10 Real World Protocols

10.1 Application Layer

Secure e-mail
There are differences between the connection-oriented protocols that we considered previously and e-mail:

- e-mail is a store-and-forward protocol.
  - The originator of a message is not online when it is received.
- Therefore, we cannot provide entity authentication.
  - However, we can provide message origin authentication and non-repudiation of origin.
- While many e-mails are relatively small, for efficiency reasons we still need to use symmetric cryptography to provide confidentiality.
  - As before, we will use a session key, but in this context it is normally referred to as a message key.
  - Since e-mails may be stored after receipt, we need some secure way of keeping message keys.
- In general, a message may have multiple recipients.
  - Again for efficiency reasons, we do not necessarily want to send different encrypted messages to each recipient.

Secure e-mail

- An e-mail message consists of 2 parts:
  1. The headers (or envelope) that are used by the e-mail system to route and deliver the message to the recipients
  2. A body (or contents) that contain the information being sent by the originator.

  In general, as a message is transferred through an e-mail system, the headers are modified.
- With e-mail, we want to supply end-to-end security, i.e., we don't want message transfer agents to:
  - Be able to decrypt and read messages.
  - Have to re-sign messages if changes are made to the headers.

  To achieve this, we will only apply security to the body of a message.

Protocol Elements

- We assume that all e-mail users have certified public keys i.e., they have certificates issued by a CA of a recognized PKI.
- We represent an e-mail message by the pair $\langle H, M \rangle$ where H is the headers and M the body.
- We will consider the case where an originator A is sending a message to t recipients $B_1 \cdots B_t$.
- In this case, the headers H will identify the originator A and the recipients $B_1 \cdots B_t$.

Signed Messages

- Consider a message $\langle H, M \rangle$.
- To sign this message, A uses their private key K_A^- to obtain the signed message $\langle H, \{|M|\}_{K_A^-} \rangle$
- This message can be transferred to the recipients and will arrive as the message $\langle H', \{|M|\}_{K_A^-} \rangle$ where H' represents the modified headers produced during the message transfer.
- The recipients (or in fact anyone) can check the signature was produced by the originator A
  - They can obtain a valid certificate for A and extract A's public key K_A^+.
  - This provides message origin authentication.
- If there is a legal framework and we are using a recognized PKI, we also have non-repudiation of origin.
- Anyone can read the message, i.e., we do not have confidentiality.
- If need be, we can include A's certificate or indeed, A's entire certificate path in the message.

Encrypted Messages

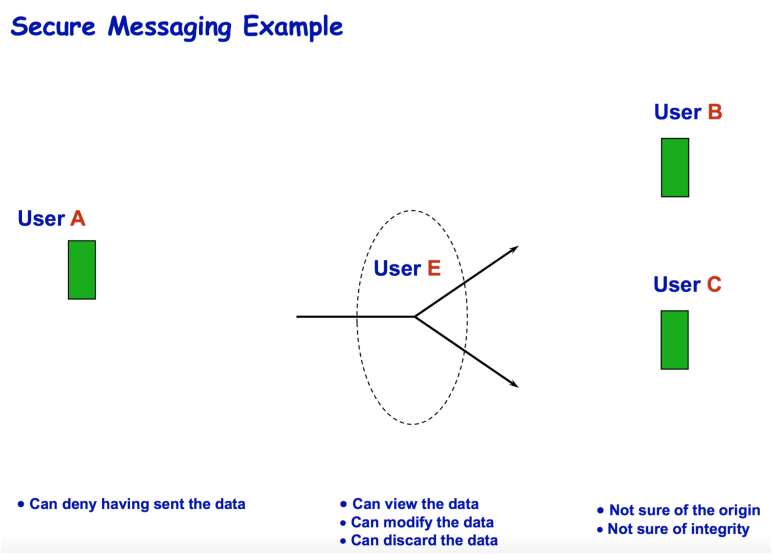
- Again, consider a message $\langle H, M \rangle$.
- To encrypt this message, A can generate a new symmetric message key K_m and encrypt the body M to obtain the message $\langle H, \{M\}_{K_m} \rangle$
- A recipient can only decrypt this message if they know the message key K_m. So we need to send the message key in some protected form to each recipient.

- We do this by creating an encrypted token containing the message key. For B_i we have $\{K_m\}_{K_{B_i}^+}$
- We need one token for each recipient, so we have the encrypted message $\langle H, \{M\}_{K_m}, \{K_m\}_{K_{B_1}^+}, \cdots, \{K_m\}_{K_{B_t}^+} \rangle$
- We can get the public key for each recipient from their certificate.
- This provides confidentiality as only the recipients of the message can recover the plaintext of the body.

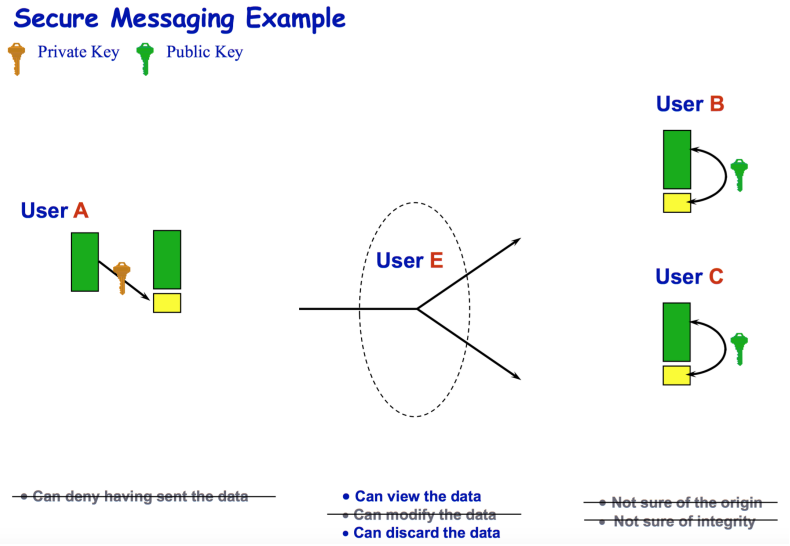
Signed and Encrypted Messages

- We can combine these two techniques to produce a signed and encrypted message $\langle H, \{\{|M|\}_{K_A^-}\}_{K_m}, \{K_m\}_{K_{B_1}^+}, \cdots, \{K_m\}_{K_{B_t}^+} \rangle$
- This protocol provides the following security services:
  - Confidentiality.
  - Integrity.
  - Message Origin Authentication.
  - Non-repudiation of origin (provided the appropriate legal framework exists).
- However, other threats still exist.
  - Replay.
  - Message Deletion.
  - Other denial of service threats, e.g., flooding a network with junk mail.

