Software Security Technologies

Richard Sinn
Principal Architect / Security Architect,
Yahoo, Inc
Lecturer, SJSU

Agenda

- Basic Security Theory
- Essential PKI
- Trust Models
- Threat Models
- Security Challenges in Email/Instant Communication

Basic Security Theory

The Problems

- Fundamental
  - People can read data in plain/clear text
  - Data is sent over public network
- Specific
  - Spoofing – Fake the source of data
  - Tampering – Change data in transit
  - Snooping – Monitor sensitive data
  - Replay – Intercept and resend data at a later time

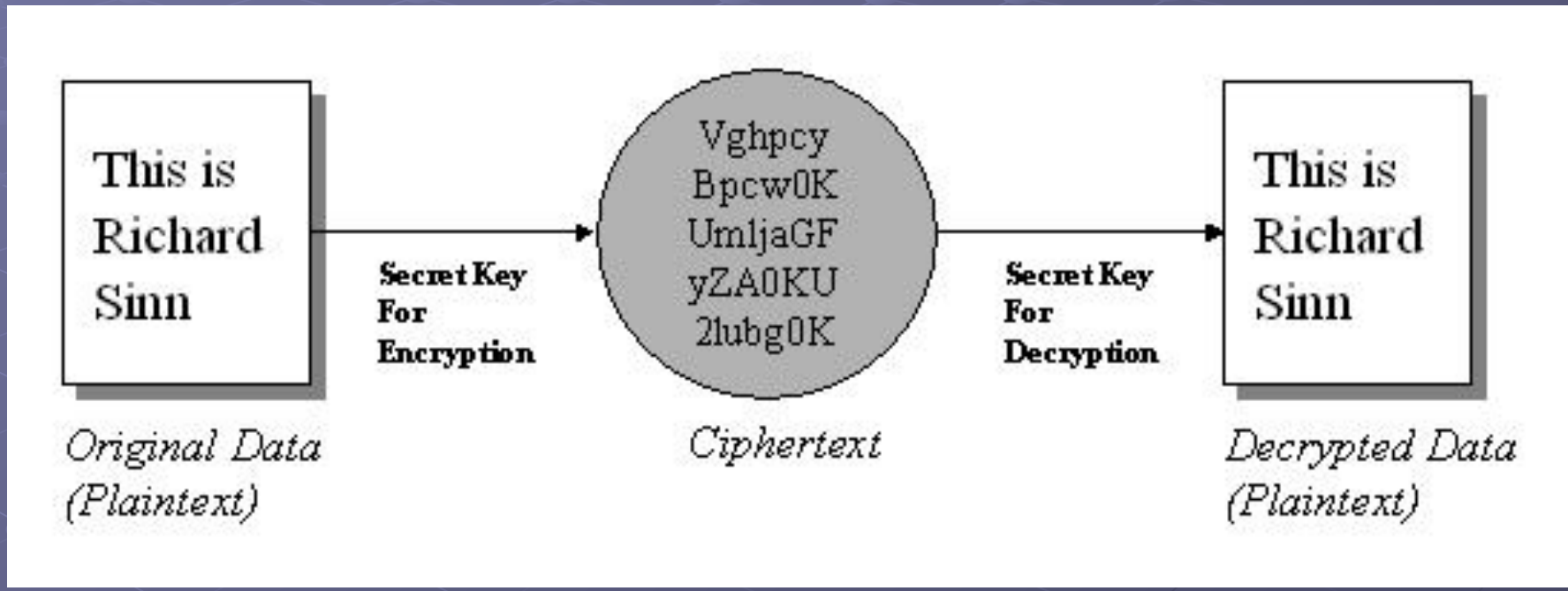
The Goals

- Confidentiality – Assurance of data privacy
- Authentication – Assurance that an entity is who he/she/it claims to be
- Integrity – Assurance of non-alteration
- Availability – Assurance of data / services are available

The Toolbox

- Symmetric Key Encryption
  - Stream ciphers
  - Block ciphers
- Message Digest and MACs
- Public Key Encryption
  - DH
  - RSA
- Digital Signature

Symmetric Key Encryption



Symmetric Key Encryption

- Plaintext – Original Data
- Ciphertext – The Gibberish
- Encryption – Transformation from plaintext to cipher-text
- Decryption – Transformation from cipher-text to plaintext
- Algorithm – Cipher
- Symmetric – Same key for encryption/ decryption

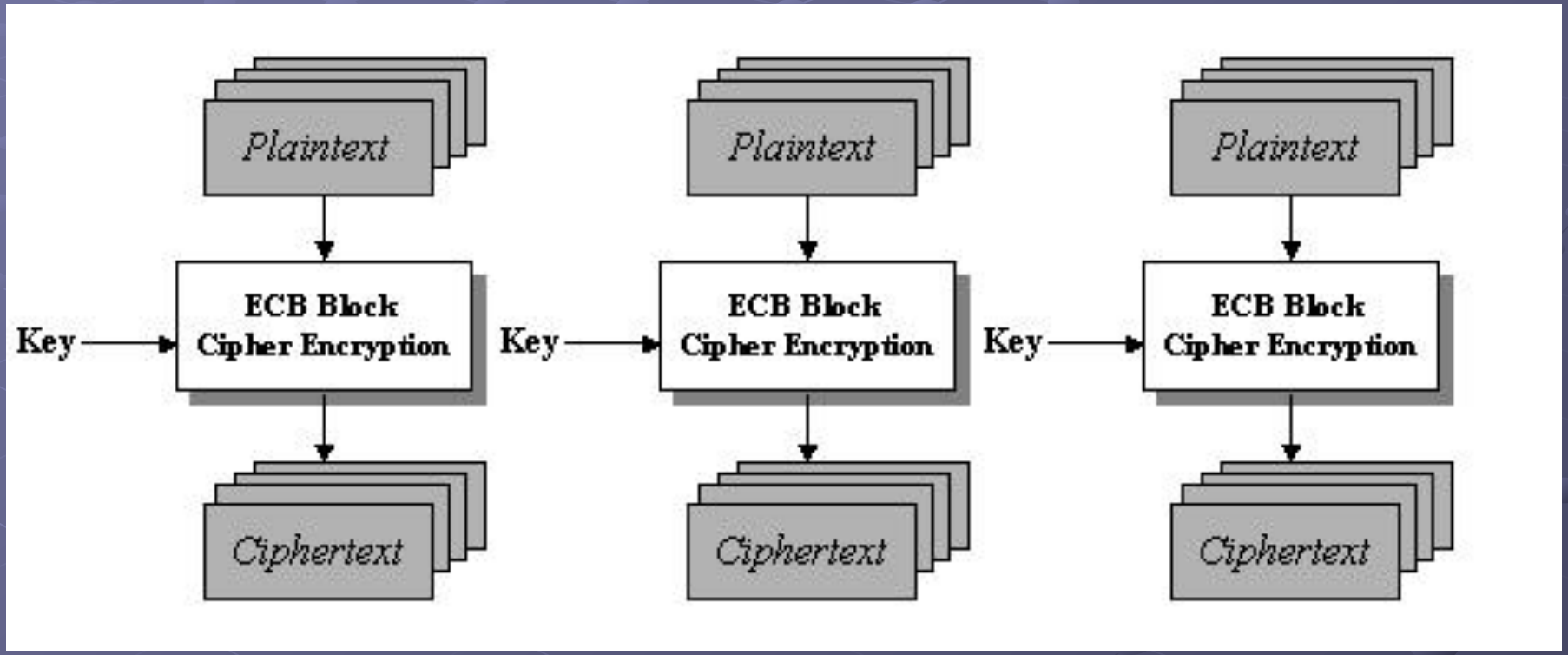
Stream Ciphers

- *Takes the original data, divides it into digits, and encrypts each digit one at a time.*
- *C[i] = KS[i] XOR P[i]*
- *P[i] = KS[i] XOR C[i]*
- *RC4 – Stream cipher with variable key length between 8 to 2048 bits*

