

IT 4823

Information Security Administration

Linux and Windows Security
Wireless Security
Review of Public Key Cryptography



Notice: This session is
being recorded.

Linux Security

- Linux has evolved into one of the most popular and versatile operating systems
- Many features mean broad attack surface
- You can create highly secure Linux systems
- We will review:
 - Discretionary Access Controls
 - Typical vulnerabilities and exploits in Linux
 - Best practices for mitigating those threats
 - Improvements to Linux security model

Linux Security Model

- Linux's traditional security model is:
 - people or processes with “root” privileges can do anything
 - other accounts can do much less
- Hence, attackers want to get root privileges
- One can run robust, secure Linux systems
- **Discretionary Access Controls (DAC)**

SetUID and SetGID

- setuid bit means program “runs as” owner
 - no matter who executes it
- setgid bit means run as a member of the group which owns it
 - again regardless of who executes it
- “run as” = “run with same privileges as”
- *Are very dangerous* if set on a file owned by root or other privileged account or group

setuid root Vulnerabilities

- A **setuid root** program runs as root *no matter who executes it*
- Used to provide unprivileged users with access to privileged resources
- Must be very carefully programmed because errors could be exploited.
- Distributions now minimize setuid-root programs
- System attackers still scan for them!

Kernel vs. User Space

- Kernel space
 - refers to memory used by the Linux kernel and its loadable modules (*e.g.*, device drivers)
- User space
 - refers to memory used by all other processes
- Since the kernel enforces Linux DAC and security, it is critical to isolate kernel from user
 - so kernel space is never swapped to disk
 - only root may load and unload kernel modules

Web Vulnerabilities

- A very broad category of vulnerabilities with big and visible attack surfaces
- When written in scripting languages
 - not as prone to classic buffer overflows
 - can suffer from poor input-handling
- Few “enabled-by-default” web applications
- But users install vulnerable web applications
- Or write custom web applications having easily-identified and easily-exploited flaws

Rootkits

- Allow attackers to cover their tracks
- If successfully installed before detection, all is very nearly lost
- Originally, collections of hacked commands for hiding attacker's files, directories, processes
- Now rootkits use loadable kernel modules
 - intercepting system calls in kernel-space
 - hiding attacker from standard commands
- May be detectable with chkrootkit
- Generally have to wipe and rebuild system

Logging

- Effective logging is a key resource
- Linux logs using syslogd or Syslog-NG
 - receive log data from a variety of sources
 - sorts by **facility** (category) and **severity**
 - writes log messages to local/remote log files
- Syslog-NG preferable because it has:
 - variety of log-data sources / destinations
 - much more flexible “rules engine” to configure
 - can log via TCP which can be encrypted
- One should check and customize defaults

Application Security

- Many security features are implemented in similar ways across different applications
- Approaches:
 - running as unprivileged user/group
 - running in chroot jail
 - modularity
 - encryption
 - logging

Running As Unprivileged User/Group

- Every process “runs as” some user
- Extremely important that this user is not root since any bug can compromise entire system
- May need root privileges, *e.g.* to bind a port
 - have root parent perform privileged function
 - but main service from unprivileged child
- User/group used should be dedicated, making it easier to identify source of log messages

Mandatory Access Controls

- Linux uses a DAC security model
- But Mandatory Access Controls (MAC) impose a global security policy on all users
 - users may not set controls weaker than policy
 - normal admin done with accounts without authority to change the global security policy
 - but MAC systems have been hard to manage
- RedHat Enterprise Linux has SELinux
- Use “pure” SELinux for high-sensitivity, high-security

SELinux

- NSA's powerful implementation of mandatory access controls for Linux
- Linux DACs still applies, but if it allows the action, SELinux then evaluates it against its own security policies
- Subjects are processes (run user commands)
- Actions are permissions
- Objects are not just files and directories
- To manage complexity SELinux has:
 - “that which is not expressly permitted, is denied” (default deny)
 - groups of subjects, permissions, and objects

Windows Security Architecture

- Active Directory (AD)
 - Microsoft's LDAP directory
 - Kerberos-type security
 - All Windows clients can use AD to perform security operations including account logon
 - Authentication uses AD when the user logs on using a domain rather than local account
 - User's credential information is sent securely across the network to be verified by AD
- WinLogon (local) and NetLogon (net) handle login requests

Local vs. Domain Accounts

A networked Windows computer can be:

- Domain joined
 - can login with either domain or local accounts
 - ~~• if local may not access domain resources~~
 - centrally managed and much more secure
- In a workgroup
 - a collection of computers connected together
 - only local accounts in Security Accounts Manager (SAM) can be used
 - no infrastructure to support AD domain

