Processz algebrai eszközök a szenzorhálózatok biztonsági vizsgálatában

GÉMESI ROLAND*, IVÁDY BALÁZS**, ZÖMBIK LÁSZLÓ***

**BME-TMIT, gemesi@alpha.tmit.bme.hu*
***BME-TMIT, ivady@alpha.tmit.bme.hu*
****Ericsson Magyarország, BME-TMIT, laszlo.zombik@ericsson.com*

Reviewed

Kulcsszavak: biztonsági protokoll vizsgálat, szenzorhálózat kódolása, CSP, kulcs-csere

A kommunikációs és hálózati technológiák nagymértékű fejlődése, valamint a mind nagyobb fokú miniatűrizáció lehetővé tette napjainkra vezeték nélküli érzékelőrendszerek megvalósítását. A szenzor-számítógépek önszerveződő ad hoc hálózatot alakítanak ki, melyeknek biztonsága a hagyományos rendszerekhez képest nehezebben garantálható. Cikkünkben bemutatjuk, miként alkalmazhatóak a hagyományos távközlőhálózatokban bevált processz algebrai eszközök ilyen rendszerek biztonsági tulajdonságainak ellenőrzésére.

1. Szenzorhálózatok

A beágyazott és a távközlési technológiák az utóbbi években rohamos fejlődésen mentek keresztül. Mára az egyre kisebb és takarékosabb eszközökben található mikroszámítógépek viszonylag egyszerűen és olcsón kiegészíthetőek a vezeték nélküli kapcsolat képességével. Ez lehetővé teszi, hogy különálló egységeink együttműködésével összetettebb funkciók valósulhassanak meg.

A *szenzorhálózatok* számos vezeték nélküli érzékelő egységet egy közös rendszerbe kapcsolnak. A kommunikációban résztvevő egységek az érzékelőelemen kívül magukba foglalnak egy komplett mobil mikroszámítógépet, azaz energiaforrást, processzort, memóriát, valamint a vezeték nélküli kapcsolat létesítésének képességét is. A további integrációval egyetlen chipbe zsugorított, majd a méretek további csökkentésével akár porszemnyi méretű szenzorok is olcsón elérhetővé válnak. A nagyszámú és területileg elszórt egységek összekapcsolására használt vezeték nélküli technológia jelentős előnyöket kínál és új lehetőségeket nyit [1].

A parányi szenzor-számítógépek erősen korlátozott erőforrásokkal rendelkeznek, melyek csak korlátozott számítási kapacitást tesznek lehetővé. Ennek ellenére a sok egység együttműködésével az érzékelt bemeneteken elosztott jelfeldolgozási megoldásokkal akár komplex mintafelismerési feladatok is megvalósíthatóak.

A kommunikáció felépítésének automatikusan és önszerveződő módon kell végbemennie. *Mobil ad hoc hálózatoknak* nevezzük az olyan vezeték nélküli hálózatokat, melyek nem igényelnek előzőleg kiépített infrastruktúrát, vagyis előfeltételezések nélkül is képesek működni. Ilyen esetekben a központi funkciók ellátását elosztott módon kell végezni.

Biztonsági oldalról közelítve megállapíthatjuk, hogy a vezeték nélküli szenzorhálózatok számos fenyegetésnek vannak kitéve [3]. A kommunikáció nyilvános médiumon keresztül történik, melyhez rosszindulatú felek is hozzáférhetnek. A korlátozott erőforrások miatt a rendszer biztonságának védelméhez nem használhatóak a túlságosan nagy számítási igényű kriptográfiai algoritmusok. Az egyes kicsiny szenzorok kompromittálódása is jelentős fenyegetést jelent, hiszen a bennük található információk fizikai védelme nehezen oldható meg. További probléma, hogy a klasszikus távközlési rendszerekben elterjedt hitelesítési mechanizmusok gyakran igényelnek megbízható harmadik felet, melyek ad hoc környezetben nem állnak rendelkezésre.

E problémák ellenére szeretnénk, hogy rendszerünk biztonságosan és megbízhatóan működjön. Létezik néhány olyan biztonsági mechanizmus, amely a teljes önszerveződésre támaszkodik [2], ám ezek számos kérdést hagynak maguk mögött. Rendszerünket biztonságosnak tekinthetjük, ha bizonyosságot szereztünk arról, hogy az alkalmazott mechanizmusok tetszőleges támadói viselkedés esetén is megfelelő biztonságot nyújtanak.

Az elmúlt évizedekben a klasszikus kommunikációs rendszereken processz algebrai eszközökkel végzett biztonsági analízisek jelentős eredményeket hoztak. Mind támadások prezentálására, mind a megfelelő biztonság bizonyítására alkalmasnak bizonyultak.

