CSE543 - Introduction to Computer and Network Security

Module: Authentication

Professor Trent Jaeger
Kerberos

- History: from UNIX to Networks (late 80s)
  - Solves: password eavesdropping
    - Also mutual authentication
  - Online authentication
    - Variant of Needham-Schroeder protocol
  - Easy application integration API
  - First *single sign-on system* (SSO)
  - Genesis: rsh, rcp
    - authentication via assertion

- Most widely used (non-web) centralized password system in existence (and lately only ..)

- Now: Windows 2K/XP/Vista/etc network authentication
  - Old Windows authentication was a cruel joke.
An aside …

• Authentication
  ‣ Assessing identity of users
  ‣ By using credentials …

• Authorization
  ‣ Determining if users have the right to perform requested action (e.g., write a file, query a database, etc.)

• Kerberos authenticates users, but does not perform any authorization functions …
  ‣ … beyond identify user as part of Realm
  ‣ Typically done by application.

• Q: Do you use any “Kerberized” programs?
  ‣ How do you know?
The setup …

• The players
  ‣ Principal - person being authenticated
  ‣ Service (verifier) - entity requiring authentication (e.g., AFS)
  ‣ Key Distribution Center (KDC)
    • Trusted third party for key distribution
    • Each principal and service has a Kerberos password known to KDC, which is munged to make a password $ke$, e.g., $k^A$
  ‣ Ticket granting server
    • Server granting transient authentication

• The objectives
  ‣ Authenticate Alice (Principal) to Bob (Service)
  ‣ Negotiate a symmetric (secret) session key $k^{AB}$
The protocol

• A two-phase process
  1. User authentication/obtain session key (and ticket granting ticket) key from Key Distribution Center
  2. Authenticate Service/obtain session key for communication with service

• Setup
  ‣ Every user and service get certified and assigns password
A Kerberos Ticket

- A kerberos ticket is a token that …
  - Alice is the only one that can open it
  - Contains a session key for Alice/Bob ($K_{AB}$)
  - Contains *inside it* a token that can only be opened by Bob

- Bob’s Ticket contains
  - Alice’s identity
  - The session key ($K_{AB}$)

- Q: What if issuing service is not trusted?
Phase 1 (obtaining a TGT)

- $\text{Time}_{\text{exp}}$ - time of expiration
- $n$ - nonce (random, one-use value: e.g., timestamp)

\[
[A, TGS, \text{Time}_{\text{exp}}, n] \rightarrow KDC
\]

\[
E(k^A, [k^{A, TGS}, TGS, \text{Time}_{\text{exp}}, n]), E(K^{TGS}, [A, k^{A, TGS}, \text{Time}_{\text{exp}}])\]

TGT
Phase 2 (authentication/key dist.)

\[ \begin{align*}
[B, \text{Time}_{\text{exp}}, n, \text{E}(k^A, TGS, [B, \text{Time}_{\text{exp}}, n])], & \text{ E}(K^{TGS}, [A, k^A, TGS, \text{Time}_{\text{exp}}])] \\
\end{align*} \]
Kerberos Reality

• V4 was supposed to be replaced by V5
  ‣ But wasn’t because interface was ugly, complicated, and encoding was infuriating

• Assumes *trusted path* between user and Kerberos

• Widely used in UNIX domains

• Robust and stable implementation

• **Problem**: trust ain’t transitive, so not so good for large collections of autonomous enterprises
Meeting Someone New

• Anywhere in the Internet
What is a certificate?

• A certificate …
  ‣ … makes an association between a user identity/job/attribute and a private key
  ‣ … contains public key information \( \{e,n\} \)
  ‣ … has a validity period
  ‣ … is signed by some certificate authority (CA)
  ‣ ... identity may have been vetted by a registration authority (RA)

• Issued by CA for some purpose
  ‣ Verisign is in the business of issuing certificates
  ‣ People trust Verisign to vet identity
What is a certificate?

• A certificate …
  ‣ … makes an association between a user identity/job/attribute and a private key
  ‣ … contains public key information \{e,n\}
  ‣ … has a validity period
  ‣ … is signed by some certificate authority (CA)
  ‣ ... identity may have been vetted by a registration authority (RA)

• Issued by CA for some purpose
  ‣ Verisign is in the business of issuing certificates
  ‣ People trust Verisign to vet identity
What is a certificate?

