Criptosisteme

Securitate informatică

March 18, 2020

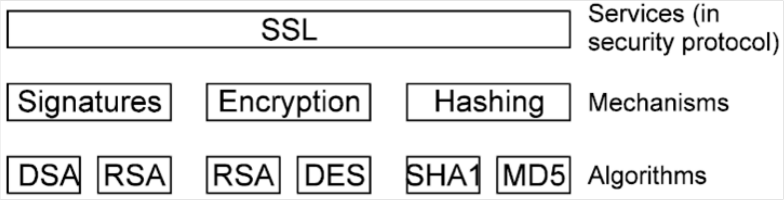
Mihai-Lica Pura

Cuprins

Security services, security protocols, security mechanisms, cryptosystems

- **Security services** are implemented through security protocols
- A typical security protocol provides one or more security services
- Security protocols are implemented using one or more **security mechanisms**
- Security mechanisms are implemented using **algorithms**
- **Cryptosystems** are one particular type of suits of algorithms used to implement security mechanisms

Security services, security protocols, security mechanisms, cryptosystems



Cryptosystems

- **cryptosystem** - a suite of cryptographic algorithms needed to implement a particular security service
- typically, a cryptosystem consists of algorithms for key generation, encryption, and decryption
- **cipher (or cypher)** - an algorithm for performing encryption or decryption
  - the cipher depends on a piece of auxiliary information, called **key**
  - without knowledge of the key, it should be unfeasible to decrypt the resulting ciphertext into readable plaintext

Cryptosystems

Ciphers classification:

- **symmetric key** algorithms - the same key is used for both encryption and decryption
- **asymmetric key** algorithms - a different key is used for each
- **block ciphers** - work on blocks of symbols usually of a fixed size
- **stream ciphers** - work on a continuous stream of symbols

Cryptosystems for security mechanism

- **Cryptographic hash algorithms**
  - used to provide integrity protection
  - can provide authentication (using Message Authentication Codes - MACs)
- **Encryption**
  - used to provide confidentiality
  - can provide authentication and integrity protection
- **Digital signatures**
  - used to provide authentication, integrity protection, and non-repudiation

Cryptographic Hash Functions

- are designed to take a string of any length as input and produce a fixed-length hash value
- are used to assure integrity and authentication
- - SHA1(securitate informatica) = 7ea5a44914425634e7ad6153bbcc4548fd98b51b
  - SHA256(securitate informatica) = bdbc3602d0a30e171ba3fa1f839701693303de05cf347cf050bcdf27428a
  - SHA256(compilatoare) = 05ace8a1c5faf8fbc2c72cffee83b15131a09292b69f9eefa6e5a4f2ce0347

Cryptographic Hash Functions

Requirements for a Cryptographic Hash Function:

- H can be applied to a block of data of any size
- H produces a fixed-length output
- H(x) is relatively easy to compute for any given x, making both hardware and software implementations practical
- For any given value h, it is computationally infeasible to find x such that H(x) = h (**one-way property**)
- For any given block x, it is computationally infeasible to find $y \neq x$ such that H(y) = H(x) (**weak collision resistance**)
- It is computationally infeasible to find any pair (x, y) such that H(x) = H(y) (**strong collision resistance**)

Cryptographic Hash Functions

Use:

- verifying the integrity of files or messages (MAC)
- password protection and verification (with care)
- in the generation of pseudorandom bits, or to derive new keys or passwords from a single, secure key or password
- widely used as file or Object identifier in e.g. Git, Mercurial and some p2p-file-sharing networks

Cryptographic Hash Functions

General purpose Hash functions:

- **MD4(128)** - not recommended anymore
- **MD5(128)** - not recommended anymore
- **SHA-1 (160)** - not recommended anymore
- **SHA-2 (224, 256)**
  - is state of the art and is recommended function to be used in e.g. X.509 Certificates
- **SHA-3 (224, 256, 384, 512)**
  - is build for future and very new and is not broadly supported at the moment

Cryptographic Hash Functions

SHA parameters

	SHA-1	SHA-256	SHA-384	SHA-512
Message digest size	160	256	384	512
Message size	$<2^{64}$	$<2^{64}$	$<2^{128}$	$<2^{128}$
Block size	512	512	1024	1024
Word size	32	32	64	64
Number of steps	80	64	80	80
Security	80	128	192	256

Notes: 1. All sizes are measured in bits.