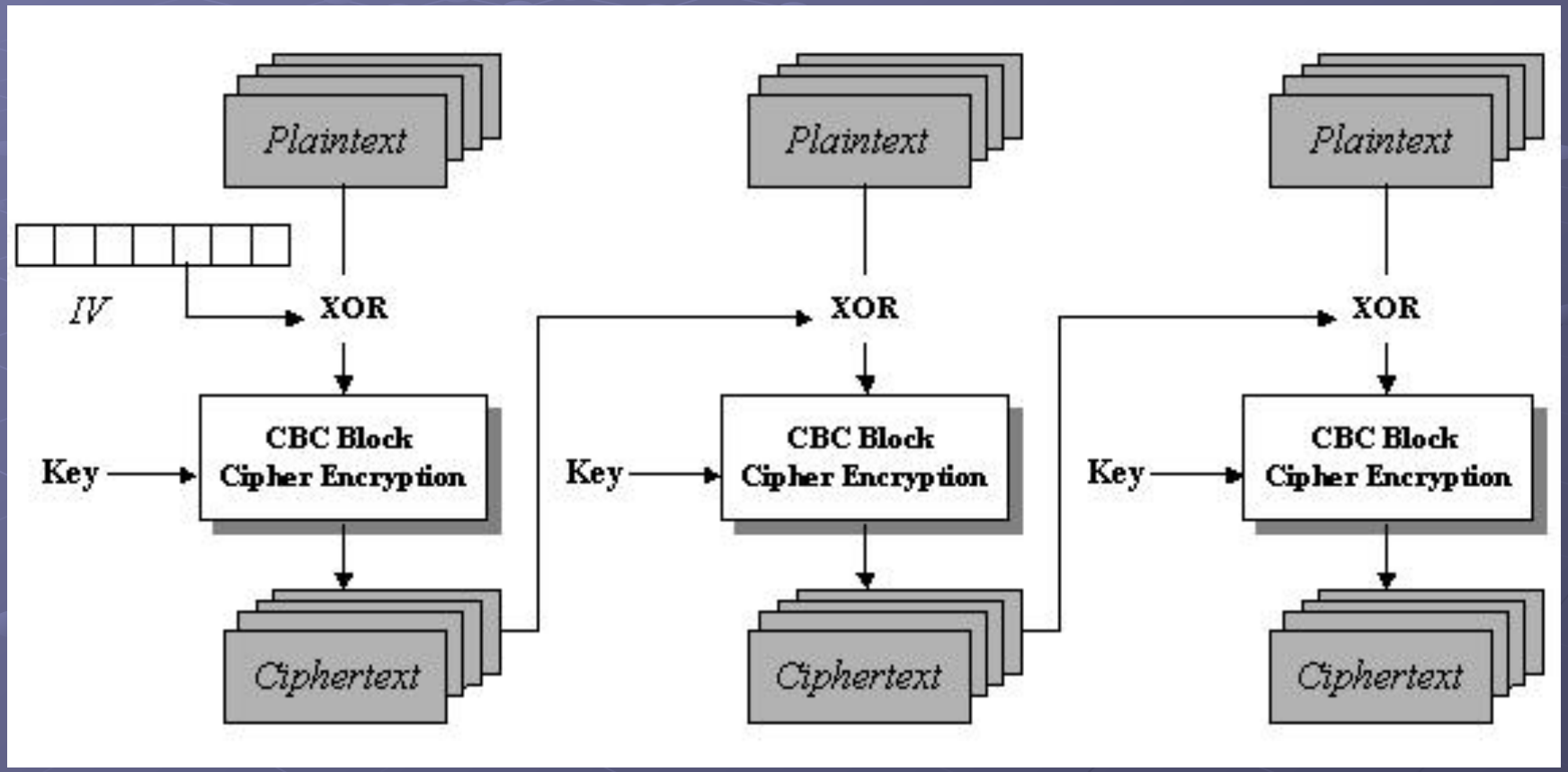
Block Ciphers

- Divided up data into blocks of fixed length
- 64 or 128 bits
- Various different modes
  - Electronic Code Book (ECB)
  - Cipher-Block Chaining (CBC)
  - Cipher Feedback (CFB)
  - Output Feedback (OFB)
  - Counter (CTR)

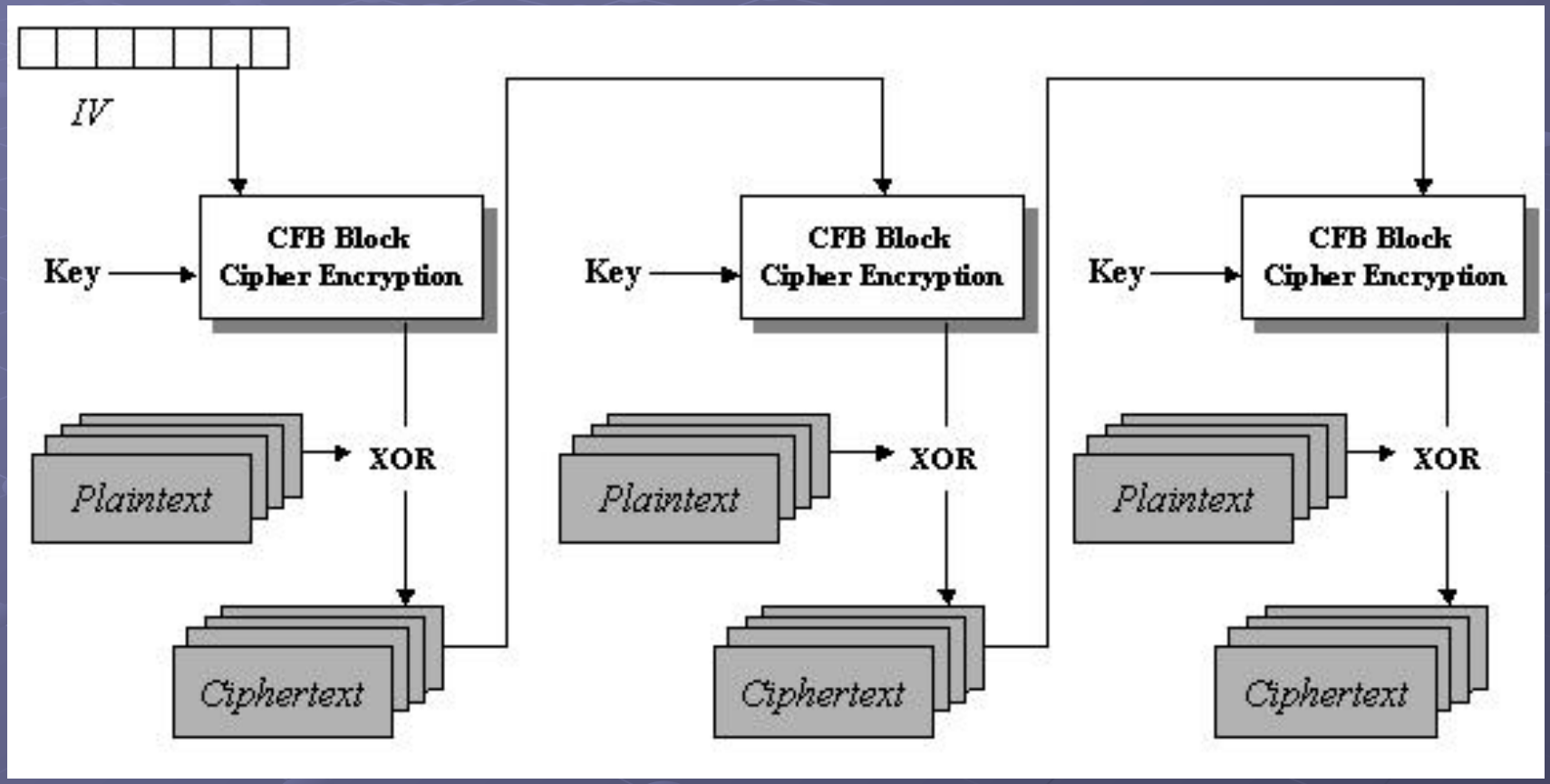
Electronic Codebook (ECB)



