

Lab. 2-3 Good to know

1. Terminologie

Plain Text = text în clar	Mesajul inițial, înainte de a fi criptat
Cifru	Sistem de semne convenționale cu care se transmit comunicări secrete.
Ciphertext (Criptograma)	Versiunea criptată a mesajului inițial
Cod	O regulă cu ajutorul căreia se va înlocui o literă (cuvânt) din mesajul inițial cu alt obiect, nu neapărat de același tip. Un exemplu este codul Morse: SOS = "...---..."
Criptare	Procesul de transformare al unui plaintext în ciphertext.
Decriptare	Procesul de transformare al unui ciphertext în plaintext.
Cheie	Informația secretă cunoscută doar de expeditor și de destinatar.
Criptografie	Știința criptării și decriptării mesajelor cu cheie sau sistem de criptare cunoscute.
Criptanaliză	Știința decriptării mesajelor fără a cunoaște cheia sau algoritmul de criptare folosit.
DEA	Data Encryption Algorithm
IDEA	International Data Encryption Algorithm
DES	Data Encryption Standard
AES	Advanced Encryption Standard
Criptosistem = Sistem criptografic	Un sistemul care realizează operații de criptare și decriptare.

2. DES - Data Encryption Standard

(https://ro.wikipedia.org/wiki/Data_Encryption_Standard) este un criptosistem simetric, bazat pe Arhitectura Feistel (https://en.wikipedia.org/wiki/Feistel_cipher). Metoda a fost selectată ca standard federal de procesare a informațiilor în Statele Unite în 1976. A fost dezvoltată de un colectiv de la IBM, o versiune inițială numindu-se LUCIFER. Criptosistemul a fost spart în 1998 prin brute-force = cautare exhaustivă, de către Electronic Frontier Foundation folosind o aparatură în valoare de 250000\$. DES este astăzi considerat nesigur, datorită cheii de 56 de biți, considerată prea scurtă. **Imbunătățire:** chei mai lungi: 3DES = Triple DES. În 3DES mesajul este criptat cu un algoritmul de criptare folosind setul de chei K, algoritmul de decriptare este aplicat cu un set de chei K' și din nou se aplică algoritmul de criptare cu setul de chei K. 3DES este considerat mai sigur.

Super imbunătățiri: posibilitatea de a folosi chei de 128, 192 și 256 biți: **AES**

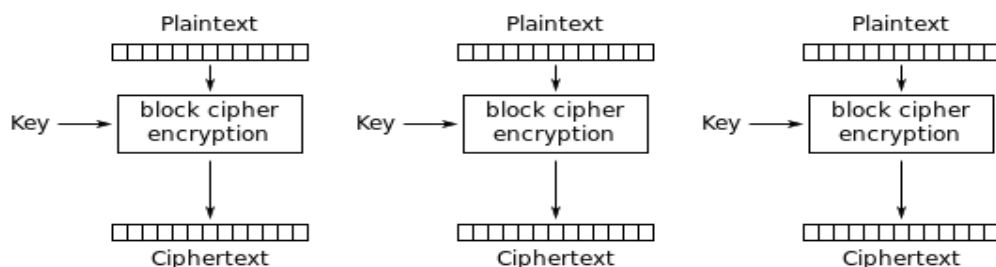
3. AES - Advanced Encryption Standard (<https://ro.wikipedia.org/wiki/AES>) Este un sistemul de criptare pe blocuri, fiind cea mai bună substituție pentru DES. A fost creat de Vincent Rijmen și Joan Daemen - **cifrul Rijndael**: operează pe blocuri de 128 de biți, operații orientate pe octet-cheie de criptare de lungime 128, 192 sau 256 de biți (doar 56 pentru DES), pentru o cheie de 128 biți numărul de combinații posibile: 3.4×10^{38} !

