Anonimitási szolgáltatásokat nyújtó hálózati protokoll

(Anonimity Enhancing Protocol, AEP)

TDK Dolgozat

Készítette:

Tóth Gergely *tgm@mit.bme.hu*

Sili Attila *jeanlucp@mit.bme.hu*

Konzulens:

Hornák Zoltán *hornak@mit.bme.hu*

Méréstechnikai és Információs Rendszerek Tanszék

Összefoglaló

Anonimitási szolgáltatásokat nyújtó hálózati protokoll

Tóth Gergely, műszaki informatika hallgató, V. évfolyam
Sili Attila, villamosmérnök hallgató, V. évfolyam

Konzulens: Hornák Zoltán
Méréstechnika és Információs Rendszerek Tanszék

A hálózati kommunikáció elterjedésével elsőként olyan biztonságtechnikai funkcionalitást nyújtó protokollok és szabványok iránt jelentkezett igény, melyek olyan szolgáltatásokat tudnak nyújtani, mint bizalmasság, integritás védelem, vagy letagadhatatlanság. Napjainkban már számos ilyen ismert, elterjedt, megbízható és mély szakmai tudás nélkül is alkalmazható protokoll áll rendelkezésre. Ezek használatának elterjedésével a legégetőbb problémák megoldása után a kommunikációval szemben új biztonsági követelmények merültek fel az "egyszerű", titkos és hiteles kommunikáción túl. Az utóbbi időben egyre nagyobb hangsúlyt kap a személyi és személyes adatok védelme (*privacy*) és ennek megfelelően a kommunikáció során elérhető minél nagyobb szintű anonimitás. Ennek az új igénynek a kielégítésére csak részmegoldások léteznek. Munkánk célja egy olyan hálózati protokoll tervezése és specifikációja, amely ötvözi a már bevált és ismert biztonságtechnikai eljárásokat, alapmegoldásokat, valamint a már meglevő anonimitási módszereket és általánosan alkalmazható külön hálózati rétegben megvalósított anonimitási szolgáltatásokat nyújt.

Számos olyan hálózati kommunikációs réteg ismert, amely biztonságtechnikai funkcionalitást nyújt. Ilyen az Internet világában ismert SSL, vagy utódja a TLS, valamint az SSH, de hasonló a WAP világában használt WTLS is. Ezek általában kliens-szerver kommunikációt tételeznek fel, melyek során biztosítják a PKI alapú kulcscserét, titkosítást, integritás-védelmet valamint a szerver és esetleg kliens oldali azonosítást is. Ezen rétegek mintájára terveztük az anonimitási funkciókat is megvalósító protokollunkat.

Anonimitás biztosítására a két legjobban elterjedt módszer a vak-aláírás és az álnév. Az eredetileg David Chaum által internetes anonim fizetésre kidolgozott háromszereplős (bank, ügyfél, szolgáltató) **vak-aláírás** protokoll (DigiCash) teljes, visszakövethetetlen anonimitást biztosít, melynek általánosított verziója számos más területen is – mint például az anonim szavazás – használható. A másik alapmódszer az **álnév (***pszeudo-identitás*), amely esetében az alany identitása védelmet élvez, de visszaélés esetén az álnév használója visszakövethető.

Az általunk bemutatott protokoll olyan hálózati rétegre épül, amely biztosítja a megbízható adatátvitelt, valamint az alap biztonságtechnikai funkcionalitást (titkosítás, integritás-védelem, szerver és kliens oldali azonosítás), ugyanis ezen funkciók implementálásával nem foglalkoztunk.

A protokoll specifikálásához először egy megfelelő leíró nyelvet választottunk. Ezek után ennek a leíró nyelvnek a segítségével a következő fázisban a különböző esetekben megjelenő lehetséges absztrakt kommunikációs partnerek és azok lehetséges tevékenységeit írtuk le. Végül a konkrét esetekhez tartozó hálózati kommunikációt specifikáltuk, melyhez a protokoll az XML szabvány leíró nyelvet használja.

Az így elkészült protokoll megoldási módszereket kínál az anonimitással együtt felmerülő visszaélések elleni védekezésre és azok utólagos felderítésének támogatására is.

Az elméleti specifikálás mellett a protokollt JAVA környezetben TCP/IP, valamint SSH2 kommunikációs réteg felett meg is valósítottuk. Példának, demonstrációs alkalmazásnak a hallgatói rendszer néhány anonimitást igénylő funkcióját választottuk.

1. Tartalomjegyzék

2. Bevezetés

Az elmúlt évtizedek során a számítástechnika, a hardver, a szoftver valamint a kommunikáció területén tapasztalható rohamos fejlődés lehetővé tette az ügyviteli rendszerek egyre nagyobb fokú elektronizálását és integrálását. Ezen tendenciák bebizonyították, hogy a hálózati kommunikáció napjaink kulcsfontosságú eleme.

A kommunikációval kapcsolatos alapkövetelmény a végpontok közötti megbízható és minél gyorsabb **adattovábbítás**. Ennek megvalósítására már régóta léteznek különböző szabványosított megoldások. A kommunikációs protokollok rétegszerkezetű felépítésnek köszönhetően a felsőbb rétegek már olyan szolgáltatásokat nyújtanak, melyeket a felhasználó mélyebb szakismeret nélkül is tud hatékonyan használni.

A kommunikációval kapcsolatos további követelményként merül fel **biztonság**. Itt olyan problémákra kell megoldást nyújtani, mint a távoli azonosítás (az van-e a kapcsolat túloldalán, akinek mondja magát), a bizalmasság kérdése (csak azok tudják elolvasni az üzenetet, akiknek szánták), az integritás-védelem (biztosan ugyanaz érkezzen meg, mint ami el lett küldve) vagy a letagadhatatlanság (ha valami megérkezik, arról bizonyítható legyen, hogy a feladója azt küldte).

Ezekre a feladatokra mára már léteznek matematikailag megalapozott, a gyakorlatban is bevált módszerek, melyeket szabványosított felületen keresztül lehet elérni. Bár az ISO-OSI hétrétegű modell nem tüntet ki külön biztonságtechnikai réteget (a fenti funkciókat alapvetően az adatkapcsolati illetve az alkalmazói rétegbe sorolja), a gyakorlatban bebizonyosodott, hogy a kommunikációs hierarchiában szükség van egy olyan réteg kialakítására, mely kifejezetten csak biztonságtechnikai feladatokat lát el. Fontos, hogy ilyen rétegek léteznek. Alapvető tulajdonságuk, hogy nem egy konkrét rejtjel algoritmuson alapulnak, hanem a protokollon belül több módszer is cserélhető, annak érdekében, hogy hosszú távon megfelelő, fejleszthető megoldást lehessen nyújtani.

A biztonságtechnikai réteg használatának elterjedésével a kommunikációval szemben napjainkban egyre gyakrabban jelenik meg követelményként az **anonimitás**. Miután a kommunikáció során megbízható az adatkapcsolat, lehallgathatatlan és módosíthatatlan az adatfolyam, és mindkét fél biztos lehet a másik fél személyazonosságában, felmerül annak az igénye, hogy a kommunikációs fél minél kevesebben tudjon meg partneréről, ne ismerje meg személyi és főleg személyes adatait, ne tudja meg személyazonosságát. Ezzel együtt természetesen rögtön felmerül a visszaélések elleni védekezés szükségessége is, melyet szorosan együtt kell kezelni az anonimitási módszerekkel.

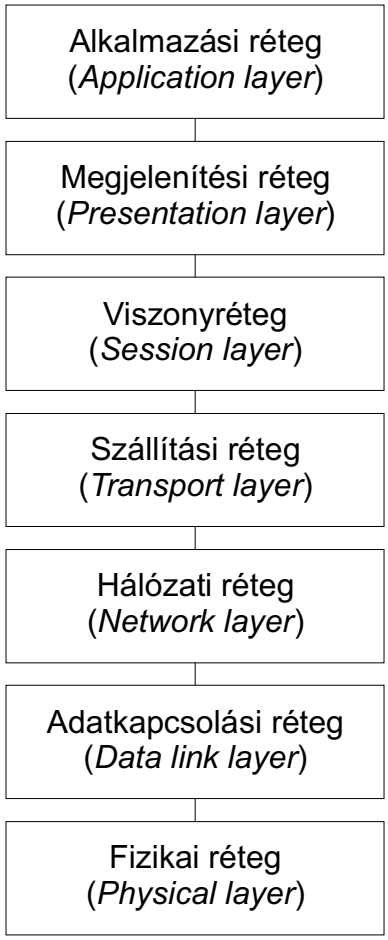
Az anonimitás különböző szintjeinek megvalósítására már léteznek módszerek. Ugyanakkor hiányzik az az összefoglalás, egységesítés, ami a biztonságtechnika más területein már megtörtént. Ugyanígy szükség van egy olyan hálózati rétegre, vagy tulajdonképpen egy "fekete dobozra", mely az anonimitási funkcionalitást anélkül látja el, hogy a felhasználónak komoly ismeretekre kellene szert tennie az anonimitási módszerek működéséről.

Ez a TDK munka azt tűzte ki célul, hogy egy olyan **hálózati protokollt** írjon le, mely felhasználja az absztrakt biztonságtechnikai réteg által nyújtott funkcionalitást, azaz követelményként előírja, hogy alatta a réteghierarchiában szerepel egy biztonságtechnikai réteg, és erre támaszkodva valósít meg az anonimitási módszereket egységes, könnyen használható formában.

3. Ismertető

Ez a fejezet röviden bemutatja a megvalósított protokollhoz felhasznált, már meglevő módszereket. Ezek egyrészt a biztonságtechnika, másrészt pedig az anonimitás témaköréből származnak.

A bemutatott módszerek a kommunikációs réteghierarchia különböző szintjein helyezkednek el. Az ISO-OSI hétrétegű referenciamodell az itt ismertetése kerülő eljárások számára nem tüntet ki külön önálló réteget, azonban a gyakorlati megvalósítások során ez szükségesnek bizonyult, mivel a biztonságtechnikai protokoll kidolgozása nagy szakértelmet kíván, ellenőrzésük szakma általi elfogadtatásuk hosszú folyamat.



1. ábra – **Az ISO-OSI hétrétegű referenciamodell**

3.1 Biztonságtechnikai réteg

Régen a számítógép-biztonság veszélye sokkal kisseb volt, hiszen akkor még a számítógépek központi helyeken, zárt szobákban voltak elhelyezve, ahol fizikailag is védve voltak, így a bejutás kívülről nem volt egyszerű. A legnagyobb veszélyeket a belső jogosult személyek általi visszaélések jelentették.

A 90-es években a számítástechnika radikálisan kezdett változni. Egyre több számítógép jelent meg magánhivatalokban, laboratóriumokban melyek irányítása már volt nem központosított. Az Internet megjelenésével a számítógépeket már távolról is el lehetett érni,

ami további biztonsági kérdéseket vetett fel, mint például hogy tényleg az a jogosult személy használja-e a számítógépet (azonosítás kérdése), akinek mondja magát. Az Internet amellett, hogy lehetővé tette a gyors információcserét, magával vonta a számítógépes logikai károkozók (vírusok, férgek) rohamos terjedését is.

Emellett az internetes kommunikációnak további gyengeségei melyeket lehetnek, melyeket kihasználva rosszindulatú emberek komoly károkat tudnak okozni. Ezek a gyengeségek alapvetően a következők lehetnek:

- **Lehallgatás** (*Eavesdropping*)

  Az üzenetek lehallgathatók, anélkül hogy azt bárki is észrevenné. Az üzenetek elvesztik bizalmasságágukat.

- **Módosítás** (*Tampering*)

  Az üzenetek az átvitel során meghamisíthatóak. A fogadó félhez már hamis adatok juthatnak. Az üzenetek elvesztik integritásukat.

- **Megszemélyesítés** (*Impersonation*)

  A kommunikáció során az üzenetek módosítás nélkül kerülnek továbbításra, viszont a fogadó fél nem lehet biztos az üzenetküldő személyazonosságában. Az üzenetek elvesztik letagadhatatlanságukat.

- **Visszajátszás** (*Replay*)

  A kommunikáció során lehallgatott üzeneteket a támadó újra elküldheti a kommunikációs partnereket, akik azt mint hiteles üzeneteket kezelik, hiszen azok is voltak.

A fent felvázolt problémák elindították azt a tendenciát, mely a kommunikációval szemben alapkövetelményként nevezi meg az azonosítás, integritás védelem, bizalmasság és a letagadhatatlanság biztosítását. A tendencia következménye a hálózati architektúrában egy olyan külön réteg bevezetése, mely a fent említett problémákra ad megoldási lehetőséget. Ez a réteg az úgynevezett **biztonságtechnikai réteg**.

3.1.1 Rejtjelezési alapmegoldások

Hogyan valósították meg az azonosítást, az integritás védelmet, a bizalmasságot és a letagadhatatlanságot?

A biztonságtechnika egyik alappillérének tekintett titkosítás (kriptográfia) történelme szinte az írás történelmével egyidős. Napjainkra alapvetően két típusa alakult ki. A szimmetrikus kulcsú titkosítás esetén a kommunikáló partnerek ugyanazon titkos kulccsal végzik a műveleteket. Emiatt a kulcscsere lényeges pontja ennek a típusnak. A másik fajtája a titkosításnak a nyilvános kulcsú kriptográfia. Ennek lényege, hogy minden kommunikáló fél rendelkezik egy titkos kulccsal, amit csak ő ismer, és egy hozzá tartozó nyilvános kulccsal, amit pedig bárki megtudhat. A titkos kulccsal bekódolt üzenetet csak a nyilvános kulccsal lehet visszafejteni, és fordítva. Ennek megfelelően amennyiben valakinek üzenetet akarnak küldeni, elég azt a fogadó nyilvános kulcsával kódolni, és azt csak ő tudja ezután visszafejteni. Ennek a megoldásnak az előnye, hogy a kulcscsere nem probléma, az egyetlen kérdés, amit meg kell oldani, a nyilvános kulcsok hiteles terjesztése.

Mint már említettük a bizalmasság (erősen) összefüggésben áll a lehallgatás fogalommal. Ahhoz hogy más illetéktelen személyek ne tudják elolvasni az üzeneteket, titkosítják azt. Az

üzenetek titkosítva (például szimmetrikus rejtjelezési technika alkalmazásával) mennek át a nyilvános csatornán és a visszafejtéshez szükséges információt (kulcsot) csak a két egymással kommunikáló fél ismeri. Ilyen elterjedt szimmetrikus titkosítási algoritmus a DES, 3DES, AES stb.

Az integritás megőrzését digitális aláírással lehet biztosítani. Az üzenetet, mielőtt küldésre kerülne, további információval egészítenek ki, melyet hash (*one-way hash*) értéknek vagy üzenet kivonatnak (*message digest*) neveznek. A hash érték kiszámításához egyirányú matematikai függvényt használnak (például SHA1, MD5 stb.), azaz az üzenetből a hash érték számítása viszonylag egyszerű, ugyanakkor egy adott hash értékhez kapcsolódó üzenet előállítása nagyon bonyolult feladat. A hash értéket a küldendő üzenetből egy titkos kezdeti érték (*intialization vector, IV*) felhasználásával készítik, aminek eredményeképpen az így kapott MAC-ot (*message authentication code*) egy támadó már nem tudja kiszámítani. A fogadó fél az üzenethez csatolt hash MAC értéket ki tudja számítani, így amennyiben a számított és a küldött MAC egyezik, akkor az üzenetet nem módosították, azaz az integritása megmaradt.

A digitális aláírás megoldást nyújt a letagadhatatlanság biztosítására is. Az egyszer már aláírt üzenetet nehéz letagadni, hiszen az aláírás a bizonyíték. Mivel az ehhez szükséges titkos kulcsot csak a küldő ismeri, az aláírást más nem készíthette, hacsak a titkos kulcsot nem szerezték meg illetéktelen személyek.

Az azonosítás biztonsági igazolványokkal (*certificate*, például X.509) történik. Emellett szükség van egy harmadik megbízható félre is, mely garantálja a kommunikáló felek személyazonosságának valódiságát, aki az igazolványokat kibocsátotta.

A visszajátszás elleni védekezés legelterjedtebb módszere az időbélyeg (*timestamp*) alkalmazása, mely az üzenet elküldésének pontos idejét tartalmazza. Ennek valódiságát ellenőrizve eldönthető, hogy az érkezett üzenetek az eredetiek-e vagy a visszajátszottak.

3.1.2 Gyakorlati megvalósítások

A fejezet további része a gyakorlatban megvalósított és elterjedt biztonságtechnikai protokollokat és azok tulajdonságait ismerteti:

- **SSL (Secure Sockets Layer)** [SSL3]**:**

  Az SSL protokoll TCP/IP felett és az alkalmazás réteg (HTTP vagy IMAP) között helyezkedik el. Tulajdonságai:

  - Szerver és kliensazonosítás.
  - Rejtjelezési algoritmusok kiválasztásainak lehetősége, melyeket a szerver és kliens egyaránt támogatnak.
  - Szimmetrikus kulcsú rejtjelezés használata a megosztott titkos adatok továbbítására.
  - Integritás védelem.

- **SSH (Secure Shell)** [SSH]**:**

  Az SSH protokoll biztonságos távoli hozzáférést biztosit. Tulajdonságai:
  - Szerver és kliensazonosítás.
  - Rejtjelezési algoritmusok kiválasztásainak lehetősége, melyeket a szerver és kliens egyaránt támogatnak.
  - Szimmetrikus kulcsú rejtjelezés használata a megosztott titkos adatok továbbítására.
  - Integritás védelem.
  - Tömörítési lehetőség.

