On the (In)Security of IPsec in MAC-then-Encrypt Configurations

Jean Paul Degabriele Kenneth G. Paterson

Information Security Group
Royal Holloway, University of London

CCS 2010

Outline

1 Motivation

2 Security of MAC-then-Encrypt in IPsec
- Preliminaries
- Using an ESP Trailer Oracle to Recover Plaintext
- An Oracle Based on IP Fragmentation

3 Concluding Remarks

IPsec

- IPsec is a suite of protocols that provide security at the IP layer.
- Three main protocols – AH, ESP, IKE that can be combined in various ways, giving higher configurability.
- Encryption is provided by ESP, normally using a block cipher in CBC mode.
- Data origin authentication can be provided either by AH or ESP.
- Keys can be set manually or automatically through IKE.

Configuring IPsec

Royal Holloway University of London Information Security Group

- An admin who wants to use IPsec to ensure the confidentiality of network traffic, has to make a number of choices:
  - Encryption-only, Encrypt-then-MAC, MAC-then-encrypt.
  - Each of AH or ESP can be operated in Transport or Tunnel mode.
  - Is replay protection necessary to achieve confidentiality?
  - Should AH or ESP be used for authentication.
- The RFCs provide very little guidance on this matter.
- There exists no systematic security analysis of the resulting configurations.

Why Use MAC-then-Encrypt?

- SSL uses MAC-then-encrypt, and is widely perceived to be secure.
- A popular textbook by Stallings discusses several benefits that accrue from MAC-then-encrypt in IPsec.
- Ferguson and Schneier claim that encrypt-then-MAC as applied in ESP is **wrong**, and in their book ‘Practical Cryptography’ they recommend MAC-then-encrypt for constructing secure channels.
- Horton principle: *‘Authenticate what is meant not what is said’*.
- Krawczyk’s proof that MAC-then-encrypt in CBC mode is secure.

Why NOT MAC-then-Encrypt?

- Our paper presents practical attacks against **ALL** possible MAC-then-encrypt IPsec configurations:
  - AH in Transport mode followed by ESP in Transport mode
  - AH in Transport mode followed by ESP in Tunnel mode
  - AH in Tunnel mode followed by ESP in Transport mode
  - AH in Tunnel mode followed by ESP in Tunnel mode
- Even when replay protection is enabled.
- Also in a repeated ESP configuration (ESP in MAC-only followed by ESP in encryption-only).

Why NOT MAC-then-Encrypt?

- Our paper presents practical attacks against **ALL** possible MAC-then-encrypt IPsec configurations:
  - AH in Transport mode followed by ESP in Transport mode.
  - AH in Transport mode followed by ESP in Tunnel mode.
  - AH in Tunnel mode followed by ESP in Transport mode.
  - AH in Tunnel mode followed by ESP in Tunnel mode.
- Even when replay protection is enabled.
- Also in a repeated ESP configuration (ESP in MAC-only followed by ESP in encryption-only).

Why NOT MAC-then-Encrypt?

Royal Holloway University of London Information Security Group

- Our paper presents practical attacks against **ALL** possible MAC-then-encrypt IPsec configurations:
  - AH in Transport mode followed by ESP in Transport mode.
  - AH in Transport mode followed by ESP in Tunnel mode.
  - AH in Tunnel mode followed by ESP in Transport mode.
  - AH in Tunnel mode followed by ESP in Tunnel mode.
- Even when replay protection is enabled.
- Also in a repeated ESP configuration (ESP in MAC-only followed by ESP in encryption-only).

Why NOT MAC-then-Encrypt?

- Our paper presents practical attacks against **ALL** possible MAC-then-encrypt IPsec configurations:
  - AH in Transport mode followed by ESP in Transport mode.
  - AH in Transport mode followed by ESP in Tunnel mode.
  - AH in Tunnel mode followed by ESP in Transport mode.
  - AH in Tunnel mode followed by ESP in Tunnel mode.
- Even when replay protection is enabled.
- Also in a repeated ESP configuration (ESP in MAC-only followed by ESP in encryption-only).

Why NOT MAC-then-Encrypt?

Royal Holloway University of London Information Security Group

- Our paper presents practical attacks against **ALL** possible MAC-then-encrypt IPsec configurations:
  - AH in Transport mode followed by ESP in Transport mode.
  - AH in Transport mode followed by ESP in Tunnel mode.
  - AH in Tunnel mode followed by ESP in Transport mode.
  - AH in Tunnel mode followed by ESP in Tunnel mode.
- Even when replay protection is enabled.
- Also in a repeated ESP configuration (ESP in MAC-only followed by ESP in encryption-only).

Why NOT MAC-then-Encrypt?

Royal Holloway University of London Information Security Group

- Our paper presents practical attacks against **ALL** possible MAC-then-encrypt IPsec configurations:
  - AH in Transport mode followed by ESP in Transport mode.
  - AH in Transport mode followed by ESP in Tunnel mode.
  - AH in Tunnel mode followed by ESP in Transport mode.
  - AH in Tunnel mode followed by ESP in Tunnel mode.
- Even when replay protection is enabled.
- Also in a repeated ESP configuration (ESP in MAC-only followed by ESP in encryption-only).

