

Closing the gap between cryptography theory and security practice

Chris Mitchell
Information Security Group
Royal Holloway, University of London

Agenda

1. Introduction
2. Developments in cryptography
3. Encryption and integrity – a case study
4. Specification issues in secure protocols
5. Concluding remarks

Main goal of talk

- Highlight the importance for users of cryptography:
 - to *keep in touch* with theoretical developments in subject;
 - to ensure that security schemes are *implemented as intended* by their designers.
- Support this with two case studies.

Background

- In the last 10-15 years, cryptography has become a much more theoretical subject, with a serious divide opening up between theoreticians and practitioners:
 - some theoreticians don't seem to be interested in ensuring best practice is followed in the 'real world';
 - some practitioners regard newer developments in cryptography as being of theoretical interest only.

Consequences

- Some of the issues addressed by theoreticians have serious practical implications.
- That is, genuine attacks on deployed protocols may be possible if the cryptographic state of the art is not followed.

Agenda

1. Introduction
2. Developments in cryptography
3. Encryption and integrity – a case study
4. Specification issues in secure protocols
5. Concluding remarks

Scope

- Interested here in ‘regular’ cryptography, as has been widely used for many years.
- This includes both symmetric crypto:
 - secret key encryption;
 - MACs;and asymmetric crypto:
 - public key encryption;
 - signatures;(as well as hash-functions).

What's new?

- Despite the fact that these crypto-primitives have been widely used for many years, we are still learning important new things about them.
- By this, I don't mean new cryptanalysis (e.g. breaking MD4 and MD5, and denting SHA-1) although this is also very important.
- Instead I mean how to use these primitives safely (assuming they are sound).

Examples

- Examples of relevant improvements in understanding include:
 - use of symmetric encryption without MACing is potentially insecure;
 - it is a good idea to encrypt before (rather than after) MACing;
 - use of RSA for encryption without data encapsulation and randomisation is dangerous;
 - use of RSA for signature without appropriate formatting is dangerous.
- What is particularly disturbing is that these statements would all be regarded as ‘old hat’ and ‘obvious’ by the theoretical crypto community, but they are not always observed by the applications builders!

Encryption without integrity

- One major change in understanding has been to view any kind of encryption without simultaneous integrity protection as highly dangerous.
- Combining encryption with integrity is not only required in order to achieve proofs of security, but over the last few years a wide range of attacks have been devised against schemes using just encryption (or poorly designed combinations of encryption and integrity).

Error oracle attacks

- One general class of attacks on encryption without integrity are the *error oracle* attacks (see *Proc. ISC 2005*).
- If an attacker can intercept ciphertext, and repeatedly insert manipulated ciphertext into a channel, then the presence or absence of error messages (e.g. whether decryption is successful, why decryption failed, or whether the plaintext causes errors) can reveal information about the plaintext.
- In general, this problem cannot be removed, since errors may arise in higher level protocols, completely independent of the layer at which encryption is performed.

Order of cryptographic operations

- Another major change in view arising from developments in crypto theory affects the order in which encryption and integrity protection should be applied.
- Theory says that it is safer to encrypt first and then MAC protect (must verify MACs before attempting decryption, and **do not attempt to decrypt if MAC verification fails**).
- It is possible to achieve proofs of security for the ‘MAC then encrypt’ model, but such schemes are more complex and more at risk from side channel attacks (since decryption may fail prior to MAC verification, which could either trigger error messages or cause an early abort of processing).

Authenticated encryption modes

- To make life simple for users, a number of *authenticated encryption modes* have been devised.
- These allow a single secret key to be used to both integrity and confidentiality protect data in a ‘safe way’.
- A standard for such techniques (ISO/IEC 19772) is due to be published very shortly.

Agenda

1. Introduction
2. Developments in cryptography
3. Encryption and integrity – a case study
4. Specification issues in secure protocols
5. Concluding remarks

IPsec in encryption-only mode

- Paterson and Degabriele have recently described and demonstrated attacks on IPsec when used in encryption only mode.
- These attacks apply to widely used IPsec implementations which conform to the relevant IETF RFCs (see *Proc. IEEE Security and Privacy 2007*).

