Applied Cryptography and Network Security

Adam J. Lee

adamlee@cs.pitt.edu

6111 Sennott Square

Lecture #14: PKI Models

February 25, 2014

University of Pittsburgh

Announcements

HW 1 will be graded this week.

Project 3 due next week! Discuss writeups with me.

Exam next Tuesday in class

Outline

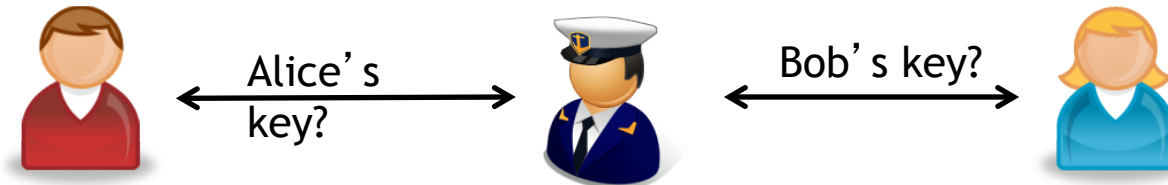What is a public key infrastructure (PKI)?

Some common PKI models
- Monopoly
- Delegated CAs
- Oligarchy
- Anarchy

Case study: PKIX/X.509

Case study: PGP

What is a digital certificate anyway?

Alice's key?   Bob's key?

Keeping track of public keys is simply not enough!

For instance, what if
- A transient fault switches Alice's and Bob's keys
- You and Trent have different opinions about who "Alice" is
- Trent is actually malicious and provides you with incorrect information
- Trent becomes compromised and the attacker provides false data
- …

In reality, key distribution servers manage digital certificates, rather than simple (name, key) pairs

Digital certificates are verifiable and unforgeable bindings between a user, a public key, and a trusted certifier (a certificate authority, or CA)

Terminology

If Alice issues a certificate vouching for Bob's name and key, then Alice is considered the issuer, and Bob is the subject

If Alice is verifying a certificate or chain of certificates, then she is called the verifier or relying party

Any user, machine, or service that has a certificate is called a principal

A trust anchor is a public key that is trusted by a verifier to certify the public keys of principals

Now that we have a basic vocabulary, let's explore a few ways that PKIs can be organized…

Most PKI models are hierarchical

A monopoly is the simplest form of PKI
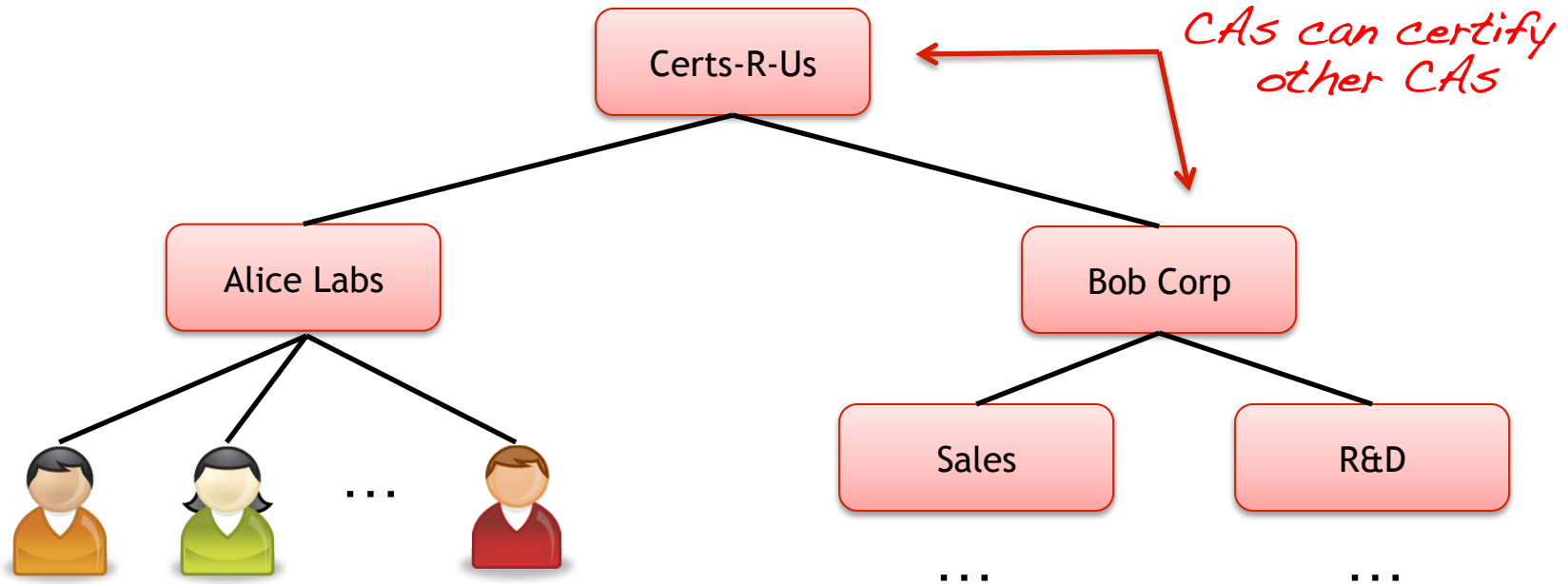
Certs-R-Us

