Key Management and Distribution

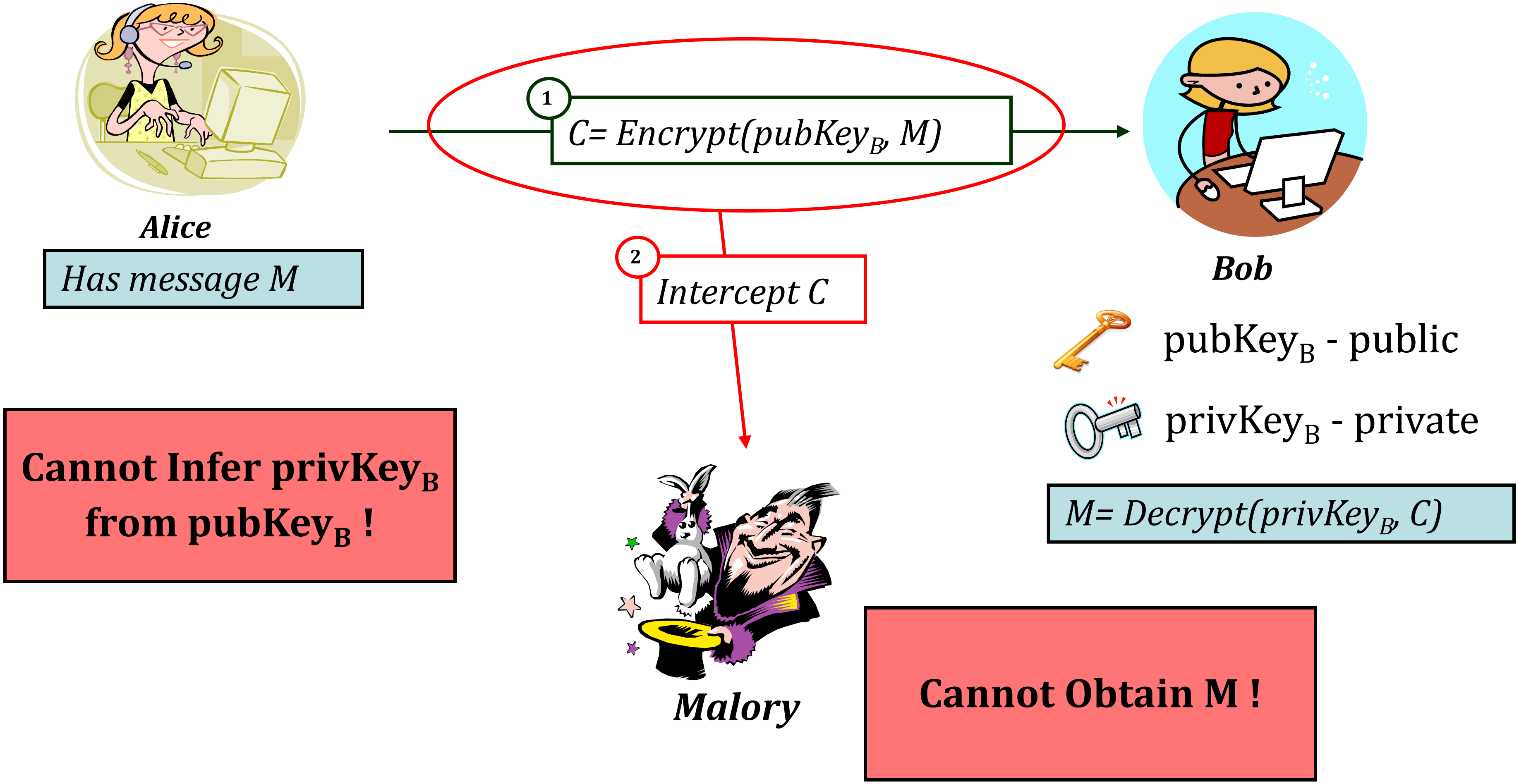
Class 5

Stallings: Ch 14

Announcement

- Homework 1 due today
- Still waiting for paper assignments

Key Distribution: The Problem



Alice

Has message M

1 *C= Encrypt(pubKey$_B$, M)*

2 *Intercept C*

Bob

pubKey$_B$ - public

privKey$_B$ - private

M= Decrypt(privKey$_B$, C)

Malory

Cannot Infer privKey$_B$ from pubKey$_B$!

Cannot Obtain M !

Key Distribution: Symmetric Key Scenario

- *Deliver a key to two parties that need to communicate securely*
    - Delivery needs to be secure: only the two parties have access to the key

Symmetric Key Scenario

- Two parties A and B
- Symmetric encryption: most efficient way to send encrypted data
- Both parties need to share a secret
- For N parties, this means N(N-1)/2 secrets !
  - Not all are needed

- How to *securely* and *efficiently* establish pairwise secrets

How To Distribute Keys

- Session key distribution with symmetric crypto
- Session key distribution with public key crypto
- Distribution of authentic public keys
- X.509 certificates

Symmetric Crypto Notations

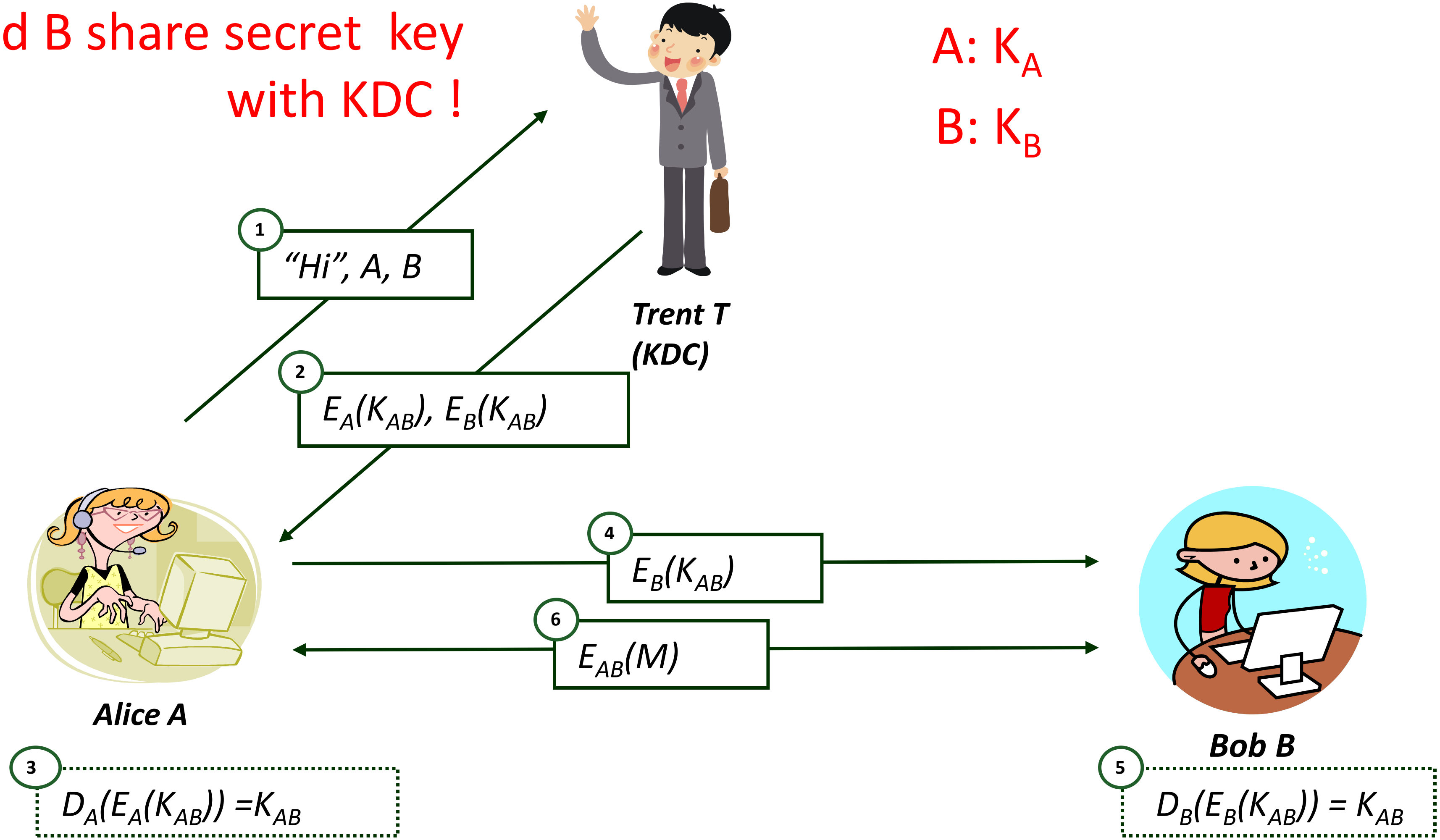
- Alice shares key K_A with KDC
    - Encryption: $E_A(M)$ = $E(K_A, M) = C$
    - Decryption: $D_A(M)$ = $D(K_A, C)$