Why look at this example?

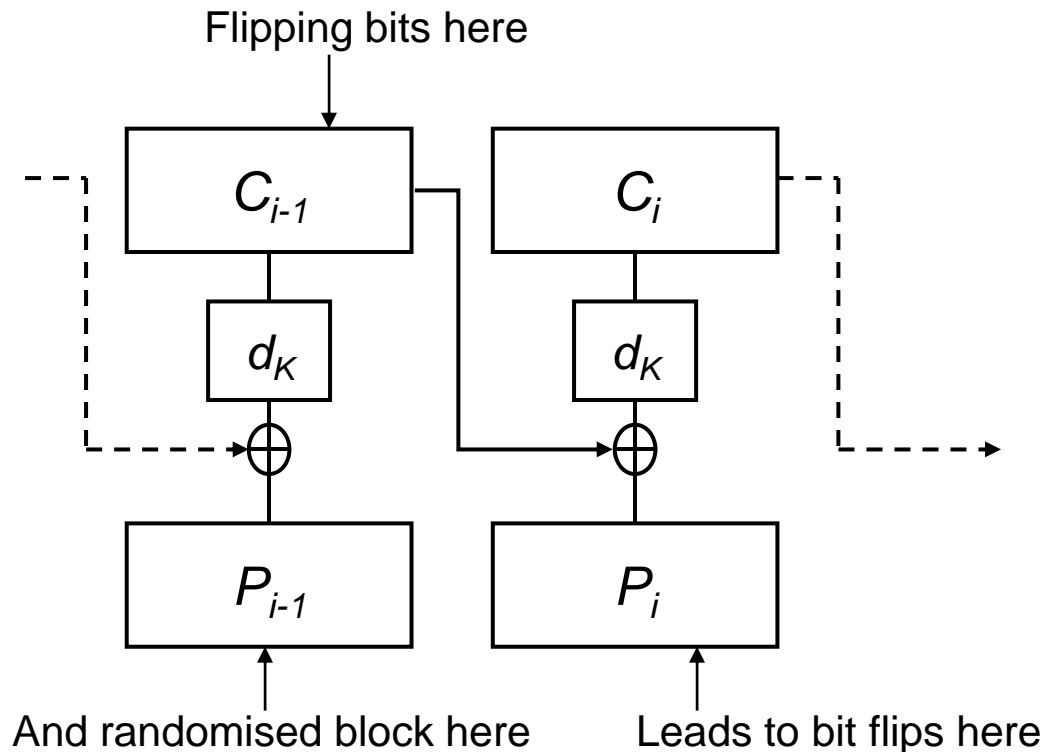
- This provides an example of the type of attack that is possible if an error oracle exists (and how error oracles can arise).
- It also illustrates the possible problems arising from the (complacent) belief that there is no need to follow crypto 'best practice'.
- Such lessons almost certainly apply to many existing widely used security protocols, which do not implement current best practice.

Main ideas

- Extension of Vaudenay's *padding oracle* attacks on CBC mode (*Eurocrypt 2002*) combined with Paterson-Yau techniques (*Eurocrypt 2006*).
- A padding oracle (a special type of error oracle):
 - attacker sends a ciphertext and learns only whether or not the underlying plaintext was correctly padded.
- Vaudenay showed that a padding oracle can be “leveraged” to build a decryption algorithm:
 - for CBC mode encryption;
 - for certain padding methods.

