CSL759: Cryptography and Computer Security

Ragesh Jaiswal

CSE, IIT Delhi

Key Distribution

Diffie Hellman Key Exchange

g^x →

← g^y

Both parties share g^{xy} which is the secret key for the session.

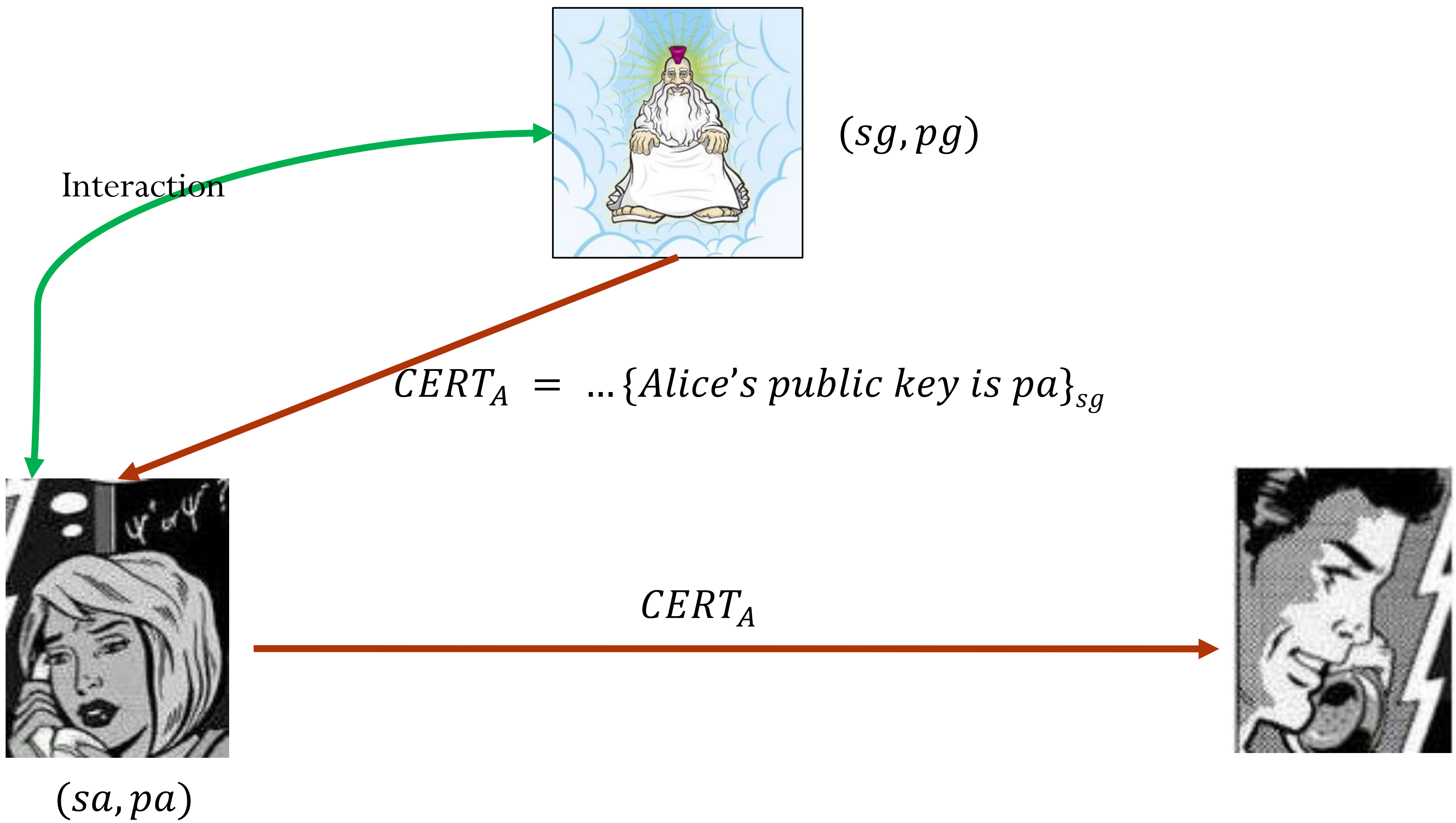
Authentication

Diffie Hellman Key Exchange

The adversary will be able to read all messages being exchanged between Alice and Bob

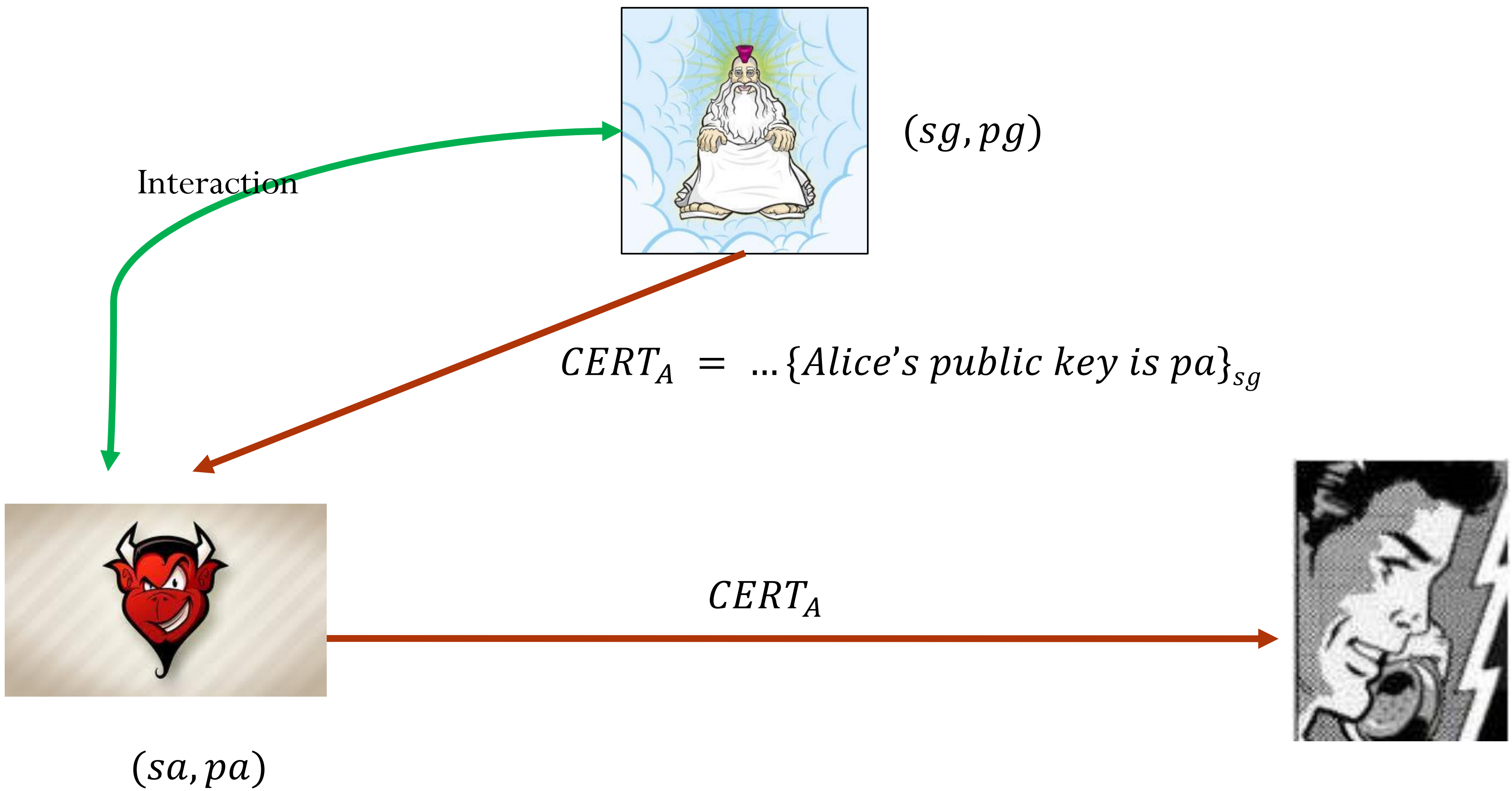
Key Distribution in Public Key Setting

- Public key cryptography:



Key Distribution in Public Key Setting

- Certificate Process



Key Distribution in Public Key Setting

Certificate Process

- Alice generates *pk* and sends it to CA
- CA does identity check
- Alice proves knowledge of secret key to CA
- CA issues certificate to Alice
- Alice sends certificate to Bob
- Bob verifies certificate and extracts Alice's *pk*

Key Distribution in Public Key Setting

Generate key and send to CA

Key generation: Alice generates her keys locally via $(pk, sk) \xleftarrow{\$} \mathcal{K}$

Send to CA: Alice sends $(Alice, pk)$ to a certificate authority (CA).

Key Distribution in Public Key Setting

Identity check

Upon receiving $(Alice, pk)$ the CA performs some checks to ensure pk is really Alice's key:

- Call Alice by phone
- Check documents

These checks are out-of-band.

Key Distribution in Public Key Setting

Proof of knowledge

The CA might have Alice sign or decrypt something under pk to ensure that Alice knows the corresponding secret key sk.

This ensures Alice has not copied someone else's key.

Key Distribution in Public Key Setting

Certificate Issuance

Once CA is convinced that *pk* belongs to Alice it forms a certificate

$$CERT_A = (CERTDATA, \sigma),$$

where σ is the CA's signature on $CERTDATA$, computed under the CA's secret key $sk[CA]$.

$CERTDATA$:

- pk, ID (Alice)
- Name of CA
- Expiry date of certificate
- Restrictions
- Security level
- ...

The certificate $CERT_A$ is returned to Alice.

Key Distribution in Public Key Setting

Certificate usage

Alice can send $CERT_A$ to Bob who will:

- $(CERTDATA, \sigma) \leftarrow CERT_A$
- Check $\mathcal{V}_{pk[CA]}(CERTDATA, \sigma) = 1$ where $pk[CA]$ is CA's public key
- $(pk, Alice, expiry, \ldots) \leftarrow CERTDATA$
- Check certificate has not expired
- ...

If all is well we are ready for usage.

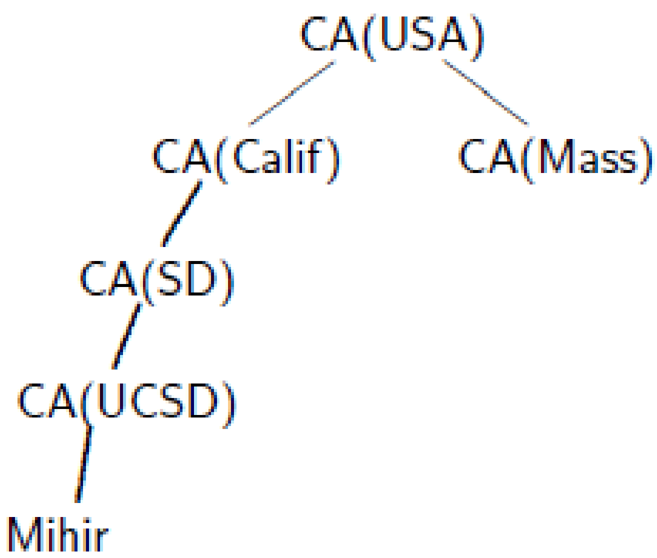
Key Distribution in Public Key Setting

How does Bob get $pk[CA]$?

CA public keys are embedded in software such as your browser.

Key Distribution in Public Key Setting

Certificate hierarchies



<u>$CERT_{Mihir}$</u>

$CERT[CA(USA) : CA(Calif)]$
$CERT[CA(Calif) : CA(SD)]$
$CERT[CA(SD) : CA(UCSD)]$
$CERT[CA(UCSD) : Mihir]$

$$CERT[X : Y] = (pk[Y], Y, \ldots, \mathcal{S}_{sk[X]}(pk[Y], Y, \ldots))$$

To verify $CERT_{Mihir}$ you need only $pk_{CA[USA]}$.

Key Distribution in Public Key Setting

Why certificate hierarchies?

- It is easier for CA(UCSD) to check Mihir's identity (and issue a certificate) than for CA(USA) since Mihir is on UCSD's payroll and UCSD already has a lot of information about him.
- Spreads the identity-check and certification job to reduce work for individual CAs
- Browsers need to have fewer embedded public keys. (Only root CA public keys needed.)

Key Distribution in Public Key Setting

Revocation

Suppose Alice wishes to revoke her certificate $CERT_A$, perhaps because her secret key was compromised.

- Alice sends $CERT_A$ and revocation request to CA
- CA checks that request comes from Alice
- CA marks $CERT_A$ as revoked

Key Distribution in Public Key Setting

Certificate revocation lists (CRLs)

CA maintains a CRL with entries of form

$$(CERT, \text{Revocation date})$$

This list is disseminated.

Before Bob trusts Alice's certificate he should ensure it is not on the CRL.

Key Distribution in Public Key Setting

Revocation Issues

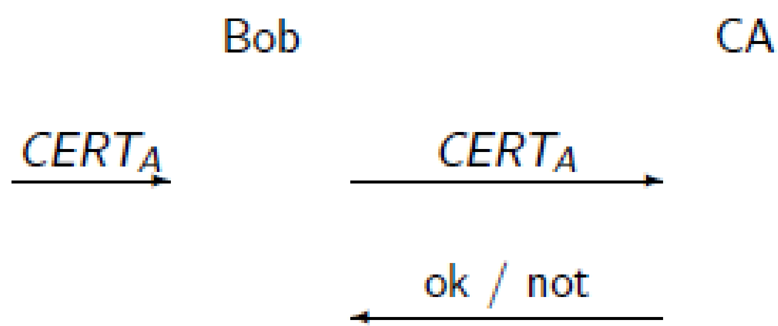
- November 22: Alice's secret key compromised
- November 24: Alice's $CERT_A$ revoked
- November 25: Bob sees CRL

In the period Nov. 22-25, $CERT_A$ might be used and Bob might be accepting as authentic signatures that are really the adversary's. Also Bob might be encrypting data for Alice which the adversary can decrypt.

Key Distribution in Public Key Setting

OCSP

The On-line Certificate Status Protocol (OCSP) enables on-line checks of whether or not a certificate has been revoked.



But on-line verification kind of defeats the purpose of public-key cryptography!