Symmetric Key Based Distribution

A and B share secret key with KDC !

A: K_A

B: K_B



Trent T (KDC)

Alice A

Bob B

1 “Hi”, A, B

2 $E_A(K_{AB}), E_B(K_{AB})$

3 $D_A(E_A(K_{AB})) = K_{AB}$

4 $E_B(K_{AB})$

5 $D_B(E_B(K_{AB})) = K_{AB}$

6 $E_{AB}(M)$

Key Distribution Problems

- Trent (the KDC) is absolutely trusted
  - If Malory corrupts KDC, all is gone
  - Malory can read all user communication
  - *Why ?*
- Trent is a bottleneck
  - *If Trent fails, the entire system is disrupted*

How To Distribute Keys

- Session key distribution with symmetric crypto
- Session key distribution with public key crypto
- Distribution of authentic public keys
- X.509 certificates

Public Key Based Distribution

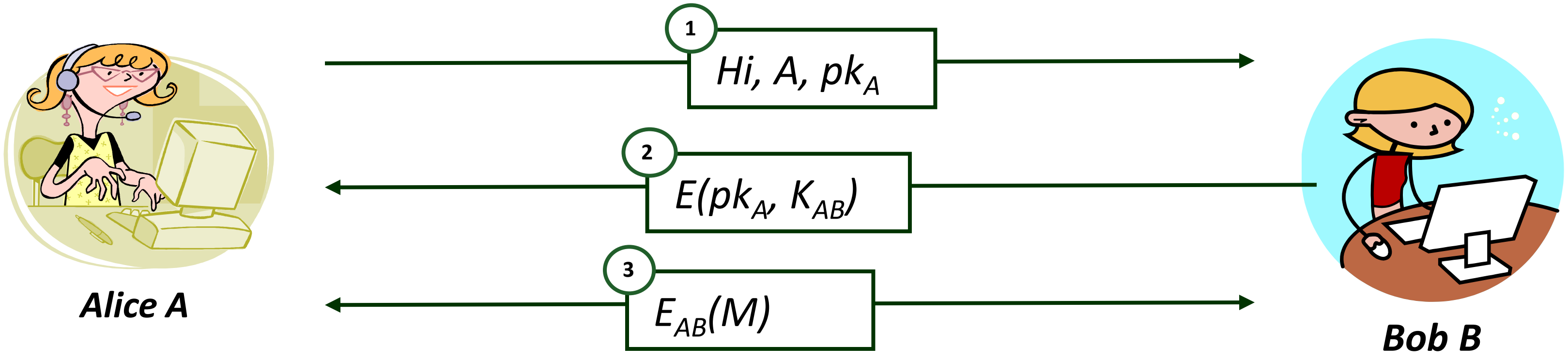
- A and B use public key crypto
  - To agree on a session key
  - Session key is used to encrypt communications
- *How do A and B know each other’s public keys?*

Public Key Crypto Notations

- Alice has key pair (pk_A, pr_A)
  - pk_A is the public key
  - pr_A is the private key
- Encryption/Decryption
  - C = E_A(Msg) = E(pk_A, Msg) – anyone can do
  - D_A(C) = D(pr_A, C) – only Alice can do this

