VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

DATABÁZE SPECIFIKACÍ BEZPEČNOSTNÍCH PROTOKOLŮ

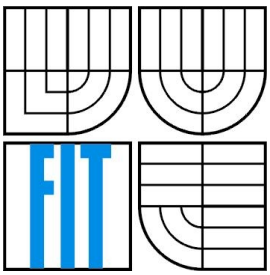
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. DAVID ONDRÁČEK

BRNO 2008

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

DATABÁZE SPECIFIKACÍ BEZPEČNOSTNÍCH PROTOKOLŮ

SPECIFICATIONS DATABASE OF SECURITY PROTOCOLS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. DAVID ONDRÁČEK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. PAVEL OČENÁŠEK

BRNO 2008

Zadání

1. Nastudujte vybrané bezpečnostní komunikační protokoly a proveďte jejich vzájemné srovnání.
2. Proveďte hloubkovou studii nástrojů a prostředí pro analýzu, simulaci a implementaci komunikačních protokolů.
3. Vzájemně jednotlivé nástroje porovnejte. Diskutujte vhodnost jednotlivých nástrojů pro různé druhy komunikačních protokolů. Zaměřte se na bezpečnostní protokoly.
4. Ve zvoleném prostředí (po konzultaci s vedoucím práce) namodelujte vybrané bezpečnostní protokoly tak, aby je bylo možné simulovat a ověřit jejich chování a vlastnosti. Specifikaci proveďte tak, aby ji bylo možné využít v dalších projektech.
5. Porovnejte vlastnosti zkoumaných protokolů s již publikovanými informacemi.
6. Zhodnoťte dosažené výsledky a diskutujte možnosti rozšíření projektu.

Licenční smlouva

Licenční smlouva je uložena v archivu Fakulty informačních technologií Vysokého učení technického v Brně.

Abstrakt

Původní protokoly, které vznikaly v počátcích vývoje počítačových sítí, již nejsou pro zajištění potřebné bezpečnosti dostačující. Proto se stále vyvíjejí a implementují protokoly nové. Důležitou součástí tohoto procesu je formální verifikace. Jde o analýzu protokolu po formální stránce, kdy se zjišťuje, zda lze protokol úspěšně napadnout. Diplomová práce se zabývá analýzou vybraných bezpečnostních protokolů a nástrojů pro jejich formální verifikaci. Výstupem praktické části je databáze specifikací protokolů v LySa kalkulu a výsledky jejich verifikace nástrojem LySatool.

Klíčová slova

Bezpečnost, komunikace, bezpečnostní protokoly, formální verifikace, analýza protokolů, verifikační nástroje, autentizace, důvěrnost.

Abstract

Original protocols, which were created during early development of computer networks, no longer provide sufficient security. This is the reason why new protocols are developed and implemented. The important component of this process is formal verification, which is used to analyze the developed protocols and check whether a successful attack is possible or not. This thesis presents selected security protocols and tools for their formal verification. Further, the selected protocols are specified in LySa calculus and results of their analysis using LySatool are presented and discussed.

Keywords

Security, communication, security protocols, formal verification, protocol analysis, verification tools, authentication, confidentiality.

Citace

Ondráček David: Databáze specifikací bezpečnostních protokolů. Brno, 2008, diplomová práce, FIT VUT v Brně.

Databáze specifikací bezpečnostních protokolů

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Pavla Očenáška. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
David Ondráček
19.5.2008

Poděkování

Rád bych tímto poděkoval Ing. Pavlu Očenáškovi za vedení a přínosné rady v průběhu tvorby diplomové práce.

Obsah

1 Úvod

V dobách prvních počítačových sítí byl kladen důraz především na jejich funkčnost a spolehlivost. Co se bezpečnosti informací týče, spoléhalo se především na kolegialitu a poctivost všech účastníků. Neexistovalo prakticky žádné zabezpečení komunikace a autentizace komunikujících stran probíhala pomocí hesla, které se po síti přenášelo v nezašifrované podobě. Tento přístup fungoval v raných začátcích, kdy byly sítě budovány a užívány skupinami nadšenců, programátorů a vědců. S nárůstem počtu uživatelů a vznikem veřejných sítí již tento přístup nebyl dostačující.

Příkladem může být dnes patrně nejznámější a nejrozsáhlejší veřejná počítačová síť – Internet. Za předchůdce Internetu lze považovat experimentální síť ARPANET, jež vznikla v 60. letech minulého století jako projekt financovaný grantovou agenturou ministerstva obrany USA – ARPA (Advanced Research Projects Agency). Tehdy šlo o propojení několika málo počítačů na čtyřech různých univerzitách za účelem ověření fungování systému při využití přepojování paketů. Síť se však postupně rozrůstala, uživatelů přibývalo, došlo ke komercionalizaci a vznikla tak celosvětová počítačová síť s více než miliardou připojených počítačů.

Dnes je Internet lidmi využíván především jako zdroj informací, jako prostředek pro přenos dat a komunikaci na dálku, je zdrojem pracovních příležitostí a slouží také k odpočinku a zábavě. Vzhledem k současné rozsáhlosti a povaze Internetu již samozřejmě není možné spolehnout se na kolegialitu a poctivost všech jeho uživatelů. Existují lidé, kteří se snaží využít neoprávněně získaná důvěrná data k vlastnímu obohacení či prospěchu, případně k poškození jiné osoby. Pro označení takového člověka se vžil termín cracker (často nesprávně zaměňováno s pojmem hacker). Původní protokoly a služby, které vznikaly v počátcích vývoje a které se používají dodnes (např. sada protokolů TCP/IP), nejsou pro zajištění potřebné bezpečnosti dostačující. Proto se stále vyvíjejí a implementují protokoly nové. Důležitou součástí tohoto procesu je formální verifikace. Jde o analýzu protokolu a jeho pravidel po formální stránce, kdy se zjišťuje, zda lze protokol úspěšně napadnout.

Diplomová práce se zabývá analýzou vybraných bezpečnostních protokolů a nástrojů pro jejich formální verifikaci. V první části práce (kapitola 2) jsou vymezeny základní pojmy nezbytné pro další výklad. Dále je vysvětlena a na konkrétních příkladech demonstrována formální specifikace protokolů včetně popisu proveditelných útoků (kapitola 3). V kapitole 4 jsou nastíněny hlavní přístupy formální verifikace. Stěžejní část práce tvoří kapitola 5, která obsahuje detailní popis čtyř vybraných verifikačních nástrojů a jejich vzájemné srovnání. Výsledky verifikací protokolů jedním z nástrojů jsou uvedeny v praktické části práce (kapitola 6). Databáze vytvořených specifikací je přiložena na CD.

2 Základní pojmy

Problematika síťové bezpečnosti je značně rozsáhlá. Ačkoliv toto není hlavní náplní práce, bude nejprve třeba některé základní pojmy vyskytující se v dalším textu alespoň stručně vymezit.

2.1 Protokol

Protokol je soubor syntaktických a sémantických pravidel pro navázání spojení, komunikaci a přenos dat mezi dvěma entitami spojenými prostřednictvím počítačové sítě. Podle definice standardního referenčního modelu ISO/OSI a modelu TCP/IP probíhá komunikace v síti v několika vrstvách (od fyzické až po aplikační). V tabulce 1 jsou uvedeny příklady protokolů podle příslušnosti k jednotlivým vrstvám.