Cikkünk célja, hogy bemutassa, miként alkalmazhatóak e módszerek szenzorhálózatok biztonságának analízisére. Bemutatjuk a *CSP (Communicating Sequential Processsses)* processz algebra alapelveit és fő szintaktikai elemeit. Áttekintjük a biztonsági vizsgálatra használt modellt, majd szenzorhálózatok biztonsági protokolljain végzett analíziseinket mutatjuk be. A cikk végén a modell további kiterjeszthetőségének kapcsán biztonságos útvonalválasztás vizsgálatának lehetőségeit vizsgáljuk meg.

2. Biztonsági protokollok és ellenőrző módszereik

A biztonságos kommunikációs rendszerekkel szemben támasztott követelmények már kiforrottnak tekinthetőek. Bár az önszerveződő hálózatok működése gyöke-

resen más szemléletet rejt, a biztonsági igények hasonlóak maradtak. Mobil ad hoc hálózatokban felmerülő biztonsági követelmények [2] közül munkánk során a titkosság, integritás, hitelesség és frissesség tulajdonságokra térünk ki.

A biztonsági protokollok olyan üzenetváltási szabályok, amelyek végrehajtásuk befejeztével valamilyen biztonsági tulajdonságot alakítanak ki. Egy támadó úgy igyekszik beavatkozni az üzenetváltásokba, hogy meghiúsítsa e kialakuló tulajdonságot. A biztonsági protokollok tervezése nagy körültekintést igényel, jóságuk bizonyítása nehéz feladat.

Hogy a kérdéshez egzaktul közelítsünk, rögzítenünk kell a támadó feltételezett képességeit. A legszélesebb körben használt támadó modell a *Dolev-Yao* támadó, amely az ellenséges médiumot reprezentálja. Hozzáfér az összes távközlési csatornához, így üzeneteket távolíthat el, lehallgathatja az üzenetváltásokat és újakat is beszúrhat. A megszerzett információk szétbontásával és a komponensek újraösszeállításával, valamint következtetésekkel új elemekhez is juthat. Egyetlen korlátja a *tökéletes-kriptográfia (perfect cryptography)* feltevés, amely szerint a kriptográfiai építőelemek tökéletes működésűek, azok próbálgatáson alapuló kompromittálódása nem lehetséges.

Nem létezik olyan általános algoritmus, melyet tetszőleges biztonsági protokollon lefuttatva, bizonyíthatná annak megfelelőségét [4]. Garantált biztonsághoz a protokollok formális analízisével juthatunk.

Az utóbbi időkben biztonsági protokollok analízisében jelentős sikereket hozott a CSP processz algebra: sokáig biztonságosnak hitt protokollokat kompromittáló új támadási viselkedéseket mutatott [5]. A módszer egy eloszott rendszer állapotgépként való modellezésére és ellenőrzésére alkalmas. Mivel e processz algebrához gépi modell ellenőrző eszköz is létezik, az analízis folyamata teljesen automatikusan, emberi beavatkozás nélkül történhet meg. A módszer megadja a támadói viselkedésének részleteit, illetve véges résztvevőből és protokollfutamokból álló rendszeren a megfelelőség bizonyítására is alkalmas.

A protokoll futamainak korlátozása jelentős gyengítésnek tűnik, azonban Lowe [5]-ban bizonyította a protokollok egy igen széles osztályára, hogy hiba esetén a támadás már néhány, meglehetősen kevés, lépés esetén is jelentkezik. A protokollok jelentős része ide tartozik, így általában már viszonylag kis állapottéren végzett ellenőrzéssel is általános állításra juthatunk.

A következőkben ezért rátérünk a CSP azon elemeinek bemutatására, melyeket a későbbi modellezésben és az analízisekben felhasználunk.

3. A CSP alapelemei

A CSP egy párhuzamos rendszerek leírására hivatott nyelv. A párhuzamos rendszerekben egyidejűleg több független folyamat (processz) is fut, melyek egymással kölcsönhatásba kerülhetnek, azaz kommunikációt folytathatnak. A CSP leírónyelv kezdetben algebrai rendszerként létezett mely lehetővé tette a különböző jelenségek formális leírását és vizsgálatát (például deadlock, livelock, nemdeterminizmus) [6].

Párhuzamos rendszerek CSP modellezése során az egyes processzek viselkedését lehet kezelni. Egy processz különféle állapotokban lehet, melyek között diszkrét események (event) hatására átmenet lehetséges. Egy processz számos eseményt ismerhet, melyek teljes halmazát a processz abc-jének (Σ) nevezzük. Egy processz egyes állapotában bizonyos eseményeket elfogadhat, vagy visszautasíthat. Egy processz *Trace*-énekk nevezzük az általa végrehajtható összes eseménysorozatok halmazát.

