

Security & Privacy

Larry Rudolph



1

Pervasive Computing MIT 6.883 SMA 5508 Spring 2006 Larry Rudolph

Who cares about Privacy?

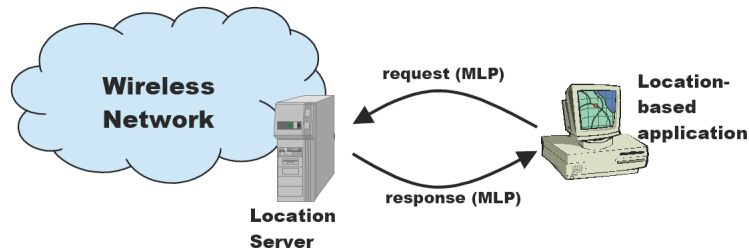
- ➡ Everybody? Nobody? Criminals? Governments? Corporations?
- ➡ Privacy is the right to keep information hidden. But there is lots of information.
 - ➡ You do not have to use stuff that can be tracked, or do you?
- ➡ We in academia can have real impact here



Pervasive Computing MIT 6.883 SMA 5508 Spring 2006 Larry Rudolph

Do we have privacy?

- ▶ Mobile Location Protocol
- ▶ another standard



- ▶ Used to keep track of your mobile phone
- ▶ In case you make an emergency call

Who cares about Security?

- * Today, nearly everyone
 - * Limit discussion to information security
- * Are you allowed to use this device?
 - * authorization
- * Are you who you say you are?
 - * authentication
- * Is this device what you think it is?
 - * authentication



Anonymity

- ➡ I am having a conversation with Alice
 - ➡ Do you know we are talking?
 - ➡ Do you know what we are talking about?
 - ➡ Do you know who we are?
- ➡ Need indirection
 - ➡ but also many-to-one (so things get lost)
 - ➡ think about publish/subscribe
 - ➡ think about multiple personalities
 - ➡ multiple credit cards, cell phones, BlueTooth IDs
 - ➡ rent personality every 5 min from trusted server



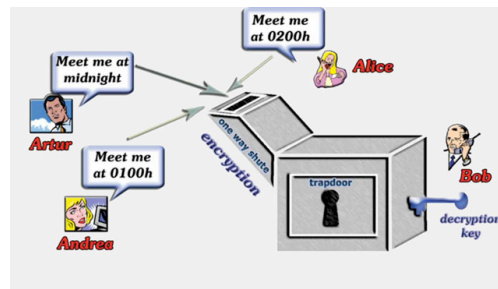
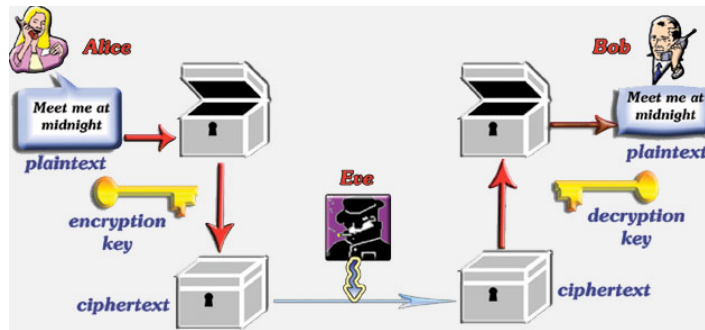
Pervasive Computing MIT 6.883 SMA 5508 Spring 2006 Larry Rudolph

Where security?



- ▶ Communicate thru insecure area
 - ▶ Over internet
 - ▶ Through air (bluetooth, 802.11, ..)
- ▶ USB keyboard, Monitor via cable
 - ▶ Assumed to be secure
- ▶ Shared resources or devices
- ▶ RFID Tags -- very insecure
 - ▶ stand next to you and listen to your cards response and replicate it later
- ▶ Want active RFID tags
 - ▶ use radio power to drive computation

Public Key Crypto-system



MIT 6.883 SMA 5508 Spring 2006 Larry Rudolph

Public Key Crypto-system

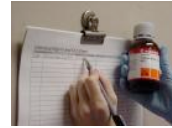
- * Very quick overview
 - * K_e = Public Key, K_d = Private Key
 - * Encrypt message, $E = \text{Encrypt}(M, K_e)$
 - * Decrypt message, $M = \text{Decrypt}(E, K_d)$
 - where $M = \text{Decrypt}(\text{Encrypt}(M, K_e), K_d)$
 - and $M = \text{Encrypt}(\text{Decrypt}(M, K_d), K_e)$
- * Given K_e , M , $\text{Encrypt}(M, K_e)$
- * cannot easily compute K_d .