<table>
<tr><th>OSI</th><th>TCP/IP</th><th colspan="8">Aplikace a protokoly</th></tr>
<tr><td>7. aplikační
6. prezentační
5. relační</td><td>aplikační vrstva</td><td>telnet</td><td>FTP</td><td>TFTP</td><td>SMTP</td><td>RIP</td><td>DNS</td><td colspan="2">ostatní</td></tr>
<tr><td>4. transportní</td><td>transportní vrstva</td><td colspan="4">TCP</td><td colspan="4">UDP</td></tr>
<tr><td rowspan="2">3. síťová</td><td rowspan="2">síťová vrstva</td><td colspan="3" rowspan="2">IP</td><td colspan="2">ICMP</td><td colspan="3"></td></tr>
<tr><td colspan="2">ARP</td><td>RARP</td></tr>
<tr><td>2. linková
1. fyzická</td><td>vrstva síťového rozhraní</td><td colspan="2">token ring</td><td colspan="3">ethernet</td><td colspan="3">jiné typy protokolů</td></tr>
</table>

Tab. 1: Příklady protokolů podle příslušnosti k jednotlivým vrstvám síťového modelu

2.2 Kryptografie

Kryptografie, neboli šifrování, je nauka o metodách transformace otevřeného textu na šifrovaný a (obvykle) naopak. Zašifrovaná zpráva je bez speciální znalosti (klíče) nečitelná / nepoužitelná. Oblast kryptografie je součástí vědního oboru kryptologie. Jeho další součástí je tzv. kryptoanalýza, která se zabývá transformací šifrovaného textu na otevřený bez znalosti klíče. Šifrování lze podle použitých algoritmů a klíčů rozdělit na symetrické a asymetrické.

V případě symetrického šifrování používají všichni účastníci komunikace (principals) stejný (symetrický) klíč.

Asymetrické šifrování využívá dvou korespondujících párových klíčů. Jeden z nich je soukromý (privátní), druhý veřejný. Každá komunikující strana disponuje svým soukromým klíčem

(který nezná nikdo jiný) a veřejným klíčem (ten je k dispozici všem účastníkům komunikace). Pokud chce například subjekt *A* poslat zprávu určenou pro *B*, zašifruje ji veřejným klíčem subjektu *B*. Takto zabezpečenou zprávu může dešifrovat pouze subjekt *B*, jelikož pouze on disponuje odpovídajícím soukromým klíčem. Žádným jiným klíčem zprávu dešifrovat nelze. Tento systém také umožňuje použití tzv. elektronického podpisu, kdy se zpráva zašifruje soukromým klíčem odesílatele a výsledek se přidá k odesílané zprávě. Příjemce pak pomocí dešifrování zprávy veřejným klíčem odesílatele může zjistit, zda je odesílatel skutečně původcem zprávy.

Velkou výhodou asymetrické kryptografie oproti symetrické je fakt, že není nutné před zahájením vlastní komunikace domlouvat / distribuovat tajný klíč pro šifrování. Nevýhodou je však možnost podvrhnout veřejný klíč, čili předložit klíč s nepravou identitou. Tomu se snaží zabránit tzv. certifikační autority, které garantují pravost veřejných klíčů uložených v jejich databázích.

2.3 Bezpečnostní protokol

Bezpečnostní (nebo také kryptografické) protokoly zajišťují bezpečnostní funkce pomocí kryptografických metod a algoritmů. Protokol popisuje, jak by takový algoritmus měl být použit.

V základě lze bezpečnostní protokoly rozdělit do dvou skupin – na autentizační protokoly a protokoly pro distribuci klíčů. Většina bezpečnostních protokolů však plní obě funkce současně.

2.3.1 Autentizační protokol

Autentizační protokol je soubor pravidel, pomocí nichž si účastníci komunikace prokazují svoji identitu (viz. kapitola 2.5.2).

2.3.2 Protokol pro distribuci klíčů

Pro navázání bezpečné komunikace musí všichni účastníci disponovat šifrovacími (dešifrovacími) klíči. K jejich bezpečné distribuci slouží protokoly pro výměnu klíčů.

2.4 KDC, KTC

U protokolů využívajících symetrického šifrování je před zahájením vlastní komunikace nutné mezi účastníky domluvit / distribuovat tajný klíč pro šifrování. Tuto úlohu zastává třetí důvěryhodná strana, která se ve formálních popisech protokolů nejčastěji označuje jako *S* (server). Tento server může fungovat buď jako KDC (Key Distribution Server) nebo KTC (Key Translation Server). Rozdíl mezi nimi je ten, že KDC klíč sám generuje, kdežto KTC klíč pouze přeposílá (vytváří jej jeden z účastníků).

2.5 Bezpečnostní služby

Bezpečnost systému je zajišťována pomocí tzv. bezpečnostních služeb. Jde o služby, které využívají bezpečnostních mechanismů pro předcházení útokům, detekci útoků a zotavení po úspěšném útoku.

2.5.1 Důvěrnost (Confidentiality)

Zajišťuje přístup k důvěrným informacím pouze autorizovaným uživatelům. Za důvěrnou informaci se považuje i informace o existenci objektu. Nejčastěji používaným prostředkem pro zajištění důvěrnosti je kryptografie.

2.5.2 Autentizace (Authentication)

Autentizace je proces, kdy účastník distribuovaného systému prokazuje jinému účastníkovi svoji identitu.

Systém umožňuje tzv. silnou autentizaci v případě, že je splněna tato podmínka: pokud subjekt *A* obdrží zprávu, ve které je jako odesílatel uveden subjekt *B*, pak *B* odeslal právě tuto zprávu subjektu *A*.

Vzhledem k parametrům a kvalitě komunikačních kanálů často není možné uvedený požadavek splnit. Systém pak umožňuje tzv. slabou autentizaci v případě, že je splněna podmínka: pokud subjekt *A* obdrží zprávu, ve které je jako odesílatel uveden subjekt *B*, pak buď *B* poslal právě tuto zprávu subjektu *A* nebo *B* tuto skutečnost popře.

2.5.3 Integrita (Integrity)

Integrita je nejčastěji požadovanou vlastností pro přenos dat. Obdržená data musí být totožná s uloženými / odeslanými. To znamená, že nesmí být při přenosu neoprávněně modifikována, uměle zadržována či opakovaně vysílána po odposlechu. Integrita se zajišťuje pomocí kontrolních součtů, hašovacích funkcí nebo samoopravných kódů.

2.5.4 Nepopiratelnost (Nonrepundation)

Nepopiratelností se rozumí skutečnost, že odesílatel zprávy nemůže popřít její odeslání a příjemce její přijetí.

2.5.5 Dostupnost (Availability)

Dostupnost zajišťuje oprávněným uživatelům přístup k systémovým zdrojům vždy, když je potřeba.

2.6 Bezpečnostní mechanismy

Bezpečnostní mechanismy jsou prostředky, jež mají zajistit předcházení útokům. V případě, že to není možné, měly by být schopny útok alespoň detekovat, případně zajistit co možná nejlepší zotavení napadeného systému. Podstatná část požadavků na bezpečnostní mechanismy je zajištěna kryptografickými metodami a algoritmy.

2.7 Útoky na protokoly

Jelikož mnoho bezpečnostních protokolů obsahuje chyby a zranitelná místa, jež může útočník využít k získání tajných informací, jejich použití samo o sobě bezpečnost komunikace nezajišťuje. Jakákoliv akce provedená za účelem ohrožení některé z bezpečnostních služeb je považována za útok. Jelikož takových útoků existuje velké množství, tvrzení, že protokol je bezpečný, by mělo být vždy doplněno o informaci proti kterým útokům je ochrana zaručena. Útoky lze podle jejich povahy rozdělit do dvou základních skupin – na pasivní a aktivní.

2.7.1 Pasivní útoky

Pasivní útoky nepředstavují tak velké nebezpečí jako útoky aktivní. Útočník „pouze“ odposlouchává komunikaci a snaží se získat přenášené informace. Jelikož nedochází k pozměnění dat či přerušení komunikace, jsou tyto útoky velmi těžce detekovatelné.

Při přenosech dat sensitivní povahy by tedy vždy mělo být použito kryptografických metod. Pokud útočník nedokáže data dešifrovat, nemá možnost je využít. Přesněji řečeno nemá možnost využít přímo obsah přenášených dat. Pomocí metod analýzy provozu na lince (traffic analysis) však určité informace zjistit lze. Jde například o zjištění identity účastníků komunikace, jejich umístění nebo také frekvenci a délku přenášených zpráv. Tyto informace mohou být využity k odhadnutí povahy komunikace.

