Course Admin

- HW#3 due at midnight coming Sunday (09/30)
- HW#4 will be posted by coming weekend
- Solutions will be posted soon

Regarding programming portions of the homework
- Submit the whole modified code that you used to measure timings
- Comment the portions in the code where you modified the code
  - Include a small “readme” for us to understand this
- If you did not submit the code for HW#2, do so now
  - Upload it on MyPoly
Outline of Today’s lecture

- Hash Functions
  - Properties
  - Known Hash Function
    - MD-5
    - SHA-1
  - Message Authentication using hash fns: HMAC
- “Private Key” Distribution
- “Public Key” Distribution: PKI
  - Certification
  - Revocation

Cryptographic Hash Functions: motivation

- Messages are large
- Public key encryption/decryption and signing/verification are costly
- Also, messages can be massaged??

- Why not compress/randomize the message first?
Cryptographic Hash Functions

- Requirements of cryptographic hash functions:
  - Can be applied to data of any length.
  - Output is fixed length.
  - Relatively easy to compute \( h(x) \), given \( x \).
  - Infeasible to get \( x \), given \( h(x) \). 
    One-wayness property
  - Given \( x \), infeasible to find \( y \) such that \( h(x) = h(y) \). 
    Weak collision property.
  - Infeasible to find any pair \( x \) and \( y \) such that \( h(x) = h(y) \). Strong collision property.

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 \( 1/2 \)?
- **Birthday paradox** - a collision can be found in roughly \( \sqrt{N} = 2^{n/2} \) trials for an \( n \) bit hash
  - In a group of 23 (\( \sim \sqrt{365} \)) people, at least two of them will have the same birthday (with a probability > \( 1/2 \))
- Hence \( n \) should be at least 160
Birthday Paradox

- Probability that hash values of k random messages are distinct is (that is, no collisions) is:

\[
\left(1 - \frac{1}{N}\right)\left(1 - \frac{2}{N}\right)\ldots\left(1 - \frac{k-1}{N}\right) = \prod_{i=1}^{k-1} \left(1 - \frac{i}{n}\right)
\]

\[
\approx \prod_{i=1}^{k-1} \left(e^{-i/n}\right) \quad \text{(as for small } x, \ e^{-x} \approx 1-x \text{ as } e^{-x} = 1 - x + \frac{x^2}{2!} - \frac{x^3}{3!} \ldots \)
\]

\[
e^{-\frac{k(k-1)}{2N}}
\]

So for at least one collision we have probability of

\[
\left(1 - e^{-\frac{k(k-1)}{2N}}\right)
\]

whose value is above 0.5 when \( k = 1.17\sqrt{N} \)

---

**Generic Hash Function**

**Construction**

- A hash function is typically based on internal compression function \( f() \) that works on fixed-size input blocks \( (M_i) \)

\[
\begin{array}{c}
\text{IV} \\
f \\
h_1 \\
f \\
h_2 \\
f \\
\ldots \\
h_{n-1} \\
f \\
h
\end{array}
\]

- Works sort of like a Chained Block Cipher

  - Produces a hash value for each fixed-size block based on its content and based on the hash value for the previous block
  - "Avalanche" effect (1-bit change in input produces "catastrophic" changes in output)

- In fact, can use symmetric encryption \( f=\text{E()} \), and use \( M \) as the key (but it won’t be fast)
Secure Hash Algorithm (SHA)

- SHA was published by NIST as a standard in 1993
- Revised in 1995 as SHA-1
  - Input: Up to 2^n bits
  - Output: 160-bit digest
  - 80-bit collision resistance
- Pad with at least 64 bits to resist padding attack
  - 1000...0 (message length)
- Processes 512-bit block
  - Initiate 5x32-bit MD registers
  - Apply compression function
    - 4 rounds of 20 steps each
    - each round uses different non-linear fi
    - registers are shifted and switched

Digest Generation with SHA-1
Basic Steps

Step 1: Padding
Step 2: Appending length as 64 bit unsigned
Step 3: Initialize MD buffer 5 32-bit words
   A|B|C|D|E
   A = 67452301
   B = efcdb89
   C = 98badcfe
   D = 10325476
   E = c3d2e1f0
Basic Steps...

Step 4: the 80-step processing of 512-bit blocks: 4 rounds, 20 steps each
Each step t (0 <= t <= 79):
- Input:
  - \( W_t \) - a 32-bit word from the message
  - \( K_t \) - a constant
  - \( ABCDE \): current MD
- Output:
  - \( ABCDE \): new MD

Basic Steps...

