

Data Security: Protocols

Integrity

Az üzenethitelesítés (integritásvédelem) feladata az, hogy a vételi oldalon detektálhatóvá tegyünk azon eseményeket, amelyek során az átviteli úton az üzenet valamilyen módosulást szenvedett el.

Módszerek:

- kriptográfiai ellenőrző összeg (CBC-MAC)
- kulcsolt hash
- rejtjelezés
- digitális aláírás
- spec.

Data Security: Protocols

Integrity

Külföldön dolgozunk, s alkalmanként szeretnénk rejtetten párbeszédet folytatni otthoni barátunkkal. Tilos azonban rejtjelezni a határon átlépő üzeneteket. Hitelesítő protokollok használata viszont nem tiltott (amikor is egy nyílt szöveg nyílt marad). Van megoldás?

A párbeszéd egy részlete, ahol Feri és Sanyi "beszélgetnek":

A: Halló, Feri vagyok.

A: *Halló, itt Jóska.*

B: *Szia, Laci.*

B: Szia, itt Sanyi.

A: *Jövő hétfőn továbbmegyek Rómába.*

A: Holnap indulok Athénba.

B: Menjek én is?

B: *Sajnos én nem tudok menni?*

A: *Mindenképp gyere.*

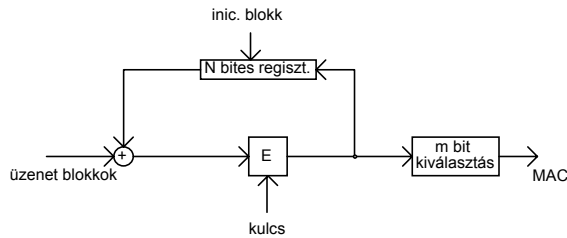
A: Semmiképp ne gyere.

Data Security: Protocols

Integrity - MAC

CBC-MAC (Message Authentication Code)

$[x_1, \dots, x_n, \text{MAC}(x_1, \dots, x_n)]$



$$y_i = E_k(x_i \oplus y_{i-1}), \quad y_0 = \text{inic.}, \quad i=1, \dots, n$$

$$\text{MAC} = g(y_n)$$

Data Security: Protocols

Integrity-MAC

CBC-MAC támadása: adaptívan választott üzenetek technikája

Tf. $IV=0$

1.)

$m_1 = m, \quad \text{MAC}_k(m)$

$m_2 = \text{MAC}_k(m), \quad \text{MAC}_k(\text{MAC}_k(m)) = E_k(\text{MAC}_k(m) + IV) = E_k(\text{MAC}_k(m))$

$m_3 = [m, 0], \quad \text{MAC}_k([m, 0]) = E_k(0 + \text{MAC}_k(m)) = E_k(\text{MAC}_k(m)) = \text{MAC}_k(\text{MAC}_k(m))$

2.)

$m_1, \quad \text{MAC}_k(m_1) = z_1$

$m_2, \quad \text{MAC}_k(m_2) = z_2$

$m_3 = m_1 \parallel z_1 + z_2 + y, \quad \text{MAC}_k(m_3) = z_3, \quad y \text{ tetszőleges bitsorozat}$

$m_4 = z_2 + y, \quad \text{MAC}_k(m_4) = E_k(z_2 + y) = z_3$

Data Security: Protocols

Integrity-MAC

Kulcsolt hash

titok prefix: $MAC_k(m) = h(k|m)$

$$MAC_k(m \parallel pad \parallel m') = h(k \parallel m \parallel pad \parallel m')$$

$$h'_{h(k|m)}(m') = h'_{MAC_k(m)}(m')$$

titok suffix: $MAC_k(m) = h(m|k)$

$$h(m) = h(m') \rightarrow h(m \parallel pad \parallel k) = h(m' \parallel pad \parallel k)$$

(születésnap paradoxon)

szendvics módszer: $MAC_{k,k'}(m) = h(k \parallel m \parallel k')$

Data Security: Protocols

Integrity-HMAC

HMAC:

