# Security

Module 22

# Security and the OS

Security means many things - we need to narrow our study a bit

| Goal | Threat |
| --- | --- |
| Confidentiality | Exposure of data |
| Integrity | Tampering with data |
| Availability | Denial of service |

# Security and the OS

Security means many things - we need to narrow our study a bit

| Goal | Threat |
| --- | --- |
| Confidentiality | Exposure of data |
| Integrity | Tampering with data |
| Availability | Denial of service |

Securing resources through **credentials**

How operating systems store passwords

How attackers could impersonate highly privileged users through buffer overflow

# Passwords

- Password are never stored as plain text:
  - A user's plain text password is stored as a crytographic hash

    `“super_password” => 8F420912D32AB23C`
  - It's trivial to go from plain text to “cypher-text”, impossible (or really hard) to go the otherway.

# Login

- On login, user enters username (plain text)
  - Username is looked up in plain text file
  - Username give OS the cypher-text password:
    - Username: sfrees
    - Password: `8F420912D32AB23C`
  - Now the user types in their password
  - OS hashes “super_password” to `8F420912D32AB23C`
  - Since they match, login is correct.

# Login security

- The easiest way to break in is to simply “guess” a password.  Choose good passwords.
- Password length and entropy is a better indicator than “special characters”
- Often attackers are searching for ***any*** user, not you in particular.  Don’t be the first one cracked.

# Cracking password

- **Complete brute-force:** Try every password a computer can generate
  - Easily thwarted by delayed login response and tripwires
- But what if you have the list of hashed passwords?
  - This is actually not as hard as you think - in UNIX it used to be completely public!
  - Pre-hash dictionary, see if you see any matches!

# Thwarting a dictionary attack

- Hashing algorithm is always known, it's relatively trivial to compute hashes of "common" passwords, and look for matches.
- Salt: Assign a randomly generated string for each user (stored in plain text, with username)
  - Instead of hashing password, hash password+salt.
  - Renders dictionary attacks computationally infeasible

| |
|---|
| Bobbie, 4238, e(Dog, 4238) |
| Tony, 2918, e(6%%TaeFF, 2918) |
| Laura, 6902, e(Shakespeare, 6902) |
| Mark, 1694, e(XaB#Bwcz, 1694) |
| Deborah, 1092, e(LordByron,1092) |

# Why steal usernames/password

- Generally, damage is limited to individual
  - Steal data from them, potentially financially or reputationally damaging
  - However, sometimes the target is not data, but **control**.
- Attackers want their code to execute in **kernel mode**.
  - Can be used to get to other, bigger, more enticing data… or to control devices, etc.

