Stronger Security Guarantees for Authenticated Encryption Schemes

Alexandra Boldyreva, **Jean Paul Degabriele**, Kenny Paterson, and Martijn Stam

DIAC Workshop - 5th July 2012

Scope of This Talk

- Arguably the best way we have of assessing the security of an Authenticated Encryption (AE) scheme is through its security proof.
- There are various criteria for assessing security proofs: security notion, tightness and quantitative bounds, assumptions, etc. but most importantly we want security to hold in in practice!
- This relates to how well our theoretic framework captures real-world scenarios. In this talk we consider two aspects that that current cryptographic theory fails to address.
- We here outline recent and upcoming work of ours to address these, and propose new design criteria for AE schemes.

Outline

Outline

Ciphertext Fragmentation

Royal Holloway
University of London
Information Security Group

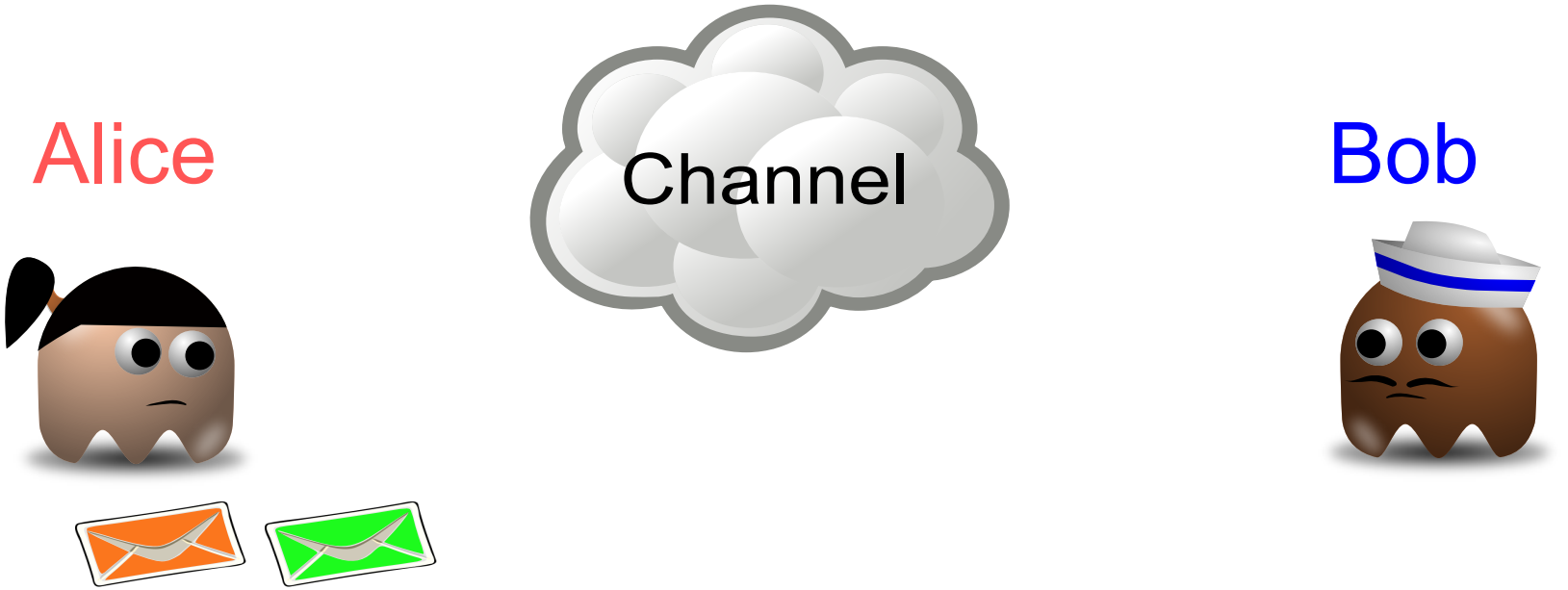
Under *normal operation* the channel delivers ciphertexts in a fragmented fashion, where:

a) The fragmentation pattern is arbitrary.

b) But the order of the fragments is preserved.

Ciphertext Fragmentation

Royal Holloway
University of London
Information Security Group

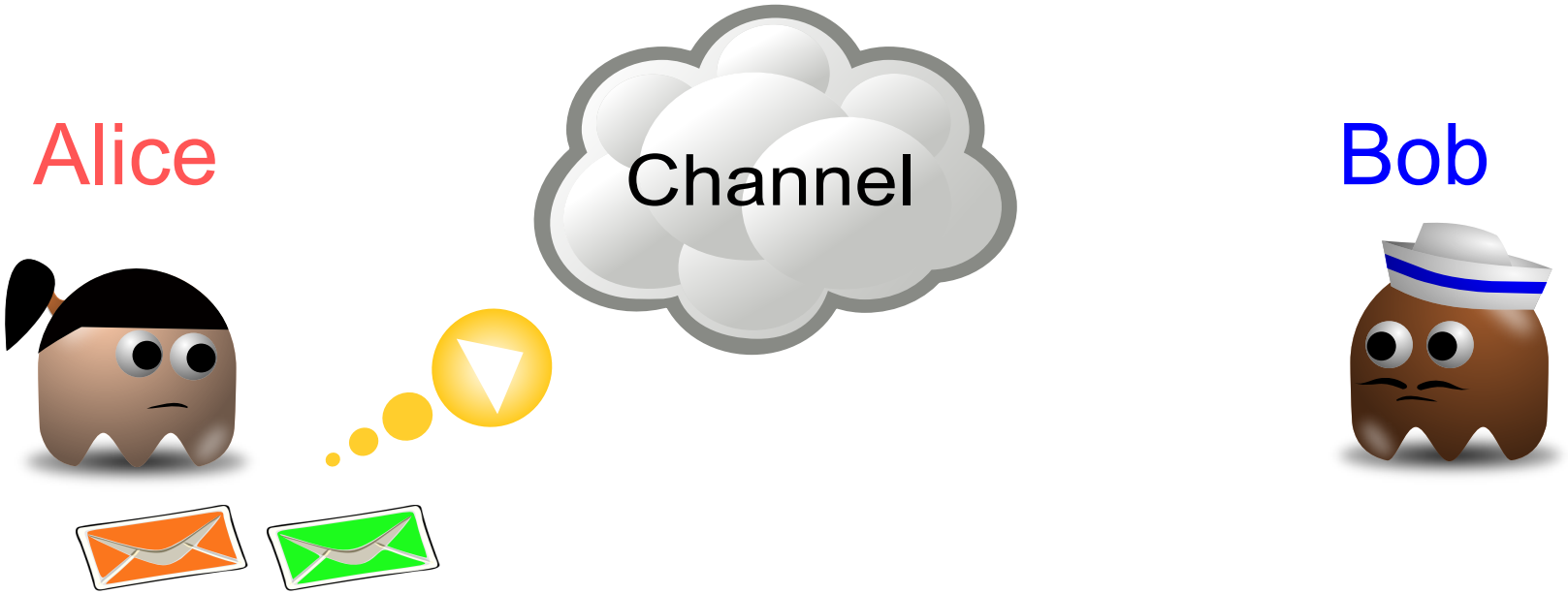


Under *normal operation* the channel delivers ciphertexts in a fragmented fashion, where:

a) The fragmentation pattern is arbitrary.

b) But the order of the fragments is preserved.

Ciphertext Fragmentation



Under *normal operation* the channel delivers ciphertexts in a fragmented fashion, where:

a) The fragmentation pattern is arbitrary.

b) But the order of the fragments is preserved.

