ELTE
EÖTVÖS LORÁND
UNIVERSITY

CRYPTOGRAPHY AND SECURITY

Practice

IP-18FKVKRBG

Lecture 8

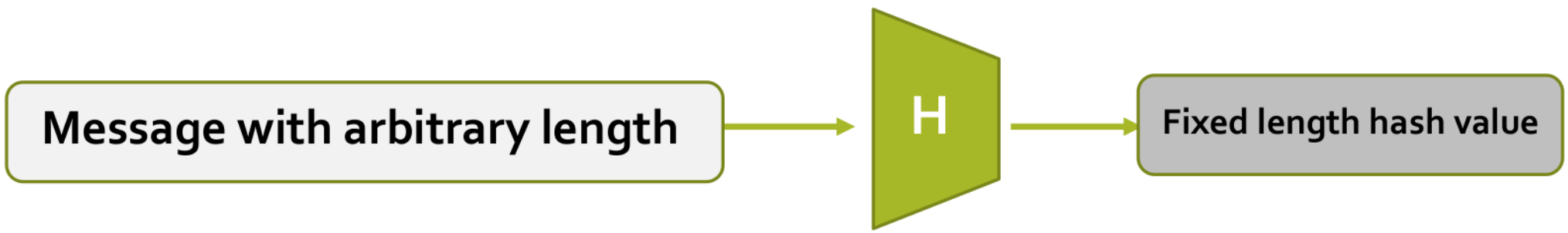
2022-2023-2

Hash Functions

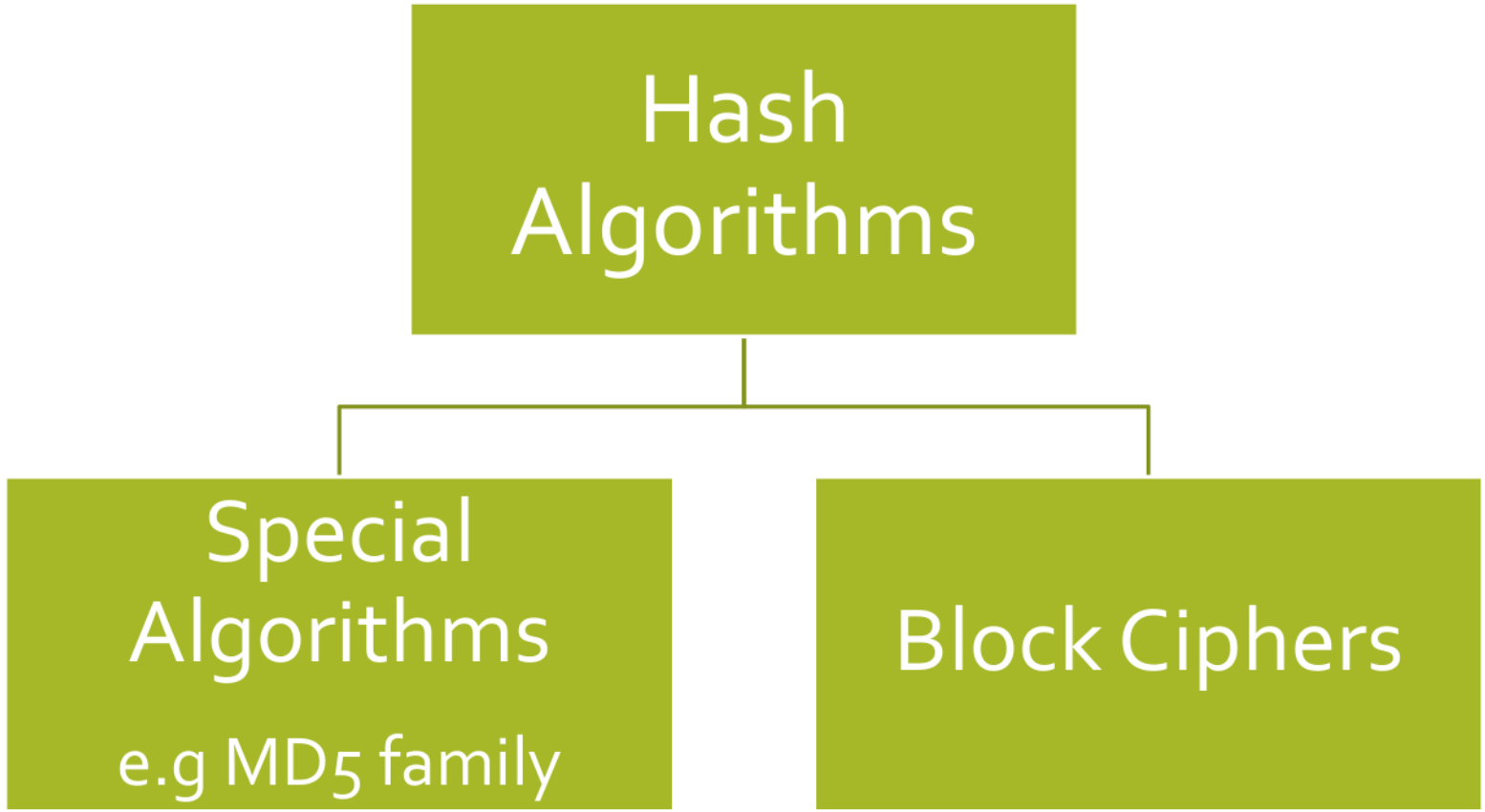
Hash Function

- A **hash function** is a computationally efficient function mapping binary strings of arbitrary length to binary strings of some fixed length, called *hash-values*.

