



Truly User-Centric PKI

Radia Perlman

Fellow

radia.perlman@sun.com



**2008
Sun Labs
Open House**



A heretic's view

Motivation

- It seems to me things should be simple
- I assumed decades ago we'd be doing mutual authentication based on public keys

Quick rant on username/pwd

Perils of Perlman

Buying something

- Scenario: Buy something from a merchant you haven't bought from recently
- All prepared with your info, credit card, etc.
- It asks you for your email address...

You're a returning user!

- Type your username and password!

You're a returning user!

- Type your username and password
- Of course you can't remember it, so...

You're a returning user!

- Type your username and password
- Of course you can't remember it, so...
 - > you manage to find "recover username"

You're a returning user!

- Type your username and password
- Of course you can't remember it, so...
 - > you manage to find "recover username"
 - > suddenly you are in a Monty Python movie
 - > Answer the following questions three:
 - Telephone number
 - Address
 - Mother's maiden name

New Rule

- It should be no more onerous to be a returning user than a new user

Security questions for password/username recovery

- Favorite sports team
- 2nd grade teacher's name
- Pet's name
- Father's middle name
- My middle name

New Rule

- Security questions must be specifiable by the user
- I'd say "or selectable from a very large list", but I'm sure they can come up with an arbitrarily long list of questions I can't answer

Security question in comedy routine

(Q&A Chosen by user, to be asked by the bank for phone verification of customer)

Security question in comedy routine

- Question: “Are you wearing underwear”?

Security question in comedy routine

- Question: “Are you wearing underwear”?
- Answer: “I don’t think that’s an appropriate question”

Keeping customer information

- I do not want to do “single click ordering”
- I do not mind typing in my address
- I do not mind typing in my credit card number
- Merchants insist on keeping all of this information
- And eventually this information gets stolen

New Rule

- After a merchant is paid, any subset of information about a customer (including all of the information) must be expunged by the merchant at the customer's request

Existing and future things

- A lot of things are similar in spirit
- I don't care about formats
- I will talk conceptually

Kerberos vs Public Key

- Kerberos
 - > On-line KDC knows one key for everything
 - > For A to talk to B, A requests “ticket” to B from the KDC, consisting of a shared key K_{AB} , and a message to B, encrypted with B's key, saying “A” knows K_{AB}
 - > To talk to B, A sends ticket, proves A knows K_{AB}
- Public key
 - > Offline CA signs “certificates”
 - > A and B send each other certificates, and authenticate using each's public key pair

KDC

Alice/Ka

Alice/Ka
Bob/Kb
Carol/Kc
Ted/Kt
Fred/Kf

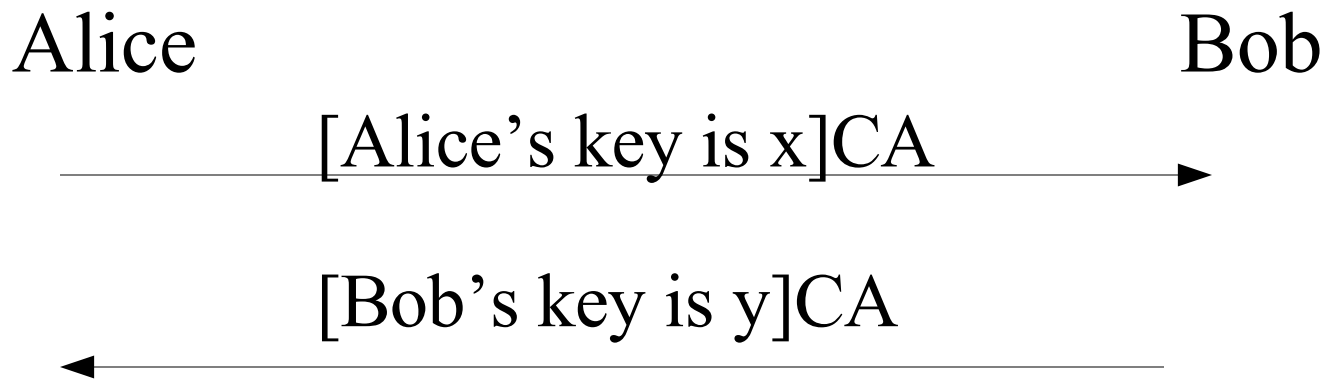
Ted/Kt

Bob/Kb

Fred/Kf

Carol/Kc

Key distribution with CAs



Mutual authentication, etc.