3.2 Anonimitási módszerek

Több szereplős kommunikáció során anonimitási funkció alatt bizonyos információ (pl. személyazonosság) a többi szereplők egy része előtti elrejtését értjük. Napjaink tendenciája az egyre növekvő igény minél nagyobb szintű anonimitás elérésére. Egyre nagyobb hangsúlyt kap a személyi és személyes adatok védelme, ezek kezelése már törvényileg szabályozott.

Az ISO-OSI hétrétegű modellben az anonimitási funkcionalitás megvalósítására nem különítettek el különálló réteget, ilyen feladatokat az applikációs rétegnek kell megoldania. Felhasználói szempontból azonban szükség van egy olyan különálló rétegre, mely az anonimitás során felmerülő problémákat megoldja, különböző feladatokhoz különböző anonimitási szintet tud biztosítani, szabványosított, cserélhető és konfigurálható módszereket használ, valamint amelynek alkalmazásához nem szükséges az anonimitási problémák és megoldási módszerek mély szakmai ismerete.

Fontos megemlíteni, hogy az anonimitási igények nagyrészt a kommunikáció egy speciális formájára, az úgynevezett kliens-szerver kommunikációra vonatkoznak. Ebben az esetben a szerver általában egy intézmény, vagy szervezet, az úgynevezett **szolgáltató**, a kliens pedig egy személy, az úgynevezett **alany**. A kommunikáció során az alany az intézmény valamely szolgáltatását akarja igénybe venni. A kapcsolat során követelmény, hogy a szolgáltató hitelesen azonosítsa magát, azaz ő nem anonim, ugyanakkor az alany a lehető legmagasabb szintű anonimitás elérésére törekszik. Ezzel együtt természetesen a szolgáltató részéről felmerül a visszakövethetőség igénye, azaz a visszaélések elkerülésének és nyomonkövethetőségének biztosítása.

3.2.1 Anonimitási szintek

A következőkben az anonimitás öt szintje kerül ismertetésre, melyek az alany egyre növekvő anonimitási szintjét képviselik.

3.2.1.1 Anonimitás hiánya

A leggyengébb anonimitási szint az anonimitás hiánya, amikor is a kommunikáció során az alany **személyazonossága** (*real identity*) ismert. Ebben az esetben a visszaélések nyomon követése nem anonimitási feladat.

3.2.1.2 Visszakövethető alany

A következő anonimitási szint során az alany már rendelkezik egy **álnévvel** *(pseudo identity*), azaz a kapcsolat során nem valós személyazonosságát használja, ugyanakkor a rendszer lehetőséget nyújt arra, hogy az álnévhez tartozó személyazonosságot bárki megtudhassa. Ebben az esetben az álnév-személyazonosság megfeleltetést és a szolgáltatás nyújtását ugyanaz a szereplő végzi, így visszaélés esetén az elkövető személyazonosságát a szolgáltató saját hatáskörében megtudhatja.

Erre tipikus példa a UNIX rendszerek felhasználói neve, melyről a legtöbb esetben kideríthető a felhasználó személyazonossága. A BME hallgatói szerverén, az URAL2-n a szerzők álnevei *tg227* és *sa246*, de a `finger` paranccsal bárki megtudhatja személyazonosságukat.

3.2.1.3 Korlátozottan visszakövethető alany

Magasabb anonimitási szinten szétválik az álnév-személyazonosság megfeleltetés és a szolgáltatás nyújtása. Az előbbi elvégzését egy úgynevezett **megbízható harmadik fél** (*trusted third party, TTP*) végzi, míg a szolgáltatást továbbra is az előzőekben ismertetett szolgáltató végzi.

Fontos fejlemény, hogy a normális üzemmenet során a szolgáltató az alanyt csak álnevén keresztül ismeri, azaz személyazonossága ismeretlen. Ugyanakkor visszaélés esetén a megbíztató harmadik fél bevonásával a szolgáltató kérheti az alany személyazonosságának felfedését.

Ezt az anonimitási szintet valósítja meg a lentebb ismertetett **Álnév módszer**.

3.2.1.4 Visszakövethetetlen alany

Ezen az anonimitási szinten a szolgáltató semmilyen módon nem tudhatja meg az alany személyazonosságát. Ugyanakkor az alanynak rendelkeznie kell egy speciális **engedéllyel** (*permission*), mely egyértelműen igazolja, hogy jogosult a szolgáltatás igénybevételére. Ilyen engedélyek kibocsátására csak az **engedélyező hatóság** (*permission authority*) jogosult. Fontos követelmény, hogy az engedélyből ne lehessen az alany személyazonosságára következtetni, ugyanakkor azt a tényt, hogy az alany valamilyen szolgáltatás igénybevételére engedélyt kapott, tárolni kell.

Ezt az anonimitási szintet valósítja meg a lentebb ismertetett **Vak-aláírás módszer**.

3.2.1.5 Teljes anonimitás

Teljes anonimitás szint esetén a szolgáltatás igénybevétele során az alany személyazonossága nem ismert, a szolgáltatónak nincs is lehetősége annak kiderítésére, a szolgáltatást bárki igénybe veheti.

Ennek a szintnek két változatát különböztetjük meg:

- Az első esetben a szolgáltató a szolgáltatásról bizonyos adatokat tárol (szolgáltatás leírása, igénybevétel időpontja, esetleges információk az alanyhoz menő adatkapcsolati útról stb.). Tipikus példa erre a webes böngészés, melynek során a legtöbb szolgáltató a megtekintett oldal nevét, az időpontot és az alany IP címét tárolja.
- A második esetben a szolgáltató semmilyen adatot nem tárol a szolgáltatásról. Ilyen például egy tipikus Windows-os környezetben egy program indítása.

3.2.2 Anonimitási módszerek

Az anonimitási módszereket általánosságban PET-eknek (*Privacy Enhancing Technology*) szokás nevezni. Ezek közül a következőkben két különböző anonimitási szintet megvalósító anonimitási módszer kerül bemutatásra. Az anonimitási módszerek alkalmazásához egy megbízható biztonságtechnikai réteg használata szükséges.

A módszerek során bemutatásra kerülnek a kommunikációban résztvevő szereplők, megfogalmazásra kerülnek a módszer megvalósításával kapcsolatos követelmények és

bemutatjuk a módszer alkalmazásának fő lépéseit. A módszerek konkrét megvalósítására a Protokoll specifikáció, a Megvalósított rendszer fejezetek és a Melléklet mutat majd példákat.

3.2.2.1 Álnév módszer

Az itt bemutatásra kerülő módszer az előzőekben ismertetett **korlátozottan visszakövethető alany** anonimitási szintjét valósítja meg. Ezt a módszert széles körben alkalmazzák hozzáférés-védelmi megoldásoknál.

3.2.2.1.1 Szereplők

A kommunikációban három eltérő feladatkörű szereplő különböztethető meg:

- **szolgáltató**: szolgáltatásokat nyújt az alanynak, úgy hogy őt csak álneve alapján ismeri,
- **alany**: anonim módon, álneve felhasználásával szolgáltatásokat vesz igénybe a szolgáltatótól,
- **megbízható harmadik fél**: az alany álnév-személyazonosság megfeleltetését hivatott tárolni, a szolgáltató felé az alany anonim azonosítását végzi.

3.2.2.1.2 Követelmények

A módszerrel szemben támasztott következő követelmények biztosítják a szereplők számára a megcélzott anonimitási szint elérését:

- az alany személyazonosságát és álnevét csak a megbízható harmadik fél tudja összekötni,
- a szolgáltató az alanyt csak álneve alapján ismeri,
- a szolgáltató számára az alanyt a megbízható harmadik fél anonim módon azonosítja,
- a szolgáltató visszaélés esetén megtudhatja az alany személyazonosságát.

3.2.2.1.3 A módszer használata

A módszer használata három jól elkülöníthető részre osztható:

- Az első **előkészítő lépés** során az alany regisztrálja magát a megbízható harmadik félnél. Ennek során az alany azonosítja magát, megkapja álnevét, a megbízható harmadik fél pedig eltárolja az álnév-személyazonosság megfeleltetést, valamint az alany álnév alapú azonosításához szükséges információt. Ezen lépés során kerül megállapításra, hogy az alany milyen szolgáltatónál milyen jogosultságokkal rendelkezik. Ez a lépés a módszer alkalmazása során csak egyszer zajlik le.

  Az előkészítő lépés a biztonságtechnikai rétegtől mindkét fél hiteles azonosítását, integritás védelmet, valamint a kommunikáció bizalmasságát igényeli.

- A következő lépés maga az álnév segítségével történő szolgáltatás igénybevétel a szolgáltatónál, az **álnév használata**. Ennek során az alany közli a szolgáltatóval az álnevét, az igényét, valamint olyan adatokat, melyek alapját őt a megbízható harmadik fél azonosíthatja. Ezekez a szolgáltató továbbítja a megbízható harmadik félnek, aki ezek alapján eldönti, hogy egyrészt tényleg az az alany akarja a szolgáltatást igénybe

venni, aki jogosult az álnév használatára, valamint az alany jogosult-e a szolgáltatás igénybevételére. Amennyiben a megbízható harmadik fél engedélyezi a műveletet, a szolgáltatás teljesítendő.

Ez a lépés a biztonságtechnikai rétegtől az alany-szolgáltató kommunikáció során a szolgáltató hiteles azonosítását, integritás védelmet, valamint bizalmasságot; a szolgáltató-megbízható harmadik fél kommunikáció során mindkét fél hiteles azonosítását, integritás védelmet, valamint bizalmasságot igényel.

- **Visszaélés** esetén valamely külső szerv (pl. nyomozóhatóság) bevonásával a szolgáltató kötelezheti a megbízható harmadik felet az alany személyazonosságának felfedésére.

3.2.2.2 Vak-aláírás módszer

Ez a módszer az előzőekben ismertetett **visszakövethetetlen alany** anonimitási szintet valósítja meg. Ezt a módszert eredetileg David Chaum írta le a DigiCash internetes anonim fizetési rendszer alapjaként [CHAUM] (ebben az esetben az alany a vásárló, a szolgáltató a bolt és az engedélyező hatóság a bank), azonban a módszer tovább általánosítható szélesebb körű alkalmazásokhoz, mint például anonim szavazás (az alany a szavazó, a szolgáltató gyakorlatilag az "urna", az engedélyező hatóság pedig az, aki a szavazócédulákat kibocsátja).

3.2.2.2.1 Szereplők

A kommunikációban három eltérő feladatkörű szereplő különböztethető meg:

- **szolgáltató**: szolgáltatásokat nyújt az alanynak, azonban nincs lehetősége az alany személyazonosságának megismerésére. A szolgáltatás feltétele, hogy az alany rendelkezzen az engedélyező hatóság által kibocsátott speciális engedéllyel,
- **alany**: anonim módon, az engedélyező hatóság által részére kibocsátott engedély segítségével szolgáltatásokat vesz igénybe a szolgáltatótól,
- **engedélyező hatóság**: az alany számára olyan speciális engedélyt bocsát ki, mellyel az alany a szolgáltatónál szolgáltatásokat tud igénybe venni.

3.2.2.2.2 Követelmények

A módszerrel szemben támasztott következő követelmények biztosítják a szereplők számára a megcélzott anonimitási szint elérését:

- a szolgáltatás egyszeri igénybevételét az engedélyező hatóságnak engedélyezni kell az alany számára, az engedélyezés során az alany személyazonossága ismert,
- az engedéllyel a szolgáltatást bármely szolgáltatónál igénybe lehet venni,
- az engedélyből nem lehet az alany személyazonosságára következtetni,
- engedélyt csak az engedélyező hatóság tud kibocsátani, annak hitelességét bármelyik szereplő el tudja dönteni, az engedély meghatározza a szolgáltatás típusát,
- a szolgáltatás igénybevételénél a szolgáltató nem ismeri az alany személyazonosságát,
- az engedély többszöri felhasználásának megakadályozása végett, a szolgáltatás nyújtása előtt az engedélyt az engedélyező hatóság ellenőrzi,
- az adott szolgáltatónál történt engedély felhasználását az engedélyező hatóság lekönyveli.

3.2.2.2.3 Használat

A módszer használata három jól elkülöníthető részre osztható:

- A módszer használatának első lépése az **engedélykérés**. Ennek során az alany közli az engedélyező hatósággal a kívánt szolgáltatás típusát. A kommunikáció során az alany személyazonossága ismert. A kérés és annak lekönyvelése után az alany megkapja az igényelt engedélyt.

  A módszer lényege abban rejlik, hogy az alany valójában nem a tényleges engedélyt, hanem csak annak az alapját, egy úgynevezett **elő-engedélyt** (*pre-permission*) kap meg. Ezután az alany még végrehajt bizonyos módosításokat a kapott engedélyen, hogy az eredmény és az engedélyező hatóságtól kapott engedély közötti kapcsolatot ne lehessen megteremteni. Fontos, hogy a módosítások után is el lehessen dönteni az engedély hitelességét, valamint, hogy az alany az engedélyező hatóság elő-engedélye nélkül ne tudjon igazi engedélyt generálni.

  Ez a lépés a biztonságtechnikai rétegtől mindkét fél hiteles azonosítását, integritás védelmet, és bizalmasságot igényel.

- A második lépés a **szolgáltatás igénybevétele**. Ennek során a szolgáltató és az alany megegyezik a szolgáltatásban és az ahhoz szükséges engedélyekben, ez a tárgyalás nem része a módszernek. Miután mindez megtörtént, az alany megmondja a szolgáltatónak a kívánt szolgáltatás leírását, valamint átadja a szükséges engedélyeket. Amennyiben az engedélyeket az engedélyező hatóság ellenőrizte (lásd 3. lépés), úgy a szolgáltatás teljesítendő.

  Ez a lépés a biztonságtechnikai rétegtől a szolgáltató hiteles azonosítását, integritás védelmet, valamint bizalmasságot igényel.

- A harmadik lépés során az engedélyező hatóság **ellenőrzi**, hogy a szolgáltató által benyújtott engedélyek nem kerültek-e már felhasználásra, és ha nem, akkor azok felhasználását **lekönyveli** a konkrét szolgáltatónál.

  Ez a lépés a biztonságtechnikai rétegtől mindkét fél hiteles azonosítását, integritás védelmet és bizalmasságot igényel.

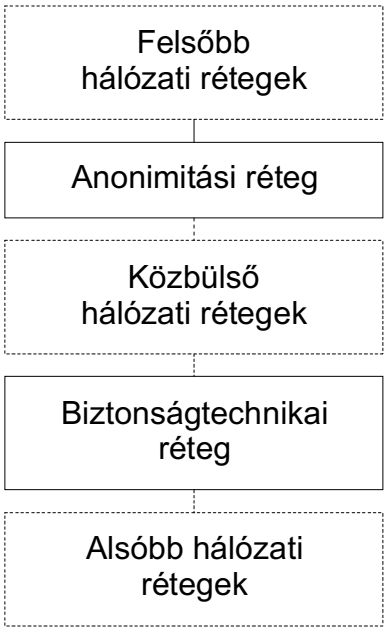
4. Rendszerkoncepció

Ez a fejezet röviden összefoglalja és bemutatja a specifikált rendszert, ismerteti a következőkben részletesen leírt rétegek elhelyezkedését és viszonyukat más rétegekkel, valamint bemutatja a protokoll absztrakt szereplőit és funkciójukat.

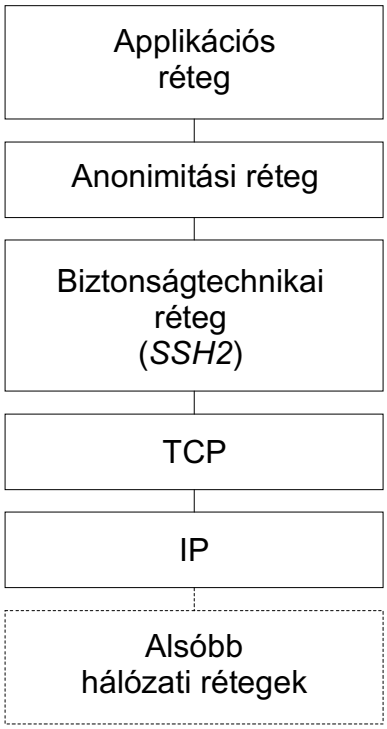
4.1 A protokoll rétegei

Az elemzések és a gyakorlati alkalmazások alátámasztják a különálló **biztonságtechnikai réteg** létjogosultságát. Hasonló megfontolások alapján a specifikált rendszerben az **anonimitási** funkcionalitás ellátására is külön réteg került bevezetésre, mely a biztonságtechnikai réteg által nyújtott szolgáltatásokat felhasználja.

A konkrét megvalósítás során az applikációs réteg (a felhasználói program) közvetlenül használja az anonimitási réteg által nyújtott szolgáltatásokat. Alattuk helyezkedik el ebben a konkrét példában az SSH2 réteg, mely közvetlenül a TCP és IP protokollokra épül.



2. ábra – **Általános rétegszerkezet**



3. ábra – **Egy konkrét rétegszerkezet**

A rétegszerkezet kialakításánál fontos, hogy a biztonságtechnikai réteg az anonimitási alatt helyezkedik el, támogatva azt. Ennek következtében az anonimitási réteget használva mindkettő funkcionalitása együtt érhető el.

4.2 Általános koncepció

Az anonimitási módszerek elemzése során azok között számos hasonlóság vehető észre:

- Háromszereplős módszerek, ahol a résztvevők a lehető legmagasabb anonimitási szint elérésére törekedő **alany**; a szolgáltatásokat nyújtó **szolgáltató**, aki az alany személyazonosságát nem ismeri; és végül valamilyen **anonimitási hatóság**, aki a kellő anonimitási szintet biztosítja, és akiben minden másik fél megbízik.
- A kommunikáció sémája hasonló:

  Előkészítő szakasz

  A kommunikáció az alany és az engedélyező hatóság között zajlik, melynek során mindkét fél hitelesen azonosítja magát.

