Distributed Systems

Dr. Philipp Leitner
Distributed Systems Group
Vienna University of Technology

leitner@infosys.tuwien.ac.at

Distributed Systems Group

Outline

1. Introduction
2. Authentication
3. Integrity
4. Access Control
5. Some State-of-the-Art Attacks

6. Advanced Security Lectures

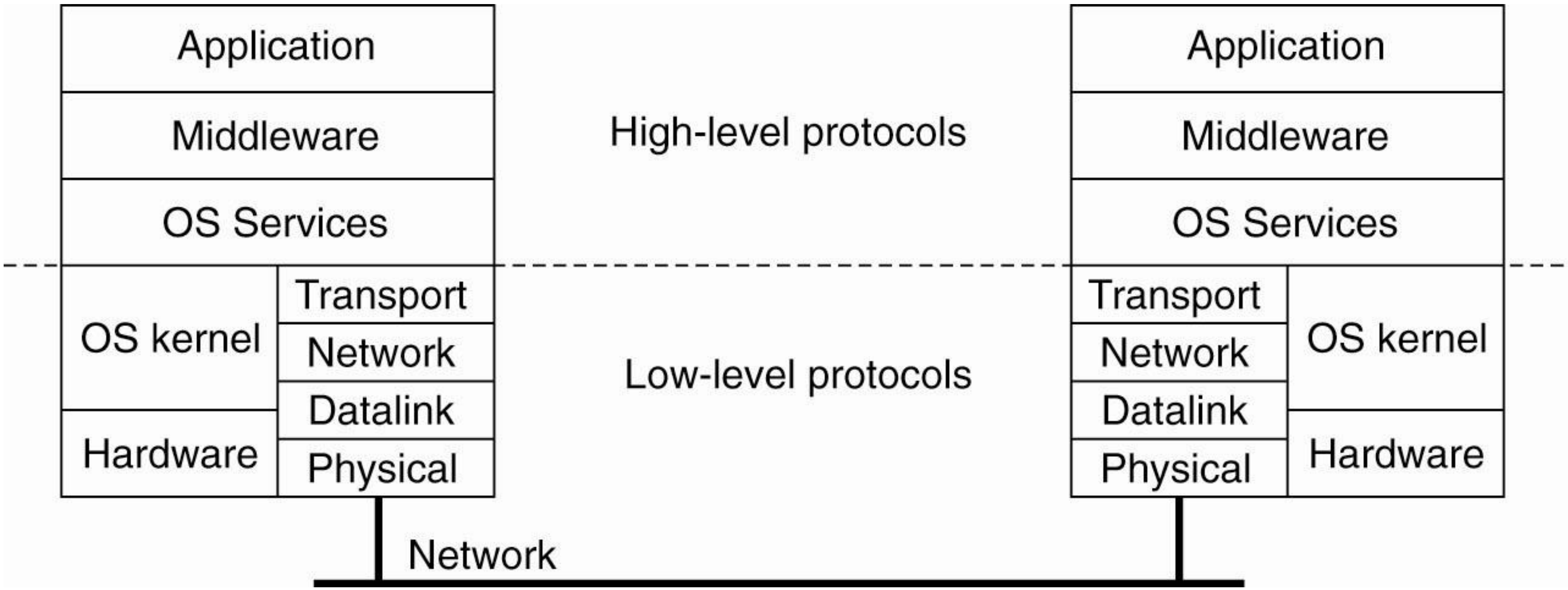
INTRODUCTION

General Concepts of Security in Distributed Systems

- Two main areas:
  - How to establish a **secure channel** between users / processes across process and machine borders?
  - How to **authorize** users and processes?

- Threats:
  - Interception
  - Interruption
  - Modification
  - Fabrication

Layering of Security Mechanisms (1)



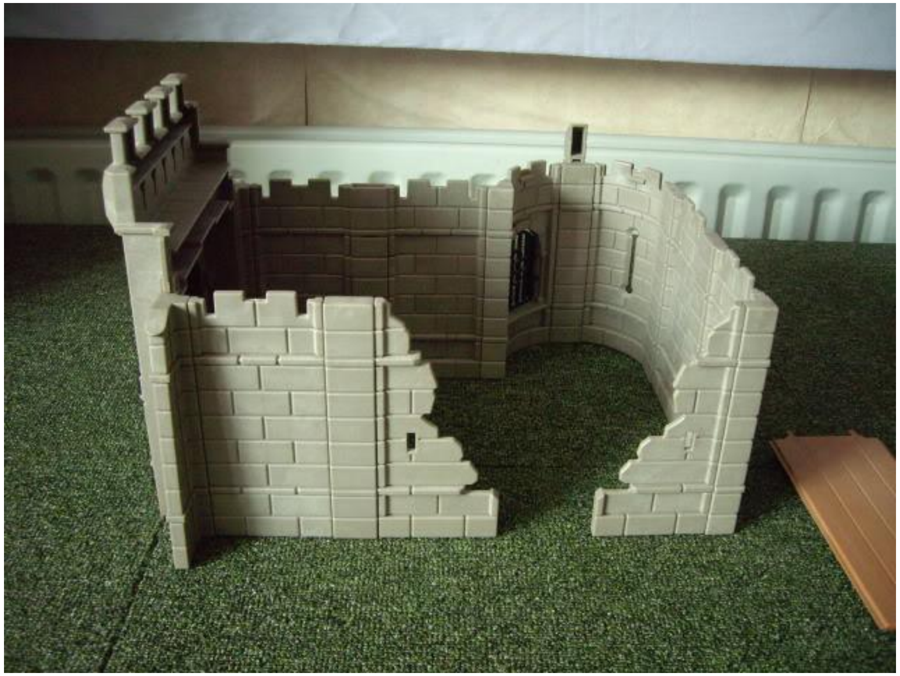
Source: Andrew S. Tanenbaum and Maarten van Steen, Distributed Systems – Principles and Paradigms, 2nd Edition, 2007, Prentice-Hall

Layering of Security Mechanisms (2)

- Considerations:
  - Security on a lower layer is often more convenient
  - Security on a higher layer may allow secure communication over an otherwise insecure channel
    - E.g., SSL / TLS (secure communication over insecure TCP via an Transport Layer protocol)

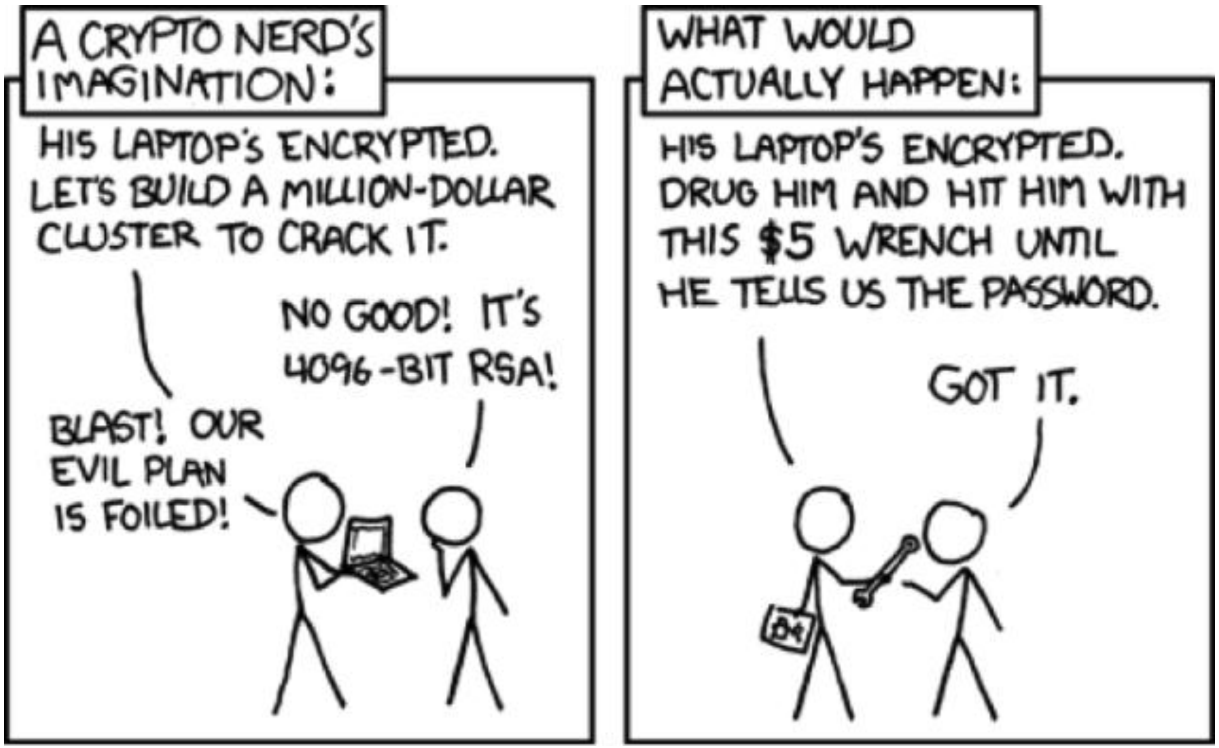
Fundamental Laws of Security (1)

- The security of any distributed system is **exactly** as good as its weakest component.