Windows Privileges

- Are system wide permissions assigned to user accounts
 - *e.g.* backup computer, or change system time
- Some are deemed “dangerous” such as:
 - *act as part of operating system* privilege
 - *debug programs* privilege
 - *backup files and directories* privilege
- Others are deemed “benign” such as
 - *bypass traverse checking* privilege

Windows Privileges

- Are system wide permissions assigned to user accounts
 - *e.g.* backup computer, or change system time
- Some are deemed “dangerous” such as:
 - *act as part of operating system* privilege
 - *debug programs* privilege
 - *backup files and directories* privilege
- Others are deemed “benign” such as
 - *bypass traverse checking* privilege

Access Control Lists

Two forms of access control list (ACL):

- Discretionary ACL (DACL)
 - grants or denies access to protected resources such as files, shared memory, named pipes etc
- System ACL (SACL)
 - used for auditing
 - Beginning with Windows Vista, to enforce mandatory integrity policy

Access Control Lists

- Objects needing protection are assigned a DACL (and possible SACL) that includes
 - SID of the object owner
 - list of access control entries (ACEs)
- Each ACE includes a SID and access mask
- Access mask could include ability to:
 - read, write, create, delete, modify, etc
- Access masks are object-type specific
e.g. service abilities are create, enumerate

Security Descriptor (SD)

- Data structure with object owner, DACL, & SACL, *e.g.*
Owner: CORP\Blake
ACE[0]: Allow CORP\Paige Full Control
ACE[1]: Allow Administrators Full Control
ACE[2]: Allow CORP\Cheryl Read, Write and Delete
- There is no implied access, if there is no ACE for requesting user, then access is denied
- Applications must request correct type of access:
if just request “all access” when need less (*e.g.* read)
some users who should have access will be denied

Mandatory Access Control

- Windows Vista, 7, 8 have “Integrity Control” that limits operations changing an object’s state
- Objects and principals are labeled (using SID) as:
 - Low integrity (S-1-16-4096)
 - Medium integrity (S-1-16-8192)
 - High integrity (S-1-16-12288)
 - System integrity (S-1-16-16384)
- When a write operation occurs, first check whether subject’s integrity level dominates object’s integrity level
- Much of O/S marked medium or higher integrity

Windows Vulnerabilities

- Windows, like all O/S's, has security errors
 - and errors have been exploited by attackers to compromise customer operating systems
- Microsoft now uses a process called the Security Development Lifecycle
- Net effect: approx 50% reduction in errors
- Windows Vista used SDL start to finish
- IIS v6 (in Windows Server 2003) had only 3 vulnerabilities in 4 years, none critical

Windows Security Defenses

- Attackers are now criminals rather than poorly-socialized pre-adolescents, and are highly motivated by money
- Windows has categories of security defenses:
 - account defenses
 - network defenses
 - buffer overrun defenses.
 - browser defenses

Wireless Security Overview

- Concerns for wireless security are similar to those found in a wired environment
- Security requirements are the same:
 - confidentiality, integrity, availability, authenticity, accountability
- Most significant source of risk is the underlying communications medium

Wireless Network Threats

- Accidental association
- Malicious association
- Ad hoc networks
- Nontraditional networks
- Identity theft (MAC spoofing)
- Man-in-the middle attacks
- Denial of service (DoS)
- Network injection

Securing Wireless Transmissions

- Principal threats are eavesdropping, altering or inserting messages, and disruption.
- Countermeasures for eavesdropping:
 - signal-hiding techniques
 - **encryption**
- The use of encryption and authentication protocols is the standard method of countering attempts to alter or insert transmissions

Securing Wireless Networks

- The main threat involving wireless access points is unauthorized access to the network
- Principal approach for preventing such access is the IEEE 802.1X standard for port-based network access control. The standard provides an authentication mechanism for devices wishing to attach to a LAN or wireless network
- Use of 802.1X can prevent rogue access points and other unauthorized devices from becoming insecure backdoors

Wireless LAN Security Algorithms: WEP

Wired Equivalent Privacy (WEP) algorithm

- 802.11 privacy
- **Trivial to crack**
- There is no circumstance in which WEP is safe.

Wireless LAN Security Algorithms WPA

Wi-Fi Protected Access (WPA)

- Security mechanisms that eliminate most WEP security issues.
- Based on the current state of the 802.11i standard at the time.

Wireless LAN Security Algorithms: WPA2

- Robust Security Network (RSN) a.k.a WPA2
 - Final form of the 802.11i standard
 - Has shared key and enterprise flavors
- Wi-Fi Alliance certifies vendors in compliance with the full 802.11i specification under the WPA2 program.
- Modern wireless networks should be using one of the WPA2 flavors.