KD with Public Key – Direct Exchange

Merkle proposed this very simple scheme



1 Hi, A, pk_A

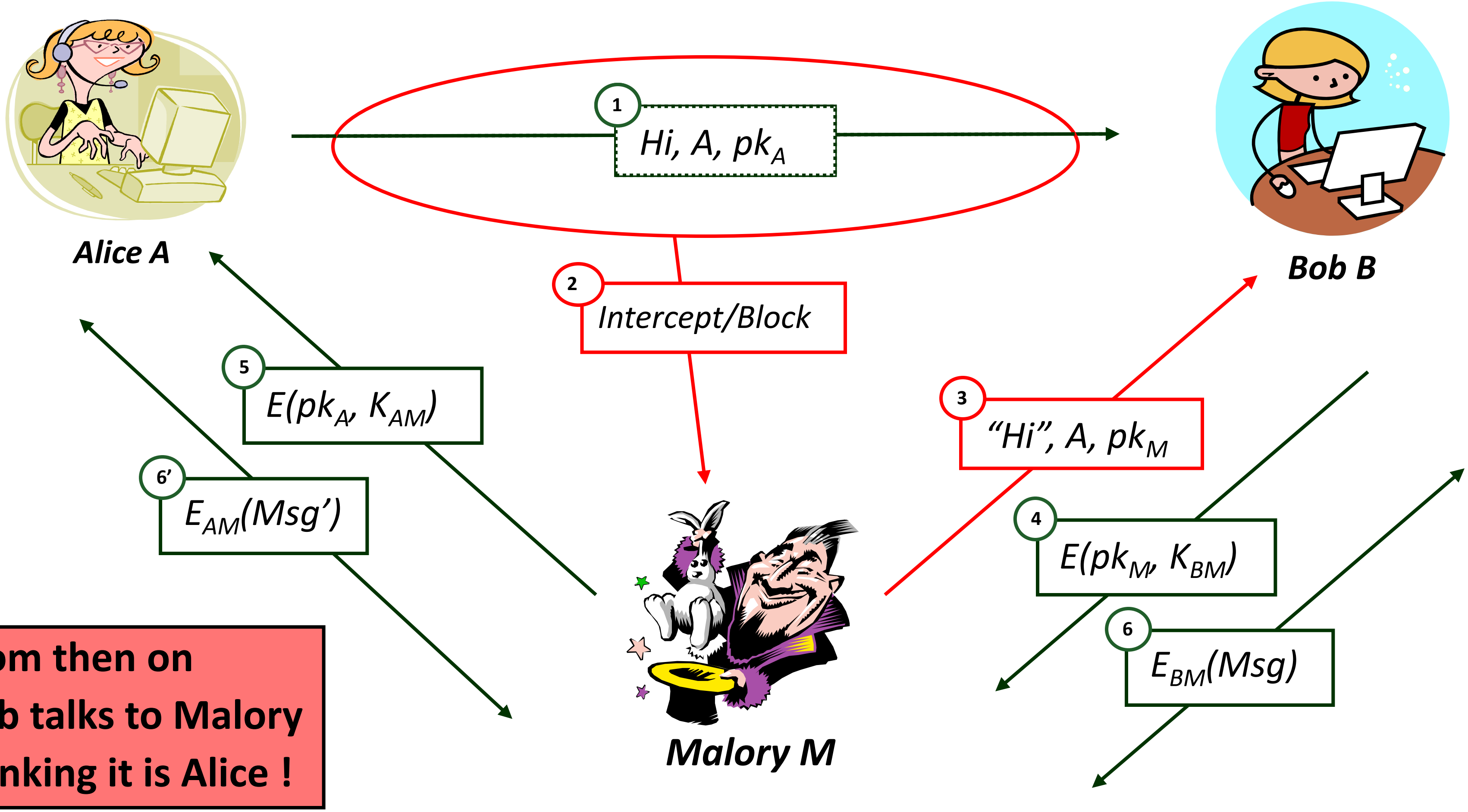
2 $E(pk_A, K_{AB})$

3 $E_{AB}(M)$

Alice A

Bob B

Man-in-the-Middle Attack



Alice A

Bob B

Malory M

1 *Hi, A, pk_A*

2 *Intercept/Block*

3 *“Hi”, A, pk_M*

4 *$E(pk_M, K_{BM})$*

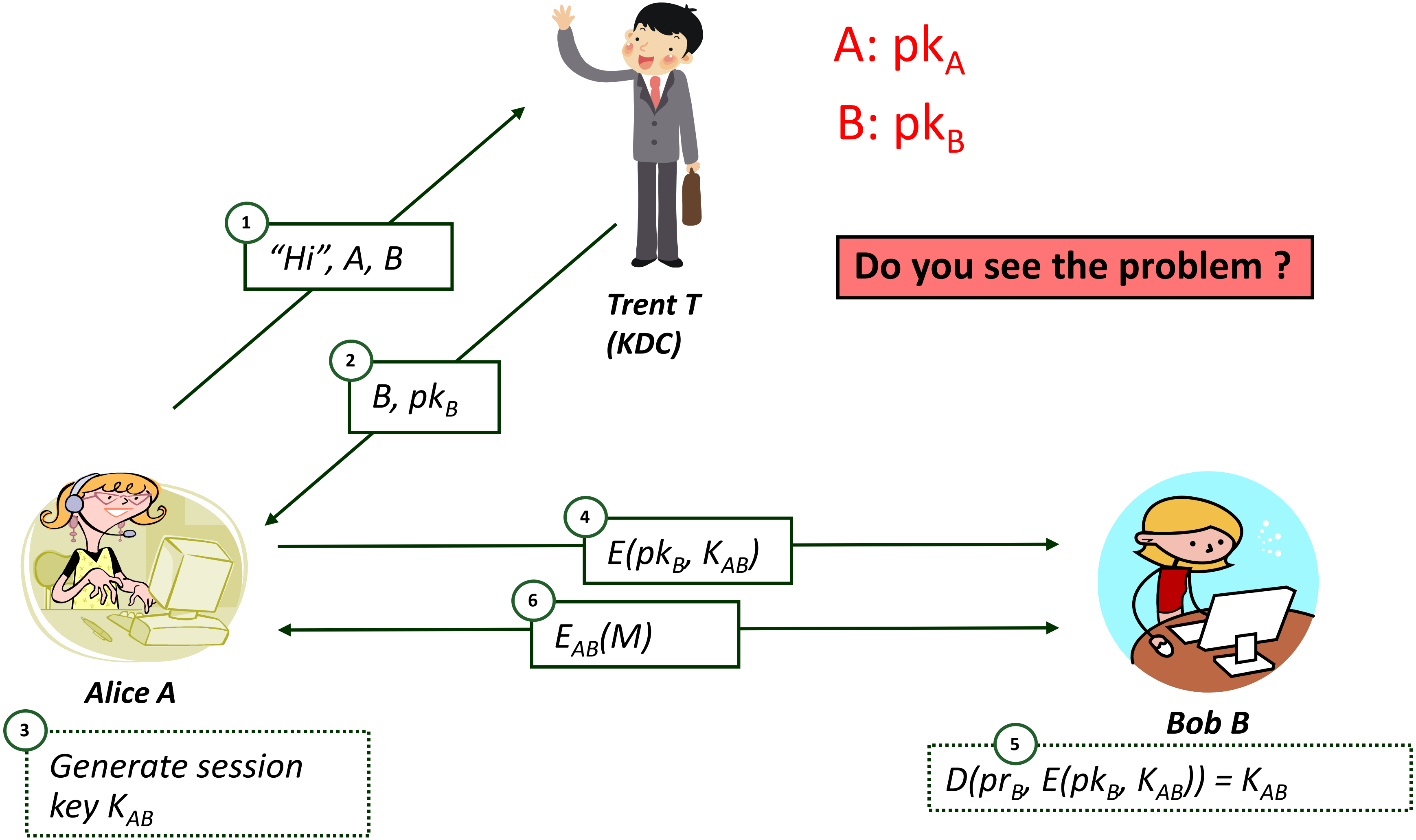
5 *$E(pk_A, K_{AM})$*

6 *$E_{BM}(Msg)$*

6’ *$E_{AM}(Msg’)$*

From then on Bob talks to Malory thinking it is Alice !

Key Distribution with Public Key and KDC



A: pk_A

B: pk_B

Do you see the problem ?

Trent T (KDC)

1 *"Hi", A, B*

2 *B, pk_B*

Alice A

3 *Generate session key K_{AB}*

4 *$E(pk_B, K_{AB})$*

Bob B

5 *$D(pr_B, E(pk_B, K_{AB})) = K_{AB}$*

6 *$E_{AB}(M)$*

How To Distribute Keys

- Session key distribution with symmetric crypto
- Session key distribution with public key crypto
- Distribution of authentic public keys
- X.509 certificates

Distribution of Public Keys !

