Cryptography for Secure Channels

Kenny Paterson

Information Security Group
Royal Holloway, University of London

kenny.paterson@rhul.ac.uk

Outline

Royal Holloway
University of London

Introduction

Introduction

Royal Holloway
University of London

- The aim of these two talks is to highlight some of the challenges in the design, implementation and security analysis of secure channel protocols.
- IPsec, SSL/TLS, SSH as case studies.
- Some theory.
- Lots of practice!
- Some lessons.

The Secure Channel Concept

Royal Holloway
University of London

- We often want to guarantee the confidentiality and integrity of data travelling over untrusted networks.
- Applications:
  - Branch office connectivity.
  - Connecting to business partners at remote site.
  - Remote access for employees.
  - Remote administration of network devices and servers.
  - E-commerce: protecting credit card numbers in transactions.
  - Secure file transfers.
  - …

The Secure Channel Concept

Royal Holloway
University of London

- Achieved by building a *secure channel*.
- Typically this secure channel will offer:
  - Data origin authentication
  - Data integrity
  - Confidentiality
  - Anti-replay
- But usually not:
  - Non-repudiation
  - Any security services once data has been received

The Secure Channel Concept

Royal Holloway
University of London

Secure channels are usually constructed in 3 steps:

- An authenticated key establishment protocol.
  - During this one or both parties is authenticated and a fresh, shared secret is established.
  - Using asymmetric (public key) or symmetric cryptography, or a combination of the two.
- A key derivation phase.
  - MAC and symmetric encryption keys are derived from the shared secret established during protocol.
- And then further traffic protected using derived keys.
  - MAC gives data integrity mechanism and data origin authentication.
  - Encryption gives confidentiality.
    - Usually mode of operation of block cipher (CBC, CTR) or stream cipher.
  - Using symmetric cryptography for speed.

Royal Holloway
University of London

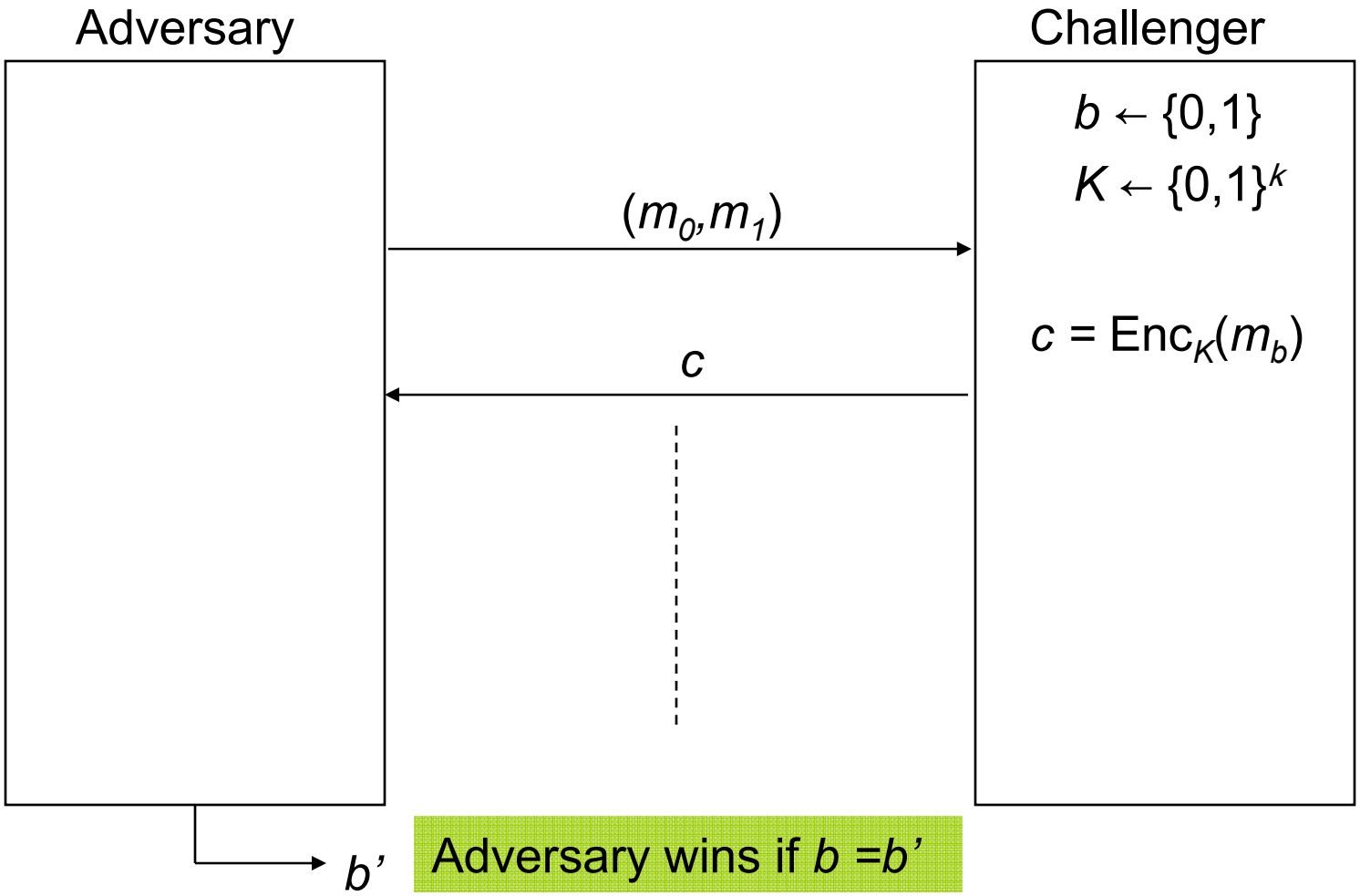
Introduction

Theory for Secure Channels

Short Theoretical Interlude

- Security models for symmetric encryption are well established.
- SE = (KGen,Enc,Dec)
- IND-CPA security:
  - Adversary has access to *Left-or-Right (LoR)* encryption oracle.
  - Adversary submits pairs of equal length messages (m_0,m_1) to the oracle.
  - Receives *c*, an encryption of either m_0 or encryption of m_1.
  - Adversary has decide which message c encrypts.
  - Adversary wins if it decides correctly.
- Formalised as a security game between the adversary and a challenger.

IND-CPA Security



IND-CPA Security

Royal Holloway
University of London

- Adversary's advantage is defined to be:

$$|\Pr(b = b') - 1/2|.$$

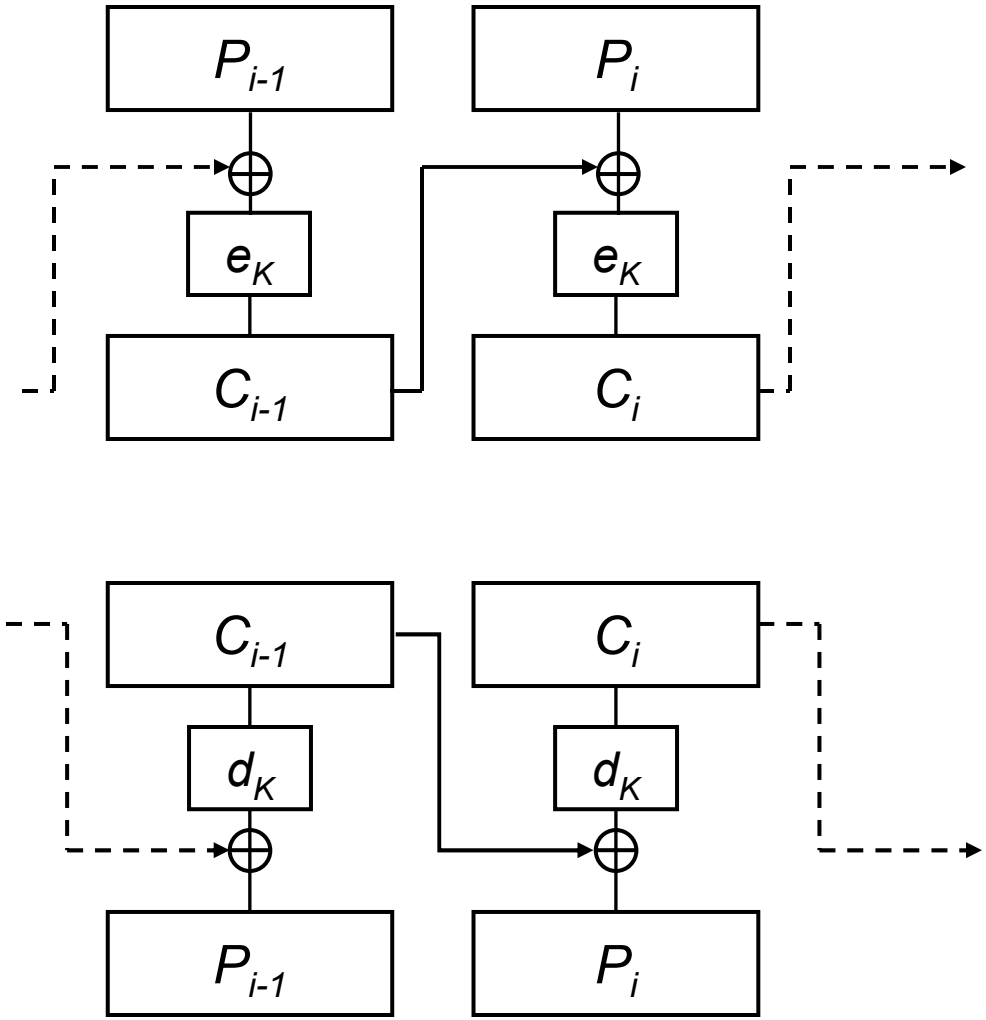
- Scheme SE is said to be IND-CPA secure if advantage is "small" for any adversary using "reasonable" resources.
  - Concepts of "small" and "reasonable" can be formalised using either an asymptotic approach or a concrete approach.
  - Requires non-deterministic (randomised) encryption.
  - IND = Indistinguishable.
  - CPA = Chosen Plaintext Attack.

IND-CPA Security

Royal Holloway
University of London

- Informally, IND-CPA is a computational version of perfect security.
  - Ciphertext leaks nothing about the plaintext.
  - Stronger notion than requiring the adversary to recover plaintext.
- Easy to achieve using suitable mode of operation of block cipher:
  - Block cipher in CBC mode with random IVs;
  - Block cipher in CTR mode
  - See [BDJR97] for analysis.
  - Requires modelling of block cipher as PRF.

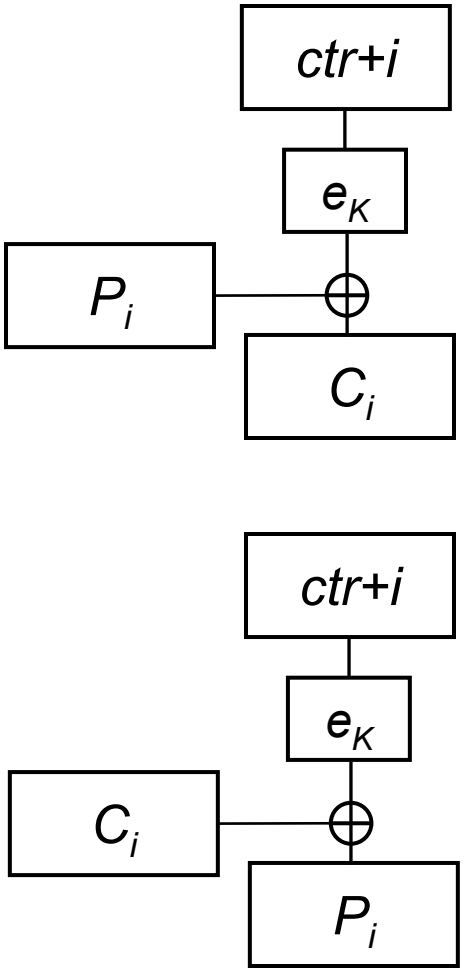
Reminder of CBC Mode



Initialisation Vector (IV):
- Defines C_0 for processing first block.
- IV often taken as random;
- Chained IVs also common

CBC mode needs some form of padding if plaintext lengths are not multiple of block length.
- More on padding later.

Reminder of CTR Mode



- CTR mode uses a block cipher to build a stream cipher.
- Random initial value chosen for *ctr*.
- Encrypt blocks

  ctr, ctr+1, ctr+2, …

  to create a sequence of ciphertext blocks.
- Use this sequence as keystream.
- Same process to decrypt.
- LoR-CPA secure assuming block cipher is a PRF.

