Advanced Programming for Cryptographic Methods

Diego Oniarti

Anno 2025-2026

Contents

1 introduction

Schedule Iniziamo alle 14:00

topics

- introduction to applied cryptography. Not the quantum stuff yet.
- How the theoretical aspects of cryptography translate on physical machines
- Software development and security
- JAVA using JCA and JCR
- C and OpenSSL

exam

- exercises assigned burring the semester (0 to 5 points)
- software project and report (0 to 27 points)
- oral exam with questions stating from the project and spanning the whole course (-5 to 5 points)

no fixed groups for exercises

Cryptography overview Two actors (Alice and Bob) have to communicate over an *insecure* channel. The channel is not secure because a third actor (Eve) is trying to tamper with the communication.

Eve could have different capabilities in terms of computation power, system knowledge, etc.
Existing systems have to constantly be updated to keep up with Eve's potential evolutions.

Three things that we want from our systems are

- Confidentiality
- Integrity
- Availability

Cryptography can work on the first two. But availability is not our problem.

Theoretical vs applied In theory numbers, keys, and other models can be arbitrarily big. In the real world whey need to be stored into memory or storage and processed in binary.

Hash Function Function $f(x) = d$ that takes an element x from a set and returns a sequence of bits d.

- d must be of fixed size
- f must not be reversible
- $f(x)$ must be unique. Collisions violate *integrity*.
  In the real word collisions are inevitable. But we must strive to minimize them as much as possible

Cryptosystem A cryptosystem is a 5-tuple $< E, D, M, K, C >$ where

- E is an encryption algorithm
- D is a decryption algorithm
- M is the set of plaintexts
- K is the set of keys
- X is the set of ciphertextx

E and D can be characterized as functions

$$\begin{aligned} E &: M \times K \to C \\ D &: X \times K \to M \\ D(E(m, k), k) &= m \end{aligned}$$

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

This principle makes interoperability of cryptographic primitives possible.

modularity Cryptographic primitives are not *modular*. This means that even a combination of secure primitives can create an insecure application.

2 Hash Functions

Hash functions should be strictly one-way and it should be unfeasible to generate a collision on demand.

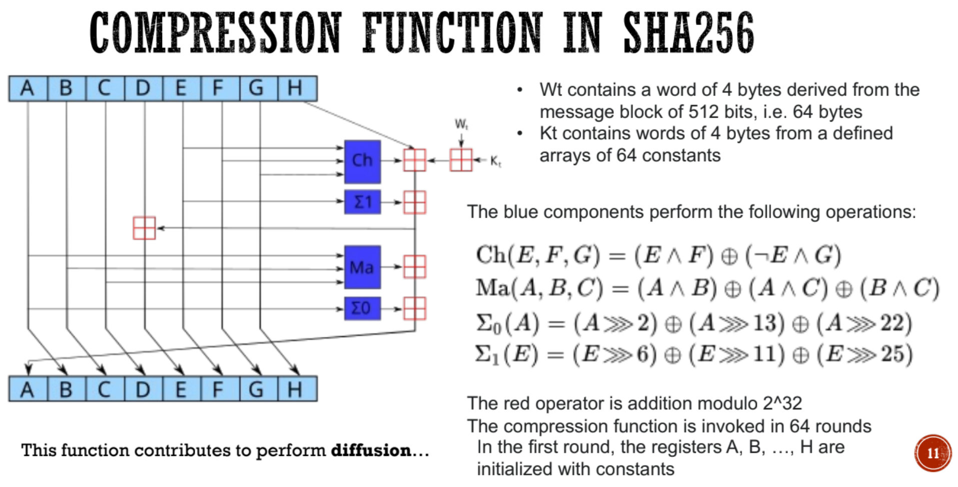
We will use the *Merkle construction* for this.
The input message is partitioned into t number of bit blocks, each one being n bits.
If necessary the final block is padded so that it is of the same length as the others.

The construction is parametric with respect to a compression function f which takes 2 arguments.

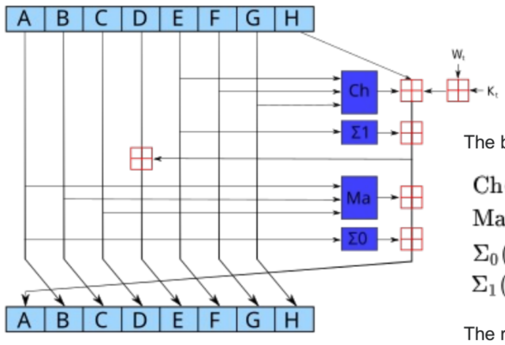
We take an initialization vector with the size m of the final message. Then f is applied iteratively on the initialization vector, passing a different block of the message as the second parameter each time.

The compression function in `SHA256` takes in some constant as well to further increase the difficulty in cracking it. It is a composition of linear and non-linear operations, like bit-shifts and xor operations.

2.1 Linear functions, diffusion and digression



COMPRESSION FUNCTION IN SHA256



- Wt contains a word of 4 bytes derived from the message block of 512 bits, i.e. 64 bytes
- Kt contains words of 4 bytes from a defined arrays of 64 constants

The blue components perform the following operations:

$$\mathrm{Ch}(E,F,G) = (E \land F) \oplus (\neg E \land G)$$
$$\mathrm{Ma}(A,B,C) = (A \land B) \oplus (A \land C) \oplus (B \land C)$$
$$\Sigma_0(A) = (A \ggg 2) \oplus (A \ggg 13) \oplus (A \ggg 22)$$
$$\Sigma_1(E) = (E \ggg 6) \oplus (E \ggg 11) \oplus (E \ggg 25)$$

The red operator is addition modulo 2^32

The compression function is invoked in 64 rounds

In the first round, the registers A, B, …, H are initialized with constants

This function contributes to perform **diffusion**…

11

Figure 1: Compression function in SHA256