Ciphertext Fragmentation

Under *normal operation* the channel delivers ciphertexts in a fragmented fashion, where:

a) The fragmentation pattern is arbitrary.

b) But the order of the fragments is preserved.

Ciphertext Fragmentation

Under *normal operation* the channel delivers ciphertexts in a fragmented fashion, where:

a) The fragmentation pattern is arbitrary.

b) But the order of the fragments is preserved.

Why is it a Problem?

- This setting emerges in practice, one such instance is that of **secure network protocols**.
- AE schemes are NOT designed to operate in this setting, and it is left to the protocol designer to adapt the scheme into one that supports ciphertext fragmentation (hoping that security is preserved).
- However as the following two examples show, security in the usual ‘atomic’ setting does not guarantee security in the fragmented setting.

Ciphertext-Fragmentation Attacks

Royal Holloway
University of London
Information Security Group

SSH:

- A proof of security (IND-sfCCA) for SSH was given in **[BKN 04]**.
- Yet **[APW 09]** presented plaintext-recovery attacks against SSH.

IPsec in MAC-then-encrypt (CBC):

- **[Kra 01]** proves that MAC-then-encrypt with CBC encryption is secure (secure channel [CK 01]).
- **[MT 10]** show that MAC-then-encode-then-encrypt (injective / CBC) is secure (secure channel [Mau 11]).
- **[DP 10]** present ciphertext-fragmentation attacks against such IPsec configurations.

Ciphertext-Fragmentation Attacks

SSH:

- A proof of security (IND-sfCCA) for SSH was given in **[BKN 04]**.
- Yet **[APW 09]** presented plaintext-recovery attacks against SSH.

IPsec in MAC-then-encrypt (CBC):

- **[Kra 01]** proves that MAC-then-encrypt with CBC encryption is secure (secure channel [CK 01]).
- **[MT 10]** show that MAC-then-encode-then-encrypt (injective / CBC) is secure (secure channel [Mau 11]).
- **[DP 10]** present ciphertext-fragmentation attacks against such IPsec configurations.

Ciphertext-Fragmentation Attacks

SSH:

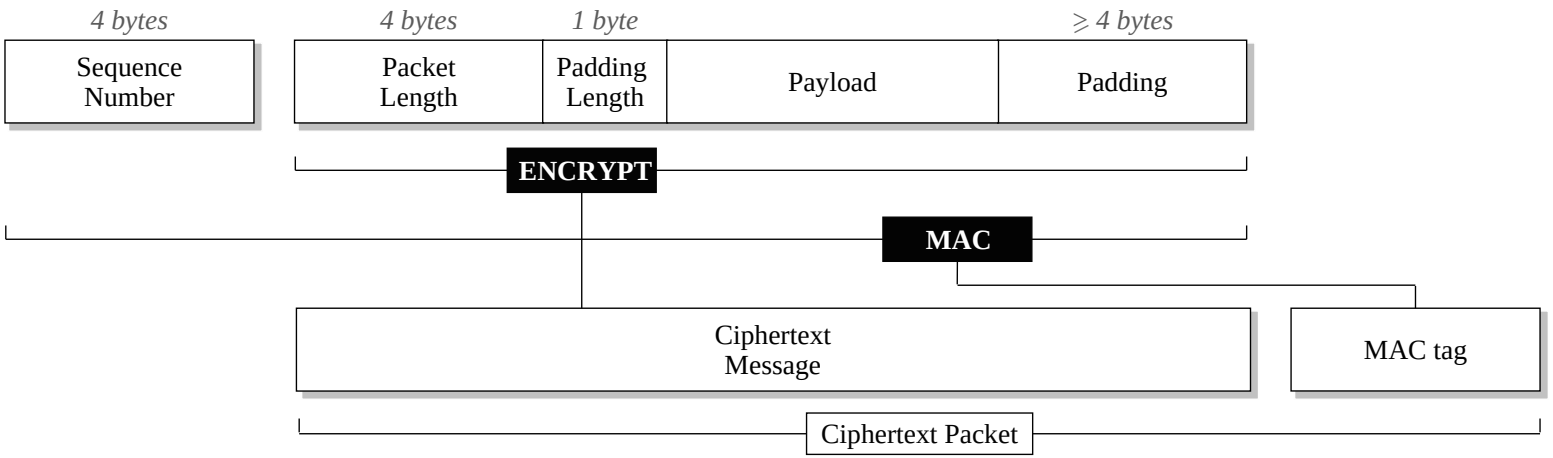
- A proof of security (IND-sfCCA) for SSH was given in **[BKN 04]**.
- Yet **[APW 09]** presented plaintext-recovery attacks against SSH.

IPsec in MAC-then-encrypt (CBC):

- **[Kra 01]** proves that MAC-then-encrypt with CBC encryption is secure (secure channel [CK 01]).
- **[MT 10]** show that MAC-then-encode-then-encrypt (injective / CBC) is secure (secure channel [Mau 11]).
- **[DP 10]** present ciphertext-fragmentation attacks against such IPsec configurations.

The Case of SSH

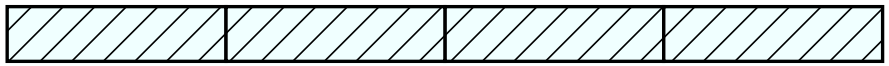
- SSH encrypts messages in the following format:



- SSH commonly uses CBC mode for encryption.

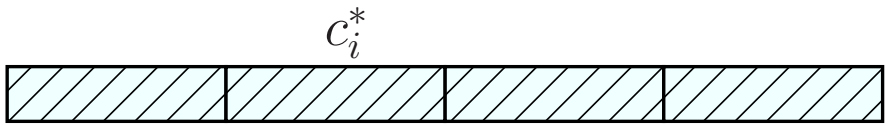
The SSH Attack (Main Idea)

Intercepted Ciphertext



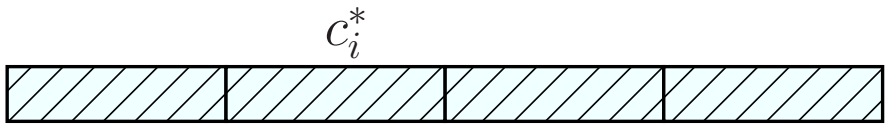
The SSH Attack (Main Idea)

Intercepted Ciphertext

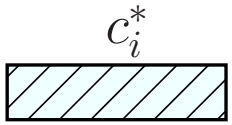


The SSH Attack (Main Idea)

Intercepted Ciphertext

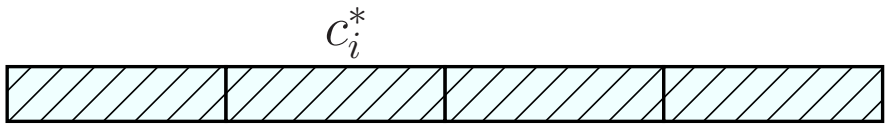


Submit for Decryption

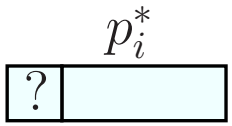


The SSH Attack (Main Idea)