Key Distribution in Public Key Setting

Revocation in practice

- VeriSign estimates that 20% of certificates are revoked
- In practice, CRLs are huge

Revocation is a big problem and one of the things that is holding up widespread deployment of a PKI and use of public-key cryptography.

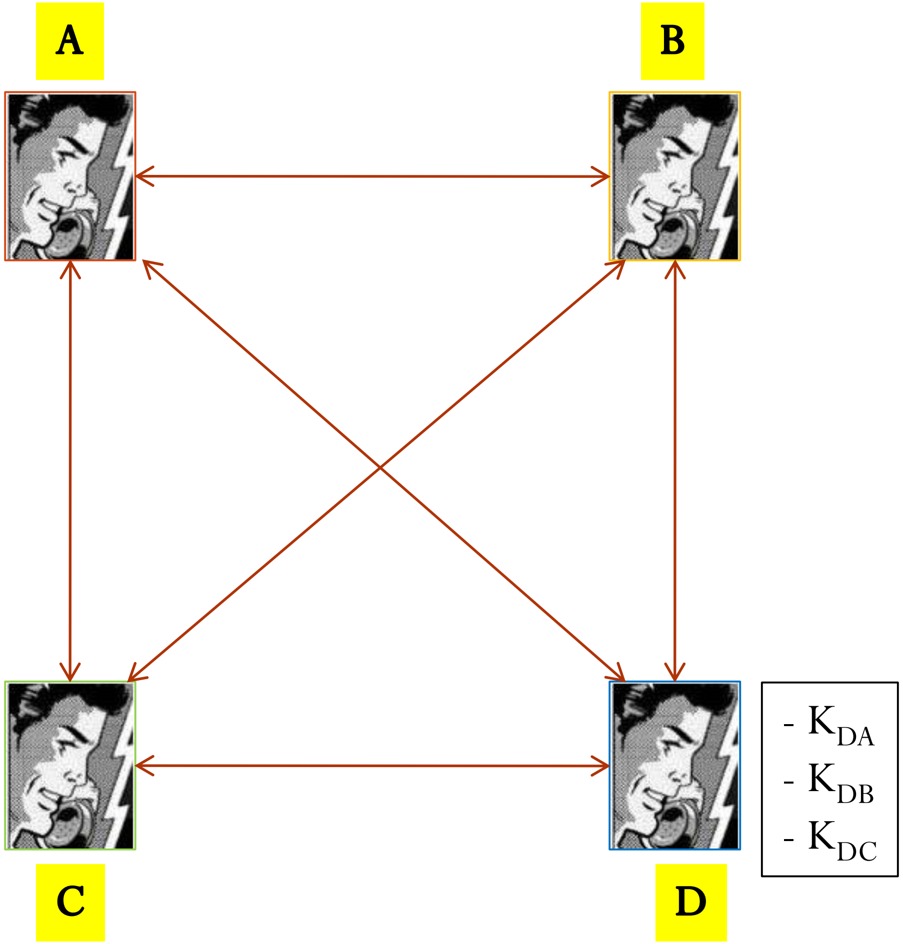
Key Distribution in Public Key Setting

PGP

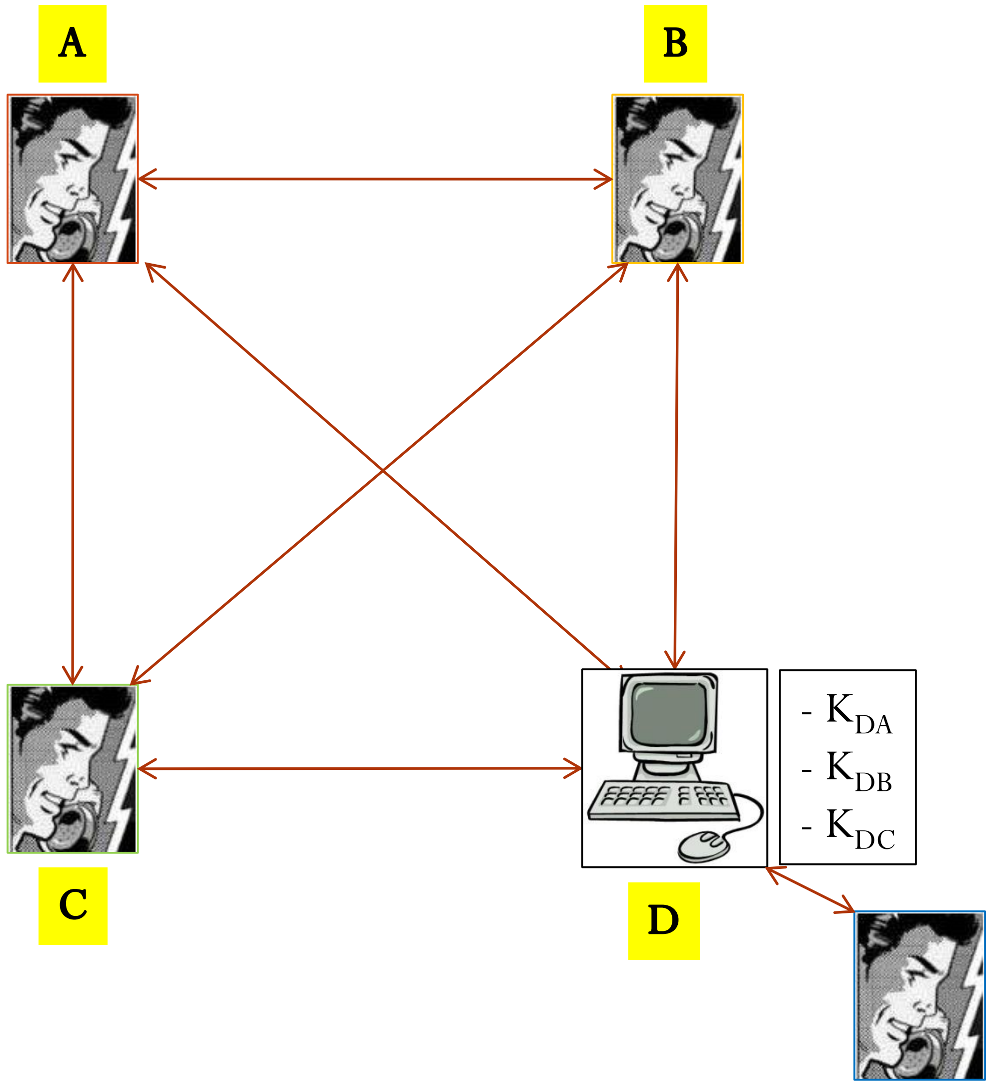
In PGP, there are no CAs. You get Alice's public key from Carol and decide to what extent you want to trust it based on your feelings about Carol. Requires user involvement.

