

Systems 3

Security and Cryptography

Marcel Waldvogel

(Handout)

Department of Computer and Information Science
University of Konstanz

Winter 2019/2020



Photo by Marcel Waldvogel (CC-BY 4.0)

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

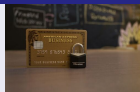
Something you know

- 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

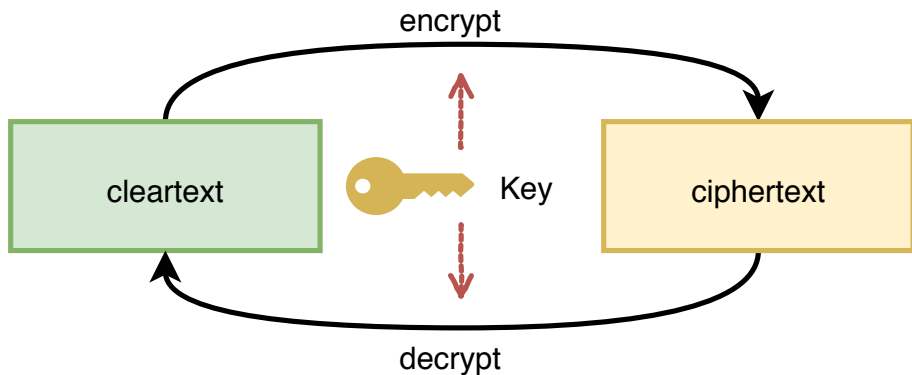
Failing safely

```
1 int verify = check_access(username, password, operation);  
2 if (verify == ERROR_ACCESS_DENIED) {  
3     display_error("Access denied");  
4 } else {  
5     perform(operation);  
6 }
```

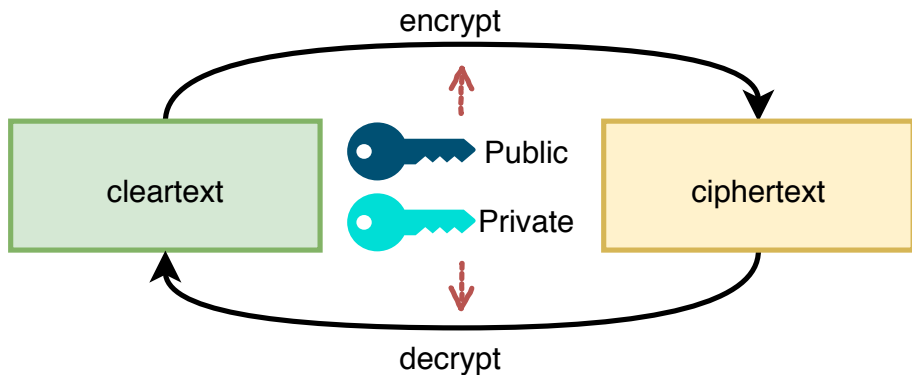
```
1 int verify = check_access(username, password, operation);  
2 if (verify == NO_ERROR) {  
3     perform(operation);  
4 } else {  
5     display_error("Access denied");  
6 }
```

