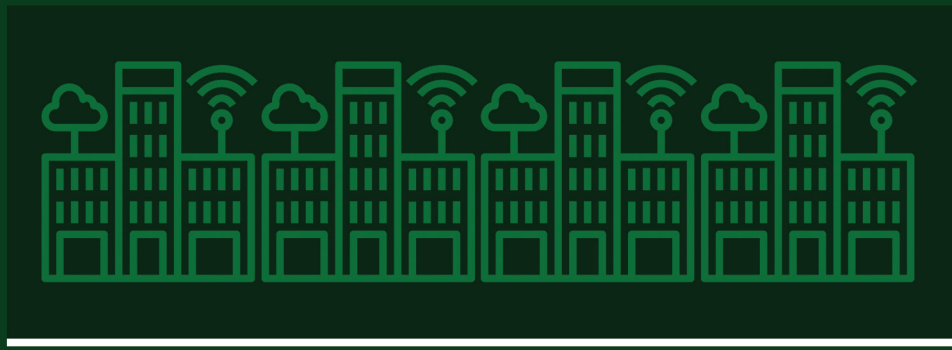
OKOSVÁROS-TECHNOLÓGIÁK XIV.

Szenzorhálózatok biztonsági kihívásai az okos városokban



Fehér Gábor

Nemzeti Közszolgálati Egyetem Budapest

SZÉCHENYI 2020

Magyarország Kormánya

Európai Unió
Európai Szociális Alap

BEFEKTETÉS A JÖVŐBE

OKOSVÁROS-TECHNOLÓGIÁK
A technológia fejlődésének irányai és hatása
XIV. kötet

Sorozatszerkesztő:
Sallai Gyula

Fehér Gábor

SZENZORHÁLÓZATOK BIZTONSÁGI KIHÍVÁSAI AZ OKOS VÁROSOKBAN

Nemzeti Közszolgálati Egyetem
Közigazgatási Továbbképzési Intézet
Budapest, 2020

A kötet a Nemzeti Közszolgálati Egyetem **KÖFOP-2.1.2-VEKOP-15-2016-00001 „A jó kormányzást megalapozó közszolgálat-fejlesztés”** projektje keretében, a Budapesti Műszaki és Gazdaságtudományi Egyetem Villamosmérnöki és Informatikai Karán létesült **„Okos város – okos közigazgatás”** kutatóműhelyben (2017/162/BME-VIK) készült.

Szakmai lektor:
Vida Rolland egyetemi docens, BME-VIK

A kézirat lezárásának dátuma:
2018. március 21.

TARTALOM

1. BEVEZETŐ

Az okos városokban fellelhető szenzorhálózatok a biztonság szemszögéből sok hasonlóságot mutat más, nem speciális számítógép hálózatok biztonságával. Ugyanakkor vannak jellegzetes eltérések is, amelyek leginkább két tényezőre vezethetőek vissza. Egyrészt a szenzorhálózatot alkotó szenzorok korlátozottak erőforrásokban. Ez korlátozott feldolgozó teljesítményt, kódméretet és tápellátást jelent leginkább. A másik problémát az alkalmazások jellegéből adódó kitettség jelenti. A szenzorok ugyanis sokszor olyan helyeken vannak elhelyezve, amelyeket nem őriznek és nincs is elzárva biztonságosan a külvilágtól, így aki gonosz szándékkal manipulálni szeretné azokat, sokszor könnyen hozzáférést nyer az eszközhöz. Ebből adódóan tehát a szenzorokat ebben a környezetben kell megvédenünk, ahol sok veszélyre és korlátozott védelemre számíthatunk.

A számítógépes biztonság területén a tökéletes védelem szinte nem is létezik. Ezt persze úgy kell értenünk, hogy bár az összes biztonsági problémára létezhet védelmi megoldás, mégis ezek alkalmazása annyira költséges és akár más tekintetben is unpraktikus lehet, hogy nem célravezető az alkalmazásuk. A védelem szempontjából ezért nagyon fontos meghatározni, hogy mennyit ér a biztonság, mennyit kockáztatunk a támadhatósággal. Ehhez érdemes megismerni a támadó motivációt is, aki valószínűleg ugyanezeket a számításokat végzi el, kiszámítja, hogy mennyi befektetéssel és mekkora kockázattal mekkora nyereséghez juthat. Amennyiben a befektetés és kockázat olyan magas, hogy nem éri el azt a szintet, amennyi bevételt remél a támadó a támadás hatására, úgy valószínűsíthető, hogy nem fogja a támadást végrehajtani.

Egy példát tekintve, egy okos otthonban lévő vezeték nélküli villanykapcsolót nem célszerű magas költségű védelmi rendszerrel ellátni, hiszen valószínűtlen, hogy a támadó a támadás magas költségei ellenére azzal szeretné bosszantani a lakót, hogy helyette kapcsolja a lámpákat. A számítógépes biztonságban a mai támadók és támadások jelentős része gazdasági alapon működik, így a megfelelő védelem nem feltétlenül jelenti a tökéletes védelmet, csupán egy olyan védelmet, amely gazdaságilag megalapozatlanná teszi a támadást. Ugyanakkor tudnunk kell, hogy létezik egy másik, egyre jelentősebbé váló támadó csoport, ahol már nem gazdasági megfontolások dominálnak, hanem hatalmi, vallási érdekek. Ebben a környezetben sokszor a pénz nem számít, így az előbbi gondolatmenet sem teljesül, a támadás akkor is megvalósul, ha annak a költsége jelentősen nagyobb a támadásból remélt haszonnál. A korábbi példához visszatérve, amennyiben a szomszédunk utál minket és számára mindent pénzt megér, ha a villanykapcsoló vezérlésével bosszanthat minket, akkor bizony fel kell készülnünk a támadásokra. Megemlítenénk még egy speciálisabb esetet, amikor a támadás hatására emberélet van veszélyben. Bár van olyan számítás, ahol megjelenik az emberi élet értéke pénzben is kifejezve, mégis sokszor leginkább egy felbecsülhetetlen magas számnak tartjuk. Így, ha a támadás emberi életet veszélyeztet, ott bizony muszáj mindent megtenni a védelem érdekében.

A kismonográfia második fejezetében felsoroljuk a tipikus okosváros alkalmazásokat, ahol szenzorhálózatot használnak az alkalmazás működéséhez. A felsorolásnál elemezzük, hogy melyik alkalmazás típusnál milyen támadókra és milyen támadásokra számíthatunk, illetve megnézzük azt is, hogy pontosabban milyen szenzorhálózati technológia áll az adott alkalmazások hátterében.

A harmadik fejezetben a kriptográfiával foglalkozunk. Megmutatjuk azt, hogy milyen olyan algoritmusok vannak, amelyeket a szenzorhálózatokban feltételezett gyenge erőforrásokkal rendelkező mikrokontrollerek is képesek jó teljesítménnyel futtatni. A gyenge erőforrások miatt ugyanis arra kell számítanunk, hogy a jól ismert és a gyakorlatban elterjedt kriptográfia algoritmusok egy része sajnos ebben a környezetben nem alkalmazható. Megmutatjuk, hogy milyen alternatívák állnak rendelkezésre és milyen kompromisszumot kell hoznunk ezek alkalmazásával.

A negyedik fejezet a szenzorhálózati kommunikáció támadásáról szól. Sorra vesszük a hálózati rétegeket és megvizsgáljuk a szenzorhálózatokat érintő lehetséges támadásokat. A támadások mellett azt is megmutatjuk, milyen technológiák állnak rendelkezésre a védekezéshez. Az ötödik fejezet aztán már teljes egészében elemzi azoknak a szenzorhálózati technológiáknak a védelmét, amelyeket a gyakorlatban is alkalmazunk. Megmutatjuk, hogyan van implementálva a védelem a különféle szenzorhálózat típusokban. A hatodik fejezet végül összegzi a leírtakat.

A szerző[1]

[1] *Dr. Fehér Gábor*, a BME Villamosmérnöki és Informatikai Kara Távközlési és Médiainformatikai Tanszékének egyetemi docense, tanszékvezető-helyettes. PhD-fokozatát a BME-n 2004-ben szerezte. Kutatási területe a tárgyak internete, szenzorhálózatok kommunikációs technológiái, illetve a hálózat- és információbiztonság.

2. OKOSVÁROS SZENZORHÁLÓZAT TÁMADÓK ÉS TÁMADÁSOK

Az alábbiakban felmérjük, hogy milyen támadókra és támadásokra számíthatunk az okos városok területén lévő szenzorhálózatok esetén. Maga az okos város helyszín meghatározza azt, hogy milyen alkalmazásokról beszélhetünk, illetve a szenzorhálózat technológia használata tovább szűkíti ezt a témakört. Felsorolunk tipikus okos város szenzorhálózat felhasználási területeket és megmutatjuk, hogy ott milyen támadások és támadási motivációk lehetségesek. Ebből fel lehet mérni a támadás veszélyeit, az okozott károkat, így végül majd következtetni lehet a védelem nagyságára.

A vizsgálatok bemutatását megelőzően csoportosítjuk a támadásokat is. A támadás célpontja lehet az adat vagy információ, amelyet a szenzor gyűjt. Ebben az esetben támadás lehet az adat megmásítása, illetve akár az adat eltűntetése vagy hamis adat gyártása is. A támadás történhet a szenzor helyszínén, akár az érzékelők fizikai befolyásolásával, történhet az adatátvitel során, amíg az adat eljut a szenzortól a gyűjtőig, illetve történhet a feldolgozás helyén is. Lehet célpont maga az eszköz is. Egyrészt pusztán az értéke miatt, másrészt pedig lehetséges, hogy más támadásokhoz kapcsolódóan szükséges az adott eszköz megtámadása. Jellemző az is, hogy a támadás célja pusztán a rombolás. Leggyakoribb a szolgálat megtagadás támadás (DoS – Denial of Service) illetve ennek elosztott változata (DDoS – Distributed Denial of Service). A támadás során az eszközt működésképtelen állapotba viszik. Itt is jellemző, hogy nem pont maga az eszköz a célpont, hanem csak eszköz egy másik támadásban. DoS és DDoS esetében például ezek az eszközök állíthatóak át egy másik célpont támadására.

2.1 KÖRNYEZET MEGFIGYELÉSE

Jellemző okosváros szenzorhálózat felhasználási terület, hogy megfigyeljük a város területén mérhető környezeti paramétereket, amelyek egyrészt az időjárást jellemzik: hőmérséklet, napsütés, páratartalom, csapadék, légnyomás, szélsebesség, szélirány, napfény, UV sugárzás; másodrészt pedig a levegő minőségére adnak meghatározást: oxigéntartalom, ózonmérés, káros gázok mennyisége, por mennyisége és részecskenagysága; végül pedig további fizikai hatásokat vizsgálhatnak: vibráció, zaj. A mért értékeket felhasználva időjárás előrejelzéseket lehet készíteni, amely segítségével megjósolhatóak a közeljövő veszélyes időjárási jelenségei (viharok, jégesők), másrészt segítik a városi polgárok életvitelét is. Az adatok ugyancsak alkalmasak arra, hogy megmutassák, mennyire egészséges az adott időben a városi környezet és felhívják a figyelmet, ha valamilyen okból kifolyólag (UV sugárzás vagy légszennyezettség nagysága) a városi levegő és így a városban kint tartózkodás egészségtelen, sőt akár az allergiások életét is megkönnyíthetik.

A környezetmegfigyelő szenzorokat tipikusan vezeték nélküli szenzorhálózatban kötik össze és így juttatják el az adatokat a feldolgozó helyekre. A kihelyezett mérő állomások száma közepesen nagy, egy nagyvárosban párszáz ilyen eszközt helyeznek ki. Jellemzően egy ilyen mérőállomás több szenzorral is rendelkezik a különböző

mérésekhez. Az alkalmazást tekintve a sok szenzor adatát összevetve, múltbéli megfigyelésekre és modellekre építve határozzák meg az egész várost lefedő időjárási és egészségügyi információkat. A vezeték nélküli technológiát tekintve egyrészt a hagyományos mobil internetes kommunikáció használható a 2G technológiától kezdve a ma elérhető 4G technológiáig, másrészt a közelmúltban megjelent nagy kiterjedésű, alacsony energiafogyasztású rádiókat (LPWAN – Low Power Wide Area Network) lehet használni, mint például a SigFox [Zuniga 2016] és a LoRaWAN [Lora 2015]. Közös jellemzője a technológiáknak, hogy a mérőegységek egyenként vannak a mérési adatokat fogadó szolgáltatáshoz kötve, így egymástól teljesen függetlenül működnek. Az 2.1 ábra egy ilyen eszközt mutat, amely egy lámpaoszlopra van szerelve.

2.1. ábra. A Libelium cég környezet megfigyelő szenzora lámpaoszlopra szerelve

Forrás: a kép a Libelium cég oldaláról publikusan elérhető

Támadásokat és motivációkat tekintve meg kell vizsgálnunk, mennyire értékesek az eszközök, illetve az adatok, végső soron maga a szolgáltatás. Az környezeti információ, amelyet az eszköz mér és továbbít nagyon értékes a szolgáltatás szempontjából, azonban ma még ezekre az adatokra épülő szolgáltatások nincsenek úgy elterjedve, hogy kiesésük leküzdhetetlen problémát okozna az üzleti- vagy magánéletben. A támadásokat tekintve így az értékük nem túl nagy. Megfigyelhető az is, hogy az okos városokban található szenzorok mellett ugyanennek az információnak vannak alternatív forrásai is, amelyek talán drágábbak vagy kevésbé pontosak, de az alapvető funkciókat el tudják látni. Éppen ezért az is megállapítható, hogy noha a szolgáltatás figyelmeztet kritikus időjárási helyzetekre és egészségre ártalmas állapotokra, az alternatív adatforrások miatt emberélet még sincs veszélyben. A szenzor speciális felépítése és a kommunikáció jellege miatt az eszköz nem alkalmas más eszközök elleni támadás kivitelezésére. Összességében tehát elmondható, hogy a támadónak kevés haszna van az egység megtámadásával vagy megbénításával, emberi életek nem kerülnek veszélybe. A szolgáltatott adatok könnyen ellenőrizhetőek, így a támadás ténye gyorsan felderíthető. A támadó oldalt tekintve ugyanakkor az is elmondható, hogy az eszközök általában szabadon hozzáférhetőek és kis kockázattal módosíthatók vagy tönkretehetők. A támadás célpontja lehet maga az eszköz is, mert jelenleg ezek az eszközök értékes alkatrészeket tartalmaznak.

2.2 KÖZMŰVEK MÉRÉSE

Az okos városok szenzorhálózatainak egy tipikus alkalmazása a közmű fogyasztások mérése (2.2. ábra). A víz, gáz, elektromosáram-fogyasztás leolvasását így nem a leolvasók vagy a fogyasztók végzik, hanem azok akár a hagyományos leolvasási periódusnál gyakrabban, automatikusan leolvasódnak és az adatok megjelennek a szolgáltatónál. Az eszközök tipikusan a mérésre alkalmas szenzorból állnak, illetve egy rádiós kommunikációs eszközből, amely a mért adatokat a szolgáltató felé továbbítja. Egyelőre az implementációk zömében az eszközök közvetlenül kommunikálnak az adatgyűjtő szolgáltatással, de vannak olyan termékek is, ahol a mérők önmagukban is egy szövevényes vezeték nélküli szenzorhálózatot alkotnak, és egymást segítve továbbítják az adatokat. Kihívást jelent, hogy a vízmérők esetében leggyakrabban lebetonozott és mély vízaknákból kell a mért adatot elküldeni, amely problémát okoz a rádiós kapcsolatok esetében. A többi mérő esetében is előfordulhat, hogy a lakások belsejében már nem kedvezőek a rádiós viszonyok, illetve lakótelepeken gondot jelenthet a felhasználók nagy száma és közelsége. A mérő berendezések a felhasználók, illetve a szolgáltatók magánterületén helyezkednek el. Ahhoz, hogy egy támadó hozzá tudjon férni az eszközökhöz, be kell hatolnia az adott területre, amely jelentősen emeli a támadás felderítésének kockázatát.

2.2. ábra. Okos gázmérő a FŐGÁZ okosmérő projektjéből

Forrás: a kép a FŐGÁZ oldaláról publikusan elérhető

Közvetlen szenzor és gyűjtő kapcsolatok esetében a már említett mobil internet, illetve LPWAN technológiák jelentik a kapcsolatot, amelyen az adatokat továbbítják. Azokban az esetekben, amikor szövevényes szenzorhálózat biztosítja az adattovábbítást, ott tipikusan valamilyen alacsony fogyasztású, rövid hatótávolságú rádiós modult használnak. Tipikus a ZigBee [ZIGBEE 2009] technológia használata. Gyakran ötvöződik a két technológia és a házon/lakáson belüli fogyasztásmérőket rövid hatótávolságú rádiókkal kapcsolják össze, míg az egyik ilyen mérő kitüntetett szerepet kap, és átjáróként ő továbbítja a társaitól kapott adatokat a szolgáltatók felé valamelyik nagy hatótávolságú rádió segítségével.

Támadások esetén kézenfekvő annak a lehetősége, hogy a fogyasztó maga szeretné meghamisítani az adatokat azért, hogy a mérőóra kevesebbet mérjen, és így ne fizessen az igénybe vett szolgáltatásért. Ezeknek a támadásoknak egy része a mérő szenzorra irányul. Klasszikus megfelelője volt a mérőórák megakasztása. A támadás másik része a kommunikációra irányul, amely meghamisításával elérhető lenne, hogy a szolgáltató felé hamis fogyasztási adatok menjenek. A szolgáltató általában más eszközökkel is figyeli a felhasználók fogyasztását, így ezeknél a támadásoknál a támadás felismerésének nagy a kockázata. Mivel az elkövető az érdekek miatt maga a fogyasztó, így a lebukás veszélyének még magasabb a kockázata. Ettől függetlenül érdemes mégis megfelelően védeni a mérőműszereket és a kommunikációt. A szolgáltatás ideiglenes megbénításának nem sok értelme van a támadó részéről, ezek a mérők amúgy is csak ritkán mérnek.

Más a helyzet azonban azokkal a mérésekkel, ahol a mérések célja nem csak az időszakos leolvasás kiváltása, hanem a sűrű mérések segítségével fogyasztás előrejelzés és az elosztás segítése. A villamos áram fogyasztásánál a szolgáltatónak nagyon hasznos lehet az az információ, hogy hogyan alakul a fogyasztás a közeljövőben az elektromos hálózaton. Ennek segítségével a szabályozás egyszerűbbé, hatékonyabbá és olcsóbbá válik. Ebben az esetben, amikor a szabályozás részévé válik a fogyasztás mérése, a szolgáltatónak nagyon értékessé válnak az adatok. A téves adatok alkalmasak arra, hogy komoly veszteségeket okozzanak a szolgáltatónak, illetve a szolgáltatás megtagadás támadás is már nagy csapás. A támadó motivációja ebben az esetben egyértelműen a rombolás, hiszen számára anyagi előny nem igazán származik a szolgáltató veszteségeinek keletkezésével.

2.3 EGÉSZSÉGÜGYI ALKALMAZÁSOK

Az okos városon ugyan túlmutatnak, de mindenképpen az okos város alkalmazások részei azok a megoldások, ahol a városi polgárok életvitelét figyeljük meg szenzorokkal és segítjük őket abban, hogy fittek tudjanak maradni. Mára Magyarországon is elterjedt a fitnesz karkötők használata, amelyek az emberek aktivitását, mozgását vagy éppen annak hiányát mérik, ezen kívül pedig figyelik az alvását és ezeket kiértékelve próbálnak segíteni, hogy viselőik fittek lehessenek. Bizonyos modelleknél már megjelenik a pulzusmérés funkció, sőt véroxigén szintre is próbálnak következtetni. Az eszközök gyakran csak a megfelelő szenzort és egy kommunikációs modult tartalmazzák, a kiértékelés és visszajelzés már nem az eszközön történik, hanem valahol egy olyan szolgáltatónál, aki fogadja és feldolgozza ezeket az adatokat. A szolgáltató nem feltétlenül eszköz specifikus, a Google például nem gyárt ilyen eszközöket, mégis foglalkozik adatfeldolgozással és gyárt is alkalmazásokat, amelyek megjelenítik ezeket az adatokat.

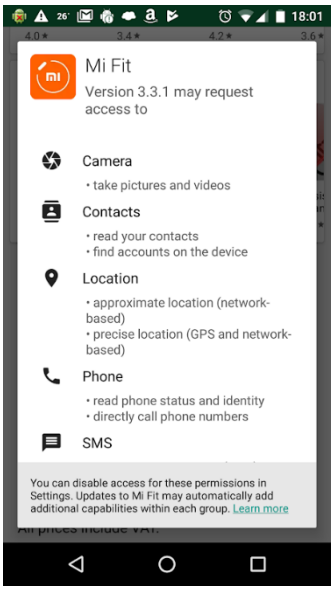
2.3. ábra. Fitneszkarperec és pedométer

Forrás: a kép a szerző saját felvétele

A fitneszeszközök méretükből adódóan is nagyon kevés tápellátással rendelkeznek, ugyanakkor a viseléseknél elvárt, hogy hosszú távon működőképesek maradjanak (2.3 ábra). Mivel sok esetben a közel valós idejű adatfeldolgozás is elvárt, így mindenképpen sűrűn kommunikálnak egy feldolgozó felé, amit a kényelmes viselet miatt vezeték nélkül tesznek meg. Éppen ezért kis fogyasztású, kis hatótávolságú rádiókat találunk ezeken az eszközökön. Leggyakoribb megoldás a Bluetooth LE [Sandhya 2012] protokoll alkalmazása, de komolyabb sporteszköz esetén szóba jön az ANT+ [Smith 2011] protokoll is. A rövid távú kapcsolat a felhasználó okostelefonjáig tart, majd onnan már mobil internet vagy WiFi kapcsolaton és utána vezetékes összekötésen keresztül jut el egy feldolgozóhoz. A feldolgozott adatok is az okostelefon képernyőjén keresztül jelennek meg.

A támadó szándékokat vizsgálva az adatok ugyan más számára értéktelennek tűnnek, ugyanakkor mégis sok ember nagyobb jelentőséget tulajdonít annak, hogy a magánszféra védelmének részeként ne lehessen ezeket mások által megismerni. Az adatok tehát azért értékesek, mert a magánszférához tartoznak és a védelmet is ennek megfelelően kell felállítani, még úgy is, hogy a támadó motivációja nem igazolt. Az eszközök kapcsán azonban felmerül egy másfajta dimenziója is a támadásoknak, mégpedig az, hogy a kommunikációs és prezentációs elemnek megjelenik az okostelefon is. Mint látni fogjuk az okostelefon része lesz más okosváros alkalmazásoknak is, sőt sok esetben még inkább érintett lesz, mégis már itt, ennél a pontál érdemes foglalkozni vele. Az okostelefon még nagyobb mértékben érintett a magánszféra támadásokban, mint azok az adatok, amik a fitnesz eszközök és az okostelefon között mennek. A támadó célja lehet tehát irányítást szerezni az okostelefon vezérlése felett és onnan, a telefonból szerezni meg információkat. Itt már anyagi érdekeltség is van.

A telefon tárolhat titkokat, amelyek pénzügyi tranzakciók indítását eredményezhetik, illetve tárolhat egyéb olyan információkat, amelyek nem kerülhetnek máshoz. Ez utóbbi esetben vagy maga az információ értékes és akár eladható, vagy a felhasználható megzsarolható az információ, általában privát fotók, rögzített hanganyagok, videók nyilvánosságra hozásával. A támadás érkezhet a fitnesz eszközhöz ajánlott app képében, ahol vagy egy olyan alkalmazásról van szó, amely átveri a felhasználót, vagy maga az eszközgyártó és szolgáltató él vissza a helyzetével. Ezekben az esetekben tehát különös figyelmet kell fordítani az okostelefonok védelmére főleg abban a tekintetben, hogy az ott futó alkalmazások mihez férhetnek hozzá. A 2.4. ábra egy népszerű Kínai fitnesz alkalmazást, a Mi Fit app hozzáférési kérelmeit mutatja. Látható, hogy az app futtatásával a felhasználó megengedi azt is, hogy az alkalmazás hozzáférjen a telefonon tárolt kapcsolatokhoz, telefonhívásokat és SMS üzeneteket is kezelhet, illetve a kamera segítségével képet, hangot és videót rögzíthet.



2.4. ábra. A Mi Fit fitneszalkalmazás által igényelt hozzáférés-engedélyek

Forrás: a kép a szerző saját felvétele

2017 novemberében a Strava cég büszkélkedett el vele [Hsu 2018], hogy alkalmazása, amelyet többszázezren használnak, milyen sok helyre eljutott már és egy nyilvánosan elérhető térképre tette fel az adatgyűjtések helyét. A probléma az lett, hogy a térképre felkerültek titkosnak hitt katonai bázisok is idegen országokban, pusztán azért, mert az ottani katonák egy része is használta az alkalmazást. Ebben a konkrét esetben így már túlmutat a magánszféra védelmén az alkalmazás adatainak biztonsága.

Más, egészségügy témakörben említhető okosváros alkalmazások beteg emberek életmódját javítják, segítenek sérült vagy tartós betegséggel élő embereket. Bár az öregség nem betegség, de ide sorolhatóak azok a szenzorhálózati alkalmazások is, amelyek idős embereknek nyújtanak segítséget azáltal, hogy megfigyelik életvitelüket. Az alkalmazások egy része tájékoztató jellegű, de sajnos sok olyan felhasználó is van, aki rákényszerül ezeknek az eszközöknek a használatára egészségügyi állapota miatt. Ilyenek a vakok, gyengén látók, siketek. Ide sorolhatóak azok a cukorbetegek is, akik az inzulint automatikusan mérik és adagolják műszerek segítségével, valamint minden egyéb olyan eset, ahol a gyógyszer adagolásában és akár bevitelében egy alkalmazás segít az embernek. Bár a legtöbb ilyen eszköz önmagában működőképes, több olyan is van, amely pont azért, mert megfigyeléseket végez vagy azért, hogy naprakész információkkal rendelkezzen, hálózatba van kötve. Itt a legkülönfélébb technológiákkal találkozhatunk az összeköttetés területén a direkt kapcsolatoktól kezdve az okostelefonon keresztül átjátszott adatokig.

A támadásokat tekintve itt egyértelműen van életveszély, hiszen a beteg vagy sérült emberek megbíznak ezekben az eszközökben, amely bizalom bizony életveszélybe is sodorhatja őket. A feltételezett támadó ugyanakkor anyagilag nem motivált ilyen támadások kivitelezésében. Kimondottan emberélet ellen irányuló támadások sem jellemzően ilyen célpontot választanak. Éppen ezért megfigyelhető, hogy ezeknek az eszközöknek a védelme nincsen az emberélet feltételezett árához igazítva. Ezt az értékelést az is nehezítené, hogy a felhasználóknak vagy a róluk gondoskodó társadalomnak jellemzően nincsen pénze költséges védelmi megoldások megvételére, így a valóban megfelelő védelemmel ellátott eszközök nem üzletképesek.

2.4 OKOSOTTHON VEZÉRLÉS

Az okosotthon vezérlés esetében már Magyarországon is észlelhető az előrelépés, egyre több otthon egyre több eszköze válik vezérelhetővé, illetve szolgáltat adatokat. Az eszközök ma már a ház építése során is beépítésre kerülnek, ebben az esetben leginkább a vezetékes technológiát alkalmazzák, míg az utólagos beépítésnél a vezeték nélküli telepítések vannak többségben. Az alkalmazások szenzorokat és beavatkozókat fednek le, amelyek folyamatosan figyelik a ház, kert és lakók állapotát, illetve szükség esetén vagy kérésre beavatkoznak. Tipikus szenzorok a lakótéren belül a hőmérők, páratartalom mérők, fénymérők, kapcsolók, szénmonoxid és füst mérők, illetve a vagyonbiztonsággal kapcsolatos szenzorok, mint nyitás érzékelők, mozgás érzékelők, rezgés érzékelők. A beavatkozók területén sok és különféle világító berendezéssel találkozunk, hőmérséklet szabályozókkal, találunk ablak és ajtó nyitót, redőnymozgatót, sőt ma már több háztartási kis és nagyberendezés is része lehet a szenzorhálózatnak, úgymint hűtőgépek, mosógépek, mosogatógépek, sütők stb.

A vezeték nélküli kommunikációs esetekben általában valamelyik rövid hatótávolságú alacsony fogyasztású rádiós technológiát választják, mint Bluetooth LE vagy ZigBee, de az sem ritka, hogy éppen azért választanak alacsony fogyasztású, nagy hatótávolságú rádió modult, mint például a LoRa, hogy a rádiós támadásokat kivédve is kommunikációképes maradjon az eszköz. A rövid hatótávolságú rádiók esetében az otthon területén van egy átjáró (gateway) is, amely vagy lokálisan vezérli az eszközöket, vagy létezik kapcsolat egy szolgáltató felé.

A támadó szempontjából itt elsősorban az otthonokban található vagyon a célpont. Az támadó egyrészt információt szerezhet arról, hogy a lakó otthon tartózkodik-e vagy esetleg elutazott hosszabb időre. Megjegyzésként, pont ilyen céllal vannak olyan okosotthon eszközök, amelyek hosszabb távollét alatt pont az szimulálják, mintha a felhasználó otthon lenne például a fények kapcsolgatásával. Elutazásra utaló jel lehet a termosztát beállítása vagy a lakás hőmérséklete vagy akár a mozgásszenzorok jelzései. A támadónak hozzá kell férnie a szenzorhálózathoz, amely vezeték nélküli kivitelezés esetében akár behatolás nélkül is megtehető. A betörés esetén további célpontok lehetnek a vagyonbiztonsági érzékelők, amelyeket a betörő összezavarhat vagy megbéníthat. A támadó az eszközök hibáit kihasználva akár magát a betörést is egyszerűbben kivitelezheti. Az okos zárak esetében nagyon fontos ezért a megfelelő védelem. Egy 2016-os konferencián az amerikai Merculite Security munkatársai 16 okos zárból 12-nél találtak biztonsági problémát [Rose 2016], többek között például nyíltan átvitt jelszavakat.

A vagyonvédelem mellett a támadó megbéníthat olyan szenzorokat is, amelyek emberéletek mentésére lettek telepítve, úgymint füst és szénmonoxid szenzorok. Az ilyen esetek szerencsére önmagukban ritkák, a támadónak nem érdeke ez a támadás, így inkább, mint egy másik támadáshoz kapcsolódóan bekövetkezett esemény jelenhet meg. Az otthonok beavatkozói működésük során nem alkalmasak emberélt kioltására, így a támadásoknak ez szintén nem lehet célja.

Hasonlóan a korábbi okosváros alkalmazásokhoz, itt is felmerül, hogy a támadó célja magánszférába tartozó információk lopása, egyrészt az információk értéke miatt, másrészt akár zsarolásokhoz. Az érzékeny információt az okosotthonok kamerái és mikrofonjai szolgáltathatják, amelyek rögzítik a lakásban történő eseményeket akár a nap 24 órájában.