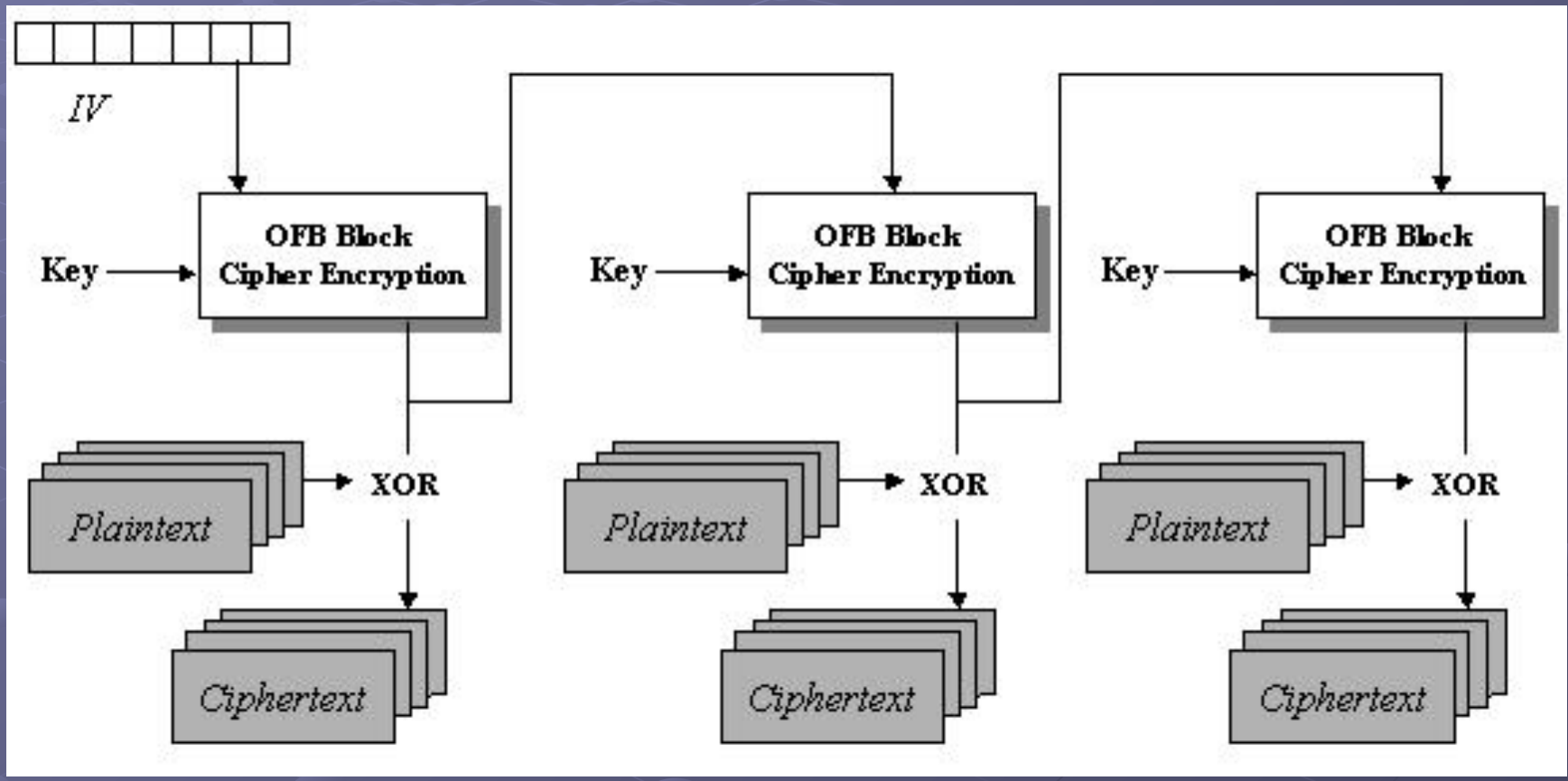
Cipher Block Chaining (CBC)



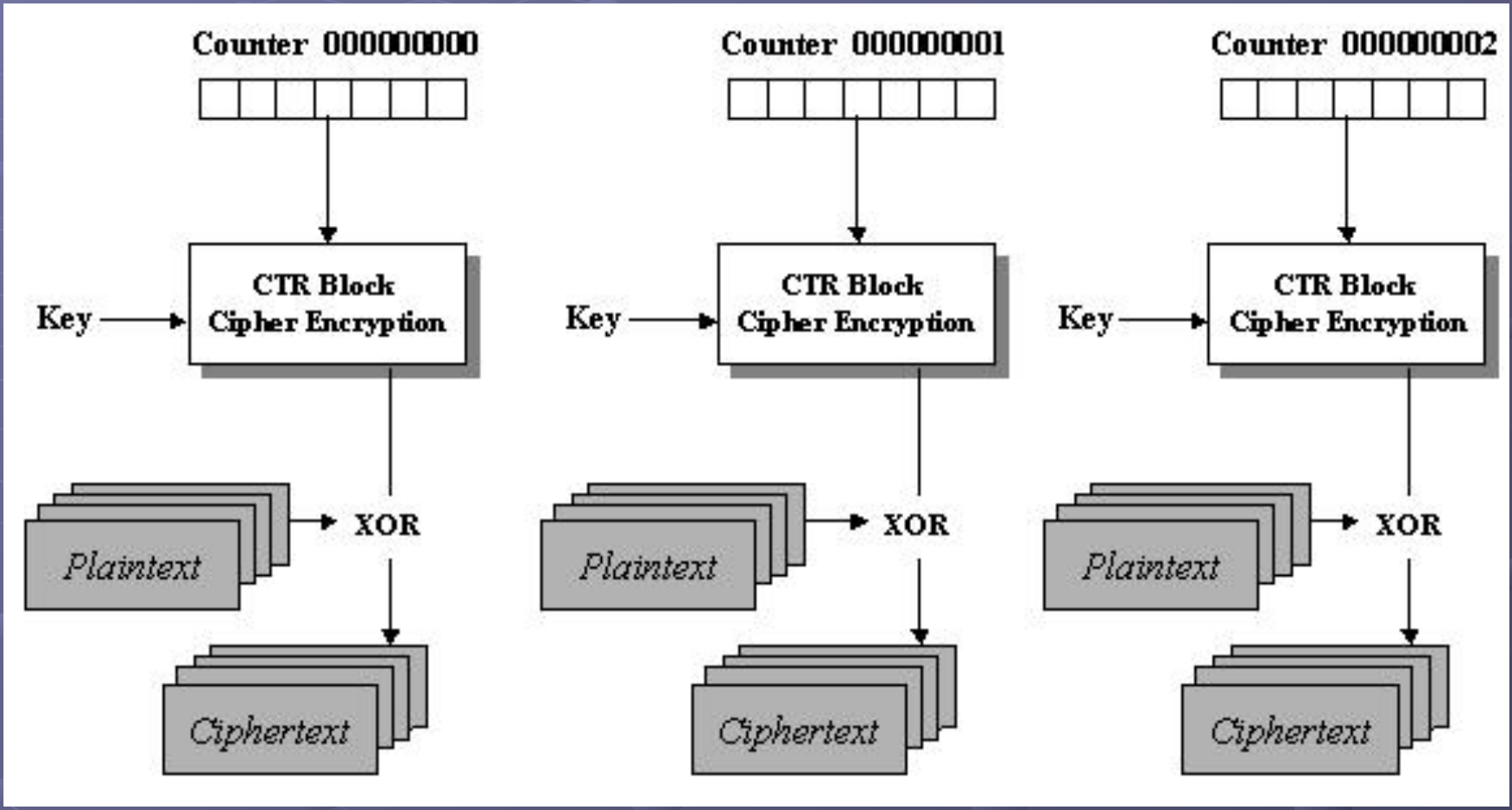
Cipher Feedback (CFB)