  ① Az alany **kéri** az anonimitási hatóságtól a személyazonosságának, a kívánt anonimitási szintnek és a kívánt szolgáltatás típusának egyeztetését.

  ② Az anonimitási hatóság **kiállítja** az alanynak az anonimitási okmányokat, majd **lekönyveli** a kívánt szolgáltatás igénybevételéhez a jogosultságokat.

  Használat

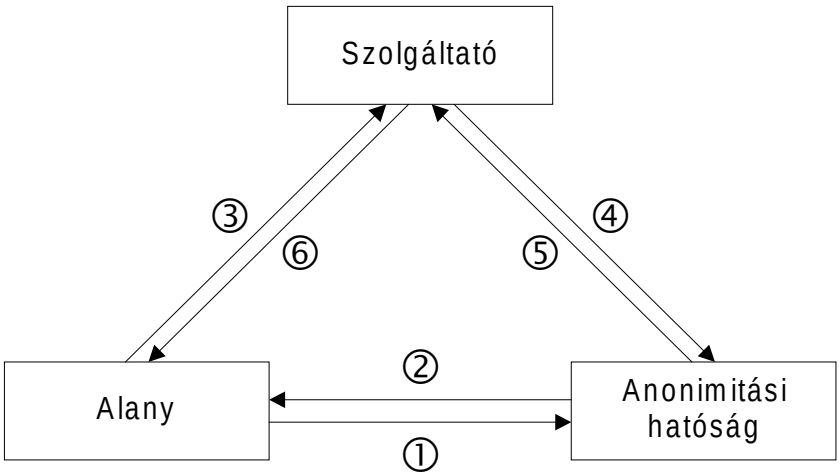
  A kommunikáció során az alany a kívánt szintű anonimitást használja, míg a többi fél hitelesen azonosítja magát.

  ③ Az alany **kéri a szolgáltatást** a szolgáltatótól és átadja az anonimitási okmányokat.

  ④ A szolgáltató az okmányok **hitelesítését kéri** az anonimitási hatóságtól.

  ⑤ Az anonimitási hatóság **ellenőrzi** az anonimitási okmányok hitelességét és **lekönyveli** a szolgáltatás igénybevételét a szolgáltatónál.

  ⑥ A megfelelő hitelesítés esetén a szolgáltató **teljesíti** a kívánt szolgáltatást.



4. ábra – **Általános anonimitási kommunikációs séma**

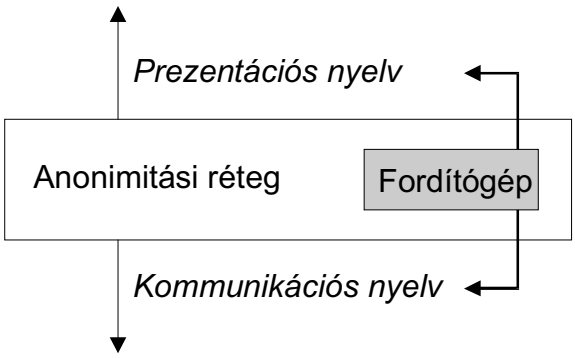
5. Protokoll specifikáció

Ez a fejezet mutatja be teljes részletességgel a megtervezett protokollt, ismerteti a felhasznált prezentációs nyelvet, a kommunikáció nyelvét, leírja az absztrakt szereplők tulajdonságait, a közöttük lefolyó kommunikáció üzeneteinek szintaxisát, az adatstruktúrákat.

A **prezentációs nyelv** az a nyelv, amelynek segítségével a protokollban használt adatstruktúrákat írjuk le. A anonimitási réteg feletti entitások a funkcionalitást ezeknek az adatstruktúráknak a használatával érik el, azaz a prezentációs nyelvet a felsőbb szintű rétegek használják. A prezentációs nyelv szintaxisa a biztonságtechnikából jól ismert TLS szintaxisához és a C programozási nyelvéhez hasonlít. A protokollban használt adatstruktúrák leírásához olyan nyelvet kerestünk, mely megfelel az igényeinknek, használata könnyű, ugyanakkor eléggé elterjedt és könnyen érthető, hogy mások hosszas beletanulás nélkül is tudják alkalmazni.

Az anonimitási rétegből lefelé induló kommunikáció nyelve a **kommunikációs nyelv**. A protokoll architektúrájának kialakításakor az volt a cél, hogy a későbbi verziók során a prezentációs nyelv lényegében állandó maradjon (esetleg azt majd ki lehet egészíteni), azonban a kommunikációs nyelvet igény szerint ki lehessen cserélni, módosítani lehessen.

Ennek megfelelően a kommunikációs nyelv, nem a nyelv definícióján keresztül került megadásra, hanem egy fordítógépen keresztül, mely a prezentációs nyelvet fordítja le a kommunikációs nyelvre. Ez a fordítógép tekinthető akár egy doboznak, mely a két nyelv közötti kölcsönös megfeleltetést biztosítja.



5. ábra – **A prezentációs és a kommunikációs nyelv kapcsolata**

A kommunikációs nyelv megválasztása során arra fektettük a hangsúlyt, hogy annak értelmezése könnyű legyen, legyen meg hozzá a támogatottság a különböző programozási nyelveken, támogassa a strukturáltságot, és hogy az ember számára viszonylag könnyen kezelhető legyen. A jelen megvalósítás során a választás az XML [XML] nyelvre esett, mert rendelkezik a fent említett tulajdonságokkal.

A fejezet további részeiben bemutatásra kerülnek a szereplők és a közöttük zajló kommunikáció lehetséges formái. A kommunikációs diagrammok bemutatják az üzenetek lehetséges típusait és azok lehetséges sorrendjét.

Végül leírásra kerülnek a prezentációs nyelv segítségével az üzenetekhez felhasznált adatstruktúrák, ismertetésre kerül az azok mögötti rejlő információ.

5.1 Prezentációs nyelv

Ez az alfejezet mutatja be a prezentációs nyelvet, mely a protokoll definíciója során használt adatstruktúrák és ezzel együtt a felsőbb rétegekkel való kommunikáció leírására szolgál. A prezentációs nyelv a protokollban használandó adatok illetve adatstruktúrák formázási képet kívánja leírni emberhez közeli szinten, ezzel biztosítva a könnyebb áttekinthetőséget és megértését.

Irodalomkutatás során arra a következtetésre jutottunk, hogy a C programozási nyelv és a hozzá hasonló, a TLS [TLS] protokollban használt prezentációs nyelv (mely hasonlít az SSL és a WTLS [WTLS] protokolloknál használt prezentációs nyelvre is), használata elterjedt, és ezek a céljainknak is megfelelnek.

Arra törekedtünk, hogy egy olyan nyelvet használjunk mely széles körben ismert, ezzel is követve a jól bevált szabványok használati tendenciát.

Ugyanakkor éltünk néhány kiegészítéssel is, melyek a nagyobb szintű felhasználó-barátság elérése miatt voltak szükségesek.

5.1.1 Általános szabályok

5.1.1.1 Megjegyzések

Az itt bemutatott prezentációs nyelv **kisbetű-nagybetű érzékeny** (*case-sensitive*).

A következő foglalt szavakat nem lehet adattípus- vagy változó-névként használni:

```
basic, case, default, extension, list, select, struct, type, value
```

A prezentációs nyelv alap adatmérete a **bájt**, ami egy oktett, azaz 8 bit.

A megjegyzéseket a `/*` és a `*/` betűk zárják közre.

5.1.1.2 Általános BNF definíciók

Ez a rész azokat a BNF (Backus Naur Form) definíciókat tartalmazza, melyek a további definíciók alapjául szolgálnak.

```
<kisbetű> ::= "a" | "b" | "c" | "d" | "e" | "f" | "g" | "h" |
   "i" | "j" | "k" | "l" | "m" | "n" | "o" | "p" | "q" | "r" |
   "s" | "t" | "u" | "v" | "w" | "x" | "y" | "z"
<nagybetű> ::= "A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" |
   "I" | "J" | "K" | "L" | "M" | "N" | "O" | "P" | "Q" | "R" |
   "S" | "T" | "U" | "V" | "W" | "X" | "Y" | "Z"
<szám> ::= "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" |
   "9" | "0"
<egyéb> ::= "_"
<kiskezdés> ::= <kisbetű> | <egyéb>
<nagykezdés> ::= <nagybetű>  | <egyéb>
<minden> ::= <kisbetű> | <nagybetű> | <szám> | <egyéb>
<változónév> ::= <kiskezdés> {<minden>}
<adattípusnév> ::= <nagykezdés> {<minden>}
<változódeklaráció> ::= <adattípusnév> <változónév> ";"
```

5.1.2 Adattípus fajták

Ez az alfejezet röviden bemutatja a prezentációs nyelvben használható adattípus fajtákat.

5.1.2.1 Alap típus

Alap típus alatt olyan adattípust értünk, mely nem épül már típusfajtákra, azaz értékkészlete vagy explicit módon vagy szabályok útján adott.

```
<basic> ::= basic <változónév>;
```

Ez a protokollspecifikáció a következőkben definiált alap típusokat határozza meg:

```
basic RString;
basic HEXBlob;
basic HEXNumber;
basic DECNumber;
```

5.1.2.1.1 RString

Az `Rstring` típus tetszőlegesen hosszú ASCII karaktersorozat tárolására alkalmas, azonban csak az ASCII 32-127 intervallum közötti karakterekből állhat, kivéve a ’ # ’, ’ \ ’, ’ ” ’, ’ < ‘, ’ > ’ karaktereket.

5.1.2.1.2 HEXBlob

A `HEXBlob` típus tetszőleges ASCII karaktersorozat hexadecimális reprezentációja. Az ábrázolás minden ASCII karakterhez egy kétkarakteres hexadecimális számot rendel, ami a "*network byte order*" szerint kódolt, azaz bájtonként is és összességben is az *MSB* van baloldalt, az *LSB* pedig jobboldalt.

```
<hex> ::= "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" |
    "8" | "9" | "A" | "B" | "C" | "D" | "E" | "F"
<hex2> ::= <hex><hex>
<HEXBlob> ::= <hex2> {<hex2>}
```

Példa:

```
"41"
```

ez az ASCII 65-os karakterkódnak felel meg ami nem más mint az “A” karakter

5.1.2.1.3 DECNumber

A `DECNumber` tetszőlegesen hosszú pozitív vagy negatív egész számok decimális formában való ábrázolására alkalmas.

```
<DECNumber> ::= ["+" | "-"] <szám> {<szám>}
```

Példa:

```
"-52"
```

5.1.2.1.4 HEXNumber

A `HEXNumber` a `HEXBlob` egy tovább értelmezett válfaja, tetszőlegesen hosszú pozitív vagy negatív egész számok kettes komplemensű hexadecimális formában való ábrázolására alkalmas. Amennyiben egy pozitív szám első bitje 1 lenne az ábrázolás folytán (és így a kettes

komplemens ábrázolás miatt negatívnak értelmeződne), úgy szám ábrázolása elé `"00"`-t írunk.

```
<HEXNumber> ::= <HEXBlob>
```

Példa:

```
"00"   ⇨ (decimálisan)    0
"80"   ⇨ (decimálisan) –128
"0080" ⇨ (decimálisan)  128
"FF"   ⇨ (deximálisan)   -1
```

5.1.2.2 Kiegészítés

Konkrét esetekben szükség lehet a protokoll specifikációjában megadott adattípusok kiegészítésére, arra a lehetőségre, hogy ahhoz többlet-információt fűzzünk. Erre való a kiegészítés típus. Ennek konkrét formája mindig a kommunikációs nyelvtől függ. XML esetén ez egy szintaktikailag helyes (*well-formed*) XML karaktersorozat.

```
<extension> ::= Extension <változónév>;
```

5.1.2.3 Lista típus

A lista típus már előre definiált típusú változók sorrendhelyes felsorolása.

```
<list> ::= list "<" <adattípusnév> ">" <adattípusnév> ";"
```

Példa:

```
enum {RED, GREEN, BLUE} Color;
list<Color> ColorList;
```

Ennek megfelelően a `ColorList` típus egy `Color` típusú változókból álló lista.

5.1.2.3.1 Műveletek:

- Deklarálás

```
ColorList cl;
Color c1;
Color c2;
```

- Értékadás: értékadás során a listaelemeket kapcsos zárójelek között vesszővel tagolva kell felsorolni (`{..,..}`).

```
cl={c1, c2};
```

- Hozzáférés: listaelemekhez a szögletes zárójel (`[]`)operátorral lehet hozzáférni, az elemek sorszámozása nullával kezdődik.

```
c2=cl[1];
```

- Hosszlekérdezés: a `.length` operátorral lehet a lista hosszát (elemeinek számát) lekérdezni.

```
c1.length
```

5.1.2.4 Felsorolt típus

A felsorolt típus olyan adattípust jelöl, melynek lehetséges értékei egy előre definiált halmazból kerülnek ki. A halmaz elemeihez lehetséges számértékeket rendelni (minden elemhez eltérőt), ezeket nem kell sorfolytonosan kiosztani.

```
<enumszám> ::= "(" <szám>{<szám>} ")"
<enumméret> ::= <enumszám>
<enumelemérték> ::= <nagybetű>{<nagybetű>}
<enumhalmazelem> ::= <enumelemérték> [<enumszám>]
<enum> ::= enum "{"
        <enumhalmazelem>
        {"," <enumhalmazelem>}
        ["," [<enumméret>]]
    "}" <adattípusnév> ";"
```

A felsorolt típus reprezentációja annyi helyet (bájtot) foglal, amennyi az `<enumméret>` által reprezentált szám decimális leírásához kell. Amennyiben ezt az opcionális tagot elhagyjuk, úgy a halmazelemeknek megfeleltetett számok legnagyobbja lesz a mérvadó.

```
enum {RED(0), GREEN(2), BLUE(4), (99) } Color;
```

Ebben az esetben a RED-hez rendelt érték a 2 lesz, a felsorolt típus pedig 2 bájtot fog elfoglalni (a 99-et két bájton lehet decimálisan leírni).

5.1.2.4.1 Műveletek:

- Deklarálás

```
Color c1;
```

- Értékadás

```
c1=RED;
```

5.1.2.5 Struktúra típus

A struktúra típus több tetszőleges előre definiált típusú változó összefogása egy komplexebb típussá. Szintaktikája többé-kevésbe a C nyelvben használt struktúra típuséval egyezik meg.

A struktúra típusnak van egy speciális változata, amit variáns struktúra típusnak neveznek. A variáns struktúra típus több variáns részt tartalmaz. Minden variáns részhez tartozik egy felsorolt típus. A felsorolt típus által felvehető lehetséges értékekhez különböző változódefiníció halmazokat lehet rendelni. Ennek megfelelően, a variáns részből csak azok a változók érhetők el, amelyet a felsorolt elem értéke meghatároz.

```
<selectorbefejezés> ::= "{"
    <változódeklaráció>
    {<változódeklaráció>}
    "}" ";"
<caseselector> ::= case <enumérték> <selectorbefejezés>
<defaultselector> ::= default <selectorbefejezés>
<variánsrész>=<változódeklaráció>
    select "(" <változónév> ")" "{"
        {<caseselector>}
        [<defaultselector>]
    "}" ";"
```

A *case* ággal konkrétan meg lehet határozni a felsorolt elem kiválasztandó értékét, a *default* ág az összes eddig nem specifikált érték esetében lesz érvényes. Ennek megfelelően a komplett struktúra definíciója a következőképpen alakul:

```
<struct> ::= struct "{"
        <változódeklaráció>
        {<változódeklaráció>}
        {<variánsrész>}
    "}" <adattípusnév> ";"
```

Példa:

```
/* S1, S2, S3 már definiált */
enum {APPLE, ORANGE, PEACH, BANANA} VariantTag;
struct {
    VariantTag variant;
    select (variant) {
        case APPLE {
            S1 s1;
        }
        case APPLE {
            S2 s1;
        }
        default {
            S3 s1;
        }
    }
} VariantStruct;
```

5.1.2.5.1 Műveletek:

- Deklarálás

  ```
  VariantStruct v;
  ```

- Hozzáférés: belső változókhoz a "." operátor segítségével lehet hozzáférni.

  ```
  v.variant=APPLE;
  ```

5.2 Kommunikációs nyelv

Ez az alfejezet az anonimitási rétegből az alatta elhelyezkedő réteghierarchia felé irányuló kommunikációt írja le. Ennek a rétegstruktúrának tartalmaznia kell a biztonságtechnikai réteget. A réteghierarchiával szemben támasztott követelmények ismertetése után kerül sor a kommunikációs nyelv, a fordítógép ismertetésére.

5.2.1 Követelmények

A kommunikációs nyelv megválasztása során arra fektettük a hangsúlyt, hogy az szabványosított legyen, annak értelmezése könnyű legyen, legyen meg hozzá a támogatottság a különböző programozási, támogassa a hierarchikus, strukturáltság adatábrázolást, és hogy az ember számára viszonylag könnyen kezelhető legyen (*human readable*).

A jelen megvalósítás során a választás az XML nyelvre esett, mert rendelkezik a fent említett tulajdonságokkal.

Ugyanakkor vannak az XML nyelvnek hátrányai is, legfőképpen a relatív hosszúsága, azaz ugyanazt az adatmennyiséget sokkal kompaktabb módon is át lehet küldeni egy csatornán egy megfelelő nyelv segítségével, ami alacsonykapacitású kommunikáció esetén rendkívül előnyös lenne. A protokoll továbbfejlesztése során ennek a problémának a részletes kielemzése mindenképpen kulcsfontosságú feladat lesz.