2.5 OKOSVÁROS KÖZLEKEDÉSI ALKALMAZÁSOK

Sajnos a nagyvárosi élet ma már sokszor együtt jár a dugókkal és parkolási problémákkal, amelyeken okosvárosok szenzorhálózatára épülő alkalmazások próbálnak enyhíteni. A dugók esetében felhasználhatóak úttestbe épített szenzorok vagy lámpákhoz esetleg külön póznákra kihelyezett kamerák képei. Minden esetben a forgalmat figyelik és az adatokból nyert információt juttatják el a feldolgozóig. Az eszközökből nyert adatok mobil interneten vagy saját, többnyire vezetékes hálózaton keresztül jutnak el az eszközöket üzemeltetőhöz, aki utána már a feldolgozott információt értékesíti. Másik megoldás ugyanebben a problémakörben, hogy a járművekre, tipikusan taxikra, tömegközlekedési eszközökre szerelnek fel szenzorokat és így figyelik azok mozgását, majd abból következtetnek a forgalmi helyzetekre. Itt mindenképpen mobil kapcsolat, leginkább mobil internetes összeköttetés van, az adatok az eszközöket üzemeltető szolgáltatóhoz mennek. Van azonban egy harmadik megoldás is, amely műszakilag az előző megoldáshoz hasonlít, azonban nagy különbség, hogy a felhasználók a saját eszközeiket használják és az adat feldolgozása már nem az üzemeltetőnél történik. Ezek az úgynevezett közösségi alkalmazások, ahol a felhasználó csatlakozik a közösséghez és egyúttal ő maga is adatot szolgáltat a szolgáltatáshoz. A legnépszerűbb ilyen alkalmazás az itthon is jól ismert Waze, amely éppen azt a célt tűzi ki, hogy a közlekedésben segítsen gyorsabban és problémamentesebben eljutni A-ból B-be. A szenzorok itt a felhasználók GPS vevővel ellátott okostelefonjai és a kapcsolat mobil internetes.

A támadó anyagi hasznait kutatva nehezen található olyan szituáció, amely anyagi előnyhöz juttatná a támadót a városi dugó megelőzést és elkerülést szolgáló alkalmazások között. A dugó szituáció nem fordítható közvetlenül anyagi haszonra. Elképzelhető viszont, hogy a támadónak itt nem anyagi céljai vannak, hanem a saját közlekedését szeretné elősegíteni azáltal, hogy a szenzorhálózaton keresztül a szolgáltatásba beavatkozva eltereli a forgalmat azokról a pontokról, ahol ő maga szeretne gyorsabban átjutni. Szintén valós lehet a rombolási szándék, a szenzorhálózat adatainak megzavarásával ugyanis a támadó könnyen visszájára fordíthatja az alkalmazás erejét és idézhet elő forgalmi dugókat. Jelen implementációk esetében emberi élet nem kerül még veszélybe, mert a jelzőlámpák vezérlése nem közvetlenül az alkalmazásból történik, de a jövőben nem elképzelhetetlen, hogy az egyre nagyobb integráció miatt akár ez is megtörténhet. Természetesen az integráció várhatóan nem a közösségi szolgáltatásokra fog épülni. A támadó így potenciálisan gyilkolhat is, amely a közlekedéssel kapcsolatos terrorizmus egy újabb formája lehet. Ezeknél a rendszereknél így kiemelten fontos a megfelelő védelem az adatok és eszközök számára. A lámpákra, póznákra közterületeken kihelyezett eszközök esetében a támadó el is tulajdoníthatja az eszközöket, hogy abból értékes információkat nyerjen vagy értékesítse darabjait.

A másik jellemző szenzorhálózatra épülő okosváros alkalmazás az utcai parkolás problémáját segítő alkalmazások. Itt egy alkalmas, jellemzően a parkolóhely helyén telepített szenzor figyeli a parkolásokat és a be- és kiparkolási eseményeket jelenti egy szolgáltatás felé. A szenzorok vagy az úttest burkolatában helyezkednek el, vagy rálátással rendelkeznek a parkolóhelyre. Előbbi esetben mágneses, ultrahangos vagy optikai érzékelőkkel érzékelik a parkolást. Az aszfaltba épített változat esetén vezeték nélküli kommunikációt használnak a szenzortól a szolgáltatás gyűjtőjéig. Beépítés esetén cél, hogy a burkolatot ne kelljen sűrűn megbontani az elemek cseréje miatt, így mindenképpen alacsony fogyasztású komponenseket használnak. A megoldások között találunk olyat, ahol egy alacsony fogyasztású és rövid hatótávolságú rádiót használ, majd egy közeli átjátszó, ami tipikusan még az utcában található egy villanyoszlopon kihelyezve, továbbítja az adást mobil internet vagy nagy hatótávolságú rádió segítségével a szolgáltatóhoz. Olyan megoldás is ismert, ahol a nagy hatótávolságú, de alacsony fogyasztású

rádió rögtön a szenzorhoz kerül. A burkolatba helyezett szenzor esetén, amely felett jó eséllyel még egy autó is parkol, még a nagy hatótávolságú rádió esetén is kellhet a közeli átjátszó állomás (2.5. ábra).

2.5. ábra. Burkolatba helyezhető parkolásszenzorok a Libeliumtól

Forrás: a képek a gyártó honlapján publikusan elérhetők

A videós megfigyelés esetén általában vezetékes kapcsolat van, amely a szolgáltatóhoz viszi a videó adatokat elemzésre.

A támadó szemszögéből van anyagi haszon, amennyiben nem kell fizetnie a parkolásért, azonban ez csak abban az esetben éri meg a támadónak, ha kevés befektetésbe kerül számára az alkalmazott védelem kijátszása. Mivel a parkolási díjak, bár sokszor magasnak tűnnek, mégis olyan értéket képviselnek, amely nehezen összemérhető az alapvető védelem kijátszásának költségével, így a szolgáltatónak nem kell nagyon magas szintű biztonsági megoldásokat alkalmaznia. Annál nehezebb ugyanakkor védekeznie a rombolásos és szolgáltatás megbénításos támadások ellen. Az eszközök közterületeken vannak, ráadásul a burkolatba épített változatok parkolás esetén takarva is vannak. Fennáll a veszélye, hogy az eszközt szoftveresen feltörik, működésre alkalmatlanná teszik vagy eltulajdonítják. Gondatlan implementáció esetén egyetlen eszköz megtámadása esetén akár a teljes hálózathoz hozzáférést szerezhetnek. A rongálás esetén így nem csak az adott eszközt, hanem a szolgáltató nagyobb, akár teljes hálózatát megbéníthatják, jelentős gazdasági veszteséget okozva ezzel számára. Szintén rongálás és szolgáltatás megtagadás veszélye áll fenn a közterületen kihelyezett gyűjtő és továbbító pontok esetén. Ennek megrongálásával egy körzet tehető elérhetetlenné, de itt is fennáll a veszélye az imént említett szoftveres törésnek is, amelynek vége egy nagyobb területet érintő szolgáltatás kiesés lehet.

A kamerás megoldások esetében is hasonló a helyzet, a korábban említett gondok itt is fennállnak. Itt azonban szintén, mint minden kamerás megoldás esetében előkerül a magánszféra védelmének problémája. A kamera óhatatlanul rögzít olyan eseményeket is, amelyek alkalmasak lehetnek arra, hogy információt szerezzünk meg magánszemélyekről, akik erről nem is tudnak. A képen láthatóak és felismerhetőek lehetnek emberek, rendszámok, így mozgásukat nyomon lehet követni. Félő lehet az is, hogy a támadások direkt ezen információk megszerzésére irányulnak és nincs is közük a parkolási alkalmazáshoz.

A közlekedés témakörben végül érdemes még megvizsgálni az önvezető autókat, bár jelenleg még nincsenek elterjedve nálunk fejlettebb országokban sem. Az önvezető autók is szenzorhálózatot alkotnak, mert a közlekedésük során keletkező információkat, többek között videó anyagokat, közvetítik a gyártók felé, hogy azokat felhasználva ők tökéletesíteni tudják a vezetést. Feltehetően a jövőben az autókat majd lokális döntések alapján vezeti a gép, de nem zárható ki olyan eset sem, amikor majd az autókat távolról vezérlik, ilyen megoldások ugyanis már ma is léteznek. A lokális esetben is valószínűsíthető, hogy az autónak lesz kapcsolata a külvilággal így ezekben az

esetekben sem zárható ki a távvezérlés, akár úgy, hogy az autót kívülről átprogramozzák. Az autók a mobilitásuk miatt mindenképpen vezeték nélküli technológiával kapcsolódnak majd a hálózathoz, várhatóan a 4G-n túli technológiával, amely már valóban képes lesz kiszolgálni a valós idejű követelményeket is.

Az önvezető autó rendkívül nagy támadási felületet ad. Egyrészt maga az autó nagy értéket képvisel, ezért eltulajdonítása sok támadót vonzhat. Ebben az esetben egyébként nem is kell, hogy valóban önvezető autóról beszéljünk, ma egyelőre szélesebb körben terjedt el az úgynevezett „connected car" (hálózatba kötött autó), amely hálózati összeköttetést biztosít az autóval, és ezen az összeköttetésen keresztül is törhet fel autót vagy okozhat problémát a támadó. Mivel az autó nagy értékű, ezért a támadó nyeresége is nagyobb, így nagyobb befektetéseket vagy magasabb kockázatot is vállal. A másik, rettegett támadási felület a rombolás, illetve itt már bizony előkerül az emberélet kioltása is. Egyrészt az önvezető autó kerülhet olyan szituációba, ahol veszélyezteti utasait, másrészt viszont az autó távolról vezérelve más autókat és utasokat, gyalogosokat és akár egész épületeket veszélyeztethet. Bár ilyen incidenst még nem ismerünk, de kutatók már mutattak rá példát, hogy átvettek hatalmat egy arra alkalmas autó felett vezetés közben, amelynek aztán könnyűszerrel baleset lehetett volna a vége [Miller 2015]. Sajnos a terrorizmus világában pedig van rá példa, hogy kocsival emberek közé hajtanak, így jogos a félelem, hogy ugyanezt önvezető autókkal még könnyebben megtehetik. Az autók esetében látható, hogy muszáj fejlett védelmet alkalmazni, amelynek magasabb is a költsége. Az autók esetében azonban nagyobb értékről beszélünk, így a drágább védelmi megoldások nem okoznak olyan problémát, hogy akadályoznák a védelem kialakítását és fenntarthatóságát.

2.6 ÖNKORMÁNYZATI ALKALMAZÁSOK

A szenzorhálózatok az okosvárosokban sokszor önkormányzati feladatokkal és alkalmazásokkal is kapcsolatban vannak. Egyik jellemző ilyen feladat a közvilágítás, amelyet ma egyre több helyen próbálnak okos megoldásokkal üzemeltetni. Az okos közvilágítás esetén a lámpák csak ott világítanak, ahol aktuálisan valóban hasznos a fényük, egyéb helyeken és időben energiát megtakarítva a lámpák lekapcsolnak. A lámpákat azért kapcsolják szenzorhálózatra, mert így egyetlen, a lámpával érzékelt esemény több más lámpa fényére is hatással lehet. A kommunikációhoz használhatnak olyan megoldásokat, amelyek az elektromos hálózaton viszik a jeleket. Ahol ez nem valósítható meg, ott általában rádiós átvitelt használnak méghozzá vagy mobil internetet vagy nagy távolságú, alacsony fogyasztású rádiókat.

A támadások szempontjából nem igazán vonzó a közvilágítás, hiszen a támadónak nincsen közvetlen gazdasági nyeresége a támadás által. Rombolás, szolgáltatás megtagadás elképzelhető, illetve az is lehet, hogy a támadó szándékosan sötétít el egy adott helyet, amelyen aztán más támadást hajt végre.

Szintén az önkormányzati csoportba tartoznak azok a mobil alkalmazások, amelyek segítségével a városlakó polgárok bejelenthetik azokat az eseményeket, amelyekkel találkoznak napi tevékenységük során. Bejelenthetnek például egy kátyút, amelyet az úton látnak vagy egy kidőlt fát, amely a járdára borult. A felhasználók a saját okostelefonjaikat használják, és ezzel készítenek és osztanak meg eseményekről készített felvételeket, megjegyzéseket. Kommunikációra is az okostelefon szolgál, vagy otthoni WiFi-kapcsolattal vagy mobilinternet segítségével.

A támadások szempontjából az okostelefon jelentheti a célpontot. Itt ugyanazok a veszélyek vannak, amelyeket a fitnesz alkalmazások kapcsán említettünk. A támadások a magánszférát érik azáltal, hogy visszaélnek azokkal a jogosultságokkal, amelyeket a felhasználó jóváhagy a telepítés során. Annyival azonban talán szerencsésebb a helyzet, hogy az appok forrása a lokalitás miatt kevésbé idegen, így a támadó számára ez magasabb kockázatot jelent.

2.7 TÁMADÁSOK SZENZORHÁLÓZATOS OKOSVÁROS-ALKALMAZÁSOK ELLEN

A fenti alkalmazáscsoportok és támadásaik ismertetése után egy táblázatban röviden összefoglaljuk az egyes eseteket.

2.1. táblázat. Alkalmazások, technológiák, támadások és védekezések

Alkalmazási terület	Szenzorhálózati technológia	Támadások	Védekezések
Környezet megfigyelés	• *Mobil internet* • *LPWAN*	• *Nincs közvetlen anyagi haszon* • *Közterületen elhelyezett eszközök*	• *Lopás elleni védelem*
Okos közművek - Mérőleolvasás	• *Mobil internet* • *LR-WPAN* • *LPWAN*	• *A felhasználó, mint támadó* • *Közvetett anyagi haszon* • *Magas kockázat*	• *Adatok védelme, hitelessége*
Okos közművek – Villamosáram elosztás	• *Mobil internet* • *LR-WPAN* • *LPWAN*	• *Szolgálatmegtagadás* • *Nagy károk*	• *Adatok védelme, hitelessége*
Egészségügy – fitnesz	• *LR-WPAN* • *WiFi*	• *Magánszféra elleni támadások* • *Kapcsolt támadások*	• *Magánszféra védelme* • *Átjáró védelme*
Egészségügy – betegellátás, sérültek, öregedés	• *Mobil internet* • *LR-WPAN* • *WiFi*	• *Emberi élet veszélyben, de nem célpont* • *Nincs közvetlen anyagi haszon* • *Magánszféra elleni támadások*	• *Alacsony szintű és költségű védelem*
Okosotthon - otthonvezérlés	• *WiFi* • *LR-WPAN*	• *Kapcsolt támadás (betörés)*	• *Adatok, vezérlés védelme* • *Hozzáférés védelem* • *Átjáró védelme*
Okosotthon – biztonság	• *Mobil internet* • *LPWAN* • *WiFi*	• *Jelentős anyagi haszon* • *Magánszféra elleni támadások*	• *Hozzáférés védelem* • *Magánszféra védelme* • *Átjáró védelme*

Alkalmazási terület	Szenzorhálózati technológia	Támadások	Védekezések
Okos közlekedés – Forgalmi dugók	• *Vezetékes kamerák* • *Mobil internet*	• *Minimális haszon* • *Rombolás* • *Szolgálatmegtagadás* • *Közterületen elhelyezett eszközök*	• *Hozzáférés védelem* • *Adatok védelme, hitelessége* • *Magánszféra védelme* • *Lopás elleni védelem*
Okos közlekedés – Parkolás	• *Vezetékes kamerák* • *LR-WPAN* • *LPWAN*	• *Kis mértékű anyagi haszon* • *Rombolás* • *Szolgálatmegtagadás* • *Magánszféra elleni támadások* • *Közterületen elhelyezett eszközök*	• *Hozzáférés védelem* • *Adatok védelme, hitelessége* • *Magánszféra védelme* • *Lopás elleni védelem*
Okos közlekedés – önvezető autók	• *Mobil internet (5G+)*	• *Jelentős anyagi haszon* • *Terrorizmus, rombolás* • *Magánszféra elleni támadások*	• *Hozzáférés védelem* • *Adatok, vezérlés védelme, hitelessége* • *Magánszféra védelme*
Önkormányzati alkalmazások – közvilágítás	• *Vezetékes megoldások* • *LPWAN*	• *Nincs közvetlen anyagi haszon* • *Szolgálatmegtagadás* • *Kapcsolt támadások*	• *Vezérlés védelme*
Önkormányzati alkalmazások – közösségi alkalmazások	• *Mobil internet* • *WiFi*	• *Nincs közvetlen anyagi haszon* • *Magánszféra elleni támadások* • *Kapcsolt támadások*	• *Magánszféra védelme*

A táblázatból az is látható, hogy az okosváros alkalmazások rendkívül sokfélék, mind az alkalmazott szenzorhálózati technológiát, mind a támadásokat és ezzel kapcsolatos védekezéseket tekintve. Általános védelem számukra nem létezik. Létezik azonban egy alap szintű védelem, leggyakrabban a szenzorhálózati technológiához kapcsoltan. Ezzel a védelemmel a támadások egy jókora részét meg lehet akadályozni, de természetesen a speciális igényeket nem feltétlenül elégíti ki. A biztonságot azonban nem csak a szenzorhálózati részben kell elhelyezni, vannak olyan védelmi elemek, amelyek más rétegekben sokkal hatékonyabbak tudnak lenni. Leginkább a magánszféra védelme ilyen. Ebben a kismonográfiában csak azokat a megoldásokat mutatjuk be, amelyek a szenzorhálózatokhoz kapcsolódnak.

Több okosváros alkalmazás mellett is szerepel a mobil internet technológia, hiszen ezt a technológiát valóban alkalmazzák az adott megoldásnál. A mobil internet a 2G rendszerekben megjelent GPRS (General Packet Radio Service) technológiával kezdődik és ma az LTE (Long Term Evolution) megoldásoknál jár. A közelmúltban megjelentek olyan elemek is a szabványokban, amelyek kimondottan szenzorhálózatos működést, illetve gépek közötti kommunikációt céloznak meg, mint például az NB-IoT (Narrowband IoT) és LTE-CatM1. Mindazonáltal a 2G-3G-4G

mobil technológia nem tartozik a klasszikus szenzorhálózati technológiák köze, a biztonsági kihívásokat így csak az utóbb említett NB-IoT és LTE-CatM1 körben fogjuk vizsgálni.

A táblázatból és a részletesebb kifejtéséből is látható, hogy az önvezető autók kimondottan magas kockázatot jelentenek, a támadások irányulhatnak vagyonra és emberéletre is és a támadók magasabb kockázatot is vállalnak. Az ott felhasznált kommunikációs technológia még kidolgozás alatt van így erről a témakörről nem szól ez a kismonográfia. A mostani dokumentumban szereplő technológiák azonban megmutatják azt, hogy mit lehet összerakni a biztonság területén egy korszerű protokoll esetében. Várhatóan az 5G megoldások nem hoznak majd merőben új biztonsági megközelítéseket, hanem a jelenleg elérhető legjobb technológiákat választják vagy fejlesztik tovább.

3. SZENZORHÁLÓZATOKHOZ IGAZÍTOTT KRIPTOGRÁFIA

A szenzorhálózatok esetében a hálózatban résztvevő elemek, főként a szenzorok számítási és kommunikációs kapacitása erősen korlátozott. Ennek legfőbb okai, hogy a szenzor mobilis, amely kizárja a vezetékes összeköttetést és így a tartós tápellátást, illetve sokszor a méret is lényeges, így hatalmas akkumulátorok sem jöhetnek szóba. Hozzájön még az is, hogy az alkalmazáshoz általában sok szenzor szükséges, amely viszont csak akkor eladható, ha ezek a szenzorok gazdaságosan állíthatóak elő, illetve üzemeltethetőek és ez azt is jelenti, hogy a teljesítményben is korlátozottak lesznek. A méret és teljesítmény korlátozás miatt, a ma ismert legtöbb klasszikus kriptográfiai algoritmus nem alkalmazható, hiszen ezek számítási igénye és futás közbeni tárigénye sok esetben jelentős. Problémát jelent az érzékeny adatok, mint például jelszavak és kulcsok titkos tárolása is, ugyanis a hagyományos mikrokontrollerek esetében nincs olyan tárhely, ahol ezeket valóban biztonságosan tárolni lehetne. Sok mikrokontroller ugyan rendelkezik kívülről nem olvasható memóriaterülettel, de fizikai hozzáférés esetében ez nem mindig működik tökéletesen. A szenzorok védelme pedig sok esetben szintén nehezen megoldható. További problémát jelent, hogy amennyiben a támadó hozzájut egy szenzorhoz, úgy nem csak az ott tárolt szenzitív adatokat, kulcsokat, programkódokat, protokollokat képes kiolvasni belőle, hanem a szenzort felhasználva, de már hamis adatokkal vezérelve akár aktívan is be tud avatkozni a hálózat forgalmába, ezzel pedig potenciálisan képes több más szenzor adatait is megszerezni.

A szenzorokhoz igazított kriptográfia nem egy teljesen új ága a kriptográfiának, hanem azokat a klasszikus megoldásokat választjuk ki, amelyek alkalmazhatóak szenzorhálózati körülmények között is. Az algoritmusoknak leginkább teljesítményben, valamint statikus és dinamikus tárhelyben illeszkednie kell a szenzorok tulajdonságaihoz. Érdemes azért áttekinteni, hogy mi az, amit elvárhatunk ma egy szenzor, pontosabban a hozzá illeszkedő mikrokontroller teljesítményétől. Az okosváros alkalmazások esetén, mint ahogy sokféle alkalmazással, úgy sokféle szenzorral is találkozhatunk. Az egyszerűbb érzékelők esetén, mint amilyen a hőmérséklet, páratartalom adatok begyűjtése, egyszerű mikrokontrollereket találunk. A gazdaságosság miatt pont akkora teljesítményük és memóriájuk van, ami a feladat elvégzéséhez szükséges. Teljesítmény területén jellemző a 10-100 MHz órajel, órajelenkénti műveletvégzés, csak egész számok kezelése, speciális, akár kriptográfiai utasításkészlet hiánya.

Tárhely kapcsán a programok elhelyezésére általában 8-64 KB memória áll rendelkezésre, futás közben pedig a programok 2-16 KB méretű területet foglalhatnak dinamikusan. Azok az alkalmazások, amelyek bonyolultabb műveleteket végeznek, esetleg már videót is kezelnek, komolyabb mikrokontrollerrel, sőt sok esetben már mini PC-vel rendelkeznek. Ezek teljesítménye jócskán felülmúlja az előbb említett példányokat, azonban még így is elmaradnak egy normál felhasználású PC mögött. Leginkább egy belépő szintű okostelefonra hasonlítanak mind teljesítményben, mind tárhelyben. A mikrokontrollerek vagy processzorok sebessége eléri a GHz nagyságrendet, több magjuk segítségével több szálat képesek egyszerre futtatni, rendelkezhetnek speciális utasításkészlettel kriptográfia vagy jelfeldolgozás területén. Tárhelyben gyakran bővíthetőek is így a tényleges méreteket az alkalmazásokhoz lehet igazítani, de alap esetben is jellemző a MB körüli statikus és dinamikus memória méret. Az utóbbi

csoporthoz tartozó szenzorok esetében a legtöbb kriptográfiai művelet nem jelent kihívást teljesítmény és tárhely szempontból, itt azok a műveletek érdekesek, amelyek még egy normál PC jellegű gépet is alapos munkavégzésre bírnak. A szenzorokhoz igazított kriptográfiát így inkább az előbb ismertetett csoporthoz választjuk, ahol valóban léteznek korlátok.

A kriptográfia algoritmusok lefutásán túl azonban van még egy tényező, amely jelentős hatással bír a rendszer összteljesítményére, ez pedig a kommunikációs protokoll. Szenzorhálózatok esetében ugyanis maga a hálózat is lehet korlátozó tényező, amikor a rádiós modulok alacsony fogyasztásra vannak kötelezve. Ezekben az esetekben ugyanis az alacsony fogyasztás egyben azt is jelenti, hogy a kommunikációs egységek mérete kisebb és lassabb a megszokottnál. Nem ritka a 32-64 byte közötti csomagméret és LPWAN esetében a bit per másodpercben (bps) mérhető sebesség. A kis méretből adódóan a kriptográfiánál használt paramétereknek is alkalmazkodniuk kell, különben túlságosan aránytalan lesz a biztonság miatt növekmény és ez a rendes forgalom rovására megy. Példaként egy mindössze 32 bájtos keretben aránytalan egy 8 bájt hosszú kriptográfiai védőkód használata. A lassú sebesség pedig korlátozást jelenthet az üzenetek számában, így a protokoll nem lehet tetszőlegesen bonyolult. Ezek a megszorítások szintén okosváros alkalmazás specifikusak és nem minden esetben állnak fenn.

3.1 ALAPVETŐ KRIPTOGRÁFIAI MŰVELETEK SZENZORHÁLÓZATOKBAN

A számítógépes információbiztonságban az alapvető megközelítést a CIA (Confidentiality – Integrity – Availability) modell [Cherdantseva 2013] mutatja (3.1. ábra). Ez a három alapkövetelmény, amelyet a legtöbb rendszernek teljesítenie kell. A *Confidentiality* szó bizalmasságot jelent, azt, hogy az adatokat csak azok a felek érhetik el, akik jogosultak az elérésre. A bizalmasság a kriptográfiában titkosítás segítségével érhető el. A titkosítást kulcsok segítségével végezzük és a kulcsokat csak azoknak juttatjuk el, akiknek hozzáférést biztosítunk a titkosított adathoz. Az *Integrity* az integritás védelmet, sértetlenséget jelent. Valójában nem tudjuk garantálni, hogy az adatok ne sérüljenek meg szándékos vagy természetes eredetű rongálódás hatására, azonban a kriptográfia segítségével ellenőrizni tudjuk az adat integritását és jelezni tudjuk, ha sérülés történt. A kriptográfiában védő kódokat használunk a sértetlenség ellenőrzésére, amelyeket leggyakrabban kulcsok segítségével irányítunk. A kódot csak az tudja elhelyezni, illetve az tudja ellenőrizni, aki rendelkezik hozzá a szükséges kulccsal. Amíg a titkosítás nem feltétlenül változtatja meg a védett adat méretét, addig az integritás védelem szükségszerűen megnöveli a védelemmel ellátott adat méretét. A növekedés mértéke arányban van a biztonsággal. A harmadik szó, az *Availability* rendelkezésre állást jelent. Ez a terület túlmutat a kriptográfián, hiszen a kriptográfiai algoritmusok sok esetben segíthetnek ugyan, de önmagukban nem tudják a rendelkezésre állást garantálni.

3.1. ábra. CIA-háromszög az információ biztonságban

Forrás: a szerző saját szerkesztése

3.2 SZENZORHÁLÓZATOKHOZ IGAZÍTOTT BIZALMASSÁG

3.2.1 Az AES algoritmus

Ma, több szempontot is figyelembe véve legjobb titkosítónak az AES (Advanced Encryption Standard) titkosítót tartják [Daemen 2013]. A szempontok között szerepel a teljesítmény is, így szerencsére ez az algoritmus alkalmas arra, hogy szenzorhálózatok esetében is alkalmazzák. Az algoritmus nevében is utal rá, hogy egy szabványról van szó és ez további előnyt jelent, hiszen így egy olyan titkosító eljárással van dolgunk, amely széles körben elterjedt és számíthatunk több termékben is az előfordulására.

Az AES algoritmust több versenyző algoritmus közül választották ki 2000-ben. A cél az volt, hogy egy olyan algoritmust találjanak, amely megfelelően biztonságos és megfelelő teljesítményt nyújt használata során, gazdaságos a mérete, egyszerűen megvalósítható és licensz-mentes. A 15 jelentkező közül végül a Vincent Rijmen, Joan Daemen alkotását a Rijndael algoritmust választották győztesnek. Sokak kritizálták ugyan a döntést, miszerint ebben a versenyben nem a legbiztonságosabb algoritmus győzőt, hiszen más elvárásoknak is meg kellett felelni, de ugyanakkor tény, hogy komoly biztonsági problémát az AES működésével szemben a mai napig nem találtak és nem is valószínű, hogy valaha is fognak. A létező problémák ismertek és a jelenlegi számítási kapacitások mellett elhanyagolható kockázatuk.

Az AES titkosító egy blokktitkosító algoritmus, amely azt jeleni, hogy a titkosítandó információt fix méretű blokkokban várja és a kimeneten is egy teljes blokk jelenik meg. A blokkok fix, 128 bit méretűek függetlenül egyéb paraméterektől. További jellemző az AES titkosítóra, hogy szimmetrikus kulcsú titkosító. Az információ titkosításához és a titkosítás feloldásához pontosan ugyanarra a kulcsra van szükség. Ez azt is jelenti egyben, hogy amennyiben az információ megosztásáról van szó, úgy a titkosítónak ezt a kulcsot is meg kell osztania az összes olyan féllel, aki hozzáférést kap a titkosított adatokhoz. Ez nem feltétlenül szükséges, ismerünk ugyanis és ismertetni is fogunk aszimmetrikus kulcsú titkosítókat, amelyek esetén a kulcsok megosztása nem szükséges. A kulcsméretet tekintve az AES algoritmus háromféle lehetőséget kínál: 128, 192 és 256 bites kulcsok. Az algoritmus működése is minimálisan eltér a három különböző méretű kulcs esetében, amely segítségével így a normális méretűnél

erősebb biztonságot is el lehet érni. Az AES titkosító felépítése alapján egy helyettesítő-permutáló hálózat, azaz elemi műveletei helyettesítésekre és permutációkra bonthatóak. Ez a felépítés lehetővé teszi, hogy hardveresen és szoftveresen is egyszerű legyen az implementáció, ugyanakkor képes arra is, hogy a megfelelő biztonságot nyújtsa. Valójában egyszerű építőelemekből épül fel egy bonyolultabb folyamat. A műveleteket az AES titkosító egy 4 x 4 bájt méretű tömbön végzi el, amelyet állapotként tárol az egyes lépések között. A titkosítást ugyanis az AES titkosító lépésenként végzi el, a lépések köröket alkotnak és a körök többször ismétlődnek, mielőtt kialakulna a végeredmény. A körök száma a legkisebb, de a gyakorlatban a legtöbbet alkalmazott, 128 bites kulcsméret esetében 10, a 192 bites kulcsméret esetén 12, míg a legbiztonságosabb 256 bites kulcs esetén 14. A titkosítás feloldása során a titkosításhoz használt kulcsot, de fordított köröket alkalmaznak, így nyerik vissza az eredeti információt.

A titkosító a működése során legelőször a kulcsot alakítja át úgy, hogy minden egyes körhöz különböző 128 bites kulcs tartozzon, illetve külön kulcs áll elő a legelső lépés számára is, amely még a körök megkezdése előtt megtörténik. Ez a folyamat a *KeyExpansion*. Ezt követően történik meg az első lépés a titkosítással. A titkosító bemenetére érkező információt a titkosító állapotába helyezik, majd végrehajtanak egy *AddRoundKey* műveletet. Az *AddRoundKey* művelet a titkosító pillanatnyi állapotát matematikai XOR (eXclusive OR – kizáró vagy logikai művelet) kapcsolattal módosítja a körhöz tartozó kulcs paraméterrel, hatására megváltozik a titkosító állapota. A legelső lépés külön kört alkot, külön kulccsal és a körök számolásában ez nem számít. Hiányában azonban a titkosító kevésbé lenne biztonságos. A legelső lépést követően indulnak el a titkosító körök, ahol az utolsó kört kivéve minden egyes körben pontosan ugyanaz a négy lépés ismétlődik. A kör első lépése a *SubBytes* művelet, amely egy helyettesítő művelet, az állapotban tárolt bájt értékeket helyettesíti egy másik értékre egy megadott táblázat alapján. A helyettesítő táblázat speciálisan van megalkotva, ügyelve arra, hogy az ismert támadásokat kivédje. Ez a lépés független a kulcs értékétől. A kör második lépése a *ShiftRows* művelet. A művelet során a 4 x 4-es tömbnek kezelt titkosító állapot sorait tolja el a pozíciójuk alapján. Az első sor nem kerül eltolásra, a második sor egyel tolódik, a harmadik kettővel, végül a negyedik hárommal. A művelet szintén nem igényli kulcs jelenlétét, ahogy majd a következő művelet sem. A harmadik lépés a körön belül a *MixColumns* művelet. Ennél a lépésnél a tömb oszlopain végzünk műveletet, valójában egy modulo mátrix szorzás történik egy meghatározott értékkel. Végül a kört a már ismertetett *AddroundKey* művelet zárja le, amely felhasználja a körhöz tartozó kulcsot is. Minden kör hasonlóan ismétli az ismertetett lépéseket, kivéve azonban az utolsó kört, ahol a *MixColumns* művelet elmarad. A titkosító kimenete az utolsó lépés után a titkosító állapotában maradt érték.

