Paweł Rajba
pawel@cs.uni.wroc.pl
http://itcourses.eu/

Information Systems Security

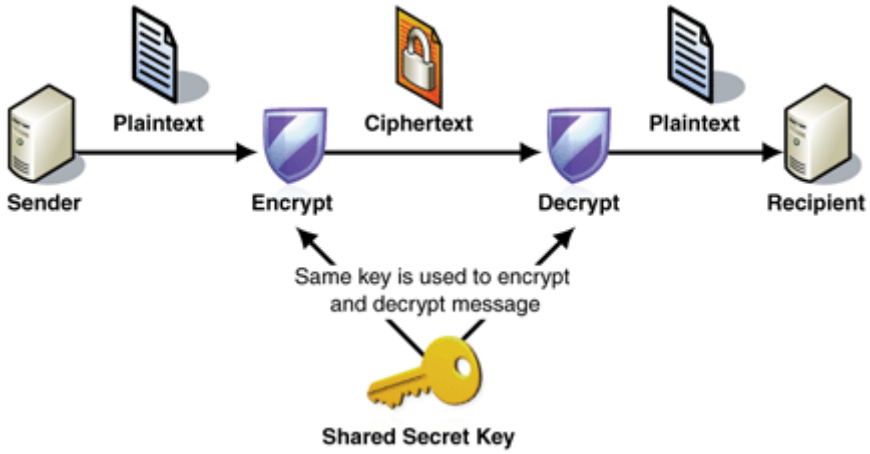
Cryptography introduction

Agenda

- Cryptography primitives
  - Secret key cryptography, Public key cryptography
  - Hash functions
  - Message Authentication Codes, Digital signatures
  - Key exchange, Key derivation function
- .NET Implementation
  - Classes hierarchy
  - Random numbers generation, Hash functions
  - Symmetric and asymmetric encryption
  - Slow functions, Recommendations, DPAPI

Cryptography primitives

- Secret key cryptography (1/2)

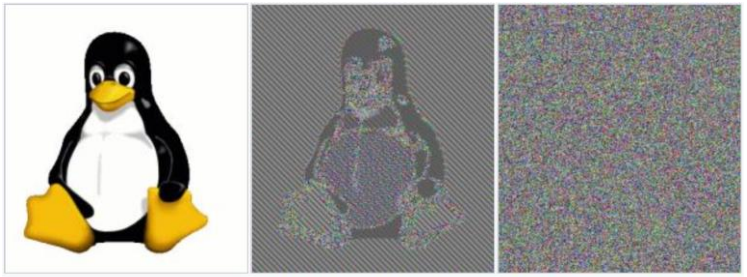


- Authenticated encryption

http://redpinata-development.com/bitcoin-academy/index.php/reader/items/public-key-cryptography.html

Cryptography primitives

- Secret key cryptography (2/2)
  - Realization: AES, Serpent, 3DES, Blowfish, Twofish
  - Block cypher mode: EBC, CTR, CBC, CFB, OFB, …



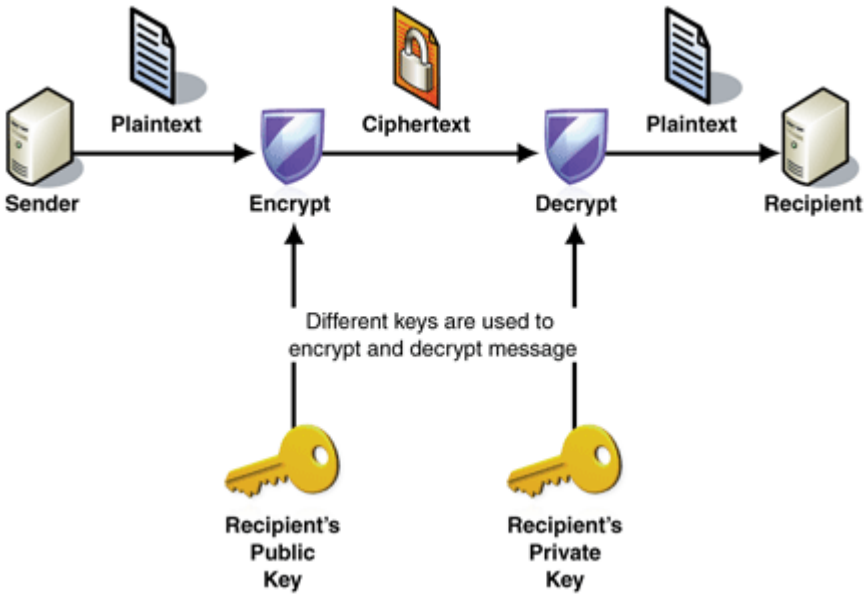
Original image | Encrypted using ECB mode | Modes other than ECB result in pseudo-randomness

