Szenzorhálózatok biztonsága

Dr. Fehér Gábor

Mit értsünk biztonság alatt?

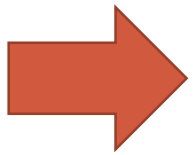
- Mit védjünk a szenzorhálózatban?
  - Értékes adatok? Adathalászat?
  - Behatolások a hálózaton keresztül?
  - Botnet hálózatok? Botcoin?

  - Szenzorok adatai
    - Otthon vagyunk? (Okos otthon)
    - Ingyen parkolás (Okos város)
    - Forgalmi dugó kialakítása (Okos város)
    - Katonai műveletek (Katonai alkalmazás)
  - Vezérlő irányítása
    - Ajtó nyitás, ablak nyitás, riasztó kapcsolás (Okos otthon)
    - Közlekedés irányítása (Okos város)
    - Konkurencia letörése (Okos földművelés)

  - Denial of Service !!!

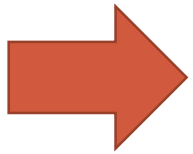
Felépítésből adódó tulajdonságok

- Szenzorhálózatok, szenzorok
  - Feladat orientált felépítés – olcsó
  - Sok kell belőle – még olcsóbb



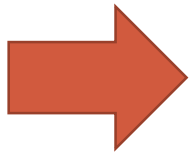
Korlátozott erőforrás

  - Szűkös tápellátás



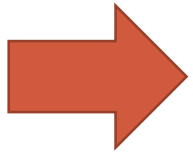
Erősen korlátozott erőforrás
memória (kód), CPU

  - Véletlen kihelyezés (bizonyos alkalmazások)
  - Szabadon hozzáférhető helyek



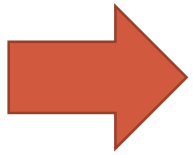
Fizikai védelem hiánya,
Dinamikusan változó környezet

  - Kis hatótávolságú rádió
  - Alvó csomópontok



Korlátozott kommunikáció (kis méret)
Véletlen és szándékosság
nehezen megkülönböztethető

  - Gyakran hiányzik a globális idő mérése



Idő alapú protokollok
nem használhatóak

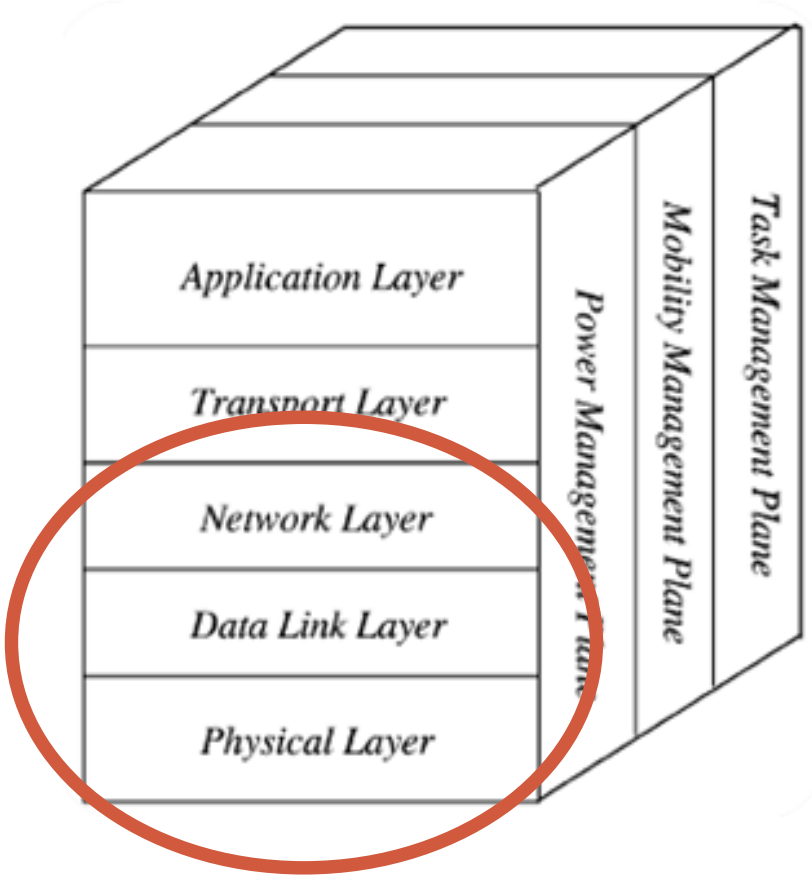
Biztonság megközelítése (klasszikus)

- CIA hármas
  - Confidentiality – Adatok bizalmassága ↔ lehallgatás
  - Integrity – Adatok sérthetetlensége ↔ adat felülírása
  - Availability – Adatok elérhetősége ↔ szolgálatmegtagadás

  - Authenticity – Hiteles, ellenőrzött felek
  - Accountability – Felelősségre vonható felek
  - Non-repudiation – Nem megszemélyesíthető felek

Biztonság rétegről rétegre

- Biztonság vizsgálata rétegenként
  - Fizikai réteg
  - Adatkapcsolati réteg
  - Hálózati réteg
  - Szállítási réteg
  - Alkalmazás réteg
- A szenzorhálózat vizsgálata
- Nem vizsgáljuk:
  - Mérések megváltoztatása
  - Vezérlők befolyásolása



