

Principles of Information Security CS 5323 Lecture 15

Prof. William Winsborough
November 11, 2008

Business

- Projects will be due Monday 12/15
- Questions from previous lectures?
- Slides 3-21 are ©2004 Matt Bishop

11 November 2008

Winsborough CS 5323 Lecture 15

2

Public Key Key Exchange

- Here interchange keys known
 - e_A, e_B Alice and Bob's public keys known to all
 - d_A, d_B Alice and Bob's private keys known only to owner
- Simple protocol
 - k_s is desired session key

Alice $\xrightarrow{\{k_s\} e_B}$ Bob

11 November 2008

Winsborough CS 5323 Lecture 15

3

Problem and Solution

- Vulnerable to forgery or replay
 - Because e_B known to anyone, Bob has no assurance that Alice sent message
- Simple fix uses Alice's private key
 - k_s is desired session key

Alice $\xrightarrow{\{\{k_s\} d_A\} e_B}$ Bob

11 November 2008

Winsborough CS 5323 Lecture 15

4

Notes

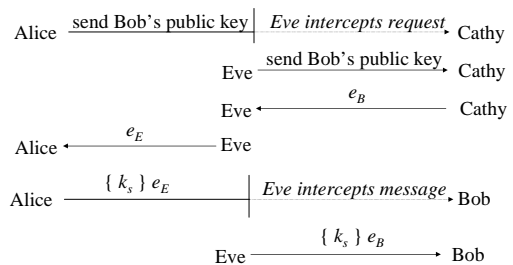
- Can include message enciphered with k_s
- Assumes Bob has Alice's public key, and *vice versa*
 - If not, each must get it from public server
 - If keys not bound to identity of owner, attacker Eve can launch a *man-in-the-middle* attack (next slide; Cathy is public server providing public keys)
 - Solution to this (binding identity to keys) discussed later as public key infrastructure (PKI)

11 November 2008

Winsborough CS 5323 Lecture 15

5

Man-in-the-Middle Attack



11 November 2008

Winsborough CS 5323 Lecture 15

6

Cryptographic Key Infrastructure

- Goal: bind identity to key
- Classical: not possible as all keys are shared
 - Use protocols to agree on a shared key (see earlier)
- Public key: bind identity to public key
 - Crucial as people will use key to communicate with principal whose identity is bound to key
 - Erroneous binding means no secrecy between principals
 - Assume principal identified by an acceptable name

11 November 2008

Winsborough CS 5323 Lecture 15

7

Certificates

- Create token (message) containing
 - Identity of principal (here, Alice)
 - Corresponding public key
 - Timestamp (when issued)
 - Other information (perhaps identity of signer)
- signed by trusted authority (here, Cathy)

$$C_A = \{ e_A \parallel \text{Alice} \parallel T \} d_C$$

11 November 2008

Winsborough CS 5323 Lecture 15

8

Use

- Bob gets Alice's certificate
 - If he knows Cathy's public key, he can decipher the certificate
 - When was certificate issued?
 - Is the principal Alice?
 - Now Bob has Alice's public key
- Problem: Bob needs Cathy's public key to validate certificate
 - Problem pushed "up" a level
 - Signature chains

11 November 2008

Winsborough CS 5323 Lecture 15

9

Certificate Signature Chains

- Create certificate
 - Generate hash of certificate
 - Encipher hash with issuer's private key
- Validate
 - Obtain issuer's public key
 - Decipher enciphered hash
 - Recompute hash from certificate and compare
- Problem: getting issuer's public key

11 November 2008

Winsborough CS 5323 Lecture 15

10

X.509 Chains

- Some certificate components in X.509v3:
 - Version
 - Serial number
 - Signature algorithm identifier: hash algorithm
 - Issuer's name; uniquely identifies issuer
 - Interval of validity
 - Subject's name; uniquely identifies subject
 - Subject's public key
 - Signature: enciphered hash