Intercepted Ciphertext

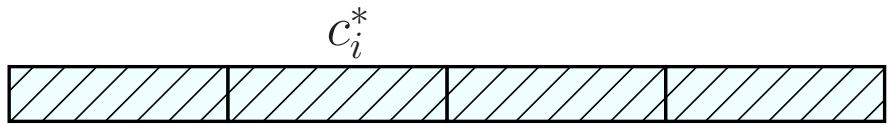


Submit for Decryption

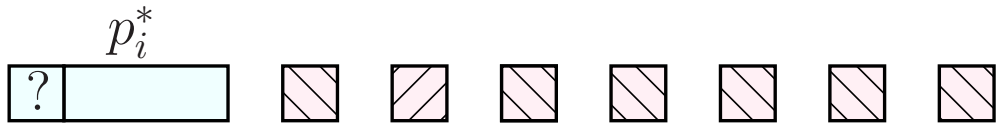


The SSH Attack (Main Idea)

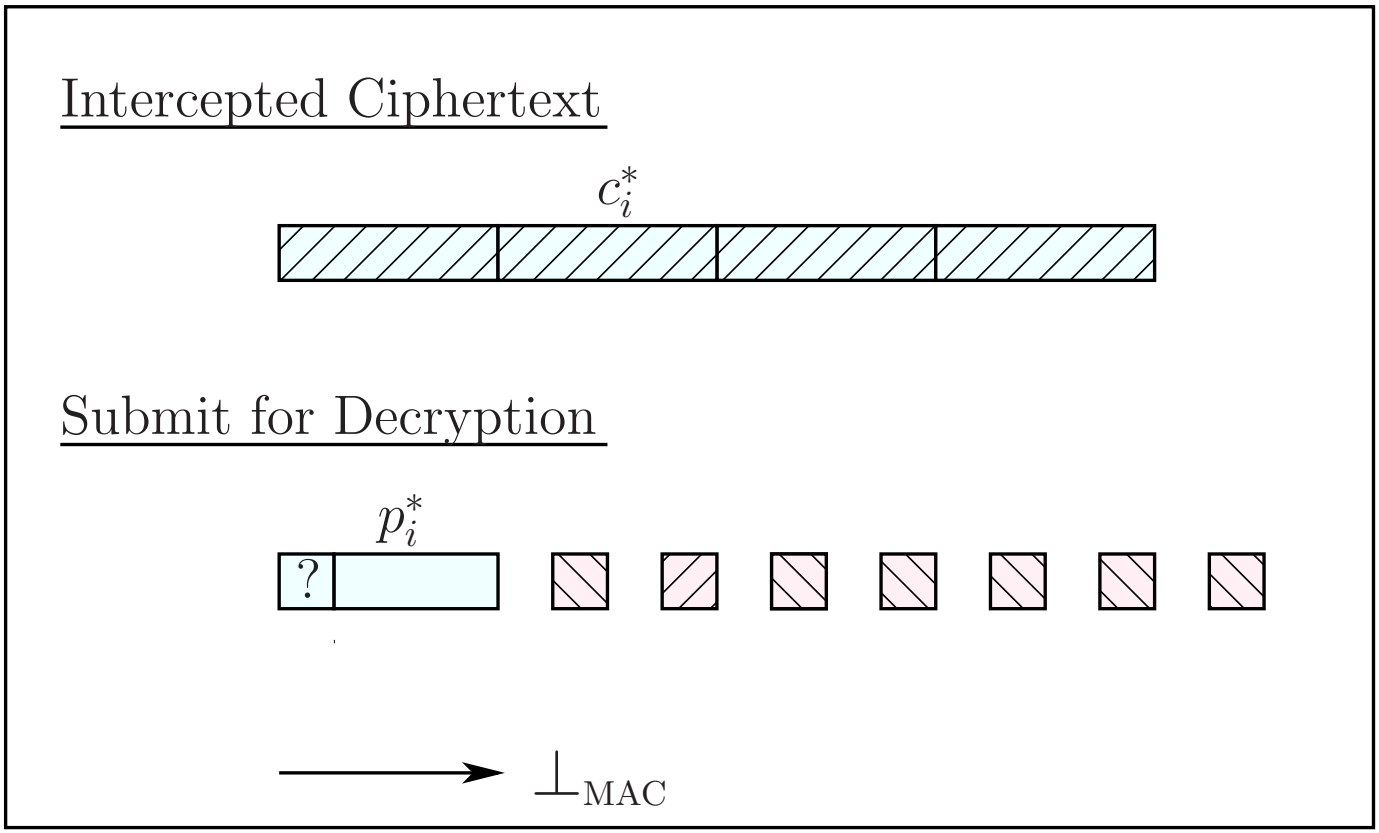
Intercepted Ciphertext



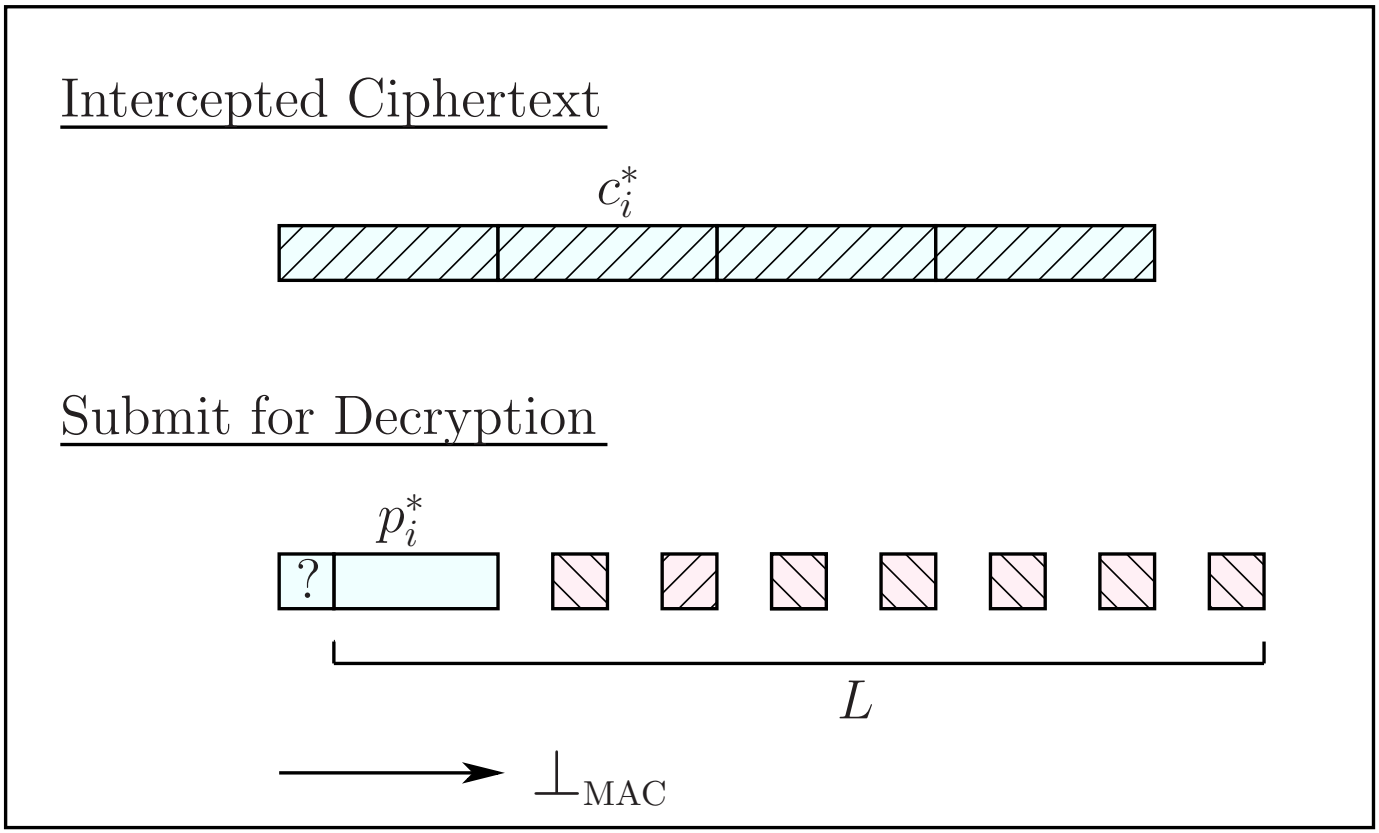
Submit for Decryption



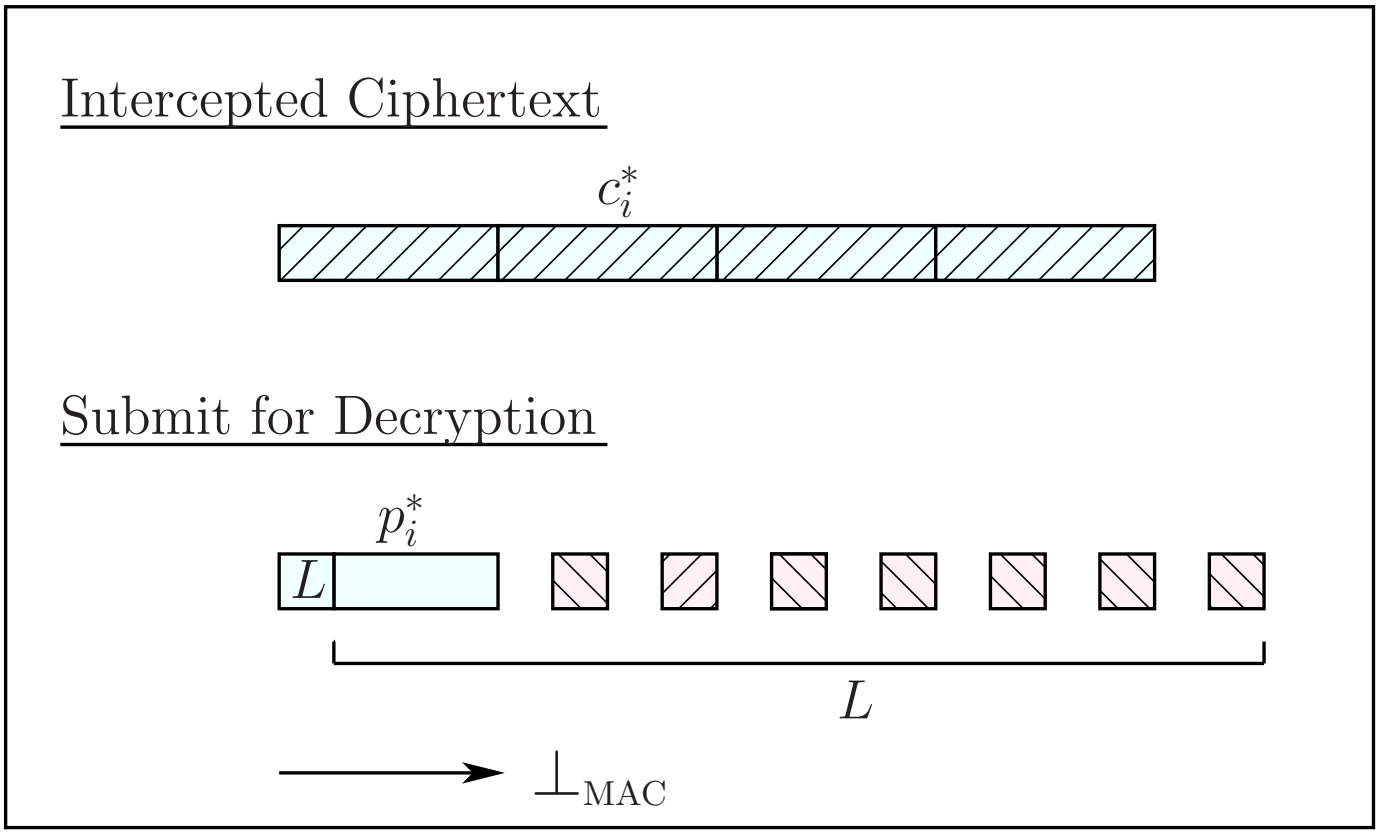
The SSH Attack (Main Idea)



The SSH Attack (Main Idea)



The SSH Attack (Main Idea)



Our Treatment of Fragmentation

From EUROCRYPT 12

Royal Holloway
University of London
Information Security Group

- We define a **syntax** and a **correctness requirement** for encryption in the fragmented setting.
- We introduce indistinguishability under **chosen-fragment attacks**.
- We identify and formalise **two other security goals** that arise in relation to ciphertext fragmentation.
- We construct a scheme, **InterMAC**, that meets all three of our security notions.

Chosen-Fragment Security

- IND-sfCCA **[BKN 04]** extends IND-CCA to protect against **replay** and **out-of-order delivery** attack.
- We extend IND-sfCCA to the fragmented setting, **IND-sfCFA** (Chosen Fragment Attack).
- We provide a **generic construction** for transforming an atomic scheme into a fragmented scheme.
- Starting from an atomic **IND-sfCCA** secure scheme, and a **prefix-free encoding**, the construction gives a fragmented scheme that is **IND-sfCFA** secure.

Chosen-Fragment Security

- IND-sfCCA **[BKN 04]** extends IND-CCA to protect against **replay** and **out-of-order delivery** attack.
- We extend IND-sfCCA to the fragmented setting, **IND-sfCFA** (Chosen Fragment Attack).
- We provide a **generic construction** for transforming an atomic scheme into a fragmented scheme.
- Starting from an atomic **IND-sfCCA** secure scheme, and a **prefix-free encoding**, the construction gives a fragmented scheme that is **IND-sfCFA** secure.