Fizikai réteg támadásai és védelme

Támadások

- DoS (Denial of Service) támadások
  - Rádiófrekvencia zavarása (jamming)
- Fizikai hozzáférés
  - Szenzor ellopása, adatok kinyerése
  - Szenzor átprogramozása

Védelem

- A kommunikáció legtöbb esetben az ISM sávban zajlik. A rádiók nem átállíthatóak
  - Szoftver rádió
  - Szórt spektrumú kommunikáció

- Szenzor őrzése sok esetben nem biztosítható
  - „Lelakatolt” szenzorok
  - Secure Element (SE)
    - Tamper resistant HW platform

Adatkapcsolati réteg támadásai és védelme

Támadások

- Lehallgatás
- Adatok megváltoztatása
- Csomagok törlése
- Visszajátszás

- Ütközések (jamming), elárasztás
  - Szenzor lemerítése

Védelem

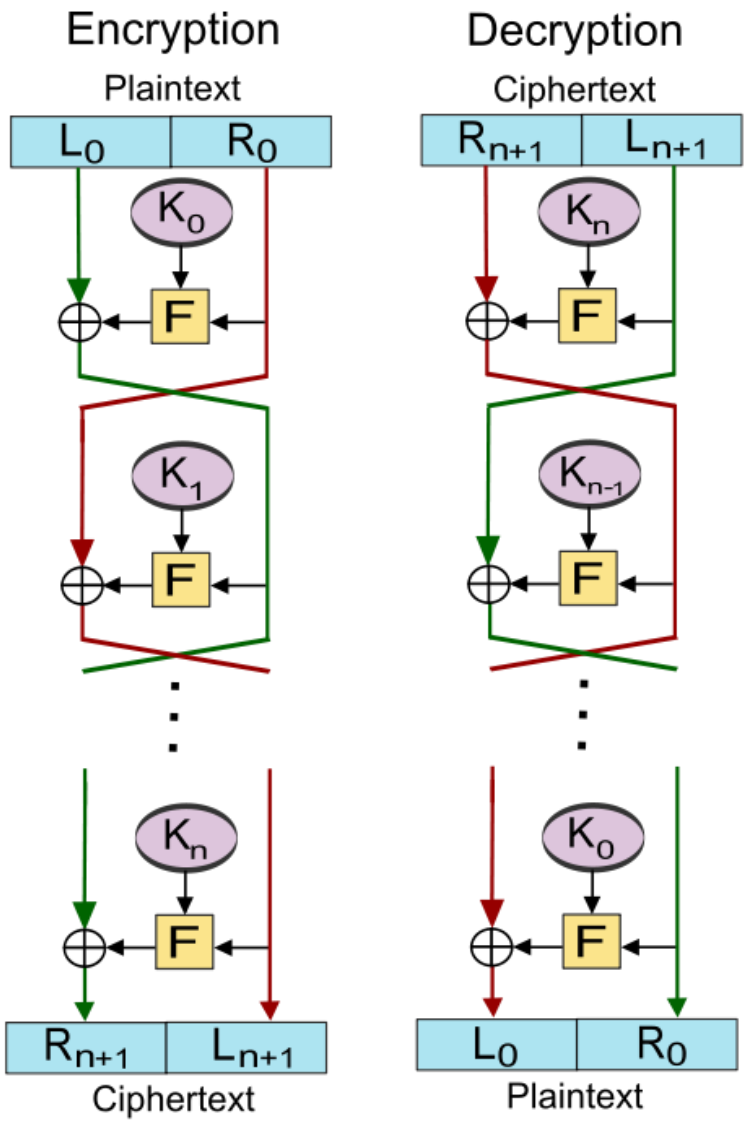
- Titkosítás
- Integritás védelem
- Visszajátszás elleni védelem

Blokk titkosítás

- Blokk titkosítás
  - Adatok tikosítása blokkonként, blokk mérete tipikusan : 64 – 128 bit
  - Népszerű titkosítók SKIPJACK (80 bit), RC6 (128, 192, 256bit), XXTEA (128 bit), AES (128, 192, 256bit)
  - Sebesség, kód mérete, energiafogyasztás lényeges
    - Többféle változat is készülhet: sebesség optimalizált, kód méret optimalizált
  - Kitöltés
  - Blokkok összefűzése, továbbítása