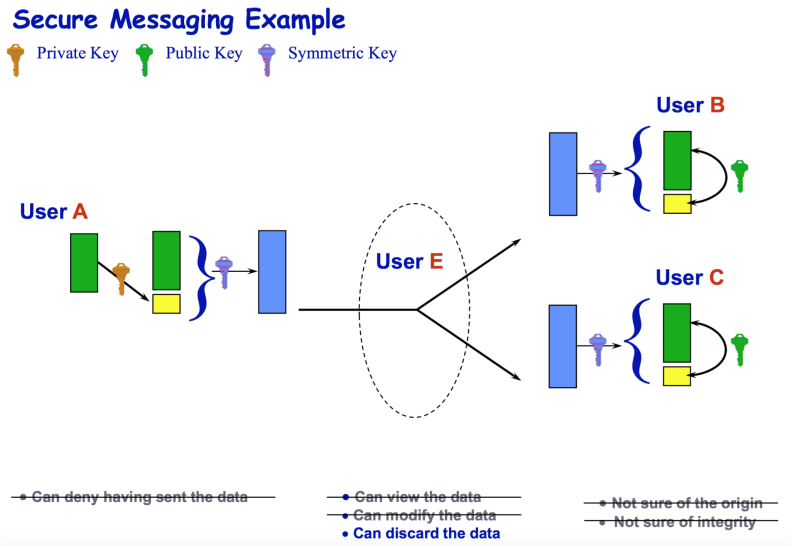
Secure Messaging



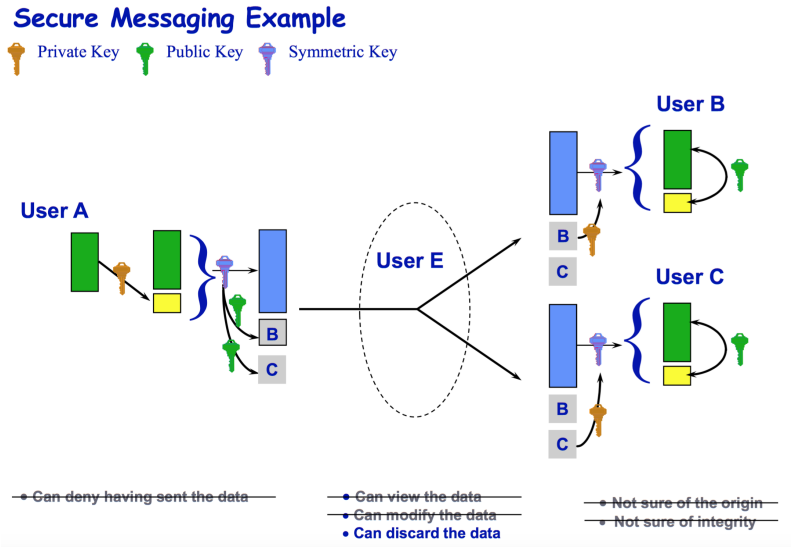
Secure Messaging



Secure Messaging



Secure Messaging



PGP (Pretty Good Privacy)

- Widely used secure email implementation.

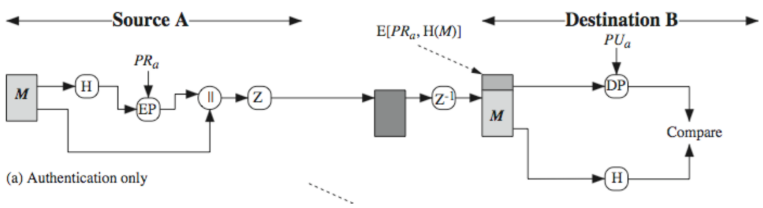
- Originally developed by Phil Zimmermann in 1991.
- For three years, Philip Zimmermann, was threatened with federal prosecution in the United States for his actions. Charges were finally dropped in January 1996.
- At the end of 1999, granted a full license by the U.S. Government to export PGP world-wide, ending a decade-old ban.
- Selected best available cryptographic algorithms and integrated into a single program.
- Works across a variety of platforms.
- Originally free; now also have commercial versions available but free for non-commercial use.

PGP
Services provided:

- Authentication using digital signature
- Confidentiality using message encryption
- Compression
- Email compabitility
- Segmentation

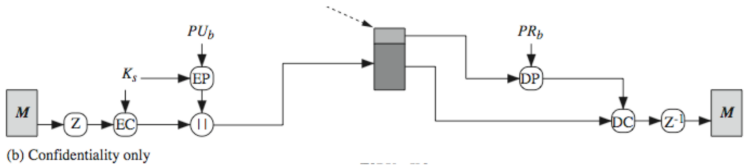
PGP Authentication

1. Sender creates message
2. SHA-1 used to generate 160-bit hash code of message
3. Hash code is encrypted with RSA or DSS using the sender's private key, and result is attached to message
4. Receiver uses RSA or DSS with sender's public key to decrypt and recover hash code
5. Receiver generates new hash code for message and compares with decrypted hash code, if match, message is accepted as authentic



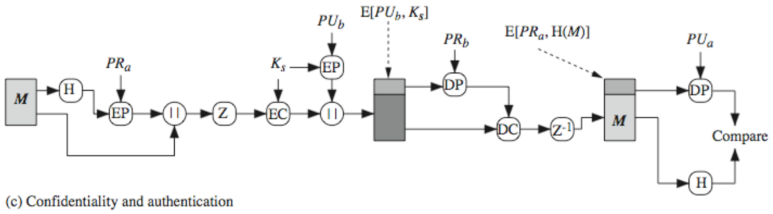
PGP Confidentiality

1. Sender generates message and random 128-bit number to be used as session key for this message only
2. Message is encrypted, using CAST-128/IDEA/3DES with session key
3. Session key is encrypted using RSA/ElGamal with recipient's public key, then attached to message
4. Receiver uses RSA/ElGamal with its private key to decrypt and recover session key
5. Session key is used to decrypt message



PGP Confidentiality and Authentication
Uses both services on same message:

- Create signature and attach to message
- Encrypt both message and signature
- Attach RSA/ElGamal encrypted session key



PGP Compression

- By default PGP compresses message after signing but before encrypting
  - Can store uncompressed message and signature for later verification
- Compression is non-deterministic
  - Uses ZIP compression algorithm

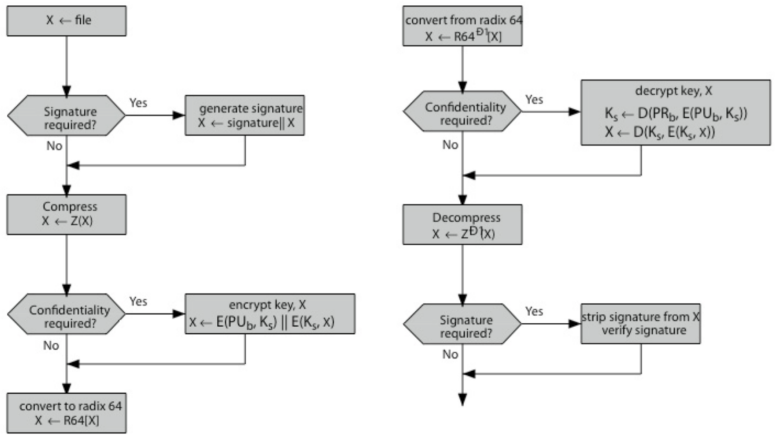
PGP Email Compatability

- When using PGP, will have binary (encrypted) data to send
- However, email was designed only for text
  - Hence PGP must encode raw binary data into printable ASCII characters
- Uses radix-64 or base-64 algorithm
- Maps 3 bytes to 4 printable chars
- Also appends a CRC

PGP Segmentation

- Email systems impose maximum length
  - 50 Kb, for example
- PGP provides automatic segmentation
  - Done after all other operations
  - Thus only one session key needed

PGP Operation Summary



(a) Generic Transmission Diagram (from A)

(b) Generic Reception Diagram (to B)

PGP Operation Summary

- Generating unpredictable session keys
- Identifying keys
  - Multiple public, private key pairs for a user
- Maintaining keys
  - Its own public, private keys of a PGP entity
  - Public keys of correspondents

PGP Session Keys

- Algorithm used: CAST-128
- Input to CAST-128:
  - 128-bit key
  - Two 64-bit plaintexts to be encrypted
- Output using cipher feedback mode:
  - Generates 2 64-bit ciphers from session key
- Plaintexts are from 128-bit randomized number:
  - Based on key stroke of user (timing and actual keys)
  - Then combined with previous session key

PGP Key Identifiers

- Receiver has multiple public keys
  - How do we know which private key is the right one?
- Approach:
  - Send the least significant 64 bits as key ID
  - Need to send the receiver's public key ID used for encrypting the session key
  - Need to send the sender's public key ID, whose corresponding private key was used for signature

PGP Key Rings

- Private key rings

Private Key Ring

Timestamp	Key ID*	Public Key	Encrypted Private Key	User ID*
•	•	•	•	•
•	•	•	•	•
•	•	•	•	•
T_i	$PU_i \bmod 2^{64}$	PU_i	$E(H(P_i), PR_i)$	User i
•	•	•	•	•
•	•	•	•	•
•	•	•	•	•

