Introduction to
Computer Security

Lecture 7
Digital Signature

October 9, 2003

Digital Signature

- Construct that authenticates origin, contents of message in a manner provable to a disinterested third party (“judge”)
- Sender cannot deny having sent message (service is “nonrepudiation”)
  - Limited to *technical* proofs
    - Inability to deny one’s cryptographic key was used to sign
  - One could claim the cryptographic key was stolen or compromised
    - Legal proofs, *etc.,* probably required;

Common Error

- Classical: Alice, Bob share key k
  - Alice sends $m \| \{ m \}k$ to Bob
  - Does this satisfy the requirement for message authentication? How?
  - Does this satisfy the requirement for a digital signature?
- **This is not a digital signature**
  - Why? Third party cannot determine whether Alice or Bob generated message

Classical Digital Signatures

- Require trusted third party
  - Alice, Bob each share keys with trusted party Cathy
- The judge must trust the trusted party Cathy

Alice $\xrightarrow{\{ m \}k_{Alice}}$ Bob

Bob $\xrightarrow{\{ m \}k_{Alice}}$ Cathy

Cathy $\xrightarrow{\{ m \}k_{Bob}}$ Bob

- To resolve dispute, judge gets $\{ m \}k_{Alice}$, $\{ m \}k_{Bob}$, and has Cathy decipher them; if messages matched, contract was signed, else one is a forgery

Public Key Digital Signatures (RSA)

- Alice's keys are d_{Alice}, e_{Alice}
- Alice sends Bob

$$m \,\|\, \{ m \}d_{Alice}$$

- In case of dispute, judge computes

$$\{ \{ m \}d_{Alice} \}e_{Alice}$$

- and if it is m, Alice signed message
  - She's the only one who knows d_{Alice}!

RSA Digital Signatures

- Use private key to encipher message
  - Protocol for use is *critical*
- Key points:
  - Never sign random documents, and when signing, always sign hash and never document
    - Mathematical properties can be turned against signer
  - Sign message first, then encipher
    - Changing public keys causes forgery

Attack #1

- Example: Alice, Bob communicating
  - $n_A = 95$, $e_A = 59$, $d_A = 11$
  - $n_B = 77$, $e_B = 53$, $d_B = 17$
- 26 contracts, numbered 00 to 25
  - Alice has Bob sign 05 and 17:
    - $c = m^{d_B} \bmod n_B = 05^{17} \bmod 77 = 3$
    - $c = m^{d_B} \bmod n_B = 17^{17} \bmod 77 = 19$
  - Alice computes $05 \times 17 \bmod 77 = 08$; corresponding signature is $03 \times 19 \bmod 77 = 57$; claims Bob signed 08
  - Note: $[(a \bmod n) \times (b \bmod n)] \bmod n = (a \times b) \bmod n$
  - Judge computes $c^{e_B} \bmod n_B = 57^{53} \bmod 77 = 08$
    - Signature validated; Bob is toast!

Attack #2: Bob's Revenge

- Bob, Alice agree to sign contract 06
- Alice enciphers, then signs:
  - Enciper: $c = m^{e_B} \bmod n_B = (06^{53} \bmod 77)^{11}$
  - Sign: $c^{d_A} \bmod n_A = (06^{53} \bmod 77)^{11} \bmod 95 = 63$
- Bob now changes his public key
  - Bob wants to claim that Alice singed N (13)
  - Computes r such that $13^r \bmod 77 = 6$; say, $r = 59$
  - Computes $r.e_B \bmod \varphi(n_B) = 59 \times 53 \bmod 60 = 7$
  - Replace public key e_B with 7, private key $d_B = 43$
- Bob claims contract was 13. Judge computes:
  - $(63^{59} \bmod 95)^{43} \bmod 77 = 13$
  - Verified; now Alice is toast
- Solution: sign first and then enciher!!

El Gamal Digital Signature

- Relies on discrete log problem
- Choose p prime, g, $d < p$;
- Compute $y = g^d \bmod p$
- Public key: (y, g, p); private key: d
- To sign contract m:
  - Choose k relatively prime to p–1, and not yet used
  - Compute $a = g^k \bmod p$
  - Find b such that $m = (da + kb) \bmod p–1$
  - Signature is (a, b)
- To validate, check that
  - $y^a a^b \bmod p = g^m \bmod p$

Example

- Alice chooses $p = 29$, $g = 3$, $d = 6$
  $y = 3^6 \bmod 29 = 4$
- Alice wants to send Bob signed contract 23
  - Chooses $k = 5$ (relatively prime to 28)
  - This gives $a = g^k \bmod p = 3^5 \bmod 29 = 11$
  - Then solving $23 = (6\times 11 + 5b) \bmod 28$ gives $b = 25$
  - Alice sends message 23 and signature (11, 25)
- Bob verifies signature: $g^m \bmod p = 3^{23} \bmod 29 = 8$ and $y^a a^b \bmod p = 4^{11} 11^{25} \bmod 29 = 8$
  - They match, so Alice signed