Fundamental Laws of Security (2)

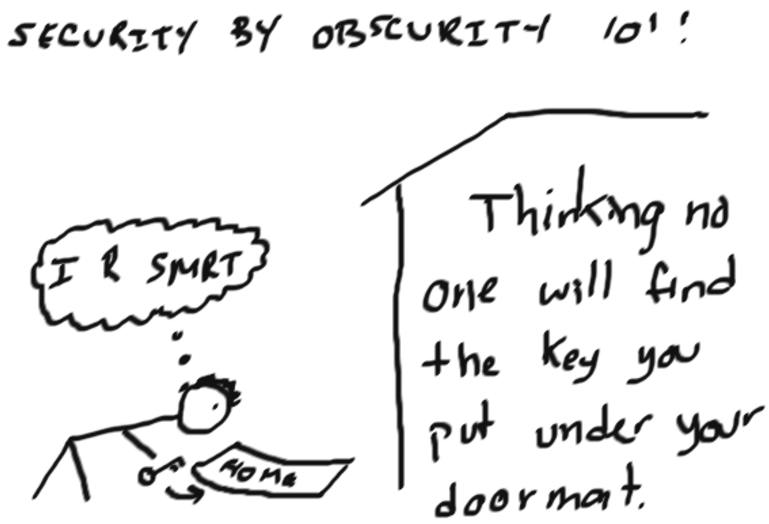
- This weakest component is typically the **human** in the loop.



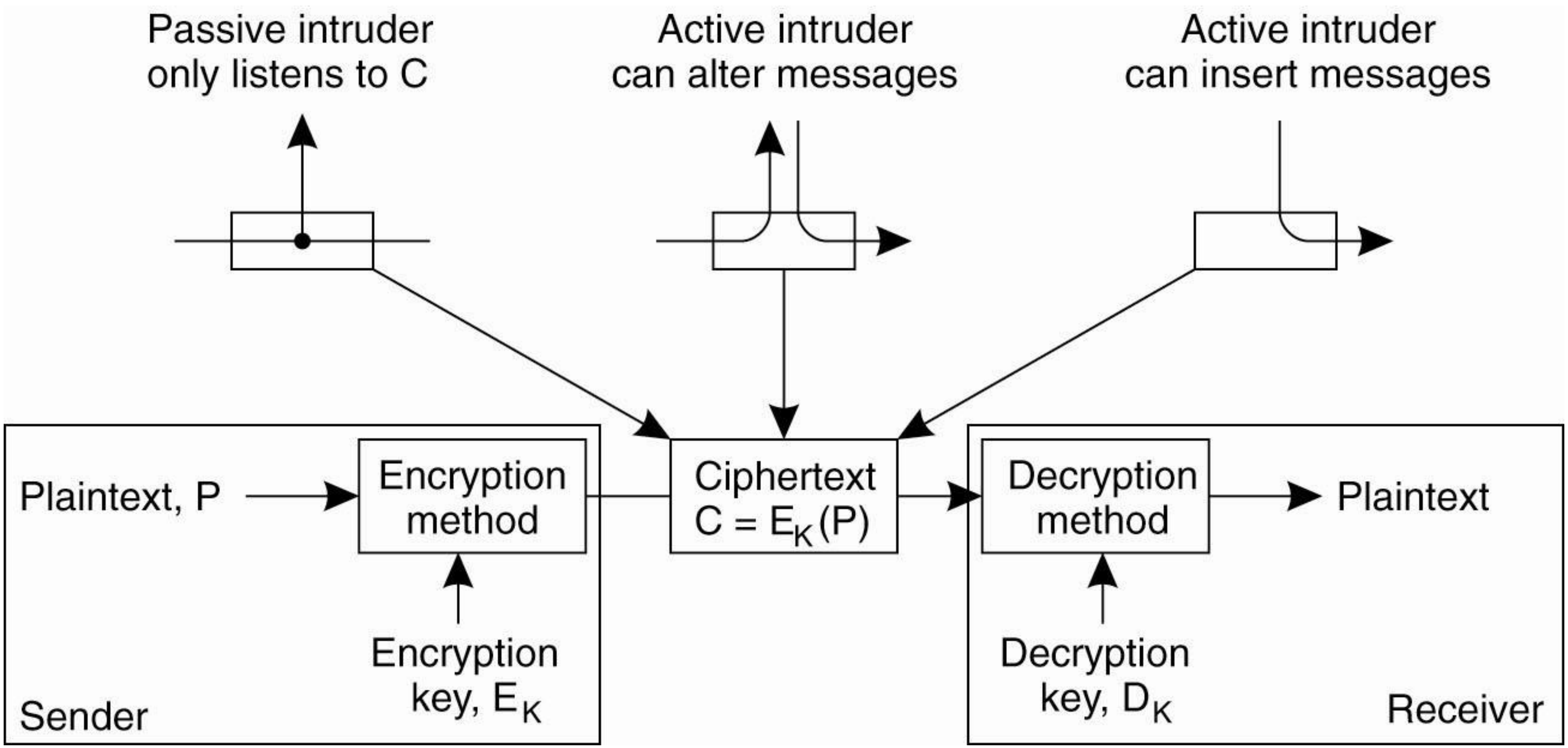
Distributed Systems Group

Fundamental Laws of Security (3)

- The security of your system needs to depend on technical and mathematical facts, and never on **hidden information**.



Cryptography (1)



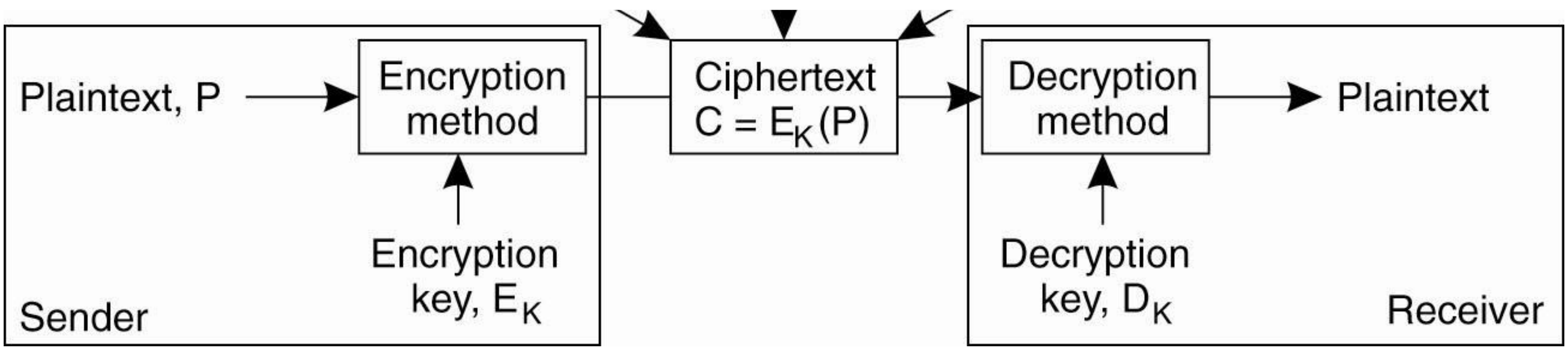
Source: Andrew S. Tanenbaum and Maarten van Steen, Distributed Systems – Principles and Paradigms, 2nd Edition, 2007, Prentice-Hall

Cryptography (2)

P = ‘Hello World’

C = ‘xYYnuYY542b’

P = ‘Hello World’