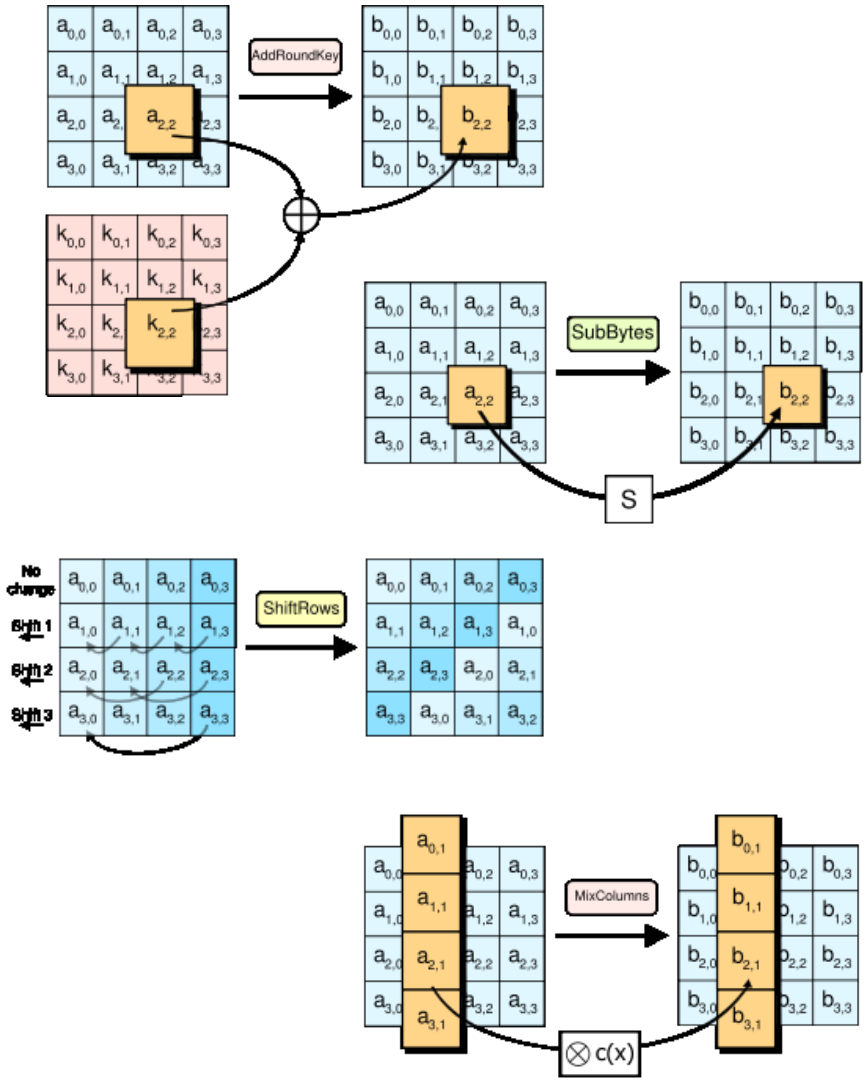
Blokk titkosítás – Titkosítók felépítése 1.

- Feistel architektúra
  - A titkosítás iterációkban/körökben történik (round)
  - Az egyes körök megegyeznek, de a használt kulcs eltér
    - A körönkénti alkulcsok az kulcsból származnak
  - A titkosítás feloldása esetén a felépítés megegyezik, de a kulcsokat fordított sorrendben használjuk
    - F titkosító függvény nem kell, hogy invertálható legyen!
  - A nem kiegyensúlyozott Feistel architektúra esetén a bal és jobb oldali adatok mérete eltérhet
  - F titkosító függvény általában egyszerű felépítésű
    - Permutáció, helyettesítés, kibővítés, vágás
- Példa:
  - DES algoritmus (Feistel) – Data Encryption Standard, Lucifer
  - 56 bites kulcs, 48 bites alkulcsok, 16 kör, S box és P box felépítés



Blokk titkosítás – Titkosítók felépítése 2.

- AES titkosító – Advanced Encryption Standard, Rijndael
- Jelenleg elfogadott standard titkosító (úgy tűnik a legjobb titkosító hosszú távon is)
- Substitution-permutation network architektúra
- Nem Feistel, de egyszerű műveletekből épül fel
  - 4x4 „matrix” műveletek
    - AddRoundKey
    - SubBytes
    - ShiftRows
    - mixColumns
  - Több körös titkosítás. A körök száma függ a kulcs méretétől: 10 – 12 – 14 kör
  - Alkulcsok használata az egyes körök között



Blokk titkosítás – Kitöltés

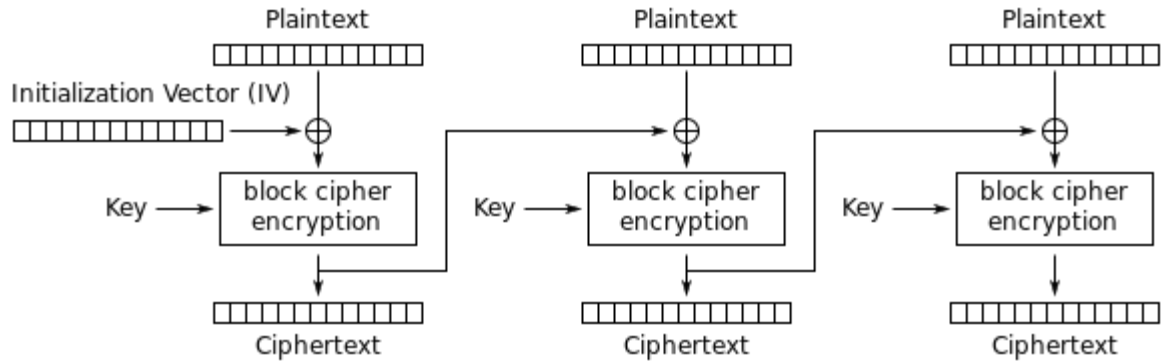
- A blokkok fix hosszúságúak a titkosítás esetében
  - Az adat mérete gyakran eltér a blokk méretétől
  - Adat kevesebb, mint a titkosítási blokk: kitöltés szükséges (padding)
  - Adat több, mint a titkosítási blokk: több blokkra kell darabolni (blokk chaining - modes of operation)