But There's More!

- Our construction shows that Chosen-Fragment Security is not that hard to achieve!
- Protocol designers aim to reduce susceptibility to **Traffic Analysis**. Heuristic approach: *Leak the least information possible*.
- Many mechanisms to support ciphertext fragmentation expose the scheme to **fragmentation-related DoS attacks**.
- Additionally meeting these security goals without compromising confidentiality is more difficult! - as exemplified by the details of the SSH attack.

Boundary Hiding

- As noted earlier **SSH encrypts the length field**. This does not conceal the message length but does hide ciphertext boundaries (in the passive case).
- **Boundary Hiding** (Informally): Given a concatenation of ciphertexts, no adversary can determine where the ciphertext boundaries lie.
- Intuitively this captures (in part) a weak form of resilience against Traffic Analysis.
- **Conflicting goals**: intuitively to support ciphertext fragmentation (correctness), the decryption algorithm requires the ability to determine ciphertext boundaries.

Denial of Service

- The SSH standard (RFC 4253) suggests limiting the maximum value of the length field in order to mitigate against certain denial-of-service attacks.
- Otherwise an adversary could alter the contents of the length field to indicate a very large value. The receiver would then interpret all subsequent ciphertexts as part of this large ciphertext – **connection hang**.
- Such denial-of-service attacks are not specific to SSH, but to encryption schemes supporting fragmentation in general.
- Informally a scheme is **N-DOS-sfCFA** secure, if no adversary can produce an N-bit long sequence of ciphertext fragments (not output by the encryption oracle) such that the decryption algorithm returns ε throughout.