- Public key rings

Public Key Ring

Timestamp	Key ID*	Public Key	Owner Trust	User ID*	Key Legitimacy	Signature(s)	Signature Trust(s)
•	•	•	•	•	•	•	•
•	•	•	•	•	•	•	•
•	•	•	•	•	•	•	•
T_i	$PU_i \bmod 2^{64}$	PU_i	$trust\_flag_i$	User i	$trust\_flag_i$		
•	•	•	•	•	•	•	•
•	•	•	•	•	•	•	•
•	•	•	•	•	•	•	•

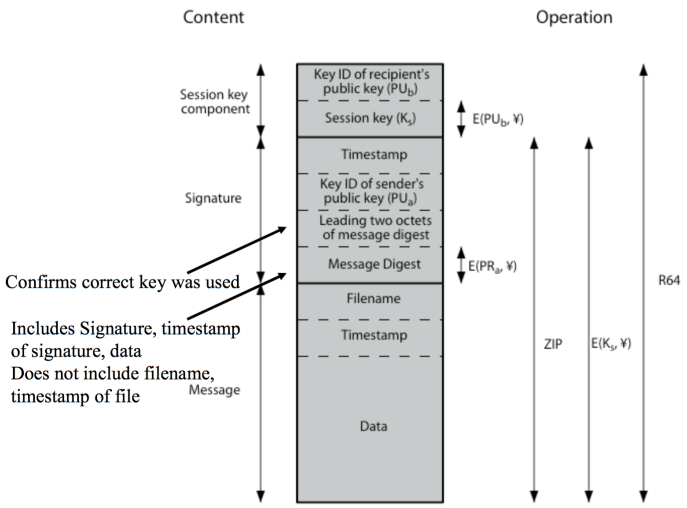
PGP Public Key Management

- A public key attributed to B may belong to C
  - C can send messages to A forging B's signature
  - C can read any encrypted message to B
- Approach to getting true public key:
  - Physically get key from B
  - Obtain B's key from mutual trusted authority
  - Using key legitimacy field computed from the signature trust field and number of certificates for the key

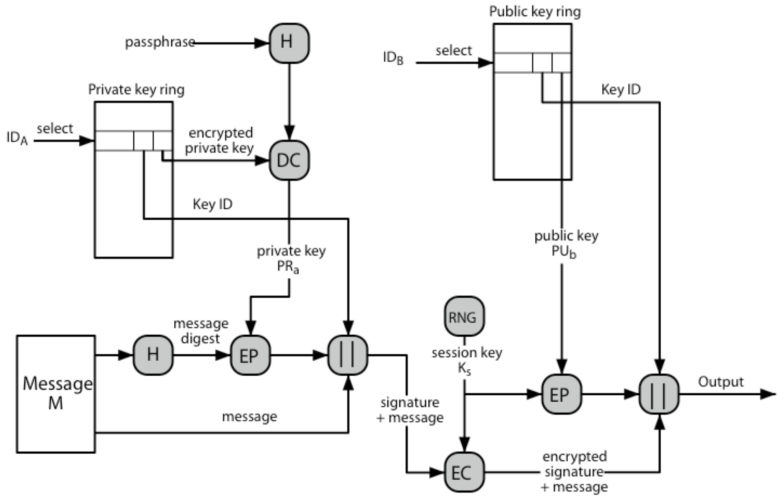
PGP Revoking Public Key

- Possible reasons:
  - It is compromised; private key is open
  - Simply to avoid use of same key for a period
- Approach:
  - Owner issues key revocation certificate, signed by owner
  - Using corresponding private key to sign the certificate
  - Disseminate the certificate as widely and as quickly as possible

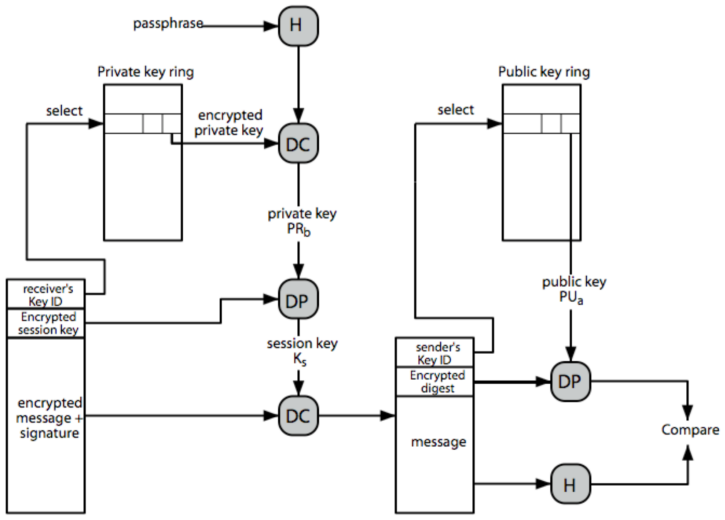
PGP Message Format



PGP Message Generation



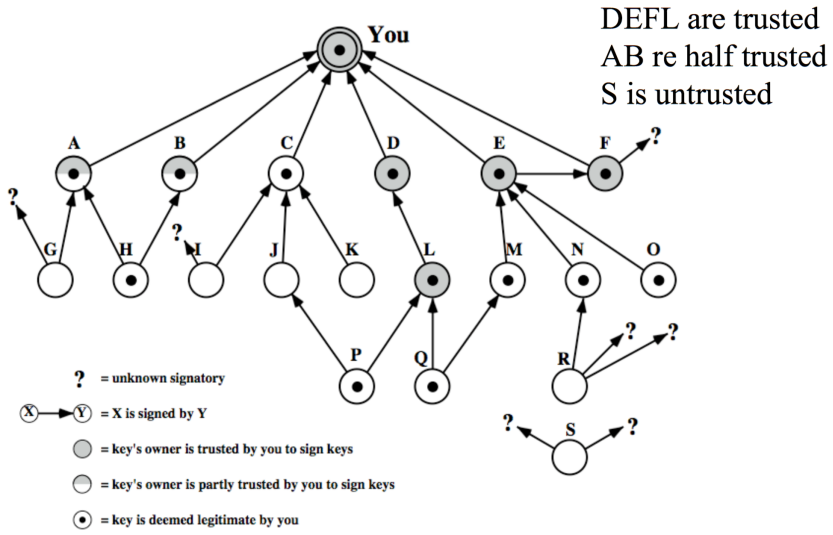
PGP Message Reception



PGP Web of Trust

- There is no need to buy certificates from companies
- A user can sign other user's certificates
- If you trust someone, you can trust users that they sign for
- You can assign a level of trust to each user and hence to the certificate they sign for
- For example:
  - A certificate that is signed by a fully trusted user is fully trusted
  - A certificate signed by two half trusted users is fully trusted
  - A certificate signed by one half trusted user is half trusted
  - Some certificates are untrusted.
- Owners can revoke public key by issuing a revocation certificate signed with the revoked private key
- New web-of-trust certificates have expiry dates

PGP Trust Model



S/MIME (Secure/Multipurpose Internet Mail Extensions)

- Original Internet RFC822 email was text only
- MIME for varying content types and multi-part messages
  - With encoding of binary data to textual form
- S/MIME added security enhancements
  - Enveloped data: Encrypted content and associated keys
  - Signed data: Encoded message + signed digest
  - Clear-signed data: Clear text message + encoded signed digest
  - Signed and enveloped data: Nesting of signed and encrypted entities
- Have S/MIME support in many mail agents
  - E.g., MS Outlook, Mozilla, Mac Mail etc

S/MIME Cryptographic Functions

- Digital signatures: DSS and RSA
- Hash functions: SHA-1 and MD5

- Session key encryption: ElGamal and RSA
- Message encryption: AES, Triple-DES, RC2/40 and others
- MAC: HMAC with SHA-1
- Have process to decide which algorithms to use

S/MIME Cryptographic Messages

- S/MIME secures a MIME entity with a signature, encryption, or both
- Forming a MIME wrapped PKCS object (Public Key Cryptography Standard originally by RSA Inc Now by IETF)

Type	Subtype	Smime parameter	Meaning
Multipart	Signed		clear msg w signature
Application	Pkcs7-mime	signedData	Signed entity
Application	Pkcs7-mime	envelopedData	Encrypted entity
Application	Pkcs7-mime	Degenerate signedData	Certificate only
Application	Pkcs7-mime	CompressedData	Compressed entity
Application	Pkcs7-signature	signedData	Signature

```
Content-Type: application/pklcs7-mime; smime-type=signedData; name=smime.p7m
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7m
```

S/MIME Certificate Processing

- S/MIME uses X.509 v3 certificates
- Managed using a hybrid of a strict X.509 CA hierarchy and enterprise's CAs
- Each client has a list of trusted CA's certificates and his own public/private key pairs and certificates
- Several types of certificates with different levels of checks:
  - Class 1: Email and web browsing
  - Class 2: Inter-company email
  - Class 3: Banking, ...