A $P \mathrel{\hat{=}} e \rightarrow Q$ egy olyan P processzt ír le, amely az e eseményben való részvétel után Q processzként viselkedik. A *Stop* névre hallgató processz semmilyen eseményben nem vesz már részt, míg a *Skip* állapot jelzi egy processz sikeres befejeztét.

Általános koncepció, hogy az események csatornákon jelentkeznek, ilymódon a $c.e$ esemény a c csatornán bekövetkező e eseményt jelöli. Ilymódon összetettebb konstrukciók is használhatóak, mint például *c.e.f.g.*

Az olyan szituációkat, amikor a processz többféle eseményt is elfogadhat, *választásnak* nevezzük. Egyik fajtája a külső választás *(external choice)*: a $P \Box Q$ processz a külvilágtól függően P-ként vagy Q-ként viselkedik. Ez gyakran kiegészül a *guarded alternative* operátorral, amely az egyes választásokat explicit feltételhez köti.

Az eddigi jelölések segítségével független processzek írhatóak le. Kommunikáció, vagyis közös esemény leírására hivatott a párhuzamosság *(parallel)* operátor. Ennek hatására bizonyos processzek csak egyszerre (párhuzamosan) vehetnek részt egy eseményben. Például a $P \mid\mid_e Q$ jelöléssel megadott processz a P és Q processzek olyan egyesítése, amelyben az e esemény csakis közösen hajtható végre. A teljesen független működésű processzek leírója az összefésülés *(interleave)* operátor. A $P \mid\mid\mid Q$ processz a P és Q olyan kombinációja, melyek minden eseményben csakis külön-külön vehetnek részt.

Egy operátor sokszoros alkalmazását segítik a replikált oprátorok. Ennek jelölési szisztémája a külső választás operátorra a következő: $\Box\, a : S \bullet a \rightarrow P$. Ennek jelentése, hogy bármely $a \in S$ esemény bekövetkezhet, mellyel a P processzhez jutunk.

Összetettebb rendszerek modellezéséhez szükség lehet moduláris zárt komponensek létrehozására. Lehetséges ezért az elrejtő *(hiding)* operátorral bizonyos eseményeket elrejteni a külvilág elől. A $P \setminus E$ processz P-ként viselkedik, ám az E eseményhalmaz a P-n kívülről rejtett. Megvalósítható események átnevezése is, a $P[[e \leftarrow f]]$ processz a P-vel megegyező működésű, ám annak eredeti e eseménye most kívülről f-ként látható.

Ezen operátorok segítségével lehetséges a különálló processzek leírása, valamint azok egymással való összekapcsolása. Eljuthatunk egy olyan processzhez, amely egy teljes összekapcsolt kommunkációs rend-

szert reprezentál, hordozza annak minden lehetséges eseménysorozatát.

Az általunk ellenőrzésre használt reláció a *finomítás*. Egy *Q* processz *traces finomítottja* egy *P* processznek, ha a *Q* által elvégezhető összes eseménysorozat elvégezhető *P* által is. Ezt a $P \sqsubseteq_T Q$ fejezi ki, vagyis ez esetben $Traces(Q) \subseteq Traces(P)$.

Ha *S* egy specifikációul szolgáló processz és *I* egy implementáció, a megvalósítás konformanciája ellenőrizhető a $S \sqsubseteq_T I$ finomítás ellenőrzésével. Véges állapotterű processzek közötti finomítás ellenőrzést az *FDR2* modell ellenőrző segítségével automatikusan elvégezhetjük.

4. Kommunikációs rendszerek modellezése CSP-ben

A kommunikációs protokollok modellezésére kiválóan alkalmas a CSP keretrendszer. A kommunikáció résztvevői adott szabályoknak eleget téve, a protokoll szabályai szerint működnek.

A résztvevők tehát processzek, melyek a protokoll egyes lépései során a küldés *(send)*, illetve a fogadás *(recv)* csatornákon eseményekben vesznek részt, majd ezzel új állapotba kerülnek. Egy-egy ilyen esemény reprezentálja a forrás és célentitásokat, valamint az aktuális üzenetet is. Így például az *A*-tól a *B*-be irányuló üzenetváltás során a *send.A.B.üzenet*, illetve a *recv. A.B.üzenet* események jelennek meg.

A modell az üzenetek konstruálását oszthatatlan adattagok egy véges halmazából kiindulva, a különféle kriptográfiai műveleteket megtestesítő konstrukciós operátorokkal képezi. Ilyen konstrukciós operátor például a titkosítást leíró *Encr.(Adat, Kulcs)*.