Output Feedback (OFB)



Counter (CTR)


Counter 000000000
Key
CTR Block Cipher Encryption
Plaintext
XOR
Ciphertext
Counter 000000001
Key
CTR Block Cipher Encryption
Plaintext
XOR
Ciphertext
Counter 000000002
Key
CTR Block Cipher Encryption
Plaintext
XOR
Ciphertext

Common Block Ciphers

- DES – IBM 1970. 64-bit, 56-bit key. 8 for parity check
- 3DES - *C = DESk3(DES-1k2(DESk1(P))).*
- RC2 – Ron Rivest, RSA, 64-bit with variable size key up to 128 bytes
- AES - Fixed block size of 128 bits and a key size of 128, 192 or 256 bits

The Toolbox

- Symmetric Key Encryption
  - Stream ciphers
  - Block ciphers
- Message Digest and MACs
- Public Key Encryption
  - DH
  - RSA
- Digital Signature

Message Digest

- Call digest or hash
- Fixed-size checksum created by cryptographic hash functions
- Properties
  - Preimage resistant – *H = Hash (m)*, Given H cannot find m
  - Second preimage resistant – Given m1, infeasible to find m2. *Hash (m1) = Hash (m2)*
  - Collision-resistant: Infeasible to find *m1* and *m2* such that *Hash (m1) = Hash (m2)*
- Examples: MD5, SHA-1, etc.

Message Authentication Code

- MAC
- Use secret key to address spoof data + hash problem
- Examples: Data Authentication Code, HMAC
- *HMACk(m) = H((K XOR opad) || H((K XOR ipad) || M))*

The Toolbox

- Symmetric Key Encryption
  - Stream ciphers
  - Block ciphers
- Message Digest and MACs
- Public Key Encryption
  - DH
  - RSA
- Digital Signature

Public Key Cryptography

- Each party has two keys: one is the **private key** where it must be kept secret, and one is the **public key** that can be freely distributed
- One encrypts and the other one decrypts
- Performance limit
- PK is usually used as means for secret key exchange

Diffie-Hellman (DH)

- Used for key agreement
- Exchange of information over an insecure medium that allows each of two parties (sender and recipient) to compute a value that will be used to construct a secret key for a symmetric cipher during the rest of the communication.
- DH is invented by Whitfield Diffie and Martin Hellman in 1976

Diffie-Hellman (DH)

Action	**Example Values**
Alice and Bob agree on two integers: p and g, where p is a large prime number and g is called the base.	Let $p = 29$, $g = 3$.
Alice chooses a secret integer a. Then, she sends Bob $g^a \bmod p$.	Let $a = 5$, then $g^a \bmod p = 3^5 \bmod 29 = 11$.
Bob chooses a secret integer b. Then, he sends Alice $g^b \bmod p$	Let $b = 10$, then $g^b \bmod p = 3^{10} \bmod 29 = 5$.
Alice computes $k_a = (g^b \bmod p)^a \bmod p$.	$k_a = (g^b \bmod p)^a \bmod p = 5^5 \bmod 29 = 22$.
Bob computes $k_b = (g^a \bmod p)^b \bmod p$.	$k_b = (g^a \bmod p)^b \bmod p = 11^{10} \bmod 29 = 22$.
Since $k_a = k_b = k$, a secret value is exchanged.	The value 22 is secretly exchanged.

RSA

- RSA is the public key algorithm invented in 1977 by Ron Rivest, Adi Shamir, and Len Adelman at MIT.
  - Choose two large prime numbers *p* and *q* randomly, unpredictably and independently of each other.
  - Compute $N = p\,q$.
  - Compute $\varphi = (p - 1)(q - 1)$.
  - Choose an integer *e* where $1 < e < \varphi$ and is coprime to φ. Coprime indicates that e and φ does not have common factor except 1 and –1.
  - Compute *d = e-1 mode* (φ)

RSA – Encryption Example

Action	**Example Values**
First prime (kept private): p Second prime (kept private): q Modulus (public): $N = pq$ Public exponent (public): e Private exponent (kept private): d The public key is (N, e). The private key is (N, d).	Let: $p = 11$ $q = 3$ $N = pq = 33$ $e = 3$ $d = 7$ The public key is (33, 3). The private key is (33, 7).
Plaintext = n	Let $n = 7$
encrypt(n) = c = n^e mod N where m is plaintext, and c is ciphertext.	$encrypt(n) = n^3 \bmod 33$ $= 7^3 \bmod 33$ $= 13$. Thus, ciphertext = 13.

Digital Signature

- Use to ensure integrity
- MAC is not useful due to the need of same secret key
- Alice can sign a message with her private key, and anyone who has her public key can verify the signed message.
- The message is usually cryptographically hashed, and then only the hash of the message is signed.

Digital Signature - RSA

Action	**Example Values**
Sign message m. Compute $h = Hash(m)$ Signature = $s = h^d \bmod N$	Reuse values from the RSA encryption / decryption example: $d = 7, N = 33, e = 3$ Let $m = 123$ Assume $h = Hash(123) = 5$ Signature = $s = h^d \bmod N = 5^7 \bmod 33 = 14$

RSA Signature Generation

Digital Signature - RSA

Action	**Example Values**
Verify signature Receive message m, and signature s Compute $x = Hash(m)$ Compute $y = s^e \bmod N$ Message is verified if and only if $x = y$	Given $s = 14$, $e = 3$, and $m = 123$ $x = Hash(m) = 5$ $y = s^e \bmod N = 14^3 \bmod 33 = 5$ Message is verified since $x = y$.