Types of Cryptographic Methods (1)

- **Symmetric** cryptosystems
  - The same key is used as encryption key E_K and decryption key D_K
  - E.g., DES, AES

- **Asymmetric** cryptosystems
  - E_K and D_K differ, but form a pair
  - Other common name: **public-key** cryptosystem
  - E.g., RSA

Types of Cryptographic Methods (2)

- **Hashing** cryptosystems
  - Basically encryption method where no decryption key D_K exists
  - E.g., MD5, SHA-1

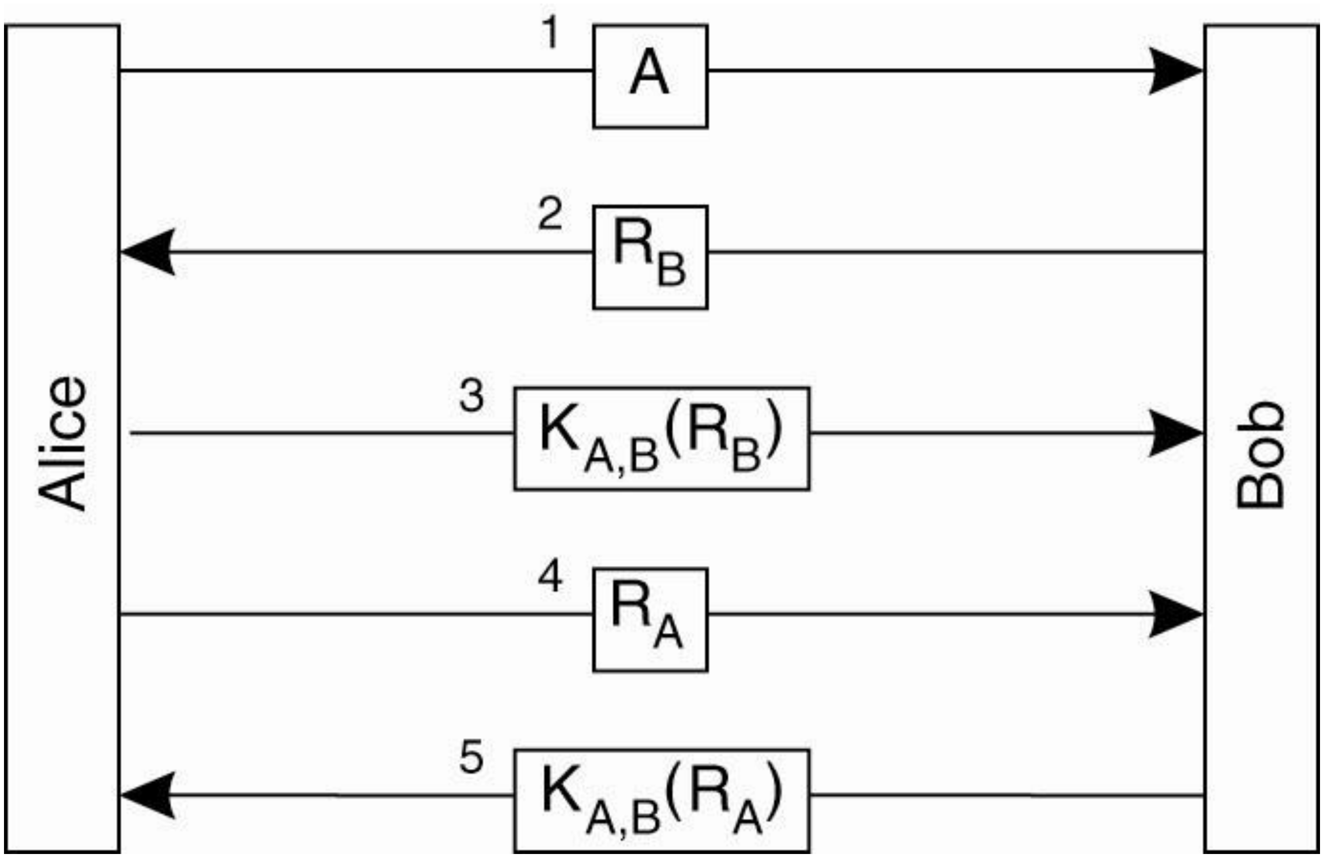
  - **Properties:**
    - **One-way functions**
      - Given a hash value, it is infeasible to find the original value
    - **Weak collision resistance**
      - Given a hash and an original value, it is infeasible to find another original value that leads to the same hash
    - **Strong collision resistance**
      - It is infeasible to find two original values that lead to the same hash

Typical Use Cases

- Symmetric cryptosystems
  - Encryption (prevention of **interception**)
- Asymmetric cryptosystems
  - Authentication (prevention of **fabrication**)
- Hashing cryptosystems
  - Integrity (prevention of **modification**)

