

Principles of Information Security CS 5323 Lecture 16

Prof. William Winsborough
November 13, 2008

Business

- Projects (papers and implementations) will be due Monday 12/15
 - Presentations will take place 11/25, 12/2, and 12/4
 - 25 minutes per speaker
- Questions from previous lectures?
- Slides 3-10 are ©2004 Matt Bishop

13 November 2008

Winsborough CS 5323 Lecture 16

2

Digital Signature

- Construct that authenticated 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; not dealt with here

13 November 2008

Winsborough CS 5323 Lecture 16

3

Common Error

- Classical: Alice, Bob share key k
 - Alice sends $m || \{ m \} k$ to Bob
- This is a digital signature
- WRONG**
- This is not a digital signature**
- Why? Third party cannot determine whether Alice or Bob generated message

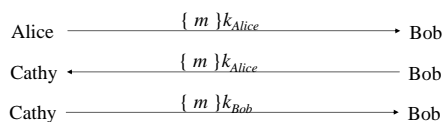
13 November 2008

Winsborough CS 5323 Lecture 16

4

Classical Digital Signatures

- Require trusted third party
 - Alice, Bob each share keys with trusted party Cathy
- To resolve dispute, judge gets $\{ m \} k_{Alice}$, $\{ m \} k_{Bob}$, and has Cathy decipher them; if messages matched, contract was signed



13 November 2008

Winsborough CS 5323 Lecture 16

5

Public Key Digital Signatures

- 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} !

13 November 2008

Winsborough CS 5323 Lecture 16

6

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

13 November 2008

Winsborough CS 5323 Lecture 16

7

Recall RSA Algorithm

- Choose two large prime numbers p, q
 - Let $n = pq$; then $\phi(n) = (p-1)(q-1)$ (totient function)
 - Choose $e < n$ such that e is relatively prime to $\phi(n)$.
 - Compute d such that $ed \bmod \phi(n) = 1$
- Public key: (e, n) ; private key: d
- Encipher: $c = m^e \bmod n$
- Decipher: $m = c^d \bmod n$

13 November 2008

Winsborough CS 5323 Lecture 16

8

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:
 - $c = m^{e_B} \bmod n_B = 05^{17} \bmod 77 = 3$
 - $c = m^{e_B} \bmod n_B = 17^{17} \bmod 77 = 19$
 - Alice computes $05 \times 17 \bmod 77 = 08$; corresponding signature is $03 \times 19 \bmod 77 = 57$; claims Bob signed 08
 - Judge computes $c^{e_B} \bmod n_B = 57^{53} \bmod 77 = 08$
 - Signature validated; Bob is toast

13 November 2008

Winsborough CS 5323 Lecture 16

9

Attack #2: Bob's Revenge

- Bob, Alice agree to sign contract 06
- Alice enciphers, then signs:
 $(m^{e_B} \bmod 77)^{d_A} \bmod n_A = (06^{53} \bmod 77)^{11} \bmod 95 = 63$
- Bob now changes his public key
 - Computes r such that $13^r \bmod 77 = 6$; say, $r = 59$
 - Computes $re_B \bmod \phi(n_B) = 59 \times 53 \bmod 60 = 7$
 - 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

13 November 2008

Winsborough CS 5323 Lecture 16

10

Key Points

- Key management critical to effective use of cryptosystems
 - Different levels of keys (session vs. interchange)
- Keys need infrastructure to identify holders, allow revoking
 - Key escrowing complicates infrastructure
- Digital signatures provide integrity of origin and content
 - Much easier with public key cryptosystems than with classical cryptosystems

13 November 2008

Winsborough CS 5323 Lecture 16

11

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

13 November 2008

Winsborough CS 5323 Lecture 16

12

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

13 November 2008

Winsborough CS 5323 Lecture 16

13

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

13 November 2008

Winsborough CS 5323 Lecture 16

14

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

13 November 2008

Winsborough CS 5323 Lecture 16

15

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

13 November 2008

Winsborough CS 5323 Lecture 16

16

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

13 November 2008

Winsborough CS 5323 Lecture 16

17

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.r₁
A.r contains role B.r₁ as a subset
 - A.r \leftarrow A.r₁.r₂
A.r contains B.r₂ as a subset, for each B in A.r₁
 - A.r \leftarrow A₁.r₁ \cap A₂.r₂
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

13 November 2008

Winsborough CS 5323 Lecture 16

18

A Brief Intro to Logic Programming

- A program P is a set of clauses:
 - $h(\mathbf{t}_h) :- b_1(\mathbf{t}_1), \dots, b_n(\mathbf{t}_n)$ where h and b_i are predicates and \mathbf{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 $?- b_1(\mathbf{t}_1), \dots, b_n(\mathbf{t}_n)$
 - $?- p(?U, ?V).$
- An answer is an instance \mathcal{Q} of the query Q that is logically entailed by the program ($\mathcal{P} \models \mathcal{Q}$), e.g., $\mathcal{Q} = p(c, d).$

13 November 2008

Winsborough CS 5323 Lecture 16

19

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

13 November 2008

Winsborough CS 5323 Lecture 16

20

$SP(\mathcal{P})$: A Logic-Programming Semantics for RT_0 policy \mathcal{P}

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

13 November 2008

Winsborough CS 5323 Lecture 16

21

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

13 November 2008

Winsborough CS 5323 Lecture 16

22

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`

13 November 2008

Winsborough CS 5323 Lecture 16

23

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 $SP(\mathcal{P})$ 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 \mathcal{P}

13 November 2008

Winsborough CS 5323 Lecture 16

24