Motivating Stronger Security

Royal Holloway
University of London

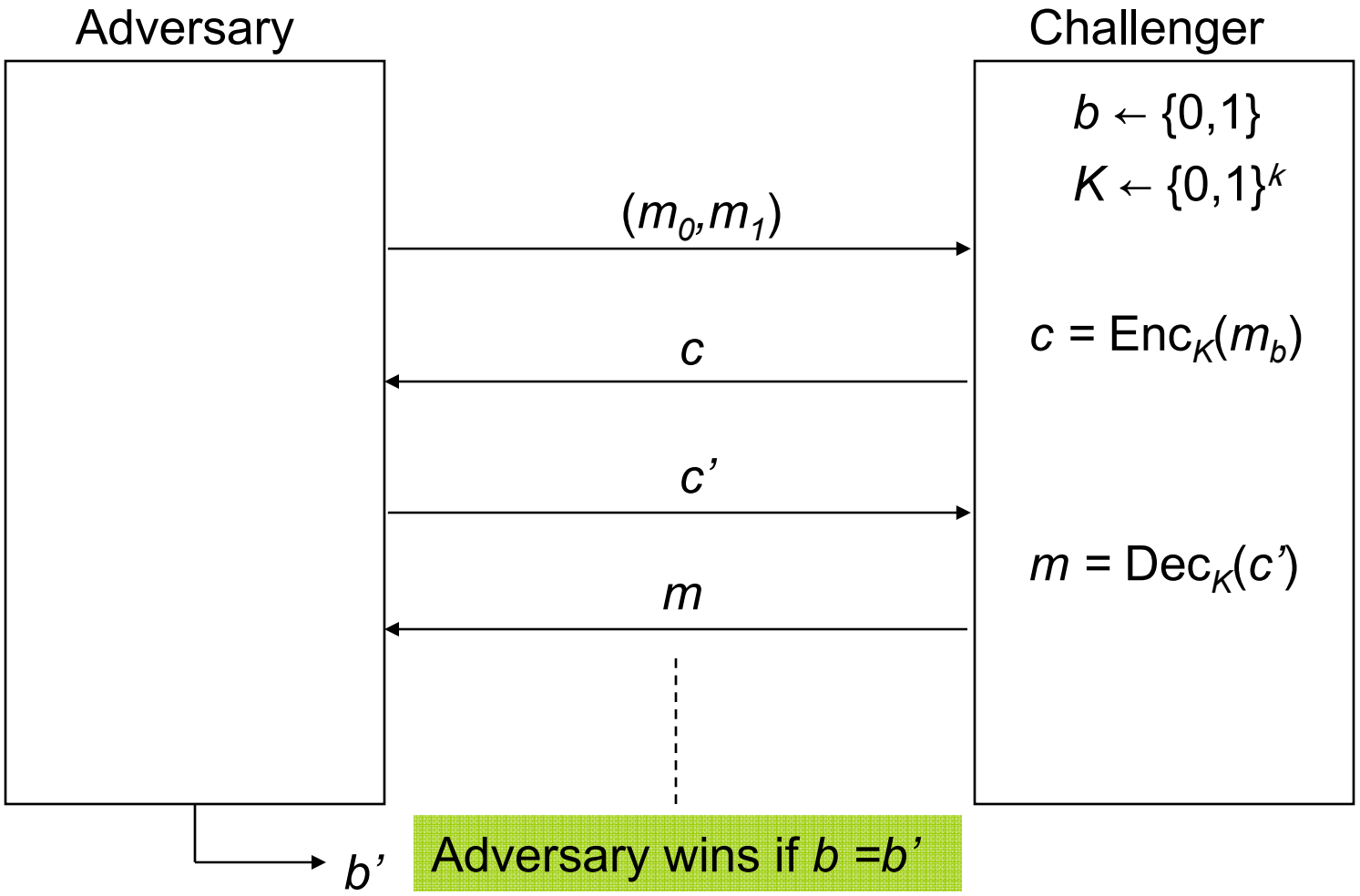
- With CBC mode and CTR mode, an adversary can manipulate ciphertexts.
  - Modify *c* to *c'* and change the underlying plaintext from *p* to *p'.*
  - Does not break IND-CPA security, but is clearly undesirable.
  - We really want *non-malleable* encryption, guaranteeing integrity as well as confidentiality.
- We may also want to consider chosen-ciphertext attacks, where adversary can get ciphertexts of his choice decrypted.
  - This may arise in practice depending on application.

IND-CCA Security

Royal Holloway
University of London

- IND-CCA security:
  - Attacker now has access to LoR encryption oracle and decryption oracle.
  - LoR encryption oracle as before.
  - Decryption oracle takes any c as input, and outputs either $\mathrm{Dec}_K(c)$, which is either a message m or a failure symbol $\perp$.
  - Adversary not permitted to submit output of LoR encryption oracle to its decryption oracle.
- All basic modes of operation are insecure in this model!

IND-CCA Security

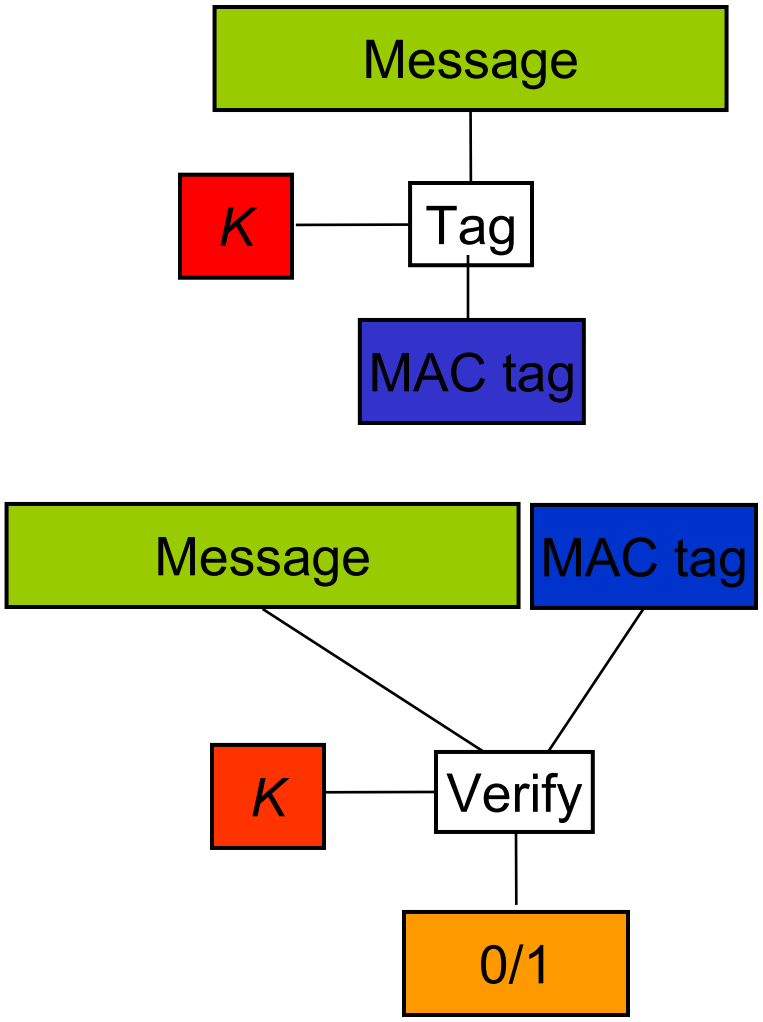


MACs

Royal Holloway
University of London

- Message Authentication Codes (MACs) provide authenticity/integrity protection for messages.
- Symmetric analogue of a digital signature.
- MAC = (KGen,Tag,Verify)
- Algorithm Tag has as input a key *K*, a message *M* of arbitrary length, and outputs a short MAC tag τ.
- Algorithm Verify has as input a key *K*, a message *M,* a MAC tag τ and outputs 0 or 1, indicating correctness of tag τ.
- HMAC is a general method for building a MAC scheme from a hash function.
  - Very widely used in secure protocols.

MACs



- Key security requirement is *unforgeability*.
- Having seen MAC tags for many chosen messages, an adversary cannot create the correct MAC tag for another chosen message.
- Strong and weak forms of unforgeability:
  - New MAC tag on (possibly) queried message versus MAC tag on unqueried message.
  - SUF-CMA and (W)UF-CMA security

Achieving IND-CCA Security

Royal Holloway
University of London

- [BN00] considered how to achieve IND-CCA security by combining IND-CPA secure encryption and (S)UF-CMA secure MAC.

- Encrypt-then-MAC: SECURE
  - As used by IPsec ESP enc+auth.
- Encrypt-and-MAC: INSECURE
  - MAC can leak plaintext information.
  - But specific instantiations may be IND-CCA secure, e.g. as used in SSH [BKN02].
- MAC-then-Encrypt: INSECURE
  - But MAC followed by CBC mode encryption or stream cipher is IND-CCA secure (extension of [K01]).
    - As used by SSL/TLS.
- Also available: dedicated authenticated encryption algorithms – e.g. CCM, EAX, GCM, OCB.

Stateful Security

Royal Holloway
University of London

- [BKN02] developed *stateful* security models for symmetric encryption.
  - Reflecting wide use of sequence numbers in secure channel protocols.
- IND-SFCCA security:
  - Attacker has access to an LoR encryption oracle and a decryption oracle.
  - Both oracles are stateful (e.g. sequence numbers).
  - Model allows adversary to advance states to any chosen value via queries to LoR encryption and decryption oracles.
  - Adversary wins game if he can guess hidden bit *b* of encryption oracle.

Royal Holloway
University of London

Introduction

Theory for Secure Channels

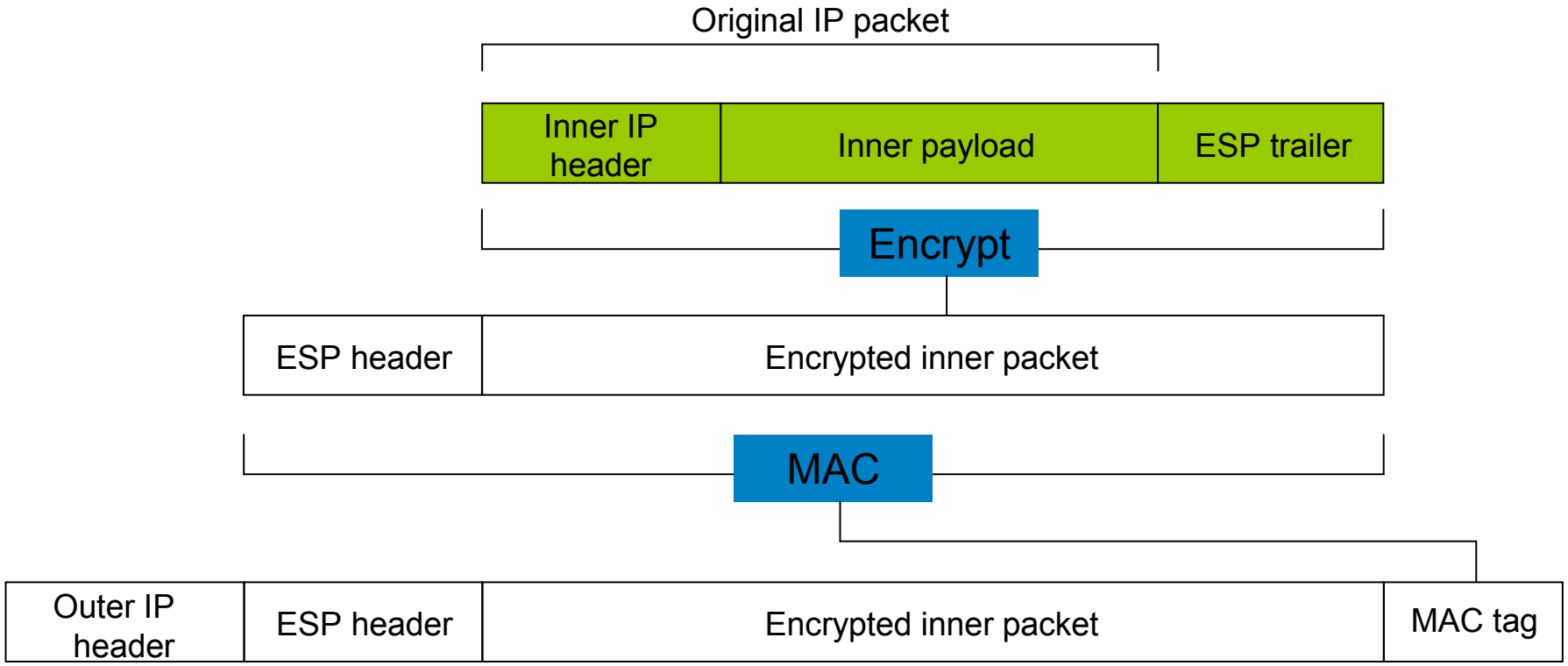
Evolution of Encryption in IPsec

Encryption in IPsec

Royal Holloway
University of London

- IPsec provides security at the IP layer.
  - Cryptographic protection of IP packets and their payloads.
- Widely used in VPNs and remote access systems.
- ESP = Encapsulating Security Protocol.
  - v1, v2, v3 in IETF RFCs 1827, 2406, 4303.
  - IPsec's "encryption workhorse".
- ESP provides one or both of:
  - Confidentiality for packet/payload (v1, v2, v3).
  - Integrity protection for packet/payload (v2, v3).
- ESP uses symmetric encryption and MACs.
  - Usually CBC mode of block cipher for encryption.
  - HMAC-SHA1 and HMAC-MD5 for integrity protection.
  - Increasing support for authenticated encryption algorithms.