KDC vs. CA Tradeoffs

- KDC solution less secure
 - > Highly sensitive database (all user secrets)
 - > Must be on-line and accessible via the net
 - > complex system, probably exploitable bugs, attractive target
 - > Must be replicated for performance, availability
 - > each replica must be physically secured
 - > Privacy issue: KDC knows everything A attempts to talk to

KDC vs. CA

- KDC more expensive
 - > big, complex, performance-sensitive, replicated
 - > CA glorified calculator
 - > can be off-line (easy to physically secure)
 - > OK if down for a few hours
 - > not performance-sensitive
- Performance
 - > Have to talk to extra box before communicating

Revocation

- Levels the playing field somewhat, if want to check certificate validity on every interaction
- But, the revocation server is not as security sensitive as the CA or KDC
- And revocation can be “batched” with CRLs, delta CRLs, having client periodically get a non-revocation certificate

Now Federated Identity stuff

My view of federated things

- Microsoft created the “Passport” vision, with Microsoft the one identity provider
- Users authenticate to the identity provider, which vouches for them at affiliated sites
- Others said, “Hey, let’s not anoint one organization to be an eternal monopoly
- So, the notion of lots of IDPs, and a federation is the set of SPs that trust that IDP

If there is just one IDP

- User authenticates to that IDP
- That IDP vouches for the user at all the affiliated sites
- KDC vs IDP
 - > With KDC you cryptographically prove you know the secret inside the ticket
 - > With IDP, philosophy is “bearer token”, where no further authentication is necessary

Implications of bearer token vs ticket

- If client tricked into talking to the wrong thing, or if the same token is shared with multiple servers, client can be impersonated
- Bearer tokens could, in theory, be made somewhat more secure
 - > Limited in time
 - > Created for one transaction
 - > Created for one server
 - > Carry server's public key inside, and sent to server encrypted with that server's public key

But what if there are lots of IDPs?

- And what if the SPs the user wants to use affiliate with different subsets of them?
- No longer have single sign-on
- Less of an issue within a single trust domain than when a user is acting on their own behalf on the Internet

**And what value does the IDP
give, anyway?**

Downside of IDP (vs. peer-to-peer mutual authentication)

- Security (on-line IDP can impersonate all users)
- Availability (if IDP is down, nothing works)
- Performance (bouncing around between boxes)
- Privacy (IDP knows everyone you talk to)

Upside of IDP

- Within an organization, it's a single point of auditing (i.e., think of the fact that the one box knows who is talking to each other, as a feature)
 - > However...it only knows A is talking to B, not what is going on
 - > So you'd still need to do auditing at B
- Perhaps it can be an adaptation point for lots of versions of client software and server software (everything just needs to talk to it)

Term “user-centric”

- Often applied to the IDP-centric model
- Perhaps because a user gets to set policy on every attribute stored at the IDP, as to which service provider gets what information

Truly User-Centric

- I'll talk about public key-based models
 - > Within an organization (one CA)
 - > On the web (no CAs)
 - > Interorganizational

Simple intra-organizational PKI

Within an organization

- Should be trivial, single CA
- To create an account
 - > Sysadmin told username, takes new badge
 - > Inserts badge, types name into tool
 - > Tool generates certificate, stores cert in directory or on card, along with corporate CA's public key
- User logs in
 - > Activates smart card, inserts it into client machine
 - > Accesses resource: authenticates with public key

Until smart cards

- Could store the user's private key, encrypted with the password, and download it to the client machine the user is using

And, IDP and Kerberos also work

- Within an organization, also easy to have everything trust the same KDC, or IDP
- But better without having an online trusted box
 - > Performance, availability, security
 - > (Privacy not so much an issue in Enterprise scenario)

Individuals on the web

How about individuals?

