Introduction to Cryptography
Lecture 12

Public Key Infrastructure (PKI), secret sharing

Benny Pinkas

Trusting public keys

- Public key technology requires every user to remember its private key, and to have access to other users' public keys
- 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

Certification Authorities (CA)

- A method to bootstrap trust
  - Start by trusting a single party and knowing its public key
  - Use this to establish trust with other parties (and associate them with public keys)

- 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)*.

Certification Authorities (CA)

- 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 online interaction with the CA.

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

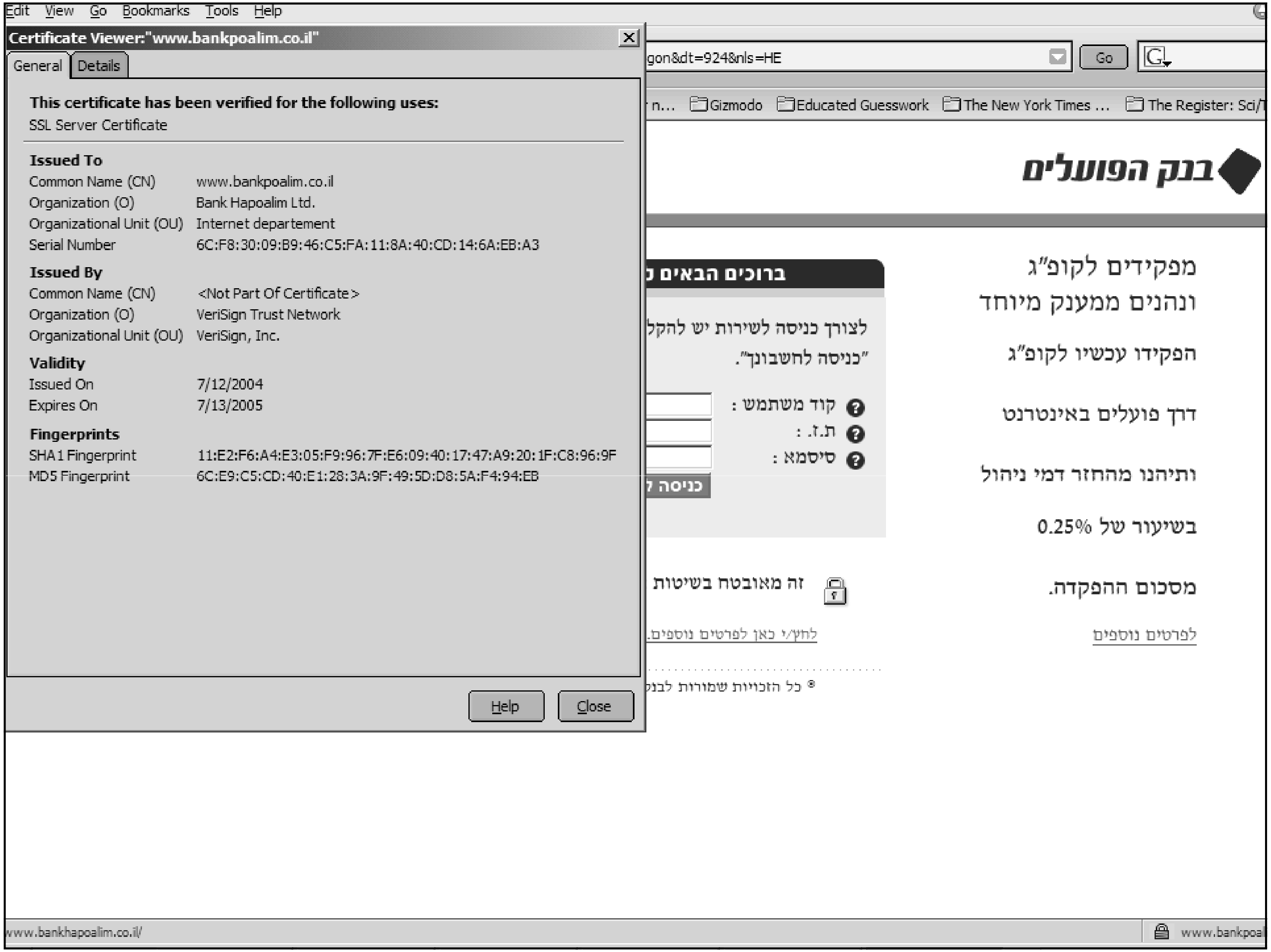
Certification Authorities (CA)

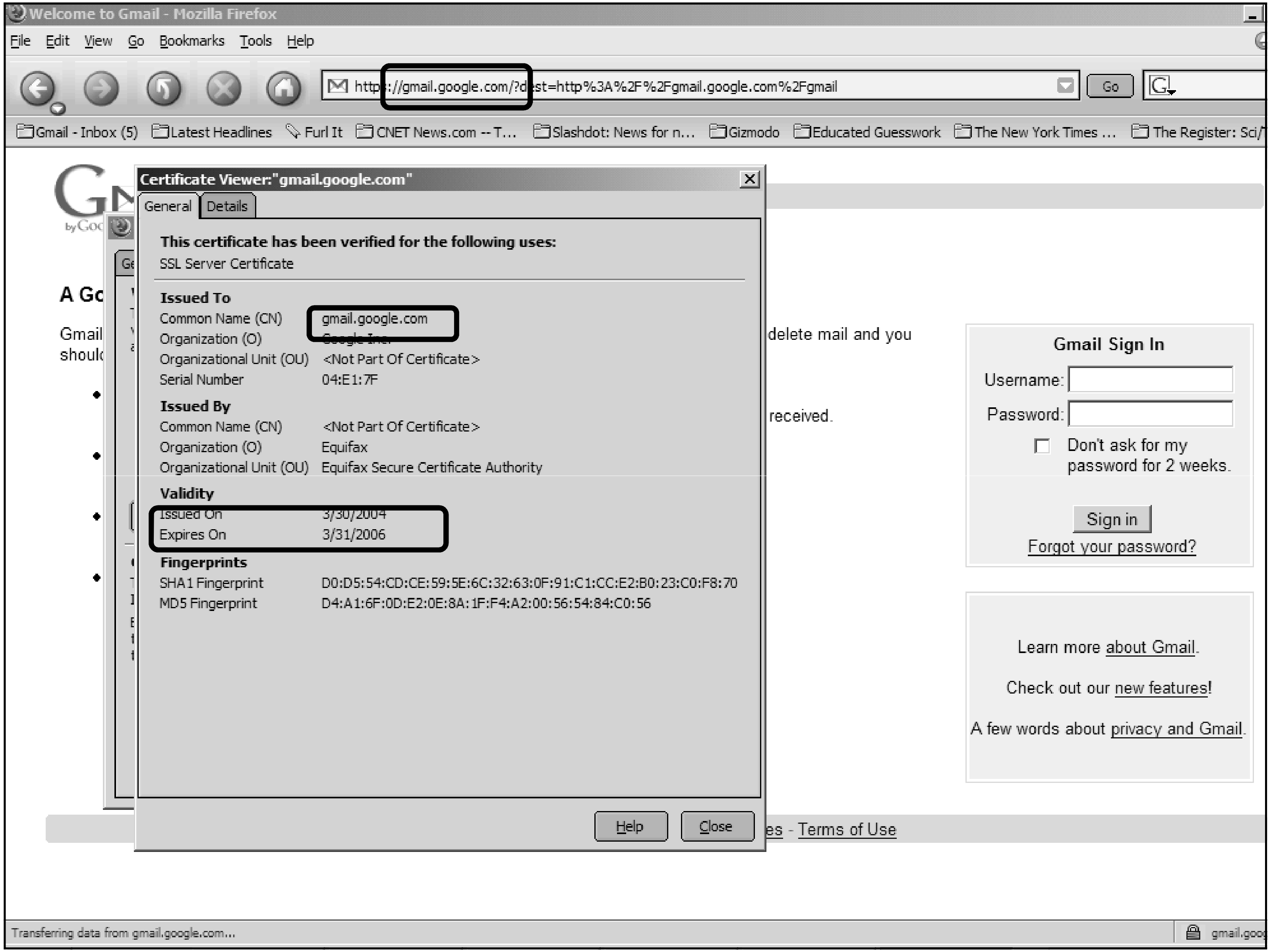
- Unlike KDCs, the CA does not have to be online to provide keys to users
  – It can therefore be better secured than a KDC
  – The CA does not have to be available all the time
- Users only keep a single public key – of the CA
- The certificates are not secret. They can be stored in a public place.
- 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.