  - Padding
- The biggest challenge?

https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation

Cryptography primitives

- Public key cryptography

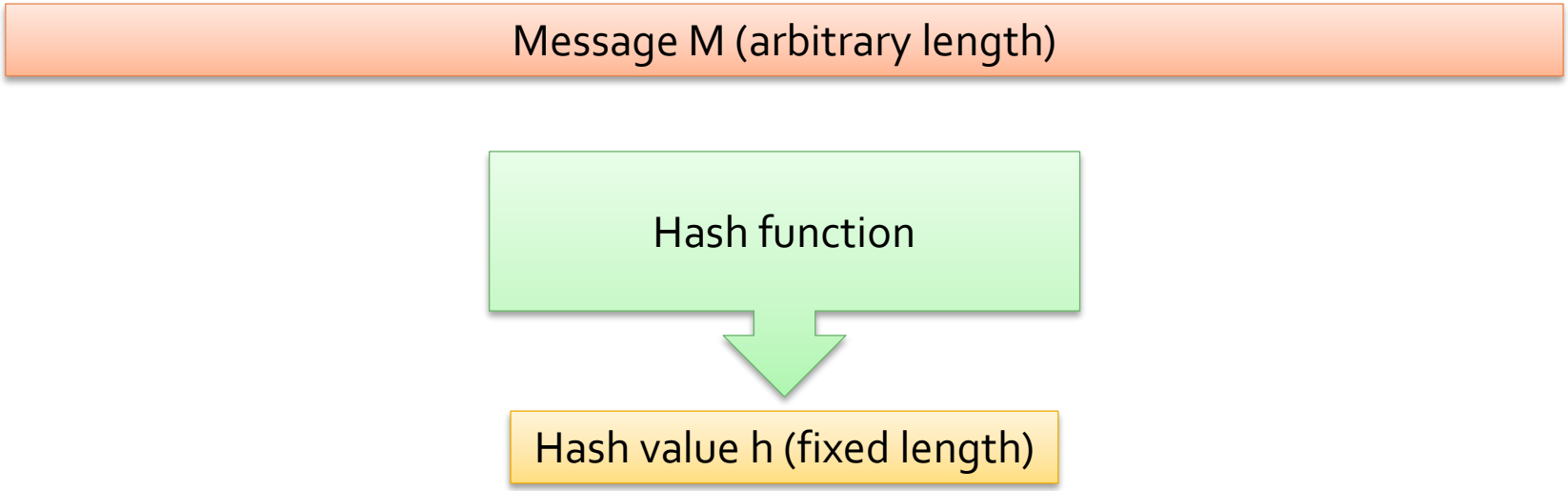


- Realization: RSA, DSA

http://redpinata-development.com/bitcoin-academy/index.php/reader/items/public-key-cryptography.html