$H(\text{key1} \parallel H(\text{key2} \parallel \text{message}))$

Pseudocode of HMAC

function hmac (key, message)

if (length(key) > blocksize) **then** key = hash(key) *// keys longer than blocksize are shortened*

else if (length(key) < blocksize) **then** key = key \parallel zeroes(blocksize - length(key))
// keys shorter than blocksize are zero-padded

end if

opad = [0x5c * blocksize] \oplus key *// Where blocksize is that of the underlying hash function*

ipad = [0x36 * blocksize] \oplus key *// Where \oplus is XOR*

return hash(opad \parallel hash(ipad \parallel message))

end function

Data Security: Protocols

Integrity

Tekintsük az

$m|E_k(\text{MDC}(m))$

integritásvédő kódolást. Milyen tulajdonsággal kell ez esetben az MDC-nek rendelkeznie?

Ütközés-ellenállóknak kell lennie:

Legyen $(m; m')$ MDC-ütközésre vezető üzenetpárt, m "nem gyanús" üzenetre kérve a kódolást, $E_k(\text{MDC}(m)) = E_k(\text{MDC}(m'))$ egyenlőség miatt m' üzenetre is ismert lesz a MAC.

Data Security: Protocols

Key exchange

Kriptográfiát alkalmazó rendszer biztonsága nem haladhatja meg a kulcsgondozása biztonságát.

Kulcsgondozási alapfeladatok:

- kulcsgenerálás
- kulcstárolás
- kulcscsere (szállítás, megegyezés)
- kulcsfrissítés
- kulcsvisszavonás

Data Security: Protocols

Key exchange

Egy kriptorendszer kulcshossza 100 bit. A kulcsot olyan generátorból nyerjük, amely 5 bites blokkokat állít elő. Ezen blokkokból azonos valószínűséggel olyanokat generál, amelyekben az 1 bitek darabszáma mindig kevesebb, mint a 0 bitek darabszáma. Az egymás utáni blokkok függetlenek.

a.) 20 db blokkot használunk kulcsként. Mennyi a tényleges kulcshossz, azaz mennyi az így generált kulcs entrópiája?

b.) Lehetséges-e, hogy 100 bit entrópiájú kulcsot előállítani az adott generátorra támaszkodva?

a.)

blokkfajta: db

3db 1bit 2db 0 bit \rightarrow 10

4db 1bit 1db 0 bit \rightarrow 5

5db 1bit 0db 0 bit \rightarrow 1

Összesen: 16-féle blokk $\rightarrow \log_2 16 = 4$ bit /blokk $\rightarrow 100/5 \cdot 4 = 80$ bit

b.) A 16 lehetséges blokkot egyértelműen leképezzük a 0000, ..., 1111 16 db félbájt egyikébe. 25 db blokkot ilyen módon leképezve 100 db pénzfeldobás bitet kapunk.

Data Security: Protocols

Key exchange

A kulcscsere protokollok szolgáltatásai (A fél számára):

Implicit kulcsHITELESÍTÉS

a protokoll sikeres lefutása után A meg lehet győződve arról, hogy rajta kívül csak a feltételezett másik fél, mondjuk B, és esetleg egy megbízható harmadik fél (pl. a kulcsszerver) férhet hozzá a protokoll során létrehozott kapcsolatkulcshoz

KulcsKONFIRMÁCIÓ

a protokoll sikeres futása után A meggyőződhet arról, hogy a másik résztvevő (B), valóban birtokában van a protokoll futása során létrehozott kapcsolatkulcsnak

Explicit kulcsHITELESÍTÉS

a protokoll egyszerre biztosítja az implicit kulcsHITELESÍTÉST és a kulcsKONFIRMÁCIÓT

KulcsFRISSESSÉG

a protokoll sikeres futása után A meg van győződve arról, hogy a létrehozott kapcsolatkulcs új, és nem egy korábban már használt (esetleg megfejtett) kulcs

PartnerHITELESÍTÉS

a protokoll sikeres futtatása után A meg van arról győződve, hogy a feltételezett másik résztvevő valóban részt vett a protokoll végrehajtásában.