Outline

Bit Flipping in CBC Mode

CBC encryption

$$C_i = \mathcal{E}_k(P_i \oplus C_{i-1});\ C_0 = IV$$

CBC decryption

$$P_i = \mathcal{D}_k(C_i) \oplus C_{i-1};\ C_0 = IV$$

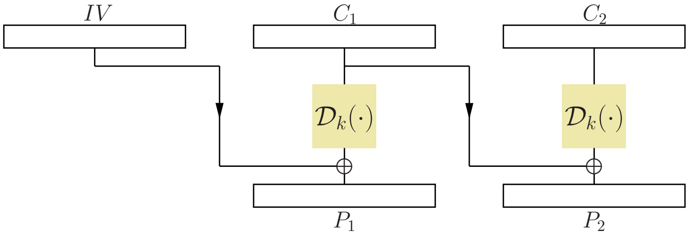
Bit Flipping in CBC Mode

CBC encryption

$$C_i = \mathcal{E}_k(P_i \oplus C_{i-1});\ C_0 = IV$$

CBC decryption

$$P_i = \mathcal{D}_k(C_i) \oplus C_{i-1};\ C_0 = IV$$



Bit Flipping in CBC Mode

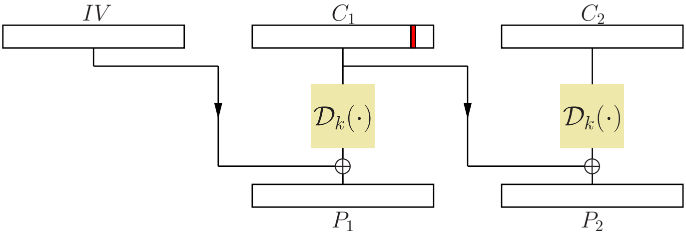
Royal Holloway University of London Information Security Group

CBC encryption

$C_i = \mathcal{E}_k(P_i \oplus C_{i-1});\ C_0 = IV$

CBC decryption

$P_i = \mathcal{D}_k(C_i) \oplus C_{i-1};\ C_0 = IV$



Bit Flipping in CBC Mode

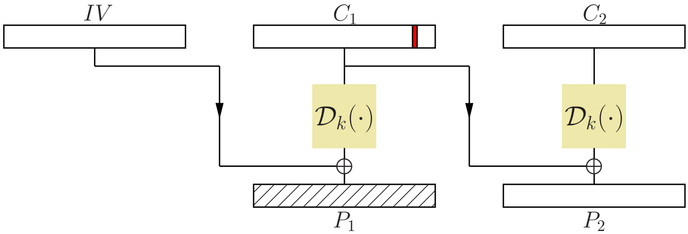
Royal Holloway University of London Information Security Group

CBC encryption

$C_i = \mathcal{E}_k(P_i \oplus C_{i-1});\ C_0 = IV$

CBC decryption

$P_i = \mathcal{D}_k(C_i) \oplus C_{i-1};\ C_0 = IV$



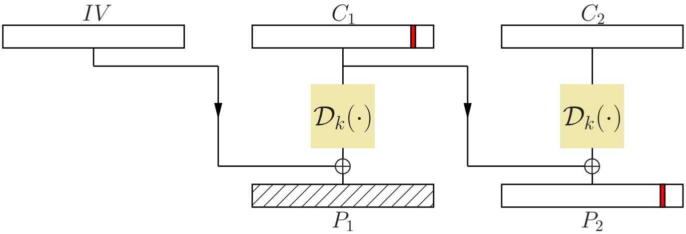
Bit Flipping in CBC Mode

CBC encryption

$$C_i = \mathcal{E}_k(P_i \oplus C_{i-1});\ C_0 = IV$$

CBC decryption

$$P_i = \mathcal{D}_k(C_i) \oplus C_{i-1};\ C_0 = IV$$



Bit Flipping in CBC Mode

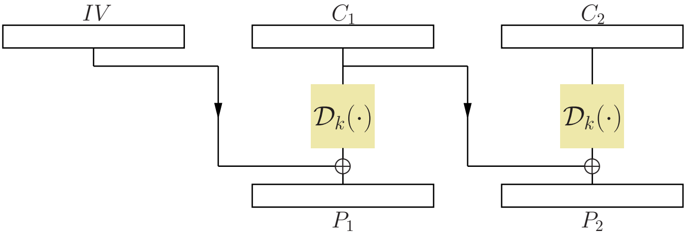
Royal Holloway University of London Information Security Group