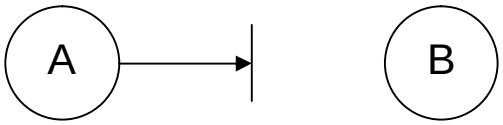
2.7.2 Aktivní útoky

Při aktivních útocích již může dojít k modifikaci přenášené zprávy či jejímu podvržení, čili detekce je zde mnohem jednodušší než u pasivních útoků. Většinou je však obtížné zabránit úspěšnému provedení útoku, opatření jsou tedy zaměřena především na detekci a rychlé zotavení.

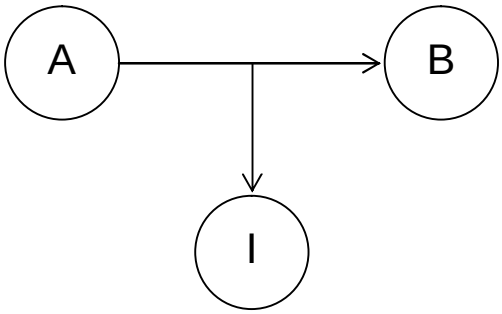
2.7.3 Základní útoky

Na obrázku 1a) je znázorněn normální datový tok od zdroje k cíli. V případě provedení útoku je tento normální tok narušen. Podle způsobu narušení lze rozlišit základní typy útoků znázorněné na obrázcích 1b) – 1e).

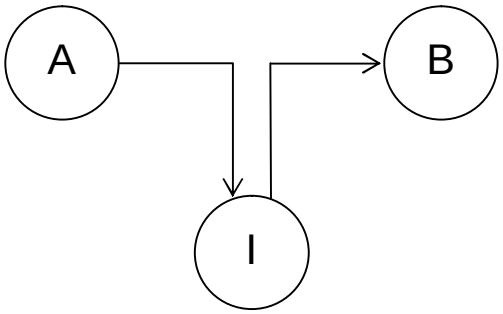
a) Normální datový tok



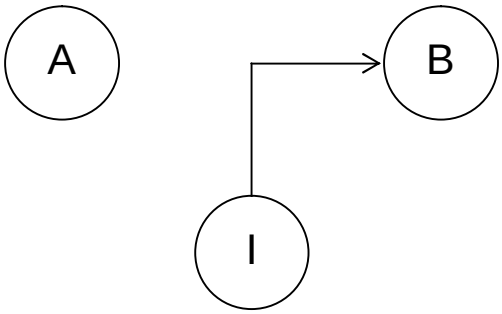
b) Přerušení komunikace



c) Odposlouchávání



d) Modifikace zprávy



e) Podvržení identity

Obr. 1: Základní typy útoků na zprávu protokolu
(*A*, *B* – autorizovaní účastníci komunikace, *I* – útočník)

Přerušení komunikace (Interruption)

Jedná se o útok na dostupnost. Přenášená zpráva je zničena nebo zadržena. Normální datový tok je přerušen. Příkladem může být přerušení komunikačního spoje.

Odposlouchávání (Interception)

Odposlouchávání je útokem na důvěrnost. Neautorizovaný subjekt (může to být osoba, program nebo počítač) získá přístup k přenášeným zprávám. Nejčastěji se jedná o snahu odposlechnout heslo např. při autentizaci klienta ke vzdálenému serveru. Tato hesla se často přenáší po síti v nezašifrované podobě. Dalším příkladem je nepovolené kopírování přenášených souborů či programů.

Modifikace zprávy (Modification)

Neautorizovaný účastník komunikace získá přístup k přenášeným datům a cíleně je modifikuje. Jde tedy o útok na integritu – příjemce sice obdrží zprávu odeslanou autorizovanou stranou, ovšem s pozměněným obsahem. Příkladem je modifikace údajů v přenášených dokumentech nebo úprava programů za účelem změny jejich činnosti či chování.

Podvržení identity (Fabrication)

Útočník vytvoří zprávu s falešnou identitou. Příkladem může být šíření falešných informací po síti.

2.7.4 Další útoky

Přehrávání (Replay)

Útočník se snaží podvrhnout svoji identitu zasláním dříve odposlechnuté zprávy. Pokud příjemce zprávy nemá možnost ověřit její „čerstvost“ (freshness), může dojít k úspěšné autentizaci. K zamezení tohoto typu útoku se používají tzv. nonces (viz. dále).

Útok ze středu (Man in the Middle Attack)

Jedná se o aktivní odposlouchávání, kdy se útočník vloží do komunikace například mezi dva subjekty *A* a *B* a s oběma naváže spojení, přičemž se vydává za regulérního partnera. Pro *A* se tedy jeví jako *B* a pro *B* jako *A*. Kromě odposlouchávání komunikace má tedy možnost zasílat falešné zprávy.

2.8 Nonce

Termín nonce se někdy překládá jako „keksík“. Jde o vygenerované číslo, které se přidává do zpráv protokolu. Pomocí něj mohou účastníci komunikace ověřit „čerstvost“ přijaté zprávy a zabránit tak útokům založeným na přehrávání. Podle způsobu generování čísel lze nonces rozdělit na odhadnutelná (sekvenčně generovaná čísla, časová razítka) a neodhadnutelná (náhodná čísla).

3 Formální popis protokolů

Jak již bylo zmíněno, mnoho bezpečnostních protokolů obsahuje chyby a zranitelná místa. Pro jejich nalezení je možné použít metod formální verifikace. Nejprve je však nutné protokol formálně zapsat.

Protokol je definován jako soubor syntaktických a sémantických pravidel pro komunikaci a přenos dat mezi dvěma entitami. Základním prvkem takové komunikace je zpráva. Ta může obsahovat:

- označení účastníků *A*, *B*, *S*, …
- nonces *Na*, *Nb*, …
- klíče *Ka*, *Kb*, *Kab*, …
- složenou zprávu { *X*, *X′* }
- hašovanou zprávu *Hash X*
- zašifrovanou zprávu *Crypt K X* (zpráva *X* zašifrovaná klíčem *K*);
  používá se také značení $\{ X \}_K$

Klíč *Ka* je obvykle symetrický klíč subjektu *A*, který je sdílen se serverem *S*. Obdobně to platí pro klíč *Kb*. Klíč relace *Kab* je sdílen mezi účastníky *A* a *B*.

V případě asymetrického šifrování se používá označení K^{-1} pro inverzní klíč ke klíči *K*. Pokud platí rovnost $K^{-1} = K$, znamená to, že *K* je symetrický klíč. Dále se předpokládá, že $(K^{-1})^{-1} = K$ pro všechny klíče *K*. Zašifrovaná zpráva nemůže být přečtena nebo upravena bez znalosti správného klíče.

Formální specifikace zprávy může vypadat například takto:

$$A \rightarrow B : \{ X \}_{Kab}$$

Zápis vyjadřuje skutečnost, že subjekt *A* poslal subjektu *B* zprávu *X* zašifrovanou sdíleným klíčem *Kab*.

3.1 Příklady protokolů

Účelem této kapitoly je demonstrovat formální zápis, případně proveditelný útok, na konkrétních protokolech. Jedná se o čtyři základní bezpečnostní protokoly – Otway-Rees, Wide Mouthed Frog, Yahalom a Needham-Schroeder.

3.1.1 Protokol Otway-Rees

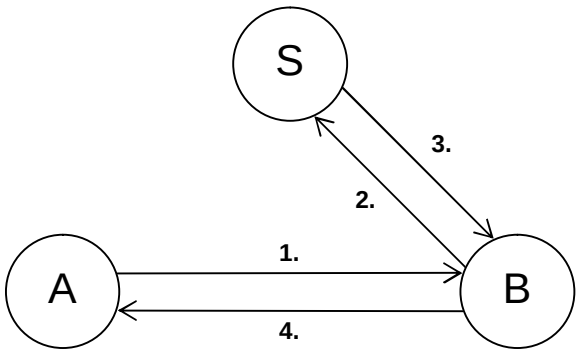
Protokol Otway-Rees je symetrický autentizační protokol navržený pro použití v nezabezpečených sítích (jako je například Internet). Umožňuje účastníkům komunikace vzájemně si prokázat identitu.