- Only 4 per-round distinctive additive constants
  
  \[0 \leq t \leq 19 \quad K_t = 5A827999\]
  
  \[20 \leq t \leq 39 \quad K_t = 6ED9EBA1\]
  
  \[40 \leq t \leq 59 \quad K_t = 8F1BCDC\]
  
  \[60 \leq t \leq 79 \quad K_t = CA62C1D6\]
Basic Steps - The Heart Of The Matter

Basic Logic Functions

- Only 3 different functions

<table>
<thead>
<tr>
<th>Round</th>
<th>Function $f_t(B,C,D)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 ≤ $t$ ≤ 19</td>
<td>$(B \land C) \lor (\neg B \land D)$</td>
</tr>
<tr>
<td>20 ≤ $t$ ≤ 39</td>
<td>$B \oplus C \oplus D$</td>
</tr>
<tr>
<td>40 ≤ $t$ ≤ 59</td>
<td>$(B \land C) \lor (B \land D) \lor (C \land D)$</td>
</tr>
<tr>
<td>60 ≤ $t$ ≤ 79</td>
<td>$B \oplus C \oplus D$</td>
</tr>
</tbody>
</table>
Other Hash Functions

- Many other hash functions
  - MD5 – Message Digest algorithm 5
    - Very similar to SHA – study on your own
  - RIPEM
  - MD4
  - MD6
  - Etc.

Twist With $W_t$'s

- Additional mixing used with input message 512-bit block
  - $W_0|W_1|...|W_{15} = m_0|m_1|m_2...|m_{15}$
  - For $15 < t < 80$:
    - $W_t = W_{t-16} \oplus W_{t-14} \oplus W_{t-8} \oplus W_{t-3}$

- XOR is a very efficient operation, but with multilevel shifting, it should produce very extensive and random mixing!
Current Security of MD5 and SHA-1

- SHA-1
  - B'day attack requires $2^{80}$ calls
  - Faster attacks $2^{69}$ calls
- MD5
  - Output is 128-bits, so B'day attack requires $2^{64}$ calls only
  - Faster attacks to find a collision:
- Better use stronger versions, such as SHA-256
- Although, these attacks are still not practical - they only find two random messages that collide

Message Authentication Codes

- Integrity as well as authentication
- (m, MAC)
- We want MAC to be as small and as secure as possible
- Security based on the length of the key and also how the MAC is computed
- A MAC can be constructed based on any good symmetric cipher - though this can be computationally expensive.
Recall MAC Using DES in CBC mode

Time = 1
\[
\text{DES Encrypt} \quad \text{(56 bits)} \\
O_1 \quad \text{(64 bits)}
\]

Time = 2
\[
\text{DES Encrypt} \\
\text{DES Encrypt} \\
O_2 \quad \text{(64 bits)}
\]

Time = N - 1
\[
\text{DES Encrypt} \\
\text{DES Encrypt} \\
O_{N-1} \quad \text{(64 bits)}
\]

Time = N
\[
\text{DES Encrypt} \\
\text{DES Encrypt} \\
O_N \quad \text{(64 bits)}
\]

DAC (16 to 64 bits)

Security notion for MAC

- Very similar to the security notion for a digital signature scheme
- Existential forgery under (adaptively) chosen message attack
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 (0x36) repeated $b/8$ times.
- $opad$ - 1011100 (0x5c) repeated $b/8$ times.
- Essentially
  \[
  HMAC_K = H((K^+ \ xor \ opad) || H((K^+ \ xor \ ipad) || M))
  \]
Security of HMAC

- Proven secure under assumptions stronger than that of being able to find collisions of the underlying hash function
  - Finding collisions even when the IV is secret and random
  - Computing the hash value even when the IV is secret and random
- See HMAC paper, if interested in details
  
  http://www-cse.ucsd.edu/~mihir/papers/hmac.html
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!

“Private Key” Distribution: attempt 1

- 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 || 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}(TS_1, 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(TS_2), E_{K_B}(TS_1, K || ID_A)
  (I would like to talk using key in envelope sent by KDC; here is an authenticator)
- B -> A: E_K(TS_2+1)
  (OK Alice, here is a proof that I am really Bob)
Kerberos - Goals

- Security
  - Next slide.
- Reliability
- Transparency
  - Minimum modification to existing network applications.
- Scalability
  - Modular distributed architecture.

Kerberos - Security Goals

- No cleartext passwords over network.
- No cleartext passwords stored on servers.
- Minimum exposure of client and server keys.
- Compromises should only affect current session.
- Require password only at login.
Kerberos - Assumptions

- Global clock.
- There is a way to distribute authorization data.
  - Kerberos provides authentication and not authorization.

