Lecture 13
The CIA Triad

ECE 422: Reliable and Secure Systems Design

Instructor: An Ran Chen
Term: 2024 Winter

Schedule for today

- Key concepts from last classes (Reliability)
- The CIA triad
  - Confidentiality
  - Integrity
  - Availability
- Digital Signature
  - Birthday problem
  - Hash function

SRE vs DevOps

SRE focuses on reliability

- Operating and monitoring
- Incident response
- Capacity planning

DevOps focuses on delivery

- CI/CD
- Release automation
- Configuration management

Note: SRE is to DevOps is what Scrum is to Agile; one implementation of a philosophy and principles, but not a 100% subset.

Fault tolerance techniques

- Single version techniques
  - Exception handling
- Multiple version techniques: use two or more versions of the same software module to achieve fault tolerance through software redundancy.
  - Recovery blocks
  - N-version programming
  - N self-checking programming
- Multiple data representation techniques (not covered in class)
  - Retry blocks
  - N-copy programming

Fault removal

- Why? Despite fault tolerance efforts, not all faults are tolerated, so we need fault removal.
  - To keep in mind: fault tolerance is a must-have property for safety-critical systems.
  - But for software that we use everyday, we want to remove faults to improve user experience.
- Improving system dependability by:
  - Detecting existing faults through software verification and validation
  - Eliminating the detected faults

Fault removal as two concepts:

1. as a solution to improve availability, and thus dependability
2. as a solution for faults that affect user's daily activities (not tolerated)

Fault localization

Fault localization techniques have been proposed to assist in locating and understanding the root causes of faults.

Why? Fault localization as a debugging technique to maintain the system:

- Pinpoint the location to fix in the code
- Recover fast from bugs, and reduce its impact on the users
- Reduce manual debugging efforts, more time for new feature

Information redundancy

Information redundancy tolerate faults by adding information to the original data.

- Tolerate faults by means of coding
- Avoid unwanted information changes
- E.g., information loss during data storage or transmission

The term “coding” is used in the context of communication and data storage

Byzantine fault tolerance

Byzantine fault tolerance: the ability of a system to continue functioning even if some of its nodes/components fail randomly or act maliciously.

- The system can keep functioning even if certain components stop working
- For example, safety-critical systems need to be able to work when if some of its components fail
  - E.g., train, airplane, spacecraft
  - E.g., heart-lung machine, robotic surgery

Schedule for today

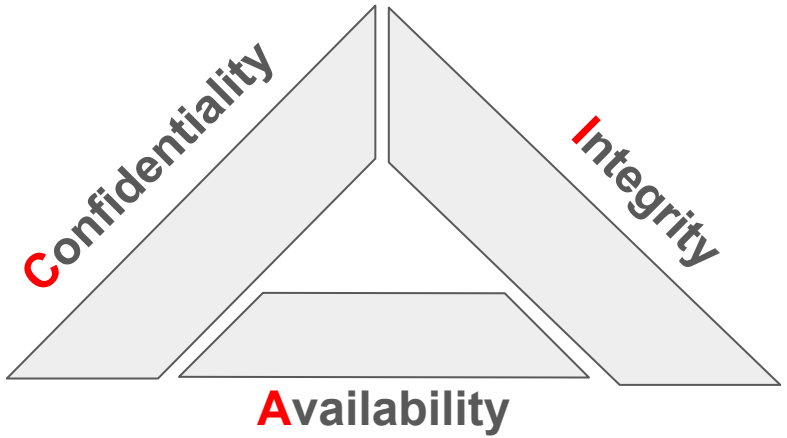
- Key concepts from last classes (Reliability)
- The CIA triad
  - Confidentiality
  - Integrity
  - Availability
- Digital Signature
  - Birthday problem
  - Hash function

The CIA triad

CIA: Confidentiality, integrity and availability

- Guiding model in information security
- Also known as the AIC Triad

Despite the acronym, this is not related to the Central Intelligence Agency in any way.



