

CS 3700

Networks and Distributed Systems

Lecture 16: Security Basics

- Definitions**
- Models**
- Principals**
- Basics**
- Vulnerabilities**

Defining “security”

3

- **Policies and mechanisms for preserving desirable protection properties over data and resources.**
- **We reason about security in terms of properties**
 - **Policies specify what we want to enforce**
 - **Mechanisms are the means by which we enforce policies**
- **Always in the context of an attacker**

Security properties

4

- Let's consider an example where a general wants to give the order "Attack at dawn."
- In a network, messages must be distributed from one principal to various other principals.
- What are the properties we would like to enforce on messages?
- Alternatively, what are the bad things that could be done to this message?

Confidentiality

5

"Hey, we're attacking at dawn!"

- Data must only be released to *authorized principals*
- Cryptography has historically focused on providing confidentiality
 - But, there are other mechanisms
- Can have a temporal aspect

Integrity

6

"Retreat at dawn."

- ❑ Data must not be modified (in an undetectable manner)
- ❑ What constitutes a modification?
 - ❑ Corruption
 - ❑ Dropped, replayed, or reordered messages
- ❑ Cryptography has also historically provided this
 - ❑ e.g, (cryptographic) hash functions, HMAC

Authenticity

7

Enemy commander: "Attack at dawn."

- Establishment of identity
 - Or, verification of "genuineness"
- Again, cryptography has long considered this
 - e.g., HMAC, signatures

Availability

8

“Xfk3^#M3mf a __ q3rf” – jamming results in garbled message

- Data and resources must be accessible when required
- Related to integrity, but more concerned with denial of service (DoS) attacks
 - Resource exhaustion (e.g., CPU, memory, network bandwidth)
 - Usually easy to perform, can be difficult to defend

Non-repudiation

9

"I never said to attack at dawn!"

- Data must be bound to identity
 - Prevents denial of message transmission or receipt
- Cryptographic techniques
 - e.g., HMAC, certificates

Access Control

10

- Policy specifying how entities can interact with resources
 - i.e., *Who can access what?*
 - Requires authentication and authorization
- Access control primitives

Principal Users of a system

Subject Entity that acts on behalf of principals

Object Resource acted upon by subjects

Authentication

11

- Verification of identity claim made by a subject on behalf of a principal
- Involves examination of *factors*, or *credentials*
 - Something you *have* – e.g., a badge
 - Something you *know* – e.g., a password
 - Something you *are* – e.g., your fingerprint
- Desirable properties include being *unforgeable*, *unguessable*, and *revocable*

Authorization

- Authorization follows authentication
 - If asking what someone can do, you must know who they are

- Usually represented as a policy specification of what resources can be accessed by a given subject
 - Can also include the nature of the access

Types of Access Control

13

- Discretionary Access Control (DAC)
 - Owners of objects specify policy
- Mandatory Access Control (MAC)
 - Policy based on sensitivity levels – e.g., clearance
 - Owners do not specify their own policies
- Role-based Access Control (RBAC)
 - Central authority defines policy in terms of *roles*
 - Roles \approx permission sets

Access Control Matrices

14

- Introduced by Lampson in 1971
- Static description of system protection state
- Abstract model of concrete systems

Given subjects $s_i \in S$, objects $o_j \in O$, rights $\{R, W, X\}$,

| | O ₁ | O ₂ | O ₃ |
|----------------|----------------|----------------|----------------|
| S ₁ | RW | RX | |
| S ₂ | R | RWX | RW |
| S ₃ | | RWX | |

- Definitions
- Models**
- Principals**
- Basics**
- Vulnerabilities**

Abstract Security Models

16

- Access control lists
- Capabilities
- Bell-LaPadula
- Biba Integrity
- Clark-Wilson
- Brewer-Nash
- Non-interference
- Information flow

Practical Security Models

17

- UNIX permissions
- Windows access control
- Java permissions
- Web (same-origin policy)
- Android permissions
- iOS (MAC model)

Limitations of Access Matrices

18

- The Unix model is very simple
 - Users and groups, read/write/execute
- Not all possible policies can be encoded

| | file 1 | file 2 |
|---------------|------------|------------|
| <i>user 1</i> | --- | <i>rw-</i> |
| <i>user 2</i> | <i>r--</i> | <i>r--</i> |
| <i>user 3</i> | <i>rw-</i> | <i>rwX</i> |
| <i>user 4</i> | <i>rw-</i> | --- |

- file 1: two users have high privileges
 - If user 3 and user 4 are in a group, how to give user 2 read and user 1 nothing?
- file 2: four distinct privilege levels
 - Maximum of three levels (user, group, other)

Access Control List (ACL)

19

⟨object, subject, operation⟩

- ❑ Authorization verified for each request by checking list of tuples
- ❑ Instantiation of access control matrices with update
- ❑ Used pervasively in filesystems and networks
 - ❑ "Users *a*, *b*, and *c* and read file *x*."
 - ❑ "Hosts *a* and *b* can listen on port *x*."
- ❑ Drawbacks?

Capabilities

20

- In this model, authorization is synonymous with possession of a *capability*
 - Capabilities represented as transferable, unforgeable tokens
- Many implementations
 - Hardware
 - Systems (EROS, Capsicum)
 - Languages (E, Caja, Joe-E)
- Drawbacks?

Covert Channels

21

- Access control is defined over "legitimate" channels
 - e.g., shared memory, pipes, sockets, files
- However, isolation in real systems is imperfect

- External observations can be used to create *covert channels*
 - Requires collusion with an insider
- Can be extremely difficult to detect
 - Difficulty is proportionate to channel bandwidth

Side Channels

22

- Side channels result from *inadvertent* information leakage
 - Timing – e.g., password recovery by timing keystrokes
 - Power – e.g., crypto key recovery by power fluctuations
 - RF emissions – e.g., video signal recovery from video cable EM leakage
 - Virtually any shared resource can be used
- Countermeasures?
 - Remove access to shared resource
 - Introduce noise (chaff) or blind the resource

- Definitions
- Models
- Principals**
- Basics**
- Vulnerabilities**

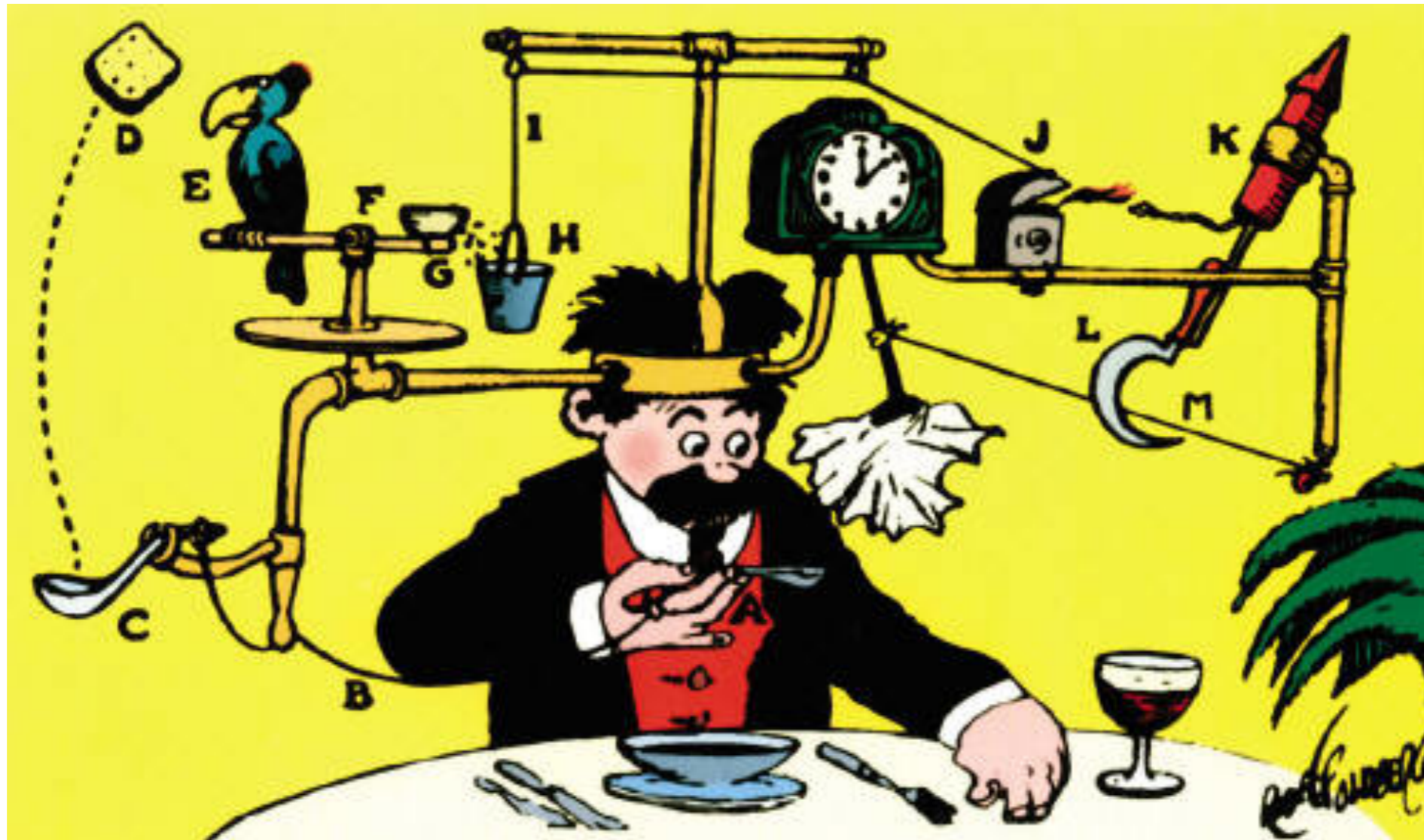
Security Principles

24

- We've seen some basic properties, policies, mechanisms, models, and approaches to security
- But, designing secure systems (and breaking them) remains an art
- Security principles help bridge the gap between art and science
 - Let's look at a few

Economy of Mechanism

25



Would you depend on a defense designed like this?

Economy of Mechanism

26

Simplicity of design implies a smaller attack surface

- Correctness of protection mechanisms is critical
 - "Who watches the watcher?"
 - We need to be able to trust our security mechanisms
 - (Or, at least quantify their efficacy)
- Essentially the KISS principle

Defense in Depth

27

Don't depend on a single protection mechanism, since they are apt to fail

- ❑ Even very simple or formally verified defenses fail
- ❑ Layering defenses increases the difficulty for attackers
- ❑ Defenses should be complementary!