11 November 2008

Winsborough CS 5323 Lecture 15

11

X.509 Certificate Validation

- Obtain issuer's public key
 - The one for the particular signature algorithm
- Decipher signature
 - Gives hash of certificate
- Recompute hash from certificate and compare
 - If they differ, there's a problem
- Check interval of validity
 - This confirms that certificate is current

11 November 2008

Winsborough CS 5323 Lecture 15

12

Issuers

- **Certification Authority (CA):** entity that issues certificates
 - Multiple issuers pose validation problem
 - Alice's CA is Cathy; Bob's CA is Don; how can Alice validate Bob's certificate?
 - Have Cathy and Don cross-certify
 - Each issues certificate for the other

11 November 2008

Winsborough CS 5323 Lecture 15

13

Validation and Cross-Certifying

- **Certificates:**
 - Cathy<<Alice>>
 - Dan<<Bob>
 - Cathy<<Dan>>
 - Dan<<Cathy>>
- Alice validates Bob's certificate
 - Alice obtains Cathy<<Dan>>
 - Alice uses (known) public key of Cathy to validate Cathy<<Dan>>
 - Alice uses Cathy<<Dan>> to validate Dan<<Bob>>

11 November 2008

Winsborough CS 5323 Lecture 15

14

PGP Chains

- OpenPGP certificates structured into packets
 - One public key packet
 - Zero or more signature packets
- **Public key packet:**
 - Version (3 or 4; 3 compatible with all versions of PGP, 4 not compatible with older versions of PGP)
 - Creation time
 - Validity period (not present in version 3)
 - Public key algorithm, associated parameters
 - Public key

11 November 2008

Winsborough CS 5323 Lecture 15

15

OpenPGP Signature Packet

- **Version 3 signature packet**
 - Version (3)
 - Signature type (level of trust)
 - Creation time (when next fields hashed)
 - Signer's key identifier (identifies key to encipher hash)
 - Public key algorithm (used to encipher hash)
 - Hash algorithm
 - Part of signed hash (used for quick check)
 - Signature (enciphered hash)
- **Version 4 packet more complex**

11 November 2008

Winsborough CS 5323 Lecture 15

16

Signing

- Single certificate may have multiple signatures
- Notion of "trust" embedded in each signature
 - Range from "untrusted" to "ultimate trust"
 - Signer defines meaning of trust level (no standards!)
- All version 4 keys signed by subject
 - Called "self-signing"

11 November 2008

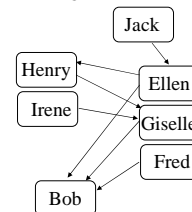
Winsborough CS 5323 Lecture 15

17

Validating Certificates

- Alice needs to validate Bob's OpenPGP cert
 - Does not know Fred, Giselle, or Ellen
- Alice gets Giselle's cert
 - Knows Henry slightly, but his signature is at "casual" level of trust
- Alice gets Ellen's cert
 - Knows Jack, so uses his cert to validate Ellen's, then hers to validate Bob's

Arrows show signatures
Self signatures not shown



11 November 2008

Winsborough CS 5323 Lecture 15

18

Storing Keys

- Multi-user or networked systems: attackers may defeat access control mechanisms
 - Encipher file containing key
 - Attacker can monitor keystrokes to decipher files
 - Key will be resident in memory that attacker may be able to read
 - Use physical devices like “smart card”
 - Key never enters system
 - Card can be stolen, so have 2 devices combine bits to make single key

11 November 2008

Winsborough CS 5323 Lecture 15

19

Key Revocation

- Certificates invalidated *before* expiration
 - Usually due to compromised key
 - May be due to change in circumstance (e.g., someone leaving company)
- Problems
 - Entity revoking certificate authorized to do so
 - Revocation information circulates to everyone fast enough
 - Network delays, infrastructure problems may delay information

11 November 2008

Winsborough CS 5323 Lecture 15

20

CRLs

- *Certificate revocation list* lists certificates that are revoked
- X.509: only certificate issuer can revoke certificate
 - Added to CRL
- PGP: signers can revoke signatures; owners can revoke certificates, or allow others to do so
 - Revocation message placed in PGP packet and signed
 - Flag marks it as revocation message