5.2.2 Specifikáció

A kommunikációs nyelv arra hivatott, hogy a prezentációs nyelven definiált változókat közvetítse az anonimitási réteg alatti réteghierarchia számára.

A kommunikációt üzenetek sorozataként kell értelmezni, ahol minden egyes üzenet egy-egy összetett változó és minden egyes üzenet típus egy-egy adattípusnak felel meg.

5.2.2.1 Általános megjegyzések

Legyen ***v*** egy változó, legyen a típusa `T(`***v***`)` és az értéke `V(`***v***`)`. Az XML sorokat a # jel vezeti be. Amennyiben sor ## jelekkel kezdődik, úgy az a jelek utáni szöveg XML-é konvertált változatást jelenti.

A legtöbb esetben egy XML struktúra belsejében a változó nevéből annak típusa is kikövetkeztethető, így annak feltüntetésére az XML kimenetben nem szükséges. Amennyiben ez nem lehetséges, úgy a változónak megfelelő XML tag-ot el kell látni egy `type` attribútummal, ami a változó típusát határozza meg. Ez a helyzet áll fent kiterjesztések alkalmazásánál, valamint az XML-é fordítandó struktúra legkülső változójának esetében.

5.2.2.2 Konverziós szabályok

5.2.2.2.1 Alap típus

A fordítás során keletkező XML kimenet a következő:

```
# <v value="V(v)" />
pl.
    RString v1="a1";
    #<v1 value="a1" />
```

A változó értékét a prezentációs nyelv szerint kell értelmezni.

5.2.2.2.2 Lista típus

A fordítás során keletkező XML kimenet a következő:

```
# <v>
## 1. elem
## 2. elem
## ...
# </v>
```

A lista elemei úgy konvertálódnak XML-é a `<v></v>` részek között, mintha `element_i` nevű változók lennének, ahol `i` a listaelem sorszáma.

```
pl.
    list<DECNumber> decList;
    decList lista={3,2,1};
    #<lista>
    #  <element_0 value="3" />
    #  <element_1 value="2" />
    #  <element_2 value="1" />
    #</lista>
```

5.2.2.2.3 Felsorolt típus

A fordítás során keletkező XML kimenet a következő:

```
# <v value="1" />
```

A változó értéke a felsorolt típus által felvehető elemekhez rendelt számérték decimális reprezentációja, ami maximálisan annyi bájtot foglal el, amennyit a felsorolt típus definíciója erre megszab.

```
pl.
    enum {RED(0), BLUE(1), GREEN(50), (99)} Color;
    Color c1=GREEN;
    # <c1 value="50" />
```

5.2.2.2.4 Struktúra típus

A fordítás során keletkező XML kimenet a következő:

```
# <v>
## belső változók XML reprezentációja
# </v>
```

Ha a struktúra variáns részt is tartalmaz, akkor csak azoknak a belső változóknak kell feltüntetni, amelyeket a variáns rész felsorolt típusának értéke meghatároz.

```
# pl.
    enum {RED(0), GREEN(1), (9)} Color;
    struct {
        Color c1;
        select (c1) {
            case RED {
                RString s;
            }
            case BLUE {
                DECNumber n;
            }
        }
    } StructType;
    StructType sVar;
    sVar.c1=RED;
    sVar.s="string";
    # <sVar>
    #   <c1 value="0" />
    #   <s value="string" />
    # </sVar>
```

5.2.2.2.5 Kiegészítés

XML alapú kommunikációs nyelv esetén a kiegészítés szintaktikailag helyes XML karaktersorozatot tartalmaz.

```
# <v>
## a kiegészítés tartalma
# </v>
pl.
    extension ext;
    extension="<message value=\"Alert\" type=\"AlertMsg\" />"
    # <ext>
    #   <message value="Alert" type="AlertMsg" />
    # </ext>
```

5.3 Üzenetek

Az anonimitási réteg a funkcionalitását a hálózati kommunikáció során elküldött üzenetek által valósítja meg, melyek a konkrét módszereket és elveket tükrözik. Hogy az összes anonimitási módszert egységesen lehessen kezelni, egy olyan üzenethalmaz és üzenetsorrend került kialakításra, ami támogatja a további kiegészítések lehetőségét és a jelenleg feldolgozandó összes módszert képes megvalósítani. Ennek megfelelően ez az alfejezet bemutatja a specifikált anonimitási réteg által használt egységes üzeneteket, azok elküldésének sorrendjét és tartalmát.

A következőkben először egy általános, absztrakt kommunikációs megoldás kerül ismertetésre, mely aztán a konkrét esetekben alkalmazható a különböző anonimitási szintet biztosító módszerek kivitelezésére.

Ezután kerül bemutatásra a korlátozottan visszakövethető alany anonimitási szintet megvalósító álnév módszer esetében alkalmazható konkrét kommunikációs struktúra, végül pedig a visszakövethetetlen alany anonimitási szintet megvalósító vak-aláírásos módszerhez tartozó üzenetek.

5.3.1 Előfeltételek

A biztonságtechnikai réteggel szembeni követelmények kielégítése érdekében szükséges, hogy a kommunikációban résztvevő összes fél rendelkezzen olyan digitális azonosítókkal, melyek őt a többi fél számára egyértelműen megnevezi (*identification*) és olyan kulcsokkal és kriptográfiai képességgel (pl. nyilvános kulcsú kriptográfiai képesség és a hozzá tartozó nyilvános-titkos kulcs-pár), melyek segítségével az azonosítása (*authentication*) megtörténhet.

Ennek a gyakorlati megvalósítása során a legtöbb esetben bevezetésre kerül egy újabb szereplő, a *Certification Authority* (CA), melynek szerepe hamisíthatatlan **igazolványok** kiállítása. Ez a legtöbb esetlen a PKI (*Public Key Infrastructure*) – mint a legelterjedtebb azonosítási módszer – használtából ered. A módszerek lényege, hogy minden résztvevő rendelkezik egy **nyilvános** és egy hozzá tartozó **titkos kulcs-párral**. A nyilvános kulcsot és az úgynevezett megkülönböztető nevet (*Distinguishing Name*, DN) tartalmazó igazolványt (*Certificate*) a CA állítja ki, és titkos kulcsával aláírja, így annak hitelességét is biztosítja. Ennek felhasználásával lehet a kommunikációs feleket megnevezni, és az ehhez tartozó titkos kulccsal azonosítani.

Ennek megfelelően az itt leírt protokoll specifikáció feltételezi, hogy a kommunikációban résztvevő összes fél rendelkezik a fent leírt típusú igazolvánnyal és a hozzá tartozó titkos kulccsal, és az igazolványok hitelességének tanúsítására léteznek megbízható CA-k.

5.3.2 Általános módszer

A különféle anonimitási módszerek tanulmányozása során felismert összefüggések alapján a kommunikáció során három fő szereplőtípus tüntethető ki: az **alany**, a **szolgáltató** és az **anonimitási hatóság**.

Az anonimitási módszerek alkalmazása során a kommunikációt két szakaszra lehet osztani. Az első szakasz, az **előkészítő szakasz** során az alany személyazonossága ismert, ő csak az anonimitási hatósággal áll kapcsolatban és ennek a szakasznak a feladata a következő szakasz paramétereinek egyeztetése, az igényelt anonimitási szint, a később igénybe veendő szolgáltatás típusának és egyéb körülményeknek megbeszélése.

A második szakasz során, melyet **szolgáltatás igénybevételének** nevezünk, az alany az előbb egyeztetett körülményeknek megfelelően vesz részt a kommunikációban, azaz a kívánt anonimitási szinten, és a már megbeszélt szolgáltatás típust veszi igénybe.

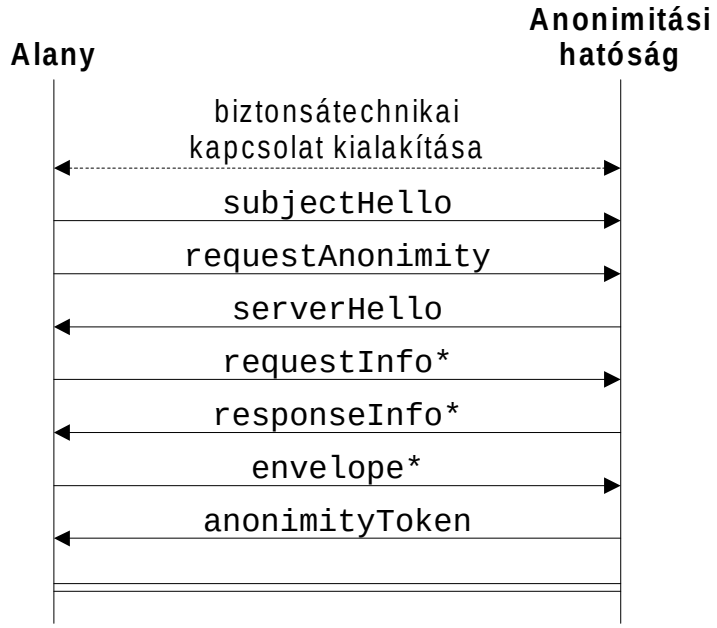
Ez a fejezet a protokollban szereplő összes üzenetet ismerteti. Az egyes konkrét esetekben nem biztos, hogy az igényelt funkcionalitás ellátásához az összes üzenetre szükség lesz, és emiatt elképzelhető, hogy azok a tényleges kommunikáció során nem is fognak megjelenni.

Ennek megfelelően a következőkben bemutatásra kerülő kommunikációs diagrammokban az opcionális, nem kötelező üzenetek neve után egy csillag (*) szerepel. Ezek az üzenet a kívánt funkcionalitástól függően elhagyhatók. A többi üzenet minden alkalommal megjelenik a konkrét kommunikációban.

A hálózati párbeszéd kialakulása során a réteghierarchia alulról felfelé épül ki, ennek megfelelően mivel az anonimitási réteg a biztonságtechnikai felett helyezkedik el, először a biztonságtechnikai funkcionalitás jelenik meg. Ennek megfelelően az anonimitási rétegbeli kommunikáció során a biztonságtechnikai rétegtől már igényelhetünk szolgáltatásokat.

5.3.2.1 Előkészítő szakasz

Az előkészítő szakasz feladata, hogy az alany által igényelt anonimitási szint és szolgáltatáshoz szükséges anonimitási okmányokat. Ennek során a kommunikáció csak az alanyra és az anonimitási hatóságra terjed ki.



6. ábra – **Az általános előkészítő szakasz kommunikációs diagrammja**

Az első lépés a kellő biztonságtechnikai kapcsolat kiépítése. Ez a szakasz a biztonságtechnikai rétegtől mindkét fél hiteles azonosítását, az integritás védelm és a bizalmasság biztosítását feltételezi.

A kommunikáció üzenetei a következő lényegi adatokat hordozzák:

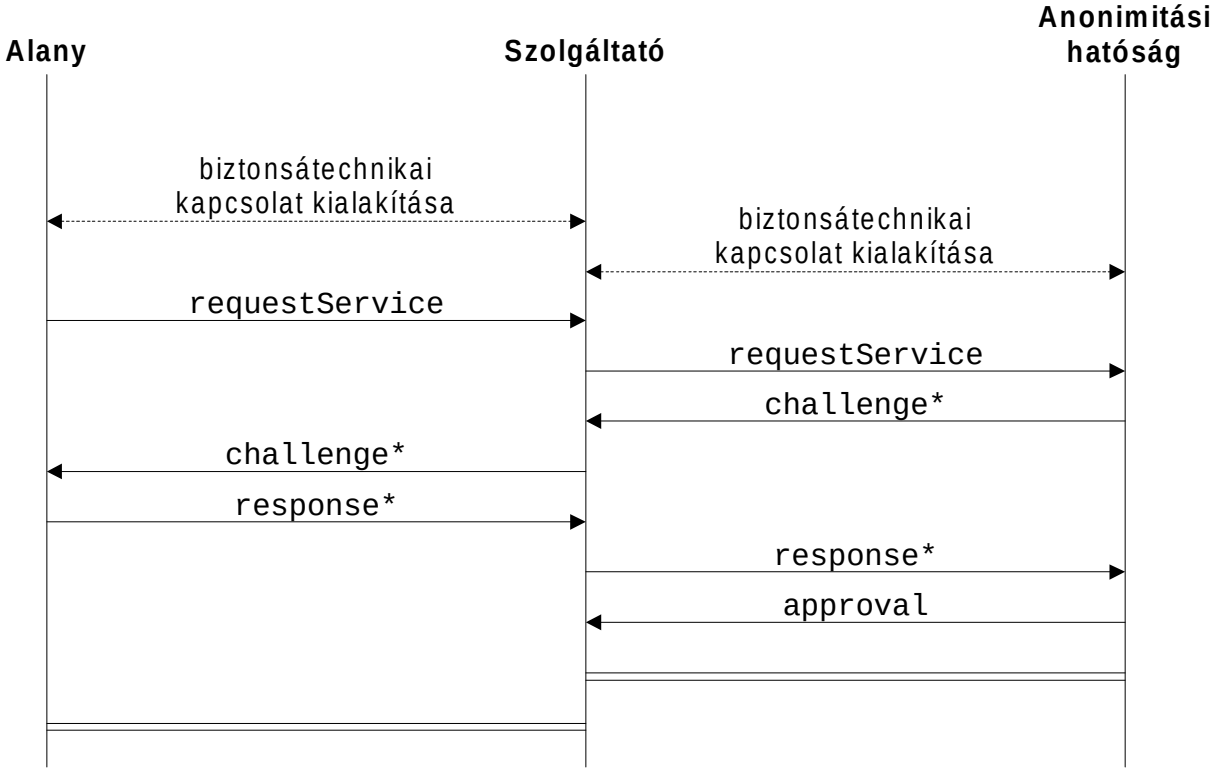
- Először az alany elküldi az anonimitási hatóságnak, hogy milyen módszereket képes megvalósítani (*subjectHello* üzenet).
- Ez után az alany elküldi, hogy milyen típusú szolgáltatást kíván igénybe venni, aközben milyen anonimitási szintet kíván elérni, és szükség esetén kiegészítő információkat is közölhet (*requestAnonimity* üzenet).
- Ez után az anonimitási hatóság az alany által küldött listából kiválasztja azt a konkrét módszert, aminek megfelelő anonimitási okmányt fog majd az alanynak adni (*serverHello* üzenet).
- Az alany, amennyiben nem rendelkezik a szükséges algoritmusok megvalósításához szükséges paraméterekkel, ezeket kérheti az anonimitási hatóságtól (*requestInfo* opcionális üzenet).
- Az anonimitási hatóság, amennyiben szükséges és rendelkezik velük, megadhatja ezeket a paramétereket (*responseInfo* opcionális üzenet).
- Amennyiben az igényelt anonimitási szint eléréséhez az alany részéről szükséges olyan adat továbbítása az anonimitási hatóság felé, melyet kriptográfiailag kell védeni az anonimitási hatóságtól, akkor azt a **boríték módszerrel** (részletes leírását lásd a vak aláírás módszer ismertetésénél lentebb) küldheti el (*envelope* opcionális üzenet).
- Utolsó üzenetként az anonimitási hatóság kiállítja, lekönyveli és az alany rendelkezésére bocsátja a szolgáltatás igénybevételéhez, valamint az igényelt anonimitási szint eléréséhez szükséges anonimitási okmányokat (*anonimityToken* üzenet).

Az utolsó üzenet elküldése után az előkészítő szakasz kommunikációs része lezártnak tekinthető, így a két fél közötti kapcsolat bontható. Elképzelhető, hogy az alanynak a megkapott anonimitási okmányokon még módosításokat (esetleges kriptográfiai műveleteket) kell végeznie, ezekez a konkrét módszerek leírják.

5.3.2.2 Szolgáltatás igénybevétele

A használat során az alany az anonimitási okmányok segítségével igénybe veszi a szolgáltatótól a szolgáltatást, melyet az anonimitási hatóság engedélyez. Fontos, hogy a kommunikáció során az alany végig az előkészítő szakaszban eldöntött anonimitási szinten tartózkodik.

A használat során először az alany-szolgáltató kapcsolat épül ki, melynek a biztonságtechnikai rétegtől a szolgáltató hiteles azonosítását, az integritás védelmet és a biztonságot követeli meg, majd ezután épül csak ki a szolgáltató-anonimitási hatóság kapcsolat, ahol a biztonságtechnikai réteg feladata mindkét fél hiteles azonosítása, az integritás védelem és a bizalmasság biztosítása.



7. ábra – **Az általános szolgáltatás igénybevételének kommunikációs diagrammja**

A szolgáltatás igénybevételéhez szükséges, hogy az alany és a szolgáltató megegyezzenek a konkrét szolgáltatásban és az ahhoz szükséges anonimitási okmányokban. Ennek a tárgyalásnak a menete nem része a protokollnak, feltételezzük, hogy ez sikeres volt, mindketten ismerik és elfogadják ennek végeredményét. A protokoll szempontjából csak ez a végeredmény mérvadó.

