Module 3: Cryptography III; Email Security

Nitesh Saxena

Overview of the Module

1.1 Hash Functions

1.2 Message Authentication Code

1.3 Key Distribution: Private Key Setting

1.4 Key Distribution: Public Key Setting

1.5 Email Security

Module 3, Lecture 1

Hash Functions

Cryptographic Hash Functions

- Requirements of cryptographic hash functions:
  - Can be applied to data of any length.
  - Output is fixed length, usually very short
  - Relatively easy to compute h(x), given x
  - Function is **deterministic**
  - Infeasible to get x, given h(x). **One-wayness property**
  - Infeasible to find any pair x and y (x ≠ y) such that h(x) = h(y). **Collision resistance property**

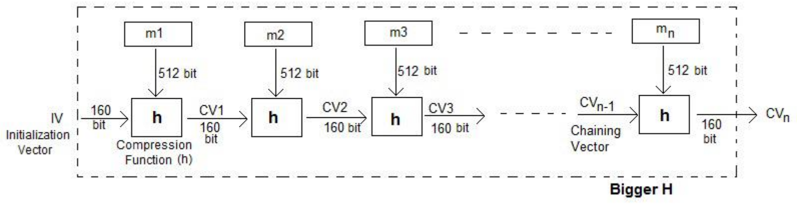
Some Applications of Hash Functions

- In general, can be used as a checksum for large amounts of data
- Password hashing
- Digital signatures
- Message authentication codes (will study in next lecture)
- Used also in RSA-OAEP, and many other cryptographic constructions

Hash Output Length

- How long should be the output (n bits) of a cryptographic hash function?
- To find collision - randomly select messages and check if hash matches any that we know.
- Throwing k balls in $N = 2^n$ bins. How large should k be, before probability of landing two balls in the same becomes greater than ½?
- ***Birthday paradox*** - a collision can be found in roughly $\text{sqrt}(N) = 2^{(n/2)}$ trials for an n bit hash
  - In a group of 23 (~ sqrt(365)) people, at least two of them will have the same birthday (with a probability > ½)
- Hence n should be at least 160

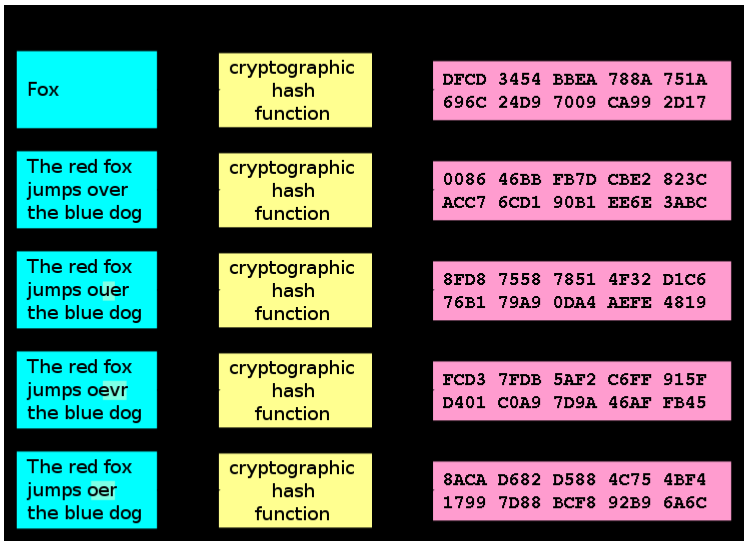
Generic Hash Function – Merkle-Damgard Construction



- This design for H() is collision-resistant given that h() is collision resistant
- Intuitively, this is because there is a avalanche effect – even if the inputs differ in just 1 bit, the outputs will be completely different
- IV is a known public constant

7

An Illustrative Example



Fox	cryptographic hash function	DFCD 3454 BBEA 788A 751A 696C 24D9 7009 CA99 2D17
The red fox jumps over the blue dog	cryptographic hash function	0086 46BB FB7D CBE2 823C ACC7 6CD1 90B1 EE6E 3ABC
The red fox jumps ouer the blue dog	cryptographic hash function	8FD8 7558 7851 4F32 D1C6 76B1 79A9 0DA4 AEFE 4819
The red fox jumps oevr the blue dog	cryptographic hash function	FCD3 7FDB 5AF2 C6FF 915F D401 C0A9 7D9A 46AF FB45
The red fox jumps oer the blue dog	cryptographic hash function	8ACA D682 D588 4C75 4BF4 1799 7D88 BCF8 92B9 6A6C