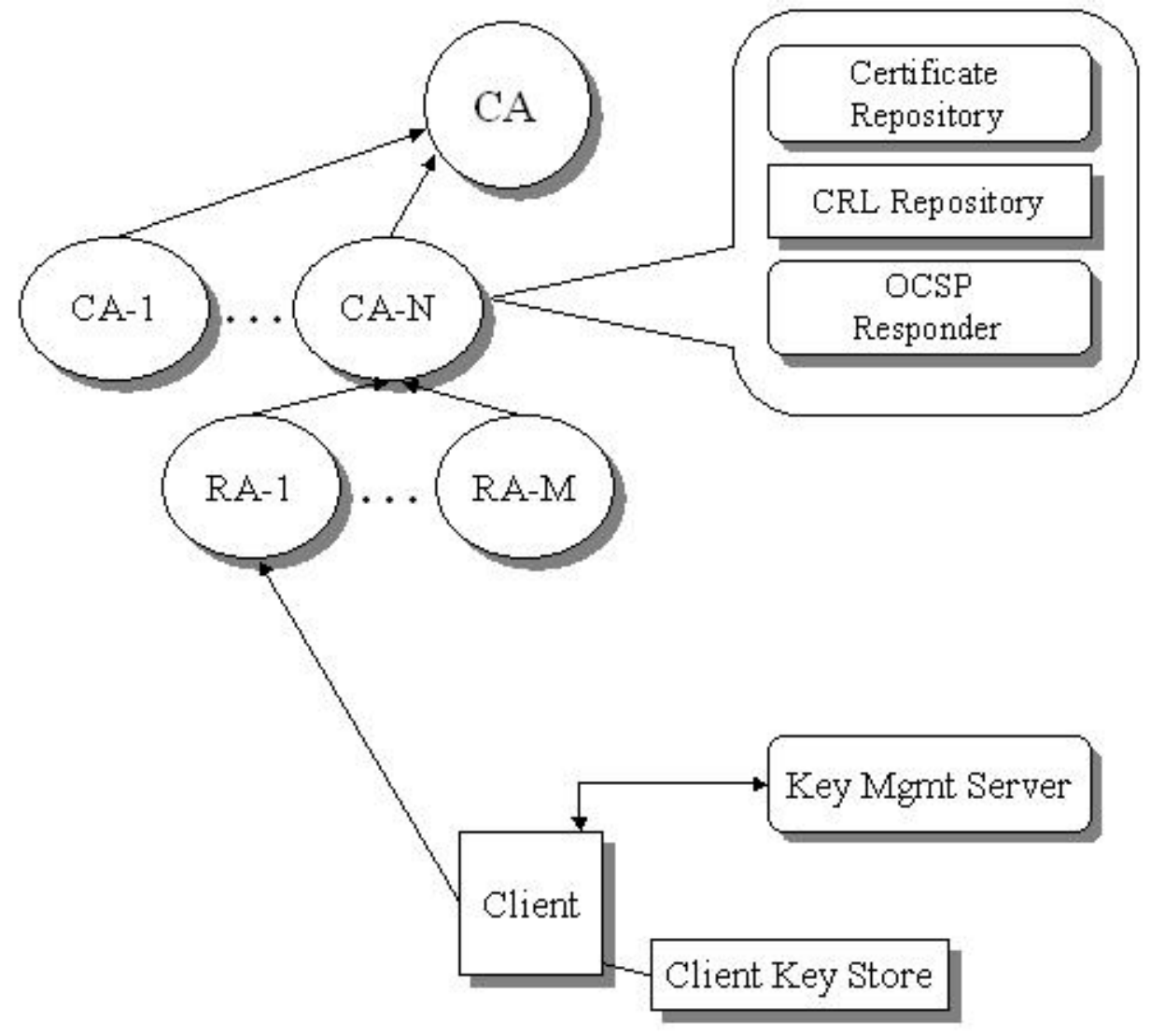
RSA Signature Verification

Essential PKI

Goals

- How do you get someone else's public key?
- An infrastructure is needed:
  - *Well-defined Entry Points*
  - *Predictable Security*
  - *End-User Transparency*
- Services:
  - Authentication
  - Integrity
  - Confientiality

PKI Structure



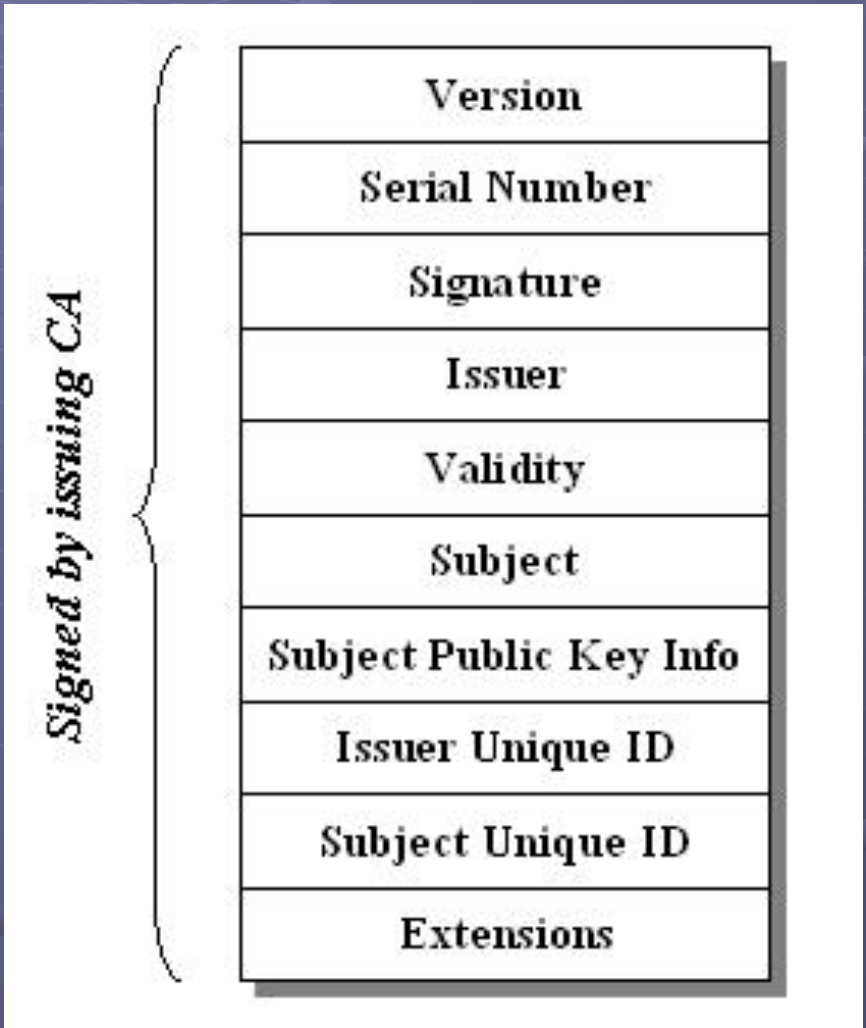
PKI Structure

- Certificate Authority
- Registration Authority
- Certificate Repository
- CRL Repository
- OCSP Responder
- Client
- Client Key Store
- Key management server

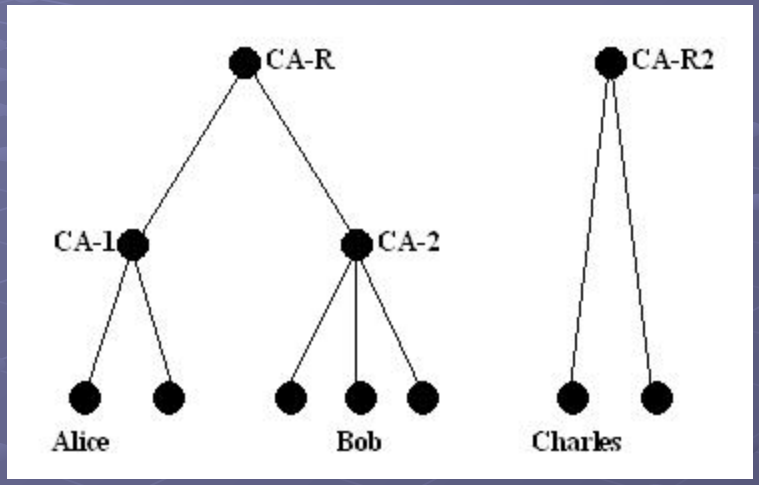
Certificate

- The goal of a certificate in PKI is to provide a mechanism that binds the public key to the claimed owner in a trustworthy manner.
- This binding needs to ensure that the integrity of the public key is preserved, and the pubic key and any other associated identity information has been bound to the claimed owner.

