-then-MAC

1. $c = \text{Enc}(k_E, m)$
2. $t = \text{MAC}(k_M, c)$
3. return (c, t)

Example: IPSec

II. MAC-then-Encrypt

1. $t = \text{MAC}(k_M, n)$
2. $c = \text{Enc}(k_E, m||t)$
3. return c

Example: SSL

- Encrypt-then-MAC always provides AE
- MAC-then-Encrypt provides AE when Enc is randomized CTR/CBC mode encryption
- Other combinations of Mac and Encryption usually do not provide secure AE

AE standards

1. **GCM (Galois Counter Mode)**. Encrypt-then-MAC
   Encryption: CTR mode + fast Mac (Carter-Wegman Mac).
   Application: TLS
   Advantages: somewhat fast (on Intel)

AE standards

1. **GCM (Galois Counter Mode)**. Encrypt-then-MAC
   Encryption: CTR mode + fast Mac (Carter-Wegman Mac).
   Application: TLS
   Advantages: somewhat fast (on Intel)
2. **CCM**. MAC-then-Encrypt
   Encryption: CBC MAC (AES)+ CTR mode (AES)
   Application: 802.11i
   Advantages: less code

AE standards

1. **GCM (Galois Counter Mode)**. Encrypt-then-MAC
   Encryption: CTR mode + fast Mac (Carter-Wegman Mac).
   Application: TLS
   Advantages: somewhat fast (on Intel)

2. **CCM**. MAC-then-Encrypt
   Encryption: CBC MAC (AES)+ CTR mode (AES)
   Application: 802.11i
   Advantages: less code

3. **ChaCha20-Poly1305.** Encrypt-then-MAC
   Encryption: ChaCha20 Encryption + Poly1305 MAC
   Application: TLS
   Advantages: fast

AE standards

1. **GCM (Galois Counter Mode)**. Encrypt-then-MAC
   Encryption: CTR mode + fast Mac (Carter-Wegman Mac).
   Application: TLS
   Advantages: somewhat fast (on Intel)

2. **CCM**. MAC-then-Encrypt
   Encryption: CBC MAC (AES)+ CTR mode (AES)
   Application: 802.11i
   Advantages: less code

3. **ChaCha20-Poly1305.** Encrypt-then-MAC
   Encryption: ChaCha20 Encryption + Poly1305 MAC
   Application: TLS
   Advantages: fast

These three are implemented in OpenSSL.
I do not know of Russian AE standards (although one can replace Enc and MAC by Russian GOSTs).

AEAD: Authenticated Encryption with Associated Data

Often not all data needs to be encrypted.

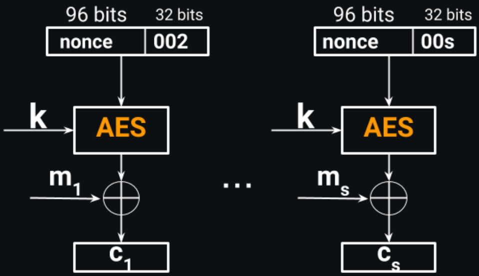
$$\underbrace{[\texttt{Associated data}||\texttt{Encrypted data}]}_{\text{Authenticated}}$$