8

Practical Examples

- SHA-1
  - Output 160 bits
  - B'day attack requires 2^{80} calls
- MD5
  - Output is 128 bits
  - B'day attack requires 2^{64} calls only
- Better use stronger versions, such as SHA-256
  - B'day attack requires 2^{128} calls

Further Reading

- Stallings Chapter 3
- HAC Chapter 9

Module 3, Lecture 2

Message Authentication Codes

Message Authentication Codes

- Provide integrity as well as authentication
- Send (m, MAC); MAC is created on m using the key shared between two parties
- Has to be **deterministic** to enable verification
  - Unlike encryption schemes
- We want MAC to be as small and as secure as possible
- Can not provide non-repudiation
  - Why not?

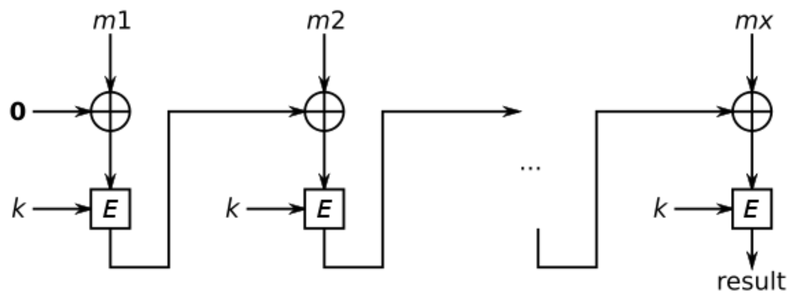
MAC – Functions

- KeyGen – outputs a key
- MAC – creates a checksum on m using key K
- Verify – validates whether the checksum on m is computed correctly
  - Just create MAC and compare

Security Notion for MAC

- Very similar to the security notion for a digital signature scheme
- Existential forgery under (adaptively) chosen message attack

MAC Based on Block Cipher in the CBC mode – CBC-MAC



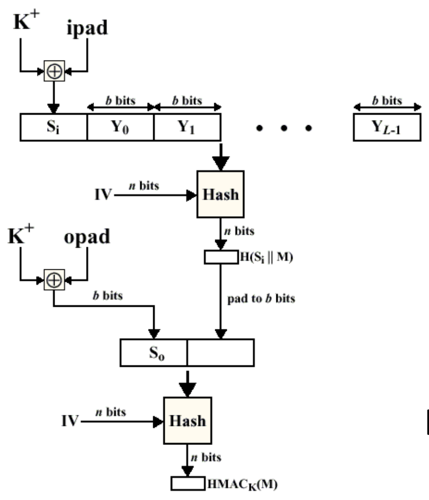
CBC-MAC

- Note that this is deterministic
  - IV = [0]
  - Unlike CBC encryption
- Only the last block of the output is used as a MAC
- This is secure under CMA attack
  - For pre-decided fixed-length messages
  - Intuitively because of the presence of an avalanche effect

HMAC: MAC using Hash Functions

- Developed as part of IPSEC - RFC 2104. Also used in SSL etc.
- Key based hash but almost as fast as non-key based hash functions.
- Avoids export restrictions unlike DES based MAC.
- Provable security
- Can be used with different hash functions like SHA-1,MD5, etc.

HMAC



Block size b bits.

K^+ - K padded with bits on the left to make b bits.

ipad – 0110110 (ox36) repeated b/8 times.

opad – 1011100 (0x5c) repeated b/8 times.