CBC encryption

$$C_i = \mathcal{E}_k(P_i \oplus C_{i-1});\ C_0 = IV$$

CBC decryption

$$P_i = \mathcal{D}_k(C_i) \oplus C_{i-1};\ C_0 = IV$$



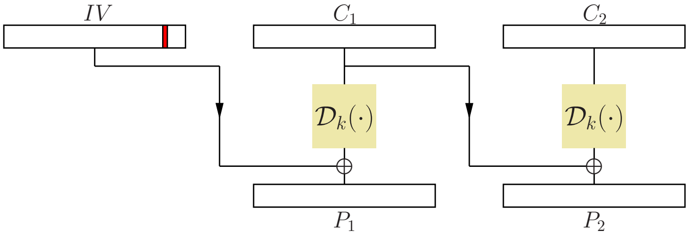
Bit Flipping in CBC Mode

CBC encryption

$$C_i = \mathcal{E}_k(P_i \oplus C_{i-1});\ C_0 = IV$$

CBC decryption

$$P_i = \mathcal{D}_k(C_i) \oplus C_{i-1};\ C_0 = IV$$



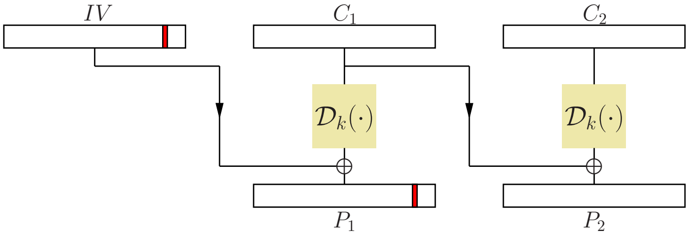
Bit Flipping in CBC Mode

CBC encryption

$$C_i = \mathcal{E}_k(P_i \oplus C_{i-1});\ C_0 = IV$$

CBC decryption

$$P_i = \mathcal{D}_k(C_i) \oplus C_{i-1};\ C_0 = IV$$



The ESP Trailer Format

- Prior to encryption an *ESP trailer* is appended to the plaintext, extending its length to an integer multiple of the blocksize.

Plaintext

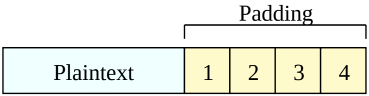
- PL = Padding Length
  NH = Next Header
- In Tunnel mode NH = 4

0,4
1,1,4
1,2,2,4
1,2,3,3,4

The ESP Trailer Format

Royal Holloway University of London Information Security Group

- Prior to encryption an *ESP trailer* is appended to the plaintext, extending its length to an integer multiple of the blocksize.

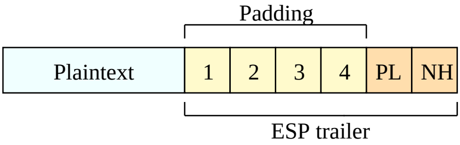


- PL = Padding Length
  NH = Next Header
- In Tunnel mode NH = 4

0,4
1,1,4
1,2,2,4
1,2,3,3,4

The ESP Trailer Format

- Prior to encryption an *ESP trailer* is appended to the plaintext, extending its length to an integer multiple of the blocksize.

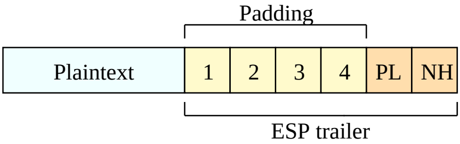


- PL = Padding Length
  NH = Next Header
- In Tunnel mode NH = 4

0,4
1,1,4
1,2,2,4
1,2,3,3,4

The ESP Trailer Format

- Prior to encryption an *ESP trailer* is appended to the plaintext, extending its length to an integer multiple of the blocksize.



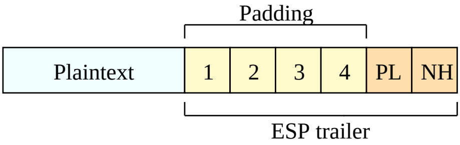
- PL = Padding Length
  NH = Next Header
- In Tunnel mode NH = 4

0,4
1,1,4
1,2,2,4
1,2,3,3,4

The ESP Trailer Format

Royal Holloway University of London Information Security Group

- Prior to encryption an *ESP trailer* is appended to the plaintext, extending its length to an integer multiple of the blocksize.

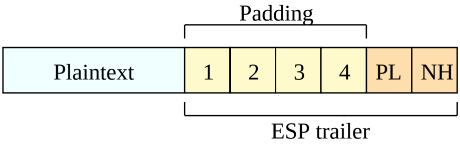


- PL = Padding Length
  NH = Next Header
- In Tunnel mode NH = 4

0,4
1,1,4
1,2,2,4
1,2,3,3,4

The ESP Trailer Format

- Prior to encryption an *ESP trailer* is appended to the plaintext, extending its length to an integer multiple of the blocksize.



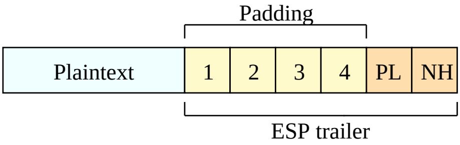
- PL = Padding Length
  NH = Next Header
- In Tunnel mode NH = 4

0,4
1,1,4
1,2,2,4
1,2,3,3,4

The ESP Trailer Format

Royal Holloway University of London Information Security Group

- Prior to encryption an *ESP trailer* is appended to the plaintext, extending its length to an integer multiple of the blocksize.