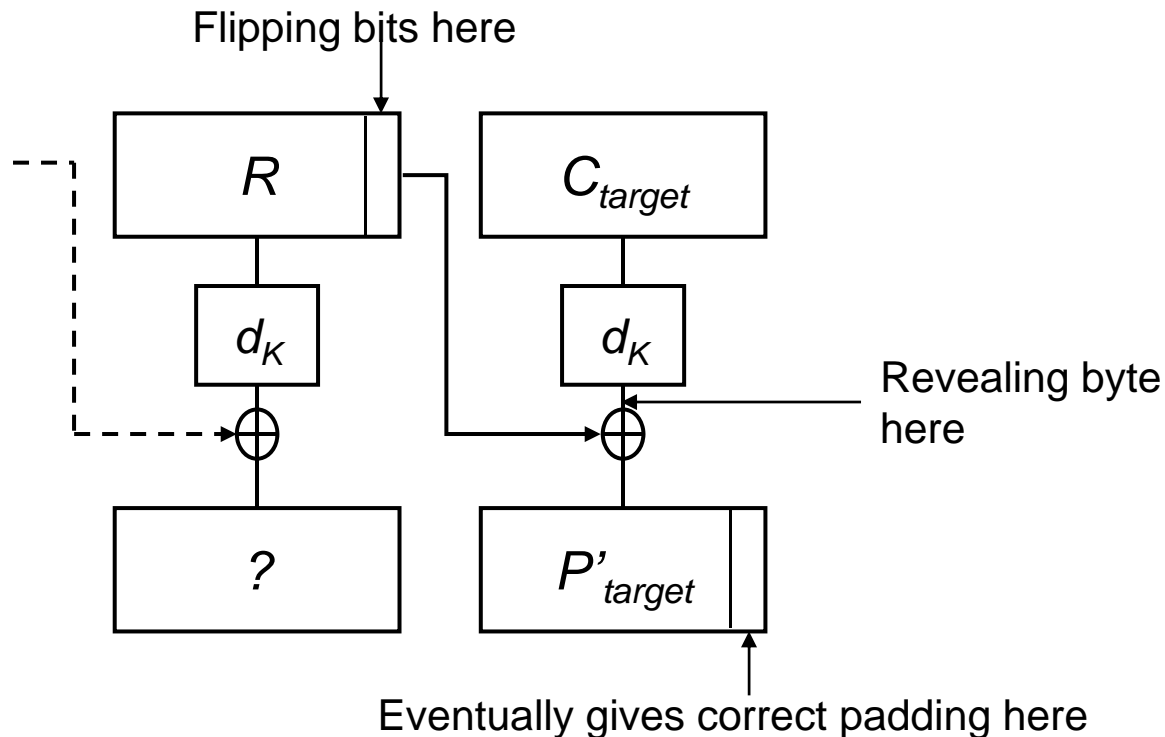
Bit flipping in CBC mode

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



Padding oracles – a toy example

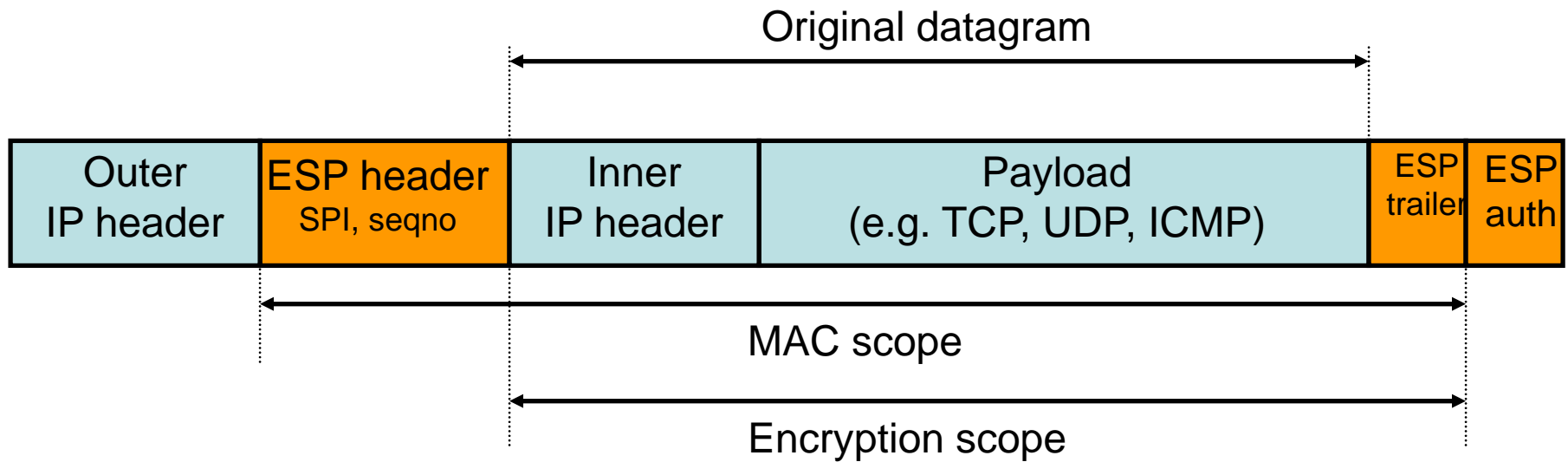
- Suppose only requirement for correct padding is last byte be “01” (hex).
- Repeatedly flip bits in last byte of R and submit to padding oracle.
- Padding oracle says “yes” iff:
 Last byte of $R \oplus$ last byte of $d_K(C_{target}) = 01$
- This reveals last byte of $d_K(C_{target})$ and so last byte of the original P_{target} .