$$\boldsymbol{H} : \{0,1\}^* \rightarrow \{0,1\}^k$$

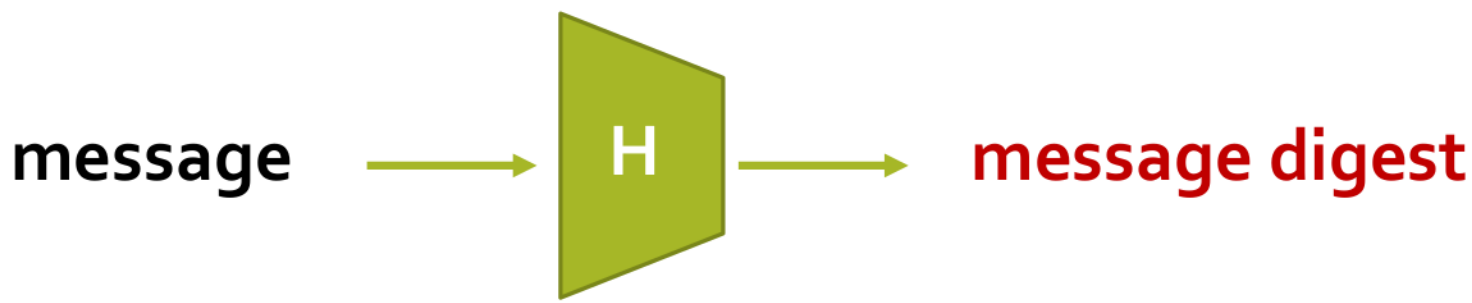


Hash Function Algorithms



Cryptographic hash function

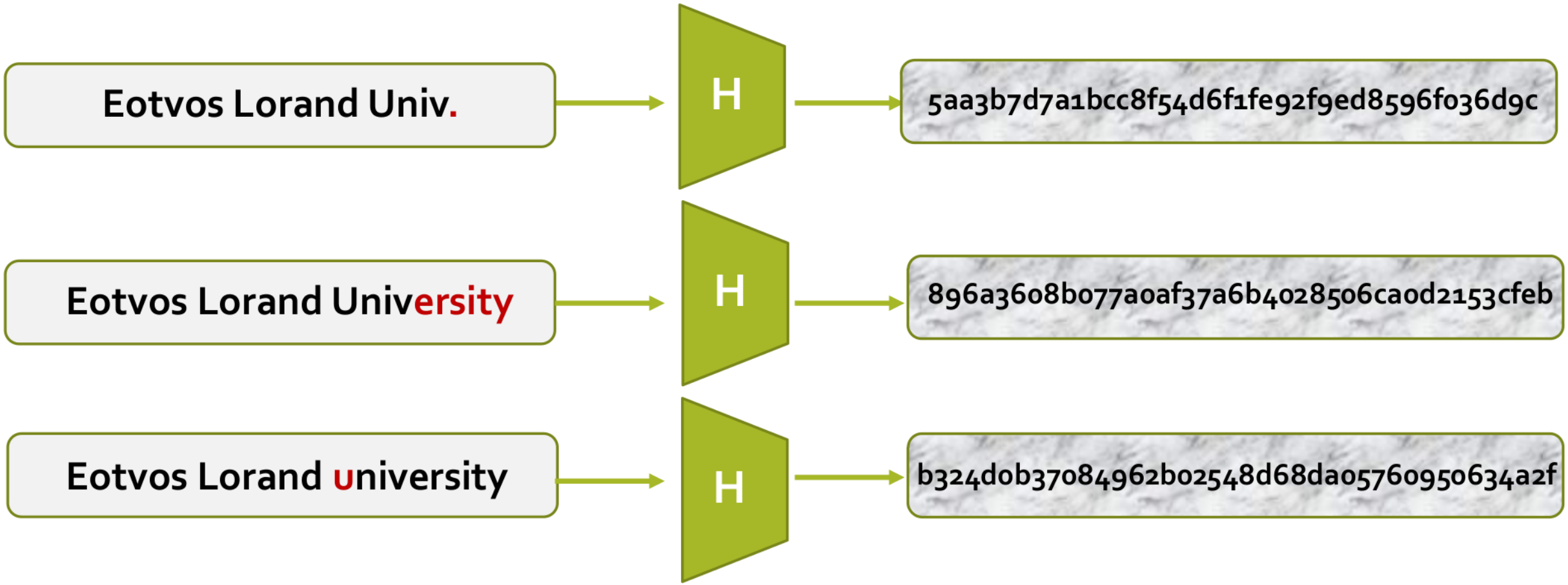
- A cryptographic hash function is a mathematical algorithm that maps data of arbitrary size to a bit string of a fixed size (a hash function) which is designed to also be **one-way function**, that is, a function which is **infeasible to invert**.
- The input data is often called the **message**, and the output (the hash value or hash) is often called the **message digest** or simply the digest.



Properties of cryptographic hash functions

- it is quick to compute the hash value for any given message.
- it is infeasible to generate a message from its hash value except by trying all possible messages.
- a small change to a message should change the hash value so extensively that the new hash value appears uncorrelated with the old hash value.
- it is infeasible to find two different messages with the same hash value.

Principal behavior of hash functions



* sha1

Applications of one-way hash

- Password files (one way)
- Data integrity
- Digital signatures (collision resistant)
- Keyed hash for message authentication