A fent ismertetett lépések nem bonyolultak ahhoz, hogy egy szerényebb képességű mikrokontroller megvalósítsa azokat elfogadható teljesítménnyel. Mégis, ahol a megfelelő tárterület rendelkezésre áll, ott ezeket a műveleteket is lehet gyorsítani. 32 bites mikrokontroller architektúrák esetén a kör első három műveletét ugyanis lehet helyettesíteni 16 helyettesítés művelettel és 12 XOR művelettel, ahol a helyettesítések négy darab 256 elemű táblázatot használnak. A négy táblázat 256 eleme egyenként 4 bájtot foglal el, így csak az AES titkosítás elvégzéséhez 4 KB memóriára lesz szükség, de segítségével valóban gyorsul a körök végrehajtási sebessége. Ez természetesen nem csak a mikrokontrollerek esetében igaz, bár komolyabb architektúrák esetén akár beépített utasításkészletet is találhatunk az AES titkosításhoz. Ugyanezt a feladatot meg lehet oldani egyetlen táblázat és eltolások segítségével, így a futás ugyan lassabb, de elegendő csupán 1 KB memória.

Mint látható az AES titkosító kedvező tulajdonságokkal rendelkezik és tárhely igénye is olyan, hogy akár egy egyszerűbb mikrokontroller is alkalmazhatja. Éppen ezért valóban alkalmazzák szenzorhálózatokban. Sőt, sok esetben a rádiós modul is tartalmazza ezt a titkosítót, így a szenzor mikrokontrollerének akár nem is kell implementálnia. Mindezek ellenére az AES mégsem jelenti mindig a tökéletes választást a szenzor adatok és egyéni

információk titkosítási feladatainak ellátására. Előfordul, hogy még az AES-nél is kisebb tárhely igényű vagy éppen gyorsabb, esetleg költséghatékonyabb vagy energiatakarékosabb implementáció szükséges. Minthogy az AES egy adott elvárás mentén történt optimalizálás eredménye, így sejthető, hogy jobb titkosítót csak valamilyen tulajdonság feladásával tudunk majd találni. Ez akár lehet a bizalmasság is. A következőkben bemutatunk olyan titkosítókat is, amelyek bizonyos helyzetekben alternatívái lehetnek a tökéletes AES titkosítónak.

3.2.2 Twofish titkosító

A Twofish [Schneier 1998] nevű titkosító Bruce Schneier kriptológus algoritmusa, amelyet 1998-ban készített. Az algoritmus versengett az AES címért, az 5 algoritmusos döntő körig jutott, de ott alulmaradt a Rijndael algoritmussal szemben. A Twofish algoritmus a Blowfish algoritmus továbbfejlesztése. A Twofish algoritmus kiemelkedő tulajdonsága, hogy a megfelelő biztonsági szint megtartása mellett lehetséges a futási sebességet a kódméret rovására növelni, illetve fordítva a kódméretet csökkenteni a sebesség rovására. Ha tehát egy igen gyors titkosító eljárásra van szükség, feltételezve egy lassú, de sok tárhellyel rendelkező mikrokontrollert, úgy a Twofish algoritmus implementációja gyorsabb lehet az AES implementációnál.

A Twofish szintén szimmetrikus blokktitkosító, akárcsak az AES. A blokk mérete itt is 128 bit, de a kulcsméret tekintetében kevésbé válogatós, ugyanis minden méretet elfogad 128 bit és 256 bit között. A szimmetrikus kulcsok miatt itt is pontosan ugyanarra a kulcsra van szükség a titkosításhoz, mint annak feloldásához. A titkosító a Feistel felépítést követi [Feistel 1973], amely egy korábban gyakran alkalmazott megoldás. A Feistel titkosító is több megismételt titkosító lépésből áll. Az architektúra előnye azonban, hogy a Rijndael felépítésével ellentétben nem kell minden egyes művelethez inverz műveletet biztosítani a titkosítás feloldásához, hanem a Feistel felépítésből adódóan elegendő a körökben használt részkulcsokat fordított sorrendben használni és máris a kívánt művelethez jutunk. A titkosítás itt is egyszerűbb műveletekből áll össze. A szerző szerint csak a szükséges lépéseket hagyták benn, minden, aminek a jelenléte nem volt matematikailag igazolható, azt elhagyták. Szintén fontosnak tartja a szerző, hogy az algoritmust úgy alkotta meg, hogy az akkor ismert titkosítás algoritmus-törő analitikáknak (differenciális kriptoanalízis és lineáris kriptoanalízis) ellenálljon, és egyben a még ismeretlen analitikák ellen is hatásos legyen.

A titkosítás itt is körönként történik. Minden körben a 128 bites blokknak csak a felével történik művelet. A 64 bites félblokkot tovább bontják két részre, majd a keletkező 4-4 darab 8 bites blokkot helyettesítik egy táblázat segítségével. A táblázat nem a véletlen műve, úgy van előállítva, hogy a legjobb titkosító teljesítményt nyújtsa és ezt tesztekkel igazolták. A helyettesítő táblázat ráadásul kulcsfüggő. A helyettesítés után egy mátrix műveletet végeznek az kapott eredményekkel, amely esetben szintén nem véletlen érték a művelet másik paramétere, hanem gondosan megválasztották és tesztelték azt. Az adott körben végzett műveleteket egy összeadás és egy XOR művelet zárja, ahol az utóbbi paramétere a körhöz kiszámított kulcsérték. Azért, hogy a kriptoanalízist még jobban megzavarják, a kör előtt és annak végén még egy 1 bites eltolást is alkalmaznak. A titkosító összesen 16 kört tesz meg. A körök elvégzése előtt és az összes kör elvégzése után egy egyszerű helyettesítés található, amely a biztonság további fokozása miatt van ott.

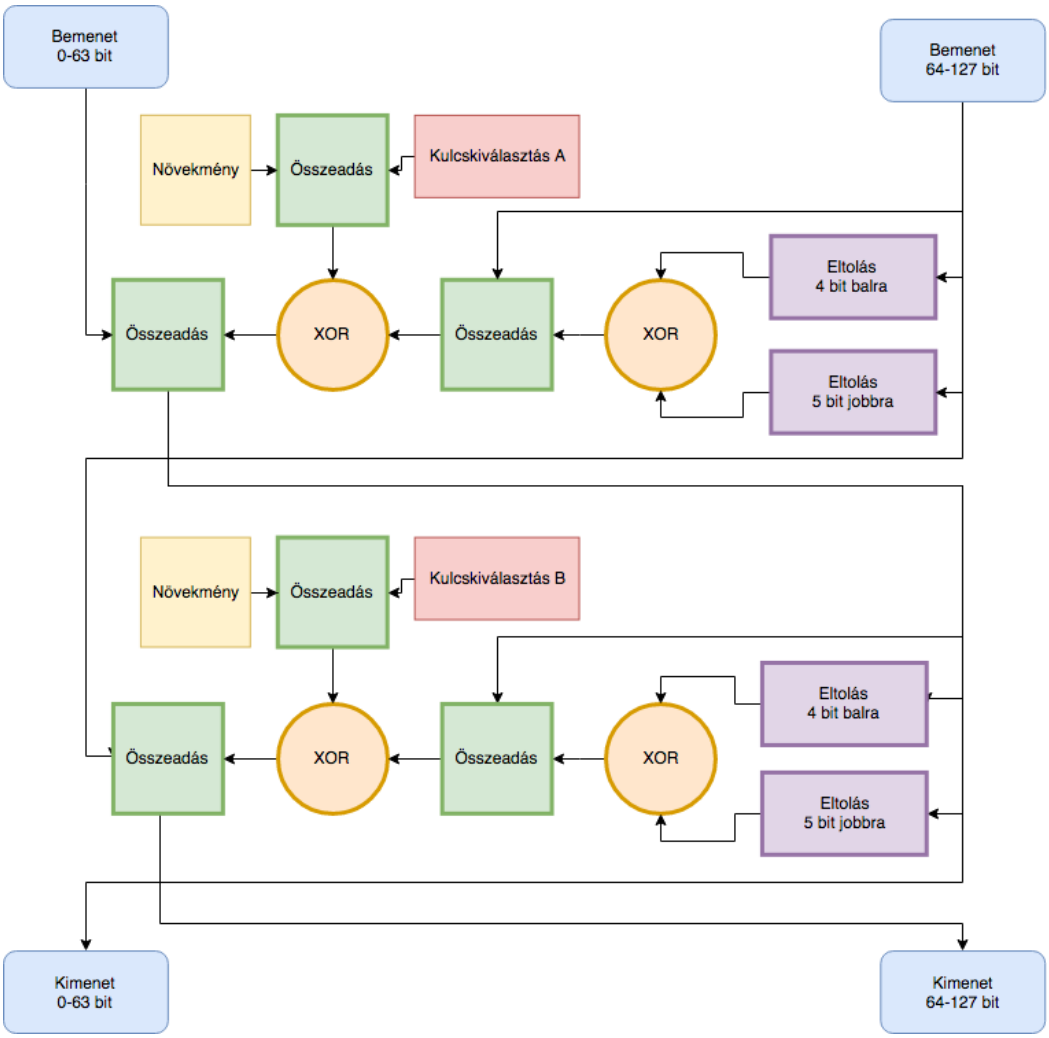
Ahogy írtuk, a Twofish algoritmus azért kiemelkedő, mert sokféleképpen kombinálható a futási teljesítmény és a tárterület igény. Egyrészt kombinálható, hogy a körönkénti kulcsok és helyettesítő táblázatok milyen gyorsan álljanak elő a körökben végzett műveletek rovására. Amennyiben nagymennyiségű adatot kell titkosítani, úgy

jobban járunk, ha a kulcsok előkészítése lassú, de azután, azokat megtartva a körönkénti titkosítások gyorsabbak. Gyorsan változó kulcsok és rövid adatok esetén viszont érdemes a kulcsok és táblázatok előkészítését gyorsabban elvégezni még akkor is, ha a titkosítás így egy kicsit lassabb lesz az előzőhöz képest. Több köztes fokozat is lehetséges, de példaként a leggyorsabb kulcselőkészítés esetén egy adott CPU architektúrán a kulcs előkészítése 1250 ciklus, míg a titkosítás 850 ciklus. Ugyanakkor, a leggyorsabb titkosítás esetén, a kulcs előkészítése 12700 ciklus és a titkosítás 285 ciklus. Ugyanazt a kulcsot használva, 21 blokk után már megéri az utóbbi implementáció választása. Nem csak futási idővel lehet operálni, hanem az elfoglalt memória méretével is. Egy egyszerűbb 8 bites mikroprocesszoron (mint például a több mint 40 éves MOS 6502) is elegendő 1760 bájt memória a titkosító kódjának a tárolására. Amennyiben a kulcs is tárolható a mikrokontroller statikus memóriájában, úgy a futáshoz mindössze 36 bájt memória szükséges. A statikus memória mérete tovább is csökkenthető, a szerző szerint további 350 bájt spórolás érhető el, azonban ekkor már a titkosítási sebesség a tizedére esik vissza.

A Twofish algoritmus végül alulmaradt a Rijndael algoritmussal szemben. A kutatók azonban kiszámolták azt is, hogy hány kört kellene megtennie az egyes titkosítóknak ahhoz, hogy az ismert módon feltörhetetlenek legyenek. A Twofish esetében azt találták, hogy a 16 kör helyett csupán 7 kör is elegendő lenne. A Rijndael algoritmus esetében is van csökkenés ebben a számban, azonban koránt sem ekkora, ott 8 kör esetén ismert támadás. A minimálisra beállított körszámok esetén a Twofish gyorsabb a Rijndael algoritmusnál hasonló biztonsági szint mellett. A szenzorhálózati alkalmazásnál opció lehet a körök számának csökkentése is a gyorsaság érdekében.

3.2.3 XTEA és XXTEA algoritmusok

Az XTEA és XXTEA algoritmusok mind a TEA (Tiny Encryption Algorithm) titkosító leszármazottai. Az X betű elvileg bővítést jelent, azonban az alap algoritmust nem volt elegendő csak bővíteni, hanem javítani is kellett. A nevét viszont onnan kapta az algoritmus, hogy a titkosító nagyon kicsi kódmérettel rendelkezik. Az XTEA algoritmus 1997-ben született meg [Wheeler 1997]. A titkosító Feistel architektúrájú blokk titkosító szimmetrikus kulccsal. A Blokk mérete 64 bit, a kulcs fixen 128 bit méretű és a titkosító 64 kört teszt meg minden egyes blokk titkosításánál. A körök egyszerű műveletekből: összeadásból, XOR bitenkénti logikai műveletből és bitenkénti eltolás műveletekből állnak. A körönkénti kulcsok előállítása is egyszerű, egy bitenkénti shift művelet segítségével választ a 128 bites kulcsból egy 32 bites részt. A szerzők által kiadott minta forráskódban a titkosítás, illetve titkosítás feloldás művelete csupán 10-10 sor. A 64 körös ismétlés ellenére a titkosítók nem nevezhetőek lassúnak sem. A titkosító egy körének működését a 3.2 ábra mutatja be. Amint látható, egyetlen körben 2 Feistel műveleti blokkot is végrehajt, amelyek között minimális a változás. A növekmény minden egyes lépésnél meghatározott értékkel növekedik. A kulcskiválasztás egy 32 bites értéket választ ki a kör sorszáma és a növekmény értéke alapján.



3.2. ábra. Az XTEA titkosító egy körének műveleti lépései

Forrás: a szerző saját szerkesztése

Az XTEA titkosító megjelenésével együtt megjelent egy Block TEA nevű algoritmus is, amely megkapta az XTEA körének műveleteit, de kiterjesztette a titkosítót úgy, hogy a blokkméret tetszőleges lehetett. Praktikusan a blokkméret a teljes adatot lefedte, azaz egyetlen nagy blokkot alkotott. A titkosítóban később hibát találtak, így javítani kellett. 1998-ban így született meg a Corrected Block TEA titkosító [YARRKOV 2010], amely az XXTEA rövidítést kapta. Az XXTEA titkosító esetében szintén szabadon választható a blokkméret, úgy, hogy az a 32 többszöröse és legalább 64 bit. Ebben az esetben is van tehát lehetőség arra, hogy egy blokk a teljes titkosítandó adatot lefedje. A körök száma változó és a blokkmérettől függ, de biztosan nagyobb, mint 6. A szabadon elérhető implementációban a 6 + 52 / n értéket adták meg, ahol n a blokkméret 32 bites egységekben. A körökben végzett műveleteket lecserélték, azonban azok továbbra is viszonylag egyszerű műveletek maradtak. Mint később kiderült, az XXTEA algoritmus sem hibamentes, ugyanakkor a törés használatához nagyon sok titkosított adatot kellene megfigyelni, így a gyakorlati életben egyelőre használható maradt.

3.2.4 Skipjack titkosító

A Skipjack titkosítót [Kim 2009] az NSA (US National Security Agency) alkotta meg és a kezdetekben a kriptográfiával foglalkozó kutatók éppen ezért negatívan fogadták, hiszen az NSA inkább titkosítók feltörésével vált hírhedté, nem pedig titkosítók megalkotásával. A kutató társadalom félt, hogy az NSA eleve olyan titkosítót gyárt, amelyet saját maga már fel tud törni. A kételyeket fokozta, hogy a megjelenésével egyidőben nem volt szabadon hozzáférhető a titkosító kódja, a titkosítót, mint egy feltörés biztos hardver elem bocsátották ki. Végül 1998-ban nyilvánosságra hozták a titkosító működését. A Skipjack algoritmust úgy alkották meg, hogy a lehető legegyszerűbb műveletei legyenek, így nagy sebességet tudjon elérni. A titkosító a már korábban is többször szóba került Feistel architektúrára épül. Blokktitkosító 64 bites blokkokkal, azonban érdekessége, hogy a kulcsok mindössze 80 bit nagyságúak. A titkosítás és titkosítás eltávolítása alatt a körök 32 ismétlést tesznek.

A Skipjack algoritmust publikálása után már többen megvizsgálták és sok publikáció született arról, hogy mennyire nem biztonságos a használata. Bár itt is igaz, hogy a publikált törések csak közelebb visznek az adatok feltöréséhez és csak elméletben működnek, a valóságban ezek nem működőképesek. Sokkal nagyobb gondot jelent azonban a kulcs mérete. Egy máig érvényes ökölszabály szerint minden titkosító feltörhetőnek számít, amelynek a kulcsmérete nem éri el a 100 bitet. Mivel a Skipjack titkosító csupán 80 bites kulcsot tartalmaz, ezért nem éri el a biztonságos szintet. Az USA sokáig kitartott mellette, hogy ennek ellenére a Skipjack biztonságos és használható, végül 2016-ban visszavonta a titkosító korábbi felhatalmazását, így már nem használható a kormány által is használt alkalmazásokban. Mindezek ellenére éppen gyorsasága és rövid kulcsmérete miatt alkalmas lehet szenzorhálózatokban történő felhasználásra. Mint látni fogjuk, éppen emiatt a tulajdonsága miatt több szenzorhálózati kommunikációs protokoll is használni fogja.

3.2.5 HIGHT titkosító

Végül pedig bemutatjuk a HIGHT algoritmust [Hong 2006], amelyet kimondottan úgy fejlesztettek ki, hogy a hardveres implementáció egyszerű legyen. A hardver esetében a mérőszám a kapuk száma, amelyeket a gyártás során a lapra kell integrálni. A HIGHT algoritmus szinte ugyanannyi kapu segítségével valósítják meg, mint az AES hardveres implementációját, mégis közel háromszor gyorsabban titkosít. A titkosítás során itt is Feistel architektúrát találunk, 64 bites blokkokkal, 128 bites kulccsal és 32 körrel. A körönkénti műveletek nagyon egyszerűek: 8 bites összeadás, kivonás, XOR és eltolás műveletek. Nem szerepel a lépések között helyettesítés, így a memóriával is takarékos tud lenni.

A HIGHT kimondottan hardverre optimalizált titkosítás, szoftveresen megvalósítva már nem annyira előnyös a használata.

3.3 BLOKKOK ÖSSZEFŰZÉSE

A most bemutatott, szenzorhálózatokba szánt titkosítók mindegyike blokktitkosító, azaz az adatokat blokkokban tagolva várják. Kivételt egyedül az XXTEA titkosító képvisel, amely ugyan szintén blokktitkosító, de itt a blokkmérete közel tetszőleges, így akár a teljes adatot is jelentheti. Minden más esetben, amikor a titkosítandó adat a

blokkméret többszöröse, szükség lesz két műveletre. Az egyik művelet a titkosítandó adatot darabolja blokkokra, a másik művelet pedig az esetlegesen részben megtöltött utolsó blokkot egészíti ki teljes blokká. Például egy 99 bájt méretű információ esetén, 64 bites blokkmérettel a daraboló 13 blokkot fog alkotni, és az utolsó blokkban majd csak 3 bájtnyi hasznos információ lesz. A műveletet lehet egyszerűen is végezni és a blokkok kiegészítési művelete (padding) valóban egyszerű is. Itt tulajdonképpen tetszőleges kitöltést lehetne alkalmazni, csupán arra kell ügyelni, hogy az, aki a titkosítás feloldását végzi, tudja majd, hogy melyik bitek tartoznak a kitöltéshez és melyikek nem. Erre a műveletre több egyszerű szabványos megoldás is létezik, általában a kitöltést bájtokban mérik, és a blokkban megjelölik a kitöltés hosszát.

Az üzenet blokkokra darabolása azonban nem ennyire egyszerű feladat. Pontosabban a darabolás egyszerű, hiszen elegendő a hosszú üzenetet blokk méretű darabokra vágni, azonban, ha a titkosításra így küldenénk el az egyes blokkokat, akkor az biztonsági problémához vezetne. A blokkokat ugyanis egy feltételezett támadó fel tudná cserélni és így más, de akár továbbra is értelmes üzenethez jutnánk a titkosítás eltávolítása után. Ahhoz, hogy ez ne történhessen meg, a blokkokat egymáshoz kell fűzni, hogy csak a helyes sorrendben lehessen megfelelően értelmezni azokat. A klasszikus információbiztonság esetében a CBC (Cipher Block Chaining) [Bellare 1994] és a CTR/ICM (Counter mode vagy Integer Counter mode) [Lipmaa 2000] a leggyakrabban használt eljárások blokkösszefűzésre. Minden esetben a titkosító változatlan marad, tehát tetszőleges blokktitkosítóval működik az eljárás. A blokkok összefűzése a titkosító blokk működése előtt, illetve után történik. A CBC blokkösszefűzés esetében az információt nyíltan tartalmazó aktuális bemeneti blokk titkosítása után a keletkező titkos blokkot az XOR művelet segítségével a következő bemeneti blokkhoz fűzzük. Így tehát a bemeneti blokkok függenek attól, hogy előtte milyen más blokkokat titkosítottunk már az információból. A titkosítás feloldása hasonlóan történik, így, ha a sorrendet a támadó összekeverné, az összekeverés pillanatától már nem kapnánk helyesen értelmezhető információt a kimeneten. A CBC művelet problémája viszont, hogy az titkosítás feloldásánál a blokkok csak pontosan egymás után érhetőek el. Ha valaki a sokadik blokk értékére kíváncsi csak, úgy előtte az összes korábbi blokkot fel kell oldania. Sok alkalmazás esetében, mint ahogy tipikusan a szenzorhálózati alkalmazások esetében sem jelent ez problémát, mert az adatok vagy eleve kisméretűek, vagy amúgy sem tudunk alkalmazni párhuzamosított feldolgozást. Azonban akár ezeken a területeken is érdemesebb a CTR blokkösszefűző algoritmust használni előnyös tulajdonságai miatt. A CTR mód esetén egy számlálót használunk, amely számolja, hogy sorrendben melyik blokkról van szó, es a számláló értékét XOR művelet segítségével összefűzzük az aktuális, titkosításra szánt információs blokk adataival, majd így végezzük el a titkosítást. A számláló értéke minden blokkra egyedi, így amennyiben egy támadás hatására rossz sorrendben próbálkoznánk, úgy nem kapnánk értelmezhető kimenetet a titkosítás feloldása után. A CTR megoldás előnye, hogy a blokkok titkosítását és a titkosítás eltávolítását tetszőleges sorrendben végezhetjük, sőt akár egymással párhuzamosan is történhet több blokkon az aktuális kriptográfiai művelet. Mindkét megoldás elterjedt a klasszikus információ- és hálózatbiztonsági alkalmazásokban.

3.4 FOLYAM TITKOSÍTÓK SZENZORHÁLÓZATOKHOZ

A korábban említett titkosítók mind kivétel nélkül blokktitkosítók voltak, amelyek az információt blokkonként titkosították. Léteznek azonban olyan titkosítók is, amelyek nem blokkokra épülnek, hanem a legkisebb egységenként, bitenként végzik a titkosítást. Ezek az úgynevezett *folyam titkosítók*. A folyam titkosítók esetében nincs szükség blokkokra és így a velük járó blokkösszefűzésekre és kitöltésekre sem. A folyam titkosítók is lehetnek

rendkívül gyorsak, sőt sok esetben gyorsabbak is, mint a blokktitkosítók. Éppen ezért szenzorhálózatokban ideális választás lehet. Általánosságban elmondható, hogy a gyakorlatban alkalmazott folyam titkosítók mind ugyanarra a működési mechanizmusra épülnek. A titkosító információval egyenlő bithosszúságú álvéletlen sorozatot állítanak elő, majd ezt a sorozatot az XOR művelet segítségével hozzákeverik a titkosítandó információhoz és már kész is a titkosított információ. A titkosítás feloldása ugyancsak egyszerű feladat, ugyanezt az álvéletlen sorozatot kell a titkosított információhoz keverni az XOR művelettel és máris visszaáll a titkosítatlan információ. Sőt, az is megállapítható, hogy amennyiben valóban véletlen bitsorozatot használnánk a titkosításnál, akkor a tökéletes titkosítás állna elő, amely feltörhetetlen. A probléma viszont ott van, hogy a véletlen bitsorozat előállítása nem egyszerű művelet, és ha elő is áll, akkor is kommunikáció esetén egy az egyben a másik félhez kellene szállítani, ami se nem hatékony, se nem egyszerű. Éppen ezért a folyam titkosítók a véletlen bitsorozatok helyett úgynevezett álvéletlen bitsorozatot használnak, amely ugyan véletlen bitsorozatnak tűnik, de az egyes bitek nem a véletlen, hanem valamilyen matematikai művelet eredményei. A folyam titkosítók ezen a ponton térnek el egymástól, azaz hogyan állítják elő az álvéletlen bitsorozatot egy kezelhetően rövid kulcs segítségével.

A folyam titkosítók kevésbé elterjedtek, mint a blokktitkosítók, ennek köszönhetően sajnos a kriptoanalitikusok is kevesebbet foglalkoznak velük. Ez egyben azt is jelenti, hogy a folyam titkosítók hátterében húzódó matematika kevésbé kidolgozott, mint ugyanez a blokk titkosítók esetében. A most megalkotott titkosítók így félig meddig ismeretlen terepen készülnek és ez sok kutató szemében jelent biztonsági kockázatot. Hasonlóan az AES felhíváshoz, Európában az eSTREAM projekt próbálkozott meg azzal, hogy kinevezze a legjobb folyam titkosítót egy verseny alapján. A felhívás során benevezett titkosítókat 2008-ig értékelték, majd 2012-ben egy második körös értékelés is történt. A benevezett titkosítók nagy figyelmet kaptak a kutatók részéről és az ott kialakult eredmények alapján a győztes titkosítókat ma már nagy biztonsággal használhatjuk.

A folyam titkosítók hibájaként szokták felróni, hogy használatuk több körültekintést igényel, mint a blokk titkosítók használata. Mi tagadás, valóban születtek olyan protokollok is, amelyeket megalkotásuknál tökéletesnek hittek, aztán később kiderült, hogy a folyam titkosító helytelen alkalmazása miatt a protokoll feltörhető. Ilyen protokoll a kezdeti WiFi titkosításért felelős WEP (Wired Equivalent Privacy) protokoll. A folyam titkosítók esetén például egy nagyon fontos szabály, hogy kétszer ugyanazt az álvéletlen bitsorozatot tilos felhasználni.

3.4.1 Az RC4 titkosító

Az 1987-ben megalkotott RC4 folyam titkosító [Rivest 1996] egy olyan kivételes titkosító, amely már a blokk titkosítók korában is használt volt és sok helyen a mai napokig is használatban van. Teljesítményét tekintve nagyon jó paramétereket tudott felmutatni, így nem véletlen, hogy 2000-ben a WiFi protokoll titkosításához is kiválasztották. Ebben a környezetben akkor még olyan eszközökről volt szó, amelyek kis számítási teljesítménnyel és kis memóriaterülettel rendelkeztek. Az RC4 protokoll tökéletesen illeszkedett ehhez a feladathoz. Végül azonban a mai WiFi protokollt már nem az RC4, hanem a már ismertetett AES protokoll védi. Ez köszönhető annak, hogy egyrészt a WiFi-t védő WEP protokollban súlyos hibákat találtak, másrészt a WiFi eszközök alkalmasak lettek komolyabb erőforrásokat igénylő algoritmusok befogadására is. A tervezők így inkább a biztos utat választották itt is. Az RC4 algoritmus azonban túlélte a WEP kudarcát és használatban maradt, de az alkalmazásánál már jobban odafigyeltek a potenciális problémákra. 2015-ben aztán végül megpecsételődött az algoritmus hivatalos felhasználásának sorsa, ugyanis az Internet biztonságával is foglakozó mérnökök a potenciális veszélyek miatt kitiltották

az algoritmust a titkosított kommunikációra használható algoritmusok sorából. A kitiltás azért érdekes, mert így ezzel egyetlen folyam titkosító sem maradt a használható algoritmusok között, holott volt már olyan időszak is, amikor az összes blokk titkosítót érintett biztonsági probléma és egyedül az RC4 nem volt érintve. Az ismert támadások hatására mára szinten teljesen eltűnt az RC4 a titkosított kapcsolatokból.

Az RC4 titkosítás, mint minden folyam titkosító, álvéletlen bitsorozatot állít elő egy kisebb, fix méretű kulcs segítségével. Az RC4 esetében a kulcs hossza minimum 40, maximum 2048 bit. A kulcsokat a biztonság tekintetében ekvivalensnek tekintjük a blokk titkosítók kulcshosszával, így itt is érvényes, hogy a 100 bit feletti kulcsokat tekinthetjük biztonságosnak. Az RC4 algoritmus két működési fázisból áll. Az első fázis a kulcstér kialakítása (KSA – Key-scheduling algorithm). Itt a bemeneti kulcsot keveri az algoritmus egy kezdeti titkosító állapotra. A belső állapot mérete 256 bájt, ezt a területet a processzornak a dinamikus memóriában kell tartania. Az RC4 legnagyobb hibája, hogy ez a fázis nem tökéletes. Több kutató is kimutatta, hogy a nem tökéletes állapot előállítása miatt a titkosító álvéletlen kimenetének első bitjei nem használhatóak teljes biztonsággal. Egyszerű, de nem túl hatékony megoldás az első bitek (akár az első 256 bit vagy még több) eldobása. Léteznek azonban RC4 variánsok is, amelyek ezt a lépést máshogy oldják meg, így biztonságosak. Az RC4 esetében az inicializálás 256 iterációból áll. Az iterációk során bájtonként permutálódik a belső állapot a kulcs által meghatározott pozíciókban. Az RC4 második művelete egy álvéletlenszám generáló algoritmus (PRGA – Pseudo-random generation algorithm), amely a már létrehozott belső állapot segítségével újabb és újabb álvéletlen biteket gyárt. A működés során a belső állapot mindig felülíródik egy újabb állapottal. Az algoritmus működése során minden egyes álvéletlen bájt előállítása során egy újabb permutáció keletkezik a belső állapotban.

Az RC4 algoritmus implementáció mindössze pár sor kód. Mivel a kulcs inicializálása hosszabb időt vesz igénybe, így inkább akkor működik jó teljesítménnyel, amikor nagyobb mennyiségű információ titkosításáról van szó. Ebben az esetben az sem gond, ha az első biteket el kell hagyni az álvéletlen sorozat előállítása során. Az RC4 algoritmust egyébként nemcsak mint titkosító algoritmust, hanem mint álvéletlenszám generátort is alkalmazzák. Bár ebben az esetben kevésbé merülnek fel biztonsági problémák, egyre több helyen ebben a pozícióban is lecserélésre kerül az algoritmus másik folyam titkosító vagy az AES blokk titkosító javára.