S/MIME

RFC	Description
2821	Simple Mail Transfer Protocol (**SMTP**)
2822	Internet Message Format
1939	Post Office Protocol - Version 3 (**POP3**)
3501	Internet Message Access Protocol (**IMAP**)
2045-2049	Multipurpose Internet Mail Extensions (**MIME**)
3369	Cryptographic Message Syntax (**CMS**)
3370	**CMS** Algorithms
3850	**S/MIME** Certificate Handling
3851	**S/MIME** Message Specification

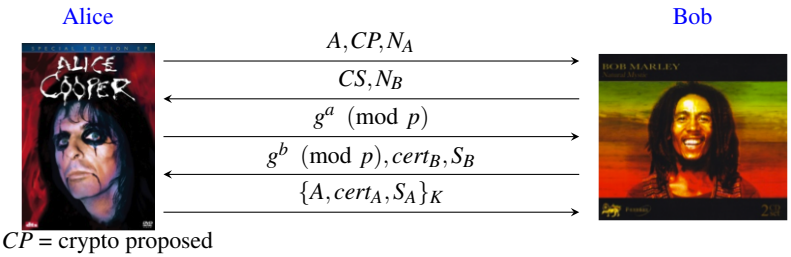
S/MIME

- RFCs 2821, 2822, 1939 & 3501 define Internet e-mail.
  - An e-mail message is just a piece of ASCII text.
  - The initial lines are the headers and the remainder the body.
- RFCs 2045-2049 define how non-ASCII data can be encoded as ASCII text and included in an e-mail message.
  - In particular, these RFCs define how the body of a message can be split into a number of body-parts each containing different types of data.
- RFCs 2269, 3370, 3350, 3851, $\cdots$ define how secure messages can be represented as MIME body-parts.
  - In particular, S/MIME uses `ASN.1` to define a concrete syntax for secure messages.
  - This syntax results in binary data (`ASN.1` encodings) that need to be converted to MIME body-parts.

SSH (Secure Shell)

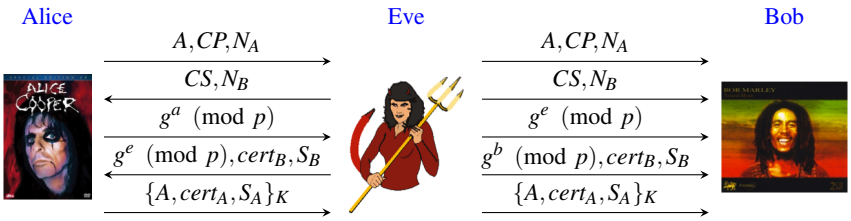
- Creates a secure tunnel
- Insecure commands sent through SSH tunnel are then secure
- SSH is a relatively simple protocol
- SSH authentication can be based on any of the following:
  - Public keys
  - Digital certificates
  - Passwords
- Here, we consider certificate mode
- We consider slightly simplified SSH

Simplified SSH



- CP = crypto proposed
- CS = crypto selected
- $H = H(A, B, CP, CS, N_A, N_B, g^a \pmod p, g^b \pmod p, g^{ab} \pmod p)$
- $S_B = \{|H|\}_{K_B^-}$
- $S_A = \{|H, A, cert_A|\}_{K_A^-}$
- $K = g^{ab} \pmod p$

Man-in-the-Middle Attack on SSH?

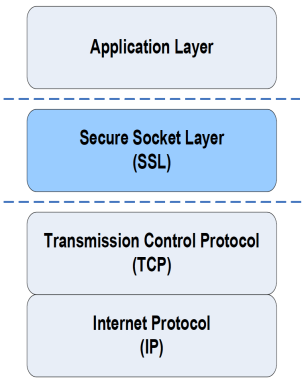


Where does this attack fail?

- Alice computes: $H_A = H(A, B, CP, CS, N_A, N_B, g^a \pmod p, g^e \pmod p, g^{ae} \pmod p)$
- But Bob signs: $H_B = H(A, B, CP, CS, N_A, N_B, g^e \pmod p, g^b \pmod p, g^{be} \pmod p)$

10.2 Transport Layer

SSL (Secure Socket Layer

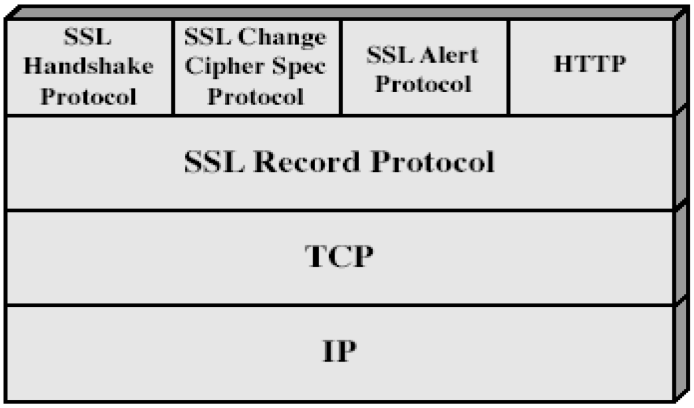


- "Socket layer" lives between application and transport layers
- SSL usually between HTTP and TCP

SSL

- SSL is the protocol used for majority of secure transactions on the Internet
- For example, if you want to buy a book from Amazon:
  - You want to be sure you are dealing with Amazon (authentication)
  - Your credit card information must be protected in transit (confidentiality and/or integrity)
  - As long as you have money, Amazon does not care who you are
  - So, no need for mutual authentication

SSL Architecture



SSL Architecture
SSL session:

- an association between client and server
- created by the Handshake Protocol
- define a set of cryptographic parameters
- may be shared by multiple SSL connections
- uses costly public key operations

SSL connection:

- a transient, peer-to-peer, communications link
- associated with one SSL session
- SSL has an efficient protocol for opening new connections given an existing session

SSL Record Protocol
Confidentiality:

- using symmetric encryption with a shared secret key defined by Handshake Protocol
- IDEA, RC2-40, DES-40, DES, 3DES, Fortezza, RC4-40, RC4-128
- message is compressed before encryption

Message Integrity:

- using a MAC with shared secret key
- similar to HMAC but with different padding

SSL Change Cipher Specification Protocol

- One of three SSL specific protocols which use the SSL Record protocol
- A single message
- Causes pending state to become current
- Hence updating the cipher suite in use

SSL Alert Protocol

- Conveys SSL-related alerts to peer entity
- Severity: warning or fatal
- Specific alert
  - Unexpected message, bad record MAC, decompression failure, handshake failure, illegal parameter
  - Close notify, no certificate, bad certificate, unsupported certificate, certificate revoked, certificate expired, certificate unknown
- Compressed and encrypted like all SSL data

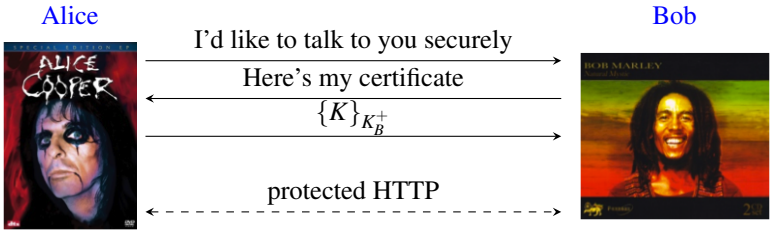
SSL Handshake Protocol
Allows server and client to:

- authenticate each other
- negotiate encryption and MAC algorithms
- negotiate cryptographic keys to be used

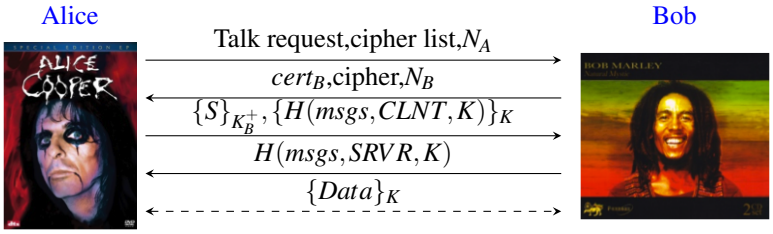
Comprises a series of messages in phases:

1. Establish Security Capabilities
2. Server Authentication and Key Exchange
3. Client Authentication and Key Exchange
4. Finish

Simple SSL-like Protocol



Simplified SSL Protocol



- S is known as the pre-master secret
- $K = H(S, N_A, N_B)$
- "msgs" means all previous messages
- $CLNT$ and $SRVR$ are constants

SSL Keys
6 "keys" derived from $K = H(S, N_A, N_B)$

- 2 encryption keys: send and receive
- 2 integrity keys: send and receive
- 2 IVs: send and receive

Why different keys in each direction?

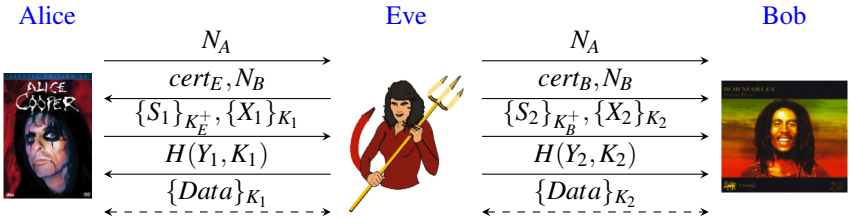
SSL Authentication
Alice authenticates Bob, not vice-versa

- How does client authenticate server?
- Why would server not authenticate client?

Mutual authentication is possible: Bob sends certificate request in message 2

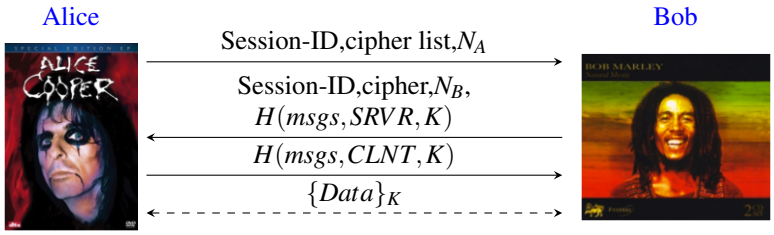
- Then client must have a valid certificate
- But, if server wants to authenticate client, server could instead require password

Man-in-the-Middle Attack on SSL?



- What prevents this MiM "attack"?
  - Bob's certificate must be signed by a certificate authority
- What does browser do if signature not valid?
- What does user do when browser complains?

SSL Connection



Geoff Hamilton

- Assuming SSL session exists
- So, S is already known to Alice and Bob
- Both sides must remember Session-ID
- $K = H(S, N_A, N_B)$
- No public key operations (relies on known S)

SSL Attacks

- Certificate Injection Attack
  - The list of trusted Certificate Authorities is altered
  - Can be avoided by upgrading the OS or switching to a safer one.
- Man-in-the-Middle
  - Cipher Spec Rollback: regresses the public key encryption algorithms
  - Version Rollback: regression from SSL 3.0 to weaker SSL 2.0
  - Algorithm rollback: modify public encryption method
  - Truncation attack: TCP FIN|RST used to terminate connection
- Timing attack
  - Can be avoided by randomly delaying the computations
- Brute force
  - Can be used on servers that accept small key sizes: 40 bits for symmetric encryptions and 512 for the asymmetric one.

TLS (Transport Layer Security)
IETF standard RFC 2246 similar to SSLv3 with minor differences:

- in record format version number
- uses HMAC for MAC
- a pseudo-random function expands secrets
- has additional alert codes
- some changes in supported ciphers
- changes in certificate negotiations
- changes in use of padding

TLS
TLS has about 30 possible cipher 'suites', combinations of key exchange, encryption method, and hashing method.

- Key exchange includes: RSA, DSS, Kerberos
- Encryption includes: IDEA(CBC), RC2, RC4, DES, 3DES, and AES
- Hashing: SHA and MD5

Some of the suites are intentionally weak export versions.

TOR (The Onion Router)

- Tor is a successful privacy enhancing technology that works at the transport layer
- Normally, any TCP connection you make on the internet automatically reveals your IP address
- Tor allows you to make TCP connections without revealing your IP address
- It is most commonly used for HTTP (web) connections
- Scattered around the Internet are about 1000 Tor nodes, also called Onion Routers

TOR
Say Alice wants to connect to a web server without revealing her IP address:

1. Alice picks one of the Tor nodes (n_1) and uses public-key cryptography to establish an encrypted communication channel to it (much like TLS)
2. Alice tells n_1 to contact a second node (n_2), and establishes a new encrypted communication channel to n_2, tunnelled within the previous one to n_1
3. Alice tells n_2 to contact a third node (n_3), and establishes a new encrypted communication channel to n_3, tunnelled within the previous one to n_2
4. And so on, for as many steps as she likes (usually 3)
5. Alice tells the last node (within the layers of tunnels) to connect to the website

TOR

- Alice now shares three symmetric keys:
  - K_1 with n_1
  - K_2 with n_2
  - K_3 with n_3
- When Alice wants to send a message M, she actually sends $\{\{\{M\}_{K_3}\}_{K_2}\}_{K_1}$

- Node n_1 uses K_1 to decrypt the outer layer, and passes the result $\{\{M\}_{K_3}\}_{K_2}$ to n_2
- Node n_2 uses K_2 to decrypt the next layer, and passes the result $\{M\}_{K_3}$ to n_3
- Node n_3 uses K_3 to decrypt the final layer, and sends M to the website

TOR

- When the website replies with message R, it will send it to node n_3
- Node n_3 will encrypt R with K_3 and send $\{R\}_{K_3}$ to n_2
- Node n_2 will encrypt that with K_2 and send $\{\{R\}_{K_3}\}_{K_2}$ to n_1
- Node n_1 will encrypt that with K_1 and send $\{\{\{R\}_{K_3}\}_{K_2}\}_{K_1}$ to Alice
- Alice will use K_1, K_2, and K_3 to decrypt the layers of the reply and recover R

TOR

- Node n_1 knows that Alice is using Tor, and that her next node is n_2, but does not know which website Alice is visiting
- Node n_3 knows some Tor user (with previous node n_2) is using a particular website, but doesn't know who
- The website itself only knows that it got a connection from Tor node n_3
- The connection between n_3 and the website is not encrypted - if you want encryption as well as the benefits of Tor, you should use end-to-end encryption in addition (like HTTPS)

TOR

Tor provides for anonymity in TCP connections over the Internet, both unlinkably (long-term) and linkably (short-term):

- There is no long-term identifier for a Tor user
- If a web server gets a connection from Tor today, and another one tomorrow, it won't be able to tell whether those are from the same person
- But two connections in quick succession from the same Tor node are more likely to in fact be from the same person

10.3 Network Layer

IPSec

- IPSec lives at the network layer and is transparent to applications
- Complex protocol
- Over-engineered
  - Lots of (generally useless) features
- Flawed
  - Some significant security issues
- Interoperability is serious challenge
  - Defeats the purpose of having a standard

IPSec
Two parts to IPSec:

- IKE: Internet Key Exchange
  - Mutual authentication
  - Establish session key
  - Two “phases” - like SSL session/connection
- ESP/AH
  - ESP: Encapsulating Security Payload - for encryption and/or integrity of IP packets
  - AH: Authentication Header - integrity only

IKE

- IKE has 2 phases:
  - Phase 1- IKE security association (SA)
  - Phase 2 - AH/ESP security association
- Phase 1 is comparable to SSL session
- Phase 2 is comparable to SSL connection
- Not an obvious need for two phases in IKE
- If multiple Phase 2’s do not occur, then it is more costly to have two phases

IKE Phase 1
Four different "key" options:

- Public key encryption (original version)
- Public key encryption (improved version)
- Public key signature
- Symmetric key

For each of these, two different "modes":

- Main mode and aggressive mode

There are 8 versions of IKE Phase 1

IKE Phase 1
We discuss 6 of 8 Phase 1 variants:

- Public key signatures (main and aggressive modes)
- Symmetric key (main and aggressive modes)
- Public key encryption (main and aggressive)

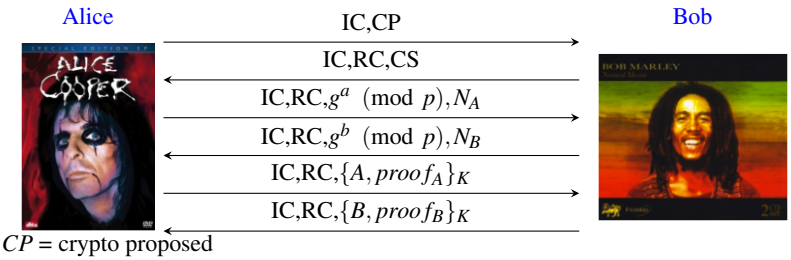
Why public key encryption and public key signatures?

- Always know your own private key
- May not (initially) know other side's public key

IKE Phase 1

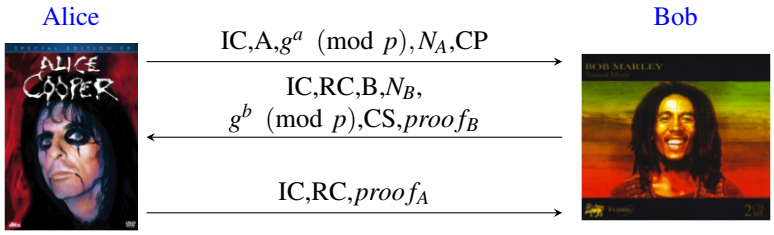
- Uses ephemeral Diffie-Hellman to establish session key
  - Provides perfect forward secrecy (PFS)
- Let a be Alice's Diffie-Hellman exponent
- Let b be Bob's Diffie-Hellman exponent
- Let g be generator and p prime
- Recall that p and g are public

IKE Phase 1: Digital Signature (Main Mode)



- CP = crypto proposed
- CS = crypto selected
- IC = initiator cookie
- RC = responder cookie
- $K = H(IC, RC, g^{ab} \pmod p, N_A, N_B)$
- $SKEYID = H(N_A, N_B, g^{ab} \pmod p)$
- $proof_A = \{|H(SKEYID, g^a \pmod p, g^b \pmod p, IC, RC, CP, A))|\}_{K_A^-}$

IKE Phase 1: Digital Signature (Aggressive Mode)



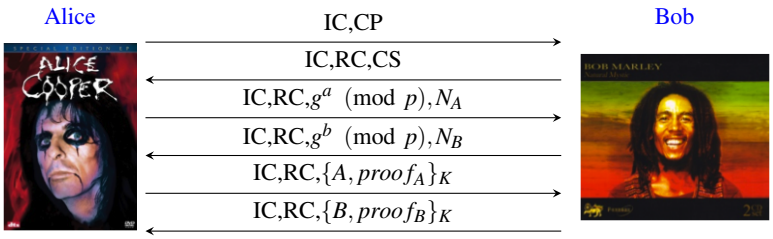
Main differences from main mode:

- Not trying to protect identities
- Cannot negotiate g or p

Main vs Aggressive Modes

- Main mode must be implemented
- Aggressive mode should be implemented
- Might create interoperability issues
- For public key signature authentication:
  - Passive attacker knows identities of Alice and Bob in aggressive mode, but not in main mode
  - Active attacker can determine Alice's and Bob's identity in main mode

IKE Phase 1: Symmetric Key (Main Mode)



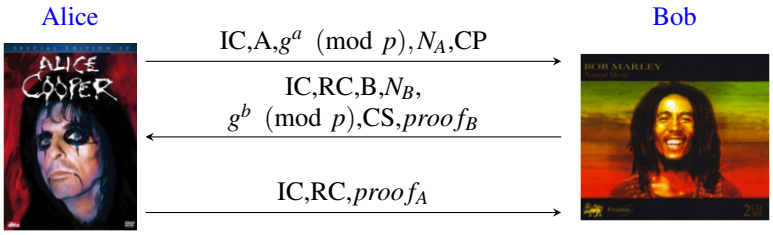
Same as signature mode except:

- K_{AB} = symmetric key shared in advance
- $K = H(IC, RC, g^{ab} \pmod p, N_A, N_B, K_{AB})$
- SKEYID = $H(K, g^{ab} \pmod p)$
- $proof_A = H(SKEYID, g^a \pmod p, g^b \pmod p, IC, RC, CP, A)$

Problems with Symmetric Key (Main Mode)

- Catch-22:
  - Alice sends her ID in message 5
  - Alice's ID encrypted with K
  - To find K Bob must know K_{AB}
  - To get K_{AB} Bob must know he's talking to Alice!
- Result: Alice's ID must be IP address!
- Useless mode for people who move around
- Why go to all of the trouble of trying to hide identities in 6 message protocol?

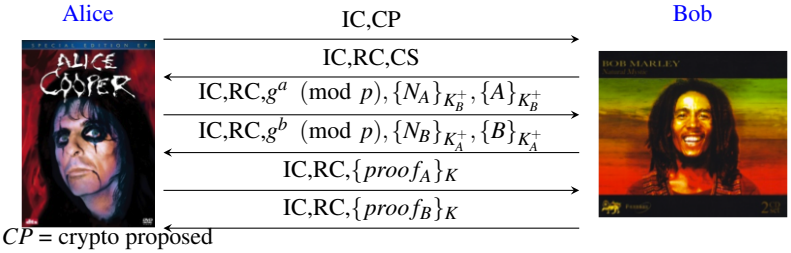
IKE Phase 1: Symmetric Key (Aggressive Mode)



- Same format as digital signature aggressive mode
- Not trying to hide identities

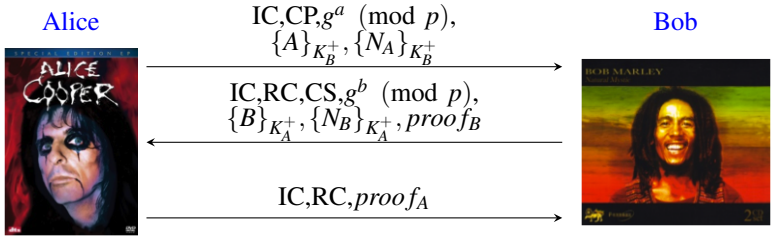
- As a result, does not have problems of main mode
- But does not (pretend to) hide identities

IKE Phase 1: Public Key Encryption (Main Mode)



- CP = crypto proposed
- CS = crypto selected
- IC = initiator cookie
- RC = responder cookie
- $K = H(IC, RC, g^{ab} \pmod p, N_A, N_B)$
- $SKEYID = H(N_A, N_B, g^{ab} \pmod p)$
- $proof_A = H(SKEYID, g^a \pmod p, g^b \pmod p, IC, RC, CP, A))$

IKE Phase 1: Public Key Encryption (Aggressive Mode)



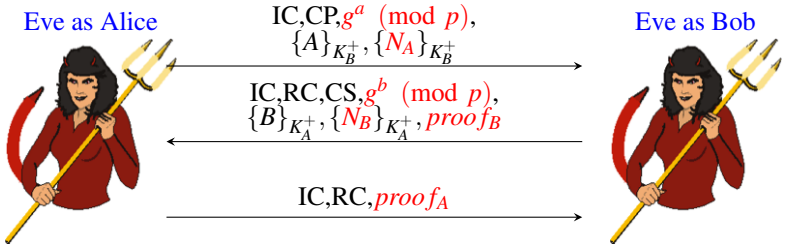
- K, $proof_A$, $proof_B$ computed as in main mode
- Note that identities are hidden
  - The only aggressive mode to hide identities
  - So, why have a main mode?

