# Kerberos, OSF/DCE, and Public Key

Burt Kaliski RSA Laboratories

1993 RSA Data Security Conference

## Outline

### Problem

Solutions

### Passwords

#### **Mediation**

Kerberos and OSF/DCE

### **Public keys**

NetWare and SPX

### Conclusions

### Problem

Alice sends a message to Bob.

How does Bob know it came from Alice, and it hasn't been modified en route?

Bob needs *identification*, *authentication*, *integrity*.

Applications: Login, database access, printing, network management, ...

### Solutions

### Passwords

Alice, Bob authenticate with a shared password.

#### Mediation

Alice, Bob authenticate with a shared session key issued by key center.

#### **Public keys**

Alice, Bob authenticate with digital signatures.

### Passwords

Alice, Bob share a password  $K_{AB}$ .

They authenticate each other by encrypting, for example, a time stamp under the password:

Alice  $\rightarrow$  Bob: A,  $K_{AB}$ (time) Bob  $\rightarrow$  Alice:  $K_{AB}$ (time+1) If they also exchange a secret key under the password, they can authenticate (and encrypt) subsequent messages without the password.

## **Passwords (cont'd)**

Good speed, but poor scalability; two copies/secret, many secrets/user.

### Mediation

Alice shares a secret key  $K_{AC}$  with key center; Bob shares a secret key  $K_{BC}$ .

Alice, Bob authenticate each other with a session key issued by the key center.

For example, Alice authenticates to the key center with  $K_{AC}$ , and the key center returns two encrypted copies of the session key:  $\langle K_{AC}(K_{session}), K_{BC}(K_{session}) \rangle$ .

# Mediation (cont'd)

Alice sends Bob his copy, and they authenticate each other with the session key:

Alice  $\rightarrow$  Bob: A,  $K_{BC}(K_{session})$ ,  $K_{session}$ (time) Bob  $\rightarrow$  Alice:  $K_{session}$ (time+1)

Alice and Bob can authenticate (and encrypt) subsequent messages with the session key. Good speed, fair scalability; two copies/secret, one secret/user; key center trusted with secrets.

### Kerberos

Mediated solution—Needham & Schroeder, 1978; MIT, 1986.

#### **Participants**

Users

Servers

Kerberos server

Ticket-granting server (TGS)

### Keys

User, Kerberos server share a secret key.

Kerberos server, TGS share a secret key.

Server, TGS share a secret key.

# Kerberos (cont'd)

### At login

User authenticates to Kerberos server with shared secret to get a *ticket* to TGS and encrypted session key.

Ticket contains user's name, time, encrypted under TGS secret key.

#### For each service

1. User authenticates to TGS with TGS ticket, session key to get server ticket, encrypted session key.

2. User authenticates to server with server ticket, session key.

# Kerberos (cont'd)

#### What about scalability?

TGS shares secret keys with servers.

For Alice to authenticate to another server requires TGS to share a secret key with that server

... or with another TGS.

Suppose Alice sends a message to Robert (a server) in France.

If Alice's local TGS shares a secret key with Robert's TGS, then Alice can get a ticket to Robert's TGS, and therefore to Robert.

Robert authenticates Alice with the ticket.

# Kerberos (cont'd)

### What about scalability? (cont'd)

TGS's must trust each other with secret keys. TGS trust hierarchy simplifies administration.

Future versions of Kerberos may include TGSto-TGS authentication with public keys.In such versions, TGS's need not trust each other with secret keys, just certification authorities with public keys.

### DCE

Distributed Computing Environment—another mediated secret-key solution—HP, OSF, 1990s.

"Commercializable" Kerberos v5 with new tools.

Participants, keys, protocols as in Kerberos, with extensions such as access control lists; support for directory names.

# Public-key versions planned, as part of Sesame effort.

# **Public keys**

Alice has public key  $R_A$ , private key  $S_A$ ; Bob has public key  $R_B$ , private key  $S_B$ .

They also have certificates  $S_C(\text{user}, R_{\text{user}})$ , where  $S_C$  is certification authority's private key.

Alice and Bob authenticate each other by encrypting, for example, a time stamp under their private keys: Alice  $\rightarrow$  Bob:  $S_{C}(\langle A, R_{A} \rangle)$ ,  $S_{A}(\text{time})$ Bob  $\rightarrow$  Alice:  $S_{C}(\langle B, R_{B} \rangle)$ ,  $S_{B}(\text{time})$ 

# Public keys (cont'd)

If they also exchange a secret key under one of their public keys, they can authenticate (and encrypt) subsequent messages.

Fair speed, good scalability; one copy/secret, one secret/user; certification authorities trusted with public keys.

But good speed in hybrid with secret key.

### **NetWare**

A public-key solution—Novell, 1993.

### Participants

Users

Servers

Certification authority

### Keys

Users have public/private key pairs.

Servers have public/private key pairs.

Certification authority issues certificates to users, servers.

# NetWare (cont'd)

### At login

User authenticates to login server with key pair, generates a short-term key pair.

### For each service

User authenticates to server with short-term key pair; includes long-term and short-term certificates.

If user, server exchange a secret key, they can authenticate (and encrypt) subsequent messages.

(Actual implementation is more efficient; has zero-knowledge proofs, avoids short-term certificate.)

# NetWare (cont'd)

What	about	scala	bility?
------	-------	-------	---------

Alice can authenticate to any server that trusts Alice's certification authority.

- No administration of secret keys.
- Certification hierarchy simplifies administration of public keys.
- TGS must trust certification authorities with public keys.

### SPX

Another public-key solution—Tardo, Alagappan & Pitkin, Digital, 1989.

Protocols as in Kerberos, with certification authority, not Kerberos server/TGS.Compatible with Generic Security Service API.

As in NetWare, but RSA session keys, rather than zero-knowledge proofs. X.509/Privacy-Enhanced Mail certificates.

### Conclusions

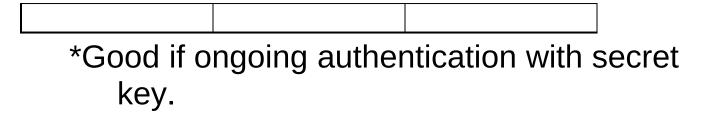
### Problem

How does Bob know it's from Alice?

#### Solutions

Passwords, mediation, public key

	passwords	mediation	public key
speed	good	good	fair*
scalability	poor	fair	good
copies/ secret	two	two	one
secrets/user	many	one	one



# **Conclusions (cont'd)**

#### Systems

Kerberos, OSF/DCE, NetWare, SPX

#### **Speed vs. scalability**

Secret key for speed, public key for scalability

Hybrid solutions achieve both