2. Security refers to the fact that a birthday attack on a message digest of size *n* produces a collision with a workfactor of approximately $2^{n/2}$

Cryptographic Hash Functions

Hash and Passwords:

- general purpose hash functions are sometimes used for password hashing
  BUT
- they are to "too fast" on modern hardware, which makes them weak against brute-force attack
  SO
- just **DON'T use general purpose hashing algorithms for password hashing**
- instead use one of the **hash functions designed for password protection**
  - PBKDF2
  - bcrypt, scrypt
  - **Argon2** - the winner of the Password Hashing Competition in July 2015

Cryptographic Hash Functions

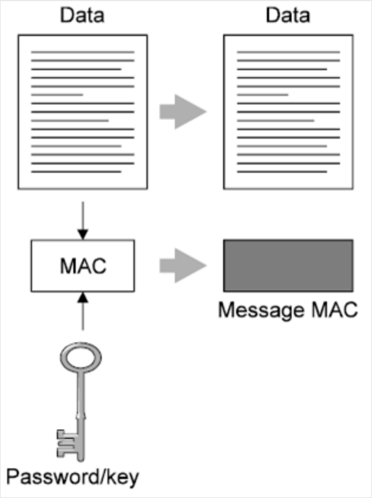
Argon2:

- designed by Alex Biryukov, **Daniel Dinu (Military Technical Academy graduate)**, and Dmitry Khovratovich from the University of Luxembourg
- source code: `https://github.com/p-h-c/phc-winner-argon2`

Message Authentication Codes

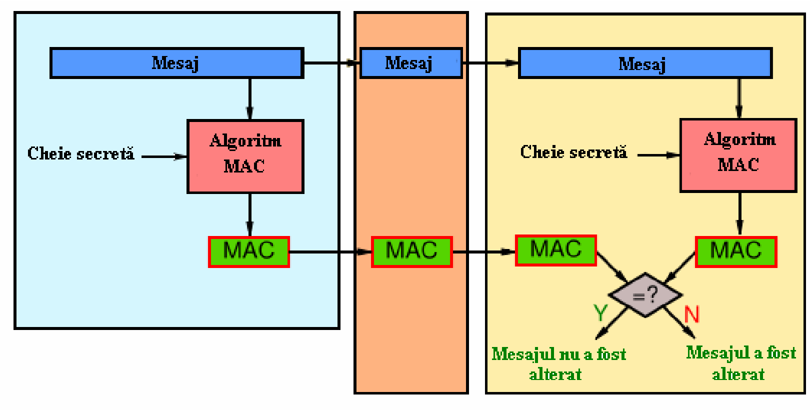
- objectives:
  - data integrity
  - data authentication
- does NOT provide non-repudiation of data origin
- similar to hash, but adds a key/password to the hash:
  - MAC = C(K, M)
- only the password holder(s) can generate/verify the MAC
- a MAC function is similar to encryption - one difference is that the MAC algorithm needs to be irreversible
- does not allow a distinction to be made between the parties sharing the key/password
- (crypto) algorithms: AES-MAC, DES-MAC, HMAC (SHA1/SHA2/SHA3), ...