Key Distribution in the symmetric setting

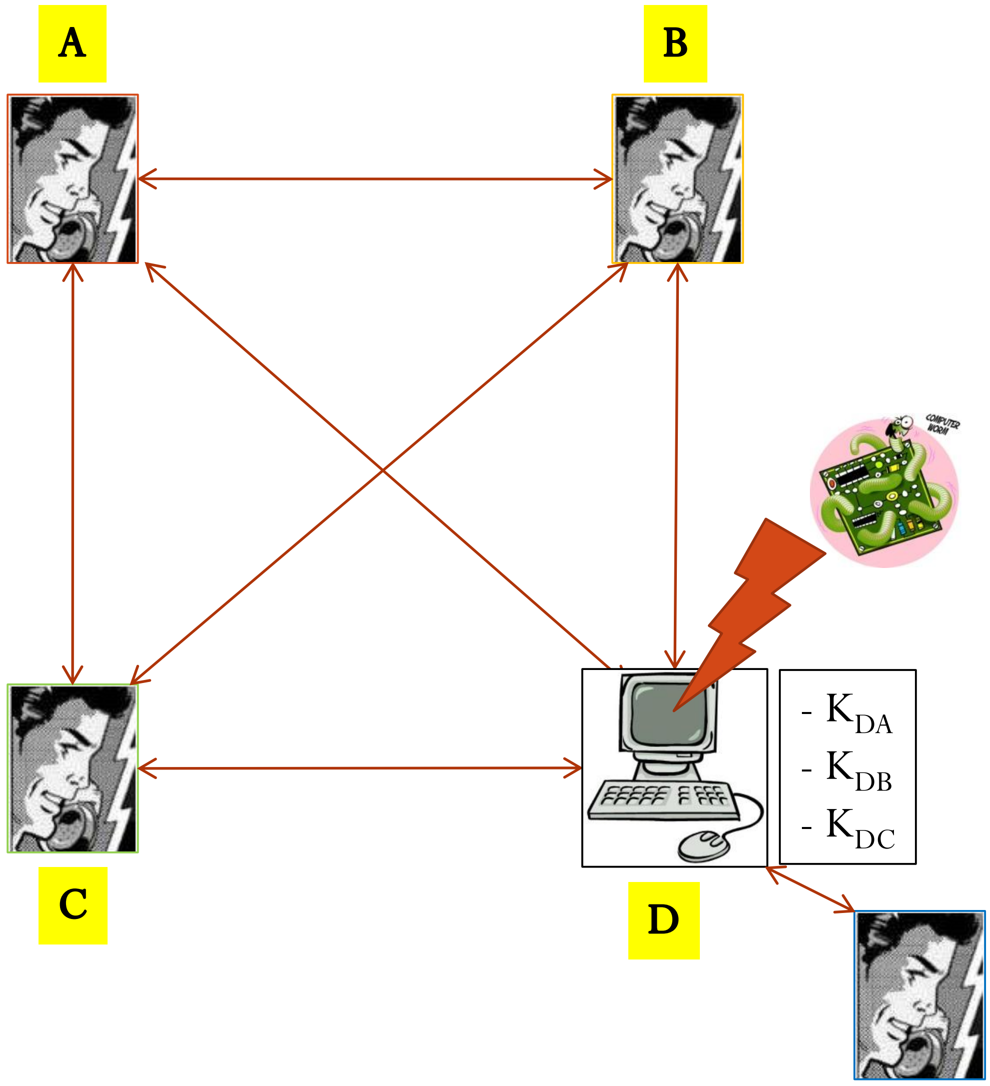
Key Distribution: Symmetric Setting



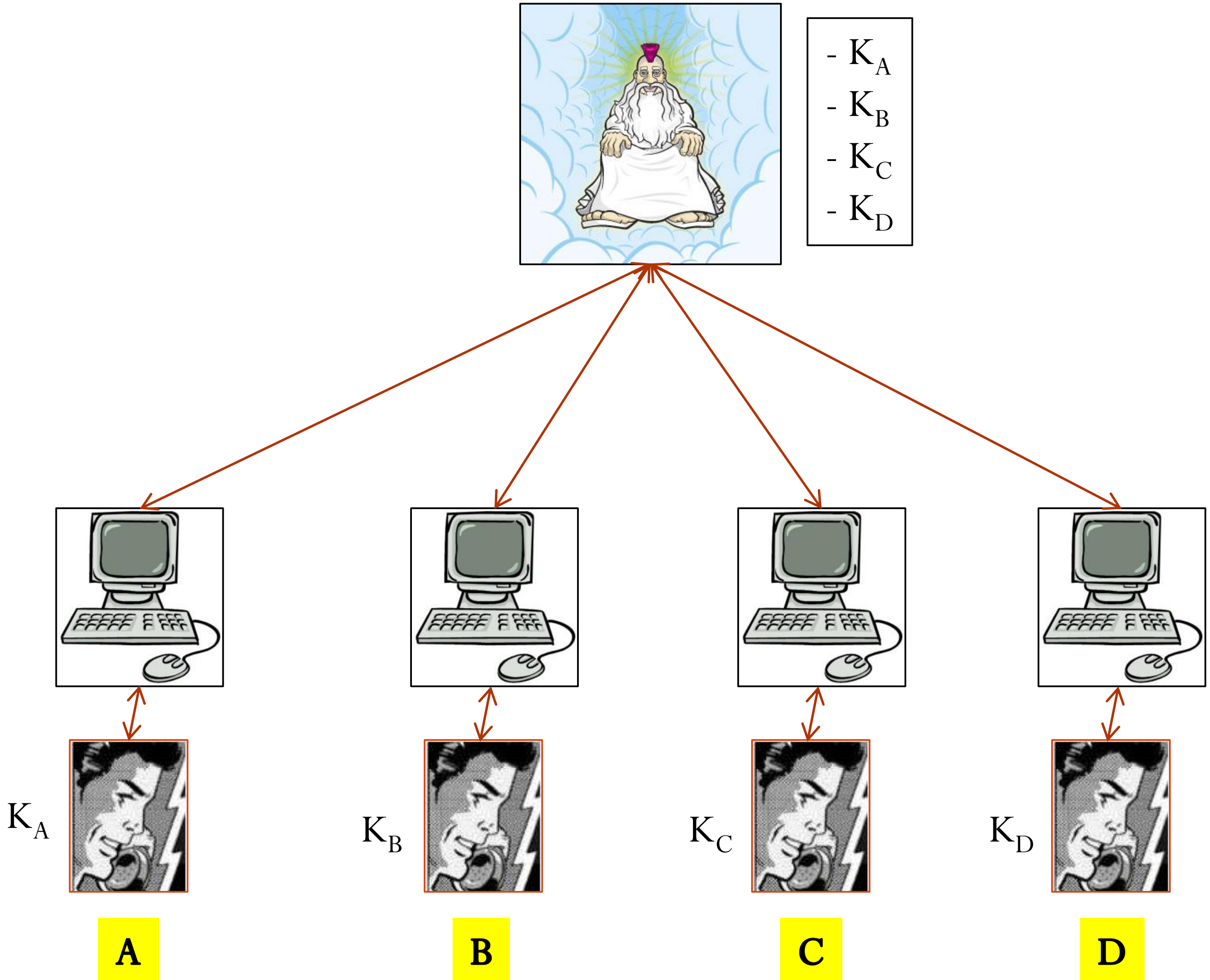
Key Distribution: Symmetric Setting



Key Distribution: Symmetric Setting



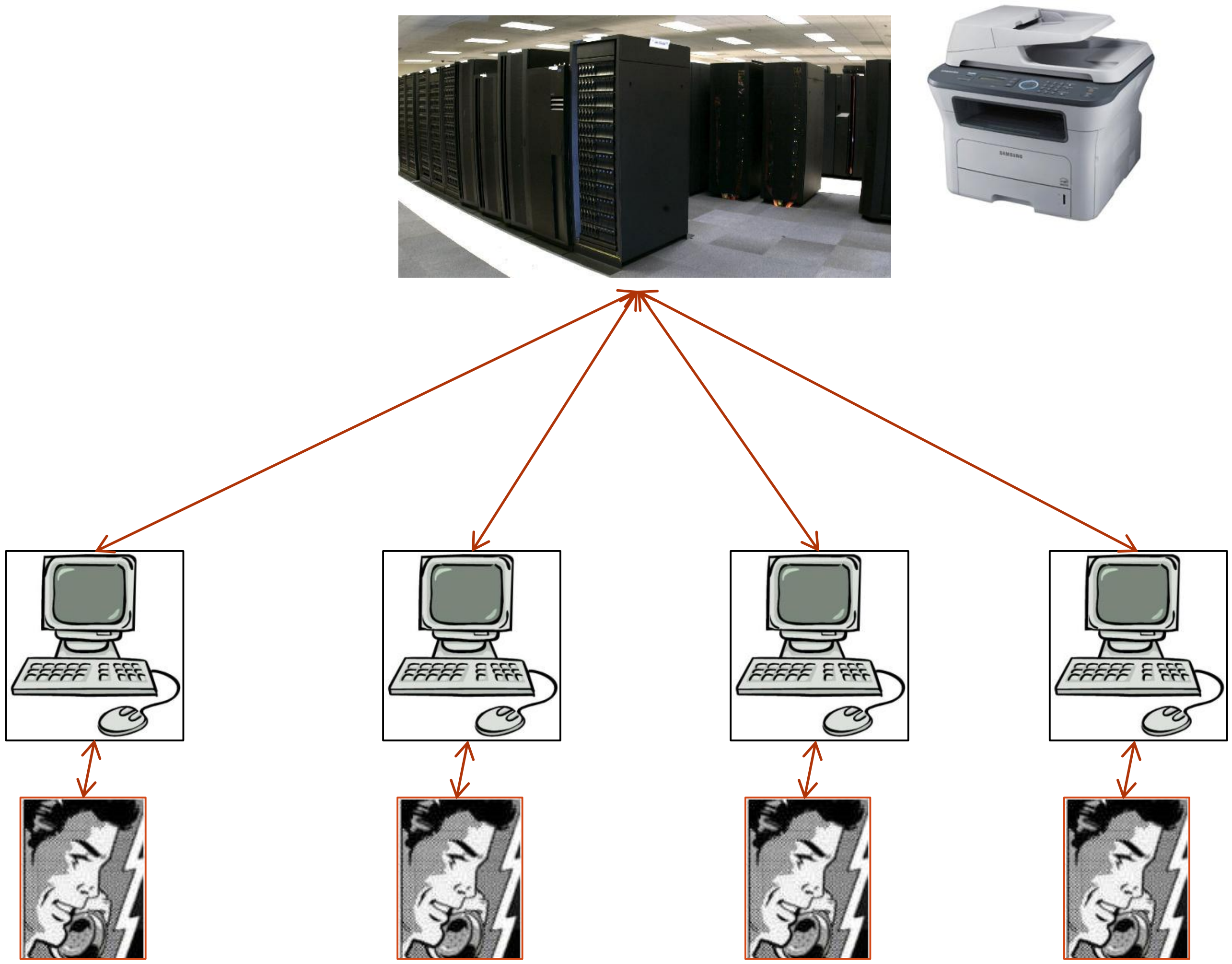
Key Distribution: Symmetric Setting



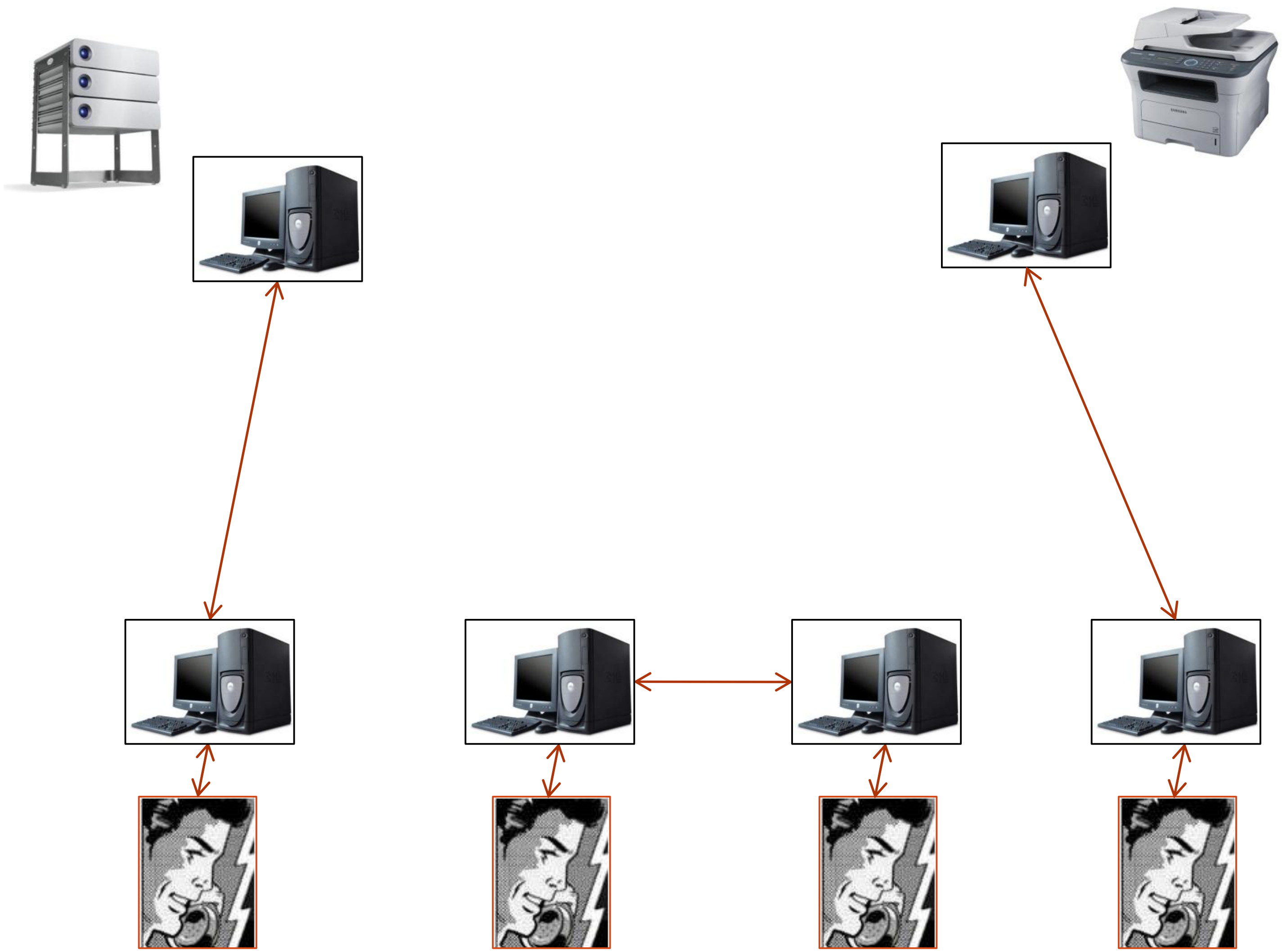
Key Distribution: Kerberos

Best understood using a dialogue in four scenes

