Systems 3

Security and Cryptography

Marcel Waldvogel

(Handout)

Department of Computer and Information Science
University of Konstanz

Winter 2019/2020


Radfahrer
absteigen

Chapter Goals

- What are the three main goals in security?
- What are the basic means of achieving confidentiality and integrity?
- How do we get the users on our side?
- How can we secure network connections?
- How should passwords be stored?

CIA

Main security objectives:

Confidentiality	Data should be secret to unintended users/recipients
Integrity	Data tampering should be impossible (or at least not go undetected)
Availability	Services should be available

(Other security objectives include authenticity, non-repudiation, ...)

How does privacy fit into this picture?

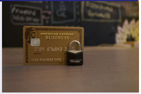
Identification: Three factors

Something you know

- Password
- PIN

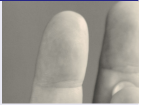
Something you have

- Bank card
- Hardware token
- Phone



Something you are

- Biometrics:
  Fingerprint, face, retina, speech, typing pattern, gait, ...



Security Design

Criteria	Goal
Kerckhoffs's principle	Open, inspectable system
Principle of least privilege	Contain results of misbehavior
Secure by default	Laziness should not cause problems
Secure by design	Not as an afterthought
Economy of mechanism	KISS means less can go wrong
Privacy by design	Data can be toxic
Psychological acceptability	Keep users on our side
Fail securely	If something fails, avoid the epic

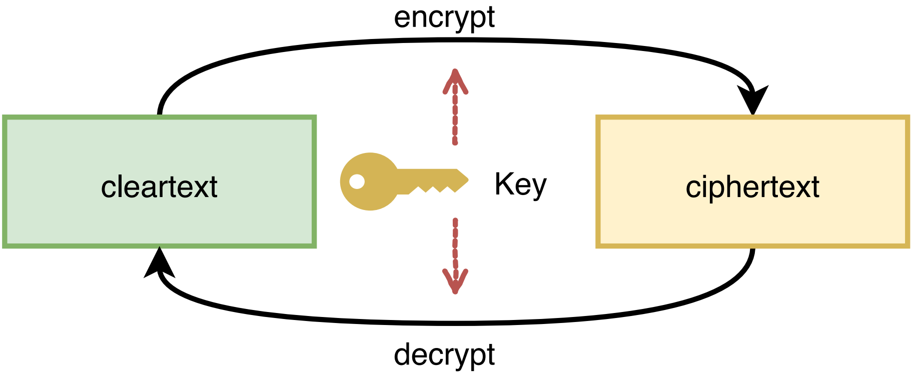
Failing safely

```c
int verify = check_access(username, password, operation);
if (verify == ERROR_ACCESS_DENIED) {
   display_error("Access denied");
} else {
   perform(operation);
}
```

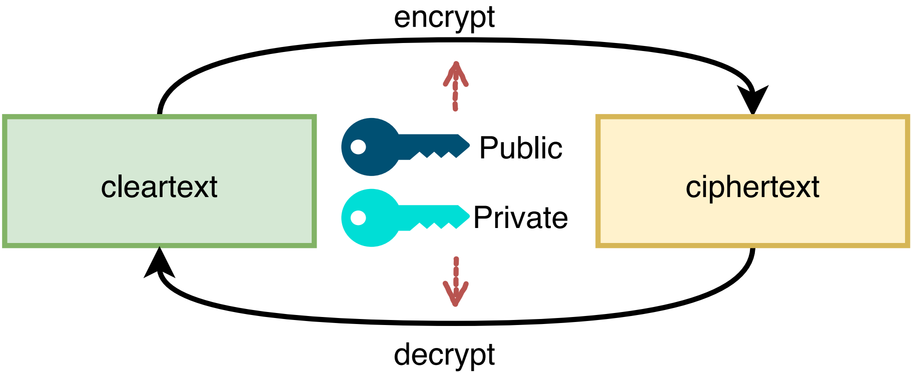
```c
int verify = check_access(username, password, operation);
if (verify == NO_ERROR) {
   perform(operation);
} else {
   display_error("Access denied");
}
```

Modeled after https://www.us-cert.gov/bsi/articles/knowledge/principles/failing-securely.