X.509 pubic-key certificate

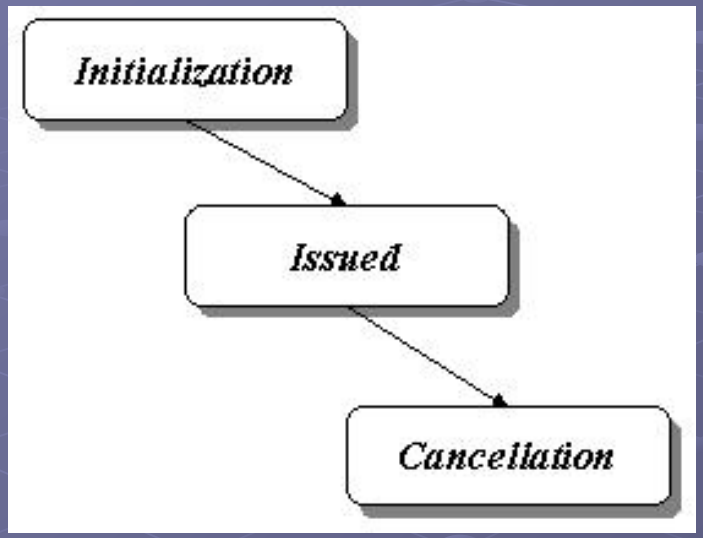


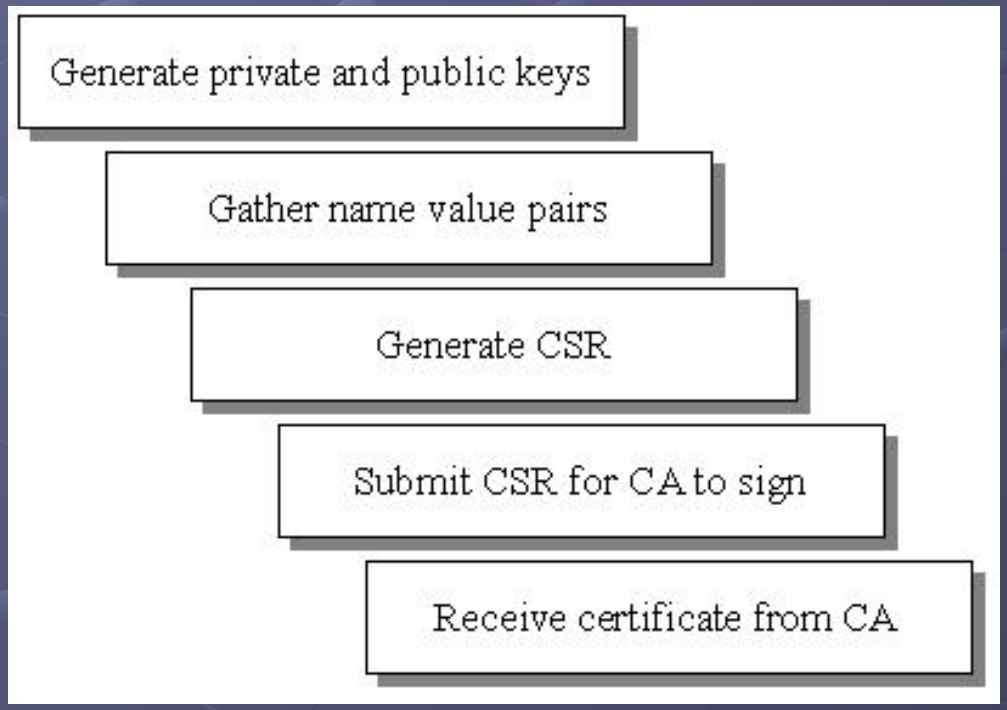
Certificate Path



- Alice holds Bob's certificate.
- Bob's certificate is signed by CA-2.
- Alice accesses the certificate repository in the PKI to obtain CA-2's certificate for examination.
- CA-2 is signed by CA-R that is also the root CA for Alice.
- Since Alice's certificate is signed by CA-1, and CA-1 is in turn signed by CA-R.
- Alice should trust CA-2.
- A valid certificate path for Bob is found: Bob->CA-2->CA-R. Another way of saying it is that Bob's certificate chained back to CA-R, and Alice successfully validate Bob's certificate.

Key and Cert Management





Trust Models

Next Question

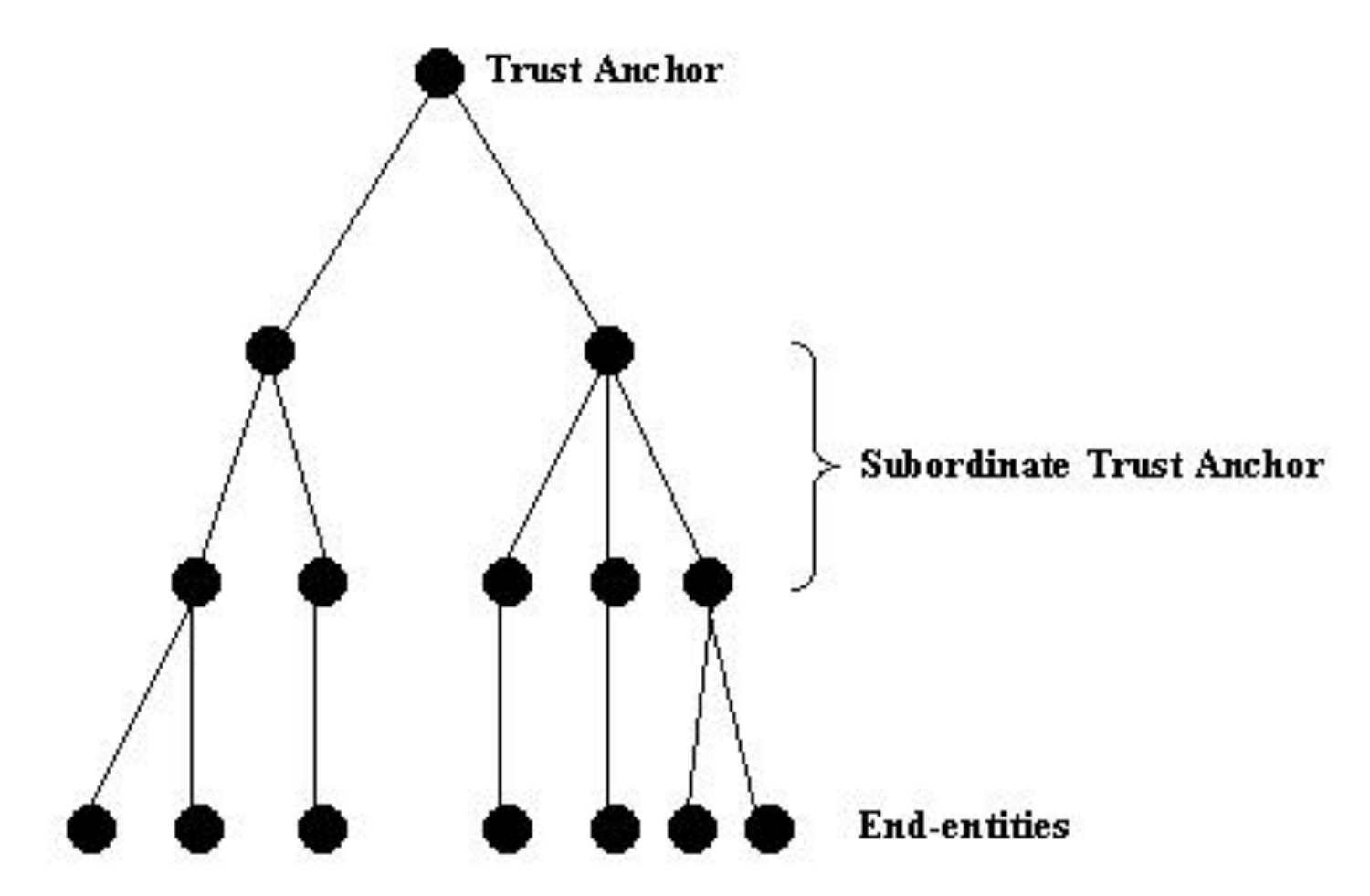
- The public key is in a certificate signed by the CA
- The public key is valid since I trust the CA
- What does Trust mean?
- Answer: (From X.509 specification)

"A" trusts "B" when "A" assumes that "B" will behave exactly as "A" expects.