Data Security: Protocols

Key exchange

Kulcscsere protokoll jellemzők:

- Szolgáltatásnyújtás iránya
- Megbízható harmadik fél használata
- Előzetesen szétszított információk
- Hatékonyság

Data Security: Protocols

Key exchange

Kulcscsere protokollok támadása

Támadás típusok

- passzív lehallgatás
- protokoll üzenetei törlése, módosítása, visszajátzása (replay)
- támadó a protokoll egyik résztvevője (megszemélyesítés, parallel sessions)

A támadó célja

(tipikusan) a protokoll által létrehozott kapcsolatkulcs megszerzése, vagy annak elhitése egy becsületes *A* résztvevővel, hogy *A* a kapcsolatkulcsot egy másik becsületes *B* résztvevővel hozta létre, miközben valójában a támadóval osztja meg azt.

A támadó rendelkezésére álló információk

a támadó rendelkezésére álló információk tekintetében általában azt feltételezzük, hogy a nyilvánosan elérhető információkon kívül a támadó nem rendelkezik további információkkal.

Data Security: Protocols

Key exchange

Tekintsük a Wide Mouth Frog protokoll következő (javított) változatát

$A \rightarrow S : A, \{“0”, B, K, T_A\}K_{as}$
 $S \rightarrow B : \{“1”, A, K, T_S\}K_{bs}$

Itt a “0” és az “1” irányjelző bitek, ahol a “0” jelöli a szervernek küldött üzeneteket és “1” jelöli a szerver által küldött üzeneteket.

- Melyik osztályba tartozik a protokoll?
- Melyik szolgáltatások nyújtja a protokoll A számára?
- Melyik szolgáltatások nyújtja a protokoll B számára?

kulcs-szállító protokoll

implicit kulcsHITELESÍTÉS
kulcsfrissesség

implicit kulcsHITELESÍTÉS
explicit kulcsHITELESÍTÉS
kulcsfrissesség
partnerHITELESÍTÉS

Data Security: Protocols

Key exchange

Tekintsük a következő kulcscsere protokollt:

1. $A \rightarrow B : \{K\}K_b$
2. $B \rightarrow A : \{N\}K$
3. $A \rightarrow B : \{\text{sig}_A(N)\}K$

- K_b B nyilvános kulcsa,
- K egy A által generált friss kapcsolatkulcs,
- N egy B által generált friss véletlenszám
- $\text{sig}_A(N)$ jelöli A aláírását N-en

A aláírása ellenére sem lehet B biztos benne, hogy a K kulcsot rajta kívül csak A ismeri!

Rosszindulató X fél meg tudja személyesíteni A-t B felé:

$A \rightarrow X : \{K\}K_x$
 $X \rightarrow B : \{K\}K_b$
 $B \rightarrow X : \{N\}K$
 $X \rightarrow A : \{N\}K$
 $A \rightarrow X : \{\text{sig}_A(N)\}K$
 $X \rightarrow B : \{\text{sig}_A(N)\}K$

Javítás

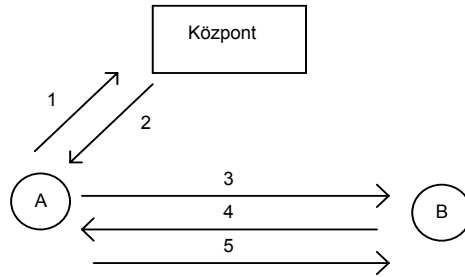
1. $A \rightarrow B : \{K\}K_b$
2. $B \rightarrow A : \{N\}K$
3. $A \rightarrow B : \{\text{sig}_A(B, K, N)\}K$

Data Security: Protocols

Key transport

Needham-Schroeder protokoll

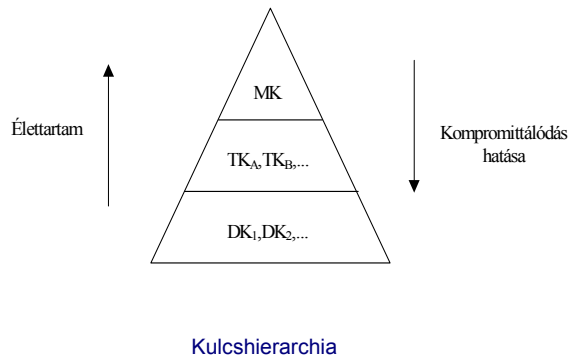
1. A → Kp: $ID_A, ID_B, R1$
2. Kp → A: $E_{TKA}(R1, ID_B, DK, E_{TKB}(DK, ID_A))$
3. A → B: $E_{TKB}(DK, ID_A)$
4. B → A: $E_{DK}(R2)$
5. A → B: $E_{DK}(R2 - 1)$