Cryptography primitives

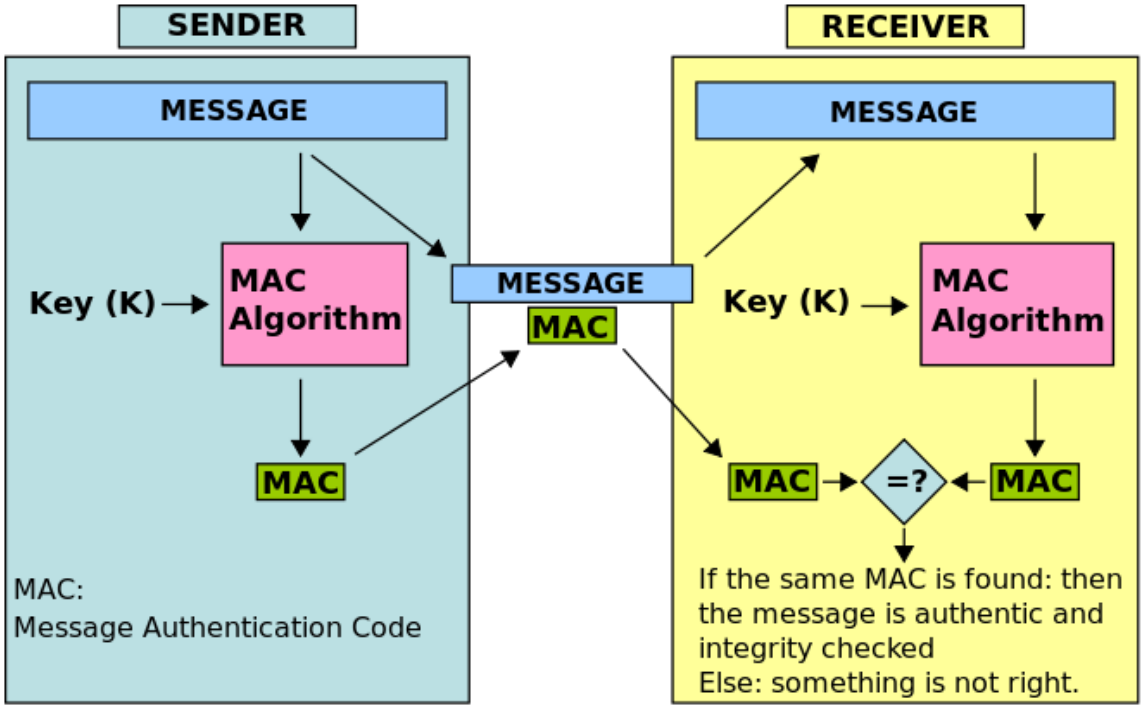
- Hash function



- Realization: SHA-2, SHA-3

Cryptography primitives

- Message Authentication Code

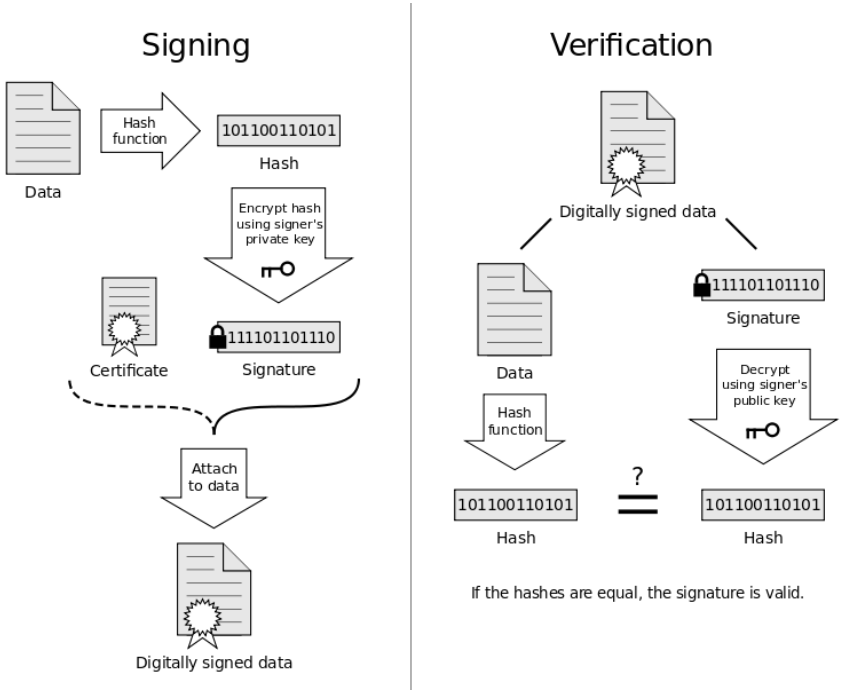


- Realization: HMAC-SHA512, CBC-MAC

https://en.wikipedia.org/wiki/Message_authentication_code

Cryptography primitives

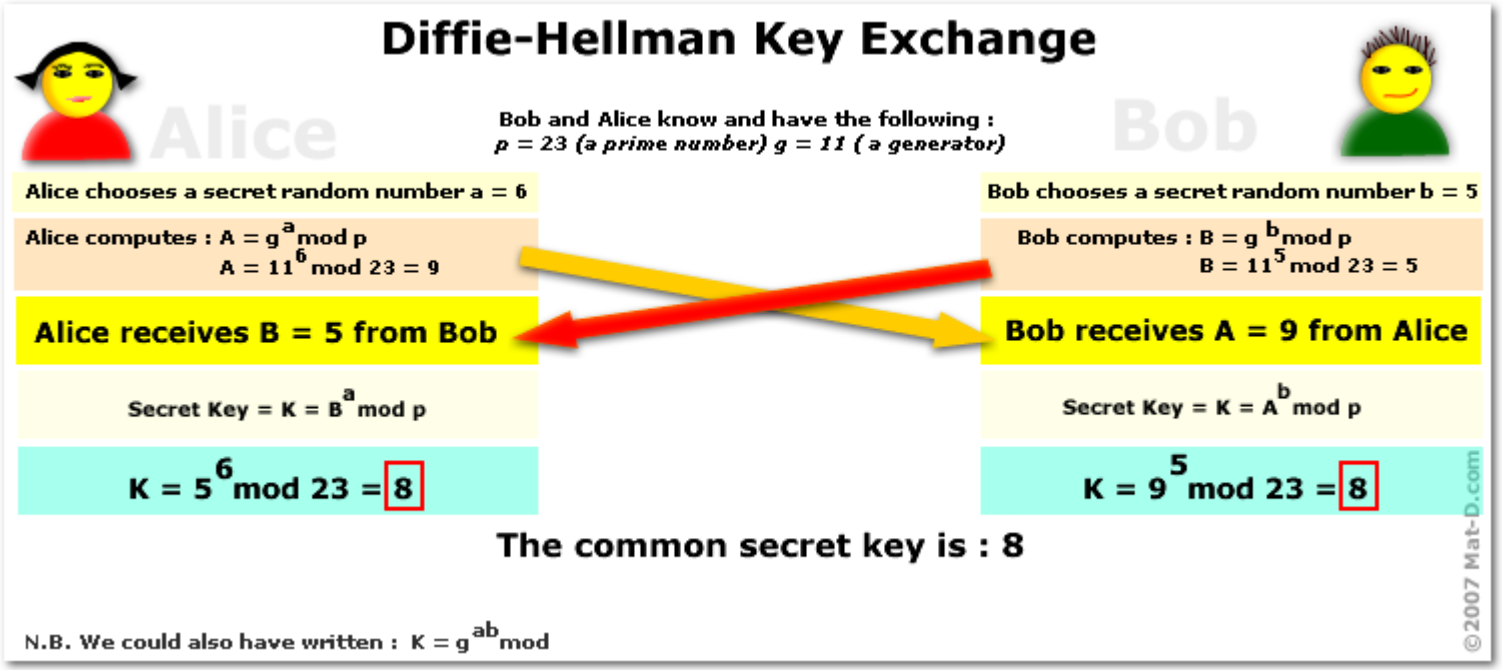
- Digital Signature



https://en.wikipedia.org/wiki/Message_authentication_code

Cryptography primitives

- Key exchange



- Realization: Diffie-Hellman, RSA

http://internetokracy.appspot.com/crypto1

Cryptography primitives

- Key derivation function
  - Take some source of initial keying material and derive from it
    one or more cryptographically strong secret keys
  - Popular functions: HKDF (fast) and PBKDF (slow)
  - Used for different scenarios, imporant one:
    - Generate a key from a password
    - Common schema:
      - PBKDF2(PRF, Password, Salt, iterations, derivedKeyLen)
        - PRF is e.g. keyed HMAC
  - More reading:
    - https://en.wikipedia.org/wiki/Key_derivation_function
    - https://en.wikipedia.org/wiki/HKDF
    - https://tools.ietf.org/html/rfc5869, https://eprint.iacr.org/2010/264.pdf
    - https://cryptography.io/en/latest/hazmat/primitives/key-derivation-functions/
    - https://crypto.stackexchange.com/questions/40971/what-is-the-difference-between-kdfs-for-key-derivation-vs-password-stretching
