CS4103-DS: Security

Jan de Muijnck-Hughes

11 April 2016, 12 April 2016, 19 April 2016

Section 1

Overview

Aims & Objectives

- Gain an understanding of salient issues surrounding Security and Distributed Systems.
- Understand the issues associated with authorisation within a Distributed System, and ways in which it can be addressed.
- Understand issues associated with authentication, and how cryptographic techniques can be used to provide authentication mechanisms.

Reading

- Andrew Tanenbaum et al. *Distributed Systems: Principles and Paradigms*. English. 3rd ed. Pearson Higher Education, 2013, p. 633. ISBN: 1292025522,
  Chp. 9:§9.1-2, §9.2.1-2&4 §9.3.1, §9.4.1&3. §9.5
- George Coulouris et al. *Distributed Systems: Concepts and Designs*. English. 5th ed. Pearson Higher Education, 2011, p. 927. ISBN: 0273760599, Chp. 11:§11.1, §11.6.1&2
- Yu Zhou et al. 'Policy Enforcement Pattern'. In: *PLoP 2002*. 2002

Security as Risk Management

Doing Security $\equiv$ Risk Management

- Asset identification
- Risk identification
  - Identifying an asset's vulnerabilities
  - Identifying relevant threats
- Risk analysis
- Risk treatment

Threat Manifestation aka Risk

$$\text{Risk} \leftarrow \text{Threat} + \text{Vulnerability} = \text{Success}$$

Threat

- Circumstances that have potential to cause loss or harm to the asset
- Threats can be:
  - **accidental**
  - **deliberate**
  - **environmental**

Vulnerability

- Weakness that can be exploited within a system
- Vulnerabilities can be:
  - **accidental**
  - **deliberate**
  - **environmental**

ISO Threat Types

- **Physical damage**
  - fire, water, dust
- **Natural events**
  - weather, volcanic activity
- **Loss of essential services**
  - loss of power
- **Disturbance due to radiation**
  - electromagnetic, thermal
- **Compromise of information**
  - Eavesdropping, Remote Spying
- **Technical failures**
  - equipment or software malfunction
- **Unauthorised Actions**
  - illegal processing of data, using pirated software
- **Compromise of functions**
  - Abuse of rights, Denial of Actions

Where can Vulnerabilities Occur?

- **Hardware**
  - environmental damage, wear and tear
- **Software**
  - well-known flaws, insufficient testing
- **Network**
  - single point of failure, unprotected comm lines
- **Personnel**
  - lack of personnel, insufficient training
- **Site**
  - located in flood plain, unstable power grid
- **Organisational**
  - lack of continuity plans, lack of email usage policy

Security Policies & Mechanisms

Policies
Describes the actions that an 'entity' are permitted to do, and not to. Essentially, security requirements: Confidentiality, Integrity, Availability...

- **Examples**
  - 'Only Jan & DoT can see the exam.'
  - 'STAFFRESS is only accessible by Staff members.'

Mechanisms
Technology or procedure employed to enforce the policy.

- **Examples**
  - Authentication, Authorisation, Encryption, & Auditing.

Intermezzo: Scope

- Security is a comprehensive and extensive subject area.
- Our interest for this lecture is:

  Security of Distributed Systems

  and

  Distributed Systems for Security.
- We won't cover other security topics.

Section 2

Security & Distributed Systems

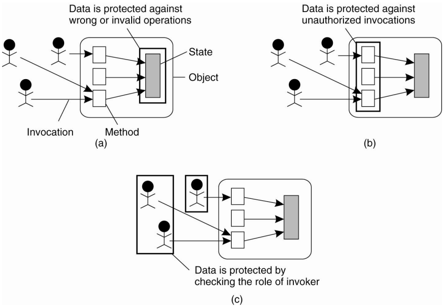
Security & Distributed Systems

How can security policies be defined and implemented over distributed resources and using what mechanisms?

Core Issues concern Identity & Access Management

1. **Data Security:**
    - How to secure data **at-rest**?
    - How to secure data **in-flight**?
2. **Identity Management:**
    - Definition and management of identities.
3. **Authentication:**
    - Authentication in distributed setting.
4. **Authorisation:**
    - Define and enforce authorisation policies.
    - Authorisation in distributed setting.