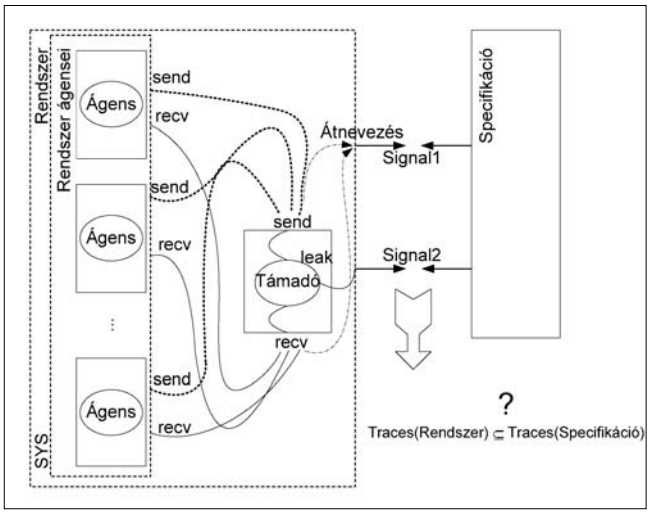
A résztvevők tehát ilyen eseményekkel érintkeznek a külvilággal. A küldés rendszerint egyértelműen adott, ám üzenet érkezése esetén gyakran előzőleg ismeretlen elemek is megjelennek a protokollfutamban. Az ilyen új adattagok típusuknak megfelelően bármilyen új értéket felvehetnek, így itt választásról van szó. Ilyen, halmaz elemei közötti választást a replikált külső választás operátor határozza meg.

A résztvevők egymástól független működésűek: a *SYS*_0 processz ezért az ágens-halmaz *(Agent)* elemeinek összefésüléssel vett egyesítéséből adódik. Ebben a küldés és fogadás események még összehangolatlanul, tetszőlegesen történhetnek meg. A kommunikáció rendjét a médium állítja elő. A *Dolev-Yao* támadó, vagyis az ellenséges médium egyben maga az összekapcsoló processz *(INTRUDER)*.

$$SYS\_0 \mathrel{\hat{=}} |||\ A : Agent \bullet A$$
$$SYS \mathrel{\hat{=}} SYS\_0\ ||_{\{send,recv\}}\ INTRUDER$$

A támadó processz meglehetősen bonyolult felépítésű. Kezdeti tudáshalmazzal rendelkezik, melyet új információkkal bővít. A lehallgatott üzenetek szétbontása után különböző dedukciós szabályokkal következtetéseket végezhet és új elemeket állíthat elő. Belső bonyolultsága ellenére kívülről is látható eseményei egyszerűen olyanok, melyekkel a többi ágens eseményeivel érintkezhet. A belső működés pusztán az analízis sebességének optimalizálása szempontjából fontos.

1. ábra Rendszerünk ellenőrzésének modellje



E Dolev-Yao médium tehát az ágensek tetszőleges üzenetküldés *(send)* eseményeivel bővíti tudáshalmazát és felajánlja az összes olyan fogadás *(recv)* eseményt, amely ezek alapján megvalósulhat. Amikor egy ágens üzenetet küld, a legegyszerűbb eset, hogy az módosítatlanul a valós címzettjéhez jut el. De a támadó felajánlja a csomag fogadását az összes többi ágensnek is. Ráadásul ha a támadó képes más ismereteiből szabályaival ugyanolyan üzenetstruktúrát képezni, a módosított csomagok fogadása is bekövetkezhet.

Az előállt rendszer tehát tartalmazza a támadóval megzavart kommunikációs folyamat összes lehetséges eseménysorozatát. A feladat ennek alapján leellenőrizni, hogy a protokoll összes kimenetele megfelelő biztonsági tulajdonságokat hordoz-e. Tehát olyan specifikációul szolgáló processzt kell készíteni, amely csak biztonságos állapotokat tartalmaz.

E cikkben csak a titkosság és autentikáció ellenőrzésére hagyatkozunk, részletesebben a [7]-ben találhatunk erről információkat. Az alkalmazott modellben a feltételezett titok kiszivárgását a támadó egy speciális eseménnyel jelezheti, ilymódon az ellenőrzés adott esemény egzisztenciájának kérdése. Autentikáció ellenőrzése során arról kell megbizonyosodni, hogy egy elem forrás általi kibocsátása valóban megelőzte-e annak vételét. Emódon itt események egymásutániságának ellenőrzéséről kell megbizonyosodnunk.

A protokoll biztonsági ellenőrzése finomítás-ellenőrzésből áll, vagyis a *specifikáció* $\sqsubseteq_T$ *implementáció* ellenőrzéséből. Ezen ellenőrzés a rendelkezésre álló FDR2 modell-ellenőrzővel automatikusan elvégezhető.