Kerberos Key Distribution (1)

Step 1
Joe to KDC

I would like to
Talk to the File Server

Step 2
KDC

Session key for User

Session key for service
Kerberos Key Distribution (2)

Step 3
KDC

Box 1

Session Key for Joe

Dear Joe,
This key for File server

Box 2

Session Key for File server

Dear File server
This key for Use with Joe

KDC

Step 4
KDC to Joe

Box 1

Box 2

Joe

Kerberos Distribution (3)

Step 5
Joe

Opened Box 1

Dear Joe,
This key for File server

Box 2

Session Key for File server

Dear File server
This key for Use with Joe

Locked With File Server’s key

Step 6
Joe

Box 3

Locked With Session key

Dear File server,
The time is 3:40 pm

Box 2

Session Key for File server

Dear File server
This key for Use with Joe

Locked With File Server’s key
Kerberos Distribution (4)

Step 7: Joe to File server

Joe

Box 2

Box 3

File Server

Unlocked
Box 3

Unlocked
Box 2

Step 8: File server

Dear File server,
The time is 3:40 pm

Dear File server,
This key for Use with Joe

Kerberos Key Distribution (5)

- For mutual authentication, file server can create box 4 with time stamp and encrypt with session key and send to Joe.
- Box 2 is called ticket.
- KDC issues ticket only after authenticating password
- To avoid entering passwords every time access needed, KDC split into two - authenticating server and ticket granting server.
Kerberos – One Slide Overview

1. User logs on to workstation and requests service on host.
2. AS verifies user’s access right in database, creates ticket-granting ticket and session key. Results are encrypted using key derived from user’s password.
3. Workstation prompts user for password and uses password to decrypt ticket and authenticator that contains user’s name, network address, and time to TGS.
4. TGS decrypts ticket and authenticator, verifies request, then creates ticket for requested server.
5. Workstation sends ticket and authenticator to server.
6. Server verifies that ticket and authenticator match, then grants access to service. If mutual authentication is required, server returns an authenticator.

Version 4 summary

(a) Authentication Service Exchange: to obtain ticket-granting ticket

| 1 | C → AS: \[ \text{ID}_c \| \text{ID}_u \| \text{TS}_1 \] |
| 2 | AS → C: \[ \text{E}_{K_{AS}} [\text{ID}_u \| \text{ID}_d \| \text{TS}_2 \| \text{Lifetime} \| \text{Ticket}_2] \] |
| Ticket$_2$ = \[ \text{E}_{K_{AS}} [\text{ID}_u \| \text{ID}_d \| \text{TS}_2 \| \text{Lifetime}] \] |

(b) Ticket-Granting Service Exchange: to obtain service-granting ticket

| 3 | C → TGS: \[ \text{ID}_c \| \text{Ticket}_2 \| \text{Authenticator}_c \] |
| 4 | TGS → C: \[ \text{E}_{K_{TGS}} [\text{K}_{AS} \| \text{ID}_c \| \text{ID}_d \| \text{TS}_4 \| \text{Ticket}_c] \] |
| Ticket$_c$ = \[ \text{E}_{K_{TGS}} [\text{K}_{AS} \| \text{ID}_c \| \text{ID}_d \| \text{TS}_4 \| \text{TS}_5 \| \text{Lifetime}] \] |
| Authenticator$_c$ = \[ \text{E}_{K_{TGS}} [\text{ID}_c \| \text{ID}_d \| \text{TS}_5] \] |

(c) Client/Server Authentication Exchange: to obtain service

| 5 | C → K: \[ \text{Ticket}_c \| \text{Authenticator}_c \] |
| 6 | K → C: \[ \text{E}_{\text{K}_c} [\text{TS}_5 + 1] \] (for mutual authentication) |
| Ticket$_c$ = \[ \text{E}_{\text{K}_c} [\text{ID}_c \| \text{ID}_d \| \text{TS}_5 \| \text{Lifetime}] \] |
| Authenticator$_c$ = \[ \text{E}_{\text{K}_c} [\text{ID}_c \| \text{ID}_d \| \text{TS}_5] \] |
Kerberos - Limitations

- Every network service must be individually modified for use with Kerberos.
- Requires a global clock
- Requires secure Kerberos server.
- Requires continuously available or online server.

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
Identity for Files and Objects

- Files and other objects identified by “names”
  - File name – humans use.
  - File descriptor or handle – process use.
  - File allocation table entry – kernel use.
- Example
  - Unix – inodes, file descriptors, relative and absolute path names.
  - URL’s – Uniform Resource Locator.