Data Security: Protocols

Key transport

Needham-Schroeder protokoll



Data Security: Protocols

Key transport

Needham-Schroeder protokoll

1. $A \rightarrow Kp: ID_A, ID_B, R1$
2. $Kp \rightarrow A: E_{TKA}(R1, ID_B, DK, E_{TKB}(DK, ID_A))$
3. $A \rightarrow B: E_{TKB}(DK, ID_A)$
4. $B \rightarrow A: E_{DK}(R2)$
5. $A \rightarrow B: E_{DK}(R2 - 1)$

R1, R2: visszajátszásos támadás megakadályozása

Mit hisznek a felek a protokoll lefutása után?

- B azt hiszi, hogy olyan féllal áll szemben, aki ismer egy "valamikor" A-nak küldött DK kapcsolatkulcsot
- 'A' nincs meggyőzve arról, hogy B a partnere

Régi, a C támadó által időközben megismert DK kulcs felhasználható támadásra (C a protokoll 3.lépésétől kezd)

Data Security: Protocols

Key transport

Javított protokoll ("Kerberizált"):

1. $A \rightarrow Kp: ID_A, ID_B$
2. $Kp \rightarrow A: E_{TKA}(T, L, DK, ID_B), E_{TKB}(T, L, DK, ID_A)$
3. $A \rightarrow B: E_{DK}(T', ID_A), E_{TKB}(T, L, DK, ID_A)$
4. $B \rightarrow A: E_{DK}(T'+1)$

T, L a DK kulcs előállításának időpontja, ill. élettartama.

2. lépés: A ellenőriz : $t2 \in [T, T+L]$?
3. lépés: B ellenőriz : $t3 \in [T, T+L]$?
- 3.lépés: $E_{DK}(T', ID_A)$
A meggyőzi B felet a jelenlétéről (1-2. lépésig egy C támadó is eljuthat),
T' friss elemmel kihívást is küld egyúttal B felé.
- 4.lépés: B meggyőzi A felet a jelenlétéről

Időszinkron és biztonság kérdése

Data Security: Protocols

Key exchange/agreement

1. Verzió:

1. A→B: $ID_A, E_B(R)$

passzív támadó: -

aktív támadó: megszemélyesítés sikeres

2. Verzió:

1. A→B: ID_A, k_A^p

2. B→A: ID_B, k_B^p

3. A→B: $E_B(R1)$

4. B→A: $E_A(R2)$

5. A: , B: $k=F(R1,R2)$

MIM (Man-In-the-Middle), "támadó közében" támadás sikeres:

C az A és B közé áll:

1'. A→C: ID_A, k_A^p

C→B: ID_A, k_C^p

2'. B→C: ID_B, k_B^p

C→A: ID_B, k_C^p

Data Security: Protocols

Key agreement

3. Verzió (interlock protokoll) :

1. A→B: ID_A, k_A^p

2. B→A: ID_B, k_B^p

3. A→B: $/E_B(R1)/$ (blokk első fele)

4. B→A: $/E_A(R2)/$

5. A→B: $//E_B(R1)//$ (blokk második fele)

6. B→A: $//E_A(R2)//$

7. A: , B: $k=F(R1,R2)$

Mi bizonyítja B számára a fenti protokollban, hogy A-val beszél? Semmi.

4. Verzió (kulcstanúsítvány alkalmazása)

1. A→B: ID_A, k_A^p, C_A

2. B→A: ID_B, k_B^p, C_B

3. A→B: $E_B(R1)$

4. B→A: $E_A(R2)$

5. A: , B: $k=F(R1,R2)$

Data Security: Protocols

Key agreement

Diffie-Hellman kulcscsere (alapprotokoll)

diszkrét hatványozás (kommutatív egyirányú függvény)

