INB255/INN255 Security

Lecture 7:

Asymmetric Cryptography and PKI

Outline

- Symmetric ciphers: key establishment problem
- Asymmetric (public key) ciphers
  - For confidentiality
  - For authentication
  - For integrity
- Public keys and required infrastructure
  - The spoofing problem
  - Digital certificates
  - Public key infrastructure
- Summary

Symmetric ciphers:
the key establishment problem

- Symmetric ciphers
  - used to provide security services such as:
    - Confidentiality
    - Integrity
  - use the *same secret key* to encrypt and decrypt
    - For <u>transmission</u> of encrypted messages
      - key is required at both sending and receiving points
    - For encrypted <u>storage</u>,
      - key is required at storage and at retrieval

- Security of encrypted data depends on security of the key:
  - Need to consider C, I, A of keys – how to provide this?

Symmetric ciphers:
the key establishment problem

- Transmission scenario:
  - Two parties (Alice and Bob) wish to communicate securely over an insecure channel.
  - MITM Carol may be eavesdropping, or modifying transmissions
- Alice and Bob decide to use cryptographic mechanisms based on <u>*symmetric ciphers*</u> to provide security services for their communications.
- <u>Before</u> they can have a secure communication session, they need to <u>establish a shared secret key</u>: a *session key*.
- Question: How can they do this if the communication channel they are using is not secure?

Symmetric ciphers:
the key establishment problem

- Key establishment options:
  1. Generate and securely pre-distribute shared keys for likely communicants
     - Possible for a small group (see next slide)
  2. Use a trusted third party (as a key server)
     - who shares a (long-term) symmetric key with each user, and can provide a shared key for any two parties upon request
       – Example: Kerberos protocol
  3. Diffie-Hellman key agreement algorithm
     - Allows 2 hosts to create a shared secret (form a secret key) without a secure distribution channel

Symmetric ciphers:
the key establishment problem

- Option 1 – Secure pre-distribution
  - Secret key distribution must happen ‘out-of-band’:
    - Not over the communication channel, but over a different secure channel.
    - Maybe physical distribution via trusted courier?
- How many *secret keys* are needed in a system?
  - If there are
    - two people, only one secret key is needed.
    - three people, and each participant may possibly communicate securely with every other participant, three secret keys are needed.
    - four people, six secret keys are needed.
    - *n* people, *n(n-1)/2* secret keys are needed

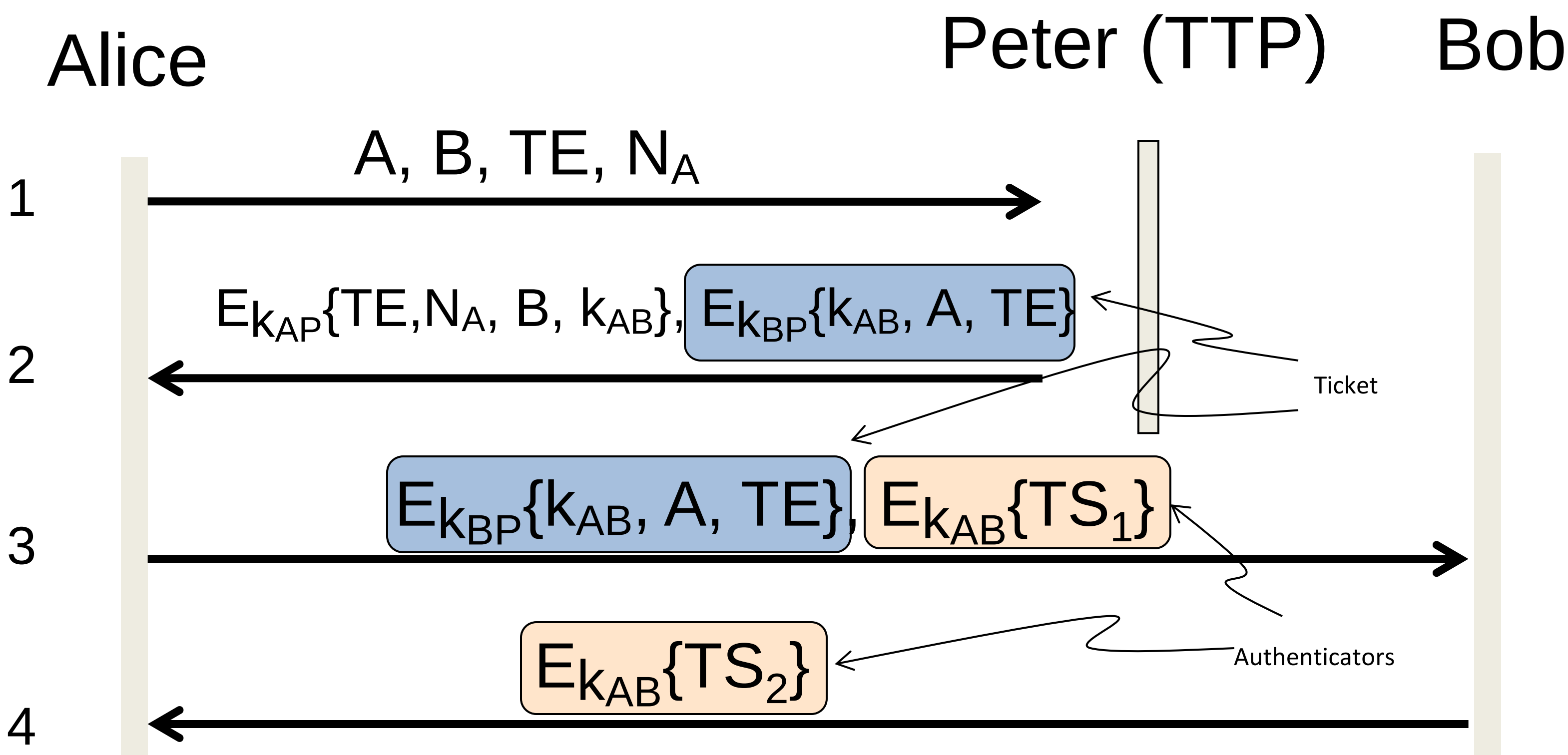
Symmetric ciphers:
the key establishment problem

- Option 1 – Secure pre-distribution
  - *What sort of numbers are we looking at?*
    - for *n* = 5, (5 x 4)/2 = 10 keys must be established.
    - for *n* = 10, (10 x 9)/2 = 45 keys must be established.
    - for *n* = 20, (20 x 19)/2 = 190 keys must be established.
    - for *n* = 100, (100 x 99)/2 = 4 950 keys.
    - for *n* = 200, (200 x 199)/2 = 19 900 keys.
    - for *n* = 350,(350 x 349)/2 = 61 075 keys.
  - Not practical
    - for large numbers of participants, or
    - if users will be added/removed from system frequently

Symmetric ciphers:
the key establishment problem

- Option 2 – Trusted third party
  - Alternative to everyone having a secret key for possible communications with everyone else:
    - have <u>one trusted party</u>
    - everyone holds shared secret keys to communicate with that party
  - When you want to communicate securely with someone else:
    - you ask the trusted third party (TTP) to send you a key (encrypted using the secret key you share with the third party)
    - You use this key in communications with the other party
    - Kerberos is a well known scheme that takes this approach

Key Establishment:
Option 2 - Simplified Kerberos Protocol



Symmetric ciphers:

the key establishment problem

- Option 3 – Diffie-Hellman key agreement alg'thm
  - 1976: Whitfield Diffie and Martin Hellman published a radical method for forming symmetric cryptographic keys
  - Made use of certain mathematical methods:
    - modular exponentiation, with careful parameter choices
  - Without prior arrangement, two parties without a secure distribution channel can agree on shared secret known only to them
    - Each sends the other a mathematical function of their chosen secret, and these can be combined to form shared secret
    - An eavesdropper **cannot** determine the shared secret by listening to the messages exchanged by the two parties
      - unless they can break the discrete log problem

Diffie-Hellman key agreement

for system parameters p (a prime) and g

Alice picks random integer a

$g^a \bmod p$

Bob picks random integer b

$g^b \bmod p$

Alice computes the shared secret
$(g^b)^a = g^{ab} \bmod p$

Bob computes the same secret
$(g^a)^b = g^{ab} \bmod p.$

Symmetric ciphers:
the key establishment problem

- Option 3 – Diffie-Hellman key agreement alg.
  - Q: Does that mean Alice and Bob can now:
    - establish a shared secret key over an insecure channel (yes),
    - and can communicate securely, without worrying about Carol (the attacker)?
  - A: Actually, no.
    - If Carol is clever, she can still eavesdrop …
    - Problem is there is no authentication of the communicating parties.
      - While they are communicating to establish the key, Alice and Bob have no assurance about who they are communicating with, so a man-in-the-middle (Carol) attack is possible
  - Important aspect of Diffie-Hellman is application of a new type of mathematics in cryptography: lead to *asymmetric cipher systems*