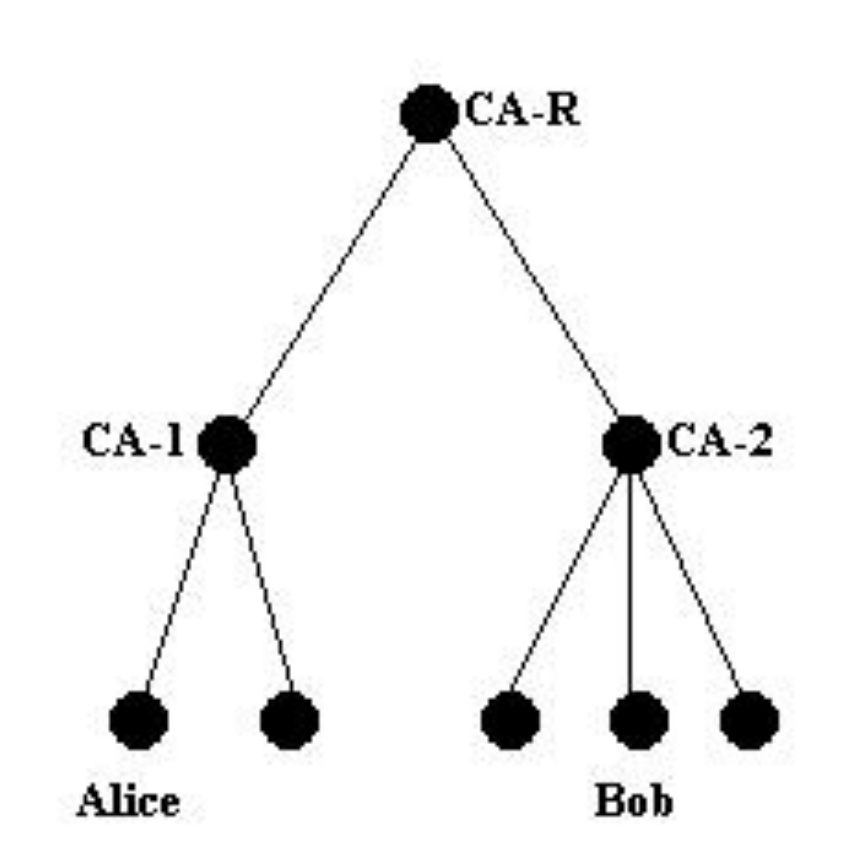
Trust Model

- Capture how trust relationship is used in software
- Once you have decided on the trust model to use, the software you build should be based on the trust model.
- Trust models definition is important because trust models might be implicitly assumed by an entity.

Strict Hierarchy Trust Model

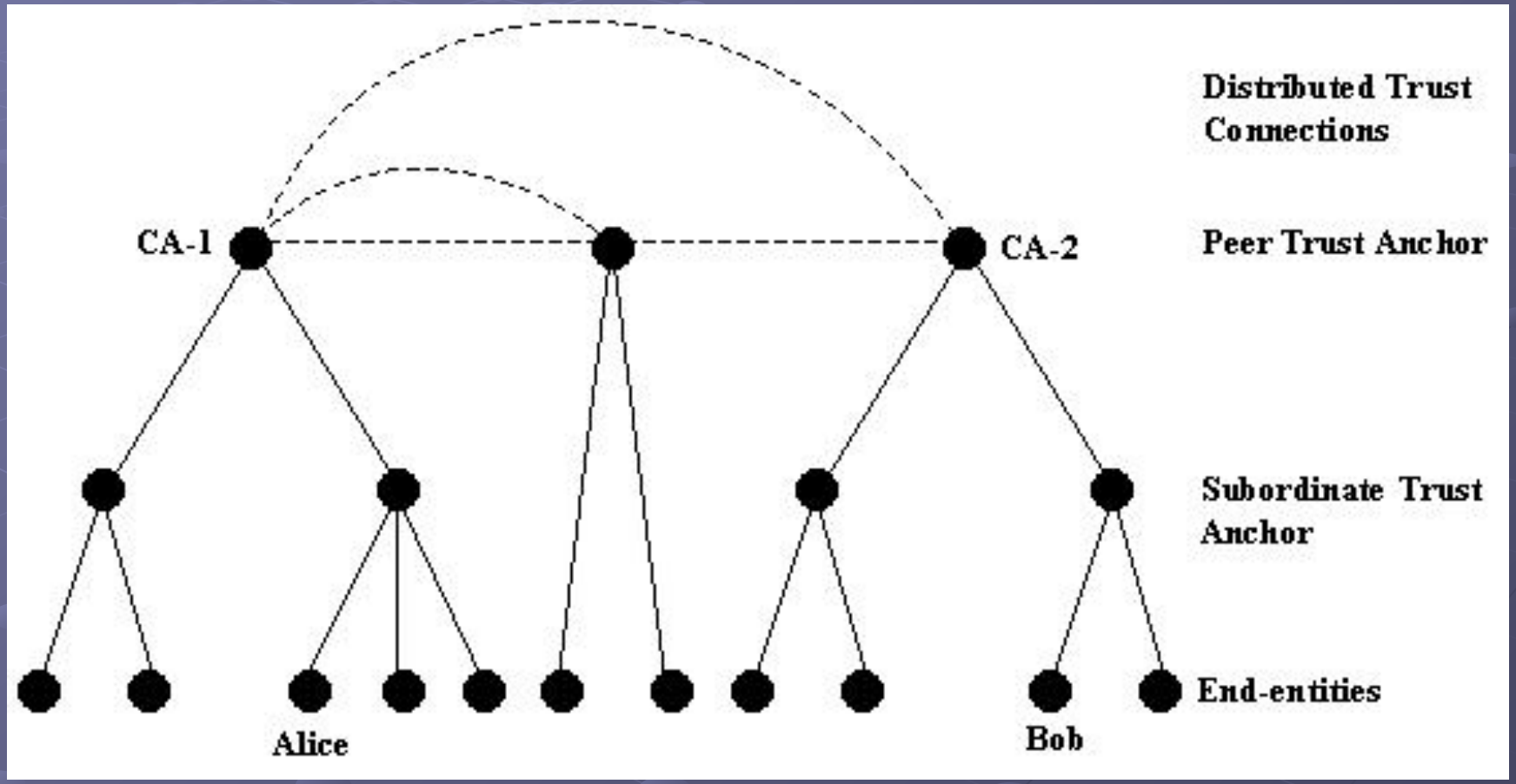


Strict Hierarchy Model - Example



- Given Bob's certificate is signed by CA-2.
- CA-2's certificate is signed by CA-R.
- CA-R is the root trust anchor for Alice as well.
- With CA-R's public key certificate, Alice can obtain and verify CA-2's certificate.
- With CA-2's certificate verified and trusted, Alice can use CA-2's certificate (with the corresponding public key) to verify Bob's certificate.
- Once Bob's certificate is verified, Bob's public key is trusted and can be used to encrypt messages for Bob, verify digital signature, etc.

