

Encryption on the Internet

Excerpt from the *Internet Security* course to be offered at CTU

Dr. Garrison Q. Kenney
NCSU Computer Training Unit

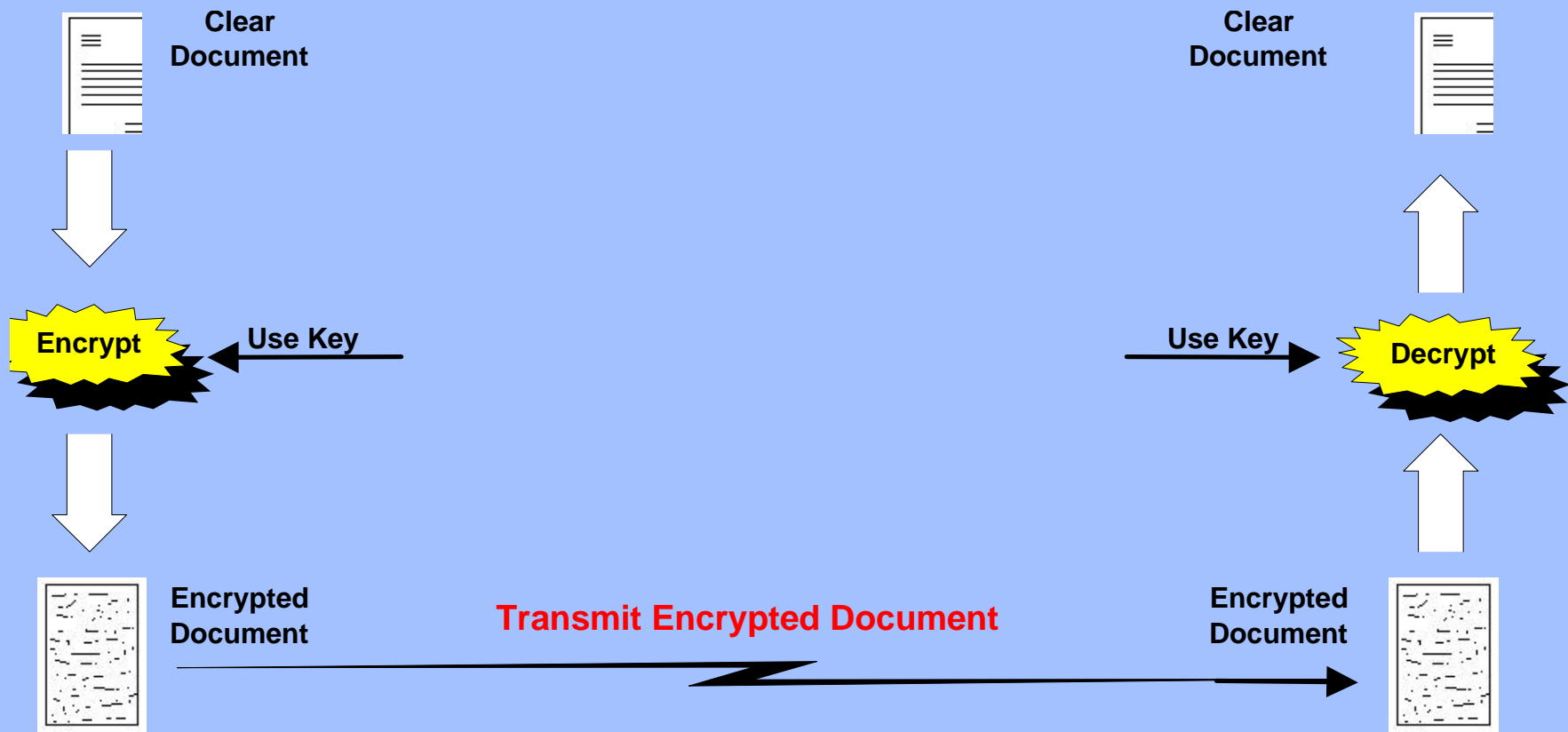
NC TECHNOLOGY SHOWCASE 2000

Raleigh NC March 23, 2000

Encryption Systems

- ◆ *Encryption* is the *transformation* of data into some *scrambled* form.
- ◆ In data communications, we usually want to encrypt a message before transmitting it over an *insecure or untrusted network*.
- ◆ *Decryption* is the reverse of encryption, putting encrypted data back *into the clear* so the intended recipient can read it.