One CA that everyone must trust
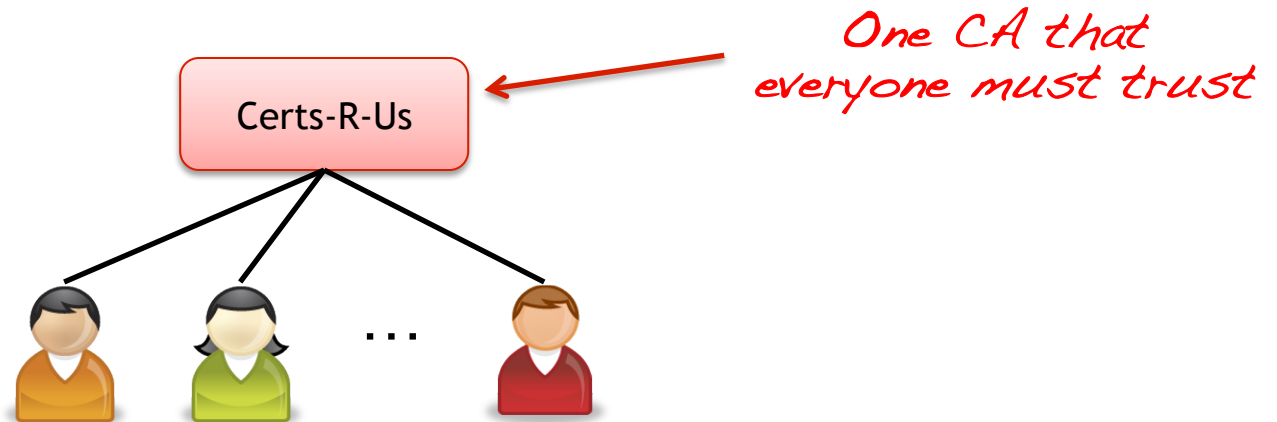
Pros:
- Very simple to model
- Only need to know a single public key to authenticate anyone

Cons:
- In real life, no organization is trusted by everyone
- Changing the key of that CA would be a nightmare
- How would a single CA verify the identity associated with every key?

Allowing delegation fixes some of these problems

Certs-R-Us

CAs can certify other CAs

Alice Labs

Bob Corp

...

Sales

R&D

...   ...