- Kitöltés
  - A lényeg, hogy egyértelmű legyen a dekódoló számára!
    - Csupa 0 (nem mindig jó)
    - Utolsó bájt jelzi a blokk méretét
      - A többi 0
      - A többi véletlen
      - A többi ugyanaz
  - A kitöltés még tikosítás előtt kerül a blokkba
  - Gyakran a kitöltést is át kell vinni a hálózaton (128 bit esetén 16 bájtos egységek!)

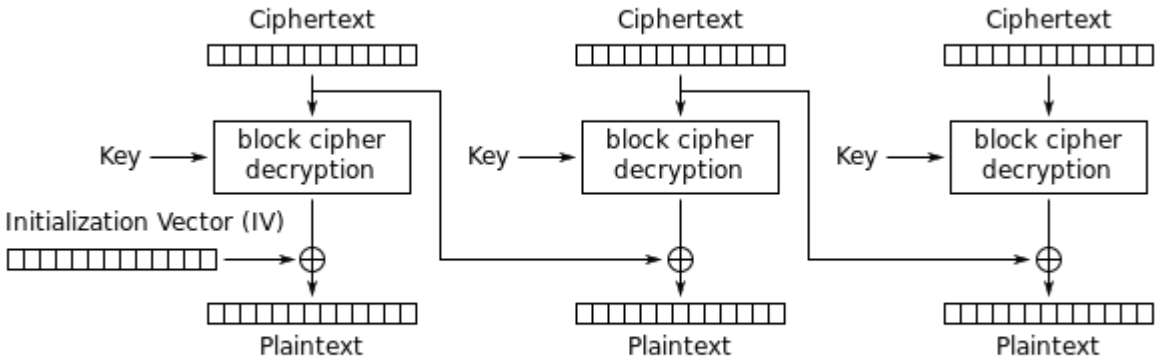
Blokk titkosítás – Blokkok összefűzése 1.

- Electronic Codebook
  - Minden blokkot ugyanúgy titkosítunk, egymástól függetlenül
  - Nem biztonságos! A blokkokat fel lehet cserélni, illetve azonos blokkok felismerhetőek!

- CBC – Cipher-block Chaining
  - A blokkok egy vektor segítségével egymáshoz vannak fűzve
  - Az inicializálási vektor nem titkos, de szükséges a szinkronizálása
  - A titkosítást muszáj sorban végezni, a feloldás végezhető párhuzamosan is
  - Átvitelnél létezik olyan változat, ahol a kitöltést nem kell átvinni (Ciphertext stealing)
  - Használják integritás védelemre is (CBC-MAC)



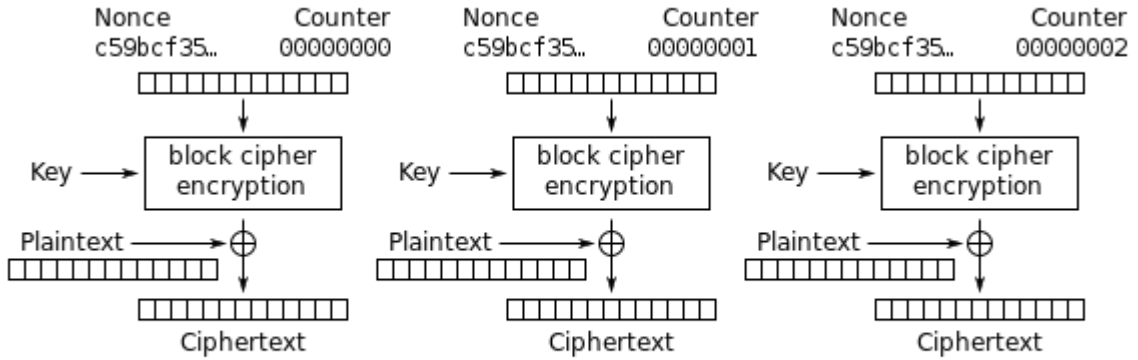
Cipher Block Chaining (CBC) mode encryption



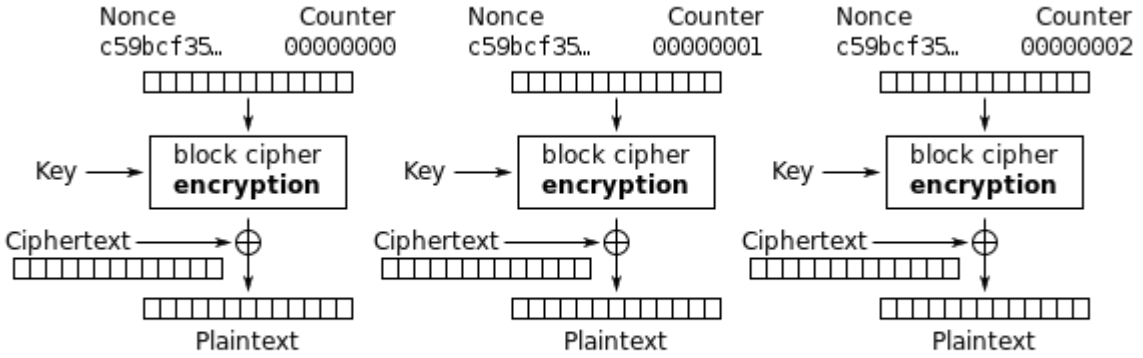
Cipher Block Chaining (CBC) mode decryption

Blokk titkosítás – Blokkok összefűzése 2.