• A certificate …
  ‣ … makes an association between a user identity/job/attribute and a private key
  ‣ … contains public key information \{e,n\}
  ‣ … has a validity period
  ‣ … is signed by some certificate authority (CA)
  ‣ … identity may have been vetted by a registration authority (RA)

• Issued by CA for some purpose
  ‣ Symantec is in the business of issuing certificates
  ‣ People trust Symantec to vet identity
What is a certificate?

• A certificate …
  ‣ … makes an association between a user identity/job/attribute and a private key
  ‣ … contains public key information \{e,n\}
  ‣ … has a validity period
  ‣ … is signed by some certificate authority (CA)
  ‣ … identity may have been vetted by a registration authority (RA)

• Issued by CA for some purpose
  ‣ Symantec is in the business of issuing certificates
  ‣ People trust Symantec to vet identity
Why do I trust the certificate?

• A collections of “root” CA certificates
  ‣ … baked into your browser
  ‣ … vetted by the browser manufacturer
  ‣ … supposedly closely guarded (yeah, right)

• Root certificates used to validate certificate
  ‣ Vouches for certificate’s authenticity

CA (signs) Certificate

Signature
Public Key Infrastructure

• System to “securely distribute public keys (certificates)”
  ‣ Q: Why is that hard?

• Terminology:
  ‣ Alice signs a certificate for Bob’s name and key
    • Alice is issuer, and Bob is subject
  ‣ Alice wants to find a path to Bob’s key
    • Alice is verifier, and Bob is target
  ‣ Anything that has a public key is a principal
  ‣ Anything trusted to sign certificates is a trust anchor
    • Its certificate is a root certificate
Possible PKI Constructions
Possible PKI Constructions

• Monarchy
  ‣ Single globally trusted third party
Possible PKI Constructions

- Monarchy
  - Single globally trusted third party
Possible PKI Constructions

- Monarchy
  - Single globally trusted third party

- Anarchy
  - No globally trusted third party
    - e.g., Using MIT’s PGP keyserver
Possible PKI Constructions

• Monarchy
  ‣ Single globally trusted third party

• Anarchy
  ‣ No globally trusted third party
    • e.g., Using MIT’s PGP keyserver
Possible PKI Constructions

• Monarchy
  ‣ Single globally trusted third party

• Anarchy
  ‣ No globally trusted third party
    • e.g., Using MIT’s PGP keyserver

• Oligarchy
  ‣ Multiple globally trusted third parties
    • Model used in the Internet
Possible PKI Constructions

• Monarchy
  ‣ Single globally trusted third party

• Anarchy
  ‣ No globally trusted third party
    • e.g., Using MIT’s PGP keyserver

• Oligarchy
  ‣ Multiple globally trusted third parties
    • Model used in the Internet
The Internet PKI?

- Rooted tree of CAs
- Cascading issuance
  - Any CA can issue cert
  - CAs issue certs for children
Certificate Validation
Certificate Validation
Certificate Validation

Certificate Validation

Certificate
Signature

Root

CA1

CA11

CA12

CA1n

CA2

CA21

CA22

CA3

Cert11a

Cert11b

Cert11c

…

…

…

…
PKI and Revocation

• Certificate may be revoked before expiration
  ‣ Lost private key
  ‣ Compromised
  ‣ Owner no longer authorized

• Revocation is hard …
  ‣ The “anti-matter” problem
  ‣ Verifiers need to check revocation state
    • Loses the advantage of off-line verification
  ‣ Revocation state must be authenticated
Revocation Mechanisms

Let's communicate securely

Here's my certificate and current status:
Issuer: CA
Subject: example.com
... Signature: e137d6...
Certificate: example.com
Status: valid
Signature: d3b073...

That status response is recent and valid, so we're all set.
Revocation Mechanisms

- Certificate revocation lists (CRL)
  - Periodically issued
  - Delta CRLs when CRLs get too large
Revocation Mechanisms

• Certificate revocation lists (CRL)
  ‣ Periodically issued
  ‣ Delta CRLs when CRLs get too large

• Online certificate revocation server
  ‣ Answers revoked = yes/no for a particular certificate
    • Implemented by OCSP protocol
  ‣ Disadvantages?
  ‣ OCSP-stapling
Where’s my PKI?
Where’s my PKI?