- PL = Padding Length
  NH = Next Header
- In Tunnel mode NH = 4

0,4
1,1,4
1,2,2,4
1,2,3,3,4

An ESP Trailer Oracle

Definition

An ESP Trailer Oracle is an oracle that when presented with an ESP-protected IP packet, outputs 1 if the underlying plaintext ends in a **valid ESP trailer**, and outputs 0 otherwise.

- Such an oracle is an adaptation of Vaudenay's padding oracle concept from Eurocrypt '02 to the IPsec setting.

An ESP Trailer Oracle

Definition

An ESP Trailer Oracle is an oracle that when presented with an ESP-protected IP packet, outputs 1 if the underlying plaintext ends in a **valid ESP trailer**, and outputs 0 otherwise.

- Such an oracle is an adaptation of Vaudenay's padding oracle concept from Eurocrypt '02 to the IPsec setting.

Outline

Royal Holloway University of London Information Security Group

A Decryption Algorithm

From an ESP Trailer Oracle

- Choose a ciphertext block from an ESP-protected IP packet that we want to decrypt.

IP $\text{header}^*_{\text{out}}$	ESP header^*	$C^*_0, C^*_1, \ldots, \mathbf{C^*_i}, \ldots, C^*_n$

- Arbitrarily pick another packet – call it the *carrier packet*.

- Append a random block R and block C^*_i to the carrier packet and submit it to the oracle.

A Decryption Algorithm
From an ESP Trailer Oracle

- Choose a ciphertext block from an ESP-protected IP packet that we want to decrypt.

IP $\text{header}^*_{\text{out}}$	ESP header^*	$C^*_0, C^*_1, \ldots, \mathbf{C^*_i}, \ldots, C^*_n$

- Arbitrarily pick another packet – call it the *carrier packet*.

IP $\text{header}_{\text{out}}$	ESP header	$C_0, C_1, C_2, \ldots, C_{n-1}, C_n$

- Append a random block R and block C^*_i to the carrier packet and submit it to the oracle.

A Decryption Algorithm

From an ESP Trailer Oracle

- Choose a ciphertext block from an ESP-protected IP packet that we want to decrypt.

IP header$^*_{\text{out}}$	ESP header*	$C_0^*, C_1^*, \ldots, \mathbf{C_i^*}, \ldots, C_n^*$

- Arbitrarily pick another packet – call it the *carrier packet*.

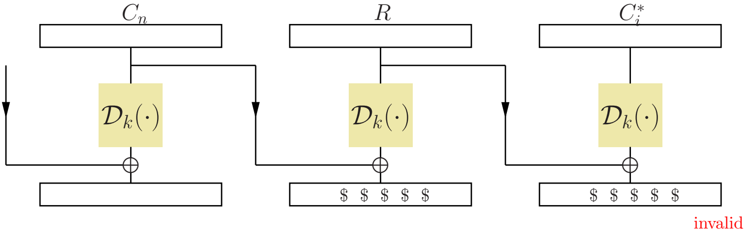
IP header$_{\text{out}}$	ESP header	$C_0, C_1, C_2, \ldots, C_{n-1}, C_n$

- Append a random block R and block C_i^* to the carrier packet and submit it to the oracle.

IP header$_{\text{out}}$	ESP header	$C_0, C_1, C_2, \ldots, C_{n-1}, C_n, R, C_i^*$

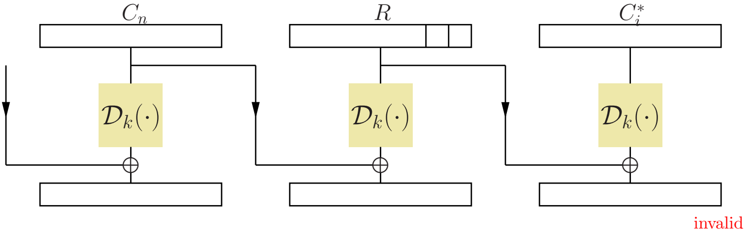
A Decryption Algorithm

From an ESP Trailer Oracle



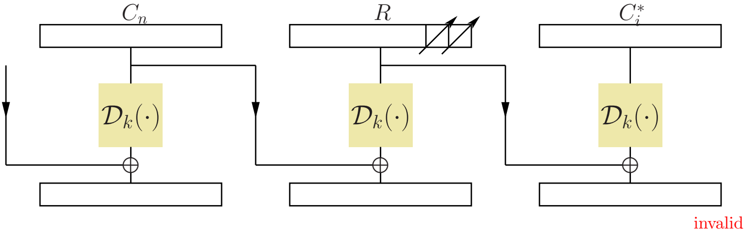
- $T_{ESP} = \mathcal{D}_K(C_i^*) \oplus R$
- $P_i^* = \mathcal{D}_K(C_i^*) \oplus C_{i-1}^*$.