Attack

- Eve learns k, corresponding message m, and signature (a, b)
  - Extended Euclidean Algorithm gives d, the private key
- Example from above: Eve learned Alice signed last message with $k = 5$

$$m = (da + kb) \bmod p-1 = 23$$
$$= (11d + 5 \times 25) \bmod 28$$

So Alice's private key is $d = 6$

Kerberos

- Authentication system
  - Based on Needham-Schroeder with Denning-Sacco modification
  - Central server plays role of trusted third party ("Cathy")
- Ticket (credential)
  - Issuer vouches for identity of requester of service
- Authenticator
  - Identifies sender
- Alice must
  1. Authenticate herself to the system
  2. Obtain ticket to use server S

Overview

- User u authenticates to Kerberos server
  - Obtains ticket $T_{u,TGS}$ for ticket granting service (TGS)
- User u wants to use service s:
  - User sends authenticator A_u, ticket $T_{u,TGS}$ to TGS asking for ticket for service
  - TGS sends ticket $T_{u,s}$ to user
  - User sends A_u, $T_{u,s}$ to server as request to use s
- Details follow

Ticket

- Credential saying issuer has identified ticket requester
- Example ticket issued to user u for service s

$$T_{u,s} = s \,||\, \{ u \,||\, u\text{'s address} \,||\, \text{valid time} \,||\, k_{u,s} \} k_s$$

where:

- $k_{u,s}$ is session key for user and service
- Valid time is interval for which the ticket is valid
- u's address may be IP address or something else
  - Note: more fields, but not relevant here

Authenticator

- Credential containing identity of sender of ticket
  - Used to confirm sender is entity to which ticket was issued
- Example: authenticator user *u* generates for service *s*

$$A_{u,s} = \{ u \parallel \text{generation time} \parallel k_t \} k_{u,s}$$

where:

- k_t is alternate session key
- Generation time is when authenticator generated
  - Note: more fields, not relevant here

Protocol

user	$user \parallel TGS$	Cathy
Cathy	$\{ k_{u,TGS} \} k_u \parallel T_{u,TGS}$	*user*
user	$service \parallel A_{u,TGS} \parallel T_{u,TGS}$	TGS
user	$user \parallel \{ k_{u,s} \} k_{u,TGS} \parallel T_{u,s}$	TGS
user	$A_{u,s} \parallel T_{u,s}$	*service*
user	$\{ t + 1 \} k_{u,s}$	*service*

Analysis

- First two steps get user ticket to use TGS
  - User *u* can obtain session key only if *u* knows key shared with Cathy
- Next four steps show how *u* gets and uses ticket for service *s*
  - Service *s* validates request by checking sender (using $A_{u,s}$) is same as entity ticket issued to
  - Step 6 optional; used when *u* requests confirmation

Problems

- Relies on synchronized clocks
  - If not synchronized and old tickets, authenticators not cached, replay is possible
- Tickets have some fixed fields
  - Dictionary attacks possible
  - Kerberos 4 session keys weak (had much less than 56 bits of randomness); researchers at Purdue found them from tickets in minutes

Midterm

Midterm

- Midterm date: October 16, 2003
- Duration: 2:30 minutes
- Coverage: Material till today
- Closed Book: Yes

Roughly speaking

- Chapter 1, 2, 4: 20%
- Chapter 3: 20%
- Chapter 5, 6, 7: 35%
- Chapter 9 and 10: 25%

- May vary slightly!!

Chapter 1

- Understand the general concepts/issues
  - Components of security: confidentiality, integrity, availability, etc.
  - Threats
  - Policy vs. mechanisms
  - Assumptions of trust
  - Assurance
    - Specification/design/implementation
  - Operational issues
    - Cost-benefit; risk analysis; Human issues, etc.
  - Organizational problems
  - Security life cycle

Chapter 2

- Understand that access control matrix is an abstract model
- Understand the notation of state transitions
- Formal definitions of primitive commands
- Structure of conditional commands
- Principle of attenuation of privilege

Chapter 3

- Understand the working of Turing machine and the mapping
- Take-grant model
  - Understand the concepts well
    - Witness
    - Sharing
    - Stealing/conspiracy
  - No need to remember definitions (e.g., initial/terminal spans, bridges etc.)
- SPM model
  - Understand link/f, cc, cr functions well
  - Understand the examples well

Chapter 4

- Policy definitions
- Types of access control
- Policy language (Pandey & Hashii)
- Security and precision
  - Observability postulate
  - Secure and precise mechanism
  - Understand the definitions – no need to memorize (they will be provided if needed)

Chapter 5, 6 and 7

- Confidentiality: Bell-LaPadula model [5]
  - Security levels, categories, dominates relation
    - Not the formal model
- Integrity policies
  - Biba's integrity models
  - Lipner's integrity model
  - Clark-wilson model
- Hybrid policies
  - Chinese wall (informal)
  - Clinical and originator control (understand the basic requirements)
  - Role-based access control (NIST)

Chapter 9

- Classical crypto systems
  - Transposition ciphers
  - Substitution ciphers (caesar cipher)
  - Vigenere cipher
  - One-time pad
  - Data Encryption Standard (DES)
    - General working of DES
    - Cipher Block Chaining mode
  - Public-key
    - Diffie-hellman
    - RSA
  - Cryptographic checkcsum

Chapter 10

- Classical cryptographic key exchange and authentication
  - Basic protocol
  - Needham-Schroeder
  - Denning and Sacco
  - Otway-Rees protocol
  - Kerberos
  - Digital Signature