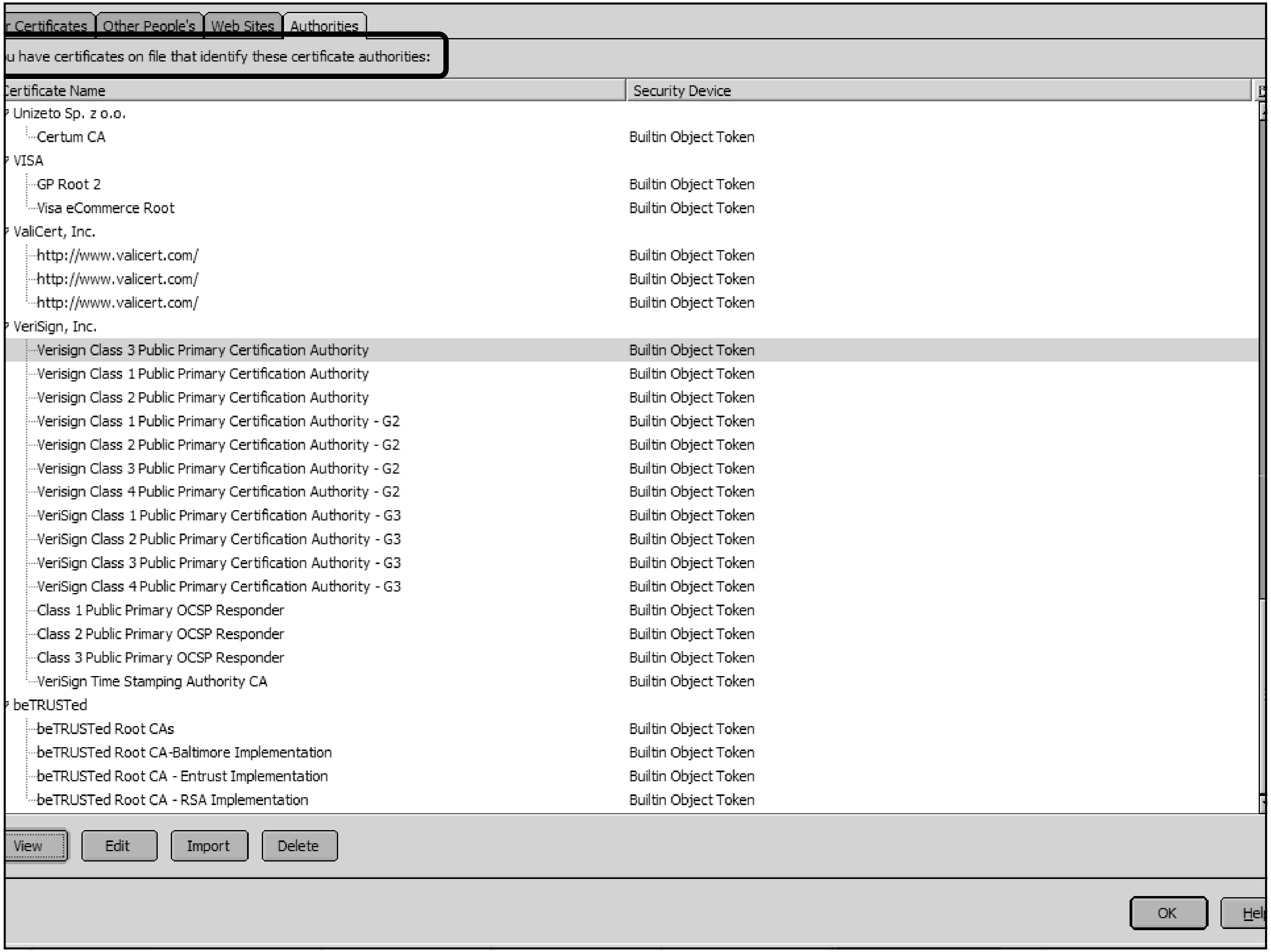
Certificates in Internet browsing

- Our browser can identify web sites if their certificates are signed by certificate authorities which are trusted by the browser.

- Last time I counted, Firefox listed more than 70 certificate authorities which it trusts.


Edit View Go Bookmarks Tools Help
Certificate Viewer:"www.bankpoalim.co.il"
General Details
This certificate has been verified for the following uses:
SSL Server Certificate
Issued To
Common Name (CN) www.bankpoalim.co.il
Organization (O) Bank Hapoalim Ltd.
Organizational Unit (OU) Internet departement
Serial Number 6C:F8:30:09:B9:46:C5:FA:11:8A:40:CD:14:6A:EB:A3
Issued By
Common Name (CN) <Not Part Of Certificate>
Organization (O) VeriSign Trust Network
Organizational Unit (OU) VeriSign, Inc.
Validity
Issued On 7/12/2004
Expires On 7/13/2005
Fingerprints
SHA1 Fingerprint 11:E2:F6:A4:E3:05:F9:96:7F:E6:09:40:17:47:A9:20:1F:C8:96:9F
MD5 Fingerprint 6C:E9:C5:CD:40:E1:28:3A:9F:49:5D:D8:5A:F4:94:EB
Help Close
gon&dt=924&nls=HE
Go
Gizmodo
Educated Guesswork
The New York Times ...
The Register: Sci/T
בנק הפועלים
ברוכים הבאים
לצורך כניסה לשירות יש להקל
"כניסה לחשבונך".
קוד משתמש :
ת.ז. :
סיסמא :
כניסה ל
זה מאובטח בשיטות
לחץ/י כאן לפרטים נוספים.
כל הזכויות שמורות לבנ
מפקידים לקופ"ג
ונהנים ממענק מיוחד
הפקידו עכשיו לקופ"ג
דרך פועלים באינטרנט
ותיהנו מהחזר דמי ניהול
בשיעור של 0.25%
מסכום ההפקדה.
לפרטים נוספים
www.bankhapoalim.co.il/
www.bankpoal