ESP

- 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.

ESP in Tunnel Mode



ESP header and trailer

- ESP specifies header and trailer fields to be added to IP datagrams.
- Fields in header include:
 - Security Parameters Index (SPI).
 - Sequence number.
- Fields in trailer include:
 - Any padding needed for encryption algorithm (may also be used to disguise payload length).
 - Padding length.
 - Next Header byte.
 - MAC value (if integrity protection used).

History of encryption in IPsec

- ESPv1 (1995) provided no integrity protection.
 - Reliant on separate AH protocol to provide this.
 - Bellare (95 and 96) sketched a series of attacks on ESPv1 without AH.
- Bellare-Wagner attack:
 - Limited recovery of plaintext from TCP segments:
 - Requires ciphertexts matching 2^{24} chosen plaintexts.
 - Requires receiver to ignore encryption padding format.
 - attack fails if padding check carried out upon decryption.
 - Recovers last byte of plaintext from TCP segments if byte length is equal to 1 modulo 8.
 - Requirements can be reduced to 2^8 *chosen* plaintexts if variable length padding acceptable.

Bellovin's Attacks (continued)

- Bellovin's paper presents a collection of attack sketches and ideas.
 - Theoretically interesting, but no attacks demonstrated to work in practice.
 - Drew attention to need for integrity protection along with encryption.
 - Sufficiently serious to influence development of RFCs.

Integrity protection and ESPv2

- IETF response to Bellovin's attacks:
 - ESPv2 (1998) recommends receiver should check format of encryption padding.
 - Also includes integrity protection as an option.
 - But implementations must still support “encryption-only” mode.
- ESPv2 represents a compromise between improving security and maintaining backwards-compatibility.

Integrity protection and ESPv3

- ESPv3 (2005):
 - still allows encryption-only ESP.
 - but no longer *requires* support for encryption-only.
 - gives strong warnings about Bellovin-Wagner attack and refers to theoretical cryptography literature to motivate need to use integrity protection.
 - *“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.”*

IPsec in **theory** and practice

- The theoretical cryptography community is well aware of the need to carefully combine integrity protection with encryption to prevent active attacks against encryption.
- Plenty of high-profile, real-world examples:
 - Kerberosv4, IEEE 802.11b, SSH, OpenSSL,...
- It is also well-known amongst IPsec experts that encryption-only configurations should be avoided - clear warnings against their use in the RFCs.
- So is there really any problem?

IPsec in theory and practice

- 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.
- End users don't read RFCs or technical papers.
- End users might reasonably assume that encryption on its own gives confidentiality.
- Many on-line tutorials do not highlight the dangers of encryption-only IPsec.