Message Authentication Codes

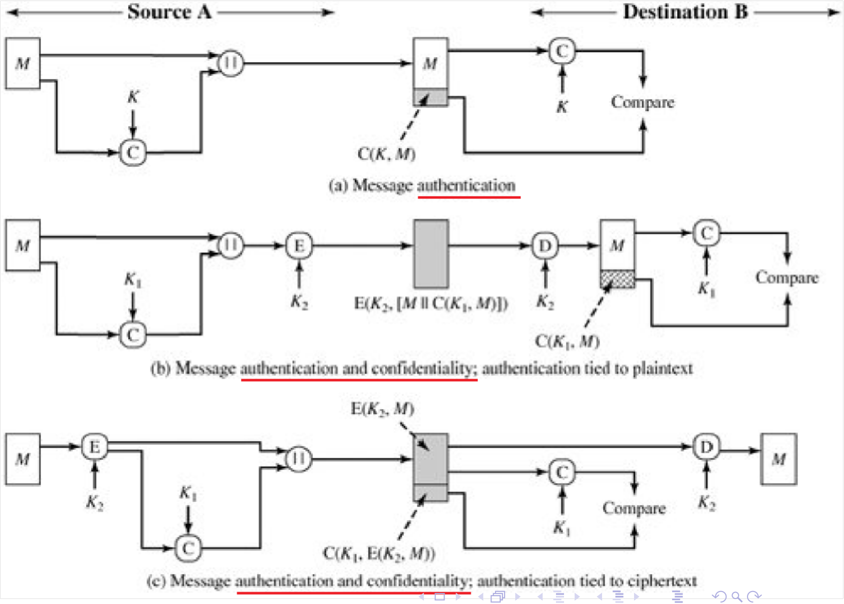


Message Authentication Codes

Message authentication



Message Authentication Codes



(a) Message authentication

(b) Message authentication and confidentiality; authentication tied to plaintext

(c) Message authentication and confidentiality; authentication tied to ciphertext

Message Authentication Codes

- **Hash Message Authentication Code (HMAC)**
  - RFC 2104 is from 1997
  - just a specific type of MAC that is based on hash functions
  - any cryptographic hash function may be used in the calculation of an HMAC
  - cryptographic strength of the HMAC depends upon:
    - the cryptographic strength of the underlying hash function
    - the size of its hash output
    - the size and quality of the key
- **Cipher-based Message Authentication Code (CMAC)**
  - CMAC or CMAC-AES (RFC 4493 from 2006)
  - MAC algorithm for block ciphers
  - CMAC can be calculated faster if target platform utilizes hardware optimization for block ciphers (e.g. dedicated AES opcodes)

Message Authentication Codes

- **Universal Hashing MAC (UMAC)**
  - RFC 4418 from 2006
  - MAC based on universal hashing
  - the resulting digest is then encrypted to hide the identity of the used hash function for additional security
  - UMAC has provable cryptographic strength and is usually a lot less computationally intensive than other MACs
  - UMAC's design is optimized for 32-bit architectures
  - VMAC - a closely related variant of UMAC that is optimized for 64-bit architectures
- **Poly1305**
  - Google has selected Poly1305 along with symmetric cipher ChaCha20 as a replacement for RC4 in TLS/SSL

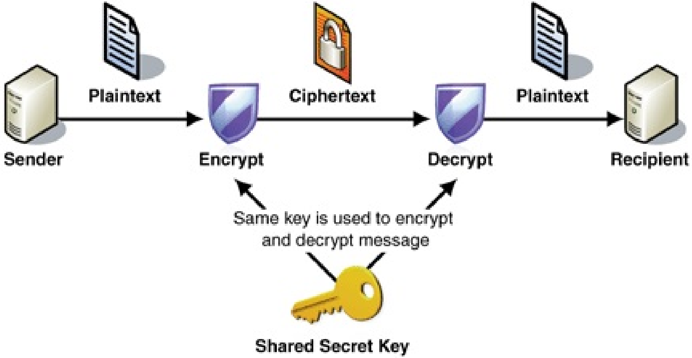
Symmetric-key cryptography

- objective:
  - data confidentiality (data privacy)
- **stream ciphers:**
  - RC4 - meanwhile considered insecure!
  - Salsa20 - very effcient and secure. ChaCha variant was selected as a replacement for RC4 in OpenSSL.
  - SEAL - one of the fastest stream ciphers
  - A5/1,A5/2 - are used in GSM and considered weak and insecure!
  - SNOW 3G - synchronous stream cipher
  - HC-256 - gains popularity
  - Rabbit - gains popularity