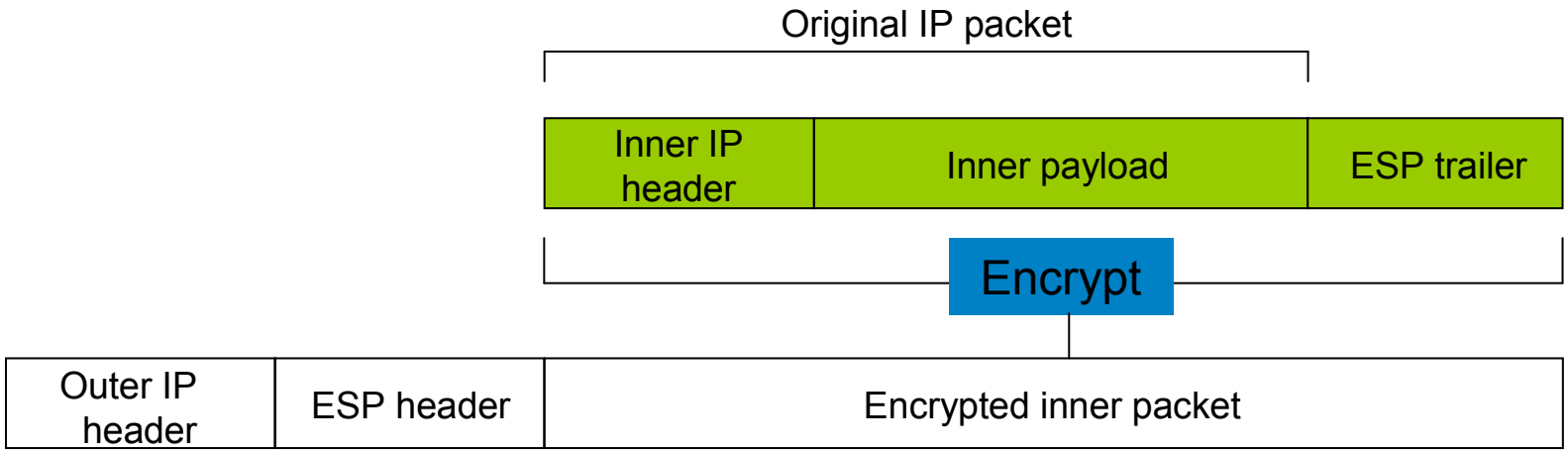
ESP in Tunnel Mode



- Encrypt-then-MAC construction, generically secure [BN00].
- ESP trailer contains padding, pad length and next header field.
- ESP commonly uses a block cipher in CBC mode for encryption.

Integrity Protection and ESPv1

- ESPv1 (RFC 1827) provided no integrity protection.
  - Reliant on separate AH protocol to provide this.



Integrity Protection and ESPv1

Royal Holloway
University of London

- Bellovin's attack on ESPv1 without AH:
  - Recovers one byte of plaintext from each TCP segment.
  - Uses ciphertexts matching 2^{24} chosen plaintexts.
  - Requires receiver to ignore encryption padding format.
    - Attack fails if padding format check carried out upon decryption.
  - An example showing that IND-CPA secure encryption on its own is not enough to prevent *active* attacks.

Integrity Protection and ESPv2

Royal Holloway
University of London

- IETF response to Bellovin's attack:
  - ESPv2 (RFC 2406) recommends receiver should check format of encryption padding.
    - A measure sufficient to stop Bellovin's attack.
  - ESPv2 also includes integrity protection as an option.
    - Effectively an Encrypt-then-MAC construction.
    - Generically secure [BN00].
  - But ESP implementations **must** still support "encryption-only" mode.
- ESPv2 represents a compromise between improving security and maintaining backwards-compatibility.

Integrity Protection and ESPv3

Royal Holloway
University of London

- ESPv3 gives strong warnings about Bellovin's attack and refers to theoretical cryptography literature to motivate need to use integrity protection.

> *"Using encryption without a strong integrity mechanism on top of it (either in ESP or separately via AH) may render the confidentiality service insecure against some forms of active attacks [Bel96, Kra01]. Moreover, an underlying integrity service, such as AH, applied before encryption does not necessarily protect the encryption-only confidentiality against active attackers [Kra01]."*

Integrity Protection and ESPv3

Royal Holloway
University of London

- ESPv3 (RFC 4303) still allows encryption-only ESP:

  "ESP allows encryption-only … because this may offer considerably better performance and still provide adequate security, e.g., when higher layer authentication/integrity protection is offered independently."

- Implying some kind of MAC-then-encrypt construction.
  - Known to be generically *in*secure [BN00], [K01].
  - But provably secure in certain combinations, e.g. MAC-then-CBC-mode [K01].
- But ESPv3 no longer *requires* support for encryption-only mode.

Encryption-only ESP in the Real World

Royal Holloway
University of London

- The theoretical cryptography community is well aware of the need to carefully combine integrity protection with encryption.
  - To achieve IND-CCA security and so prevent active attacks against confidentiality.
- It is also well-known amongst IPsec experts that encryption-only configurations should be avoided.
  - Clear warnings against their use in the RFCs.
  - Bellovin attack.
- So is there really any problem if encryption-only modes continue to be allowed in the RFCs?

Encryption-only ESP in the Real World

Royal Holloway
University of London

- Developers are *required* by RFC 2406 to support encryption-only ESP.
- Developers rarely pass RFC warnings to end users.
- Developers don't properly implement RFCs (see later).
- End users probably don't read RFCs or technical papers.
- End users might reasonably assume that encryption on its own gives confidentiality.
- On-line tutorials sometimes do not highlight the dangers of encryption-only IPsec…

Encryption-only ESP in the Real World

From the IPsec Tunnel Implementation administrator's guide of a well-known vendor (until quite recently):

> *"If you require data confidentiality only in your IPSec tunnel implementation,* ***you should use ESP without authentication****. By leaving off the authentication service, you gain some performance speed but lose the authentication service."*

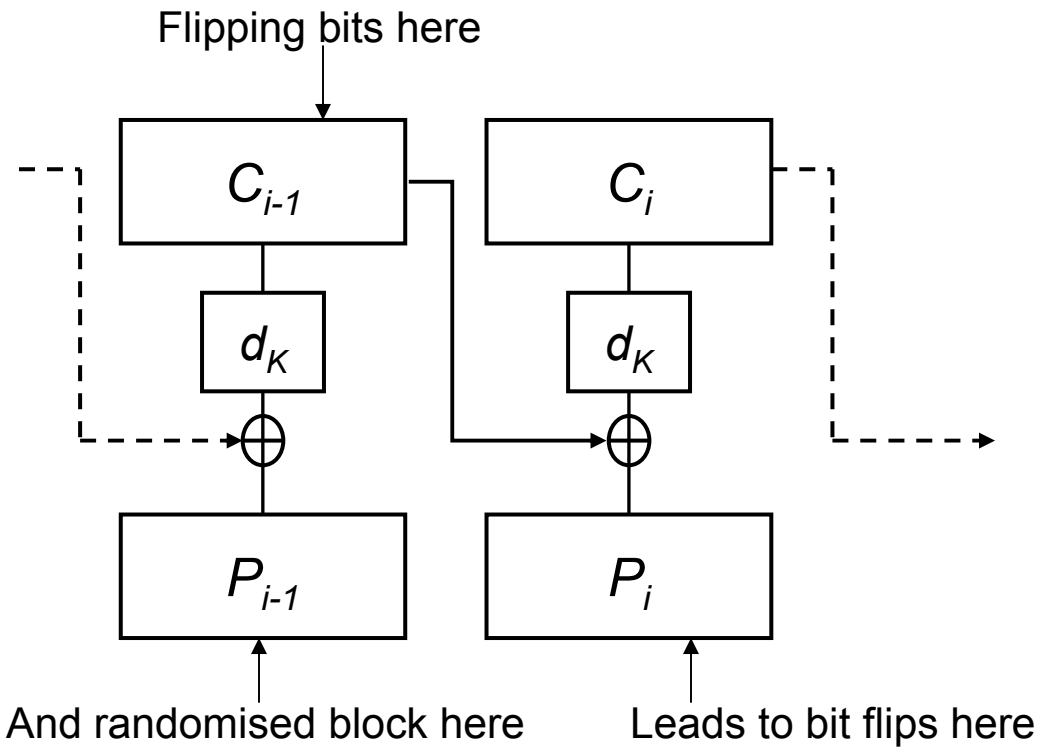
Attacking Encryption-only ESP

Royal Holloway
University of London

- [PY06]:
  - Efficient ciphertext-only attacks against encryption-only ESP.
  - Implemented in a realistic lab setting against the Linux implementation.
  - Attacks do **not** work for implementation strictly following the RFCs.
- [DP07]:
  - Ciphertext-only attacks against encryption-only ESP.
  - But less efficient than Linux attacks.
  - But attacks should work for **any** implementation strictly following the RFCs.
  - Implemented against OpenSolaris implementation.

Bit Flipping in CBC Mode

- Flipping bits in ciphertext block C_{i-1} leads to controlled changes in plaintext block P_i upon decryption.
- But block P_{i-1} is randomised.



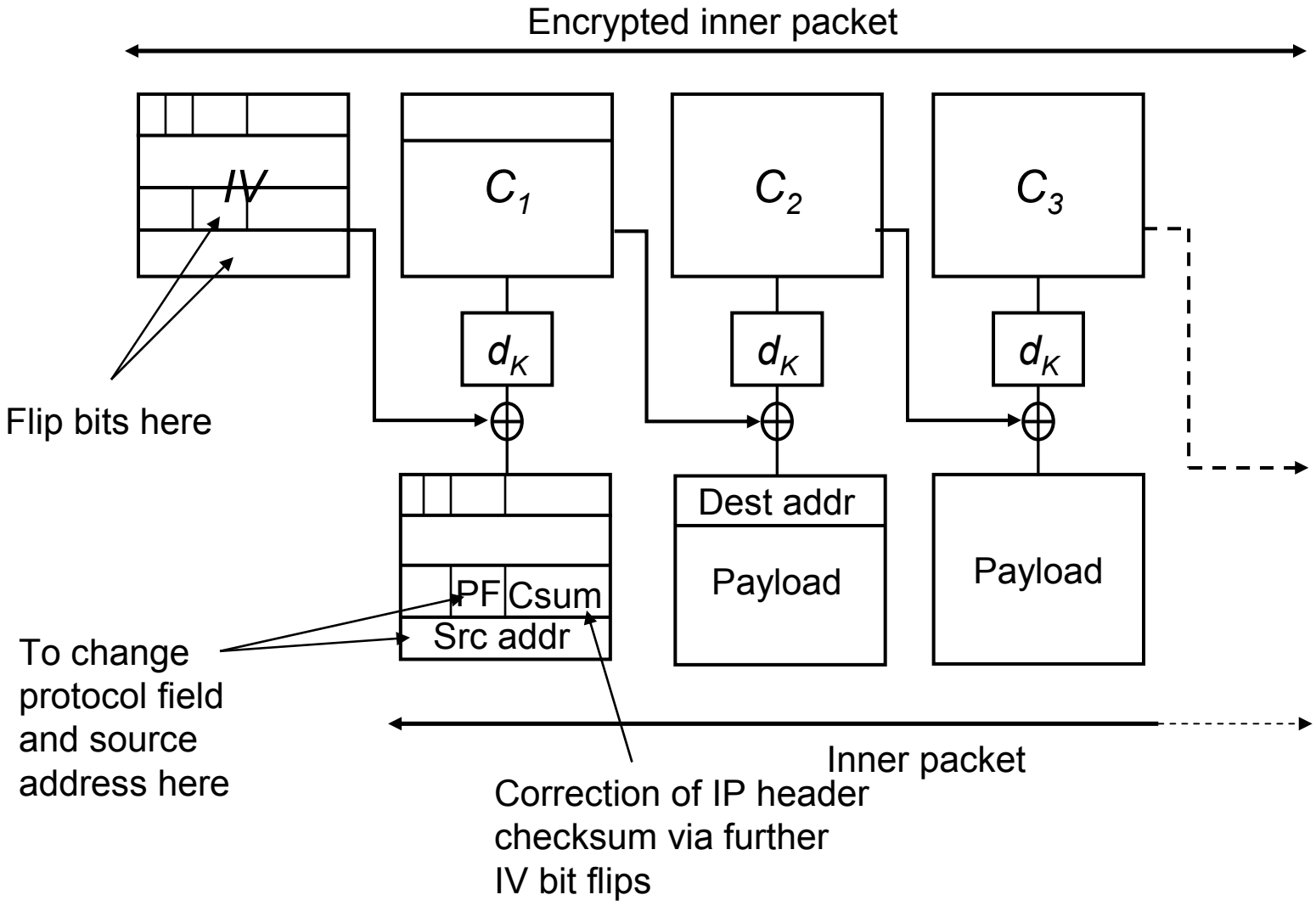
Attacking Linux ESP

Royal Holloway
University of London

Paterson and Yau [PY06]:

- Exploit bit flipping weakness of CBC mode encryption.
- Capture packets from network.
- Modify headers of inner packets so they produce error messages when processed by IP.
  - Error messages are carried by ICMP and reveal partial plaintext data.
- Modify headers so that error messages are sent outside of IPsec tunnel.
- Inject modified packets back into network.