Welcome to Gmail - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://gmail.google.com/?dest=http%3A%2F%2Fgmail.google.com%2Fgmail
Go
Gmail - Inbox (5) Latest Headlines Furl It CNET News.com -- T... Slashdot: News for n... Gizmodo Educated Guesswork The New York Times ... The Register: Sci/T
Certificate Viewer:"gmail.google.com"
General Details
This certificate has been verified for the following uses:
SSL Server Certificate
Issued To
Common Name (CN) gmail.google.com
Organization (O) Google Inc.
Organizational Unit (OU) <Not Part Of Certificate>
Serial Number 04:E1:7F
Issued By
Common Name (CN) <Not Part Of Certificate>
Organization (O) Equifax
Organizational Unit (OU) Equifax Secure Certificate Authority
Validity
Issued On 3/30/2004
Expires On 3/31/2006
Fingerprints
SHA1 Fingerprint D0:D5:54:CD:CE:59:5E:6C:32:63:0F:91:C1:CC:E2:B0:23:C0:F8:70
MD5 Fingerprint D4:A1:6F:0D:E2:0E:8A:1F:F4:A2:00:56:54:84:C0:56
Help Close
delete mail and you
received.
Gmail Sign In
Username:
Password:
Don't ask for my password for 2 weeks.
Sign in
Forgot your password?
Learn more about Gmail.
Check out our new features!
A few words about privacy and Gmail.
Terms of Use
Transferring data from gmail.google.com...


r Certificates
Other People's
Web Sites
Authorities
u have certificates on file that identify these certificate authorities:
Certificate Name
Security Device
Unizeto Sp. z o.o.
Certum CA
Builtin Object Token
VISA
GP Root 2
Builtin Object Token
Visa eCommerce Root
Builtin Object Token
ValiCert, Inc.
http://www.valicert.com/
Builtin Object Token
http://www.valicert.com/
Builtin Object Token
http://www.valicert.com/
Builtin Object Token
VeriSign, Inc.
Verisign Class 3 Public Primary Certification Authority
Builtin Object Token
Verisign Class 1 Public Primary Certification Authority
Builtin Object Token
Verisign Class 2 Public Primary Certification Authority
Builtin Object Token
Verisign Class 1 Public Primary Certification Authority - G2
Builtin Object Token
Verisign Class 2 Public Primary Certification Authority - G2
Builtin Object Token
Verisign Class 3 Public Primary Certification Authority - G2
Builtin Object Token
Verisign Class 4 Public Primary Certification Authority - G2
Builtin Object Token
VeriSign Class 1 Public Primary Certification Authority - G3
Builtin Object Token
VeriSign Class 2 Public Primary Certification Authority - G3
Builtin Object Token
VeriSign Class 3 Public Primary Certification Authority - G3
Builtin Object Token
VeriSign Class 4 Public Primary Certification Authority - G3
Builtin Object Token
Class 1 Public Primary OCSP Responder
Builtin Object Token
Class 2 Public Primary OCSP Responder
Builtin Object Token
Class 3 Public Primary OCSP Responder
Builtin Object Token
VeriSign Time Stamping Authority CA
Builtin Object Token
beTRUSTed
beTRUSTed Root CAs
Builtin Object Token
beTRUSTed Root CA-Baltimore Implementation
Builtin Object Token
beTRUSTed Root CA - Entrust Implementation
Builtin Object Token
beTRUSTed Root CA - RSA Implementation
Builtin Object Token
View
Edit
Import
Delete
OK

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

An example of an X.509 certificate

```
Certificate:
  Data:
    Version: 1 (0x0)
    Serial Number: 7829 (0x1e95)
    Signature Algorithm: md5WithRSAEncryption
    Issuer: C=ZA, ST=Western Cape, L=Cape Town, O=Thawte Consulting cc,
      OU=Certification Services Division, CN=Thawte Server
      CA/emailAddress=server-certs@thawte.com
    Validity
         Not Before: Jul 9 16:04:02 1998 GMT
         Not After : Jul 9 16:04:02 1999 GMT
    Subject: C=US, ST=Maryland, L=Pasadena, O=Brent Baccala, OU=FreeSoft,
      CN=www.freesoft.org/emailAddress=baccala@freesoft.org
    Subject Public Key Info:
         Public Key Algorithm: rsaEncryption
         RSA Public Key: (1024 bit)
         Modulus (1024 bit): 00:b4:31:98:0a:c4:bc:62:c1:88:aa:dc:b0:c8:bb:
           33:35:19:d5:0c:64:b9:3d:41:b2:96:fc:f3:31:e1:
           66:36:d0:8e:56:12:44:ba:75:eb:e8:1c:9c:5b:66:
           70:33:52:14:c9:ec:4f:91:51:70:39:de:53:85:17:
           16:94:6e:ee:f4:d5:6f:d5:ca:b3:47:5e:1b:0c:7b:
           c5:cc:2b:6b:c1:90:c3:16:31:0d:bf:7a:c7:47:77:
           8f:a0:21:c7:4c:d0:16:65:00:c1:0f:d7:b8:80:e3:
           d2:75:6b:c1:ea:9e:5c:5c:ea:7d:c1:a1:10:bc:b8: e8:35:1c:9e:27:52:7e:41:8f
         Exponent: 65537 (0x10001)
  Signature Algorithm: md5WithRSAEncryption
    93:5f:8f:5f:c5:af:bf:0a:ab:a5:6d:fb:24:5f:b6:59:5d:9d:
      92:2e:4a:1b:8b:ac:7d:99:17:5d:cd:19:f6:ad:ef:63:2f:92:…
```

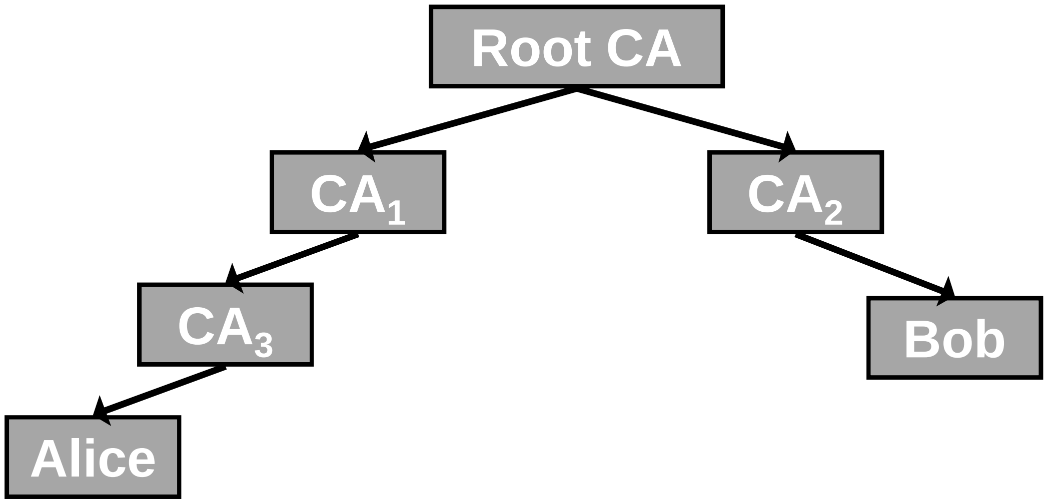
Public Key Infrastructure (PKI)