Encryption Systems



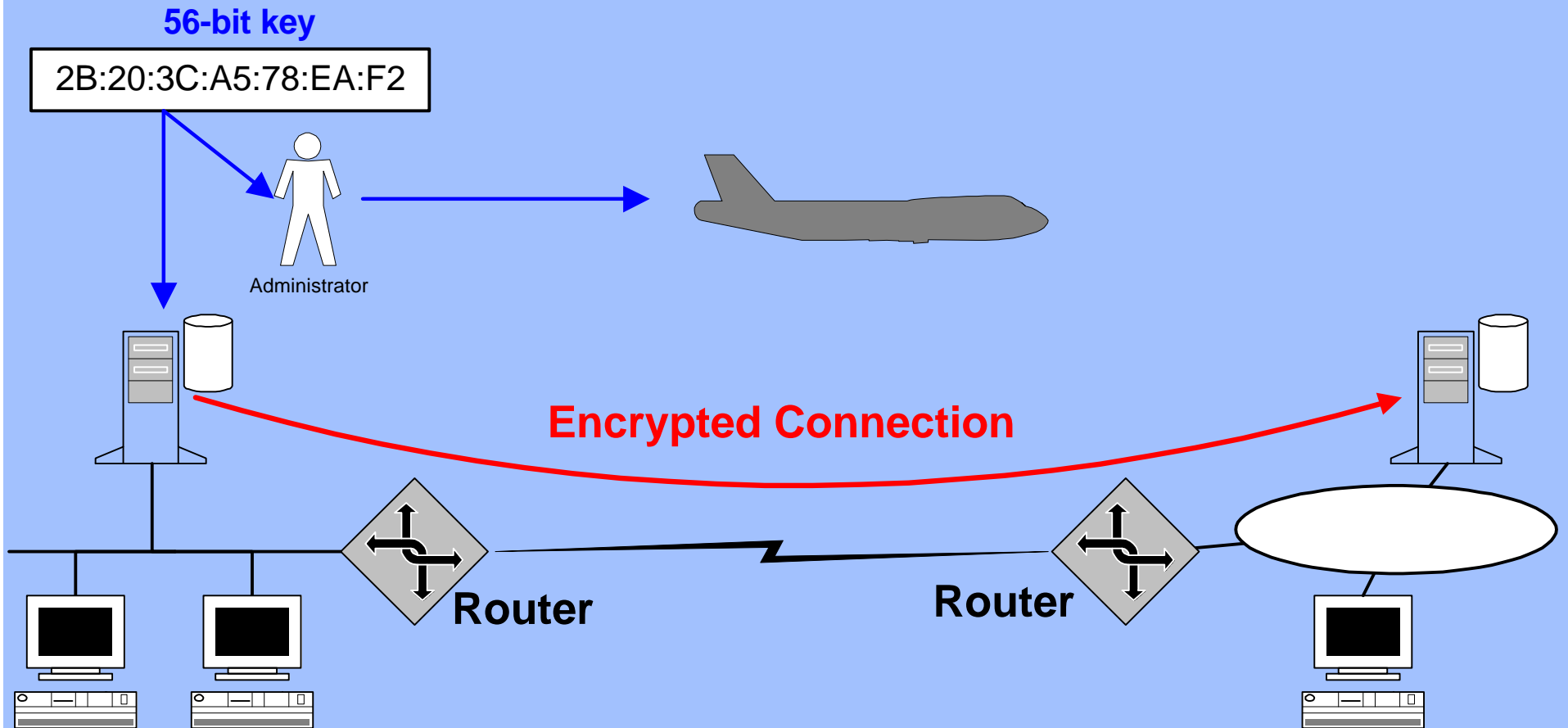
Two Types of Encryption

- ◆ Secret-key encryption systems
- ◆ Public-key encryption systems

Secret-key Encryption

- ◆ *Classic* encryption method.
- ◆ Also called *symmetric-key* encryption.
- ◆ The same key is used to *both encrypt and decrypt* a message.
- ◆ Security, in general, depends on the length of the key.
- ◆ Several good methods are in use.
- ◆ *Problem* is key distribution.

Secret-key Encryption



Secret-key Methods

- ◆ Data Encryption Standard (DES) is the most widely-used. Fixed 56-bit key.
- ◆ RC2 and RC4 are newer and faster. Variable length key from 40 to 128 bits.
- ◆ Triple-DES (3DES) is literally running DES three times to get stronger encryption.
- ◆ Several others.

Cracking the Code

- ◆ Exhaustive key search is the only way to crack good codes.
- ◆ Assume a machine that can try 100 million keys per second:
 - 40-bit key takes $2^{40} = 10^{12} = 3$ hours.
 - 56-bit key takes $2^{56} = 10^{16} = 10$ years.
 - 128-bit key takes $2^{128} = 10^{38} = 10^{23}$ years.
- ◆ 1000 such machines in parallel can crack DES in 3 days, but 128-bit is still 10^{20} years!

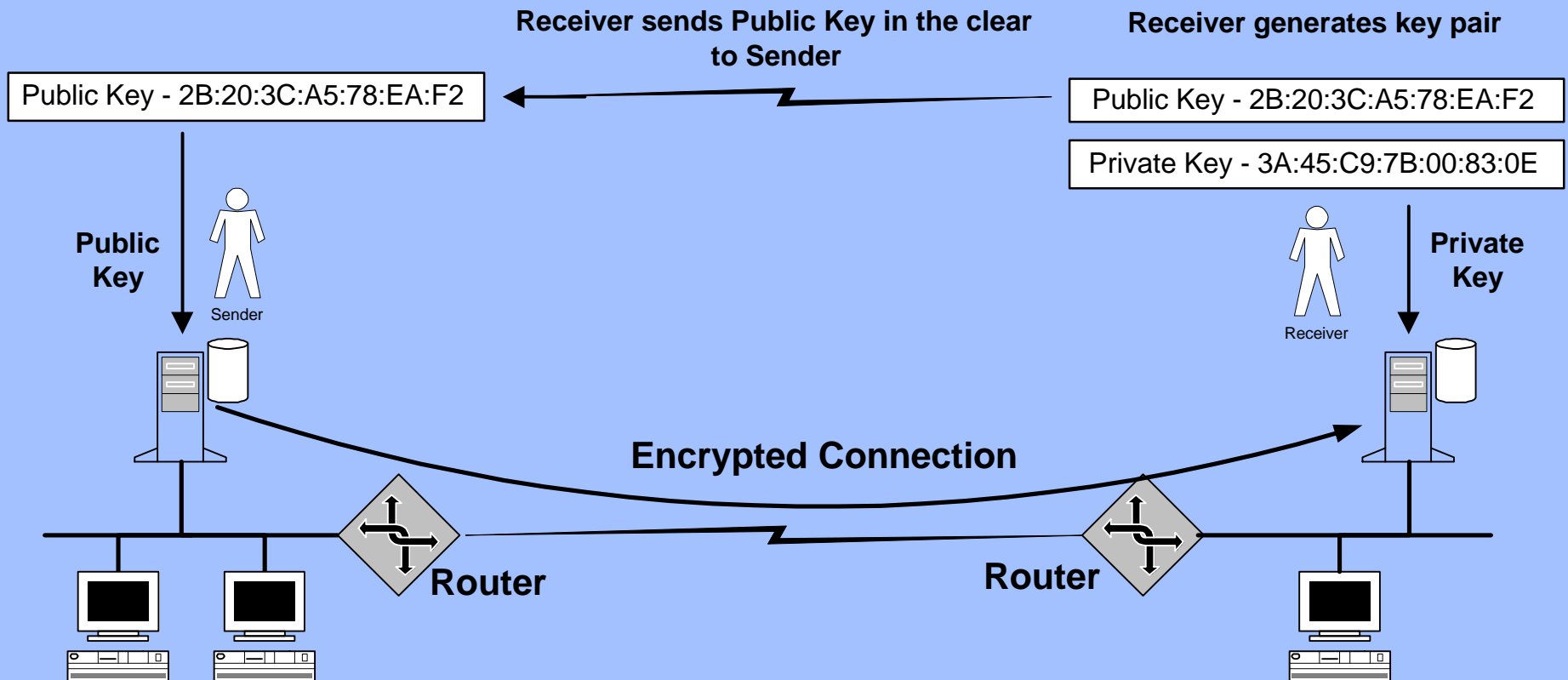
How Good is Good Enough?