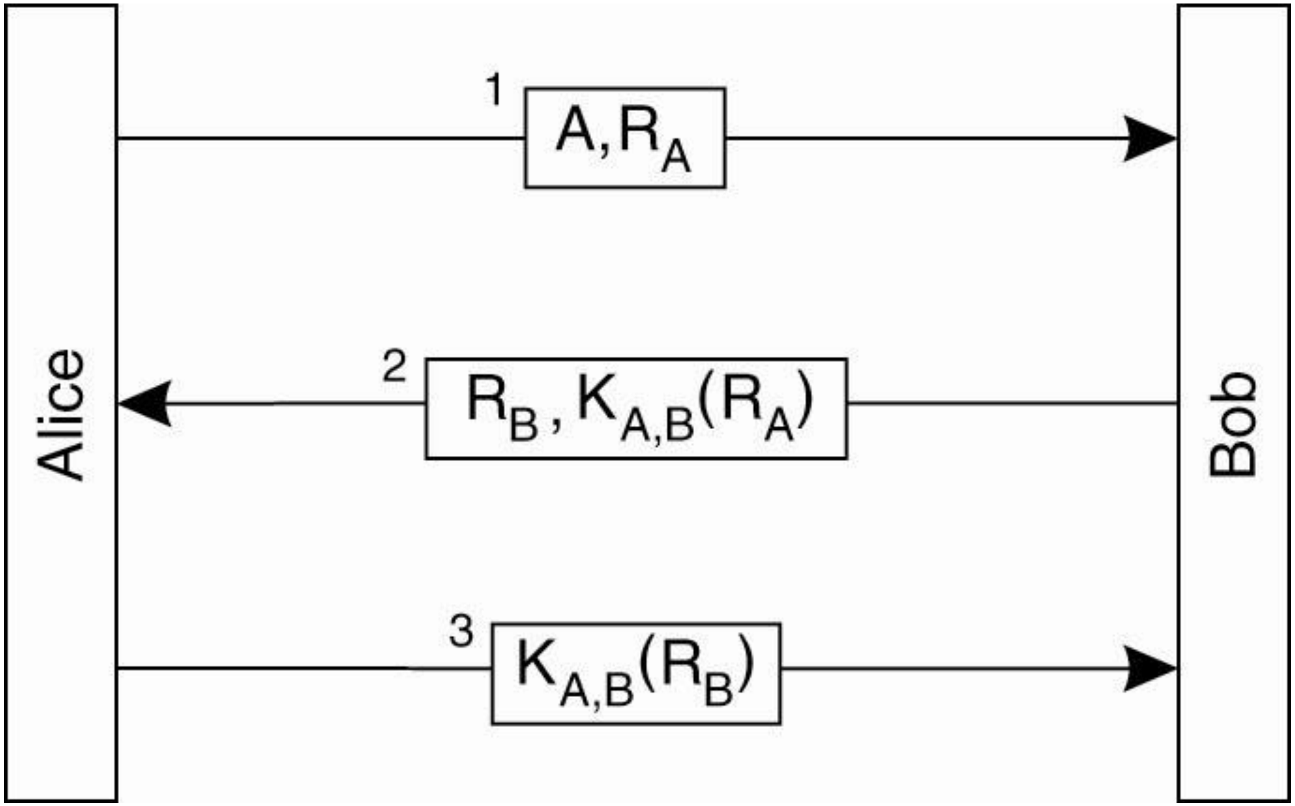
AUTHENTICATION

Authentication Based on a Shared Secret Key



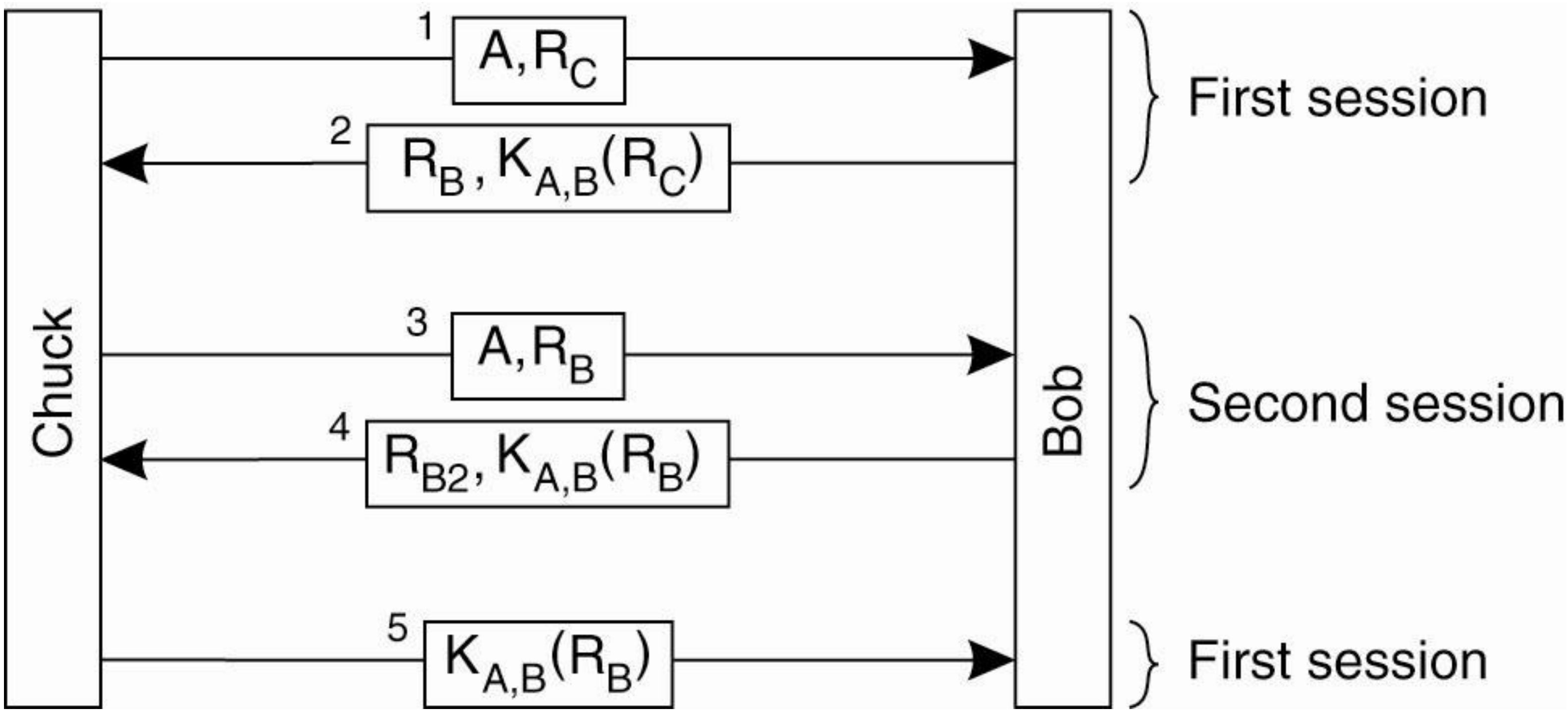
Source: Andrew S. Tanenbaum and Maarten van Steen, Distributed Systems – Principles and Paradigms, 2nd Edition, 2007, Prentice-Hall

An “Optimized” Version of This Protocol?



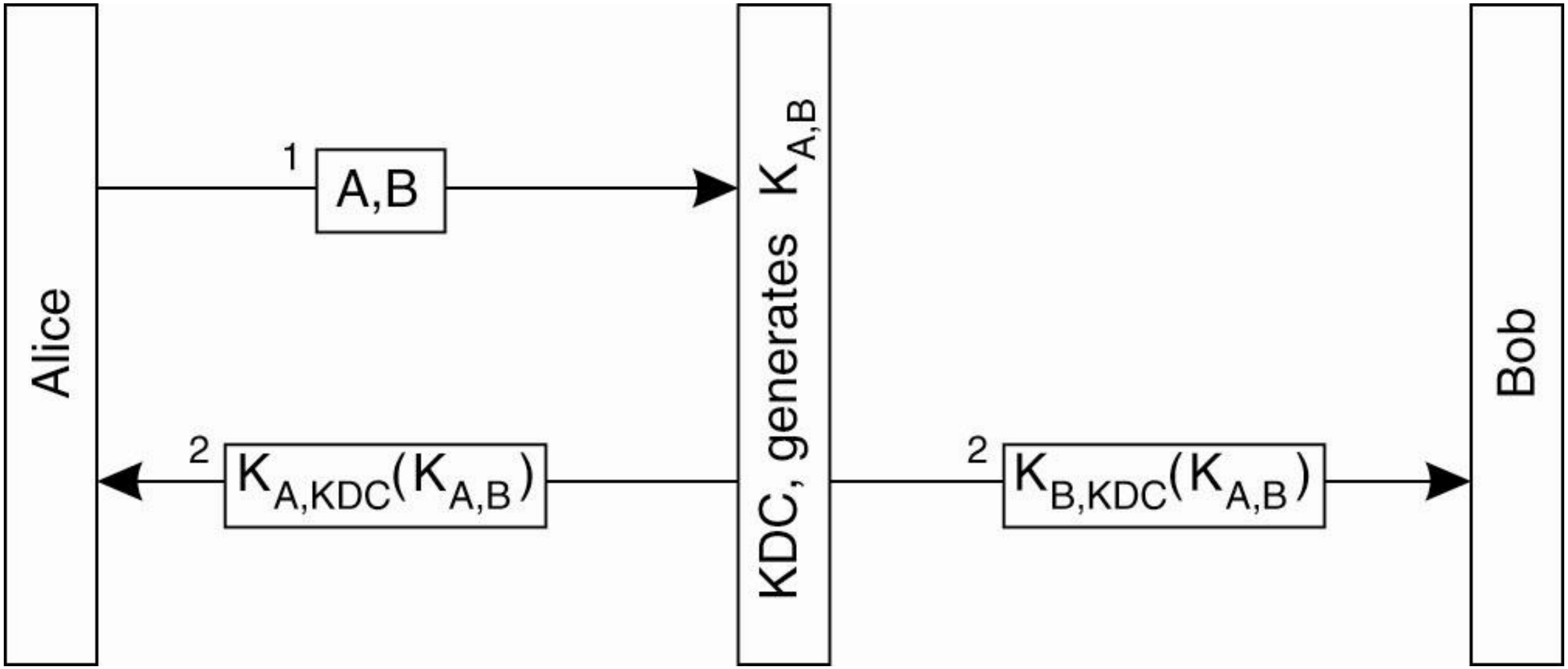
Source: Andrew S. Tanenbaum and Maarten van Steen, Distributed Systems – Principles and Paradigms, 2nd Edition, 2007, Prentice-Hall