Fail-safe Defaults

28

The absence of explicit permission is equivalent to no permission

- **Systems should be secure "out-of-the-box"**
 - **Most users stick with defaults**
 - **Security should be easy**
 - **Users should "opt-in" to less-secure configurations**

Complete Mediation

29



Complete Mediation

30

Every access to every object must be checked for authorization

- Incomplete mediation implies that a path exists to bypass a security mechanism
- Required property of *reference monitors*

Open Design

31

Kerckhoff's Principle: A cryptosystem should be secure even if everything about the system, except the key, is public knowledge

- **Generalization: A system should be secure even if the adversary knows everything about its design**
 - **Design does not include runtime parameters**
- **Contrast with "security through obscurity"**

Separation of Privilege

32

Privilege, or authority, should only be distributed to subjects that require it

- Some components of a system should be less privileged than others
 - Not every subject needs the ability to do everything
 - Not every subject is deserving of full trust
- Contrast with "ambient authority"

Least Privilege

33

Subjects should possess only that authority that is required to operate successfully

- **Closely related to separation of privilege**
 - Not only should privilege be separated, but subjects should have the *least* amount necessary to perform a task

Compromise Recording

34

Concede that attacks will occur, but record the fact

- Auditing approach to security
 - Detection and recovery
- "Tamper-evident" vs. "tamper-proof"

Threat Models

35

- When analyzing a system's security, we often speak of a *threat model*
 - Threat models bound the capabilities of an attacker
 - Many formal examples from cryptography (Dolev-Yao, IND-CPA, IND-CCA)
- Also important for systems
 - Passive network attacker, active network attacker, privileged local user

Security vs. Usability

36

- Security often comes with a trade-off between the level of protection provided and ease-of-use
 - Systems that try to provide very strong security guarantees tend to be unusable in practice
 - Completely insecure systems are usually easy to use
- Pragmatic security follows the Pareto principle, or 80/20 rule

- Definitions
- Models
- Principals
- Basics**
- Vulnerabilities**

Cryptographic Algorithms

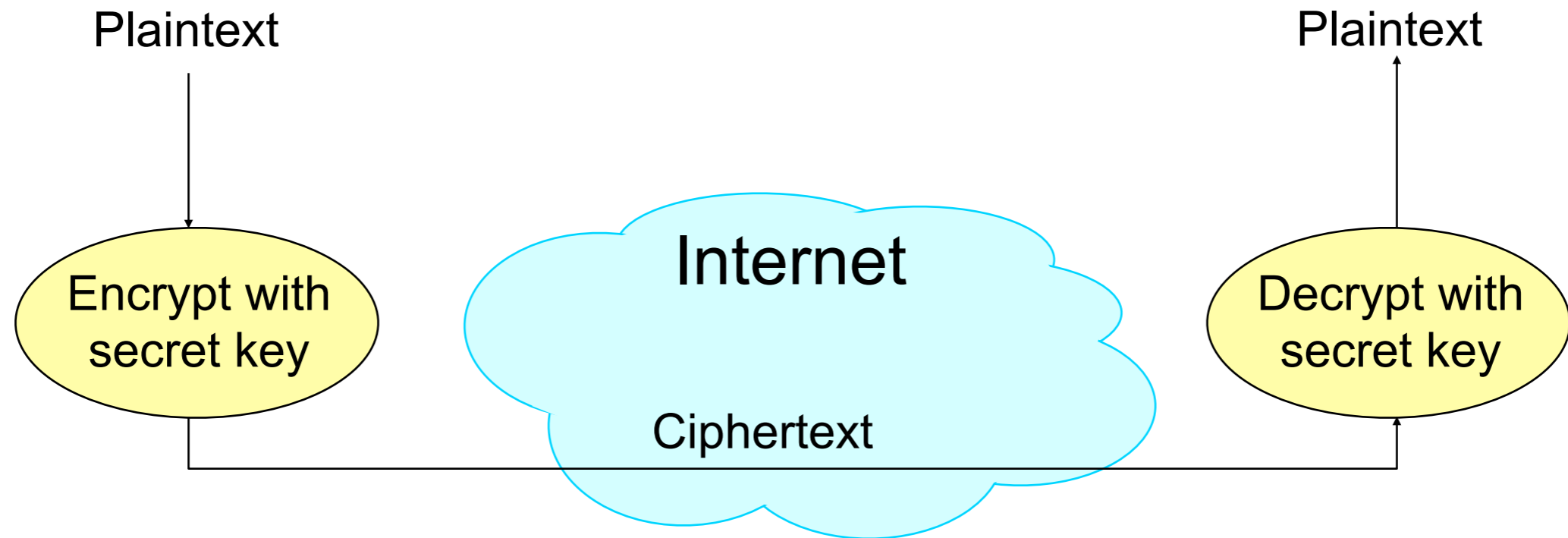
38

- Security foundation: cryptographic algorithms
 - Secret key cryptography, e.g. Data Encryption Standard (DES)
 - Public key cryptography, e.g. RSA algorithm
 - Message digest, e.g. MD5

Symmetric Key

39

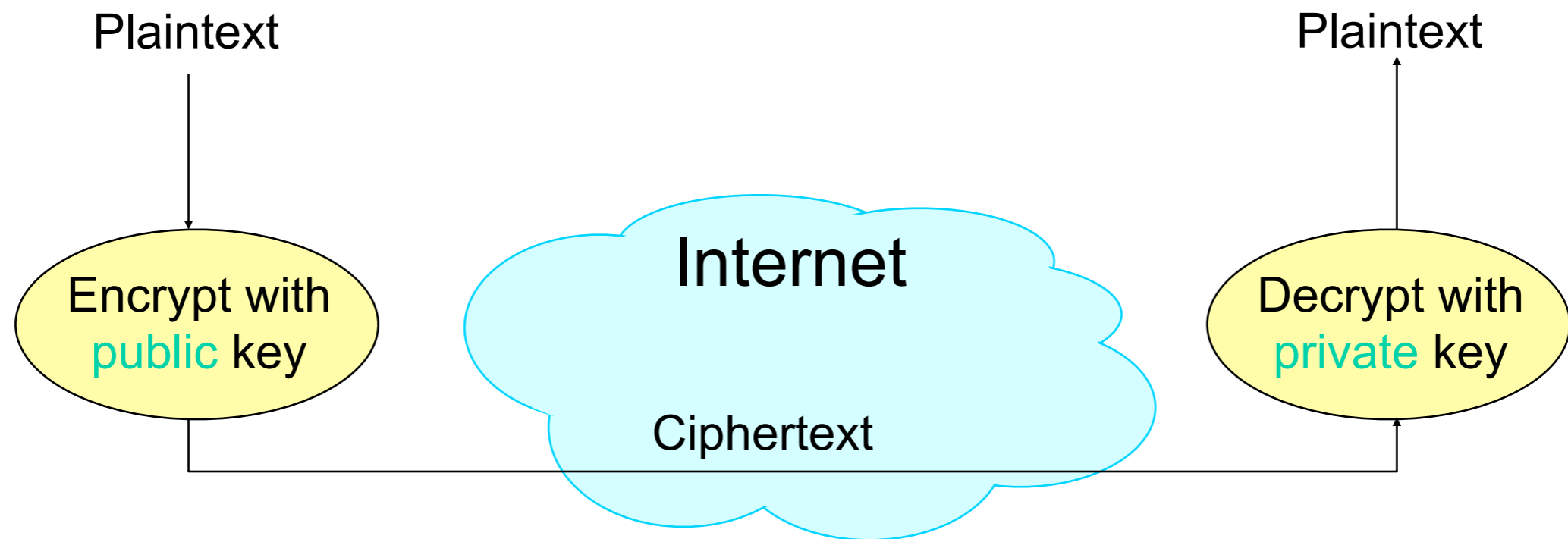
- Both the sender and the receiver use the same secret keys



Public-Key Cryptography: RSA

40

- Sender uses a **public** key
 - Advertised to everyone
- Receiver uses a **private** key



Message Digest (MD) MD5/SHA1

41

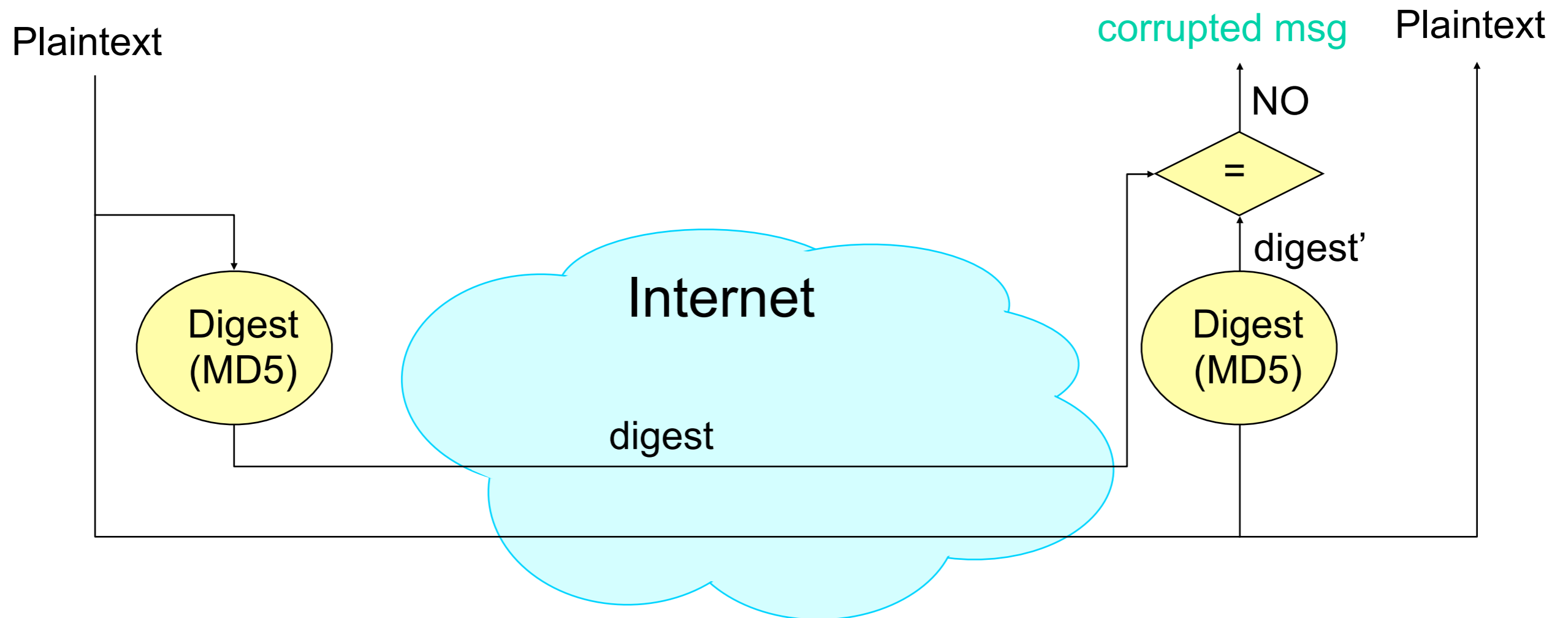
- Can provide data integrity
 - Used to verify the authenticity of a message
- Idea: compute a hash value on the message and send it along with the message
- Receiver can apply the same hash function on the message and see whether the result coincides with the received hash

- Very hard to forge a message that produces the same hash value
 - i.e. Message -> hash is easy
 - Hash -> Message is hard
 - Compare to other error detection methods (CRC, parity, etc)

MD 5 (cont'd)

42

- Basic property: digest operation very hard to invert
 - Send the digest via a different channel



Transport Layer Security

43

- Application-layer protocol for confidentiality, integrity, authenticity between clients and servers
 - Introduced by Netscape in 1995 as the Secure Sockets Layer (SSL) to encapsulate HTTP traffic – i.e., HTTPS
- Sits between application and transport layers
 - Therefore, applications must be TLS-aware
- Both client and server must have an asymmetric keypair
 - In practice, X.509 certificates and PKI rooted in certificate authorities (CAs)

X.509

44

Version: 3 (0x2)

Serial Number:

0e:77:76:8a:5d:07:f0:e5:79:59:ca:2a:9d:50:82:b5

Signature Algorithm: sha1WithRSAEncryption

Issuer: C=US, O=DigiCert Inc, OU=www.digicert.com,
CN=DigiCert High Assurance EV CA-1

Validity

Not Before: May 27 00:00:00 2011 GMT

Not After : Jul 29 12:00:00 2013 GMT

Subject: C=US, ST=California, L=San Francisco,
O=GitHub, Inc., CN=github.com

Subject Public Key Info:

Public Key Algorithm: rsaEncryption

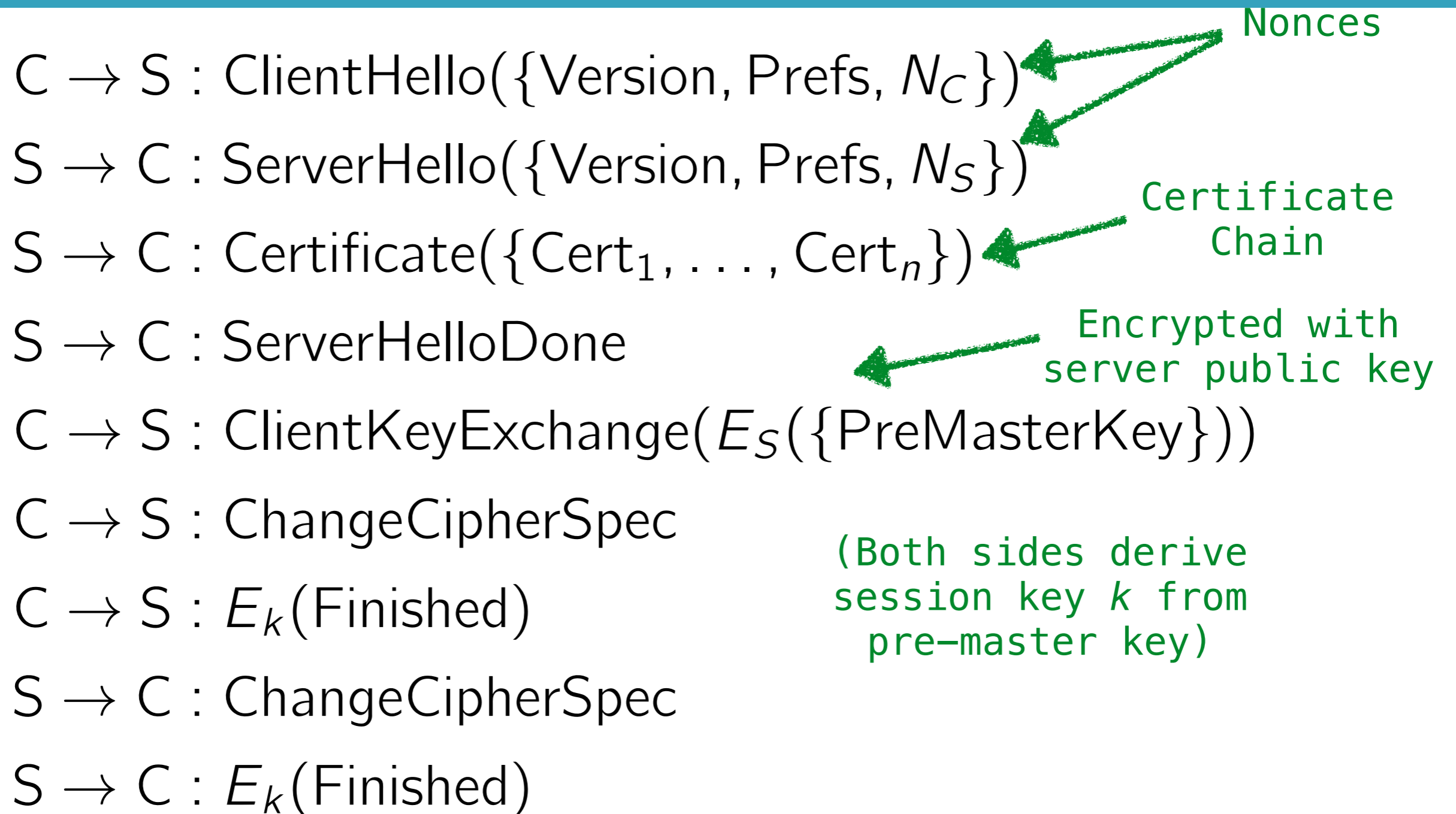
Public-Key: (2048 bit)

Modulus:

00:ed:d3:89:c3:5d:70:72:09:f3:33:4f:1a:72:74:
d9:b6:5a:95:50:bb:68:61:9f:f7:fb:1f:19:e1:da:

Connection Establishment

45



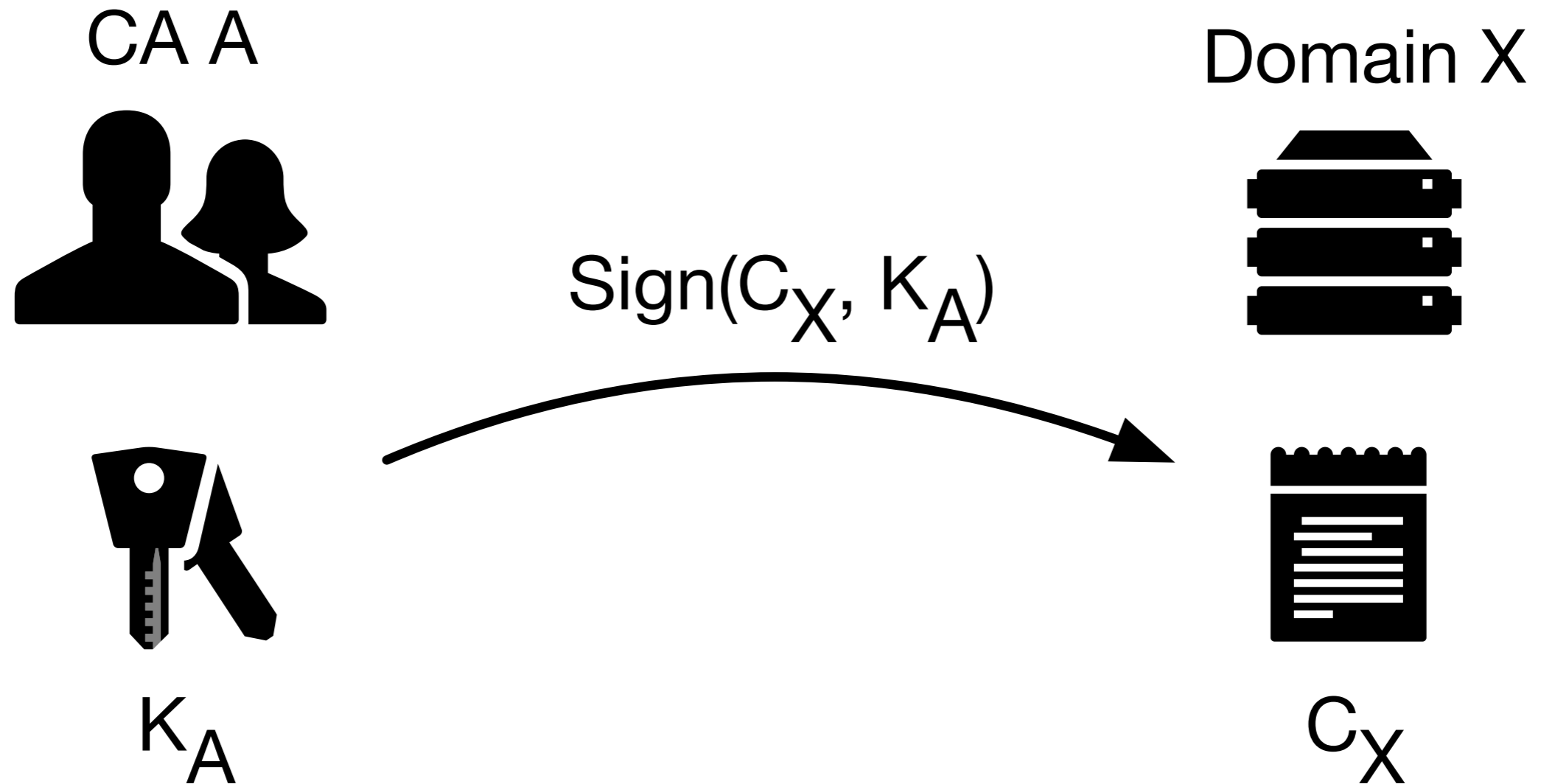
TLS Authentication

46

- Typical scenario: identify the server
 - X.509 Common Name (CN) field contains hostname
 - During connection establishment, client can check the CN, verify the CA's signature of the server's certificate, and check whether the CA is trusted
 - CA trust established via local trust anchors – i.e., a list of CA public keys obtained out-of-band
- TLS can also provide mutual authentication
 - Server can require a client certificate and perform an analogous check
 - Usually, client authentication is handled using another mechanism