Focus of Control

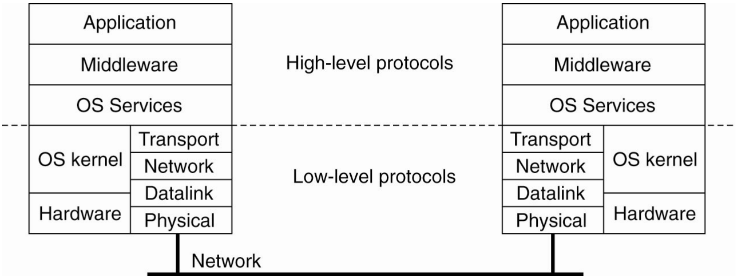


Tanenbaum et al. [TS13]

Where to focus protection?

1. **Model** Protection against invalid operations
2. **View** Protection against unauthorised invocations
3. **Controller** Protection against unauthorised users

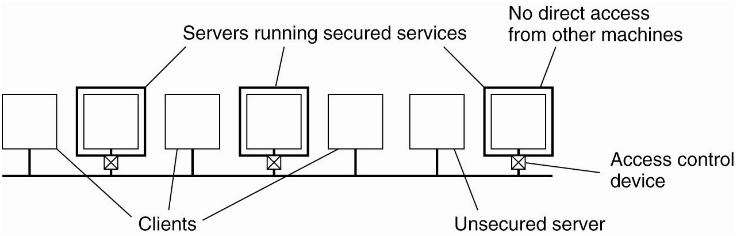
Layering of Security Mechanisms



Tanenbaum et al. [TS13]

- Protect all the layers.
- ‘Transport Layer Security is solved’

Distribution of Security Mechanisms



Tanenbaum et al. [TS13]

- Organisational & Administrative Heterogeneity.
- Trusted Computing Base
- Simplicity

Securing Data at Rest

Data at rest is data that doesn't 'move'.

- Can solve using cryptography...
  - **Secrecy with Signature with Appendix** using **KEM/DEM**
  - Public Key Encryption, Symmetric Encryption, Digital Signatures, Hash functions...
- What standards and parameters to use?

Disk, File-System	Symmetric	AES, SkipJack, Blowfish
User files, app data	KEM/DEM	RSA, DSA, ECC, ECDSA

- Key Management!?
  - Public Key Infrastructure: Centralised, Decentralised
- Expressiveness of Encryption
  - Perfect Forward Secrecy
  - Classical Schemes provide **1-2-1** Encryption.
  - Need additional mechanisms to manage permissions.

Securing Data in Flight

Data in flight is data being moved from one domain to another.

- Can solve using cryptography to construct **secure channels**
  - Send messages securely between two points: End-2-End Encryption.
- What standards and parameters to use?
  - Network Layer has IPSec
  - Transport Layer has TLS
  - Application layer has: Signal, Cryptocat, OTR...
- Key Management!?
  - Public Key Infrastructure: Centralised, Decentralised

Section 3

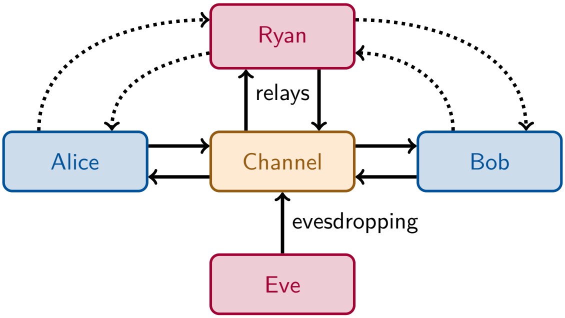
Authentication

Authentication

Given two entities Alice and Bob, how can Bob authenticate with Alice such that Alice knows that Bob is really who he says he is.

- Typically ‘Two phased’ protocols
  1. **Enrollment**: Establish credentials.
  2. **Challenge & Response**: Check validity of credentials.
- Utilise lots of cryptography.
- **Styles**
  - Direct or Brokered
- Examples
  - Network: HIP, IPSec, ILNP
  - Transport: MS-CHAP, EAP
  - Application: RADIUS, DIAMETER
  - User: SAML, OpenID, .Net Passport, KERBEROS, Shibboleth, OpenAthens

Secure Channel



- Use **Secrecy with Signature with Appendix** using **KEM/DEM**
  - Session Keys for each conversation.
  - Public Key Infrastructure to get public keys.
- How to exchange session keys?
  - Diffe-Hellman Key Exchange, Station-To-Station, Needham-Schroeder-Lowe
- How to authenticate Bob?

Kerberos

Authentication protocol using **tickets** to allow nodes to authenticate over an untrusted network.