A kommunikáció üzenetei a következő lényegi adatokat hordozzák:

- Az alany hivatkozik a kiválasztott szolgáltatásra, **kéri** annak teljesítését és átadja a szolgáltatónak az anonimitási okmányokat (*requestService* üzenet).
- Majd a szolgáltató elküldi az anonimitási hatóságnak, hogy milyen szolgáltatástípus-igény és milyen anonimitási okmányok érkeztek be hozzá (*requestService* üzenet).
- Amennyiben szükséges, az anonimitási hatóság kezdeményezheti az alany anonim azonosítását a szolgáltatón keresztül. Ennek első lépéseként az anonimitási hatóság visszaküldhet a szolgáltatónak egy kérvényt, melyet az alanynak kell majd később **hitelesítenie** (*challenge* opcionális üzenet).
- Ezután a szolgáltató továbbítja a fent megkapott hitelesítési kérvényt az alanynak (*challenge* opcionális üzenet).
- A kérvény kézhezvétele után az alany hitelesíti azt, majd visszaküldi a szolgáltatónak (*response* opcionális üzenet). Fontos, hogy a hitelesítésből a szolgáltató ne tudjon az alany személyazonosságára következtetni.
- A szolgáltató ezután továbbítja ezt a hitelesítést az anonimitási hatóságnak (*challenge* opcionális üzenet).
- Amennyiben a hitelesítést az anonimitási hatóság elfogadja és a megkapott anonimitási okmányok megfelelnek a kért szolgáltatás típusának, akkor az anonimitási hatóság **engedélyezi** és **lekönyveli** a szolgáltatás teljesítését (*approval* üzenet).

Ezek után a protokoll szempontjából a használat lezártnak tekinthető. Ezután történik meg a kommunikáció talán legfontosabb eleme, a szolgáltatás konkrét teljesítése, mely azonban nem része a protokollnak.

A teljesítés után a kommunikációs csatornák mindhárom fél között bonthatók.

5.3.2.3 Különleges esetek

5.3.2.3.1 Visszaélés

Amennyiben a protokollban leírt kommunikáció során valamelyik fél visszaélést tapasztal, úgy legrosszabb esetben a kommunikáció megszüntetésével ez a probléma tünetileg kezelhető (hogy nagyobb kár ne keletkezzen), majd megtehetők a szükséges törvényes lépések.

Amennyiben a visszaélésre csak a szolgáltatás teljesítése után derül fény, abban az esetben már korlátozottabbak a lehetőségek. Az álnév módszer által nyújtott korlátozottan visszakövethető alany anonimitási szint mellett a szolgáltató kezdeményezheti külső hatóságok (pl. bíróság) bevonásával az alany anonimitásának megszüntetését, személyazonossága felfedését. Azonban ha az alany a visszakövethetetlen alany anonimitási szinttel élt például a vak-aláírás módszer alkalmazásával, úgy utólag nincs lehetőség személyazonosságának megismerésére.

5.3.2.3.2 Hiba

Amennyiben bármelyik fél a protokoll alkalmazása során olyan hibát észlel, melyre a protokoll nem nyújt megoldási lehetőséget, úgy végső esetben alkalmazhat egy speciális kapcsolatbontó üzenetet, melyben megnevezi a hibát és lehetséges okait. (*alert* opcionális üzenet).

5.3.3 Megjegyzések a konkrét kommunikációhoz

Az üzenetek és a hozzájuk tartozó adatstruktúrák prezentációs nyelven történő konkrét bemutatását a Melléklet Adatstruktúrák című alfejezete tartalmazza.

A jelen protokoll a "0.1" verziószámot viseli.

A kommunikáció során használt valamennyi adatstruktúra tartalmaz kiegészítés típusú mezőt (ezt általában az `Extension ext;` sor deklarálja). Ezek szerepe, hogy amennyiben az adatstruktúrát küldő fél abban olyan adatokat akar küldeni, amelyek a többi deklarált mező segítségével nem lehetségesek, akkor azt ezeknek a mezőknek a segítségével megtehesse.

Az anonimitás konkrét megvalósított formájára példát a következő, A megvalósított rendszer című fejezet ad, a kommunikáció tényleges formáját mind a reprezentációs nyelven, mint pedig a kommunikációs nyelven pedig a Melléklet Példák című alfejezet tartalmazza.

Amennyiben a kommunikáció során bármelyik fél olyan hibát észlel, mely miatt nem tudja folytatni a már megkezdett tranzakciót, úgy ezt a partner felé az *alert* üzenet elküldésével jelezheti. Ennek hatására a kapcsolat bontandó.

5.3.4 Korlátozottan visszakövethető alany anonimitási szint

A korlátozottan visszakövethető alany anonimitási szint egyik konkrét megvalósítása, az álnév módszer alkalmazása során az alany a szolgáltatás igénybevételekor a kívánt anonimitási szintet az **álnév** (*pseudoIdentity*), mint az **anonimitási hatóság** által kiállított **anonimitási okmány** használatával éri él.

5.3.4.1 Előkészítő szakasz

Ennek a módszernek az alkalmazása során az előkészítő szakasz egyes üzeneteinek konkrét tartalma a következők szerint alakul:

5.3.4.1.1 *subjectHello* üzenet

Az *subjectHello* üzenetben alany megjelöli a protokoll verzióját (ez a protokoll a "0.1" verziószámot írja elő) és a támogatott anonimitási módszerek listájában (*supportedAnonimitySuites*) feltünteti az álnév módszert (*pseudoIdentity*).

5.3.4.1.2 *requestAnonimity* üzenet

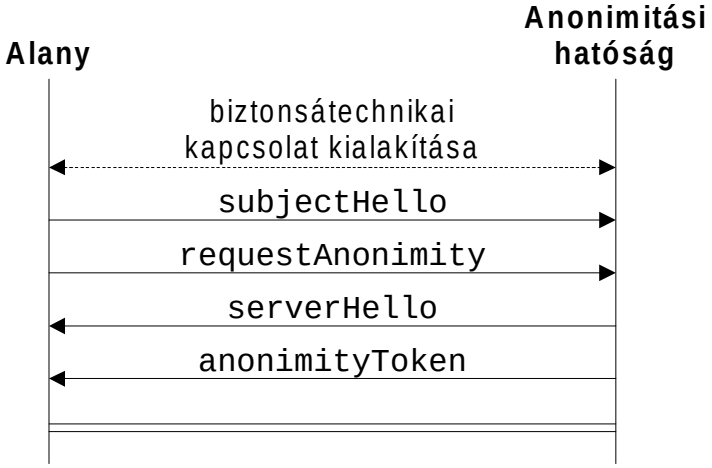
A *requestAnonimity* üzenetben az alany megjelöli a kívánt szolgáltatás típusát (*serviceType*) és kéri, hogy az anonimitási hatóságot, hogy a korlátozottan visszakövethető alany anonimitási szintnek (*anonimityLevel*) megfelelő anonimitási okmányt állítson ki. A szolgáltatás típusa például a szolgáltatónál történő csoportbesorolás lehet.

5.3.4.1.3 *serverHello* üzenet

A *serverHello* üzenetben az anonimitási hatóság kijelöli azt az anonimitási módszert (*chosenAnonimitySuite*), melyet alkalmazva majd kiállítja az anonimitási okmányt.

5.3.4.1.4 *anonimityToken* üzenet

Az *anonimityToken* üzenet többször felhasználható (*anonimityTokenUsageData*) anonimitási okmányt tartalmaz, mely megjelöli anonimitási szintként a korlátozottan visszakövethető alany szintet (*anonimityLevel*), anonimitási módszerként például az álnév módszert (*anonimitySuite*), tartalmazza az érvényességi intervallumot (*validityStart* és *validityEnd*), az anonimitási hatóság egyértelmű azonosítóját (*issuerDN*), az igénybe vehető szolgáltatás típusát (*serviceType*) és végül magát az álnevet (*pseudoIdentity*).



8. ábra – **Az előkészítő szakasz kommunikációs diagrammja korlátozottan visszakövethető alany anonimitási szint esetén**

5.3.4.2 Szolgáltatás igénybevétele

A szolgáltatás igénybevétele során az egyes üzenetek konkrét tartalma a következők szerint alakul:

5.3.4.2.1 *requestService* üzenet (alany ⇨ szolgáltató)

A *requestService* üzenet ezen típusában (alany ⇨ szolgáltató) az alany közli a szolgáltatóval a konkrét szolgáltatás azonosítóját (*service*) és átadja az annak igénybevételéhez szükséges anonimitási okmányokat (*anonimityTokenDataList*), ami az álnevét igazolja. A konkrét szolgáltatás lehet például a csoportbesorolásának megfelelő valamely erőforrás használata, azon valamilyen művelet elvégzése, például valamilyen információ letöltése.

5.3.4.2.2 *requestService* üzenet (szolgáltató ⇨ anonimitási hatóság)

A *requestService* ezen típusában (szolgáltató ⇨ anonimitási hatóság) a szolgáltató az anonimitási hatóságnak csak az anonimitási okmányokat (*anonimityTokenDataList*) küldi el azok hitelesítése céljából.

5.3.4.2.3 *challenge* üzenet

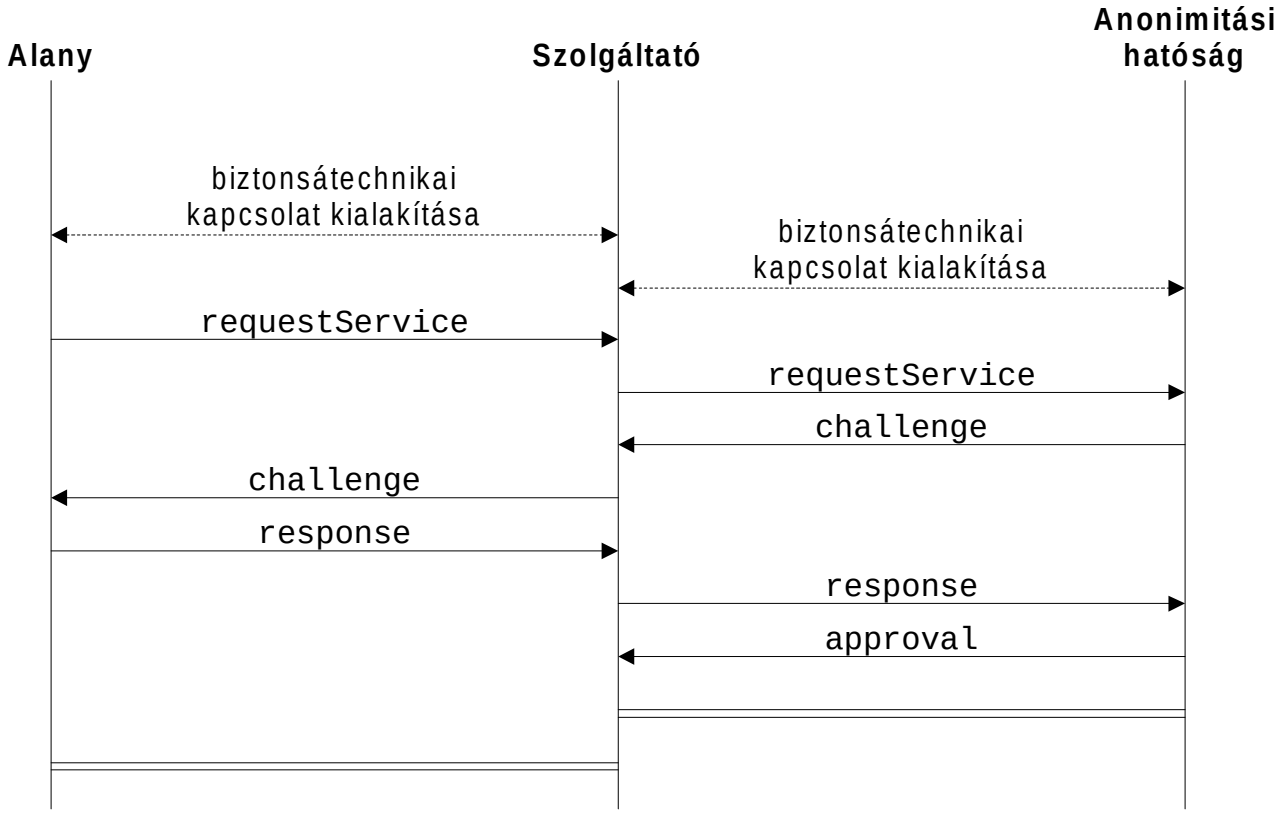
A *challenge* üzenetben az anonimitási hatóság egy véletlen adat (*challenge*) aláírását kéri az alanytól, de mivel nincsenek közvetlen kapcsolatban, ezért ezt az üzenetet a szolgáltató továbbítja. Annak érdekében, hogy erre az üzenetre válaszként érkező *response* üzenet a szolgáltató számára értelmezhetetlen maradjon, ugyanakkor abból az alanyt azonosítani lehessen, azt mind az alany titkos kulcsával (hitelesség), mind pedig az anonimitási hatóság nyilvános kulcsával le kell titkosítani (csak az anonimitási hatóság tudja ezután elolvasni). Mivel azonban mind az alany, mind az anonimitási hatóság többféle kulccsal rendelkezhet, ezeket a *challenge* üzenet egyértelműen megnevezi (*signatureAlgorithmAA* az anonimitási hatóság részéről, *signatureAlgorithmS* az alany részéről).

5.3.4.2.4 *response* üzenet

A *response* üzenetben az alany elküldi a szolgáltatón keresztül az anonimitási hatóságnak a megfelelően hitelesített véletlen adatot (*response*).

5.3.4.2.5 *approval* üzenet

Amennyiben a hitelesítést az anonimitási hatóság elfogadja, úgy ezt a szolgáltató felé az *approval* üzenettel jelzi. Ennek hatására a szolgáltató megkezdheti az igényelt szolgáltatás teljesítését.



9. ábra – **A szolgáltatás igénybevételének kommunikációs diagrammja korlátozottan visszakövethető alany anonimitási szint esetén**

5.3.5 Visszakövethetetlen alany anonimitási szint

A visszakövethetetlen alany anonimitási szint egy konkrét megvalósítása a vak aláírás módszer, melyet eredetileg David Chaum dolgozott ki internetes anonim fizetésre a DigiCash cégnek. Azonban ez a módszer általánosítva alkalmazható számos más anonimitást és biztonságtechnikai funkcionalitást is használó forgatókönyvhöz, mint például anonim szavazáshoz.

A módszer megfelel az általános anonimitási módszer leírásának, viszont lényeges különbség az előző módszerrel szemben, hogy az anonimitás szint emelését az anonimitási rétegen belüli kriptográfiai algoritmusok alkalmazásával éri el.

Ennek megfelelően a módszerek egyeztetése után a következő műveletek hajtódnak végre, ezeket a pénzverés mozzanataihoz szokták hasonlítani:

- Az alany beszerzi az anonimitási hatóságtól a szolgáltatás típusához tartozó RSA nyilvános kulcsot (ez áll egyszer a nyilvános exponensből–`AHPubExp`, és a modulusból–`AHMod`). Ezt az alany megteheti a *requestInfo–responseInfo* üzenetpárral.
- Ezután az alany generál két véletlen számot (azonosító – `randomNumber` és a boríték – `randomEnvelope`). Ezek után az alany kiszámítja a következő speciális számot, ahol a `randomNumber`$^+$ a `randomNumber` szám kétszer egymás mögé írt változata:

  $$specNumberInEnvelope = (randomNumber^+)randomEnvelope^{AHPubExp} \bmod AHMod$$

  Ezt a speciális számot küldi el az alany az anonimitási hatóságnak az *envelope* üzenetben.

  Ennek a műveletnek a szemléltetése úgy képzelhető el, hogy az alany egy átlátszatlan, más által felnyithatatlan borítékba elhelyez egy még veretlen érmét.

- Az anonimitási hatóság ezen a speciális számon elvégzi a következő műveletet, ahol az `AHPrivExp` az előbbi nyilvános kulcshoz tartozó titkos kulcs exponense:

  $$signedSpecNumberInEnvelope = specNumberInEnvelope^{AHPrivExp} \bmod AHMod$$

  Ezt az új speciális számot küldi vissza az anonimitási hatóság az *anonimityToken* üzenetben az **elő-engedély** részeként.

  Szemléltetésként ez a művelet úgy képzelhető el, mintha az anonimitási hatóság a borítékra ráütné a pénzérme tényleges domborulatait, ami a borítékon átnyomódva a benne elhelyezett veretlen pénzre is rányomódik, így most már a borítékban egy kész pénzérme helyezkedik el.

- Végül az alany elvégzi a következő műveleteket, ahol a `randomEnvelope`$^{-1}$ a `randomEnvelope` szám inverze:

  $$signedSpecNumber = signedSpecNumberInEnvelope \cdot randomEnvelope^{-1} \bmod AHMod$$

  Ezt az új speciális szám az alany végleges **engedélyének** fő alkotóeleme.

  Ez a művelet úgy is elképzelhető, mintha az alany felnyitná a borítékot és kivenné belőle a kész pénzérmét.

Ezen műveletek és kommunikációs lépések sorozata azt eredményezi, hogy az alany birtokában van egy olyan speciális számnak (*signedSpecNumber*), melyet rajta kívül senki sem ismer, még az anonimitási hatóság sem, hiszen amikor ő hitelesítette azt, az még le volt titkosítva a véletlen borítékkal. Azonban erről a speciális számról könnyű bizonyítani, hogy

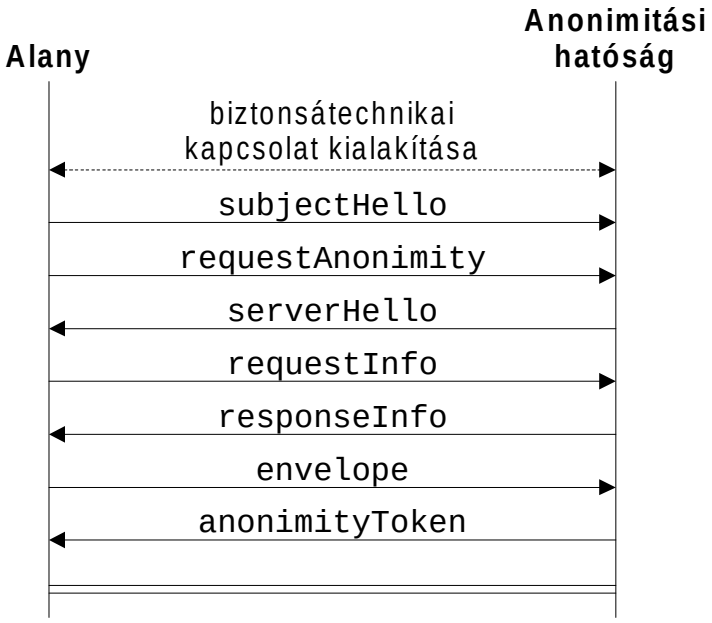
az anonimitási hatóság azt hitelesítette. Ez abból látszik, hogy ha ezen a számon végrehajtunk egy RSA műveletet az anonimitási hatóság nyilvános kulcsával, akkor egy olyan speciális új számot kapunk eredményül, amely két véletlen szám egymás után írásából keletkezett. Mivel egy ilyen RSA-val kódolt szám készítéséhez szükség lenne az anonimitási hatóság titkos kulcsára, ezt csak ő tehette meg.