- ◆ The longer the key, the more time it takes to break the code.
- ◆ But -- the longer the key, the more time it takes to *legitimately encrypt and decrypt* the message!
- ◆ Question -- how valuable is your data to an attacker?
- ◆ Difference between *prudence* and *paranoia*!

Public-key Encryption

- ◆ *Two keys are used* -- one to encrypt the message and the other to decrypt it.
- ◆ Pick two large prime numbers and generate a *matching pair of keys* from them.
- ◆ Encrypt a message with one and you can decrypt it only with the other.
- ◆ RSA security is the most widely-used.

Public-key Encryption



RSA Cryptography

- ◆ $\text{CipherText} = \text{PlainText}^{\text{PublicExp}} \bmod n$
- ◆ $\text{PlainText} = \text{CipherText}^{\text{PrivateExp}} \bmod n$
- ◆ Actually, *it doesn't matter* which of the two keys you use for the public key and the private key.

A 512-bit RSA Key

- ◆ **modulus:**

00:9a:92:25:ed:a4:77:69:23:d4:53:05:2b:1f:3a:
55:32:bb:26:de:0a:48:d8:fc:c8:c0:c8:77:f6:5d:
61:fd:1b:33:23:4f:f4:a8:2d:96:44:c9:5f:c2:6e:
45:6a:9a:21:a3:28:d3:27:a6:72:19:45:1e:9c:80:
a5:94:ac:8a:67

- ◆ **publicExponent:** 65537 (0x10001)

- ◆ **privateExponent:**

00:a9:0f:30:6c:bb:75:df:89:50:b1:7c:f5:ad:32:
1f:fd:5c:b5:26:26:19:87:3a:f4:57:e6:eb:4e:8a:
d4:a1:ff:6a:99:7b:e3:a0:d0:af:ed:83:4c:db:c3:
75:2c:d9:8a:13:f6:cb:48:f2:7e:b0:f1:9e:ed:3b:
09:73:fe:01

Public-key Encryption Problems

- ◆ Very *processor-intensive* operation -- 100 to 1000 times slower than secret-key operations.
- ◆ Sender can masquerade as anybody and send a bogus encrypted message. This is the *Authentication* problem.
- ◆ Someone could intercept the transmission and modify the (encrypted) message in transit. Known as the *man-in-the-middle* problem.

Solving the First Problem

- ◆ Use *both* secret-key encryption and public-key encryption together.
 - use the public-key system to *send just the (relatively short) single symmetric key* of a secret-key encryption system.
 - use secret-key encryption (e.g. DES) to *encrypt the actual message*.