- Think of this as just doing what we do with username/pwd, but more securely, and without torturing the user
- Assume first the user has a smart card with a secret (private key, or secret key)

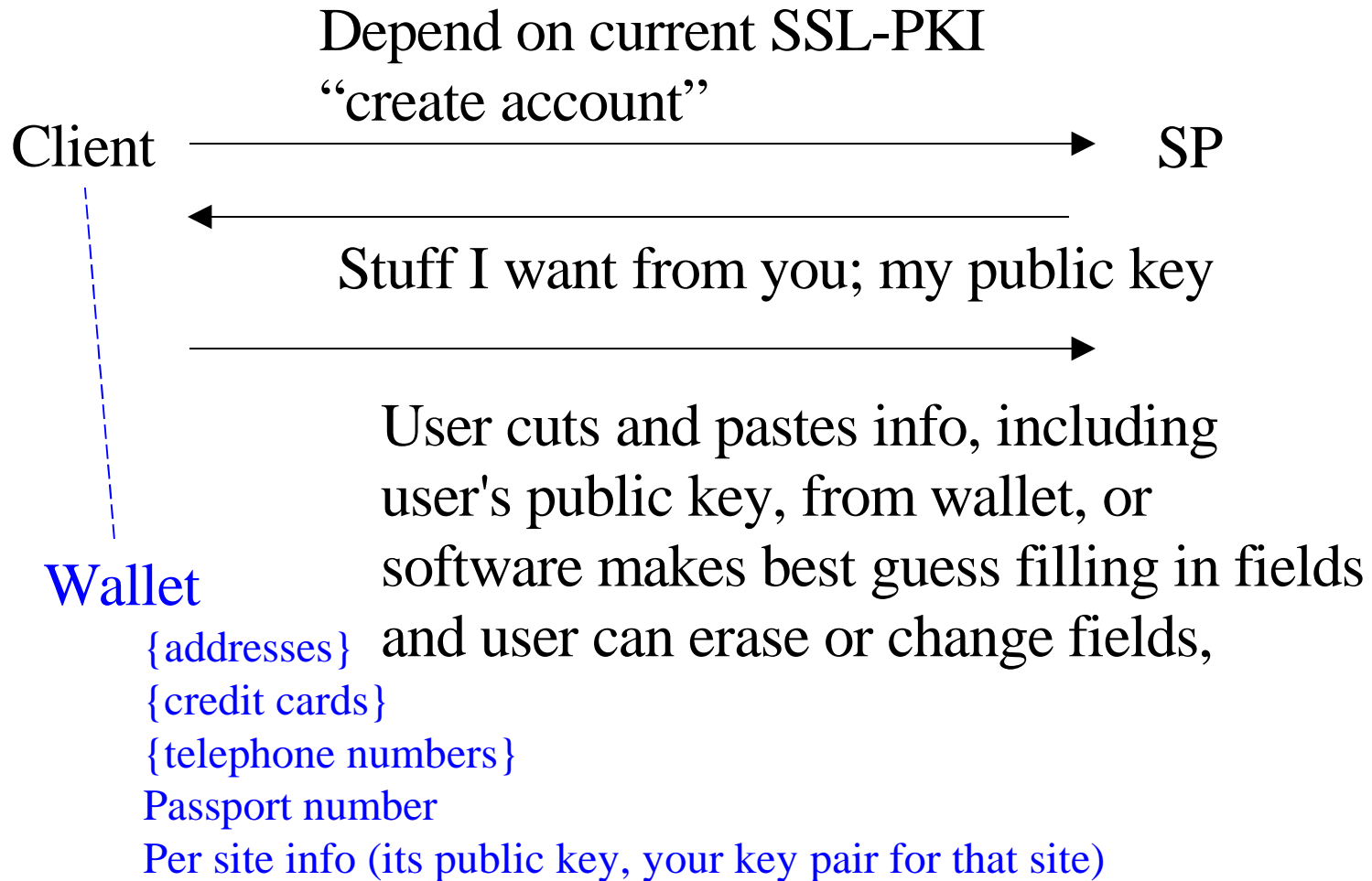
“Wallet”

- A bunch of data cryptographically protected with the user’s smart card secret
- Downloadable from one or more places
- Contains, for instance, public keys of various merchants, perhaps private keys to use with that merchant, information such as passport number and credit card numbers
- Note: CardSpace has a notion of user data stored at client, but not currently easily portable

