

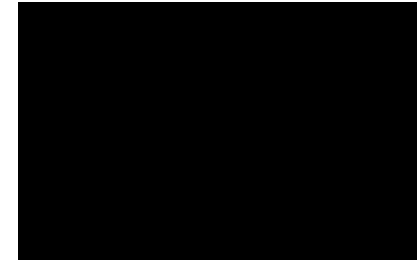
Introduction to Cryptography Lecture 9

Public Key Infrastructure (PKI),
hash chains, hash trees

Benny Pinkas

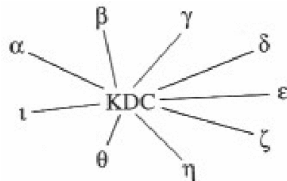
Key Infrastructure for symmetric key encryption

- Each user has a shared key with each other user
 - A total of $n(n-1)/2$ keys
 - Each user stores $n-1$ keys



Key Distribution Center (KDC)

- The KDC shares a symmetric key K_u with every user u
- Using this key they can establish a trusted channel
- When u wants to communicate with v
 - u sends a request to the KDC
 - The KDC
 - authenticates u
 - generates a key K_{uv} to be used by u and v
 - sends $Enc(K_u, K_{uv})$ to u , and $Enc(K_v, K_{uv})$ to v



Key Distribution Center (KDC)

- Advantages:
 - A total of n keys, one key per user.
 - easier management of joining and leaving users.
- Disadvantages:
 - The KDC can impersonate anyone
 - The KDC is a single point for failure, for both
 - security,
 - and quality of service
- Multiple copies of the KDC
 - More security risks
 - But better availability

Certification Authorities (CA)

- Public key technology requires every user to remember its private key, and to have access to other users' public key
- How can the user verify that a public key PK_v corresponds to user v ?
 - What can go wrong otherwise?
- A simple solution:
 - A trusted public repository of public keys and corresponding identities
 - Doesn't scale up
 - Requires online access per usage of a new public key

December26, 2004

Introduction to Cryptography, Benny Pinkas

page 5

Certification Authorities (CA)

- The Certificate Authority (CA) is trusted party.
- All users have a copy of the public key of the CA
- The CA signs Alice's digital certificate. A simplified certificate is of the form *(Alice, Alice's public key)*.
- When we get Alice's certificate, we
 - Examine the identity in the certificate
 - Verify the signature
 - Use the public key given in the certificate to
 - Encrypt messages to Alice
 - Or, verify signatures of Alice
- The certificate can be sent by Alice without any interaction with the CA.

December26, 2004

Introduction to Cryptography, Benny Pinkas

page 6

Certification Authorities (CA)

- Unlike KDCs, the CA does not have to be online to provide keys to users
 - It can therefore be better secured than a KDC
 - The CA does not have to be available all the time
- Users only keep a single public key – of the CA
- The certificates are not secret. They can be stored in a public place.
- When a user wants to communicate with Alice, it can get her certificate from either her, the CA, or a public repository.
- A compromised CA
 - can mount active attacks (certifying keys as being Alice's)
 - but it cannot decrypt conversations.

December26, 2004

Introduction to Cryptography, Benny Pinkas

page 7

Certification Authorities (CA)