Public Key Encryption Issue?
In public key encryption (aggressive mode):

- Suppose Eve generates:
  - Exponents a and b
  - Nonces N_A and N_B
- Eve can compute "valid" keys and proofs: $g^{ab} \pmod p$, K, $SKEYID$, $proof_A$ and $proof_B$

This also works in main mode.

Public Key Encryption Issue?



- Eve can create exchange that appears to be between Alice and Bob
- Appears valid to any observer, including Alice and Bob!

Plausible Deniability

- Eve can create a "conversation" that appears to be between Alice and Bob
- Appears valid, even to Alice and Bob!
- A security failure?
- In this IPSec key option, it is a feature:
  - Plausible deniability: Alice and Bob can deny that any conversation took place!
- In some cases it might create a problem:
  - E.g., if Alice makes a purchase from Bob, she could later repudiate it (unless she had signed)

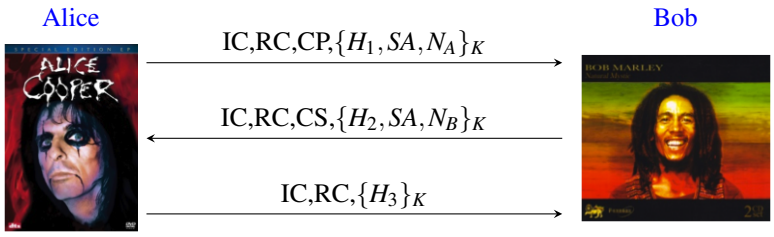
IKE Phase 1 Cookies

- IC and RC are cookies (or "anti-clogging tokens") supposed to prevent denial of service attacks (no relation to Web cookies)
- To reduce denial of service threats, Bob wants to remain stateless as long as possible
- But Bob must remember CP from message 1 (required for proof of identity in message 6)
- Bob must keep state from message 1 on, so these "cookies" offer little denial of service protection

IKE Phase 1 Summary

- Result of IKE phase 1 is:
  - Mutual authentication
  - Shared symmetric key
  - IKE Security Association (SA)
- But phase 1 is expensive
  - Especially in public key and/or main mode
- Developers of IKE thought it would be used for lots of things - not just IPSec

IKE Phase 2

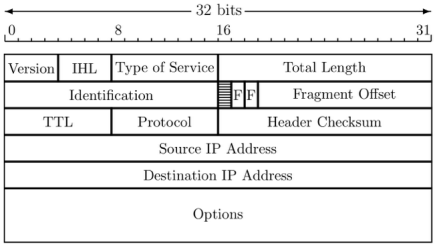


- K, IC, RC and SA known from Phase 1
- Proposed CP includes ESP and/or AH
- Hashes H_1, H_2, H_3 depend on $SKEYID$, SA, N_A and N_B
- Keys derived from KEYMAT = $H(SKEYID, N_A, N_B, junk)$
- Recall $SKEYID$ depends on phase 1 key method
- Optional perfect forward secrecy (ephemeral Diffie-Hellman exchange)

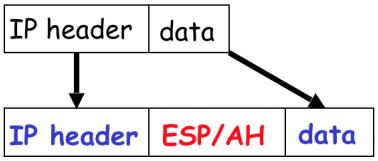
IPSec

- After IKE Phase 1, we have an IKE SA
- After IKE Phase 2, we have an IPSec SA
- Both sides have a shared symmetric key
- Now we want to protect IP datagrams of the form:

- Where IP header is of the form:

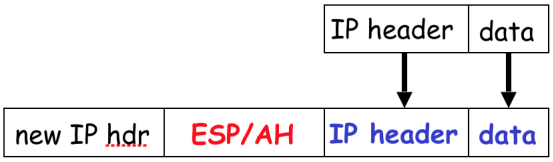


IPSec Transport Mode



- Transport mode designed for host-to-host traffic
- Transport mode is efficient
  - Adds minimal amount of extra header
- The original header remains
  - Passive attacker can see who is talking
- There may be firewalls in between

IPSec Tunnel Mode



- Tunnel mode designed for firewall-to-firewall traffic
- Original IP packet encapsulated in IPSec
- Original IP header not visible to attacker
  - New IP header from firewall to firewall
  - Attacker does not know which hosts are talking
- Local networks not protected
- Transport mode not necessary, but is more efficient

AH vs ESP
AH: Authentication Header

- Integrity only (no confidentiality)
- Integrity-protect everything beyond IP header and some fields of header (why not all fields?)

ESP: Encapsulating Security Payload

- Integrity and confidentiality both required
- Protects everything beyond IP header
- Integrity-only by using NULL encryption

Why Does AH Exist?

- Cannot encrypt IP header
  - Routers must look at the IP header
  - IP addresses, TTL, etc.
  - IP header exists to route packets
- AH protects immutable fields in IP header
  - Cannot integrity protect all header fields
  - TTL, for example, will change
- ESP does not protect IP header at all but encrypts everything beyond it (if non-null encryption)
- If ESP-encrypted, firewall cannot look at TCP header (e.g., port numbers)
- Why not use ESP with NULL encryption?
  - Firewall sees ESP header, but does not know whether null encryption is used
  - End systems know, but not the firewalls

IPSec vs SSL
IPSec:

- Lives at the network layer (part of the OS)
- OS must be aware, but not apps
- Encryption, integrity, authentication, etc.
- Is overly complex, has some security “issues”

- Often used in VPNs

SSL:

- Lives at socket layer (part of user space)
- Apps must be aware, but not OS
- Encryption, integrity, authentication, etc.
- Relatively simple and elegant specification
- Built into web early on (Netscape)

SSL session is like IKE phase 1; SSL connection is like IKE phase 2.

10.4 Link Layer

Wireless LAN Security
Standards: IEEE 802.11 Series

- 802.11b: 11Mbps @ 2.4GHz
  - No inherent security
  - Uses the WEP protocol
- 802.11a: 54Mbps @ 5.7GHz
- 802.11g: 54Mbps @ 2.4GHz
- 802.1X: security add-on
- 802.11i: high security

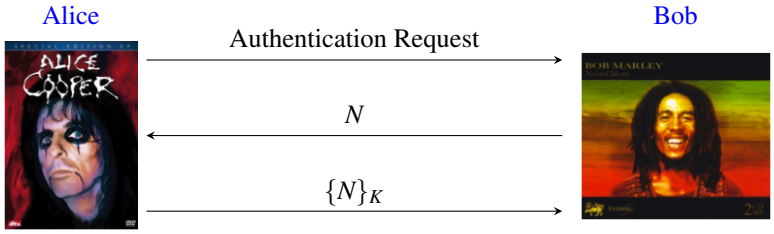
WEP (Wired Equivalent Privacy)
Very badly designed security protocol.
WEP uses RC4 for confidentiality.

- Considered a strong cipher.
- But WEP introduces a subtle flaw.

WEP uses CRC for integrity.

- Should have used a MAC instead.
- CRC is for error detection, not cryptographic integrity.

WEP Authentication



- Bob is wireless access point
- Key K shared by access point and all users
  - K seldom (if ever) changes
- WEP has many security flaws

WEP

WEP "integrity" does not provide integrity.