Enrolling at a site

- Just like today, except username/pwd is replaced by “public key”
- The wallet information (such as address) can be filled into the form, to save the user typing, or the user could drag info she wants into the form
- The SP sends the user its public key

Enrolling



Note

- Instead of enrolling with a username and password, your account name is your public key, and you authenticate with your public key
- And by saving the SP's public key (a la SSH), you can do mutual authentication, knowing you are again reaching the same site as before

Revisiting the site

- Mutual authentication using public keys (e.g., SSL with client certs)

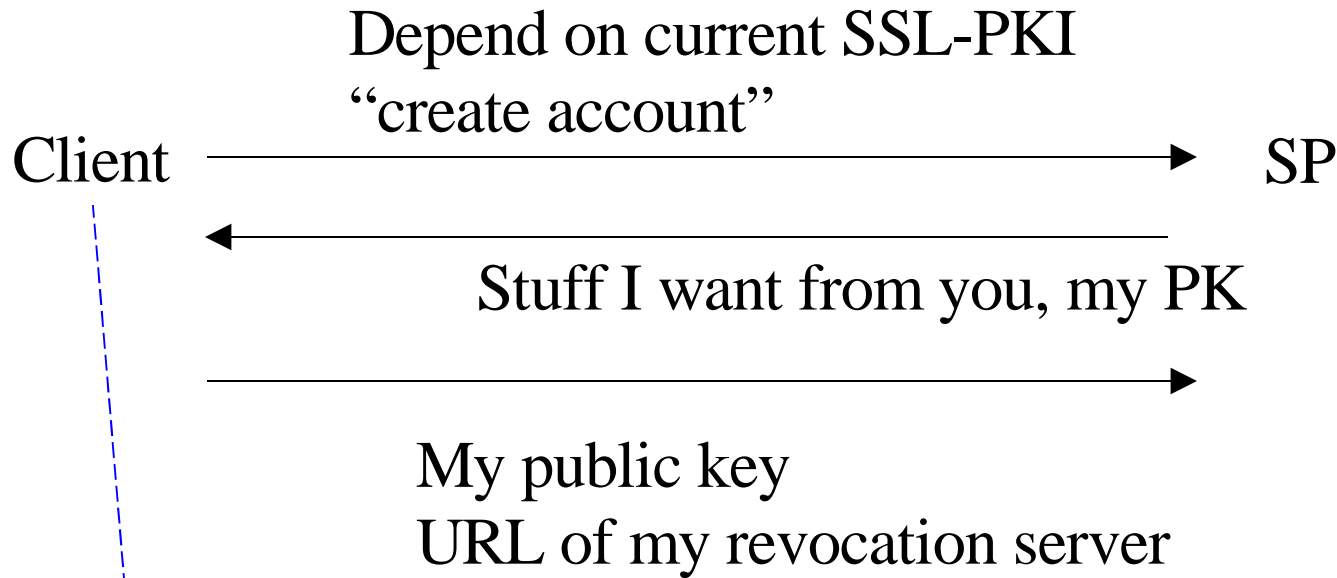
Filling in information

- Rather than deciding in advance who should get what information:
 - > Client does its best filling in form
 - > Perhaps gives user choices (like with multiple credit cards)
 - > User can decide at that point what information to delete from the form

One-step revocation

- Suppose you are using your public key at lots of sites
 - > (not sure how useful different keys for each site is)
- And someone steals it
- Use “revocation service”

Enrolling



Wallet

{addresses}

{credit cards}

{telephone numbers}

Passport number

Per site info (its public key, your key pair for that site)

Revocation service

- SP learns user's revocation server along with the user's public key
- SP can “enroll” with that revocation service, to be notified in case of revocation
- Or SP can check periodically
- User has to have some sort of out-of-band mechanism to authenticate and revoke the key

Authenticated attributes

- User can have, in wallet, certs signed by whoever is trusted to assert the attribute, that a public key associated with the user is over 18, a citizen, whatever
- Can send such certs to SP when needed, along with proof of knowledge of the private key

Other potential issues

- Authenticated attributes (e.g., qualified for AAA discount, age > 21, member of group allowed to do free IEEE library searches...)
- Web of trust between organizations
- Neither of those are harder with IDPs than with certificates

Summary

- Things would be higher performance, more available, simpler, with public key-based authentication