A biztonsági analíziséhez szükséges modell elkészítése hosszadalamas folyamat, mely számos hibázási lehetőséggel jár. De mivel a modellezési alapelvek különböző protokollok esetén is hasonlóak maradnak,

a folyamat tovább automatizálható. A CASPER (Compiler for the Analysis of Security Protocols) fordító képes egy biztonsági protokollból és specifikációinak viszonylag egyszerű szabványos leírásából [8] automatikusan elkészíteni az ellenőrzés alapjául szolgáló CSP leírást.

A bemenetül szolgáló fájl első része a protokoll változóit, az üzenetek leírását és a specifikációkat tartalmazza. A második rész a vizsgálandó véges rendszert, valamint a támadó kezdeti tudását definiálja. A CASPER fordító a modell ellenőrzés eredményeképp kapott eseménysorozatok értelmezésében is segítséget nyújt.

A CASPER fordító klasszikus rendszerek kulcs csere protokolljainak egyszerű és gyors analízisére született és az ismertetett rendszermodellt használja. A következőkben szenzorhálózatokhoz javasolt protokollok biztonságának vizsgálatára fogjuk e keretrendszert alkalmazni, majd ezután megvizsgáljuk a kiterjeszthetőség további lehetőségeit.

5. A SNEP protokoll analízise

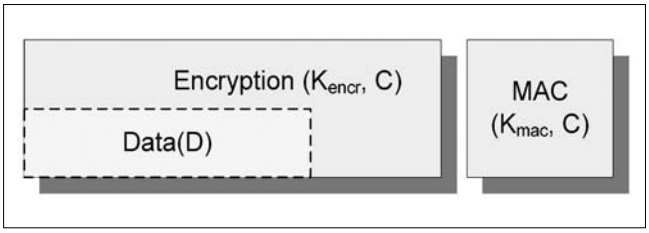
A szenzorhálózatokban, az érzékeny információk védelmét a szűkös erőforrások figyelembevételével kell biztosítani. A *SNEP (Sensor Network Encryption Protocol)* hivatott garantálni [9], hogy a kommunikáló felek között átküldött információ titkos, hiteles, és friss legyen. Ezzel akadályozza meg, hogy rosszindulatú felek hozzájuthassanak vagy módosításokat végezhessenek az átküldött adatokon. Működése feltételezi egy bázisállomás meglétét, amely minden egyes szenzorral osztott kulccsal rendelkezik. Ilyen bázisállomás gyakran része a mai szenzorhálózat-implementációknak.

A mechanizmus osztott kulcsokon alapszik. A protokoll először a közös kulcsból származtatott K_{enc} kulccsal szimmetrikus kulcsú titkosítást végez, majd egy másik származtatott (K_{mac}) kulcsot üzenet hitelesítő *(Message Authentication Code – $MAC_{K_{mac}}$)* kód képzésére használ *(2. ábra)*. Az üzenetek frissességét egy, mindkét fél által karbantartott számláló, illetve erősebb kritérium esetén egyszer használatos *nonce* elem valósít meg.

A protokoll, mely vizsgálatunk alapjául szolgált a következő:

1. $B \rightarrow A : N_b, R_b$
2. $A \rightarrow B : \{D\}\{K_{enc}, C\}, MAC_{K_{mac}, C}(N_b, \{D\}\{K_{enc}\})$

2. ábra A SNEP mechanizmus



E leírásban *B* a bázismállomás, *A* a szenzor. Első lépésben a bázis a szenzornak elküldi a frissesség alapjául szolgáló N_b nonce tagot, valamint a lekérdezést inicializáló R_b azonosítót. A kérésnek megfelelően a válaszul szolgáló információt (*D*) a szenzor az ismertetett védelemmel küldi vissza, mely során a kulcsokon kívül az aktuális *C* számlálót alkalmazza. A $\{D\}\{K_{enc},C\}$ tag a titkosított adat, míg az üzenet második tagja a generált lenyomat (MAC).

A CASPER fordító által generált modell elegendő az analízis elvégzéséhez. Az üzenetváltások az imént megadotthoz hasonló formátumban kerülhetnek megadásra. Az analízis során nem szerepeltettük az említett számlálót, hiszen a nonce által nyújtott frissesség jóval erősebb tulajdonság. A protokoll specifikációját az információ titkossága és hitelessége képezi.