Formální specifikace protokolu

1. A → B : Na, A, B, $\{ Na, A, B \}_{Ka}$
2. B → S : Na, A, B, $\{ Na, A, B \}_{Ka}$, Nb, $\{ Na, A, B \}_{Kb}$
3. S → B : Na, $\{ Na, Kab \}_{Ka}$, $\{ Nb, Kab \}_{Kb}$
4. B → A : Na, $\{ Na, Kab \}_{Ka}$

Popis komunikace

Subjekt *A* se autentizuje vůči *B* pomocí důvěryhodného serveru *S*, který zde plní funkci KDC. *Na* a *Nb* jsou nonces. *Ka* je klíč subjektu *A*, *Kb* klíč subjektu *B*. Relační klíč *Kab* je sdílen oběma účastníky. Server *S* zná všechny klíče. Protokol je graficky znázorněn na obrázku 2.



Obr. 2: Grafické znázornění protokolu Otway-Rees

7. *A* vytvoří nonce *Na* a spolu s označením účastníků a zprávou $\{ Na, A, B \}_{Ka}$ jej odešle subjektu *B*.
8. *B* vytvoří nonce *Nb*. Zpráva přijatá od *A* je zašifrovaná klíčem subjektu *A*. *B* ji tedy dešifrovat nemůže a tak ji pouze přepošle serveru *S*. K této zprávě ještě připojí nonce *Nb* a zprávu $\{ Na, A, B \}$ zašifrovanou vlastním klíčem *Kb*.
9. Protože server *S* disponuje klíči všech účastníků komunikace, může obě přijaté zprávy dešifrovat. Následně vytvoří relační klíč *Kab*. Pro subjekt *A* zašifruje zprávu obsahující *Kab* a *Na* klíčem *Ka*.

Taková zpráva se pak nazývá certifikát. Obdobně *S* vytvoří i certifikát pro *B* a oba certifikáty spolu s nonce *Na* odešle subjektu *B*.

10. *B* převezme svůj certifikát a zkontroluje přijaté *Nb*. Pokud souhlasí s *Nb* odeslaným v kroku 2, přepošle druhý certifikát subjektu *A*. Ten jej přijme a obdobně ověří.

Útok

Tato verze protokolu obsahuje chybu – lze provést následující útok:

1. $A \rightarrow I_B : Na, A, B, \{ Na, A, B \}_{Ka}$
1′. $I \rightarrow A : Ni, I, A, \{ Ni, I, A \}_{Ki}$
2′. $A \rightarrow I_S : Ni, I, A, \{ Ni, I, A \}_{Ki}, Na', \{ Ni, I, A \}_{Ka}$
2″. $I_A \rightarrow S : Ni, I, A, \{ Ni, I, A \}_{Ki}, Na, \{ Ni, I, A \}_{Ka}$
3′. $S \rightarrow I_A : Ni, \{ Ni, Kia \}_{Ki}, \{ Na, Kia \}_{Ka}$
4. $I_B \rightarrow A : Na, \{ Na, Kia \}_{Ka}$

Subjekt *A* chce kontaktovat *B*. Odeslanou zprávu ale zachytí útočník *I*, který se za *B* vydává (označení I_B). Útočník následně kontaktuje subjekt *A*. Ten se podle pravidel protokolu snaží zkontaktovat server *S*. Útočník odeslanou zprávu zachytí a modifikuje – původní nonce *Na′* nahradí nonce *Na* z první odposlechnuté zprávy. Následně server *S* odešle subjektu *A* certifikáty obsahující relační klíč. Útočník *I* zprávu opět zachytí a vydávajíc se za *B* odešle subjektu *A* „špatný“ certifikát. Ten však obsahuje platné nonce *Na*, čili subjekt *A* považuje certifikát za platný. Akceptuje tedy obdržený klíč relace, který je sdílený s útočníkem *I*.

Opravená verze protokolu šifruje nonce *Nb* v kroku 2.

3.1.2 Protokol Wide Mouthed Frog

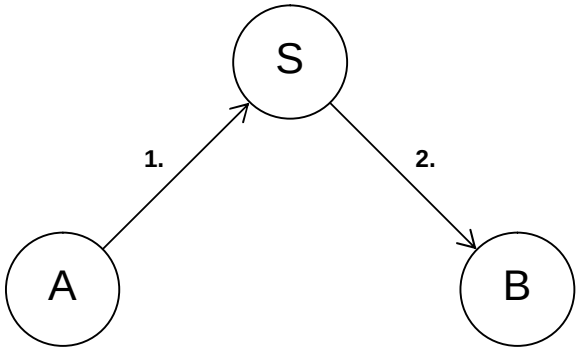
Protokol Wide Mouthed Frog je patrně nejjednodušším autentizačním protokolem využívajícím symetrické kryptografie. Jeho velkou nevýhodou je nutnost použití globálních synchronizačních hodin.

Formální specifikace protokolu

1. $A \rightarrow S : A, \{ Ta, B, Kab \}_{Ka}$
2. $S \rightarrow B : \{ Ts, A, Kab \}_{Kb}$

Popis komunikace

Subjekt *A* se autentizuje vůči *B*. Důvěryhodný server *S* zde plní funkci KTC. *Ta* a *Ts* jsou časová razítka (timestamps). *Ka* je klíč subjektu *A*, *Kb* klíč subjektu *B*. *Kab* je relační klíč. Všechny klíče jsou známy serveru *S*. Grafické znázornění protokolu je na obrázku 3.



Obr. 3: Grafické znázornění protokolu Wide Mouthed Frog

1. Subjekt *A* pošle v zašifrované zprávě serveru *S* klíč relace *Kab*. Zpráva rovněž obsahuje označení subjektu *B* a časové razítko *Ta*.
2. *S* podle časového razítka ověří platnost zprávy a pošle subjektu *B* zašifrovanou zprávu obsahující klíč relace *Kab*, označení subjektu *A* a časové razítko *Ts*. Subjekt *B* pak ověří, zda je *Ts* novější než všechny dříve přijatá časová razítka.

Útok 1

Provedením následujícího útoku lze prodloužit platnost starého (potenciálně kompromitovaného) klíče relace:

1. $A \rightarrow S : A, \{ Ta, B, Kab \}_{Ka}$
2. $S \rightarrow B : \{ Ts, A, Kab \}_{Kb}$
1′. $I_B \rightarrow S : B, \{ Ts, A, Kab \}_{Kb}$
2′. $S \rightarrow A : \{ Ts', B, Kab \}_{Ka}$
1″. $I_A \rightarrow S : A, \{ Ts', B, Kab \}_{Ka}$
2″. $S \rightarrow B : \{ Ts'', A, Kab \}_{Kb}$

Nejprve protokol proběhne bez účasti útočníka. Útočník *I* se následně vydává za *B* a v platném časovém okně přehraje dříve odposlechnutou zprávu. Server *S* zareaguje aktualizací hodnoty časového razítka, čímž se prodlouží platnost klíče *Kab*. Nyní se útočník vydává za subjekt *A* a v platném časovém okně opět přehraje dříve odposlechnutou zprávu. Server *S* znovu aktualizuje hodnotu časového razítka. Takto může útočník prodlužovat platnost klíče libovolně dlouho.

Útok 2

Další možný útok na protokol objevil Lowe v roce 1997. Ve své zprávě rovněž představuje opravenou verzi.

1. $A \rightarrow S : A, \{ Ta, B, Kab \}_{Ka}$
2. $S \rightarrow B : \{ Ts, A, Kab \}_{Kb}$
2′. $I_S \rightarrow B : \{ Ts, A, Kab \}_{Kb}$

Nejprve protokol proběhne obvyklým způsobem. Útočník *I* se následně vydává za *S* a přehraje zprávu číslo 2. Subjekt *B* nyní věří, že *A* s ním navázal dvě relace, zatímco *A* věří, že se subjektem *B* navázal pouze jednu relaci.