- ICM/CTR – Counter mode
  - A titkosító blokkok egy kulcsot állítanak elő
    - Minden blokk külön kulcs
    - A kulcsok a mester kulcsból és egy számlálóból állnak össze
    - Mindkét oldalon ugyanúgy történik a kulcsok előállítása, így csak titkosító művelet van
  - Működését tekintve folyamtitkosító
    - Tetszőleges hosszúságú bementen működik
    - Nincs szükség kitöltésre
  - Titkosítás és a feloldás is párhuzamosítható
- CCM: Counter with CBC-MAC
  - Titkosítás és hitelesítés



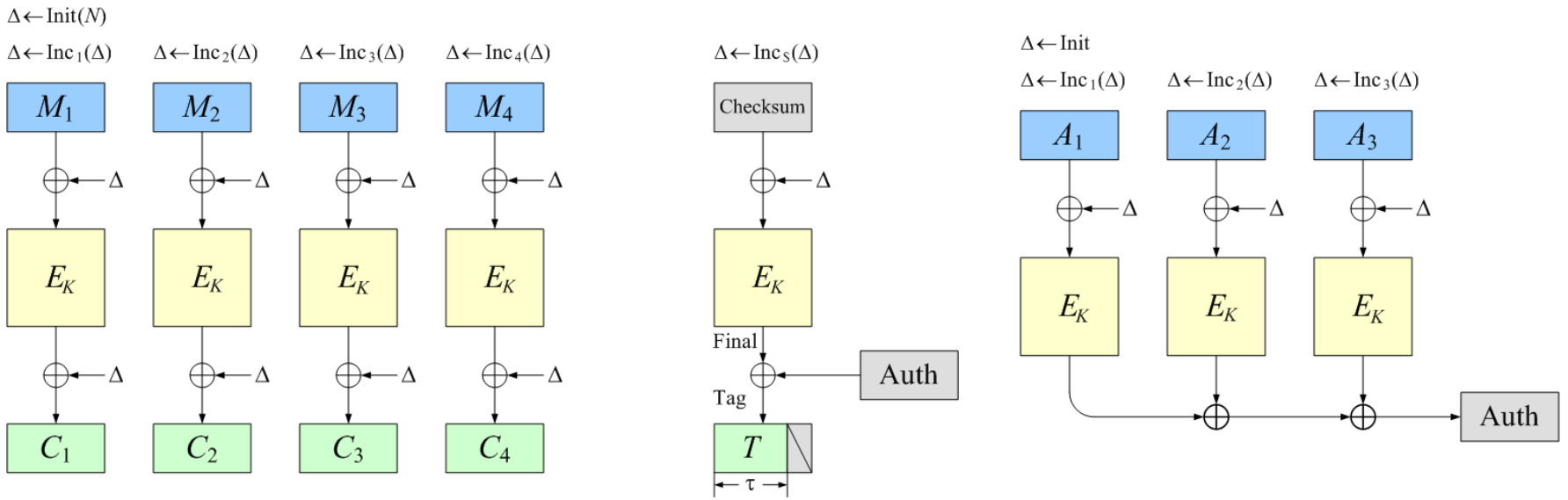
Counter (CTR) mode encryption



Counter (CTR) mode decryption

Blokk titkosítás – Blokkok összefűzése 3.

- OCB – Offset Codebook Mode (~~OCB1~~, OCB2, OCB3)
  - Titkosítás és integritás védelem egyben
  - Lehetséges nem titkosított (de integritás védett) rész csatolása
  - Az integritás védelem mérete különböző lehet
  - Kitöltés továbbítása nem mindig szükséges, hasonlóan a CBC-hez
  - Szükséges blokk titkosítás műveletek száma: m + a + 1.016 (blokk méretek, számlálót feltételezve N -nek)



Blokk titkosítás – Mérési eredmények

- Mica2 mote, 4KB RAM

Code size	SKJ	XXTEA	RC6	AES(speed)	AES(size)
RAM	800B	600B	1400B	1400B	1400B
ROM	16KB	16KB	16KB	26KB	16KB

Throughput	SKJ	XXTEA	RC6	AES(speed)	AES(size)
Encoding	170Kbps	40Kbps	62Kbps	120Kbps	58Kbps
Decoding	170Kbps	40Kbps	56Kbps	130Kbps	55Kbps

Energy	SKJ	XXTEA	RC6	AES(speed)	AES(size)
CPU	0.665uJ	0.68uJ	0.69uJ	0.672uJ	0.692uJ
Radio	1.754uJ	1.754uJ	1.759uJ	1.759uJ	1.759uJ