# Buffer Overflows

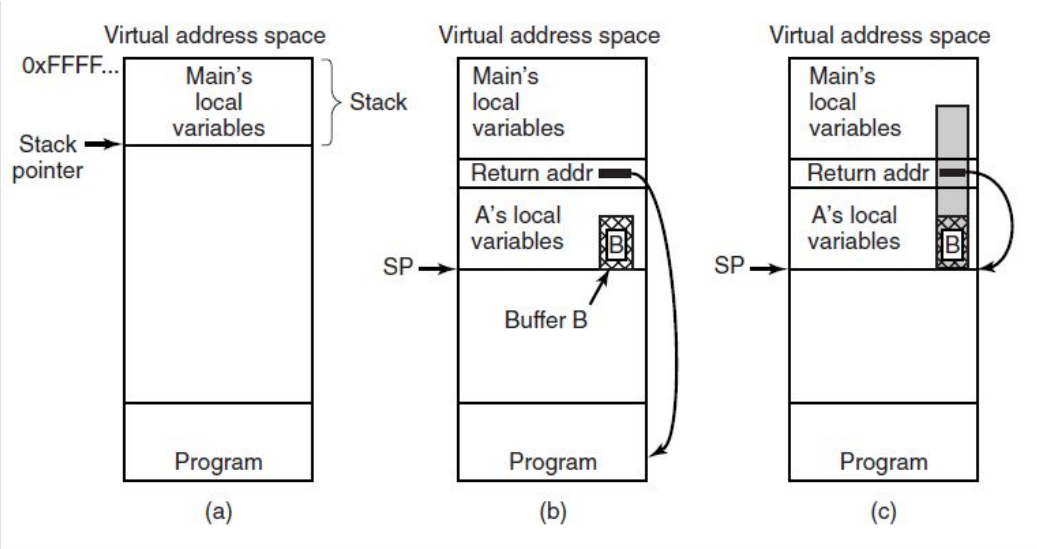



Figure 9-21. (a) Situation when the main program is running. (b) After the procedure A has been called. (c) Buffer overflow shown in gray.

If the **buffer** the user types is code, and the return address is overwritten to the location of the buffer… mischief ensues

# Security and the OS

Security means many things - we need to narrow our study a bit

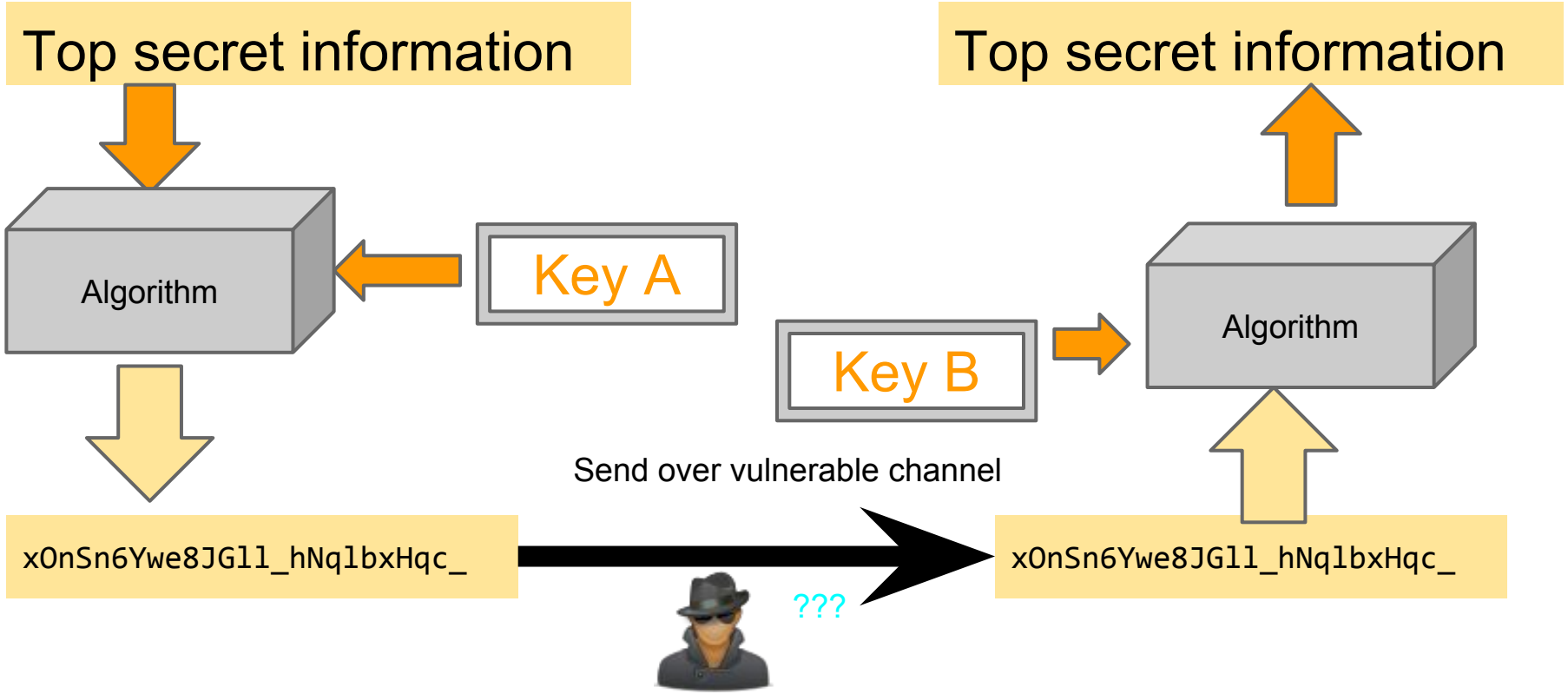
| Goal | Threat |
| --- | --- |
| Confidentiality | Exposure of data |
| Integrity | Tampering with data |
| Availability | Denial of service |