Disable WPS Support

- It's "WiFi Protected Setup"
- But it takes only 11,000 guesses to brute-force the key exchange PIN.
- The keys that secure the PIN are often predictable.
- Some manufacturers computed the PIN from the MAC address.
- Annnnd, it's printed on a sticker on the machine!

Public Key Cryptography

- Also known as “asymmetric key cryptography.”
- Encrypt with one key, publicly available, the **public key**.
- Decrypt with a different key, known only to the recipient, the **private key**.
- Solves the key exchange problem because there is no need to exchange keys.

Computationally Secure? Maybe.

- A fast method for finding the prime factors of large numbers might be developed. (It is only “thought” to be hard.)
- An agency like the NSA may already have done so in secret.
- When quantum computing is sufficiently developed, it will be possible to use Shor’s algorithm to find prime factors of large numbers.

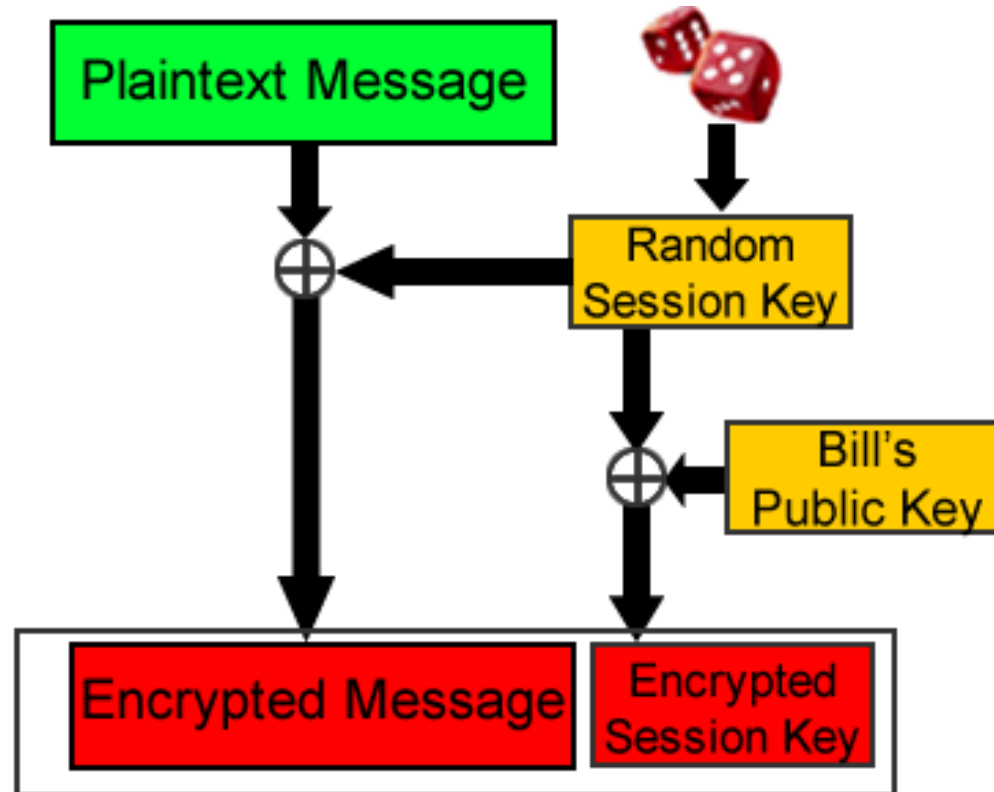
Computational Effort

- Public key encryption can be 10,000 times more computational work than secret key encryption.
- Why? We are dealing with difficult arithmetic (exponentiation) on very large (hundreds of digits) numbers.