- How are they distributed in the first place ?
  - Remember Merkle’s solution
  - … and the Man-in-the-Middle Attack
- *Need an authentic way to distribute keys !*
- Alternatives
  - Public announcement
  - Publicly available directory
  - Public-key authority
  - Public-key certificates

Public Announcement

- Similar to Merkle’s first step …
- Users distribute public keys to recipients or broadcast to community at large
  - Append keys to email messages
  - Post to news groups or email list
- *Major weakness is man-in-the-middle*
  - Anyone can create a key claiming to be someone else and broadcast it
  - Until forgery is discovered can masquerade as claimed user

Publicly Available Directory

- Register keys with a public directory
- Directory contains {name,public-key} entries
- Participants register securely with directory
  - In person or using secure authentication
- Participants can replace key at any time
- Directory can be accessed electronically
  - Needs secure, authentic communication to directory
- *Vulnerable to tampering or forgery*

Public Key Authority

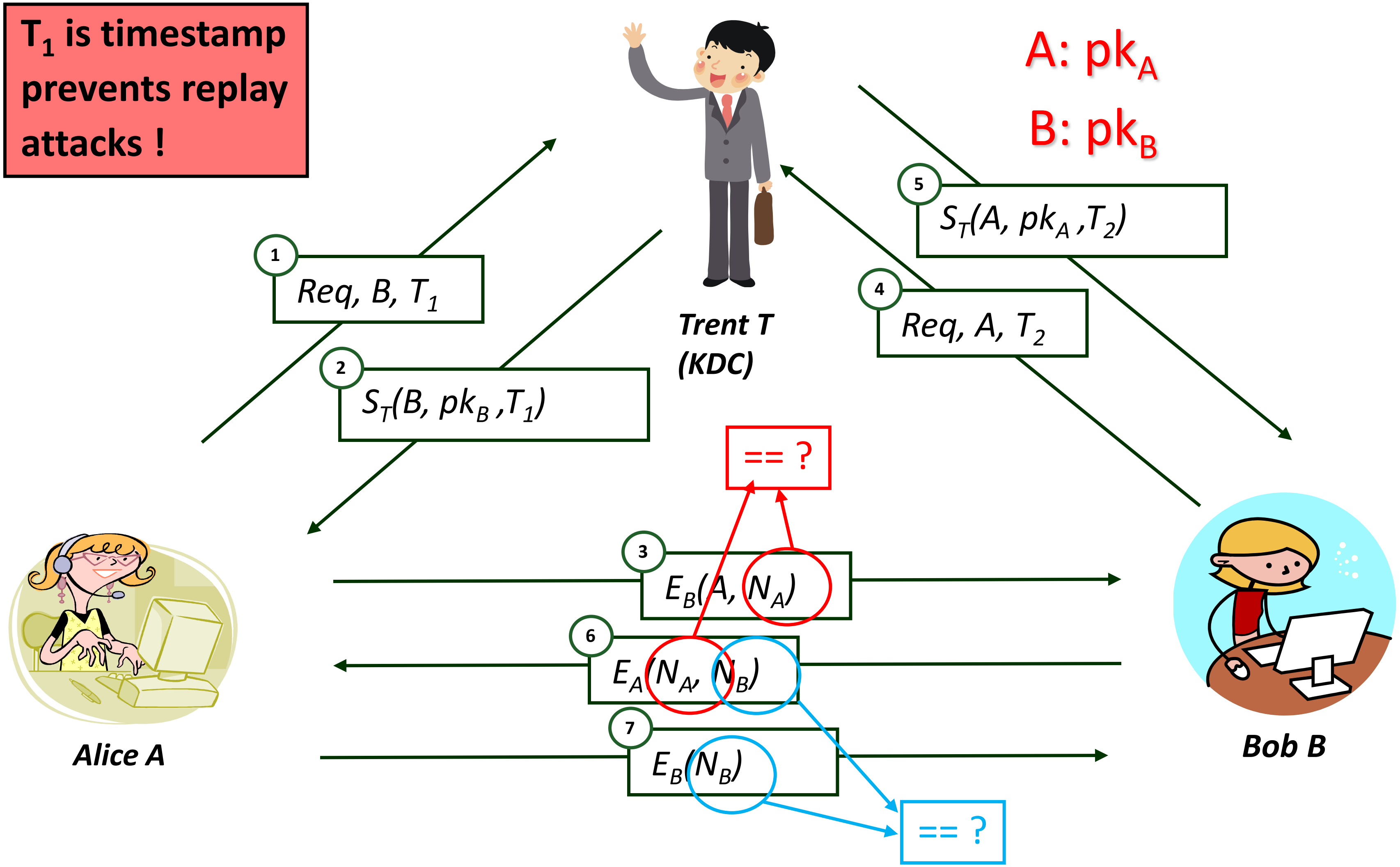
- Has properties of directory *plus*
- *Requires users to know public key of authority*
- Users interact with directory to obtain any desired public key securely
  - Requires real-time access to directory when keys are needed
- *May be vulnerable to tampering*

Public Key Crypto Notations

- Alice has key pair (pk_A, pr_A)
  - pk_A is the public key
  - pr_A is the private key
- Encryption/Decryption
  - $E_A(M) = E(pk_A,M)$ – anyone can do this
  - $D_A(M) = D(pr_A,M)$ – only Alice can do this
- Signature/Verification
  - $S_A(M)$: sign message M with private key of A
  - $V_A(M, S)$: verify that S is a signature for M
    - Uses A’s public key

Public Key Authority (Needham-Schroeder with Public Keys)

T_1 is timestamp prevents replay attacks !



A: pk_A

B: pk_B

1 Req, B, T_1

2 $S_T(B, pk_B, T_1)$

3 $E_B(A, N_A)$

4 Req, A, T_2

5 $S_T(A, pk_A, T_2)$

6 $E_A(N_A, N_B)$

7 $E_B(N_B)$

== ?

== ?

Public Key Authority Use (cont'd)

- Why do we need T's signature ?
  - A and B can be sure of the other's public key
- Why do we need steps 6 and 7 ?
  - A makes sure B knows its private key
  - Makes sure Mallory cannot impersonate B
  - … and vice-versa

How To Distribute Keys

- Session key distribution with symmetric crypto
- Session key distribution with public key crypto
- Distribution of authentic public keys
- X.509 certificates