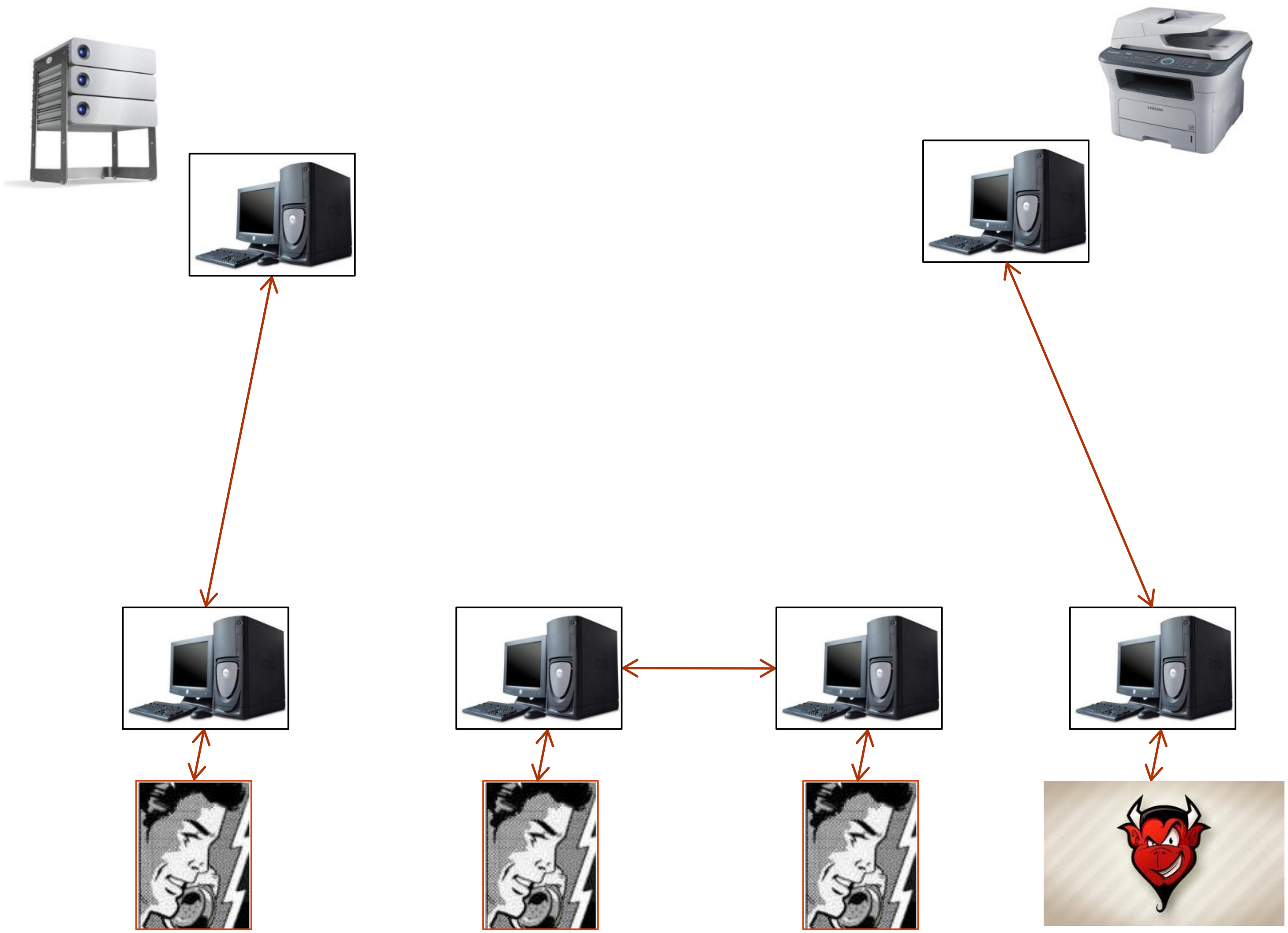
Kerberos: Scene I



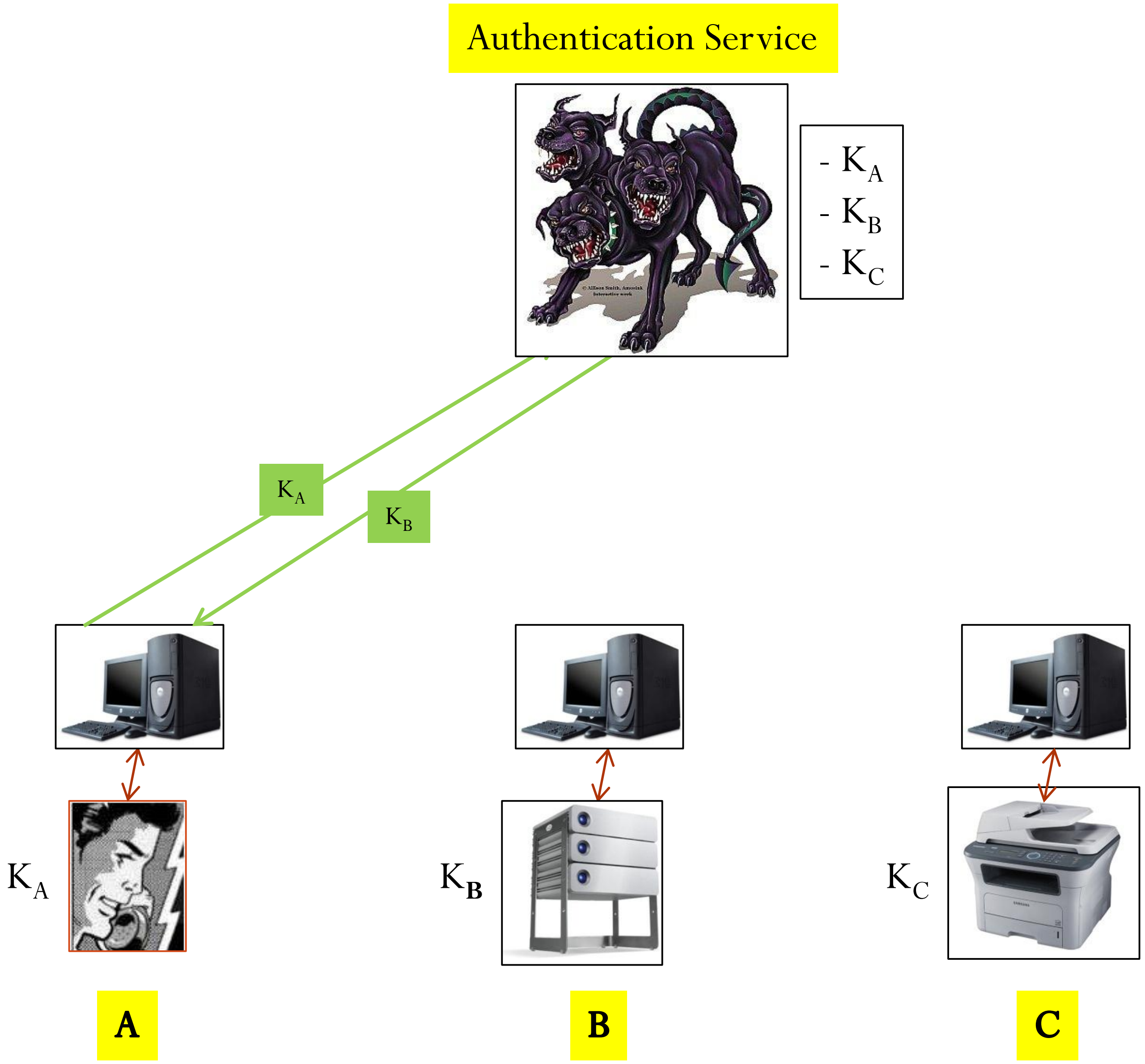
Kerberos: Scene I



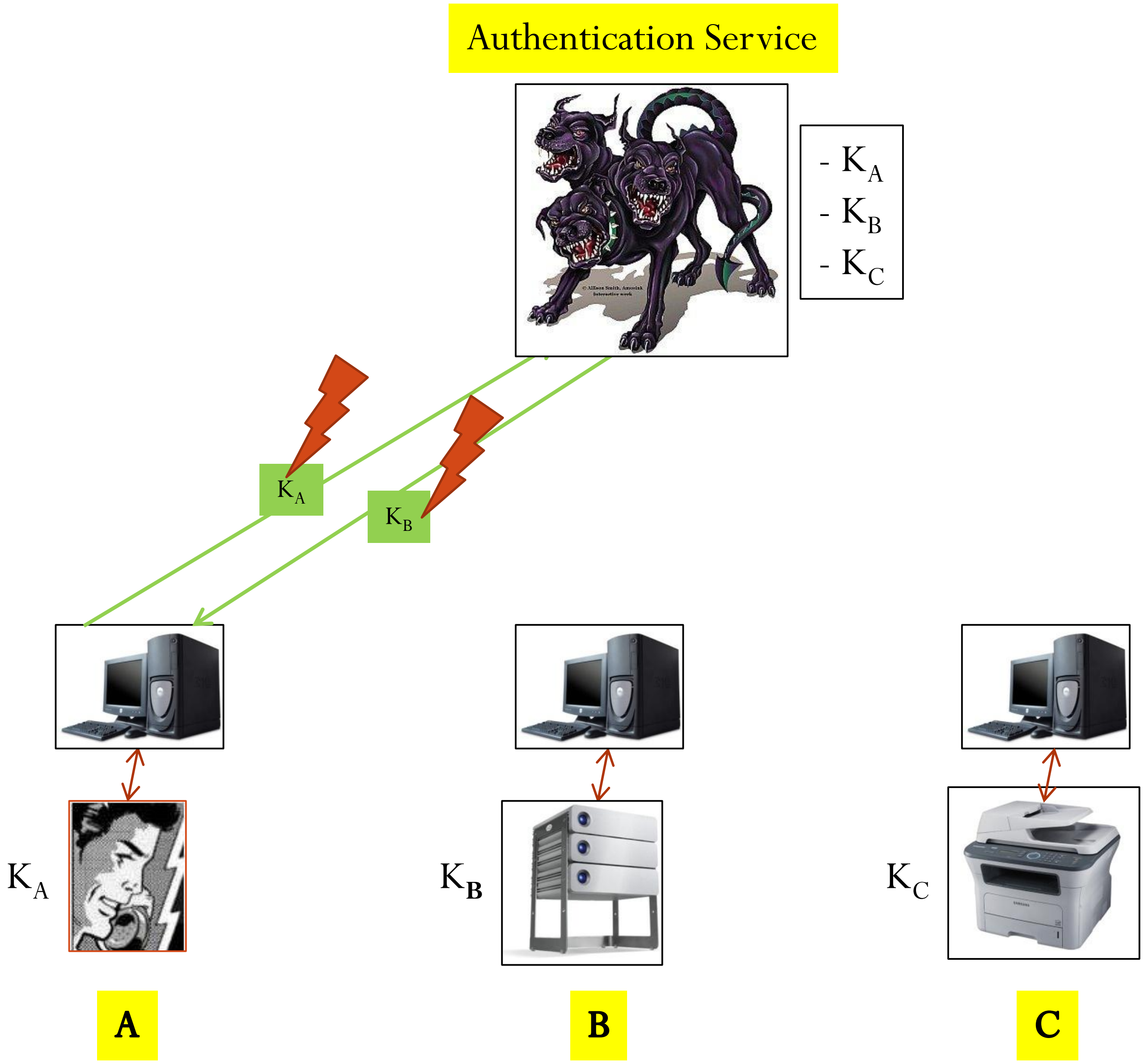
Kerberos: Scene I



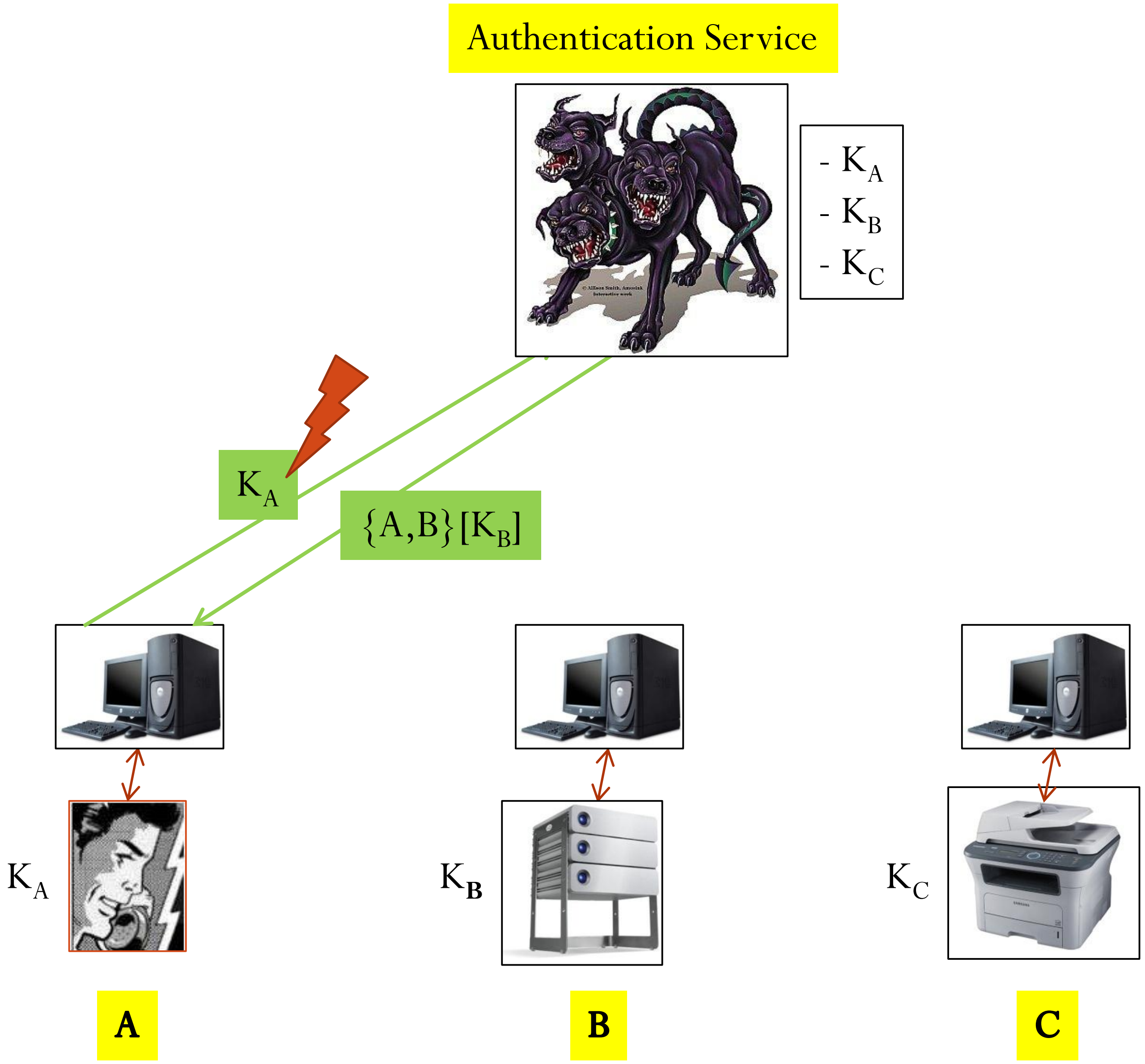
Kerberos: Scene II



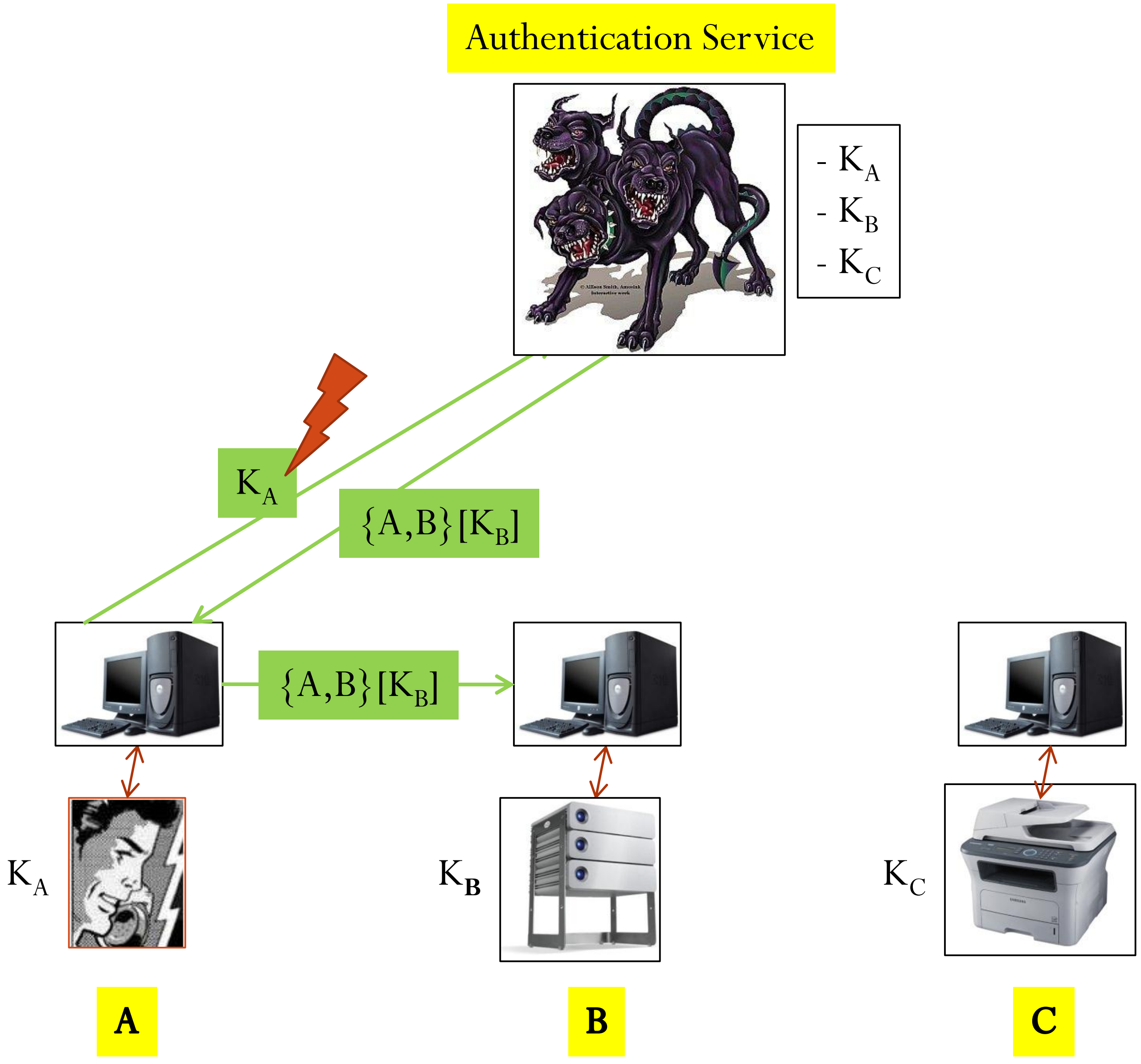
Kerberos: Scene II



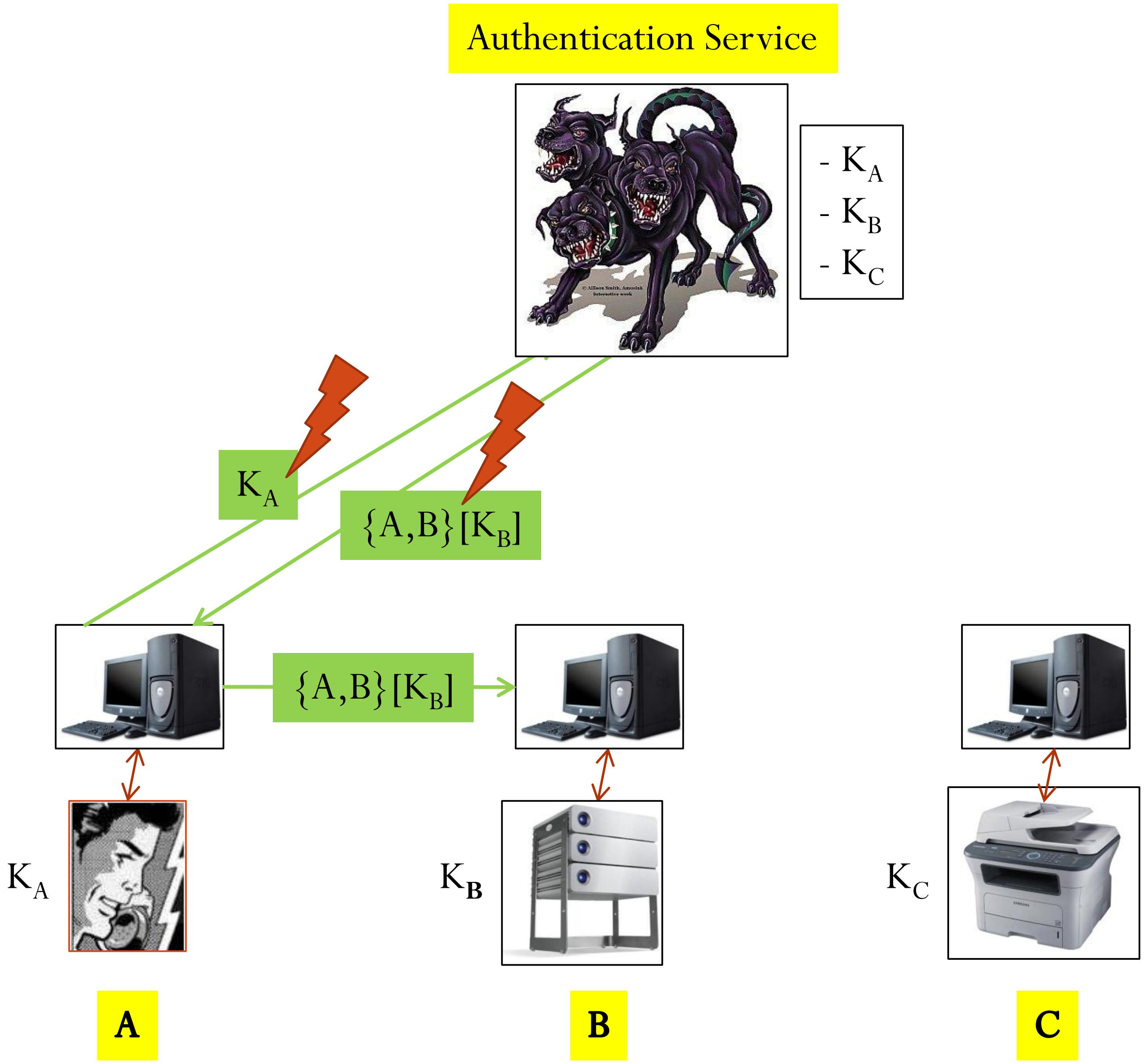
Kerberos: Scene II



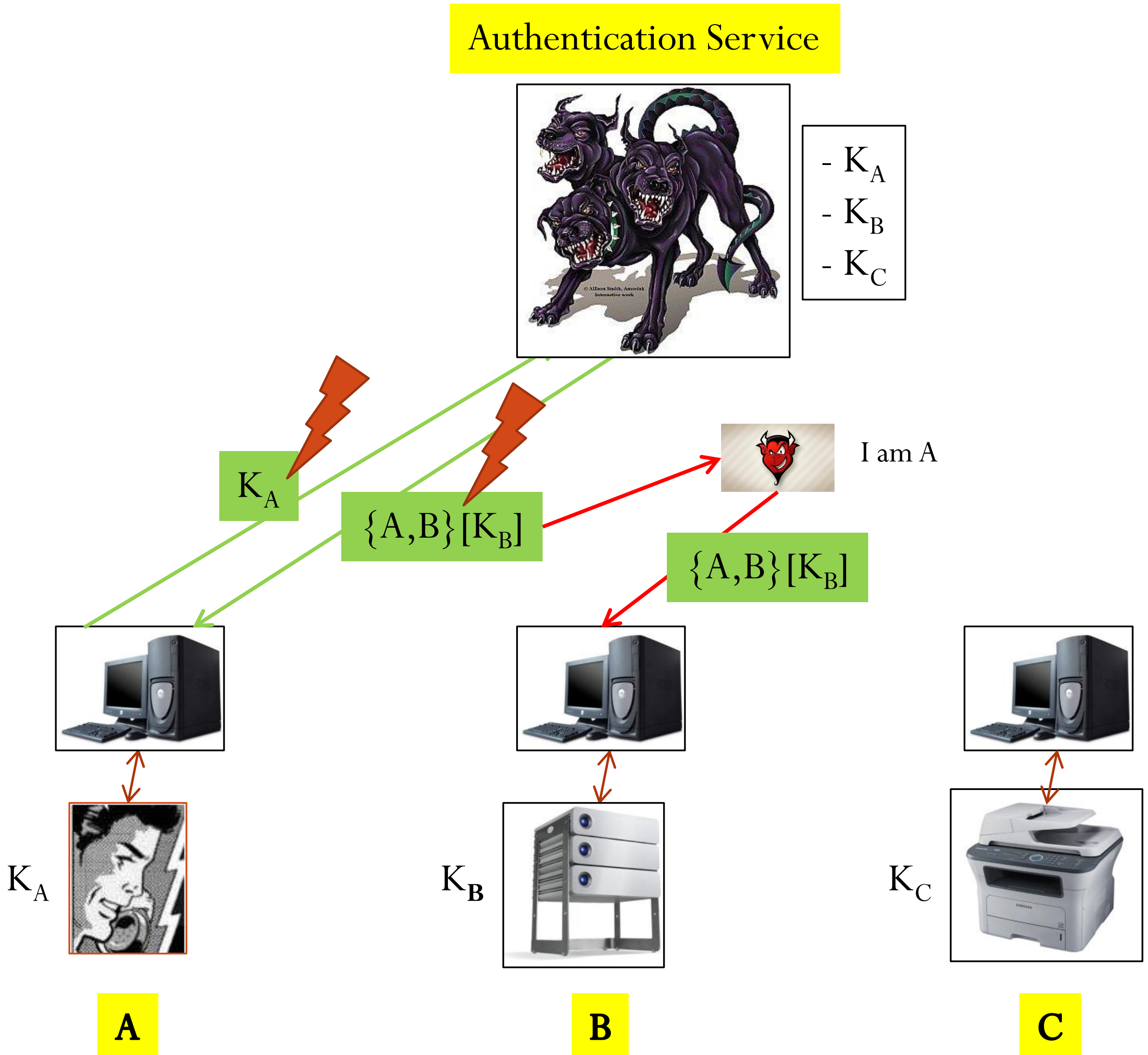
Kerberos: Scene II