Outline

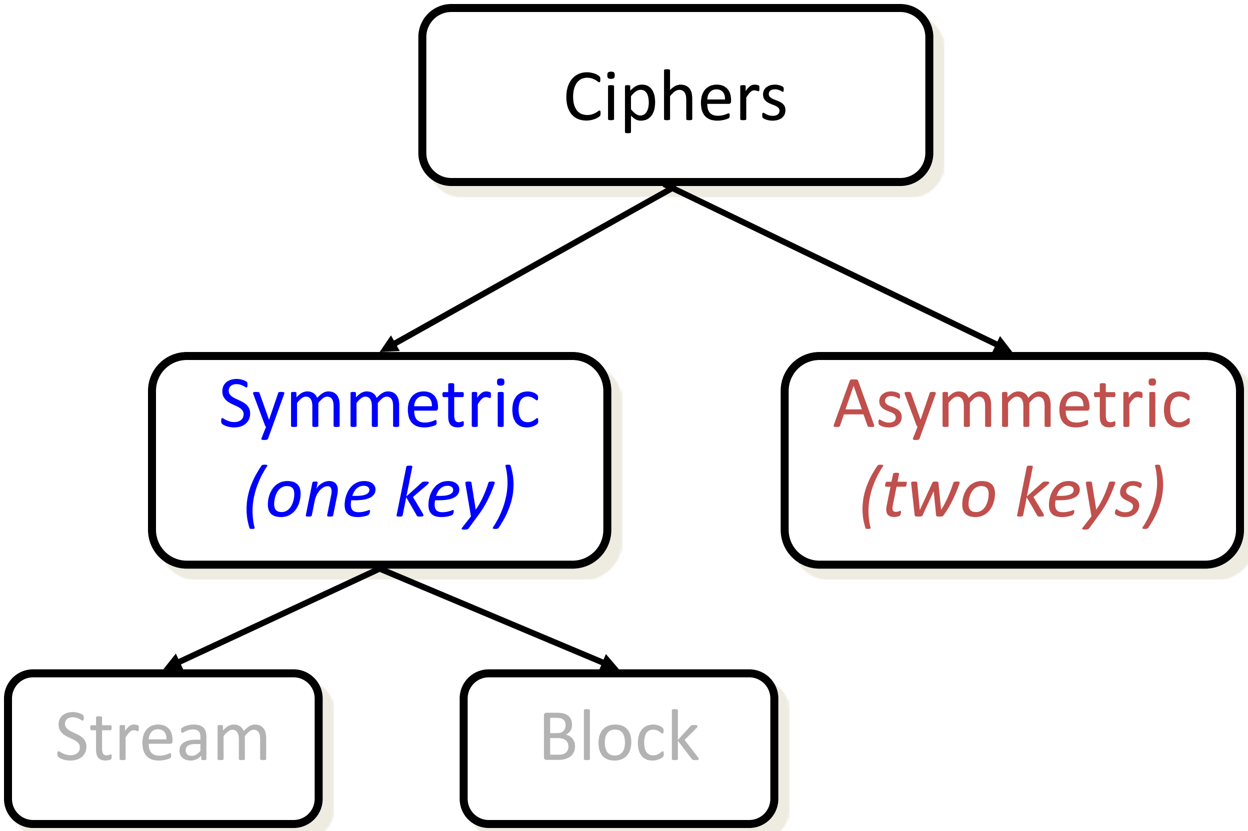
- Symmetric ciphers: key establishment problem
  - Pre-distribution
  - Trusted Third Party
  - Diffie-Hellman Key Exchange
- Asymmetric (public key) ciphers
  - For confidentiality
  - For authentication
  - For integrity
- Public keys and required infrastructure
  - The spoofing problem
  - Digital certificates
  - Public key infrastructure

Asymmetric ciphers

- Symmetric ciphers: encryption & decryption keys are the same
- Asymmetric ciphers: encryption & decryption keys are different:
  - The keys are related, but it is computationally infeasible to derive one key from the other
- Each participant needs a pair of keys:
  - One key kept private: K_{priv} but the other key is made public: K_{pub}
    - Asymmetric crypto also known as public key cryptography
- Security of system depends on:
  - The strength of the algorithm,
  - The key size, and
  - Confidentiality of the private key K_{priv}

Asymmetric ciphers

In taxonomy of modern ciphers:



Asymmetric Ciphers:

Key distribution problem solved!

- Key distribution is much simpler than for symmetric ciphers
  - as anyone may know the public key
- Each participant needs to:
  - create their asymmetric key pair
  - publish the public key
  - keep the private key secret
- If there are *n* participants, then a total of *n* <u>key pairs</u> must be created

Using asymmetric ciphers

- How do you get to know someone's public key?
  - They could:
    - Give it to you directly (have it in their email signature)
    - Put it up on their website (Check AusCERT site for theirs)
    - Make it available through a public keyserver
      - Example: http://pgp.mit.edu:11371/

Outline

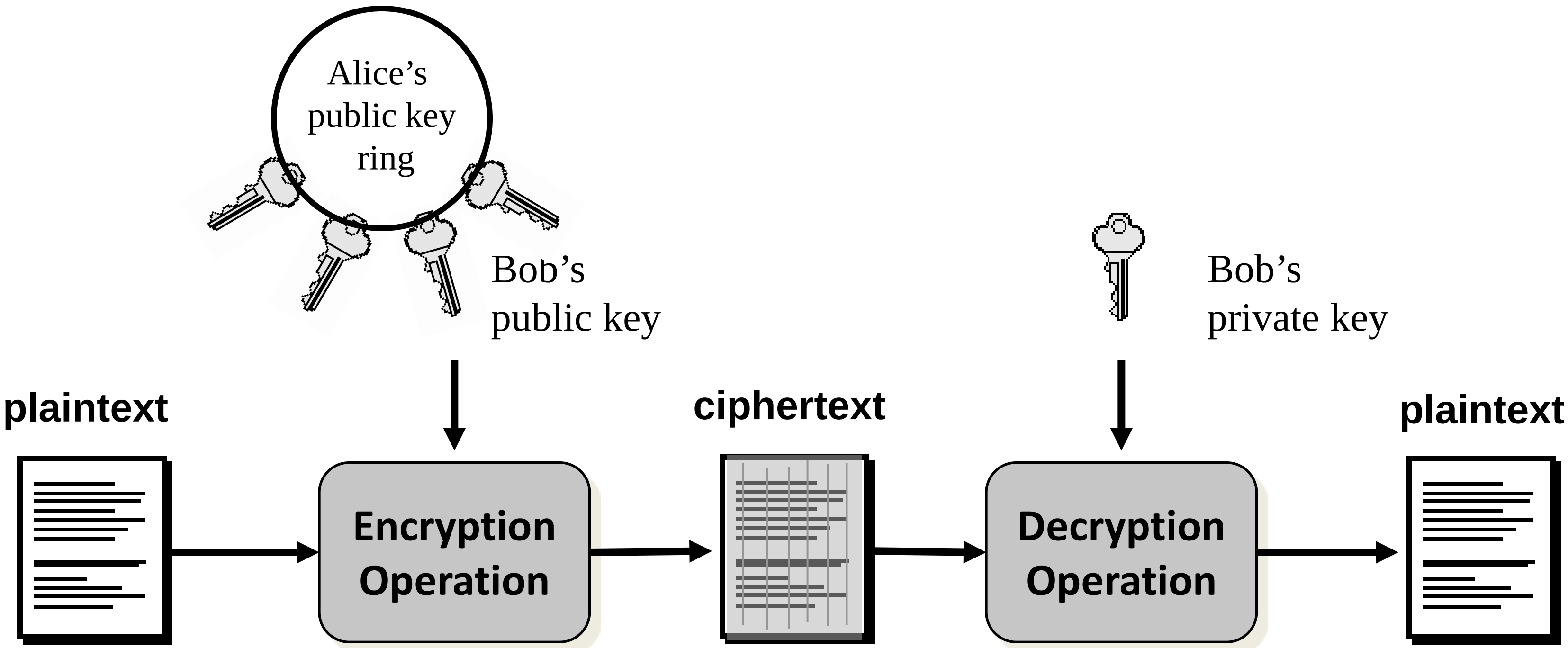
- Symmetric ciphers: key establishment problem
  - Pre-distribution
  - Trusted Third Party
  - Diffie-Hellman Key Exchange
- Asymmetric (public key) ciphers
  - For confidentiality
  - For authentication
  - For integrity
- Public keys and required infrastructure
  - The spoofing problem
  - Digital certificates
  - Public key infrastructure

Using asymmetric ciphers
for confidentiality

- For Alice to send a confidential message to Bob, Alice needs to:
  1. Know Bob's public key
  2. Encrypt the plaintext message P using the asymmetric encryption algorithm and Bob's public key
  3. Send the resulting ciphertext to Bob.
- For Bob to recover the message,
  – Bob uses his private key and the asymmetric decryption algorithm to decrypt the ciphertext
- Only Bob knows the corresponding private key, so only Bob can decrypt the encrypted message

Asymmetric ciphers for confidentiality

For a message Alice sends to Bob



$C = \mathrm{E}(P, K_{B\_pub})$

$P = \mathrm{D}(C, K_{B\_Priv})$

Asymmetric ciphers for confidentiality

Basic operation

- Notation:
  - Encryption: $C = E(P,K_{pub})$
  - Decryption: $P = D(C,K_{priv})$
- The public key of the recipient is used for encryption
- The private key of the recipient is used for decryption
- Only the private key must remain confidential (i.e. known only to its owner)
  - Anyone is allowed to know the public key
  - Only the owner may know the associated private key