3.1.3 Protokol Yahalom

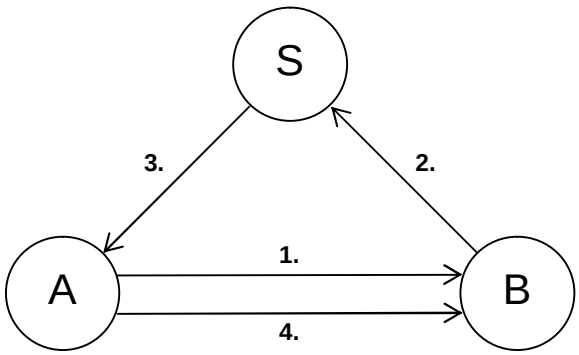
Protokol Yahalom byl představen v roce 1988. Jedná se opět o symetrický autentizační protokol.

Formální specifikace protokolu

1. A → B : A, Na
2. B → S : B, Nb, { A, Na }$_{Kb}$
3. S → A : { B, Kab, Na, Nb }$_{Ka}$, { A, Kab }$_{Kb}$
4. A → B : { A, Kab }$_{Kb}$, { Nb }$_{Kab}$

Popis komunikace

Subjekt *A* se autentizuje vůči *B*. Důvěryhodný server *S* funguje jako KDC. *Na* a *Nb* jsou nonces, *Ka* a *Kb* šifrovací klíče subjektů *A* a *B*, *Kab* je relační klíč. Protokol je graficky znázorněn na obrázku 4.



Obr. 4: Grafické znázornění protokolu Yahalom

1. *A* vytvoří nonce *Na* a spolu se svým označením jej pošle subjektu *B*.
2. *B* přijatou zprávu zašifruje svým klíčem a spolu se svým označením a nově vytvořeným nonce *Nb* odešle serveru *S*.
3. Server *S* vytvoří klíč relace a v zašifrované zprávě jej odešle subjektu *A*.
4. Subjekt *A* převezme svůj certifikát a získá tak klíč relace *Kab*. Druhý certifikát přepošle subjektu *B*.

Útok

Slabinou protokolu je skutečnost, že certifikát subjektu *B* neobsahuje žádné nonce. Toho může útočník využít v případě, že disponuje nějakým starým relačním klíčem *K* (který byl subjekty *A* a *B* již jednou sdílen) a starým certifikátem { *A*, *K* }$_{Kb}$.

1. $I_A \rightarrow B : A, Ni$
2. $B \rightarrow I_S : B, Nb, \{ A, Ni \}_{Kb}$
4. $I_A \rightarrow B : \{ A, K \}_{Kb}, \{ Nb \}_K$

Subjekt *B* takto přijme starý relační klíč *K*. Podobně jako u protokolu Otway-Rees, správná verze protokolu zašifruje nonce *Nb* v kroku 2.

3.1.4 Protokol Needham-Schroeder

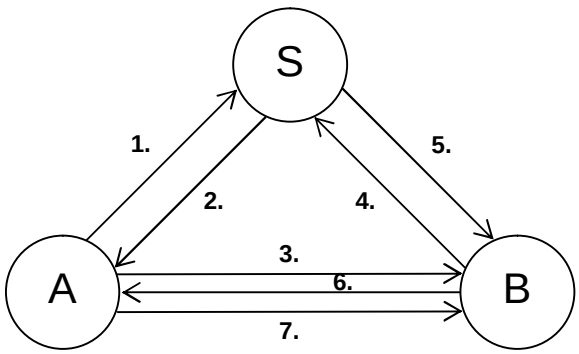
Pod tímto názvem se skrývají dva typy protokolů. Oba byly představeny autory Needhamem a Schroederem v roce 1978. Jeden z nich používá symetrickou kryptografii (Needham-Schroeder Symmetric Key Protocol), druhý funguje na principu veřejného klíče, čili používá asymetrickou kryptografii (Needham-Schroeder Public Key Protocol). Dále bude podrobněji představen druhý zmíněný. Tento protokol je určen pro vzájemnou autentizaci mezi dvěma subjekty.

Formální specifikace protokolu

1. $A \rightarrow S : A, B$
2. $S \rightarrow A : \{ Kb, B \}_{Ks}^{-1}$
3. $A \rightarrow B : \{ Na, A \}_{Kb}$
4. $B \rightarrow S : B, A$
5. $S \rightarrow B : \{ Ka, A \}_{Ks}^{-1}$
6. $B \rightarrow A : \{ Na, Nb \}_{Ka}$
7. $A \rightarrow B : \{ Nb \}_{Kb}$

Popis komunikace

Subjekty *A* a *B* se vzájemně autentizují pomocí důvěryhodného serveru pro distribuci veřejných klíčů (*S*). *Na* a *Nb* jsou nonces, *Ka* veřejný klíč subjektu *A*, *Kb* veřejný klíč subjektu *B*. Ks^{-1} značí soukromý klíč serveru *S*. Grafické znázornění protokolu je na obrázku 5.



Obr. 5: Grafické znázornění protokolu Needham-Schroeder

1. *A* si od serveru *S* vyžádá veřejný klíč subjektu *B*.
2. Server *S* pošle *A* zprávu zašifrovanou svým soukromým klíčem Ks^{-1}. Zpráva obsahuje veřejný klíč subjektu a označení subjektu *B* (pro potvrzení).
3. Subjekt *A* vytvoří nonce *Na* a pošle jej *B*.
4. *B* si od *S* vyžádá veřejný klíč subjektu *A*.
5. Server *S* pošle *B* veřejný klíč subjektu *A*.
6. *B* vytvoří nonce *Nb*. Subjektu *A* pošle zprávu zašifrovanou jeho veřejným klíčem. Kromě nonce *Nb* obsahuje zpráva i dešifrované *Na*. Tím subjekt *B* prokázal svou identitu vůči *A*.
7. *A* dešifruje nonce *Nb* a pošle jej zpět subjektu *B*. Tímto subjekt *A* prokázal svou identitu vůči *B*.

Útok

Protokol v této původní verzi není odolný vůči útoku ze středu. Pokud se útočníkovi *I* podaří přimět subjekt *A*, aby s ním zahájil relaci, může přijaté zprávy přeposílat subjektu *B* a přesvědčit jej, že komunikuje s *A*. Pokud pomineme komunikaci se serverem *S* (ta zůstává nezměněna), vypadá útok následovně:

3. $A \rightarrow I : \{ Na, A \}_{Ki}$
3′. $I_A \rightarrow B : \{ Na, A \}_{Kb}$
6. $B \rightarrow I_A : \{ Na, Nb \}_{Ka}$
6′. $I \rightarrow A : \{ Na, Nb \}_{Ka}$
7. $A \rightarrow I : \{ Nb \}_{Ki}$
7′. $I_A \rightarrow B : \{ Nb \}_{Kb}$

A chce navázat relaci se subjektem *I*. Vytvoří tedy nonce *Na* a pošle mu jej v zašifrované zprávě. *I* (vydávajíc se za *A*) přepošle zprávu subjektu *B*. Subjekt *B* odpoví zprávou obsahující přijaté *Na* a nově vytvořené *Nb*. Útočník nezná klíč *Ka*, tudíž nemůže zprávu dešifrovat. Přepošle ji tedy subjektu *A*, který zprávu dešifruje a vrátí nonce *Nb* zašifrované klíčem útočníka *I*. Útočník nyní může zprávu dešifrovat. Získané *Nb* odešle subjektu *B*. Ten nyní věří, že navázal relaci se subjektem *A*, ve skutečnosti však navázal relaci s útočníkem *I*.

Tento útok poprvé popsal Gavin Lowe ve své zprávě z roku 1995. Zpráva rovněž obsahuje opravenou verzi protokolu označovanou jako protokol Needham-Schroeder-Lowe. Oprava spočívá v přidání identity příjemce *B* do zprávy v kroku 6.