TLS Authentication

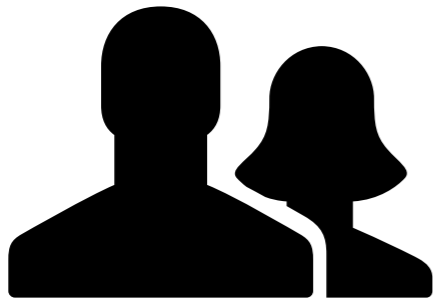
47



TLS Authentication

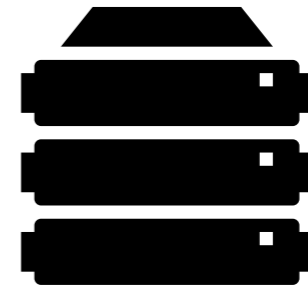
48

CA A



K_A

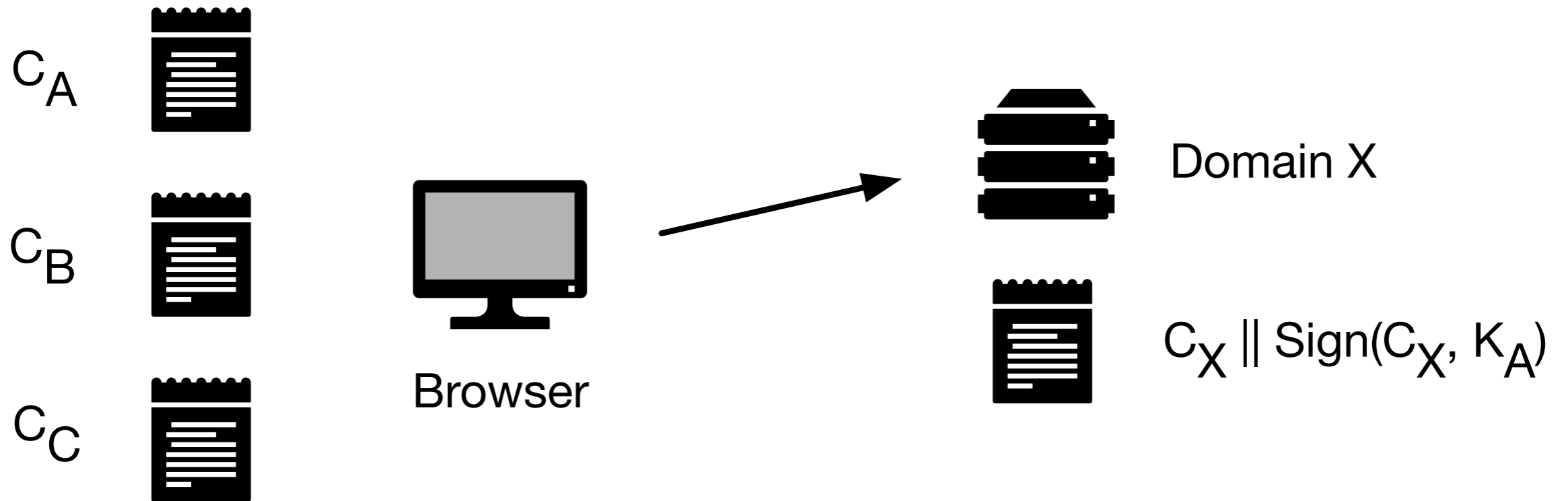
Domain X



$C_X \parallel \text{Sign}(C_X, K_A)$

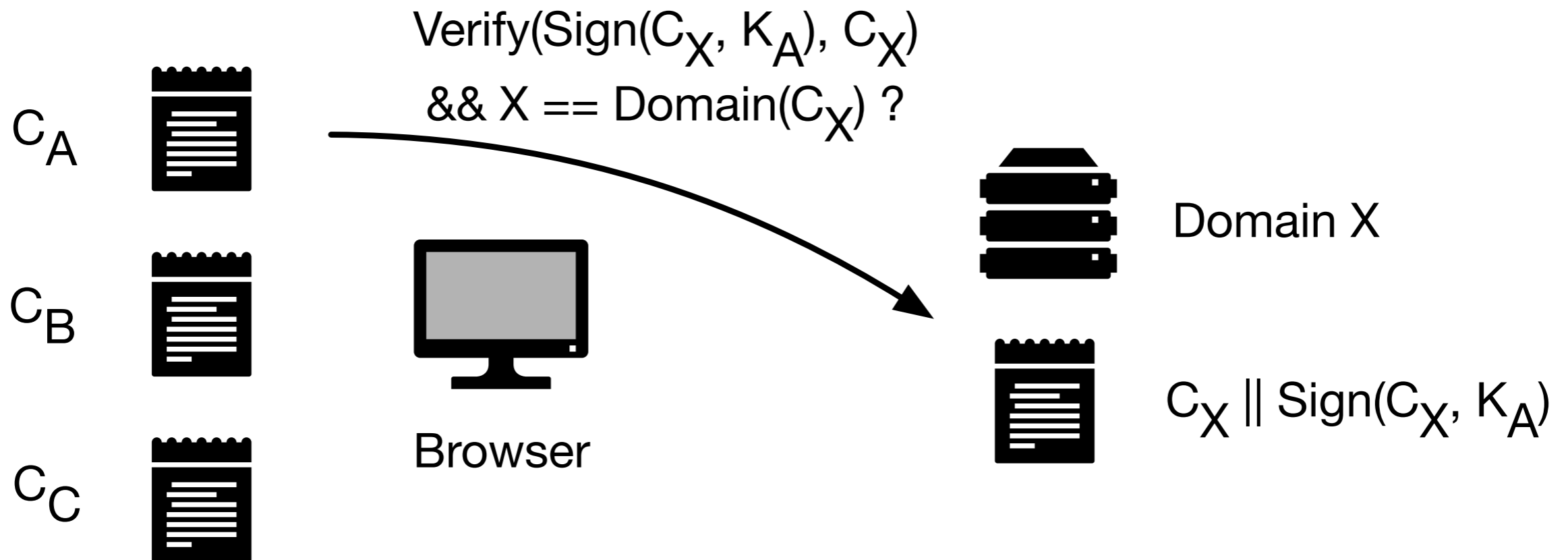
TLS Authentication

49



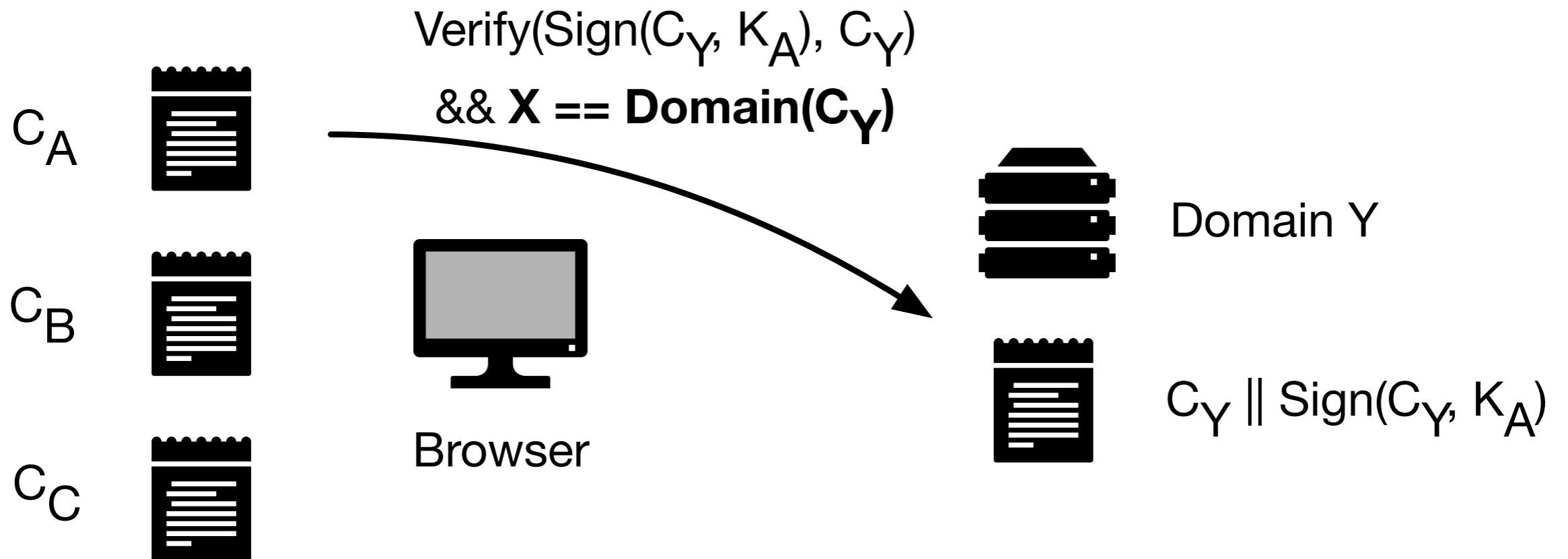
TLS Authentication

50



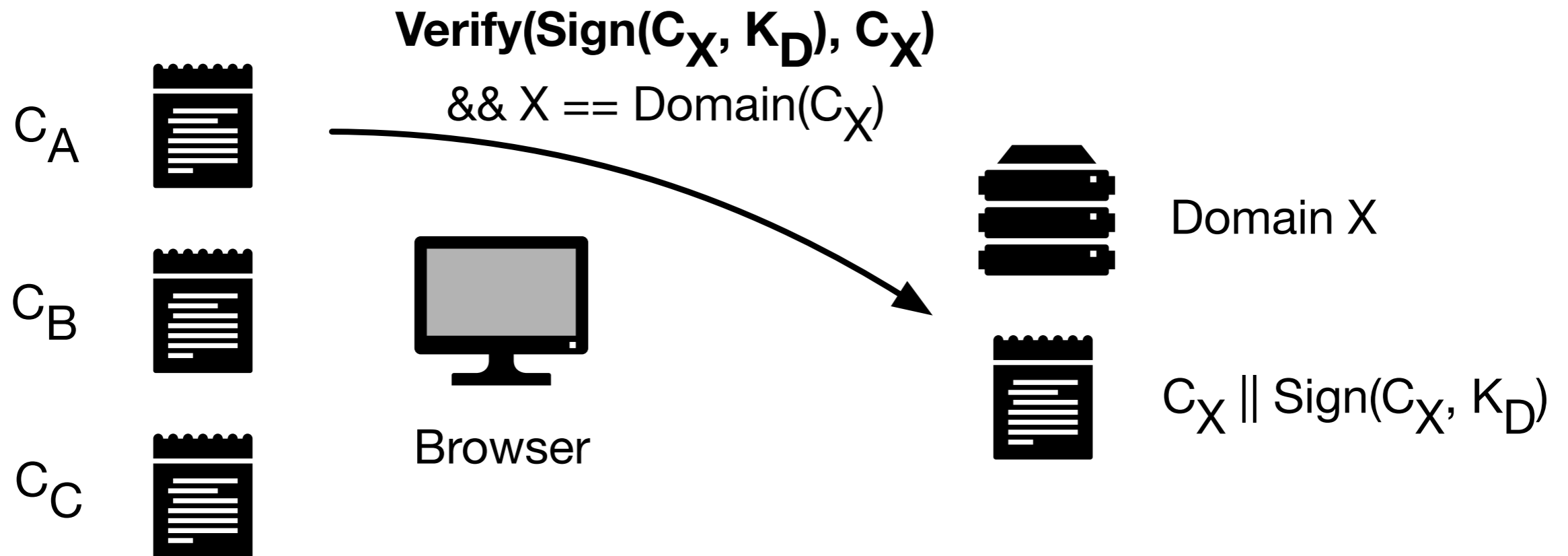
TLS Authentication

51



TLS Authentication

52



CA Trustworthiness

53

- A CA is essentially a trusted third party
 - Certificate signatures are attestations of authenticity for the server and (optionally) the client
 - Remember: trust is bad and should be minimized!
- If a CA mistakenly (or purposefully) signs a certificate for a domain and provides it to a malicious principal, TLS can be subverted
- Not only must we trust root CAs, but also *intermediate* CAs that have been delegated signing authority

CA Trustworthiness

54

- Clearly, the CA secret key must be protected at all costs
 - Possession of the CA secret key grants adversaries the ability to sign any domain
 - Attractive target for adversaries
- Signatures should only be issued after verifying the identity of the requester
 - Also known as domain validation
 - Should be easy, right?

CA Failures

55

Issued to: Microsoft Corporation
Issued by: VeriSign Commercial Software Publishers CA
Valid from 1/29/2001 to 1/30/2002
Serial number is 1B51 90F7 3724 399C 9254 CD42 4637 996A

Issued to: Microsoft Corporation
Issued by: VeriSign Commercial Software Publishers CA
Valid from 1/30/2001 to 1/31/2002
Serial number is 750E 40FF 97F0 47ED F556 C708 4EB1 ABFD

- In 2001, VeriSign issued two executable signing certificates to someone claiming to be from Microsoft
 - Could be used to issue untrusted software updates

Independent Iranian hacker claims responsibility for Comodo hack

Posts claiming to be from an Iranian hacker responsible for the Comodo hack ...

by Peter Bright - Mar 28 2011, 11:15am EDT

65

```
1. Hello
2.
3. I'm writing this to the world, so you'll know more about me..
4.
5. At first I want to give some points, so you'll be sure I'm the hacker:
6.
7. I hacked Comodo from InstantSSL.it, their CEO's e-mail address mfpenco@mfpenco.com
8. Their Comodo username/password was: user: gtdadmin password: [trimmed]
9. Their DB name was: globaltrust and instantsslcms
```

The alleged hacker's claim of responsibility on pastebin.com

The hack that resulted in [Comodo creating certificates](#) for popular e-mail providers including Google Gmail, Yahoo Mail, and Microsoft Hotmail has been claimed as the work of an independent Iranian patriot. A [post](#) made to data sharing site pastebin.com by a person going by the handle "comodohacker" claimed responsibility for the hack and described details of the attack. A second [post](#) provided source code apparently reverse-engineered as one of the parts of the attack.

Another fraudulent certificate raises the same old questions about certificate authorities

For the second time this year, Iranian hackers have created a fraudulent ...

by Peter Bright - Aug 29 2011, 11:12pm EDT

42

Earlier this year, an [Iranian hacker](#) broke into servers belonging to a reseller for certificate authority Comodo and issued himself a range of certificates for sites including Gmail, Hotmail, and Yahoo! Mail. With these certificates, he could eavesdrop on users of those mail providers, even if they use SSL to protect their mail sessions.

It's happened again. This time, Dutch certificate authority DigiNotar has issued a fraudulent certificate for google.com and all subdomains. As before, Gmail appears to be the target. The perpetrator also appears to be Iranian, with [reports](#) that the certificate has been used in the wild for man-in-the-middle attacks in that country. The certificate was issued on July 10th, and so could have been in use for several weeks prior to its discovery.

DigiNotar has revoked the certificate, which provides some protection to users (though many applications do not bother checking for revocations). However, the company has so far not disclosed how the certificate was issued in the first place, making it unclear that its integrity has been restored. As a result, Google and Mozilla have both made patches to [Chrome](#) and [Firefox](#) respectively that blacklist the entire certificate authority.