Asymmetric Ciphers:

Example: ElGamal Cryptosystem

- Designed by Taher ElGamal
  - Published in 1985
- Like Diffie Hellman, it relies on difficulty of discrete logarithms for security
- Unlike Diffie-Hellman, this is designed for encryption, not key agreement
- Feature:
  - Ciphertext is twice the length of plaintext
  - Ciphertext is randomised – multiple encryptions of same plaintext will produce different ciphertexts

Asymmetric Ciphers:

Example: ElGamal Cryptosystem

- System parameters p and g same as for Diffie-Hellman
- Alice picks a random integer a as her fixed private key and publishes her public key: $K_{A\_Pub} = g^a \bmod p$.
- To **encrypt** a message to send to Alice,
  1. Bob first encodes it as a number, m.
  2. Bob picks a random integer r , and computes

     $g^r \bmod p$, $(g^a)^r = g^{ar} \bmod p$ and $m \times g^{ar} \bmod p$
  3. Bob sends the ciphertext ($g^r \bmod p$, $m \times g^{ar} \bmod p$) to Alice.
- To **recover** the message,
  1. Alice computes the shared secret $(g^r)^a = g^{ar} \bmod p$
  2. Recovers $m = (m \times g^{ar} \bmod p) / (g^{ar} \bmod p)$

Asymmetric Ciphers:
Example: RSA Cryptosystem

- Rivest-Shamir-Adleman, MIT, 1977
- Most well-known asymmetric cryptosystem
  - Can use for encryption, also digital signature scheme
- Based on difficulty of factorising large integers
  - Calculations performed using modular arithmetic
    - Where the modulus is product of two large primes
  - Breaking RSA is no harder than factorising the modulus (although this is a really large number – typically over 1024 bits for corporate use)
- US patent owned by RSA company – expired in 2000
- Historical Note: U.K. cryptographer Clifford Cocks invented an RSA variant in 1973

Asymmetric Ciphers:
Example: RSA Cryptosystem

- RSA Key Generation:
  - Randomly choose two large primes p and q
  - Calculate: $n = pq$
  - Choose a public exponent e [where e and $(p-1)(q-1)$ have no common factors]
  - Calculate the corresponding private exponent d using the secret knowledge of primes p and q [detail outside the scope of this course]
- Public key is (n, e) and may be available to anyone.
- Private key (or exponent) is d and must be kept confidential
- Primes p and q must be kept confidential (possibly destroyed).

Asymmetric Ciphers:
RSA Cryptosystem

- Encryption (for confidentiality):
  1. Encode p/text message as string of integers; $1 \leq \boldsymbol{m} \leq n$
  2. For each of these integers, sender encrypts message (integer) m by calculating: $\boldsymbol{c = m^e \bmod n}$ using recipient's public key: (n, e)
  3. Ciphertext consists of concatenation of each of these integers

- Decryption:
  1. For each of the ciphertext integers,

     Recipient decrypts c/text (integer) c by calculating: $\boldsymbol{m = c^d \bmod n}$

     using recipient's private key: (n, d)
  2. Then decode from integers and concatenate to recover plaintext

Asymmetric Ciphers:
Example: Elliptic Curve Cryptography

- First proposed in 1985
- Uses algebraic group defined on a set of points on an elliptic curve
- Any cryptosystem based on discrete logarithm problem (such as ElGamal) can work on elliptic curves
  - So breaking the cryptosystem is as difficult as solving EC discrete log problem
- Commercialised, particularly by Certicom, in 1990s
- Advantage over other asymmetric ciphers:
  - Smaller key size and smaller ciphertext size than RSA
  - Provides same level of security with smaller key sizes

Symmetric & Asymmetric Ciphers:
Key length comparison

AES Key Size (Symmetric key)	RSA Key Size	Elliptic curve Key Size
-	1024	163
128	3072	256
192	7680	384
256	15360	512

Symmetric and asymmetric ciphers offering comparable security

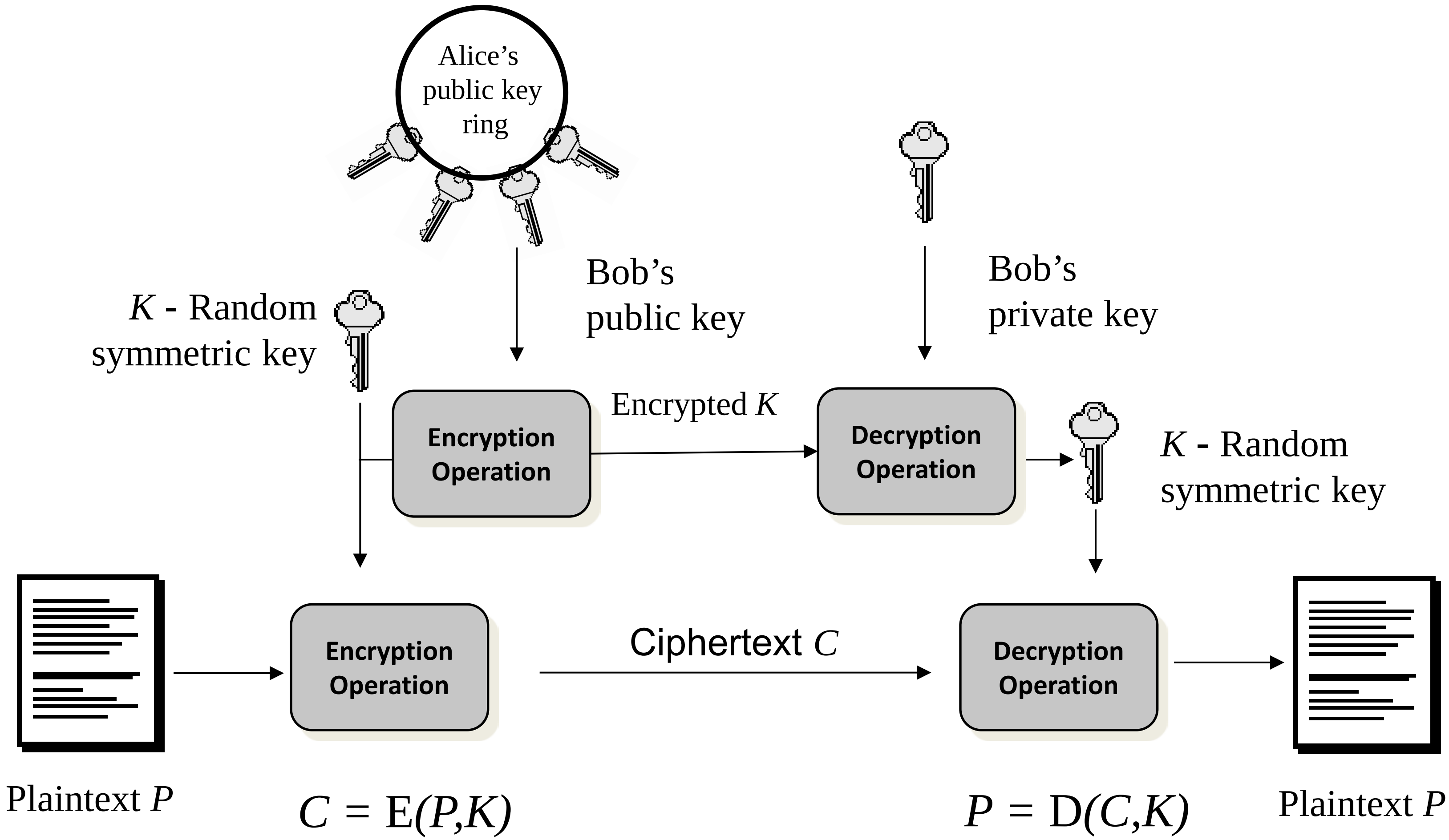
Asymmetric ciphers for confidentiality

Hybrid cryptosystems

- Symmetric ciphers are much faster than currently available asymmetric ciphers
  - because they are less computationally expensive
- However, asymmetric ciphers greatly simplify key distribution
- So, many cryptosystems use a combination:
  - First, the asymmetric cipher is used to provide confidentiality for a particular short message:
    - a randomly chosen shared secret key to be used with a particular symmetric cipher.
  - Then the symmetric cipher is used with that shared secret key for encrypting the bulk data.