Moduri de operare AES

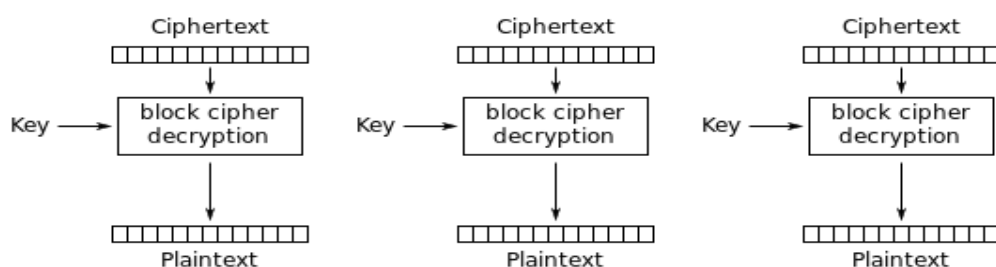
ECB - Electronic Code Book

[http://cryptowiki.net/index.php?title=Electronic_Code_Book_\(ECB\)](http://cryptowiki.net/index.php?title=Electronic_Code_Book_(ECB))

Criptarea constă în procesarea independentă a fiecărui bloc de 128 de biți cu aceeași cheie de criptare.



Electronic Codebook (ECB) mode encryption



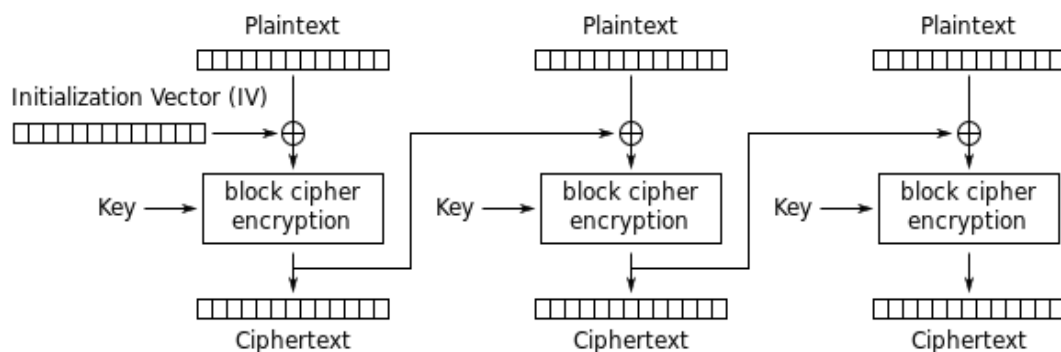
Electronic Codebook (ECB) mode decryption

Dezavantaje: mesaje în clar identice produc mesaje criptate identice; metoda nu este sigură pentru secvențe lungi de date. Un atacator poate schimba de asemenea ordinea blocurilor criptate.

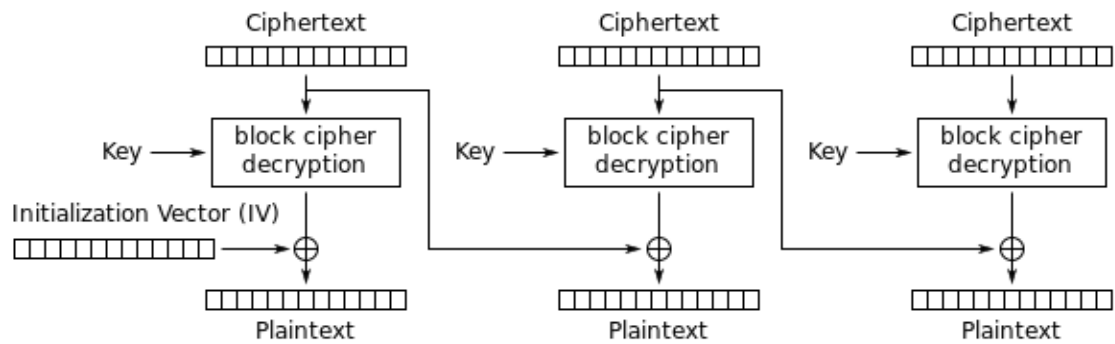
CBC - Cipher Block Chaining

[http://cryptowiki.net/index.php?title=Cipher_Block_Chaining_\(CBC\)](http://cryptowiki.net/index.php?title=Cipher_Block_Chaining_(CBC))

Fiecare bloc este criptat cu aceeași cheie; metoda presupune suplimentar o operație SAU EXCLUSIV (XOR) între blocul criptat curent și mesajul în clar următor.