Sketch: Protocol Field Manipulation for 128-bit Blocks



Sketch (ctd.)

Royal Holloway
University of London

- (Modified) inner packet is recovered upon decryption, and forwarded to the host indicated in destination address field.
- This host generates an ICMP “protocol unreachable” message in response to the (modified) protocol field in header.
- In Linux, ICMP payload carries inner packet header and up to 528 bytes of inner packet’s payload.
  - Payload now in plaintext form.
  - ICMP message is sent to host indicated in source address.
  - And we have modified this so ICMP message does not pass through IPsec tunnel.
- Attacker intercepts ICMP message to get plaintext bytes.

Characteristics of Linux Attacks

Royal Holloway
University of London

The full attacks:

- Recover complete contents of IPsec-protected datagrams.
  - But not encryption keys.
- Are efficient.
  - Recover 500 bytes of plaintext per attacker-injected packet.
- Do not require special operating conditions.
  - Attacker needs to capture packets from network, inject packets into network, and monitor gateway for ICMP messages.
- Worked in practice against Linux implementation of IPsec.

Attacking Linux ESP – Reactions

Reactions to the Linux ESP attacks:

"…the possibility of active attacks on unauthenticated but encrypted ESP packets is well known, and we advise against such use in the most recent set of IPsec documents. These documents have been approved for publication by the IESG and are in the queue to be published as RFCs. As a result, no further, substantiative changes will be made."

"This is only an attack against a specific IPsec implementation; the attacks would not work if RFCs were properly followed."

Attacking the RFCs – Main Ideas

Royal Holloway
University of London

Degabriele and Paterson [DP07]:

- Extension of *padding oracle* attacks on CBC mode from [V02] combined with previous techniques.
  - Padding oracle attacks will be discussed soon.
- Attack should work if (and only if) implementer has followed *all* the advice in IPsec RFCs:
  - Check correctness of padding format, as per RFCs 2406, 4303.
  - Post-processing policy checks (omitted in Linux).
- Resulting attack is less efficient than Linux attack, but still recovers all plaintext.
  - About 2^{16} packet injections to decrypt each block.
- So are any implementations strict enough?

Implementing the Attacks

- Linux:
  - Comment in source code:

```
/* ... check padding bits here. Silly. :-) */}
```

  - No padding check implemented, contrary to advice in RFCs 2406, 4303.
  - So our RFC attacks don't apply.
  - But then vulnerable to Bellovin's attack from 1995!
  - And a variant of the RFC attack which can efficiently extract 2 bytes per block.
    - Which we implemented as a proof of concept.
  - And the attacks from [PY06].

Implementing the Attacks

Royal Holloway
University of London

- KAME, OpenBSD, FreeBSD, NetBSD, MacOS X:
  - Crude padding check: check if pad length byte is 0 or if pad length byte = last byte of padding.
  - Not inconsistent with RFCs!
  - Not rigorous enough for the RFC attacks to work.
  - But a variant of the RFC attacks extracts 3 bytes per block for 2^{16} effort.

Implementing the Attacks

Royal Holloway
University of London

- Openswan, strongSwan, FreeS/WAN:
  - Don't allow selection of encryption-only configurations (despite mandated support in ESPv2).
    - This can be over-ridden manually.
  - All check padding carefully, but then don't drop packet if it's incorrectly padded.
    - Not inconsistent with the RFCs!
    - (RFCs don't explicitly mandate drop, but then what's the point of doing the check?)
  - So the RFC attacks won't work, but Bellovin's will.

Implementing the Attacks

Royal Holloway
University of London

- OpenSolaris
  - Explicit warning to user if encryption-only mode selected.
  - 3 different levels of padding check can be selected.
    - No check, KAME-style check, full padding check.
  - But the full check was incorrectly implemented!
  - We reported the bug to Sun.
  - They then fixed it in Release 55 of OpenSolaris.
  - After which, we were able to successfully attack the OpenSolaris implementation.
  - Attack complexity in line with theoretical results.
    - Dominated by 2^{16} trials to extract last 2 bytes of each block.

Summary of ESP Attacks

Royal Holloway
University of London

- We now have a range of attacks against encryption-only ESP.
  - Sometimes attacks that work in practice wouldn't work if the RFCs had been followed.
  - The attacks that work on paper against the RFCs often don't work in practice against implementations.
- Attacks in theory versus attacks in practice.
  - Implementations often deviate from standards (RFCs) in small ways that turn out to be important for security.
  - These deviations can result in big differences between attacks on paper and attacks that work in practice.
- What counts as success in such an endeavour?
  - Depends on your choice of programme committee!

Final Remarks on IPsec

Royal Holloway
University of London

- The attacks show that encryption-only ESP is dangerously weak in a practical sense.
  - This does not contradict theory: CBC mode is trivially insecure against a CCA attacker.
  - But a fully practical attack seems necessary to convince practitioners of the need for CCA security.
  - Even then, standards may not be changed, for many other reasons.
    - Timing relative to standards process, cost of change, backwards compatibility,…

Final Remarks on IPsec

Royal Holloway
University of London

- Recall ESPv3 (RFC 4303):

  "ESP allows encryption-only … because this may offer considerably better performance and still provide adequate security, e.g., when higher layer authentication/integrity protection is offered independently."

- In fact, little or no security is gained from the provision of upper layer integrity protection.
  - An **appropriate** combination of encryption and integrity protection is needed.
  - As provided, for example, by using ESP with encryption **and** integrity protection (Encrypt-then-MAC construction).

Final Remarks on IPsec

Royal Holloway
University of London

- The attacks arise from the interactions between cryptographic components and the “ambient” system in which they operate.
  - IPsec protects IP; IP header has particular syntax.
  - ICMP as an error-reporting side-channel for IP/IPsec
- The attacks illustrate the dangers of viewing cryptography in isolation.

Royal Holloway
University of London

Introduction

Theory for Secure Channels

Evolution of Encryption in IPsec

SSL/TLS Record Layer Protocol

SSL/TLS Overview

Royal Holloway
University of London

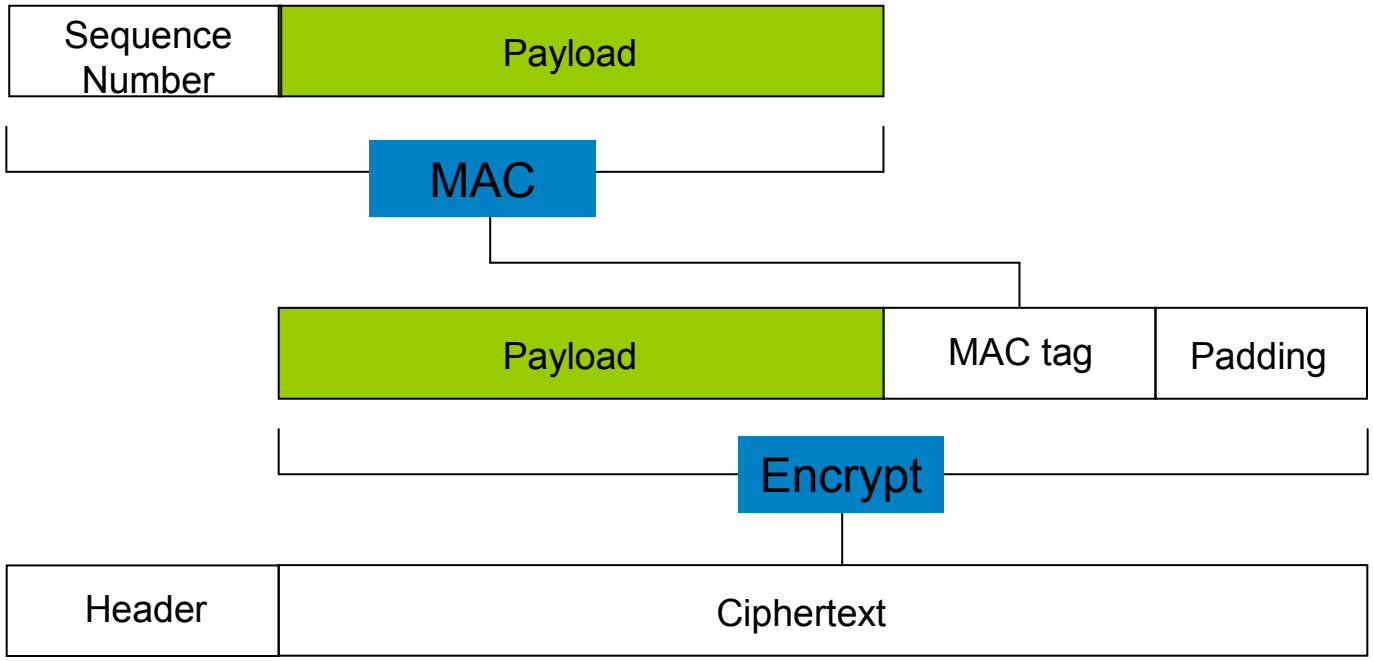
- SSL = Secure Sockets Layer.
  - unreleased v1, flawed but useful v2, good v3.
- TLS = Transport Layer Security.
  - IETF-standardised version of SSL.
  - TLS1.0=SSL3.0 with minor tweaks, RFC 2246.
  - TLS1.1=TLS1.0 with tweaks, RFC 4346 (2006).
  - TLS1.2=TLS1.1 with yet more tweaks, RFC 5246 (2008).
- SSL/TLS provides security ‘at TCP layer’.
  - Runs over TCP, using TCP to provide reliable, end-to-end transport.
  - Widely deployed in Web browsers and servers to support ‘secure e-commerce’ over HTTP.

SSL/TLS Record Protocol

Royal Holloway
University of London

- SSL/TLS Record Protocol provides:
  - Data origin authentication, integrity, anti-replay using a MAC and sequence number.
    - Algorithms supported in TLS1.2 are NULL, HMAC-MD5, HMAC-SHA1, HMAC-SHA256.
  - Confidentiality using symmetric algorithm.
    - Algorithms supported in TLS1.2 are NULL; 3DES, AES-128, AES-256 block ciphers, all in CBC mode; RC4-128 stream cipher.
- Earlier versions of SSL/TLS will support different sets of algorithms.
- Keys for the algorithms are supplied by the SSL/TLS Handshake Protocol.

SSL/TLS Record Protocol (Simplified)



Security of SSL/TLS Record Protocol

Royal Holloway
University of London

- A MAC-then-encrypt construction.
- **Not** generically secure.
- But building on results of [K01], the basic MAC-then-encrypt construction is IND-CCA secure for CBC mode encryption.
- So we are OK, right?
- Well, SSL/TLS is actually a MAC-*then-pad*-then-encrypt construction…

SSL/TLS Record Layer Padding

- Padding in SSL/TLS has a particular format:
  - Add a sequence of bytes after the MAC to complete the last plaintext block.
  - If *t* bytes are needed, then add *t* copies of the byte representation of *t*.
  - e.g. 01, 02 02, 03 03 03,…
  - Sender is allowed to include up to 255 bytes of padding.
- So what should happen if the padding format after decryption of a received ciphertext is incorrect?

SSL/TLS Record Layer Padding

- The specification for TLS1.0 in RFC 2246 describes possible error messages, including:

  decryption_failed: A TLS Ciphertext decrypted in an invalid way: either it wasn't an even multiple of the block length or its padding values, when checked, weren't correct. This message is always fatal.