3.4.2 Salsa20/12 és ChaCha20 titkosítók

A Salsa20 titkosító egyike annak a 4 folyam titkosítónak, amelyet az eSTREAM projekt választott ki, mint biztonságos és hatékony folyam titkosító. A projekt lezárása után 2008-ban jelent meg a titkosító továbbfejlesztése a ChaCha variáns. Az új titkosító még jobb biztonságot és még hatékonyabb működést ígér. Az algoritmus előnyeit a Google is felismerte, és a már említett RC4 problémák miatt az RC4 algoritmust lecserélte a ChaCha algoritmusra. Chrome és Android böngészők már bizonyos oldalakat ezzel a titkosítóval látogatnak meg. Sőt, a ChaCha algoritmus elindult a szabványosítás útján, hogy ismét lehessen folyam titkosító az ajánlott internetes kapcsolat titkosítók listáján.

A titkosítás során az algoritmus a titkos kulcsot, egy tetszőleges publikus 64 bites értéket, egy sorszámot és 4 darab 32 bites meghatározott értéket egy 512 bites belső állapottá állít össze. A belső állapotot aztán több, a titkosító neve után található számú műveleti iteráció után egy 512 bites kimeneti értékké alakítja. Az eljárás nagyon emlékeztet egy blokktitkosító működésére, amely a CTR blokkösszefűzést használja. A különbség viszont

az, hogy a Salsa20 algoritmus nagyon egyszerű műveletekkel dolgozik: 32 bites összeadás, bitenkénti eltolás és XOR műveletek. Ennek köszönhetően a titkosító lényegesen gyorsabb, mint a blokk titkosítók. Az iteráció szám állításával a titkosító teljesítménye és biztonsága is skálázható. A szerző 20 iterációt javasol a tipikus kriptográfiai alkalmazások esetén, míg a limitált környezetekben a tökéletes biztonság feladásával elegendő lehet akár 8 iteráció is. Az eSTREAM projektben a 12-es iteráció szám versenyzett, amely egy középutat jelent a biztonság és teljesítmény között, azonban mint látható, ez az iteráció szám is biztonságosnak találtatott. Ma már ismert elméleti törés a 8 iteráció esetére, így az alkalmazása esetén érdemes ennél nagyobb értéket választani.

A ChaCha20 titkosító olyan processzorokon, amelyek nem támogatják az AES utasításkészletét, nagyjából háromszor gyorsabban fut, mint a vele ekvivalens erősségű AES blokk titkosító és modern GCM (Galois Counter Mode) blokkösszefűzés.

3.5 INTEGRITÁSVÉDELEM SZENZORHÁLÓZATOKHOZ

Az adatok sértetlenségét az integritásvédő algoritmusokkal lehet ellenőrizni. Különös szerepet akkor kap az integritásvédelem, amikor a titkosított információ elhagyja a titkosítás helyét és kommunikáció révén egy másik félhez érkezik. A másik félénél ugyanis ellenőrizni kell, hogy a megérkezett adatok vajon megegyeznek-e azokkal az adatokkal, amit a küldő fél szándékozott elküldeni. Amennyiben ugyanis eltérés lenne, úgy feltételezhetjük, hogy a kommunikáció támadás áldozata lett és a támadó megmásította az üzenetet. Természetesen előfordulhat az is, hogy a kommunikáció során az üzenet nem támadó, hanem valamilyen természetben előforduló hiba miatt változott meg, mint például egy rádiós adás esetében az áthallás által okozott hibák. Ezeket a szituációkat is felismeri az integritásvédelem, bár meg kell, hogy jegyezzük, hogy ilyen feladatra hatékonyabb algoritmusok is léteznek. A támadót nehezebb felismerni.

Az integritásvédelem megvalósításához *kriptográfiai kivonatképző algoritmusokat* használunk a klasszikus hálózatbiztonság területén. A kivonatképző algoritmusok tetszőleges méretű információt sűrítenek egy fix hosszúságú kivonatba, amely aztán az adatokkal együtt, de mindenképpen biztonságosan utazik a másik félhez. A másik fél szintén elvégzi a kivonatképzést és összehasonlítja a kapott értéket a küldött értékkel. Amennyiben nincs eltérés, úgy az üzenet integritása rendben van. A jól működő kivonatképző algoritmus akár gigabájt méretű adatokból képes 16 bájtos kivonatokat képezni úgy, hogy a gyakorlatban nem fogunk ütközést találni a kivont értékében, azaz más tetszőlegesen nagy méretű adatnak sem lesz ugyanez a kivonata. A kriptográfiai kivonatképző algoritmusok pedig ezt a tevékenységet olyan jól csinálják, hogy egy támadó szándékosan sem tud olyan adathalmazt készíteni, ami egy másik adattal egyező kivonatot mutat. A kivonatképzéshez nem feltétlenül van szükség titkos kulcsra, azonban, ha a kivonatot nyilvános csatornán szeretnénk átküldeni, akkor vagy szükséges annak titkosítás vagy érdemes a kivonatkészítés során kulcsot is használni. Ebben az esetben ugyanis a támadó képtelen lesz a megváltoztatott üzenethez igazítani az integritásvédő kódot.

A kriptográfiai kivonatképző algoritmusok előnyös tulajdonsága, hogy teljesítményük nagyon jó, bár működésükhöz egy nagyobb memóriaterületre van szükségük, de akár átalakítás nélkül is használhatóak szenzorhálózatokban. A hálózatbiztonság területén ma még mindig népszerű az SHA-1 (Secure Hash Standard 1) algoritmus, holott ennek az algoritmusnak a biztonságát ma már többen megkérdőjelezik. Az algoritmus szabvány, ami miatt széleskörűen használják. Működését tekintve iteratív hash, amely egy egyszerű lépésekből álló művelettel vonul végig a bemeneti információn, miközben az új adatok a belső állapotot változtatják meg.

Az SHA-1 algoritmus korábbi elméleti támadásaiból mára már gyakorlatban is alkalmazható támadások születtek. 2017-ben az amszterdami Centrum Wiskunde & Informatica és a Google mutatott be közösen két olyan pdf dokumentumot, amelynek ugyanaz az SHA-1 lenyomata. Bár a dokumentum létrehozásánál 6500 évnyi CPU időt használtak fel, mégis a támadást már kivitelezhetőnek kell tekinteni. A korábbi elméletekre alapozva a böngészők ma már nem fogadják el az SHA-1 algoritmust a biztonsági protokollok működése során. Az SHA-1 algoritmusnak létezett továbbfejlesztése is, az SHA-2 algoritmus. Mivel ez ugyanazt a belső működési elvet használja, mint az SHA-1, ma már ennek az algoritmusnak a használata sem javasolt. Már 2006-ban az NIST (USA National Institute of Standards and Technology) felhívást írt ki, hogy hasonlóan az AES titkosító pályázatához, a kriptográfiai kivonatképzőket is megversenyeztesse. Már akkor elvárás volt, hogy az új algoritmusnak merőben eltérő alapokra kell épülnie, mint az SHA-1 vagy SHA-2 algoritmusok. A verseny győztesét 2015-ben jelentették be, ez a Keccak algoritmus lett. Érdekesség, hogy a szerzők között megtaláljuk az AES szerzőpáros egyik tagját is.

Az SHA-3 algoritmus jelenleg kielégít minden biztonsági kívánságot, amit egy kriptográfiai kivonatképző algoritmustól elvárhatunk. Problémájának azt róják fel, hogy bár a hardveres implementáció gyorsabb, mint elődei, addig a szoftveres implementáció fele olyan sebességű csak, mint az SHA-2 és harmad olyan sebességű, mint az SHA-1. Összehasonlításképpen egy modern, 2.7 GHz órajelű Intel i5 processzoron az SHA-1 540 MB/s sebességgel, az SHA-2, 512 bites kimenettel 377 MB/s sebességgel, míg az SHA-3 szintén 512 bites kimenettel csupán 141 MB/s sebességgel fut.

Az említett algoritmusoknál tehát nincsenek sebesség problémák, általában gyors és egyszerű felépítésű algoritmusokról van szó, amelyek ráadásul a memóriában is elférnek. A belső állapotuk nagy ugyan, de nem használnak helyettesítő táblákat. A szenzorhálózatos alkalmazásuk során a problémát inkább az jelenti, hogy az előálló kivonat túlságosan nagy méretű a szenzorhálózatokra jellemző méretekhez képest. Az SHA-1 algoritmus 20 bájt hosszúságú kivonatot gyártott. Ez a méret túlságosan nagy egy olyan hálózatban, ahol van, hogy a maximális csomagméret mindössze 32 bájt. Ráadásul az SHA-1 ma már a múltat képviseli és helyette az SHA-2 és SHA-3 algoritmusokat kell használni, ahol a kivonat mérete legalább 28 bájt. Ebben az esetben egy előbb említett rádiós modul használata esetén nem is marad hely hasznos adat átvitelére. Szerencsére azonban a legtöbb kivonatképző algoritmus esetében az előálló kódot tetszőlegesen lehet csonkolni a biztonság rovására. Így alacsonyabb biztonsági szinten, de elérhető, hogy az alkalmazott szenzorhálózati protokollhoz illeszkedően, rövidebb integritásvédő kódot alkalmazzunk. A csökkent biztonsági szintet majd máshol kell kompenzálni, ha szükséges.

3.6 INTEGRÁLT INTEGRITÁSVÉDELEM ÉS TITKOSÍTÁS

A titkosítás és az integritásvédelem két külön funkció és az előbbiekben bemutatott eljárások merőben mások is voltak a feladatok megvalósításához. Azonban éppen a korlátozott erőforrásokkal bíró szenzorhálózatok esetében érdemes arról beszélni, hogyan lehetne a két feladatot összevonni ahhoz, hogy időt, energiát vagy memóriahelyet spóroljunk. Az alapvető és a klasszikus hálózatbiztonságban alkalmazott megközelítés ugyanis az, hogy a titkosító egyszer végighalad az összes adaton és titkosítja azt, majd az integritásvédelem végighalad a már titkosított adatokon és kiszámolja az integritásvédő kódot. Az információn tehát kétszer haladunk át, ez a két eljárás egyesítésével egyetlen feldolgozásra csökkenne.

A megoldást a blokkösszefűzés megváltoztatása jelenti. Az OCB (Offset Codebook Mode) [Rogaway 2003] blokkösszefűzést éppen az ilyen helyzetekhez találták ki. Ráadásul még további kedvező tulajdonsága is van a szenzorhálózatos alkalmazás elősegítésére. A titkosítás közben ugyanis kijelölhetünk olyan adatrészeket is, amelyeket nem szeretnénk titkosítani, csupán integritásvédelmet akarunk biztosítani. A hálózati kommunikáció esetén ugyanis gyakran tartalmaznak az egyes adatcsomagok olyan részt is, amely nem a hasznos teherhez tartozik, hanem azért szükséges, hogy a megfelelő célponthoz jusson el az adatcsomag. A legegyszerűbb ilyen példa a célpont hálózati címe, de léteznek más, az irányítást segítő információk is. Ezeket az információkat nem lehet titkosítani, vagy legalábbis csoport szinten kell azt megtenni, különben a továbbításban résztvevő csomópontok nem tudják, mit kezdjenek az adott adatcsomaggal. Az OCB további előnye, hogy az utolsó blokkot is meg lehet hagyni a maga hosszában, nincs szükség a kitöltésre sem. Így az átvitt bájtok számán is lehet spórolni.

Az OCB működéséhez természetesen egy blokk titkosítóra van szükség, hiszen a blokkok összefűzése csak ezeknél a titkosítóknál merül fel. Az OCB jelenleg használt verzióját 2011-ben alkották meg, ez lett az algoritmus harmadik változata, egyben a legtöbb funkcióval és a legjobb teljesítménnyel. A megoldás sikerét az is mutatja, hogy ezt a blokkösszefűzési módot a WiFi szabvány is megjelölte, mint alternatíva a jelenleg alkalmazott módszerek mellé, bár onnan végül kikerült, mert az OCB csak bizonyos esetekben mentes a licenszeléstől.

Teljesítményben az OCB sebessége a CBC blokkösszefűzési technológiával egy szinten van. Ez azért nagyon jó eredmény, mert míg a CBC mód csak titkosítást végez, addig az OCB esetében már megtörténik az integritásvédelem is. Megfelelő paraméterek használata esetén az OCB titkosító 1.6 százalékkal több titkosítási műveletet használ, mint amennyit az titkosítandó információ hossza indokolna, függetlenül a tényleges hossztól. A blokk titkosításán túl blokkonként még 3 darab XOR művelet, amit a módszer pluszban végez. Nem csak a teljesítmény területén, hanem a memóriahasználat esetében is remekel ez a blokkösszefűzési mód. Az implementációban a legtöbb memóriaigénnyel az alkalmazott blokk titkosítót jelentkezik és az OCB paraméterei és kódja csak minimális tárhelyet igényelnek.

Az OCB természetesen nem az egyetlen olyan megoldás, amely egyetlen műveletben ötvözi a hitelesítést és a titkosítást. Hasonlóan a korábban bemutatott AES, SAH-3 és Salsa20 kriptográfiai algoritmusokhoz, ezen a területen is kiírtak 2014-ben egy pályázatot, amelyben keresik a legjobb integrált integritásvédő titkosítót (AE - Authenticated Encryption). A CAESAR pályázatra nagyon sok kutató jelentkezett titkosító algoritmusával. A kiértékelés 2017 végén zárult le, jelenleg 7 algoritmus szerepel a döntősök között, köztük az OCB algoritmus is.

Létezik integrált titkosító és integritásvédő megoldás folyam titkosítás esetén is. Egy ilyen a ChaCha20-Poly1305 [Procter 2014] megoldás, amely a ChaCha20 folyam titkosítót ruházza fel integritásvéső képességekkel. Azonban itt nem egészen arról van szó, mint az OCB esetében. A művelet ugyan egyetlen kulcsot használ és van valamennyi köze a két algoritmusnak egymáshoz, de valójában itt azért két külön algoritmusról van szó, jelen esetben a ChaCha20 titkosítóról és a Poly1305 integritásvédő algoritmusról.

3.7 ASZIMMETRIKUS KULCSOK HASZNÁLATA

Az eddig ismertetett algoritmusok mind szimmetrikus kulcsú algoritmusok voltak, ami azt jelenti, hogy amennyiben egy algoritmus titkos kulcsot használ egy adott művelethez, akkor a művelet inverzéhez pont ugyanazt a kulcsot kell használnia. Ez gondot csak akkor jelent, amikor nem ugyanaz a személy és eszköz végzi el a művelet inverzét, ugyanis ebben az esetben a másik fél számára is biztosítani kell a kulcsot. A kulcs megosztása alapvetően

két problémát vet fel. Az egyik a jogosultsági probléma, azaz megosztható-e a kulcs a másik féllel, hiszen például egy integritásvédő kulcs megosztása után már a másik fél is tud majd integritásvédő kódokat gyártani, akár hamis üzenetekhez is. A másik technikai probléma az, hogy hogyan juttatható el a kulcs biztonságosan a másik félhez. Itt azért nem annyira egyszerű a megoldás, hogy titkosítsuk a kulcsot, hiszen ahhoz a titkosításhoz is egy újabb megosztandó kulcs kellene.

A hálózatbiztonság témakörben nagyon sok kulcsegyeztető protokollt találunk, amelyek a legkülönfélébb helyzetekben nyújtanak megoldást, hogy a közös kulcsot biztonságosan eljuttassuk a másik fél számára. Lényegesen egyszerűbb módszer lenne azonban, ha a kulcsot nem kellene megosztani a másik féllel, hanem mindenki a saját kulcsával tudna dolgozni. Bár dolgoznak rajta, de ezt még egyelőre nem lehet elérni, az azonban már működőképes, hogy a felhasználóknak két kulcsuk van, az egyik a saját használatra szánt privát kulcs, a másik pedig egy szabadon és nyilvánosan megosztható publikus kulcs. Ez a kulcspár az aszimmetrikus kulcs és vannak jól működő algoritmusok, amelyek biztosítják a titkosítást a kulcsok használatával. A kommunikáció során tehát az egyik fél a kulcs egyik felét használja a titkosításhoz, míg a másik fél már a másik kulcsot fogja használni a titkosítás feloldásához. Mivel a publikus kulcs mindenki számára ismert, ezért ebben a helyzetben az üzenetküldő a másik fél jól ismert publikus kulcsával titkosítja az üzenetet, amit a címzett majd a saját privát kulcsával tud feloldani. Az algoritmus biztosítja, hogy senki más, aki nem ismeri a privát kulcsot, ne tudjon hozzáférni az üzenet tartalmához. A jó aszimmetrikus kulcsú titkosító esetében a publikus-privát kulcsok ugyan párban vannak, de a publikus kulcs ismeretében még nem lehet a privát kulcsot kitalálni.

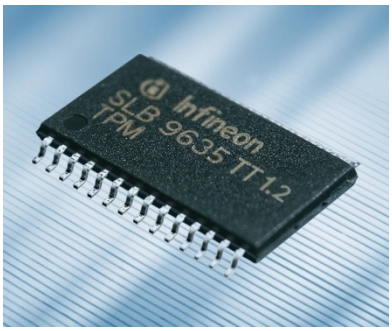
A legelterjedtebb aszimmetrikus kulcsú titkosító algoritmus az 1977-ben megalkotott RSA algoritmus [Rivest 1983], amely a feltalálóiról (Ron Rivest, Adi Shamir és Leonard Adleman) kapta a nevét. Az algoritmus működése viszonylag egyszerű, a titkosítást tulajdonképpen egy modulo hatványozás művelet jelenti. A hatványozás alapja az üzenet maga, míg a publikus és privát kulcs a kitevő. A kulcsok speciálisan szerkesztettek és egy alapvető matematikai problémára, a prímtényezős felbontás algoritmusának nem létezésének feltételezésére épülnek. Két hatalmas prímszámból gyúrnak össze egy olyan számot, amelyből származtatni lehet a publikus és privát kulcsot, úgy, hogy ezek a kulcsok teljesítik azt a követelményt, miszerint egy tetszőleges számot az egyik majd másik kulcsra, mint hatványra emelve ugyanazt a számot kapjuk vissza a nagy prímszám modulo mellett. Az RSA eljárás így már régóta alapját képezi az internetes kommunikációnak. Aki ma titkosított adatkapcsolatot használ, jó eséllyel belefut valahol az RSA használatába is. Nem csak a titkosítás területén, hanem a hitelesség ellenőrzésére is használhatjuk. A privát kulcs használata ugyanis bebizonyíthatja, hogy az azt használó fél valóban ismeri a kulcsot és ezt csak a tulajdonosáról feltételezzük. Az ismeret tényét a publikus kulcs segítségével lehet egyszerűen ellenőrizni. Ez esetben a hitelesítendő fél a privát kulcsával titkosít egy a másik fél számára is ismert szöveget, majd a hitelesítő fél a hitelesítendő fél publikus kulcsával ellenőrzi a titkosítás helyességét.

Az RSA algoritmus szenzorhálózatokban történő alkalmazása sajnos nem egyértelmű. Mint írtuk óriási számokkal operálunk és a művelet ugyan egyszerű, de annak elvégzése rendkívül időigényes. Az RSA működéséhez nagy prímszámok kellenek, itt tipikusan 1024 – 2048 – 4096 bitről beszélünk. Látható, hogy a kulcs méretek jelentősen eltérnek a szimmetrikus kulcsok méretétől. Az 1024 bites aszimmetrikus RSA kulcs egy 80 bites szimmetrikus kulcsnak felel meg, így ma már ezt nem tekintjük biztonságosnak. A 2048 bites RSA kulcs a 112 bites kulcs megfelelője, míg a szimmetrikus kulcsú titkosítóknál alkalmazott 128 bites kulcshossz 3072 bitnek felel meg az RSA világában. A probléma ráadásul az is, hogy ahogy nő a számítógépek feldolgozási teljesítménye, úgy folyamatosan emelni is kell ezeket a kulcsméreteket, tartva a kulcsfeltörésektől, a növekedés pedig exponenciális.

A jövőben tehát fel kell készülnünk a több tízezer bites RSA kulcsok kezelésére is. A tárolás az egyik probléma, hiszen a privát kulcsot csak és kizárólag az adott eszköz ismerheti, így olyan tárolást kell biztosítani, amelyből a privát kulcs értékét nem lehet kiolvasni.

A másik probléma a nagy számmal az, hogy ez fog szerepelni titkosítás és a titkosítás eltávolítása során a hatványkitevőben. Egy amúgy is nagy számot ennyiszer kell majd önmagával megszorozni és ennyi modulo műveletet kell majd végezni. Ennek a problémának a feloldását némiképpen segíti, hogy a kulcspár gyártása során az egyik kulcsot szabadon meg lehet választani. A publikus kulcsot így a viszonylag nem túl nagy 65537 értékre szokták beállítani, így elkerülve azt, hogy túl sok hatványozást kelljen végezni az egyik művelet során. Egy mikrokontroller számára azonban a 2048 bit feletti hozzáférés védett tárhely, illetve a nagy számok hatványozása így is rendkívül nehéz, sőt teljesíthetetlen feladat. Egy miniPC jellegű szenzor természetesen, még ha lassan is, de el tud boldogulni a feladattal, de egy ma, szenzorokban alkalmazott átlagos mikrokontroller nem boldogul önmagában ekkora feladattal. A megoldás az, hogy a feladatot egy dedikáltan kriptográfiával foglalkozó külső egységre kell bízni.

A TPM (Trusted Platfrom Module) modulok olyan kiegészítést jelentenek a mikrokontrollerek számára, amelyben megtalálható mind a privát kulcsok tárolására szolgáló biztonságos tárhely, mind a szükséges teljesítmény ahhoz, hogy az aszimmetrikus kulcsú kriptográfiai műveleteket elvégezze a külön egység (3.3 ábra). Általában nem csak az RSA algoritmust implementálják, hanem más szimmetrikus kulcsú titkosítók, integritásvédő algoritmusok, mint például az AES titkosító és az SHA-1 is helyet kap a modulban. A TPM modul különböző védelmi mechanizmusokkal is rendelkezhet, így nem csak a kommunikációt, hanem az indítási folyamatot és a szoftverfrissítést is biztonságossá teheti.



3.3. ábra. Az Infineon TPM-modulja

Forrás: a kép a gyártó oldalán publikusan elérhető

Az RSA mellett léteznek ma már alternatívák is. Széles körben elterjedt algoritmus az elliptikus görbékre épülő kriptográfia (Elliptic Curve Crytography, ECC) [Hankerson 2006], amely szintén képes aszimmetrikus kulcsú titkosítást biztosítani. A megoldás előnye, hogy a kulcsméret és sebesség jóval kedvezőbb az RSA megoldásához képest. Az ECC esetében jelenleg a 160 bites kulcsok is biztonságos megoldást nyújtanak, illetve a jövőben sem kell 512 bitnél nagyobb kulcsokra számítani. Bizonyos elliptikus görbére épülő algoritmusok így már kis teljesítményű mikrokontrollereken is implementálhatóak. Az ECC algoritmus már bekerült a legújabb 2.0 verziószámú TPM specifikációba is.

Szintén megoldás lehet, ha a szenzorhálózatokban kikerülik az aszimmetrikus kulcsú titkosítást és helyette biztonságos kulcsmegosztó protokollokat használnak. Ezek a kulcsmegosztó protokollok gyakran szintén meg-

lehetősen számításigényesek, azonban legalább a titkos kulcs kezelésével kevesebb a probléma. Ennek már csak azért is jelentősége van, mert a kvantumszámítógépek feltételezett terjedésével sok aszimmetrikus kulcsú algoritmus, többek között az RSA és az ECC is feltörhetővé válik a távolabbi jövőben. A szimmetrikus kulcsú algoritmusok nem érintettek ennyire súlyosan, ott várhatóan a kulcs méretének megnövelése egyszerűen megoldja a problémát, így az arra épülő megoldások egyben jövőbiztosak is.

3.8 SZENZORHÁLÓZATOKBAN HASZNÁLHATÓ KRIPTOGRÁFIAI ALGORITMUSOK ÖSSZEFOGLALÁSA

A fent bemutatott algoritmusok az RSA algoritmus kivételével mind használhatóak szenzorhálózatokban is, sőt végül az RSA használata is megoldható egy erre a feladatra dedikált kiegészítő modullal. Elmondható azonban, hogy a szenzorhálózatok szenzorainak kis teljesítményéhez és korlátos tárhelyéhez alkalmazkodva érdemes a modernebb eljárásokat választani.

A titkosítási feladatokhoz akkor érdemes az AES algoritmust választani, ha az elkészített rendszernek kompatibilisnek kell lennie valamilyen más eszközön futó szabványos megoldásokat használó szoftverrel. Bár szenzorhálózatokban nem gyakori, de szintén előnyös ennek a titkosítónak a választása, ha a felhasznált mikrokontroller rendelkezik az AES műveletek elvégzéséhez speciálisan kialakított utasításkészlettel vagy rendelkezik olyan modullal, amely képes ezt a feladatot ellátni. Ez utóbbi modul vagy egy TPM modul, amely kötelezően tartalmazza az AES algoritmus implementációját vagy lehet akár a rádiós modul is. Amennyiben az AES implementálása a szenzor mikrokontrollerén történik, érdemes olyan implementációt választani, amely jól illik a feladathoz. Vannak implementációk, amelyek lassabbak, de kevesebb memóriát foglalnak, és vannak implementációk, amelyek gyorsabbak, de több helyen férnek csak el.

Amikor nem cél az AES kompatibilitás, akkor tervezői döntés lehet egy az AES-nél gyorsabb és kisebb memória méretű, de kevésbé biztonságosabb megoldás is. Ilyen megoldások az ismertetett Twofish, XTEA, XXTEA, Skipjack és HIGHT algoritmusok. A blokk titkosítók esetében, kivétel az XXTEA titkosító, az információ blokkokba szervezésével is foglalkozni kell. Erre a feladatra ismét több megoldás lehetséges. Amennyiben szintén cél, hogy egy más szabványos rendszerrel kompatibilis legyen a rendszerünk, feltételezve, hogy a másik rendszer nem implementálja az újdonságokat, úgy a CBC vagy CTR blokkösszefűzési módokat kell választani az AES titkosító mellé, illetve ha egyben integritásvédelmet is szeretnénk, akkor pedig a GCM blokkösszefűzési módot. Amennyiben a tervező eltérhet a szabványos és széles körben elterjedt megoldásoktól, úgy érdemes az OCB blokkösszefűzési módot választani, amely jobb teljesítménnyel nyújtja az integrált hitelesítést, mint a korábban felsorolt megoldások. Az OCB mellé természetesen nem feltétlenül kell az AES algoritmust választani. Alternatívaként felmerülhet, hogy nem blokktitkosítót használunk. Itt az RC4 titkosító ma már nem igazán jó döntés, várhatóan a közeljövőben el fog tűnni minden alkalmazása. Helyette a ChaCha folyam titkosító tűnik jó választásnak és úgy tűnik a közeljövőben egyre több helyen implementálják majd. A ChaCha szenzorhálózati csomópontokon is jól implementálható, tárigénye viszonylag kicsi, sebessége gyors és nem is igényli a blokkok kezelését. Itt azonban szükség lesz egy integritásvédő kódra is, amelyet ugyan ma még az SHA-1 algoritmus biztosít, de célszerű ezt már elkerülni és helyette az SHA-3 algoritmust választani. Mindkét említett algoritmus jól alkalmazható mikrokontrollerek esetében is. A ChaCha esetében jobb választás a Poly1305 algoritmus.

A következő táblázat megmutatja, hogy bizonyos összetett megoldások esetén milyen sebességre számíthatunk egy átlagos mikrokontrollertől. Jelen esetben egy hobbi célra szánt 16 MHz órajelű Arduino UNO egységet választottunk, amelynek 32 KB programtárolója és 2KB dinamikus memóriája van. Mellette megtalálható egy másik teszt, amelyben pedig egy ennél komolyabb, Cortex-M1 architektúrájú 84 MHz órajelű Arduino Due mikrokontrollert mértünk. A tesztekben különböző üzenet hosszok mellett látható a titkosításra és titkosítás feloldására fordított idő. Ugyancsak megjelenítettük a művelet közben felhasznált dinamikus memóriaterület nagyságát is.

3.1. táblázat. Titkosítók és integritásvédők teljesítménye Arduino UNO esetén

Tesztelt eljárás	32 bájtos üzenet titkosítás/feloldás		200 bájtos üzenet titkosítás/feloldás		Memória-foglalás
AES 128 csak titkosítás	*1.22 ms*	*2.18 ms*	*6.81 ms*	*12.79 ms*	*181 bájt*
ChaCha20 csak titkosítás	*0.52 ms*	*0.52 ms*	*3.02 ms*	*3.02 ms*	*132 bájt*
GCM-AES128	*4.78 ms*	*4.76 ms*	*23.21 ms*	*23.21 ms*	*285 bájt*
ChaChaPoly	*2.22 ms*	*2.22 ms*	*9.14 ms*	*9.14 ms*	*221 bájt*

3.2. táblázat – Titkosítók és integritásvédők teljesítménye Arduino Due esetén

Tesztelt eljárás	32 bájtos üzenet titkosítás/feloldás		200 bájtos üzenet titkosítás/feloldás		Memória-foglalás
AES 128 csak titkosítás	*0.25 ms*	*0.4 ms*	*1.35 ms*	*2.32 ms*	*188 bájt*
ChaCha20 csak titkosítás	*0.03 ms*	*0.03 ms*	*0.18 ms*	*0.18 ms*	*136 bájt*
GCM-AES128	*0.6 ms*	*0.6 ms*	*2.43 ms*	*2.43 ms*	*312 bájt*
ChaChaPoly	*0.1 ms*	*0.1 ms*	*0.39 ms*	*0.39 ms*	*240 bájt*

Természetesen a tesztelt mikrokontrollerek nem támogatják az AES algoritmust segítő utasításokat, mint ahogyan a többi algoritmust sem. A tesztek is azt igazolják, hogy érdemes elmozdulni az új titkosítók irányába a szenzorhálózati elemek esetében, hiszen szignifikáns a teljesítmény növekedés, amely ugyanannál a feladatnál kevesebb energiát is jelent, illetve a memóriafoglalás is kisebb.

4. SZENZORHÁLÓZATI KOMMUNIKÁCIÓ TÁMADÁSA

Az okosváros alkalmazások támadásaink felsorolásánál a támadások jókora része irányult a szenzorhálózatokban átvitt adatok ellen. Amennyiben a támadó meghamisíthatja az átvitt adatokat, úgy több alkalmazás esetében is akár közvetlen anyagi haszonra tehet szert, éppen ezért az adathamisítás valós fenyegetés. A szenzorhálózatnak védenie kell az átvitt adatokat.