Pros:
- Fairly simple to model
- Only need to know a single public key to discover a certificate chain that authenticates anyone
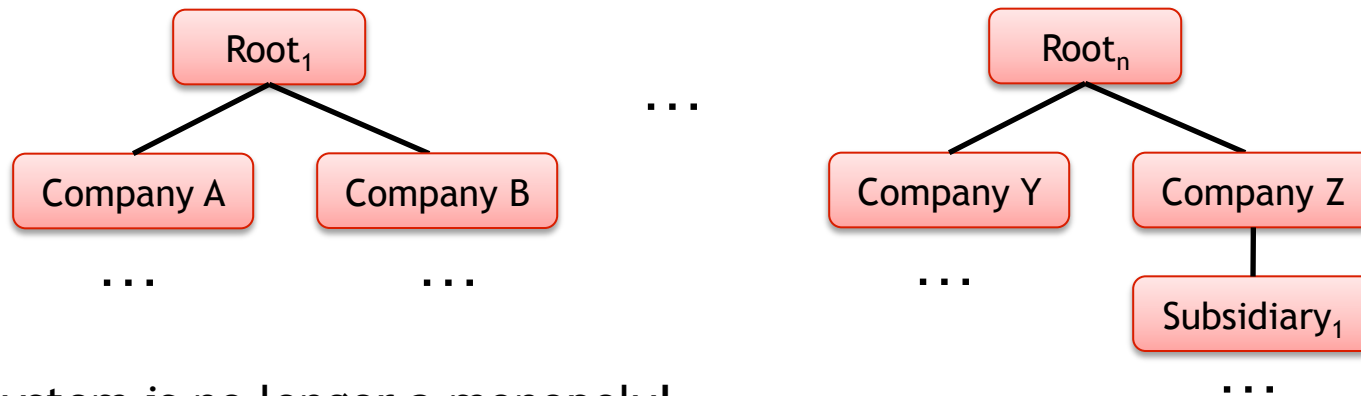- Delegation of control makes identity verification more tractable

Cons:
- In real life, no organization is trusted by everyone
- Changing the key of that CA would be a nightmare

An oligarchy is a collection of hierarchies

Rather than pre-configuring a single trust anchor, allow applications to have a number of trust anchors

```
        Root₁                            Rootₙ
       /     \          ...             /      \
Company A   Company B         Company Y   Company Z
                                                 |
   ...         ...               ...        Subsidiary₁

                                              ...
```

The system is no longer a monopoly!

- In a free market, this will make for more competitive certificate pricing
- There is no single point of attack

Unfortunately, this model is not without its problems

- Trust anchors are often chosen by application vendors, not users!
- If trust anchors are user configurable, it may be possible to trick users into adding bogus roots of trust!
- Managing a large set of trust anchors is complicated...