- The goal: build trust on a global level

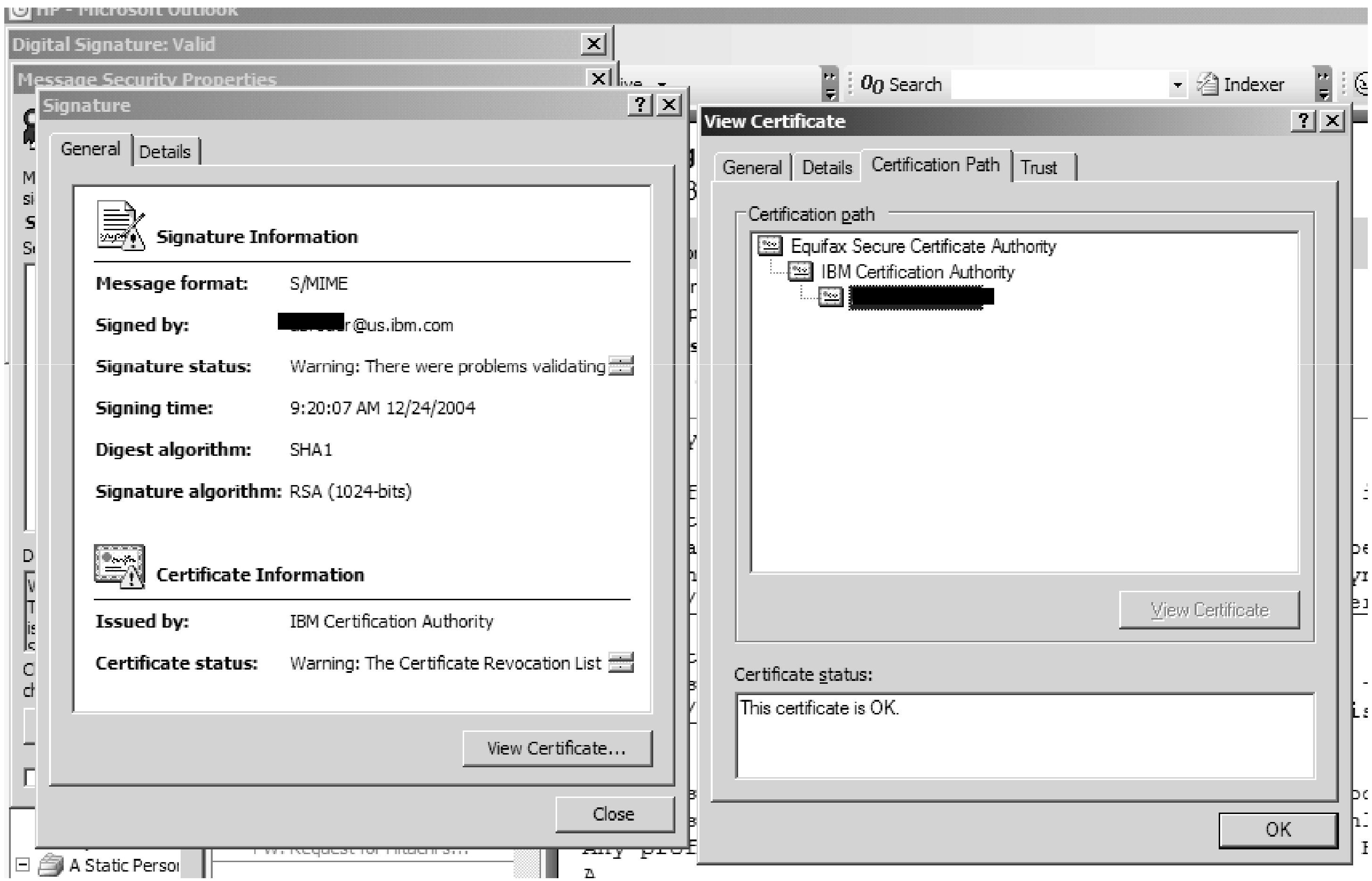
- Running a CA:
  - If people trust you to vouch for other parties, everyone needs you.
  - A license to print money
  - But,
    - The CA should limit its responsibilities, buy insurance…
    - It should maintain a high level of security
    - Bootstrapping: how would everyone get the CA’s public key?

Public Key Infrastructure (PKI)

- Monopoly: a single CA vouches for all public keys
  - Mostly suitable for enterprises.
- Monopoly + delegated CAs:
  - top level CA can issue *special* certificates for other CAs
  - Certificates of the form
    - [$(\text{Alice}, PK_A)_{CA3}$, $(CA3, PK_{CA3})_{CA1}$, $(CA1, PK_{CA1})_{ROOT\text{-}CA}$]



Certificate chain



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
  - New problem: before using a certificate we must verify that it has not been revoked
    - Often the most costly aspect of running a large scale public key infrastructure (PKI)
    - How can this be done efficiently?
    - (we won’t discuss this issue this year)

SSL / TLS

SSL/TLS

- General structure of secure HTTP connections
  - 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.

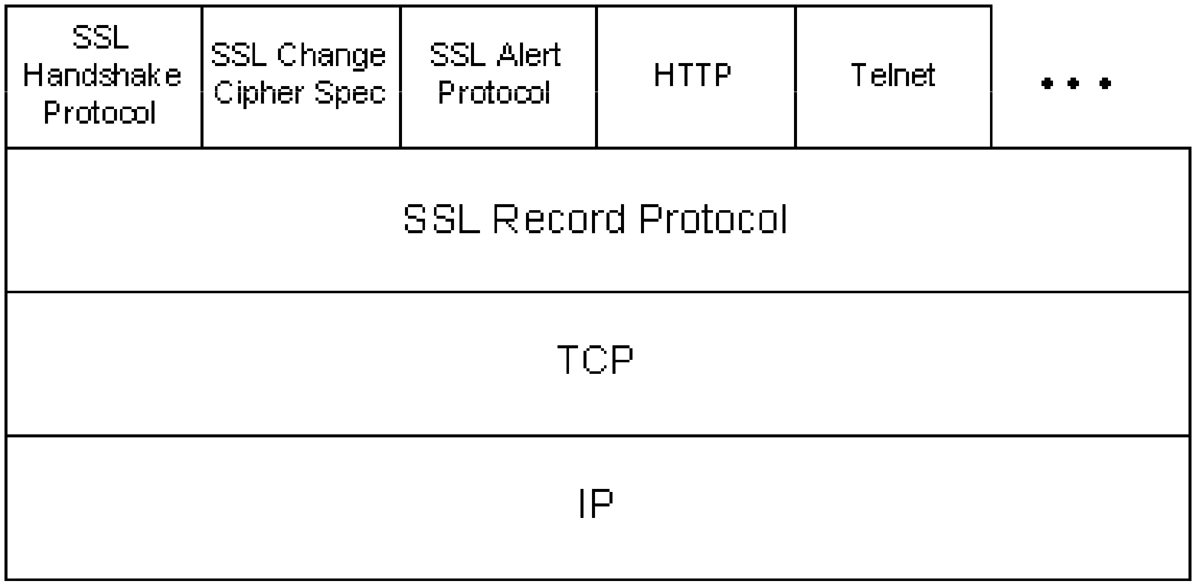
SSL/TLS

- SSL (Secure Sockets Layer)
  - SSL v2
    - Released in 1995 with Netscape 1.1
    - A flaw found in the key generation algorithm
  - SSL v3
    - Improved, released in 1996
    - Public design process
- TLS (Transport Layer Security)
  - IETF standard, RFC 2246

- Common browsers support all these protocols

SSL Protocol Stack

- SSL/TLS operates over TCP, which ensures reliable transport.
- Supports any application protocol (usually used with http).



SSL/TLS Overview

- Handshake Protocol - establishes a session
  - Agreement on algorithms and security parameters
  - Identity authentication
  - Agreement on a key
  - Report error conditions to each other

- Record Protocol - Secures the transferred data
  - Message encryption and authentication

- Alert Protocol – Error notification (including “fatal” errors).