4 Formální verifikace

Již bylo zmíněno, že použití bezpečnostních protokolů samo o sobě bezpečnost nezaručuje. Mnoho protokolů obsahuje chyby a slabiny, kterých může být využito k provedení útoku. Přitom se nejedná pouze o protokoly nové, nýbrž i o starší protokoly, jež byly dlouhou dobu považovány za bezpečné. K nalezení chyb v protokolech se používají techniky formální verifikace, kdy se provádí analýza protokolu a jeho pravidel po formální stránce a ověřuje se, zda lze protokol napadnout / prolomit.

K tomu se používají nástroje, jejichž hlavním cílem je implementace a automatizace technik pro analýzu protokolů a nalezení jejich slabin s minimálně vynaloženým úsilím. Nakolik se tento cíl daří plnit ve značné míře závisí na metodě přístupu daného nástroje. Mezi hlavní přístupy patří kontrola modelem (model checking), modální logiky (modal logics), dokazování teorému (theorem proving) a statická analýza (static analysis).

4.1 Kontrola modelem (Model Checking)

Protokol se modeluje jako systém s obvykle konečným počtem stavů. Celý stavový prostor se prochází a v každém stavu se kontroluje, zda jsou splněny zadané požadavky. Pokud je počet stavů konečný, celý proces verifikace může proběhnout automaticky.

Výhodou tohoto přístupu je skutečnost, že pokud během verifikace dojde k chybě, je možné proces okamžitě přerušit a vypsat stav, ve kterém k porušení požadavků došlo. Použití této metody je ovšem vhodné pouze v případě konkrétně zaměřených protokolů (tedy protokolů, jež nejsou příliš obecné), které navíc nemají velký stavový prostor. Nevýhodou také je, že většina technik této kategorie dokáže ověřit pouze správnost protokolu, nikoliv však jeho bezpečnost. Z těchto důvodů je současný vývoj v oblasti verifikace bezpečnostních protokolů zaměřen spíše na modální logiky a induktivní techniky (viz. dále).

4.2 Modální logiky (Modal Logics)

Verifikace protokolů za použití modální logiky je deduktivní proces. Většina technik této kategorie je založena na axiomizaci znalostí a předpokladů jednotlivých zúčastněných subjektů.

Před vlastní verifikací je nejprve nutné formálně specifikovat počáteční parametry, jednotlivé kroky protokolu v jazyce konkrétní logiky a požadované cíle. Poté následuje spuštění procesu se zadanými počátečními podmínkami, přičemž se opakovaně kontroluje, zda průběžné výsledky odpovídají požadovaným cílům.

Hlavní předností těchto technik je jejich malá časová i prostorová náročnost. Jsou tedy poměrně často používané. Nevýhodou je nutnost přepsat protokol do množiny logických formulí (obvykle manuálně), což může být zdrojem potenciálních chyb.

4.3 Dokazování teorému (Theorem Proving)

4.3.1 Základní metoda

Model protokolu je reprezentován logikou s konečným počtem pravidel, který však během procesu kontroly narůstá, tzn. pravidla se odvozují.

Pro verifikaci protokolu je nejprve nutné vytvořit konečnou množinu axiomů a pravidel. Počáteční předpoklady a zprávy protokolu jsou interpretovány jako pravidla. Verifikační nástroj prohledá všechna pravidla, která jsou odvoditelná z původní množiny pravidel a zjišťuje, zda jsou v nich obsažena pravidla reprezentující testovanou vlastnost protokolu.

4.3.2 Induktivní techniky

Induktivní techniky na rozdíl od původní základní metody neberou ohled na konečnost modelu protokolu. Lze říci, že tento přístup je kombinací výše zmíněných přístupů – kontroly modelem a modální logiky. Protokol je zde induktivně reprezentován jako množina interakcí mezi subjekty. Zápis těchto interakcí (zpráv) je prakticky přejat z technik kontroly modelem, odvozování informací zase z modální logiky. Vlastnosti protokolů jsou ověřovány indukcí.

Induktivní techniky jsou dostatečně obecné a jednoduché. V porovnání s modální logikou však vyžadují delší a mnohem detailnější testování.

4.4 Statická analýza (Static Analysis)

Techniky založené na tomto přístupu analyzují protokol aniž by byl spouštěn a vykonáván jako program, analýza se provádí staticky nad jeho zdrojovým kódem (obvykle jde o rozšířenou formální specifikaci). Díky tomu je možné protokol verifikovat v dřívějších fázích jeho vývoje než by tomu bylo u jiných přístupů. Navíc je takto možné odhalit některé chyby, které by při analýze založené na provádění kódu mohly zůstat skryty.

Mezi statické metody patří analýza toku řízení (control-flow analysis), analýza založená na omezeních (constraint-based analysis), abstraktní interpretace (abstract interpretation) a typové systémy (type systems).

V této práci je blíže představena analýza toku řízení pomocí LySa kalkulu a jeho implementace – nástroje LySatool (viz. dále).

5 Nástroje pro formální verifikaci

V současné době pro verifikaci protokolů existuje mnoho nástrojů. Cílem této kapitoly není poskytnout jejich úplný výčet. Jsou zde detailně popsány čtyři vybrané volně dostupné nástroje. První z nich (LySa) je popsán nejpodrobněji z toho důvodu, že byl použit pro verifikaci protokolů v praktické části této práce. Nástroj AVISPA sestává ze čtyř různých analyzátorů. Ty jsou zde pro nedostatek místa popsány pouze stručně a výklad je zaměřen spíše na architekturu nástroje a základy specifikačního jazyka HLPSL. V závěru této kapitoly je uvedeno srovnání jednotlivých nástrojů.

5.1 LySa

LySa je procesní kalkul (procesní algebra) pro analýzu bezpečnostních protokolů. Je odvozen od pi-kalkulu [32] a spi-kalkulu [1]. Oproti nim se liší zejména v počtu komunikačních kanálů – LySa uvažuje existenci pouze jednoho globálního kanálu (ether), k němuž mají přístup všechny procesy. Tento předpoklad tedy více odpovídá praxi, kdy je často k dispozici pouze jedno komunikační médium přístupné komukoliv (i útočníkům). V případě požadavku na tajné kanály (např. u intranetových sítí), je však možné LySa kalkul pro tyto účely rozšířit. Druhým rozdílem je způsob kontroly vstupu a šifrování, kdy LySa používá tzv. porovnávání vzorů (pattern matching).

Analýza protokolů pomocí LySa kalkulu patří mezi statické metody (konkrétně jde o analýzu toku řízení). Dokáže zaručit splnění požadavků na důvěrnost a autentizaci v případě provedení prakticky libovolného typu útoku. Předpokládá se zde perfektní kryptografie (zprávu nelze šifrovat a dešifrovat bez znalosti klíče), možnost provedení útoku hrubou silou se tedy neuvažuje. Integrita je zaručena předpokladem perfektní kryptografie. Implementací LySa kalkulu je nástroj LySatool (viz. dále).