Public Keys with Less Computation

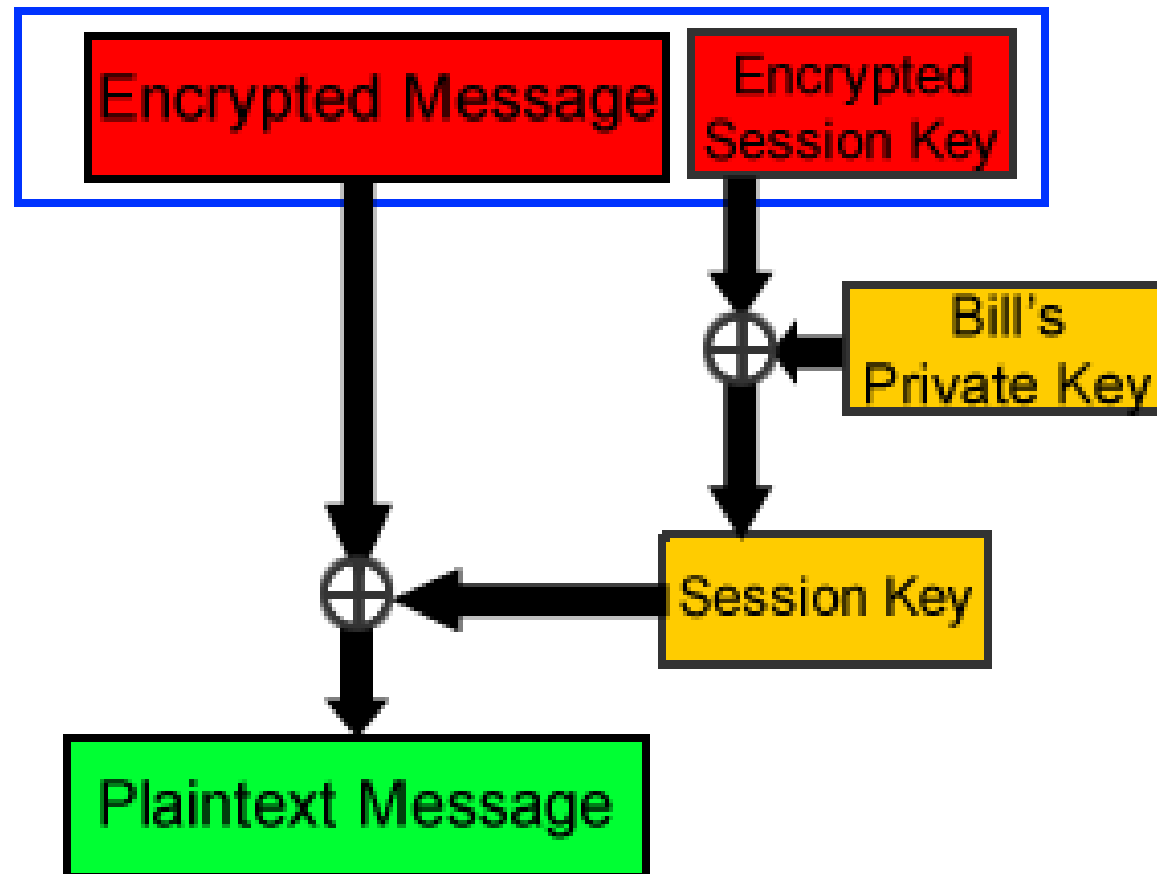
- Public key encryption is time-consuming... up to 10,000 times more work than secret key encryption.
- Solution: generate a random, one-time secret key, the *session key*.
- Encrypt the message with the secret key.
- Encrypt the secret key with the recipient's public key.
- Send encrypted message and encrypted key.

Encryption with Session Key



The session key is a symmetric key; computation for encryption is (relatively) easy.

Decrypting a Hybrid Message



Digital Signatures

- Is there a way to use cryptography to verify origin integrity? (*Hint: Yes.*)
- The public and private keys of a key pair are cryptographic inverses of each other.
 - A message encrypted with the public key can only be decrypted with the corresponding private key.
 - *But*, a message encrypted with the private key can be decrypted using the corresponding *public* key!

Public Key Digital Signature

Public key cryptography works both ways; the keys are cryptographic inverses of one another!

Bob encrypts a message with his private key

Anyone can decrypt it using Bob's public key.

Only Bob could have encrypted it, so, it is “signed” by Bob. *Bob Brown*

Uhhh, but *anyone* can decrypt it; there is *no confidentiality*. (This is exactly analogous to a handwritten signature.)

Cryptographic Hash Functions

- **Idea:** Detect tampering with messages.
- Needed: A function that “characterizes” a message in some way. That’s a cryptographic hash function.
- Characteristics of a strong hash function:
 - $h(m)$ is easy to compute
 - Deriving m from $h(m)$ is infeasible
 - Given m , finding m' such that $h(m') = h(m)$ is infeasible (protects integrity by resisting collision attacks)

Hashing vs. Encryption

- Hash:
 - Not invertible (reversible)
 - Variable length input, fixed length output.
- Encryption:
 - Reversible; you can decrypt encrypted text.
 - One to one correspondence between plain text and ciphertext.
 - Large plain text will generate large ciphertext.