Symmetric-key cryptography

- **block ciphers:**
  - DES
  - AES
  - IDEA (1990, International Data Encryption Algorithm) - used in PGP
  - Blowfish (1993, Bruce Schneier)
  - Twofish - used by Microsoft
  - CAST-128, CAST-256 (Carlisle M. Adams)
  - Serpent (Ross Anderson, Eli Biham und Lars Knudsen)
  - RC5
- usages:
  - data encryption
  - MACs

Symmetric-key cryptography



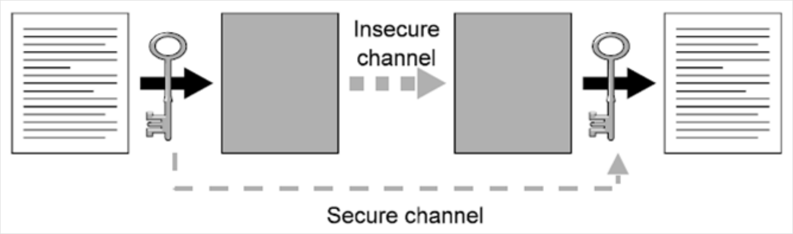
Symmetric-key cryptography

- plaintext is encrypted into ciphertext on the sender side
- the same key (key copy) is used to decrypt the ciphertext to plaintex on the recipient side
- as long as both sender and recipient know the secret key, they can encrypt and decrypt all messages that use this key
- **drawback**: both parties must know the same secret key - sharing of key must be done in a secure way

Symmetric-key cryptography

- uses a secret shared key
- the problem of communicating a large message in secret is reduced to the problem of communicating a small key in secret
- the new **Big Issue**: management (sharing) of the secret key

Symmetric-key cryptography



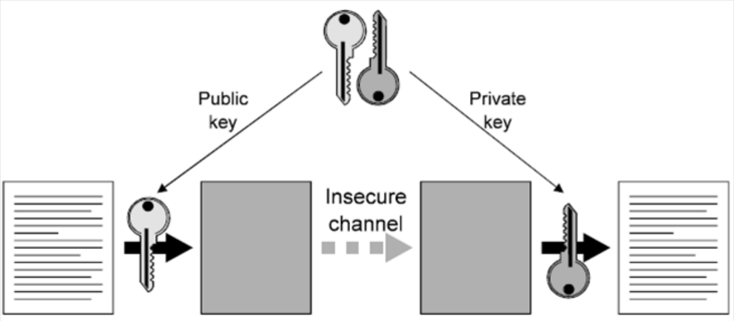
Asymmetric-key cryptography

- objective:
  - data confidentiality (data privacy)
  - data integrity
  - data authentication
  - non-repudiation
- (crypto) algorithms: RSA, DSA, Elliptic Curves-based DSA (ECDSA), El Gamal, etc.
- usages:
  - data encryption
  - key encryption (key-agreement)
  - digital signatures

Asymmetric-key cryptography

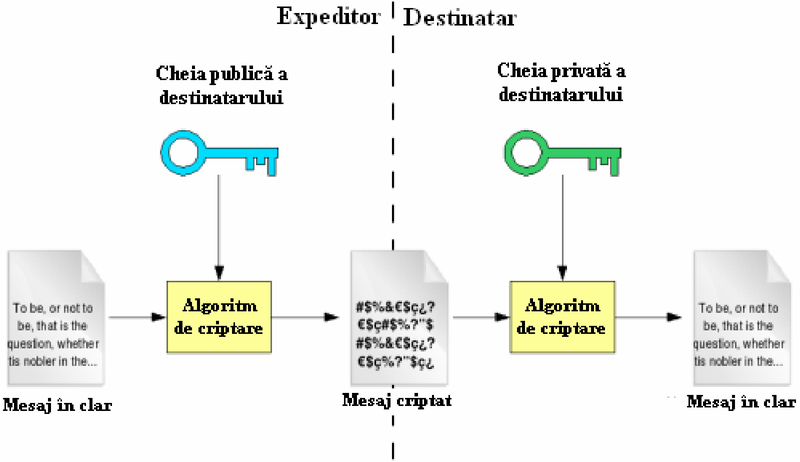
- fiecare utilizator are o pereche de chei:
  - cheie publică, disponibilă tuturor celorlalți utilizatori
  - cheie privată, care trebuie să rămână cunoscută numai posesorului acesteia
- cele două chei ale unui utilizator sunt în relație matematică, dar cheia privată nu poate fi obținută pornind de la cheia publică
- Anyone can encrypt/decrypt with the public key, only one person (the owner) can encrypt/decrypt with the private key
- Solves the sharing of the keys, but needs other infrastructures (e.g. PKI)