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

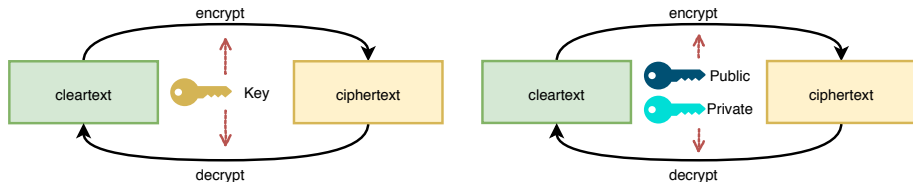
Symmetric Encryption



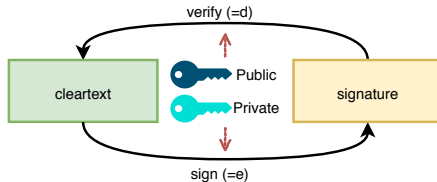
Asymmetric Encryption



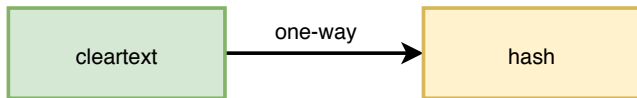
Symmetric vs. Asymmetric Encryption



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

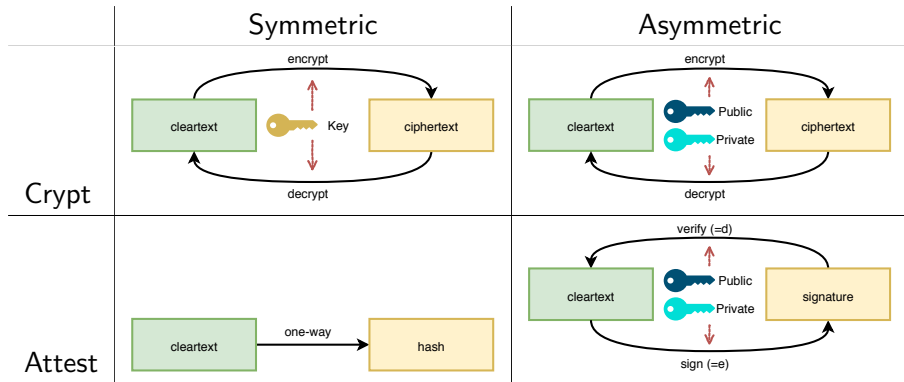


Cryptographic Hash Functions



- 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



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

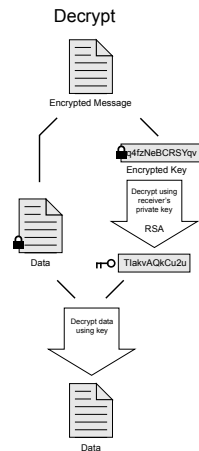
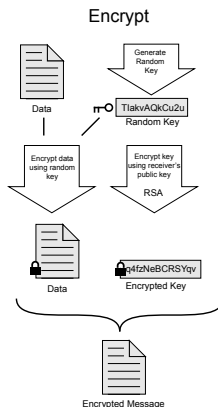
Encrypted mail

- 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: Wikipedia user [xaedes](#) & [jfreax](#) & [Acdx](#), CC BY-SA 3.0.



Certificates

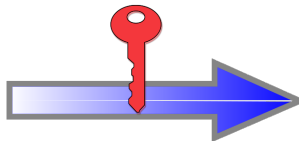
Identity Information and Public Key of Mario Rossi

Name: *Mario Rossi*
 Organization: *Wikimedia*
 Address: *via*
 Country: *United States*



Public Key
of
Mario Rossi

Certificate Authority
 verifies the identity of Mario Rossi
 and encrypts with its Private Key



Certificate of Mario Rossi

Name: *Mario Rossi*
 Organization: *Wikimedia*
 Address: *via*
 Country: *United States*
 Validity: *1997/07/01 - 2047/06/30*



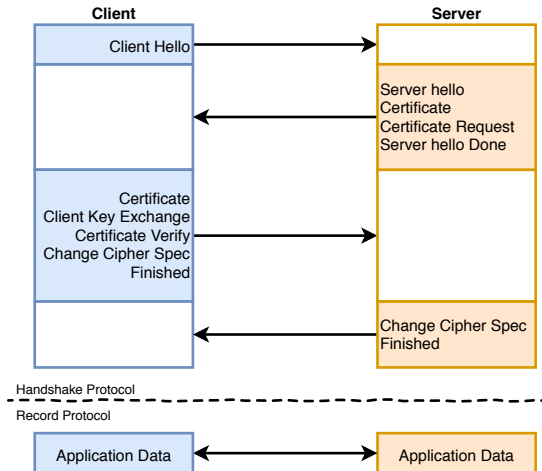
Public Key
of
Mario Rossi

Digital Signature
of the Certificate Authority

Digitally Signed by
Certificate Authority

Image credit: Wikipedia 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 hashcat¹ are able to compute 68,000,000,000² SHA1 hashes per second. Therefore we have to rise the cost of calculating a hash. One example is Bcrypt.

\$2a\$10\$N9qo8uL0ickgx2ZMRZoMyeIjZAgcf17p92ldGxad68LJZdL17lhWy

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/.](https://www.schneier.com/essays/))
- Niels Ferguson, Bruce Schneier, and Tadayoshi Kohno. Cryptography Engineering. John Wiley & Sons, 2010.