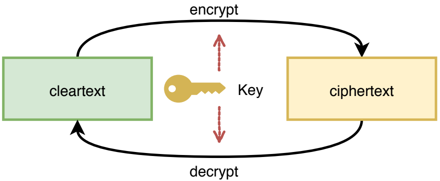
Symmetric Encryption

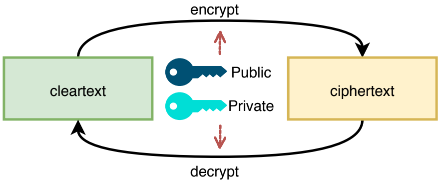


Asymmetric Encryption

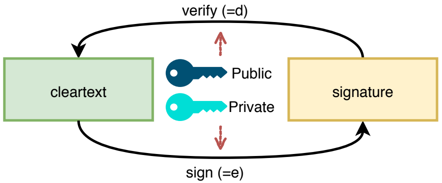


Symmetric vs. Asymmetric Encryption

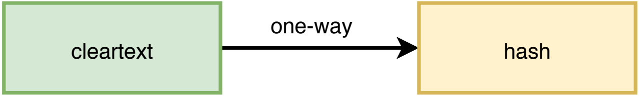




- Symmetric is (much) faster
- Asymmetric is more versatile
- Asymmetric can be used “the wrong way around”:

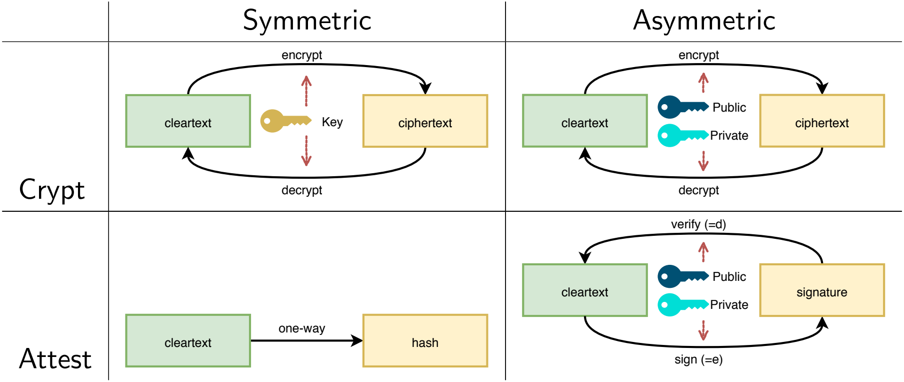


Cryptographic Hash Functions



- Not reversible (except by brute force)
- No key (but key can be included (HMAC)
- Fast
- Small change in input results in major change in hash

Comparison


Symmetric
Asymmetric
Crypt
encrypt
cleartext
Key
ciphertext
decrypt
encrypt
cleartext
Public
Private
ciphertext
decrypt
Attest
cleartext
one-way
hash
verify (=d)
cleartext
Public
Private
signature
sign (=e)

Hybrid operation

Hybrid encryption

- Generate random secret key k
- Encrypt message M symmetrically ($E_k(M)$)
- Encrypt k asymmetrically with recipient key(s)

Hybrid signature

- Hash message (symmetrical, $H(M)$)
- Sign hash (asymmetric)

Benefits

- No secret key exchange
- Only a few bytes for the (slow) asymmetric operation

PGP and S/MIME

Encrypted mail

- Hybrid encryption, signature
- PGP first published 1991.
- Public keys, distribution

Correct public key?

- Web of trust
- Certificate Authorities (CAs; "Trusted Third Party (TTP)")

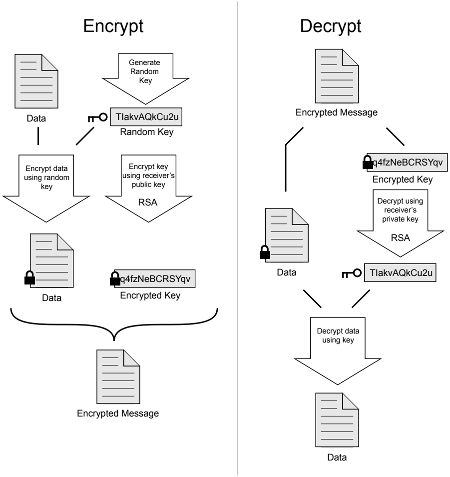


Image credit: Wikpedia user xaedes & jfreax & Acdx, CC BY-SA 3.0.