GF(q): egy 'nagy méretű' véges test

g: primitív elem

1. A → B: g^{R1}
2. B → A: g^{R2}
3. A: $(g^{R2})^{R1}$
B: $(g^{R1})^{R2}$

kommutativitás: $k=(g^{R2})^{R1}=(g^{R1})^{R2}$

- lehallgató C támadó nem képes az R1 illetve R2 véletlen elemeket megállapítani

- aktív C támadó képes "támadó a középén" támadásra

Kulcshitelesség nincs biztosítva! → hitelesítés: pl. SSL (Secure Socket Layer) protokollban

Data Security: Protocols

Public key certificate

ISO X.509 tanúsítvány

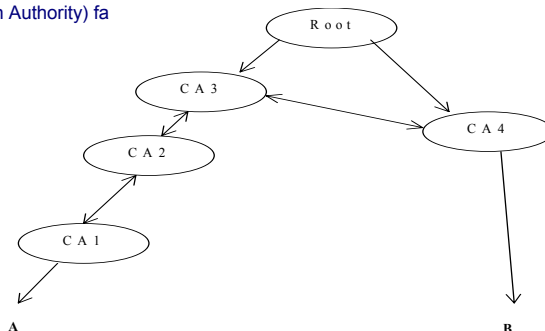
Az X.509 tanúsítvány komponensei az alábbiak:

- .Verziószám
- .Sorozatszám
- .Algoritmus azonosító
- .Tanúsítvány kibocsátó (CA) azonosítója
- .Tanúsítvány érvényességi időtartam (Not Before Date, Not After Date)
- .A tanúsítvány tulajdonosának azonosítója
- .A tanúsítandó **publikus kulcs**
- .A **digitális aláírás** (a fenti adatokra)

Data Security: Protocols

Public key certificate

CA (Certification Authority) fa



- A ismeri CA1, B ismeri CA4 hiteles publikus kulcsát.
- Nyíl iránya a hitelesítés irányát mutat.
- Root CA publikus kulcsa már nem tanúsítvánnyal igazolt, hitelesen ismertnek feltételezett.

Data Security: Protocols

Public key certificate

Amikor *A* és *B* egymással egy nyilvános kulcsú kriptográfiára épülő protokollt szeretne futtatni, be kell szereznie a másik fél nyilvános kulcsát igazoló tanúsítványokat.

Például *A* elküldi a tanúsítványok alábbi listáját *B* számára:

$$A \rightarrow B: \{k_A^p\}CA1, \{k_{CA1}^p\}CA2, \{k_{CA2}^p\}CA3, \{k_{CA3}^p\}CA4$$

ahol $\{k\}CA$ azt jelenti, hogy *k* kulcsot *CA* tanúsítja.

B a listát fordított sorrendben dolgozza fel:

1. A lista utolsó eleme és CA4 általa hitelesen ismert publikus kulcsa alapján megállapíthatja CA3 hiteles publikus kulcsát.
2. A lista megelőző, harmadik eleme alapján - most már CA3 hiteles kulcsa alapján - megállapíthatja CA2 hiteles publikus kulcsát.
3. A lista második eleme alapján - most már CA2 hiteles kulcsa alapján - megállapíthatja CA1 hiteles publikus kulcsát, majd ennek alapján az első tanúsítvány felhasználásával elér a céljához, *A* publikus kulcsa hitelességének ellenőrzéséhez.

Data Security: Protocols

Secret sharing

1.version:

Cut the secret key into N portions and distribute these portions among N people (secret-keepers)

Advantage: simple

Disadvantage:

- if one portion perishes the key cannot be restored,
- all of the secret-keepers are needed for the recovery of the key
- a subset of secret keepers with size close to N possesses too much information (the rest might be found by them using exhaustive search)

2.version:

$S = \{0, 1, \dots, q-1\}$ is the set of from which the secret is selected randomly

$s \in S$ is the secret

r_1, r_2, \dots, r_{N-1} random elements from set S

$r_1, r_2, \dots, r_{N-1}, r_N$ distributed shares of secret among N secret keepers, where

$r_N = s - (r_1 + r_2 + \dots + r_{N-1}) \text{ mod } q$

Advantage: Less than N people cannot restore the key!