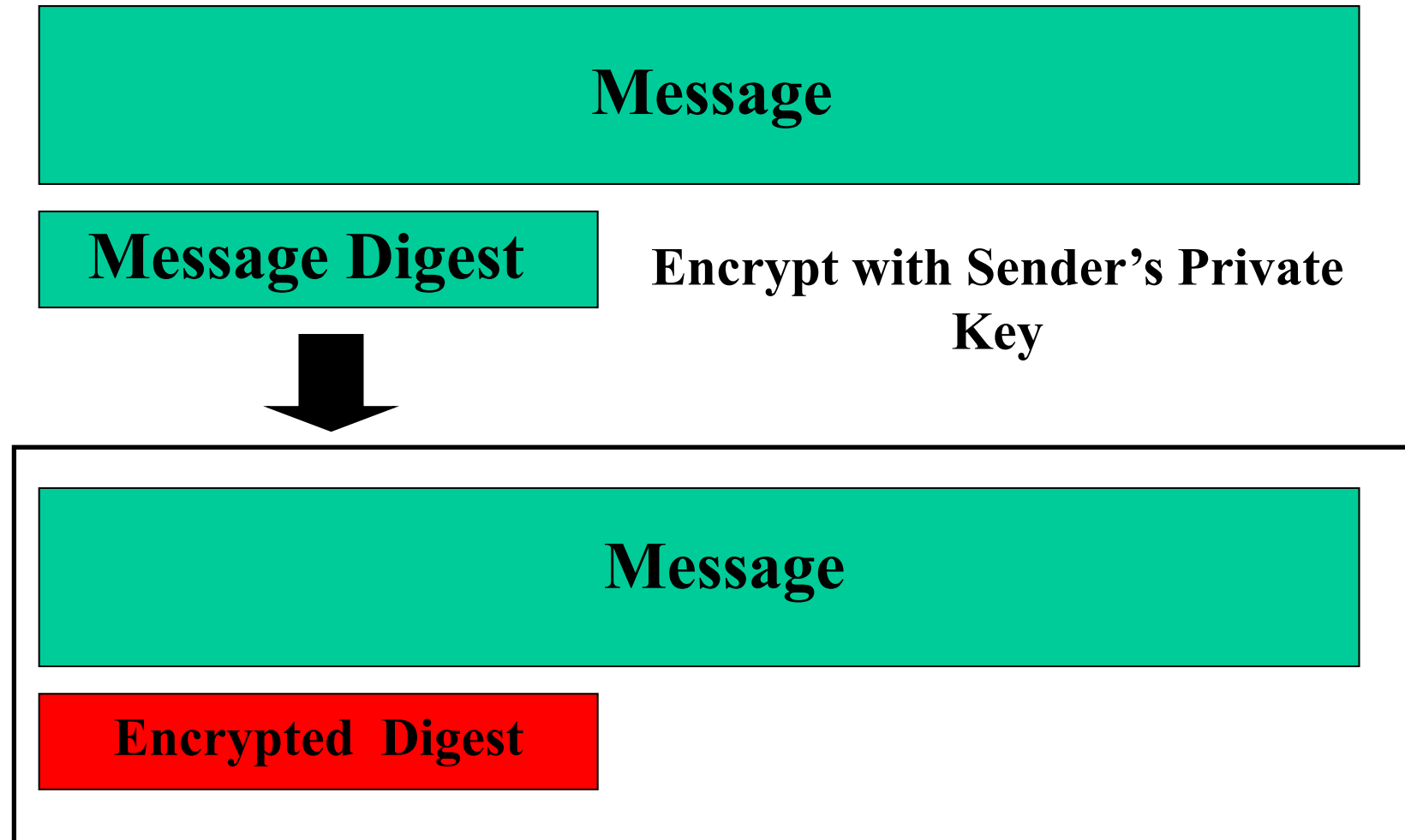
Digital Signature Improved

Bob computes a “digest” of a message using a hash function; this is smaller than the message and of fixed size, but has the property that changing the message changes the computed digest.