TrustWave

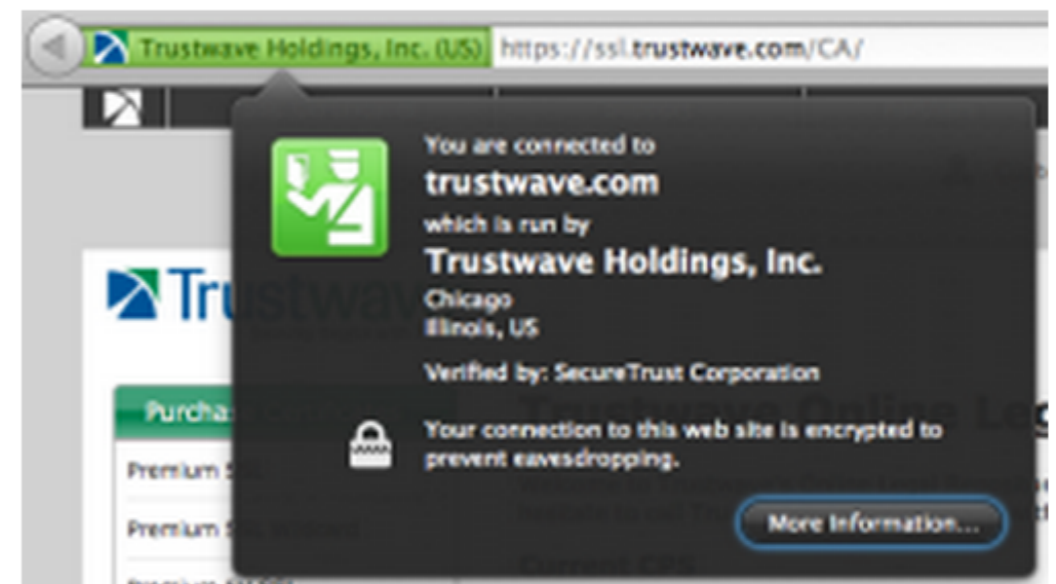
58


Trustwave issued a man-in-the-middle certificate



Certificate authority [Trustwave](#) issued a certificate to a company allowing it to issue valid certificates for any server. This enabled the company to listen in on encrypted traffic sent and received by its staff using services such as Google and Hotmail. Trustwave has since revoked the CA certificate and [vowed](#) to refrain from issuing such certificates in future.

According to Trustwave, the CA certificate was used in a data loss prevention (DLP) system,



All of the major browsers contain the Trustwave root certificate 

Certificate Pinning

59

- One approach to avoid HTTPS attacks is to *pin* certificates
 - Browser downloads server certificate as usual
 - Server certificate is validated against a trusted local copy or hash
 - Trusted data shipped with browser
- This technique was used to detect the use of fake DigiNotar-issued certificates in 2011
 - But, it doesn't scale – reserved for "critical" sites

- Definitions
- Models
- Principals
- Basics
- Vulnerabilities**

Importance of Network Security

61

- Internet currently used for important services
 - Financial transactions, medical records
- Could be used in the future for *critical* services
 - 911, surgical operations, energy system control, transportation system control
- Networks more open than ever before
 - Global, ubiquitous Internet, wireless
- Malicious Users
 - Selfish users: want more network resources than you
 - Malicious users: would hurt you even if it doesn't get them more network resources

Network Security Problems

62

- Host Compromise
 - Attacker gains control of a host
- Denial-of-Service
 - Attacker prevents legitimate users from gaining service
- Attack can be both
 - E.g., host compromise that provides resources for denial-of-service

Host Compromise

63

- One of earliest major Internet security incidents
 - Internet Worm (1988): compromised almost every BSD-derived machine on Internet
- Today: estimated that a single worm could compromise 10M hosts in < 5 min
- Attacker gains control of a host
 - Reads data
 - Erases data
 - Compromises another host
 - Launches denial-of-service attack on another host

Definitions

64

- **Worm**
 - **Replicates itself**
 - **Usually relies on stack overflow attack**

- **Virus**
 - **Program that attaches itself to another (usually trusted) program**

- **Trojan horse**
 - **Program that gives a hacker a back door**
 - **Usually relies on user exploitation**

Host Compromise: Buffer Overflow

65

- Typical code has many bugs because those bugs are not triggered by common input
- Network code is vulnerable because it accepts input from the network
- Network code that runs with high privileges (i.e., as root) is especially dangerous
 - E.g., web server

Example

66

- What is wrong here?

```
#define MAXNAMELEN 64

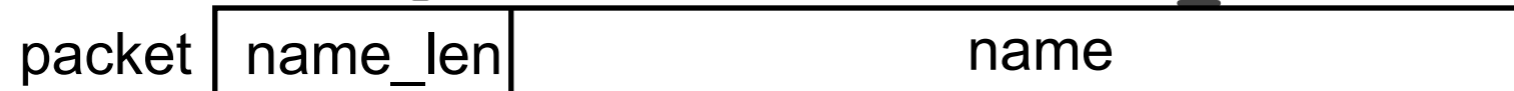
int offset = OFFSET_USERNAME;

char username[MAXNAMELEN];

int name_len;

name_len = ntohl(*(int *)packet);

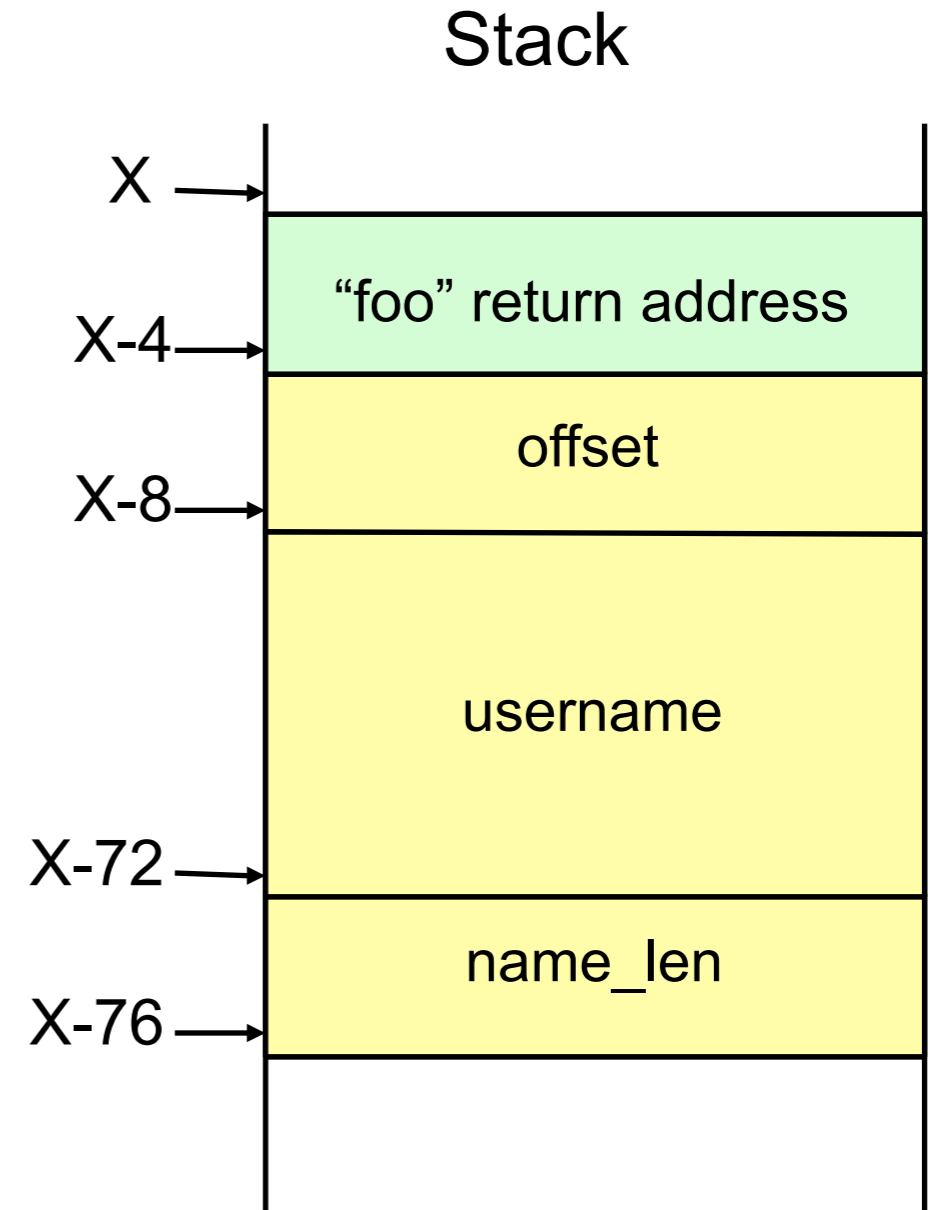
memcpy(&username, packet[offset], name_len);
```



Example

67

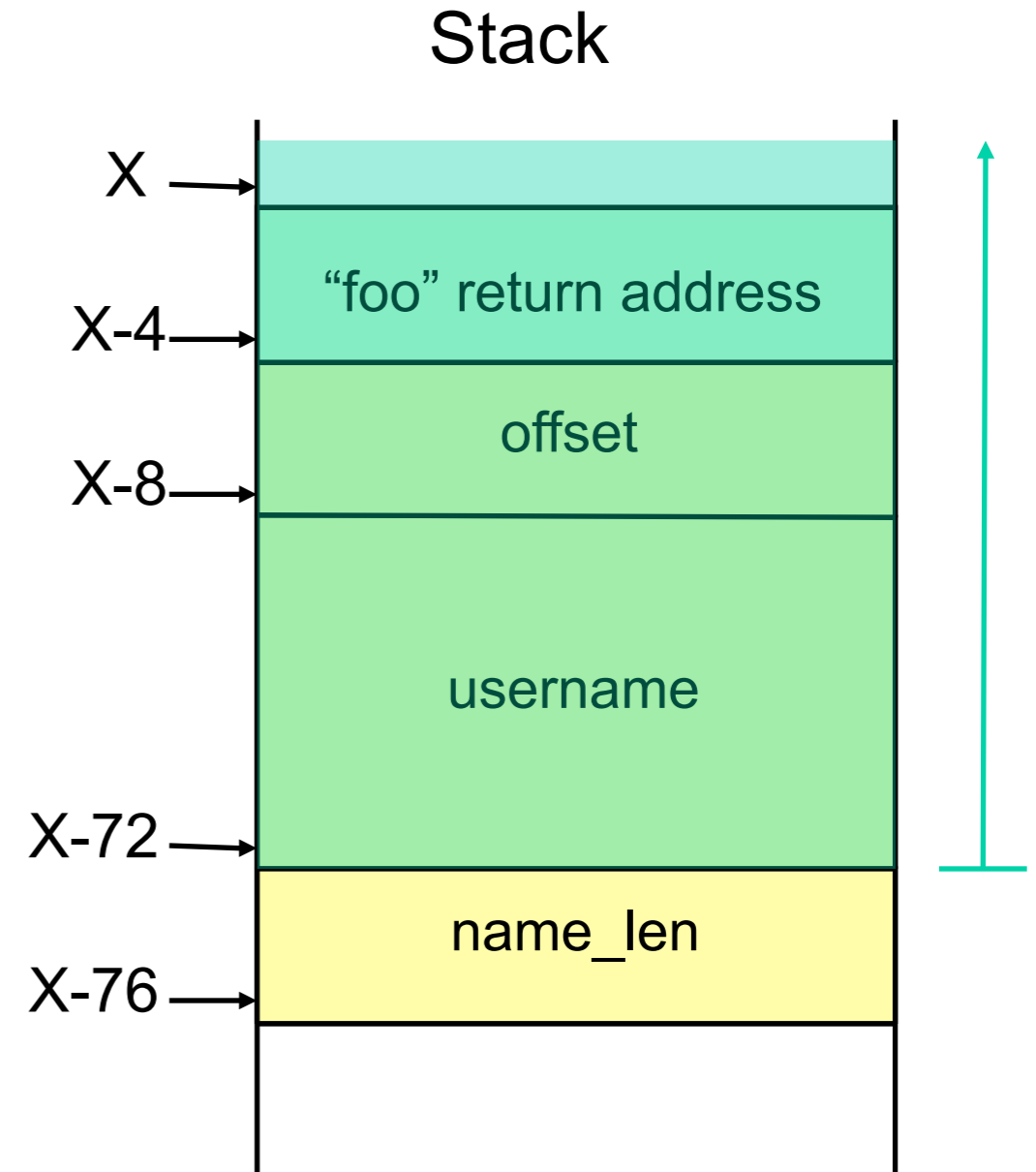
```
void foo(packet) {  
    #define MAXNAMELEN 64  
  
    int offset = OFFSET_USERNAME;  
  
    char username[MAXNAMELEN];  
  
    int name_len;  
→  
  
    name_len = ntohl(*(int*)packet);  
  
    memcpy(&username,  
          packet[offset], name_len);  
  
    ...  
}
```