Asymmetric-key cryptography



Asymmetric-key cryptography

Data encryption



Asymmetric-key cryptography

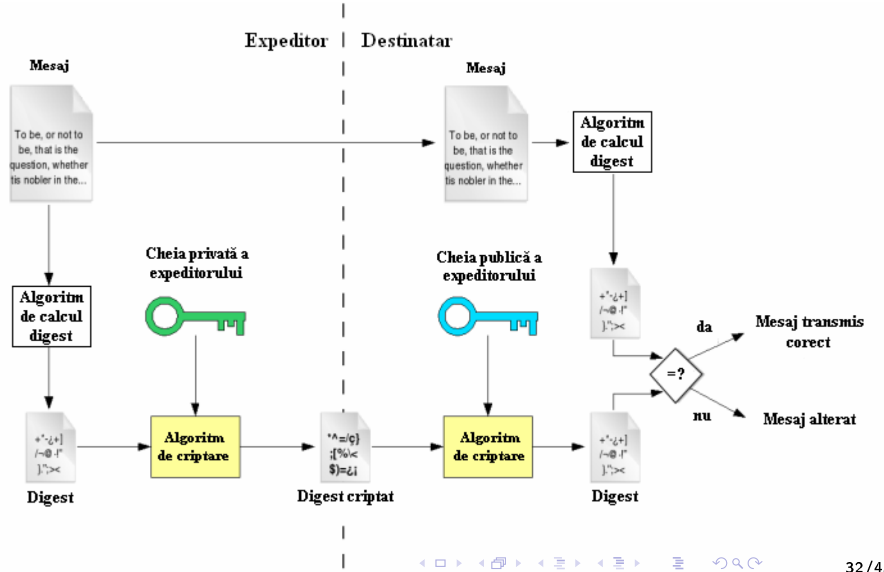
Data encryption

- Analogie
  - Cutia poștală a unei persoane:
  - Oricine care cunoaște adresa persoanei respective poate să îi pună o scrisoare în cutie
  - Numai proprietarul cutiei poștale, care are cheia acesteia, poate să citească scrisorile
- Pentru criptare și decriptare se folosesc chei diferite:
  - Cheia publică a destinatarului la criptare (ea este accesibilă oricui și deci oricine poate trimite un mesaj unui anumit destinatar)
  - Cheia privată a destinatarului la decriptare (ea este cunoscută numai de către destinatar și deci numai acesta poate decripta mesajul)

data confidentiality, data integrity

Asymmetric-key cryptography

Digital signatures



Asymmetric-key cryptography

Digital signatures

- Analogie
  - Sigilarea unui plic cu un sigiliu de ceară
  - Oricine poate să deschidă plicul
  - Numai posesorul sigiliului poate să sigileze plicul
- Numai expeditorul a putut cripta digestul în forma primită, deoarece cheia privată este cunoscută numai de către el
- Oricine poate decripta și verifica digestul, deoarece cheia publică este accesibilă tuturor