- Developed by MIT.
- Requires a **Trusted-Third Party**
    - Authentication Service
    - Ticket Granting Service
- Mutual Authentication
- ‘Dated’

Intermezzo: Crypto Notation

Key Notation

Symmetric Key	K_{AB}	**Signing Key**	$\mathsf{Enc}_{priv}(Alice)$
Public Key	$\mathsf{Enc}_{pub}(Bob)$		
Private Key	$\mathsf{Dec}_{priv}(Bob)$	**Verifying Key**	$\mathsf{Dec}_{pub}(Alice)$

Operations

Encrypt	$\mathsf{Encrypt}(\ldots)$	**Sign**	$\mathsf{Sign}(\ldots)$
Decrypt	$\mathsf{Decrypt}(\ldots)$	**Verify**	$\mathsf{Verify}(\ldots)$

Misc

Ctxt Sym	$\{M\}_{\mathsf{K}_{Bob}}$	**Ctxt ASym**	$\{\vert M\vert\}_{\mathsf{Enc}(Bob)}$
Hash	$\#(msg)$	**Send A to B**	$A \rightarrow B : msg$
Concatenate	$A \mid\mid B$	**Assignment**	$H_{msg} \leftarrow \#(msg)$

Authentication Protocol

Simplified Kerberos protocol to talk to Bob.

- **Sign into Service**
  - Session Key $\mathsf{K}_{A,AS}$ Established
  - $Alice \rightarrow AS : \mathrm{ID}(A)$
  - AS generates
    - ticket with TTL: $\mathcal{T}_{ttl} \leftarrow \{\mathrm{ID}(A) \mid\mid \mathsf{K}_{A,TGS}\}_{\mathsf{K}_{AS,TGS}}$
    - Session Key $\mathsf{K}_{A,TGS}$
  - $AS \rightarrow Alice : \{\mathsf{K}_{A,TGS} \mid\mid \mathcal{T}_{ttl}\}_{\mathsf{K}_{A,AS}}$
- **Request Ticket to Talk to Bob**
  - Session Key $\mathsf{K}_{A,TGS}$ Established
  - Timestamp t.
  - $A \rightarrow TGS : \mathcal{T}_{ttl} \mid\mid \mathrm{ID}(B) \mid\mid \{t\}_{\mathsf{K}_{A,TGS}}$
  - TGS Generates Session Key $\mathsf{K}_{A,B}$ and obtains $\mathsf{K}_{B,TGS}$.
  - $TGS \rightarrow A : \{\mathrm{ID}(B) \mid\mid \mathsf{K}_{A,B}\}_{\mathsf{K}_{A,TGS}} \mid\mid \{\mathrm{ID}(A) \mid\mid \mathsf{K}_{A,B}\}_{\mathsf{K}_{B,TGS}}$
- **Ask Bob To Talk**
  - $A \rightarrow B : \{\mathrm{ID}(A) \mid\mid \mathsf{K}_{A,B}\}_{\mathsf{K}_{B,TGS}} \mid\mid \{t\}_{\mathsf{K}_{A,B}}$
  - $B \rightarrow A : \{t+1\}_{\mathsf{K}_{A,B}}$

Kerberos cont. . .

- **Authentication Protocol**
  - Based on Needham-Schoeder-Lowe
- **'Single-Sign-On'**
  - By authenticating with the AS get timed access (24hrs) to system.
  - Ticket used to request access to other services i.e. other bobs.
  - Combine with Authorisation services
- **'Simplified'**
  - Introduce Public Key variants
  - Don't see sending $\{|\{\mathrm{ID}(A) \mid\mid \mathrm{K}_{A,B}\}_{\mathrm{K}_{B,TGS}} \mid\mid \{t\}_{\mathrm{K}_{A,B}}|\}_{\mathrm{Enc}(B)}$

Advantages & Disadvantages

- Advantages
  - Authentication in a Distributed System
  - Single-Sign-On
- Disadvantages
  - Single Point of Failure
  - Not federated
  - ‘Dated’
  - Not a cool protocol...

Section 4

Authorisation

Authorisation/Access Control

Granting access rights to a subject for resources in various environments, and ensuring that a subject has the correct permissions to access a particular resource in an particular environment.