Cipher Block Chaining (CBC) mode encryption



Cipher Block Chaining (CBC) mode decryption

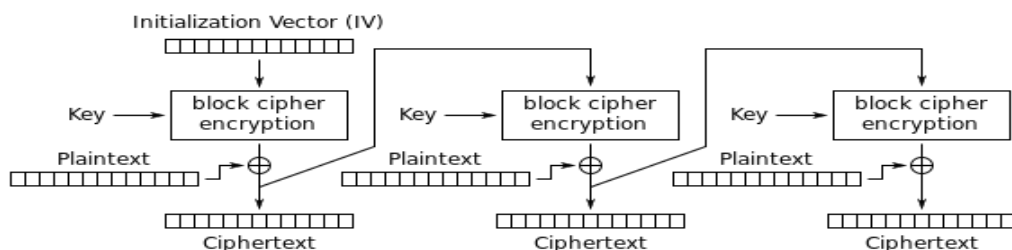
Pentru primul bloc se foloseste un vector de inițializare – asigură aleatorizarea în debutul procesului de criptare CBC; în absența acestuia mesaje în clar identice ar conduce la mesaje criptate identice-vectorul de inițializare trebuie transmis receptorului.

Dezavantaj CBC: erorile apărute în procesul de transmisie într-un bloc criptat afectează și blocurile următoare

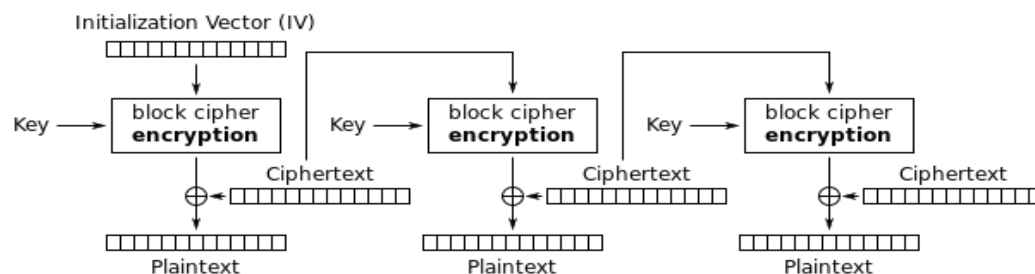
CFB - Cipher Feedback (Lab3)

[http://cryptowiki.net/index.php?title=Cipher_Feed_Back_\(CFB\)](http://cryptowiki.net/index.php?title=Cipher_Feed_Back_(CFB))

Modul de operare CFB este similar modului de operare CBC, cu diferența că se criptează blocul ciphertext obținut la pasul precedent (și nu blocul de plaintext) obținându-se un așa zis "flux de cheie" (keystream), care la rândul lui este supus unui XOR cu blocul curent de criptat obținându-se ciphertextul următor. Primul pas de criptare se face apelând din nou la un vector inițial.