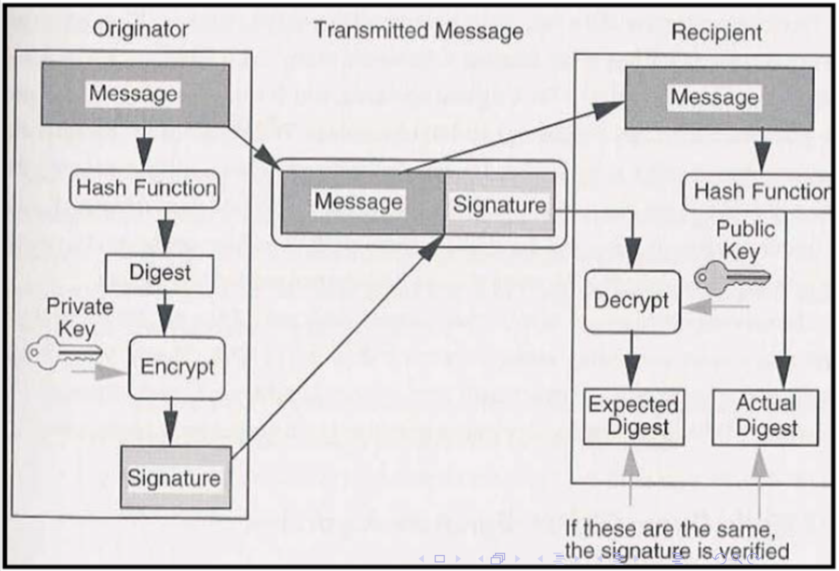
data integrity, data/origin authentication, data/origin non-repudiation

Digital signatures

- Recommended Key Lengths for electronic signatures:
  - 2048 bits key RSA
  - 2048 bits key DSA
  - 224, 256+ bits Elliptic Curves-based DSA (ECDSA)
- Un sistem de semnătură digitală implică 3 funcții (algoritmi):
  - funcția de generare de chei
  - funcția de semnare
  - funcția de validare a semnăturii

Asymmetric-key cryptography

Digital signatures



Asymmetric-key cryptography

- Problema centrală a criptografiei cu chei publice:
  - **Încrederea** că o anumită cheie publică este:
    - corectă,
    - aparține persoanei sau entității căreia se spune că aparține
    - nu a fost alterată sau modificată de către o terță parte rău-voitoare
- Soluții:
  - prin utilizarea certificatelor:
    - Web of Trust (folosită de către PGP)
    - Public-Key Infrastructure (PKI)
  - prin utilizarea identității ca și cheie publică:
    - Identity-based Cryptography (IBC)

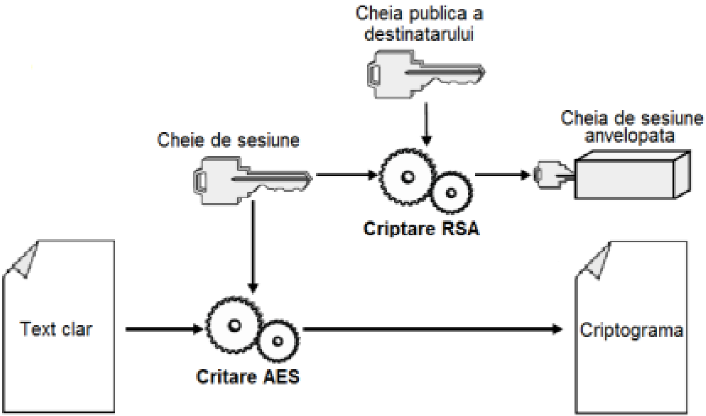
Hybrid cryptography

- Folosește pentru criptare atât criptografia simetrică, cât și cea asimetrică
- Avantaje:
  - Criptografia simetrică este mult mai rapidă decât cea cu chei publice
  - Gestiunea cheilor în cadrul criptografiei asimetrice este mult mai simplă
  - Combinarea lor elimină dezavantajul criptografiei simetrice dat de necesitatea de distribuire sigură a cheilor prin criptarea acestora folosind algoritmii asimetrici
  - Se eliminăși dezavantajul de viteză al criptografiei asimetrice prin criptarea mesajului propriu-zis folosind algoritmii simetrici