- Change Cipher Protocol – Activates the pending crypto suite

Simplified SSL Handshake

Client

Server

I want to talk, ciphers I support, R_C

Certificate (PK_{Server}), cipher I choose, R_S

$\{S\}_{PKserver}$, {keyed hash of handshake message}

compute $K = f(S, R_C, R_S)$

{keyed hash of handshake message}

compute $K = f(S, R_C, R_S)$

Data protected by keys derived from K

A typical run of a TLS protocol

- $C \Rightarrow S$
  - ClientHello.protocol.version = “TLS version 1.0”
  - ClientHello.random = T_C, N_C
  - ClientHello.session_id = “NULL”
  - ClientHello.crypto_suite = “RSA: encryption.SHA-1:HMAC”
  - ClientHello.compression_method = “NULL”
- $S \Rightarrow C$
  - ServerHello.protocol.version = “TLS version 1.0”
  - ServerHello.random = T_S, N_S
  - ServerHello.session_id = “1234”
  - ServerHello.crypto_suite = “RSA: encryption.SHA-1:HMAC”
  - ServerHello.compression_method = “NULL”
  - ServerCertificate = pointer to server’s certificate
  - ServerHelloDone

Some additional issues

- More on S $\Rightarrow$ C
  - The ServerHello message can also contain Certificate Request Message
  - I.e., server may request client to send its certificate
  - Two fields: certificate type and acceptable CAs

- Negotiating crypto suites
  - The crypto suite defines the encryption and authentication algorithms and the key lengths to be used.
  - ~30 predefined standard crypto suites
  - Selection (SSL v3): Client proposes a set of suites. Server selects one.

Key generation

- Key computation:
  - The key is generated in two steps:
  - *pre-master secret S* is exchanged during handshake
  - *master secret K* is a 48 byte value calculated using pre-master secret and the random nonces

- Session vs. Connection: a *session* is relatively long lived. Multiple TCP *connections* can be supported under the same SSL/TSL connection.
- For each connection: 6 keys are generated from the master secret *K* and from the nonces. (For each direction: encryption key, authentication key, IV.)

TLS Record Protocol

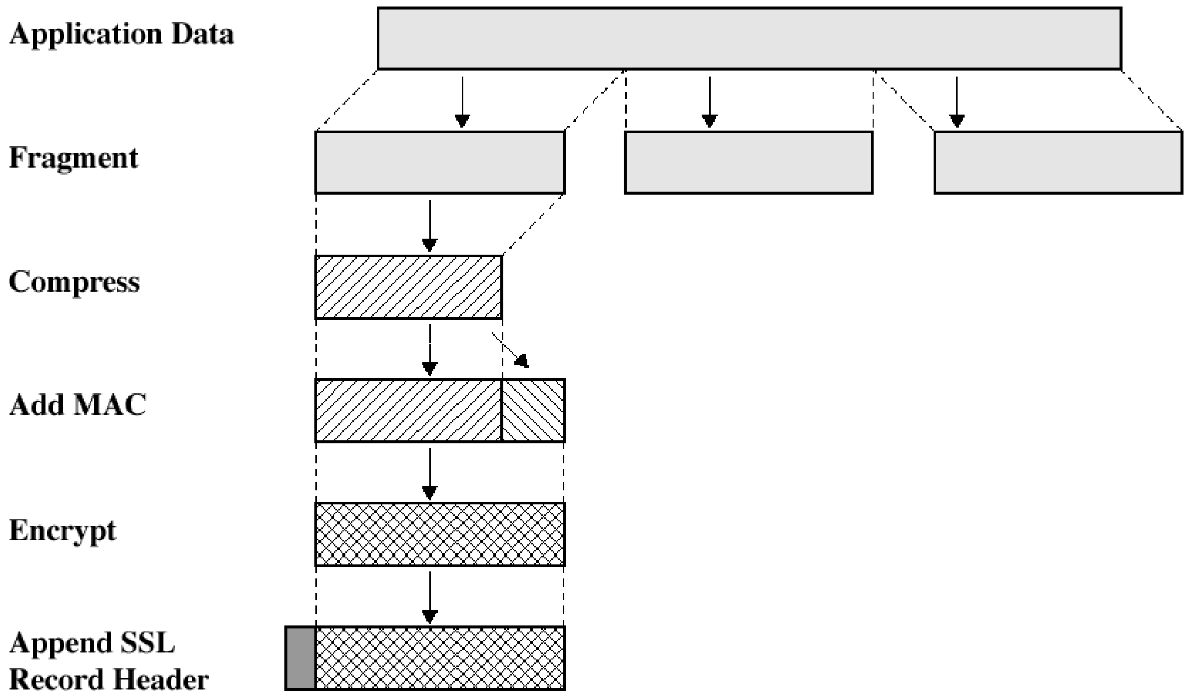


Figure 17.3 SSL Record Protocol Operation

Secret sharing

Secret Sharing

- 3-out-of-3 secret sharing:
  - Three parties, A, B and C.
  - Secret *S.*
  - No two parties should know *anything* about *S*, but all three together should be able to retrieve it.
- In other words
  - A + B + C $\Rightarrow$ S
  - But,
    - A + B $\not\Rightarrow$ S
    - A + C $\not\Rightarrow$ S
    - B + C $\not\Rightarrow$ S

Secret Sharing

- 3-out-of-3 secret sharing:
- How about the following scheme:
  - Let $S=s_1 s_2 \ldots s_m$ be the bit representation of *S. (m* is a multiple of 3)
    - Party A receives $s_1,\ldots,s_{m/3}$.
    - Party B receives $s_{m/3+1},\ldots,s_{2m/3}$.
    - Party C receives $s_{2m/3+1},\ldots,s_m$.
  - All three parties can recover *S.*

  - Why doesn't this scheme satisfy the definition of secret sharing?
  - Why does each share need to be as long as the secret?

Secret Sharing

- Solution:
  - Define *shares* for A,B,C in the following way
  - (S_A, S_B, S_C) is a random triple, subject to the constraint that
    - $S_A \oplus S_B \oplus S_C = S$
    - *or,* S_A and S_B are random, and $S_C = S_A \oplus S_B \oplus S$.

- What if it is required that any one of the parties should be able to compute *S?*
  - Set $S_A = S_B = S_C = S$

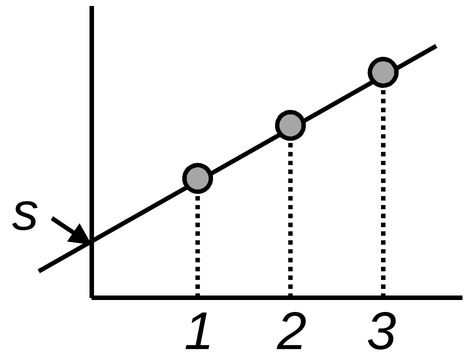
- What if each pair of the three parties should be able to compute *S?*

t-out-of-*n* secret sharing

- Provide shares to *n* parties, satisfying
  - Recoverability: any *t* shares enable the reconstruction of the secret.
  - Secrecy: any *t-1* shares reveal nothing about the secret.

- We saw *1*-out-of-*n* and *n*-out-of-*n* secret sharing.

- Consider *2*-out-of-*n* secret sharing.
  - Define a line which intersects the Y axis at *S*
  - The shares are points on the line
  - Any two shares define *S*
  - A single share reveals nothing



t-out-of-*n* secret sharing

- Fact: Let F be a field. Any $d+1$ pairs (a_i , b_i) define a unique polynomial P of degree ≤ d, s.t. $P(a_i)=b_i$. (assuming $d < |F|$).

- Shamir's secret sharing scheme:
  - Choose a large prime and work in the field Zp.
  - The secret S is an element in the field.
  - Define a polynomial P of degree $t-1$ by choosing random coefficients $a_1,\ldots,a_{t-1}$ and defining $P(x) = a_{t-1}x^{t-1}+\ldots+a_1x+\underline{S}$.
  - The share of party j is $(j, P(j))$.

t-out-of-*n* secret sharing

- Reconstruction of the secret:
  - Assume we have $P(x_1),\ldots,P(x_t)$.
  - Use Lagrange interpolation to compute the unique polynomial of degree ≤ $t-1$ which agrees with these points.
  - Output the free coefficient of this polynomial.