Hybrid Cryptosystems

Providing confidentiality (from Alice to Bob):



Outline

- Symmetric ciphers: key establishment problem
  - Pre-distribution
  - Trusted Third Party
  - Diffie-Hellman Key Exchange
- Asymmetric (public key) ciphers
  - For confidentiality
  - For authentication
  - For integrity
- Public keys and required infrastructure
  - The spoofing problem
  - Digital certificates
  - Public key infrastructure

Using asymmetric ciphers

for authentication of message sender

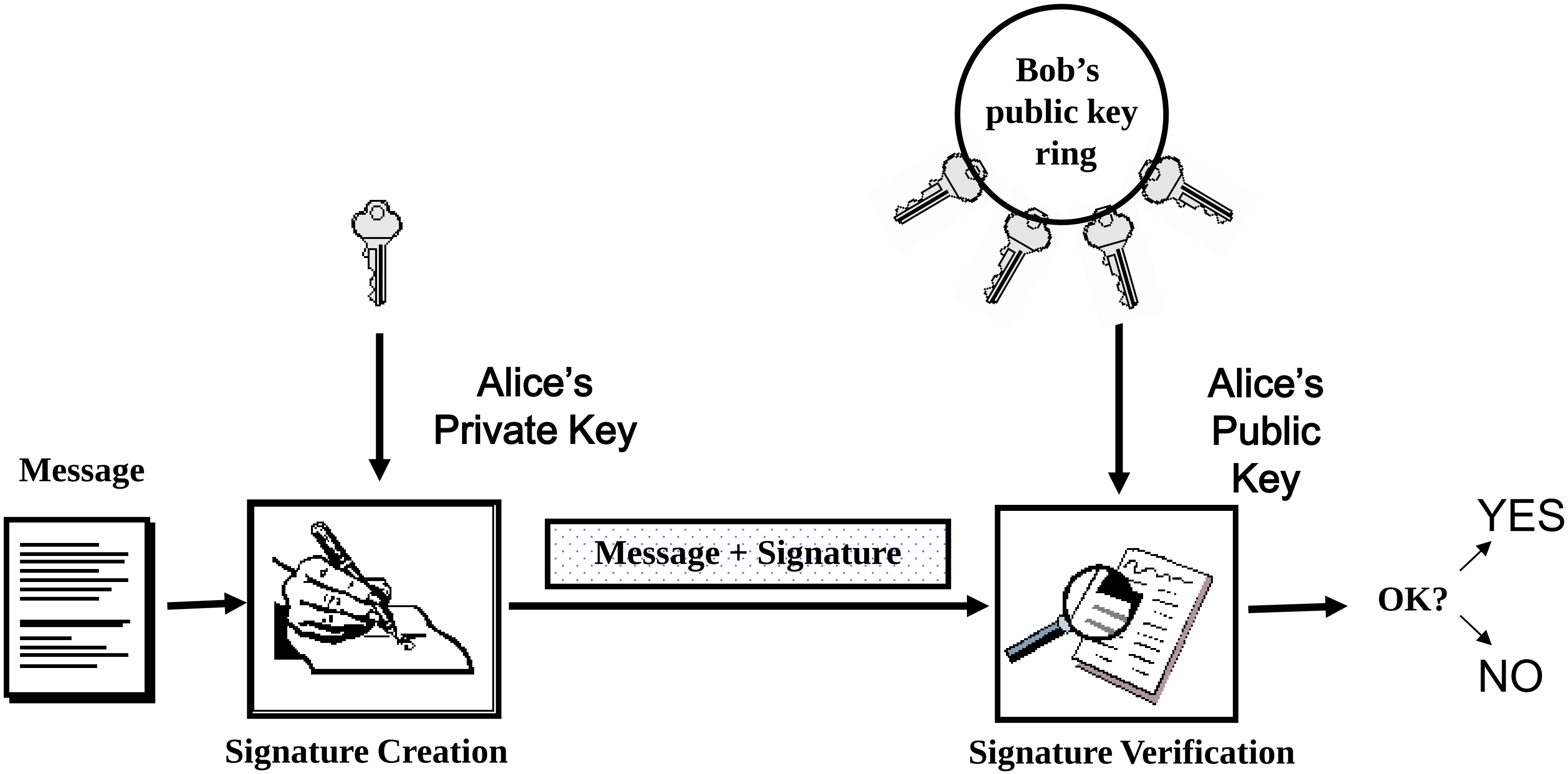
- Using asymmetric cryptography, each entity has a unique key pair:
  - One key is kept private (known only to owner of key)
  - The other key is public (anyone can know this)
- A *private key* can be used by the owner to *form a digital signature* for a particular message or file
- The corresponding *public key* can be used by others to verify the digital signature on the message
  - Provides *authentication* of the sender for a particular message
  - Since only the signer knows the private key, only the signer could have created the digital signature

Using asymmetric ciphers
Authentication - Digital signatures

- Similarities to handwritten signatures:
  - unique to signer
  - verifiable
  - legally binding
- Important differences:
  - digital signature is *different* for every document
  - digital signature must be produced and verified by a machine
- A digital signature is completely different from a *digitized signature*

Authentication - Digital Signatures:

Basic Operation: Alice signs a message



Using asymmetric ciphers

Digital signatures - Common examples:

- The most widely used digital signature schemes are:
  - **RSA**:
    - exploits symmetry in RSA encryption/decryption algorithms.
    - Relies on the difficulty of factoring large numbers.
  - **DSA** (Digital Signature Algorithm):
    - Also referred to as DSS (Digital Signature Standard)
    - Relies on the difficulty of solving the discrete log problem.
  - **ECDSA** (Elliptic Curve DSA):
- DSA, RSA, and ECDSA are currently the only FIPS-approved methods for digital signatures.

Using asymmetric ciphers

Digital signatures and hash functions

- For efficiency, signature schemes typically use a hash function to reduce amount of material processed using asymmetric cipher
- To create a signature for message *m*,
  - first compute the hash of *m,* and
  - then proceed with the other steps of the signing method

- Any change in message *m* should result in a different hash value, so digital signatures provide some assurance of message integrity
  - Important to use the same hash function during both:
    - the signature creation process, and
    - the signature verification processes

Using asymmetric ciphers

Digital Signatures: RSA Signatures

- Key generation same as RSA encryption:
  - The public signing key is **(*n*, *e*)** - should be available to anyone.
  - The private signing key is ***d*** - must be kept confidential.
- Signature generation of message *m:*
  - Calculate hash: $h = H(m)$ with $0 < h < n$
  - Calculate: $s = h^d \bmod n$
- Verification of claimed signature ***s*** on message ***m***
  - Calculate hash: $h = H(m)$ with $0 < h < n$
  - Calculate: $h' = s^e \bmod n$
  - If $h' = h$ then accept signature, otherwise reject it

Using asymmetric ciphers

Digital signatures and security services

- Digital signatures provide:
  - Authentication of message sender
  - Some assurance of message integrity
- They can also provide *non-repudiation:*
  - A third party (judge) can decide if a specific party signed a message
  - This cannot be achieved with symmetric key authentication, such as with a MAC
  - This makes digital signatures particularly useful for electronic commerce applications

Outline

- Symmetric ciphers: key establishment problem
  - Pre-distribution
  - Trusted Third Party
  - Diffie-Hellman Key Exchange
- Asymmetric (public key) ciphers
  - For confidentiality
  - For authentication
  - For integrity
- Public keys and required infrastructure
  - The spoofing problem
  - Digital certificates
  - Public key infrastructure

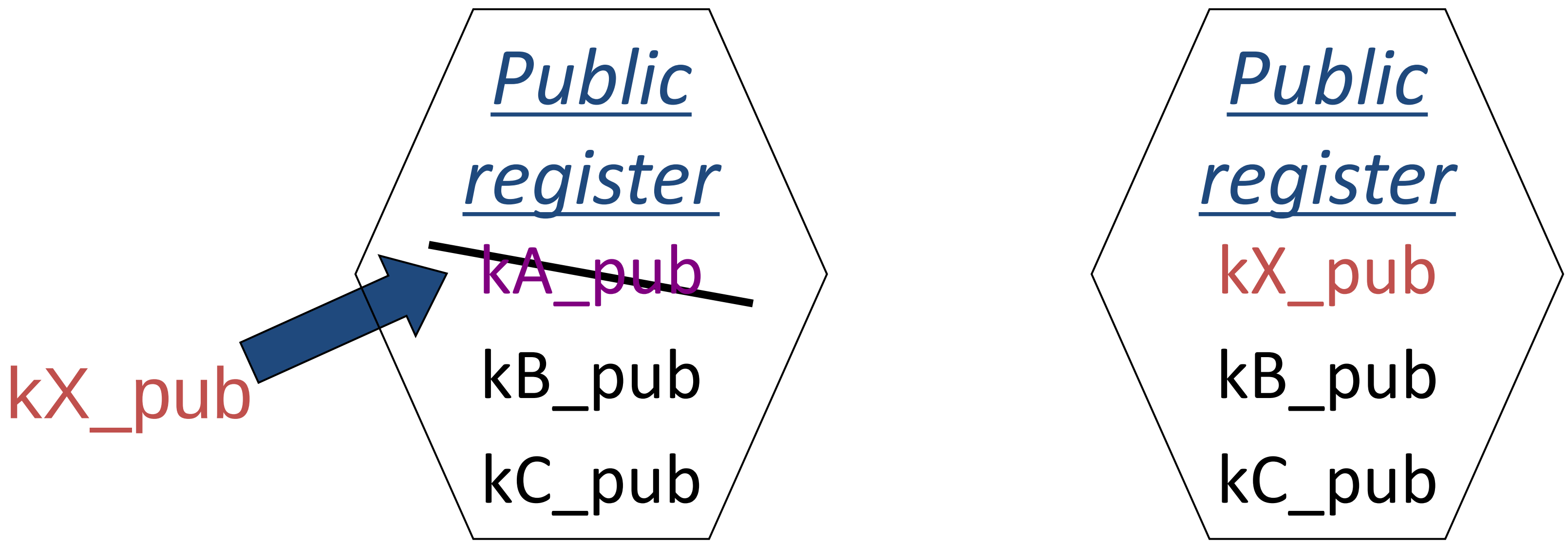
Public keys and infrastructure:

- Alice's public keys may be used by others to:
  - **Encrypt messages sent to Alice**
    - For confidentiality (Example: shared symmetric key)
  - **Verify Alice's signature** on messages
    - For authentication of Alice as message sender, and also assurance of message integrity
- Alice makes the key available on a public site
- **Question:** What might happen if another person (an attacker, say Carol) replaces Alice's public key with another public key chosen by the attacker (called a spoofing attack)?