Reflection Attack



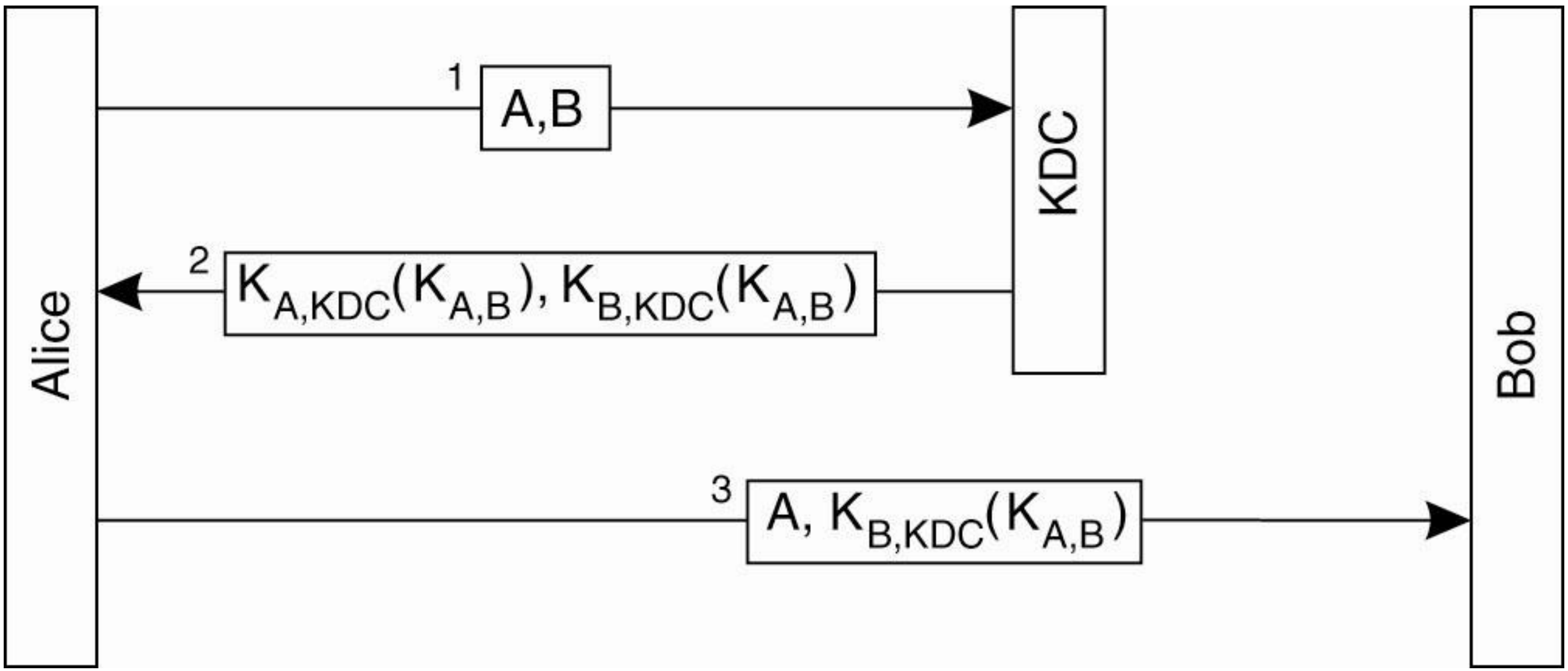
Source: Andrew S. Tanenbaum and Maarten van Steen, Distributed Systems – Principles and Paradigms, 2nd Edition, 2007, Prentice-Hall

Key Distribution Center



Source: Andrew S. Tanenbaum and Maarten van Steen, Distributed Systems – Principles and Paradigms, 2nd Edition, 2007, Prentice-Hall

Authentication using a Key Distribution Center

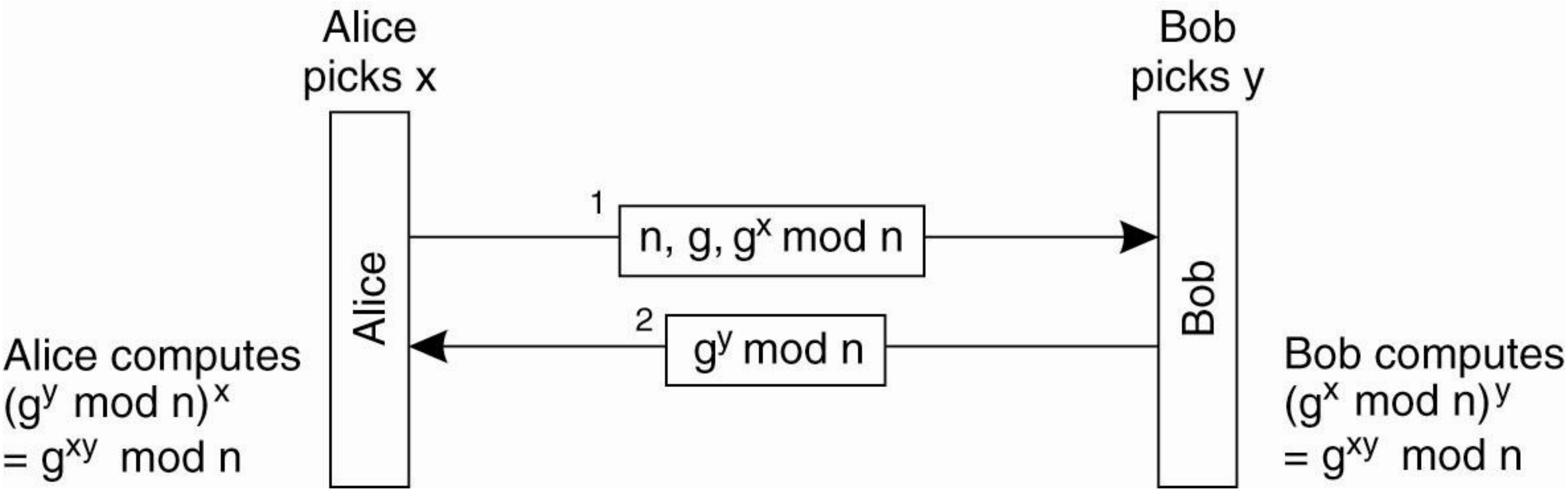


Source: Andrew S. Tanenbaum and Maarten van Steen, Distributed Systems – Principles and Paradigms, 2nd Edition, 2007, Prentice-Hall

Session Keys

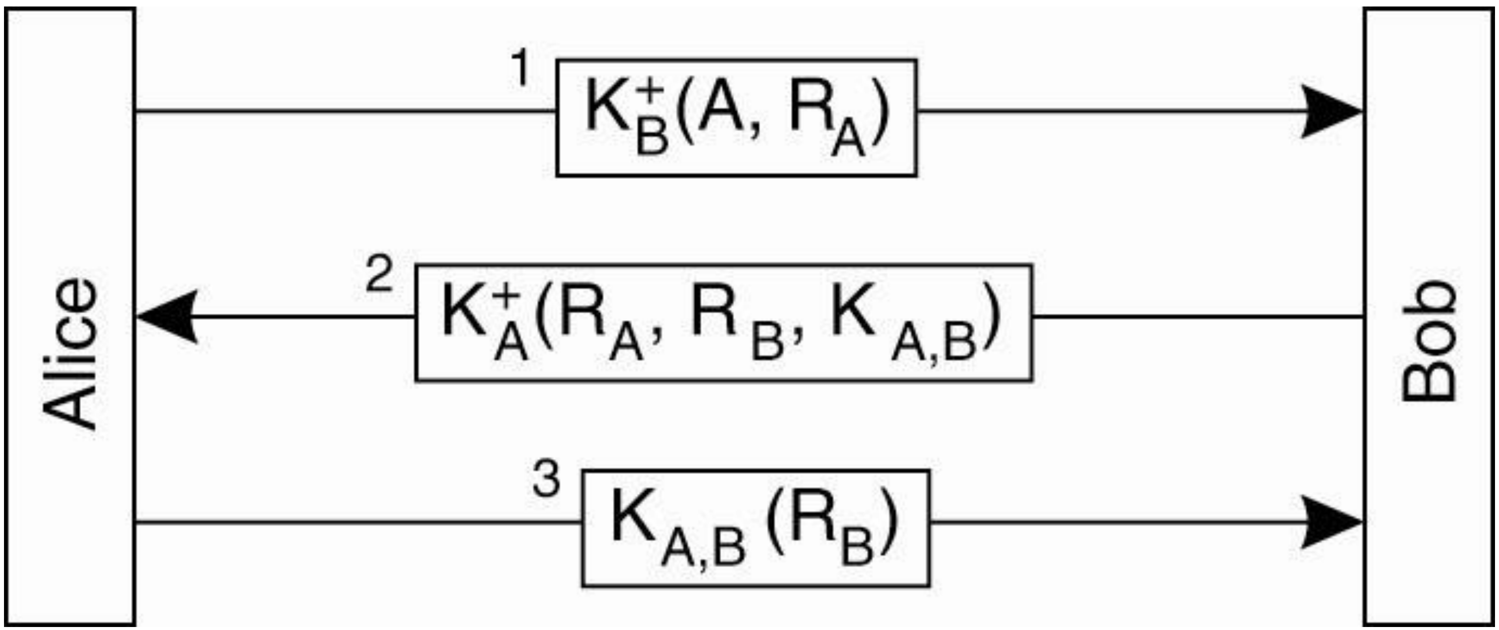
- Asymmetric cryptosystems are **computationally expensive**
  - → generally infeasible to encrypt everything using public keys
- Hence standard approach
  - Authenticate participants via public keys
  - Negotiate a shared one-time **session key**
  - Communicate using symmetric encryption using the session key
    - E.g., used in SSL/TLS