- This suggests that implementations should check the format of padding and terminate the connection if the padding format is incorrect.
- What are the security consequences of this?

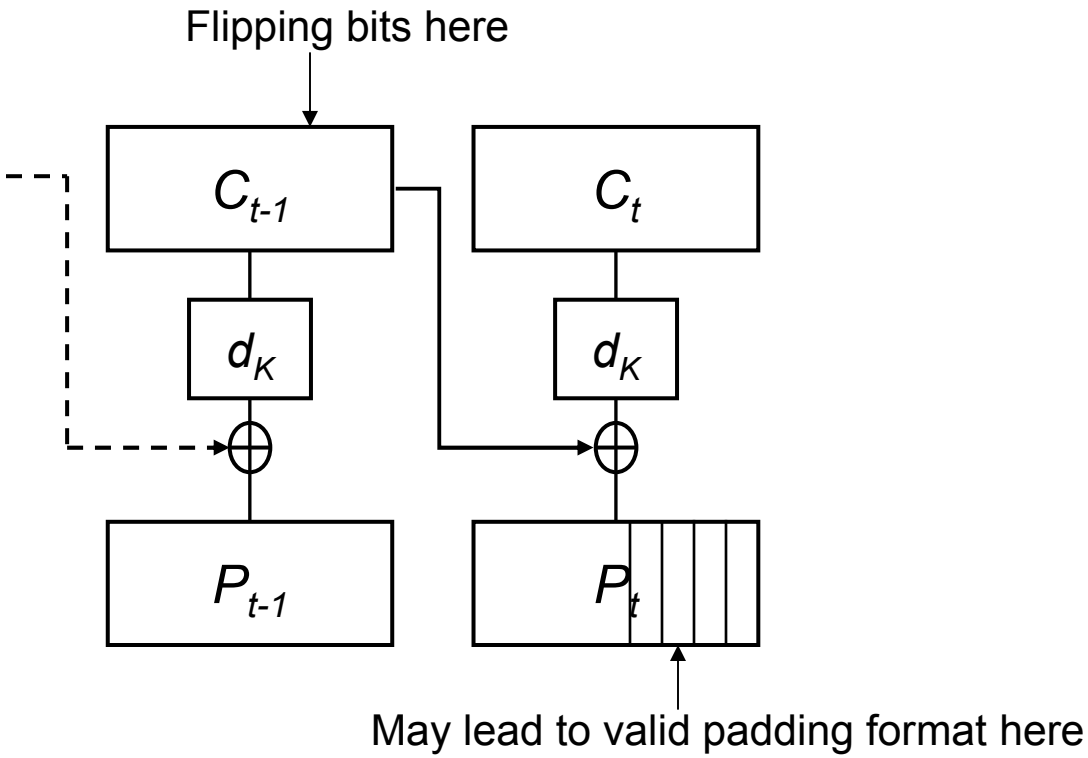
Padding Oracle Attacks

Royal Holloway
University of London

- Vaudenay [V02] proposed the concept of a *padding oracle attack*.
- A padding oracle *P* takes as input a ciphertext and outputs a bit indicating whether the underlying plaintext is correctly padded or not.
  - According to some fixed padding rule and some fixed key *K*.
- [V02] showed that, for CBC mode and for certain padding schemes, a padding oracle can be used to build a decryption oracle.
  - Let's look at SSL/TLS-style padding….

SSL/TLS Padding Oracle Attack

- Place target block C_t as last block of ciphertext.
- Flipping bits in last-but-one ciphertext block may lead to correct padding in last plaintext block.
- Padding oracle will tell attacker when this occurs.

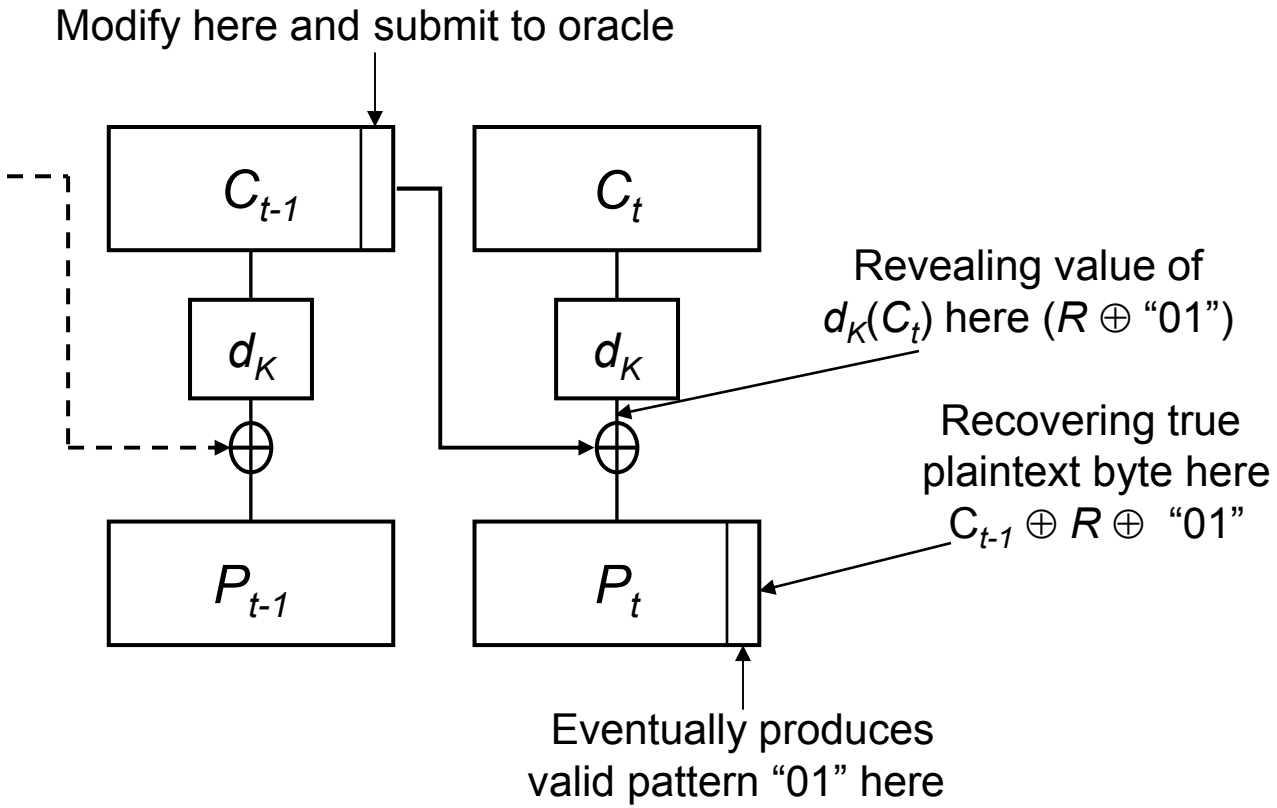


SSL/TLS Padding Oracle Attack

Royal Holloway
University of London

- For SSL/TLS, most likely "correct" output arises from valid padding pattern of length 1, namely "01".
- So:
  - Select a random block R.
  - Repeatedly modify last byte of R and submit ciphertext ending with blocks R, C_t to padding oracle….

SSL/TLS Padding Oracle Attack



SSL/TLS Padding Oracle Attack

Royal Holloway
University of London

- An average of 128 trials are needed to extract the last byte of each plaintext block.
- Can extend to the entire block.
  - Now that last byte of P_t is known, we can modify *R* to set last plaintext byte to “02”.
  - Then modify second-to-last byte of *R* until padding oracle returns “correct”.
  - Most likely padding pattern is now “02 02”.
  - Can now recover second last byte of P_t.
  - Repeat…
- Can extend to multiple blocks.
  - Can place any target block as last block of ciphertext.

Padding Oracles In Practice

Royal Holloway
University of London

- Canvel *et al.* [CHVV03]:
  - Recall that either a padding failure or MAC failure may result in an error message on the SSL/TLS channel.
  - Process of decryption will be:
    1. CBC-decrypt;
    2. check padding;
    3. check MAC.
  - Hence a MAC failure error message is likely to appear on the network **later** than a padding failure error message.
  - So timing of error message may give us a padding oracle.
  - But either type of message is a “fatal error” causing the SSL/TLS connection to be destroyed.
  - We have a *bomb oracle* instead of a padding oracle.
  - So the attacker can only learn one byte of plaintext, and with probability 1/256.

OpenSSL and Padding Oracles

- Canvel *et al.* also showed:
  - The attacker can still decrypt if a *fixed* plaintext is repeated in a *fixed* location in many SSL/TLS connections.
  - As is the case when SSL/TLS is used to protect an Outlook password, for example.
  - Timing differences between padding failures and MAC failures detectable on a LAN for the then-current OpenSSL implementation of SSL/TLS.

OpenSSL and Padding Oracles

- OpenSSL was subsequently patched to ensure uniform reporting and timing of error messages, thus preventing the attack.
- Advice to implementers in RFC 4346 (TLS1.1):

> *The receiver MUST check this padding and SHOULD use the bad_record_mac alert to indicate padding errors.*
>
> *In order to defend against this attack, implementations MUST ensure that record processing time is essentially the same whether or not the padding is correct. In general, the best way to do this is to compute the MAC even if the padding is incorrect, and only then reject the packet.*

Theoretical Impact

Royal Holloway
University of London

- The result of [K01], while valuable, is not the complete story about SSL/TLS Record Protocol security.
  - Even if the title of [K01] is *"The order of encryption and authentication for protecting communications (Or: how secure is SSL?)"* ☺
- The attack of [CHVV03] shows the importance of looking beyond the basic encryption and MAC algorithm components of secure channels
  - Order/timing/error events for cryptographic and other operations matters.
  - Padding matters.
  - (Statefulness/sequence numbers matter.)

Theoretical Impact

Royal Holloway
University of London

- Padding oracles may seem esoteric, but can be difficult to avoid in practice.
  - e.g. in IPsec, an RFC-compliant implementation will check an extended padding format and silently drop the packet if check fails, and forward the packet if it passes.
    - Basis for the attacks against IPsec RFCs in [DP07].
  - e.g. in SSL/TLS, an RFC-compliant implementation will issue an error message if the padding check fails; it is up to the implementation to ensure this does not introduce a timing side-channel.

Theoretical Impact

Royal Holloway
University of London

- [PW08] gives a formal security treatment of CBC mode in the presence of padding oracles.
  - Shows OWE-PO security for OZ-PAD scheme recommended in ISO standards.
  - Proves IND-PO-CPA security for abit/abyte padding schemes.
  - But provable security still unresolved for important MAC-then-PAD-then-encrypt construction used by SSL/TLS in presence of padding oracles.

Royal Holloway
University of London

Introduction

Theory for Secure Channels

Evolution of Encryption in IPsec

SSL/TLS Record Layer Protocol

SSH Binary Packet Protocol

Introducing SSH

Royal Holloway
University of London

Secure Shell or SSH is a network protocol that allows data to be exchanged using a secure channel between two networked devices. Used primarily on Linux and Unix based systems to access shell accounts, SSH was designed as a replacement for TELNET and other insecure remote shells, which send information, notably passwords, in plaintext, leaving them open for interception. ***The encryption used by SSH provides confidentiality and integrity of data over an insecure network, such as the Internet****.*

– Wikipedia

Introducing SSH

Royal Holloway
University of London

- SSHv1 had several security flaws.
  - Worst ones arising from use of CRC algorithm to provide integrity.
  - Enabling, for example, traffic injection attacks.
- SSHv2 was standardised in 2006 by the IETF in RFCs 4251-4254.
- RFC 4253 specifies the SSH Binary Packet Protocol (BPP).
- SSHv2 is widely regarded as providing strong security.
  - Widely used to enable secure remote administration of sensitive systems.
  - One minor flaw in the BPP that *in theory* allows distinguishing attacks ([D02]; [BKN02]).
  - Simple countermeasure adopted in OpenSSH.