Diagram

Public key cryptography:

The Spoofing Problem



- How does this affect the security of:
  - a confidential message sent to A? Why?
  - a digital signature on a message received from A? Why?

Public key crypto: spoofing problem

- Major issue associated with the use of asymmetric cryptography is the <u>integrity</u> and <u>trustworthiness</u> of public keys:
  - How can a user be sure who a public key belongs to?
  - How can a user be sure a public key has not been altered - intentionally or unintentionally?
- How can public keys be made available *in a trusted way*?
  - Use *digital certificates* issued by a trusted third party
    - a Certification Authority (CA).

Outline

- Symmetric ciphers: key establishment problem
  - Pre-distribution
  - Trusted Third Party
  - Diffie-Hellman Key Exchange
- Asymmetric (public key) ciphers
  - For confidentiality
  - For authentication
  - For integrity
- Public keys and required infrastructure
  - The spoofing problem
  - Digital certificates
  - Public key infrastructure

Public keys and infrastructure:
Digital certificates

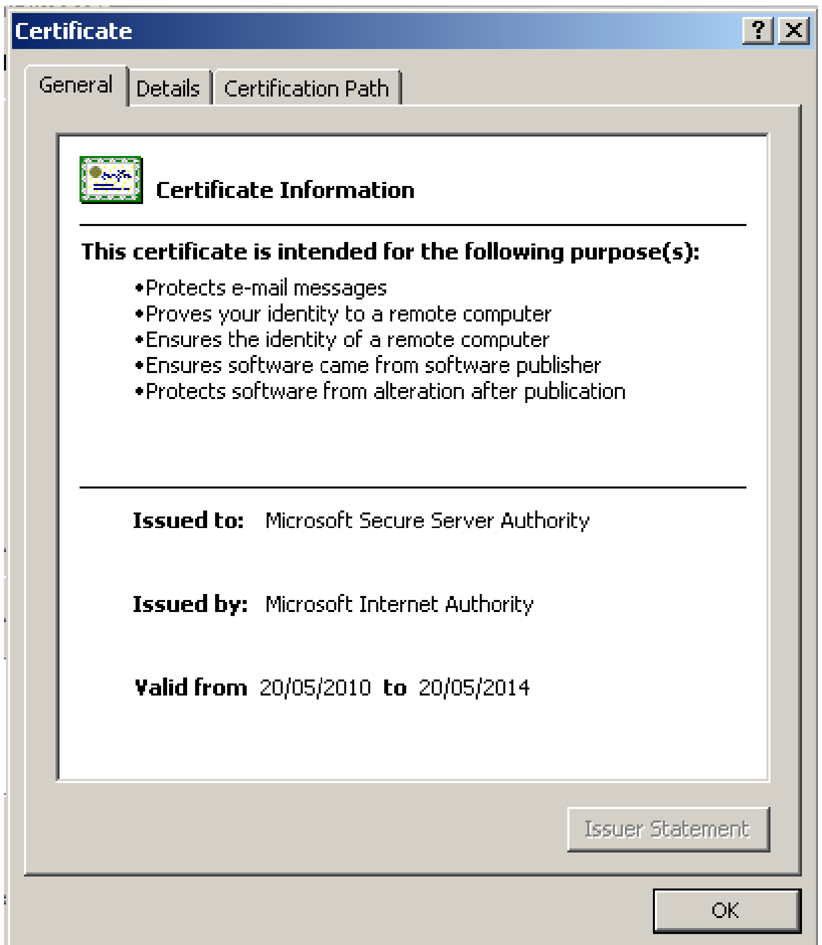
- Digital certificates solve the issue of binding a public key to an entity
  - one of the major legal issues with this technology
- A digital certificate contains:
  - the user’s public key
  - plus the user’s ID
  - plus some other information e.g. validity period
- A Certificate Authority (CA) *creates and digitally signs* the certificate
  - The CA is vouching for the information

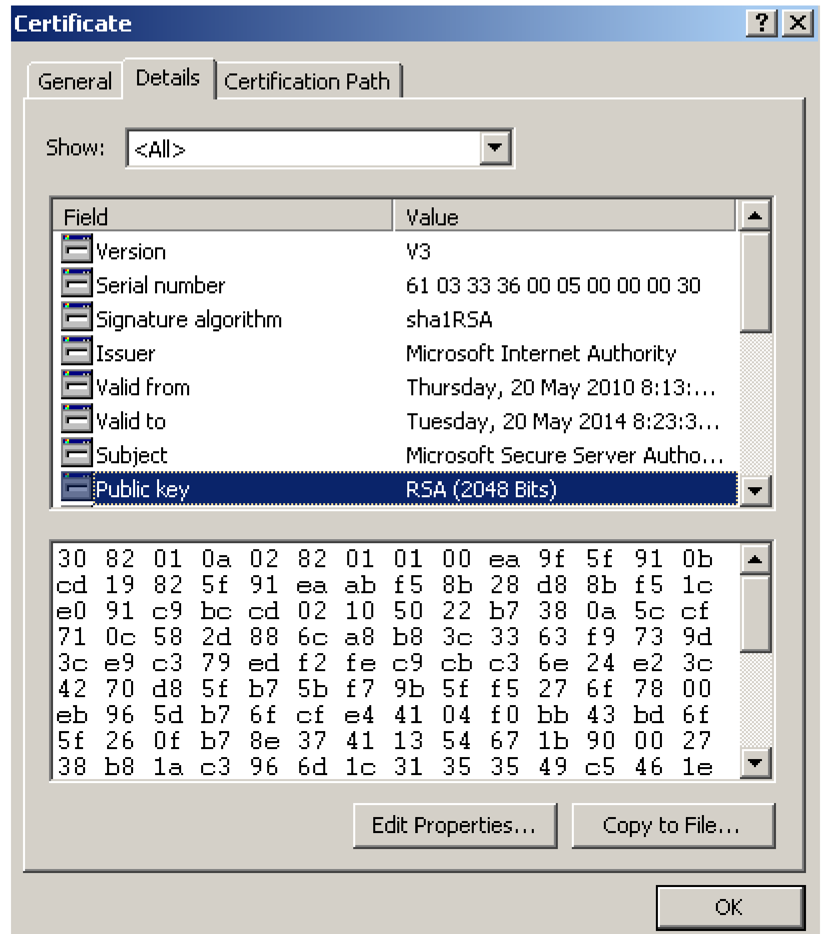
Public keys and infrastructure:
Digital certificates

- Format for digital certificates: X.509 standard
  - Most widely used standard (still evolving: now v3)
  - Recommended by International Telecommunication Union (ITU-T)
  - Important fields in X.509 digital certificates are:
    - Version number
    - Serial Number (set by the CA)
    - Signature Algorithm identifier (Algorithm used for dig sigs)
    - Issuer (Name of the CA)
    - Subject (Name of entity to which certificate has been issued)
    - Public Key Information
    - Validity period (certificate should not be used outside this time)
    - Digital signature (of the certificate, signed by the CA)

Public keys and infrastructure:

Digital certificate - example

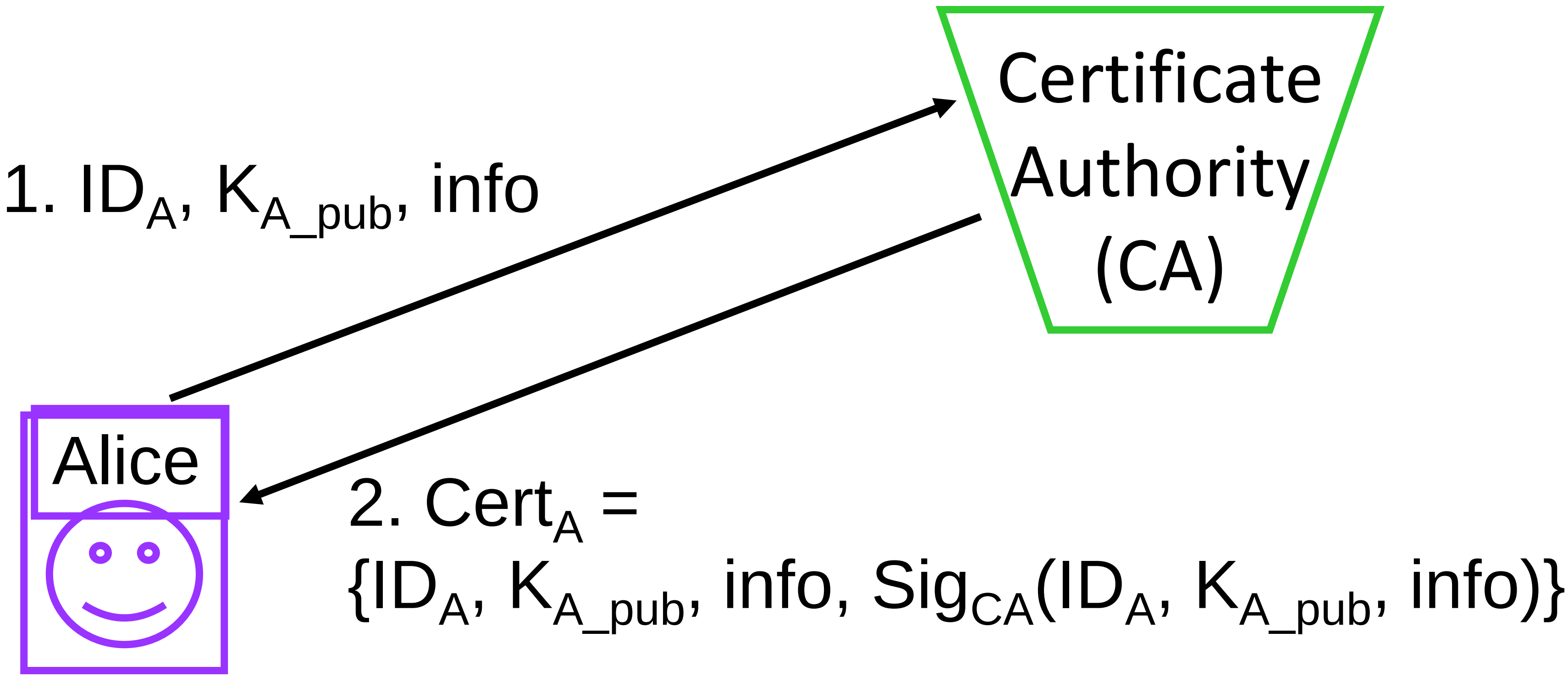




Diagram

Public keys & infrastructure: digital certificates

How does Alice obtain a digital certificate?



Public keys and infrastructure:

Digital certificates

- How does Alice obtain a digital certificate for her public key?
  1. Alice generates a key pair (keeps the private key secret) and sends to the CA:
     - her identity details, ID_A
     - her public key, K_{A_pub}, and
     - any other information required by the CA
  2. The CA then:
     - performs any required checks to verify Alice's identity,
     - creates a certificate containing ID_A and K_{A_pub}
     - sends the certificate, $Cert_A$, to Alice

- To trust a certificate, you need to trust the CA that issued the certificate has performed the necessary checks

Public keys and infrastructure:

Using digital certificates

- **How do I use Alice's digital certificate?**
- Suppose Bob wants to send a *confidential* message to Alice:

  $C = E(M, K_{A\_pub})$

  1. Bob obtains Alice's public key for encryption:
     - Bob obtains $Cert_A$
     - Bob verifies $Cert_A$
     - Bob obtains K_{A_pub} from $Cert_A$
  2. Bob uses K_{A_pub} to encrypt the message M
- If Bob:
  - <u>trusts</u> the CA that issued $Cert_A$ and
  - is certain of the CA's public key

  then Bob can be sure that only Alice will be able to decrypt the ciphertext message he sends to Alice

Public keys and infrastructure:

Using digital certificates

- **How do I use a digital certificate?**
- Suppose Alice sends a *digitally signed* message to Bob: {M, SigA, CertA}, then Bob can *verify* it:
  1. Bob obtains Alice's public signing key:
     - Bob obtains $Cert_A$
     - Bob verifies $Cert_A$
       - uses the CA's public key to verify the CA's signature on CertA
     - Bob obtains K_{A_pub} from $Cert_A$
  2. Bob uses K_{A_pub} to verify Sig_A

- If Bob:
  - trusts the CA that issued $Cert_A$ and is certain of CA's public key,
  - and if the signature verification is OK
  - then Bob has a valid signature on the message from Alice

Public keys and infrastructure:

Digital certificates

- Some questions on certificates:
  1. What advantage is there for Alice in having a digital certificate $Cert_A$?
  2. Who can have access to Alice's certificate, $Cert_A$?
  3. Why do we verify the signature in $Cert_A$?
  4. What does Bob need in order to verify $Cert_A$?
  5. After someone has verified $Cert_A$, of what can they be assured?

Public keys and infrastructure:

Digital certificates

- **Digital certificates and Trust:**
- **Q:** Can I trust Alice's digital certificate?
- *A: Do I trust the CA who issued it - and is that really the CA's public key?*

- To verify the CA's signature on $Cert_A$, I need to use the CA's public key.
  - If I don't know this already, I can obtain it from the CA's certificate
  - To verify that certificate, I need to use the public key of the CA who issued that certificate …
- Some infrastructure is required to enable implementation of public key cryptography

Outline

- Symmetric ciphers: key establishment problem
  - Pre-distribution
  - Trusted Third Party
  - Diffie-Hellman Key Exchange
- Asymmetric (public key) ciphers
  - For confidentiality
  - For authentication
  - For integrity
- Public keys and required infrastructure
  - The spoofing problem
  - Digital certificates
  - Public key infrastructure

Public key crypto: infrastructure

- Public key cryptography needs a **P**ublic **K**ey **I**nfrastructure (PKI) to provide security services
- A **PKI** is a set of
  - **Policies** (to define the rules for managing certificates)
  - **Products** (hardware and/or software) (to implement the policies and generate, store and manage certificates)
  - **Procedures** (related to key management)

that enable users to implement public key cryptography - often in large, distributed settings

Public key crypto: infrastructure

- Critical component of a PKI is the Certification Authority (CA)
  - The CA *creates and issues digital certificates* to other parties
  - The level of trust you can place in a certificate depends on the amount of checking a CA does to establish the credentials of requester before providing a certificate
- Who is the CA?
  - Some large organisations have their own CA
  - Commercial CA's charge to provide certificates, and may offer various grades
    - More thorough checking generally means greater cost

Public key crypto: infrastructure

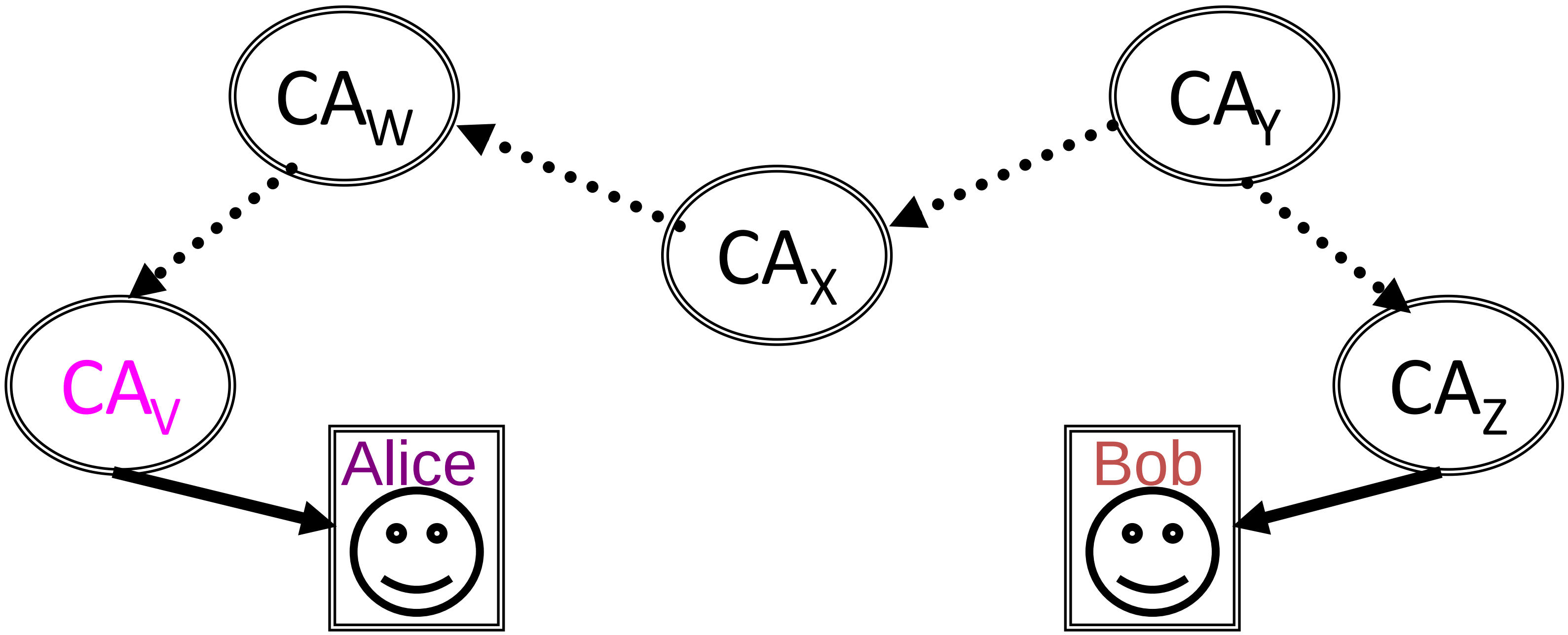
- **Q: Are all digital certificates trustworthy?**
- A: No!
- Known cases of spoofed digital certificates:
  - Dec 2012: CA linked to TURKTRUST issued fraudulant Cert for Google
  - July 2011: DigiNotar - over 500 fraudulent certificates issued, including cert's for Google, Yahoo!, Mozilla, Wordpress
  - March 2011: Nine fraudulent certificates issued by Comodo, supposedly to various google, yahoo and skype servers
  - Reaction: certificates can be revoked (cancelled before expiry date)
    - Check Certificate Revocation Lists (CRL) or Online Certificate Status Protocol (OCSP) before trusting certificate
- Also possible to create a self-signed certificate:
  - The certificate creator is also verifying the contents
  - Careful about trusting these! Would you trust credentials in a home-made driver's licence?