3. Sender picks a random secret key



Secret Key

Encrypt

4. Sender encrypts the secret key using the receiver's public key

2. Receiver transmits the public key (in the clear)

1. Receiver generates key pair

Public Key - 2B:20:3C:A5:78:EA:F2

Private Key - 3A:45:C9:7B:00:83:0E

5. Sender transmits the secret key (encrypted with receiver's public key)

Encrypted secret key

Encrypted secret key

6. Receiver decrypts the secret key using his private key

Decrypt

Secret Key

Clear Document

Clear Document

7. Sender encrypts the message using the secret key

Encrypt

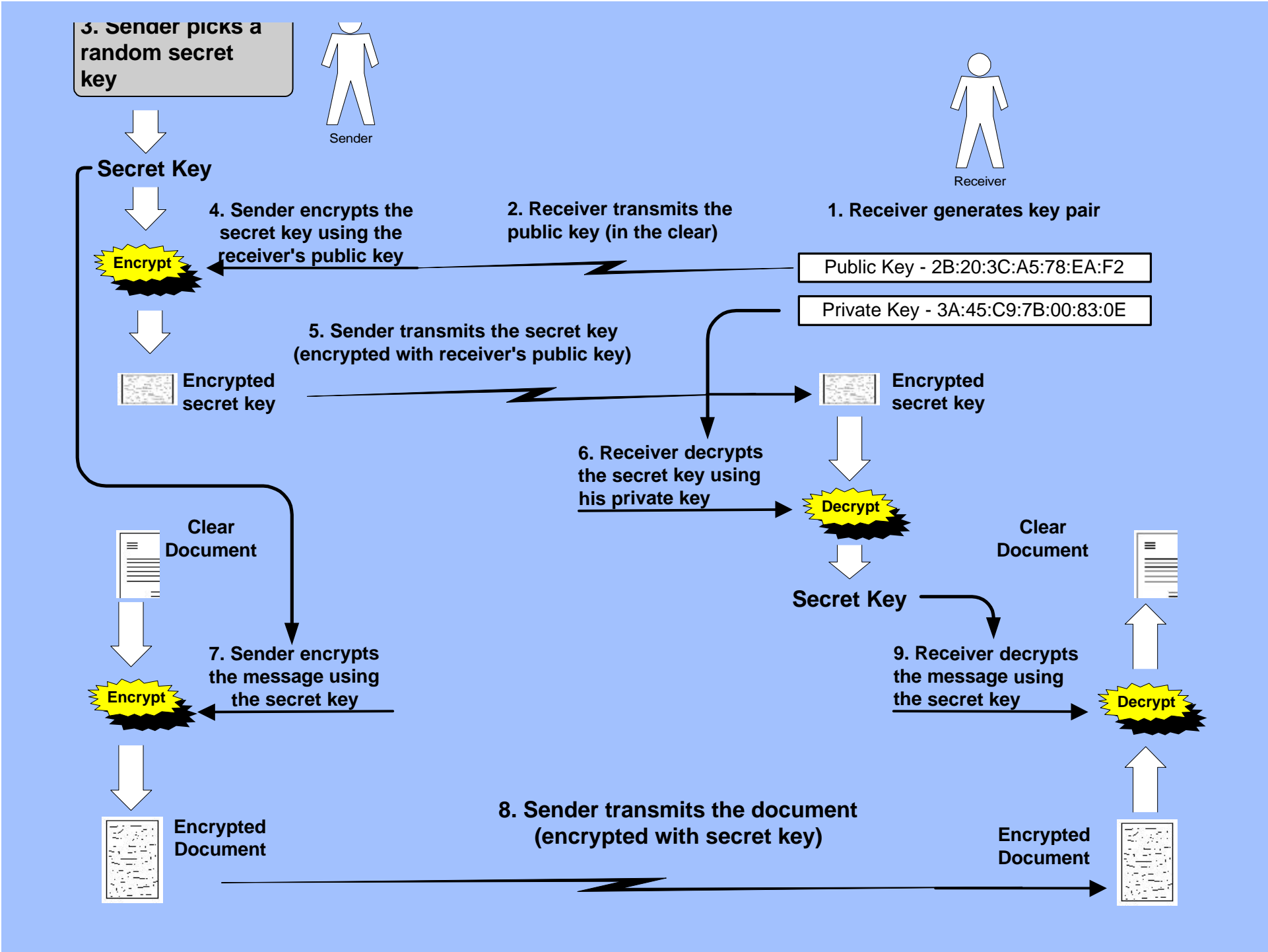
9. Receiver decrypts the message using the secret key

Decrypt

8. Sender transmits the document (encrypted with secret key)

Encrypted Document

Encrypted Document



Authenticating the Sender

- ◆ The sender "signs" the document by encrypting something with her private key.
- ◆ If the receiver can decrypt it using the sender's public key, he can be reasonably assured that it came from the owner of that public key.

Example Transfer

- ◆ Both Alice and Bob have generated a key pair and *exchanged their respective public keys beforehand*.
- ◆ Alice has the desired message to be sent in clear text.
- ◆ Alice "signs" the clear text by *encrypting her name with her private key* and adding it to the bottom of the clear text.
- ◆ Alice then *encrypts the entire message, including her encrypted signature*, using Bob's public key.
- ◆ Alice sends the encrypted message to Bob.
- ◆ Bob *decrypts the message using his private key*.
- ◆ Bob then *decrypts the embedded signature using Alice's public key*.
- ◆ If the signature comes out in the clear, Bob can be assured that it came from Alice, *because only Alice could have encrypted it with her private key*.