Bizonyos szenzorhálózatok esetében azonban külön nehézséget jelent, hogy a rádiós kommunikáció energiaigényének leszorítása miatt rövid hatótávolságú rádiókat alkalmaznak, és a távolabbi pontok elérése végett a rádiós csomópontok továbbítják egymás adatait. A szenzorhálózat esetében ez a többugrásos adatátvitel. Ebben a helyzetben a globális szinten elért energiafelhasználás csökken ahhoz képest, mintha minden egyes adó saját maga érné el a nyelőt. Sok esetben egyébként nem csak gazdasági indokok vannak a többugrásos adatátvitel mellett, hanem a rádiós viszonyok olyanok, hogy lehetetlen a közvetlen kapcsolat. A többugrásos kommunikáció ugyanakkor hatékonysága mellett sajnos lehetőséget teremt arra is, hogy a támadó, amennyiben átveszi egy adott csomópont felett az irányítást, az azon áthaladó adatforgalmat befolyásolja. Sőt, mint azt majd látni fogjuk, ebben a környezetben a támadó akár lényegesen többet is elérhet, befolyásolva olyan csomópontok működését is, amelyeknek forgalma eredetileg nem volt érintett a támadó által.

A kriptoanalízis módszertanában a támadások és fenyegetések kiértékelésénél úgy kell tekintenünk, mintha a támadónak minden eszköz rendelkezésére állna, amit a mai technológia elérhetővé tesz. Tulajdonképpen a támadó egy feltételezett gonosz szupererő, aki feltételezésünk szerint a következő szuper képességekkel rendelkezik:

- Képes egyszerre jelen lenni a hálózat több pontján is, ott hallgatózni és akár bele is avatkozni a csomópontok forgalmába;
- Képes a saját, támadásokhoz kapcsolható forgalmát a szenzorhálózatnál gyorsabban továbbítani, akár a szenzorhálózaton kívül is;
- Képes a támadás során előállított vagy tárolt adatok elküldésére tetszőleges pontra, valamint a hálózatban átvitt adatok nyomtalan eltűntetésére is;
- Képes a szenzorhálózat csomópontjainak feltörésére és az ott található információk megszerzésére, amennyiben azok nem megfelelően védettek;
- Képes az itt felsorolt műveleteket láthatatlanul, önmagát elrejtve megtenni.

Tulajdonképpen ezek a feltételezések nem szakadnak el a valóságtól, amennyiben feltesszük, hogy a támadónak jelentős anyagi forrása van a támadás kivitelezésére. A fent felsorolt műveletek ugyanis valóban megvalósíthatók, csupán egyelőre annyira költségesek, hogy azt nem mindenki tudja megengedni magának. Fontos, hogy semmi olyan képességgel nem ruházzuk fel a támadót, amely a tudomány mai állása szerint lehetetlen lenne. Sőt egyelőre a terület szakirodalma sem foglalkozik olyan, jelenleg távolinak gondolt támadási képességekkel, mint például a kvantumszámítógépek majdani megjelenésével véghez vihető kriptográfiai algoritmusokat érintő támadások [Gisin 2002]. Minden számba vett támadás tehát valóban valós fenyegetés egyben.

A szenzorhálózati kommunikáció védelmének a már felsorolt támadásokra kell választ adnia a most ismertetett támadóval szemben. Mielőtt azonban párosítanánk a támadásokat és a védekezéseket, bemutatjuk azt, hogy a mai szenzorhálózatokban mit érdemes és egyáltalán mit tudunk védeni. Bár a szenzorhálózatok az alkalmazásoktól függően sokfélék és így sokféle dolgot értünk a biztonság kifejezés alatt is – gondolva itt például egy adatgyűjtő okosmérő és egy emberek egészségügyi állapotának megfigyelésére telepített szenzorhálózat különbségére – mindazonáltal mégis vannak közös jellemzők.

4.1.1 Kommunikáció bizalmasság (confidentiality) a szenzorhálózatokban

A bizalmas kommunikáció kifejezésen azt értjük, hogy a kommunikáció során átvitt üzenetek nem kerülhetnek illetéktelen kezekbe. A többugrásos vezeték nélküli szenzorhálózatok esetében azonban ez egy kimondottan nehéz feladat, hiszen ebben az esetben az üzenetek útja nem közvetlenül a szenzor és a gyűjtő között van, hanem más csomópontok is besegítenek. Nem működhetnek tehát az olyan megoldások, amelyek csak az adott pont-pont fizikai kapcsolat védelmére alapoznak, hiszen nem feltétlenül létezik a végpontok között a közvetlen fizikai kapcsolat.

Az üzenetek bizalmasságának megőrzése nem korlátozódhat kizárólag a szenzor által mért információkat tartalmazó üzenetekre. A hálózatban a kommunikáció megfelelő kialakításához is szükség van adatcserére, legtöbbször vezérlő adatok és üzenetek formájában. A vezérlő adatok védelme kiemelten fontos a kommunikációs képesség megőrzése érdekében.

A kommunikációs üzenetek bizalmasságának megőrzésére a szenzorhálózatok esetében is kriptográfiai titkosító algoritmusokat használunk. Mint azt a kriptográfiai részek ismertetésénél elmondtuk, a szenzorhálózatokban nagyon nehéz publikus és privát kulcsokon alapuló kriptográfiát alkalmazni, így a titkosításon túl szükségünk lesz olyan protokollokra is, amelyek megteremtik azt az állapotot, hogy a kommunikációban résztvevő összes fél ismeri a megfelelő titkosító kulcsot.

4.1.2 Kommunikáció integritásvédelme (integrity) a szenzorhálózatokban

A kommunikációs üzenetek titkosítása révén nyújtott bizalmasság biztosíthatja, hogy az üzenetek tartalma ne legyen elérhető nem kívánt felek számára. Önmagában azonban egyetlen titkosító algoritmus sem véd meg attól, hogy a támadó módosítsa a már titkosított üzeneteket. A titkosítás hatására, amennyiben a titkosítás valóban jól működik, a támadó így természetesen továbbra sem fér az ott közölt adatokhoz, de a titkosított üzenet tartalma a módosítás hatására kiszámíthatatlanul megváltozhat. Az üzenetek integritásának védelme pont azt a célt szolgálja, hogy az üzenetekben szállított információ módosítását a vevő fél észlelhesse, még mielőtt az információt felhasználná. Magát a megváltoztatást egyetlen algoritmus sem tudja megakadályozni, hiszen az a kommunikáló csomópontok hatáskörén kívül történik, az azonban elkerülhető, hogy meghamisított adat kerüljön feldolgozásra. Amennyiben az üzenet útja több csomóponton keresztül vezet, úgy a köztes csomópont felismerve a hamisított üzenetet, akár már a szállítás közben jelezheti a hamisítás tényét a küldő csomópont számára.

Az üzenetek integritásának védelmére is kriptográfiai algoritmusokat használunk, akárcsak a klasszikus információbiztonság esetében. A megoldás a már ott ismertetett módon zajlik, azaz a küldő fél az üzenet mellé egy

olyan kódot is előállít, amely segítségével ellenőrizhető annak épsége. A kód előállításához egy kulcs szükséges és feltételezzük, hogy ez a kulcs csak a küldőnél és a fogadónál van jelen. Az üzenet címzettje, amikor megkapja az üzenetet, ő maga is előállítja az üzenetet védő kódot ugyanazzal a módszerrel és kulccsal, amivel a küldő tette. Amennyiben a számítás eredményeként kapott kód és az üzenettel együtt érkezett kód megegyezik, úgy az üzenetet valódinak lehet elfogadni. Amennyiben eltérés mutatkozna, úgy az üzenet tartalmát vagy a támadó vagy valami természeti jelenség megváltoztatta. Az integritásvédő algoritmusok tárgyalásánál már írtuk, hogy sokszor találkozunk titkosítóval egybeintegrált integritásvédő algoritmussal. Ezek működése az egyes csomópontokban ugyan eltér a külön titkosítástól és integritásvédelemtől, azonban az üzenet szintjén ez már nem látszik. Minden esetben a titkosított üzenetet és az integritásvédő kódot látjuk. Ahogy azt szintén írtuk már a védelem esetében a kihívást nem is igazán az algoritmus erőforrás igénye jelenti, hanem az, hogy az integritásvédő kód általában túlságosan nagy az üzenet méretéhez képest. Ezt csonkolással lehet kezelni, így azonban sérül a biztonság is. Gyakran alkalmaznak ilyen esetekben proaktív védelmet. Ilyen a WiFi TKIP (Temporal Key Integrity Protocol) esetében alkalmazott megoldás is. A részletesebb ismertetés nélkül a megoldás úgy működik, hogy amennyiben a kommunikációs felek úgy érzik, hogy az üzenetek átvitelében keletkező hibákat nem természetes jelenségek, hanem egy támadó okozza, úgy azonnal megváltoztatják a védekezésre használt kulcsokat és így a támadó próbálkozása hiábavaló lesz. A természetes hibák kiszűrésére felesleges kriptográfia integritásvédő kódokat használni, ugyanis itt nem feltételezünk intelligenciát a változások mögött. A véletlenszerű változások kiszűrésére CRC (Cyclic Redundancy Code) [Koopman 2004] kódokat használhatunk. A legtöbb rádiós kommunikációs protokoll alapból támogatja ezeknek a kódoknak a használatát, sőt terjedelmük általában nem megy a felhasználható üzenethossz méretének rovására.

4.1.3 Kommunikáció hitelesítés (authentication) a szenzorhálózatokban

Hiába biztosított az üzenetekben küldött adatok bizalmassága és hiába tudhatjuk azt is, hogy a gyűjtőhöz beérkező üzenetek valóban egy az egyben azok, amiket a küldő elküldött, még mindig nem lehetünk biztosak abban, hogy a küldő fél valóban az, akitől az üzenetekben szereplő adatnak származnia kell. A titkosítás és integritásvédelem tökéletes működése ellenére lehetséges, hogy a küldőt megszemélyesítik és így a védelem jó, csak éppen a védett adatok hamisak. A vezeték nélküli szenzorhálózatban résztvevő csomópontokat így hitelesíteni is kell, hogy tudjuk, valóban az az adatforrás, amit az adott információ feltűntet. Fontos, hogy nem kimondottan az adatok hitelesítéséről beszélünk, hanem most a csomópontokat hitelesítjük. Lesz majd olyan alkalmazásunk is, amikor viszont pont az adatokat kell hitelesíteni.

Természetesen itt is kriptográfiai algoritmusokat fogunk használni, hiszen a szenzorhálózat fizikai kiterjedése és a szenzorok sokasága nem igazán tesz lehetővé más megoldást. A klasszikus információbiztonságban erre a célra az aszimmetrikus titkosítók illetve digitális aláírók által nyújtott hitelesítések adják a legjobb megoldást. Azt azonban a kriptográfiai részben már kitárgyaltuk, hogy az aszimmetrikus kriptográfia csak korlátozottan terjedt el a szenzorhálózatokban, ugyanis ezek az algoritmusok nem implementálhatóak egyszerű mikrokontrollereken. A megoldáshoz vagy egy dedikált kriptográfiai modult kell alkalmazni, vagy erősebb mikrokontrollert, processzort. Mindkét megoldás plusz költségeket vet fel, amelyeket, ha lehetséges, akkor érdemes elkerülni. Megoldást egyrészt olyan kulcscsere protokollok jelentenek, amelyek működnek a rádiós környezetben is, illetve ismertetni fogunk külön szenzorhálózatokhoz készített protokollokat, amelyek hitelesítik a résztvevő csomópontokat.

4.1.4 Kommunikáció elérhetősége (availability) a szenzorhálózatokban

A szenzorhálózati alkalmazások egy csoportjának a működési alapja, hogy a szenzor által mért adatokat biztonságosan eljuttatja a gyűjtő számára. Mint a fentiekben láttuk, képesek vagyunk az üzeneteket bizalmasan továbbítani, észrevesszük, ha a támadó módosítaná azt menet közben, valamint az adatok forrását is hitelesítjük. Továbbra sem ér azonban az eljárásunk semmit, ha nem tudjuk biztosítani, hogy az üzenetek valóban megérkezzenek a szenzorokból a gyűjtőhöz. Biztosítani kell tehát, hogy a szenzorhálózat csomópontjai működőképesek maradjanak és elláthassák kommunikációs funkcióikat. Többugrásos szenzorhálózatokban ez még nehezebb feladat, hiszen itt nem csak a végpontokat kell védeni, hanem a köztes pontokat is, valamint a támadások is bonyolultabbak lehetnek.

Az állandó elérhetőség biztosítása nagyon nagy kihívás és a támadóról alkotott feltételezéseink szerint lehetetlen is, hiszen feltételeztük, hogy a támadó minden átvitt adatot minden egyes csomópontban nyomtalanul eltűntethet. A gyakorlat szerencsére ennél jobb képet mutat, ezt az elméleti lehetőséget olyannyira költséges lenne a támadónak megteremtenie és hosszú távon fenntartania, hogy az már valóban lehetetlen. Mindazonáltal szükséges, hogy a szenzorhálózatot felkészítsük az ilyen jellegű támadásokra, hiszen a csomópontok egy részénél ideiglenesen vagy akár tartós ideig is számítani lehet az elérhetőséget érintő támadásokra. Amennyiben nincs megfelelő védelem, úgy a többugrásos adattovábbítás esetén a támadó már kevés számú, de ügyesen kiválasztott és sikeresen megtámadott csomópont segítségével leállíthatja a teljes adatforgalmat az egész hálózatban. Okosváros alkalmazásoknál viszonylag ritka, hogy a szenzorok úgy legyenek elhelyezve, hogy az ilyen támadások esetén lehetetlen legyen a javításuk, azonban mégis érdemes legalább az átmeneti időre úgy felkészülni, hogy a szenzorhálózat önszervező módon javítja magát és a támadó által elérhetetlenné tett csomópontok nélkül is biztosítja a hálózat többi elemének helyes működését. A konkrét megoldásokról később esik majd szó.

A fenti három tulajdonsággal valójában a már ismertetett CIA háromszöget jártuk körbe. Azt írtuk, hogy ez a három tulajdonság jelenti a biztonság alapját. Ugyanakkor, ha a védelemről gondoskodunk, az még nem jelenti feltétlenül azt, hogy minden támadás ellen védve vagyunk. Érdemes tehát a CIA háromszögön túl is vizsgálódni és további támadási felületeket és védekezéseket keresni.

4.1.5 Üzenetek frissessége (freshness) a szenzorhálózatokban

Azokban a szenzorhálózatokban, ahol adatokat gyűjtünk, és azokat továbbítjuk feldolgozásra, nagy szükségünk van az adatok frissességének biztosítására. A szenzorok mérnek és a mért értéket továbbítják, de a támadó akár eltűntethet egy adott üzenetet és lecserélheti azt egy másik, más időben mért adatot tartalmazó üzenettel. Például a támadó egy időjárás szenzor reggeli és esti üzeneteit felcserélheti, ezzel okozva zűrzavart a feldolgozásban a ráépülő szolgáltatásnál. Mind a reggeli és esti adatok titkosítottak, nem módosítottak, az adatok forrása pedig hitelesített, mégsem működik jól a szolgáltatás. A szenzorhálózatnak vagy a szenzorokon futó alkalmazásnak tehát a titkosításon, integritásvédelmen, hitelesítésen túl biztosítani kell azt is, hogy a szállított információk mindig frissek legyenek, ne lehessen azokat később visszajátszani.

Az általános megoldások itt kétféle eljárást követnek. Amennyiben a szenzorhálózat csomópontjai képesek az idő megfelelő pontosságú mérésére, úgy, hogy a támadó azt képtelen legyen összezavarni, úgy a küldött üzeneteken belül elhelyezhetnek az aktuális óraállásról is információt, úgynevezett időbélyeget, amelyet aztán a gyűjtő

oldalon megítélhetnek frissesség szempontjából. Sok okosváros alkalmazásnál használnak GPS információkat is, ott az időbélyeg könnyebben megoldható. Gondot okozhat, hogy vajon tényleg befolyásolhatatlan-e az időmérés, illetve szinkronizálható-e megfelelő pontossággal a szenzorhálózatot alkotó csomópontok órája. Amennyiben a támadó befolyásolhatja a mért időt és ez igaz lehet akár a GPS esetében is, ott sajnos ez a megoldás nem használható. Szintén nem tudjuk használni a megoldást, ha a szenzorok óráit csak egy kezdeti pontban szinkronizálják, itt ugyanis akár az időjárás tényezők miatt is összejöhet akkora csúszás, hogy a csomópontok nem fogadják el egymás, egyébként az időpont mérésétől eltekintve teljesen helyes üzeneteit.

A másik megoldás szerint a küldő fél egy frissen előállított kriptográfiailag véletlen azonosítót (nonce) helyez a küldött üzenetekbe. Minden egyes küldésnél más és más azonosító kerül az üzenetbe, az adatokat gyűjtő fél pedig jegyzi, hogy milyen azonosítókat látott már, elutasítva az ismétlődéseket. Az eljárás nehézsége, hogy a gyűjtőnek fel kell jegyeznie az összes olyan kódot, amelyen már adat érkezett és ez nagyobb tárigényt és akár sok keresést jelent. Mivel feltételezhetjük, hogy a gyűjtőnél az erőforrások bővebben állnak rendelkezésre, mint más csomópontok esetén, így ez tulajdonképpen nem jelent feltétlenül kihívást. A kódokat ráadásul nem kell a végtelenségig őrizni, időnként a teljes rendszert, vele együtt a kulcsokat is újra lehet inicializálni. Ismerünk ráadásul olyan algoritmusokat, amelyeket pont ilyen feladathoz találtak ki, ezek a Bloom filterek [Bloom 1970]. Könnyebbséget jelent viszont az azonosítás és a tárolás esetében is, ha nem tetszőleges kódot, hanem egy számlálót alkalmaz a csomópont az egyediség jelzésére. Ennél a megoldásnál azonban számolni kell azzal is, hogy a természetes módon vagy támadásra visszavezethető okok miatt elveszett üzenetek hatására a számláló bizonyos értékei kieshetnek. A kieső számláló értékek ellenére is a rendszernek működnie kell. A számláló alapú megoldásnál, illetve az egyedi azonosító alkalmazása esetén a valódi frissességet nem teljesen tudjuk garantálni, csupán azt, hogy az aktuálisan átvitt adat valóban később keletkezett, mint egy már korábban megérkezett és elfogadott adat. Az egyedi azonosító alkalmazása esetén ráadásul a sorrend is meghamisítható. Az időbélyeg és a sorszám megoldások akár kombinálhatóak is.

4.1.6 További biztonsági követelmények

A fenti támadásokon és védekezéseken túlmenően még számos más hálózatbiztonságot érintő támadás ismert. Ide tartozik a letagadhatatlanság (Non repudiation) témaköre, a visszavonhatóság és az anonimitás is. Ezek a tulajdonságok azonban a szenzorhálózatok kapcsán nem hangsúlyosak, így ebben a kismonográfiában nem tárgyaljuk azokat.

Tudnunk kell azonban azt is, hogy léteznek olyan támadások is, amelyek hasonló hatásokat érnek el, mint a fent említett módszerek, de nem a szenzorokat összekötő hálózatokon keresztül működnek. Legegyszerűbb példa talán egy olyan adathamisítás, amikor nem a szenzorhálózatot befolyásolja a támadó, hogy az átvitt tartalmat megváltoztathassa, hanem magát a szenzort közvetlenül befolyásolja és így a megfigyelt eseményt változtatja meg. Konkrétan, ha egy utcára kihelyezett fénymérőt egy takaró elemmel beborítunk, az állandó sötétséget fog jelezni. A szenzor és a szenzorhálózat is ebben az esetben helyesen működik, az eredmény azonban a szolgáltatás szempontjából mégis rossz. Hasonló támadás lehet a szenzor áthelyezése, fizikai tönkretétele, működési körülményeinek befolyásolása, mint például órajelének felgyorsítása vagy lelassítása. Jelen kismonográfia ezekre az esetekre sem terjed ki. A gyakorlatban a kommunikáció elérhetőség védelméhez hasonlóan az így megtámadott csomópontokat azonosítani kell, és vagy javítani kell azokat, vagy a szenzorhálózatot nélkülük kell újjáépíteni.

4.2 TÁMADÁSOK A SZENZORHÁLÓZATOK KÜLÖNBÖZŐ RÉTEGEIBEN

A korábban ismertetett szenzorhálózati kommunikációt érintő támadások úgy is működőképesek, ha csak egyetlen egy ponton támadják a szenzorhálózatot. Léteznek azonban ennél kifinomultabb támadások, amelyek esetében több támadás együtt fejti ki hatását. A szenzorhálózatos okosváros alkalmazásokra amúgy is jellemző, hogy nagy kiterjedésű, egész városokat lefedő hálózatokban működnek, így a támadásokhoz általában szükséges is, hogy egyszerre több csomópontban fejtsék ki hatásukat. Az egyszerűbb támadások által elért hatás is megsokszorozható, ha a támadó a hálózat több pontján is jelen van. Ezeket a támadásokat elosztott támadásoknak nevezzük.

A támadó kapcsán feltételeztük azt a képességet, hogy a hálózat több pontján is jelen tud lenni egy időben. Az elosztott támadásoknál pontosan ez történik. Az elosztott támadás nem csak elméletileg létezik azonban. A szenzorhálózatok sok esetben valóban tetszőleges ponton hozzáférhetőek és ismerünk olyan technológiákat is, amellyel a támadások gyorsabban szinkronizálhatóak a hálózaton kívül, mint ahogy az üzenetek a szenzorhálózatban terjednek.

A következőkben megvizsgáljuk a szenzorhálózatot a hálózati protokollok rétegződése mentén. A klasszikus modell alapján a kommunikációt hét rétegre osztjuk fel. A legalsó rétegben található a fizikai réteg, ahol az eszköz által kibocsátott fizikai jelek vannak. Alulról a második réteg az adatkapcsolati réteg, ahol pont-pont kapcsolatok vannak, a szomszédok kommunikálnak. A következő réteg már a hálózati réteg, ahol az üzenetek eljutnak a hálózat tetszőleges pontjára. A szenzorhálózatok esetében a többi réteget már nem érdemes külön vizsgálni, innen már tulajdonképpen az Internet Protokoll érvényesül és a kommunikációnak kevésbé van szenzorhálózati specialitása.

4.2.1 Vezeték nélküli kapcsolatok megbénítása (radio jamming)

A vezeték nélküli szenzorhálózatok rádiós kommunikációt használnak, hogy elérjék az adatgyűjtő pontot, sőt, mint írtuk, a többugrásos hálózatok esetében ezt úgy teszik meg, hogy a küldött üzeneteket több köztes állomás újraküldi a gyűjtő irányába. A gyakorlatban a legtöbb szenzor alkalmazás esetében ma már éppen a rádiós kapcsolat jelenti a legnagyobb energia használatot a csomópont többi egységének fogyasztásához képest. Az adatok begyűjtése és az üzenetek összeállítása kisebb erőforrás-igényű, mint a rádiózás sok esetben, még alacsony fogyasztású rádiók esetén is. Ráadásul a rádiós modul akkor is fogyasztja egy csomópont erőforrását, ha csupán hallgató állapotban van, nem feltétlenül kell adatot küldeni.

Kihasználva a rádióhullámok terjedésének nehézségeit, gyakori támadás a rádiós kapcsolat megbénítása, amely során egy zavaró jel interferenciát okoz az elküldött rádióhullámokkal, így a rádiós adás sikertelen lesz. Az interferenciát a támadó úgy okozhatja, hogy a szenzor által használt rádiós frekvencia tartományban zajt sugároz a küldő szenzor vagy a gyűjtő csomópont közelében. A rádiós kapcsolat zavarása esetén a csomópont nem feltétlenül veszi észre az adás sikertelenségét. Amennyiben azonban észleli azt, úgy megpróbálhatja ismételten elküldeni az üzenetet, természetes hibát sejtve és reménykedve egy szerencsésebb kommunikációban. Amennyiben a támadó nem hagyja abba a zavarást, a szenzor próbálkozásai sorra kudarccal fognak végződni. A támadó ismételt üzenetküldésekre veheti rá a csomópontot és így elérheti azt is, hogy a csomópont tápellátása teljesen kimerüljön, véglegesen megállítva így a szenzort működésében. A rádiós zavarás lehet kimondottan csak az adás időtartama alatt aktív, illetve lehetséges az is, hogy a zavarás állandó. Előbbi eset költségesebb ugyan, hiszen a kommunikációt valós időben kell észlelni és értelmezni, ugyanakkor a lelepleződés kockázata kisebb.

A rádiós kapcsolat zavarásának kivédésére a szenzoroknál olyan rádió modult kell alkalmazni, amely a zavaró körülmények ellenére is képes sikeresen kommunikálni. A mobil internetet, NB IoT és LTE-CatM1 megoldásokat kivéve a szenzorhálózatoknál alkalmazott rádiók valamelyik licencmentes csatornát használják, amelyek az ISM (Industrial, Scientific and Medical radio bands) csatornák közül kerülnek ki. Tipikusan a 433 MHz, 868 MHz, 915 MHz, illetve a 2.4 GHz sávok a népszerű választások, régiónként eltérően. A rádióadások zavarása így általában ezekre a sávokra koncentrálódik, illetve mobil operátor által üzemeltetett hálózatok esetében az adott cellában működő frekvenciára. A támadó viszonylag egyszerűen elkészítheti az általános rádiójel zavarót, amely az ezeken a frekvenciákon zajt bocsát ki nagy teljesítménnyel. Ezek az eszközök nem számítanak drágának, könnyen beszerezhetőek webes áruházakból is. Használatuk legtöbbször illegális, azonban akadnak kivételek, Franciaországban például használhatják mozik és koncertek.

A zavarás ellen az egyik gyakori védekezés a frekvenciaugrásos szórt spektrumú technológia (Frequency Hopping Spread Spectrum – FHSS) [Ephremides 1987] használata. Ez egy olyan szórt spektrumú adás, amely esetében a vivő frekvenciát egy, csak az adó és vevő felek számára ismert szekvencia szerint, gyors ütemben váltogatják. Feltételezzük, hogy ezt a szekvenciát valóban nem ismeri a támadó, illetve azt is, hogy a rádióadás elég széles spektrumot használ. Ilyen esetben az átvitel sikeres lehet, azaz a zavarás sikertelen még abban az esetben is, ha az üzenet küldéséhez használt jel erőssége alatta marad a zaj jelerősségének. A módszert nem a zavarások ellen találták ki, így további előnyei is vannak az adatátvitel tekintetében. Erőssége a többutas terjedésből származó hibák kizárása, illetve az is, hogy több adás is folyhat párhuzamosan ugyanabban a frekvenciatartományban egymás tényleges zavarása nélkül. Hátránya ugyanakkor, hogy az adatok küldése jelentősen nagyobb frekvenciatartományt igényel, mintha csak egyetlen vivőfrekvenciát használnánk.

Másik sikeres védekezési megoldás lehet a közvetlen sorozatú frekvenciaszórás (Direct Sequence Spread Spectrum – DSSS) használata [Simon 1985]. Az eljárás során az adatot vivő rádiójelet egy, szintén csak az adó és vevő által ismert álvéletlen sorozattal modulálják, amelynek a frekvenciája nagyobb az eredeti rádiójelnél. A moduláció során egy nagyon széles frekvenciájú, zajhoz hasonló jel áll elő. Míg a rádióadás zajnak tűnik, addig ezt a zajt a vevő, az adásnál alkalmazott álvéletlen sorozat ismeretében szűrni tudja és így visszaállíthatja az eredeti rádiójelet. A megoldás a korábban ismertetett FHSS megoldás előnyeit is hordozza. További előny, hogy a moduláció segítségével tulajdonképpen az adás egyben titkosítva is van, hiszen a moduláció során a megfelelő álvéletlen sorozat megválasztásával gyakorlatilag egy titkosítást valósítunk meg. Szintén előnyös tulajdonsága a megoldásnak, hogy a támadó csak nehezen mérheti be a rádiós adás forrását, hiszen az nem emelkedik ki a zajból. Meg kell azonban jegyezzük, hogy a gyakorlatban alkalmazott DSSS rendszerek esetében – mint például, amit az IEEE 802.15.4 szabvány is előír – az előírt alapértelmezett paraméterek nem garantálják még, hogy az adás zavarhatatlan legyen.

Az FHSS és DSSS módszer kombinálásával már egy erősebb zavarást is túlélő megoldás alakítható ki, ötvözve a megoldások előnyös tulajdonságait is, igaz így a rádiónak már igen széles frekvenciatartomány kell átfognia.

Léteznek más megoldások is, amelyek nem elrejteni akarják a rádiójeleket, hanem éppen olyan erőssé tenni, hogy ne lehessen azokat zavarni. Szintén hatásos megoldás lehet az ultraszéles sávú (Ultra-Wide Band, UWB) jelek használata [Fowler 1990]. Ebben az esetben az adás egy szélesebb frekvenciatartományban, több rövid impulzus egyidejű átvitelével történik. A megoldás megnehezíti a támadó számára a jelek felismerését és zavarását is. A kommunikáció biztonságon kívül a szenzorhálózat alkalmazások más területén is hasznos lehet ez a technológia, mivel egyrészt csökkenti a rádiózás erőforrás használatát, másrészt akár pontos lokalizáció is megvalósítható

segítségével GPS nélkül is. A megoldás jelenlegi hátránya, hogy ugyan az IEEE 802.15.3a szabványa szabvánnyá emelte volna ezt az eljárást, azonban ez a pont meghiúsult, ugyanis két versengő változat közül képtelen volt választani a bizottság.

Végül pedig van olyan védekezési javaslat is, ami egyelőre inkább csak elméletileg létezik. Az antennák megfelelő irányításával ugyanis jelentősen csökkenthető a zavarás hatása és megnövelhető a kommunikáció során a rádiós adás és vétel jelerőssége. Egyelőre azonban ez a megoldás inkább csak elméleti marad, hiszen a legtöbb esetben a szenzorhálózatok telepítése során nincsen mód az antennák egyenkénti beállítására, illetve azok utólagos állíthatósága irreális követelményeket és költségeket vetne fel. A közeljövő megoldása lehet az együttműködéses antenna karakterisztika formálás (collaborative beamforming). Ebben az esetben ugyanis a szenzor csomópontoknak csak egyszerű antennájuk van, de az egymáshoz közel fekvő csomópontok összejátszanak a küldő oldalon és így együttesen küldik az adatokat az antenna karakterisztika formálására is képes gyűjtőpont felé. A megoldás segítségével a gyűjtőpontnál vett adás túlélheti a küldő csomópontoknál történő zavarást is. A megoldást és annak hatékonyságát egyelőre még kutatók vizsgálják, de ígéretesnek tűnik. A megoldás nem csak a zavarások ellen lenne jó, hanem képes lenne meghosszabbítani a két csomópont között áthidalható távolságot is.

4.2.2 Kommunikáció elnyomása (collision, exhaustion, unfairness)

Az előző pontban azt tárgyaltuk, hogy a rádiós kapcsolat zavarható, bár szerencsére van hatásos védekezés ellene, illetve a zavarás ténye viszonylag egyszerűen bemérhető. A következőkben már azt a szituációt feltételezzük, hogy a támadó hozzáférést szerzett az eszközhöz és képes irányítani a benne lévő rádiómodult. Az is lehetséges, hogy az alkalmazott kommunikációs megoldás nem tartalmaz védelmet, így a rádiós csatornában az adatkommunikáció szabadon hozzáférhető. A klasszikus hálózati rétegződésben ezek a támadások az adatkapcsolati réteg (data link layer) támadásai, míg az előző támadások a fizikai réteg támadásainak minősülnek

Az adatkapcsolati rétegben az eszközök egy velük közvetlenül szomszédos eszközzel kommunikálnak. Nem mindegyik, de sok protokoll esetében találunk arra megoldást, hogy mi történik akkor, amikor két eszköz egyszerre szeretné a hozzáférési közeget használni. Vannak kommunikációs protokollok, amelyek inkább megelőzni szeretnék ezt a szituációt, vannak, akik ezt felismerik és intézkednek, míg olyanok is akadnak, ahol nem létezik ez a szituáció, valamelyik eszköznek mindig elsőbbsége van és ezt érvényesíteni is tudja. Amikor a két eszköz egyszerre próbál meg kommunikálni, azt ütközésnek (collision) hívjuk. Az ütközések, minél nagyobb a forgalom a hálózatban, annál gyakoribbak, de mindenképpen létrejönnek természetes úton is.