Example

68

```
void foo(packet) {  
    #define MAXNAMELEN 64  
    int offset = OFFSET_USERNAME;  
    char username[MAXNAMELEN];  
    int name_len;  
    → name_len = ntohl(*(int *) packet);  
    memcpy(&username,  
          packet[offset], name_len);  
    ...  
}
```



Effect of Stack Based Buffer Overflow

69

- Write into part of the stack or heap
 - Write arbitrary code to part of memory
 - Cause program execution to jump to arbitrary code

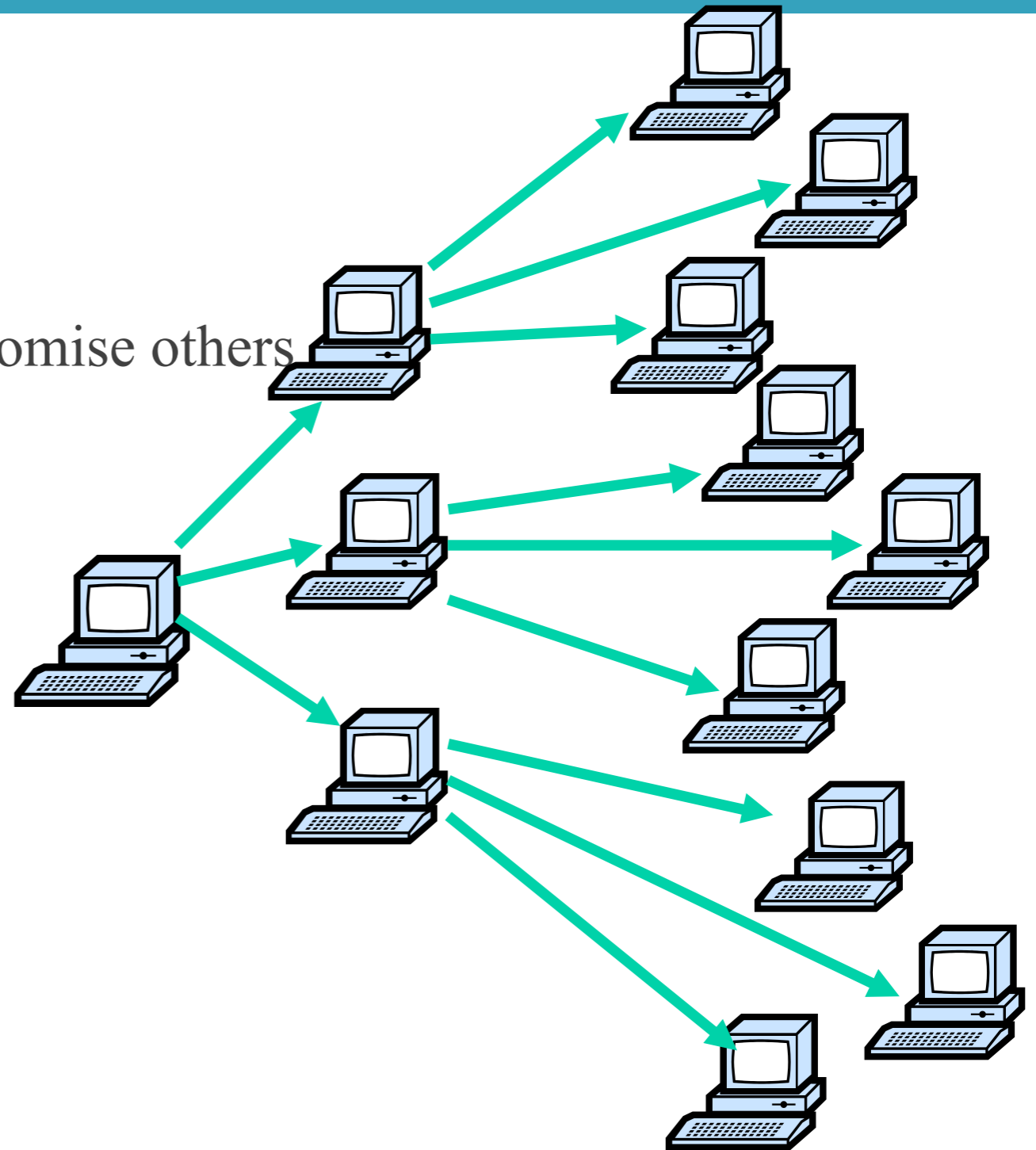
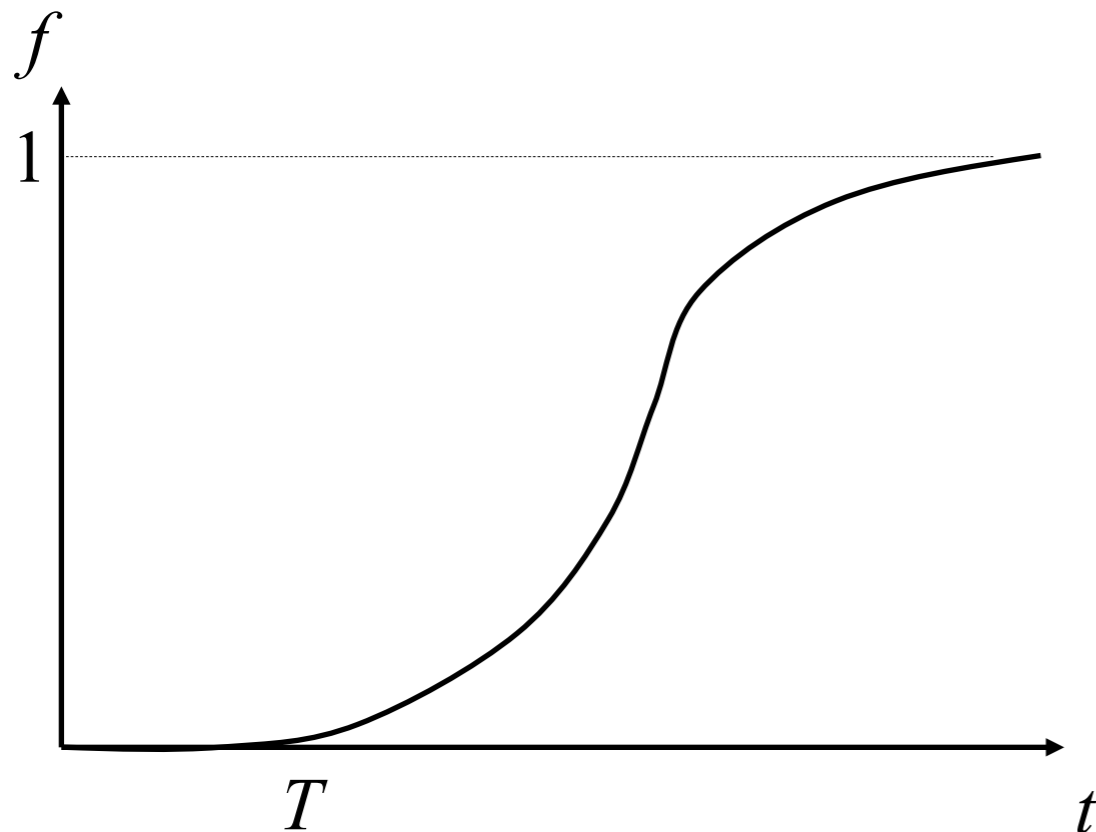
- Worm
 - Probes host for vulnerable software
 - Sends bogus input
 - Attacker can do anything that the privileges of the buggy program allows
 - Launches copy of itself on compromised host
 - Spread at exponential rate
 - 10M hosts in < 5 minutes

Worm Spreading

70

$$f = (e^{K(t-T)} - 1) / (1 + e^{K(t-T)})$$

- f – fraction of hosts infected
- K – rate at which one host can compromise others
- T – start time of the attack



Worm Examples

71

- Morris worm (1988)
- Code Red (2001)
- MS Slammer (January 2003)
- MS Blaster (August 2003)

MS SQL Slammer (January 2003)

72

- Uses UDP port 1434 to exploit a buffer overflow in MS SQL server
- Effect
 - Generate massive amounts of network packets
 - Brought down as many as 5 of the 13 internet root name servers
- Others
 - The worm only spreads as an in-memory process: it never writes itself to the hard drive
 - Solution: close UDP port on firewall and reboot

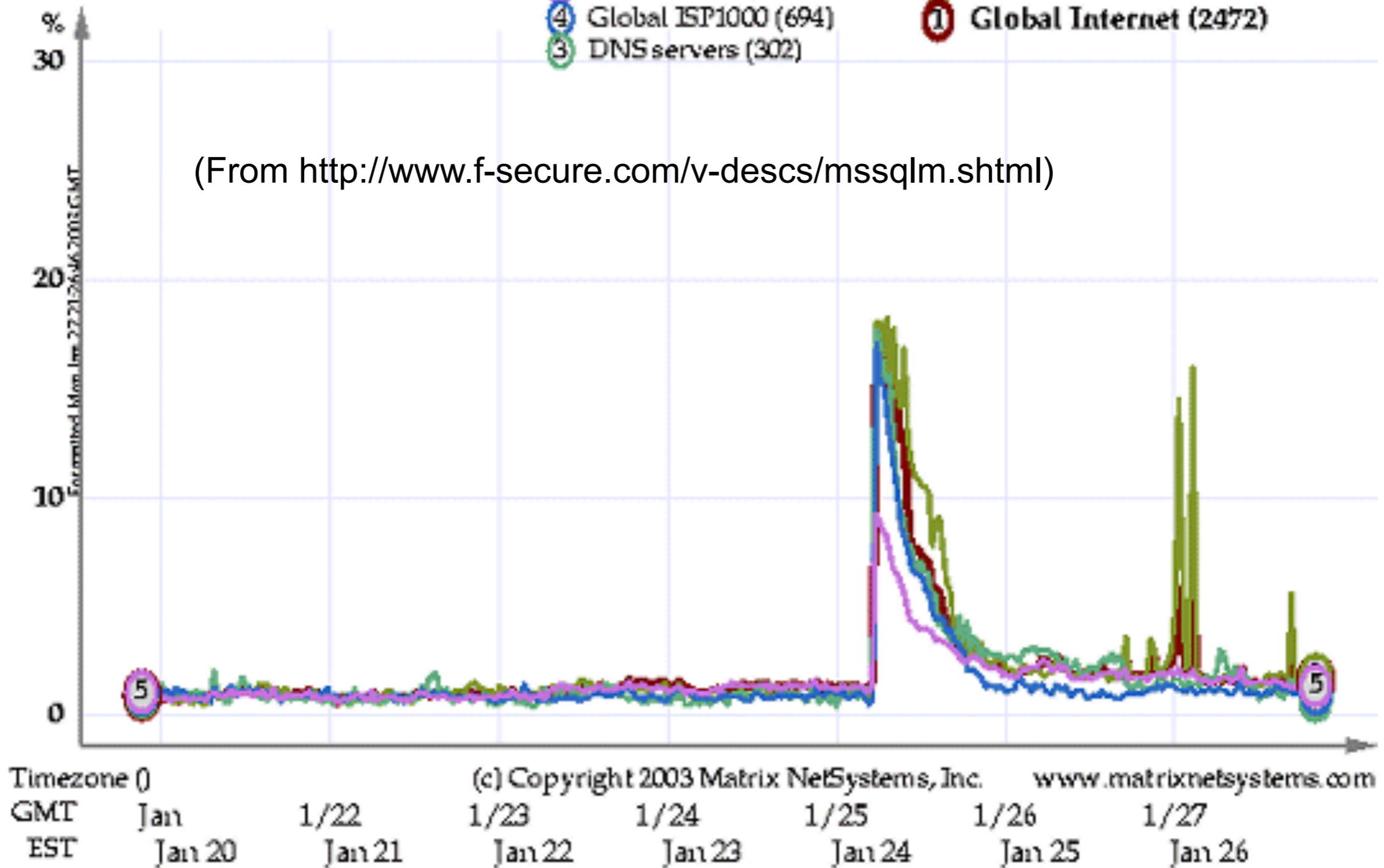
MS SQL Slammer (January 2003)