11 November 2008

Winsborough CS 5323 Lecture 15

21

Language for Policy and Credentials

- Pubs
 - Design of a Role-Based Trust Management Framework. Ninghui Li, John C. Mitchell, and William H. Winsborough. *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, May 2002
- Outline
 - Requirements
 - Examples
 - Syntax
 - Semantics
 - Language extensions

11 November 2008

Winsborough CS 5323 Lecture 15

22

Policy Language Wish List

- Decentralize authority to define attributes
 - Utilize policy and credentials from many sources
- Delegation of attribute authority
 - To specific principals
 - To principals with certain attributes
- Inference of attributes
 - E.g., derive access rights based on roles or other characteristics
- Intersection of attributes
- Parameterization
- Support for thresholds, separation of duty

11 November 2008

Winsborough CS 5323 Lecture 15

23

Role-based Trust Management (*RT*)

- A family of credential / policy languages
 - Simplest, RT_0 , has no parameterization, thresholds, or separation of duty
- RT_0 example: student discount subscription
 - $\text{EPub.studentDiscount} \leftarrow \text{StateU.student}$
 - $\text{StateU.student} \leftarrow \text{URegistrar.fulltimeLoad}$
 - $\text{StateU.student} \leftarrow \text{URegistrar.parttimeLoad}$
 - $\text{URegistrar.parttimeLoad} \leftarrow \text{Alice}$

11 November 2008

Winsborough CS 5323 Lecture 15

24

Role-based Trust Management (RT)

- A family of credential / policy languages
 - Simplest, RT_0 , has no parameterization, thresholds, or separation of duty
- RT_0 example: student discount subscription
 - $Epub.studentDiscount \leftarrow StateU.student$
 - $StateU.student \leftarrow URegistrar.fulltimeLoad$
 - $StateU.student \leftarrow URegistrar.parttimeLoad$
 - $URegistrar.parttimeLoad \leftarrow Alice$
- Credential chain proves authorization

11 November 2008

Winsborough CS 5323 Lecture 15

25

Example: Attribute-based Delegation

- Accepting student ID from **any** university
 - $Epub.studentDiscount \leftarrow FAB.accredited.student$
 - $FAB.accredited \leftarrow StateU$
 - $StateU.student \leftarrow URegistrar.fulltimeLoad$
 - $StateU.student \leftarrow URegistrar.parttimeLoad$
 - $URegistrar.parttimeLoad \leftarrow Alice$

11 November 2008

Winsborough CS 5323 Lecture 15

26

Example: Expressivity in Credentials

- Deferring a Guaranteed Student Loan
 - $BankWon.deferGSL \leftarrow FAB.accredited.fulltimeStudent$
 - $FAB.accredited \leftarrow StateU$
 - $StateU.fulltimeStudent \leftarrow URegistrar.fulltimeLoad$
 - $StateU.fulltimeStudent \leftarrow URegistrar.parttimeLoad \cap StateU.gradOfficer.phdCandidate$
 - $URegistrar.parttimeLoad \leftarrow Bob$
 - $StateU.gradOfficer \leftarrow Carol$
 - $Carol.phdCandidate \leftarrow Bob$

11 November 2008

Winsborough CS 5323 Lecture 15

27

RT_0 Syntax

- Basic structure is a role (i.e., an attribute): $A.r$
 - A is a principal (authority for $A.r$), r is a role name
- Four types of policy statement
 - $A.r \leftarrow D$
Role $A.r$ contains principal D as a member
 - $A.r \leftarrow B.r_1$
 $A.r$ contains role $B.r_1$ as a subset
 - $A.r \leftarrow A.r_1.r_2$
 $A.r$ contains $B.r_2$ as a subset, for each B in $A.r_1$
 - $A.r \leftarrow A_1.r_1 \cap A_2.r_2$
 $A.r$ contains the intersection
- A credential is a statement signed by A , the credential issuer and the authority over $A.r$
- The first 3 statement types give a language equivalent to pure SDSI