Nézzük most meg a protokoll ágenseinek CSP leírását, hogy megvizsgájunk egy valódi biztonsági protokollt felépítő processzeket:

$$
\begin{aligned}
&Base(B, N_b, R_b, K_{enc}, K_{mac}, A) = \\
&\quad send.B.A.(Msg_1, < N_b, R_b >) \rightarrow \\
&\quad \Box\ D : Message \bullet \\
&\quad recv.A.B.(Msg_2, < \{D\}\{K_{enc}, C\}, \\
&\qquad\qquad MAC_{\{K_{mac}, C\}}(N_b, \{D\}\{K_{enc}\}\ ) >) \rightarrow \\
&\quad Skip
\end{aligned}
$$

$$
\begin{aligned}
&Sensor(A, D, K_{enc}, K_{mac}, B) = \\
&\quad \Box\ N_b : Nonce \bullet \Box\ R_b : Message \bullet \\
&\quad recv.B.A.(Msg_1, < N_b, R_b >) \rightarrow \\
&\quad send.A.B.(Msg_2, < \{D\}\{K_{enc}, C\}, \\
&\qquad\qquad MAC_{\{K_{mac}, C\}}(N_b, \{D\}\{K_{enc}\}\ ) >) \rightarrow \\
&\quad Skip
\end{aligned}
$$

Mindkét ágens esetében az első paraméter a saját azonosító, majd a továbbiak az egyéb szükséges információk. A leírásban a protokoll korábbi leírásának megfelelő változónevek szerepelnek. A processzek a következő módon viselkednek.

A bázis ágens első eseménye egy küldés *(send)*, mellyel az ágens egy elágazáshoz érkezik. A következő fogadás *(recv)* esemény *D* adateleme, vagyis a mérés eredménye, tetszőleges lehet. Ezt modellezi a $\Box\ D : Message \bullet$ külső választás, mely szerint az eseményben szereplő *D* adatelem a *Message* halmaz bármely eleme lehet. Ezzel a protokoll sikeresen terminálódik, vagyis a *Skip* állapotba kerül. A szenzor ágens hasonló elvek alapján működik, az előbbiek alapján az végigkövethető.

Az analízis során támadási lehetőséget nem találtunk, több résztvevőre és protokollfutamok különféle kombinációira is elvégezve az ellenőrzést állíthatjuk hogy a protokoll nem megtéveszthető. Kisebb módosításokkal különleges, nem üzemszerű esetekre is elvégeztük az analízist. Ilyen speciális eset volt a nem megfelelő nonce tag használata. A *nonce* specifikációjának megfelelően egyszer használatos, véletlen és előre nem jósolható véletlen számra van szükségünk. A szenzorokba implementált véletlenszám-generátor gyengesé-

ge fenyegetést jelent. Vizsgálatunk ugyanis kimutatta, hogy rossz nonce esetén üzenetvisszajátszáson alapuló támadás lehetséges.

Megvizsgáltuk továbbá a kulcsok kompromittálódásának hatásait is. A szenzorok fizikai védelmének hiányában fizikai hozzáférés esetén a kulcsok kinyerhetőek az egységekből. A modellben a támadó kiindulási tudáshalmazához hozzáadtuk a K_{mac}, illetve K_{enc} kulcsokat. A K_{mac} kulcs egyedüli megszerzésével a támadó nem jut új képességhez, a K_{enc} kulccsal azonban felfedheti a titkos adattagokat. A hitelesség csak mindkét kulcs birtoklása esetén sérül, ilyenkor a *D* adattag megszerezhető és megváltoztatható. A K_{mac} kulcs azonban csak tökéletes kriptográfia esetén felesleges, hiszen egy valóságos titkosítás nem szükségszerűen biztosítja az integritásvédelmet is.

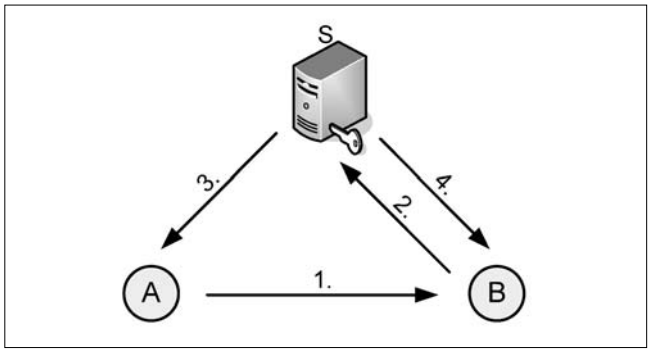
6. Szenzor kulcs-csere protokoll

Mivel a szenzorok biztonságos kommunikációja gyakran osztott kulcsok kialakítását igényli, ennek megvalósítására [9] nézzük a következő kulcs-csere protokollt. A két szenzor (*A* és *B*) egy szerver segítségével hozza létre (például a létező bázisállomás) az osztott titkot, mellyel külön-külön kulccsal (K_{AS} és K_{BS}) rendelkeznek. A protokoll lépései a következőek *(3. ábra)*:

1. $A \rightarrow B : N_A, A$
2. $B \rightarrow S : N_A, N_B, A, B, MAC_{K_{BS}}(N_A, N_B, A, B)$
3. $S \rightarrow A : \{SK_{AB}\}_{K_{AS}}, MAC_{K_{AS}}(N_A, B, \{SK_{AB}\}K_{K_{AS}})$
4. $S \rightarrow B : \{SK_{AB}\}_{K_{BS}}, MAC_{K_{BS}}(N_B, A, \{SK_{AB}\}_{K_{BS}})$

Az első lépésében az *A* szenzor kezdeményezi a protokollt, melyben egy N_A nonce tagot használ. A megszólított *B* ezután a szerverhez fordul, ahol N_B egy friss nonce és a K_{BS} a hitelesítő tag (MAC) kulcsa. A protokoll utolsó két lépése osztja ki az új kulcsot (SK_{AB}) a részvevőknek.

3. ábra Szenzor kulcs-csere



A protokoll specifikációja szerint e lépések sikeres befejezéssel a két szenzor hiteles és titokban tartott SK_{AB} kulcsokkal rendelkeznek.

Az analízis támadási lehetőséget mutatott, mely olyan esetben áll fenn, amikor *A* és *B* szenzor is kezdeményezheti a protokollt a másikkal egyidejűleg. Ez valódi fenyegetést jelent, hiszen a valóságban az *A* és *B* szenzorok egyenrangúak, nincsenek rögzített kezdeményezők.

$\alpha 1.\ I(B) \rightarrow A : N_M, B$
$\alpha 2.\ A \rightarrow I(S) : N_M, N_A, B, A, MAC_{K_{AS}}(N_M, N_A, B, A)$
$\beta 1.\ I(A) \rightarrow B : N_A, A$
$\beta 2.\ B \rightarrow S : N_A, N_B, A, B, MAC_{K_{BS}}(N_A, N_B, A, B)$
$\beta 3.\ S \rightarrow I(A) : \{SK_{AB}\}_{K_{AS}}, MAC_{K_{AS}}(N_A, B, \{SK_{AB}\}_{K_{AS}})$
$\alpha 4.\ I(S) \rightarrow A : \{SK_{AB}\}_{K_{AS}}, MAC_{K_{AS}}(N_A, B, \{SK_{AB}\}_{K_{AS}})$

A támadás leírásában két protokollfutam (α és β) összefésülését láthatjuk. A leírásban $I(A)$, illetve $I(S)$ jelenti az *A* és *S* résztvevőket megszemélyesítő támadót. A támadó a N_M nonce taggal kezdeményezi a protokollt, majd úgy juttatja el az fi futamot a sikeres befejezéshez, hogy közben a másik résztvevő valójában nem is vett részt abban a futamban. A 3. és 4. protokoll lépés lenyomatát kiegészítve a másik résztvevő nonce-ával is, a támadási lehetőség megszűnik, így a protokoll javítható.

Az előző két protokoll jó példaként szolgált biztonsági analízisünk alapjainak demonstrálására. A fenti támadási lehetőség felderítése lehetőséget adhat a hiba javítására. A SNEP protkoll esetében láthattuk, hogy a modell elég rugalmas ahhoz, hogy speciális eshetőségek (például kulcs kompromittálódás) hatásait is figyelembe vegyük.

A következőkben vizsgáljuk meg a modell nyújtotta további lehetőségeket, különös tekintettel a szenzor és ad hoc hálózatok dinamikus jellemzőire.

7. További modellezési lehetőségek

A CASPER fordító és az ismertetett modell kulcs-csere protokollok analízisére született. Ezekben, mint láttuk, minden egyes ágens meghatározott szereppel rendelkezik a protokollban. E szerep határozza meg az ágens viselkedését, az általa fogadható üzenetek és reakcióinak determinálásával.

A szenzor és ad hoc hálózatok résztvevői azonban rendszerint egyenrangú felekként viselkednek. A résztvevők egyformák, aktuális szerepük a környezettől (például szomszédok) függően alakul ki. A CSP keretrendszer lehetőséget ad ilyen dinamikus rendszerek modellezésére is.

A résztvevő ágensek azonos ismeretekkel rendelkeznek, azonban többféle üzenet vételére is képesek, ez határozza meg további működését. Mivel a vett üzenetek eseményekként jelennek meg, így események egy véges halmaza ($event_1,...,event_n$) közötti választásról van szó, melyek az ágenst szerepének megfelelő állapotba ($role_1,...,role_n$) juttatja. Ez a külső választás operátorral írható le, egy ilyen *Node* leírása a következő formájú:

$Node(A,...) =$
$(event_1 \rightarrow role_1) \Box (event_2 \rightarrow role_2) \Box ... \Box (event_n \rightarrow role_n)$

Így modellezhetővé és analizálhatóvá válnak például a biztonságos útvonalválasztó mechanizmusok. Az analízisben végrehajtásában problémát okoz a sok választás, így szerteágazó trace, okozta hatalmas állapottér. E állapottérrobbanás megelőzésére különféle optimalizálási technikák alkalmazhatóak, melyek a jövőben lerövidíthetik az analízist.