Digital Signatures

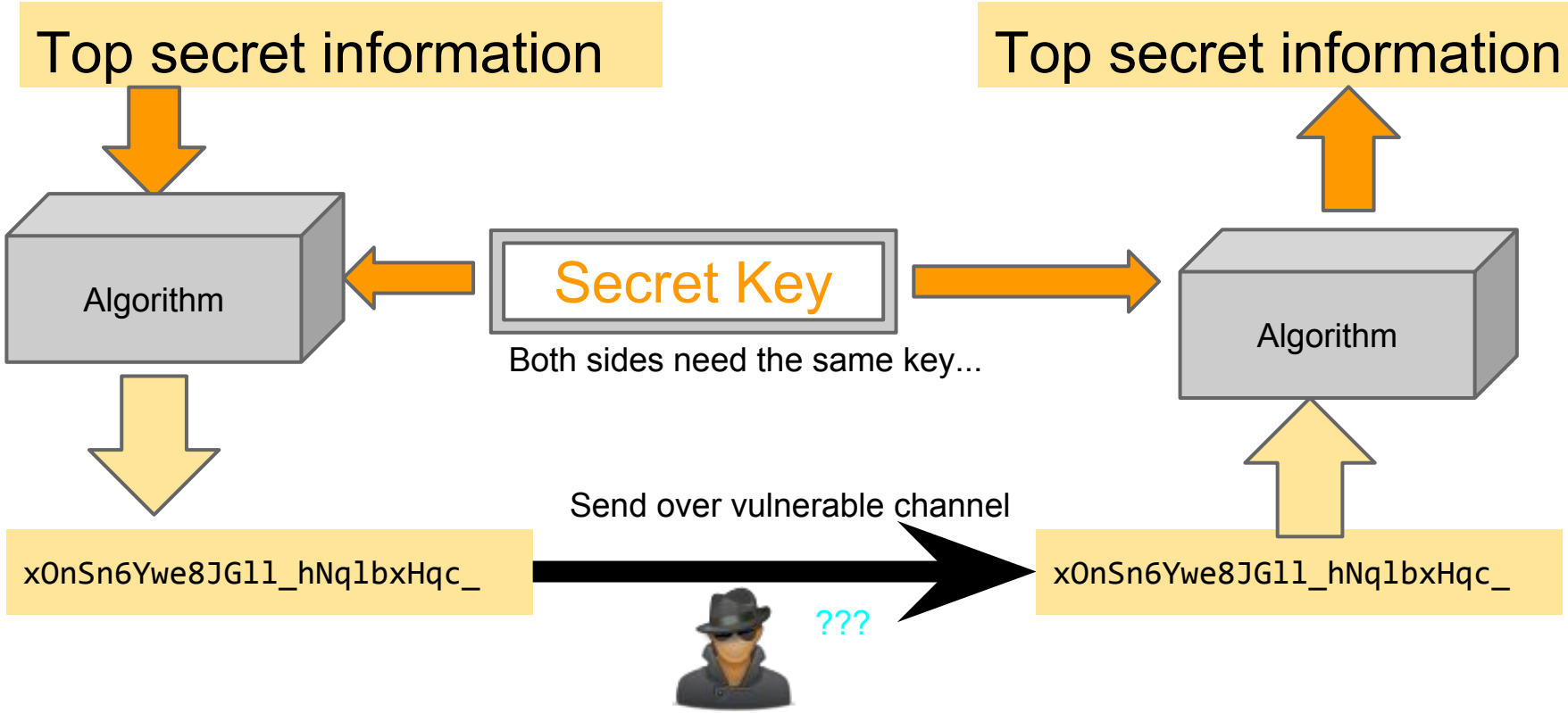
# We've talked about ECC...