Identity for Users

- User identity (UID)
  - Identity of a single entity
  - System represents user identity in different ways.
  - Not always human/physical entity.
- Example – Unix
  - Login name
  - UID – integer. UID 0 is root.
  - Logging done using login name.
- Same principal may have different identities.
  - Real and effective UID in UNIX. Used by SUID programs.
  - Saved UID – Free BSD and Solaris.
  - Audit or login UID – set at login and never changed. Allows one to track the original UID of a process.
Groups and Roles

- Users may need to share resources.
- Groups allow assignment of rights to multiple principals simultaneously.
- Group identity is static or can change.
- Example – UNIX
  - Each user assigned to one or more groups.
  - Each process has user id and group id.
- Role is a type of group that ties membership to function.
  - Sysadmin role, Backup role, webmaster role etc.
  - Allows finer grained control over access rights.
  - Mimics organizational structure of an enterprise.

Identity on the internet

- Host Identity
  - Related to network. Each network “layer” may use a different name. All names point to the same host but within different context.
  - Hostname, IP address, Ethernet (MAC) address.
  - Databases contain mappings between different names.
  - Can be spoofed. Mapping mechanism may not be secure.
- Static and dynamic identifiers.
  - DHCP, NAT. Local identifiers and global identifiers.
Naming and Certificates

- Certification authority's vouch for the identity of an entity - *Distinguished Names (DN)*.
  
  `/O=Polytechnic University/OU=CS/CN=John Doe`
  
  - Although CN may be same, DN is different.

- Policies of certification
  
  - *Authentication policy*
    
    What level of authentication is required to identify the principal.
  
  - *Issuance policy*
    
    Given the identity of principal will the CA issue a certificate?

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.
- CA could be
  - Internal CA.
  - Outsourced CA.
  - Trusted Third-Party CA.

**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)
- 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

(a) X.509 Certificate

(b) Certificate Revocation List
X.509 CA Hierarchy – Example.

Y<<X>> means the certificate of user X issued by CA Y.

To talk to B, A obtains the following chain
X<<W>>
W<<V>>
V<<Y>>
Y<<Z>>
Z<<B>>

Simpler if X has
X<<Z>>

Advantages of CA Over KDC

- CA does not need to be on-line!
- 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.
Internet Certificate Hierarchy

Types of certificates

- Organizational Certificates
  Principal’s affiliation with an organization

- Residential certificates
  Principal’s affiliation with an address

- Persona Certificates
  Principal’s Identity

- Principal need not be a person. It could be a role.
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 Netscape/Microsoft certificate servers

Problems with PKI – Private Key

- Where and how is private key stored?
  - Host – encrypted with pass phrase
  - Host – encrypted by OS or application
  - Smart Card
- Assumes secure host or tamper proof smartcard.
Problems with PKI - Conflicts

- X.509, PGP and IPRA remain silent on conflicts.
- They assume CA's and PCA's will ensure that no conflicts arise.
- But in practice conflicts may exist –
  - John A. Smith and John B. Smith may live at the same address.

Trustworthiness of Issuer

- A certificate is the binding of an external identity to a cryptographic key and a distinguished name. If the issuer can be fooled, all who rely upon the certificate can be fooled 😐
- How do you trust CA from country XYZ (your favorite prejudice).
Further Reading

- MIT Kerberos site:
- Kerberos RFC: RFC-1510
- X.509 page
- Ten Risks of PKI -

Some questions

- Schnorr signatures (SK=x in Z_q, PK=y=g^x mod p)
  - Signing
    - Choose random k in Z_q
    - Compute r = g^k mod p, Set c = H(m,r)
    - S = k + cx mod q
    - Output (m, r, s)
  - Verification? c = H(m,r) g^s = r.(y^c) mod p
- What is the length of
  - an RSA certificate?
  - a DSS certificate?
  - a Schnorr certificate?
Some questions

- Can a KDC learn communication between Alice and Bob, to whom it issued keys?
- Can a CA learn communication between Alice and Bob, to whom it issued certificates?
- What happens if the CA is online all the time?
- Alice uses her private key, public key pairs and a CA issued certificate. She learnt that Eve might have leaned her key. What should she do?

Some questions

- SHA-1 collisions can be found in $2^{69}$ operations. Does this mean that HMAC (using SHA-1) can be forged in $2^{69}$ operations too?
- DES CBC MAC is ------- than HMAC, computationally (for same key sizes)?
Sometimes when you access an https web-site, you get a security warning. What is that warning for?

Sometimes when you connect to an SSH server, you get a security warning. What is that warning for?

What is a self-signed certificate?

Computation time to MD-5 a 100 bytes long file is the same as for a 100MB file. Right?

Does DES use any modular arithmetic?

Can I use a DL-based key pairs, when the CA issuing me a certificate uses RSA keys?

Alice has only a DL-based key pairs, while Bob has only RSA keys. Can they ever be able to communicate securely?