The Man in the Middle

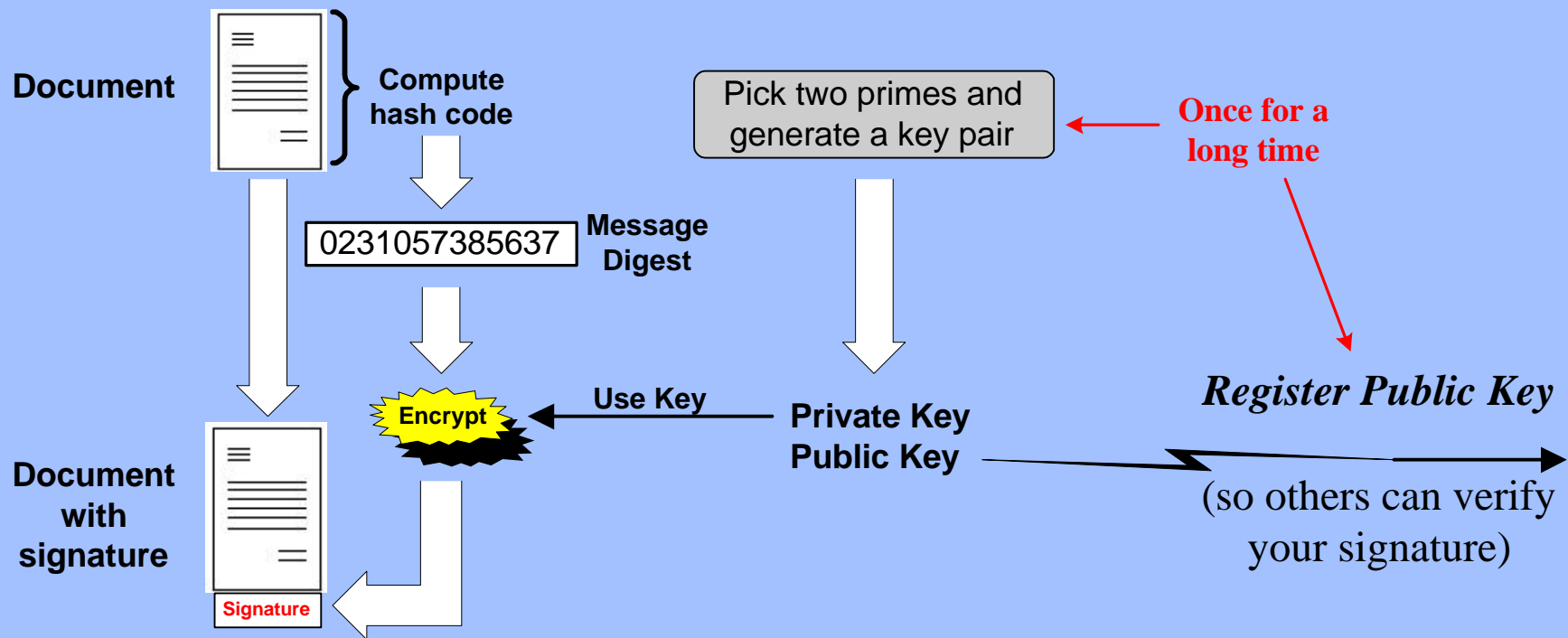
Assume that a sender encrypts the message (including the digital signature) with the receiver's public key, breaks it into multiple packets, and begins transmitting the packets over the Internet. The digital signature is contained in the last packet.

- ◆ Someone in the middle of the Internet can intercept the packets, *encrypt his own message with the intended receiver's public key*, and forward them to the receiver in place of the original sender's packets.
- ◆ When this man in the middle receives the sender's last packet (with the digital signature), he forwards it to the receiver so *the receiver believes that the entire message came from the original sender*.

We protect against this by:

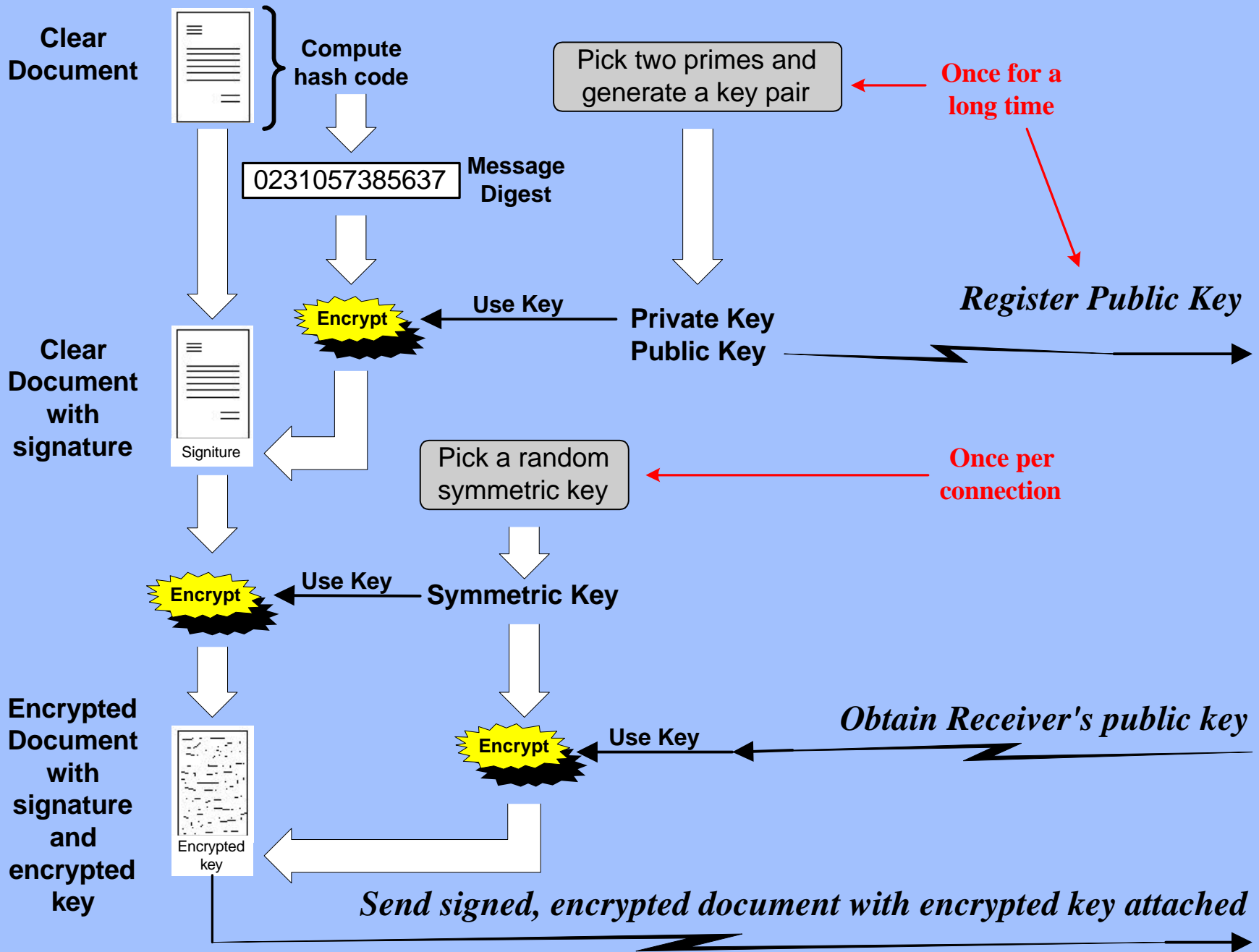
- ◆ Using a *secure hash function* (similar to a CRC) on the original document
 - ◆ to produce a *message digest* of the document
 - ◆ that we *encrypt with the sender's private key*
 - ◆ to produce a *document signature*.
-
- ◆ The document signature is *unique to both the document and the sender*.