Cipher Feedback (CFB) mode encryption

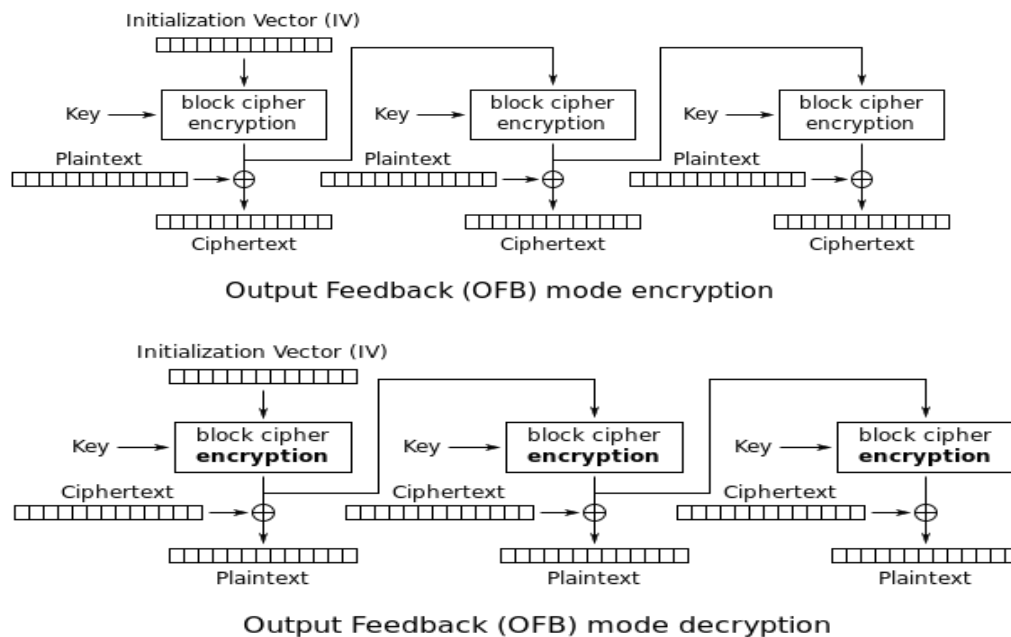


Cipher Feedback (CFB) mode decryption

OFB - Output Feedback (Lab 3)

[http://cryptowiki.net/index.php?title=Output_Feed_Back_\(OFB\)](http://cryptowiki.net/index.php?title=Output_Feed_Back_(OFB))

OFB-ul este asemănător cu CFB, cu diferența că algoritmul de criptare se aplică "fluxului de cheie", rezultatul fiind apoi supus unui XOR cu următorul bloc de plaintext, obținându-se următorul bloc criptat.



4. Aplicații Tehnici de marcă transparentă a imaginilor și a semnalelor

Criptarea poate fi privită ca o soluție pentru protecția drepturilor de autor. Dezavantajul major constă în faptul că de îndată ce informația a fost decriptată aceasta poate fi copiată fără restricții. Tehnicile de marcă transparentă rezolvă această problemă prin inserarea unei informații imperceptibile pentru sistemul auditiv sau vizual uman. Informația inserată poartă denumirea de **watermark (marcă)**. Tehnicile de marcă transparentă inserează marcajul într-un semnal gazdă de tip fișier, semnal audio, imagini statice sau secvențe video. Utilizarea tehnicilor de watermarking în aplicații reale presupune definirea a 2 operații pereche: *inserarea marcajului* și *extragerea marcajului*.

Exemple: 1. Protecția dreptului de autor: autorul introduce un marcaj cu informații legate de dreptul său de proprietate.

2. Protejarea la copiere: informație de marcaj inserată pentru permiterea sau interzicerea dreptului de copiere

3. Amprentare: inserarea unor informații specifice în fiecare copie adatelor originale.

5. Header Orice fișier imagine are un "header" în care este specificată informația referitoare la acea imagine (mărime, culoare, rezoluție, etc.). Această informație este reprezentată prin bytes. Structura headerului diferă de la format la format. De exemplu, headerul unui fișier BMP este format din primii 54 de octeți (bytes), un fișier PCX începe cu 128-bytes, etc, etc, etc. În link-urile de mai jos găsiți informații referitoare la header-ul unui fișier imagine, depinzând de formatul acesteia.

in English

http://www.fastgraph.com/help/image_file_header_formats.html

in Romanian

<http://ctmtc.utcluj.ro:8080/sites/pni/SACCDV/Laborator/Laborator01.pdf>

Când se criptează o imagine, aceasta se face cu tot cu headerul său (ceea ce corupe formatul imaginii). Asadar, pentru a (re)vizualiza imaginea criptată (cu ajutorul unui program de vizualizare) se "decupează" headerul fișierului de criptat (care poate fi

vizualizat folosind editorul hexa-Bless) si se suprapune peste headerul fisierului criptat (vizualizat cu Bless).