Pervasive Computing MIT 6.883 SMA 5508 Spring 2006 Larry Rudolph

Signing Messages

- ➡ Alice & Bob have keys A_1, A_2, B_1, B_2
- ➡ Alice sends message M to Bob
 - ➡ $\text{Encrypt}(M \ \& \ \text{Encrypt}(\text{Alice}, A_2), B_1)$
- ➡ Bob decrypts message M using B_1 and then uses Alice's public key, A_1 to decrypt the name **Alice**.
- ➡ Does Bob know that Alice sent the msg M ?



Pervasive Computing MIT 6.883 SMA 5508 Spring 2006 Larry Rudolph

Certification Authority

- ▶ No, it all depends on having correct public keys.
 - ▶ How did Alice know that B_1 is Bob's public key?
- ▶ Use a certification authority:
 - ▶ some trusted site that associates keys with names.
- ▶ Hierarchy of CA's



Pervasive Computing MIT 6.883 SMA 5508 Spring 2006 Larry Rudolph

Point to point communication

- * Public Key scheme: many people can send messages to Alice
- * But basically a one-to-one protocol:
 - * With signing and with replies
- * Not well suited for pervasive computing
 - * Environment filled with devices



Pervasive Computing MIT 6.883 SMA 5508 Spring 2006 Larry Rudolph

Want group keys

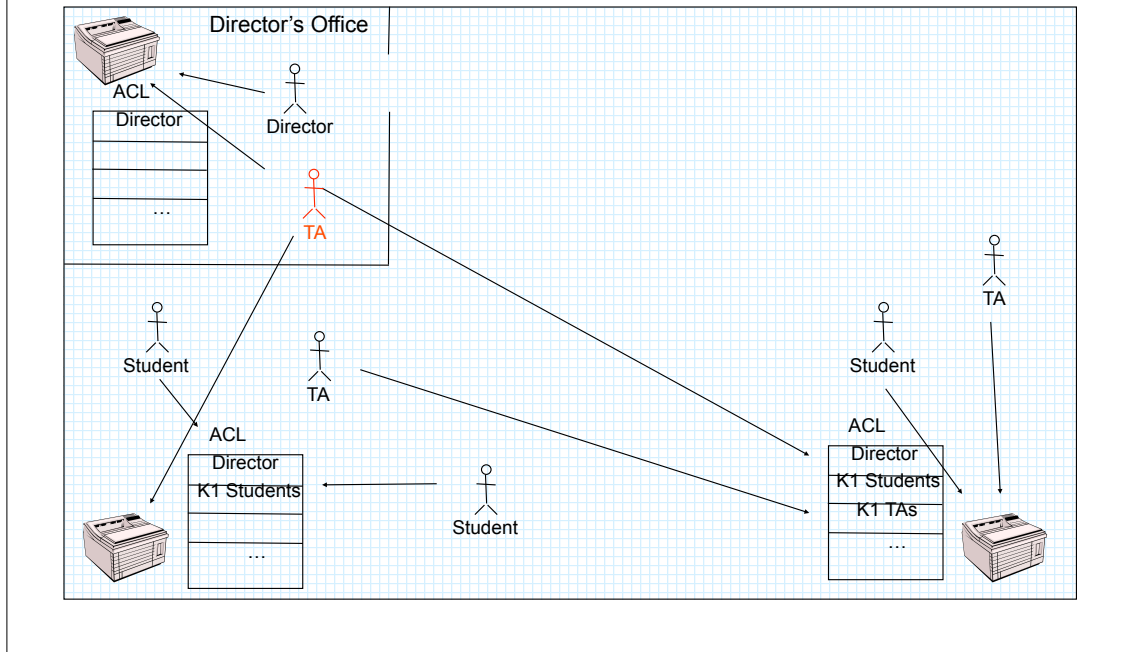


- ➔ Want all students in SMA 5508 and 6.883 to access course web site
- ➔ Want all SMA students to access SMA's main site, etc.
- ➔ A person belongs to many groups
- ➔ Grant access based on group
- ➔ Add/remove people from group