Hybrid cryptography

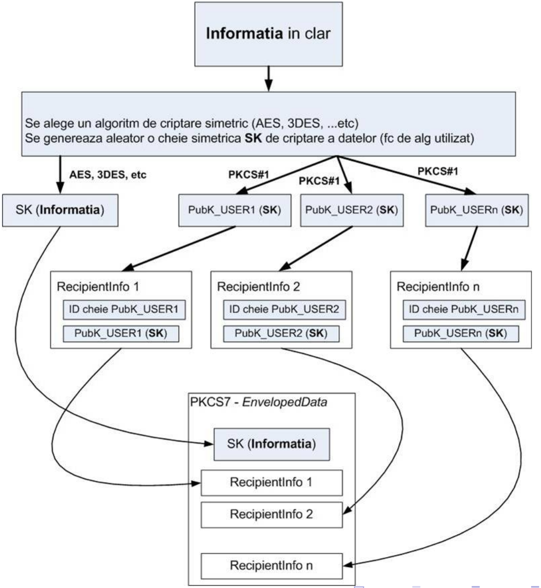
- Cum se face defapt criptarea?
  - Se alege unul din algoritmii de criptare simetrică
  - Se generează aleator o cheie pentru acest algoritm simetric
  - Se criptează mesajul folosind algoritmul simetric și cheia generată
  - Se criptează cheia simetrică generată folosind cheia publică a destinatarului
  - Mesajul criptat și cheia simetrică criptată sunt expediate destinatarului
  - Destionatarul decriptează cheia simetrică folosind cheia sa privată
  - Apoi, pe baza cheii simetrice decriptate, decriptează și mesajul propriu-zis

Hybrid cryptography



Hybrid cryptography

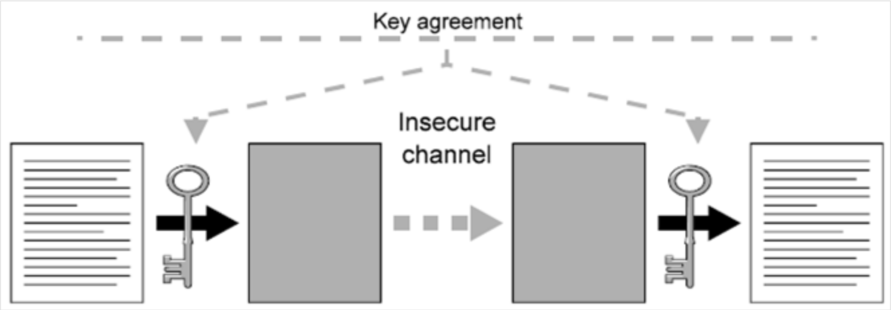
Criptarea pentru mai mulți destianatari



Key Agreement

- objective:
  - key sharing for data encryption & MACs (data confidentiality, data integrity, data authentication)
- establishment of the secret keys (key sharing), allows two parties to agree on a shared key
- public key-based algorithms: Diffie Hellman, ECDH, RSA
- provides part of the required secure channel for exchanging a conventional encryption key

Key Agreement



Public-Key Cryptography Standards

- PKCS#1 – std. pt criptografia cu alg. RSA
- PKCS#2 – incorporat in #1 - criptarea digest–urilor criptografice
- PKCS#3 – std. Diffie si Hellman Key Agreement
- PKCS#4 – incorporat in #1 – formatul cheii RSA
- PKCS#5 – std. pt. criptografia bazata pe parole
- PKCS#6 – std. pt. sintaxa extinsa a unui certificat digital – extensii
- PKCS#7 – std. pt. formatul mesajului criptografic

Public-Key Cryptography Standards

- PKCS#8 – std. pt. sintaxa cheii private
- PKCS#9 – std. pt. tipurile de atribute
- PKCS#10 – std. pt. formatul cererii de certificat
- PKCS#11 – std. pt. API–urile criptografice ale dispozitivelor hardware
- PKCS#12 – std. pt. token-uri software
- PKCS#13 – std. pt. criptografia cu curbe eliptice – in dezvoltare
- PKCS#14 – std. pt. generatoarele de nr. pseudoaleatoare – in dezvoltare
- PKCS#15 – standard pt. formatul informatiei pe token–urile criptografice

References

- Alexander Holbreich, Cryptography basics
  http://alexander.holbreich.org/cryptography-basics/
- Troy Hunt, Our password hashing has no clothes
  https://www.troyhunt.com/our-password-hashing-has-no-clothes/