Essentially

$HMAC_K = H[(K^+ \text{ xor opad}) || H[(K^+ \text{ xor ipad}) || M]]$

Security of HMAC

- Security related to the collision resistance of the underlying hash function

Further Reading

- Stallings Chapter 3
- HAC Chapter 9

Module 3, Lecture 3

Key Distribution

(Private Key Setting)

Key Distribution

- Cryptographic primitives seen so far assume
  - In private key setting: Alice and Bob share a secret key which is unknown to Oscar.
  - In public key setting: Alice has a “trusted” (or authenticated) copy of Bob’s public key.
- But how does this happen in the first place?
- Alice and Bob meet and exchange key(s)
- Not always practical or possible.
- We need key distribution, first and foremost!
- Idea: make use of a trusted third party (TTP)

"Private Key" Distribution: An Attempt

- Protocol assumes that Alice and Bob share a session key K_A and K_B with a Key Distribution Center (KDC).
  - Alice calls Trent (Trusted KDC) and requests a session key to communicate with Bob.
  - Trent generates random session key K and sends $E_{K_A}(K)$ to Alice and $E_{K_B}(K)$ to Bob.
  - Alice and Bob decrypt with K_A and K_B respectively to get K.
- This is a key distribution protocol.
- Susceptible to replay attack!

Session Key Exchange with KDC – Needham-Schroeder Protocol

- A -> KDC $ID_A \,||\, ID_B \,||\, N_1$
  (Hello, I am Alice, I want to talk to Bob, I need a session Key and here is a random nonce identifying this request)
- KDC -> A $E_{K_A}(K \,||\, ID_B \,||\, N_1 \,||\, E_{K_B}(K \,||\, ID_A))$
  Encrypted(Here is a key, for you to talk to Bob as per your request N_1 and also an envelope to Bob containing the same key)
- A -> B $E_{K_B}(K \,||\, ID_A)$
  (I would like to talk using key in envelope sent by KDC)
- B -> A $E_K(N_2)$
  (OK Alice, But can you prove to me that you are indeed Alice and know the key?)
- A -> B $E_K(f(N_2))$ (Sure I can!)
- Dennig-Sacco (replay) attack on the protocol

Session Key Exchange with KDC – Needham-Schroeder Protocol (corrected version with mutual authentication)

- A -> KDC: $ID_A \mid\mid ID_B \mid\mid N_1$

  (Hello, I am Alice, I want to talk to Bob, I need a session Key and here is a random nonce identifying this request)

- KDC -> A: $E_{K_A}(K \mid\mid ID_B \mid\mid N_1 \mid\mid E_{K_B}(\underline{TS1}, K \mid\mid ID_A))$

  Encrypted(Here is a key, for you to talk to Bob as per your request N_1 and also an envelope to Bob containing the same key)

- A -> B: $E_K(\underline{TS2}), E_{K_B}(\underline{TS1}, K \mid\mid ID_A)$

  (I would like to talk using key in envelope sent by KDC; here is an authenticator)

- B -> A: $E_K(TS2+1)$

  (OK Alice, here is a proof that I am really Bob)

Module 3, Lecture 4

Key Distribution

(Public Key Setting)

Key Distribution

- Cryptographic primitives seen so far assume
  - In private key setting: Alice and Bob share a secret key which is unknown to Oscar.
  - **In public key setting: Alice has a "trusted" (or authenticated) copy of Bob's public key.**
- But how does this happen in the first place?
- Alice and Bob meet and exchange key(s)
- Not always practical or possible.
- We need key distribution, first and foremost!
- Idea: make use of a trusted third party (TTP)

Public Key Distribution

- Public announcements (such as email)
  - Can be forged
- Public directory
  - Can be tampered with
- Public-key certification authority (CA) (such as verisign)
  - This is what we use in practice
  - CA issues certificates to the users

Naming and Certificates

- Certification authority's vouch for the identity of an entity - *Distinguished Names (DN)*.
  /O=UAB/OU=CIS/CN=Nitesh Saxena
  - Although CN may be same, DN is different.

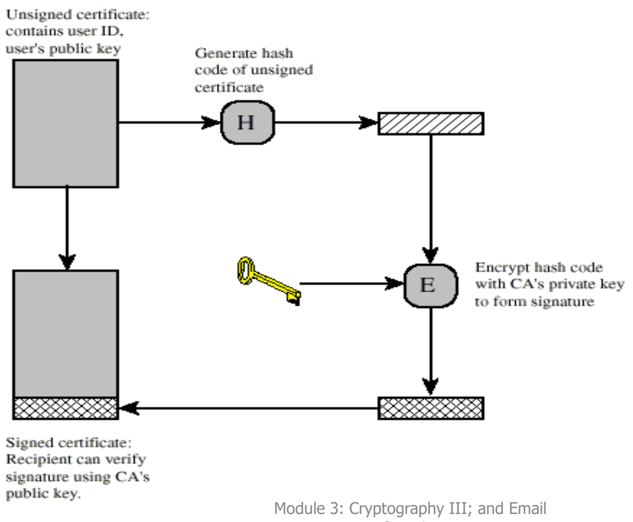
Types of Certificates

- CA's vouch at some level the identity of the principal.
- Example – Verisign:
  - Class 1 – Email address
  - Class 2 – Name and address verified through database.
  - Class 3- Background check.