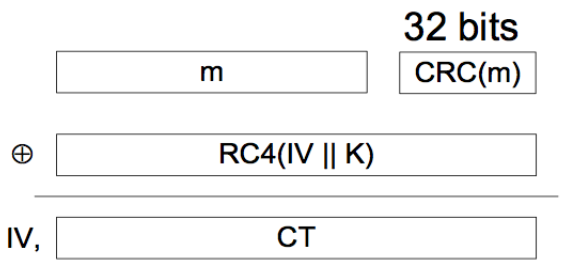
- CRC is linear, so is stream cipher XOR
- Can change ciphertext and CRC so that checksum remains correct.
- Such introduced errors go undetected.
- This requires no knowledge of the plaintext.
- Even worse if plaintext is known.
- CRC does not provide a cryptographic integrity check.
  - CRC designed to detect random errors.
  - Not designed to detect intelligent changes.

WEP

WEP uses a long-term secret 128-bit key K.
RC4 is a stream cipher, so each packet must be encrypted using a different key.

- 24-bit Initialization Vector (IV) sent with packet
- Sent in the clear (IV is not secret)

Actual RC4 key for packet is $(IV||K)$



WEP
A simple linear checksum has the property:

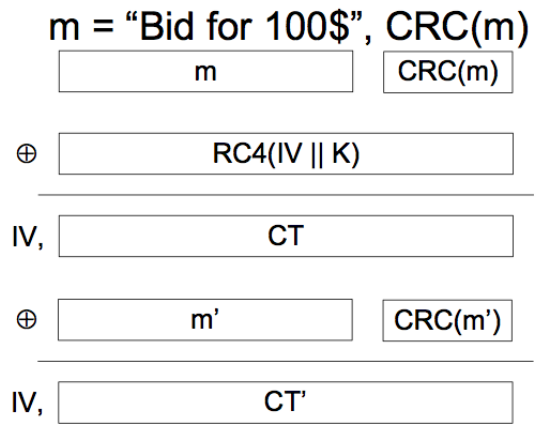
$$\text{CRC}(x \oplus y) = \text{CRC}(x) \oplus \text{CRC}(y)$$

Say the ciphertext $CT = \text{RC4}(IV||K) \oplus (m,\text{CRC}(m))$
The attacker creates $(m',\text{CRC}(m'))$ and XORs this with the ciphertext to get:

$$\begin{aligned} CT' &= CT \oplus (m',\text{CRC}(m')) \\ &= \text{RC4}(IV||K) \oplus (m,\text{CRC}(m)) \oplus (m',\text{CRC}(m')) \\ &= \text{RC4}(IV||K) \oplus (m \oplus m',\text{CRC}(m \oplus m')) \end{aligned}$$

The attacker has changed the ciphertext in such a way that it will now decrypt to $m \oplus m'$, and the CRC will still be okay.
If the attacker wants to change the message to m'', then select m' s.t. $m \oplus m' = m''$ i.e. $m' = m \oplus m''$.

WEP
Consider the following message to eBay:



Select m' such that $m \oplus m'$ = “Bid for 900$”

WEP
Each packet gets a new IV.
Long term key K seldom changes
If long-term key K and IV are the same, then same keystream is used.

- This is bad.
- It is at least as bad as reuse of one-time pad.
- If $C_1 = \text{RC4}(IV||K) \oplus P_1$ and $C_2 = \text{RC4}(IV,K) \oplus P_2$ then:

$$C_1 \oplus C_2 = P_1 \oplus P_2$$

WEP
This attack would be prevented if a different IV were used for every message encrypted using the same key.
However this was merely recommended in the standard and not required.
So many manufacturers simply reset it to 0 every time the card was powered up.
Some generated them randomly, but with only 2^{24} possible variations, a collision can be expected after just 2^{12} packets.
In practice it is common for the same IV to be re-used with the same key.

WEP
Suppose an attacker can insert traffic and observe corresponding ciphertext:

- Then they know the keystream for some IV
- They can decrypt any packets that use that IV
- If the attacker does this many times, they can then decrypt data for lots of IVs
- Known plaintext attack

802.11b: Proposed Solutions

- Virtual Private Network
- Closed Network
- Through the use of SSID
- Ethernet MAC address control lists
- Replace RC4 with block cipher
- Do not reuse IV
- Automatic Key Assignment

802.1X: Interim Solution

- Port-based authentication
  - Not specific to wireless networks
- Authentication servers
  - RADIUS
- Client authentication
  - Extensible Authentication Protocol (EAP)
- Possible attacks
  - Man-in-the-Middle
  - Session Hijacking
- Proposed solutions
  - Per-packet authenticity and integrity
  - Authenticity and integrity of EAPOL messages
  - Two-way authentication

WPA (Wi-Fi Protected Access)

- Addresses issues with WEP
  - Key management: Temporal Key Integrity Protocol (TKIP)
  - Integrity: Message Integrity Check (MIC)
- Software upgrade only
- Compatible with 802.1X
- Compatible with 802.11i

802.11i

- Finalized: June, 2004
- Robust Security Network
- Wi-Fi Alliance: WPA2
- Improvements made:
  - Authentication enhanced
  - Key Management created
  - Data Transfer security enhanced

802.11i
Authentication

- Authentication Server
- Two-way authentication
  - Prevents man-in-the-middle attacks
  - Master Key (MK)
  - Pairwise Master Key (PMK)

Key Management

- Key Types:
  - Pairwise Transient Key
  - Key Confirmation Key
  - Key Encryption Key
  - Group Transient Key
  - Temporal Key

802.11i
Data Transfer

- CCMP (CTR + CBC-MAC Protocol)
  - Long term solution - mandatory for 802.11i compliance
  - Latest AES encryption
  - Requires hardware upgrades
- WRAP (Wireless Robust Authenticated Protocol)
  - Provided for early vendor support
- TKIP (Temporal Key Integrity Protocol)
  - Carried over from WPA

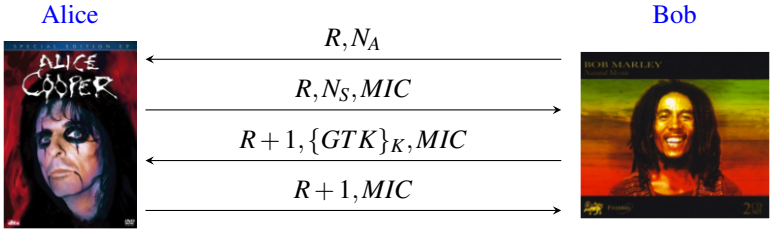
Additional Enhancements

- Pre-authentication (roaming clients)
- Client validation
- Password-to-key mappings
- Random number generation

802.11i

Key Reinstallation Attacks (KRACKS)

- Attack against the 4-way handshake of the WPA2 protocol.
- This handshake takes place between the client and the authentication server as follows:



- R = Replay Counter
- MIC = Message Integrity Check
- GTK = Group Temporal Key

802.11i

- In a key reinstallation attack, the attacker tricks a victim into reinstalling an already-in-use key.
- The client will install a PTK (Pairwise Transient Key) after receiving message 3; this is calculated from the PMK (Pairwise Master Key), N_A, N_S and the two MAC addresses.
- Once the key is installed, it will be used to encrypt normal data frames using an encryption protocol.
- The access point will retransmit message 3 if it did not receive an appropriate acknowledgment (message 4).
- Each time the client receives message 3, it will reinstall the same encryption key, and reset the incremental transmit packet number (nonce) and receive replay counter.
- An attacker can force these nonce resets by collecting and replaying transmissions of message 3.
- To guarantee security, a key should only be installed and used once; unfortunately this is not guaranteed by the WPA2 protocol.

Wireless LAN Security: Summary

- Basic 802.11b (with WEP)
  - Massive security holes
  - Easily attacked
- 802.1X
  - Good interim solution
  - Allows use of existing hardware
  - Can still be attacked
- Wi-Fi Protected Access
  - Allows use of existing hardware
  - Compatible with 802.1X
  - Compatible with 802.11i
- 802.11i
  - May require hardware upgrades
  - Can be attacked
  - Backwards compatible secure replacement is under development