A Decryption Algorithm
From an ESP Trailer Oracle



- $T_{ESP} = \mathcal{D}_K(C_i^*) \oplus R$
- $P_i^* = \mathcal{D}_K(C_i^*) \oplus C_{i-1}^*$.

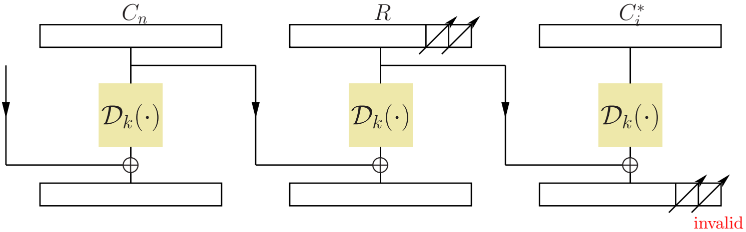
A Decryption Algorithm
From an ESP Trailer Oracle



- $T_{ESP} = \mathcal{D}_K(C_i^*) \oplus R$
- $P_i^* = \mathcal{D}_K(C_i^*) \oplus C_{i-1}^*$.

A Decryption Algorithm

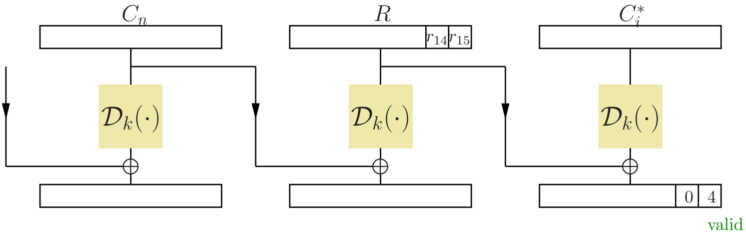
From an ESP Trailer Oracle



- $T_{ESP} = \mathcal{D}_K(C_i^*) \oplus R$
- $P_i^* = \mathcal{D}_K(C_i^*) \oplus C_{i-1}^*$.

A Decryption Algorithm

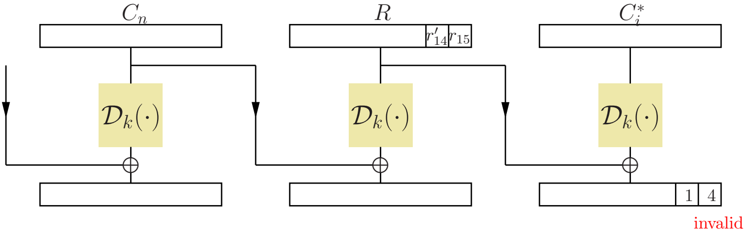
From an ESP Trailer Oracle



- $T_{ESP} = \mathcal{D}_K(C_i^*) \oplus R$
- $P_i^* = \mathcal{D}_K(C_i^*) \oplus C_{i-1}^*$.

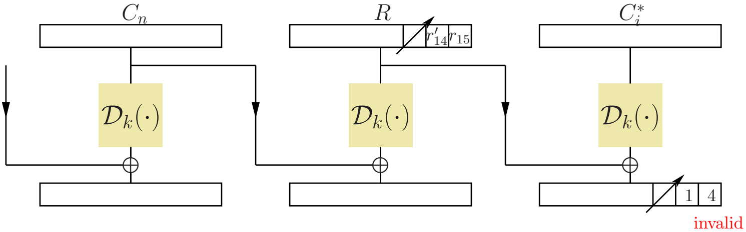
A Decryption Algorithm

From an ESP Trailer Oracle



- $T_{ESP} = \mathcal{D}_K(C_i^*) \oplus R$
- $P_i^* = \mathcal{D}_K(C_i^*) \oplus C_{i-1}^*$.

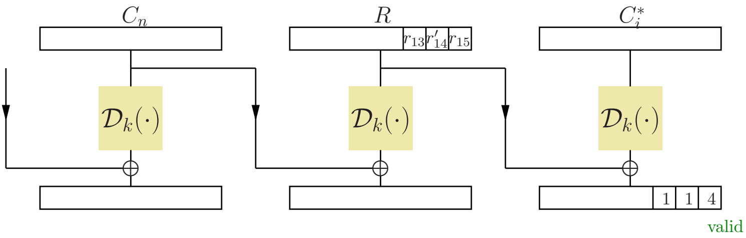
A Decryption Algorithm
From an ESP Trailer Oracle



- $T_{ESP} = \mathcal{D}_K(C_i^*) \oplus R$
- $P_i^* = \mathcal{D}_K(C_i^*) \oplus C_{i-1}^*$.