- Lagrange interpolation
  - $P(x) = \sum_{i=1..t} P(x_i)\cdot L_i(x)$
  - where $L_i(x)=\prod_{j\neq i}(x-x_j) / \prod_{j\neq i}(x_i-x_j)$
  - (Note that $L_i(x_i)=1$, $L_i(x_j)=0$ for $j\neq i$.)

  - I.e., $S = \sum_{i=1..t} P(x_i) \cdot \prod_{j\neq I} x_j \,/ \prod_{j\neq i}(x_i - x_j)$

Properties of Shamir's secret sharing

- Perfect secrecy: Any $t\text{-}1$ shares give no information about the secret: $\Pr(secret=s \mid P(1),\ldots,P(t\text{-}1)) = \Pr(secret=s)$. (Security is not based on any assumptions.)
- Proof:
  - Let's get intuition from 2-out-of-n secret sharing
  - The polynomial is generated by choosing a random coefficient a and defining $P(x)= a \cdot x+s$.
  - Suppose that the adversary knows $P(x_1)=a \cdot x_1+s$.
  - For any value of s, the value of a is uniquely defined by $P(x_1)$ and s.
  - Namely, there is a one-to-one correspondence between s and a.
  - Since a is uniformly distributed, so is the value of $P(x_1)$ (any assignment to a results in exactly one value of $P(x_1)$).
    - Therefore $P(x_1)$ does not reveal any information about s.

Properties of Shamir's secret sharing

- Perfect secrecy: Any t-1 shares give no information about the secret: $\Pr(secret{=}s \mid P(1),\ldots,P(t\text{-}1)) = \Pr(secret{=}s)$. (Security is not based on any assumptions.)
- Proof:
  - The polynomial is generated by choosing a random polynomial of degree t-1, subject to $P(0)$=secret.
  - Suppose that the adversary knows the shares $P(x_1),\ldots,P(x_{t-1})$.
  - The values of $P(x_1),\ldots,P(x_{t-1})$ are defined by t-1 linear equations of $a_1,\ldots,a_{t-1}$, s.
    - $P(x_i) = \Sigma_{i=1,\ldots,t-1} (x_i)^j a_j + s.$

Properties of Shamir's secret sharing

- Proof (cont.):
  - The values of $P(x_1),\dots,P(x_{t-1})$ are defined by $t-1$ linear equations of $a_1,\dots,a_{t-1}, s.$
    - $P(x_i) = \Sigma_{j=1,\dots,t-1} (x_i)^j a_j + s.$
  - For any possible value of s, there is a exactly one set of values of $a_1,\dots,a_{t-1}$ which gives the values $P(x_1),\dots,P(x_{t-1})$.
    - This set of $a_1,\dots,a_{t-1}$ can be found by solving a linear system of equations.
  - Since $a_1,\dots,a_{t-1}$ *are* uniformly distributed, so are the values of $P(x_1),\dots,P(x_{t-1})$.
    - Therefore $P(x_1),\dots,P(x_{t-1})$ reveal nothing about s.

Additional properties of Shamir's secret sharing

- Ideal size: Each share is the same size as the secret.
- Extendable: Additional shares can be easily added.
- Flexible: different weights can be given to different parties by giving them more shares.
- Homomorphic property: Suppose $P(1),\ldots,P(n)$ are shares of S, and $P'(1),\ldots,P'(n)$ are shares of S', then $P(1)+P'(1),\ldots,P(n)+P'(n)$ are shares for $S+S'$.

General secret sharing

- P is the set of users (say, n users).
- $A \in \{1,2,\ldots,n\}$ is an authorized subset if it is authorized to access the secret.
- Γ is the set of authorized subsets.
- For example,
  - $P = \{1,2,3,4\}$
  - Γ = *Any set containing one of* $\{ \{1,2,4\}, \{1,3,4,\}, \{2,3\} \}$
  - Not supported by threshold secret sharing

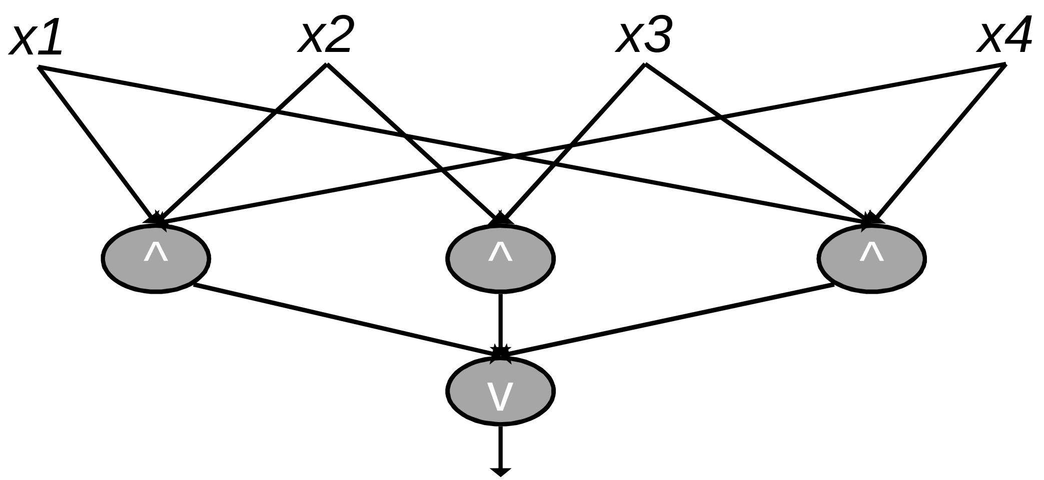
- If $A\in\Gamma$ and $A \subseteq B$, then $B\in\Gamma$.
- $A\in\Gamma$ is a minimal authorized set if there is no $C \subseteq A$ such that $C\in\Gamma$.
- The set of minimal subsets Γ_0 is called the basis of Γ.

Why should we examine general access structures?

- Some access structures can be implemented using threshold access structures.
- But not all access structures can be represented by threshold access structures
- For example, consider the access structure $\Gamma=\{\{1,2\},\{3,4\}\}$
  - Any threshold based secret sharing scheme with threshold t gives weights to parties, such that $w_1+w_2 \geq t$, and $w_3+w_4 \geq t$.
  - Therefore either $w_1 \geq t/2$, or $w_2 \geq t/2$. Suppose that this is w_1.
  - Similarly either $w_3 \geq t/2$, or $w_4 \geq t/2$. Suppose that this is w_3.
  - In this case parties 1 and 3 can reveal the secret, since $w_1+w_3 \geq t$.
  - Therefore, this access structure cannot be realized by a threshold scheme.