Public Key Certificate

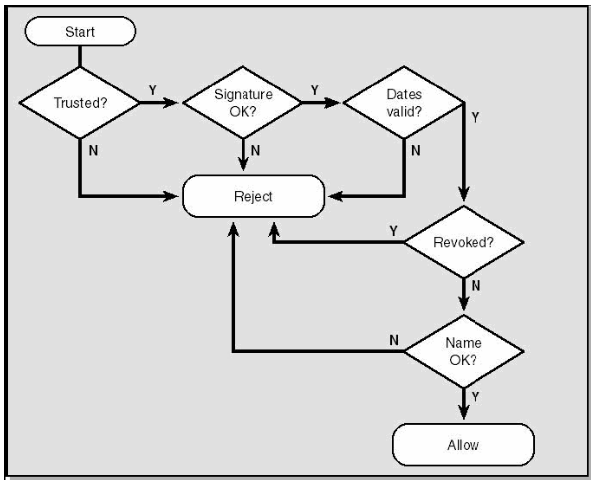
- *Public Key Certificate* – Signed messages specifying a name (identity) and the corresponding public key.
- Signed by whom – *Certification Authority* (CA), an organization that issues public key certificates.
- We assume that everyone is in possession of a trusted copy of the CA's public key.

Public Key Certificate


Unsigned certificate: contains user ID, user's public key
Generate hash code of unsigned certificate
H
Encrypt hash code with CA's private key to form signature
E
Signed certificate: Recipient can verify signature using CA's public key.

Note: Mechanism of certification and content of certificate, will vary but at the minimum we have email verification and contains ID and Public Key.

Certificate Verification/Validation



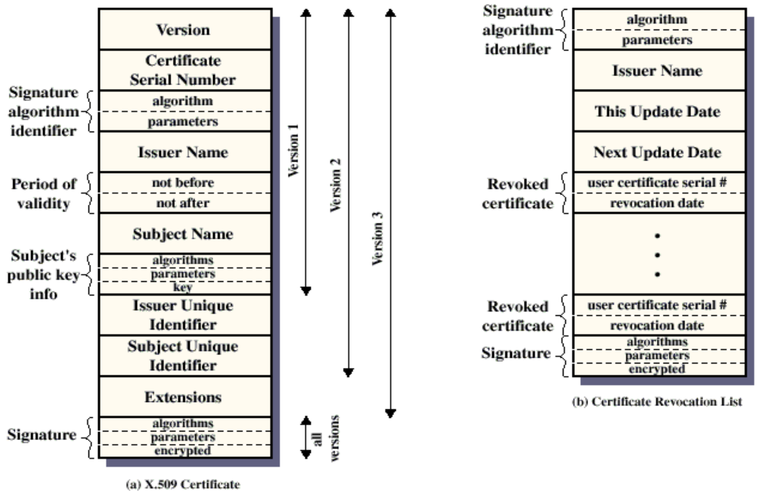
Certificate Revocation

- CA also needs some mechanism to *revoke* certificates
  - Private key compromised.
  - CA mistake in issuing certificate.
  - Particular service the certificate grants access to may no longer exist.
  - CA compromised.
- Expiration time solves the problems only partially.
- Certification Revocation Lists (CRL) – a list of every certificate that has been revoked but not expired.
  - CRL's quickly grow large!
- CRL's distributed periodically.
  - What about time period between revocation and distribution of CRL?
- Other mechanisms
  - OCSP (online certificate status protocol)

X.509

- Clearly, there is a need for standardization – X.509.
- Originally 1988, revised 93 and 95.
- X.509 is part of X.500 series that defines a directory service.
- Defines a framework for authentication services by X.500 directory to its users.
- Used in S/MIME, IPSEC, SSL etc.
- Does not dictate use of specific algorithm (recommends RSA).