Public Key Certificates

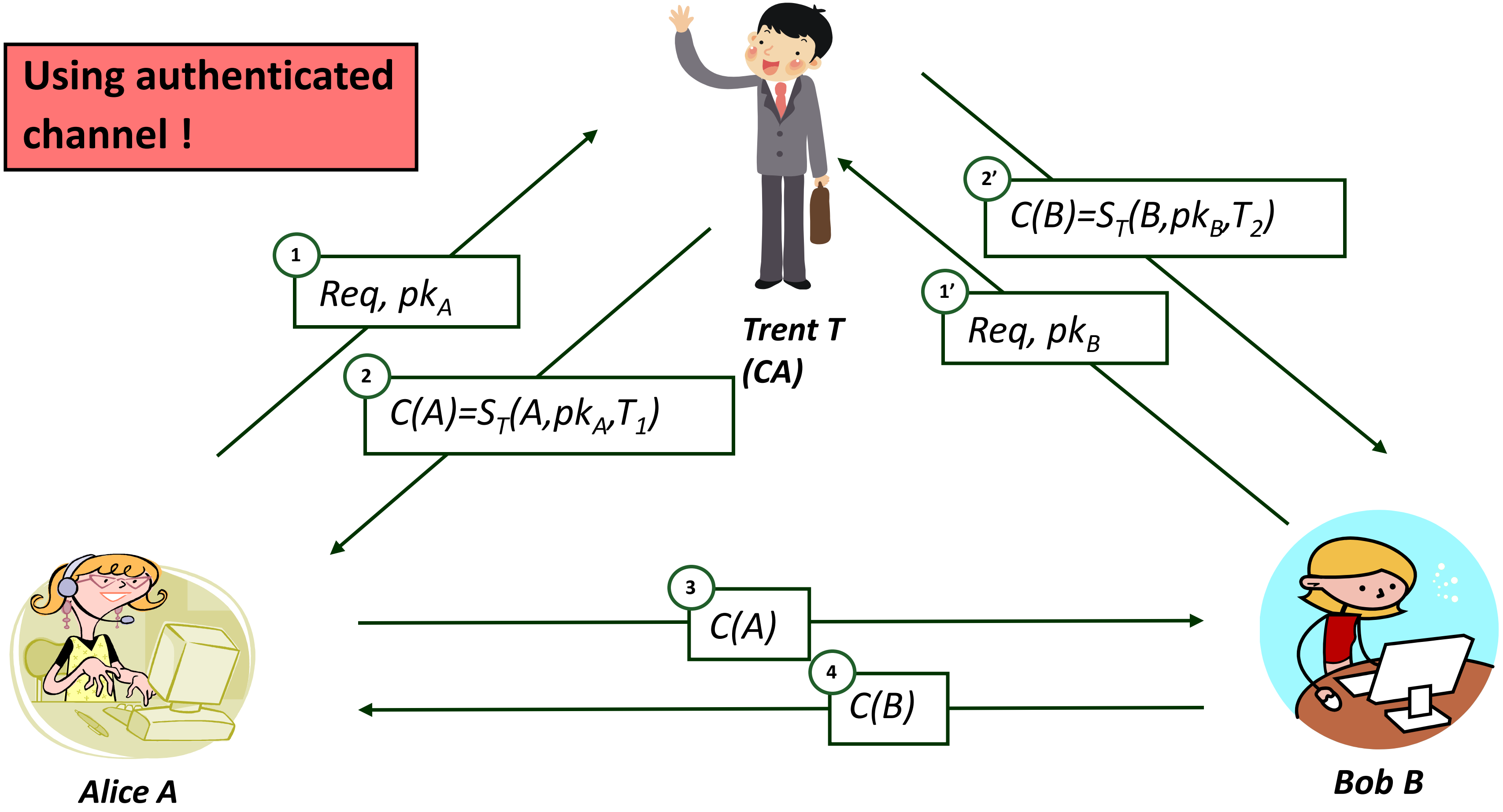
- Allow key exchange without real-time access to public-key authority
- Bind **identity** to **public key**
  - Plus other info: period of validity, rights of use etc
- All contents **signed** by a trusted Public-Key or Certificate Authority (CA)
  - Can be verified by anyone who knows the public-key authority's public-key

Certificate Requirements

- Anyone can read the name and public key from a certificate
- Only the CA can *create* and *update* certificates
- Anyone can *verify the validity* of the certificate

How are Certificates Used ?

Using authenticated channel !



1 Req, pk_A

2 $C(A)=S_T(A,pk_A,T_1)$

Trent T (CA)

1’ Req, pk_B

2’ $C(B)=S_T(B,pk_B,T_2)$

3 $C(A)$

4 $C(B)$

Alice A

Bob B

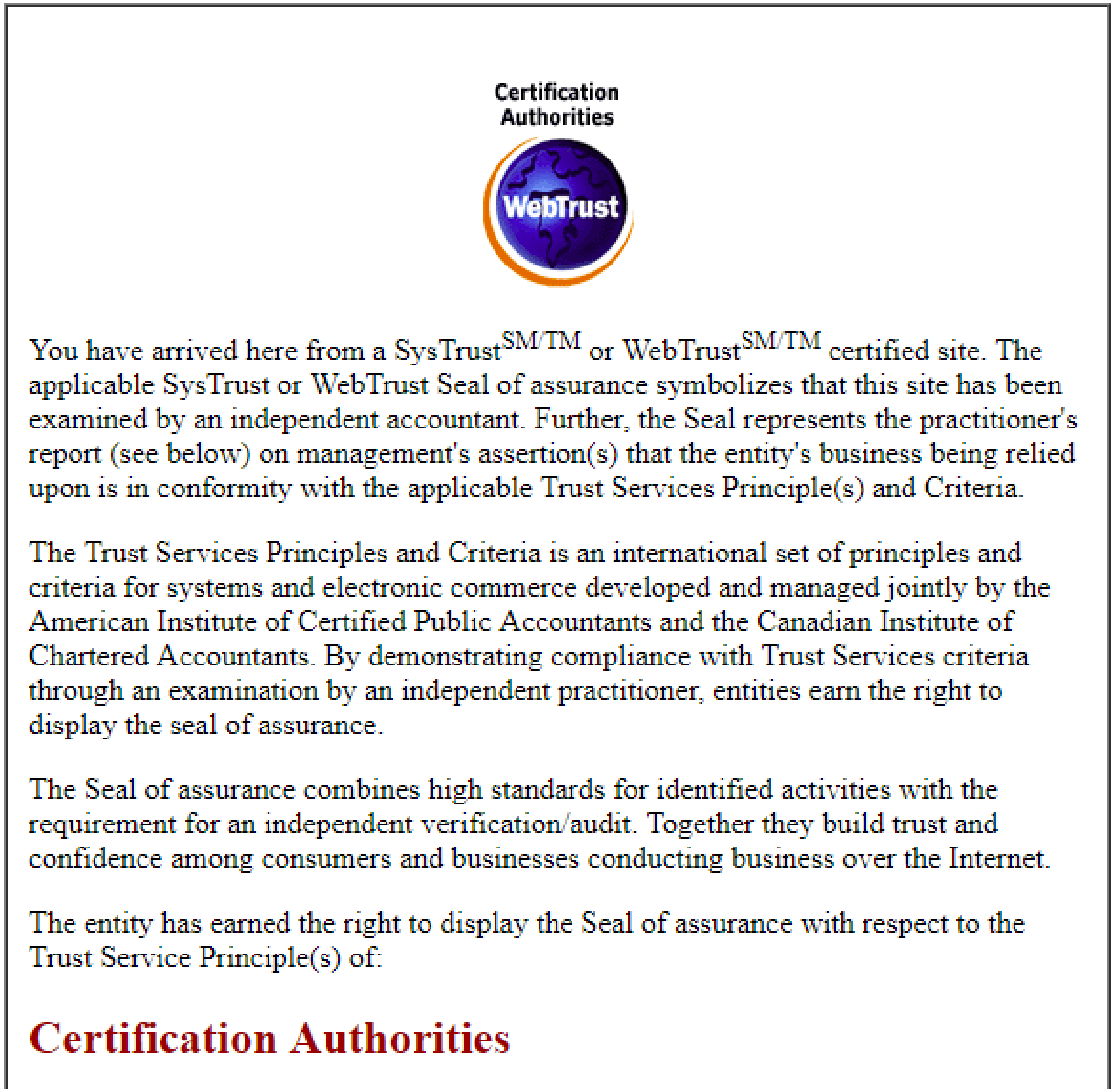
How are Certificates Used ? (cont'd)