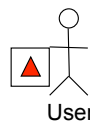
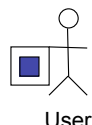
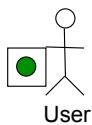
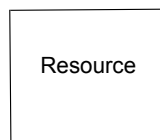
Pervasive Computing MIT 6.883 SMA 5508 Spring 2006 Larry Rudolph

Usage Scenario



Access Control

- ➡ Security Model
- ➡ Useful mechanism in guarding access to resources
- ➡ Suitable for dynamic environments
- ➡ Each resource maintains a list referencing a set of valid keys
 - ➡ Granting, delegating, revoking access
 - ➡ user/application does not know accessibility of resource without explicitly attempting access





SPKI/SDSI Introduction

(Simple Public-Key Infrastructure/Simple Distributed Security Infrastructure)

- ▶ A group key infrastructure
- ▶ Build secure, scalable distributed computing systems
- ▶ Fine-grained access control over an untrusted network



SPKI/SDSI Introduction

(Simple Public-Key Infrastructure/Simple Distributed Security Infrastructure)

- * Designed by Ron Rivest, Butler Lampson and Carl Ellison
- * Each public key is a CA
- * Name certificate: defines a name in issuer's name space
- * Authorization certificate: grants a specific authorization from issuer to subject



SPKI/SDSI: Name Certificates

Traditional:

$\{\text{MIT Larry, } K_L\}_{K_{\text{mit}}}$

- Local name spaces
- Groups

SPKI/SDSI:

$\{K_C \text{ friends, } K_d\}_K$

$\{K_C \text{ friends, } K_e\}_K$

$\{K_C \text{ friends, } K_f^c\}_K$

If 'K_C friends' is on an ACL, K_d, K_e and K_f are allowed to access the object.



Pervasive Computing MIT 6.883 SMA 5508 Spring 2006 Larry Rudolph

SPKI/SDSI: Name Certificate

```
(cert
  (issuer
    (name
      (public-key
        (rsa-pkcs1-md5
          (e #23#)
          (n
            |AMMgMuKpqK13pHMhC8kuxaSeCo+yt8TadcgnG8bEo+erdrSBveY3C
            MBkkZqrM0St4KkmMuHMxhsp5FX71XBiVW1+JGCBLfI7hxWDZCxGTMg
            bR4Fk+ctyUxIv3CQ93uYVkg9ca6awCxtS0EI7sLuEB+HKuOLjzTsH+
            +Txw9NAHq4r|)))
      friends))
  (subject
    (public-key
      (rsa-pkcs1-md5
        (e #23#)
        (n
          |AKg3tOzoJ5PGQ5q9jzxxzwxE8o6bIZ6/cE8gEL+1xJa23viE3bz68ru
          hpD5muqJ+uyDCN*xgAZ0JVXJazmX1QjiGudj9kEmuni8gJRLZRu0T5E3
          K7OU2dodu0kdDg32kym7+ooZNe/F0zWGekfESeezyQ25kvNO3XQvMHX
          afWcYjRw|))))))
```

SPKI/SDSI: Authorization Model



- * Simple trust policy model
- * Authorizations specified in flexible, user-defined tags
- * Authorizations can be defined as specific or as general as desired
- * Delegation (specific)



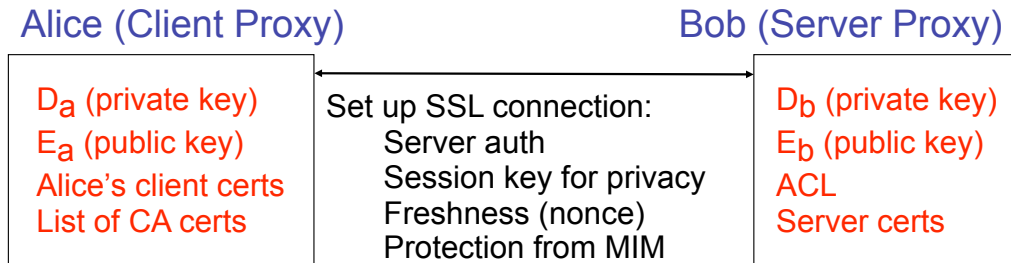
Pervasive Computing MIT 6.883 SMA 5508 Spring 2006 Larry Rudolph

