



Trust Management

CS 6463 Lecture Four

Prof. William Winsborough
January 30, 2006

Origin of Trust Management

- Decentralized Trust Management. Matt Blaze, Joan Feigenbaum, Jack Lacy. IEEE Symposium on Security and Privacy, 1996 (Oakland 96). IEEE Press, pp.164-173.
 - Seminal paper to identify the need for a coherent framework for organizing (1) security policies, (2) security credentials, and (3) trust relationships in decentralized systems
 - They coined the term the *trust management problem* for this collection of network services

Design Goals

- Unified mechanism – Policies, credentials, and trust relationships are all treated within the same framework
- Flexibility – Expressive enough to support complex trust relationship in very large distributed systems

Design Goals, Part 2

- Locality of Control – The parties that rely on the credentials can decide by whom such credentials must be issued.
 - No monolithic hierarchy of “certifying authorities”
- Separation of mechanism from policy – credential verification is independent of the meaning of the credentials with respect to the application

Prior Work Studying Parts of the Trust Management Problem

- Prior work that solves parts of the problem
 - Hierarchical public key infrastructure (PKI) systems such as X.509 [e.g. Adams & Lloyd 03]
 - PGP [Zimmermann 94] – web of trust
 - Both of these serve to bind a user's name to a public key
 - This assists with *authentication* (e.g., for email, remote login)
 - But at best it solves only half the problem when trying to make *authorization* decisions
 - Neither approach scales well

Prior Certificate Systems

- Bind keys to names (or email addresses), but do not address the problem of who has what rights
- PGP and X.509 (at least in early versions) attempt to solve the same problem:
 - Finding a suitably trustworthy copy of the public key of someone with whom one wishes to communicate
 - Enables sending confidential messages, verifying signatures from known parties

Quick Review of What Public/Private Key Crypto Gives Us

- Public and private keys come in pairs.
 - Clear text encrypted using the public key and be decrypted only by using the corresponding private key, and vice versa
- This enables:
 - Sending messages readable only by the person possessing the private key
 - Signing documents with signatures that are unforgeable and are verifiable by anyone knowing the corresponding public key

PGP Overview

- Public key record:

((Name, EmailAddress), PublicKey, CreationTimeStamp)

- E.g., a public key record for (and generated by) B:
((B, B@hotmail.com), BKEY, 1/1/06)
- If A has a copy he trusts he signs it and can pass it to C
(The the singed record is a *key certificate*)
- Here A acts as an introducer of B to C
- C can assign various degrees of trust to different introducers (such as A) and use them to organize authorization policies
 - E.g., one highly trusted introducer is adequate, but with marginally trusted introducers, 2 are required

What Can Be Verified?

- The identity of the sender of an email message
 - Message is signed by using the sender's public key
 - Recipient verifies the signature by decrypting it with the public key received from a trusted introducer
- Note that A's signature on B's public-key record does not mean A trusts B or any property of B other than his ID

Where Do Trusted Introducers Come From In PGP?

- Not solved by the PGP system
- In particular, introducer trust need not be transitive
 - Each participant must decide in an ad hoc manner the level of trust to accord each introducer
- This approach has lead to a decentralized “web of trust”
 - Each participant is responsible for creating his own key records and for acting as introducers of people know to him for others

The Hierarchical Approach of X.509

- Attempts to solve the same problem: binding ID to key
- In X.509, only official *certifying authorities* (CAs) provide certificates
- Certificates are registered in an official directory service, from which relying parties subsequently retrieve them
- If A and B want to communicate, but were not certified by the same CA, the directory service must create a *certification path* from A to B:
 $CA_1, cert_1, CA_2, cert_2, \dots, CA_n, cert_n$ where $cert_i$ is a certificate of CA_{i+1} signed by CA_i , for $1 \leq i < n$, and $cert_n$ is a certificate of B
- Rests on the assumption that all CAs are organized into a global certifying authority tree