11 November 2008

Winsborough CS 5323 Lecture 15

28

A Brief Intro to Logic Programming

- A program P is a set of clauses:
 - $h(t_i) :- b_1(t_1), \dots, b_n(t_n)$ where h and b_i are predicates and t_i are tuples of logical terms
 - $:-$ is read "if"
 - $p(c, ?X) :- q(b, ?Z), r(?Z, ?X).$
 - $q(b, a).$
 - $r(a, d).$
- Basically, relational database + rules for deriving relations
- A query Q has the form $?- b_1(t_1), \dots, b_n(t_n)$
 - $?- p(?U, ?V).$
- An answer is an instance Q' of the query Q that is logically entailed by the program
($\mathcal{P} \models Q'$), e.g., $Q' = p(c, d).$

11 November 2008

Winsborough CS 5323 Lecture 15

29

Benefits of LP Semantics

- Makes complexity results easy
- Facilitates extending RT_0
 - Parameters, thresholds, sep. of duty
 - Other semantic foundations do not easily support important extensions
 - String rewriting [Clarke et al., JCS 2001]
 - Sets provide a good intuition
 - A role is the set of principals in the role
 - Parameterization requires generalization

11 November 2008

Winsborough CS 5323 Lecture 15

30

$SP(\varphi)$: A Logic-Programming Semantics for RT_0 policy φ

- Translate each statement of φ to a clause:
 - For each $A.r \leftarrow D$ in φ , add $m(A, r, D)$.
 - For each $A.r \leftarrow B.r_1$ in φ , add $m(A, r, ?X) :- m(B, r_1, ?X)$.
 - For each $A.r \leftarrow A.r_1.r_2$ in φ , add $m(A, r, ?X) :- m(A, r_1, ?Y), m(?Y, r_2, ?X)$.
 - For each $A.r \leftarrow A_1.r_1 \cap A_2.r_2$ in φ , add $m(A, r, ?X) :- m(A_1, r_1, ?X), m(A_2, r_2, ?X)$.
- D is in $A.r$ if $SP(\varphi) \models m(A, r, D)$
- Pose queries like $?- m(A, r, D)$, $?- m(A, r, ?X)$, and $?- m(?X, ?u, D)$

11 November 2008

Winsborough CS 5323 Lecture 15

31

Globally Unique Role Names

- Application Domain Specification Document (ADSD)
 - Declares a collection of related role names
 - Unique name space for each ADSD
 - Role names declared in different ADSDs are different
 - They refer to the URI of the ADSD in which they are declared
- In RT_1 , where roles are parameterized, ADSD also gives type signature

11 November 2008

Winsborough CS 5323 Lecture 15

32

RT_1 : Adding Role Parameters

- Roles have the form $A.R = A.r(h_1, \dots, h_n)$
- Each h_i is a data term whose type is that declared for r 's i^{th} parameter in the ADSD
- Example:
 - `BigCorp.evaluatorOf(?Y) ← BigCorp.managerOf(?Y)`
 - `BigCorp.raise ← BigCorp.evaluatorOf(this).exceedsExpectations`

11 November 2008

Winsborough CS 5323 Lecture 15

33

Parameterization: Semantics and Complexity

- LP semantics is straightforward:
 - E.g., $A.r(h_1, \dots, h_n) \leftarrow B.r_1(s_1, \dots, s_m)$ translates to $m(A, r, h_1, \dots, h_n, ?X) :- m(B, r_1, s_1, \dots, s_m, ?X)$
Alternatively: $r(A, h_1, \dots, h_n, ?X) :- r_1(B, s_1, \dots, s_m, ?X)$
- Apply known complexity results: The atomic implications of $SP(\varphi)$ can be computed in $O(N^{v+3})$
 - v is the max number of variables per statement
 - Each role name has at most p arguments
 - $N = \max(N_0, pN_0)$
 - N_0 is the number of statements in φ

11 November 2008

Winsborough CS 5323 Lecture 15

34