X.509 Certificate



Advantages of CA Over KDC

- CA does not need to be on-line all the time!
- CA can be very simple computing device.
- If CA crashes, life goes on (except CRL).
- Certificates can be stored in an insecure manner!!
- Compromised CA cannot decrypt messages.
- Scales well.

Public-key Infrastructure (PKI)

- Combination of digital certificates, public-key cryptography, and certificate authorities.
- A typical enterprise's PKI encompasses
  - issuance of digital certificates to users and servers
  - end-user enrollment software
  - integration with corporate certificate directories
  - tools for managing, renewing, and revoking certificates; and related services and support
- Verisign, Thawte and Entrust – PKI providers.
- Your own PKI using Mozilla certificate servers

Further Reading

- Stallings Chapter 4
- HAC Chapter 12

Module 3, Lecture 5

Email Security via PGP

Email Security

- Email is one of the most widely used and regarded network services
- By default, email communication is NOT "secure"
  - may be inspected either in transit, or by suitably privileged users on destination system
  - may be impersonated/spoofed

Email Security Properties

- Confidentiality
  - protection from disclosure
- Authentication
  - of sender of message
- Message integrity
  - protection from modification
- Non-repudiation of origin
  - protection from denial by sender

Pretty Good Privacy (PGP)

- Open source, freely available software package for secure e-mail
- De facto standard for secure email
- Developed by Phil Zimmermann
- Selected best available crypto algorithms to use
- Runs on a variety of platforms like Unix, PC, Macintosh and other systems
- Originally free (now also have commercial versions available)

PGP Operation – Authentication

Just use digital signatures:

1. Sender creates message
2. Generates a digital signature for the message
3. Use SHA-1 to generate 160-bit hash of message
4. Signed hash with RSA using sender's private key, and is attached to message
5. Receiver uses RSA with sender's public key to decrypt and recover hash code
6. Receiver verifies received message using hash of it and compares with decrypted hash code

PGP Operation – Confidentiality

1. Sender generates a message
2. Generates a128-bit random number as session key
3. Encrypts the message using CAST-128 / IDEA / 3DES in CBC mode with session key
4. Session key encrypted using RSA with recipient's public key and attached to the msg
5. Receiver uses RSA with private key to decrypt and recover session key
6. Session key is used to decrypt message

PGP Operation – Confidentiality & Authentication

- Can use both services on the same message
  - create signature & attach it to the message
  - encrypt both message & signature
  - attach RSA encrypted session key

This sequence is preferred because

--one can store the plaintext message/file and its signature

--no need to store the ciphertext for future signature verification

PGP Operation – Compression

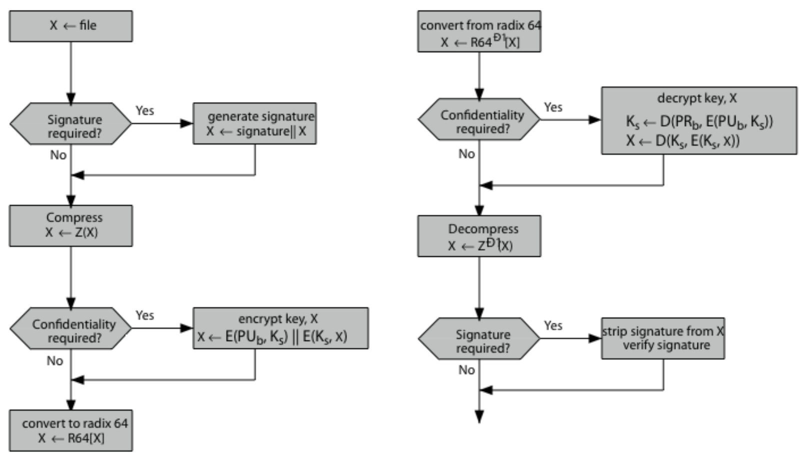
- PGP compresses messages to save space for e-mail transmission and storage
- By default, PGP compresses message after signing but before encrypting
  - so can store uncompressed message & signature for later verification
  - Encryption after compression strengthens security (because compression has less redundancy)
- uses ZIP compression algorithm