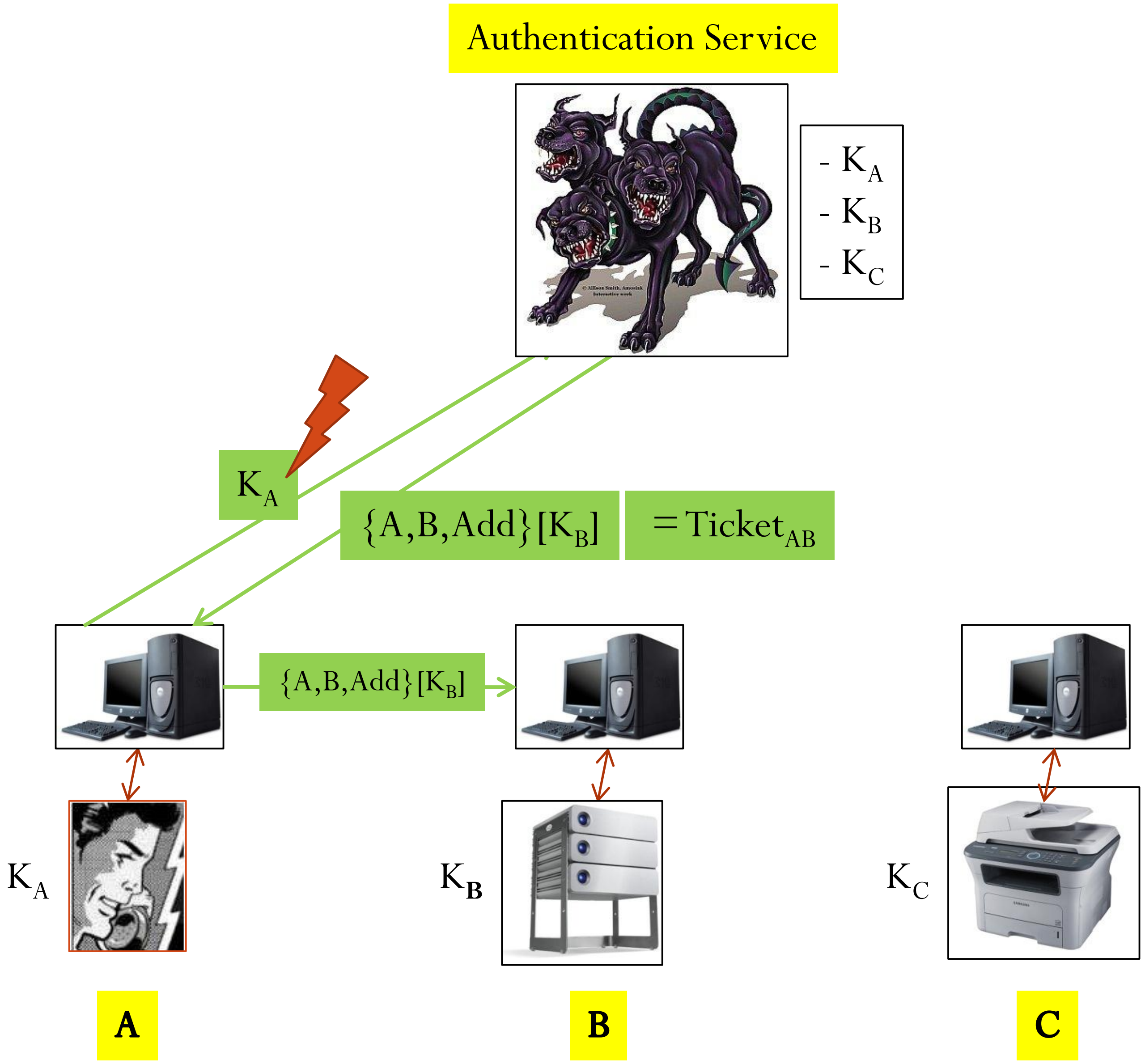
Kerberos: Scene II



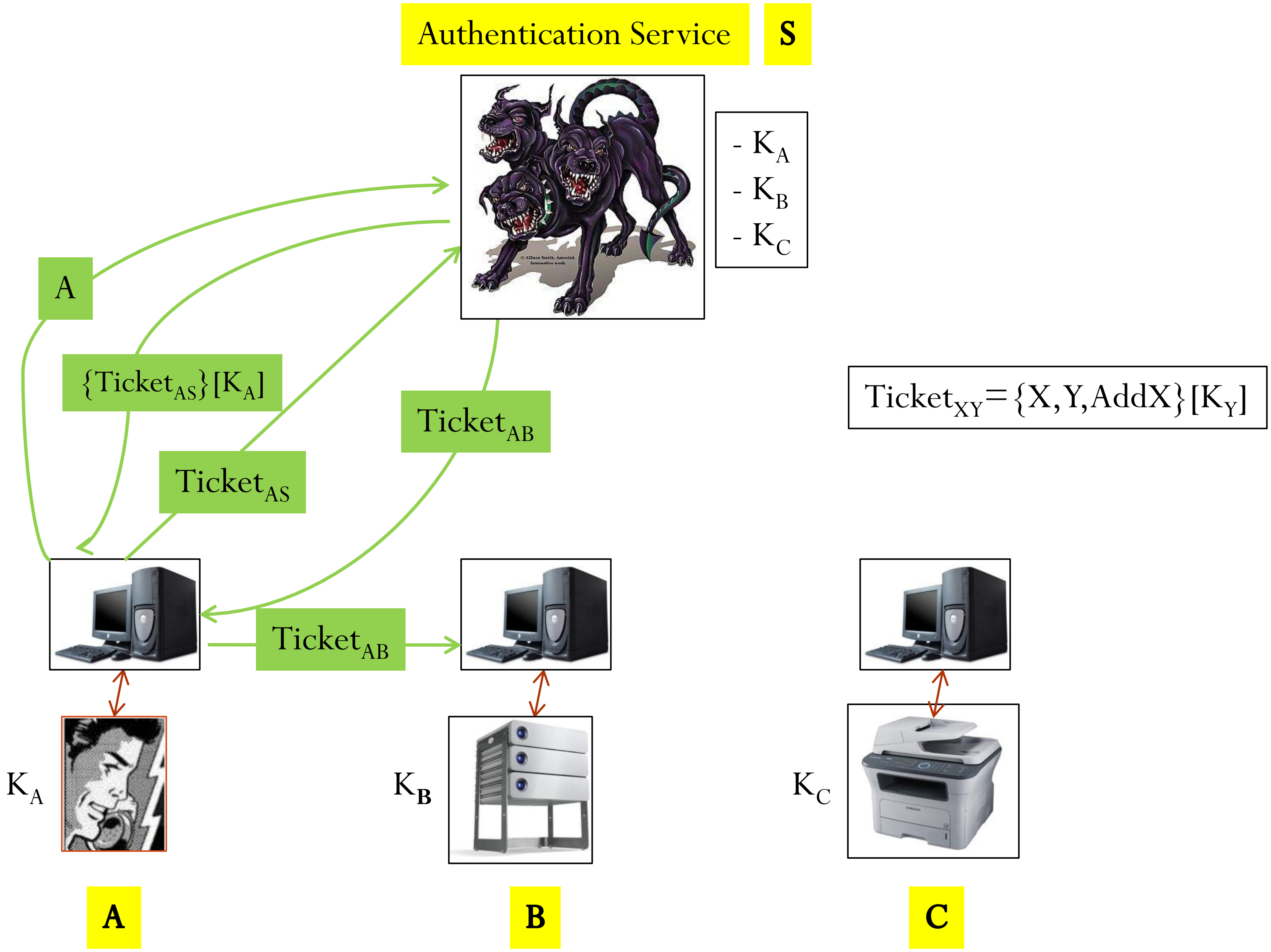
Kerberos: Scene II



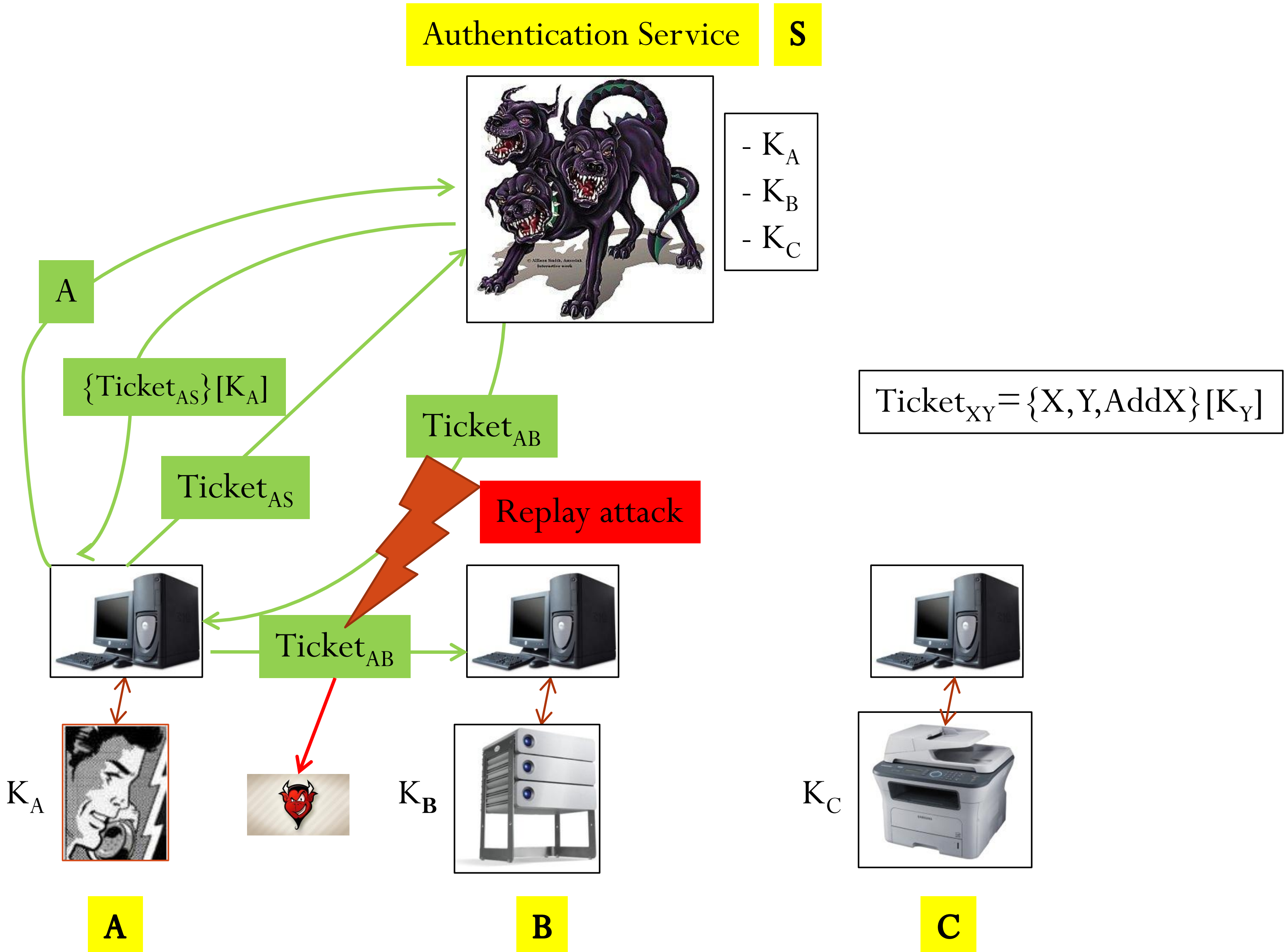
Kerberos: Scene II



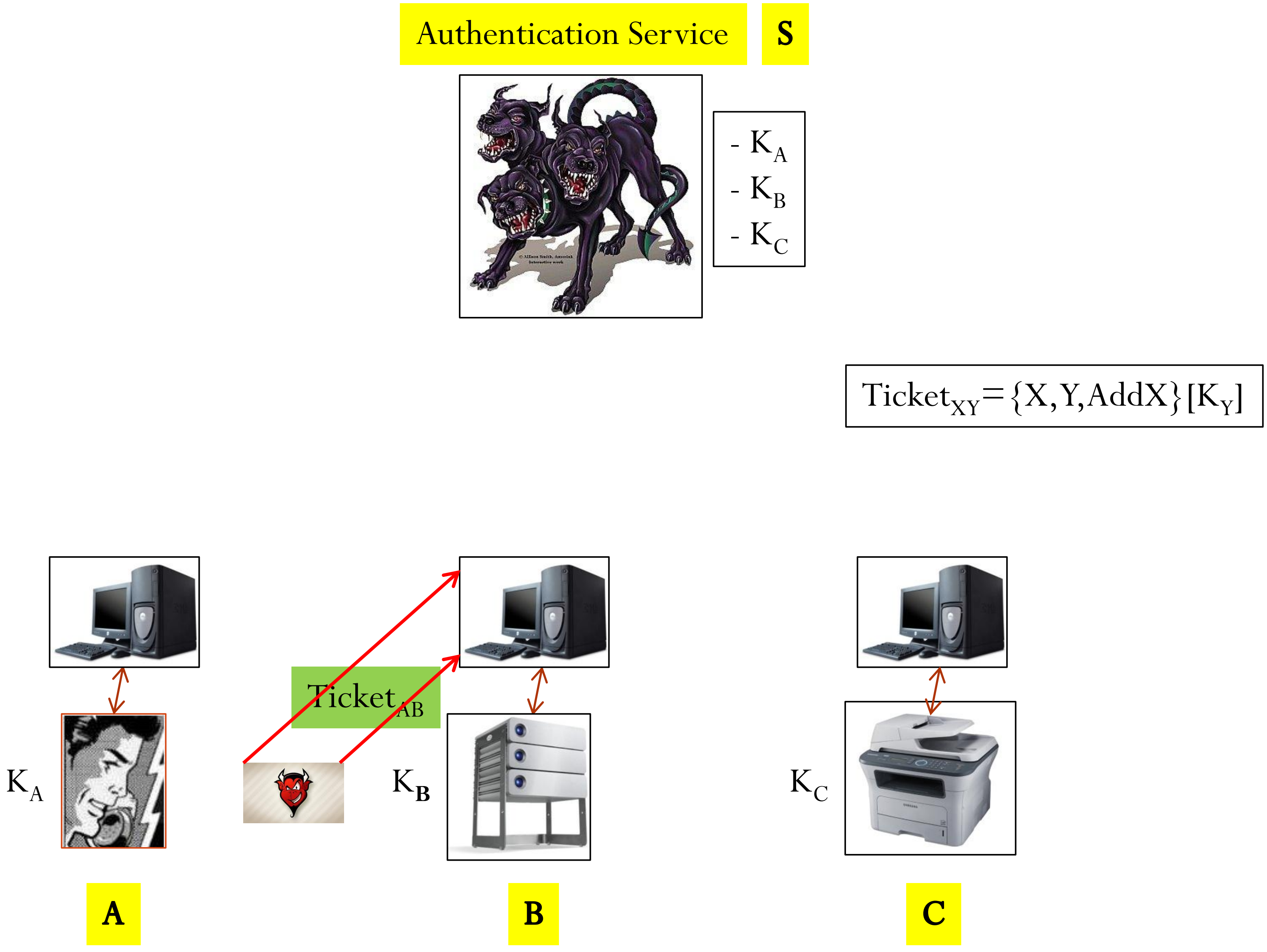
Kerberos: Scene III



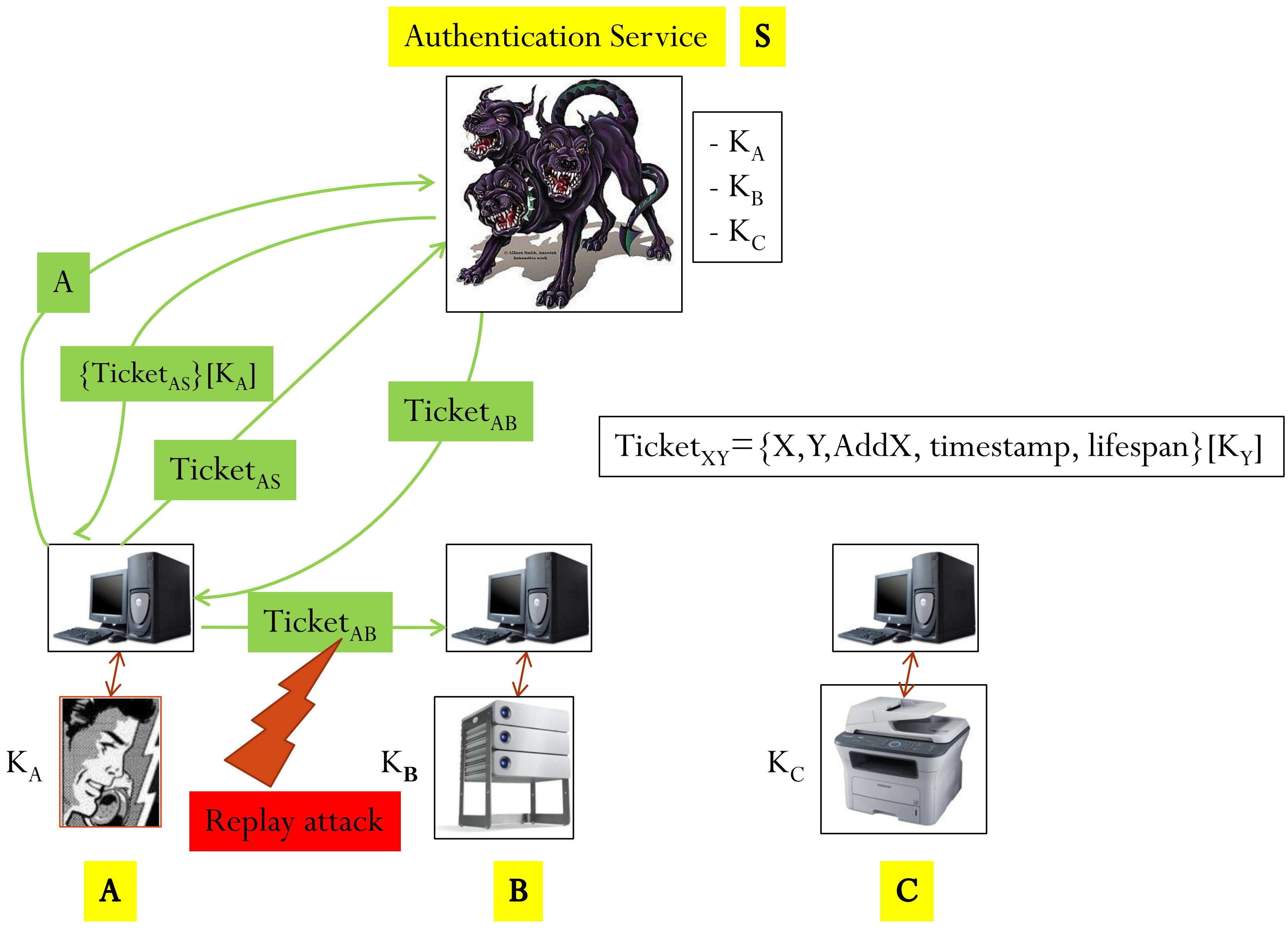
Kerberos: Scene III



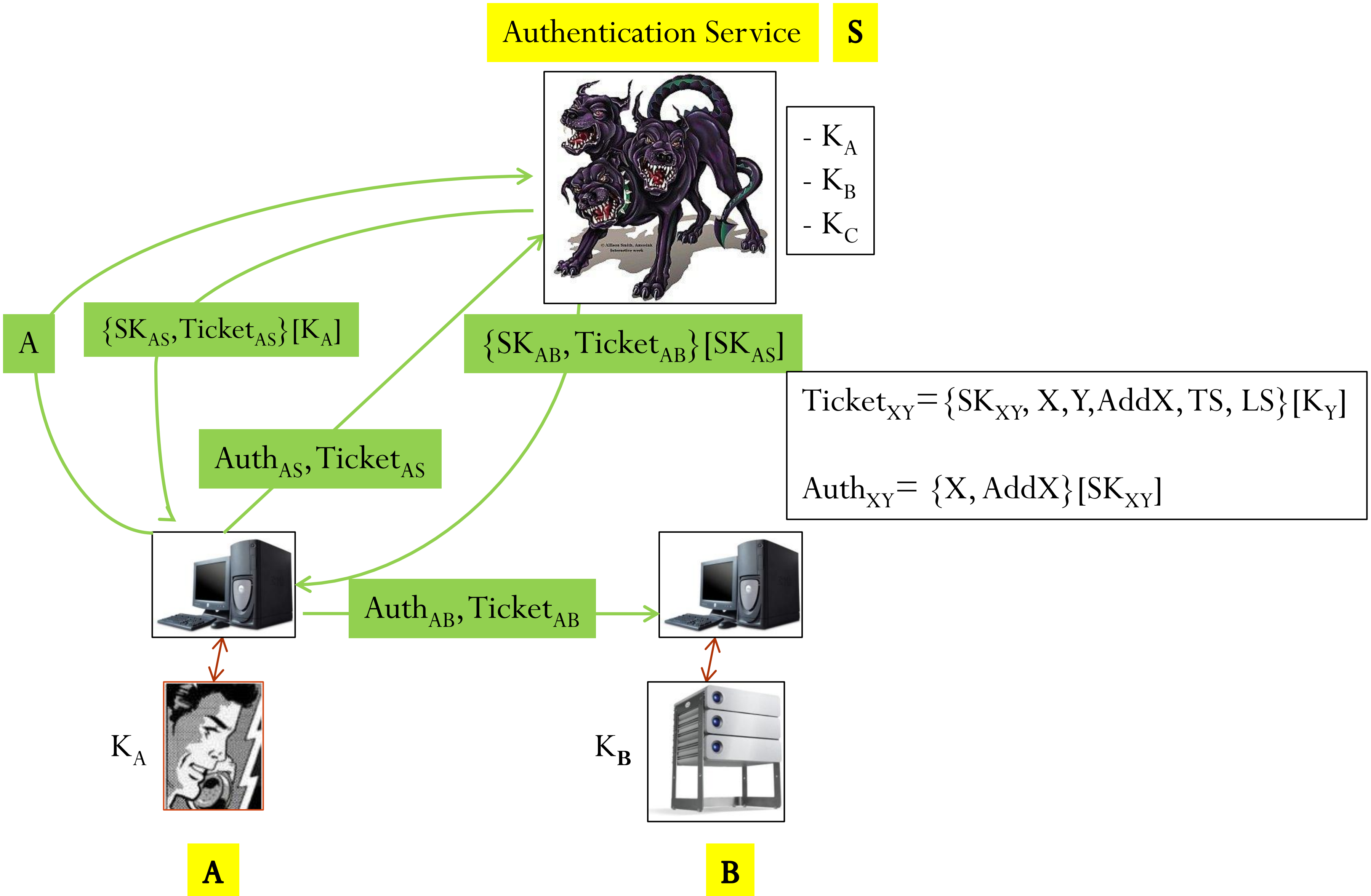
Kerberos: Scene III



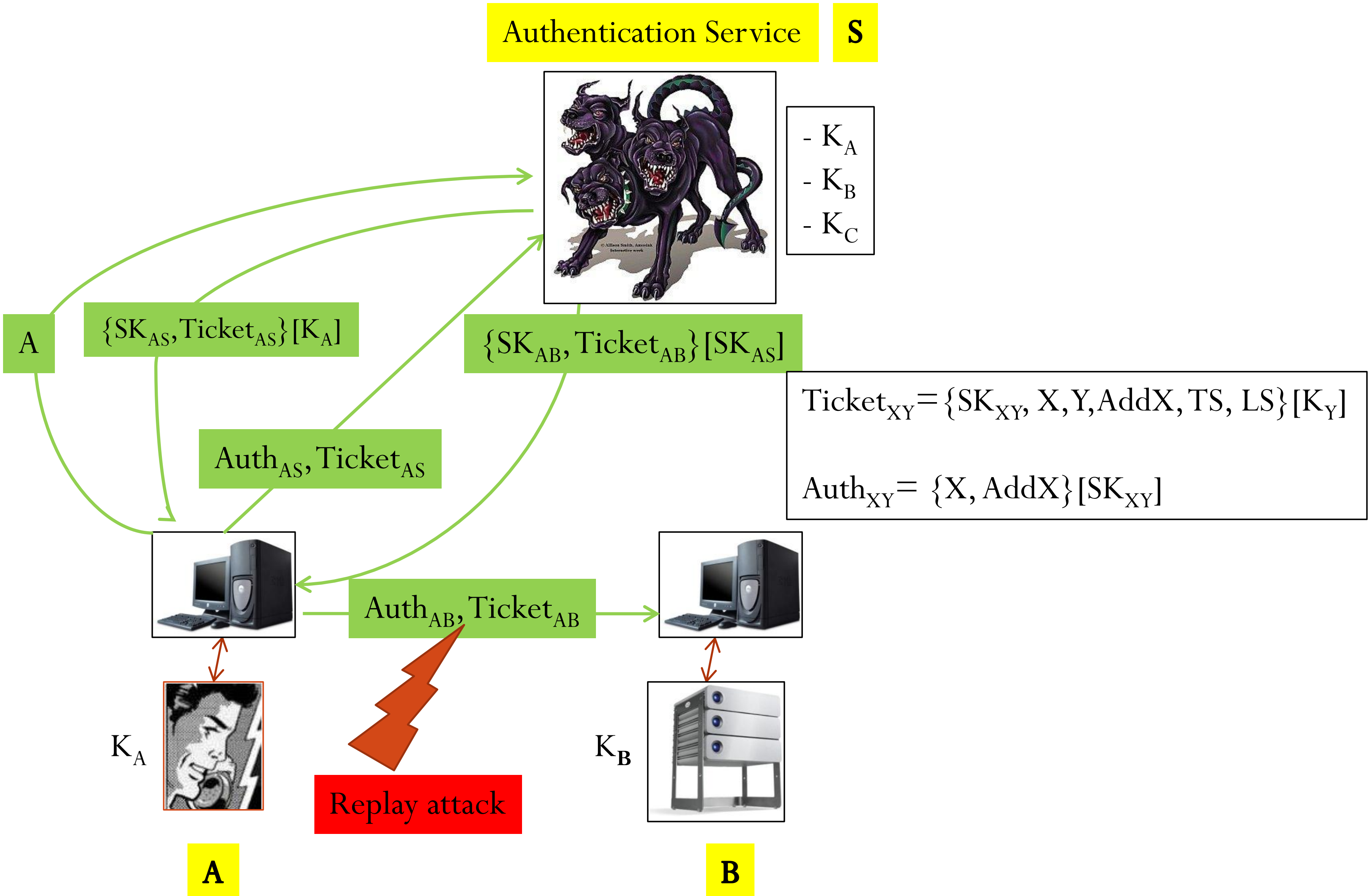
Kerberos: Scene III



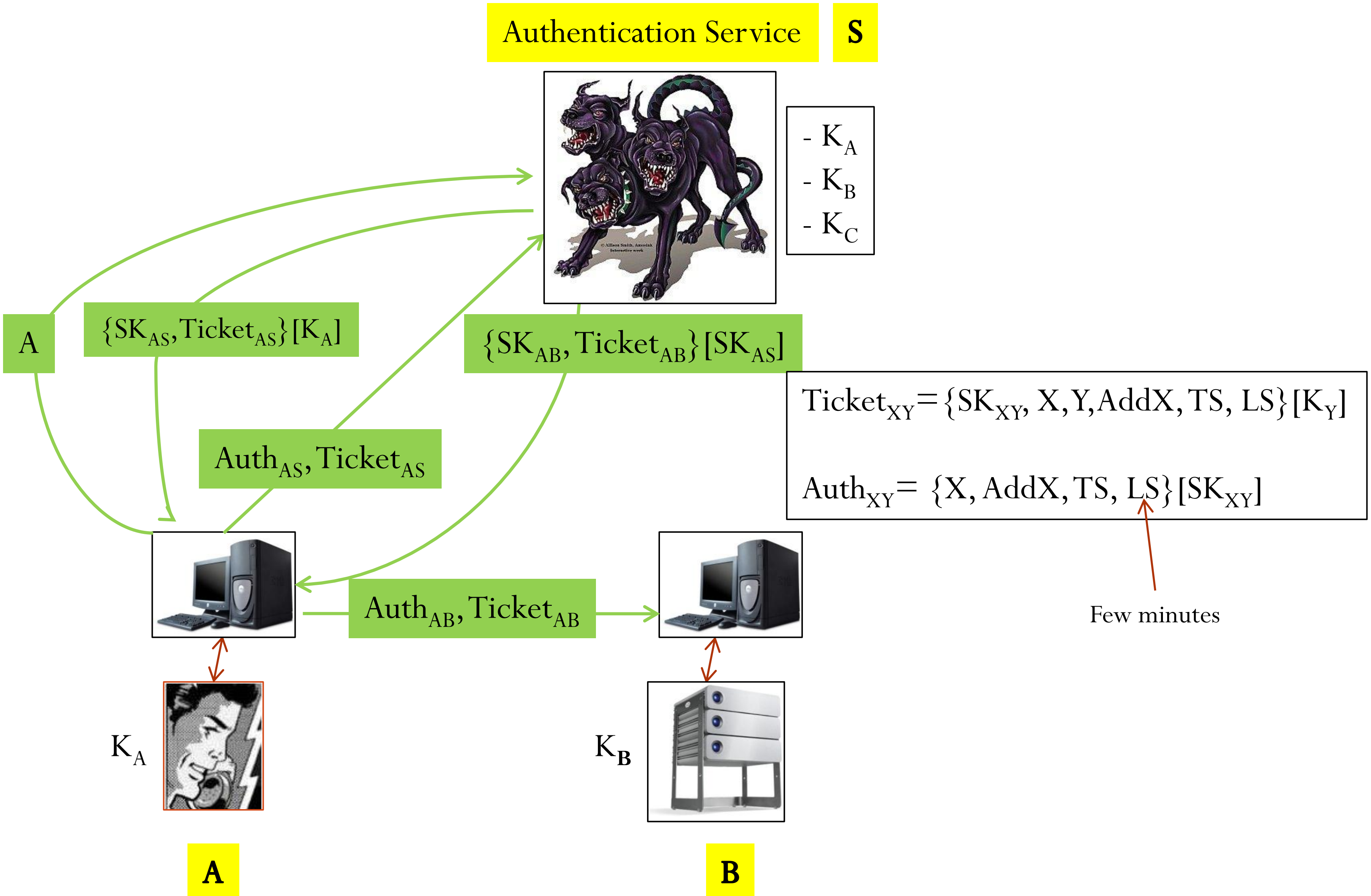