73

Packet Loss %

- 5 Global Paths (1393)
- 4 Global ISP1000 (694)
- 3 DNS servers (302)
- 2 Global Web (734)
- 1 Global Internet (2472)

(From <http://www.f-secure.com/v-descs/mssqlm.shtml>)



Hall of Shame

74

- Software that have had many stack overflow bugs:
 - BIND (most popular DNS server)
 - RPC (Remote Procedure Call, used for NFS)
 - NFS (Network File System)
 - Sendmail (most popular UNIX mail delivery software)
 - IIS (Windows web server)
 - SNMP (Simple Network Management Protocol, used to manage routers and other network devices)

Potential Solutions

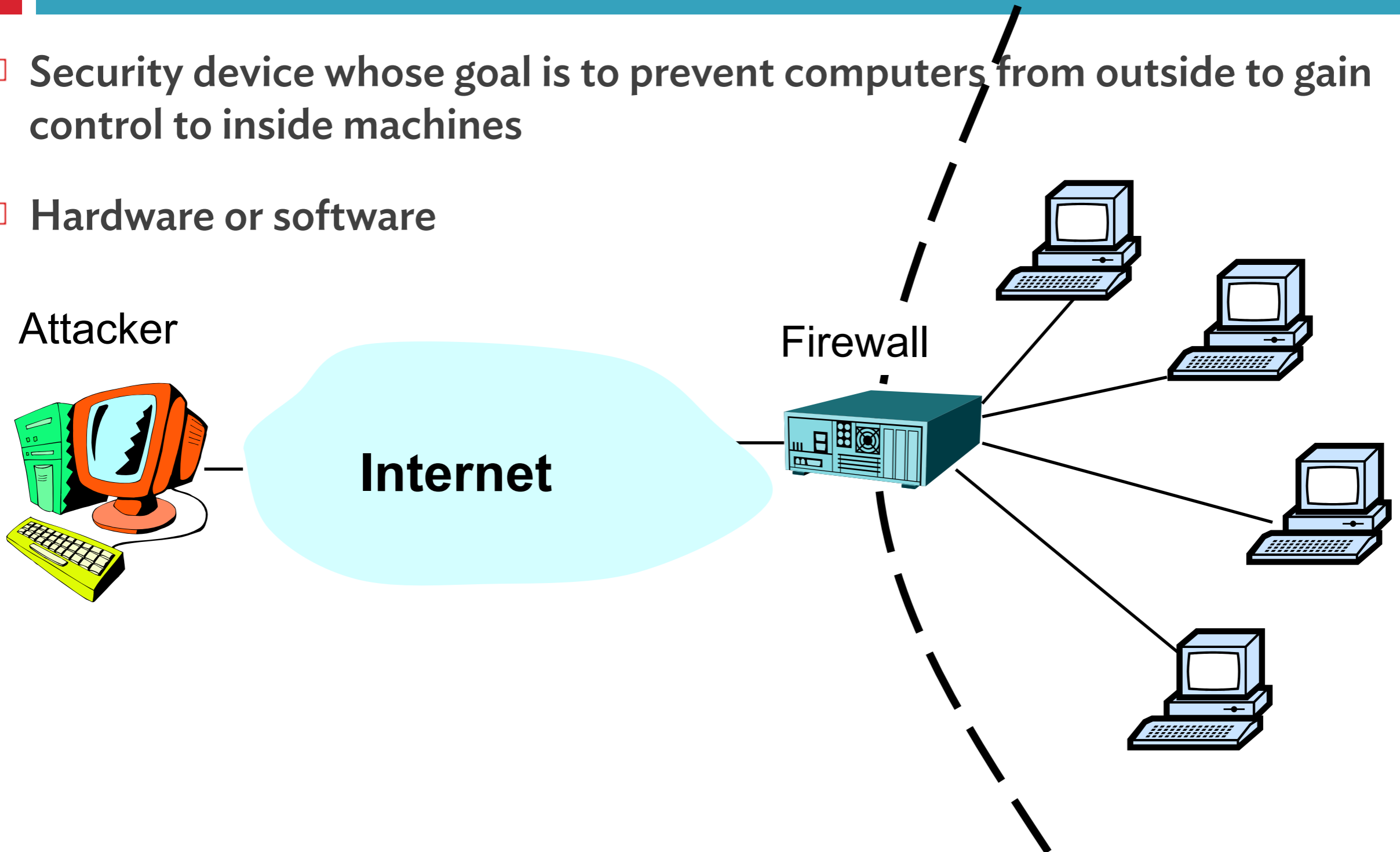
75

- Don't write buggy software
 - It's not like people try to write buggy software
- Type-safe Languages
 - Unrestricted memory access of C/C++ contributes to problem
 - Use Java, Perl, or Python instead
- OS architecture
 - Compartmentalize programs better, so one compromise doesn't compromise the entire system
 - E.g., DNS server doesn't need total system access
- Firewalls

Firewall

76

- Security device whose goal is to prevent computers from outside to gain control to inside machines
- Hardware or software



Firewall (cont'd)

77

- Restrict traffic between Internet and devices (machines) behind it based on
 - Source address and port number
 - Payload
 - Stateful analysis of data
- Examples of rules
 - Block any external packets not for port 80
 - Block any email with an attachment
 - Block any external packets with an internal IP address
 - Ingress filtering

Firewalls: Properties

78

- Easier to deploy firewall than secure all internal hosts
- Doesn't prevent user exploitation
- Tradeoff between availability of services (firewall passes more ports on more machines) and security
 - If firewall is too restrictive, users will find way around it, thus compromising security
 - E.g., have all services use port 80
- Can't prevent problem from spreading from within

Host Compromise: User Exploitation

79

- Some security architectures rely on the user to decide if a potentially dangerous action should be taken, e.g.,
 - Run code downloaded from the Internet
 - “Do you accept content from Microsoft?”
 - Run code attached to email
 - “subject: You’ve got to see this!”
 - Allow a macro in a data file to be run
 - “Here is the latest version of the document.”

User Exploitation

80

- Users are not good at making this decision
 - Which of the following is the real name Microsoft uses when you download code from them?
 - Microsoft
 - Microsoft, Inc.
 - Microsoft Corporation
- Typical email attack
 - Attacker sends email to some initial victims
 - Reading the email / running its attachment / viewing its attachment opens the hole
 - Worm/trojan/virus mails itself to everyone in address book

Solutions

81

- OS architecture
- Don't ask the users questions which they don't know how to answer anyway
- Separate code and data
 - Viewing data should not launch attack
- Be very careful about installing new software

Denial of Service

82

- Huge problem in current Internet
 - Major sites attacked: Yahoo!, Amazon, eBay, CNN, Microsoft
 - 12,000 attacks on 2,000 organizations in 3 weeks
 - Some more than 600,000 packets/second
 - More than 192Mb/s
 - Almost all attacks launched from compromised hosts
- General form
 - Prevent legitimate users from gaining service by overloading or crashing a server
 - E.g., SYN attack

Effect on Victim

83

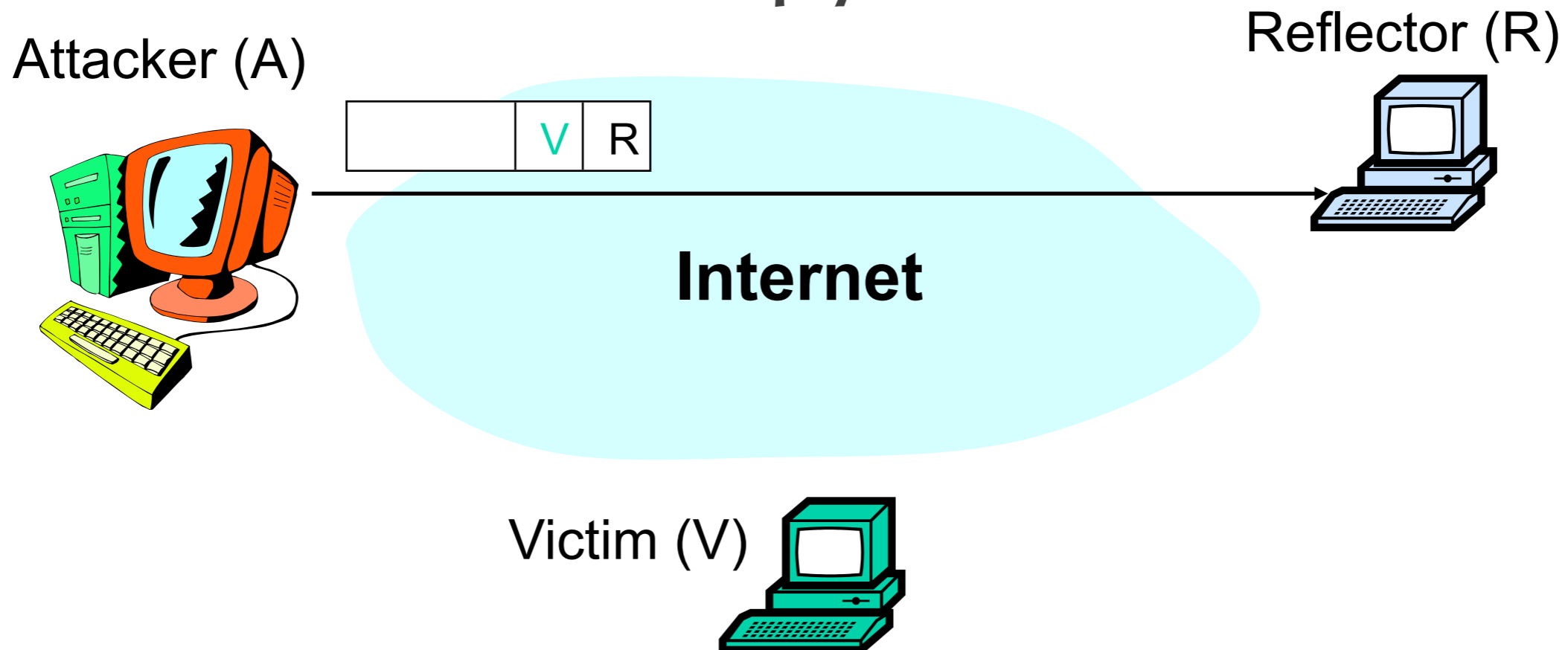
- ❑ Buggy implementations allow unfinished connections to eat all memory, leading to crash
- ❑ Better implementations limit the number of unfinished connections
 - ❑ Once limit reached, new SYNs are dropped
- ❑ Effect on victim's users
 - ❑ Users can't access the targeted service on the victim because the unfinished connection queue is full → DoS

Other Denial-of-Service Attacks

84

□ Reflection

- Cause one non-compromised host to attack another
- E.g., host A sends DNS request or TCP SYN with source V to server R. R sends reply to V