- ECC verifies network packets and disk blocks are being read as they were written
- But how do we know they were written "honestly" - by the person we think?
- Digital Signatures can tie a message to a specific machine - it centers around **encryption**

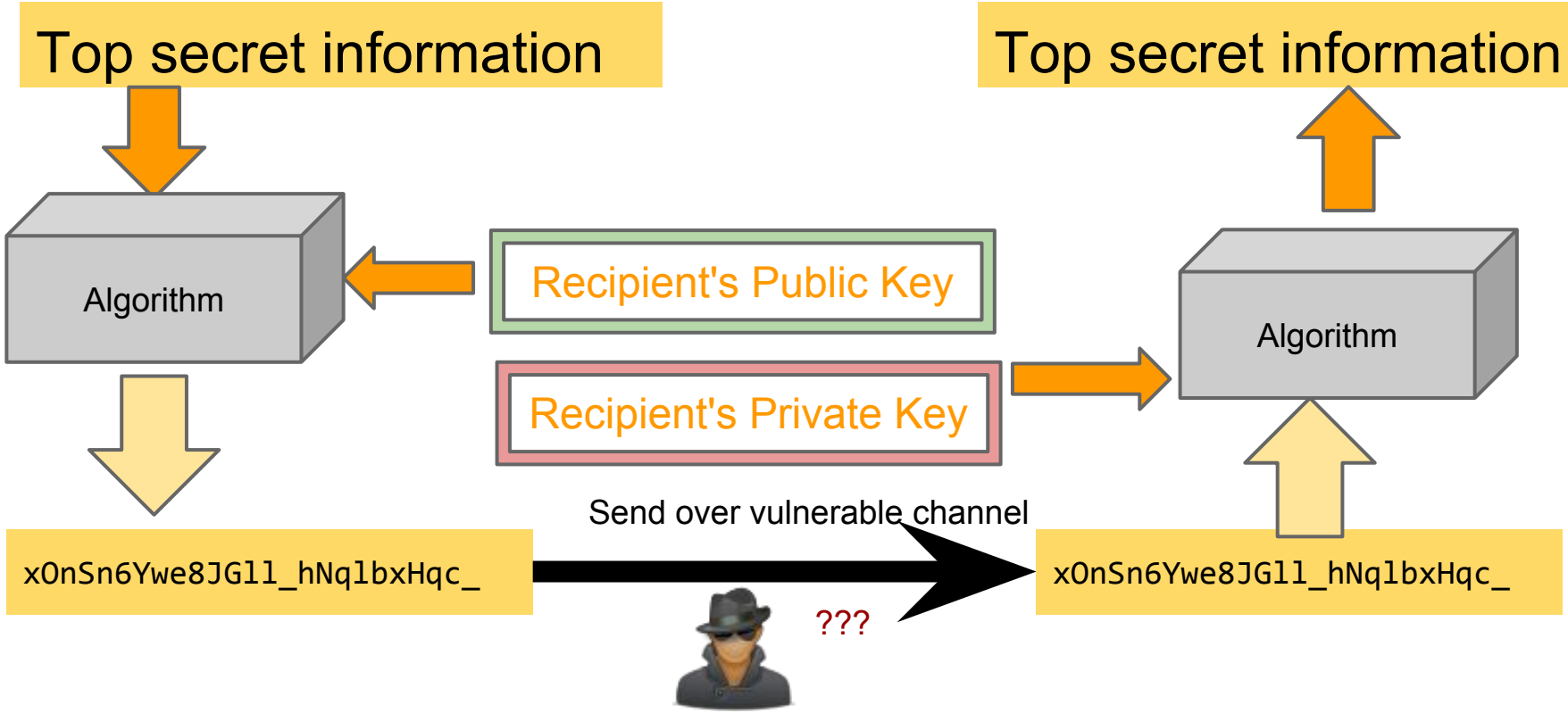
# Encryption - the general idea



# Secret-Key Encryption



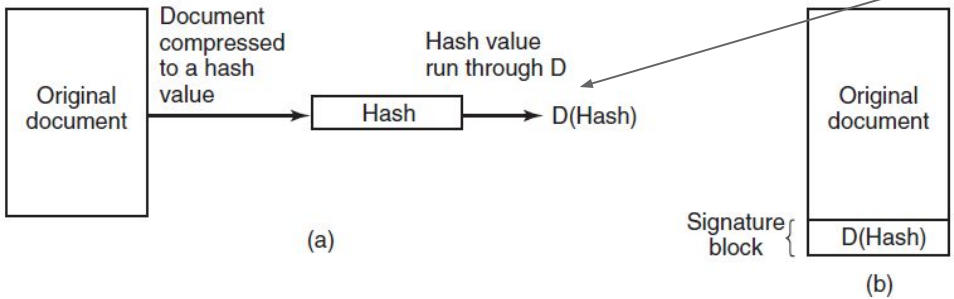
# Public/Private Key Encryption



# Digital Signatures

Actor X is sending to Actor Y

Y want's to be able to ensure message is *actually* from X.



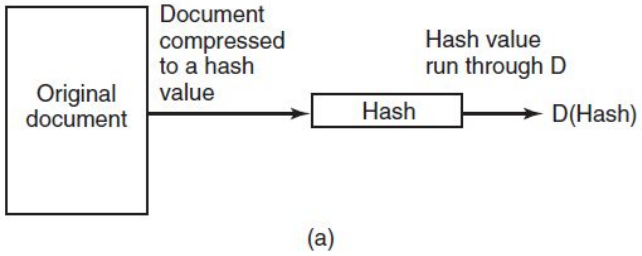

X is using it's private key to encrypt the hashed data.

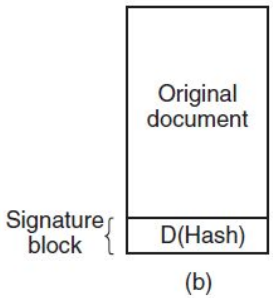
Note - private key's are usually for decrypting… but not here.

# Digital Signatures

Actor X is sending to Actor Y

Y want's to be able to ensure message is *actually* from X.




When Y receives the data, it compute's the hash independently.

It then uses X's public key to reverse the encryption and makes sure the sent hash matches the expected hash.

Essentially proves that the message was sent by the holder of X's private key.

# Security and the OS

Security means many things - we need to narrow our study a bit

| Goal | Threat |
| --- | --- |
| Confidentiality | Exposure of data |
| Integrity | Tampering with data |
| Availability | Denial of service |

Logic Bombs

Distributed Denial of Service attacks (DDS)

# Logic Bombs

```
while (1) fork();
```

Simply put - an OS must ensure that a single process cannot consume too many resources.

Viruses may contain logic bombs, but normally they instead choose to remain undected...

# DDS Attacks

- Virus’s like to stay undetected so they can initiate “logic bombs” externally.
  - Once an attacker has control of a machine, it can be considered a “bot” or “zombie”
  - Often unknown to the user of the machine.
  - Criminals have created botnets of hundreds of thousands of computers… to cripple web servers
    - Very difficult (but possible) to defend against