- **Access Control Models**
  - Access Control Matrix
  - Access Control Lists, Capabilities
  - Role-Based Access Control, Attribute-Based Access Control, Policy-Based Access Control
- **Implementations**
  - POSIX, Capsicum, XACML, SAML, Kerberos, Shibboleth, OpenID, OAuth, Facebook Connect

General Model

- **Subjects** are: nodes, processes, users. . .
- **Objects** are: files, data, databases, services. . .
- **Permissions** are actions on objects: Read, Write, Execute. . .
  - Permissions can also be time dependent.
- **Monitor** is access control mechanism to enforce permissions.
- **Schema** is a description of an instance of an access control model for a particular scenario.

Access Control Matrix

Matrix where rows denote subjects, columns denote objects, and cells the permissions that the subject has on an object.

	Slides	**Exam**	**STAFFRES**
Jan	RWX	RWX	RWX
DoT	R-X	RWX	RWX
Bob	R-X	—	—

- Common way to envisage access control.
- Monitor ‘just’ performs matrix look up.
- If Subject s or Object o not in Matrix M then failure.
- Unwieldy for large models

Access Control Lists

Each object carries a description of the subjects and their permissions.

Jan	RWX
DoT	R-X

- Classic Approach found in most OS.
  - Column Spans of a matrix.
- Monitor/Object **needs to know** who can do what
- ‘Centralised’ Approach.

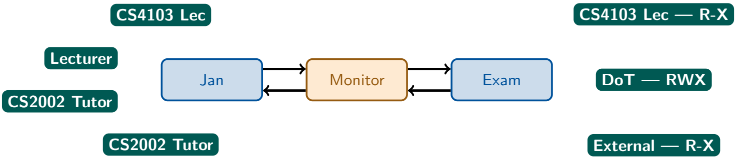
Capabilities

Each subject carries a description of the objects they can access and their assigned permissions.

Exam	RWX
...	

- Row Span of a Matrix
- Each client is given restricted list of abilities on objects.
- Monitor checks if capability can be applied.
- ‘Decentralised’ Approach.

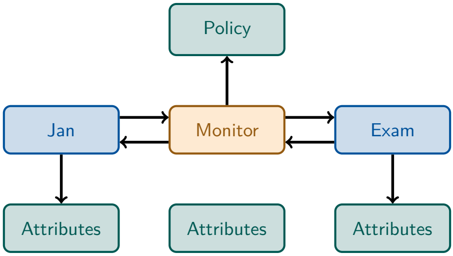
Role-Based Access Control



Application Layer Model

- Each subject has one or more (Hierarchic) Roles.
- Object has permissions based on roles.
- Monitor checks if Subject's Role allows access to Object.
- 'Decentralised' Approach.

Attribute-Based Access Control



Application Layer Model

- Attributes used to describe: **Subjects**, **Objects**, & the **Environment**.
- Policies are **Boolean Formula** over attributes.
- Monitor grants access based on policy satisfaction.
- 'Decentralised' Approach.

ABAC Example: TCPLog Access

Attributes

Subject Group, Roles, Clearance Level...

Object TCP Header Information, Ownership...

Environment Locale, Time, Date...

Access Policy

$$
\begin{aligned}
\text{Policy}(s, o, e) \leftarrow \quad & \text{Group}(s) \equiv \text{GCHQ} \\
\wedge \quad & \text{Level}(s) \geq \text{SECRECT} \\
\wedge \quad & (\text{srcPort}(o) \equiv 80 \vee \text{srcPort}(o) \equiv 8080) \\
\wedge \quad & \text{srcAddr}(o) \equiv 123.456.789 \\
\wedge \quad & \text{CurrentDate}(e) \leq 20160527 \\
\wedge \quad & \text{CurrentDate}(e) \geq 20150927
\end{aligned}
$$

XACML: eXtensible A/C Mark-up Language

Declarative access control policy language and processing model using XML to encode and evaluate policies.

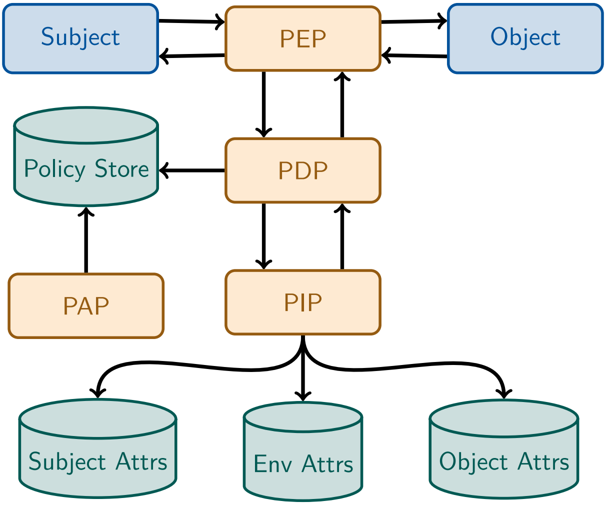
- OASIS Standard [Ris13].
- Policy Language is based on ABAC.
- ‘Policy Enforcement Points’
  - Separates decision from enforcement from definition.
  - Distributed components that work together.
- Designed for **Service Oriented Architectures**