A vezeték nélküli szenzorhálózatos környezetben egy támadási forma az ütközések generálása. A támadó ebben az esetben addig várakozik, míg az áldozatként kiszemelt szenzor csomópont meg nem kezdi a kommunikációt. Ebben a pillanatban a támadó is ugyanazon a kommunikációs csatornán adást kezdeményez, ezzel ütközést generál. A kommunikációs protokolltól függően az áldozat nem feltétlenül veszi észre rögtön az ütközést. Sőt olyan is lehetséges, hogy az ütközés az áldozattól olyan távol történik, hogy az már nem tudja azt észlelni. Amennyiben a kommunikációs protokoll úgy van felépítve, hogy a küldő értesüljön az ütközésről, akár közvetlenül a kommunikáció visszahallgatásával, akár a vevő fél nyugtázásának hiányával, úgy megpróbálhatja ismételten elküldeni az üzenetet egy későbbi időpontban. Ez a szituáció az előző fejezetben ismertetett kimerüléshez vezethet, amely a kommunikáció megbénításán túl szintén célja lehet a támadónak.

Az ütközéses támadások elleni védelem hibajavító kódok alkalmazásával történhet [MACWILLIAMS 1977]. A hibajavító kódok redundánsan ábrázolják az elküldendő adatokat és így az üzenet bizonyos fokú sérülése mellett még mindig helyreállítható marad az átküldött információ. A védőkód alkalmazása ugyanakkor megnöveli az átvitt üzenetek hosszát, és rontja a szenzor csomópontjának teljesítményét, vele együtt élettartamát is, hiszen a védőkód kiszámításával is foglalkoznia kell. Éppen ezért érdemes a hibajavító kódokat adaptívan alkalmazni. Amikor a kommunikációs kapcsolat problémamentes, akkor nincs szükség az alkalmazásukra, vagy csak minimális mértékben kell alkalmazni. Amikor viszont a szenzor csomópont túl sok ütközést tapasztal, ami esetleg támadásra is utalhat, megpróbálkozhat egyre erősebb és erősebb hibajavító kód alkalmazásával. A hibajavító kód mindenféle ütközéses szituációban segíthet, nem csak a támadások miatt érdemes alkalmazni.

Ismert a szenzorhálózatok keretén belül is olyan vezeték nélküli kommunikációs protokoll is, ahol vezérjelet (token) alkalmaznak az eszközök kommunikációjának vezérlésére. Többek között a WiFi is képes ilyen működésre, de a Bluetooth LE is ilyen protokoll. A tokenes vezérlés esetén az időt rövidebb szeletekre osztják fel és a hálózati vezérlő határozza meg a token kiküldésével, hogy melyik csomópont kommunikálhat egy adott időszeletben. A vezérlő akár elsőbbséget is adhat bizonyos csomópontoknak a kommunikáció során és számukra gyakrabban oszthat vezérlőjelet. A szenzorhálózatoknál létezik olyan protokoll is, amelyben elsőbbséget élveznek azok a szenzor csomópontok, amelyek már gyengébb erőforrásokkal rendelkeznek éppen azért, hogy azoknak már kevesebbet kelljen várakozniuk, így csökkentve energia-fogyasztásukat. Az ilyen protokollok esetén a támadó a valósnál nagyobb prioritást és kommunikációs igényt jelezve a vezérlő felé elérheti, hogy a többi szenzor csomópont ne jusson megfelelő kommunikációs csatornához és így egy igazságtalan (unfair) helyzetet teremtve megbénítsa vagy jelentősen hátráltassa a kommunikációt.

A fent említett támadásokat a szenzorhálózatot alkotó csomópontok nehezen vagy egyáltalán nem tudják érzékelni. Az egész szenzorhálózat működését azonban magasabb szinten megfigyelve láthatóvá válnak a csalások és beazonosítható akár a támadó csomópont is. A támadó csomópont kizárható a hálózatból, ezzel megállítva a támadást és megakadályozva annak folytatását. A csomópontok azonosítása hitelesítéssel megvalósítható, a konkrét megoldásokról a későbbiekben lesz szó.

4.2.3 Az üzenettovábbító protokoll támadása

Az üzenettovábbító protokollok biztosítják, hogy a hálózat tetszőleges csomópontjának üzenete elérhessen egy másik szenzor csomópont a hálózatban, illetve adott esetben a hálózaton kívülre is kommunikálni tudjanak a csomópontok. Az üzenettovábbítás nagyon sokféleképpen valósul meg a szenzorhálózatok esetén. Egyszerűbb esetben a szenzorok közvetlenül egy átjáróval kommunikálnak és a forgalom további kezelését már az átjáró intézi. Mobil internet, NB-IoT, LTE-CatM1, LPWAN protokollok mind ilyenek. Számos esetben ugyan IP kommunikáció van az átjárón túl, a szenzorhálózaton belül mégsem IP alapján közlekednek a csomagok. Az LR-WPAN megoldások esetében azonban gyakran találkozunk többugrásos megoldásokkal, ahol szükség van arra, hogy a szenzor csomópontok egymás között irányítsák az üzeneteket. Ebben a környezetben léteznek útvonalválasztó protokollok [AKKAYA 2005], amelyek kijelölik az üzenetek útját és megmondják a csomópontok számára, hogy mit kell tenniük egy adott csomag érkezésekor. A továbbiakban az útvonalválasztó protokollok legjellemzőbb támadásait mutatjuk be: szelektív továbbítás (selective forwarding), nyelő támadás (sinkhole attack), fekete lyuk

támadás (black hole attack), többszörözött jelenlét (sybil attack), féregjárat támadás (wormholes attack) és végül HELLO elárasztásos támadás (HELLO flood attack).

A vezeték nélküli szenzorhálózatokban nagyon sokféle üzenettovábbító protokoll ismert. Az optimális protokoll kiválasztása leginkább a szenzorhálózat egyéb tulajdonságaitól függ. A most ismertetett támadások esetén feltételezzük a támadóról, hogy ismeri ezt a protokollt és be tud épülni a szenzorhálózatba a csomópontok közé. Ebben a szituációban a támadó maga is részt vesz az üzenettovábbító protokoll működésében, így végső soron a hálózati üzenettovábbítás kialakításában. A támadó így többféle támadást is indíthat különféle célokkal. A klasszikus hálózati rétegmodellben ezek a támadások a hálózati rétegben (network layer) történnek.

4.2.3.1 Szelektív továbbításos támadás

A támadó, beépülve a szenzor csomópontok közé, dönthet úgy, hogy bizonyos üzenetek továbbítását nem végzi el, hanem eldobja azokat, ezzel megakadályozva az áldozatként kiszemelt csomópontok üzeneteinek célba érkezését. Ezt a támadást szelektív továbbításos támadásnak (selective forwarding) nevezzük. Az üzenetek eldobása helyett a támadó szándékosan rossz irányba is küldheti azokat, így megzavarva az üzenettovábbítási protokollt vagy akár jelentős erőforrás pazarlást okozva.

A szelektív továbbításos támadás elleni egyszerű, bár kétségtelenül költséges védekezés, hogy a szenzor csomópontok üzeneteiket több, egymástól független úton küldik el a fogadó csomópont számára. Költséges, mert ilyenkor biztosan lesz olyan útvonal is, amely nem optimális az üzenettovábbítás szempontjából. Amennyiben azonban létezik olyan útvonal, amely nem érintett a támadó vagy támadók által, úgy az üzenet sikeresen célba fog érkezni. A megoldás hátránya, hogy sajnos nem biztos, hogy egyáltalán létezik olyan útvonal, amely nem érintett a támadó által, így a megoldás nem minden esetben működőképes. Több útvonal esetén ráadásul az útvonalak által érintett csomópontok plusz munkát végeznek, akár jelentősen is csökkentve a kommunikációra szánt erőforrásaikat, így végső soron rövidebb működési időtartamot eredményezve.

Az előbbi megoldásnál hatásosabb védekezés lehet, ha beazonosítható a támadó csomópont és így az kizárható a szenzorhálózat forgalmából. A támadó csomópont azonosításához szükség van az összes szenzor csomópontok egyedi azonosítására. A támadás megállításához továbbá fel kell ismerni a támadó csomópont normálistól eltérő viselkedését. Ezt az eltérést a támadó csomópont közelében lévő, az adott csomópont forgalmát közvetlenül is észlelő csomópontok képesek felfedezni.

4.2.3.2 Nyelő támadás

A szelektív továbbításos támadás azonban az ismertetettnél is még hatékonyabbá fejleszthető a támadó által, amennyiben képes elérni, hogy a környezetében lévő összes csomópont a támadó felé irányítsa a forgalmát. Ezt a támadást nyelő támadás (sinkhole attack) néven ismerik [Ngai 2006]. Bizonyos szenzorhálózatban működő útválasztási protokollok esetén a támadó hamisan tűntetheti fel magát, mint igen kedvező továbbítási pont. Jelenthet magáról kedvező, adatgyűjtőhöz rendkívül közeli pozíciót, jó erőforrás kondíciót, jó rádiós kapcsolati lehetőséget. A protokoll így a támadót fogja kijelölni sok kapcsolat továbbítására. A támadás következtében aztán az üzenettovábbítás, hasonlóan a korábban ismertetett szelektív továbbításos támadáshoz, vagy meg sem történik, vagy megtörténik ugyan, de nagyon erőforrás pazarló. Mivel a támadó feltételezhetően sok útvonalban érintett, így a

támadás az egész szenzorhálózat számára jelentősebb veszteségeket okoz, súlyosabbat, mint az egyszerűbb szelektív továbbításos támadás. A védekezés történhet hasonló módon, mint a korábban ismertetett szelektív továbbításos támadás ellen.

4.2.3.3 Fekete lyuk támadás

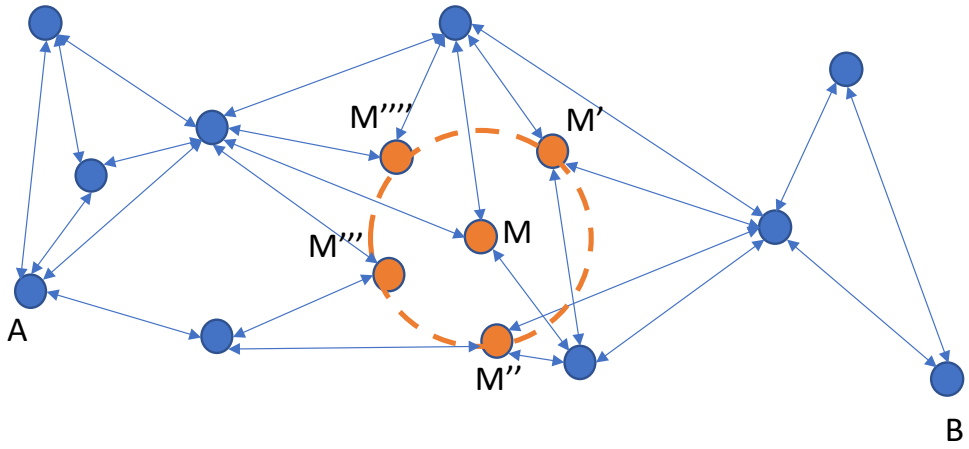
A nyelő támadás esetében a támadó célja lehet az adatforgalom megakadályozása a szenzorhálózat minél nagyobb részében. A támadást ekkor fekete lyuk támadásnak (black hole attack) nevezzük [Al-Shurman 2004], mert a támadó szinte minden forgalmat elnyel.

Ugyanakkor lehet a támadás célja a lehallgatás is. A támadó ilyenkor ugyanúgy magához vonzza a szenzorhálózat forgalmának nagy részét, de az eldobás helyett most továbbítja, csak éppen le is hallgatja az átmenő üzeneteket. A csomópontok üzeneteinek lehallgatása ellen a kriptográfia segítségével, titkosítással lehet védekezni.

4.2.3.4 Többszörözött jelenlét támadás

Az üzenettovábbítást a támadó sikeresebben befolyásolhatja, ha önmagát többszörözötten tünteti fel a szenzorhálózatban. Ez a többszörözött jelenlét támadás (sybil attack) [Newsome 2004]. A támadás esetén a támadót körülvevő szenzor csomópontok úgy érzékelik, hogy több csomópont is található a támadó csomópont helyén (4.1. ábra). A többszörözött jelenlét segítségével a támadó nagyobb befolyást szerezhet az üzenetirányításban, így nagyobb eséllyel irányítja saját kedve szerint a többi csomópont forgalmát. Szintén eredményes lehet e támadási mód az olyan szelektív továbbításos támadás védekezési módszer esetén, ahol a küldő csomópont több független utat szeretne választani a továbbításhoz. A küldő csomópont ugyanis így azt fogja feltételezni, hogy sikerült több független utat választania, de valójában az általa küldött üzenetek ugyanazon a támadó csomóponton fognak keresztül menni.

A többszörözött jelenlét támadást a csomópontok pontos hitelesítésével meg tudjuk akadályozni. Amennyiben egy adott csomópont csak egyetlen eszköz, saját maga azonosítására képes, úgy nem lesz lehetséges megszemélyesítenie több más csomópontot. Egy másfajta védekezés, amikor azt használják ki, hogy a szenzorhálózat csomópontjainak általában csupán egyetlen rádiójuk van és az képtelen egyszerre adni és venni is. A többszörözött csomópontok így felderíthetőek. Felderítésekor a szomszédos csomópontok megszólítják egymást, választ várva. Amennyiben adott csomópontoktól többször nem érkezik válasz, fennállhat a gyanú, hogy azért nem tud válaszolni, mivel egy többszörözött csomópont és éppen a sok közül csak egyetlen személyisége van aktívan a rádióra hangolva. A támadás felderítésének ezen módszere nem túl pontos és ráadásul felettébb erőforrás igényes, így, ha lehetőség van rá, akkor inkább a kriptográfiai algoritmusokon alapuló hitelesítést kell alkalmazni.

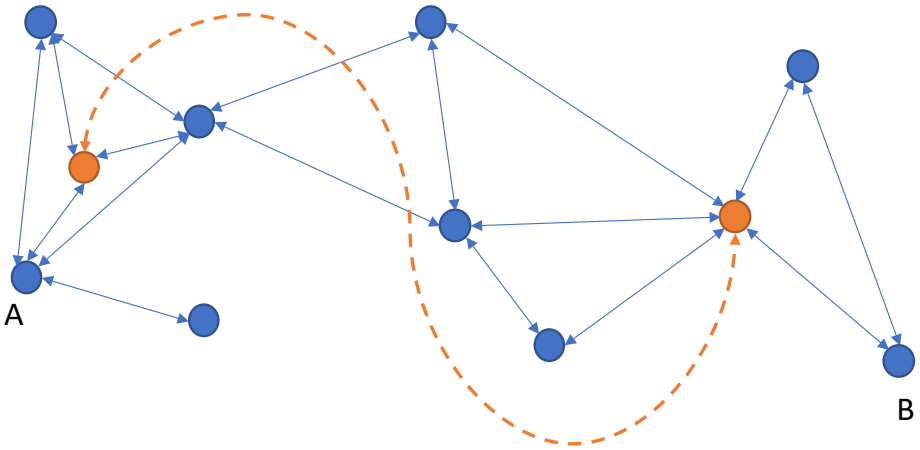


4.1. ábra. Többszörözött jelenlét támadás esetén a csomópontok azt hiszik, hogy M’, M’’, M’’’, M’’’’ mind létezik, de valójában csak egyetlen M csomópont van

Forrás: a szerző saját szerkesztése

4.2.3.5 Féregjárat támadás

Összejátszó csomópontok támadása lehetséges a féregjárat támadás (wormhole attack) [Win 2008] formájában. Ebben az esetben a szenzorhálózat két távoli pontján lévő, a támadó irányítása alatt lévő csomópont összejátszik és egymás közötti, valótlanul rövid távolság adatokat közöl az szenzorhálózat útválasztási protokollja számára. A támadás segítségével így a támadó egy rövidebb és gazdaságosabb útvonalat hirdet a két összejátszó pont között, mint amit a többi lehetséges útvonal lehetővé tesz. Így a támadó elérheti, hogy a hálózat két távoli részéből a forgalom mindig csomópontjain keresztül haladjon. Az üzeneteket ezután a támadó vagy lehallgatja és akár módosítja vagy szelektíven továbbítja a korábban említett károkat okozva. A féregjáratok esetében megvalósítható, de költséges lehet a két pont között egy a szenzorhálózatnál gyorsabb útvonalat felépíteni. Amennyiben az útvonalválasztó protokoll nem tartalmaz a féregjárattal szembeni védelmet, úgy egyszerűbb, ha a támadó az útválasztási üzenetek meghamisításával csak hazudja a jó összeköttetésű szomszédsági kapcsolatot (4.2. ábra).



4.2. ábra. A féregjárat miatt A és B úgy gondolja, hogy csak egyetlen csomópont van köztük

Forrás: a szerző saját szerkesztése

A féregjáratok felismerése nem könnyű feladat, a detektálásukra irányuló eljárások megpróbálják feltérképezni az egymással kommunikáló szomszédos csomópontok helyzetét vagy megpróbálják megmérni a szomszédos csomópontok közötti kommunikáció tényleges időtartamát. Sajnos azonban ezek bármelyike sem minden szenzorhálózat esetében tehető meg. Előbbihez szükséges, hogy a csomópontok fel tudják térképezni környezetüket akár a rádiós modul segítségével. Amennyiben a feltérképezés sikeres, úgy kellően nagy valószínűséggel meg lehetne állapítani, hogy két szomszédosnak gondolt csomópont távolsága nagyobb, mint amit a rádiójuk paramétere megenged, így a féregjárat felismerhető. Utóbbi megoldáshoz pedig tökéletes óraszinkronizációra van szükség. Ebben az esetben ugyanis felismerhető, hogy az üzenetek több időt töltenek el a két pont között, mint amennyit az útvonalválasztó protokoll ígér. Léteznek olyan útvonalválasztó protokollok is, ahol eleve nem alakulhat ki féregjárat, ugyanis az útvonalválasztás a leggyorsabb útvonal mentén halad és nem a csomópontok által közölt információk alapján van kijelölve.

4.2.3.6 HELLO elárasztásos támadás

Végül az utolsó itt említett támadás a HELLO elárasztásos támadás (HELLO flood attack) nevet kapta [Singh 2010]. A név onnan ered, hogy több szenzorhálózatos útvonalválasztási protokoll esetében is létezik egy speciális üzenet, amellyel két szomszédos csomópont jelzi egymásnak, hogy ők szomszédok. Ez a HELLO üzenet. Amikor egy szenzor csomópont HELLO üzenetet kap és az erre küldött válaszra is nyugta érkezik, akkor a két csomópont szomszédosnak tekinthető. Mindkét csomópont felveszi a másikat a szomszédsági listára. A támadó azonban ezt a protokoll üzenetet ki tudja használni, hogy valótlan szomszédossági viszonyokat hozzon létre. A támadó egy erős rádió segítségével több csomópontot is elérhet, valamint azzal szomszédságot alakíthat ki. A megvalósított támadás alkalmas lehet arra, hogy a támadó nyelő támadást hajtson végre és forgalmat vonzzon el a hálózat megtámadott részéről. Az üzenetek továbbá felhasználhatóak egy olyan kimerítéses támadás megvalósítására is, ahol a támadó nem nyugtázza a HELLO üzenetére érkezett válaszokat, de azokat folyamatosan küldi. A megtámadott csomópontok így aztán erőforrásokat pazarolnak egy valójában nem is létező szomszéd felderítésére.

A HELLO elárasztásos támadás ellen védelmet nyújthat a csomópontok megfelelően erős kriptográfiai hitelesítése és a már korábban ismertetett védelmek is. Amennyiben a támadó nem nyelő támadást hajt végre, hanem a csomópontokat akarja kimeríteni, úgy az egyszer átvert áldozat csomópont megjegyezheti, hogy ki volt a nem létező szomszéd, így a következőkben már nem szükséges erőforrásait pazarolni a HELLO üzenetekre küldött válaszokkal.

4.2.4 Szolgáltatások támadása

A hálózati protokollok felett a szenzorhálózatok esetében már az alkalmazás helyezkedik el. Innen nézve a hálózati rétegek már transzparensek, azaz a szolgáltatás szempontjából mindegy, hogy milyen hálózati technológia szállítja azt a szenzorok között, az alkalmazási rétegnek erre már nincsen befolyása. Itt tehát már nem a szenzorhálózat elleni támadásokat vizsgáljuk, hanem konkrétan a szenzorok, illetve az általuk nyújtott szolgáltatások elleni támadásokat.

A szenzorok működésüket tekintve csoportosíthatóak aszerint is, hogy az általuk gyűjtött adatokat maguktól közlik időnként, események hatására közölnek adatváltozást vagy igényvezérelt módon működnek, és amikor másnak szüksége van az általuk szolgáltatott adatra, akkor kérdezik csak le őket. Az utóbbi két esetben a támadó kimerítéses támadást tud végrehajtani a szenzor csomópont ellen abban az esetben, ha sokkal több munkára készteti, mint amennyire a szenzor élettartamát tervezték. Az eseményvezérelt szenzorok esetében a támadónak közel kell kerülnie a megtámadott szenzorhoz, hiszen ekkor tud csak eseményeket generálni számára. Ez a támadás a gyakorlatban nem túl valószínű, hiszen amennyiben a támadó célja a szenzor blokkolása, úgy akár tönkre is teheti azt. Az igényvezérelt működés esetében azonban a támadó akár távolról is kimerítheti a megtámadott szenzor erőforrásait. A támadás során a támadó lekérdezésekkel árasztja el az áldozat szenzort és a szenzor, minden lekérdezésre válaszolva, előbb vagy utóbb kimeríti a működésre szánt erőforrásait.

Ez a támadás a klasszikus hálózatbiztonságban is előfordul és az ott alkalmazott védekezés akár itt is sikeresen alkalmazható lehet. A védekezés során, amennyiben az áldozathoz rendellenesen sok lekérdezés érkezik egy adott idő alatt, és ez felveti annak gyanúját, hogy támadás áldozata lett, úgy a szolgáltatás teljesítését egy feladvány (client puzzle) teljesítéséhez kötheti [Jules 1999]. Ezek a feladványok úgy vannak megalkotva, hogy a megfejtésük nehéz, de az ellenőrzésük nagyon egyszerű. A gyakorlatban a feladvány legtöbbször egy kriptográfiai kivonatot készítő algoritmus kimenete. A kriptográfiai kivonatképző algoritmusokat egyirányú függvényeknek is nevezik, mert teljesül rájuk nézve az, hogy a kimenetből csak kimerítő találgatással lehet a bemenetet meghatározni. A szenzor így az egyirányú függvény által gyártott kimenethez kér majd bemenetet. A feladvány erőssége állítható olyan módon, hogy rövidebb kimenethez kevesebb találgatás szükséges, míg hosszabbhoz több. Itt a növekedés exponenciális. A lekérdezőtől érkező válasz ellenőrzése egyszerű feladat a kivonatképző algoritmus gyorsan lefuttatható. Amennyiben az idő paraméter is szerepel a feladványban, úgy a szenzornak még a kiadott feladatokat sem szükséges, hogy tárolja, így a védelem működése tárhelyben sem korlátos. A védekezést aktiválva sajnos azok a csomópontok is munkára kényszerülnek, amelyek nem támadó céllal végzik a lekérdezést, éppen ezért a védelmet csak akkor szabad alkalmazni, ha a szenzor támadást gyanít.

5. OKOSVÁROS SZENZORHÁLÓZAT TÍPUSOK VÉDELME

A második fejezetben már megállapítottuk, hogy a szenzorhálózatos okosváros alkalmazások átviteli technikája a gyakorlatban a következő megoldásokra korlátozódik:

- LR-WPAN (Low Rate Wireless Personal Area Network) megoldások, mint ZigBee, Bluetooth LE és társaik
- LPWAN (Low Power Wide Area Network) megoldások, mint LoRaWAN és SigFox
- Mobil Internet megoldások, valamit NB-IoT és LTE-CatM1 megoldások

Az alábbiakban áttekintjük ezen hálózatok szenzorspecifikus részét és megmutatjuk, hogy a feltárt támadások esetén az ismertetett algoritmusok segítségével konkrétan milyen védekezéseket alkalmaznak.

5.1 LR-WPAN HÁLÓZATOK VÉDELME

Az LR-WPAN megoldások majdnem minden okosváros alkalmazásnál, konkrétan az itt bemutatott alkalmazás csoportok közül az okos közművek, egészségügyi alkalmazások, okosotthon vezérlés, okos közlekedés esetében fordul elő. A szenzorhálózat rövid hatótávolságú rádiókra épül, amelyek akár egymással együttműködve alkotnak nagyobb szenzorhálózatot. Több alkalmazás esetében is, mint például az okos közművek, okosotthon vezérlés, okos közlekedés jellemző a többugrásos adattovábbítás is, azaz a szenzorhálózatban szereplő csomópontok itt egymás forgalmát is továbbítják.

A szenzorhálózatokat a kutatók évtizedek óta kutatják, és kezdetben az elképzelésük az volt, hogy egy többugrásos szenzorhálózat akár annyira heterogén lehet, hogy több felhasználó csomópontja is kooperál az üzenetek továbbjuttatásának hatékonysága miatt. A mai gyakorlat ennél konzervatívabb, egy szenzorhálózatban csak az adott szolgáltatás szenzorait képzeljük el és még akkor sincsen közreműködés, ha két szenzorhálózat működési területe átfedésben van. Bár így is fel kell készülni a különböző támadásokra, de a szenzorhálózat így jobban védhető, egyszerűbb például a hitelesítés és a támadások megállításában együttműködés is várható.

Támadásokat tekintve az elemzésünk alapján fel kell készülni adatlopásra, adathamisításra, vezérlő üzenetek megváltoztatására, illetve hamisan előállított vezérlőüzenetekre. Veszélyben lehet az az átjáró is, amely biztosítja az adatok továbbítását más hálózatok felé, sőt itt felmerül a magánszféra védelme is. Az adatot érintő támadásokra kriptográfiai titkosítást tudunk majd alkalmazni, a hamisítások ellen integritásvédelmet és hitelesítést.

Az átjáró védelme túlmutat a szenzorhálózatok védelmén, hiszen itt elsősorban a másik hálózatból, tipikusan az internet felől érkezik a támadás. Ugyanez igaz a magánszféra védelmére is, itt arról volt szó, hogy vannak esetek, amikor a felhasználó okostelefonja az átjáró és a szolgáltatás igénybevétele miatt kénytelen olyan alkalmazást telepíteni, amely visszaélhet a számára biztosított jogosultságokkal. Látható, hogy ebben az esetben nem a szenzorhálózat a támadás célpontja és a támadás azon kívül történik. Sajnos a ma használt hivatalos okostelefonos

ökoszisztémák mellett nincs is remény arra, hogy ezeket az alkalmazásokat megállítsák. A felhasználó természetesen választhat, hogy telepíti-e a szolgáltatást vagy sem, de ennél több döntést ritkán végezhet. A védekezés így tulajdonképpen úgy történik, hogy a felhasználó nem veszi meg az érintett terméket és így elkerüli a magánszféráját érintő támadásokat. Nagy valószínűséggel talál a piacon olyan alkalmazást, amellyel eléri a kívánt funkcionalitást, magánszférájának veszélyeztetése nélkül is. Bár feltételezve, hogy a gyűjthető adatok valóban értékesek, a tiszta alkalmazás várhatóan drágább lesz.

A szenzorhálózat csomópontjait megvizsgálva az LR-WPAN hálózatok legtöbb esetében azt találjuk, hogy nagyon sok szenzor alkotja a hálózatot, így rendkívüli módon törekednek a csomópontok gazdaságos előállítására. A szenzorok mikrokontrollere éppen a szükséges teljesítményt tartalmazza, éppen a szükséges tápellátással. Lehetőleg kis méretűek és jellemző a tervezésben, hogy évekig kívánjuk elemről működtetni azokat. A tervezési döntések így egyben azt is behatárolják, hogy milyen védelmet tudunk alkalmazni. Tipikusan ez a terület lesz az, ahol alkut kell kötni a biztonság rovására és csak azokat a területeket lehet majd jól védeni, ahol valóban nagy a támadások fenyegetése.

5.1.1 Komplett biztonsági protokollok

A következőkben olyan vezeték nélküli szenzorhálózatokban gyakran alkalmazott, komplett biztonsági protokollokat ismertetünk, amelyek azzal a céllal készültek, hogy átfogó védelmet nyújtsanak több támadás ellen is. Nem egy konkrét támadás védelmére, hanem az egész hálózat minél teljesebb védelmére készültek. Amint látni fogjuk, teljesítik azt a követelményt is, hogy minél kevesebb erőforrással, és így szerényebb képességű szenzorok esetében is, megfelelő szintű biztonságot nyújtsanak. Ezek a biztonsági komponensek kiegészíthetik a szenzorokban működő operációs rendszereket és így biztonságossá teszik a teljes szenzorhálózati infrastruktúrát. A bemutatott protokollok egyszerre védik az adatok biztonságát, a vezérlő üzenetek biztonságát, nyújtanak integritás védelmet és biztosítanak hitelesítést a szenzorhálózatban.

5.1.1.1 SPINS – Security Protocols for Sensor Networks

A SPINS rendszert [Perrig 2001] a Berkeley egyetemen dolgozták ki és publikálták még 2001-ben. A rendszernek két alapvető építő eleme van, az egyik a SNEP (Secure Network Encryption Protocol) a másik pedig a µTESLA (micro version of Timed, Efficient, Streaming, Loss-tolerant Authentication) eljárás. A SNEP a két csomópont között átvitt adatok és üzenetek biztonságáért felelős, míg a µTESLA a broadcast üzenetek hitelesítését támogatja. A broadcast üzeneteket a csomópontok úgy küldik el, hogy nincsen megjelölve címzett, azt a szenzorhálózat összes szenzora észlelheti. Erre a két építőelemre alapozva a vezeték nélküli szenzorhálózat legtöbb támadása kivédhető, illetve segítségükkel akár új protokollok is építhetőek. Egy ilyen protokoll a biztonságos útvonal választó protokoll, amely, mint láttuk, kritikus eleme a vezeték nélküli szenzorhálózatoknak.

A SNEP építőelem felelős a védelem legtöbb pontjáért: kezeli az adatok és üzenetek titkosítását, ellátja integritásvédő kóddal a védendő kommunikációt, hitelesítést biztosít a csomópontok számára, sőt még az adat frissessége is ellenőrizhető. A SNEP protokollt kimondottan vezeték nélküli szenzorhálózatokhoz alkották, így ügyeltek arra, hogy a megfelelő biztonsági szint nyújtása mellett kevés plusz erőforrást igényeljenek. Ügyeltek rá, hogy a

biztonság ne növelje meg jelentősen a mikrokontroller számítási terhelését és a felhasznált memóriaterületet mind a programtároló, mind a dinamikus, futás közbeni memóriaterület esetén, illetve arra is ügyeltek, hogy a két csomópont között átvitt adatmennyiség ne növekedjen túlságosan a biztonság hatására. Mivel a protokollt 2002-ben készítették, ezért nem fogunk találkozni olyan megoldásokkal, mint például az OCB blokkösszefűzési mód vagy a modern folyam titkosítók, amelyeket csak később alkottak meg.