Összességében azt kapjuk, hogy az alany birtokába kerül egy olyan engedélynek, melyet az anonimitási hatóság hitelesített, azonban azt nem ismeri. Azáltal azonban, hogy hitelesített, felhasználható a megfelelő típusú szolgáltatás igénybevételéhez. A szolgáltatás igénybevételekor a hitelesítés ellenőrzését (amennyiben a szolgáltató ismeri a szolgáltatás típusához tartozó nyilvános kulcsot) a szolgáltató az anonimitási hatóság bevonása nélkül elvégezheti. Azonban annak ellenőrzésére, hogy ez az anonimitási okmányt nem használták-e fel már más alkalommal is, mindenképpen be kell vonnia az anonimitási hatóságot.

A gyakorlatban ennek a módszernek a megvalósítása során a következőket kell figyelembe venni:

- Az anonimitási hatóság minden különböző típusú szolgáltatáshoz más titkos és nyilvános kulcspárt használ. Azaz például internetes anonim fizetés esetén minden egyes pénzérme/papírpénz típusnak külön kulcspár felel meg, így más az 1000 forintos és az 5000 forintos kulcspárja, vagy anonim szavazás esetén minden szavazáskor különböző kulcspár használandó.
- A speciális szám hitelesítésekor az alany személyazonossága ismert, így azt mindenképpen lekönyvelhető, hogy ő rendelkezésére bocsátottak egy bizonyos típusú anonimitási engedélyt. Ez a gyakorlatban azt jelenti, hogy a pénzérme kibocsátásának esetében annak értékével az alany számláját az anonimitási hatóság (ebben az esetben a bank) megterheli. Ugyanilyen alapon a szolgáltatás igénybevételekor a felhasznált pénzérme értékét a szolgáltató számláján jóváírja. Anonim szavazás esetén az anonimitási okmány a szavazócédulának felel meg, ennek kiadása lekönyvelhető. A szolgáltató a szavazás esetén az urnának felel meg.

5.3.5.1 Előkészítő szakasz



10. ábra – **Az előkészítő szakasz kommunikációs diagrammja visszakövethetetlen alany anonimitási szint esetén**

Ennek a módszernek az alkalmazása során az előkészítő szakasz egyes üzeneteinek konkrét tartalma a következők szerint alakul:

5.3.5.1.1 *subjectHello* üzenet

Az *subjectHello* üzenetben alany megjelöli a protokoll verzióját (ez a protokoll a "0.1" verziószámot írja elő) és a támogatott anonimitási módszerek listájában (*supportedAnonimitySuites*) feltünteti az vak-aláírás módszert (*blindSignature*).

5.3.5.1.2 *requestAnonimity* üzenet

A *requestAnonimity* üzenetben az alany megjelöli a kívánt szolgáltatás típusát (*serviceType*) és kéri az anonimitási hatóságot, hogy visszakövethetetlen alany anonimitási szintnek (*anonimityLevel*) megfelelő anonimitási okmányt állítson ki.

5.3.5.1.3 *serverHello* üzenet

A *serverHello* üzenetben az anonimitási hatóság kijelöli azt az anonimitási módszert (*chosenAnonimitySuite*), melyet alkalmazva majd kiállítja az anonimitási okmányt.

5.3.5.1.4 *requestInfo* üzenet

A *requestInfo* üzenetben az alany olyan adatokat kérhet az anonimitási hatóságtól, amelyekre az esetleges kriptográfiai műveletek során szüksége lehet. Ilyen adat lehet az anonimitási hatóság által kibocsátott, a szolgáltatás típusának megfelelő nyilvános kulcs, mely a vak-aláírás módszer esetében, a *specNumberInEnvelope* speciális szám előállításához szükséges.

5.3.5.1.5 *responseInfo* üzenet

A *responseInfo* üzenetben az anonimitási hatóság az előző üzenetben igényelt adatokat szolgáltatja, azaz például megadja az igényelt nyilvános kulcsot.

5.3.5.1.6 *envelope* üzenet

Az *envelope* üzenet az olyan kriptográfiai adatot tartalmaz, amely az anonimitási szint eléréséhez szükséges. Ilyen lehet az előzőekben már ismertetett módon kiszámolt és ismert tulajdonságokkal rendelkező *specNumberInEnvelope* szám.

5.3.5.1.7 *anonimityToken* üzenet

Az *anonimityToken* üzenet egyszer felhasználható (*anonimityTokenUsageData*) anonimitási okmányt tartalmaz, mely megjelöli anonimitási szintként a visszakövethetetlen alany szintet (*anonimityLevel*), anonimitási módszerként a például vak-aláírás módszert (*anonimitySuite*), tartalmazza az érvényességi intervallumot (*validityStart* és *validityEnd*), az anonimitási hatóság egyértelmű azonosítóját (*issuerDN*), az igénybe vehető szolgáltatás típusát (*serviceType*) és végül magát az elő-engedélyt (*prePermission*), azaz a *signedSpecNumberInEnvelope* számot.

Az üzenet megérkezte után az alany ebből állítja elő a valódi **engedélyt**, a szolgáltató által elfogadott, és a megfelelő anonimitási szintet biztosító anonimitási okmányt, amely kriptográfiai alapként a *signedSpecNumber* számot tartalmazza.

5.3.5.2 Szolgáltatás igénybevétele

A szolgáltatás igénybevétele során az egyes üzenetek konkrét tartalma a következők szerint alakul:

5.3.5.2.1 *requestService* üzenet (alany ⇨ szolgáltató)

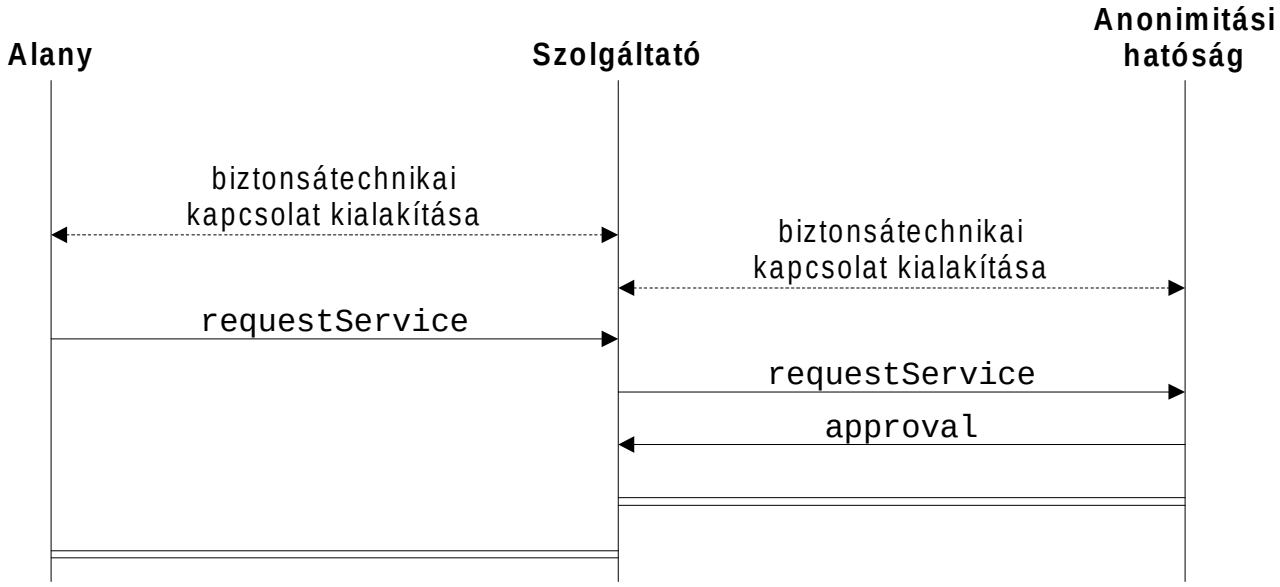
A *requestService* üzenet ezen típusában (alany ⇨ szolgáltató) az alany közli a szolgáltatóval a konkrét szolgáltatás azonosítóját (*service*) és átadja az annak igénybevételéhez szükséges anonimitási okmányokat (*anonimityTokenDataList*), ami jelen esetben a visszakövethetetlen alany anonimitási szintet biztosító engedély. A konkrét szolgáltatás lehet például valamilyen információ megszerzése, de lehet maga az anonim szavazás is.

5.3.5.2.2 *requestService* üzenet (szolgáltató ⇨ anonimitási hatóság)

A *requestService* ezen típusában (szolgáltató ⇨ anonimitási hatóság) a szolgáltató az anonimitási hatóságnak csak az anonimitási okmányokat (*anonimityTokenDataList*) küldi el azok hitelesítése céljából.

5.3.5.2.3 *approval* üzenet

Amennyiben a hitelesítést az anonimitási hatóság elfogadja, úgy ezt a szolgáltató felé az *approval* üzenettel jelzi. Ennek hatására a szolgáltató megkezdheti az igényelt szolgáltatás teljesítését.



11. ábra – **A szolgáltatás igénybevételének kommunikációs diagrammja visszakövethetetlen alany anonimitási szint mellett**

6. A megvalósított rendszer

6.1 Implementáció

A protokoll megalkotásakor fontosnak éreztük, hogy ne csak az elméleti specifikáció készüljön el, hanem egy olyan gyakorlati általános implementáció, melyet későbbi alkalmazások is tudnak hasznosítani.

Az implementáció során a JAVA nyelvre esett a választásunk. Ennek több oka is volt. Egyrészt a JAVA nyelv egy kellően magas szintű, objektum-orientált nyelv, mely megfelel az anonimitási rétegben alkalmazott absztrakciók, adatstruktúrák kezelésére. Hozzájárult a döntéshez, hogy a kommunikációs nyelvet megtestesítő XML nyelvhez széleskörű JAVA támogatás áll rendelkezésre. Másrészt az is befolyásolta a döntést, hogy a biztonságtechnikai rétegnek választott SSH2 réteg JAVA implementációja szabadon rendelkezésre állt.

A JAVA implementáció során megvalósításra került az anonimitási réteg alsó és felső kimenetének, a kommunikációs nyelven és a prezentációs nyelven megfogalmazott adatstruktúráknak a kezelése.

Végeredményben az SSH2 rétegre építve az anonimitási réteggel elkészült azon réteghierarchia JAVA implementációja, mely biztosítja mind a biztonságtechnikai, mint pedig a protokollban specifikált anonimitási funkcionalitást.

6.2 Példa forgatókönyvek

Az implementáció gyakorlati alkalmazhatóságának bemutatására a feldolgozott két anonimitási módszer és ezzel együtt a két anonimitási szint mindegyikére készült egy-egy példaforgatókönyv. Ezeket az egyetemi élet mindennapjai közül választottuk, mivel az itt felmerülő anonimitási kérdésekkel a hallgatók nap mint nap találkoznak.

6.2.1 Korlátozottan visszakövethető alany anonimitási szint

A korlátozottan visszakövethető alany anonimitási szint elérésére az álnév módszer alkalmas. Az egyetemi életből ennek bemutatására az egyetemi beiratkozást választottuk.

Amikor a leendő hallgató felvételizik, a felvételi eredményétől függően az Országos Felsőoktatási Felvételi Iroda (OFFI) segítségével kerül eldöntésre, hogy a hova nyert felvételt a hallgató.

Annak érdekében, hogy a leendő hallgató az egyetemre a korlátozottan visszakövethető alany anonimitási szint mellett tudjon beiratkozni, az OFFI a leendő hallgató rendelkezésére bocsátja azt az álnevet, amivel ezt megteheti.

Ennek a példának megfelelően az anonimitási szerepek a következőképpen alakulnak:

- Az **alany** a **leendő hallgató**, aki az OFFI-tól kapja álnevét, az anonimitási okmányát, mellyel azonosíthatja magát az egyetemnél.
- Az **OFFI** az **anonimitási hatóság**, aki a leendő hallgató rendelkezésére bocsátja az anonimitási okmányát, az álnevét, aminek felhasználásával a hallgatót majd később anonim módon azonosítani lehet az egyetem számára.
- A **szolgáltató** ebben az esetben maga az **egyetem**, aki az álnevet elfogadja és így a hallgatónak megadja az OFFI által megítélt hallgatói státuszt.

6.2.2 Visszakövethetetlen alany anonimitási szint

A visszakövethetetlen alany anonimitási szint elérésére a vak-aláírás módszer alkalmas. Az egyetemi élet mindennapjaiból ennek a módszernek a gyakorlati használatának bemutatására a BME VIK Hallgatói Képviseleti választások lebonyolítását választottuk.

Ennek a példának megfelelően az anonimitási szerepek a következőképpen alakulnak:

- Az **alany** az **egyetemi hallgató**, aki a BME VIK karára jár. Amennyiben ez teljesül, jogosult a Hallgatói Képviseleti választásokon részt venni.
- Az **anonimitási hatóság** ebben az esetben maga az **egyetem**, neki a feladata nyilvántartani, hogy ki jogosult a szavazásra, és amennyiben az kéri, részére az anonimitási okmányt, az elő-engedélyt kiállítani, ami ebben az esetben nem más, mint a szavazócédula.
- A **szolgáltató** ebben az esetben az **urna**, akinek a feladata a szavazócédulák összegyűjtése, azok hitelességének ellenőrzése és az azokon megjelölt szavazatok begyűjtése, összesítése. Ezt a feladatot elláthatja az éppen aktuális (leköszönő) Hallgatói Képviselet.

7. Értékelés

Az általunk felvázolt protokoll a hálózati kommunikáció során napjainkban tapasztalható egyre növekvő anonimitási igény egységes kezelésére próbál egyfajta megoldást találni. Ez a dokumentum egy olyan keretrendszert próbál specifikálni, mely támogat különböző anonimitási szinteket megvalósító algoritmusokat.

A protokoll lényege, hogy a mára már kialakult biztonságtechnikai réteg fölé bevezeti az anonimitási funkcionalitást ellátó anonimitási réteget. Ennek szükségessége abból adódik, hogy az anonimitási igénnyel együtt mindig fellép a hitelesség, az azonosítás igénye is, és így a kettő ellenpont közötti egyensúly megtalálása, az arányok megállapítása a legkényesebb feladat, amire nem lehet egyértelmű megoldást találni. Ezért is kerültek bevezetésre a különböző anonimitási szintek.

Célunk nem az volt, hogy az összes konkrét anonimitási módszert feldolgozzuk (bár igyekeztünk minél többet figyelembe venni), hanem hogy egy olyan protokollt definiáljunk, melyet a jövőben könnyen ki lehet egészíteni újabb módszerekkel, anélkül, hogy a protokollban lényegi változtatásokat kellene eszközölni.

A specifikált és megvalósított protokoll előnye, hogy egységes felületet nyújt anonimitási módszerek megvalósításához, lehetővé teszi különböző anonimitási szintek alkalmazását, az anonimitási módszerek az igényeknek megfelelően választhatók. Mindezt egy biztonságtechnikai rétegre épülve a lehető legnagyobb biztonság mellet nyújtja.

Ugyanakkor fontosnak éreztük azt is, hogy az elméleti specifikálás mellett létrejöjjön egy a gyakorlatban is használható implementáció is. Mind a biztonságtechnikai réteg (SSH2), mind pedig az általános anonimitási réteg elkészült JAVA környezetben és szabadon elérhető a GNU GPL licensze alapján. Anonimitási módszerként pedig az álnév és a vak-aláírás módszer került megvalósításra.

A protokoll továbbfejlesztési irányzatainak meghatározásakor fontos szerepet fog kapni újabb anonimitási módszerek feldolgozása és integrálása, hogy az anonimitási szintek mindegyikét támogassa az implementáció, valamint a kommunikációs nyelv megfelelő (sávszélesség-hatékony) megválasztása.

Távlati célunk a protokoll továbbfejlesztése diplomatervként (esetleg PhD munkaként), annak érdekében, hogy a specifikált és megvalósított protokoll működését tüzetesen megvizsgáljuk és teszteljük, valamint funkcionalitását minél szélesebb anonimitási körre kiterjesszük.

A protokoll használatával a felhasználók egy olyan eszközt kapnak kézhez, mellyel megoldhatók az egyre fontosabbnak ítélt anonimitási kérdések. Ezzel újabb lépést lehet tenni a személyi és főleg a személyes adatok hatékonyabb védelmének elérése felé.