SSHv2 Architecture

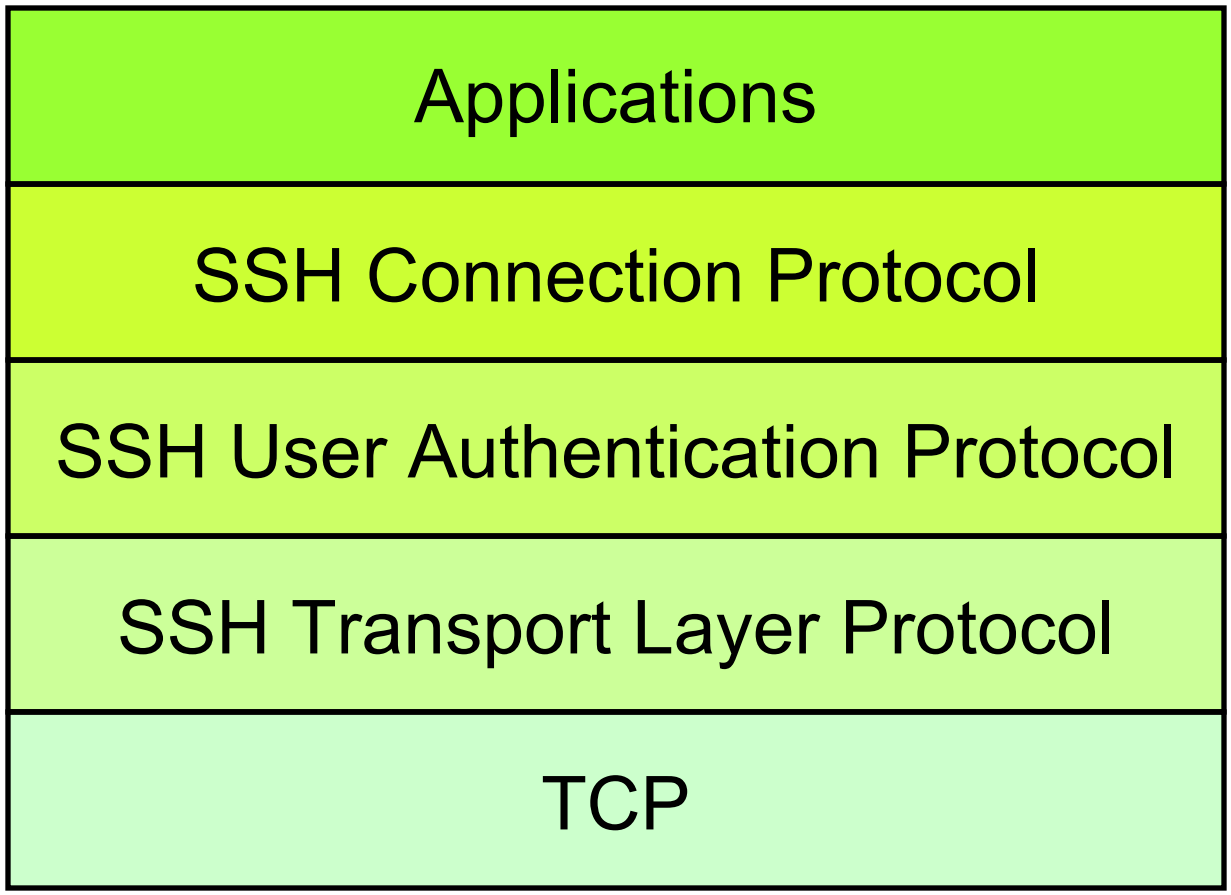
Royal Holloway
University of London

SSHv2 adopts a three layer architecture:

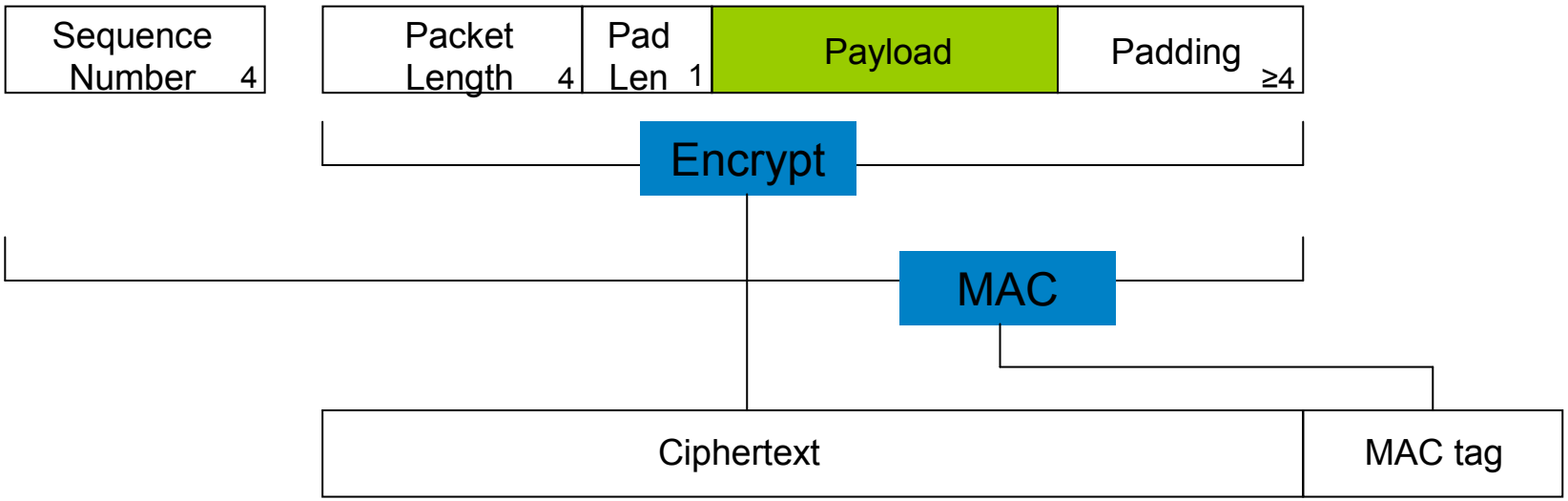
- SSH Transport Layer Protocol.
  - Initial connection establishment and key exchange.
  - Server authentication (almost always).
  - Sets up a secure channel between client and server, using the **SSH Binary Packet Protocol** specified in RFC 4253.
- SSH User Authentication Protocol.
  - Client authentication over secure Transport Layer channel.
- SSH Connection Protocol.
  - Supports multiple concurrent connections over a single Transport Layer secure channel.
  - Efficiency (session re-use) and support for multiple applications.

SSHv2 Architecture

Royal Holloway
University of London

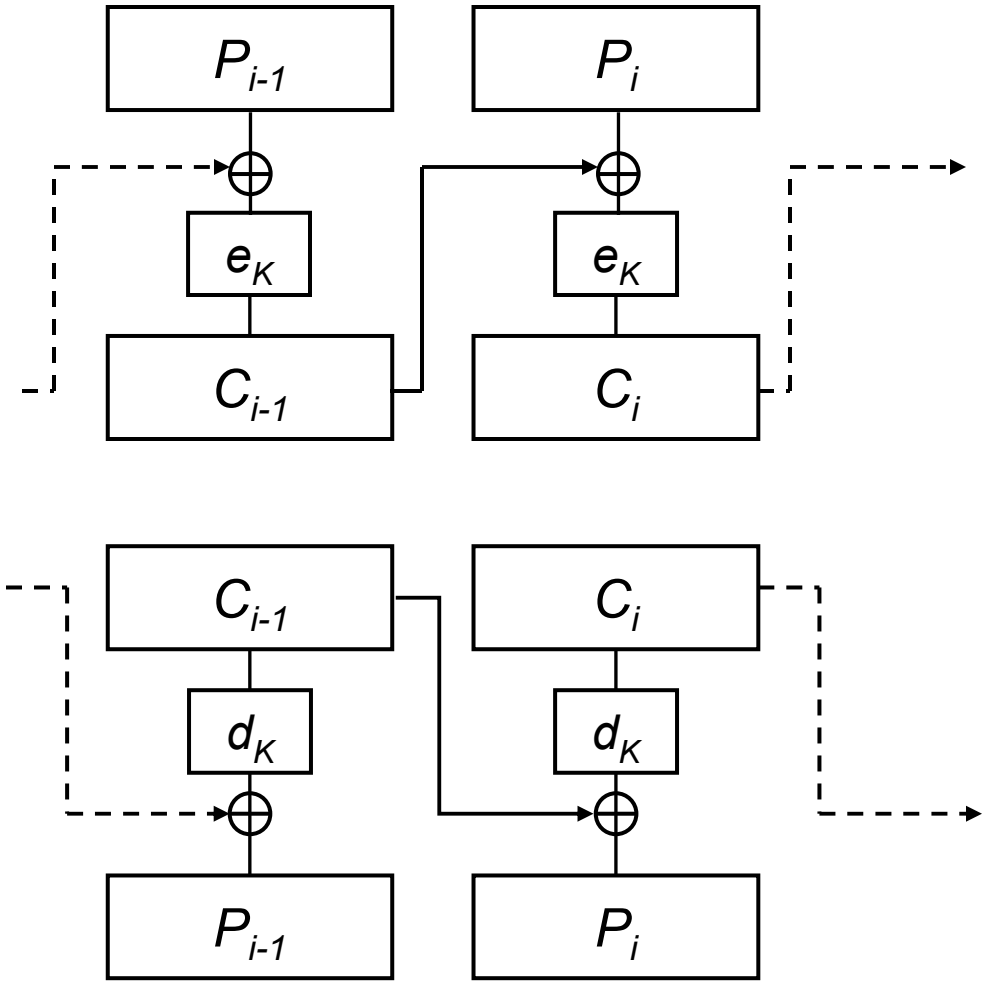


The SSH BPP



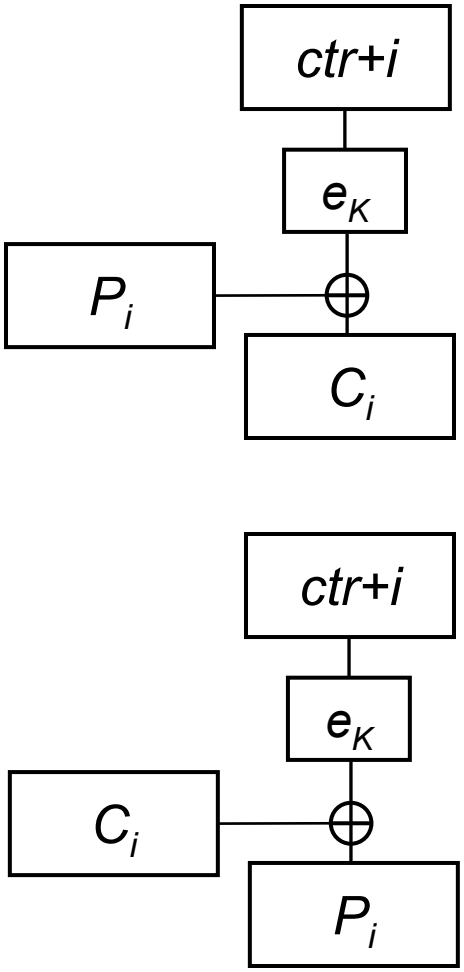
- Encode-then-Encrypt&MAC construction, **not** generically secure.
- Packet length field measures the size of the packet on the wire in bytes and is encrypted to hide the true length of SSH packets.
- Variable length padding is permissible; padding needed for CBC mode and carried over to CTR mode.

CBC Mode in SSH



- RFC 4253 mandates 3DES-CBC and recommends AES-CBC.
  - In fact, all originally specified optional configurations involve CBC mode, and ARCFOUR was the only optional stream cipher.
- SSH uses a chained IV in CBC mode:
  - IV for current packet is the **last** ciphertext block from the **previous** packet.
  - Effectively creates a single stream of data from multiple SSH packets.

CTR Mode in SSH



- CTR mode uses block cipher to build a stream cipher.
- CTR mode for SSH standardised in RFC 4344.
  - Initial value of counter is obtained from handshake protocol.
  - Packet format is preserved from CBC case.
  - Recommends use of AES-CTR with 128, 192 and 256-bit keys, and 3DES-CTR.

Security of the SSH BPP

Royal Holloway
University of London

- Attack of [D02], [BKN02] exploits chained IVs in CBC mode.
  - Breaks semantic security of the SSH BPP in a chosen ciphertext attack model.
    - Attacker can distinguish which one of two chosen messages was encrypted.
  - Low success probability against SSH implementations because of specifics of packet format.
  - Prevented in OpenSSH by optional use of dummy packets to hide IVs until it is too late for attacker to make use of them.
- **Basic message**: SSH BPP using CBC mode with chained IVs is **in**secure according to the standard theoretical notion of security.

Security of the SSH BPP

Royal Holloway
University of London

- Using the IND-SFCCA model, [BKN02] proved the security of variants of the SSH BPP under reasonable assumptions concerning:
  - The encryption component.
    - Essentially, IND-CPA security.
  - The MAC component.
    - Strong unforgeability and pseudo-randomness.
  - The randomness of the padding scheme.
  - Collision properties of the encoding scheme.
    - In practice, for SSH BPP, this means not too many packets can be encrypted.