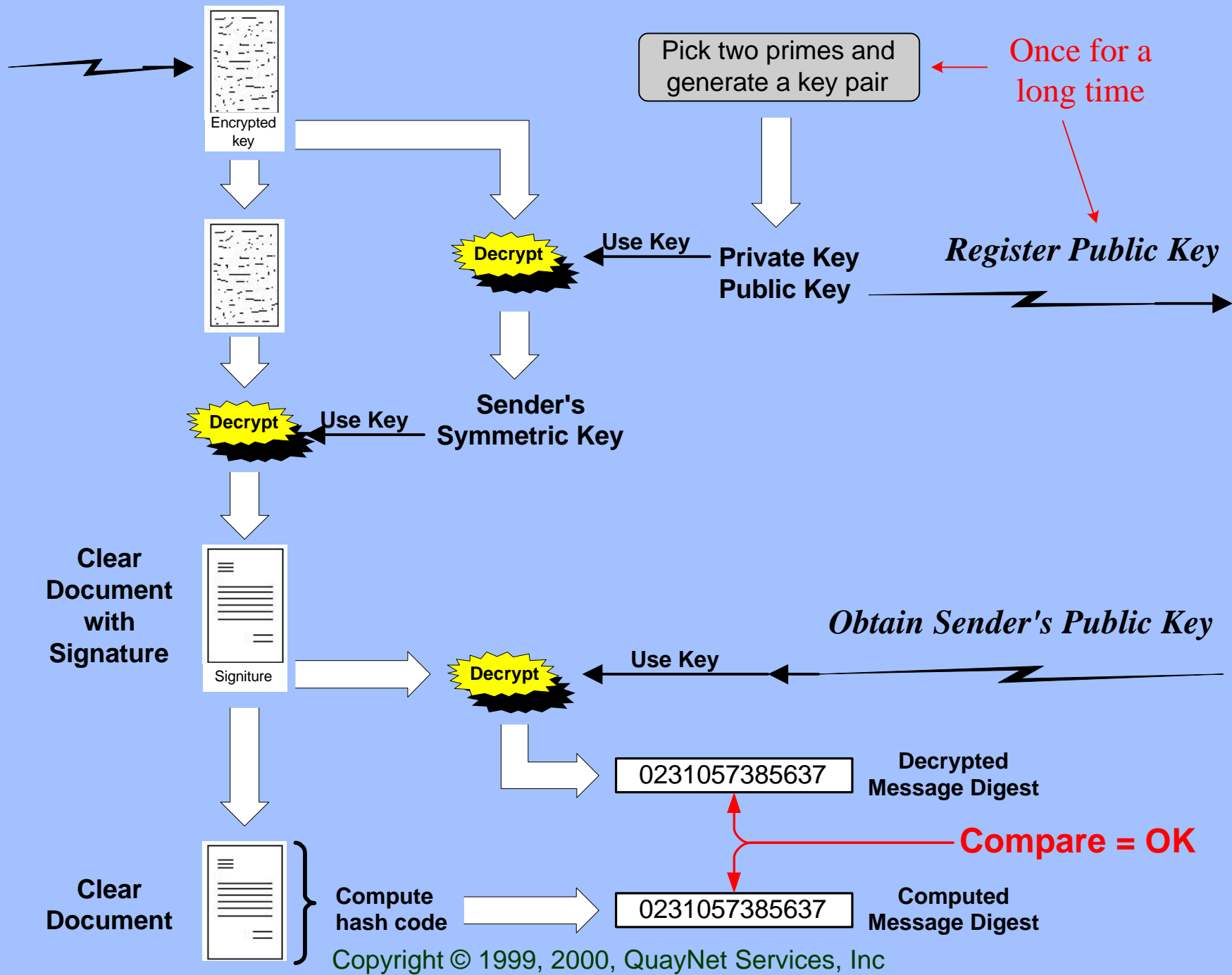
Document Signature



Putting it All Together

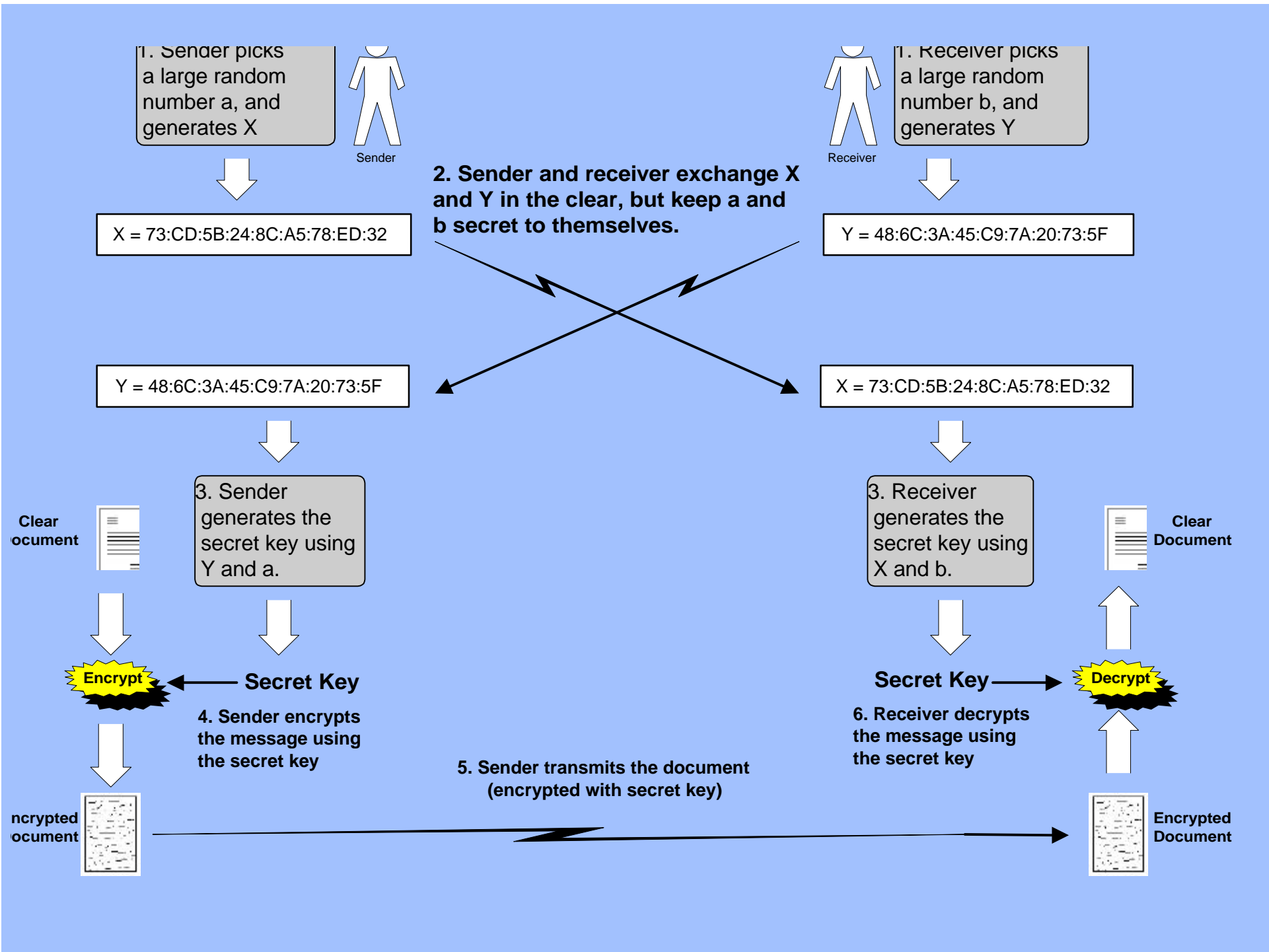
- ◆ *We encrypt a document using a secret-key encryption system.* We generate a new secret key every time we want to send something.
- ◆ *We encrypt the secret key using the receiver's public key,* and send it along with the document.
- ◆ *We generate a message digest* of the document using a secure hash algorithm.
- ◆ *We encrypt the message digest using the sender's private key* to form the document signature.





Diffie-Hellman Key Agreement

- ◆ We can use the Diffie-Hellman algorithm if our only use of Public-key encryption is to *generate a secure secret key* for use with a secret-key encryption system.
- ◆ Two parties can agree on a secret key without sending it over the wire in the clear.



Digital Signature Algorithm

- ◆ We can use the Digital Signature Algorithm (DSA) if our only use of Public-key encryption is to *generate a digital signature*.
- ◆ Based on a variant of the El-Gamal signature algorithm, it cannot be used for encryption.
- ◆ The Digital Signature Standard (DSS) required the use of DSA until December 1998. Now, either DSA or RSA can be used.

Summary of Security Technologies

- ◆ RSA -- Public-key encryption system.
 - Both secret-key agreement and signing.
- ◆ Diffie-Hellman -- Secret-key agreement only.
- ◆ DSA -- Signing only.
- ◆ SHA-1 -- Standard secure hash algorithm.
- ◆ MD2, MD4, MD5 -- Secure hash algorithms.