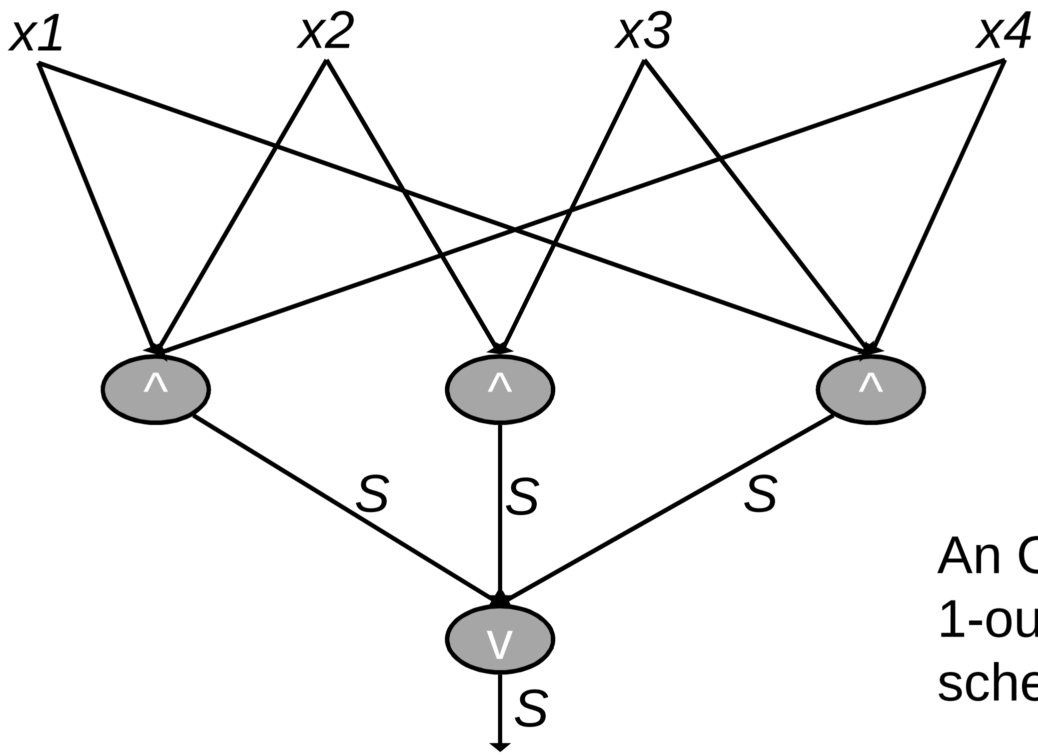
The monotone circuit construction (Benaloh-Leichter)

- Given Γ construct a circuit C s.t. $C(A)=1$ iff $A \in \Gamma$.
  - $\Gamma_0 = \{ \{1,2,4\}, \{1,3,4,\}, \{2,3\} \}$
- This Boolean circuit can be constructed from OR and AND gates, and is *monotone.* Namely, if $C(x)=1$, then changing bits of *x* from 0 to 1 doesn't change the result to 0.