Your web browser uses the oligarchy model
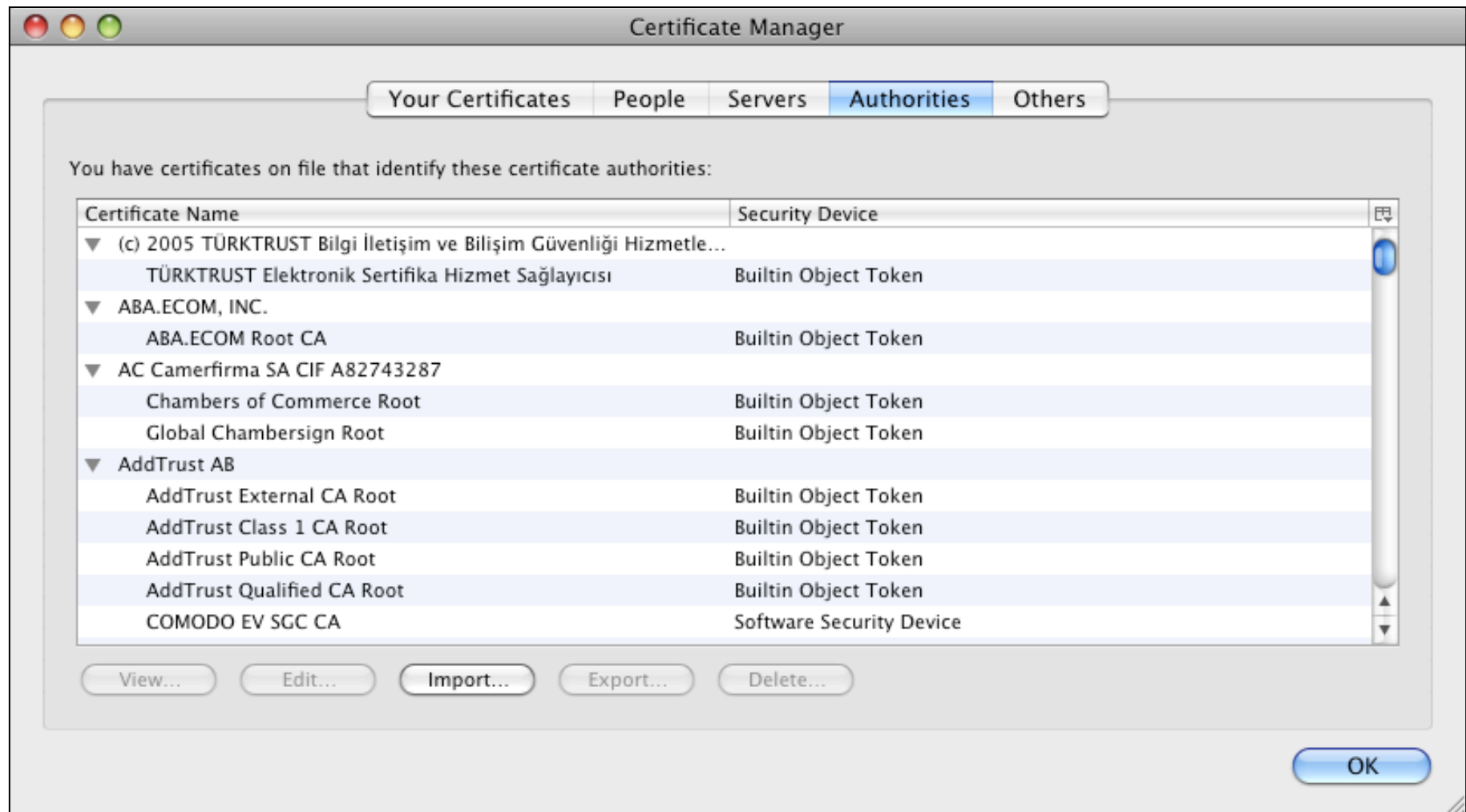
In any PKI model, certificate issuance is a tricky issue

Say you find a certificate for Adam Lee, who is it *really* for?
- An assistant professor at Pitt? (http://www.cs.pitt.edu/~adamlee/)
- A professional balloon twister? (http://www.adamlee.net/)
- A graphic designer? (http://www.adamleedesign.com/)
- A winemaker? (http://www.siduri.com/)
- …

A certificate is only helpful if it unambiguously vouches for an identity
- Am I really talking to Amazon.com, or is this a phishing site?
- Did I just take CS course advice from a balloon twister?

Question: How can we *unambiguously* specify identities?

In our case studies, we'll see how this can be accomplished using unique identifiers and social connections

How can we deal with revoked certificates?

Over time, it may become necessary to revoke a certificate. For example,

- The private key becomes compromised (and the owner finds out)
- The binding between the name and certificate becomes invalidated

The revocation process can be handled in an offline manner using a certificate revocation list (CRL)

- Digitally-signed list of revoked certificates
- Issued periodically by the CA
- Always consulted prior to accepting a certificate

Note: This is how credit card were checked for revocation in the "old" days


Question: Why do CRLs need to be signed?

Question: How can we make the distribution of large CRLs more efficient?

If we assume a reliable online service, we can implement a more timely revocation system

An online revocation server (OLRS) is an online service that is queried to check the validity of certificates