- Certificates *issued*
  - Over authenticated channels
  - In person
- Certificates are re-issued infrequently
  - Steps 1 and 2 are done once
- Certificates contain timestamp and validity period
  - User can verify certificate validity
- Example CAs: Symantec (VeriSign), Comodo, GoDaddy

Symantec (former VeriSign)

- For websites
- Examines
  - Traditional documents like articles of incorporation and business licenses
  - Digital verification of each site operated by the organization

WebTrust https://cert.webtrust.org



You have arrived here from a SysTrust$^{SM/TM}$ or WebTrust$^{SM/TM}$ certified site. The applicable SysTrust or WebTrust Seal of assurance symbolizes that this site has been examined by an independent accountant. Further, the Seal represents the practitioner's report (see below) on management's assertion(s) that the entity's business being relied upon is in conformity with the applicable Trust Services Principle(s) and Criteria.

The Trust Services Principles and Criteria is an international set of principles and criteria for systems and electronic commerce developed and managed jointly by the American Institute of Certified Public Accountants and the Canadian Institute of Chartered Accountants. By demonstrating compliance with Trust Services criteria through an examination by an independent practitioner, entities earn the right to display the seal of assurance.

The Seal of assurance combines high standards for identified activities with the requirement for an independent verification/audit. Together they build trust and confidence among consumers and businesses conducting business over the Internet.

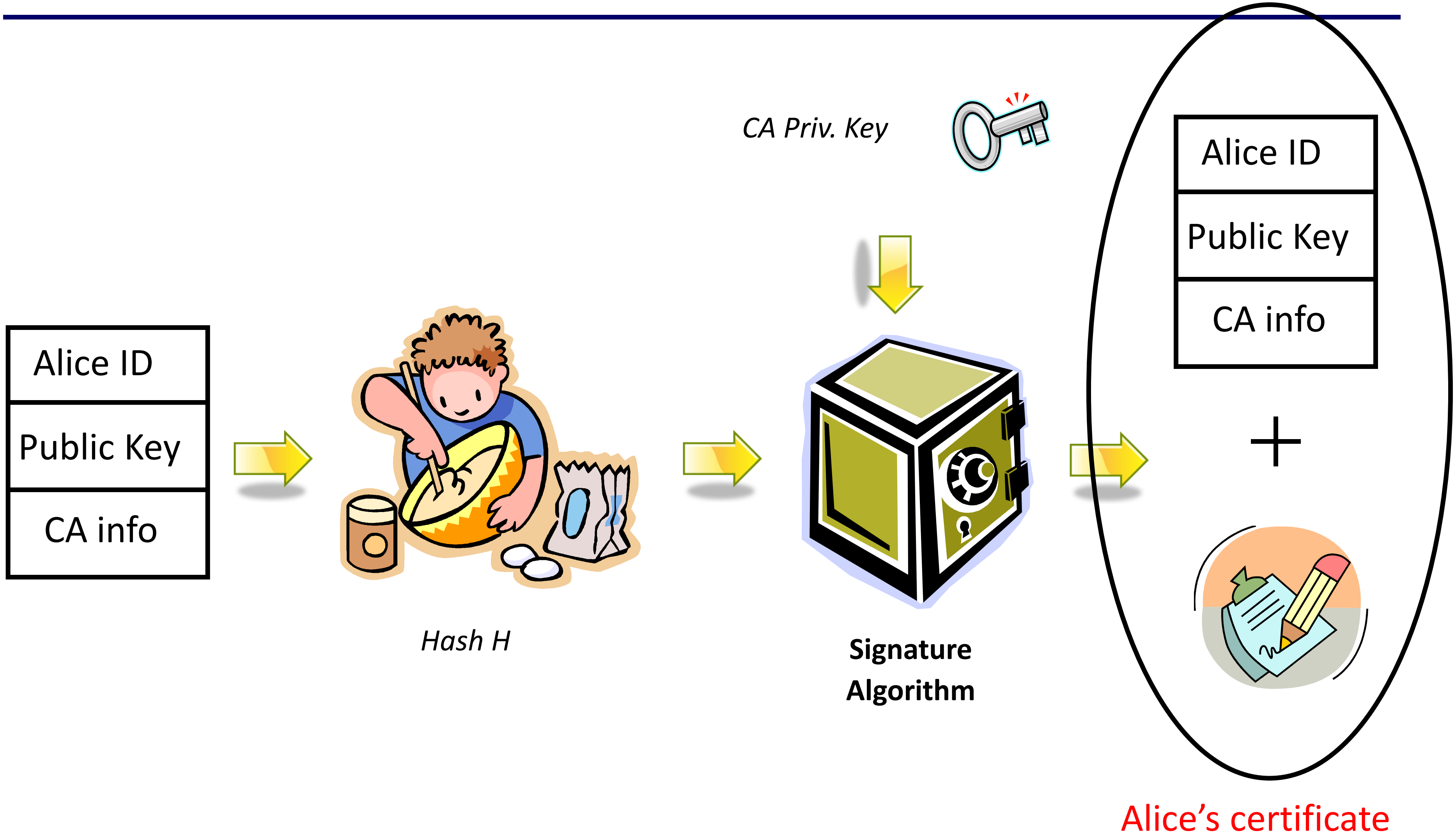
The entity has earned the right to display the Seal of assurance with respect to the Trust Service Principle(s) of:

Certification Authorities

X.509 Certificates

- Part of CCITT X.500 directory service standards
  - Distributed servers maintaining user info database
- *Defines framework for authentication services*
  - Directory may store public-key certificates
  - Public key of user signed by certification authority
- Defines authentication protocols
- Uses public-key crypto & digital signatures
  - Algorithms not standardised, but RSA recommended
- X.509 certificates are widely used
  - have 3 versions