IPsec in theory and practice

- From the IPsec Tunnel Implementation administrator's guide of a well-known vendor:
 - “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.”
- http://www.cisco.com/en/US/docs/security/security_management/vms/router_mc/1.3.x/user/guide/U13_bldg.html#wp1068306 (last accessed 28/1/2008).

ESP trailer format

- Append a byte pattern of the form:
01 02 ... y
- Append the PL byte (y again).
- Append the NH byte (04 in tunnel mode).

- So valid ESP trailer formats are:

00 04,

01 01 04,

01 02 02 04,

01 02 03 03 04, ...

ESP trailer oracles

- An *ESP trailer oracle* could be exploited to perform decryption.
 - This is an oracle telling the attacker if a trailer's format is valid or invalid.
 - Initial 2-byte pattern "00 04" implies 2^{16} calls to the oracle are needed to extract first 2 bytes.
 - 2^8 calls per byte for remaining bytes of each block.
- To make this work, we need to find a reliable ESP trailer oracle.

ESP Trailer Oracles

- Wrongly formatted ESP trailers should lead to packet drops.
 - Padding checks SHOULD be carried out:
 - but packet drop on failure is not mandated explicitly;
 - NH byte must equal 04 to pass policy checks
- So a packet drop would indicate an incorrectly formatted ESP trailer.
- But we also need an indication of when an ESP trailer is *correctly* formatted.

Building an ESP Trailer Oracle

- If ESP trailer is correctly formatted, then inner packet is eventually processed by IP.
- Paterson-Degabriele built a single tunnel mode packet that *always* results in an ICMP response when its inner packet is processed.
 - Capture a tunnel mode packet.
 - Modify TTL, Protocol or IP header length fields in inner packet by bit flipping.
 - Correct Header Checksum field by bit flipping.
 - One-time cost for construction.

ESP Trailer Oracle Attack

- For any target ciphertext block:
 - Splice random block and target ciphertext block onto end of ICMP-generating packet.
 - Inject this new packet into tunnel.
 - Target block interpreted as ESP trailer
 - ICMP message created if and only if ESP trailer correctly formatted.
 - ICMP message sent encrypted on reverse tunnel
 - Detectable by its length.
- This is an *ESP trailer oracle attack*.
 - Using behaviour of IPsec implementation at receiver coupled with presence/absence of ICMP messages as the oracle.

Implementing the RFC Attacks

- These RFC attacks work “in theory” against any IPsec implementation that strictly follows the RFCs.
- But many practical issues may interfere with the correct operation of the attacks.
- Are any implementations sufficiently strict?
- And what happens in reality?
- Look at open source implementations...

Implementing the RFC attacks

- **Linux:**

- Comment in source code:

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

- No padding check implemented.

- So the RFC attacks don't apply because of incorrect implementation ... but then vulnerable to Bellovin-Wagner attack from 1995!

- Also vulnerable to:

- a variant of the RFC attack which can efficiently extract two bytes per block, implemented as a proof of concept;
- another Paterson-Yau attack.

Implementing the RFC attacks

- **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 rigorous enough for the RFC attacks to work.
 - But a variant of the Paterson-Degabriele RFC attacks extracts three bytes per block for 2^{16} effort.

Implementing the RFC attacks

- **Openswan, strongSwan, FreeS/WAN:**
 - Don't allow selection of encryption-only configurations (despite mandated support in ESPv2).
 - All check padding carefully, but then don't drop packet if it's incorrect!
 - (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 attacks will.

Implementing the RFC attacks

- **OpenSolaris:**

- 3 different levels of padding check can be selected.
 - No check, KAME-style check, full padding check.
- But the full check was incorrectly implemented!
- Paterson and Degabriele reported the bug to Sun.
- Sun fixed it in Release 55 of OpenSolaris.
- After which, they successfully attacked 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 attacks

- There is a range of attacks against encryption-only ESP that work:
 - against any implementation strictly following the RFCs, e.g. OpenSolaris;
 - against many implementations not following the RFCs, e.g. Linux.
- The attacks that work in practice shouldn't work against the RFCs.
- The attacks that work against the RFCs often don't work in practice.