A SNEP a bizalmasságot a titkosítás segítségével éri el, mint minden más biztonsági protokoll. A protokoll megalkotói ügyeltek arra, hogy amennyiben kétszer ugyanaz az adat kerülne átvitelre, mert például a szenzor által mért adat megegyezik mindkét esetben, az adatot tartalmazó titkosított üzenet akkor is különböző legyen. A támadók így az üzenetek lehallgatásával még összefüggéseket sem tudnak felfedezni a közölt adatokban. Ezt a kriptográfiában szemantikus biztonságnak nevezik. Más működési környezetben a szemantikus biztonságot véletlen bitek beszúrásával lehet egyszerűen elérni, a szenzorhálózatok esetében azonban ez túlzott erőforrás pazarlás lenne. Éppen ezért, e helyett a két kommunikáló fél inkább monotonon növekvő számlálókat használ az üzenetekben erre a célra, így biztosítva minden egyes üzenet egyedi tartalmát és így egyben egyedi titkosított tartalmát is.

Az alkalmazott számláló olyan nagy, hogy jogosan feltételezhető, hogy a szenzorhálózat működése során egyszer sem fog túlcsordulni, így az ismétlődés ki van zárva. Egy ilyen nagyméretű számláló üzenetek mellé csatolása azonban méretbeli gondot jelentene. Ebben az esetben viszont a számláló értéke nem utazik a hálózaton, azt a kommunikáló felek a saját memóriájukban tárolják csak. A számlálót felhasználják a titkosítás közben és ez biztosítja a nem ismétlődő titkosított üzenetet még ismétlődő adat esetén is. A számláló egyben az adat frissességét is hivatott garantálni, legalábbis az üzenetek így már nem felcserélhetőek a támadó által. Az integritásvédelem és hitelesítés feladatát egy kulcsos kriptográfiai kivonatképző algoritmussal gyártott üzenet hitelesítő kód látja el. A szenzor hitelességét az hivatott igazolni, hogy ismeri a titkos kulcsot, amely a titkosításhoz és a kivonatképzéshez szükséges. A titkosítás maga ugyan nem növeli meg az átvitt üzenet méretét, hiszen nem csatolunk hozzá semmilyen plusz információt, viszont a hitelesítő kód megnöveli azt. Amennyiben a protokoll 64 bites blokkméretű blokk titkosítót használ, úgy ez esetben 64 bittel nő minden egyes átvitt üzenet mérete, ennyi ugyanis az integritásvédő kód hossza.

Fontos azonban megjegyezni, hogy mint arra már kitértünk, a fenti eljárással csak egy gyengébb biztonságú frissességet lehet biztosítani. Az üzenetet vevő csomópont a számláló állása miatt abban biztos lehet, hogy az üzenetet nem játszotta újra egy esetleges támadó, de azt nem tudhatja, hogy valóban akkor keletkezett, amikor az üzenet megérkezett. Ennél nagyobb biztonságot nyújtó adat frissesség is elérhető, ekkor azonban a kommunikációt egy újabb üzenettel kell kiegészíteni. Amennyiben az alkalmazás működése igényvezérelt, úgy a lekérdező fél kezdeményezi a kapcsolatot és abban az üzenetben közölhet egy frissen előállított véletlen kódot a frissesség ellenőrzése véget. A lekérdezett csomóponttól érkező válaszban a szenzornak pontosan ugyanezt a kódot kell visszaküldenie, úgy, hogy azt a támadó ne legyen képes megváltoztatni. A lekérdező a válasz megérkezésekor ellenőrzi ezt a kódot és csak abban az esetben fogadja el a közölt értéket, ha egyezést talál ezen a ponton. A támadó így képtelen bejátszani egy korábban rögzített üzenetet, mert elvárható, hogy a lekérdező által küldött érték mindig más és más. A lekérdező által küldött véletlen kód akár utazhat nyíltan is a szenzorhálózatban, hiszen csak azonosításként fog szerepelni a lekérdezőnél. A támadó nem éri el célját, ha ezt a kódot a lekérdezés közben megváltoztatja. Ugyanakkor az sem kizárt, hogy már a lekérdezés is titkosított, így a SNEP segítségével a lekérdező védi ezt a paramétert is az üzenetben. A plusz kód azt garantálja tehát, hogy a válasz pontosan arra a lekérdezésre készült, amelyben ez a kód szerepelt. Ahhoz, hogy a frissesség időben is korlátozva legyen, a lekérdezőnek mérnie kell a kérés és a válasz között eltelt időt.

A SNEP működésének lényeges pontja a titkosításhoz és integritásvédelemhez használt kulcsok kiosztása. Itt a protokoll megalkotói minden egyes szenzor csomópont és adatgyűjtő párkapcsolatra egy statikusan telepített mester kulcsot helyeztek el az eszközökben. A mester kulcs nem kerül közvetlenül felhasználásra, hanem a titkosításhoz és hitelesítéshez használt kulcsokat a szenzor és az adatgyűjtő ebből származtatja. Az aktuálisan használt kulcsok meghatározásához az eszközök a hitelesítésnél is használt kriptográfiai kivonatképző algoritmust használják fel. A kommunikáció közben, amennyiben felmerül a gyanú, hogy az éppen használt kulcsok kompromittálódtak, úgy a felek új kulcsot készítenek a mesterkulcs segítségével. A megoldás azt igényli, hogy az eszközökben a mesterkulcsok előre legyenek telepítve. Ez különösen akkor lehet probléma, amikor egy nagy számosságú szenzorhálózat esetén a csomópontoknak túlságosan sok kulcsot és számlálót kell nyilvántartaniuk. A kulcsoknak nem csak a konzisztens konfigurálása nehéz ugyanis, hanem a szenzornak elegendő és biztonságos tárhelyet kell biztosítani a mesterkulcsok és a kapcsolatokhoz tartozó számlálók számára. A helyzet megoldása az lehet, hogy ebben az estben korlátozzuk a kommunikációt. A szenzor csomópontok amúgy is csak ritkán szoktak egymással érdemben is kommunikálni, sőt sok alkalmazás ezt nem is igényli, a cél, hogy az adatok a gyűjtőt elérjék. Ebben az esetben viszont elegendő, ha a mesterkulcs és számláló csak az adott szenzor és a gyűjtő viszonyában létezik. Amennyiben az feltétlenül szükséges, még így is tud két csomópont egymással kommunikálni, amennyiben az adatgyűjtő az üzeneteket átjátssza a szenzorok között.

A µTESLA protokoll broadcast üzenetek hitelesítésére lett kitalálva. A broadcast üzenetek a teljes szenzorhálózatban szétterjednek, ezt a megoldást ezért elsősorban a nyelő használhatja olyan üzenetek szétosztására, amelyek minden csomópontot érintenek. A megoldásban a már létező TESLA protokollt [Perrig 2000] alakították úgy, hogy az a szenzorhálózatokban is jól tudjon működni. A TESLA protokoll megalkotói között találjuk a SPINS és SNEP protokollok szerzőit is. A TESLA protokollt még ad-hoc hálózatokhoz fejlesztettek ki, ahol a hálózat felépítése ad-hoc módon történik és számítottak idegen szenzorokra is. Ez a környezet nem pontosan fedi a szenzorhálózatokat ezért a SPINS megalkotói ezeket az itteni környezetben kedvezőtlen tulajdonságokat változtatták meg úgy, hogy a protokoll lépéseit az erőforrásokban gyengélkedő szenzorok is végre tudják már hajtani. Az eredeti protokoll ugyanis tartalmazott digitális aláírást is, amely az aszimmetrikus kulcsú titkosításhoz hasonlóan rendkívül erőforrásigényes és ezért, ahogy azt korábban megállapítottuk, nem is kivitelezhető egy gyengébb szenzorokból álló szenzorhálózatban. A digitális aláírást szimmetrikus kulcsú titkosítás váltotta ki. Az üzenetek méretét is csökkenteni kellett, ez azzal a hátránnyal járt, hogy így az eredeti protokollhoz képest lassabb ütemben működik csak a hamarosan ismertetett kulcsfelfedés művelete. Csökkent az üzenetek száma is. Végül pedig szűkítették azon csomópontok számát, akik jogosultak broadcast üzenetek küldésére és ezáltal csökkent a kulcsok és egyéb állapotok tárolásához szükséges memóriaterületet csomópontonként.

A µTESLA protokoll működése megkívánja, hogy a csomópontok órája lazán szinkronizálva legyen egymáshoz. A laza szinkronizáció azt jelenti, hogy nem követeljük meg a mikroszekundumra pontos együttjárást, de túl nagy eltérést nem is mutathatnak egymáshoz lépest. A protokoll működéséhez időszeleteket definiálunk, és ezeknek az időszeleteknek kell átfedésben lennie az együttműködő csomópontokban. Az órák pontosságának tehát az időszeletekhez mérten kell jónak lennie. Az időszelet nagyságát minden egyes csomópont ismeri, ez egy hálózatra jellemző paraméter. A µTESLA működésének alapja, hogy a küldő csomópont egy olyan titkos kulcsot használ az elküldött broadcast üzenet hitelesítéséhez, amelyet csak ő ismer, soha előtte nem használt és nem is fedett fel, ezért mindenki más számára biztosan titkos. Amikor a vevők megkapják az üzenetet, ellenőrzik, hogy a kulcs még valóban ismeretlen. A titkos kulcs csak a megadott időszeletben lesz használható és azután már nem is lesz titkos.

Éppen ehhez a ponthoz szükséges a laza időszinkronizáció, a kulcs érvényességének eldöntéséhez az időszeleteknek szinkronban kell lenniük. A vevők az üzenet megérkezésének a pillanatában még nem tudják ellenőrizni annak hitelességét, hiszen még nem ismerik azt a kulcsot, amivel hitelesítésük történt. A szenzor csomópont ezért eltárolja az üzenetet egy későbbi alkalomig. A következő időszeletben ugyanis a küldő csomópont felfedi az előző időszeletben használt kulcsot, így a vevők már képesek lesznek a korábbi üzenet hitelességének ellenőrzésére. Fontos, hogy a vevők arról is meg tudjanak hitelesen győződni, hogy az éppen felfedett kulcs valóban a küldő által használt kulcs. Ellenkező esetben bárki küldhetne ilyen jellegű üzeneteket. Ezt úgy oldották meg, hogy a küldőnél az egymást követő időszeletekben használt hitelesítő kulcsok generálása egy egyirányú függvény ismételt segítségével történik. Az egyirányú függvény biztosítja, hogy az aktuális időszeletben felfedett kulcsból kiindulva csak találgatással lehessen kitalálni az aktuális időszelet titkos kulcsát. A kulcs pedig természetesen olyan méretű, hogy ezt a találgatást az időszelet idejében lehetetlen legyen eredményesen végrehajtani. Az üzenetet vevő csomópontok viszont nagyon egyszerűen ellenőrizhetik az éppen felfedett kulcs hitelességét, hiszen az egyirányú függvény segítségével pont a korábbi időszeletben felfedett kulcsot kell megkapniuk. Az egyirányú függvény ebben az esetben is egy kriptográfiai kivonatképző algoritmus lesz. Az időszeletek kulcsai azonban pont ellenkező sorrendben használhatóak, mint ahogy az egyirányú függvény gyártja azokat, így a küldő félnek először le kell gyártania egy hosszabb láncot a kulcsokból, majd azokat fordított sorrendben kell használnia.

A μTESLA protokoll esetében minden egyes csomópontnak számon kell tartania, hogy melyik küldő milyen kulcsot fedett fel a legutóbbi időszeletben. Ez sajnos azt is jelenti, hogy annyi tárhellyel kell rendelkeznie, hogy tárolni tudja az összes ilyen küldő kulcsát. Bár a μTESLA protokoll tetszőleges számú küldő csomópontra működne, a tárhely korlátossága miatt segítségével leginkább csak az adatgyűjtők tudnak hitelesített információt közölni a csomópontokkal. Ebben az esetben ugyanis a csomópontoknak csak egyetlen, illetve több adatgyűjtő esetén is csak korlátozott számú kulcsot kell tárolniuk.

Amennyiben elegendő tárhely áll rendelkezésre, vagy a hálózatban csak kevés útválasztó csomópont van, úgy az eljárás az útválasztó protokoll biztosítására is használható. A szomszédos csomópontok itt az állapotáról szóló információkat hitelesítik és terjesztik.

A μTESLA protokoll további előnye, hogy nem csak az üzenet titkosítását és egyben az eszköz hitelességét biztosítja, hanem egyben az információ frissességét is garantálja, hiszen az biztosan a konkrét időszeletben lett titkosítva.

Bár a protokollt még 2001-ben alkották, mai szemmel nézve a SPINS protokoll elemei még mindig megállják a helyüket. Abban az évben már ugyan létezett a ma is ismert AES titkosító, azonban a szerzők még helyette egy korábbról ismert és kompaktabb kódot biztosító blokk titkosítót az RC5 titkosítót használták [Rivest 1994]. A protokoll elemeit megtartva ez lecserélhető egy tetszőleges másik blokk titkosítóra, illetve akár a SNEP protokoll elem is helyettesíthető az OCB blokkösszefűzési módszerrel, amely szintén biztosítja az adatok hitelesítését. A SNEP esetében a hitelesítő kód előállításához a CBC blokk összefűzési módot használták úgy, hogy nem titkosítottak vele, csupán a lánc utolsó elemét tartották meg ellenőrző kódnak. Mivel az RC5 titkosító 64 bites blokkokkal dolgozott, így lett a hitelesítő kód is 64 bites.

5.1.1.2 Az IEEE 802.14.5 biztonsági megoldásai

Az IEEE 802.14.5 szabvány az LR-WPAN hálózatok működését írja le. A szabvány 2003-ban született meg és tartalmaz a kommunikáció biztonságára vonatkozó pontokat is [Sastry 2004]. A szenzorhálózat működtetése közben

alapvetően háromféle biztonsági mód közül választhatunk: nincs semmilyen biztonság, hozzáférési lista (Access Control List – ACL) használata, illetve biztonságos mód. Mint látható majd, a szabvány igyekszik a legmodernebb biztonsági megközelítést alkalmazni és segítségével megoldható a bizalmasság, integritás védelem, üzenet hitelesítés, és gyenge frissesség biztonság problémája.

Hozzáférési lista (ACL) alkalmazásának segítségével az IEEE 802.15.4 protokoll esetén az alkalmazás maga döntheti el, hogy egy adott csomópont mely másik csomópontoktól fogadhat el egyáltalán üzeneteket. Amennyiben a küldő szenzor csomópont azonosítója nem szerepel az üzenetet aktuálisan vevő szenzor hozzáférési listájában, úgy a vevő az üzenetet egyszerűen eldobja. Ez a biztonsági üzemmód csak a vevő hozzáférését korlátozza, a továbbított és vevőhöz beérkezett üzenetek bizalmassága még nincsen védve. Ahhoz, hogy a kommunikáció üzenetei is valóban védettek legyenek, a következő bekezdésben ismertetett biztonságos üzemmódot kell használni.

Az IEEE 802.15.4 biztonságos üzemmódja esetén a kommunikáció titkosításra kerül és kriptográfiai kivonatképző algoritmus által előállított integritásvédő kód védi azt a módosításától, illetve számlálókkal kiegészítve azt, az újraküldésétől is. A kriptográfiai kivonat készítéséhez használt kulcson keresztül a hitelesítés is megoldott. Ebben az üzemmódban szintén használja a rendszer a hozzáférési listákat is a korábban leírt módon. A protokoll különböző típusú üzenetei azonban nincsenek egyformán kezelve. Négyféle üzenettípust különböztetnek meg: átjátszó, nyugtázó, adat és vezérlő üzenet. A nyugta üzenetek esetében nincsen semmilyen alkalmazott biztonsági megoldás, azok védelem nélkül közlekednek. A többi üzenet esetében nyolcféle titkosítás és hitelesítés kombináció közül lehet választani, hogy a rendszer melyiket alkalmazza. A nyolcféle kombinációban négyféle hitelesítő kód méretet választhatunk: 0, 32, 64 és 128 bit, illetve választhatjuk, hogy társul-e mellé titkosítás: igen vagy nem. Ez a négy és két kombináció összesen kiadja a nyolc lehetőséget. A 0 bites hitelesítő kód tulajdonképpen azt jelenti, hogy nincs hitelesítés, így a nyolc kombináció egyike az, hogy sem titkosítás, sem hitelesítés nincs. Minden esetben a titkosításhoz az AES blokk titkosító CTR blokkösszefűző üzemmódját használják. A hitelesítéshez pedig minden esetben az AES titkosító CBC-MAC [Bellare 1994] algoritmusát. Ezekkel az algoritmusokkal nem csak szenzorhálózatokban, hanem a modern kommunikációs protokollokban is találkozunk, akár hasonló paraméterekkel. Például a mai WiFi hálózatokat is pont ezek az algoritmusok védik 128 bites kulccsal és 64 bites integritásvédő kóddal. Ez azt is jelenti, hogy amennyiben a szenzor azt erőforrásokban megengedheti, akár a klasszikus hálózatokkal egyenértékű biztonságot tudhat magáénak. Mivel sokszor maga a rádió modul implementálja a biztonsági algoritmusokat, mint például a ZigBee esetén, így a szenzor még a kis teljesítménye ellenére is használhatja azokat.

A jó algoritmus még nem jelenti a megfelelő biztonságot, ugyanis a védelemhez használt kulcsokat is helyesen kell kezelni. Az IEEE 802.15.4 esetében alapértelmezésben háromféle kulcskezelés támogatott. Az első megoldás a minden hálózatot alkotó eszközzel közösen megosztott, úgynevezett hálózati kulcs. Ez gyakorlatban hasonló az otthoni WiFi hálózatokban használt közös kulcshoz. Nyilvánvaló biztonsági probléma, hogy amennyiben a támadók hozzáférnek egy adott csomóponthoz fizikailag és sikerül belőle a kulcsot kinyerni, úgy ezzel egyszerre az egész hálózat kompromittálódik. A másik lehetőség, hogy a kommunikáló eszközök páronként egyedi kulccsal rendelkeznek. Itt akárcsak azt már kitárgyaltuk a SNEP esetében, a problémát a potenciálisan magas számú kommunikációs pár jelenti. Itt is a megoldást a gyakorlatban az jelenti, hogy a szenzorhálózat csomópontjai csak a gyűjtő ponttal vagy azon keresztül kommunikálnak. A harmadik lehetőség csoport kulcsok kialakítása, ez ötvözi a korábbi két kulcskezelési módszert. A csoportoknak lesz közös kulcsa, de több csoport lesz a teljes hálózatban. Léteznek olyan alkalmazások is, amelyek kombinálva használják a megadott módszereket, azaz például páronként egyéni kulccsal kommunikálnak a gyűjtő ponttal, míg a többi csomóponttal a hálózati kulcs segítségével teszik azt.

Az IEEE 802.15.4 biztonsági megoldásai alapvetően ott jók, ahol azok implementációja hardveres támogatást kap, hiszen ott tetszőleges mikrokontroller esetén is használható. Láthatóan nem törekedtek arra, hogy egyéb esetben szerényebb képességű szenzorok is a hálózat részét képezhessék. Az integritásvédő kód hosszának megválasztásánál pedig figyelembe kell venni a rádiós kommunikációs protokoll üzeneteinek lehetséges hosszát is.

Több, kriptográfiával foglalkozó szakember problémának tartja, hogy a nyugtázó üzeneteket nem védi a rendszer, és ezzel lehetővé teszi a nyugták meghamisítását.

5.1.1.4 TinySEC – Link Layer Security Architecture for WSNs

A TinySEC protokoll [Karlof 2004] az adatkapcsolat szintű biztonság megvalósításáért felelős. Akárcsak a SPINS protokollt, ezt is a Berkeley egyetemen fejlesztették ki és publikálták 2004-ben. A fejlesztésekor már felhasználták a SPINS protokoll SNEP építőelemével kapcsolatos szerzett tapasztalatokat.

A TinySEC protokoll kimondottan csak az adatkapcsolati szintet célozza meg, azaz hogyan lehetséges két szomszédos csomópont között üzenetet váltani. Többugrásos adattovábbítás során a köztes szenzor csomópontok a továbbítás előtt még ellenőrzik a továbbítandó üzenet biztonságát is. Támadás esetén így annak ténye már két szenzor csomópont közötti adatkapcsolatban kiderülhet, és ezzel megakadályozható, hogy az erőforrás pazarlóan elterjedjen a többi csomópontnál is.

Akárcsak a SNEP esetében a TinySEC protokoll tervezése közben is cél volt a szenzor erőforrások kímélése mellett a bizalmasság, üzenet integritás védelem és az eszköz hitelességének ellenőrizhetősége. Emellett ennél a protokollnál is megtaláljuk a frissesség védelmet, igaz itt is csak egy gyengébb védelemről beszélhetünk. A hitelesítés megvalósításához a kutatók a Skipjack titkosítót választották és az CBC blokkösszefűzési technológiát. A Skipjack szintén rövid kódot tartalmaz és rendkívül gyors titkosító, igaz biztonságát korlátozza, hogy mindössze 80 bites kulcsot használ. A TinySEC, ahogy neve is mutatja, rendkívüli mértékben kívánja leszorítani a biztonsághoz szükséges kódméretet, így a blokkösszefűzés kiválasztásánál cél volt egy olyan algoritmus kiválasztása, amely egyben integritásvédelemre is használható. A CBC üzemmód így egyben a blokkokat fűzi össze, másrészt a SNEP-hez hasonlóan az integritás védelmet és hitelesítést is szolgálja. Az üzenetméret csökkentése végett azonban az integritásvédő kódhoz nem használják fel mind a 64 bit kimenetet, hanem abból levágnak 32 bitet. Ennél a protokollnál tehát a hitelesítő kód mérete csupán 4 bájt.

A TinySEC protokoll kétféle üzenet primitívet különböztet meg. Az egyik, a titkosítást és hitelesítést is nyújtó TinySEC-AE üzenet típus, míg a másik típus a TinySEC-Auth, ahol csak hitelesítés van, titkosítás nincs. Mindkét üzenet típus esetében a protokoll a hitelesítő kódot a teljes üzenetre számítja, az első bittől az utolsóig. Mivel a TinySEC-et alkalmazó TinyOS operációs rendszer már eleve tartalmazott olyan mezőket, ahol hibavédő kódot szállítottak, ezért a hitelesítő üzenet csupán egyetlen bájtos növekedést fog okozni. A titkosított üzenet azonban már 5 bájtos növekedést okoz, ugyanis a titkosításhoz szükség van egy számlálóra, illetve az üzenetben jelölni kell az küldő fél azonosítóját is. Ez utóbbi paraméterek segítségével a gyenge frissesség ellenőrzése is megoldható, mint ahogy azt a SNEP esetén írtuk. A frissesség tehát alapesetben csak az üzenetcserék ellehetetlenítésére vonatkozik.

A TinySEC protokoll 80 bites titkosító kulcsa és 32 bites integritásvédő kódja mutatja, hogy itt egyértelműen a biztonság szintjének rovására megy az erőforrások faragása. Érdekes kérdés, hogy vajon a TinySEC által nyújtott biztonság elegendő-e az alkalmazások számára. Ez természetesen függ a támadó modelljétől és az pedig a konkrét szenzorhálózat alkalmazástól, de biztosan vannak olyan területek is, ahol ez a biztonság még sokáig megfelelő lesz. A korábban ismertetett proaktív védelmi megoldásokkal pedig ez a módszer is erősebbé tehető.

5.1.1.5 MiniSec protokoll

A MiniSec protokollt [Luk 2007] 2007-ben publikálták. Alkotói úgy gondolták, hogy a létező szenzorhálózatok biztonságával foglalkozó protokollok nem tökéletesek és lehet még mit javítani rajtuk. A SPINS esetén teljesítmény gondokat találtak, az IEEE 802.15.4 szerintük túlságosan erőforrás pazarló volt, míg a TinySEC nem volt elég biztonságos. Ezért hát alkottak egy újabb protokollt, amely a már meglévő alapokon, de a hibákat kijavítva működik. A protokollba újdonságként bekerült, hogy az üzenetekhez rendelt védelmet a szerint választották ketté, hogy broadcast vagy csak egyetlen csomópontnak szóló unicast üzenetről van szó. Új elem, hogy a broadcast üzenetek számára is biztosítottak gyenge frissesség védelmet.

A protokoll implementációja továbbra sem szakított a blokk titkosítókkal, azonban a korábbi CBC és CTR blokkösszefűzési módszerek helyett a már modernebb és erőforrás takarékosabb OCB protokollt használja fel. Az OCB esetén ugyanis a titkosítás és a hitelesítő kód előállítása egyetlen lépésben történik, így, mint azt mi is bemutattuk, a teljesítménye lényegesen jobb más megoldásoknál. Újdonság a számláló kezelése, amely nélkülözhetetlen eleme a protokollnak az üzenetek visszajátszásának megakadályozására és így végső soron a frissesség biztosítása miatt is. A számlálót a SNEP protokoll egyáltalán nem vitte át az üzenetekben, ezzel a forgalom méretével takarékoskodva. A TinySEC és az IEEE 802.15.4 szabvány egy az egyben átvitte a számlálót az üzenetben. A TinySEC esetében ugyanis ez nem sok növekedést okozott, hiszen volt már egy ilyen mező az üzenetben, amelyet csak bővíteni kellett. Az IEEE 802.15.4 protokollt pedig nem érdekli az üzenetek növekedése. A SNEP esetén a megfigyelt teljesítmény gondok pontosan abból adódnak, hogy amennyiben sok az elveszett üzenet, a számlálókat folyamatosan újra kell szinkronizálni.

A MiniSec egy köztes megoldást alkalmaz, üzenetei nem viszik át a teljes számláló tartalmát, hanem annak csak egy kis részét. Így az átvitt üzenetek hosszával is takarékoskodik, illetve ellenálló lesz az üzenetvesztések okozta problémákkal szemben is. A MiniSec emiatt viszonylag nagy, 32 bites számlálókat tud alkalmazni, amelyekből üzenetenként csak az utolsó 3 bitet viszi át szinkronizálás céljából. Ráadásul ezt a 3 bitet az üzenet fejlécében az üzenet hossz 8 bites mezőjébe szúrták be, ahol a korlátozott üzenet méret miatt ezeket a biteket amúgy sem használták. Itt is, ahogyan a többi protokoll esetében is, kérdéses volt még a hitelesítő kód hossza is. Ez a hossz végül itt is 32 bit lett, amely egy megfelelő középutat jelöl a biztonság és az erőforrásokkal való spórolás között. A protokoll megalkotásakor a Skipjack titkosítót használták 80 bites kulcsokkal, de jelezték, hogy amennyiben szükséges és megtehető, a biztonság megnövelése miatt át lehet állni az AES titkosítóra.

A MiniSec az üzenet típusa szerint a MiniSec-U unicast és MiniSec-B broadcast protokollokra bontható. Az üzenetkezelési eljárás a biztonságot illetően alapjában véve mindkét típus esetén ugyanaz, ugyanakkor a MiniSec-B esetén nem annyira triviális a frissesség biztosítása. Itt ugyanis több forrás is lehetséges, mindegyik egyedi számlálóval és a korrekt védelemhez minden egyes forrás számlálóját külön tárolni lenne szükséges. A számlálók mérete továbbra is 32 bit, azonban broadcast esetben már 8 bitet használnak az átvitel során, 3 bitet továbbra is az üzenet hosszba, míg 5 bitet a küldő cím mezőjében helyeznek el. Ezáltal ugyan csökken a broadcast címek száma a hálózatban, de még így is 2048 ilyen cím lehet a teljes szenzorhálózatban. A helyett azonban, hogy valóban minden egyes broadcast címre külön eltárolják a számláló értékét, kétfajta megoldást ajánlanak. Az egyik egy időbélyeg alapú megoldás, amely akkor működőképes, ha a szenzorhálózat csomópontjainak órája legalább lazán szinkronizált. A számlálók ilyenkor itt feleslegesek is, a frissesség az időbélyeg alapján van megállapítva. Amennyiben biztonságos az óraszinkronizáció, úgy ráadásul egyből erős frissesség védelem valósul meg. A másik megoldás a számlálókon alapul, de itt sem tároljuk el a különböző források számlálóját külön-külön. Korábban már írtuk, hogy

használható a Bloom filter. A megfelelő paraméterű Bloom filter használata esetén meg tudjuk mondani, hogy az adott üzenetet megkaptuk-e már vagy sem. A Bloom filter tulajdonsága, hogy tévedhet is, ha egy konkrét értéket keresünk, viszont amennyiben a keresett értéket már egyszer hozzáadtuk a filterhez, úgy ennek tényét tévedhetetlenül, egyértelműen jelzi. Egy kifejezetten a forrásokra és sorszámokra épített Bloom filter erőforrás takarékos megoldást jelent az összes broadcast számláló eltárolása helyett. Itt azonban csak azt ellenőrizzük, hogy az üzenet visszajátszott-e vagy sem, a sorszámból származó információt nem tudjuk használni.

A kulcsok kiosztásáról a MiniSec protokoll nem rendelkezik, azt más protokollokra bízza.

5.1.1.6 LR-WPAN biztonsági protokollok összehasonlítása

Összességében látható, hogy a különböző protokollok alapjában véve hasonló megközelítéssel dolgoztak, bár mindenhol találunk egyedi megoldásokat is. Az adatkapcsolati üzenetküldés esetében mindenhol egy blokk titkosítóval, egy integritásvédő és hitelesítő kóddal, valamint egy számláló segítségével megvalósított, frissességet biztosító, visszajátszás elleni védelemmel találkozunk. A protokollok különböznek abban, hogy milyen konkrét kriptográfiai algoritmusokat alkalmaznak, mekkora a kulcsok és mezők mérete, valamint az algoritmusok működéséhez szükséges számlálót, hitelesítő kódot és egyéb paramétereket hogyan helyezik el az üzenetekben. Ez utóbbi egyben meghatározza az üzenet hosszának növekedését is, amely a szenzorhálózatokban igen kritikus lehet. Sok alkalmazás esetében már 1-2 bájt eltérésnek is jelentősége lehet. Az alábbi táblázat foglalja össze az ismertetett protokollok főbb paramétereit.

5.1. táblázat. LR-WPAN biztonsági protokollok jellemzői

Protokoll	Blokk titkosító	Blokk-összefűzés	Hitelesítő algoritmus	Hitelesítő kód hossza (bitben)	Üzenet hossz növekmény (bájtban)
SPINS/SNEP	*RC5*	*CTR*	*CBC-MAC*	*64*	*8*
TinySEC	*Skipjack/RC5*	*CBC*	*CBC-MAC*	*32*	*1/5*
IEEE802.15.4	*AES*	*CTR*	*CBC-MAC*	*0/32/64/128*	*9 – 21*
MiniSec	*Skipjack/AES*	*OCB*	*OCB*	*32*	*3*

A pont-pont kapcsolatok biztonságán kívül a protokollok érintik a broadcast üzeneteket is. A SPINS esetében egyedi megoldásként működik a µTesla protokoll. A MiniSec esetén pedig a broadcast üzenetek frissességét időbélyeg vagy a Bloom filteres megoldás biztosítja. Az IEEE 802.15.4 esetében kiemelendő, hogy több különböző biztonsági szint közül választhatunk, így a védelem jobban illeszthető az alkalmazáshoz, bár mint látjuk a megoldások még így is pazarlóbbak a többi protokollhoz képest.