5.1.1 Syntaxe

Základem LySa kalkulu jsou termy a procesy. Jejich syntaxe je následující:

E ::=		termy
	n	jméno ($n \in N$)
	x	proměnná ($x \in X$)
	k^+, k^-	soukromý a veřejný klíč
	$\{ E_1, \ldots, E_k \}^{\mathbf{I}}_{E_0}$ [dest L]	symetrické šifrování ($k \geq 0$)
	$\{\| E_1, \ldots, E_k \|\}^{\mathbf{I}}_{E_0}$ [dest L]	asymetrické šifrování ($k \geq 0$)

P ::=	procesy
0	nil
$< E_1, \ldots, E_k >.P$	výstup
$(E_1, \ldots, E_j ; x_{j+1}, \ldots, x_k).P$	vstup (s porovnáváním vzorů)
$P_1 \mid P_2$	paralelní kompozice
$(\nu\, n) P$	omezení
$(\nu_{\pm}\, m) P$	vytvoření páru klíčů
$!P$	replikace
decrypt E as $\{ E_1, \ldots, E_l ; x_{j+1}, \ldots, x_k \}^{\mathbf{l}}_{E_0}$ [orig L] in P	symetrické dešifrování s porovnáváním vzorů ($k \geq 0$)
decrypt E as $\{\mid E_1, \ldots, E_l ; x_{j+1}, \ldots, x_k \mid\}^{\mathbf{l}}_{E_0}$ [orig L] in P	asymetrické dešifrování s porovnáváním vzorů ($k \geq 0$)

Termy modelují zprávy, jež se skládají ze jmen, proměnných a klíčů. Jména slouží například pro označení účastníků komunikace. Symbol *N* označuje množinu jmen, *X* množinu proměnných. K-tice termů ($E_1, \ldots, E_k$) je šifrována (dešifrována) klíčem reprezentovaným termem E_0.

Proces modeluje vstup, výstup, vytváření klíčů a dešifrování. Procesy lze zpracovávat paralelně, lze je replikovat a ukončovat.

V případě vstupu a dešifrování se provádí tzv. porovnávání vzorů (pattern matching), kdy se termy testují na shodu s předpokládanou hodnotou. Příjemce tak má například možnost zjistit, zda je zpráva skutečně určena pro něj, případně zkontrolovat identitu odesílatele (podle předchozí komunikace). Termy určené pro porovnávání tvoří prefix dešifrované k-tice (resp. k-tice na vstupu). Ostatní termy jsou odděleny středníkem. Jejich hodnoty se pouze navážou na proměnné (porovnávání se neprovádí).

Pro kontrolu autentizačních vlastností se používají tzv. crypto-points. Crypto-point **l** je prvkem nějaké spočetné množiny *C* (disjunktní s *N* a *X*). Označuje místo ve specifikaci protokolu, kde dochází k šifrování (resp. dešifrování). Současně je v závorce uvedeno, na kterých místech může dojít k dešifrovaní zprávy ([*dest L*]), respektive kde byla zpráva zašifrována ([*orig L*]). Množina *L* je podmnožinou *C* ($L \subseteq C$).

5.1.2 Operační sémantika

Operační sémantika je založena na následujících odvozovacích pravidlech:

$$\frac{\wedge_{i=1}^{j} \S E_i \text{"} = \S E'_i \text{"}}{< E_1, \ ..., \ E_k >.P \mid (E'_1, \ ..., \ E'_j \ ; \ x_{j+1}, ..., x_k).Q \rightarrow_R P \mid Q \ [E_{j+1} / x_{j+1}, \ ..., E_k / x_k]}$$

(Com)

$$\frac{\wedge_{i=0}^{j} \S E_i \text{"} = \S E'_i \text{"} \ \wedge \ R(\mathbf{l}, L', \mathbf{l}', L)}{\text{decrypt } (\{ E1, ..., Ek \}_{E_0}^{\mathbf{l}} [\text{dest } L]) \text{ as } \{E'_1, ..., E'_j \ ; x_{j+1}, ..., x_k \}_{E'_0}^{\mathbf{l}'} [\text{orig } L'] \text{ in } P \rightarrow_R P[E_{j+1} / x_{j+1}, \ ..., E_k / x_k]}$$

(Decr)

$$\frac{\wedge_{i=1}^{j} \S E_i \text{"} = \S E'_i \text{"} \ \wedge \ \{ E_0, E'_0 \} = \{ m^{\pm}, m^{\mathbb{m}} \} \ \wedge \ R(\mathbf{l}, L', \mathbf{l}', L)}{\text{decrypt } (\{|E1, ..., Ek|\}_{E_0}^{\mathbf{l}} [\text{dest } L]) \text{ as } \{|E'_1, ..., E'_j \ ; x_{j+1}, ..., x_k |\}_{E'_0}^{\mathbf{l}'} [\text{orig } L'] \text{ in } P \rightarrow_R P[E_{j+1} / x_{j+1}, \ ..., E_k / x_k]}$$

(As-decr)

$$\frac{P \rightarrow_R P'}{P \mid Q \rightarrow_R P' \mid Q} \qquad \frac{P \rightarrow_R P'}{(\nu \ n) P \rightarrow_R (\nu \ n) P'} \qquad \frac{P \rightarrow_R P'}{(\nu_{\pm} \ m) P \rightarrow_R (\nu_{\pm} \ m) P'} \qquad \frac{P \equiv Q \wedge Q \rightarrow_R Q' \wedge Q' \equiv P'}{P \rightarrow_R P'}$$

(Par) (Res) (As-res) (Congr)

První část pravidla je tzv. předpoklad, který musí být splněn pro provedení pravidla pod čarou. Symboly $\S.\text{"}$ označují term s odstraněnými anotacemi. Redukční relace $P \rightarrow_R P'$ vyjadřuje provedení přechodu z procesu *P* do procesu *P'* v případě, že je splněn parametr *R*. V základě existují dvě varianty operační sémantiky:

- Standardní sémantika – parametr *R* je vždy považován za splněný (pravdivý).
- Referenční monitor – parametr *R* je nahrazen parametrem *RM* definovaným jako $RM\ (\mathbf{l}, L', \mathbf{l}', L) = (\mathbf{l} \in L' \wedge \mathbf{l}' \in L)$. V případě porušení podmínek je proces přerušen.

Substituce a kongruence struktury zde mají stejný význam jako v pi-kalkulu:

- Substituce *P*[*E*/*X*] je nahrazení všech volných výskytů prvku *E* v procesu *P* prvkem *X*.
- Mezi dvěma procesy existuje strukturní kongruence ≡ v případě, že jsou identické až po strukturní úroveň.

Pravidlo *Com* vyjadřuje výše zmíněné porovnávání vzorů. Pokud je prvních *j* prvků *E* na vstupu a výstupu po párech shodných, dojde k navázání prvků E_i na proměnné x_i (pro $i = j + 1 .. k$). Podobně pravidlo *Decr* (resp. *As-decr* pro asymetrické šifrování) porovnává prvky *E* v případě šifrování a dešifrování. Následují pravidla pro paralelní kompozici (*Par*), asymetrickou paralelní kompozici (*As-par*), omezení (*Res*) a kongruenci (*Congr*).

5.1.3 Specifikace protokolu

Pro zápis protokolu v LySa kalkulu je vhodné nejprve rozšířit jeho standardní formální specifikaci. Rozšířená specifikace (extended narration) rozlišuje mezi vstupy a korespondujícími výstupy, resp. šifrováním a korespondujícím dešifrováním. Na vstupech je také možné explicitně uvádět kontrolu obdržených hodnot. Každá zpráva je na začátku doplněna o označení odesílatele a příjemce.

Specifikace protokolu ve standardní podobě je obvykle založena na oddělených rolích jednotlivých účastníků – odesílatel zprávy (*A*), příjemce zprávy (*B*), server (*S*) apod. LySa specifikace typicky předpokládá existenci libovolného množství účastníků, přičemž každý může vykonávat jak roli iniciátora komunikace, tak roli příjemce. Dále se předpokládá, že server s každým účastníkem sdílí šifrovací klíč. Útočník se může chovat jako legitimní účastník komunikace.

Příklad rozšířeného zápisu protokolu a jeho specifikace v LySa kalkulu je uveden v případové studii (kapitola 5.1.8).

5.1.4 Analýza toku řízení

Analýza bezpečnostních protokolů pomocí LySa kalkulu je založena na aproximaci (over-approximation) množiny všech zpráv na síti a množin potenciálních hodnot každé z proměnných. Dále je zaznamenáváno případné porušení podmínek určených body pro šifrování a dešifrování (crypto-points). Jak již bylo zmíněno, jde o statickou metodu – analýza se provádí aniž by byla vstupní specifikace protokolu spouštěna jako program.