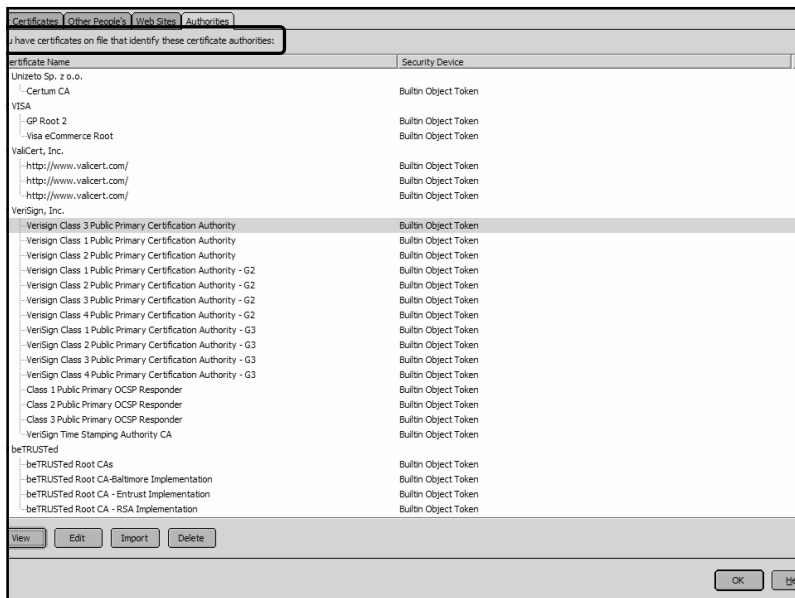
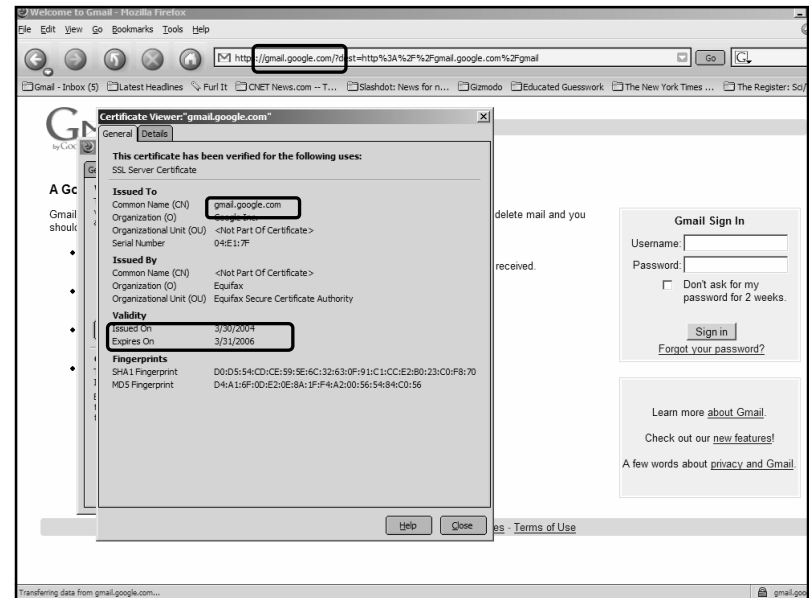
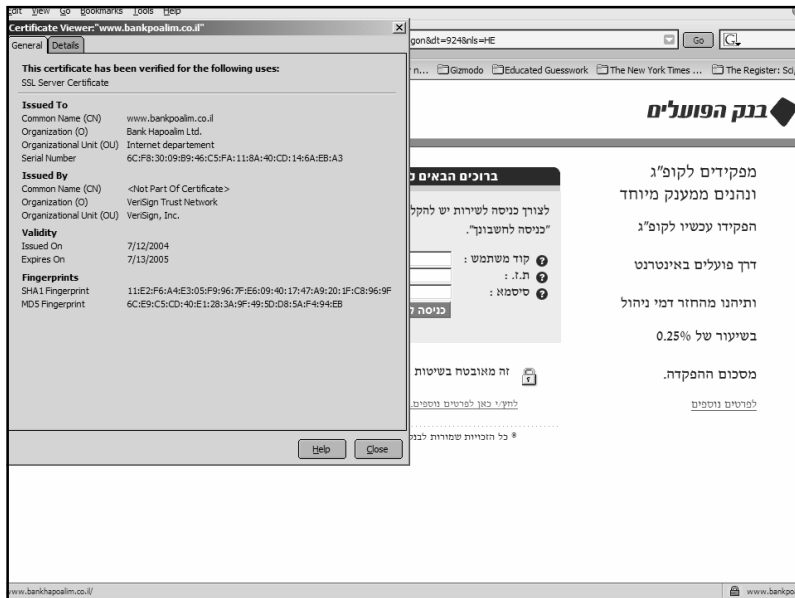
- For example.
 - To connect to a secure web site using SSL or TLS, we send an `https://` command
 - The web site sends back a public key⁽¹⁾, and a certificate.
 - Our browser
 - Checks that the certificate belongs to the url we're visiting
 - Checks the expiration date
 - Checks that the certificate is signed by a CA whose public key is known to the browser
 - Checks the signature
 - If everything is fine, it chooses a session key and sends it to the server encrypted with RSA using the server's public key

⁽¹⁾ This is a very simplified version of the actual protocol.