Kerberos: Scene IV



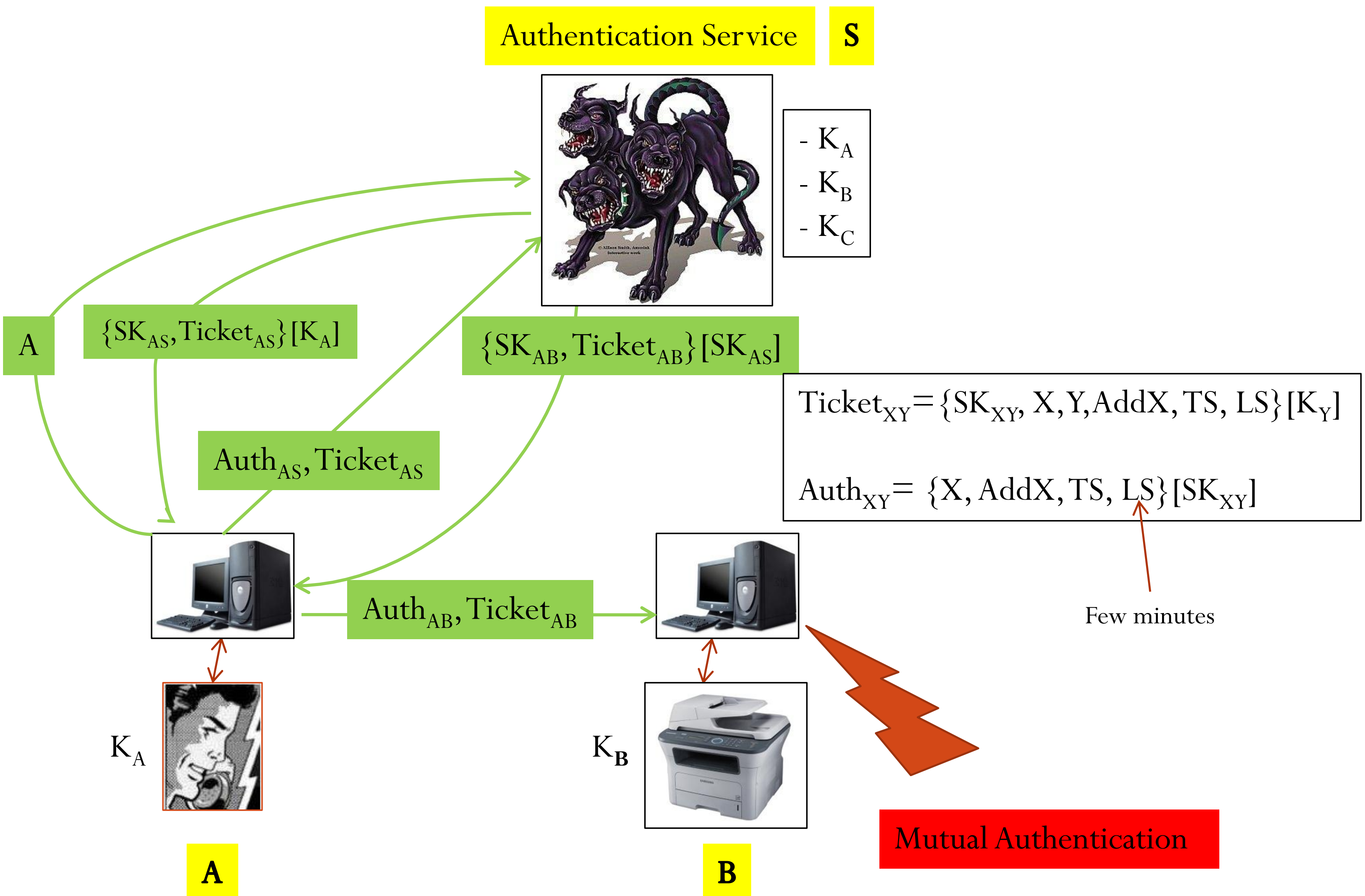
Kerberos: Scene IV



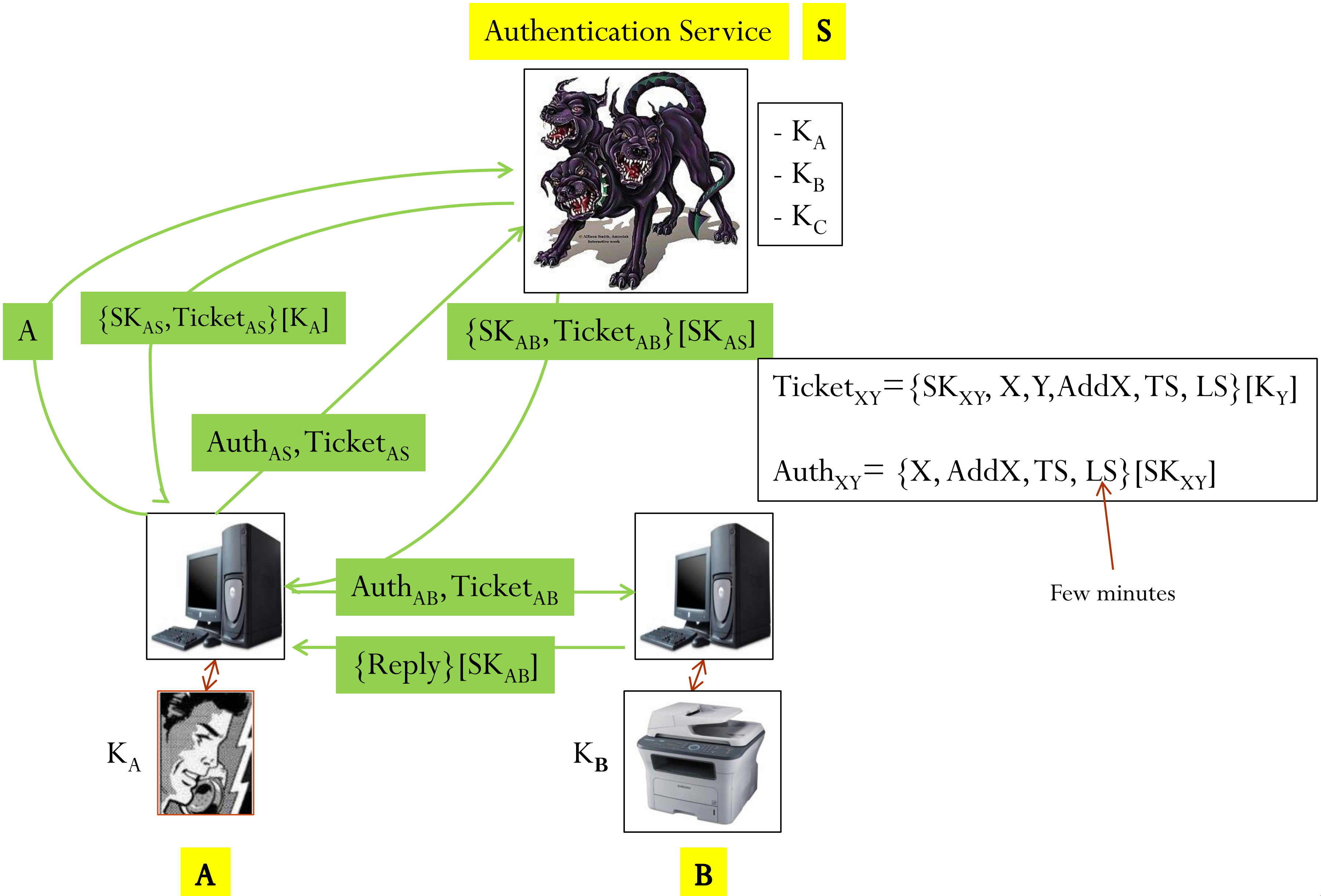
Kerberos: Scene IV



Kerberos: Scene IV



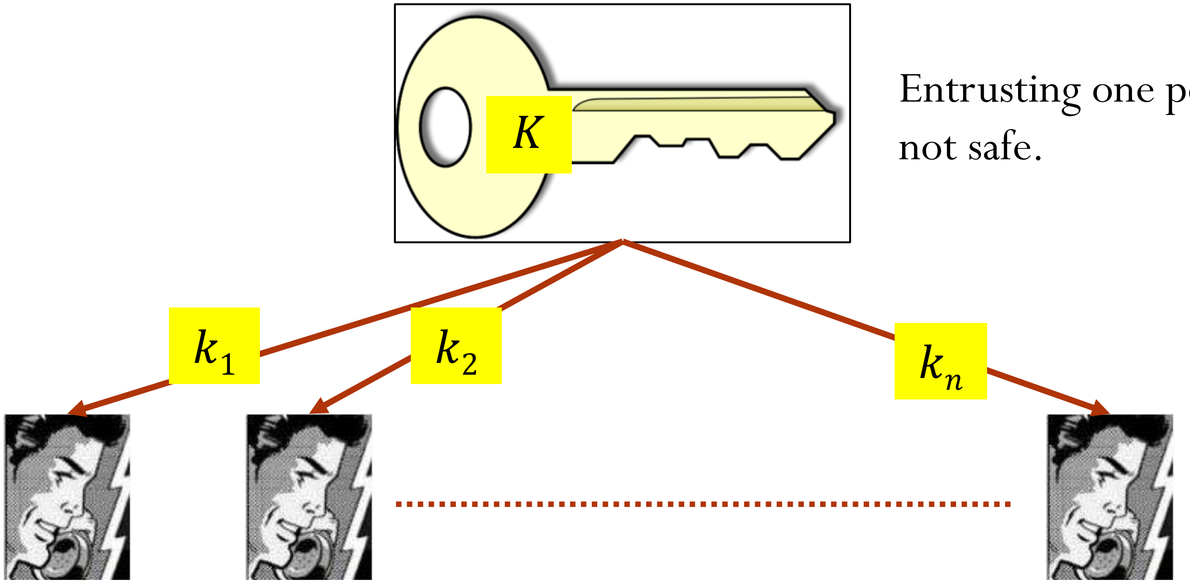
Kerberos: Scene IV



Other Cryptographic Protocols

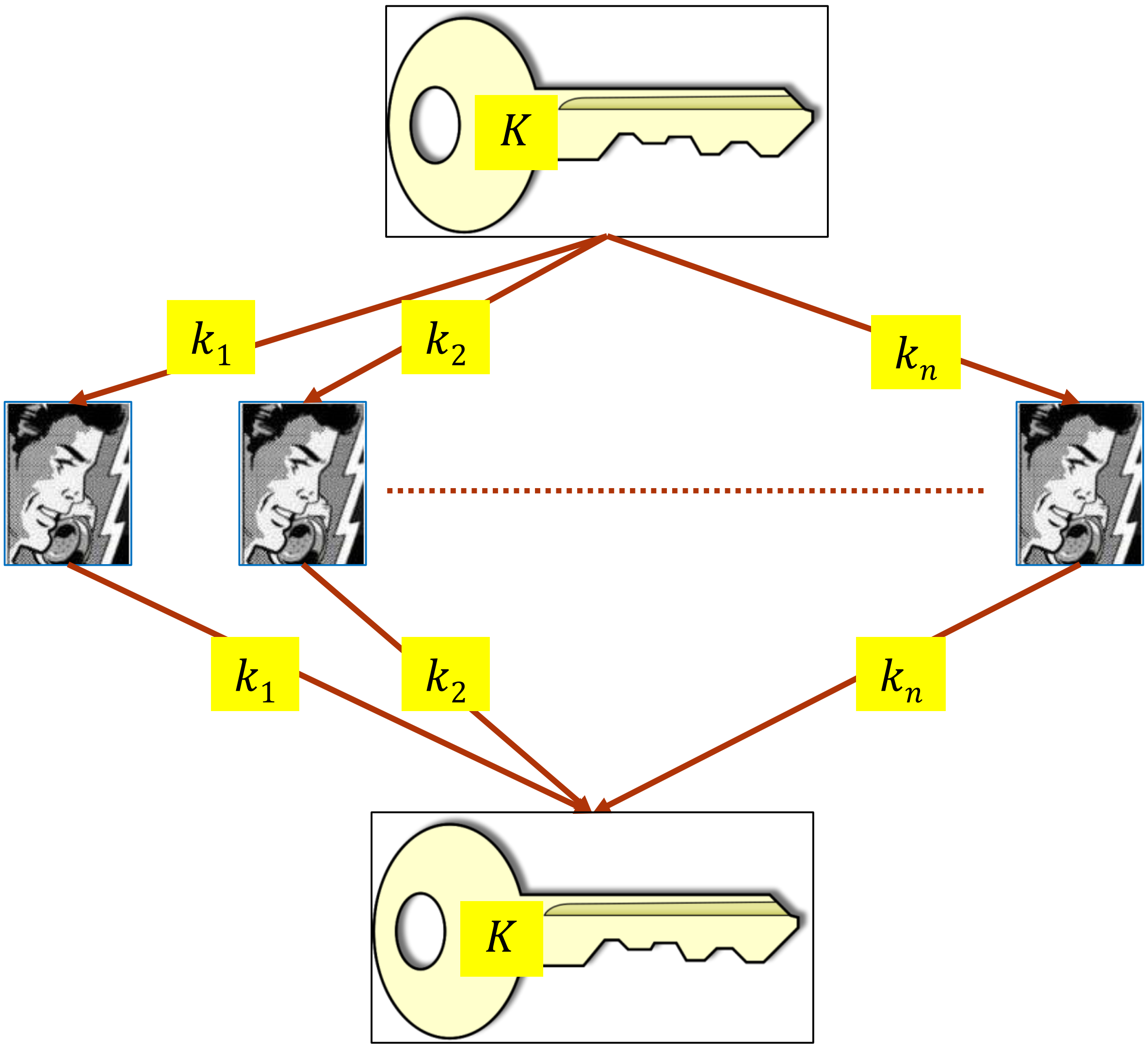
- Secret sharing
- Coin flipping over phone
- Oblivious transfer

Secret Sharing

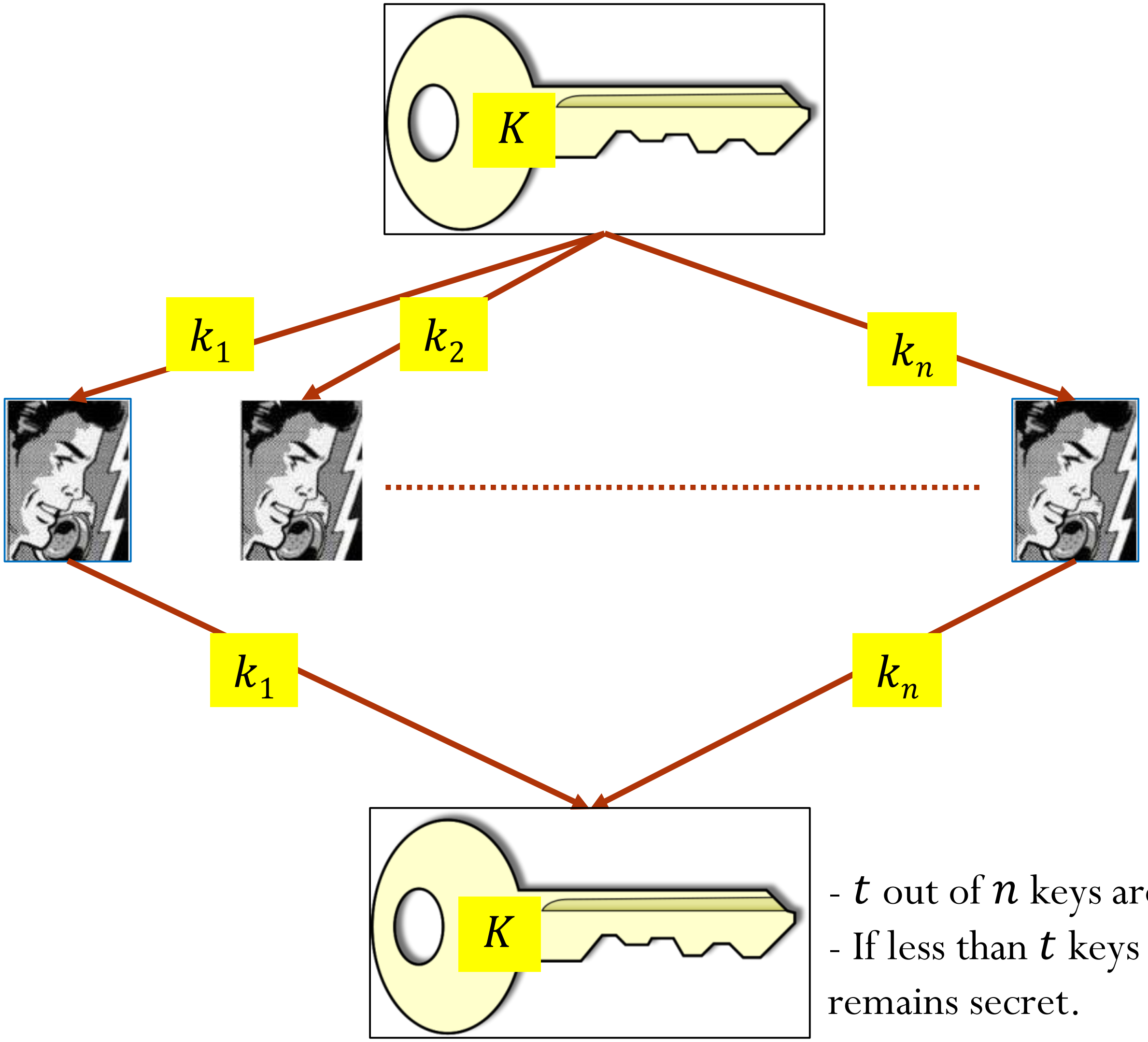