O alta posibilitate este de a folosi comenzi in Linux

Exercitiul 1. laborator

Instalati editorul Bless pe masina virtuala

- instalare pachet Bless (editor Hexa): `sudo apt-get install bless`
- configurare editor Bless (Ubuntu14, Lubuntu14):
`mkdir /home/nume_cont_utilizator/tempBless //creare director temporar pt Bless`
`sudo pico /usr/share/bless/default-preferences.xml // deschide fisierul`

Se modifica ultima linie din fisier `<pref name="ByteBuffer.TempDir"> </pref >` astfel:

```
<pref name="ByteBuffer.TempDir">/home/nume_cont_utilizator/tempBless</pref>
(fara spatii intre > si calea catre director, respectiv cale director <)
se salveaza fisierul si se inchide editorul pico
```

Exercitiul 2. laborator

Folosind comanda `openssl enc` criptati si decriptati un fisier imagine folosind modurile CBC, ECB, CFB sau OFB.

Hint 1: folositi comanzile `man openssl` si `man enc` pentru a consulta instructiunea `enc` in manual. Altfel,

Hint 2: `$ openssl enc ciphertype -e -in plain.txt -out cipher.bin \`
`-K 00112233445566778899aabbccddeeff \`
`iv 0102030405060708`

unde `ciphertype` face apel la unul din algoritmii de criptare CBC, ECB, CFB sau OFB. Mai precis `ciphertype` se inlocuieste cu `-aes-128-cbc`, `-aes-128-cfb`, `-bf-cbc`, `-des-cbc`, `-des-cfb`, etc.

`-in <file>` = input file

`-out <file>` = output file

`-e` = encrypt

`-d` = decrypt

`-K/-iv` = the key/ the initialization vector

Comparati fisierele criptate. Care dintre moduri este mai eficient?

6. The Crypto Library

Biblioteca OpenSSL este dotata cu un API numit EVP, care faciliteaza realizarea si compilarea programelor in C++. Detalii privind programarea in C++ utilizand functiile din API EVP pot fi gasite accesand link-urile

<https://www.openssl.org/docs/man1.0.2/crypto/crypto.html>

<https://wiki.openssl.org/index.php/EVP>

sau

https://www.openssl.org/docs/man1.0.2/man3/EVP_EncryptInit.html

Cu ajutorul interfetei API EVP puteti crea propriile voastre programe de criptare si decriptare.

Exercitiul 3. laborator

Utilizand functiile API EVP, scrieti un program C/C++ care primeste ca date de intrare numele a doua fisiere plaintext, criptotext, si un mod de operare (ECB, CBC, CFB sau OFB, la alegere), unde fisierul criptotext este rezultatul operatiei de criptare cu o cheie K (si eventual un vector de initializare, de exemplu,

IV = "\x01\x02\x03\x04\x05\x06\x07\x08\x09\x0a\x0b\x0c\x0d\x0e\x0f")

a fisierului plaintext. Programul trebuie sa determine cheia K utilizata pentru criptarea fisierului plaintext, stiind ca:

- aceasta cheie, K, are o lungime de 128 biti (16 octeti)
- cheia K este reprezentarea in hexa a unui cuvant din dictionarul englez cu o lungime de 16 caractere (completat cu caracterul spatiu, de cod hexa "\x20", in cazul in care aceasta lungime este mai mica de 16 caractere). De exemplu, pentru cuvantul "cipher" cheia K va fi
"\x63\x69\x70\x68\x65\x72\x20\x20\x20\x20\x20\x20\x20\x20"

Pe langa cheia K, programul trebuie sa listeze cuvantul care sta la baza ei, precum si numarul de incercari efectuate pana la gasirea cheii K.

La adresa <http://profs.info.uaic.ro/~nica.anca/is/words.txt>

se gaseste un fisier text cu cuvinte din dictionarul limbii engleze.