Diffie-Hellman



Source: Andrew S. Tanenbaum and Maarten van Steen, Distributed Systems – Principles and Paradigms, 2nd Edition, 2007, Prentice-Hall

Authentication using Asymmetric Cryptosystems (1)



Source: Andrew S. Tanenbaum and Maarten van Steen, Distributed Systems – Principles and Paradigms, 2nd Edition, 2007, Prentice-Hall

Authentication using Asymmetric Cryptosystems (2)

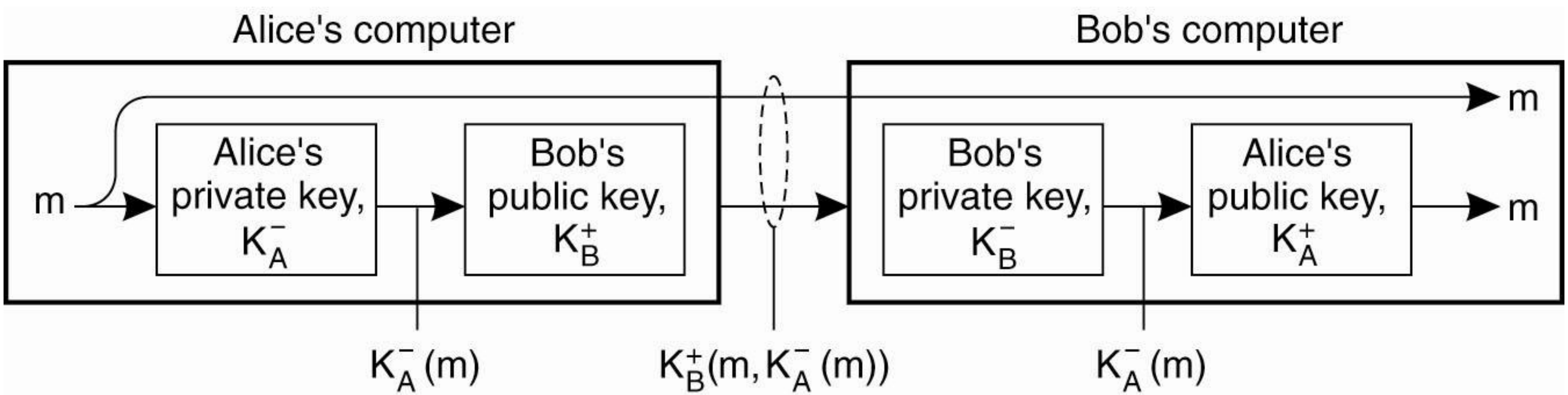
- How do participants learn about $K_{[A|B]}^{+}$?
- Typically:
  - **Certification Authorities**
  - Trusted entities that generate public-private keypairs and validate the public key – participant mapping

Key Lifecycle Management

- Keys in a cryptosystem exhibit some **wear-and-tear**
  - Risk of compromisation increases with duration of usage
    - Key expiration
  - In the real world, some keys get compromised anyway (loss, security breaches, …)
    - Key revocation (e.g., **Certificate Revocation Lists**, CRLs)

INTEGRITY

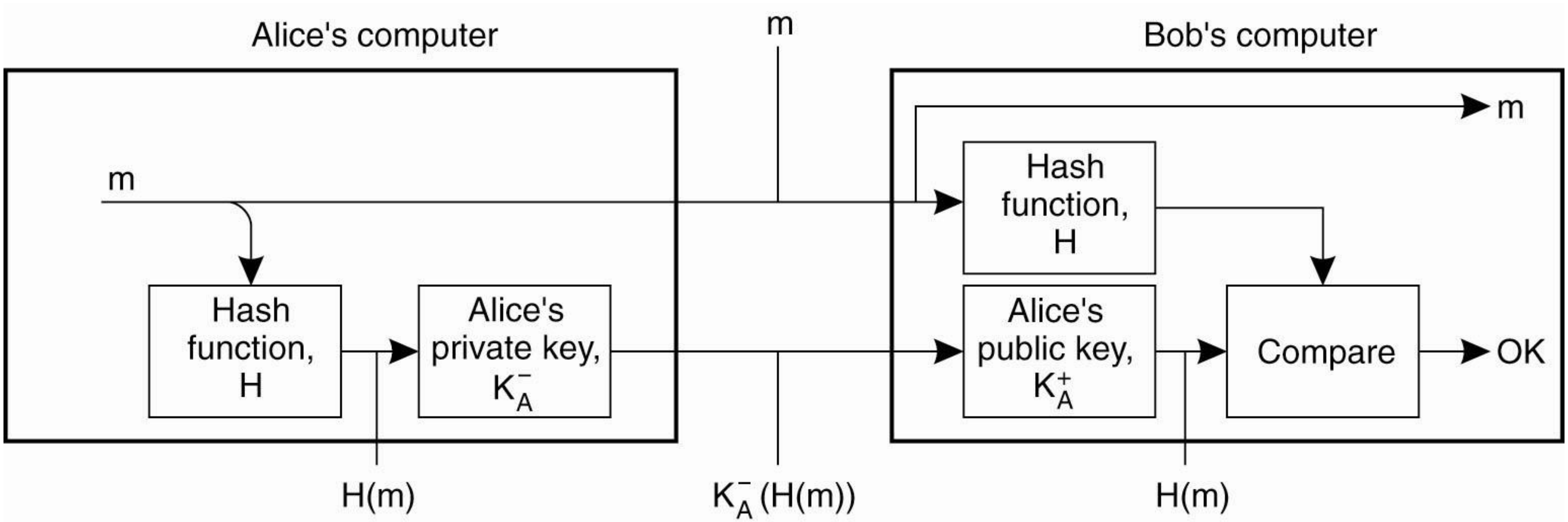
Signing a Message using Asymmetric Cryptosystems



Source: Andrew S. Tanenbaum and Maarten van Steen, Distributed Systems – Principles and Paradigms, 2nd Edition, 2007, Prentice-Hall

Distributed Systems Group

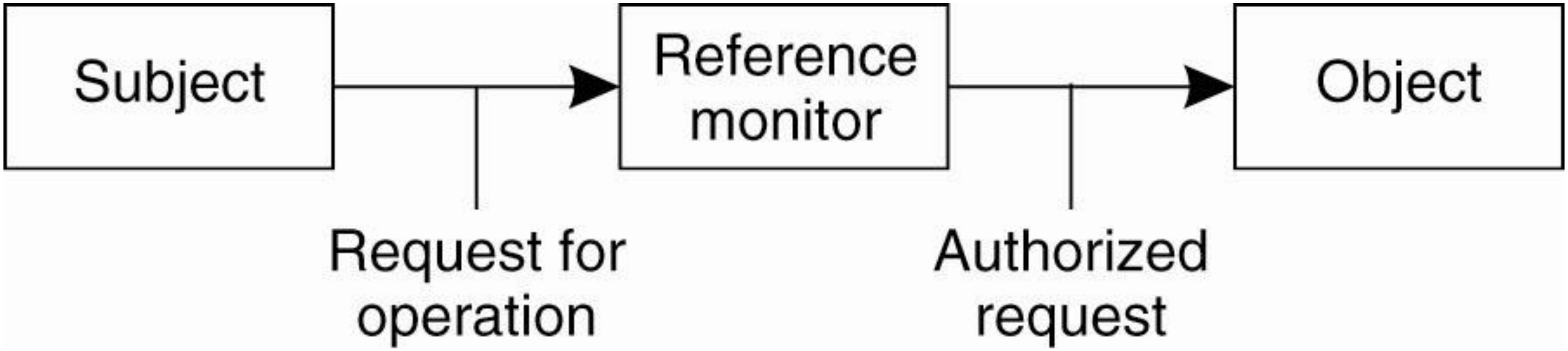
Signing a Message using Digital Signatures



Source: Andrew S. Tanenbaum and Maarten van Steen, Distributed Systems – Principles and Paradigms, 2nd Edition, 2007, Prentice-Hall