8. Rövidítések

Ez a TDK dolgozat a következő rövidítéseket használja:

3DES	Triple-DES
AES	Advanced Encryption Standard
ASCII	American Standard Code for Information Interchange
BNF	Backus Naur Form
CA	Certification Authority
DES	Data Encryption Standard
DH	Diffie-Hellman
DN	Distinguishing Name
DSA	Digital Signature Algorithm
ECC	Elliptic Curve Cryptography
GPL	General Public License
HTTP	Hypertext Transfer Protocol
IMAP	Internet Message Access Protocol
IP	Internet Protocol
LSB	Less Significant Bit
MD5	Message Digest 5
MSB	Most Significant Bit
PET	Privacy Enhancing Technologies
PKI	Public Key Infrastructure
RSA	Rivest, Shamir & Adleman
SHA1	Secure Hash Algorithm 1
SSH	Secure Shell
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TTP	Trusted Third Party
WAP	Wireless Application Protocol
WTLS	Wireless Transaction Layer Security
XML	Extensible Markup Language

9. Referenciák

[SSL3] Alan O. Freier, Philip Karlton, Paul C. Kocher:(1996): The SSL Protocol Version 3.0, http://home.netscape.com/eng/ssl3/ssl-toc.html

[INTROSSL] Netscape Communication Inc.(1998): Introduction to SSL, http://developer.netscape.com/docs/manuals/security/sslin/index.htm

[PCKINTRO] Netscape Communication Inc.(1998):. Introduction to Public-Key Cryptography, http://developer.netscape.com/docs/manuals/security/pkin/contents.htm

[TLS] RFC 2246 The TLS Protocol Version 1.0

[RFC10244] RFC 1244 Site Security Handbook

[RFC2054] RFC 2504 User's Security Handbook

[SSH] SSH IETF: Secure Shell internet-drafts, http://www.ietf.org/ids.by.wg/secsh.html

[OPENSSL] OpenSSL, www.openssl.org

[SZAMH] Andrew S. Tanenbaum, Számíotgéphálozatok

[WAP] Wireless Application Protocol http://www.wapforum.org

[WTLS] Wireless Transport Layer Security http://www.wapforum.org

[XML] Extensible Markup Language http://www.w3.org/xml

[CHAUM] D. Chaum, R.L.Rivest & A.T.Sherman: Blind Signature for Untraceable Payments

10. Melléklet

10.1 Adatstruktúrák

Ebben a fejezetben megadásra kerülnek a protokollhoz specifikált adatstruktúrák.

```
enum AnonmityLevel {NONE(0), TRACEABLE(1),
RESTRICTED_TRACEABLE(2), NON_TRACEABLE(3), FULL(4)}

struct {
  AnonimityLevel anonimityLevel;
  RString name;
  Extension parameters;
  Extension ext;
} AnonimtySuite;
```

Megnevezés	**Magyarázat**
`anonimityLevel`	Anonimitási szint azonosító.
`name`	Az anonimitás módszer neve.
`parameters, ext`	Kiegészítő paramétereknek fenntartott hely.

```
list<AnonimitySuite> AnonimitySuiteList;

struct {
  RString version;
  AnonimitySuiteList supportedAnonimitySuites;
  Extension ext;
} SubjectHello;
```

Megnevezés	**Magyarázat**
`version`	Protokoll verzió.
`supportedAnonimitySuites`	Az alany által ismert anonimitás módszerek listája.
`ext`	Kiegészítő paramétereknek fenntartott hely.

```
struct {
  RString name;
  Extension ext;
} ServiceType;
```

Megnevezés	**Magyarázat**
`name`	Szolgáltatás-típus neve.
`ext`	Kiegészítő paramétereknek fenntartott hely.

```
struct {
  AnonimityLevel anonimityLevel;
  ServiceType serviceType;
  Extension ext;
} RequestAnonimity;
```

Megnevezés	**Magyarázat**
`anonimityLevel`	Anonimitási szint azonosító.
`serviceType`	Szolgáltatás típus.
`ext`	Kiegészítő paramétereknek fenntartott hely.

```
struct {
  AnonimitySuite chosenAnonimitySuite;
  Extension ext;
} ServerHello;
```

Megnevezés	**Magyarázat**
`chosenAnonimitySuite`	Az anonimitási hatóság által választott anonimitási módszer.
`ext`	Kiegészítő paramétereknek fenntartott hely.

```
enum {PUBKEY(0)} RequestInfoType;

struct {
  RequestInfoType resultInfoType;
  Extension ext;
} RequestInfoData;
```

Megnevezés	**Magyarázat**
`resultInfoType`	További szükséges paraméter típus melyet az anonimitási hatóságtól kíván az alany kérni.
`ext`	Kiegészítő paramétereknek fenntartott hely.

```
list<RequestInfoData> RequestInfoDataList;

struct {
  RequestInfoDataList requests;
  Extension ext;
} RequestInfo;
```

Megnevezés	**Magyarázat**
`requests`	Az alany a kért paramétereket listában közli az anonimitási hatósággal.
`ext`	Kiegészítő paramétereknek fenntartott hely.

```
struct {
  RequestInfoType requestInfoTzpe;
  Extension data;
} ResponseInfoData;
```

Megnevezés	**Magyarázat**
`requestInfoType`	A kért paraméter típus azonosítója.
`data`	Kiegészítő paramétereknek fenntartott hely. Ebben kerül átadásra a kért adat.

```
list<ResponseInfoData> ResponseInfoDataList;

struct {
  ResponseInfoDataList responses;
  Extension ext;
} ResponseInfo;
```

Megnevezés	**Magyarázat**
`responses`	Az anonimitási hatóság listában küldi vissza a kívánt paramétereket az alanynak.
`ext`	Kiegészítő paramétereknek fenntartott hely.

```
struct {
  HEXBlob envelope;
  Extension ext;
} Envelope;
```

Megnevezés	**Magyarázat**
`envelope`	Boríték. Visszakövethetetlen alany anonimitási szint esetében használják. Az alany általa generált adatot továbbit benne, melyet kriptográfiailag védeni kell.
`ext`	Kiegészítő paramétereknek fenntartott hely.

```
enum {SINGLE(0), FIXEDNUMBER(1), MULTIPLE(2)}
AnonimityTokenUsageType;

struct {
  AnonimityTokenUsageType anonimityTokenUsageType;
  select (type) {
    case FIXEDNUMBER {
      DECNumber numberOfUsages;
    }
    default {
    }
  }
} AnonimityTokenUsageData;
```

Megnevezés	**Magyarázat**
`anonimityTokenUsageType`	A kiállított okmány felhasználási módját definiálja.
`numberOfUsages`	Az okmány újrafelhasználhatóság felső korlatja. E változó értéke szabja meg, hogy hányszor lehet felhasználni a kiállított okmányt.
`ext`	Kiegészítő paramétereknek fenntartott hely.

```
struct {
  RString pseudoIdentity;
  Extension ext;
} PseudoIdentity;
```

Megnevezés	Magyarázat
pseudoIdentity	A létrehozott álnév.
ext	Kiegészítő paramétereknek fenntartott hely.

```
struct {
  HEXBlob permissionData;
  Extension ext;
} PermissionData;
```

Megnevezés	Magyarázat
permissionData	Az anonimitási hatóság által kiállított engedély.
ext	Kiegészítő paramétereknek fenntartott hely.

```
enum {PRE(0), FINAL(1)} PermissionStatus;

struct {
  PermissionStatus status;
  PermissionData permissionData;
  Extension ext;
} Permission;
```

Megnevezés	Magyarázat
status	A kiállított engedély státusza. Ha az értéke PRE akkor az alanynak még el kell végeznie módosításokat a kapott okmányon mielőtt használná. Ha pedig FINAL akkor pedig már felhasználható.
permissionData	Az anonimitási hatóság által kiállított engedély.
ext	Kiegészítő paramétereknek fenntartott hely.

```
struct {
   DECNumber year;
   DECNumber month;
   DECNumber day;
   Extension ext;
} Date;
```

Megnevezés	Magyarázat
year	Dátum év értéke.
month	Dátum hónap értéke.
day	Dátum nap értéke.
ext	Kiegészítő paramétereknek fenntartott hely.

```
struct {
   DECNumber hour;
   DECNumber minute;
   DECNumber second;
   RString timeZone;
   Extension ext;
} Time;
```

Megnevezés	**Magyarázat**
hour	Az idő óra egysége.
minute	Az idő perc egysége.
second	Az idő másodperc egysége.
timeZone	Az időzóna.
ext	Kiegészítő paramétereknek fenntartott hely.

```
struct {
   Date date;
   Time time;
   Extension ext;
} DateTime;
```

Megnevezés	**Magyarázat**
date	Dátum értéke.
time	Idő értéke.
ext	Kiegészítő paramétereknek fenntartott hely.

```
struct {
  AnonimityLevel anonimityLevel;
  AnonimitySuite anonimitySuite;
  DateTime validityStart;
  DateTime validityEnd;
  AnonimityTokenUsageData;
  RString issuerDN;
  ServiceType serviceType;
  select (level) {
    case RESTRICTED_TRACEABLE {
      PseudoIdentity      ;
    }
    case NON_TRACEABLE {
      Permission permission;
    }
    default {
    }
  }
  Extension ext;
} AnonimityTokenData;
```

Megnevezés	**Magyarázat**
anonimityLevel	Anonimitási szint azonosító.
anonimitySuite	Az anonimitás módszer struktúra.
validityStart	Az okmány érvényességének kezdete.
validityEnd	Az okmány érvényességének lejárata.
AnonimityTokenUsageData	A kiállított okmány felhasználási módja.
issuerDN	Az okmány kiadó neve.
serviceType	Szolgáltatás típusa.

`pseudoIdentity`	A létrehozott álnév.
`permission`	Az anonimitási hatóság által kiállított engedély.
`ext`	Kiegészítő paramétereknek fenntartott hely.

```
struct {
  AnonimityTokenData data;
  Extension ext;
} AnonimityToken;
```

Megnevezés	**Magyarázat**
`data`	A kiállított okmány.
`ext`	Kiegészítő paramétereknek fenntartott hely

```
struct {
  RString serviceID;
  Extension ext;
} Service;
```

Megnevezés	**Magyarázat**
`serviceID`	Az engedélyezett szolgáltatás azonosítója.
`ext`	Kiegészítő paramétereknek fenntartott hely

```
list<AnonimityTokenData> AnonimityTokenDataList;

enum {SUBJECT_PROVIDER(0), PROVIDER_ANONIMITY_AUTHORITY(1)
     } UsageConnection;

struct {
  UsageConnection usageConnection;
  select (usageConnection) {
    case SUBJECT_PROVIDER {
      Service service;
    }
    default {
    }
  }
  AnonimityTokenDataList anonimityTokenDataList;
  Extension ext;
} RequestService;
```

Megnevezés	**Magyarázat**
`usageConnection`	Az értéke megmondja kihez irányul az alany kérése. Ha értéke 0, akkor a szolgáltató fele, ha értéke 1 anonimitási hatóság felé irányul.
`service`	A kívánt szolgáltatás.
`anonimityTokenDataList`	A kiállított okmányok listában összefűzve.
`ext`	Kiegészítő paramétereknek fenntartott hely

```
enum {DES(0), 3DES(1), AES(2)} SymmetricKeyType;

struct {
    SymmetricKeyType keySpecType;
    HEXBlob key;
    Extension ext;
} SymmetricKeyData;
```

Megnevezés	**Magyarázat**
keySpecType	Szimmetrikus rejtjelezésben használt algoritmusazonosító.
key	A szimetrikus kulcs.
ext	Kiegészítő paramétereknek fenntartott hely.

```
struct {
  HEXNumber e;
  HEXNumber m;
  Extension ext;
} RSAPubKey;
```

Megnevezés	**Magyarázat**
e	Az RSA nyilvános kulcs exponense.
m	Az RSA nyilvános kulcs modulusa.
ext	Kiegészítő paramétereknek fenntartott hely.

```
struct {
  HEXNumber d;
  HEXNumber m;
  Extension ext;
} RSAPrivKey;
```

Megnevezés	**Magyarázat**
d	Az RSA titkos kulcs exponense
m	Az RSA titkos kulcs modulusa.
ext	Kiegészítő paramétereknek fenntartott hely.

```
enum {RSA(0)} AsymmetricKeyType;

struct {
  AsymmetricKeyType keySpecType;
  select (keySpecType) {
    case RSA {
      RSAPrivKey data;
    }
  }
  Extension ext;
} PrivKey;
```

Megnevezés	**Magyarázat**
keySpecType	Aszimmetrikus rejtjelezésben használt algoritmus azonosító.
data	A titkos kulcs.
ext	Kiegészítő paramétereknek fenntartott hely.

```
struct {
  AsymmetricKeyType keySpecType;
  select (keySpecType) {
    case RSA {
      RSAPubKey data;
    }
  }
  Extension ext;
} PubKey;
```

Megnevezés	Magyarázat
keySpecType	Aszimmetrikus rejtjelezés algoritmus típusjelző.
data	A nyilvános kulcs.
ext	Kiegészítő paramétereknek fenntartott hely.

```
enum {PUBLIC(0), PRIVATE(1)} AsymmetricKeySide;

struct {
  AsymmetricKeySide side;
  AsymmetricKeyType keySpecType;
  select (side) {
    case PRIVATE {
      PrivKey keySideData;
    }
    case PUBLIC {
      PubKey keySideData;
    }
  }
  Extension ext;
} AsymmetricKeyData;
```

Megnevezés	Magyarázat
side	Aszimmetrikus rejtjelezés kulcs típust azonosítja. Ha értéke 0 nyilvános kulcsról, ha pedig 1 akkor titkos kulcsról van szó.
keySideData	Kulcs tartalom. A tartalma függ a kulcs típusától.
ext	Kiegészítő paramétereknek fenntartott hely.

```
enum {ASYMMETRIC(0), SYMMETRIC(1), NONE(2)} KeyType;

struct {
  KeyType keyType;
  select (keyType) {
    case ASYMMETRIC {
      AsymmetricKeyData keyData;
    }
    case SYMMETRIC {
      SymmetricKeyData keyData;
    }
    case NONE {
    }
  }
  Extension ext;
} Key;
```

Megnevezés	Magyarázat
keyType	Rejtjelezési módszerazonosító. Ha értéke 0 akkor aszimmetrikus, ha értéke 1 akkor szimmetrikus rejtjelezésről van szó.
keyData	Az absztrakt kriptográfiai kulcs.
ext	Kiegészítő paramétereknek fenntartott hely.

```
enum {SHA1(0), MD5(1)} HashAlgorithm;

struct {
  HashAlgorithm hashAlgorithm;
  HEXBlob hash;
  Extension ext;
} Hash;
```

Megnevezés	Magyarázat
hashAlgorithm	Hash algoritmus típusa
hash	Hash értéke
ext	Kiegészítő paramétereknek fenntartott hely.

```
struct {
  HashAlgorithm hashAlgorithm;
  AsymmetricKeyType cypherAlgorithm;
  Extension ext;
} SignatureAlgorithm;
```

Megnevezés	Magyarázat
hashAlgorithm	Hash algoritmus típusa.
cypherAlgorithm	Aszimmetrikus rejtjelezéshez használt algoritmusazonosító.
ext	Kiegészítő paramétereknek fenntartott hely.

```
struct {
  SignatureAlgorithm signatureAlgorithm;
  HEXBlob signature;
  Extension ext;
} Signature;
```

Megnevezés	Magyarázat
signatureAlgorithm	Digitális aláírásban használt algoritmus leírása.
signature	A digitálisan aláírt üzenet.
ext	Kiegészítő paramétereknek fenntartott hely.

```
struct {
  SignatureAlgorithm signatureAlgorithmAA;
  SignatureAlgorithm signatureAlgorithmS;
  HEXBlob challenge;
  Extension ext;
} Challenge;
```

Megnevezés	Magyarázat
signatureAlgorithmAA	Az alany az anonimitási hatóság kulcsai közül melyiket és hogyan használja.
signatureAlgorithmS	Az alany saját kulcsai közül melyiket és hogyan használja.
challenge	Az aláírandó üzenet.
ext	Kiegészítő paramétereknek fenntartott hely.

```
struct {
  HEXBlob response;
  Extension ext;
} Response;
```

Megnevezés	Magyarázat
response	Alany által aláírt challenge üzenet.
ext	Kiegészítő paramétereknek fenntartott hely.

```
struct {
  Extension ext;
} Approval;
```

Megnevezés	Magyarázat
ext	Kiegészítő paramétereknek fenntartott hely. Az alanyt értesíteni kell kérelmének elfogadásáról elutasításáról.

```
struct {
  DECNumber id;
  RString reason;
  Extension ext;
} Alert;
```

Megnevezés	Magyarázat
id	Hibakód
reason	A hiba okának értelmezése
ext	Kiegészítő paramétereknek fenntartott hely.

10.2 Példák

Ebben az alfejezetben két konkrét anonimitási módszer, az álnév módszer és a vak-aláírás módszer konkrét alkalmazását mutatjuk be a prezentációs nyelv és a kommunikációs nyelv szintjén a Megvalósított rendszer fejezetben leírt mintaalkalmazásoknak megfelelően.

A prezentációs nyelv szintjén a => jel után következő változó kerül elküldésre, míg a kommunikációs nyelv a tényleges XML karaktersorozatot bemutatja.

A következőkben az S az alany (*subject*), az AA az anonimitási hatóság (*anonimity authority*) és a P a szolgáltató (*provider*) rövidítése.

10.2.1 Korlátozottan visszakövethető alany anonimitási szint

Ez a fejezet bemutatja az álnév módszer alkalmazását az egyetemi környezetben.