certificate id, nonce

[certificate id, nonce, status]k_{OLRS}^{-1}

Why is this message signed?
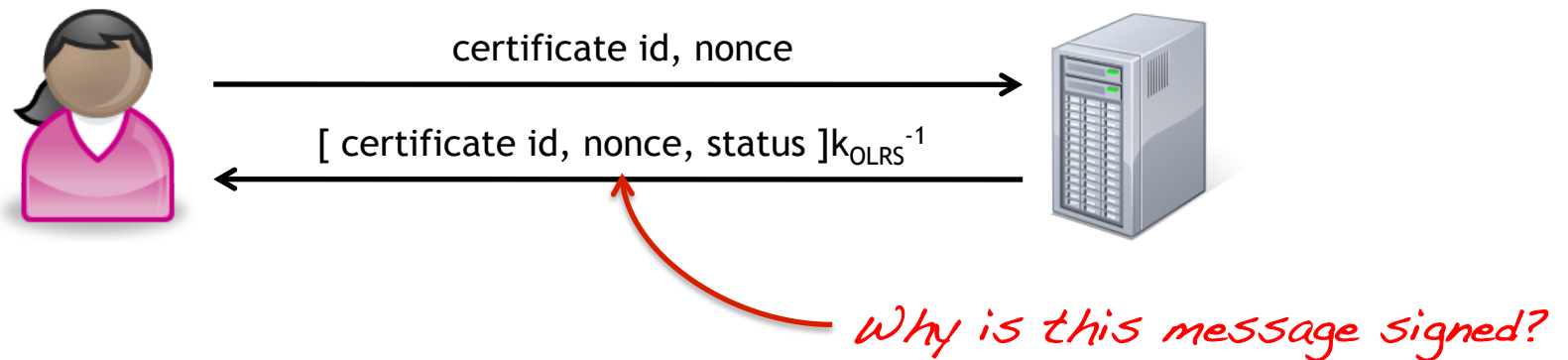
Usually, the OLRS is not the CA itself, but instead a service operating on behalf of the CA (Why?)

Alternatively, some OLRS pre-compute validity responses offline
- E.g., "Valid as of 08:00 on 2/26/2013"

Question: What are the tradeoffs between these two approaches?

Case study: X.509

The IETF's PKIX working group is tasked with developing standards for deploying public key infrastructures based on the X.509 standard

This group has released a number of RFCs describing things like
- Certificate contents (RFC 5280)
- Certificate path validity checking (RFC 3280)
- Handling revocation lists (RFC 5280)
- Online certificate status protocols (RFC 2560)
- …

X.509 certificates uniquely identify principals through the use of X.500 distinguished names (DNs)
- /O=University of Pittsburgh ← *Organization*
  /OU=Arts and Sciences
  /OU=Computer Science ← *Organizational units*
  /CN=Adam J. Lee ← *Common name*

X.509 Certificates

An X.509 certificate contains quite a bit of information

- Version
- Serial number: Must be unique amongst all certificates issued by the same CA
- Signature algorithm ID: What algorithm was used to sign this certificate?
- Issuer's distinguished name (DN): Who signed this certificate
- Validity interval: Start and end of certificate validity
- Subject's DN: Who is this certificate for?
- Subject's public key information: The public key of the subject
- Issuer's unique ID: Used to disambiguate issuers with the same DN
- Subjects unique ID: Used to disambiguate subjects with the same DN
- Extensions: Typically used for key and policy information
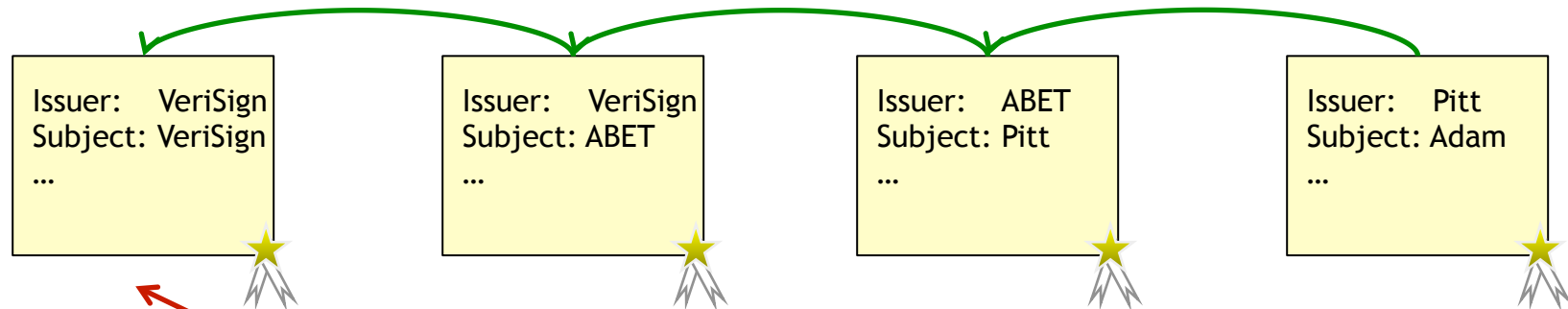- Signature: A digital signature of (a hash of) all other fields