Security of the SSH BPP

Royal Holloway
University of London

- In particular, [BKN02] established the IND-SFCCA security of SSH-$NPC and SSH-CTR.
  - SSH-$NPC = SSH using a block cipher in CBC mode with explicit, per-packet, random IV and with random padding.
    - In contrast to chained IVs used in SSH BPP.
  - SSH-CTR = SSH using a block cipher in counter mode, with counter maintained at sender and receiver.

Attacking the SSH BPP

Royal Holloway
University of London

- [APW09]: plaintext recovering attacks against SSH BPP.
  - Much stronger than distinguishing attack of [D02]!
- These attacks exploit the interaction of the following features of the BPP specification:
  - The attacker can send data on an SSH connection in small chunks (TCP).
  - CBC mode is mandated.
  - A MAC failure is visible on the network.
  - The packet length field encodes how much data needs to be received before the MAC is received and the integrity of the packet can be checked.

Attacking the SSH BPP (Theory)

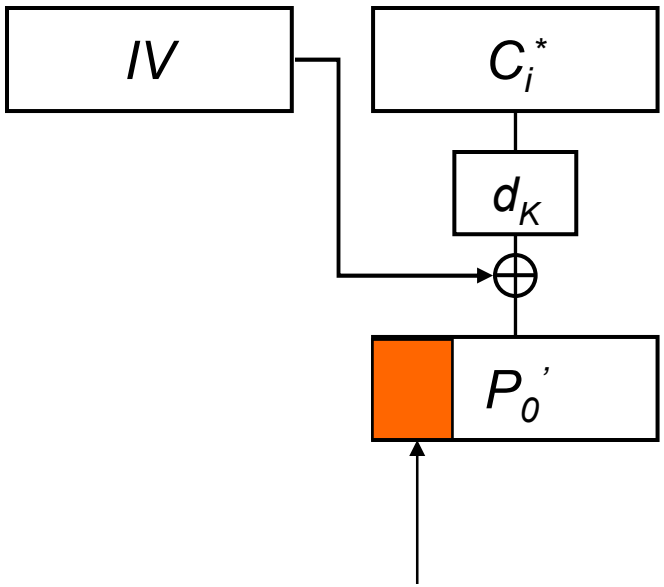
Royal Holloway
University of London

- The attacker monitors an SSH connection and selects any target ciphertext block C_i^*. Here:

$$C_i^* = e_K(C_{i-1}^* \oplus P_i^*), \text{ i.e. } P_i^* = C_{i-1}^* \oplus d_K(C_i^*)$$

- The attacker injects C_i^* so it as seen as the *first* block of a new SSH packet by the receiver…

Attacking the SSH BPP (Theory)

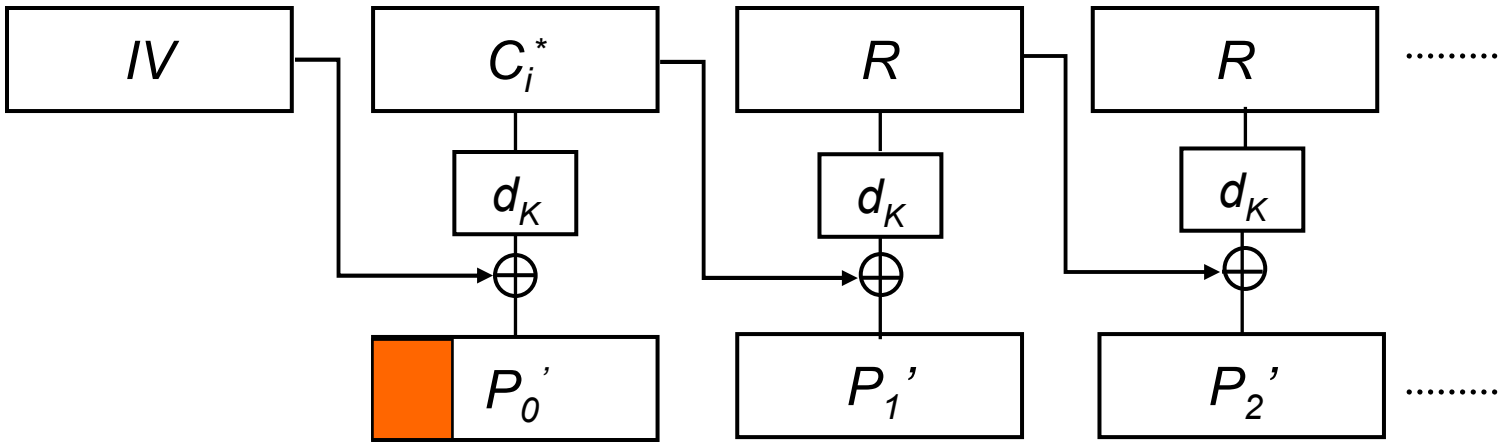


The receiver will treat the first 32 bits of the calculated plaintext block as the packet length field for the new packet. Here:

$$P_0' = IV \oplus d_K(C_i^*)$$

where *IV* is known from the previous packet.

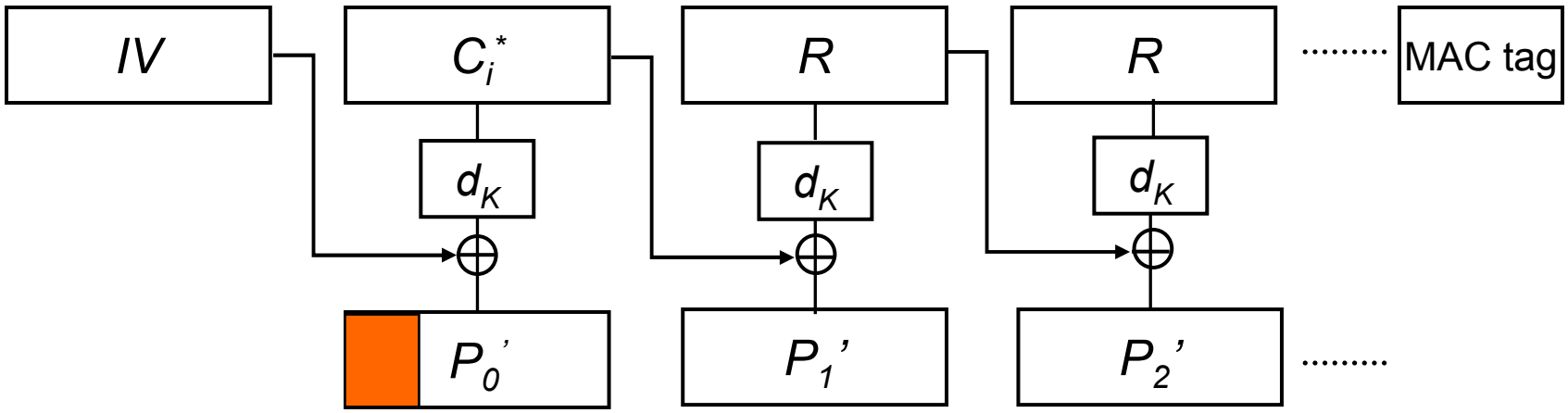
Attacking the SSH BPP (Theory)



The attacker then feeds random blocks to the receiver.

– One block at a time, waiting to see what happens at the server when each new block is processed.

Attacking the SSH BPP (Theory)



- Eventually, once enough data has arrived, the receiver will receive what it thinks is the MAC tag.
- The receiver will then check the MAC.
  - This check will fail with overwhelming probability.
  - Consequently the connection is terminated (with an error message).
- How much data is “enough” so that the receiver decides to check the MAC?

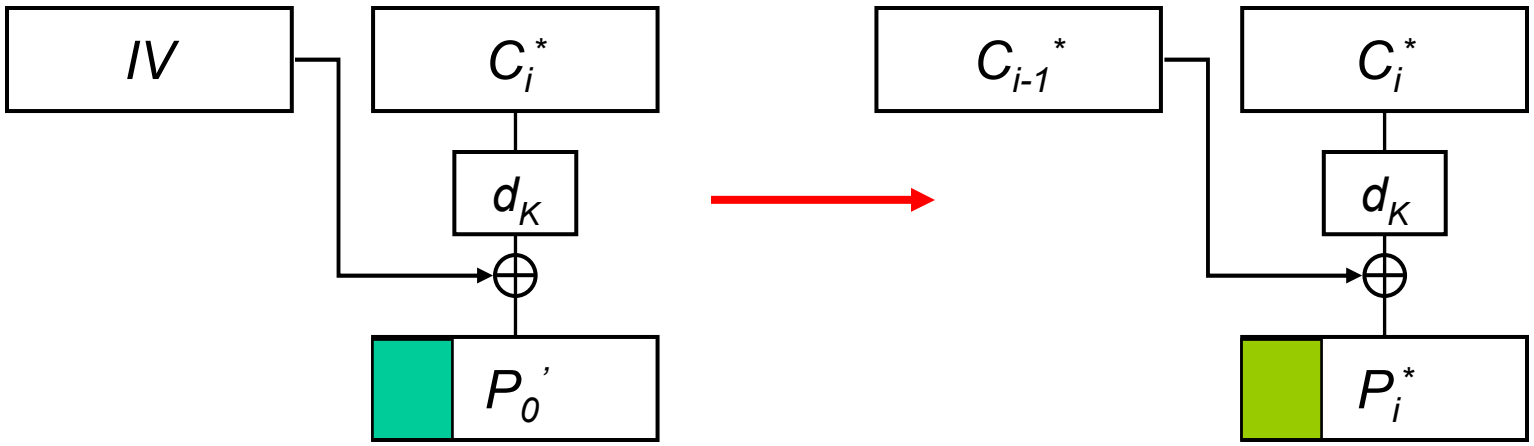
Attacking the SSH BPP (Theory)

Royal Holloway
University of London

- The receiver **has** to use the packet length field to decide when the MAC tag has arrived.
- Hence an attacker who counts the number of blocks needed to cause connection termination learns the packet length field.
- That is, the attacker learns the first 32 bits of:

$$P_0' = IV \oplus d_K(C_i^*).$$

Attacking the SSH BPP (Theory)



- Knowing IV and 32 bits of P_0', the attacker can now recover 32 bits of the **target** plaintext block:

$$P_i^* = C_{i-1}^* \oplus d_K(C_i^*) = C_{i-1}^* \oplus IV \oplus P_0'$$

Attack Performance (Theory)

Royal Holloway
University of London

- As described, this simple attack succeeds in recovering 32 bits of plaintext from an arbitrary ciphertext block with probability 1.
  - But requires the injection of about 2^{31} random bytes to trigger the MAC check.
  - And leads to an SSH connection tear-down.
- The attack breaks the SSH BPP.
- The attack still works if a fresh IV is used for each new SSH packet.
  - Breaking SSH-$NPC that was proven secure in [BKN02].

Attacking OpenSSH

Royal Holloway
University of London

- OpenSSH is the most popular implementation of the SSH RFCs.
  - Open-source, distributed as part of OpenBSD.
  - OpenSSH webpages state that OpenSSH accounts for more than 80% of all deployed SSH servers.
  - www.openssh.org/usage/index.html
- We worked with OpenSSH 5.1.
  - Version 5.2 released 23/02/2009 partly as a consequence of our work, current version is 5.3.

Attacking OpenSSH

Royal Holloway
University of London

- In OpenSSH 5.1, two sanity checks are carried out on the packet length field after the first block is decrypted.
- When each of the checks fails, the SSH connection is terminated in subtly different ways.
  - This difference leaks some information, but also reduces success prob. of the attack.
- If the length checks pass, then OpenSSH 5.1 waits for more bytes.
- Finally, when the MAC check fails, a third type of connection termination is seen.

Attacking OpenSSH

Royal Holloway
University of London

- The manner in which OpenSSH 5.1 behaves on failure allows:
  - A first attack verifiably recovering 14 bits of plaintext with probability 2^{-14}.
  - A second attack verifiably recovering 32 bits of plaintext with probability 2^{-18} (for a 128-bit block cipher).
  - The attacks require injection of (roughly) 2^{18} bytes.
- Both attacks result in termination of the SSH connection.
  - But the attacks can be iterated if a plaintext is repeated across multiple connections.
- The attacks worked in practice.

Iterating the attacks

Royal Holloway
University of London

- If a *fixed* plaintext is repeated at a *fixed* position in SSH packets over *multiple* connections, then the attacks can be iterated to boost success rate.
  - Application to password extraction.
  - Some clients automatically reconnect on session termination.
  - By carefully selecting after which IV to inject the target ciphertext block, we can reduce the number of connections consumed during the attack to $2^{14} + 2^{4}$.