A Decryption Algorithm

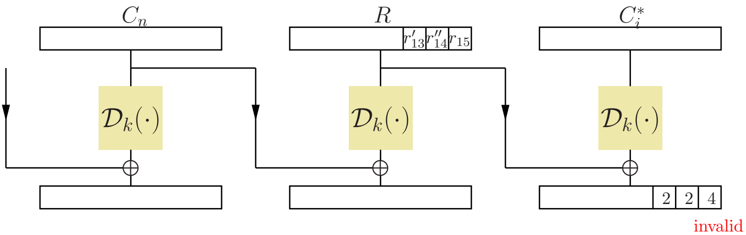
From an ESP Trailer Oracle



- $T_{ESP} = \mathcal{D}_K(C_i^*) \oplus R$
- $P_i^* = \mathcal{D}_K(C_i^*) \oplus C_{i-1}^*$.

A Decryption Algorithm

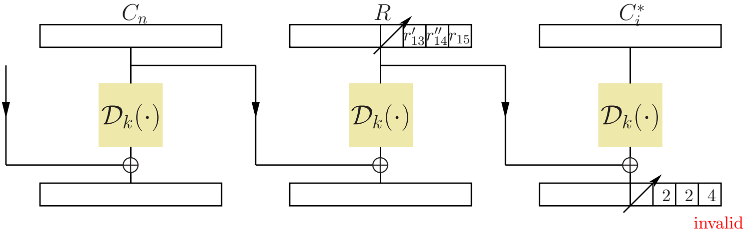
From an ESP Trailer Oracle



- $T_{ESP} = \mathcal{D}_K(C_i^*) \oplus R$
- $P_i^* = \mathcal{D}_K(C_i^*) \oplus C_{i-1}^*$.

A Decryption Algorithm

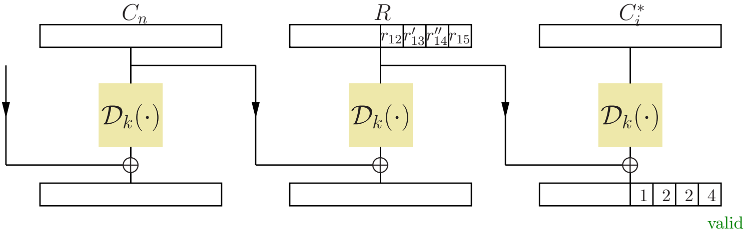
From an ESP Trailer Oracle



- $T_{ESP} = \mathcal{D}_K(C_i^*) \oplus R$
- $P_i^* = \mathcal{D}_K(C_i^*) \oplus C_{i-1}^*$.

A Decryption Algorithm

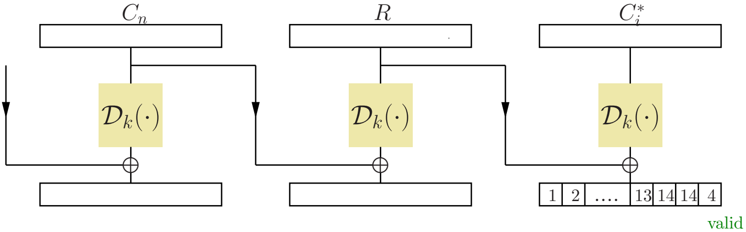
From an ESP Trailer Oracle



- $T_{ESP} = \mathcal{D}_K(C_i^*) \oplus R$
- $P_i^* = \mathcal{D}_K(C_i^*) \oplus C_{i-1}^*$.

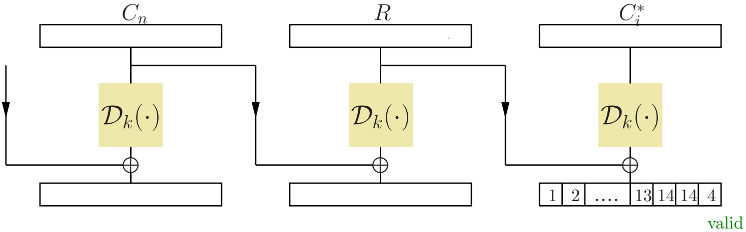
A Decryption Algorithm

From an ESP Trailer Oracle



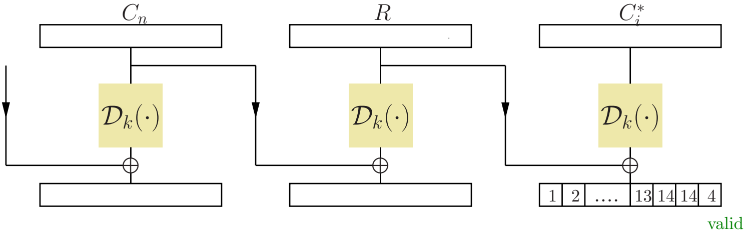
- $T_{ESP} = \mathcal{D}_K(C_i^*) \oplus R$
- $P_i^* = \mathcal{D}_K(C_i^*) \oplus C_{i-1}^*$.

A Decryption Algorithm
From an ESP Trailer Oracle



- $T_{ESP} = \mathcal{D}_k(C_i^*) \oplus R$
- $P_i^* = \mathcal{D}_k(C_i^*) \oplus C_{i-1}^*$.