Note that each certificate has exactly one issuer

Question: What if you wanted a certificate to be authorized by more than one entity, but didn't want to deviate from the X.509 standard?
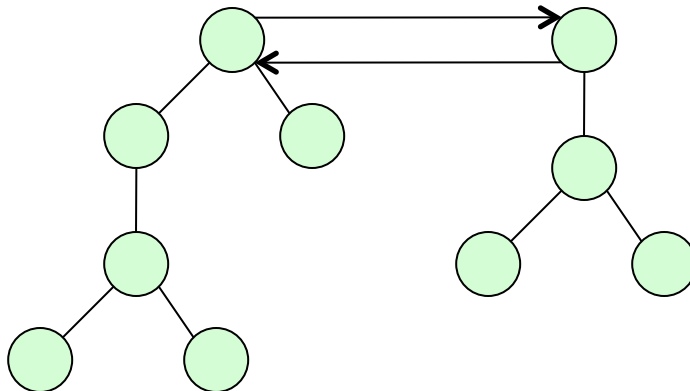
How can we validate X.509 certificates?

Since each certificate has exactly one issuer, we can build chains of trust

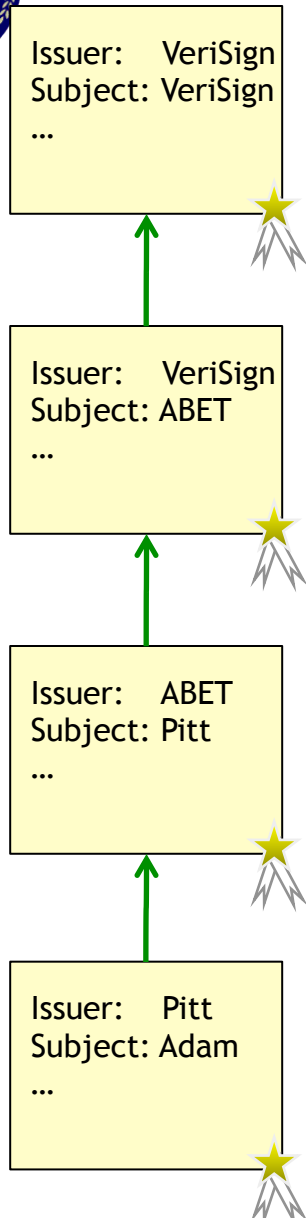| Issuer: VeriSign
Subject: VeriSign
… | Issuer: VeriSign
Subject: ABET
… | Issuer: ABET
Subject: Pitt
… | Issuer: Pitt
Subject: Adam
… |

In this example, VeriSign represents a trust anchor

As you might expect, X.509 CAs form a hierarchy

Note: Cross-certification allows certificates from one domain to be interpreted in another

Question: Is there any value to allowing the use of self-signed client certificates within the X.509 model?

RFC 3280 describes how X.509 certificate chains are supposed to be validated

| Issuer: VeriSign |
| Subject: VeriSign |
| ... |

| Issuer: VeriSign |
| Subject: ABET |
| ... |

| Issuer: ABET |
| Subject: Pitt |
| ... |

| Issuer: Pitt |
| Subject: Adam |
| ... |

Checks run on certificate 1:
- Is this certificate a trust anchor?
- Is this certificate valid at the current time?
- Has this certificate been revoked?
- Is the self signature on this certificate valid?

Checks run on certificates 2 through n-1:
- Are these certificates CA certificates?
- Are these certificates currently valid?
- Have any of these certificate been revoked?
- Was certificate i signed by certificate i-1?

Checks run on certificate n:
- Is this certificate valid at the current time?
- Has this certificate been revoked?
- Was this certificate signed by certificate n-1?