Policy Enforcement Points

General architectural model to describe a scalable distributed authorisation framework.

- Described as a **Design Pattern** in Zhou et al. [ZZP02].
- Generalisation of AAA Framework [Vol+00]
- **Key Features**
  - Distributed components that work together.
  - Separates decision from enforcement from definition.
  - Policies are made on demand, or pre-made.
  - Policies are taken from ABAC

PEP Architecture



PEP Architecture

Taken from Rissanen [Ris13]

- **PDP** Policy Decision Point
  - The system entity that evaluates applicable policy and renders an authorization decision.
- **PEP** Policy Enforcement Point
  - The system entity that performs access control, by making decision requests and enforcing authorization decisions.
- **PIP** Policy Information Point
  - The system entity that acts as a source of attribute values.
- **PAP** Policy Administration Point
  - The system entity that creates a policy or policy set.

Section 5

Summary

Summary

- Security is **hard**; Security is a socio-technical problem.
- Four ‘core’ security issues for Distributed systems:
  - **Data Security**: In Flight, At Rest.
  - **Identity Management**: Describing and managing entities.
  - **Authentication**: Verify entities identity.
  - **Authorisation**: Verify their permissions.
- Establishing **Secure Channels** often requires brokered authentication.
- Access Control Models help manage permissions at OS and Application Level.
- **Policy Enforcement Points** design pattern to provide distributed access control.

Section 6

Crypto Basics

Why use Cryptography?

Cryptography can be used to provide **mathematical** guarantees towards:

- **Confidentiality**
  - Public Key Encryption i.e. RSA, ElGamal, ECC
  - Block Ciphers i.e. Blowfish, TripleDES, Skipjack, AES
  - Stream Ciphers i.e. RC4
- **Integrity**
  - Cryptographic Hash function i.e. MD-family, SHA-family
  - Message Authentication Codes
- **Authenticity** & **Non-Repudiation**
  - Digital Signatures i.e. DSS, (EC)DSA

Cryptographic Hash Functions/Message Digests

Definition
Function to compute a unique (random) signature for some data:

$$\# : \{0,1\}^n \rightarrow^R \{0,1\}^n$$

- **Provides guarantees towards:** Data Integrity.
- **Properties**
    - **Pre-image resistance**: Given $\#(m)$, hard to find m.
    - **Second Pre-image resistance**: Given m_1, hard to find m_2 such that $m_1 \neq m_2$ & $\#(m_1) == \#(m_2)$
    - **Collision Resistance**: Hard to find m_1, m_2 such that $\#(m_1) == \#(m_2)$
- **Implementations**
    - MD-Family, SHA-Family

Block Ciphers: Symmetric Cryptography

Definition
Set of functions to encrypt data.

$$C \leftarrow \mathrm{Encrypt}(M, \mathrm{K}_M)$$
$$M \leftarrow \mathrm{Decrypt}(C, \mathrm{K}_M)$$

- **Provides guarantees towards:** Confidentiality
- **Properties**
  - Same key used to encrypt and decrypt.
  - Implementations are very efficient for large messages.
- **Implementations**
  - Blowfish, TripleDES, Skipjack, AES

Asymmetric Ciphers

Definition

Set of functions to encrypt data.

$$
\begin{aligned}
(\mathrm{Enc}(Alice), \mathrm{Dec}(Alice)) &\leftarrow \mathrm{KeyGen}(\lambda) \\
C &\leftarrow \mathrm{Encrypt}(M, \mathrm{Enc}(Alice)) \\
M &\leftarrow \mathrm{Decrypt}(C, \mathrm{Dec}(Alice))
\end{aligned}
$$

- **Provides guarantees towards:** Confidentiality
- **Properties**
  - Use of Key Pairs.
  - One key used to encrypt, the other decrypt.
  - Two modes of use: Encrypting & Signing
  - Very inefficient on large data.
- **Implementations**
  - DSA, (EC)DSA, ECC, RSA, ElGamal

Cryptographic Workflows

Ways in which crypto primitives can be combined/used to provide one or more security guarantees.

