Cryptography

- Basics
- Encryption
- Hashes
- (MAC)
- (Public Key Infrastructure)
- Key Derivation Functions

Basics

- Kerckhoffs' Principle
  - Algorithm should not be “secure”
  - Security depends on the secret
- Don’t ever trust secret algorithms
- Don’t build your own cryptographic primitives
- Describe the algorithms used in a system
- Design for the future!
- Consider that things may change

Encryption - Block Cypher

- Encrypt a block of data (64/128 bits)
- DES (Data Encryption Standard)
  - Old - don’t use
- AES (American Encryption Standard)
  - Rijndael (.NET)
  - 128/192/256 bits key size
  - No practical attack exists
- Serpent
  - AES-finalist
  - 1/3 of the performance of Rijndael
  - Higher theoretical security
- Twofish
  - Expensive to exchange/generate the key

Encryption - Mode of Operation

- Block Cipher Mode
  - Padding
    - Fix the block size
    - Reversible
  - ECB (Electronic Codebook)
    - Don’t use!
    - Generates the same output for equal blocks
  - CBC (cipher block chaining)
    - XOR the previous block with the next
    - Use a random IV (initialisation vector)

Encryption

Summary

- Use 256 bit key size
- Use AES in CBC mode with a random IV
- Don’t use Stream-Cyphers or nonce based Cyphers

Hashes

- One way function
- Random mapping
- SHA-family is one of the most popular
- Round-based approached with work factor (BCrypt)

Hashes

Summary

- Depending on you use-case
- SHA-2 is fast
- BCrypt and SHA-3 (not in .NET yet) are slower
- If you use it to store a password use a salt

MAC (Message Authentication Code)

- Shared key
- Uses a hash function
- Used to ensure **integrity**
- Slow
- HMAC is the most popular

MAC

Summary

- Don’t use the same key
- Use HMAC-SHA-256
- Don’t use any weaker hash functions in combination with HMAC

Public Key Infrastructure

- RSA as the best known system
- Can be used for signing and encryption
- Used in combination with symmetric keys (hybrid encryption)

PKI - Encryption

- Use a Padding like OAEP (Optimal Asymmetric Encryption Padding)
- Don’t encrypt large data - instead encrypt a symmetric key and use that one for the data
- Encryption with RSA is slow - be aware of that
- The public key is used to encrypt - the private to decrypt - or the other way around (signature)

PKI - Signing

- Use the private key to sign and the public key to verify
- Verification is fast
- Hash the data before signing it

PKI

Summary

- Use a key length of at least 2048 bits
- Don’t use it to encrypt large data
- Use a padding for encryption (OAEP)

Key Derivation Functions

- PBKDF2 (Password-Based Key Derivation Function 2)
- Authentication or key-generation
- Hashing + Salt
- Iteration count
- Designed to be slow

Key Derivation Functions

Summary

- Use PBKDF2 with one of the known (secure) hash functions
- Provide a random salt
- Use at least 10’000+ iterations (Server)

References

- Cryptography Engineering
  - Niels Ferguson, Bruce Schneier, Tadayoshi Kohno
- https://crypto.stackexchange.com/
  - Good page for questions about cryptography
- Consider vendor specific manuals
  - https://android-developers.googleblog.com/2013/02/using-cryptography-to-store-credentials.html
  - https://docs.microsoft.com/en-gb/dotnet/api/system.security.cryptography