Disclosure of the Attacks

- We worked with the UK Centre for Protection of National Infrastructure (CPNI) to disclose the attacks.
  - www.cpni.gov.uk/Docs/Vulnerability_Advisory_SSH.txt
  - Advisory published 14/11/2008.
  - Vendors notified well ahead of time, giving opportunity to prepare fixes.
  - Recommends switching to counter mode encryption.

Reactions and Countermeasures

- OpenSSH published a statement and committed a first fix (21/11/2008).
  - www.openssh.com/txt/cbc.adv
  - Both the statement and the bugfix addressed only the 2^{-14} attack.
- Then OpenSSH released OpenSSH 5.2 (23/02/2009).
  - Offers AES in counter mode and arcfour256 stream cipher ahead of CBC mode block ciphers.

Reactions and Countermeasures

Royal Holloway
University of London

- www.openssh.org/txt/release-5.2:
  - *"This release also adds countermeasures to mitigate CPNI-957037-style attacks against the SSH protocol's use of CBC-mode ciphers."*
  - If length checks fail, then set length field to 2^{18} and carry on.
  - This renders OpenSSH *more* vulnerable to DoS attacks!
  - And there are still plaintext recovery and distinguishing attacks.
    - Attacker who knows a certain18 bits of a block can recover a further 14 bits.

Some Countermeasures

Royal Holloway
University of London

- Use counter mode.
  - *Stateful* version of counter mode needed, as standardised in RFC 4344.
  - Our attacks no longer apply.
- *Enforce* use of counter mode.
  - Not standards compliant with the RFCs as they are currently written.
  - Some implementations do not support counter mode at all, creating backwards compatibility issue.
  - "*Only a cryptographer would suggest this...*"

Further Countermeasures

Royal Holloway
University of London

- Don't encrypt the length field.
  - Invasive and makes certain DoS attacks easier.
- Separately MAC the length field.
  - Invasive.
- Use authenticated encryption algorithm in place of SSH's *ad hoc* construction.
  - Invasive, and still can't safely encrypt the length field.

Impact of the Attacks

Royal Holloway
University of London

- SSH was meant to be bullet-proof, but our attacks are really quite simple.
- The specific attacks are easily circumvented by switching to CTR mode or by modifying error handling in CBC mode.
  - Unfortunately, this does not constitute a proof of security against attacks of the type presented here.
- And the basic attack applied to the proven secure variant SSH-$NPC
  - Hinting at inadequacies of the approach used in [BKN02].

Limitations of [BKN02]

Royal Holloway
University of London

- The security model of [BKN02] *does* model errors arising during the BPP decryption process.
  - Connection teardown is modeled by disallowing access to decryption and encryption oracles after any error event.
  - Errors can arise from decryption, decoding or MAC checking.
- But only a single type of error message is output.
  - The 2^{-14} attack against OpenSSH exploits the fact that different error events *are* distinguishable.
- And the model assumes that decoding errors arise before MAC errors.
  - While the OpenSSH implementation only does decoding *after* the MAC has been checked.

Limitations of [BKN02]

Royal Holloway
University of London

- The model assumes that plaintexts and ciphertexts are "atomic".
  - All oracle queries in the model involve complete plaintexts or ciphertexts.
  - But the attacks exploit the ability to deliver ciphertexts one block (or even one byte!) at a time and observe behaviour.
    - For example, distinguishing the wait state from a MAC failure.
- The model does not allow for *plaintext-dependent* decryption.
  - The packet length field never appears in the model.
  - But implementations *must* make use of this field during the decryption process.
  - And, as we've seen, the manner in which this field is treated is critical for security.

A New Security Analysis of SSH

Royal Holloway
University of London

- In [PW10], we:
  - Develop a new security model addressing limitations of the model used in [BKN02].
    - LOR-BSF-CCA security.
    - Allows byte-by-byte delivery of ciphertexts to decryption oracle, and buffering of any as-yet-unprocessed ciphertext bytes.
  - Build an accurate description of SSH-CTR as specified in RFCs and implemented in OpenSSH;
  - Prove the security of this description of SSH-CTR in our new model.

A New Security Analysis of SSH

Royal Holloway
University of London

- Our description of SSH-CTR involves:
  - Accurate modelling of errors, based on specification in RFCs and ‘C’ source code for OpenSSH.
    - Errors from length sanity checking.
    - Errors from MAC verification failure.
    - Errors from parsing failures during decoding.
    - Session teardown in event of any error.
  - Use of the packet length field from plaintext to determine the amount of ciphertext required before the MAC check is performed.
    - Plaintext-dependent decryption.

Modelling the Encryption Algorithm

Algorithm E-SSH-CTR$_{Ke,Kt}$ (*m*)
if st_e = *fail* then
  return *fail*
(m_e,m_t) = encode(*m*)
if m_e = *fail* then
  st_e = *fail*
  return *fail*
else
  c = *E*-CTR$_{Ke}$(m_e) \\ counter mode encryption
  tau = T_{Kt}(m_t) \\ MAC computation
  return *c* || *tau*
end if

Modelling the Decryption Algorithm

Algorithm D-SSH-CTR$_{Ke,Kt}$ (*c*)
if st_d = *fail* then
  return *fail*
end if
{Stage 1}
cbuff = *cbuff* || *c*
{Stage 2}
if m_e = *empty* and |*cbuff*| >= *L* then
  Parse *cbuff* as *c'*||*A* (where |*c'*| = *L*)
  m_e[1] = *D*-CTR$_{Ke}$ (*c'*)
  LF = len(m_e[1]) \\ len checking
  if *LF* = $fail_L$ then
    st_d = *fail*
    return $fail_L$
  else
    need = 4 + *LF* + *maclen*
  end if
end if

{Stage 3}
if |*cbuff*| >= *L* then
  if |*cbuff*| >= *need* then
    Parse *cbuff* as *c*[1…*n*] || *tau* || *B*,
    where |*c*[1…*n*] || *tau*| = *need*,
    and |*tau*| = *maclen*
    m_e[2…*n*] = *D*-CTR$_{Ke}$ (*c*[2…*n*]) \\ CTR mode
    m_e = m_e[1] || m_e[2…*n*]
    m_t = *SN* || m_e
    v = V_{Kt} (m_t, *tau*) \\ MAC checking
    if *v* = 0 then \\ MAC failure
      st_d = *fail*
      return $fail_A$
    else
      m = decode(m_e) \\ decoding plaintext
      m_e = *empty*; *cbuff* = *B*
      return *m*
    end if
  end if
end if

Main Security Result

Royal Holloway
University of London

Theorem: SSH-CTR is IND-BSF-CCA secure under the assumptions that:

- F, the function family used to construct CTR mode is pseudo-random;
- The MAC scheme is strongly unforgeable;
- The MAC tagging algorithm is pseudo-random;
- Minimal requirements on the length checking function len are met.

- The theorem can be made *concrete.*
  - The advantage of any IND-BSF-CCA adversary is meaningfully related to advantages of adversaries against F and the MAC.
  - Good for practice!

What Does the Proof Mean?

Royal Holloway
University of London

- The model is rich enough to encompass usual LOR-CCA attacker, as well as attacks of [APW09].
- The model includes all “failure modes” of SSH-CTR (as implemented in OpenSSH BPP).
  - So cryptanalysis based on error side-channels is covered.
- But:
  - Timing side-channels are not covered.
  - The model does not include anything “outside” the BPP.
  - The proof is specific to the OpenSSH implementation of SSH-CTR.
  - Completeness of model is guaranteed only by manual code inspection.

Final Remarks on SSH

Royal Holloway
University of London

- Our attacks on SSH illustrate further limitations of current approaches to provable security.
  - How do we know we have the right model?
  - How do we know what features should be included in a protocol description?
    - What is the right amount of abstraction?
    - How close to the implementation level do we need to go to capture all significant attack vectors?
    - Does [PW10] really get it right?

Royal Holloway
University of London

Introduction

Theory for Secure Channels

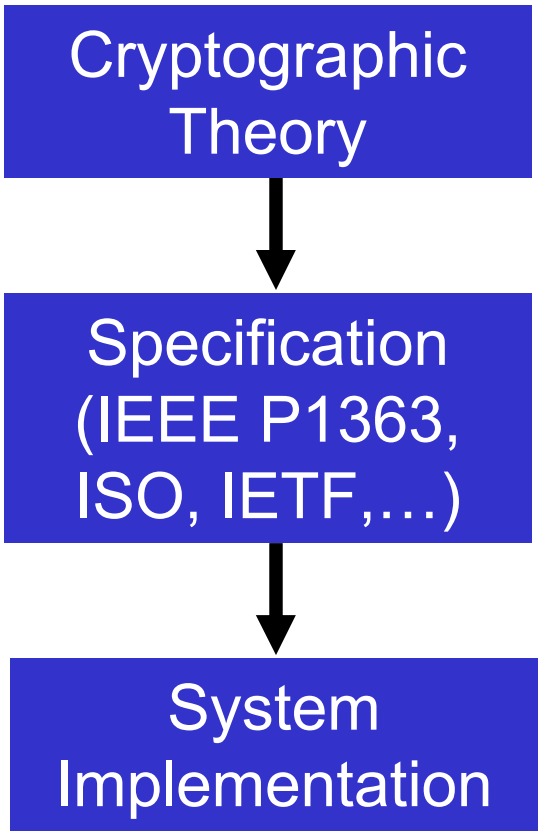
Evolution of Encryption in IPsec

SSL/TLS Record Layer Protocol

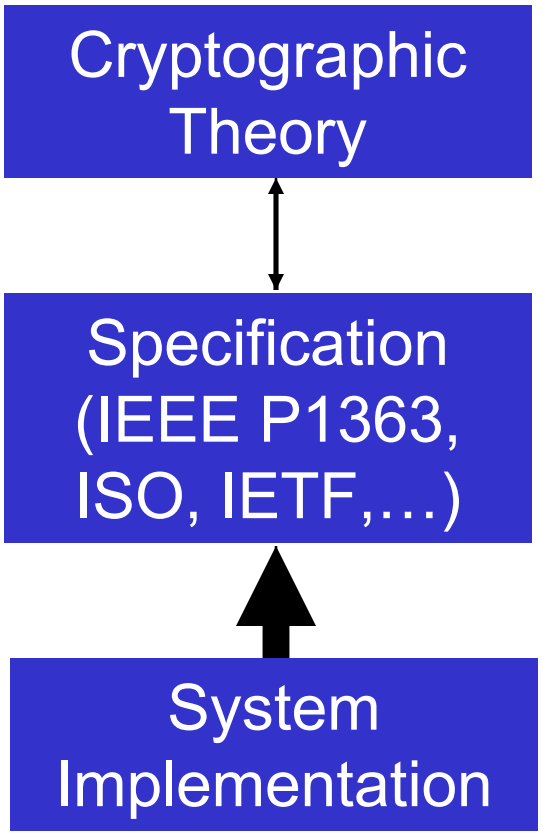
SSH Binary Packet Protocol

Concluding Remarks

The Big Picture – Idealised View



The Big Picture – Possible Realities



Concluding Remarks

Royal Holloway
University of London

- Theory is great – in theory!
  - In fact, it rules out many classes of attack in each of our case studies.
- But there is a complex interplay between theory, specification and implementation.
  - Many practitioners are not yet convinced by what provable security has to offer.
  - Fragility of proofs when building implementations.
  - What constitutes an attack?
  - Provable security is itself an evolutionary process.
    - But better than break-then-fix loop.
  - Cultural effects.

Concluding Remarks

Royal Holloway
University of London

- Cryptography is usually only a component in a larger system or protocol.
  - Integrating it can introduce security weaknesses.
- Implementing cryptography in real systems is fraught with dangers.
  - We have focussed almost exclusively on error-based side channels and format-based attacks here.
  - Many other types of implementation-based attack are known.

Last Words

Royal Holloway
University of London

A (mis)quote from Eugene Spafford:

"*Using encryption on the Internet is the equivalent of arranging an armored car to deliver credit-card information from someone living in a cardboard box to someone living on a park bench.*"

Thank You

Some Further Reading

[APW09]: Albrecht *et al.,* IEEE S&P 2009.

[BDJR97]: Bellare *et al.,* FOCS 1997.

[BKN02]: Bellare *et al.,* ACM-CCS, 2002.

[BN00]: Bellare and Namprempre, Asiacrypt 2000.

[D02]: Dai, unpublished, 2002.

[DP07]: Degabriele and Paterson, IEEE S&P 2007.

[CHVV03]: Canvel *et al.*, Crypto 2003.

[K01]: Krawczyk, Crypto 2001.

[PY06]: Paterson and Yau, Eurocrypt 2006.

[PW08]: Paterson and Watson, SCN 2008.

[PW10]: Paterson and Watson, Eurocrypt 2010.

[V02]: Vaudenay, Eurocrypt 2002.