A bemutatott megoldások biztonságának mindegyike blokk titkosítón alapul, holott a kriptográfiai fejezetben tisztáztuk, hogy a folyam titkosítás lényegesebben jobb teljesítményt nyújt, mint a blokk titkosítás. A megoldások a MiniSEC kivételével mind azelőtt születtek, hogy a folyam titkosítókat komolyabban elismerték volna.

Tulajdonképpen ez valójában inkább napjaink folyamata, így egyáltalán nem különös, hogy egyik protokoll sem veszi ezt számításba. Bár ma már ismerünk biztonságosnak vélt folyam titkosítókat, mint például a ChaCha folyam titkosító algoritmus, továbbra is igaz, hogy a folyam titkosítók esetében nagyobb körültekintést igényel azok alkalmazása, mint ugyanaz blokk titkosító segítségével. Várhatóan azonban, a hálózatbiztonság területén történt mostani hódításokkal együtt a közeljövőben, a szenzorhálózatokban is megjelennek majd a folyam titkosításra és vele integrált integritásvédelemre épülő protokollok és azok teljesítménye felülmúlja majd a jelenlegi megoldások teljesítményét.

5.2 LPWAN SZENZORHÁLÓZATOK VÉDELME

Az LPWAN (Low Power Wide Area Network) hálózatok jellemzője, hogy a kapcsolatok nagyobb távolságot fednek le, mint ahogy az a névből is következik. Tipikusan a szenzor egyetlen rádió kapcsolaton keresztül éri el a szolgáltató hálózatát, ahonnan az üzeneteit már vezetékes rendszerben továbbítják. Nincs tehát többugrásos adattovábbítás, egészen pontosan, ha lenne is, akkor ebben az adott LPWAN eszköz már az átjátszó szerepét tölti be és nem a szenzorét. Kutatás részeként ismerünk többugrásos LPWAN hálózatokat, ahol a szenzorok és átjárók egyformán LPWAN eszközök, azonban ezeket a gyakorlatban még nem alkalmazzák.

Az LPWAN technológiák viszonylag újdonságnak számítanak a kommunikáció világában, nem volt olyan rég, hogy valóban alacsony fogyasztású eszközöket tudnak gyártani, amelyek nagy távolságok áthidalására képesek. Jelen projekt keretén belül korábbi kismonográfiákban már bemutattunk ilyen LPWAN technológiákat részletesebben is. Hazánkban és általában szerte a világban egyelőre a nem túl sok megoldás közül kettővel érdemes jobban foglalkozni. Az egyik a LoRaWAN a másik a Sigfox.

A megvizsgált okosváros alkalmazások között a környezet-megfigyelés, okos közművek, okosotthon és okos közlekedés alkalmazásoknál találjuk meg ezt a technológiát. A támadások az adatok és vezérlő üzenetek meghamisítására törekednek és azt is megállapítottuk, hogy a támadónak valós motivációja van, így muszáj a védekezés.

5.2.1 A LoRaWAN hálózatok védelme

A LoRaWAN hálózati technológiáról egy korábbi kismonográfiában már részletesen írtunk [Fehér 2018], így ebben a műben csak kiemelnénk a lényeget a biztonsággal kapcsolatban.

A LoRa rádiók CSS (Chirp Spread Spectrum) modulációt használnak, amely rendkívül ellenállóvá teszi a kommunikációt a rádiós zavarásokkal szemben. A rádiós modulok rendkívül érzékenyek, így még akkor is képesek az adást fogadni, ha a zaj erőssége magasabb, mint a jel. A rádiós átvitelnél többféle spreading faktort (SF) képesek alkalmazni és a hibajavító kód is többféleképpen paraméterezhető. Minden ugyan a sebesség és így végső soron az energiafogyasztás rovására megy, de hozzásegíti az eszközt ahhoz, hogy nem csak távoli kapcsolatoknál, de a támadó által generált rádiós zavarások esetén is működőképes maradjon a kapcsolat. A LoRaWAN protokoll fel van készítve arra, hogy a kapcsolat paramétereit adaptívan igazítsa úgy, hogy azok a lehető legjobb teljesítményt nyújtsák, miközben leküzdik a rádiójel terjedését gátoló hatásokat.

A LoRaWAN estében minden egyes elküldött üzenet titkosítva és hitelesítve is van. A hitelesítéshez számlálókat is használ, amelyek számolják a legutolsó kulcscsere óta eltelt üzenetek számát, így a támadó által elfogott

üzeneteket nem lehet később visszajátszani. Amikor a LoRaWAN eszköz a hálózatra csatlakozik, megtörténik a kulcscsere, az eszközben és a LoRaWAN szerveren tárolt mesterkulcsokból egy adott viszonyra érvényes kulcsokat generálnak. Két kulcs születik, az egyik az integritásvédelmet a másik az adatok titkosítását vezérli. Előbbi a hálózati kulcs, utóbbi az alkalmazás kulcs. A kulcscserével együtt az eszköz egyedi hálózati azonosítót kap, így nem lehet az eszközt megszemélyesíteni sem. Az automatikus kulcscsere mellett létezik az az üzemmód is, amikor az eszköz az említett kulcsokat és azonosítót manuálisan állítja be. Ezt az üzemmódot csak tesztelésre érdemes használni. A jelenleg széleskörben implementált LoRaWAN 1.03 szabványban mind a hálózati kulcsot, mind pedig az alkalmazás kulcsot tulajdonképpen a LoRaWAN szerver állítja elő. A két kulcs előállítása a LoRaWAN 1.1 szabványban már markánsabban elkülönül, és a felhasználóhoz helyezi az alkalmazás kulcs előállításának szerepét. Ez azért előnyös, mert így a szervernek nem lesz tudomása a titkosított adatokról, nem lehetséges az adatszivárgás.

A rádiós LoRaWAN üzeneteket a kihelyezett átjárók fogadják. Egy adott eszköz üzenetét minden LoRaWAN átjáró fogadja, amelyhez az adás fizikailag még elérkezik, így akár más szolgáltatóhoz tartozó átjárók is. A rádiós üzenetben jelölve van, hogy az eszköz melyik szolgáltatóhoz tartozik, így az idegen szolgáltató választ, hogy vagy eldobja az üzenetet, vagy átadja a másik szolgáltató részére. Fontos, hogy az üzenet titkosítva van, így a másik szolgáltató nem ismerheti meg annak tartalmát. Az átjárók továbbítják az üzenetet a LoraWAN szerver felé. A szerver ismeri a hálózati kulcsot és azzal ellenőrizheti a kapott üzenet integritását. Amennyiben az azt mutatja, hogy az üzenet sérülésmentes, vagyis nem érte támadás, úgy az üzenetet továbbítja az alkalmazás számára. Az alkalmazás az alkalmazás kulcs segítségével fér az üzenethez. Amennyiben a szenzor fogad üzenetet, a leírtaknak megfelelően történik a titkosítás. Itt is ugyanazt a két kulcsot használjuk. Az alkalmazás csak a saját kulcsával titkosít, majd az integritásvédelmet már a LoRaWAN szerver helyezi el. A szerver csak egyetlen átjáróhoz küldi el az üzenetet, majd az továbbítja a szenzor felé azt.

A titkosító algoritmusokat tekintve a LoRaWAN az AES titkosítót használja 128 bites kulcsokkal és CTR blokkösszefűzési móddal. Az integritást védő kód 32 bites, és előállításához az AES titkosítót és a CBC blokkösszefűzést használják pont úgy, ahogy azt az IEEE 802.15.4 esetében írtuk. Az üzenetszámlálók 16 bitesek.

Összességében tehát a LoRaWAN jól védett hálózatnak tekinthető. Az eddig ismert problémák a LoRaWAN szerver szerepére irányulnak, amely még mindig nincs megfelelően leválasztva az alkalmazás szerverről a biztonságot tekintve. A felhasználók természetesen megtehetik, hogy egy újabb rétegben saját titkosítást és integritásvédelmet használnak, amely teljesen független a LoRaWAN működésétől.

5.2.3 Sigfox hálózatok védelme

A Sigfox rádiómodulok az UNB (Ultra Narrow Band) modulációs technológiát használják a rádiós üzenetek célba juttatásához. Itt egy-egy üzenet egy 100Hz szélességű csatornán kerül átvitelre. Ez a csatorna egy 192KHz tartomány része, azon belül véletlenszerűen helyezkedik el, sőt a pontos helyet még a vevő sem ismeri. Minden üzenet összesen háromszor kerül átvitelre gyors egymás után. A vevő figyeli a teljes sávot és azonosítja az egyes üzeneteket. Az átvitt teljesítmény rendkívül kis sávra koncentrálódik, így amennyiben zavarás lenne, akkor is képes túlélni azt. A Sigfox rádiók esetén amennyiben az átvitel még 8 dB értékkel a zaj felett tud maradni, az adás vehető. Hasonlóan a LoRa rádiókhoz a Sigfox rádiók is rendkívül érzékenyek. A támadó nehezen tudja az adást zavarni. Ehhez az is hozzájárul, hogy akárcsak a LoRaWAN esetén, a rádiójeleknek itt sem egyetlen állomásra kell befutniuk, hanem azokat több átjátszó veszi, és ezek mindegyike továbbítja az üzenetet a szerver felé.

A Sigfox esetében az üzenetek alapértelmezésben nincsenek titkosítva, azonban opcionálisan ez megtehető. Az adatokat integritásvédelemmel látják el és sorszámok segítik a visszajátszások elkerülését. Mivel minden adatküldés esetén három üzenet kerül kijátszásra, így a sorszámokat nem szigorúan ellenőrzik, hanem azoknak egy adott tartományban kell lenniük. A tartomány alsó értéke mindig az aktuálisan utolsó elfogadott üzenet sorszámánál eggyel nagyobb, míg a felső érték ahhoz van igazítva, hogy hány üzenetet küldhet az eszköz naponta. Az integritásvédő kódot, akárcsak a LoRaWAN esetében, itt is a szerver ellenőrzi, ezt a feladatot az átjátszók nem tudják elvégezni a megfelelő kulcs hiányában. Az integritásvédő kód hossza 16 bit. Ahogy írtuk, alapértelmezésben a Sigfox nem használ titkosítást az üzenetekhez. Ez a felhasználó feladata. A felhasználónak érdemes olyan titkosítót választania, amely illeszkedik a Sigfox üzenetekhez, azok ugyanis pontosan 96 bit hosszúak, ami másfélszerese a legtöbb blokk titkosító blokkméretének. A felhasználó megbízhat a Sigfox szolgáltatóban is, ekkor az fogja titkosítani az üzenetet.

A Sigfox protokoll az üzenetek integritásvédelméhez egy 128 bites hálózati kulcsot (NAT – Network Access Key) és az eszköz sorszámát használja (device ID). A Sigfox úgy gyártatja az eszközöket, hogy még maga a Sigfox rádió gyártója sem ismerheti meg az eszközökhöz rendelt egyedi kulcsokat. A felhasználó, amikor megvásárolja a SigFox rádiós modult, kap vele együtt egy hozzáférési kódot is (PAC – Porting Authortization Code). A hozzáférési kóddal és az eszköz azonosítójával tudja az eszközt regisztrálni a Sigfox hálózatban, és innentől kezdve használhatja azt. A PAC-t a regisztráció során megváltoztatják, nehogy ugyanazt az eszköz azonosítót kétszer is fel lehessen használni. Így minden eszköznek egyedi azonosítója és egyedi kulcsa van. Bár kicsit bonyolultnak tűnik az eljárás, azonban így az eszköz kulcsa, illetve a kapcsolathoz használt hitelesítő kulcs sohasem utazik a hálózaton, amely eleve kivéd kulcs elleni támadásokat. A kulcsot csak a Sigfox ismeri. Az integritásvédő kulcs mellett egy szintén 128 bit hosszú titkosító kulcsot (Encryption key – Ke) is tárolnak. Erre is vonatkozik, hogy csak a Sigfox ismeri azt. Amennyiben a felhasználó kéri a titkosítást, ezt a kulcsot fogja használni.

A Sigfox tehát szintén jól védett hálózatnak tekinthető. Ellentétben a LoRaWAN-nal, itt nincsen gond a Sigfox szerver és az alkalmazás szerver szerepének szétválasztásával, hiszen a Sigfox eleve a felhasználóra bízza a titkosítást. Amikor a titkosítást mégis a Sigfox végzi a felhasználó tudatában van, hogy a Sigfox megismerheti az ott átvitt adatokat.

5.3 MOBIL INTERNET, NB-IOT, LTE-CATM1 VÉDELEM

A szenzorhálózatok esetében itt egy külön fejezetben tárgyaljuk a mobil internet, NB-Iot és LTE-CatM1 technológiákat, holott az NB-IoT és LTE-CatM1 megoldásokat az LPWAN megoldások közé sorolják. Ebben a kismonográfiában azért történt meg a külön fejezetre bontás, mert egyrészt ezen három technológia közös jellemzője, hogy a hálózatot a szolgáltató egy kimondottan számára dedikált frekvenciatartományban, cellás rendszerben nyújtja, másrészt mindhárom technológia biztonsági megoldásai megegyeznek.

A mobil internet tekintetében, mint ahogy azt említettük akkor is, amikor az okosváros alkalmazásokról írtunk, LTE (Long Term Evolution) hálózatot feltételezünk. Az ide sorolható okosváros alkalmazások: környezet megfigyelés, okos közművek, egészségügy, okosotthon, okos közlekedés, önkormányzati alkalmazások. Nem meglepetés a sok átfedés a korábban említett Sigfox és LoRaWAN hálózati technológiákkal, hiszen itt egymással versengő hálózati technológiákról van szó, illetve tulajdonképpen az NB-IoT és az LTE-CatM1 is LPWAN technológia. Ami

itt megjelenik újdonságként, az az egészségügyi és önkormányzati alkalmazások. Mindkét esetben arról van szó, hogy a felhasználók a saját okostelefonjaikat használják és így érik el az alkalmazásokat. Itt valójában a mobil internet szerepel kizárólag, eddig még nem gyártottak NB-IoT vagy LTE-CatM1 képes okostelefont és ennek nem is lenne sok értelme. Akárcsak a korábban említett okosotthon biztonság esetén, itt sem fogunk kitérni részletesebben a magánszférát érintő támadásokra, ugyanis itt nem közvetlenül a szenzorhálózatok okozzák a problémákat és a védelem is kimutat a szenzorhálózatok köréből. Támadásokat tekintve itt is megállapítható, hogy elsősorban az adatok és vezérlő jelek meghamisítása áll a támadók célkeresztjében, ez ellen kell védekeznie a hálózatnak. A támadónak itt is megállapítható a motivációja, így ezen a területen is kötelező a védekezés.

A védelem leírásánál az említett technológiákat egyetlen ismertetésbe lehet összevonni, ugyanis mindhárom technológia az LTE technológia védelmével van ellátva. Ez még akkor is így van, ha működésében az NB-IoT és az LTE-CatM1 funkcionálisan kevesebb, mint a klasszikus LTE, illetve az NB-IoT nem is igazán LTE technológia, hiszen más modulációt és más sávokat használ.

5.3.1 Az LTE hálózatok védelme

Az LTE hálózatok cellás felépítésűek, azaz a hálózat cellákra van felosztva. A cellákban az eszközök a bázisállomásokhoz csatlakoznak, amelyek aztán a maghálózatra. Végül a maghálózat pedig az internetre csatlakozik. Az eszközök ugyanis az interneten keresztül kommunikálnak, ez a már ismertetett LPWAN protokolloktól eltér. Bár itt is nagyon érzékeny rádiómodulokról van szó, az alkalmazott rádiós technológia érzékeny a rádiófrekvenciás zavarásokra. A technológia nem alkalmaz külön védelmet, talán éppen azért, mert a frekvenciasáv elméletileg dedikálva van a szolgáltató számára így elvben nincs zavarás. A gyakorlatban sajnos ez nem így van, sőt mint ahogy azt írtuk is, bizonyos esetekben legális is a mobiltelefonos hálózatok zavarása. Erre a támadásra ez a technológia nincs felkészülve.

A felhasználók egy SIM (Subscriber Identity Module) azonosító modullal rendelkeznek mindhárom tárgyalt változat esetén. Az azonosító modul tárolja a felhasználó nemzetközileg is egyedi azonosítóját az IMSI-t (International Mobile Subscriber Identitity). Ezen kívül az eszköznek is van azonosítója, az IMEI (International Mobil Equipment Identity). Az IMSI 15 számjegyből áll, ebből 3-3 a szolgáltatóra vonatkozik és 9 egyedi. Ez a 9 számjegy nem a felhasználó telefonszáma. A SIM kártya nem csak az azonosító tárolásával foglalkozik, hanem a kriptográfia műveleteket is ez az eszköz végzi. A hálózatra való csatlakozáskor ez az eszköz először egy kulcscserét végez a maghálózattal, amiből aztán az adott kapcsolathoz szükséges kulcsok alakulnak ki mindkét félnél. Ez a protokoll az AKA (Authentication and Key Agreement) protokoll [Ghosh 2010]. A kulcscsere végeredményeként a titkosításhoz 256 bit hosszúságú kulcsok állnak elő, az integritásvédelemhez pedig 128 bit hosszú kulcsok. Az adat és az LTE vezérlő üzenetek külön kulcsot kapnak. Az adatok titkosítása a szolgáltató beállítása. Három különböző algoritmus közül választhat: SNOW 3G [Ekdahl 2002], AES és ZUC [Xiu-Tao 2011]. Az első és az utolsó egy svéd, illetve egy kínai folyamtitkosító. Mindhárom titkosító algoritmus képes egyben az integritásvédő kódot is előállítani.

Az LTE biztonságát összességében jobbnak mondják, mint az azt megelőző mobil generációk védelmét, azonban még mindig maradtak olyan pontok, ahol a rendszer támadható. A leggyakoribb támadások éppen ezért elhitetik a felhasználó eszközével, hogy azok egy korábbi hálózatban futnak és így azok feltörhető biztonsági algoritmusokat fognak használni. A támadások részben érintik az adatbiztonságot is, és akár sikeresek is lehetnek. A probléma nem az LTE által definiált kriptográfiai algoritmusokkal van.

6. ÖSSZEGZÉS

Ebben a kismonográfiában a szenzorhálózatokon futó okosváros alkalmazások biztonsági kihívásait és a támadásokat megállító védekezési módszereket tekintettük át. Összességében elmondható, hogy még a szenzorhálózatokban is van a támadóknak motivációja, hogy az ott átvitt adatokat és vezérlő üzeneteket megváltoztassák, vagy valótlan eszközök által valótlan forgalmat generáljanak. Emiatt az alkalmazásokat védeni kell a támadásoktól, még akkor is, ha sajnos erre sok szenzorhálózati elem, a korlátozott erőforrások miatt kevésbé képes.

Megvizsgáltunk több szenzorhálózat típust is, hogy milyen támadások lehetségesek és a ma elérhető védekezési mechanizmusok hogyan teljesítenek ezen támadások ellen. Az LR-WPAN hálózatokban a többugrásos üzenettovábbítás több támadást is lehetővé tesz, amely más hálózat típusokra nem jellemző. Ráadásul itt számíthatunk leginkább arra, hogy a szenzorok kevésbé lesznek képesek védekezni a szűkös erőforrások miatt. Mégis azt találtuk, hogy léteznek megoldások, amelyek biztonságossá teszik az LR-WPAN szenzorhálózatok használatát, akár úgy, hogy nem támasztanak túlságosan nagy követelményeket a szenzorok és a kommunikáció felé. Ráadásul a ma gyakorlatban is használt megoldások még olyan kriptográfiai algoritmusokat használnak, amelyek ugyan nem számítanak elavultnak, de léteznek már jobb megoldások, akár úgy is, hogy nem kell kompromisszumot kötni a biztonságban. Várhatóan az új kriptográfiai algoritmusok a közeljövőben beépülnek ezekbe a protokollokba is.

Az LPWAN szenzorhálózatoknál először a LoRaWAN és Sigfox protokollokat vizsgáltuk. A jelenlegi megoldások megfelelően biztonságosnak tűnnek, azonban mindkét esetben gondot jelent, hogy az üzenetek titkosítása nagymértékben függ a szolgáltatótól. A megfelelő bizalom hiányában éppen ezért ajánlott, hogy mindkét esetben az alkalmazás maga is titkosítsa az átvitt adatokat a szenzorhálózattól függetlenül. Az LTE hálózatokra épülő technológiáknál ugyan a biztonság sokat javult a korábbi generációs megoldásokhoz képest, de a hálózat még mindig nem megfelelően védett. A mobil operátorok által működtetett hálózatok nincsenek felkészülve a rádiós zavarásos támadásokra, hiszen ez eddig a dedikált frekvenciasávok miatt nem jelentett az átvitelben problémát. Problémát jelent az is, hogy a támadó képes az eszközöknek azt hazudni, hogy a 4G technológia helyett csak egy korábbi generáció érhető el. Amennyiben az eszköz átvált a korábbi technológia használatára, úgy léteznek támadások, ahol a támadó átveheti az irányítást a kommunikáció felett. Azok a protokollok, amelyek nem tudnak ilyen módon degradálódni, nem érintettek a támadásban. A problémákat az LTE által alkalmazott protokollokban kell orvosolni.

IRODALOMJEGYZÉK

[Akkaya 2005] Akkaya, Kemal, and Mohamed Younis. *„A survey on routing protocols for wireless sensor networks."* Ad hoc networks 3.3 (2005): 325–349.

[Al-Shurman 2004] Al-Shurman, Mohammad, Seong-Moo Yoo, and Seungjin Park. *„Black hole attack in mobile ad hoc networks."* Proceedings of the 42nd annual Southeast regional conference. ACM, 2004.

[Bellare 1994] Bellare, Mihir, Joe Kilian, and Phillip Rogaway. *„The security of cipher block chaining."* Annual International Cryptology Conference. Springer, Berlin, Heidelberg, 1994.

[Bloom 1970] Bloom, Burton H. *„Space/time trade-offs in hash coding with allowable errors."* Communications of the ACM 13.7 (1970): 422–426.

[Cherdantseva 2013] Cherdantseva, Yulia, and Jeremy Hilton. *„A reference model of information assurance & security."* Availability, reliability and security (ares), 2013 eighth international conference on. IEEE, 2013.

[Daemen 2013] Daemen, Joan, and Vincent Rijmen. *The design of Rijndael: AES-the advanced encryption standard.* Springer Science & Business Media, 2013.

[Ekdahl 2002] Ekdahl, Patrik, and Thomas Johansson. *„A new version of the stream cipher SNOW."* International Workshop on Selected Areas in Cryptography. Springer, Berlin, Heidelberg, 2002.

[Ephremides 1987] Ephremides, Anthony, Jeffrey E. Wieselthier, and Dennis J. Baker. *„A design concept for reliable mobile radio networks with frequency hopping signaling."* Proceedings of the IEEE 75.1 (1987): 56–73.

[Fehér 2018] Fehér G, Vida R, Jakab L. *„Vezeték nélküli szenzorhálózatok az elmélettől a gyakorlatig.* KÖFOP-2.1.2-VEKOP-15- 2016-00001 számú projekt, Nemzeti Közszolgálati Egyetem, 2018.

[Feistel 1973] Feistel, Horst. *„Cryptography and computer privacy."* Scientific american 228.5 (1973): 15–23.

[Fowler 1990] Fowler, Charles, John Entzminger, and James Corum. *„Assessment of ultra-wideband (UWB) technology."* IEEE Aerospace and Electronic Systems Magazine 5.11 (1990): 45–49.

[Ghosh 2010] Ghosh, Amitava, et al. *„LTE-advanced: next-generation wireless broadband technology."* IEEE wireless communications 17.3 (2010).

[Gisin 2002] Gisin, Nicolas, et al. *„Quantum cryptography."* Reviews of modern physics 74.1 (2002): 145.

[Hankerson 2006] Hankerson, Darrel, Alfred J. Menezes, and Scott Vanstone. *Guide to elliptic curve cryptography.* Springer Science & Business Media, 2006.

[Hong 2006] Hong, Deukjo, et al. *„HIGHT: A new block cipher suitable for low-resource device."* International Workshop on Cryptographic Hardware and Embedded Systems. Springer, Berlin, Heidelberg, 2006.

[Hsu 2018] Hsu, Jeremy. *„The Strava Heat Map and the End of Secrets."* Wired.com Security, January 2018.

[Jules 1999] Juels, Ari, and John G. Brainard. *„Client puzzles: A Cryptographic countermeasure against connection depletion attacks."* In NDSS, vol. 99, pp. 151–165.

[Karlof 2004] Karlof, Chris, Naveen Sastry, and David Wagner. *„TinySec: a link layer security architecture for wireless sensor networks."* Proceedings of the 2nd international conference on Embedded networked sensor systems. ACM, 2004.

[Kim 2009] Kim, Jongsung, and Raphael C-W. Phan. *„A cryptanalytic view of the NSA's Skipjack block cipher design."* International Conference on Information Security and Assurance. Springer, Berlin, Heidelberg, 2009.

[Koopman 2004] Koopman, Philip, and Tridib Chakravarty. *„Cyclic redundancy code (CRC) polynomial selection for embedded networks."* Dependable Systems and Networks, 2004 International Conference on. IEEE, 2004.

[Lipmaa 2000] Lipmaa, Helger, Phillip Rogaway, and David Wagner. *„CTR-mode encryption."* First NIST Workshop on Modes of Operation. 2000.

[LoRa 2015] Alliance, LoRa. *„Lorawan specification."* LoRa Alliance (2015).

[Luk 2007] Luk, Mark, et al. *„MiniSec: a secure sensor network communication architecture."* Proceedings of the 6th international conference on Information processing in sensor networks. ACM, 2007.

[MacWilliams 1977] MacWilliams, Florence Jessie, and Neil James Alexander Sloane. *The theory of error-correcting codes*. Elsevier, 1977.

[Miller 2015] Miller, Michael E. *„Car Hacking Just Got Real: In Experiment, Hackers Disable SUV on Busy Highway."* The Washington Post (2015).

[Newsome 2004] Newsome, James, et al. *„The sybil attack in sensor networks: analysis & defenses."* Proceedings of the 3rd international symposium on Information processing in sensor networks. ACM, 2004.

[Ngai 2006] Ngai, Edith CH, Jiangchuan Liu, and Michael R. Lyu. *„On the intruder detection for sinkhole attack in wireless sensor networks."* Communications, 2006. ICC'06. IEEE International Conference on. Vol. 8. IEEE, 2006.

[Perrig 2000] Adrian Perrig, Ran Canetti, Doug Tygar, and Dawn Song. *„Efficient Authentication and Signature of Multicast Streams over Lossy Channels".* In Proceedings of the IEEE Symposium on Research in Security and Privacy, pages 56–73, Oakland, CA, May 2000.

[Perrig 2001] Adrian Perrig, Robert Szewczyk, Victor Wen, David Culler, and J. D. Tygar. *„SPINS: Security Protocols for Sensor Networks".* In Seventh Annual ACM International Conference on Mobile Computing and Networks (MobiCom 2001), Rome, Italy, July 2001.

[Procter 2014] Procter, Gordon. *„A Security Analysis of the Composition of ChaCha20 and Poly1305."* IACR Cryptology ePrint Archive 2014 (2014): 613.

[Rivest 1983] Rivest, Ronald L., Adi Shamir, and Leonard M. Adleman. *„Cryptographic communications system and method."* U.S. Patent No. 4,405,829. 20 Sep. 1983.

[Rivest 1994] Rivest, Ronald L. *„The RC5 encryption algorithm."* International Workshop on Fast Software Encryption. Springer, Berlin, Heidelberg, 1994.

[Rivest 1996] Rivest, Ron. *„Rc4."* Applied Cryptography by B. Schneier, John Wiley and Sons, New York (1996).

[Rogaway 2003] Rogaway, Phillip, Mihir Bellare, and John Black. *„OCB: A block-cipher mode of operation for efficient authenticated encryption."* ACM Transactions on Information and System Security (TISSEC) 6.3 (2003): 365–403.

[Rose 2016] Anthony Rose, Ben Ramsey. *„Picking Bluetooth Low Energy Locks a Quarter Mille Away."* DEF CON® 24 Hacking Conference. Las Vegs, USA, August 2016.

[Sandhya 2012] Sandhya, S., and KA Sumithra Devi. *„Analysis of Bluetooth threats and v4.0 security features."* Computing, Communication and Applications (ICCCA), 2012 International Conference on. IEEE, 2012.

[Sastry 2004] Sastry, Naveen, and David Wagner. *„Security considerations for IEEE 802.15. 4 networks."* Proceedings of the 3rd ACM workshop on Wireless security. ACM, 2004.

[Schneier 1998] Schneier, Bruce, et al. *„Twofish: A 128-bit block cipher."* NIST AES Proposal 15 (1998): 23.

[Simon 1985] Simon, Marvin K., et al. *„Spread spectrum communications. Volume 1, 2 & 3."* NASA STI/Recon Technical Report A 87 (1985).

[Singh 2010] Singh, Virendra Pal, Sweta Jain, and Jyoti Singhai. *„Hello flood attack and its countermeasures in wireless sensor networks."* IJCSI International Journal of Computer Science Issues 7, no. 11 (2010): 23–27.

[Smith 2011] Smith, Phill. *„Comparisons between Low Power Wireless Technologies, Bluetooth low energy, ANT, ANT+, RF4CE, ZigBee, WiFi, Nike+, IrDA and NFC."* HBU Marketing document, CSR plc (2011).

[Wheeler 1997] Wheeler, D., and R. Needham. *„TEA extensions (October 1997), Also Correction to XTEA (October 1998)."* Available via: www. ftp. cl. cam. ac. uk/ftp/users/djw3.

[Win 2008] Win, Khin Sandar. *„Analysis of detecting wormhole attack in wireless networks."* In World Academy of Science, Engineering and Technology. 2008.

[Xiu-tao 2011] Xiu-tao, F. E. N. G. *„ZUC Algorithm: 3GPP LTE International Encryption Standard [J]."* Information Security and Communications Privacy 12 (2011): 031.

[Yarrkov 2010] Yarrkov, Elias. *„Cryptanalysis of XXTEA."* IACR Cryptology ePrint Archive 2010 (2010): 254.

[ZigBee 2009] Alliance, ZigBee. *„IEEE 802.15. 4, ZigBee standard."* (2009).

[Zuniga 2016] Zuniga, Juan Carlos, and Benoit Ponsard. *„Sigfox system description."* LPWAN@ IETF97, Nov. 14th (2016).

A Nemzeti Közszolgálati Egyetem kiadványa

Kiadó:
Nemzeti Közszolgálati Egyetem;
Közigazgatási Továbbképzési Intézet
www.uni-nke.hu

Felelős kiadó:
Prof. Dr. Kis Norbert rektorhelyettes
Címe: 1083 Budapest, Üllői út 82.

Olvasószerkesztő:
Kiss Eszter

Tördelőszerkesztő:
Vörös Ferenc

ISBN (pdf) 978-963-498-050-6