10.2.1.1 Előkészítési szakasz

Irány	Prezentációs nyelv	Kommunikációs nyelv
S⇨AA	AnominitySuite pseudoIdentity; pseudoIdentity.level = RESTRICTED_TRACEABLE; pseudoIdentity.name = "PseudoIdentity"; AnonimitySuiteList asl; asl={pseudoIdentity}; SubjectHello subjectHello; subjectHello.version = "0.1"; subjectHello. supportedAnonimitySuites = asl; => subjectHello	<subjectHello type="SubjectHello"> <verstion value="0.1" /> <suppportedAnonimitySuites> <element_0> <level value="2" /> <name value="PseudoIdentity" /> <parameters> </parameters> <ext> </ext> </element_0> </suppportedAnonimitySuites> <ext> </ext> </subjectHello>
S⇨AA	ServiceType serviceType; serviceType.name = "BME-VIK INFO"; RequestAnonimity requestAnonimity; requestAnonimity.level = RESTRICTED_TRACEABLE; requestAnonimity.serviceType=serviceType; => requestAnonimity	<requestAnonimity type="RequestAnonimity"> <level value="2" /> <serviceType> <name="BME-VIK INFO" /> <ext> </ext> </serviceType> <ext> </ext> </requestAnonimity>
AA⇨S	ServerHello serverHello; serverHello.chosenAnonimity Suite=pseudoIdentity; => serverHello	<serverHello type="ServerHello"> <chosenAnonimitySuite> <level value="2" /> <name value="PseudoIdentity" /> <parameters> </parameters> <ext> </ext> </chosenAnonimitySuite> <ext> </ext> </serverHello>

Irány	Prezentációs nyelv	Kommunikációs nyelv
AA⇨S	DateTime validityStart; DateTime ValidityEnd; Date startDate; Date endDate; Time startTime; Time endTime; startDate.year = 2001; startDate.month = 10; startDate.day = 26; startTime.hour = 0; startTime.minute = 0; startTime.second = 0; startTime.timeZone = "CET"; endDate.year = 2002; endDate.month = 10; endDate.day = 25; endTime.hour = 23; endTime.minute = 59; endTime.second = 59; endTime.timeZone = "CET"; validityStart.date = startDate; validityStart.time = startTime; validityEnd.date = endDate; validityEnd.time = endTime; AnonimityTokenUsageData atud; atud.type=MULTIPLE; PseudoIdentity pit; pseudoIdentityToken.pseudoId entity="H7RIQY"; AnonimityTokenData pitd; atd.anonimityLevel = RESTRICTED_TRACEABLE; atd.anonimitySuite = pseudoIdentity; atd.validityStart = validityStart; atd.validityEnd = validityEnd; atd.anonimityTokenUsageData = atud; atd.issuerDN = "CN = DiagigazolvanyKozpont, C=HU"; atd.serviceType = serviceType; atd.pseudoIdentity = pseudoIdentity;	<anonimityToken type="AnonimityToken"> <data> <anonimityLevel value="2" /> <anonimitySuite> <level value="2" /> <name value="PseudoIdentity" /> <parameters> </parameters> <ext> </ext> </anonimitySuite> <validityStart> <date> <year value="2001" /> <month value="10" /> <day value="26" /> <ext> </ext> </date> <time> <hour value="0" /> <minute value="0" /> <second value="0" /> <timeZone value="CET" /> <ext> </ext> </time> <ext> </ext> </validityStart> <validityEnd> <date> <year value="2002" /> <month value="10" /> <day value="25" /> <ext> </ext> </date> <time> <hour value="23" /> <minute value="59" /> <second value="59" /> <timeZone value="CET" /> <ext> </ext> </time> <ext> </ext> </validityEnd> <anonimityTokenUsageData> <type value="2" /> <ext> </ext> </anonimityTokenUsageData> <issuerDN value="CN=DiagigazolvanyKozpont, C=HU" />

Irány	Prezentációs nyelv	Kommunikációs nyelv
folyta-tás	AnonimityToken anonimityToken; anonimityToken.data=atd; => anonimityToken	<serviceType> <name="BME-VIK INFO" /> <ext> </ext> </serviceType> <pseudoIdentity> <pseudoIdentity value="H7RIQY" /> <ext> </ext> </pseudoIdentity> <ext> </ext> </data> <ext> </ext> </anonimityToken>

10.2.1.2 Használat

Irány	Prezentációs nyelv	Kommunikációs nyelv
S⇨P	Service service; service.seviceID = "Beiratkozas" AnonimityTokenDataList atdl; atdl={atd}; RequestService requestService; requestService. usageConnection = SUBJECT_PROVIDER; requestService.service = service; requestService. anonimityTokenDataList = atdl; => requestService	<requestService type="RequestService"> <usageConnection value="0" /> <service> <serviceID value="Beiratkozas" /> <ext> </ext> </service> <anonimityTokenDataList> <element_0> <anonimityLevel value="2" /> <anonimitySuite> <level value="2" /> <name value="PseudoIdentity" /> <parameters> </parameters> <ext> </ext> </anonimitySuite> <validityStart> <date> <year value="2001" /> <month value="10" /> <day value="26" /> <ext> </ext> </date> <time> <hour value="0" /> <minute value="0" /> <second value="0" /> <timeZone value="CET" /> <ext> </ext> </time> <ext> </ext> </validityStart>

Irány	Prezentációs nyelv	Kommunikációs nyelv
folyta-tás		<validityEnd> <date> <year value="2002" /> <month value="10" /> <day value="25" /> <ext> </ext> </date> <time> <hour value="23" /> <minute value="59" /> <second value="59" /> <timeZone value="CET" /> <ext> </ext> </time> <ext> </ext> </validityEnd> <anonimityTokenUsageData> <type value="2" /> <ext> </ext> </anonimityTokenUsageData> <issuerDN value="CN=DiagigazolvanyKozpont, C=HU" /> <serviceType> <name="BME-VIK INFO" /> <ext> </ext> </serviceType> <pseudoIdentity> <pseudoIdentity value = "H7RIQY" /> <ext> </ext> </pseudoIdentity> <ext> </ext> </element_0> </anonimityTokenDataList> <ext> </ext> </requestService>
P⇨AA	requestService. usageConnection = PROVIDER_ANONIMITY_AUTHORITY ; => requestService	<requestService type="RequestService"> <usageConnection value="1" /> <anonimityTokenDataList> <element_0> <anonimityLevel value="2" /> <anonimitySuite> <level value="2" /> <name value="PseudoIdentity" /> <parameters> </parameters> <ext> </ext> </anonimitySuite>

Irány	Prezentációs nyelv	Kommunikációs nyelv
folyta-tás		<validityStart> <date> <year value="2001" /> <month value="10" /> <day value="26" /> <ext> </ext> </date> <time> <hour value="0" /> <minute value="0" /> <second value="0" /> <timeZone value="CET" /> <ext> </ext> </time> <ext> </ext> </validityStart> <validityEnd> <date> <year value="2002" /> <month value="10" /> <day value="25" /> <ext> </ext> </date> <time> <hour value="23" /> <minute value="59" /> <second value="59" /> <timeZone value="CET" /> <ext> </ext> </time> <ext> </ext> </validityEnd> <anonimityTokenUsageData> <type value="2" /> <ext> </ext> </anonimityTokenUsageData> <issuerDN value="CN=DiagigazolvanyKozpont, C=HU" /> <serviceType> <name="BME-VIK INFO" /> <ext> </ext> </serviceType> <pseudoIdentity> <pseudoIdentity value="H7RIQY" /> <ext> </ext> </pseudoIdentity> <ext> </ext> </element_0>

Irány	Prezentációs nyelv	Kommunikációs nyelv
foly- tatás		</anonimityTokenDataList> <ext> </ext> </requestService>
AA⇨P és P⇨S	SignatureAlgorithm sigA; SignatureAlgorithm sigAA; sigA.hashAlgorithm=SHA1; sigA.cypherAlgorithm=RSA; sigAA.hashAlgorithm=SHA1; sigAA.cypherAlgorithm=RSA; Challenge challenge; challenge. signatureAlgorithmAA = sigAA; challenge. signatureAlgorithmA = sigA; challenge.challenge = _CHALLENGE_; => challenge	<challenge type="Challenge"> <signatureAlgorithmAA> <hashAlgorithm value="0" /> <cypherAlgorithm value="0" /> <ext> </ext> </signatureAlgorithmAA> <signatureAlgorithmA> <hashAlgorithm value="0" /> <cypherAlgorithm value="0" /> <ext> </ext> </signatureAlgorithmA> <challenge value="_CHALLENGE_" /> <ext> </ext> </challenge>
S⇨P és P⇨AA	Response response; response.response = _RESPONSE_; => response	<response type="Response"> <response value="_RESPONSE_" /> <ext> </ext> </response>
AA⇨P	Approval approval; => approval	<approval type="Approval"> <ext> </ext> </approval>

10.2.2 Visszakövethetetlen alany anonimitási szint

Ez a fejezet bemutatja az álnév módszer alkalmazását az egyetemi környezetben.

10.2.2.1 Előkészítő szakasz

Irány	Prezentációs nyelv	Kommunikációs nyelv
S⇨AA	AnonimitySuite blindSignature; blindSignature. anonimityLevel = NON_TRACEABLE; blindSignature.name = "BlindSignature"; AnonimitySuiteList asl = {blindSignature}; SubjectHello subjectHello; subjectHello.version = "0.1"; subjectHello. supportedAnonimitySuites = asl; => subjectHello	<subjectHello type="SubjectHello"> <version value="0.1" /> <supportedAnonimitySuites> <element_0> <anonimityLevel value="3" /> <name value="BlindSignature" /> <param> </param> <ext> </ext> </element_0> </supportedAnonimitySuites> <ext> </ext> </subjectHello>
S⇨AA	ServiceType serviceType; serviceType.name = "BME-VIK HK Valasztas" RequestAnonimity requestAnonimity; requesetAnonimity. anonimityLevel = NON_TRACEABLE; requestAnonimity. serviceType = serviceType; => requestAnonimtiy	<requestAnonimity type="RequestAnonimity"> <anonimityLevel value="3" /> <serviceType> <name value="BME-VIK HK Valasztas" /> <ext> </ext> </serviceType> <ext> </ext> </requestAnonimity>
AA⇨S	ServerHello serverHello; serverHello. chosenAnonimitySuite = blindSignature; => serverHello	<serverHello type="ServerHello"> <chosenAnonimitySuite> <anonimityLevel value="3" /> <name value="BlindSignature" /> <param> </param> <ext> </ext> </chosenAnonimitySuite> <ext> </ext> </serverHello>

Irány	Prezentációs nyelv	Kommunikációs nyelv
S⇨AA	RequestInfoData rid; rid.requestInfoType=PUBKEY; requestInfoDataList aidl; ridl={rid}; RequestInfo requestInfo; requestInfo.requests=ridl; => requestInfo	<requestInfo type="RequestInfo"> <requests> <element_0> <requestInfoType value="0" /> <ext> </ext> </element_0> </requests> <ext> </ext> </requestInfo>
AA⇨S	ResponseInfoData rid; rid.requestInfoType=PUBKEY; RSAPubKey rsaPubKey; rsaPubkey.e=_EXPONENT_; rsaPubKey.m=_MODULUS_; PubKey pubKey; pubKey.keySpecType=RSA; pubkey.data=rsaPubKey; rid.ext=__XML(pubKey); ResponseInfoDataList ridl; ridl={rid}; ResponseInfo responseInfo; responseInfo.responses=ridl; => responseInfo	<responseInfo type="ResponseInfo"> <responses> <element_0> <requestInfoType value="0" /> <ext> <pubKey type="PubKey"> <keySpecType value="0"> <data> <e value="_EXPONENT_" /> <m value="_MODULUS_" /> <ext> </ext> </data> <ext> </ext> </pubKey> </ext> </element_0> </responses> <ext> </ext> </responseInfo>
S⇨AA	Envelope envelope; envelope.envelope = _specNumberInEnvelope_; => Envelope	<envelope type="Envelope"> <envelope value="_specNumberInEnvelope" /> <ext> </ext> </envelope>
AA⇨S	DateTime validityStart; DateTime ValidityEnd; Date startDate; Date endDate; Time startTime; Time endTime; startDate.year = 2001; startDate.month = 10; startDate.day = 26; startTime.hour = 0; startTime.minute = 0; startTime.second = 0; startTime.timeZone = "CET";	<anonimityToken type="AnonimityToken"> <data> <anonimityLevel value="3" /> <anonimitySuite> <anonimityLevel value="3" /> <name value="BlindSignature" /> <param> </param> <ext> </ext> </anonimitySuite>

Irány	Prezentációs nyelv	Kommunikációs nyelv
folyta-tás	endDate.year = 2002; endDate.month = 10; endDate.day = 25; endTime.hour = 23; endTime.minute = 59; endTime.second = 59; endTime.timeZone = "CET"; validityStart.date = startDate; validityStart.time = startTime; validityEnd.date = endDate; validityEnd.time = endTime; AnonimityTokenUsageData atud; atud.type=SINGLE; PermissionData pd; pd.permissionData = _signedSpecNumberInEnvelope_ ; Permission permission; permission.status=PRE; permission.permissionData = pd; AnonimityTokenData atd; atd.anonimityLevel = NON_TRACEABLE; atd.anonimitySuite = blindSignature; atd.validityStart = validityStart; atd.validityEnd = validityEnd; atd.anonimityTokenUsageData = atud; atd.issuerDN = "CN=BME, C=HU"; atd.serviceType = serviceType; atd.permission = permission; AnonimityToken anonimityToken; anonimityToken.data=atd; => anonimityToken	<validityStart> <date> <year value="2001" /> <month value="10" /> <day value="26" /> <ext> </ext> </date> <time> <hour value="0" /> <minute value="0" /> <second value="0" /> <timeZone value="CET" /> <ext> </ext> </time> <ext> </ext> </validityStart> <validityEnd> <date> <year value="2002" /> <month value="10" /> <day value="25" /> <ext> </ext> </date> <time> <hour value="23" /> <minute value="59" /> <second value="59" /> <timeZone value="CET" /> <ext> </ext> </time> <ext> </ext> </validityEnd> <anonimityTokenUsageData> <type value="0" /> <ext> </ext> </anonimityTokenUsageData> <issuerDN value="CN=BME, C=HU" /> <serviceType> <name value="BME-VIK HK Valasztas" /> <ext> </ext> </serviceType>

Irány	Prezentációs nyelv	Kommunikációs nyelv
folyta-tás		<permission> <status value="0" /> <permissionData> <permissionData value = "_signedSpecNumberInEnvelope_" /> <ext> </ext> </permissionData> <ext> </ext> </permission> <ext> </ext> </data> <ext> </ext> </anonimityToken>

10.2.2.2 Használat

Irány	Prezentációs nyelv	Kommunikációs nyelv
S⇨P	Service service; service.serviceID = _VOTE_; AnonimityTokenDataList atdl; atdl={atd}; RequestService requestService; requestService. usageConnection = SUBJECT_PROVIDER; requestService.service = service; requestService. anonimityTokenDataList= atdl; => requestService	<requestService type="RequestService"> <usageConnection value="0" /> <service> <serviceID value="_VOTE_" /> <ext> </ext> </service> <anonimityTokenDataList> <element_0> <anonimityLevel value="3" /> <anonimitySuite> <anonimityLevel value="3" /> <name value="BlindSignature" /> <param> </param> <ext> </ext> </anonimitySuite> <validityStart> <date> <year value="2001" /> <month value="10" /> <day value="26" /> <ext> </ext> </date> <time> <hour value="0" /> <minute value="0" /> <second value="0" /> <timeZone value="CET" /> <ext> </ext> </time> <ext> </ext> </validityStart>

Irány	Prezentációs nyelv	Kommunikációs nyelv
folyta-tás		<validityEnd> <date> <year value="2002" /> <month value="10" /> <day value="25" /> <ext> </ext> </date> <time> <hour value="23" /> <minute value="59" /> <second value="59" /> <timeZone value="CET" /> <ext> </ext> </time> <ext> </ext> </validityEnd> <anonimityTokenUsageData> <anonimityTokeUsageType value="0" /> <ext> </ext> </anonimityTokenUsageData> <issuerDN value=" CN=BME, C=HU" /> <serviceType> <name value="BME-VIK HK Valasztas" /> <ext> </ext> </serviceType> <permission> <status value="1" /> <permissionData> <permissionData value="_signedSpecNumber_" /> <ext> </ext> </permissionData> <ext> </ext> </permission> <ext> </ext> </element_0> </anonimityTokenDataList> <ext> </ext> </requestService>

Irány	Prezentációs nyelv	Kommunikációs nyelv
P⇨AA	requestService. usageConnection = PROVIDER_ANONIMIY_AUTHORITY => requestService	<requestService type="RequestService"> <usageConnection value="0" /> <anonimityTokenDataList> <element_0> <anonimityLevel value="3" /> <anonimitySuite> <anonimityLevel value="3" /> <name value="BlindSignature" /> <param> </param> <ext> </ext> </anonimitySuite> <validityStart> <date> <year value="2001" /> <month value="10" /> <day value="26" /> <ext> </ext> </date> <time> <hour value="0" /> <minute value="0" /> <second value="0" /> <timeZone value="CET" /> <ext> </ext> </time> <ext> </ext> </validityStart> <validityEnd> <date> <year value="2002" /> <month value="10" /> <day value="25" /> <ext> </ext> </date> <time> <hour value="23" /> <minute value="59" /> <second value="59" /> <timeZone value="CET" /> <ext> </ext> </time> <ext> </ext> </validityEnd> <anonimityTokenUsageData> <anonimityTokeUsageType value="0" /> <ext> </ext> </anonimityTokenUsageData>

Irány	Prezentációs nyelv	Kommunikációs nyelv
foly- tatás		<issuerDN value=" CN=BME, C=HU" /> <serviceType> <name value="BME-VIK HK Valasztas" /> <ext> </ext> </serviceType> <permission> <status value="1" /> <permissionData> <permissionData value="_signedSpecNumber_" /> <ext> </ext> </permissionData> <ext> </ext> </permission> <ext> </ext> </element_0> </anonimityTokenDataList> <ext> </ext> </requestService>
AA⇨P	Approval approval; => approval	<approval type="Approval"> <ext> </ext> </approval>