PGP Operation – Email Compatibility

- When using PGP will have binary data (8-bit octets) to send (encrypted message, etc)
- However, email was designed only for text
- Hence PGP must encode raw binary data into printable ASCII characters
- uses radix-64 algorithm
  - maps 3 bytes to 4 printable chars
  - also appends a CRC
- PGP also segments messages if too big

  (maximum length 50,000 octets)

PGP Operation – Summary


X ← file
Signature required?
Yes
No
generate signature X ← signature|| X
Compress X ← Z(X)
Confidentiality required?
Yes
No
encrypt key, X
X ← E(PU_b, K_s) || E(K_s, X)
convert to radix 64 X ← R64[X]
convert from radix 64 X ← $R64^{-1}$[X]
Confidentiality required?
Yes
No
decrypt key, X
K_s ← D(PR_b, E(PU_b, K_s))
X ← D(K_s, E(K_s, X))
Decompress X ← Z^{-1}(X)
Signature required?
Yes
No
strip signature from X verify signature

(a) Generic Transmission Diagram (from A) (b) Generic Reception Diagram (to B)

PGP Session Keys

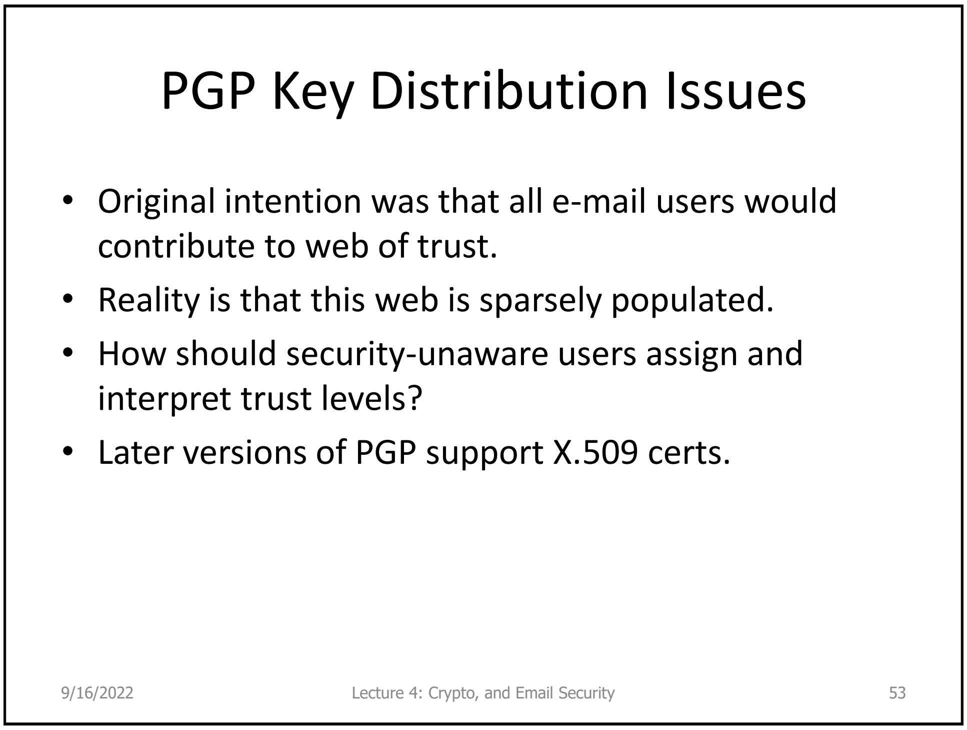
- Need a session key for each message
  - of varying sizes: 56-bit DES, 128-bit CAST or IDEA, 168-bit Triple-DES
- Uses random inputs taken from
  - actual keys hit
  - keystroke timing of a user
  - mouse movement

PGP Key Distribution

- Public keys for encrypting session keys / verifying signatures.
- Where do these keys come from and on what basis can they be trusted?

PGP Key Distribution

- PGP adopts a trust model called the *web of trust.*
- No centralized authority
- Individuals sign one another's public keys, these "certificates" are stored along with keys.
- PGP computes a *trust level* for each public key in key ring.
- Users interpret trust level for themselves.



PGP Key Distribution Issues

- Original intention was that all e-mail users would contribute to web of trust.
- Reality is that this web is sparsely populated.
- How should security-unaware users assign and interpret trust levels?
- Later versions of PGP support X.509 certs.