- ◆ DES, 3-DES, RC2, RC4, IDEA, CAST, etc.
 - Secret-key systems.

Digital Certificates

- ◆ How do we know that someone's public key really belongs to them?
 - Have them *personally give it to you* on a diskette.
 - Obtain the key from a signed *certificate*.
- ◆ A certificate is a *signed digital document* attesting that *the included public key really belongs to an individual* or other entity.
- ◆ The most widely accepted format for certificates is defined by the ITU-T **X.509** standard.

The X.509 Certificate Contains

- ◆ Your public key.
- ◆ Your name.
- ◆ The start date of your public key.
- ◆ The expiration date of your public key.
- ◆ The name of the certificate issuer.
- ◆ A serial number of the certificate.
- ◆ The digital signature of the certificate issuer.



Public Certificate Authorities

- ◆ The only trust we can place in a digital certificate, and thus the public key contained within it, is that derived from the place and method by which we obtain the certificate.
- ◆ A public *Certification Authority* (CA) will register your public key for a fee and make it available to anyone who asks.
- ◆ When they deliver the certificate, they will sign it with their own key to verify that it is authentic.

Public Certificate Authorities

- ◆ VeriSign, Inc.
- ◆ Thawte Consulting cc
- ◆ RSA Data Security, Inc.
- ◆ GTE Corporation
- ◆ AT&T
- ◆ Microsoft Corporation
- ◆ Uptime Commerce Ltd
- ◆ IBM World Registry
- ◆ BelSign NV
- ◆ Etc.

The CA's Public Key?