December26, 2004

Introduction to Cryptography, Benny Pinkas

page 8



Certificates

- A certificate usually contains the following information
 - Owner's name
 - Owner's public key
 - Encryption/signature algorithm
 - Name of the CA
 - Serial number of the certificate
 - Expiry date of the certificate
 - ...
- Your web browser contains the public keys of some CAs
- A web site identifies itself by presenting a certificate which is signed by a chain starting at one of these CAs

December 26, 2004
Introduction to Cryptography, Benny Priskas
page 12

Public Key Infrastructure (PKI)

- The goal: build trust on a global level
- Running a CA:
 - If people trust you to vouch for other parties, everyone needs you.
 - A license to print money
 - But,
 - The CA should limit its responsibilities, buy insurance...
 - It should maintain a high level of security
 - Bootstrapping: how would everyone get the CA's public key?

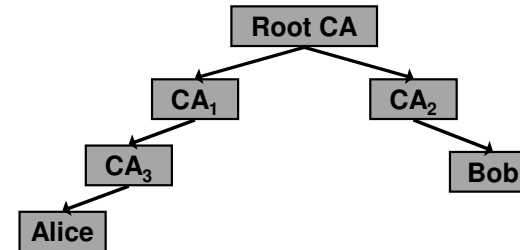
December26, 2004

Introduction to Cryptography, Benny Pinkas

page 13

Public Key Infrastructure (PKI)

- Monopoly: a single CA vouches for all public keys
- Monopoly + delegated CAs:
 - top level CA can issue certificates for other CAs
 - Certificates of the form
 - [(Alice, PK_A)_{CA3}, (CA3, PK_{CA3})_{CA1}, (CA1, PK_{CA1})_{TOP-CA}]

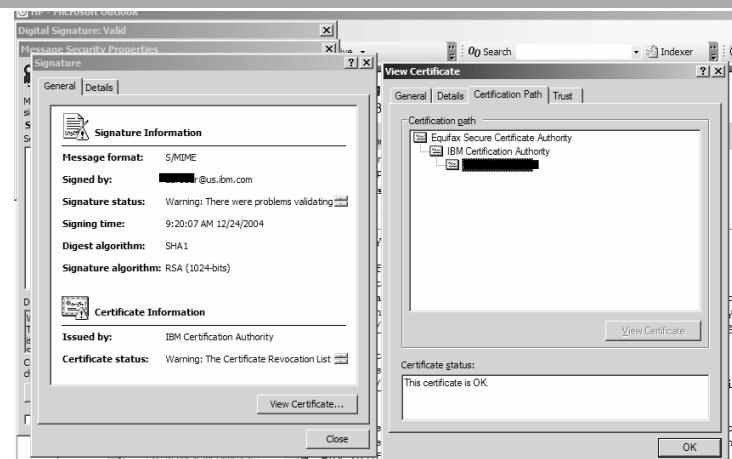


December26, 2004

Introduction to Cryptography, Benny Pinkas

page 14

Certificate chain



December26, 2004

Introduction to Cryptography, Benny Pinkas

page 15

Public Key Infrastructure

- Oligarchy
 - Multiple trust anchors (top level CAs)
 - Pre-configured in software
 - User can add/remove CAs
- Top-down with name constraints
 - Like monopoly + delegated CAs
 - But every delegated CA has a predefined portion of the name space (il, ac.il, haifa.ac.il, cs.haifa.ac.il)
 - More trustworthy

December26, 2004

Introduction to Cryptography, Benny Pinkas

page 16

Revocation

- Revocation is a key component of PKI
 - Each certificate has an expiry date
 - But certificates might get stolen, employees might leave companies, etc.
 - Certificates might therefore need to be revoked before their expiry date
 - Before using a certificate we must verify that it has not been revoked

Certificate Revocation Lists (CRLs)

- A revocation agency (RA) issues a list of revoked certificates (i.e., "bad" certificates)
 - The list is updated and published regularly (e.g. daily)
 - Users must consult the most recent CRL in addition to checking the expiry date
- Advantages: simple.
- Drawbacks:
 - Scalability. CRLs can be huge. There is no short proof that a certificate is valid. (The proof is global, rather than individual.)
 - There is a vulnerability windows between a compromise of certificate and the next publication of a CRL.
 - Need a reliable way of distributing CRLs.
- Improving scalability using "delta CRLs"