SPKI/SDSI: Authorization Certificate

```
(cert
  (issuer
    (public-key
      (rsa-pkcs1-md5
        (e #23#)
        (n
          |AMMgMuKpqK13pHMhC8kuxaSeCo+yt8TadcgnG8bEo+erdrSBveY3C
          MBkkZqrM0St4KkmMuHMxhsp5FX71XBivW1+JGCBLfI7hxWDZCxCxGTMg
          bR4Fk+ctyUxIv3CQ93uYVkg9ca6awCxtSOEI7sLuEB+HKuOLjzTsH+
          +Txw9NAHq4r|)))
    (subject
      (public-key
        (rsa-pkcs1-md5
          (e #23#)
          (n
            |AKg3tOzoJ5PGQ5q9jzxxzwxE8o6bIZ6/cE8gEL+1xJa23viE3bz68ru
            hpD5muqJ+uyDCN*xgAZ0JVXJazmX1QjiGudj9kEmuni8gJRLZRu0T5E3
            K7OU2dodu0kdDg32kym7+ooZNe/F0zWGekefESeezyQ25kvNO3XQvMHX
            afWcYjRw|)))
      (tag
        (http
          (* set GET POST)
          (* prefix http://ostrich.lcs.mit.edu/demo/)))
        (propagate))
```

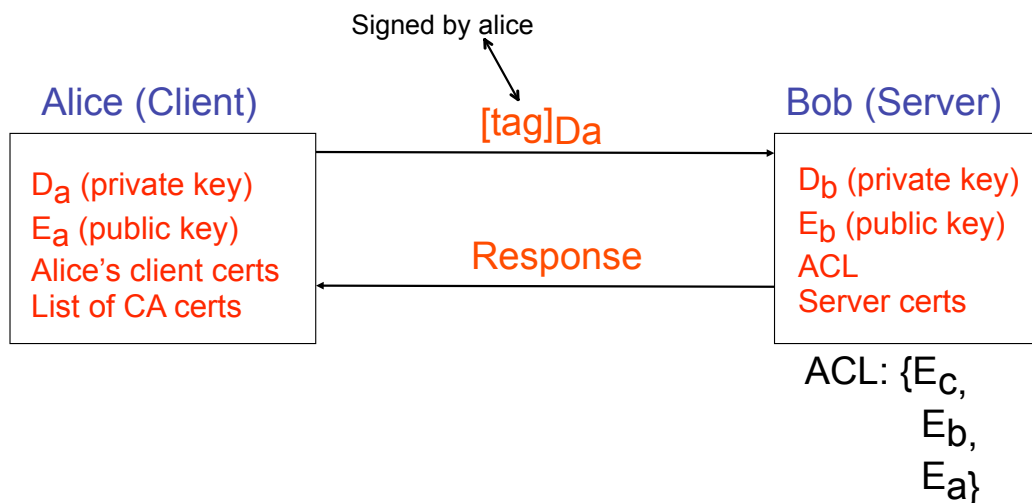
Proxy to Proxy

Initialization:



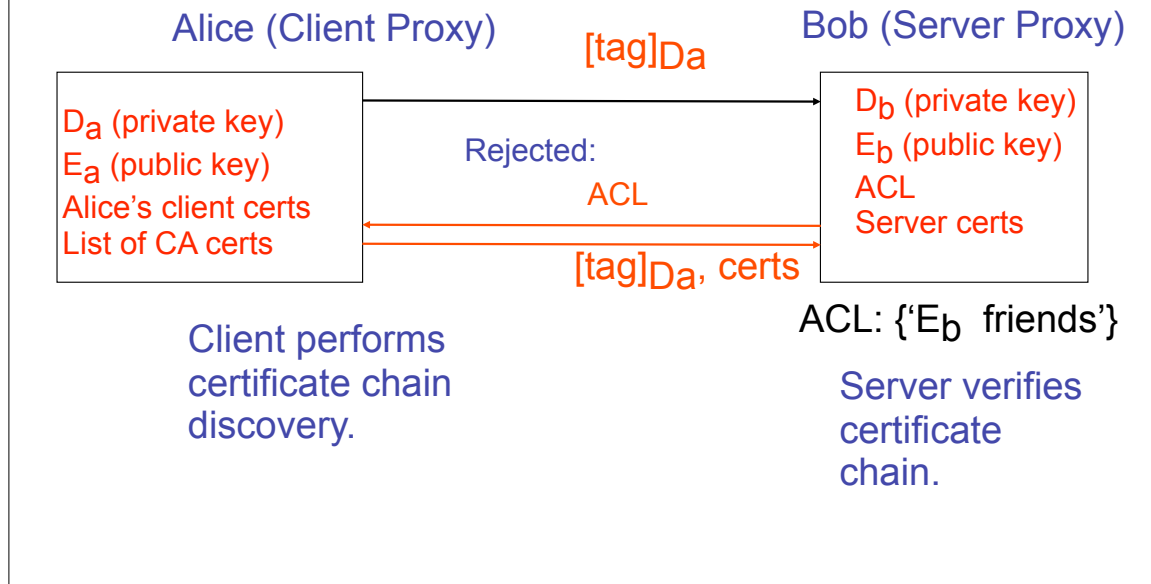
Proxy to Proxy

Case I: user's key is directly on the ACL



Proxy to Proxy

Case 2: user's key is "indirectly" on the ACL



Certificate Chaining Example

- ➔ Bob's ACL says only **MIT faculty** are allowed to access his server.
- ➔ Alice's first request is simply signed with Alice's key, and Bob rejects this request.
- ➔ Alice's second request contains a chain consisting of the following certificates:
 - ➔ A certificate saying she is an **CSAIL Professor**
 - ➔ A second certificate saying **CSAIL Professors are MIT faculty**



Certificate Chain Discovery (Client Proxy)

- Derive certificate chains
- Input: device's ACL, requestor's public key, requestor's set of signed certificates, tag
- Output: a chain of certificates leading from an entry on the ACL to the requestor's public key.

(The certificate chain consists of signed certificates. It proves that the requestor is authorized to perform the tag's operations on the device.)

* Recall, intuitively, a tag is a set of requests.



Pervasive Computing MIT 6.883 SMA 5508 Spring 2006 Larry Rudolph

Certificate Chain Verification (Server Proxy)

- Verify certificate chains
- Input: device's ACL, requestor's public key, requestor's certificate chain, tag
- Output: 1 if certificate chain proves that the public key is authorized to perform the tag's operations on the device; 0 otherwise.

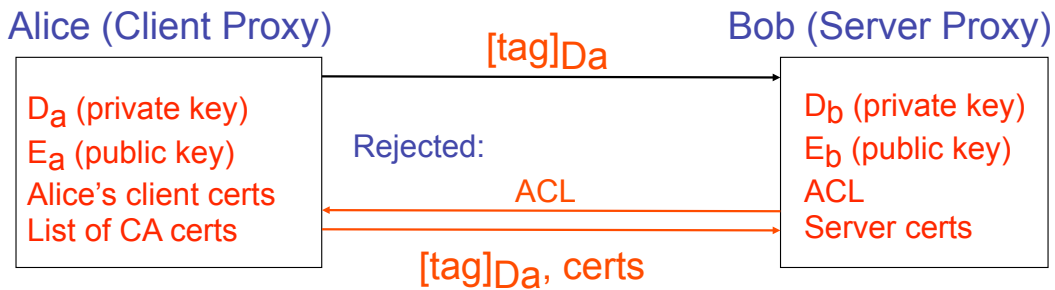


Pervasive Computing MIT 6.883 SMA 5508 Spring 2006 Larry Rudolph

Proxy to Proxy

Case 2 revisited

user's key is "indirectly" on the ACL



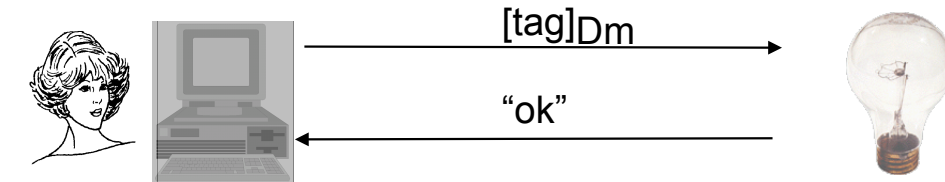
- Signed request provides proof of authenticity of the request
- Certificate chain provides proof that the request is authorized



