Cryptography and Key Management Basics

Erik Zenner

Technical University Denmark (DTU)
Institute for Mathematics
e.zenner@mat.dtu.dk

DTU, Oct. 23, 2007

Plan for Today

1. Talk 1: Cryptography and Key Management Basics
   (Erik Zenner)
2. Talk 2: Public Key Infrastructure
   (Christian D. Jensen)
3. Discussion
   Identify open questions

If you have questions, don't hesitate to ask (anytime).

1. Cryptographic Basics
   - Goals, Algorithms, and Keys
   - Symmetric vs. Asymmetric Cryptography
   - Important Examples

2. Key Management
   - Key Setup
   - Key Life-Cycle

3. Final Remarks

Outline

Protection Goals

Cryptography is not only about encryption. There exist many potential protection goals:

- Confidentiality
- Data Authentication
  - Integrity
  - Authenticity
  - Non-Repudiation
- Entity Authentication
- Key Establishment
- Anonymity
- ...

From Algorithm to Solution

Cryptography is only about the lowest "layers" when building a security solution. Higher layers are typically handled by *Security Engineers*.

Layer	**Example**
Algorithm / Primitive	AES, RSA
Scheme	AES-128-CTR, OAEP
Protocol (math)	Diffie-Hellman, Kerberos
Protocol (tech)	SSL/TLS, IPSec
Implementation	OpenSSL (C/C++)
Deployment	Portalen Single Sign-on

Cryptographic Keys

Standard Assumption:
The attacker knows everything about the security solution with the exception of the key. (Kerckhoffs' Principle)

Why?

- Protecting keys is easier than protecting whole implementations.
- Managing keys (generating, exchanging, storing, changing...) is easier than managing whole implementations.
- If only the key is secret, all other aspects of the security solution can be publicly scrutinised.

Consequence:
Protect the key by all means!

Purpose of Cryptographic Keys

The following is a categorisation of cryptographic keys according to what they are used for:

- **Data key:** Directly used for the cryptographical purpose, e.g. encryption or authentication.
- **Key-encryption key:** Used to encrypt other keys, e.g. in key exchange or key storage.
- **Master key:** Used to generate other keys, using a *key derivation function* (KDF).
  E.g.: $Session\_Key := KDF(Master\_Key, Session\_Number)$.

Symmetric Keys

Cryptographic operations typically involve a sender and a receiver (can be the same person).

Symmetric Keys: Sender and receiver use the same key (traditional case).

Properties:

- Short keys (80-256 bit)
- Fast algorithms

Special case: Passwords.

Asymmetric Keys

Asymmetric Keys: Sender and receiver use different keys:

- Public key: publicly available (e.g. for encryption)
- Private key: personal secret (e.g. for decryption)

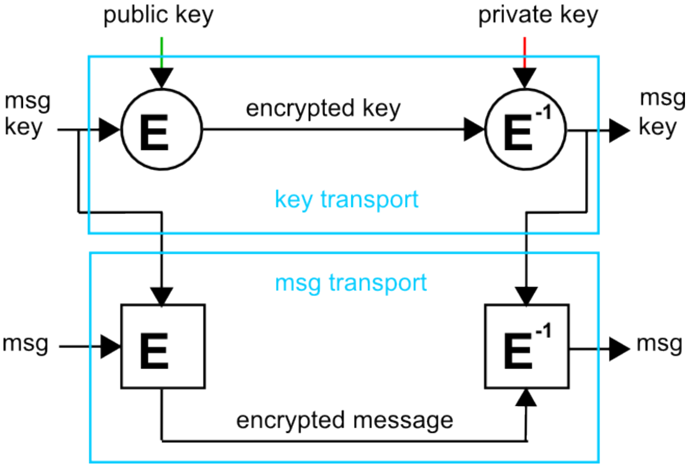
Properties:

- Long keys (e.g. RSA: 768-4095 bit)
- Slow algorithms

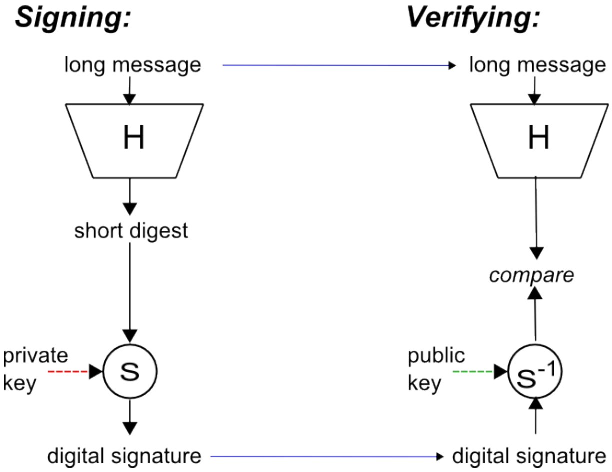
Advantage: Makes key transport easy if implemented properly.