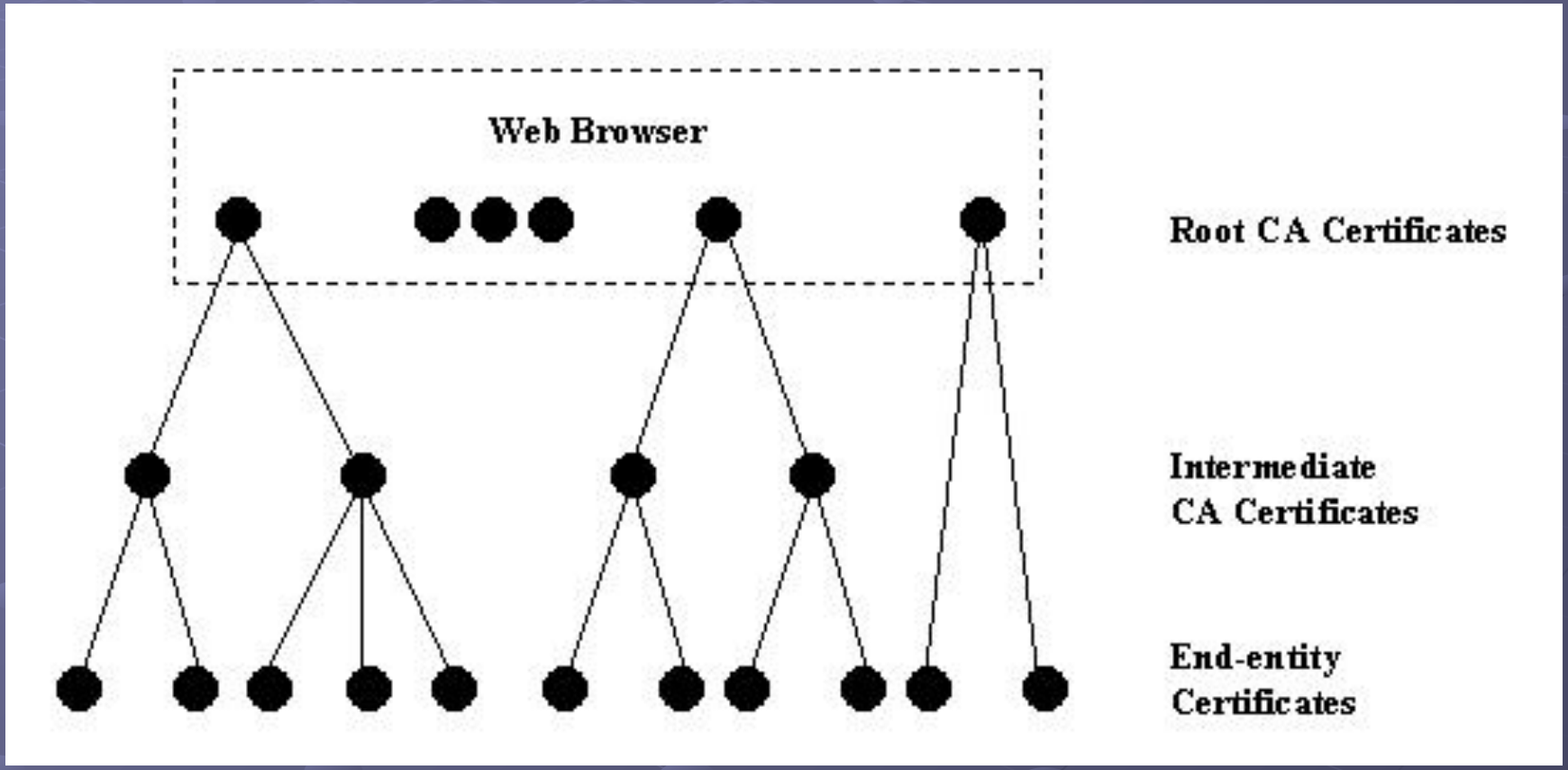
Distributed Trust Model



Distributed Trust Model

- Hub Configuration – It is also called Star Configuration. Under this configuration, each peer trust anchor is connected to a *Hub*. This central hub is used to bridge communication gaps between pairs of peer trust anchors.
- Mesh Configuration –In this configuration, all peer trust anchors are potentially cross-certified among each other. In the fully connected scenario (also called *full mesh*), this configuration requires n2 cross-certification agreements for *n* peer trust anchors.

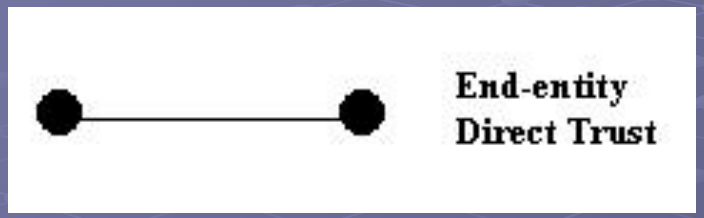
Web Trust Model

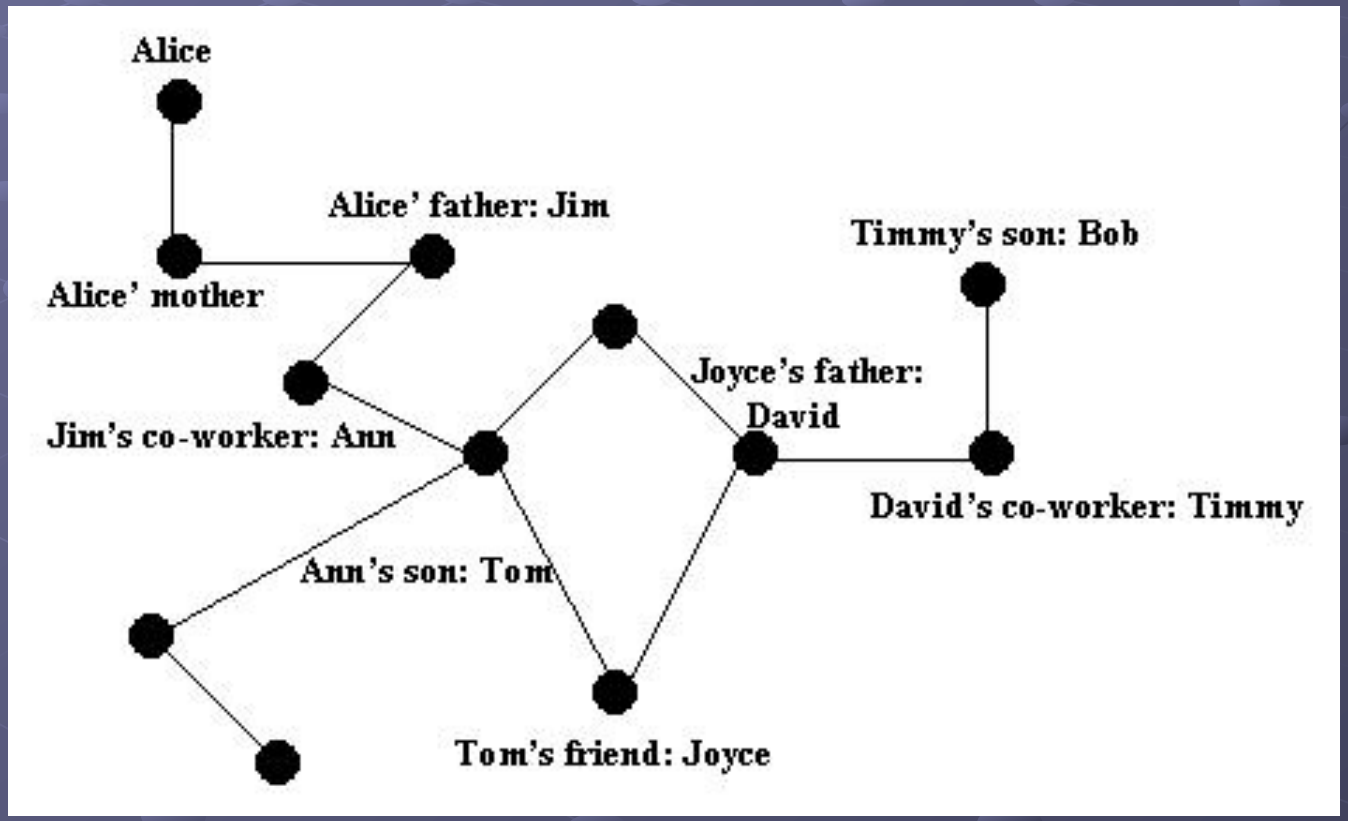


Web Trust Model

- Advantages: Easy to implement, support Internet, it is there.
- Disadvantages:
  - Identify Spoofing with “bad” CA cert
  - Decentralized Trust Management – user level management
  - Inability for Revocation – No CRL, OCSP
  - Distribution Problem – Who get what version of browsers?

User-Centric Trust Model





Reputation Trust Model

- Centralized System
  - An authority is responsible for accumulating evaluations of agents from other agents. This authority then scores each agent with a reputation score.
  - All reputations are public and global.
  - Reputations of service providers are scored by consumers agents, and consumer agents' reputations are scored by service providers agent.
  - Reputations are built and own by the centralized system.

Reputation Trust Model

- Centralized System Disadvantages
  - Artificial reputation score
  - Agent Spoofing
  - The revenge factor
- Decentralized System
  - Agent A can obtain agent B's reputation by proactively requesting and collecting other agent's evaluations for B
  - This evaluations will then be combined to form agent A's reputation score for agent B