Denial of Service

- The SSH standard (RFC 4253) suggests limiting the maximum value of the length field in order to mitigate against certain denial-of-service attacks.
- Otherwise an adversary could alter the contents of the length field to indicate a very large value. The receiver would then interpret all subsequent ciphertexts as part of this large ciphertext – **connection hang**.
- Such denial-of-service attacks are not specific to SSH, but to encryption schemes supporting fragmentation in general.
- Informally a scheme is **N-DOS-sfCFA** secure, if no adversary can produce an N-bit long sequence of ciphertext fragments (not output by the encryption oracle) such that the decryption algorithm returns ε throughout.

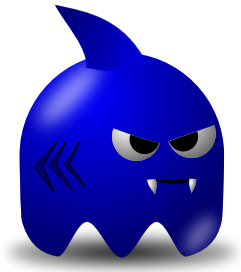
Design Criteria I

Security Under Ciphertext Fragmentation

- AE schemes are used extensively to secure communication over packet-based networks, such as TCP/IP. The channel over which the scheme has to operate is often a fragmentation channel.
- As such a good AE scheme should support or be easily extendible to **operate over fragmentation channels**.
- Specifically the scheme should **preserve confidentiality** (chosen-fragment security), and it should be **robust against fragmentation-related DoS attacks** (i.e. tunable to be N-DOS secure for a reasonable range of N values).
- We don't consider Boundary Hiding to be essential, but it would be a nice additional feature.