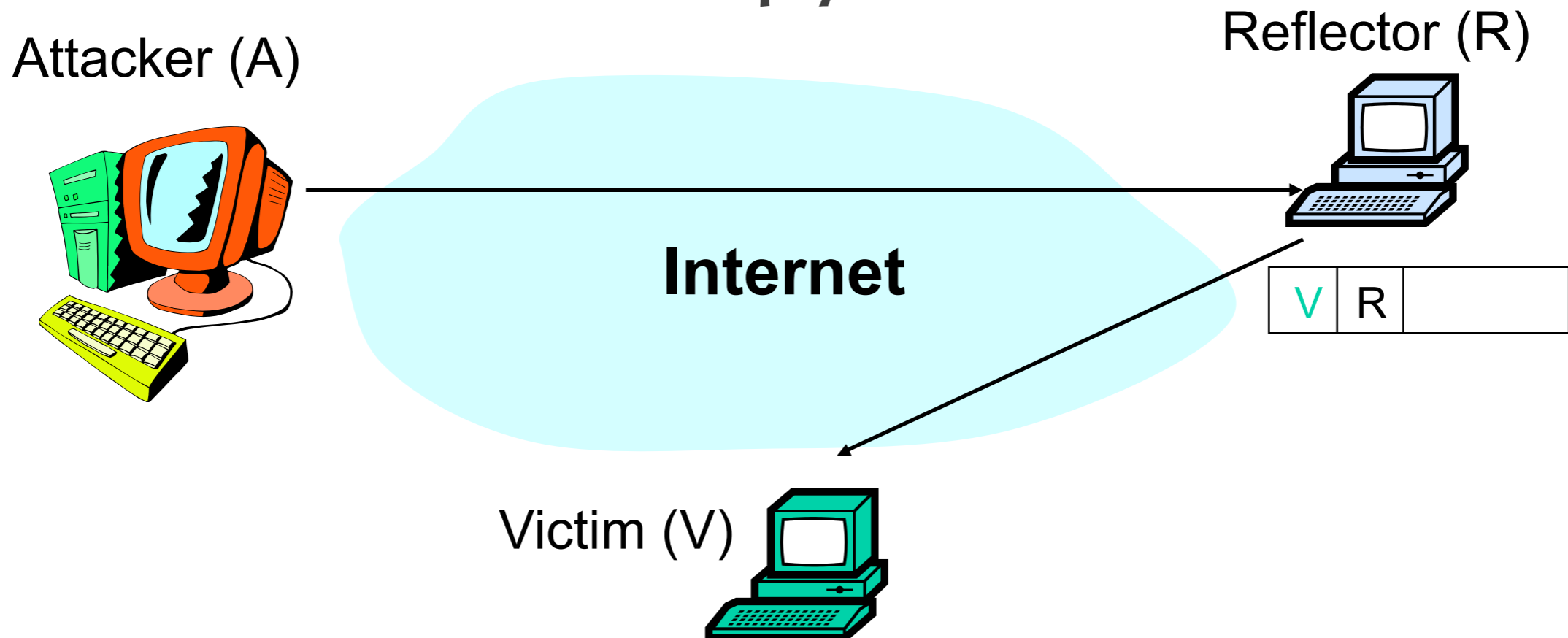


Other Denial-of-Service Attacks

85

□ Reflection

- Cause one non-compromised host to attack another
- E.g., host A sends DNS request or TCP SYN with source V to server R. R sends reply to V



Other Denial-of-Service Attacks

86

□ DNS

- Ping flooding attack on DNS root servers (October 2002)
- 9 out of 13 root servers brought down
- Relatively small impact (why?)

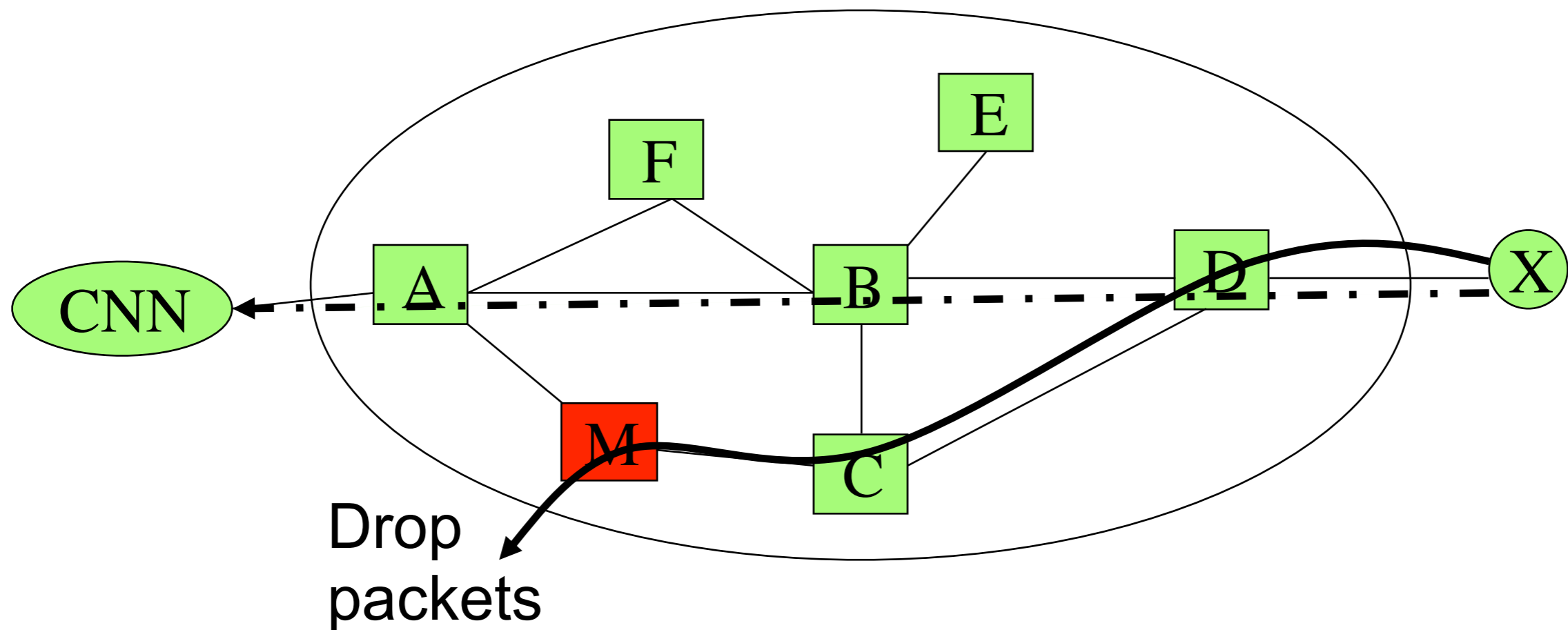
□ BGP

- Address space hijacking: Claiming ownership over the address space owned by others
 - October 1995, Los Angeles county pulled down
- Also happen because of operator mis-configurations

Address Space Hijacking

87

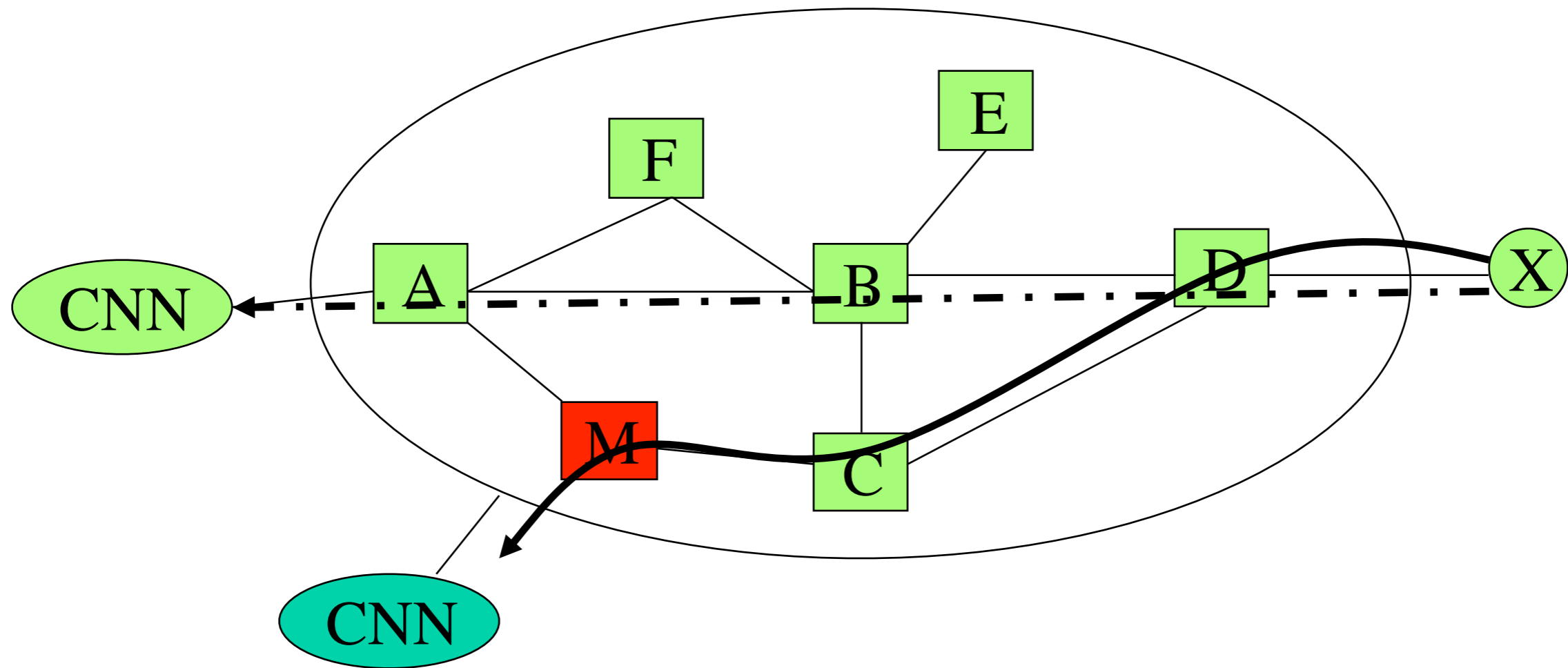
- M hijacks the address space of CNN



Renders Destination Network Unreachable

Address Space Hijacking

88



Impersonates end-hosts in destination network

Dealing with Attacks

89

- Distinguish attack from flash crowd
- Prevent damage
 - Distinguish attack traffic from legitimate traffic
 - Rate limit attack traffic
- Stop attack
 - Identify attacking machines
 - Shutdown attacking machines
 - Usually done manually, requires cooperation of ISPs, other users
- Identify attacker
 - Very difficult, except
 - Usually brags/gloats about attack on IRC
 - Also done manually, requires cooperation of ISPs, other users

Incomplete Solutions

90

- Fair queueing, rate limiting (e.g., token bucket)
- Prevent a user from sending at 10Mb/s and hurting a user sending at 1Mb/s
- Does not prevent 10 users from sending at 1Mb/s and hurting a user sending a 1Mb/s

Identify and Stop Attacking Machines

91

- ❑ Defeat spoofed source addresses
- ❑ Does not stop or slow attack
- ❑ Ingress filtering
 - ❑ A domain's border router drop outgoing packets which do not have a valid source address for that domain
 - ❑ If universal, could abolish spoofing
- ❑ IP Traceback
 - ❑ Routers probabilistically tag packets with an identifier
 - ❑ Destination can infer path to true source after receiving enough packets

Summary

92

- Network security is possibly the Internet's biggest problem
 - Preventing Internet from expanding into critical applications
- Host Compromise
 - Poorly written software
 - Solutions: better OS security architecture, type-safe languages, firewalls
- Denial-of-Service
 - No easy solution: DoS can happen at many levels