The CIA triad

The CIA triad:

- Confidentiality: only the authorized user can access particular resources
- Integrity: ensure data are trustworthy, complete, and have not been modified by unauthorized parties
- Availability: ensure data are accessible when needed

Both security and reliability are concerned with these three concepts

- Difference: the presence or lack of a malicious adversary

Confidentiality

Confidentiality

- Only the authorized user can access particular resources

Methods to achieve confidentiality:

- Encryption: encoding/decoding of the plaintext
  - E.g., Symmetric/asymmetric encryption
- Access controls: restricted access
  - E.g., Our library website
- Two-factor authentication: additional security checks
  - E.g., Mobile authentication for faculty and staff

UoA's Information Services & Technology

Integrity

Integrity

- Ensure data are trustworthy, complete, and have not been modified by unauthorized parties

Methods to achieve integrity:

- Hashing: transforms any given data into fixed-size values
  - E.g., Comparing stored data
- Digital signature: verifies the authenticity of data
  - E.g., Emails, software application codes

Availability

Availability

- Ensure data are accessible when needed

Methods to achieve integrity:

- Fault tolerance
  - E.g., Airplane, heart-lung machine
- Redundancy
  - E.g., Hardware duplicates
- Testing and maintenance
  - E.g., Function or structural tests

The CIA triad

Confidentiality

Integrity

Availability

A hacker intercepts wireless traffic from users

A push-to-talk microphone stuck in the transmit position

A hacker modifies the messages from users

Server overload

Trade-off: Redundancy

Reliability assumption: some things will go wrong at some point.

- The primary risks are non malicious in nature
- In the absence of an adversary, systems often fail safely (open)

Therefore, a reliable design should consider redundancy.

- E.g., an electronic lock is designed to remain open in case of power failure, to allow safe exit through the door.

However, redundancy increases the attack surface.

- An adversary only needs find one vulnerability in one path to be successful.

Trade-off: Incident management

Security assumption: an adversary can make things go wrong at any point.

- The risks are malicious in nature
- The presence of an adversary necessities the systems to fail securely (remain closed)

Therefore, a secure design should consider incident management.

- E.g., a door fails securely and remains closed when not powered.

However, incident management shares information on a need-to-know basis.

- A larger volume of logs may be useful to reduce the time to recovery, and thus keeping the system reliable.
- But, these logs can also be a valuable target for an attacker.

Schedule for today

- Key concepts from last classes (Reliability)
- The CIA triad
  - Confidentiality
  - Integrity
  - Availability
- Digital Signature
  - Birthday problem
  - Hash function

Birthday problem

Let P be the probability that two people not having their birthday on the same day.

- $P_1 = \frac{365}{365}$
- $P_2 = \frac{365}{365} \times \frac{364}{365}$
- $P_3 = \frac{365}{365} \times \frac{364}{365} \times \frac{363}{365}$
- $P_n = \frac{365 \times 364 \times .. \times (365 - n + 1)}{365^n}$

Birthday problem

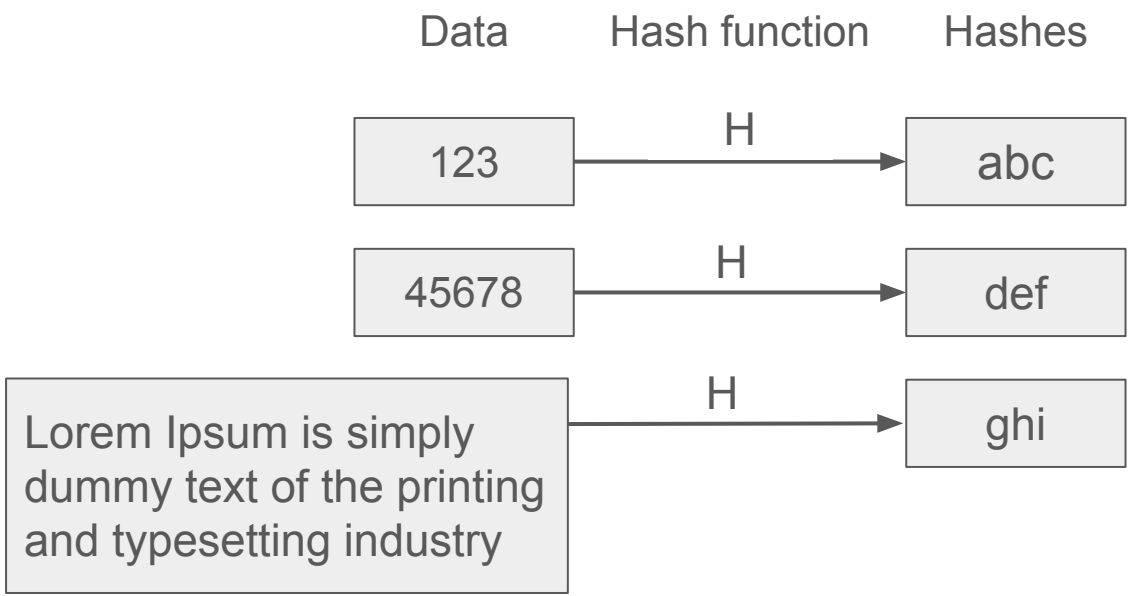
Let Q = (1 - P), the probability that two people have the same birthday, then:

- P_{10} = 88.31% Q = 1 - P = 11.69%
- P_{25} = 43.13% Q = 1 - P = 56.87%
- P_{50} = 2.96% Q = 1 - P = 97.04%
- P_{78} = 00.01% Q = 1 - P = 99.99%

Hash function

Hash function: transforms any given data into fixed-size values

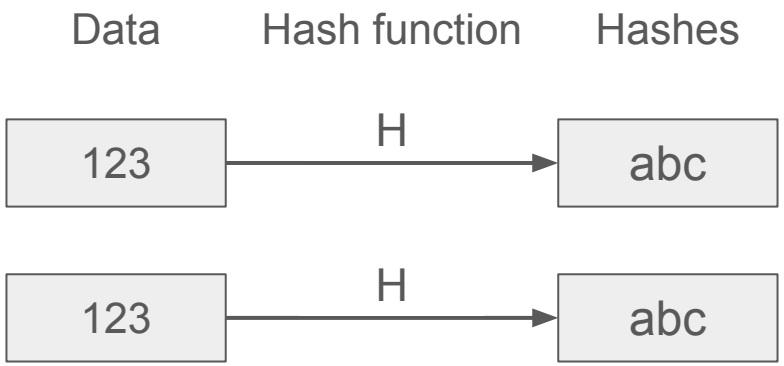
For example



Hash function

Hash function is deterministic:

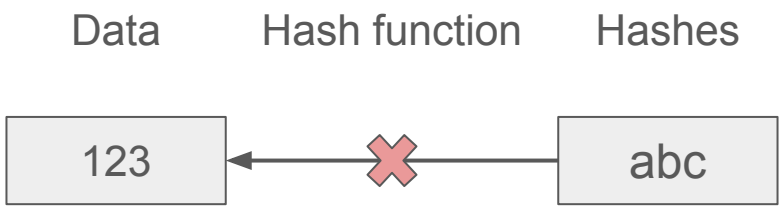
- It always produces the same hash given the same input
- Verifies the integrity of the data
  - E.g., to compare that a downloaded file is the same as the original file



Hash function

Hash function is an one-way function (irreversible):

- It is impossible to recover the original data from the hash
- Technically possible, but the computational power needed makes it infeasible



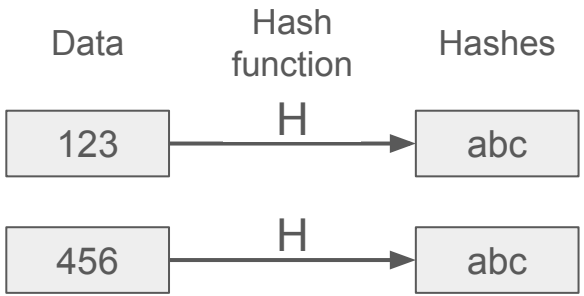
Hash collision

Hash function is unaware of its set of inputs.

- Performs arithmetic operations on the input passed to it
- Produces hashes of a fixed size

Problem? Hash collision is likely to happen.

- Hash collision happens when two pieces of data in a hash table share the same hash value
- Similar to the idea of birthday collision



Hash collision

Given a hash function that produces *n* hashes, it requires $1.2\sqrt{n}$ distinct values for the probability of a hash collision to be larger than 50%.

For example:

- A hash function with 100 hashes, we only need to hash 12 values to get 50% chance of having a hash collision.
- A hash function with 10,000 hashes, we only need 120 values.