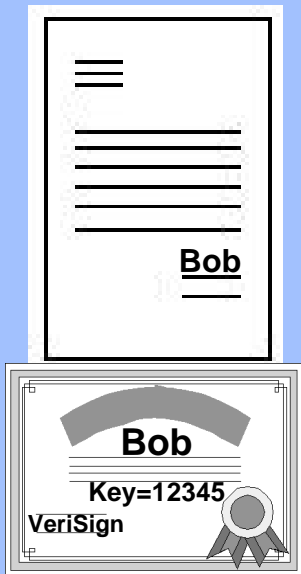
How do you get the CA's public key, so you can trust its signature, so you can trust the public key in certificates signed by the CA, so you can trust someone's signature, etc.?

- ◆ The CA's certificate comes *embedded in your browser*, web server, email program, etc. You must trust Microsoft or Netscape!!!

Certificate Hierarchy



Obtain Root
Cert with OS



Transmit Cert with Document

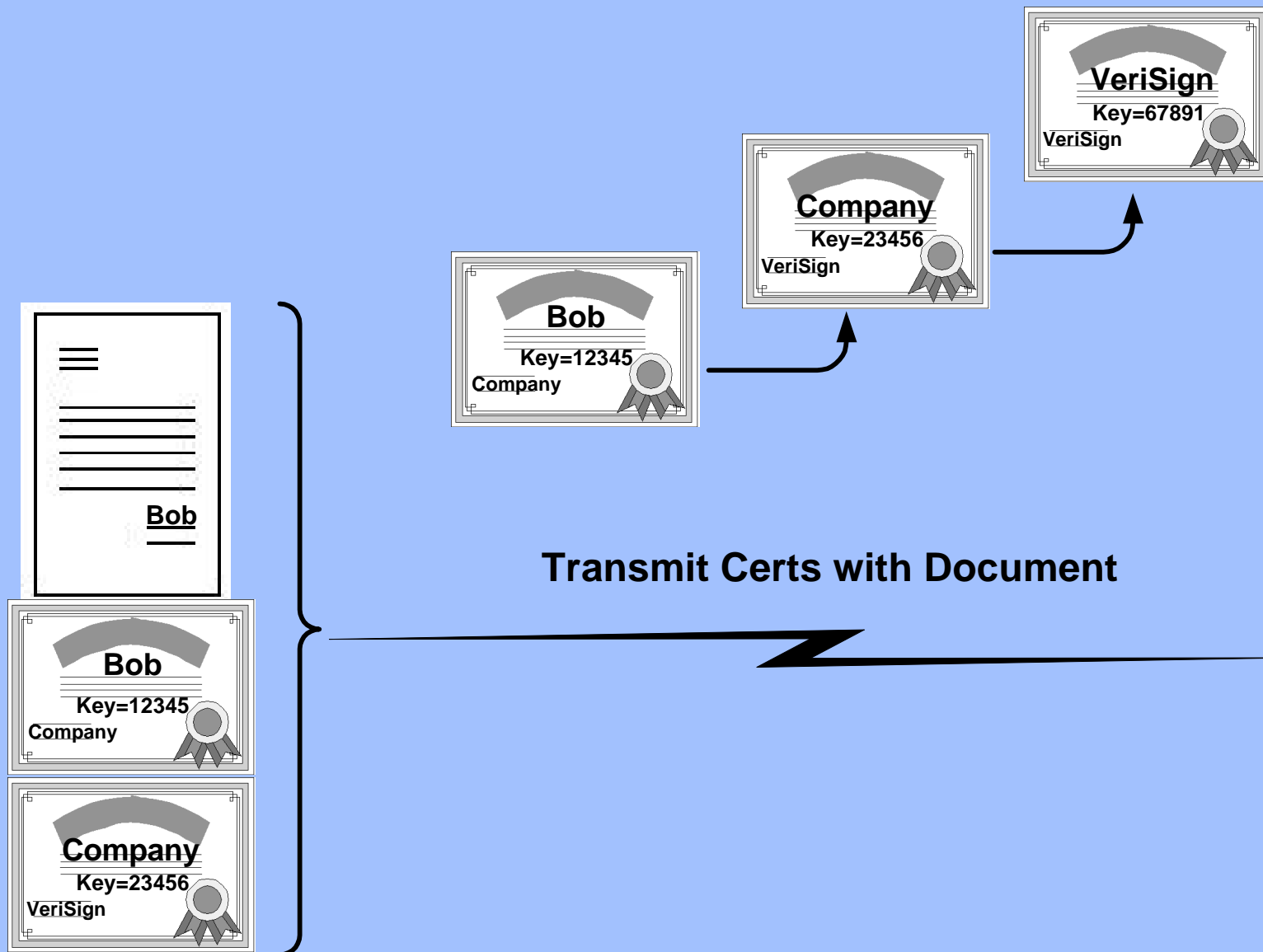
1. Verify message integrity and authentic (from Bob) by using the public key in Bob's cert that Bob sent.

2. Verify Bob's cert integrity and authentic (from VeriSign) by using the public key in VeriSign's cert that was obtained with the browser/email/etc. program.

A Company-issued Certificate

- ◆ A private company can issue certificates to its employees as well. They are signed by the “company” and serve to authenticate employee email and other transfers.
- ◆ The company gets a certificate signed by a public CA to validate its public key.
- ◆ If email is sent outside the company, the receiver probably does not have a company certificate, and so we send along the company certificate signed by a public CA.

Sending Outside the Company



Now You Know

- ◆ Secret-key encryption
- ◆ Public-key encryption
- ◆ Document signatures
- ◆ Digital certificates

The End

Dr. Garrison Q. Kenney

NCSU Computer Training Unit

QuayNet Services, Inc.

kenney@quaynet.com

www.quaynet.com