- **Information Secrecy**
- **Efficient Information Secrecy**
- **Sender Authentication**
- **Secrecy with Authentication**
- **Secrecy with Signature**
- **Secrecy with Integrity**
- **Signature with Appendix**
- **Secrecy with Signature with Appendix**

Public Key Cryptography

Asymmetric Crypto can be used to provide:

- **Confidentiality**

Key Generation

$$(\mathsf{Enc}_{pub}(Bob), \mathsf{Dec}_{priv}(Bob)) \leftarrow \mathsf{KeyGen}(\lambda)$$

Alice

1. $C \leftarrow \mathsf{Encrypt}(M, \mathsf{Enc}_{pub}(Bob))$
2. $Alice \rightarrow Bob : C$

Bob

1. $C' \leftarrow C$
2. $M' \leftarrow \mathsf{Decrypt}(C', \mathsf{Dec}_{priv}(Bob))$

Digital Signatures

Asymmetric Crypto and **Message Digests** can be used to provide:

- Authenticity of message origin
- Non-Repudiation of message origin
- Message Integrity

Key Generation

$$(\mathsf{Enc}_{priv}(Alice), \mathsf{Dec}_{pub}(Alice)) \leftarrow \mathsf{KeyGen}(\lambda)$$

Alice

1. $H \leftarrow \#(M)$
2. $S \leftarrow \mathsf{Sign}(H, \mathsf{Enc}_{priv}(Alice))$
3. $Alice \rightarrow Bob : (M \mid\mid S)$

Bob

1. $(M', S') \leftarrow (M \mid\mid S)$
2. $H' \leftarrow \mathsf{Verify}(S', \mathsf{Dec}_{pub}(Alice))$
3. Accept iff $\#(M') \equiv H'$

Public Key Encryption & Digital Signatures

Combining the previous primitives provides the following:

- Message Confidentiality and Integrity
- Authenticity of message origin
- Non-Repudiation of message origin

Alice

1. $H \leftarrow \#(M)$
2. $S \leftarrow \mathrm{Sign}(H, \mathrm{Enc}_{priv}(Alice))$
3. $C \leftarrow \mathrm{Encrypt}(M, \mathrm{Enc}_{pub}(Bob))$
4. $Alice \rightarrow Bob : (C \mid\mid S)$

Bob

1. $(C', S') \leftarrow (C \mid\mid S)$
2. $M' \leftarrow \mathrm{Decrypt}(C', \mathrm{Dec}_{priv}(Bob))$
3. $H' \leftarrow \mathrm{Verify}(S', \mathrm{Dec}_{pub}(Alice))$
4. Accept iff $\#(M') \equiv H'$

KEM/DEM

Improve encryption **efficiency** through a hybrid encryption scheme:

- **Symmetric** Encryption to encrypt **data**; and
- **Asymmetric** Encryption to encrypt **symmetric key**.

AKA **Key Encapsulation**/**Data Encapsulation** Mechanism

Alice

1. $C_M \leftarrow \mathrm{Encrypt}(M, \mathrm{K}_{Random})$
2. $C_K \leftarrow \mathrm{Encrypt}(\mathrm{K}_R, \mathrm{Enc}_{pub}(Bob))$
3. $Alice \rightarrow Bob : (C_K \mid\mid C_M)$

Bob

1. $(C'_K, C'_M) \leftarrow (C_K \mid\mid C_M)$
2. $\mathrm{K}'_R \leftarrow \mathrm{Decrypt}(C'_K, \mathrm{Dec}_{priv}(Bob))$
3. $M' \leftarrow \mathrm{Decrypt}(C'_M, \mathrm{K}'_R)$

Some Cryptographic Algorithms

- **RSA Systems**
  - Security lies in the **hardness** of factorising large numbers.
  - Examples: RSA Encryption, RSA Digital Signatures
- **Discrete Logarithm Systems**
  - Security lies in the **hardness** of taking discrete logarithms over **finite fields**.
    - Key Exchange i.e. Diffie-Hellman Key Exchange
    - Digital Signature Algorithm
    - Discrete Logarithm Integrated Encryption Scheme
- **Note:** Many variants of DL schemes in different settings e.g. Elliptic Curves.

This list is far from complete. . .