Digital signature

Digital signatures verify the authenticity

- Detect the identity of the sender/signer

Digital signatures check the integrity

- Verify that the message was not changed

Digital signatures ensure non-repudiation

- Verify that the signature is not fake

Basic of asymmetric encryption

Asymmetric encryption uses a pair of related keys for encryption and decryption:

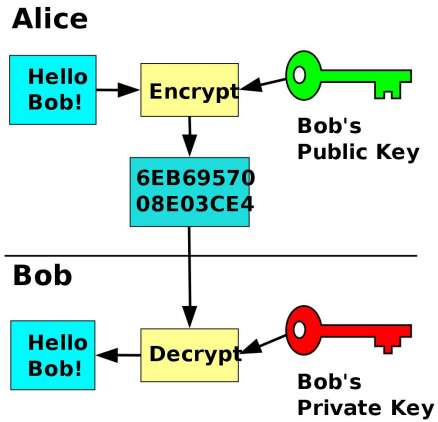
- One key for encryption, the other for decryption
- The keys are mathematically related
- File encrypted with one key can only be decrypted with the other key

Basic of asymmetric encryption

When Alice and Bob need to communicate, they publish one of their keys (the public key)

- The public key (made public) is used for encryption
- While the private key (kept private) is used for decryption
- We cannot derive the private key from the public key.

… more later



Digital signature

How is hashing used in digital signatures?

- To create a digital signature, the sender first applies a hash function to the message, then encrypts the hash with his/her private key.
- To verify the signature, the receiver decrypts the hash with the (sender's) public key, and compares it to the hash of the message. If they match, the signature is valid.

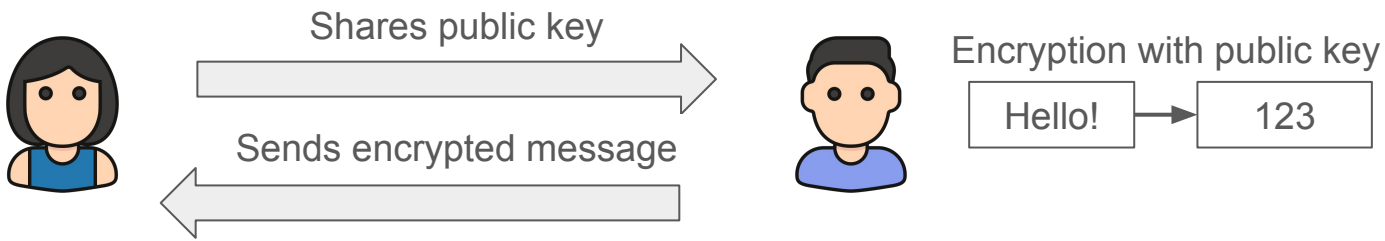
Public key: anyone can use to verify the signature

Private key: only the sender knows

Basic asymmetric encryption

Alice needs to talk to Bob

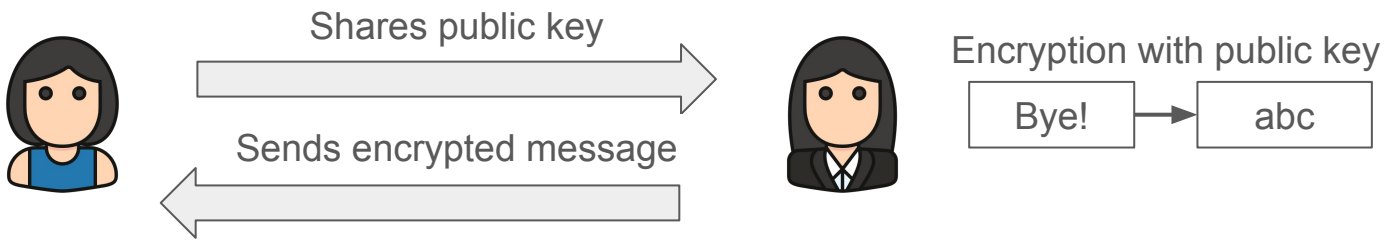
- Alice shares her public key.
- Bob uses the public key to encrypt his message, and sends back the encrypted message “123” back to Alice.
- Alice uses her private key to open the message.