Question: How do we find certificate chains in the first place?

Case study: PGP

Pretty Good Privacy (PGP) is a hybrid encryption program that was first released by Phil Zimmerman in 1991

In the PGP model
- Users are typically identified by their email address
- Users create and manage their own digital certificates
- Certificates can be posted and discovered by using volunteer "key servers"
- Key servers also serve a function similar to an OLRS

Email addresses are GUIDs, so they effectively disambiguate identities

However, note that there are no CAs in the system! As such, users can create certificates for any email address or identity that they want!

How can we deal with this anarchy?!

Unlike X.509, PGP takes a more "grassroots" approach to certificate validation

Users create their own certificates, and sign the certificates of others
- "I am Alice and I have verified that this certificate belongs to Bob"
- Four levels of certification: Generic, Persona, Casual, and Positive

This is essentially a cryptographic social network!
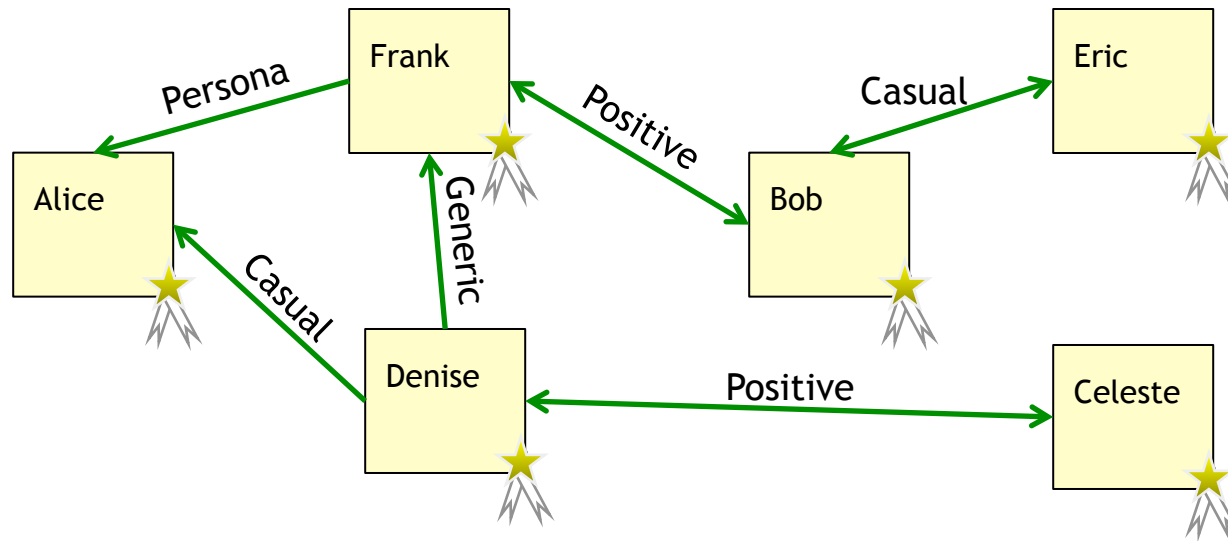
Question: Can Celeste trust Frank? Can Eric trust Frank?

PGP can handle two types of certificate formats

In addition to supporting X.509 certificates, PGP also has its own native certificate format

A PGP certificate contains the following information:

- PGP version number
- Public key: E.g., an RSA, DH, or DSA public key
- The certificate holder's "information": Free-form contents
- Signature: A self-signature on the certificate
- Validity period: When was this certificate created? How long is it valid?
- Preferred algorithms: What cipher suites should be used?

Question: If PGP keys are self-issued, why is a signature needed?

Discussion

Question: Which approach do you think is better? Why?

Conclusions

Digital certificates act as verifiable and unforgeable bindings between an identity and a public key

There are many PKI models out there
- Some based on hierarchies (e.g., X.509)
- Some more ad-hoc (e.g., PGP)

In any deployment model, it is essential to unambiguously identify users

Handling revoked certificates can be done either online or offline

Next time: Real-time communication security