Exemplu de fisier Makefile pentru compilarea unui program C++ ce utilizeaza functii din API EVP:

```
INC=/usr/local/ssl/include/  
LIB=/usr/local/ssl/lib/  
  
all:  
    gcc -I$(INC) -L$(LIB) -o out source.c -lcrypto -ldl
```

7. One Way Hash Functions (Lab 3)

Do they really exist? - 1,000,000 US\$ Millennium Prize for a correct answer, offered by the Clay Mathematics Institute

Resurse in limba engleza:

<https://www.garykessler.net/library/crypto.html#skc>

http://www.aspencrypt.com/crypto101_hash.html

http://www.aspencrypt.com/manual_03.html

Resurse in limba romana:

<http://www.aut.upt.ro>

O **funcție hash** (hash function) este o funcție care primește ca intrare mesaje de dimensiune variabilă și returnează un mesaj de lungime fixă din care mesajul inițial nu poate fi recuperat in mod eficient (problema intractabila). O functie hash

reprezinta un algoritm matematic criptografic care genereaza un checksum (rezumat criptografic) unic pentru fiecare mesaj. Acest checksum se numeste Message Digest, Digest sau Hash.

Proprietati ale functiilor hash:

1. Doua mesaje diferite genereaza doua hash-uri diferite, fiecare mesaj genereaza un hash unic, nu exista doua mesaje diferite avand acelasi hash (collision-free).
2. Dimensiunea hash-ului este intotdeauna aceeași indiferent de marimea datelor care genereaza hash-ul.
3. Aceste functii se numesc one-way functions deoarece nu exista posibilitatea practica (in timp polinomial) ca din hash sa se recupereze mesajul initial (se presupune a fi intractabila - 1,000,000 US\$ for a correct answer!).
4. Functiile hash sunt extrem de sensibile la orice modificare oricat de mica (efect de avalansa). Vezi problema tema.
5. Hash-ul unui mesaj este mereu acelasi (pe acelasi mesaj).
6. Din punct de vedere computational, proprietatea de baza a unei functii hash este eficienta. Scopul este de a construi o functie hash cat mai simplu de implementat si cat mai rapida, atunci cand dimensiunea fisierului este oricat de mare. In general hash-ul unui fisier variaza intre 128-256 biti (maximum este de 512 biti) indiferent de dimensiunea intrarii.

Aplicatii ale functiilor hash

1. **Sistemele de parole.** Parola nu este salvata pe hard-disk ci se salveaza un hash al acesteia. In momentul in care userul introduce parola, se calculeaza hash-ul acesteia care este comparat apoi cu hash-ul salvat in momentul setarii initiale a parolei din fisierul de pe hard disk. Daca cele doua hash-uri sunt egale atunci parola este corecta. Daca parola a fost introdusa gresit atunci hash-ul ei va fi diferit de cel salvat pe hard disk, fiindca fiecare mesaj are propriul sau hash, nu exista doua mesaje (sau doua parole cu acelasi hash). Avantaj: daca un cracker compromite sistemul si are astfel acces la fisierul cu parole, acesta poate observa hash-ul parolelor si nu parolele. Iar din hash nu se poate obtine parola in timp polinomial (proprietatea 3).
2. **Garantarea integritatii unui fisier, program executabil etc.** Toti producatorii de software includ pe langa fisierul binar care reprezinta programul si hash-ul acestuia. Astfel dupa ce se descarca fisierul, se calculeaza hash-ul acestuia apoi se compara cu cel afisat pe site-ul producatorului. Daca hash-urile nu sunt identice atunci fisierul a fost modificat (virusat sau copiat cu erori). Un singur bit modificat in fisierul al carui hash il calculam, genereaza un Digest complet diferit (prop. 4 si 5).
3. **Semnarea digitala a unui mesaj.** Hash-ul mesajului se cripteaza cu cheia privata, iar rezultatul se numeste semnatura digitala. Fizic semnatura digitala este reprezentata print-un fisier care reprezinta o informatie (un text) ce se adauga la sfarsitul fisierului daca fisierul semnat este de tip ASCII.

Se foloseste pentru a garanta:

1. Integritatea mesajului;

2. Non-repudiation (identitatea celui care trimite mesajul, identitatea sursei) si ca acesta nu a fost impersonalizat (trimis de catre altcineva).