Basic asymmetric encryption

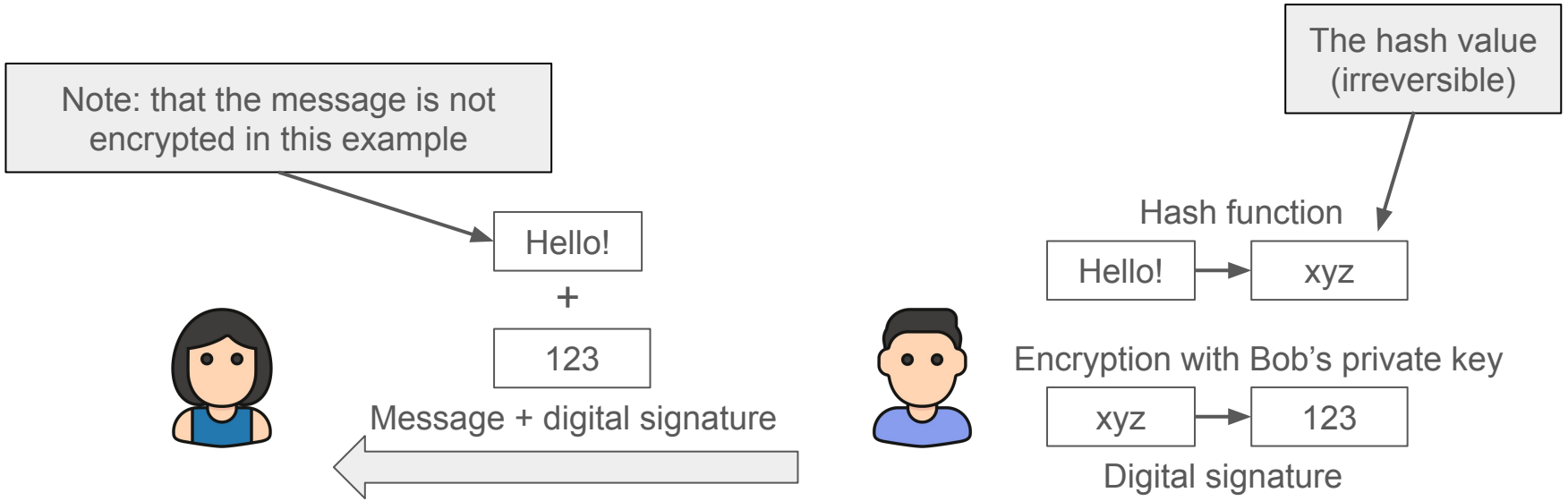
What happens if there is a third person, Eve:

- Alice shares her public key.
- Eve pretends to be Bob and sends back the message “Bye!” back to Alice.
- Alice uses her private key to open the message.
- Alice: “Does it come from Bob or someone else?”



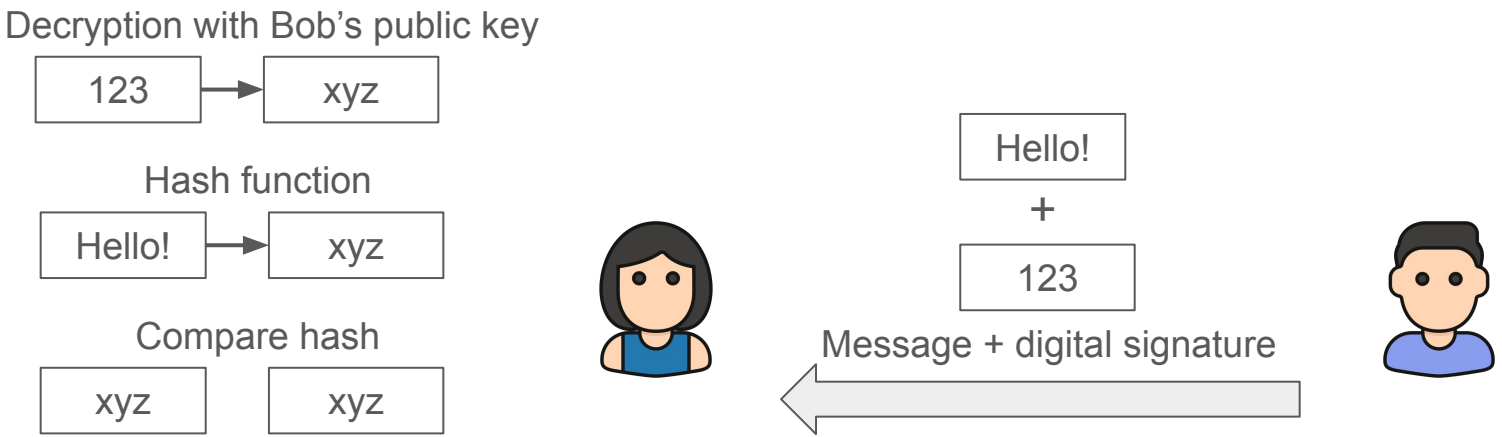
With digital signature

- Bob converts the message “Hello” into a hash “xyz”.
- Bob encrypts the hash with his private key, and sends it back together with the message “Hello” to Alice.



With digital signature

- Alice uses Bob's public key to decrypt the message.
- Alice create a hash of the message by herself.
- Alice verifies whether the hash matches what Bob sends.



Digital signature

What happens if Eve pretends to be Bob?