Discussion I

- Encryption-only ESP is dangerously weak in a very practical sense.
- No security is gained from provision of upper layer integrity protection, despite claims to contrary in ESPv3:

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.”

Discussion II

- Attacks reveal poor lines of communication in the IPsec community in its widest sense.
 - Configurations known to be weak to IPsec insiders are still allowed in the standard.
 - These configurations get deployed by end-users.
 - 10 years on, many IPsec implementations don't follow advice given in standards anyway.
 - Why is that? What can we do to address it?

Discussion III

- Ultimately RFCs are standards for interoperability, but this causes problems for RFCs concerned with security.
 - Applying patches to security in each new revision is not the answer.
 - Such standards should **take a more conservative approach and adopt defensive designs.**
 - Despite the many real-world constraints imposed on standards development process.⁴³

Discussion IV

- Attacks reveal disconnect between theory and practice in cryptography.
 - Need for strong integrity protection well understood in theoretical cryptography, but not so well by practitioners and users.
 - Cryptographic implementation details are vital for security, but are not currently considered in theoretical security models.
 - Strong anecdotal evidence suggests these points apply equally to security API designs.
- Unfortunately, the gulfs between cryptographers, users of cryptography, and implementers appear to be growing.

Discussion V

- Implementers seem likely to ignore implementation requirements specified in standards if the reason for them is not blindingly obvious.
- One way of avoiding this problem is to design security APIs to protect implementers against their own ignorance.
- That is, we **should design security APIs to only permit safe use of cryptography.**
- For the confidentiality/integrity issue, this is easily addressed by providing access only to authenticated encryption.

Agenda

1. Introduction
2. Developments in cryptography
3. Encryption and integrity – a case study
4. Specification issues in secure protocols
5. Concluding remarks

Background

- Over last 4 years, new attacks published on long-established standardised authenticated key establishment protocols.
- These protocols derive from the seminal 1979 paper of Needham and Schroeder.
- The protocols had been widely studied and were believed to be secure.
- ISO/IEC 11770-2 appeared in 1996, and no problems were identified until 2004.

New attacks

- In 2004 Cheng and Comley described two attacks on mechanism 12 from ISO/IEC 11770-2.
- ISO published a corrigendum withdrawing the broken mechanism.
- In 2008 a new edition of the standard was published containing a replacement mechanism.
- In mid-2008, Mathuria and Sriram published new 'type attacks' on mechanism 13 from ISO/IEC 11770-2.

Type attacks

- A type attack is an attack in which protocol fields of one type are misrepresented as fields of a different type.
- Typically involves message i of one instance of a protocol being replayed as message j in a different protocol instance.

Parsing ambiguity attacks

- Liqun Chen and I recently described (see IACR eprint 2008/419) a much wider class of attack which applies to almost all standardised key management and authentication protocols.
- Attacks require data string input to a crypto-algorithm to be capable of being interpreted in more than one way.

Relationship to security proofs

- Attack applies to protocols with security proofs.
- Proof models either implicitly or explicitly assume that parsing is unambiguous.
- Problem is that the standards do not cover the unique parsing requirement in the specifications.

Example (Kerberos-like scheme)

- Key distribution protocol using authenticated encryption (mechanism 8 from ISO/IEC 11770-2).
- Requires two parties who wish to set up a shared key (A , B) to share secret keys with a TTP (P) – K_{AP} and K_{BP} , say.
- Has four steps.

Example (continued)

1. $A \rightarrow P: t_A \parallel i_B;$
2. $P \rightarrow A: e_{KAP}(t_A \parallel K \parallel i_B \parallel data) \parallel e_{KBP}(t_P \parallel K \parallel i_A \parallel data)$
3. $A \rightarrow B: e_{KBP}(t_P \parallel K \parallel i_A \parallel data) \parallel e_K(t'_A \parallel i_B \parallel data)$
4. $B \rightarrow A: e_K(t_B \parallel i_A \parallel data)$

where t_A , t'_A and t_B are timestamps, i_A and i_B are identifiers for A and B , \parallel denotes concatenation of bit-strings, K is the newly established session key, and $data$ is reserved for application-specific purposes.