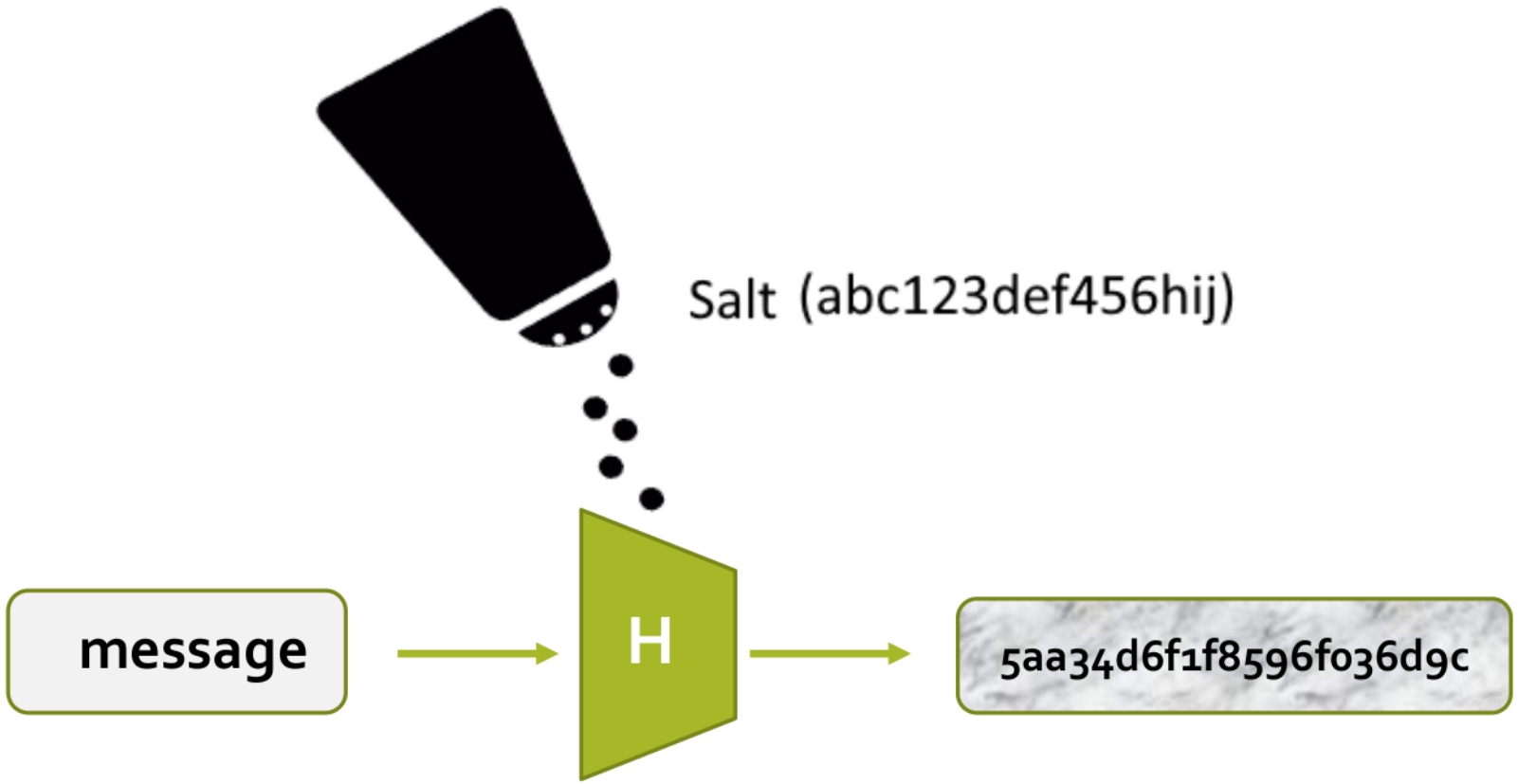
A simple Hashing

```
from hashlib import sha1


def generate_hash(name):
   return sha1(name.encode()).hexdigest()



def test():
    name = input('Enter your name: ')
    print('The hashed value of your name is = ', generate_hash(name))

test()
```

Simple Password Hashing

```
from hashlib import sha1

def hash_password(password):
   return sha1(password.encode()).hexdigest()

def check_password(stored_hashed_password, plain_password):
   password_validity = stored_hashed_password == sha1(plain_password.encode()).hexdigest()
   return password_validity

def test(entered_password):
    hashed_password = 'd20a09545c7aff14a4f596ddba19296d58f6c101'
    valid_pass = check_password(hashed_password, entered_password)
    if valid_pass:
      print('[x] Password is valid')
    else:
      print('[x] Password is not valid')

test('secret password')
```

Salted Hashing



Salted Hashing

```
import uuid
from hashlib import sha1

def hash_salted_password(password):
    salt = uuid.uuid4().hex  # generate a random unique ID as a 'salt' value
    return sha1(salt.encode() + password.encode()).hexdigest() + ':' + salt

def check_salted_password(stored_hashed_password, plain_password):
    password_hash, salt = stored_hashed_password.split(':')
    password_validity = password_hash == sha1(salt.encode() + plain_password.encode()).hexdigest()
    return password_validity

def generate_salted_password(plain_password):
    print('[x] The plain password : ', plain_password)
    hashed_password = hash_salted_password(plain_password)
    print('[x] Stored in the db : ', hashed_password)

generate_salted_password('secret password')
```