X.509 Certificate Generation



Alice ID

Public Key

CA info

Hash H

CA Priv. Key

Signature Algorithm

Alice ID

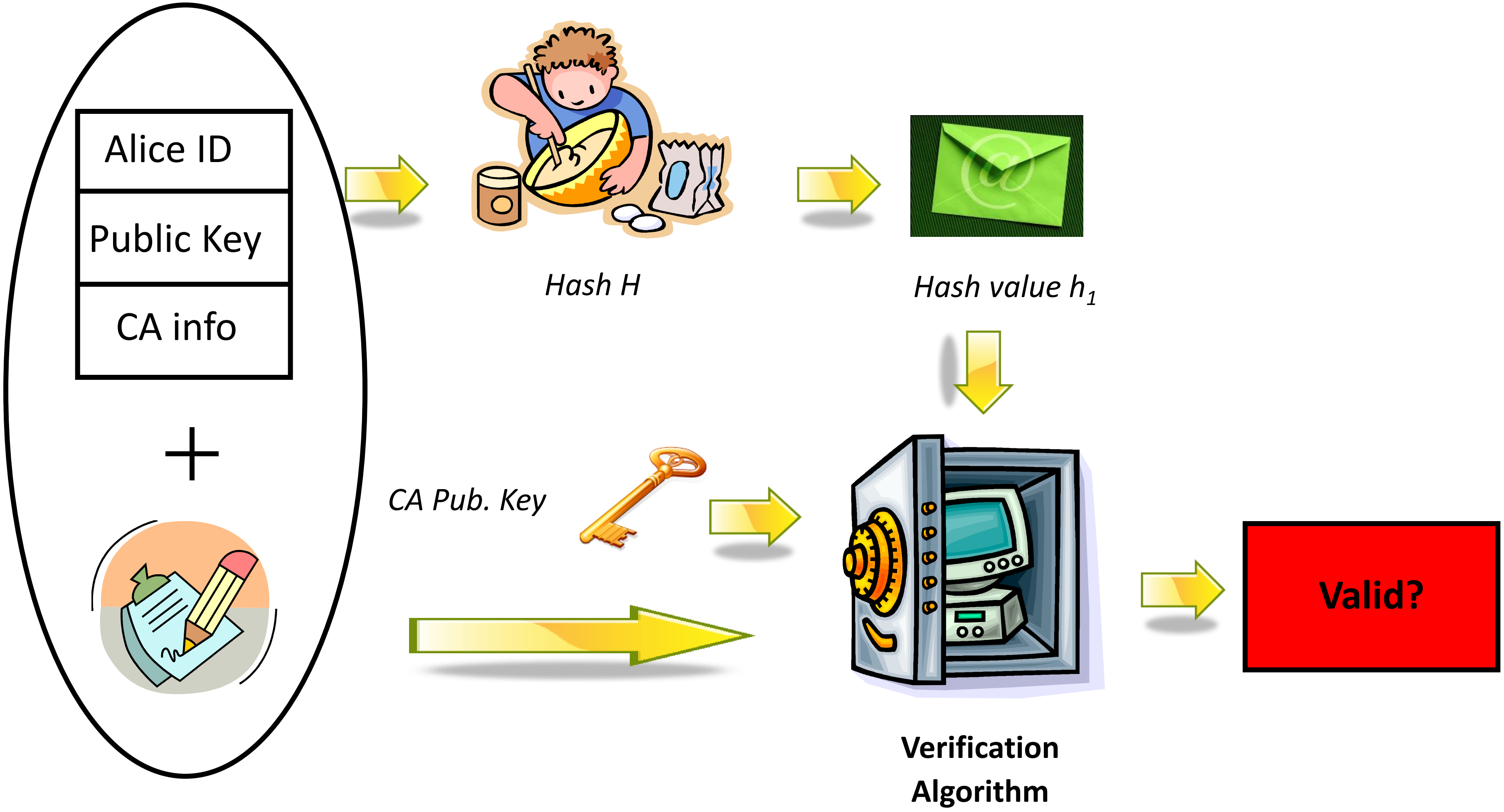
Public Key

CA info

+

Alice's certificate

X.509 Certificate Verification



Alice ID

Public Key

CA info

+

Hash H

Hash value h_1

CA Pub. Key

Valid?

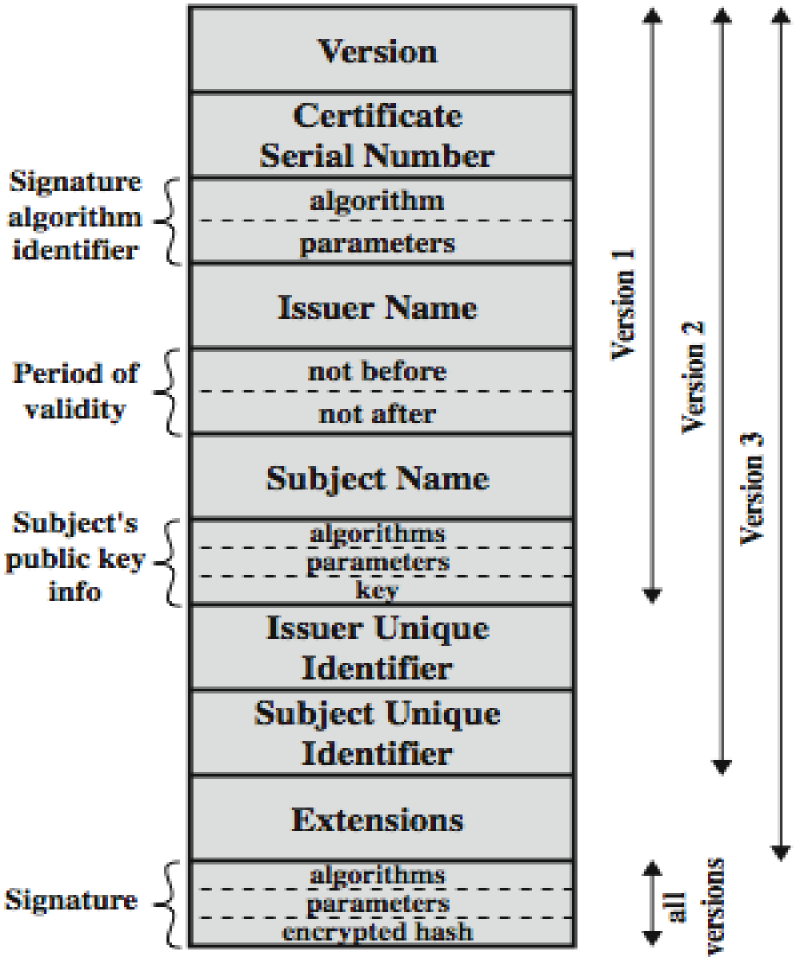
Verification Algorithm

Alice’s certificate

X.509 Certificate Format

- Issued by a Certification Authority (CA), containing:
  - version (1, 2, or 3)
  - serial number (unique within CA) identifying certificate
  - signature algorithm identifier
  - issuer X.500 name
  - **period of validity (from - to dates)**
  - subject X.500 name (name of owner)
  - subject public-key info (algorithm, parameters, key)
  - issuer unique identifier (v2+)
  - subject unique identifier (v2+)
  - extension fields (v3)
  - **signature (of hash of all fields in certificate)**
- Notation: C_{CA}(A) is certificate for A signed by CA

X.509 Certificate Format (cont'd)



(a) X.509 Certificate

From Stallings book

X.509 Version 3

- Additional information may be needed in a certificate
  - E-mail/URL, policy details, usage constraints
- Hard to explicitly name new fields
  - Backward compatibility issues
- *Define a general extension method*
  - Extension identifier
  - Criticality indicator
  - Extension value

Obtaining a Certificate

- Any user with access to CA can get any certificate from it
- Only the CA can modify a certificate
- Certificates can be placed in a public directory
- *Certificates cannot be forged*