A Decryption Algorithm
From an ESP Trailer Oracle



- $T_{ESP} = \mathcal{D}_k(C_i^*) \oplus R$
- $P_i^* = \mathcal{D}_k(C_i^*) \oplus C_{i-1}^*$.

Realising the Oracle

- An ESP trailer oracle can be realised by using a carrier packet that *always* generates a reply *after* IPsec processing.
- If the ESP trailer is invalid the packet is **discarded** and no reply is generated.
- Contrarily if the ESP trailer is valid a **reply** will be sent over the network.

Practical Complications

- A packet may be dropped for several other reasons – if this happens *the oracle is lost*.
- Flipping bits and appending blocks may invalidate the MAC.
- Replay protection prevents us from using the same carrier packet more than once.
- We need to preserve a valid internal structure of the IP packet.
- Our paper describes several ways to solve these problems – we will now present one approach.

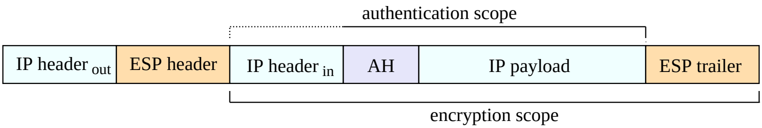
Outline

MAC-then-Encrypt

Example Configuration

Royal Holloway University of London Information Security Group

- In the following attack we will assume AH in Transport mode, followed by ESP in Tunnel mode.



IP Fragmentation

- An IP packet may be split into smaller autonomous *fragments* by an intermediate gateway.
- The **MF** bit is a flag in the IP header indicating that the IP packet is indeed a fragment and there are *More Fragments* to come.

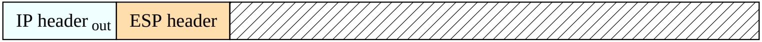
An Oracle from IP Fragmentation

Royal Holloway University of London Information Security Group

- We flip bits in the IV to set the MF bit to 1 and correct the checksum in the inner IP header.
- The inner IP packet is not complete ⇒ cannot verify the MAC!
- The receiver enters a wait stage, waiting for more fragments.
- No other fragments exist, the wait stage will eventually timeout and an **ICMP Time Exceeded** message is sent.
- The packet is discarded and no further processing takes place.
- We realised the ESP trailer oracle, and completely bypassed AH processing!

An Oracle from IP Fragmentation

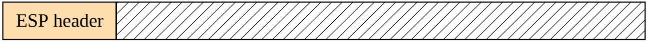
- We flip bits in the IV to set the MF bit to 1 and correct the checksum in the inner IP header.



- The inner IP packet is not complete ⇒ cannot verify the MAC!
- The receiver enters a wait stage, waiting for more fragments.
- No other fragments exist, the wait stage will eventually timeout and an **ICMP Time Exceeded** message is sent.
- The packet is discarded and no further processing takes place.
- We realised the ESP trailer oracle, and completely bypassed AH processing!

An Oracle from IP Fragmentation

- We flip bits in the IV to set the MF bit to 1 and correct the checksum in the inner IP header.



- The inner IP packet is not complete $\Rightarrow$ cannot verify the MAC!
- The receiver enters a wait stage, waiting for more fragments.
- No other fragments exist, the wait stage will eventually timeout and an **ICMP Time Exceeded** message is sent.
- The packet is discarded and no further processing takes place.
- We realised the ESP trailer oracle, and completely bypassed AH processing!

An Oracle from IP Fragmentation

- We flip bits in the IV to set the MF bit to 1 and correct the checksum in the inner IP header.

ESP header	IP header $_{in}$	AH	IP payload	?

- The inner IP packet is not complete $\Rightarrow$ cannot verify the MAC!
- The receiver enters a wait stage, waiting for more fragments.
- No other fragments exist, the wait stage will eventually timeout and an **ICMP Time Exceeded** message is sent.
- The packet is discarded and no further processing takes place.
- We realised the ESP trailer oracle, and completely bypassed AH processing!

An Oracle from IP Fragmentation

Royal Holloway University of London Information Security Group

- We flip bits in the IV to set the MF bit to 1 and correct the checksum in the inner IP header.

ESP header	IP header $_{in}$	AH	IP payload	ESP trailer

- The inner IP packet is not complete ⇒ cannot verify the MAC!
- The receiver enters a wait stage, waiting for more fragments.
- No other fragments exist, the wait stage will eventually timeout and an **ICMP Time Exceeded** message is sent.
- The packet is discarded and no further processing takes place.
- We realised the ESP trailer oracle, and completely bypassed AH processing!

An Oracle from IP Fragmentation

Royal Holloway University of London Information Security Group

- We flip bits in the IV to set the MF bit to 1 and correct the checksum in the inner IP header.