Blokk titkosítás – Konklúzió

- Létezik titkosító WSN környezetben is
  - SKIPJACK:
    - Minimális kódméret, gyors futás, alacsony fogyasztás
    - Alacsony szintű biztonság! - 80 bites kulcs
  - XXTEA:
    - Minimális kódméret, alacsony fogyasztás, magas szintű biztonság
    - Alacsony futási sebesség! (Sok iteráció (32) az algoritmusban)
  - OCB használata előnyös, minden más móddal szemben
  - Szinkronizáció szükséges az adó és vevő között
    - Initialization Vector, Nonce

Folyam titkosítás

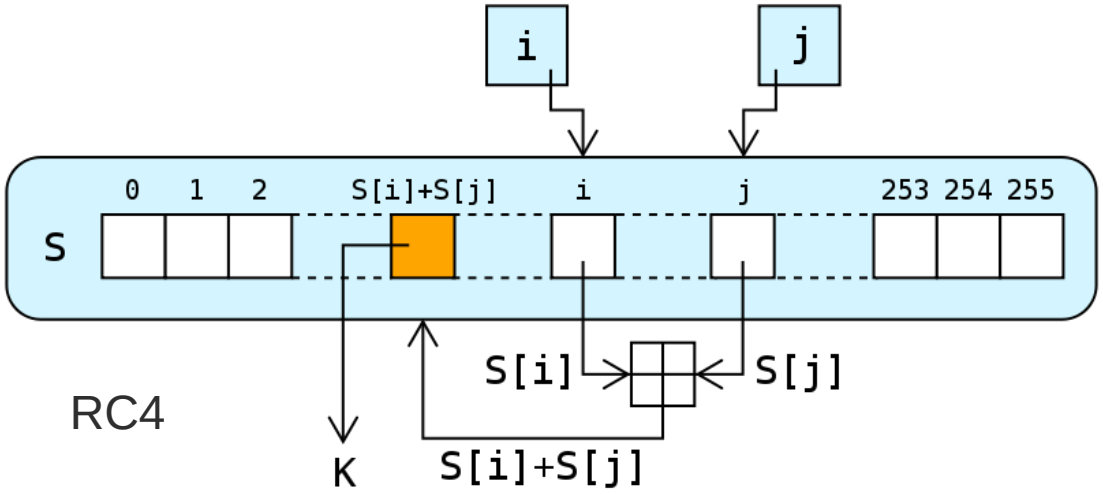
- Folyam titkosítók
  - Nincsenek meghatározott blokkok, nincs kitöltés
  - Leggyakoribb esetben Vernam titkosító: $c_i = p_i$ XOR k_i
    - One Time Pad (OTP)
    - Tökéletes titkosítás, amennyiben a kulcs valóban véletlen (Shannon bizonyítása)
    - Nem a titkosítás a nehéz, hanem a hosszú „véletlen” kulcs előállítása
    - SOHA nem szabad ugyanazt a kulcsot használni két különböző adathoz!

$$c_1 = p_1 \text{ XOR } \mathbf{k_1}$$
$$c_2 = p_2 \text{ XOR } \mathbf{k_1}$$
$$\downarrow$$
$$c_1 \text{ XOR } c_2 = p_1 \text{ XOR } p_2$$

  - RC4 titkosító (SW)
  - LFSR titkosítók (HW)

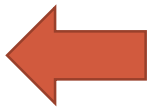
  - Sebességük sokszor a blokk titkosítók többszöröse

- Keressük a legjobb folyam titkosítót!
  - eSTREAM, ECRYPT



Integritás védelem

- Védelem az adatok megváltoztatása ellen
  - Kriptográfiai ellenőrző összeg, amely adat változása esetén eltérést jelez
    MAC - Message authentication code
  - Kulcsos egyirányú (hash) függvények használata
    - Az egyirányú függvény biztosítja, hogy ne lehessen következtetni az adatra
    - A kulcs biztosítja, hogy a számolás egyedi, forráshoz kötött legyen
    - Szükséges, hogy az adat változása ne legyen kiszámítható (titkosított CRC nem jó!)
  - A MAC hossza meghatározza a védelem erősségét is
    - Kriptográfiai egyirányú függvények és a kód hossza:
      - ~~MD5 (128bit)~~, SHA1 (160bit) SHA2-3 (224bit-512bit)
    - A „születésnapi paradoxon” miatt, valójában nem is ennyire erős
    - A hash függvény kimenete utólagosan vágható: csökken a méret, de egyben a biztonság is
  - Hagyományos megoldások
    - HMAC-SHA1
    - CBC-MAC



„Lassú” futás és nagy kimenet

WSN megoldás:
OCB

Titkosítás, integritás védelem – Kulcsok

- A kulcskiosztás meghatározó a biztonság esetén

- Hálózati kulcs használata
  - Minden eszköz ugyanazt a kulcsot használja
  - **Nem biztonságos**, mert a kulcs védelme nem biztosítható (pl. eszköz lopás)

költség

- Egyedi kulcsok használata
  - Publikus-privát kulcspár
  - Szimmetrikus kulcs generálása és hitelesítése
  - Aláírás és ellenőrzés: RSA (304mJ, 11.9mJ) és ECC (22.82mJ, 45.09mJ)
  - Kulcsok mérete jelentős (ECC kisebb)

költség

- Egyedi kulcsok használata
  - „Predistribution”, a szimmetrikus kulcsok előzetes kiosztása
  - Skálázhatósági problémák. Mindenki-mindenkivel?

költség

Aktívan keressük a legjobb megoldást!