    - https://crypto.stackexchange.com/questions/4033/whats-the-difference-between-a-key-derivation-function-and-a-password-hash

Cryptography primitives

- Key stretching
  - Techniques to make a possibly weak key (a password) more secure against a brute force attack
  - Usually used in combination with KDF
- Key strengthening
  - Extending original password with a random salt, but afterward the salt is removed
  - This forces both the attacker and legitimate users to perform a brute-force search for the salt value

.NET implementation

- Main namespace
  - System.Security.Cryptography
- Classes hierarchy
  - SymmetricAlgorithm, AsymmetricAlgorithm, HashAlgorithm, RandomNumberGenerator (abstract)
  - Aes, Rsa, RC2 (abstract)
  - Implementations
    - *CryptoServiceProvider (e.g. AesCryptoServiceProvider)
      - Based on Windows Cryptography API (CAPI)
      - Certified by FIPS (Federal Information Processing Standards)
    - *Managed (e.g. AesManaged)
      - Based fully on manager code
      - Not certified by FIPS
    - *Cng (e.g. ECDsaCng) (CNG = cryptography next generation)
      - Available since Windows 2008/Vista
      - Only several algorithms supported nowadays

.NET implementation

- Random numbers generation
  - RNGCryptoServiceProvider (better)
  - System.Random (faster)