ACCESS CONTROL

General Idea of Access Control



Source: Andrew S. Tanenbaum and Maarten van Steen, Distributed Systems – Principles and Paradigms, 2nd Edition, 2007, Prentice-Hall

- Approaches
  - Access Control Matrices
  - Access Control Lists
  - Protection Domains

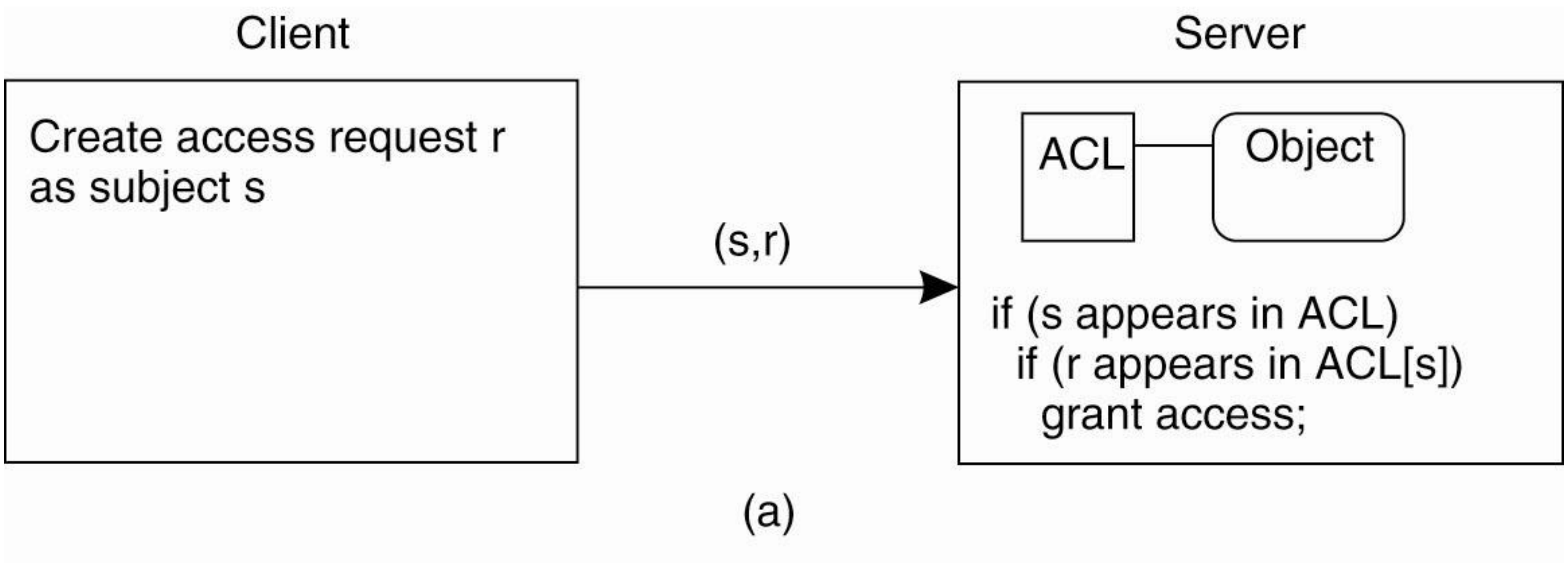
Access Control Matrices

	Object X	Object Y	Object Z
Alice	rw+	rw	r
Bob	r	r	rw+
Chuck	-	-	-

- Access levels (typically):
  - RW+ (read / write / administer rights)
  - RW (read / write)
  - R (read)
  - - (none)

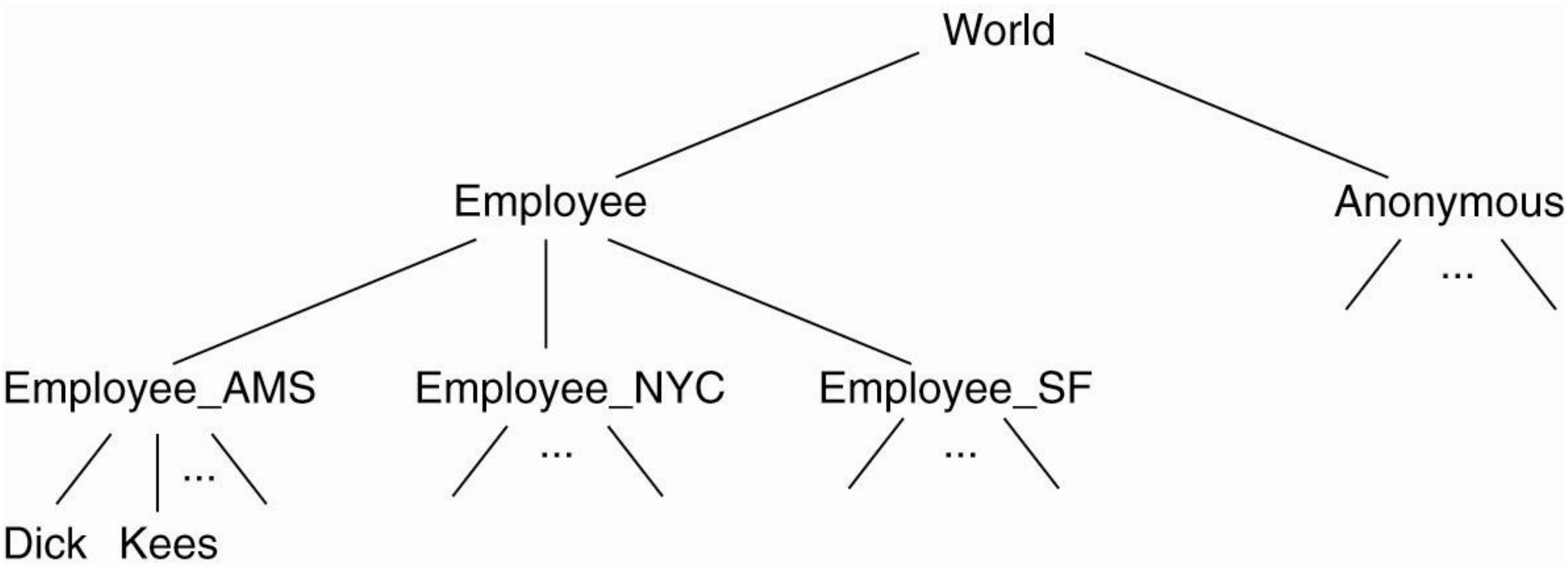
Access Control Lists (ACLs)

Object X:	Alice: rw+	Bob: r	Chuck: -
Object Y:	Alice: rw	Bob: r	Chuck: -
Object Z:	Alice: r	Bob: rw+	Chuck: -



Source: Andrew S. Tanenbaum and Maarten van Steen, Distributed Systems – Principles and Paradigms, 2nd Edition, 2007, Prentice-Hall

Protection Domains



Source: Andrew S. Tanenbaum and Maarten van Steen, Distributed Systems – Principles and Paradigms, 2nd Edition, 2007, Prentice-Hall

SOME COMMON ATTACK SCENARIOS

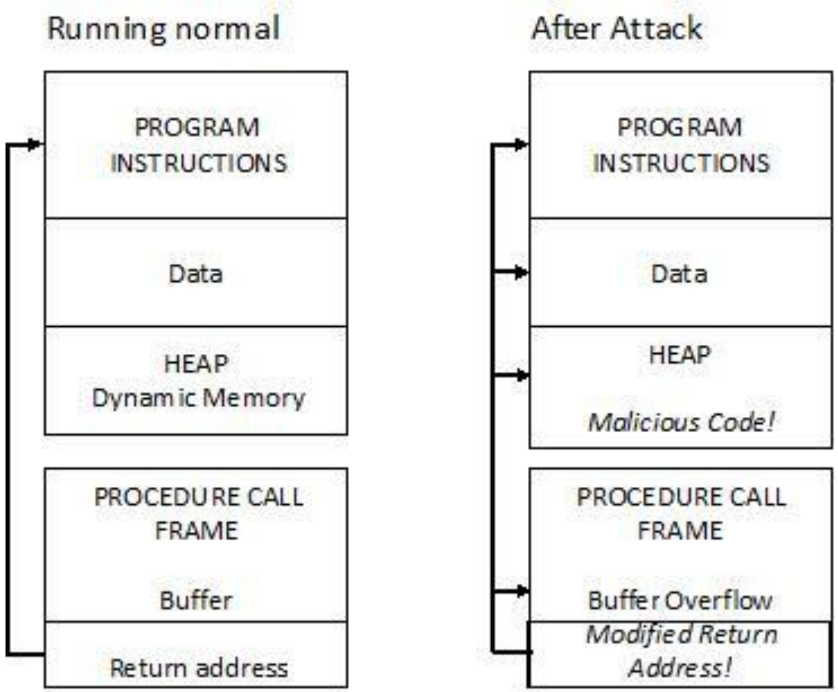
- Distributed systems security can be compromised on any layer
  - → and remember: any security breach potentially renders the entire system insecure