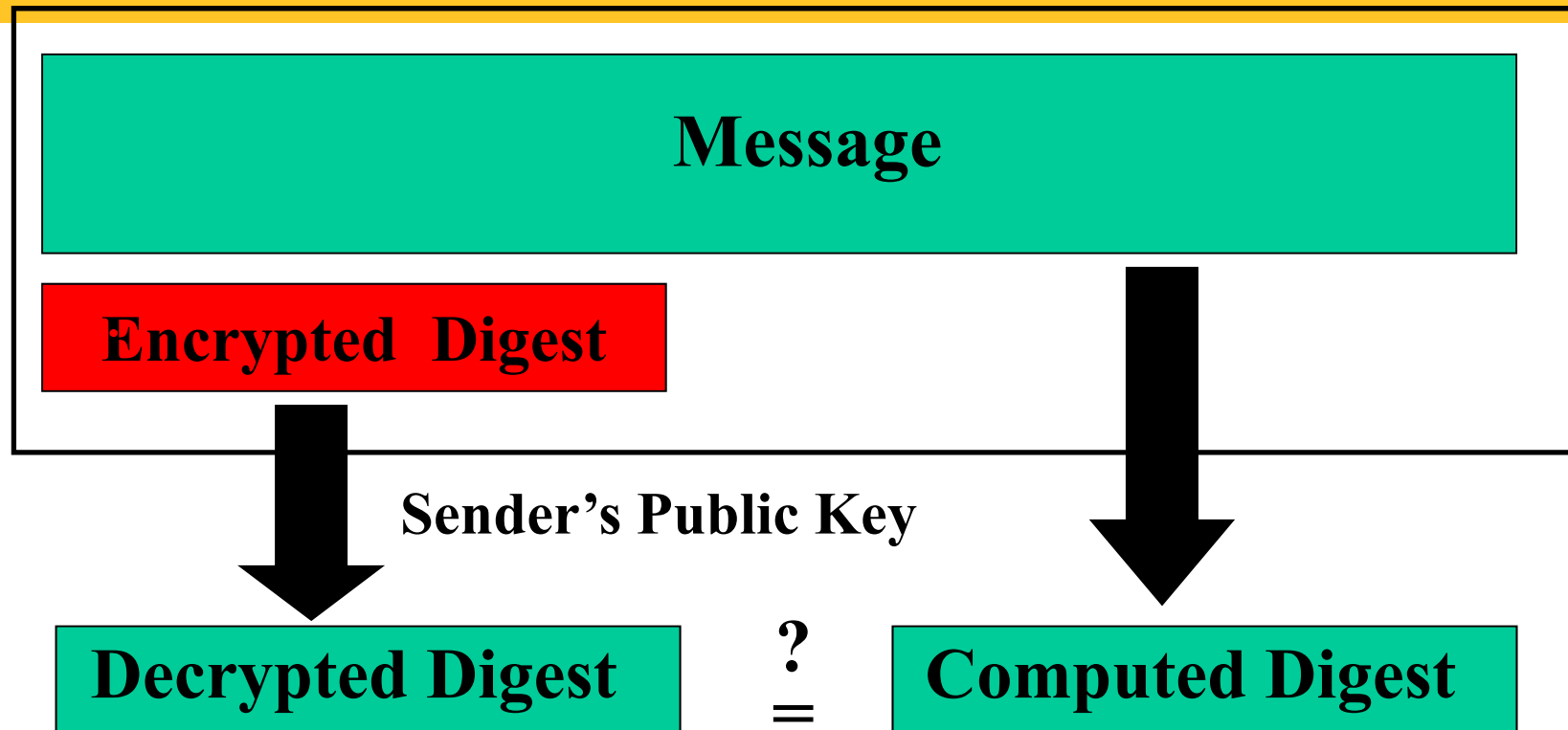
Bob encrypts the digest with his private key
Anyone can decrypt the digest using Bob’s public key, but *only Bob* could have encrypted it.
The message is digitally signed.

If the decrypted digest matches one computed from the message, the message has not been altered.