Entrusting one person with K is not safe.

Secret Sharing



Secret Sharing



- t out of n keys are sufficient to obtain K
- If less than t keys are available, then K remains secret.

Secret Sharing

- How do we construct such a protocol?
  - Ideas?

Secret Sharing

- How do we construct such a protocol?
  - Shamir's secret sharing protocol: A degree d polynomial is completely determined by d points evaluated on the polynomial.

Coin flipping

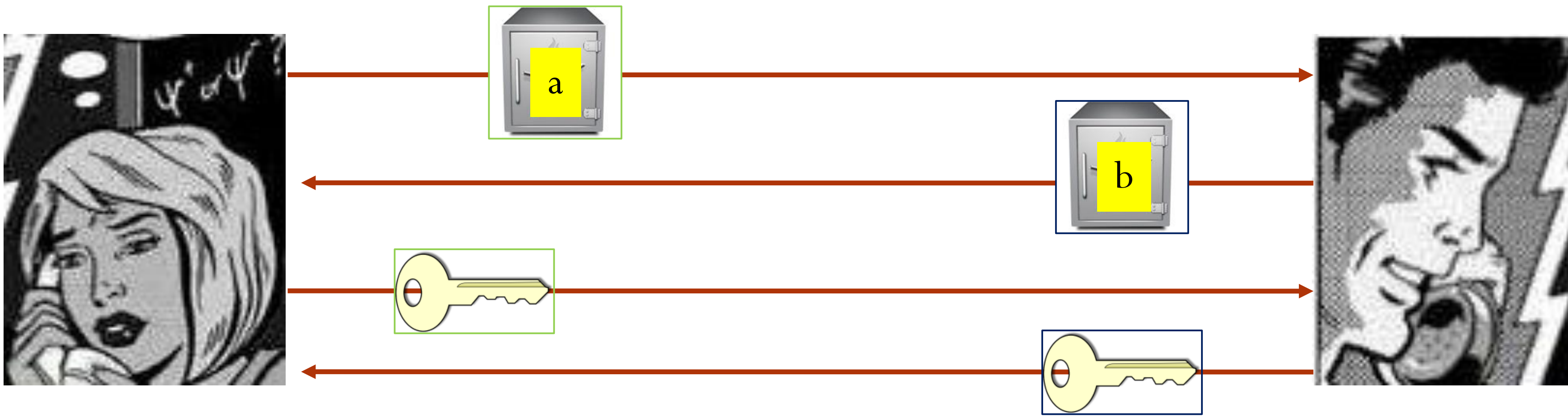
Alice and Bob want to agree on a secret bit.

Coin flipping

Alice and Bob want to agree on a secret bit.

Coin flipping

Alice and Bob want to agree on a secret bit.



Bit commitment protocol

Other protocols we did not talk about

- Oblivious transfer.
- Multi-party computation.
- Electronic voting.
- Homomorphic Encryption.

End

Slides 7-21 have been borrowed from Prof. Mihir Bellare's lecture slides.