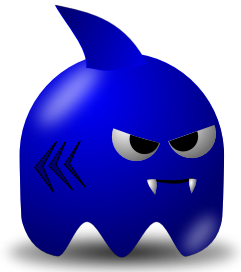
Outline

Justifying IND-CCA Security



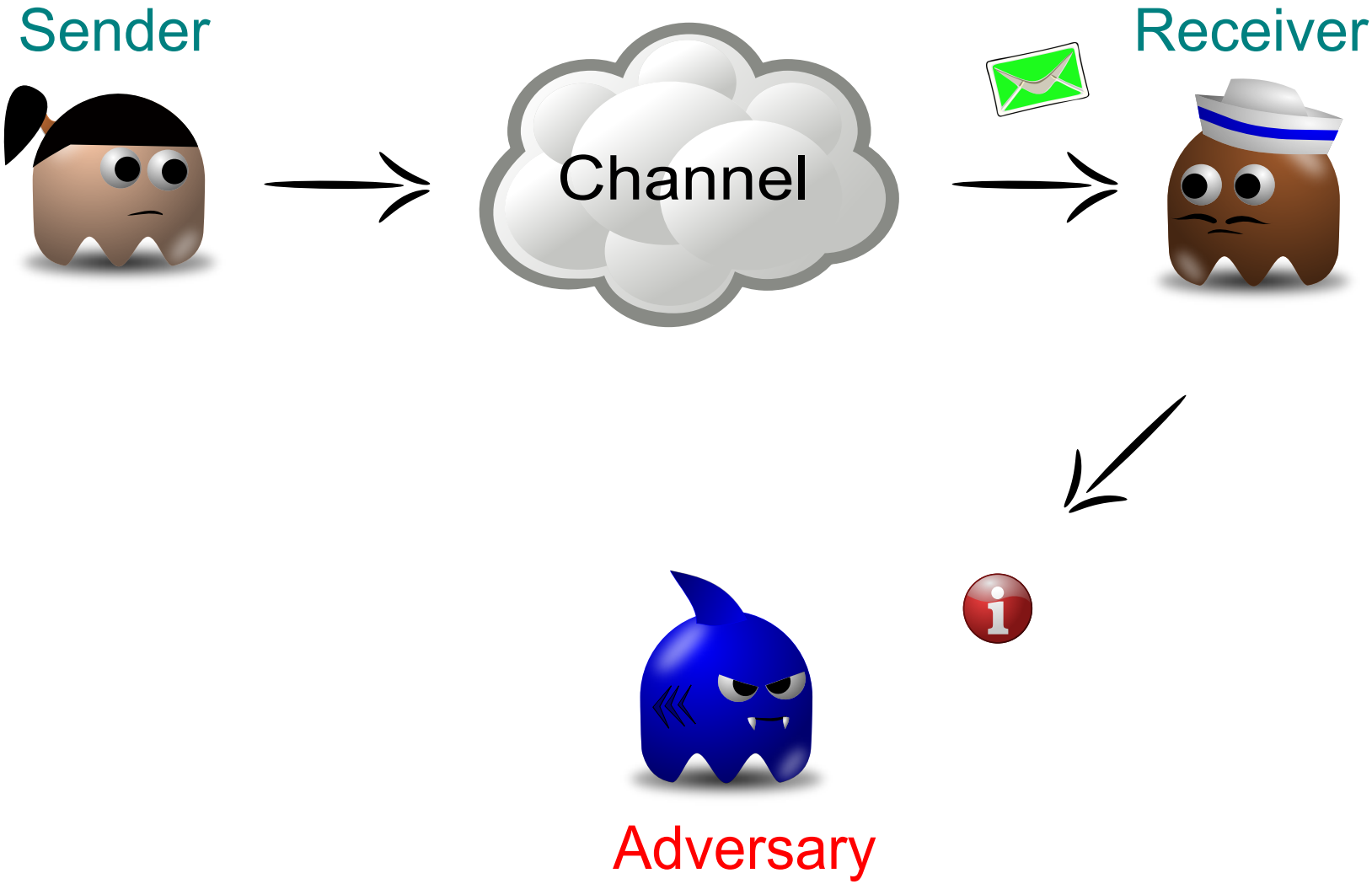
Adversary

Justifying IND-CCA Security

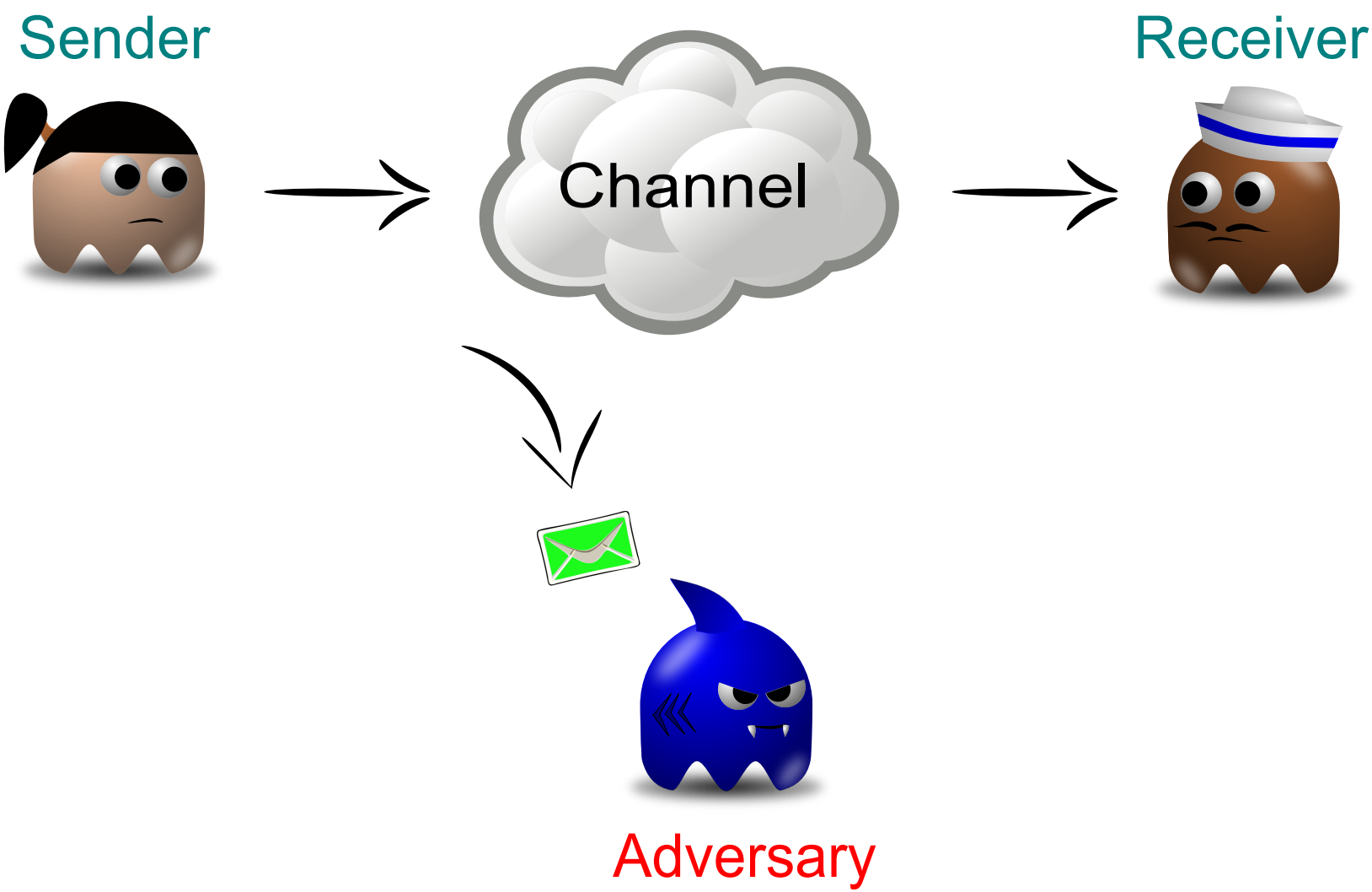


Adversary

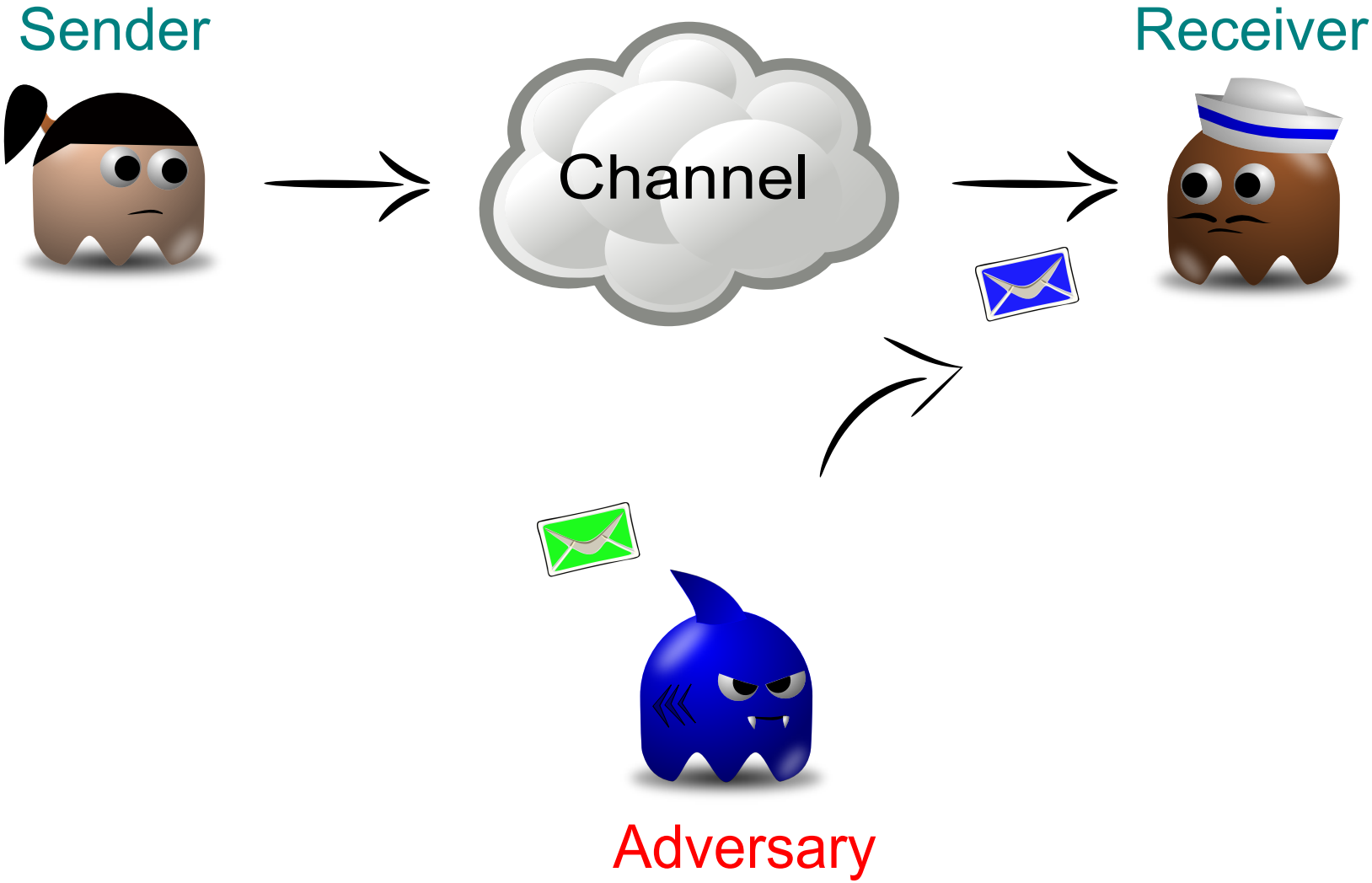
Justifying IND-CCA Security



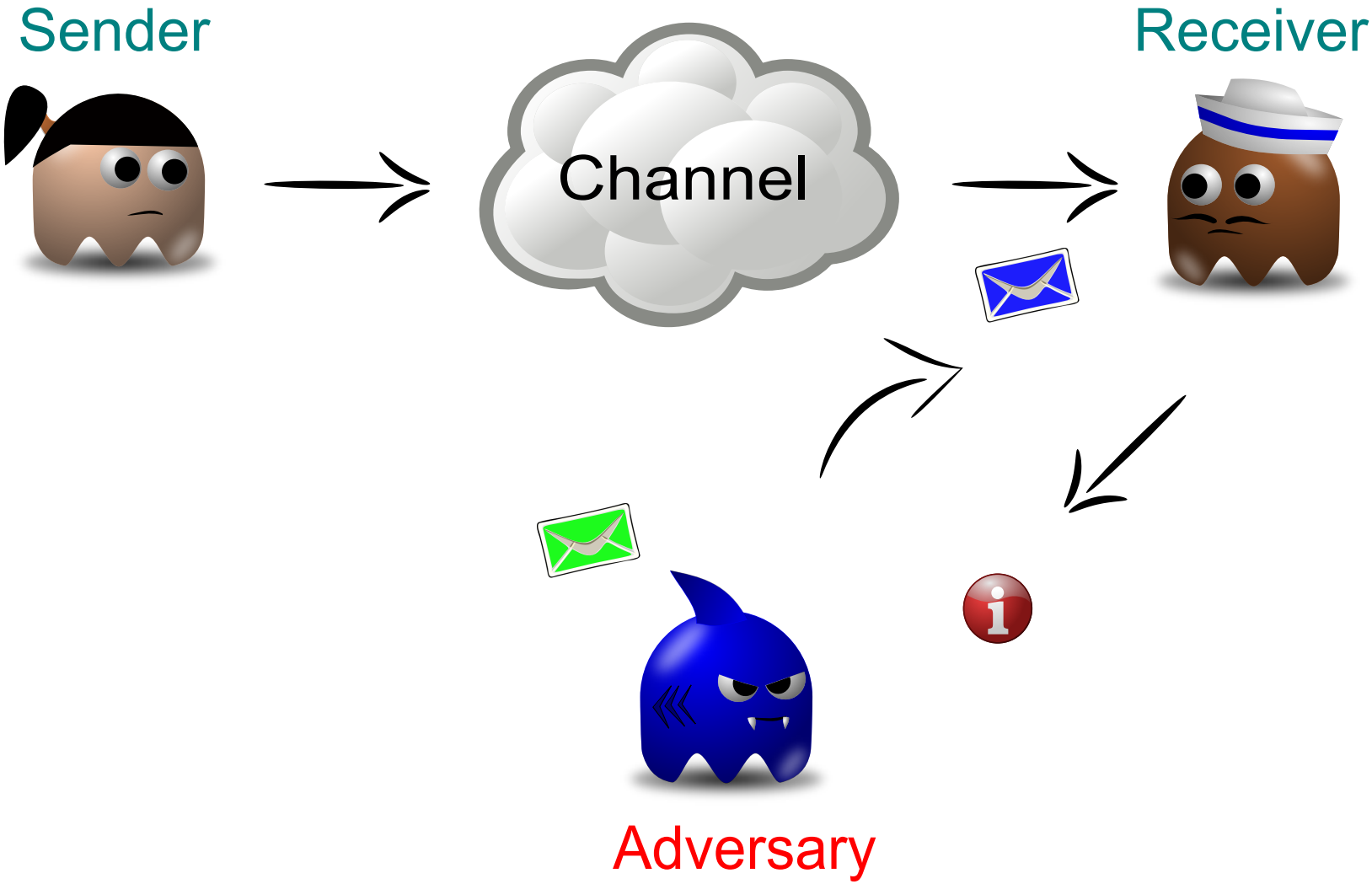
Justifying IND-CCA Security



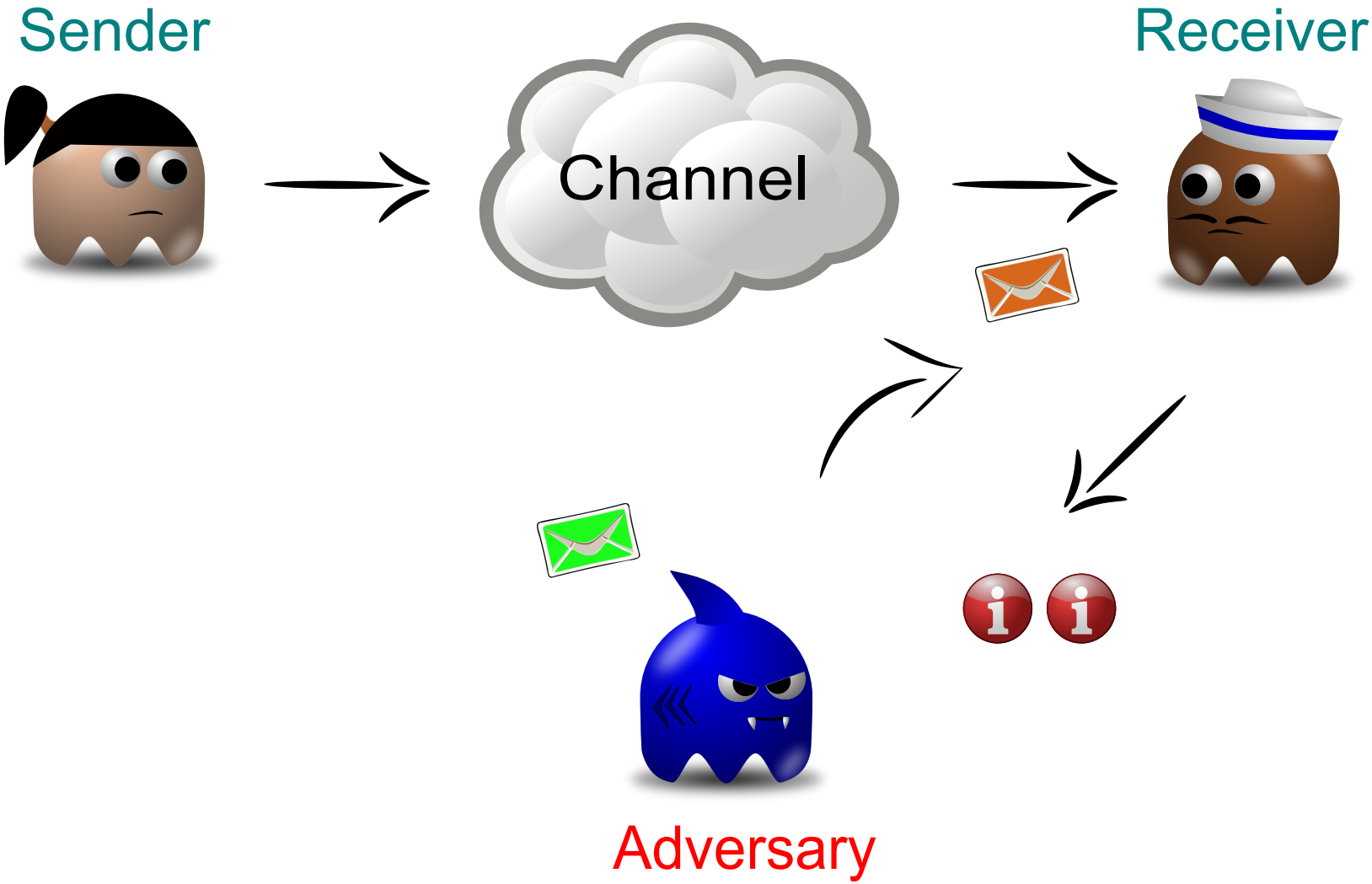
Justifying IND-CCA Security



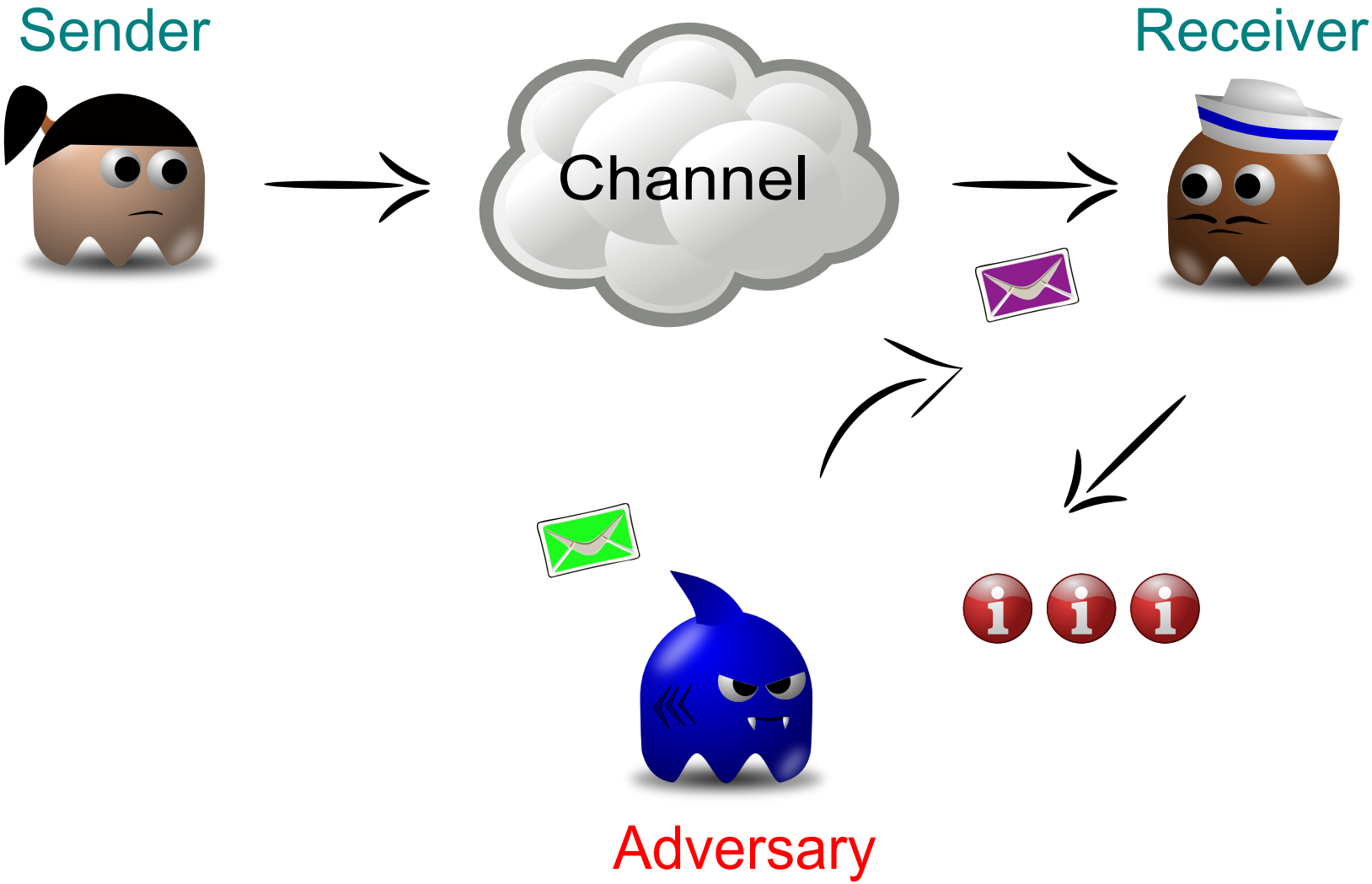
Justifying IND-CCA Security



Justifying IND-CCA Security



Justifying IND-CCA Security



A Gap in the Argument

- Krawczyk's proof can be extended to show that MAC-then-encrypt with CBC encryption is IND-CCA secure.
- Yet, attacks against TLS **[CHVV 03]**, DTLS **[AP 12]**, and IPsec **[DP 10]** have successfully managed to break instantiations of this construction through this type of attacks.
- **PROBLEM:** Our formalism implicitly assumes that an adversary cannot distinguish between decryption failures (Distinct decryption failures always return $\perp$).
- We do not capture the fact that **invalid ciphertexts** may leak more information beyond their invalidity!

A Gap in the Argument

- Krawczyk's proof can be extended to show that MAC-then-encrypt with CBC encryption is IND-CCA secure.
- Yet, attacks against TLS **[CHVV 03]**, DTLS **[AP 12]**, and IPsec **[DP 10]** have successfully managed to break instantiations of this construction through this type of attacks.
- **PROBLEM:** Our formalism implicitly assumes that an adversary cannot distinguish between decryption failures (Distinct decryption failures always return $\perp$).
- We do not capture the fact that **invalid ciphertexts** may leak more information beyond their invalidity!

Possible Remedies

- **Suggestion A:** Ensure at the implementation stage that distinct decryption failures are indistinguishable.
- History shows that this is rather difficult to achieve in practice.