Example: $[\texttt{header}||\texttt{payload}]$ in internet protocols

Most used AEAD: AES-GCM AEAD

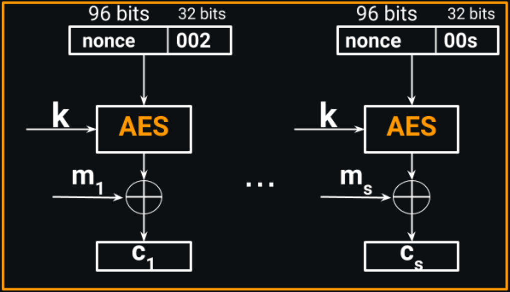
AES-GCM AEAD

Message $m = (m_1, \ldots, m_s)$



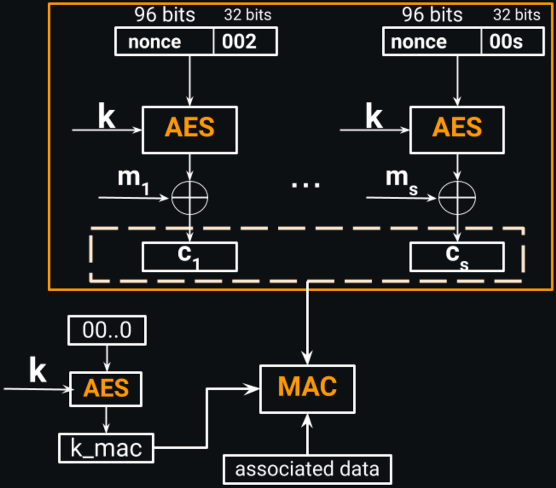
AES-GCM AEAD

Message $m = (m_1, \ldots, m_s)$



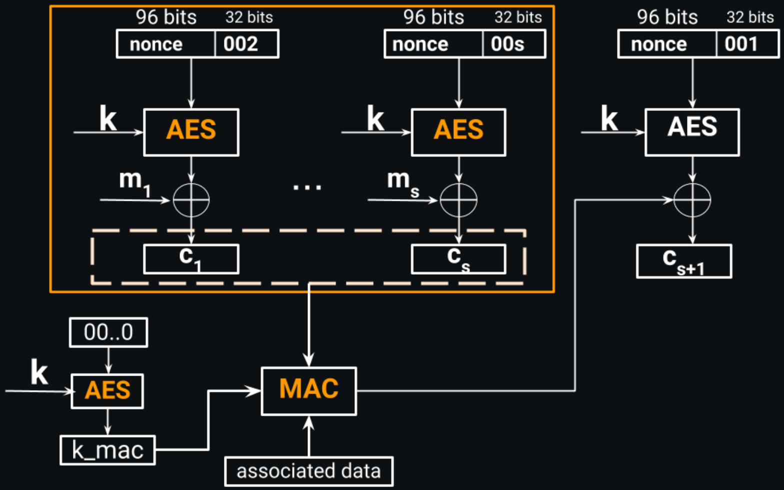
AES-GCM AEAD

Message $m = (m_1, \ldots, m_s)$



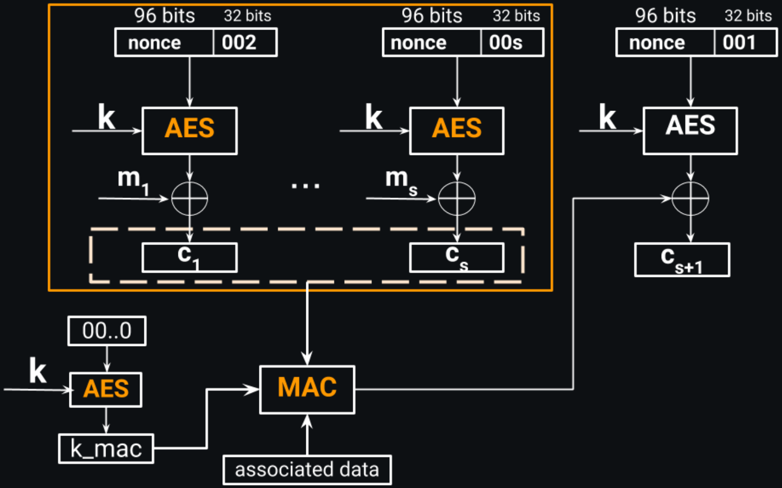
AES-GCM AEAD

Message $m = (m_1, \ldots, m_s)$



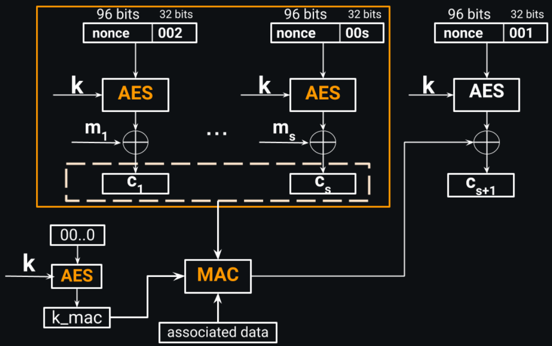
AES-GCM AEAD

Message $m = (m_1, \ldots, m_s)$



Output $(c_1, \ldots, c_s, c_{s+1})$

AES-GCM AEAD



- Uses just one key
- MAC: GHASH (Galois Hash) - uses finite field arithmetic (fast)
- Decryption:
  1. Verifies MAC
  2. Dec$(c_1, \ldots, c_s)$

AEAD in TLS 1.3

Browser	Phase 1 Handshake	Web server
	Asymmetric Encryption	
	Common keys are derived	
$k_{b \to s}$		$k_{b \to s}$
$k_{s \to b}$		$k_{s \to b}$

AEAD in TLS 1.3

`Browser` Phase 1 Handshake `Web server`

Asymmetric Encryption

Common keys are derived

$k_{b \to s}$ $k_{b \to s}$

$k_{s \to b}$ $k_{s \to b}$

Phase 2 TLS record protocol

AEAD

TLS record protocol

Data = $[m_1, \ldots, m_s]$

Browser		Web server
$k_{b\to s}$		$k_{b\to s}$
$k_{s\to b}$		$k_{s\to b}$
$ctr_{b\to s}$		$ctr_{b\to s}$
$ctr_{s\to b}$	[Meta data$\|\|\ m_i\ \|\|$ Nonce]	$ctr_{s\to b}$

$\underbrace{[\texttt{Meta data} \,||\, m_i \,||\, \texttt{Nonce}]}_{\text{AES-GCM-AEAD}(k_{b\to s})}$ $\longrightarrow$

TLS record protocol

Data = $[m_1, \ldots, m_s]$

Browser		Web server
$k_{b\to s}$		$k_{b\to s}$
$k_{s\to b}$		$k_{s\to b}$
$ctr_{b\to s}$		$ctr_{b\to s}$
$ctr_{s\to b}$	$\underbrace{[\texttt{Meta data} \| m_i \| \texttt{Nonce}]}_{\text{AES-GCM-AEAD}(k_{b\to s})} \longrightarrow$	$ctr_{s\to b}$
$ctr_{b\to s}{+}{+}$		$ctr_{b\to s}{+}{+}$

TLS record protocol

Data = $[m_1, \ldots, m_s]$

Browser		Web server
$k_{b \to s}$		$k_{b \to s}$
$k_{s \to b}$		$k_{s \to b}$
$ctr_{b \to s}$		$ctr_{b \to s}$
$ctr_{s \to b}$	$\underbrace{[\texttt{Meta data} \,\|\, m_i \,\|\, \texttt{Nonce}]}_{\text{AES-GCM-AEAD}(k_{b \to s})} \longrightarrow$	$ctr_{s \to b}$
$ctr_{b \to s}++$		$ctr_{b \to s}++$

`Meta data` includes: record on the phase (1 or 2), TLS Version, `len`(c)

Counters ctr are used to prevent replay attacks

Programming Assignment # 5

OpenSSL provides interfaces to GCM, CCM AEs via `EVP`

This PA: to implement Encryption and Decryption Interfaces for any two Authenticated Encryption

- GCM
- CCM
- ChaCha20-Poly1305

See `https://wiki.openssl.org/index.php/EVP_Authenticated_Encryption_and_Decryption` for code