Titkosítás, integritás védelem – Egyedi csomagkulcsok

- Egyedi csomagkulcsok gyártása
  - Az átvitel során két csomag nem titkosítható ugyanazzal a kulccsal
    - Egyedi csomag azonosító + mester kulcs a csomag titkosításához
    - Az egyedi azonosító sokszor sorszám (de van, hogy teljesen véletlen érték)
    - Minél hosszabb az azonosító, annál több csomag titkosítható
      - Ha kimerül, akkor új kulcsot kell készíteni
      - 48 bit jónak tűnik (Pl.: WiFi)
  - A csomag azonosítót szinkronizálni kell a kommunikáló felek között
    - Minden egyes csomagban átvitelre kerül (Pl. ZegBee)
      - Túlságosan nagy a mérete
    - Egyszer szinkronizáljuk és utána mindenki maga számolja (Pl. SPINS)
      - Elveszett csomagok esetén elvész a szinkron
      - Skálázhatósági problémák sok kommunikáció esetén
    - Vegyes módszer: az azonosító csak egy részét visszük át (Pl. MiniSec)
      - Kis méretű azonosító (akár 4-8 bit)
      - Elveszett csomagok esetén robusztusság
      - Skálázhatósági problémák csökkennek, de maradnak

Visszajátszás elleni védelem

- Több esetben lényeges, hogy az egyszer küldött csomagot ne lehessen újraküldeni
  - Valójában 2x ne lehessen fogadni. Az újraküldés természetes csomagvesztés esetén
  - Igazán fontos szerepe a vezérlő üzenetek esetén van

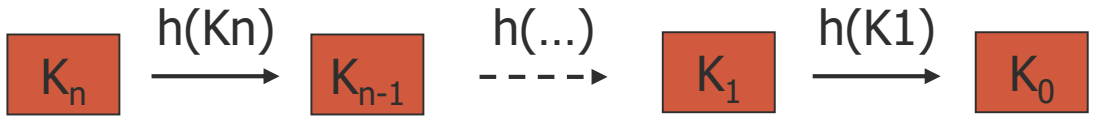
- Visszajátszás elleni védelmi stratégiák:
  - Szigorúan monoton növekedő csomagazonosítók (sorszámok) (Pl.: SPINS és ZigBee)
    - Csak teljes átvitel esetén használható
    - Skálázhatósági problémák, jegyezni kell, hogy melyik node melyik azonosítónál jár
  - Időablakos megoldás
    - A kulcsok csak egy adott időablakban használhatóak (a kulcs kiegészítése idővel)
    - Visszajátszás csak az adott időablakban lehetséges. Rövid időablakok
    - Szinkronizációs problémák
  - Bloom Filter alapú megoldás
    - A csomag azonosító Bloom filterbe kerül (hosszabb időablakkal kiegészítve)
    - A Bloom filter segít eldönteni, hogy ismétlés vagy sem (lehetséges hibák kezelése!)
    - Nem szükséges az azonosítók tárolása
- A visszajátszás elleni védelem magasabb rétegben is megvalósulhat!

Adatkapcsolati réteg – Megvalósítások

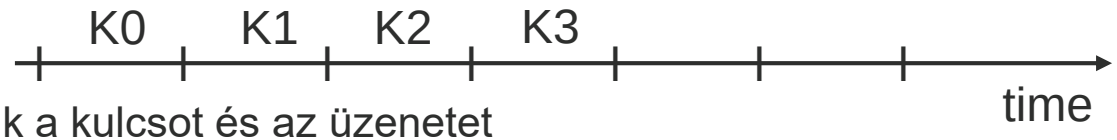
- Konkrét megvalósítások
  - ZigBee
    - Magas biztonság, magas energiafelhasználás
    - Teljes csomagazonosító minden egyes csomagban
    - Visszajátszás elleni védelem, minden nodehoz eltárolva a legutoljára küldött csomagazonosító
  - SPINS (Security Protocols for Sensor Networks)
    - Magas biztonság, közepesen hatékony energia kezelés
    - Csomagazonosító szinkronizálás
    - Visszajátszás elleni védelem, minden nodehoz eltárolva a legutoljára küldött csomagazonosító
  - MiniSec
    - Magas biztonság, alacsony energiafelhasználás
    - Csomagazonosító szinkronizálás részleges csomagazonosító alapján
    - Visszajátszás elleni védelem valószínűség alapon, pontatlan időszinkronizáció is megengedett
  - TinySec
    - Alacsony biztonság, alacsony energiafelhasználás
    - Nincs visszajátszás elleni védelem
    - Nincsenek tárolt állapotok