Digital Signature with Digest



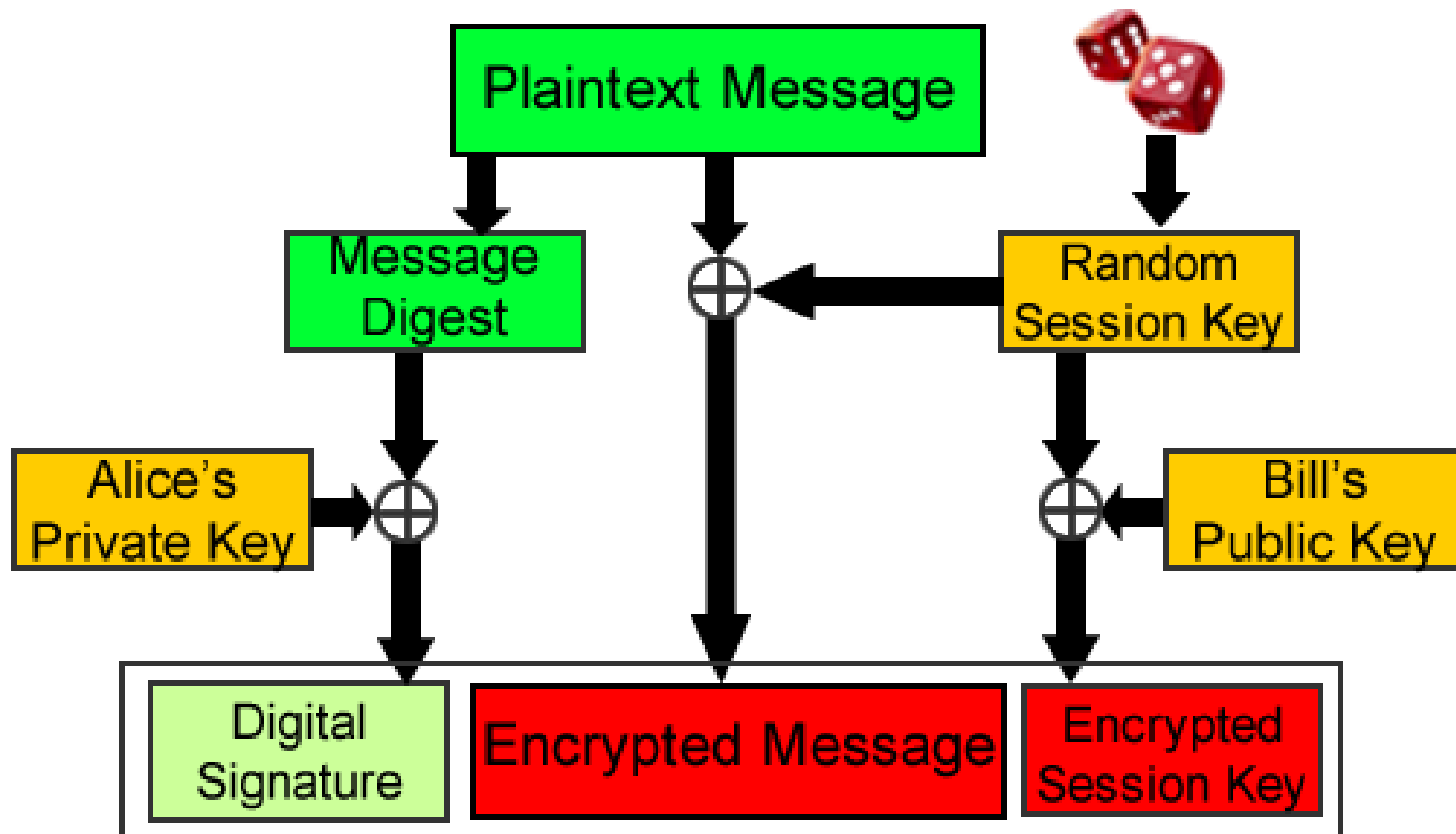
Validating the Digital Signature



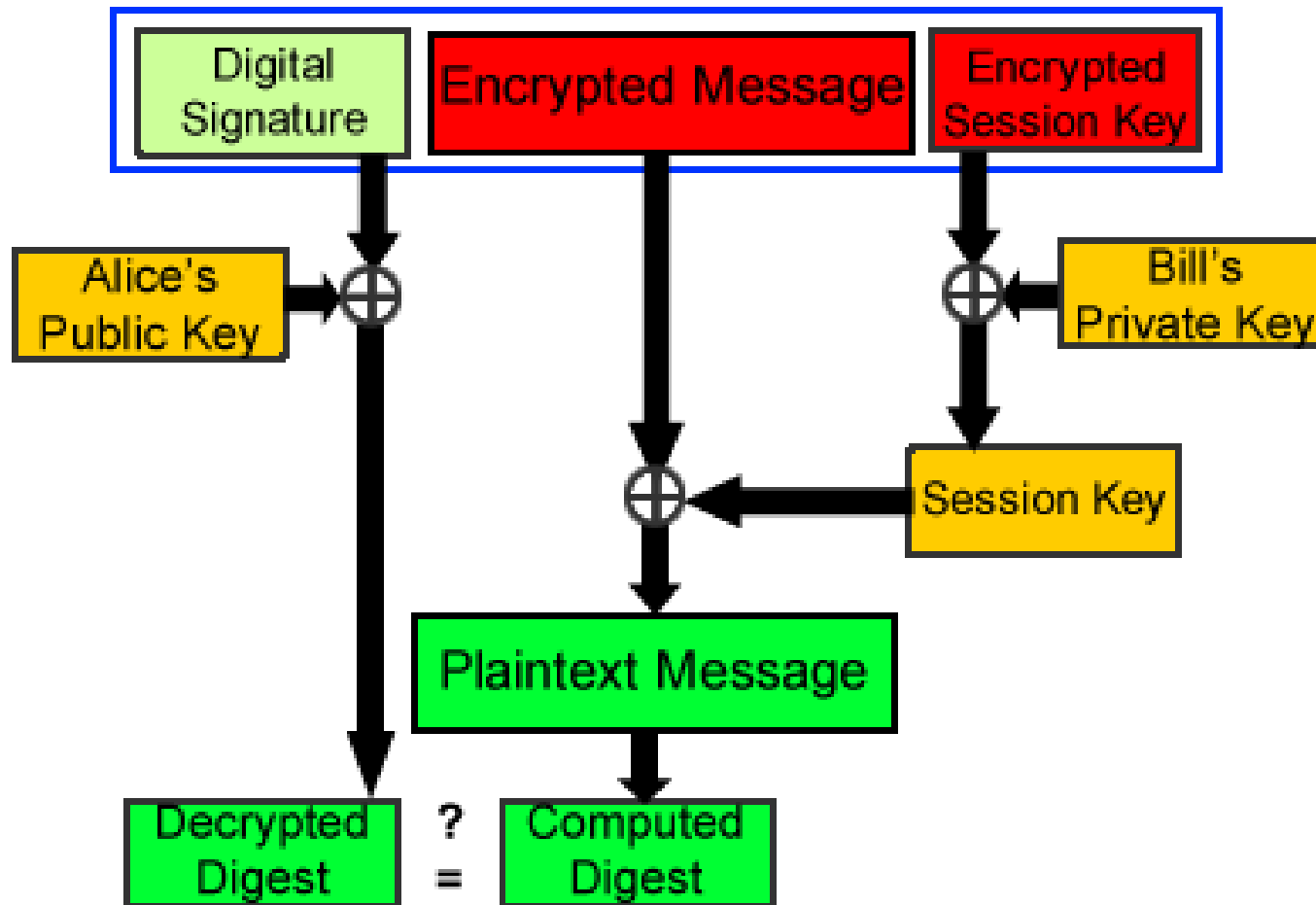
If the digests are equal, the sender is assured and no tampering has occurred.