Exemplu: Alice doreste sa transmita un mesaj lui Bob dorind ca Bob sa poata fi sigur ca mesajul nu a fost modificat pe drum (integritate) dar si ca Alice este cea care a generat mesajul (garantarea identitatii sursei). Pentru aceasta Alice ataseaza mesajului (care nu trebuie neaparat sa fie criptat) o semnatura digitala obtinuta astfel:
a) Alice trece mesajul (fisierul) pe care doreste sa-l trimita printr-o functie one-way care genereaza un hash (Exemplu: md5);

b) Acest hash, care este unic, este apoi criptat folosind cheia privata a lui Alice. Rezultatul poarta denumirea de semnatura digitala;

c) Mesajul criptat obtinut este atasat mesajului initial si trimis catre destinatie;

d) La destinatie, Bob trece mesajul primit prin aceeasi functie de hashing ca si Alice, obtinand un sir unic pentru acel mesaj;

e) Bob decripteaza semnatura digitala atasata mesajului folosind cheia publica a lui Alice. Daca rezultatul obtinut este acelasi cu cel de la punctul d) este garantat faptul ca Alice este cea care a trimis mesajul si ca acesta nu a fost modificat pe drum.

Algoritmi Hashing

1. MD5 - Message Digest Version 5

- genereaza un hash pe 128 biti exprimat in 32 cifre hexazecimale;
- a fost creat de prof. Ronald Rivest de la MIT in 1991;
- a fost standardizat in RFC1321;
- este unul dintre cei mai folositi algoritmi de hashing in prezent (2009);
- din 2004 au inceput sa fie descoperite diferite vulnerabilitati ale algoritmului.

2. SHA1 - Secure Hash Algorithm Version 1

- genereaza un hash output pe 160 biti exprimat in 40 cifre hexazecimale;
- a fost creat si publicat de guvernul USA (NSA) in 1993;
- opereaza pe mesaje de maximum $2^{64}-1$ biti;
- este unul dintre cei mai folositi algoritmi de hashing in prezent (2009);
- este vulnerabil, se considera ca va fi inlocuit de un alt algoritm mai sigur;
- SHA2 este o noua familie de algoritmi de Hash publicati in 2001 care contine SHA-224, SHA-256, SHA-384 si SHA-512 dupa nr. de biti ai outputului;

Tipuri de atacuri asupra functiilor hash

Atacul unei functii criptografice (adica recuperarea unui mesaj criptat sau a cheii de criptare) are la baza exploatarea unei vulnerabilitati a functiei hash, care nu a fost luata in considerare atunci cand a fost implementata. Tipuri de atacuri:

1. Collision attack (atac la coliziune) Presupune gasirea a doua mesaje M si M' cu acelasi hash, $h(M) = h(M')$ in mai putin de $2^{(L/2)}$ iteratii. Acest tip de vulnerabilitate nu reprezinta o problema de securitate.

2. First pre-image attack (atac principal la preimage) Presupune gasirea unui mesaj M astfel incat $h(M) = H$, unde H este cunoscut, in mai putin de 2^L iteratii. Acest tip de vulnerabilitate reprezinta o grava problema de securitate.

3. Second pre-image attack (atac secundar la preimage) Presupune gasirea unui

mesaj M' , cunoscand un mesaj M , astfel incat $h(M) = h(M')$, in mai putin de 2^L iteratii. Acest tip de vulnerabilitate reprezinta o grava problema de securitate. L = lungimea hash-ului rezultat

MAC - Message Authentication Code

MAC se foloseste pentru garantarea identitatii sursei si integritatii mesajului. Ideea de baza din spatele unui MAC este urmatoarea: in loc de a calcula doar hash-ul unui mesaj, sursa adauga la mesaj o cheie secreta stiuta doar de catre destinatar si calculeaza hash mesaj + cheie. Catre destinatie se trimite mesajul in clar si MAC-ul sau. Astfel destinatarul calculeaza hash-ul mesajului primit + cheia secreta. Daca MAC-ul primit de la sursa este identic cu cel calculat local de catre destinatie exista garantia integritatii mesajului si identitatii sursei (non-repudiation).