Adatkapcsolati réteg – Hitelesített Broadcast

- TESLA / uTESLA – (Micro) Timed Efficient Stream Loss-tolerant Authentication
  - Broadcast üzenetek hitelesítéséhez
  - uTESLA
    - Korlátozott forrás, főleg a gyűjtő -> node irányban használható, de megoldható vele kulcscsere is
    - Nincs digitális aláírás (TESLA esetében szükséges)
  - Egyirányú függvény által generált kulcs láncolat
    - A kulcsokat ellenőrizni lehet a lánc egy későbbi példányával, de előre megtudni a későbbi kulcsot nem lehet
    - A kulcsok gyártása a felhasználással ellentétes irányban történik. A kezdő kulcsot terítik a hálózatban



  - Az időt intervallumokra osztjuk, egy adott intervallumban egy kulcs van érvényben
    - Az aktuális kulcsot nem ismerik a nodeok, az csak a következő intervallumban derül ki számukra
    - Az adott kulcs segítségével hitelesítik az üzeneteket
    - A kulcsot egy következő intervallumban felfedik
    - A nodeok az egyirányú függvény segítségével ellenőrzik a kulcsot és az üzenetet
    - Üzenet vesztés esetén sincs gond, a felfedett kulcs ellenőrzi az összes korábbi kulcsot



Hálózati réteg támadásai és védelme

Támadások

- Hamis routing információk
- Black hole
  - Üzenetek elnyelése
- Sink hole
  - A forgalom vonzása más nodeok felől

Védelem

- Routing információ hitelesítés
- Többutas adatküldés
- Szerepkörök rögzítése?
- Routing megfigyelése
- Lokáció alapú routing?

Új megoldási javaslatok

Routing védelme – ARAN 1.

- Authenticated Routing for Ad hoc Networks (AODV alapon - on demand)
- Hitelesítés digitális aláírással
  - Hitelesítő-központ, aki mindenkit hitelesít certA={IPA,KA+,t,e}KT-

IP_A:	A node IP címe
K_{A+}:	A publikus kulcsa
t:	időbélyeg
e:	lejárat
K_{T-}:	hitelesítő kulcsa
$\{\}K_{T-:}$	digitális aláírás

- Út felderítése – Route Discovery Protocol (RDP)
  - broadcast üzenet
    - A ⇨ {RDP,IPx,NA,t}KA-,certA
  - Feldolgozása
    - B ⇨ {{RDP,IPx,NA,t}KA-,certA}KB-,certB
    - C ⇨ {{RDP,IPx,NA,t}KA-,certA}KC-,certC
    - ...
    - Ha már volt ilyen sorszám/idő, akkor eldobja

- Út kialakítása – Route Reply
  - A legelső befutó RDP! (Talán nem a legrövidebb)
  - Unicast üzenet visszafelé, akitől jött az üzenet
    - X ⇨ {REP,IPA,NA,t}KX-,certX
  - Feldolgozása
    - C ⇨ {{REP,IPA,NA,t}KX-,certX}KC-,certC
    - ...

Routing védelme – ARAN 2.

- Karbantartás – Route Maintenance
  - Hibaüzenet az út szakadásánál
    - B ⇨ {ERR,IP_A, IP_X,N_B,t}K_{B-},$cert_B$
  - Nehezen detektálható, hogy támadás vagy természetes
  - Gyanús a sok hibát jelző csomópont
  - „Hibás” csomópontok elkerülése
- Kulcs visszavonása
  - broadcast
  - T ⇨ {revoke,cert $_Y$}K_{T-}

ARAN – Biztonság és alkalmazhatóság

- Csak hitelesített csomópontok vesznek részt a kommunikációban
- Nincs megszemélyesítés a digitális aláírások miatt
- Nincs visszajátszás (sorszám, idő)
- Hamis üzenetek hitelesített feladótól
  - Kitiltás jár érte
- Alagutak? A célpont az első RDP-re válaszol!

- A digitális aláírás miatt nagyon költséges
- Az üzenetek hossza jelentősen megnő a tanúsítványok csatolásával

Routing védelme – ARIADNE 1.

- DSR alapú – Dynamic Source Routing
- Routing üzenetek hitelesítése
  - Közös titok a kommunikáló párok között + broadcast hitelesítés
  - TESLA használata üzenetek hitelesítéséhez

- Route Request
  - Minden egyes közbenső csomópont hitelesíti
    - Saját címének hozzáadása + az eddigi címek hashelése: h=H[A,h]
    - MAC hozzáadása az adott intervallum kulcsával
- Route Reply
  - A Request üzenet ellenőrzése: Nem járt még le?
  - Hitelesítés a forrás és cél közös kulcsával
  - Minden egyes közbenső csomópont
    - Megvárja amíg felfedheti a saját kulcsát
    - A kulcsot hozzáírja az üzenethez
- A forrás végül ellenőrzi a kulcsokat

Ariadne Biztonság

- A Route Request -ből észrevétlen nem törölhető/módosítható bejegyzés
- A támadó hosszabbíthatja az utat, de akkor már nem biztos, hogy ő a legrövidebb
- Csak hitelesített csomópontok vehetnek részt
- Nincs nagy erőforrásigény: TESLA protokoll, hash függvény

- A TESLA (nem uTESLA!) még így is költséges, skálázhatósági problémák (állapotok fenntartása)
- Üzenetek hossza jelentősen nőhet