Certificates

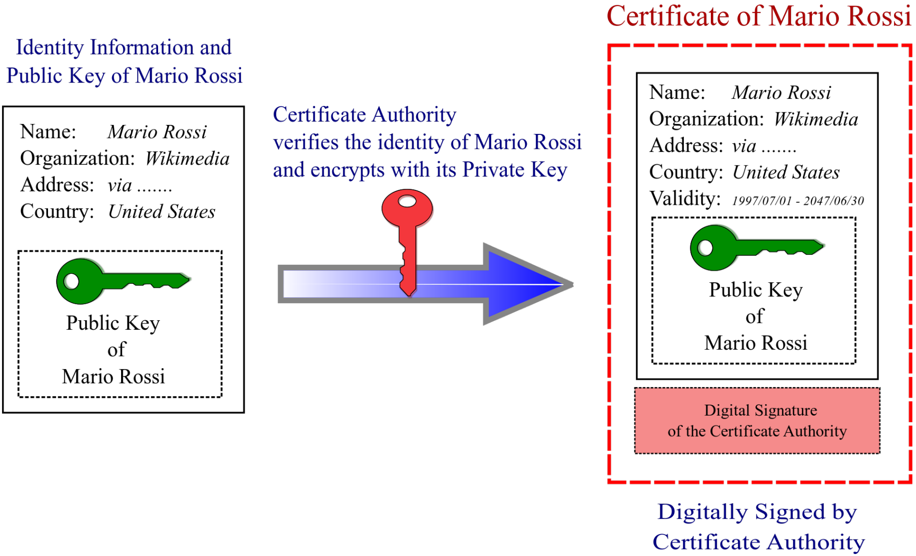
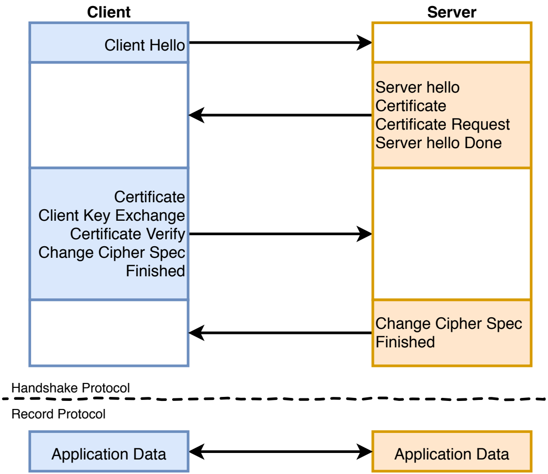


Image credit: Wikpedia user I, Giaros; CC BY-SA 3.0.

Transport Layer Security (TLS)



More information: https://developer.mozilla.org/en-US/docs/Web/Security/Transport_Layer_Security

Password storage

Never in plain text!

Hashing

Makes reversing hard.
Attack: Lookup tables.

Salted Hashing

Defeats lookup tables.
Attack: Cloud infrastructure

Iterated Hashing

Defeats cheap computing power through higher computation costs.
Common algorithms are PBKDF2, scrypt and bcrypt.

Bcrypt

Tools like hashcat[1] are able to compute 68,000,000,000[2] SHA1 hashes per second. Therefore we have to rise the cost of calculating a hash. One example is Bcrypt.

`$2a$10$N9qo8uLOickgx2ZMRZoMyeIjZAgcfl7p92ldGxad68LJZdL17lhWy`

Idea: $\text{result} = H^{2^{10}}(\text{password} + \text{salt})$

On creation, the salt is chosen randomly. On verification, the salt is extracted from the database entry and the calculated result is compared to the database entry.

[1] https://hashcat.net

[2] https://gist.github.com/epixoip/a83d38f412b4737e99bbef804a270c40

Literature

- Matt Bishop. Computer Security: Art and Science. Addison-Wesley, 2003.
- Bruce Schneier. Schneier on Security. John Wiley & Sons, 2008. (A subset of https://www.schneier.com/essays/.)
- Niels Ferguson, Bruce Schneier, and Tadayoshi Kohno. Cryptography Engineering. John Wiley & Sons, 2010.