TÉCNICO LISBOA

Cryptographic Services Technology

Segurança Informática em Redes e Sistemas
2024/25

David R. Matos, Ricardo Chaves

Ack: Miguel Pardal, Miguel P. Correia, Carlos Ribeiro

Cryptographic services (revisited)

- We can now implement cryptographic services:
  - Confidentiality
  - Integrity
  - Authenticity
- To protect against:
  - Unauthorized insertion of information
    - Loss of authenticity
  - Unauthorized modification of information in transit
    - Loss of integrity
  - Unauthorized replay of information
    - Loss of authenticity
  - Unauthorized access to information
    - Loss of confidentiality

Cipher suite

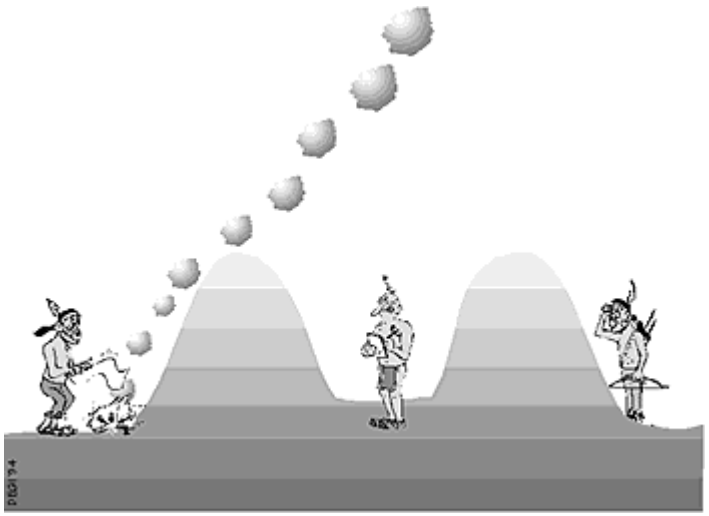
- A collection of cryptographic algorithms and protocols for digital security
- Components include encryption algorithms, message authentication codes and hash functions, like:
  - AES to encrypt data
  - HMAC with SHA-256 to allow data tampering detection
- Used in protocols like SSL/TLS for secure Internet communications:
  - Selection - server and client negotiate to choose the most suitable suite during connection setup
  - Key Exchange - algorithms like RSA or Diffie-Hellman are used to establish a secure connection *(more on this later in the course)*

Example: cipher suites in Java

- JCA (Java Cryptography Architecture):
- Framework for cryptographic functions in Java
  - Supports encryption, key generation, and digital signatures
  - Ensures uniform cryptography across Java applications
- In the code, applications ask for specific implementations from providers:
  - `Cipher.getInstance("AES/CBC/PKCS5Padding");`
  - `Cipher.getInstance("RSA/ECB/PKCS1Padding");`
  - `MessageDigest.getInstance("SHA-256");`
  - `Signature.getInstance("SHA256withRSA");`
  - `Mac.getInstance("HmacSHA256");`

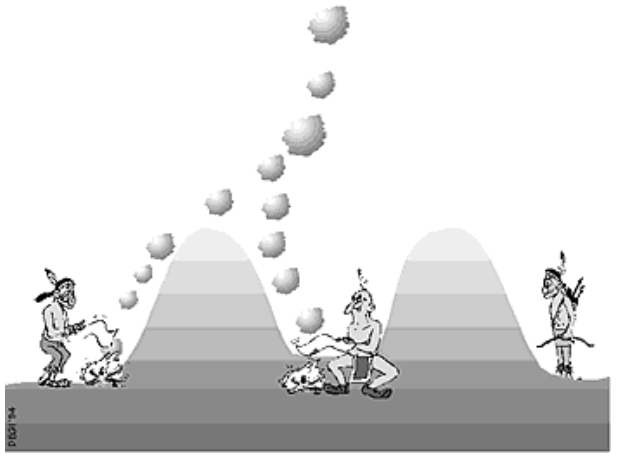
1 – Confidentiality

- Symmetric cipher
  - `Cipher.getInstance("AES/CBC/PKCS5Padding");`
- Asymmetric cipher
  - `Cipher.getInstance("RSA/ECB/PKCS1Padding");`



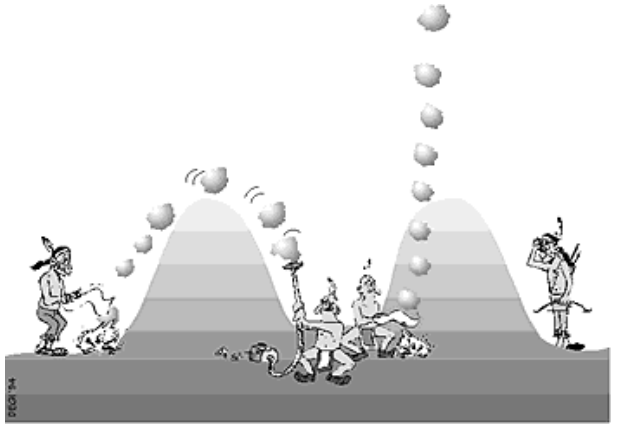
2 – Integrity

- MIC
  - `Mac.getInstance("HmacSHA256");`
- DS
  - `Signature.getInstance("SHA256withRSA");`



3 - Authenticity

- The basis of authenticity is *integrity* using MIC or DS
- Add **metadata** with *origin* information :
  - e.g. sender identification
- and *freshness* elements:
  - e.g. timestamp and random number
  - the random number must be unique inside the timestamp's acceptance interval
- The metadata and data must be protected **together**!
  - Same MIC or DS



Summary

- We need **robust** and **well-tested** cryptographic implementations to protect our applications
- Usually there are multiple cipher suites available
  - We saw the Java example, but similar capabilities are available in other mainstream programming languages
  - Algorithms are standardized, so there is compatibility between heterogeneous systems
- Using them correctly, we can provide services of:
  - Confidentiality
  - Integrity
  - Authenticity