Signature with Confidentiality

Preparing a message from Alice to Bill.



Decrypting the Message



Multiple Recipients

- This scheme can be adapted to allow a single message to be sent to multiple recipients.
- The session key must be encrypted separately, one time for each recipient, and identify the recipient.
- Each recipient uses his own private key to decrypt a copy of the session key.
- Only one copy of the message and message digest are needed.

When to Use Encryption

- Encryption is not a magic sauce.
- “Encryption is most useful when OS protections cannot work.” –*Steve Bellovin*.
 - Disk drives when attacker has physical access, backup media, etc.
 - “Cloud” storage
 - Data being transmitted over a network.
- Used inappropriately, encryption provides little protection and causes operational headaches.

What Kind of Encryption

- Symmetric: When the same subject that did the encryption will later decrypt and use the data. (No need for key exchange.)
- Asymmetric or hybrid: When two or more subjects are involved. (Use public key crypto for encryption or key exchange.)
- Cryptographic hash:
 - To check integrity of data or message
 - When stored value must be protected and compared for equality with challenge value, as with passwords.

Certificates

- Goal: Bind an identity to a public key.
- Create token (message) containing
 - Identity of principal (here, George)
 - Corresponding public key
 - Type of hash used
 - Other information (*e.g.* identity of signer)

Digitally signed by trusted party

$$C_s = \text{George} \parallel k_{GB} \parallel T \parallel \text{Signer} \parallel \{\text{Digest}\} k_{CV}$$

The Digital Certificate

Contains:

- George's identity
- George's public key
- Identity of signer and other info
- Digital Signature

Everything is plain text except the digital signature, which is a cryptographic hash (digest) encrypted by the *signer's private* key.

George's Identity
George@example.com

George's Public Key
mQGiBD9aAvwRB

Hash, other information
SHA-256

Signer's Identity
Verisign

Digital Signature
B157 ACE3 9788

Changing the Public Key

- Evil Eve substitutes her public key for George's, a man-in-the-middle attack.
- The digital signature is no longer valid (because the data have changed.)
- Evil Eve must gain access to the signer's private key (signing key) to forge a new digital signature.
We hope this is hard!

George's Identity George@example.com
Eve's Public Key DfIZOkhURN
Hash, other information SHA-256
Signer's Identity Verisign
Digital Signature B157 ACE3 9788

Validating a Digital Certificate

- Compute a cryptographic hash over all fields of the certificate except the digital signature.
- Decrypt the digital signature with the signer's public key.
- Compare the digital signature – it's also a cryptographic hash – with the computed hash.
- If equal, the certificate is valid.
- Check the Certificate Revocation List (sometimes omitted!)

Parties to a Digital Certificate

- Owner
 - Prepares an unsigned digital certificate with the **owner's public** key.
 - Submits it to the signer to be signed.
 - Uses it in place of the “bare” public key.
- Signer
 - Verifies owner
 - Signs certificate with the **signer's private** key
- User
 - Validates certificate with the **signer's public** key.
 - Uses the **owner's public** key with confidence.

Trusting Digital Certificates

We must trust that:

- The CA has correctly identified the principal.
- The CA's private key has not been compromised:
 - Theft of the key
 - Subversion of the CA's mechanisms
- This does not scale well... the more CAs there are the more opportunities for sub-version

Questions