IP header $_{in}$	AH	IP payload

- The inner IP packet is not complete $\Rightarrow$ cannot verify the MAC!
- The receiver enters a wait stage, waiting for more fragments.
- No other fragments exist, the wait stage will eventually timeout and an **ICMP Time Exceeded** message is sent.
- The packet is discarded and no further processing takes place.
- We realised the ESP trailer oracle, and completely bypassed AH processing!

An Oracle from IP Fragmentation

- We flip bits in the IV to set the MF bit to 1 and correct the checksum in the inner IP header.

MF = 1	AH	IP payload

- The inner IP packet is not complete ⇒ cannot verify the MAC!
- The receiver enters a wait stage, waiting for more fragments.
- No other fragments exist, the wait stage will eventually timeout and an **ICMP Time Exceeded** message is sent.
- The packet is discarded and no further processing takes place.
- We realised the ESP trailer oracle, and completely bypassed AH processing!

An Oracle from IP Fragmentation

Royal Holloway University of London Information Security Group

- We flip bits in the IV to set the MF bit to 1 and correct the checksum in the inner IP header.

MF = 1	AH	IP payload

- The inner IP packet is not complete ⇒ cannot verify the MAC!
- The receiver enters a wait stage, waiting for more fragments.
- No other fragments exist, the wait stage will eventually timeout and an **ICMP Time Exceeded** message is sent.
- The packet is discarded and no further processing takes place.
- We realised the ESP trailer oracle, and completely bypassed AH processing!

An Oracle from IP Fragmentation

- We flip bits in the IV to set the MF bit to 1 and correct the checksum in the inner IP header.

MF = 1	AH	IP payload

- The inner IP packet is not complete $\Rightarrow$ cannot verify the MAC!
- The receiver enters a wait stage, waiting for more fragments.
- No other fragments exist, the wait stage will eventually timeout and an **ICMP Time Exceeded** message is sent.
- The packet is discarded and no further processing takes place.
- We realised the ESP trailer oracle, and completely bypassed AH processing!

An Oracle from IP Fragmentation

Royal Holloway University of London Information Security Group

- We flip bits in the IV to set the MF bit to 1 and correct the checksum in the inner IP header.

MF = 1	AH	IP payload

- The inner IP packet is not complete $\Rightarrow$ cannot verify the MAC!
- The receiver enters a wait stage, waiting for more fragments.
- No other fragments exist, the wait stage will eventually timeout and an **ICMP Time Exceeded** message is sent.
- The packet is discarded and no further processing takes place.
- We realised the ESP trailer oracle, and completely bypassed AH processing!

An Oracle from IP Fragmentation

- We flip bits in the IV to set the MF bit to 1 and correct the checksum in the inner IP header.

MF = 1	AH	IP payload

- The inner IP packet is not complete $\Rightarrow$ cannot verify the MAC!
- The receiver enters a wait stage, waiting for more fragments.
- No other fragments exist, the wait stage will eventually timeout and an **ICMP Time Exceeded** message is sent.
- The packet is discarded and no further processing takes place.
- We realised the ESP trailer oracle, and completely bypassed AH processing!

An Oracle from IP Fragmentation

- We flip bits in the IV to set the MF bit to 1 and correct the checksum in the inner IP header.

MF = 1	AH	IP payload

- The inner IP packet is not complete $\Rightarrow$ cannot verify the MAC!
- The receiver enters a wait stage, waiting for more fragments.
- No other fragments exist, the wait stage will eventually timeout and an **ICMP Time Exceeded** message is sent.
- The packet is discarded and no further processing takes place.
- We realised the ESP trailer oracle, and completely bypassed AH processing!

Implementing the Attacks

- We implemented and verified the validity of our attacks against the OpenSolaris IPsec implementation.
- The fragmentation attack takes around 10 mins to recover a 128 bit block of plaintext due to the timeout overhead.
- On a 10Mb Hub our alternative attacks take roughly 70 seconds to recover a 128 bit block of plaintext.

Conclusions

From a Practical Perspective

- All MAC-then-encrypt IPsec configurations are vulnerable to one or more of our attacks.
- Encryption-only configurations of IPsec are already known to be insecure.
- Thus most of the flexibility provided by IPsec results in **insecure configurations** – only encrypt-then-MAC provides confidentiality in IPsec.
- Our attacks highlight the difficulty in designing a secure network protocol – its interaction with the upper and lower layer protocols has to be included in the picture.

Conclusions

From a Theoretical Perspective

Royal Holloway University of London Information Security Group

- Krawczyk's security model considers a very restricted case of MAC-then-encrypt.
- Security proofs normally assume the cryptographic processing to be atomic and do not consider distinct error messages or fragmentation.
- Care should be taken in interpreting security proofs with respect to secure protocols in practice.
- What does the proof assume? How does the model compare to real life?

Conclusions

From a Theoretical Perspective

- Krawczyk's security model considers a very restricted case of MAC-then-encrypt.
- Security proofs normally assume the cryptographic processing to be atomic and do not consider distinct error messages or fragmentation.
- Care should be taken in interpreting security proofs with respect to secure protocols in practice.
- What does the proof assume? How does the model compare to real life?