.NET implementation

- Hash functions
  - Classic hash functions
  - Message Authentication Codes (with password)
    - Based on hash functions: HMAC
    - Based block cipher algorithm: CBC-MAC
  - What is in .NET?
    - HashAlgorithm
      - MD5, MD160 (RIPEMD160), SHA1, SHA26, SHA384, SHA512
    - KeyedHashAlgorithm
      - HMACMD5, MACRIPEMD160, HMACSHA1, HMACSHA256, HMACSHA384, HMACSHA512
      - MACTripleDES

.NET implementation

- Symmetric encryption and decryption
  - Base class for symmetric algorithms
    - System.Security.Cryptography.SymmetricAlgorithm
  - Algorithms
    - Rijndael
      - Block sizes supported: 128, 160, 192, 224, and 256
    - Aes
      - Chosen by NIST to become a standard
      - Rijndael with limit set of block (128) and key sizes (128, 192, 256)
    - DES, TripleDes, RC2

.NET implementation

- Asymmetric encryption and decryption
  - Base class for symmetric algorithms
    - System.Security.Cryptography.AsymmetricAlgorithm
  - Algorithms
    - DSA
    - ECDiffieHellman
    - ECDsa
    - RSA

.NET implementation

- Asymmetric encryption and decryption
  - In this type of encryption, keys are stored in keys containers
  - Class CspParameters
    - Identified by KeyContainerName
  - Keys can be stored in a
    - Machine-Level RSA Key Container (available to everyone)
    - User-Level RSA Key Container (available to user)
      - Files in: %APPDATA%\Microsoft\Crypto\RSA

.NET implementation

- Slow functions
  - PBKDF2 (Rfc2898DeriveBytes class)
    - Example of PBKDF2
      - DK=PBKDF2(HMAC–SHA1, passphrase, ssid, 4096, 256)
        - 4096 – number of iterations
        - 256 – key length
  - BCrypt,SCrypt (no direct implementations)
- Applications:
  - Generating keys and initialize vectors
  - Slow functions can be used for storing passwords

.NET implementation

- Useful helper classes
  - BitConverter
    - ToInt32()
    - ToString()
  - Encoding.UTF8
    - GetString(byte[])
    - GetBytes(string)
  - System.Security.SecureString

.NET implementation

- Summary what to use:
  - Data privacy: Aes
  - Data integrity: HMACSHA256, HMACSHA512
  - Digital signature: ECDsa, RSA
  - Key exchange: ECDiffieHellman, RSA
  - Random number generation:
    - RNGCryptoServiceProvider
  - Generating a key from a password:
    - Rfc2898DeriveBytes

Source: http://msdn.microsoft.com/en-us/library/oss79b2x(v=vs.110).aspx

.NET implementation

- Windows Data Protection API (DPAPI)
  - Another way for symmetric encryption of data
  - One advantage: you don't care about keys
    - Windows take care
    - It can be: machine-level or user-level
  - ProtectedData class
  - File:
    - %APPDATA%\Microsoft\Protect\{SID}

References

- Cryptography model
  - http://msdn.microsoft.com/en-us/library/0ss79b2x(v=vs.110).aspx
- System.Security.Cryptography namespace
  - http://msdn.microsoft.com/en-us/library/System.Security.Cryptography(v=vs.110).aspx
- Cryptography API: Next Generation
  - http://msdn.microsoft.com/en-gb/library/windows/desktop/aa376210%28v=vs.85%29.aspx
- SecureString
  - http://www.codeproject.com/Tips/549109/Working-with-SecureString
  - http://stackoverflow.com/questions/818704/how-to-convert-securestring-to-system-string
- MAC, HMAC
  - http://en.wikipedia.org/wiki/Message_authentication_code
  - http://dev.ionous.net/2009/03/hmac-vs-raw-sha-1.html
- AES
  - http://en.wikipedia.org/wiki/Advanced_Encryption_Standard
- PBKDF2
  - http://en.wikipedia.org/wiki/PBKDF2
  - http://www.ietf.org/rfc/rfc2898.txt
  - http://www.tarsnap.com/scrypt/scrypt.pdf
- Password hashes in Java
  - http://howtodoinjava.com/2013/07/22/how-to-generate-secure-password-hash-md5-sha-pbkdf2-bcrypt-examples/
- Key containers
  - http://msdn.microsoft.com/en-us/library/f5csoacs(v=vs.110).aspx
  - http://msdn.microsoft.com/en-us/library/tswxhw92(v=vs.110).aspx
- DPAPI
  - http://msdn.microsoft.com/en-us/library/ms995355.aspx
  - http://en.wikipedia.org/wiki/Data_Protection_API