Attack assumptions

- The protocol specification in the standard does not restrict the lengths of the key, identifier or application data fields.
- Suppose C (attacker) chooses his/her identifier so that $i_C = i_A || 0$.
- Suppose also that the application writers do not fix the length of the application data field.

Attack

1. $C \rightarrow P: t_C \parallel i_B;$
2. $P \rightarrow C: e_{K_{CP}}(t_C \parallel K \parallel i_B \parallel data) \parallel e_{K_{BP}}(t_P \parallel K \parallel i_C \parallel data)$
3. $C(A) \rightarrow B: e_{K_{BP}}(t_P \parallel K \parallel i_C \parallel data) \parallel e_K(t \parallel i_B \parallel data)$
4. $B \rightarrow C(A): e_K(t_B \parallel i_A \parallel data)$

$C(A)$ means C impersonating A .

- B will accept the third message as coming from A , because C will read $e_{K_{BP}}(t_P \parallel K \parallel i_C \parallel data)$ as $e_{K_{BP}}(t_P \parallel K \parallel i_A \parallel 0 \parallel data)$

Attack variants

- If the application data field length is fixed, then a similar attack can be launched by varying the length of the session key.
- The difference in lengths of addresses does not have to be one bit – it could be one or more bytes.
- Similar attacks apply to a very wide range of standardised security protocols.

Realising the attacks

- Is it likely that addresses will have variable length?
- Could C arrange for his/her identifier i_C to have the property that $i_C = i_A || x$ (or $i_C = x || i_A$) where i_A is A 's identifier and x is some string?
- It will very much depend on the application.
- For example, if identifiers are email addresses then this is clearly possible.

Fixing the attacked schemes

- Need to ensure that cryptographically protected strings can only be parsed in one way.
- Standards are being modified (including creating seven technical corrigenda) to make this an explicit requirement.
- This could be achieved in a number of ways, e.g. fixed length fields, ASN.1 encoding, ...

A footnote

- The problem exists in the general purpose standards for authentication and key management protocols.
- So far we have not found a case where the problem exists in a ‘real’ protocol based on the standards.
- However, neither can we rule it out.

Agenda

1. Introduction
2. Developments in cryptography
3. Encryption and integrity – a case study
4. Specification issues in secure protocols
5. Concluding remarks

Understanding why I

- The motivation for using cryptography only in certain ways is often obscure to the non-theoretician.
- For example, the need to always integrity protect when encrypting arises from theory.
- The theoretical model used is one in which the cryptanalyst is given apparently unreasonably strong powers (including the ability to have any message he/she likes decrypted, apart from the target ciphertext).

Understanding why II

- If integrity protection is not provided, then the attacker can clearly win in this model.
- The attacker can take the target ciphertext and change one bit and get the result decrypted.
- If done sufficiently often, then the entire plaintext for the target ciphertext can typically be recovered.

Understanding why III

- Practitioners have probably looked at this and deduced that, in practice, attackers will not have access to such a powerful decryption oracle, and have hence concluded they can ignore the theoretical result.
- This is very dangerous, because error oracles (in particular padding oracles) are both omnipresent and very powerful attack aids.

A way forward

- This problem will arise again if we are not all much more vigilant.
- Perhaps the most significant thing we have learnt about using cryptography over the last 10-15 years is that getting it right can be very difficult.
- Just because we are using an apparently strong algorithm does not guarantee that the resulting security protocol will be robust.

The final slide!

- Thanks to:
 - Vladimir Oleshchuk for inviting me to come and speak;
 - Kenny Paterson for allowing me to borrow some of his slides on the IPsec attacks;
 - you for listening.
- Questions?

(by all means contact me offline – e.g. at c.mitchell@rhul.ac.uk – for detailed references to the various work mentioned).