Aplicatii laborator:

1. Generarea hash-ului (Message Digest) si a MAC-ului

Se vor aplica diversi algoritmi hash (MD5, SHA256, SHA1) prin intermediul comenzii

\$ openssl dgst dgsttype filename

unde **dgsttype** poate fi unul din algoritmi one-way hash: **-md5, -sha1, -sha256**

Comenzi pentru manual: **\$ man openssl** sau **\$ man dgst**

2. Keyed Hash sau HMAC

Generati o cheie hash (adica un MAC) pentru un fisier. Exemplu: se poate folosi comanda (pt algoritmul HMAC-MD5, analog pentru HMAC-SHA256, and HMAC-SHA1)

\$ openssl dgst -md5 -hmac "abcdefg" filename

Incercati aceeasi comanda cu chei de diverse lungimi. Observati rezultatul.

3. The Randomness of One-way Hash

Exercitiul 1

- Creati un fisier text de lungime arbitrara.
- Generati hash-ul acestuia, fie acesta h_1 , folosind unul din algoritmi hash (MD5, SHA256, sau SHA1).
- Schimbati un bit din fisierul de intrare (ceea ce se poate face cu editorul Bless).
- Generati hash-ul fisierului modificat, fie acesta h_2 .
- Comparati h_1 si h_2 .

Exercitiul 2 Scrieti un program C/C++ ce utilizeaza API EVP pentru implementarea functiilor hash. Programul va primi ca intrare doua fisiere text, f_1 si f_2 , ce au acelasi continut, cu exceptia unui singur caracter. Programul va calcula functiile hash asociate celor doua fisiere, utilizand algoritmi MD5 si SHA256, rezultand hash-urile $h1_md5$, $h2_md5$, respective $h1_sha256$, $h2_sha256$. De asemenea, programul va compara fisierele rezultate pentru fiecare algoritm in parte, adica: $h1_md5$ va fi

comparat cu h2_md5, iar h1_sha256 va fi comparat cu h2_sha256 (la nivel de octeti si va afisa numarul de octeti identici in cele doua fisiere.

Hint: http://www.openssl.org/docs/crypto/EVP_DigestInit.html

Tema laborator - Tema 1- 25 puncte

Implementati o infrastruktura de comunicatie ce foloseste criptosistemul AES pentru criptarea traficului intre doua noduri A si B cu urmatoarele caracteristici:

1. Se considera un nod KM (key manager) care detine trei chei pe 128 de biți K1, K2 si K3. Cheia K1 este asociata cu modul de operare CBC. Cheia K2 este asociata cu modul de operare OFB. K3 este utilizata pentru criptarea cheilor K1 si K2. Se considera ca vectorul de initializare are o valoare fixata cunoscuta din start atat de A cat si de B. De asemenea cheia K3 este detinuta din start si de A si de B.

2. Pentru a initia o sesiune de comunicare securizata nodul A trimite un mesaj catre B in care comunica modul de operare (CBC sau OFB), cerand in acelasi timp nodului KM cheia corespunzatoare. Cheia ceruta (K1 sau K2 in functie de modul de operare) este generata de catre KM in mod random, cu ajutorul unei librării criptografice, si apoi criptata ca un singur bloc cu AES folosind cheia K3, dupa care o trimite nodului A. Dupa ce A primeste cheia criptata de la KM, acesta o va trimite mai departe nodului B. A si B vor decripta cheia (K1 sau K2) pentru a incepe comunicarea. De asemenea nodul B va trimite catre A un mesaj de incepere a comunicarii.

3. Dupa primirea mesajului de confirmare de la B (referitor la inceperea comunicarii), A incepe sa trimita catre B continutul unui fisier criptat pe blocuri folosind modul selectat. Nodul B va decripta blocurile primite si va afisa rezultatul obtinut.

Observatie

Se accepta orice limbaj si folosirea oricarei librării pentru implementare. AES poate fi folosit ca algoritm de criptare pus la dispozitie de orice librerie criptografica. Modul de operare (CBC si OFB) se cere sa fie implementat in cadrul temei. Nu se cere rezolvarea de eventuale probleme de sincronizare intre noduri, interfata pentru noduri, sau un anumit protocol de comunicare.

Termen de predare strict prin e-mail la adresa liliana.cojocaru@info.uaic.ro pana pe data de 8 noiembrie (preferabil link spre un repository cu sursele). Finalizarea evaluarii temei va avea loc in laboratorul din data de 11 noiembrie.