Explicit revocation: OCSP

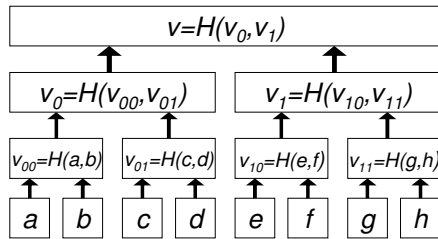
- OCSP (Online Certificate Status Protocol)
 - RFC 2560, June 1999.
- OCSP can be used in place, or in addition, to CRLs
- OCSP sends a request for certificate status information.
 - The server sends back a response of "current", "expired," or "unknown".
 - The response is signed (by the CA, or a Trusted Responder, or An Authorized Responder certified by the CA).
- Provides instantaneous status of certificates
 - Overcomes the chief limitation of CRL: the fact that updates must be frequently downloaded to keep the list current

Certificate Revocation System (CRS)

- Certificate Revocation System (Micali'96)
- Uses a hash chain
 - The certificate includes $Y_{365} = f^{365}(Y_0)$, and $N = f(N_0)$. f is one-way.
 - On day d ,
 - If the certificate is valid, then $Y_{365-d} = f^{365-d}(Y_0)$ is sent by the CA to the certificate holder or to a directory.
 - If the certificate is invalid, the value N_0 is published.
 - The certificate receiver uses the daily value ($f^{365-d}(Y_0)$) to verify that the certificate is still valid. (how?)
- Disadvantage:
 - Daily overhead, even when a cert is not revoked.

Merkle Hash Tree

- A method of hashing together n values, x_1, \dots, x_n , such that
 - The result is a single hash value
 - For any x_i , it is possible to prove that it appeared in the original list, using a proof of length $O(\log n)$.



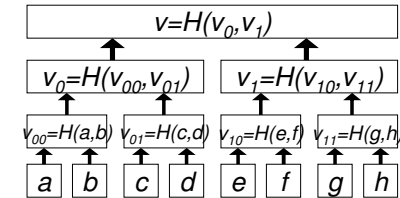
December26, 2004

Introduction to Cryptography, Benny Pinkas

page 21

Merkle Hash Tree

- H is a collision intractable hash function
- Any change to a leaf results in a change to the root
- To sign the set of values it is sufficient to sign the root (a single signature instead of n).
- How do we verify that an element appeared in the signed set?



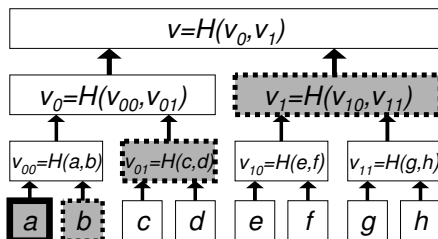
December26, 2004

Introduction to Cryptography, Benny Pinkas

page 22

Verifying that a appears in the signed set

- Provide a 's leaf, and the siblings of the nodes in the path from a to the root. ($O(\log n)$ values)
- The verifier can use H to compute the values of the nodes in the path from the leaf to the root.
- It then compares the computed root to the signed value



December26, 2004

Introduction to Cryptography, Benny Pinkas

page 23

Improving the overhead of CRS

- Originally (for a year long certificate)
 - the certificate includes $f^{365}(Y_0)$
 - On day d , certificate holder obtains $f^{365-d}(Y_0)$
 - The certificate receiver computes $f^{365}(Y_0)$ from $f^{365-d}(Y_0)$ by invoking $f()$ d times.
- Slight improvement:
 - The CA assigns a different leaf for every day, constructs a hash tree, and signs the root.
 - On day d , it releases node d and the siblings of the path from it to the root.
 - This is the proof that the certificate is valid on day d
 - The overhead of verification is $O(\log 365)$.

December26, 2004

Introduction to Cryptography, Benny Pinkas

page 24

Certificate Revocation Tree (CRT) [Kocher]

- A CRT is a hash tree with leaves corresponding to statements about ranges of certificates
 - Statements describe regions of certificate ids, in which only the smallest id is revoked.
 - For example: “if $100 \leq \text{id} < 234$, then cert is revoked iff $\text{id}=100$ ”.
 - Each certificate matches exactly one statement.
 - The statements are the leaves of a signed hash tree.
 - To examine the state of a certificate we retrieve the statement for the corresponding region.
 - A single hash tree is used for all certs.
 - Advantage: a short proof for the status of a certificate.
 - Drawback: the entire hash tree must be updated daily.