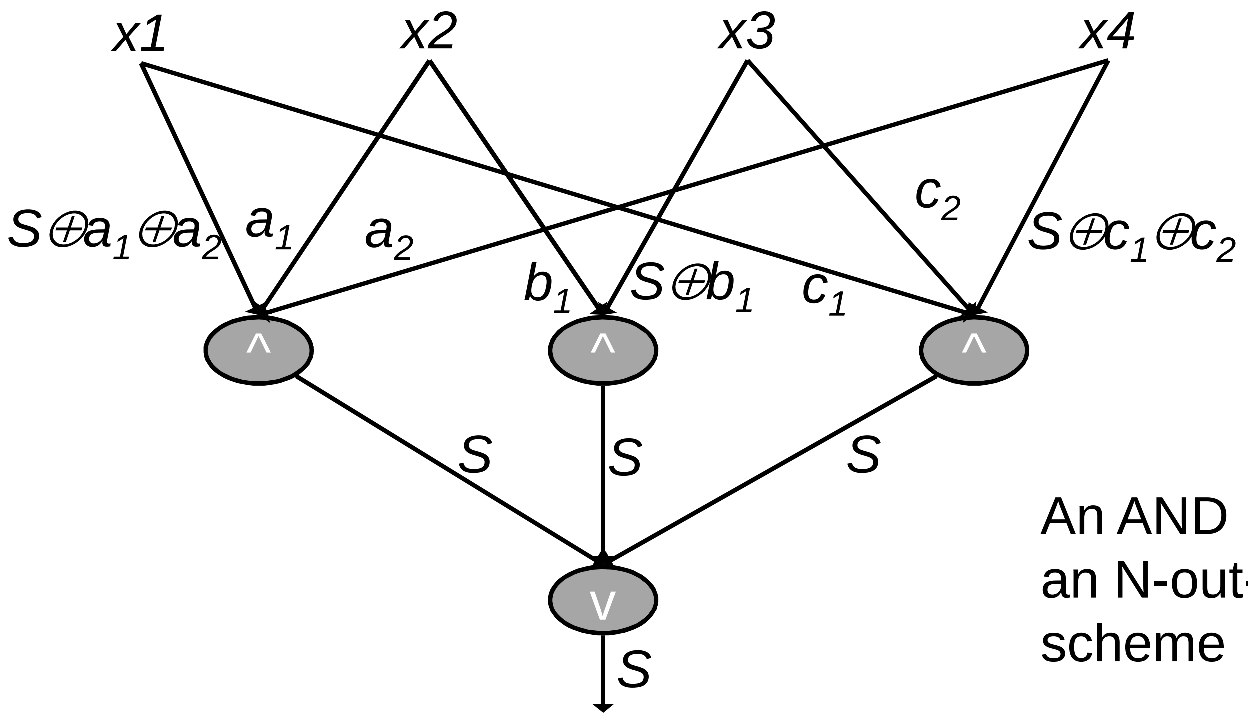
Handling OR gates

Starting from the output gate and going backwards



An OR gate is a 1-out-of-N scheme

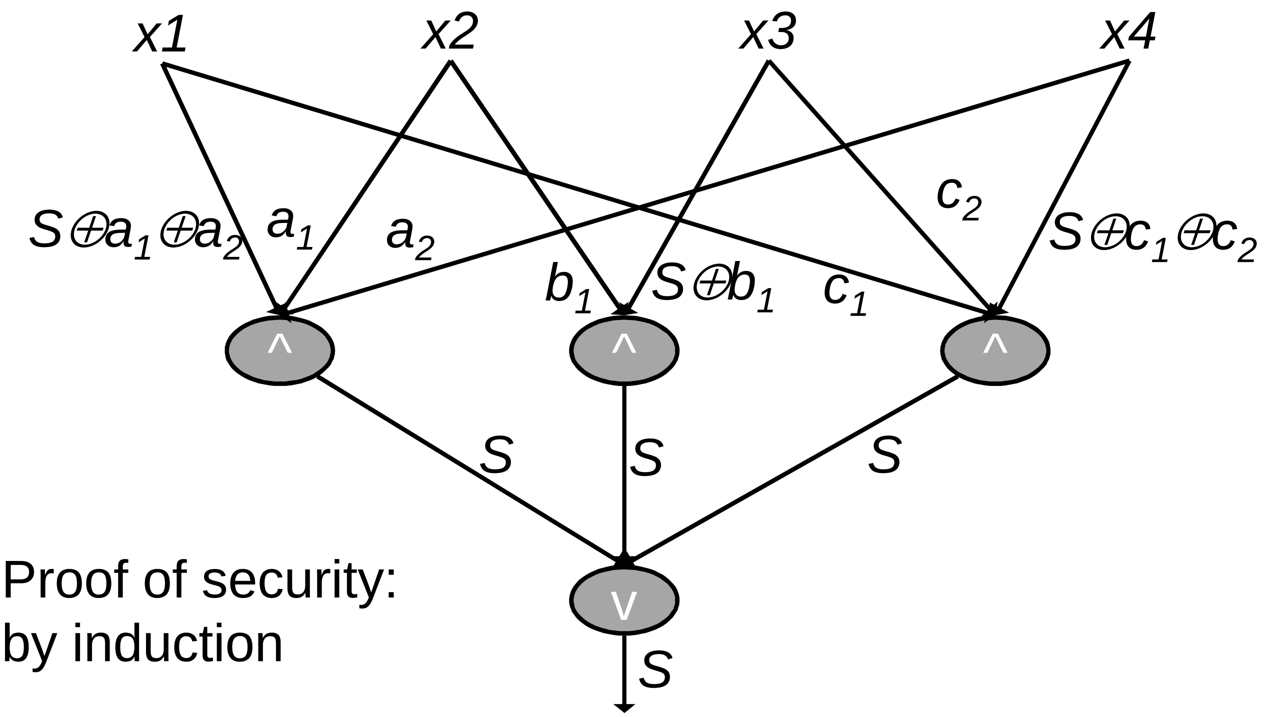
Handling AND gates



An AND gate is an N-out-of-N scheme

Handling AND gates

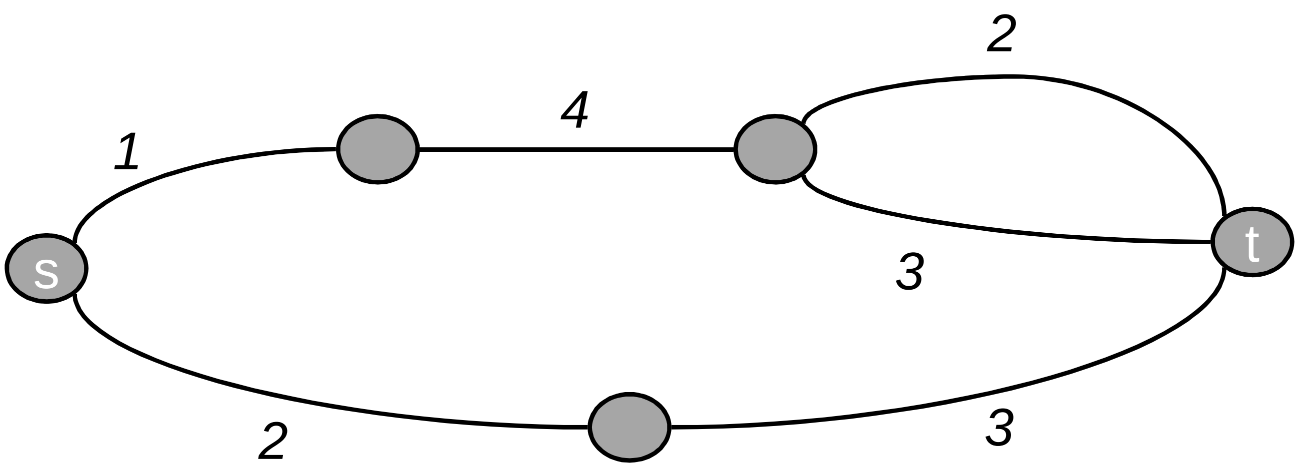
Final step: each user gets the keys of the wires going out from its variable



Proof of security: by induction

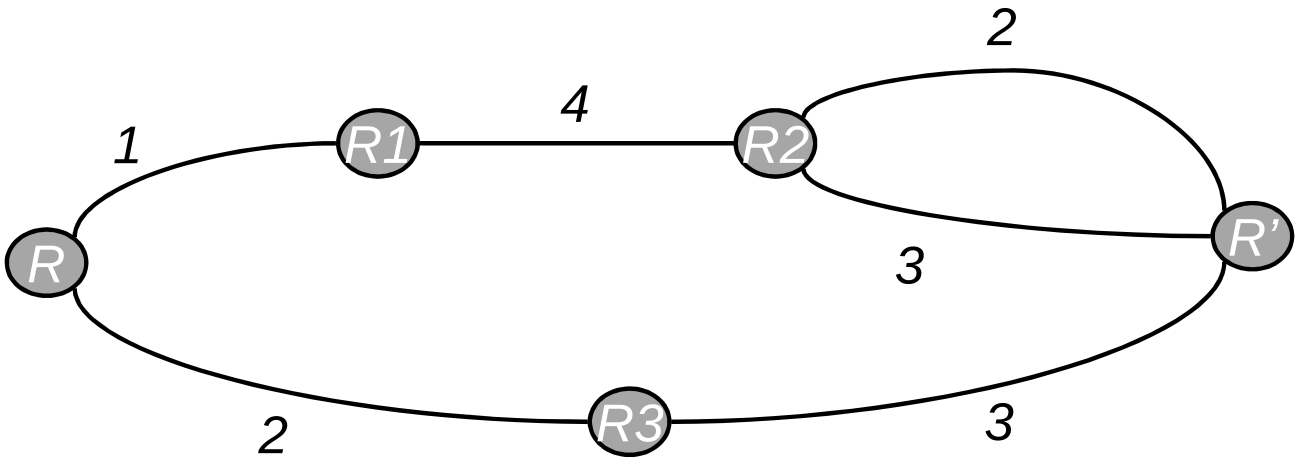
A graph based construction

- Represent the access structure by an undirected graph.
- An authorized set corresponds to a path from s to t in an undirected graph.
- Γ_0 = { {1,2,4}, {1,3,4,}, {2,3} }

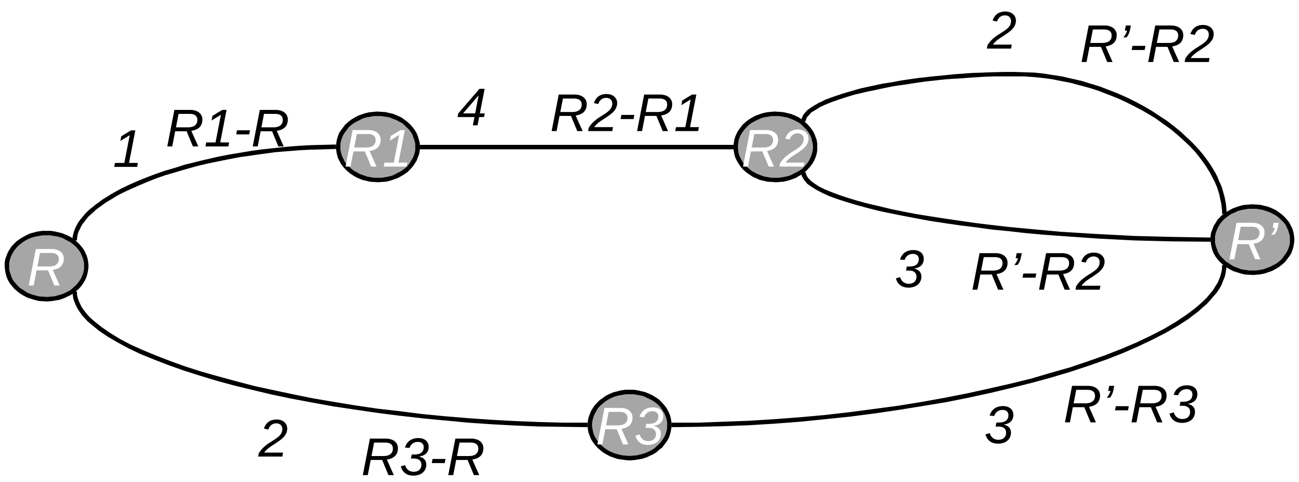


A graph based construction

Assign random values to nodes, s.t. *R'-R= shared secret (R'=R+shared secret)*



A graph based construction



- Assign to edge R1→R2 the value *R2-R1*
- Give to each user the values associated with its edges

A graph based construction

- Consider the set {1,2,4}
- why can an authorized set reconstruct the secret? Why can't a unauthorized set do that?