Certificate Revocation

- Certificates have a period of validity
- *May need to be revoked before expiry:*

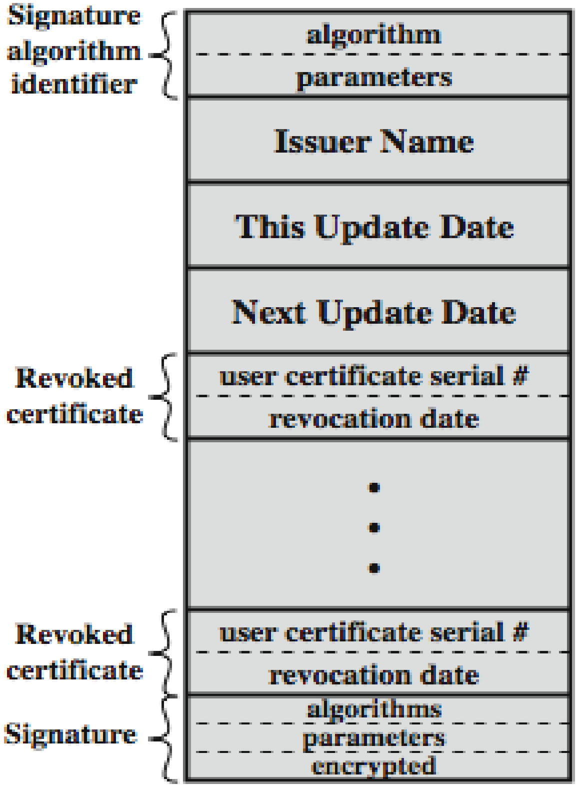
1. User's private key is compromised
2. User is no longer certified by this CA
3. CA's certificate is compromised
4. User behaves badly

- CA maintains list of revoked certificates
  - Certificate Revocation List (CRL)
- Users should check certificates with CA’s CRL

Certificate Revocation (cont'd)

- When Alice obtains Bob's certificate
  - Contact CA
  - Check that certificate is not revoked (in CRL) !

- CA needs to maintain a certificate in CRL until certificate expires



(b) Certificate Revocation List

From Stallings book

Certificate Revocation Problem

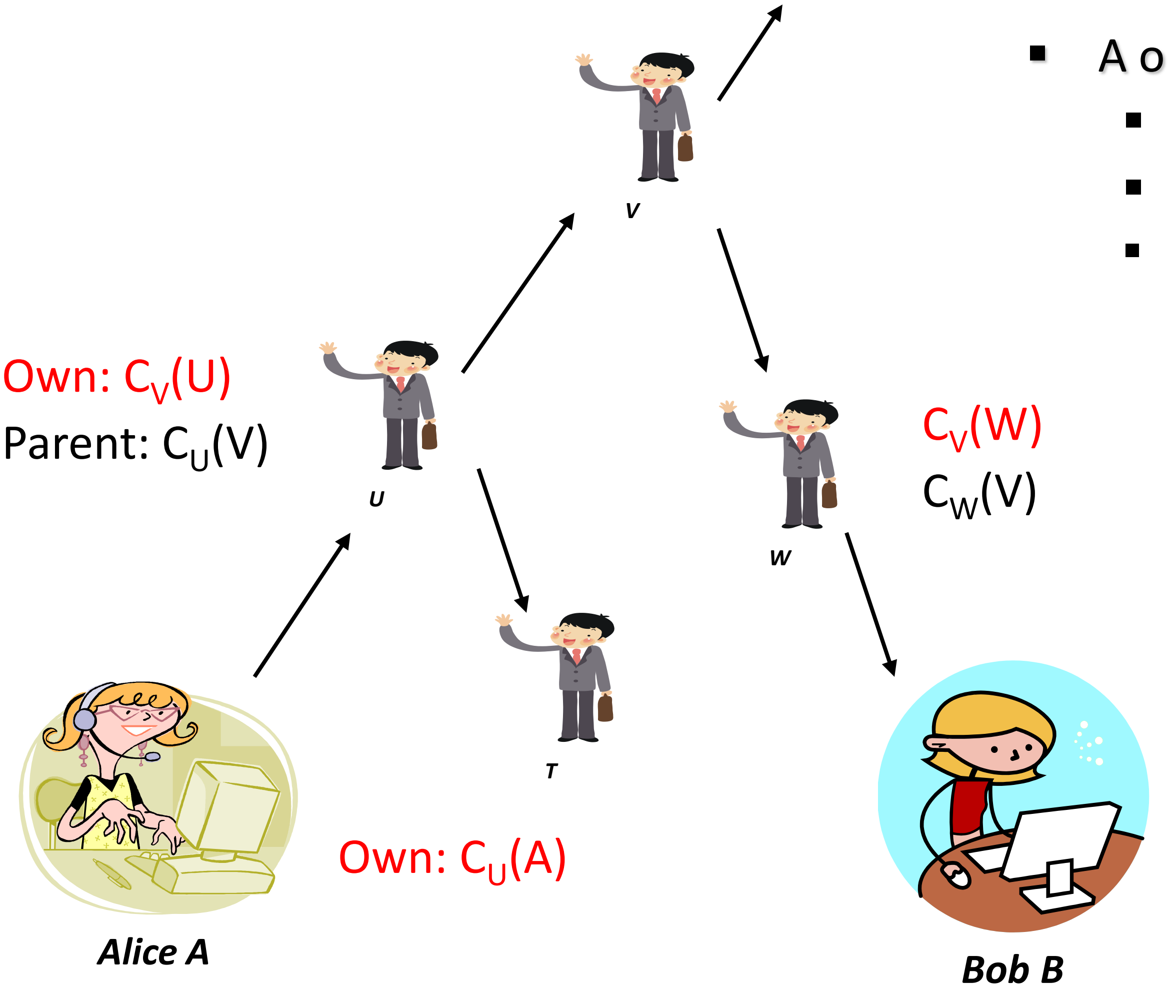
- Alice does something bad at time T_1
- CA finds out and revokes A’s certificate at T_2
  - Place A certificate into CRL
- Alice can continue to misbehave and be authenticated between T_1 and T_2 !

CA Hierarchy

- What if A and B do not share a CA ?
- Solution: CA's must form a hierarchy
- Use certificates linking members of hierarchy to validate other CA's
  - Each CA has certificates for itself (from parent CA) and for parent (backward)
- Each client trusts parents certificates

CA Hierarchy Example

- A obtains B's certificate
  - $C_U(V)$
  - $C_V(W)$
  - $C_W(B)$



Own: $C_V(U)$

Parent: $C_U(V)$

$C_V(W)$

$C_W(V)$

Own: $C_U(A)$

Own: $C_W(B)$

Certificate Types

- **Root certificate (Trust anchor)**
  - Self-signed certificate used to sign other certificates
  - E.g., Google Trust Services (https://pki.goog/)
- **Intermediate certificate**
  - Used to sign other certificates
  - Must be signed by intermediate or root certificate
- **End-entity or leaf certificate**
  - Cannot be used to sign other certificates
  - TLS/SSL server and client certificates
  - Email certificates