Public key crypto: infrastructure

- Certification trust pathways:
  - In large deployments all users may not have the same CA
  - To create a *chain of trust* between users, CAs must have their public keys certified by other CAs
  - The user may need to verify multiple certificates in order to establish a trust pathway

- All users need to trust one or more CAs in order to start constructing a trust pathway

PKI: Certification paths



For Bob to be assured of the integrity of Alice's public key, need to create a path of trust from Alice to Bob

PKI: certification paths

- Bob wants to use Alice's public key
- Bob has Alice's digital certificate $Cert_A$ containing Alice's public key issued by CA_V
  - CA_V may need to provide certificate for its' public key
  - CA_V may obtain $Cert_V$ from CA_W
  - CA_W may obtain $Cert_W$ from CA_X
  - CA_X may obtain $Cert_X$ from CA_Y

- Bob's own digital certificate is from CA_Z
  - CA_Z's $Cert_Z$ is issued by CA_Y
- A *certification path* is established

PKI: trust models

- Trust relationships
  - between different Certificate Authorities, and
  - between Certificate Authorities and end users,

  define PKI trust models

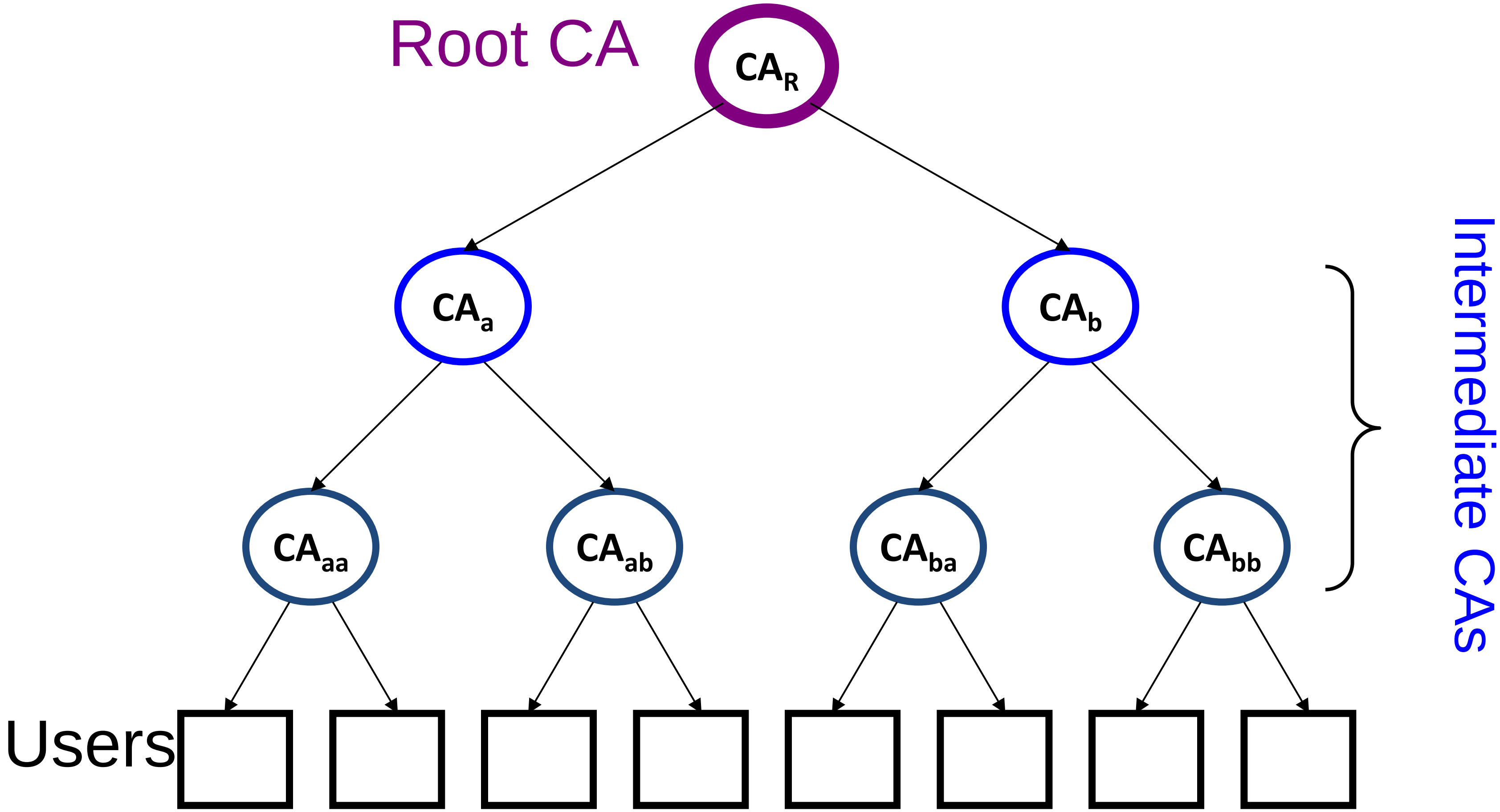
- Common PKI Trust models are:
  - Hierarchical
    - Strict hierarchical
    - Distributed trust architectures
  - User-centric
  - Browser

PKI: trust models

- Strict Hierarchical Model
  - Tree structure:
    - Single root CA
    - Users are leaves of the tree
    - Each node is certified by its immediate parent CA
  - Highly regulated:
    - Each CA must follow rules regarding to whom they may issue certificates
  - Root CA:
    - Starting point for trust
    - All users trust the root CA, and must receive its public key through a secure out-of-band channel

PKI: trust models

Strict hierarchical model

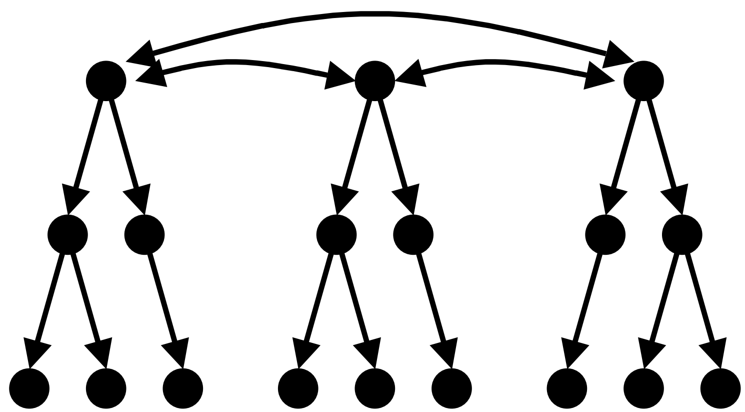


PKI: trust models

- Strict Hierarchical Model
- <u>Advantages</u>:
  - works well in highly-structured setting such as military and government
  - unique certification path between two entities (so finding certification paths is trivial)
  - scales well to larger systems
- <u>Disadvantages</u>:
  - need a trusted third party (root CA)
  - ‘single point-of-failure’ target
  - If any node is compromised, trust impact on all entities stemming from that node
  - Does not work well for global implementation (who is root TTP?)

PKI: trust models

- Distributed trust hierarchical architectures:
  - Interconnection of multiple hierarchies
  - No single root CA, multiple cross-certified root CAs
  - Trust is distributed among the root CAs