- **Suggestion B:** Formulate the scheme to have distinguishable decryption failures, then prove security under this formalism.
- Thereby making security less **implementation-dependent**.
- Syntactically the resulting scheme is different, and requires revisiting some of the established relations for AE.

Possible Remedies

- **Suggestion A:** Ensure at the implementation stage that distinct decryption failures are indistinguishable.
- History shows that this is rather difficult to achieve in practice.
- **Suggestion B:** Formulate the scheme to have distinguishable decryption failures, then prove security under this formalism.
- Thereby making security less **implementation-dependent**.
- Syntactically the resulting scheme is different, and requires revisiting some of the established relations for AE.

Possible Remedies

- **Suggestion A:** Ensure at the implementation stage that distinct decryption failures are indistinguishable.
- History shows that this is rather difficult to achieve in practice.
- **Suggestion B:** Formulate the scheme to have distinguishable decryption failures, then prove security under this formalism.
- Thereby making security less **implementation-dependent**.
- Syntactically the resulting scheme is different, and requires revisiting some of the established relations for AE.

Revisiting the Basics

- Bellare and Namprempre introduced the following simple but useful relations:

  $\text{IND-CPA} \wedge \text{INT-CTXT} \Rightarrow \text{IND-CCA}$ $\quad$ $\text{INT-CTXT} \Rightarrow \text{INT-PTXT}.$

- Since then INT-CTXT and IND-CPA have become the commonly accepted security benchmark for AE.

- Rogaway and Shrimpton later elegantly combined these two notions into a single one, sometimes referred to as IND-CCA3.

Revisiting the Basics

- Bellare and Namprempre introduced the following simple but useful relations:

  $\text{IND-CPA} \wedge \text{INT-CTXT} \Rightarrow \text{IND-CCA}$ $\qquad$ $\text{INT-CTXT} \Rightarrow \text{INT-PTXT}.$

- Since then INT-CTXT and IND-CPA have become the commonly accepted security benchmark for AE.

- Rogaway and Shrimpton later elegantly combined these two notions into a single one, sometimes referred to as IND-CCA3.

Revisiting the Basics

- For schemes with multiple errors there exists the following separation:

$$\text{IND-CPA} \wedge \text{INT-CTXT} \not\Rightarrow \text{IND-CCA}.$$

- Hence IND-CPA and INT-CTXT no longer guarantee strong confidentiality, and are not suited as the target security notion for multiple-error AE schemes.

- In ongoing work we extend these security notions to the multiple error case (non-trivial), and we establish similar relations in order to guarantee IND-CCA security and INT-PTXT.

Revisiting the Basics

- For schemes with multiple errors there exists the following separation:

$$\text{IND-CPA} \wedge \text{INT-CTXT} \not\Rightarrow \text{IND-CCA}.$$

- Hence IND-CPA and INT-CTXT no longer guarantee strong confidentiality, and are not suited as the target security notion for multiple-error AE schemes.

- In ongoing work we extend these security notions to the multiple error case (non-trivial), and we establish similar relations in order to guarantee IND-CCA security and INT-PTXT.

Design Criteria II

Distinguishable Decryption Failures

- A good AE scheme should have **implementation-robust security**; in the sense that security should be preserved in most 'reasonable' practical realisations of the scheme.
- To guard against **distinguishable decryption failures**, each test condition in the decryption algorithm that determines ciphertext validity should return a unique error symbol.
- Chosen-ciphertext security should then be proved for the resulting multiple-error scheme.
- Alternatively the approach could be applied to a scheme's **implementation** to verify that it remains secure.