Disadvantage:

- all of the secret-keepers are needed for the recovery of the key
- the shares have the same size as the secret (feasibility problem)

Data Security: Protocols

Secret sharing

Goal:

sharing a secret among N people such way, that an arbitrary, at least K size subset of them can restore the key, however smaller number of people cannot. ($K < N$).

Advantage: the secret can be restored even if N-K shares of it perishes.

3.version:

The randomization trick of Version 2. Is combined with Reed-Solomon coding (known well from coding theory)

- (N,K) RS-code over $GF(q)$, $q = |S|$

- a property RS-codes: if we know at least K elements of the codeword the message can be found uniquely ($\leq N-K$ erasures can be corrected)

Data Security: Protocols

Secret sharing

r_1, r_2, \dots, r_{K-1} randomly selected elements from $GF(q)$ -ból

$r_0 = s$

$r = (r_0, r_1, \dots, r_{K-1})$ consider as a message to be encoded by the RS code

$$c = rG,$$

G , generatormatrix over $GF(q)$ of dimension $K \times N$

$c = (c_0, c_1, \dots, c_{N-1})$ an RS codeword, the elements of which are distributed among N secret keepers

$$G = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \alpha & \alpha^2 & \dots & \alpha^{N-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha^{K-1} & \alpha^{2(K-1)} & \dots & \alpha^{(N-1)(K-1)} \end{pmatrix}$$

where $\alpha \in GF(q)$ is an element with order N (N -th root of unity).

Data Security: Protocols

Secret sharing

$$c_i = D(\alpha^i), \quad i=0, 1, \dots, N-1$$

where

$D(x) = r_0 + r_1x + \dots + r_{K-1}x^{K-1}$ polynomial over $GF(q)$

An interesting interpretation (Shamir):

There exists exactly one polynomial $y = D(x)$ with degree $K-1$, connecting points $(y_1, x_1), (y_2, x_2), \dots, (y_K, x_K)$, i.e.

$$y_i = D(x_i), \quad 1 \leq i \leq K.$$

If we select N points lying on one curve, then the curve (polynomial) can be reconstructed from a subset of at least K points and constant term of the polynomial corresponds to the secret.

Data Security: Protocols

Special protocols

Bit Commitment Protocols

Alice wants to commit to a bit or series of bits, but does not want to reveal her until sometime later. Bob, wants to make sure that Alice cannot change her mind after commitment.

Protocol 1.

1. $B \rightarrow A: R$
2. $A \rightarrow B: E_k(R, b)$
3. $A \rightarrow B: K$
4. B: checks $R \rightarrow b$

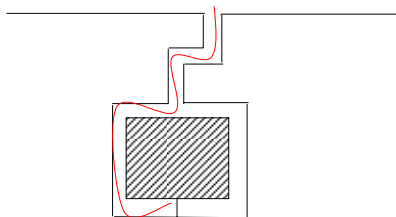
Protocol 2.

1. $A \rightarrow B: H(R1, R2, b), R1$
2. $A \rightarrow B: R1, R2, b$

Data Security: Protocols

Special protocols

Zero knowledge protocols (ZKP)



Data Security: Protocols

Special protocols

ZKP based on graph isomorphism problem

- (1) Alice randomly permutes $G1$ to produce another graph, H , that is isomorphic to $G1$.
- (2) Alice sends H to Bob.
- (3) Bob asks Alice either to:
 - (a) prove that H and $G1$ are isomorphic,
 - or
 - (b) prove that H and $G2$ are isomorphic.
- (4) Alice complies. She either:
 - (a) proves that H and $G1$ are isomorphic, without proving that H and $G2$ are isomorphic, or
 - (b) proves that H and $G2$ are isomorphic, without proving that H and $G1$ are isomorphic.
- (5) Alice and Bob repeat steps (1) through (4) n times.

Data Security: Protocols

Blind signatures

Bob has a public key, e , a private key, d , and a public modulus, n . Alice wants Bob to sign message s blindly.

- (1) Alice chooses a random value, r , between 1 and n . Then she blinds s by computing

$$m = sr^e \bmod n$$

- (2) Bob signs m

$$m^d = (sr^e)^d \bmod n$$

- (3) Alice unblinds by computing

$$s^d = m^d / r \bmod n$$

E.g. DigiCash



Data Security: **Protocols**