Cross certified strict hierarchies

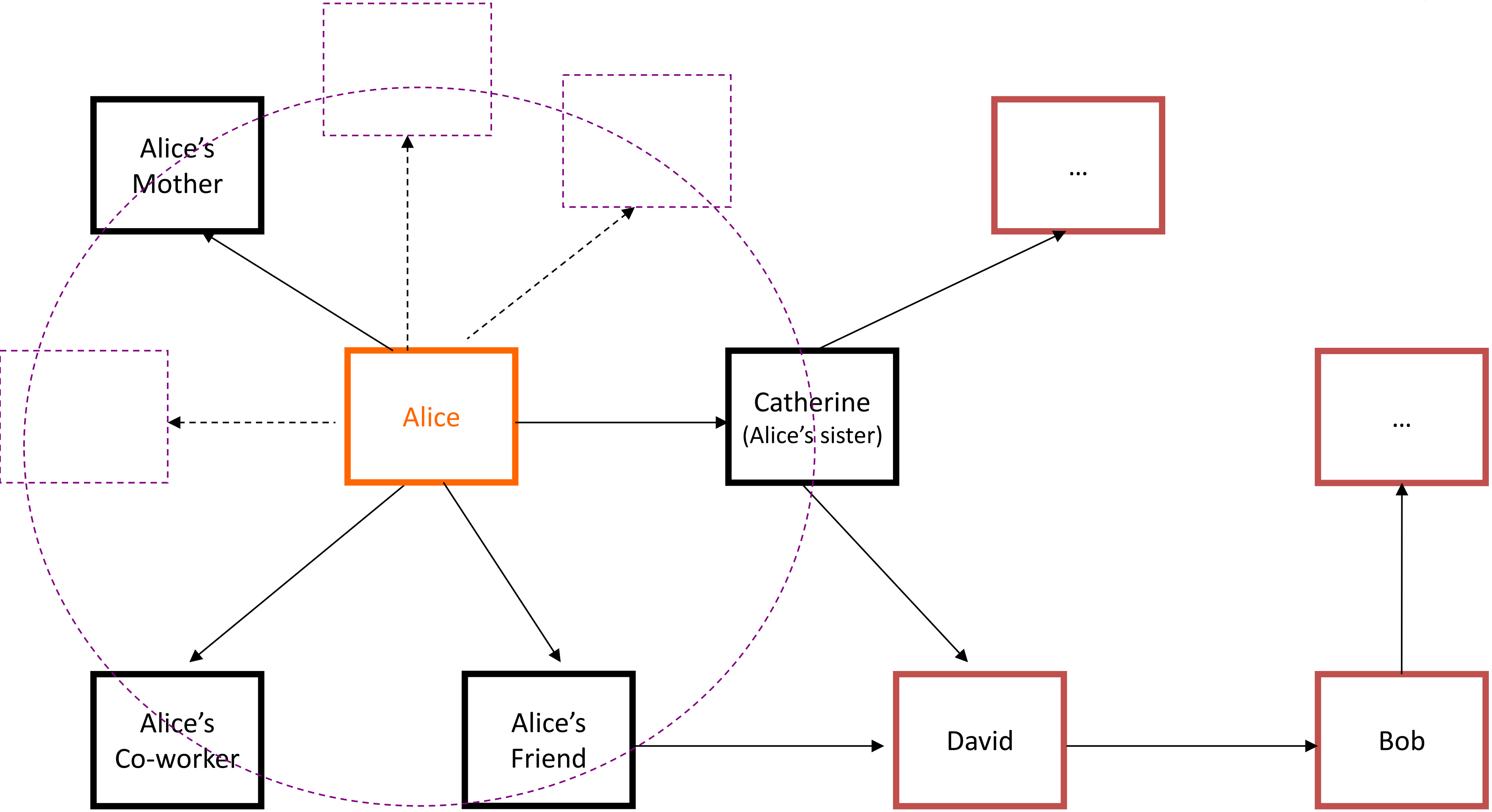
PKI: trust models

- User-centric model
  – Each user has a key ring containing public keys of other users they trust
  – Users are **completely responsible** for deciding which public keys to trust
  – Public keys can be distributed by key servers and verified by fingerprints
  – Each user may act as a CA, signing public keys that they will trust
  – OpenPGP Public Key Server: http://pgpkeys.mit.edu/
  – <u>Example</u>: *Pretty Good Privacy (PGP)* 'Web of Trust'
- PGP or GPG (Gnu Privacy Guard) – What is the difference?

Diagram

PKI: trust models

User-centric model



PKI: trust models

- User-centric model
- <u>Advantages</u>:
  - Simple and free
  - Works well for a small number of users
  - Does not require expensive infrastructure to operate
  - User-driven grass roots operation

- <u>Disadvantages</u>:
  - Relies on human judgment
    - Works well with technical users aware of the issues, but not general public
  - Not appropriate for trust-sensitive areas such as finance and government

PKI: trust models

- Browser model
  - Used by most well known browsers including *Mozilla Firefox* and *Microsoft Internet Explorer*
  - Some CA certificates pre-installed in the browser
    - Installed certificates are used as trusted ‘root’ CA certificates for verifying incoming certificates
  - The browser user is trusting the browser vendor who supplied the installed certificates, rather than a root CA
  - May also include list of ‘untrusted’ certificates
    - Check your browser certs for the fraudulent Comodo and DigiNotar certs revoked in 2011

PKI: trust models

- Browser model limitations:
  - Certification path processing is limited
    - Incoming certificates can only be verified by available 'trusted' certs
  - List of trusted certificates controlled by user - not well protected from modification attacks
    - If prompted, many users automatically accept incoming certificates that cannot be verified by 'trusted certificates'
  - Cross certification and revocation may not be supported
    - Limited opportunity for expansion and limited trust options available
  - No formal legal agreement established between users and CAs
    - Liability rests with the users and not with the CAs

PKI: trust models

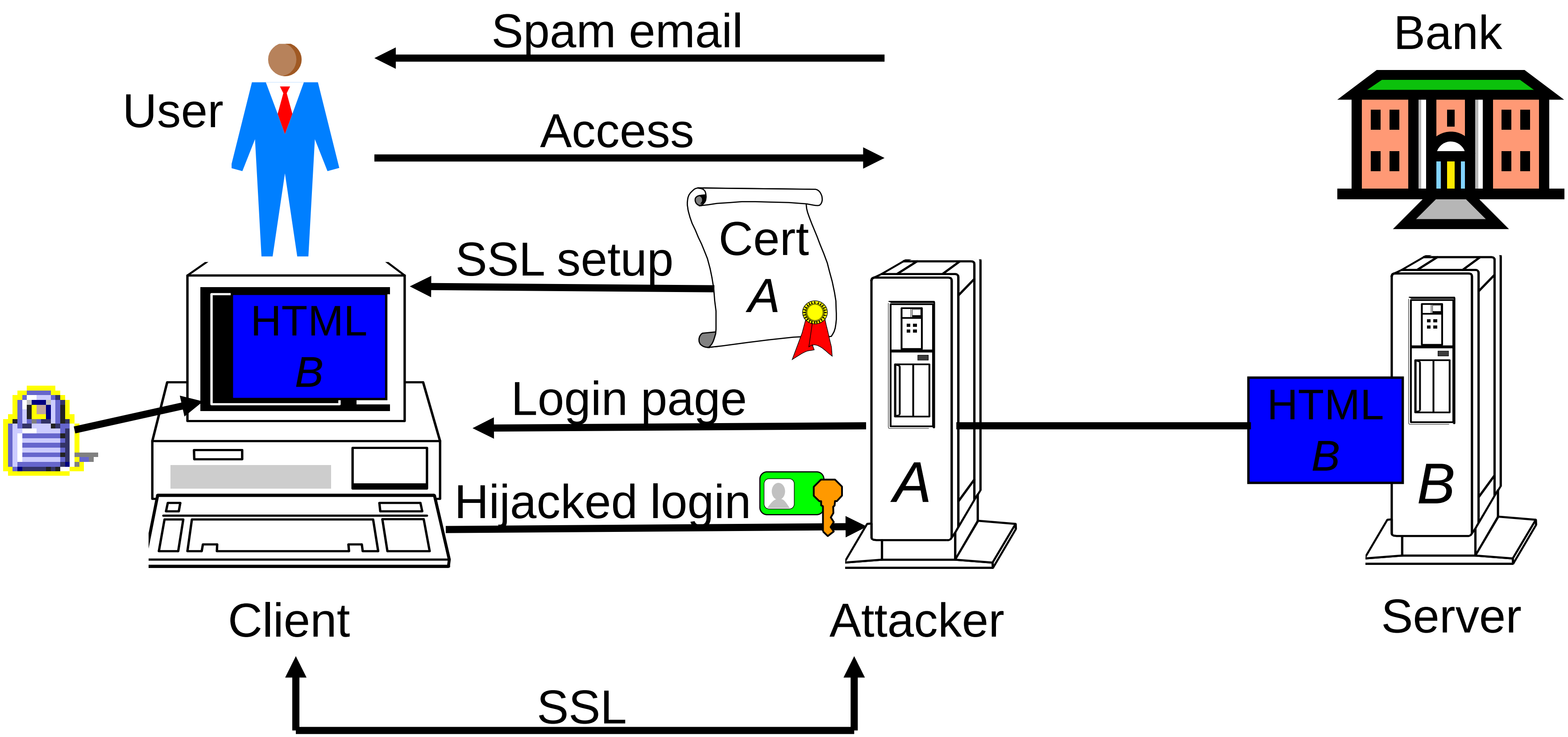
- Browser model:
- What are the security implications of user control over certificate acceptance?
  - Consider the case where certificates are used to authenticate an internet bank site.
  - Many users look for a visual symbol (padlock) for assurance that their transactions will be secure.
  - When you see this symbol, what is actually being secured?

PKI: trust models

- Browser model and website authentication:
  - Phishing emails pretend to be from your bank
    - contain a link to a fake website that looks much the same as the legitimate bank site
  - A man-in-the-middle approach and a self-signed certificate (or a low-grade certificate issued by a CA without any serious credential checking) can be used to produce convincing websites:
    - If user accepts certificate without checking details or certificate pathway, the public key details are used to secure communications between the fake site and user
  - That is, the communication is secure, but you are communicating securely with the attacker – not the entity you thought you were connecting with!

PKI: Browser trust model

Phishing and spoofing



Illustrates poor Web server authentication

Summary

- Diffie-Hellman key agreement algorithm proposed as a solution to the symmetric key distribution problem
  - Based on modular exponentiation
- New thinking about mathematical functions leads to new type of cryptosystem: asymmetric ciphers
  - Two keys, one to be made public, one to be kept private (hence the name: public key cryptography)
- Can be used for encryption and also for signatures
  - Digital signatures permit nonrepudiation, integrity assurance, authentication of sender
- Asymmetric cryptography *requires PKI for implementation* (Certificates, CAs and trust relationships)