- The following is just a small set of examples of what can go wrong
  - All of the following things happen in practice **all the time**

Buffer Overflows (1)

- Common security problem in unmanaged programming languages (e.g., C / C++)
  - Input data larger than reserved heap space
  - Hence data ***flows over*** into next frame, allowing an attacker to overwrite the return address pointer of a procedure call with a custom address
  - Hence allowing the attacker to execute arbitrary code

Buffer Overflows (2)

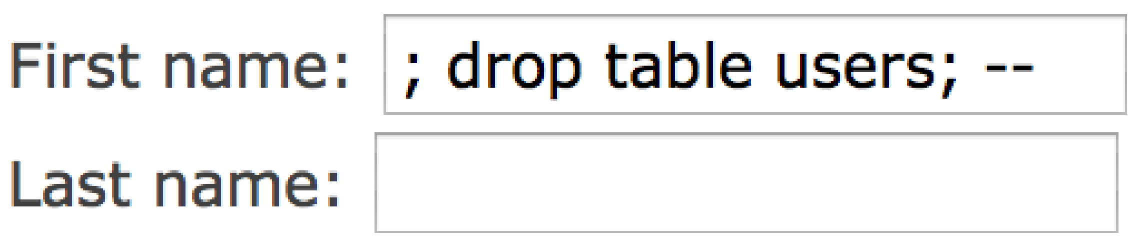


Attacker plants code that overflows buffer and corrupts the return address. Instead of returning to the appropriate calling procedure, the modified return address returns control to malicious code, located elsewhere in process memory.

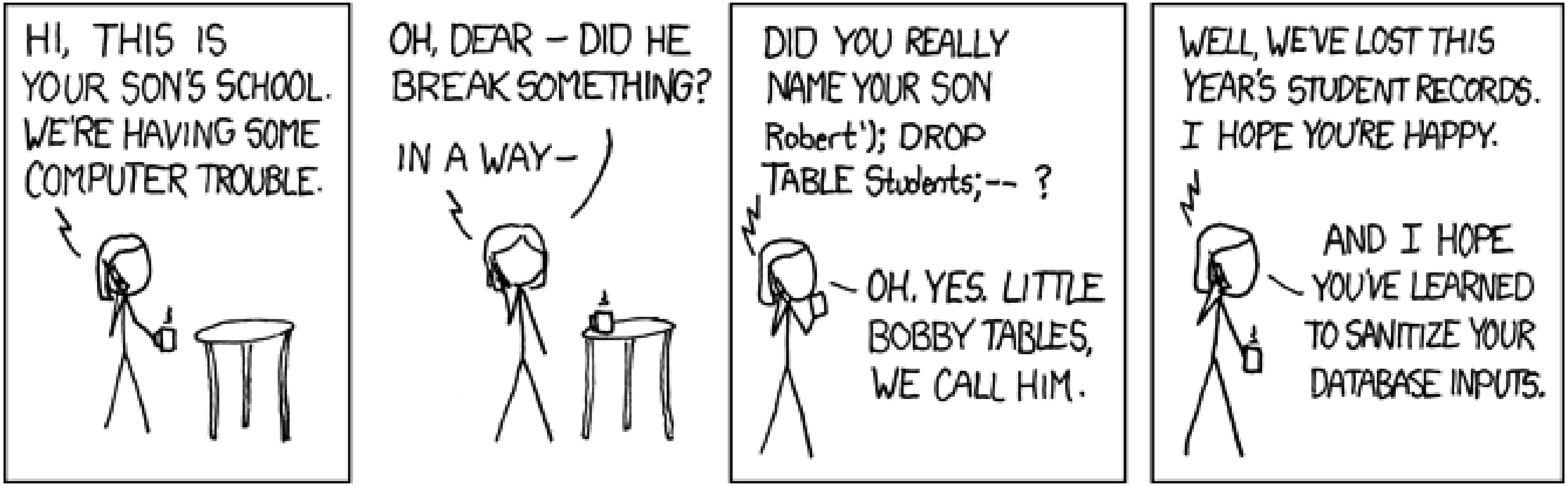
Source: http://cis1.towson.edu/~cssecinj/modules/cs2/buffer-overflow-cs2-c/

SQL Injection Attack (1)

- Some web applications do not sufficiently check data received from users before issuing SQL queries
  - `select * from users where user = $username and pw = md5($pw)`
  - → now assume e.g., `$username = '; drop table users; --'`
- Example:



SQL Injection Attack (2)

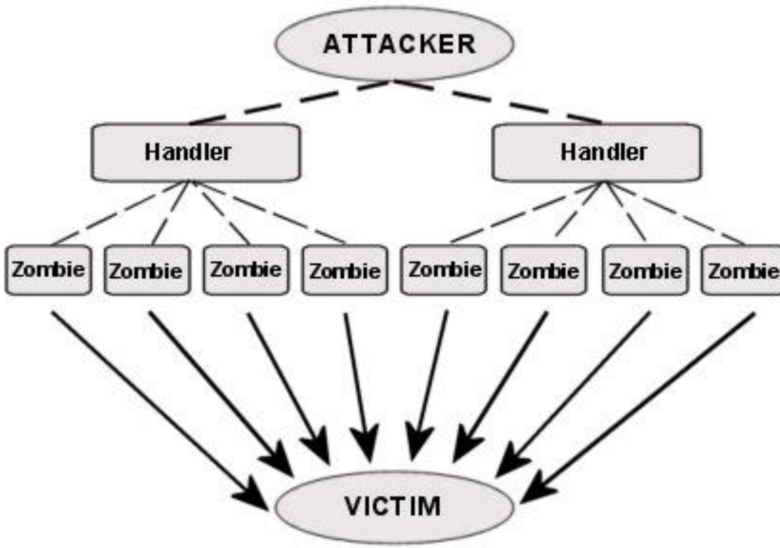


Cross-Side Scripting Attack (XSS)

- Similar principle to SQL injection
- Allows attackers to **inject arbitrary scripts** into a legit (trustable) web site
  - Example: assume you have a blog with a comment function. The comment function accepts arbitrary HTML code.

Distributed Denial-of-Service Attack (DDos)

- Attacker uses a network of hacked machines (bots, zombies) to **overload** the resources of the target with requests
  - Produces costs and load
  - Server crashes



- Difficult to protect against
- Difficult to identify the attacker
  - All requests come from unassuming zombies

Sidechannel Attacks

- Attacks that ignore the technical security mechanisms by finding out the **secret** that the mechanism was based on

- E.g.:
  - Phishing attackers steal passwords or keys
  - NSA demanding private keys from certification authorities
  - Hackers reverse-engineering keys in embedded devices by measuring energy consumption

Social Engineering

- Catch-all term for various sidechannel attacks that target the **human** behind the (secure?) technical system
- Usual assumption: people are easily manipulated by a well-prepared talker
- Most common: **Phishing**
  - E.g., calling a user and convincing her/him to tell you his account data (*“Hey, I’m Joe from IT. I understand you have had troubles logging in today?*)

FURTHER LECTURES

- Internet Security
  - https://tiss.tuwien.ac.at/course/courseDetails.xhtml?windowId=faa&courseNr=188366&semester=2014S
- Advanced Internet Security
  - https://tiss.tuwien.ac.at/course/courseDetails.xhtml?windowId=faa&courseNr=183222&semester=2013W
- Organizational Aspects of IT Security
  - https://tiss.tuwien.ac.at/course/courseDetails.xhtml?windowId=faa&courseNr=188312&semester=2013W
- Seminar aus Security
  - https://tiss.tuwien.ac.at/course/courseDetails.xhtml?windowId=faa&courseNr=183606&semester=2013W