Remark: Public "keys" are known to the attacker, i.e. no real keys.

Example 1: Hybrid Encryption



Example 2: Digital Signature



Algorithm Classification

If we organise cryptographic algorithms and protocols by

- protection goals and
- symmetric vs. asymmetric keys,

we obtain the following table:

	Symmetric	**Asymmetric**
Confidentiality	Sym. Encryption	Asym. Encryption
Data Authentication	MAC	Digital Signatures
Entity Authentication	Challenge/Response, Passwords	Challenge/Response, Zero Knowledge
Key Establishment	var.	var.

Important Examples

The following are examples for such algorithms and protocols:

- **Symmetric Encryption:** AEA (AES), DEA (DES), RC4
- **Asymmetric Encryption:** RSA, ElGamal
- **MAC:** HMAC, CBC-MAC
- **Digital Signatures:** RSA, DSA (DSS), ECDSA
- **Entity Authentication:** Password, PIN, OTP, Biometrics, Kerberos, Needham-Schroeder
- **Key Establishment:** Diffie-Hellman, IKE, Kerberos, Needham-Schroeder, TTP, Public-Key Infrastructure (PKI)

Outline

Key Generation

Any secret key material has to be generated. Main options:

- Generated by one party, then sent to the other (key transport).
- Generated by all parties working together (key agreement).
- Generated by a trusted third party and sent to all parties.

The form of the key material depends on its use (e.g., RSA keys are very different from AES-128 keys). See the relevant standard for details of format and generation.

With the exception of passwords, key generation typically requires some kind of random input.
$\Rightarrow$ Random number generation

Random Number Generation

Three types of random number generators (often confused):

- **Statistical random number generator:**
  Deterministic algorithm, not cryptographically secure
  (e.g., `rand()` from `stdlib.h` in C/C++).
  $\Rightarrow$ Never use this for cryptographic purposes!
- **Cryptographic random number generator:**
  Deterministic algorithm, cryptographically secure.
  Be very careful to seed correctly!
  Be careful to protect the inner state against attacker!
- **Real random number generator:**
  Uses measurements of physical processes to generate "real" randomness.
  Too expensive for most applications.

Key Exchange

In addition to being generated, the key also needs to be distributed to all legitimate parties.

- How to prevent others from seeing the key?
- How to authenticate the legitimate parties (sender and receiver)?
- How to distribute the key to the legitimate parties?
- How to verify that the legitimate parties received the key?

If done remotely: Use cryptography (many different solutions).
Sometimes easier: Personal key exchange.

Key Storage

Keys have to be stored somehow. Problems include:

- How to store keys such that only legitimate parties have access?
  - Use more keys?
  - Special case: Passwords (not stored in hardware)
- How to make backups such that lost keys can be recovered?
  - Prioritise: Availability or security?
  - Backups have to be secured, too!

Key Expiration

Keys can (in fact: should) expire sometime. Problems include:

- How to keep track of key expiration?
- Inform all users.
- Set up new key.
- What happens after expiration?
  - Archive old key material? How?
  - Delete old key material? How? Remember all copies!

Key Compromise

Worst case: Key has been compromised because

1. an attacker has potentially had access to the key, or
2. the corresponding cryptographic algorithm was broken.

What do we have to do?

- Key must no longer be used in the future.
  - Key Expiration (see above)
- All concerned parties have to be informed.
  - Key Revocation (see talk 2)
- Old documents have to be protected.
  - Re-Encryption? Re-Signing?
  - Destruction of old documents?

Outline

1. Cryptographic Basics
   - Goals, Algorithms, and Keys
   - Symmetric vs. Asymmetric Cryptography
   - Important Examples

2. Key Management
   - Key Setup
   - Key Life-Cycle

3. Final Remarks

How to Proceed

- No international standards on key management.
  - Probably to come in the next years
- No "one size fits all" solutions.
  - You have to know the usage scenario.
- Never build your own cryptographic solutions!
  - Use off-the-shelf (or off-the-standard) products.
  - If in doubt, ask cryptographers or IT security engineers.

References / Further Reading

The following books and references could be useful:

- N. Ferguson, B. Schneier: Practical Cryptography. Wiley, 2003.
- A. Menezes, P.C. van Oorschot, S.A. Vanstone: Handbook of Applied Cryptography.
  (parts of chapters 10,12,13; available online)
- NIST SP 800-57: Recommendation for Key Management.
  (3 parts; available online)