8. Összegzés

E cikkben megismerkedhettünk a szenzor és ad hoc hálózatok tulajdonságaival. Láthattuk a jellemzőikből és önszerveződő természetükből eredő biztonsági fenyegetéseiket. A CSP alapjaiba nyert betekintő után láthattuk, hogy miként alkalmazható az biztonsági protokollok analízisére. Láthattuk a módszer alkalmazását két, szenzorhálózatokhoz javasolt biztonsági protokollon és rámutattunk a módszer további lehetőségeire is.

Irodalom

[1] Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David E. Culler, Kristofer S. J. Pister (2000): System Architecture Directions for Networked Sensors

[2] Gémesi Roland, Ivády Balázs, Zömbik László: Mobil ad hoc hálózatok biztonsága. Híradástechnika, 2002/12.

[3] Buttyán Levente, Holczer Tamás, Schafier Péter: Kooperációra ösztönző mechanizmusok többugrásos vezeték nélküli hálózatokban. Híradástechnika, 2004/3.

[4] Hubert Comon, Vitaly Shmatikov: Is it possible to decide whether a cryptographic protocol is secure or not? (2001)

[5] Peter Ryan, Steve Schneider, Michael Goldsmith, Gavin Lowe, Bill Roscoe: Modelling and analysis of security protocols (2001)

[6] Steve Sneider: Concurrent and Real-time Systems. The CSP approach (2000)

[7] Peter Ryan, Steve Schneider, Michael Goldsmith, Gavin Lowe, Bill Roscoe: Modelling and Analysis of Security Protocols (2000)

[8] Gavin Lowe, Philippa Broadfoot, Mei Lin Hui: A Compiler for the Analysis of Security Protocols (2001)

[9] Adrian Perrig, Robert Szewczyk, Victor Wen, David Culler, J. D. Tygar: SPINS: Security Protocols for Sensor Networks (2001)

Hírek

A Sun Microsystems megalakította Globális Kormányzati Irodáját (Global Government Office), melynek feladata, hogy speciális megoldásokat dolgozzon ki a világ kormányzati szerveinek IT problémáira. A kormányzati szektornak szánt biztonságos megoldások központjában a Trusted Solaris operációs rendszer áll, amely az amerikai kormányzaton belül már gyakorlatilag platformszabványnak számít. Ez beépített hálózati biztonságot kínál: a személyes adatok védelmét, jobb elszámoltathatóságot, valamint a biztonsági rendszer sérülésének alacsonyabb kockázatát.

A Sun annak érdekében, hogy a kevésbé fejlett országok kormányzatait is kiszolgálhassa, speciális, az állampolgárok számát alapul vevő árképzési modellt jelentett be a Java Enterprise Systemre vonatkozóan. A rendszer ára polgáronként évi 0,33 dollártól 1,95 dollárig terjed, pontos értékét pedig két tényező határozza meg: az ország fejlettségi szintje az Egyesült Nemzetek meghatározása szerint, valamint az adott kormányzati egység alá eső polgárok száma.

A Sun Microsystems a Project Looking Glasst és Java 3D technológiáit a nyílt forráskódú közösség rendelkezésére bocsájtja, ezzel is megerősítve elkötelezettségét az élenjáró asztali technológiák mellett. Emellett két további, a Java fejlesztői közösségével együttműködve készülő nyílt forráskódú rendszert is bejelentett: a JDesktop Network Components és JDesktop Integration Components rendszereket. A JDNC leegyszerűsíti a gazdag funkcionalitású, hálózatos asztali alkalmazások fejlesztését, a JDIC pedig integrálja a többplatformos Java alapú alkalmazásokat a natív asztali környezettel.

A Projekt Looking Glass háromdimenziós asztali környezet fejlesztésére irányuló projekt, amelynek innovatív felülete intuitív 3D-környezetet kínál, ahol az ablakok átlátszóak, forgathatóak, lapozhatóak és tetszés szerint méretezhetők. A Project Looking Glass fejlesztői kiadásában az alábbi funkciók értetők el:

– 3D ablakkezelő platform, segítséget nyújt a fejlesztőknek a dokumentumok megtervezésében, a kezdeti specifikációkban és a prototípusok megvalósításában,
– Natív alkalmazásintegrációs modulok – Modul X11 alkalmazások a 3D környezetben futtatásához,
– Minta 3D ablakkezelő – tesztelési és demonstrációs célokra.