Pervasive Computing MIT 6.883 SMA 5508 Spring 2006 Larry Rudolph

Example: Public resource

Mary wants to turn on/off a public light switch.



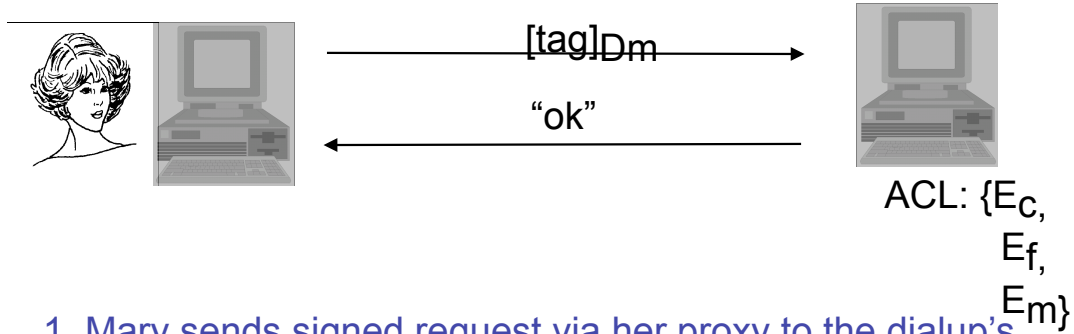
1. Mary sends request (either signed or unsigned) via her proxy to the light switch's proxy.
2. Light switch's proxy has no ACL. It honors Mary's request. Light switch's proxy may require requests to be signed for auditing purposes.



Pervasive Computing MIT 6.883 SMA 5508 Spring 2006 Larry Rudolph

User's key directly on ACL

Mary wants to log into an account on a dialup machine.



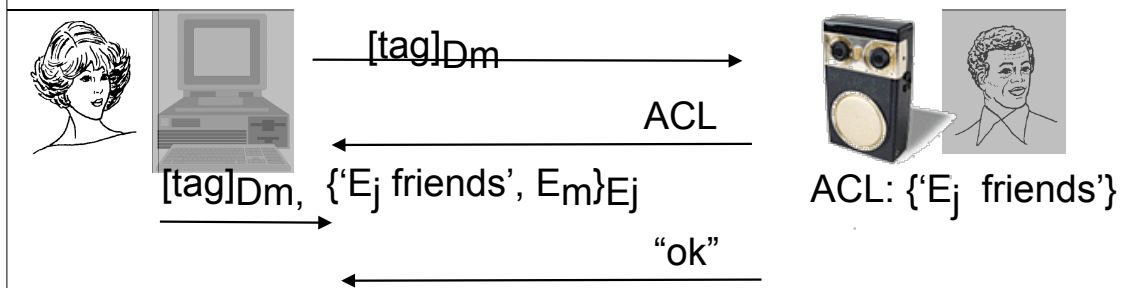
1. Mary sends signed request via her proxy to the dialup's proxy.

1. Dialup's proxy has an ACL which contains Mary's public-key. It checks the signature on Mary's request, and honors Mary's request to login if the signature verifies.



Example:user's key is indirectly on ACL

Mary wants to play music on John's speaker.



1. Mary sends signed request via her proxy to John's speaker's proxy.

2. John's speaker's proxy rejects first request, and returns the ACL.

1. Mary's proxy derives a chain of certificates and sends second request to John's speaker's proxy.

4. John's speaker's proxy verifies second request.

More “fun”

- ➡ Integrating access control with name lookup services
- ➡ Trusting untrusting devices
 - ➡ Using public terminals in Starbucks as a cache for handheld