• Really talking about a full PKI (everyone has certs.)
Where’s my PKI?

• Really talking about a full PKI (everyone has certs.)

• Why is that not a reality?
  ‣ PKI was, like many security technologies, claimed to be a panacea
  ‣ It was intended to solve a very hard problem: build trust on a global level
  ‣ Running a CA -- “license to print money”
Where’s my PKI?

• Really talking about a full PKI (everyone has certs.)

• Why is that not a reality?
  ‣ PKI was, like many security technologies, claimed to be a panacea
  ‣ It was intended to solve a very hard problem: **build trust on a global level**
  ‣ Running a CA -- “license to print money”

• Basic premise:
  ‣ Assertion #1 - e-commerce does not need PKI
  ‣ Assertion #2 - PKI needs e-commerce
Where’s my PKI?

• Really talking about a full PKI (everyone has certs.)
• Why is that not a reality?
  ‣ PKI was, like many security technologies, claimed to be a panacea
  ‣ It was intended to solve a very hard problem: build trust on a global level
  ‣ Running a CA -- “license to print money”

• Basic premise:
  ‣ Assertion #1 - e-commerce does not need PKI
  ‣ Assertion #2 - PKI needs e-commerce
• What are the problems?
Where’s my PKI?
Where’s my PKI?

- Some of the problems with creating a per-user PKI?
  - Who has the private key? (Security of client hosts)
  - How do I manage my private key(s)? (Usability)
  - Which users is a CA an authority over? (Root of Trust)
  - How do users find a legit CA? (Trusted Path)
Where’s my PKI?

• Some of the problems with creating a per-user PKI?
  ‣ Who has the private key? (Security of client hosts)
  ‣ How do I manage my private key(s)? (Usability)
  ‣ Which users is a CA an authority over? (Root of Trust)
  ‣ How do users find a legit CA? (Trusted Path)

• **Argument**: We are trying to solve a painful problem: authenticating users.
  ‣ What **technical expectations** can we make about users?
PKI (Circa 2009)

Verisign

Web.com

Google.com

Amazon.com

... x.com
Burning question ...

• Can we solve the PKI problem with better crypto?
Identity Based Cryptography

• What if your email address was your public key?
  ‣ E.g., $E(mcdaniel@gmail.com, \text{data}) = \text{ciphertext}$?
  ‣ E.g., Verify( signature, mcdaniel@gmail.com )

• 1984 - Shamir asked for such a system, but it (largely) remained out of reach until Boneh/Franklin 2001
  ‣ The public key is any arbitrary key
  ‣ Based on “Weil pairings” -- a new cryptographic device with lots and lots of uses (IBE among them)

• Advances from theory community, few systems
IBE System

• Functionally, you receive your private key from a **trusted third party** who is responsible for generating all keys in the system.

• Thereafter you (and others) can use the system as if you generated the private key yourself.

• Advantages
  ‣ No public key distribution
  ‣ No name binding problems (?)
  ‣ Key space flexibility
  ‣ Others?

1) user@gmail.com
2) key(user@gmail.com)
3) E(user@gmail.com, data)
Basic IBE Construction

• **Setup** (generate by TTP)
  
  \[ \begin{align*}
  \text{Global Parameters} & = G \\
  \text{Master Key} & = K_G
  \end{align*} \]

• **Extract** (by TTP for user, sting “str”)
  
  \[ \text{Extract}(G, K_G, Str) = K_{Str}^- \]

• **Encrypt** (by user)
  
  \[ E(G, Str, data) = \text{ciphertext} \]

• **Decrypt** (by user)
  
  \[ D(G, K_{Str}^-, \text{ciphertext}) = data \]
IBE Reality

• Many thought that IBE would lead to a revolution in public key system (solve PKI problems), it didn’t.

• Why - IBE moves the problems around
  ‣ Is there any **TTP** that everyone trusts?
  ‣ String ambiguity is still a problem? (John Robinson?)
  ‣ Revocation is still a problem (potentially worse)
  ‣ ... (see 10 reasons above)

• Fundamentally
  ‣ IBE really does not solve the CA problem, as the TTP is fulfilling that role.
  ‣ Having strings instead of obscure numbers does not get at the problems with PKI ...
  ‣ Existence of certificates is not really the problem ...