Hlavní komponenty analýzy jsou tyto:

- Prostředí proměnných (variable environment) ρ pro každou proměnnou poskytuje aproximaci jejích možných hodnot.
- Síťové prostředí (network environment) κ je aproximací množiny všech zpráv na síti.
- Komponenta chyb (error component) ψ poskytuje aproximaci potenciálních porušení podmínek definovaných krypto-pointy. Například pokud $(\mathbf{l}, \mathbf{l}') \in \psi$ znamená to, že zpráva zašifrovaná v bodě $\mathbf{l}$ byla neočekávaně dešifrována v bodě $\mathbf{l}'$ nebo že zpráva dešifrovaná v bodě $\mathbf{l}'$ měla být zašifrována v jiném bodě než $\mathbf{l}$.

Útočník, založený na modelu Dolev-Yao [33], může provádět tyto akce:

- odposlouchávat zprávy,
- dešifrovat zprávy jemu známými klíči,
- šifrovat zprávy jemu známými klíči,
- iniciovat komunikaci s legitimními účastníky.

Zpráva zašifrovaná / dešifrovaná útočníkem je v analýze označena speciálním krypto-pointem $\mathbf{l}_\bullet$. Předpokládá se, že útočník disponuje nějakými prvotními znalostmi o komunikaci. Ty mohou být v průběhu analýzy doplněny dalšími znalostmi. Veškeré znalosti útočníka jsou uchovávány ve speciální proměnné $x_\bullet$.

Analýza uvažuje různé typy útoků – odposlouchávání, modifikaci, přehrávání, atd., případně jejich kombinace. Útočník také může vystupovat jako legitimní účastník komunikace.

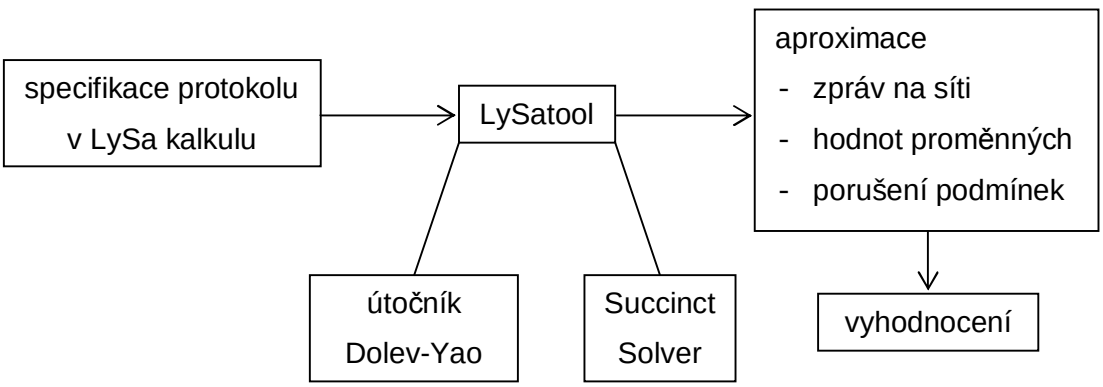
Pokud je po provedení analýzy protokolu komponenta chyb ψ prázdná, znamená to, že nedošlo k porušení podmínek definovaných krypto-pointy a tím pádem je zaručen požadavek na autentizaci. Podobně lze prověřením prostředí proměnných ρ (zejména pak proměnné $x_\bullet$) zjistit, zda protokol splňuje požadavek na důvěrnost.

Jelikož je analýza založena na aproximaci nad chováním protokolu, výsledek může obsahovat chyby, které ve skutečnosti nejsou chybami návrhu protokolu.

5.1.5 LySatool

Implementací LySa kalkulu je nástroj LySatool. Je napsán v programovacím jazyce Standard ML of New Jersey (SML/NJ). Pro analýzu využívá program Succinct Solver.

Vstupní specifikace protokolu je v průběhu analýzy převedena do logiky ALFP (Alteration-free Least Fixed Point Logic) [39], která je rozšířením Hornových klauzulí [36]. Transformace zahrnuje kódování (potenciálně nekonečných) množin termů do stromových gramatik. Takto definovaný problém je vstupem nástroje Succinct Solver, který v polynomiálním čase nalezne jeho nejmenší řešení.



Obr. 6: Analýza protokolu pomocí nástroje LySatool

Gramatika LySa procesu

Vstupem programu je specifikace LySa procesu v ASCII formátu, který je definován následující gramatikou:

proc	::=	(proc) | < term* > . proc | (term* ; var*) . proc |
		decrypt term as { term* ; var* } : term orig in proc |
		decrypt term as {| term* ; var* |} : term orig in proc |
		(new name) proc | (new +– name) proc |
		! proc | proc | proc | 0 |
		let identifier subset iset in proc | |_{ assign } proc |
		(new_{ assign+ } name) proc | (new_{ assign+ } +– name) proc
term	::=	(term) | { term* } : term dest | {| term* |} : term dest |
		name | namep | namem | var
name	::=	identifier subscript
namep	::=	identifier + subscript
namem	::=	identifier – subscript
var	::=	identifier subscript
subscript	::=	_{ index* } | ε
index	::=	identifier | number
iset	::=	{ index* } | iset union iset |
		NATURAL1 | NATURAL2 | NATURAL3 |
		NATURAL01 | NATURAL02 | NATURAL03 | ZERO
assign	::=	index in number
dest	::=	[at cryptopoint dest { cryptopoint* }] | [at cryptopoint] | ε
orig	::=	[at cryptopoint orig { cryptopoint* }] | [at cryptopoint] | ε
cryptopoint	::=	identifier subscript | CPDY

Identifikátor (*identifier*) je řetězec, který musí začínat písmenem. Následovat může jedno či více písmen nebo číslic. *Číslo* (*number*) je rovněž řetězec reprezentující nezáporné číslo.

Následující řetězce jsou považovány za klíčová slova, nemohou být tedy použity k jinému než gramatikou danému účelu:

as, at, CPDY, dest, decrypt, in, let, orig, NATURAL1, NATURAL2, NATURAL3, NATURAL01, NATURAL02, NATURAL03, new, subset, ZERO.

Jakýkoliv řetězec uzavřený mezi symboly **/*** a ***/** je považován za poznámku. Non-terminál označený symbolem + značí posloupnost jednoho nebo více těchto non-terminálů oddělených středníkem. Non-terminál se symbolem * značí taktéž posloupnost těchto non-terminálů, ta však může být i prázdná. Symbol ε v těle pravidla představuje prázdný řetězec.

Operátory vstupu, výstupu, dešifrování a omezení mají vyšší prioritu než replikace, která má vyšší prioritu než paralelní kompozice. Priority lze upravit použitím závorek.

Body pro kontrolu šifrování a dešifrování (crypto-points) jsou uzavřeny do hranatých závorek. Pokud jsou vynechány, předpokládá se, že kontrola není požadována.

Parametry

Proces analýzy je možné ovlivnit nastavením následujících parametrů (implicitní hodnoty jsou uvedeny v závorce):

withAttacker: bool (true)

Pokud je tento parametr nastaven na *true*, vstupní proces je analyzován s předpokladem existence útočníka. V opačném případě se analýza provádí bez jeho účasti.

attackerIndex: string (“”)

V případě, že se pro rozlišení legitimních účastníků komunikace používají indexy a je požadavkem modelovat útočníka jako jednoho z nich, lze tímto parametrem nastavit, jakým indexem bude tento útočník označen.

mergeExpressionLabels: bool (true)

Výrazům LySa procesu jsou automaticky přiřazována označení (labels). Tato označení pak představují non-terminály ve stromových gramatikách. Pokud je parametr *mergeExpressionLabels* nastaven na *true*, k identickým výrazům je přiřazeno stejné označení. To může přispět ke zmenšení velikosti stromových gramatik a tedy i k rychlejší analýze. Pokud je parametr nastaven na *false*, označení jsou jedinečná pro každý výraz.

mergeInputVariables: bool (true)

Výstupem analýzy protokolu je také množina znalostí, které může případný útočník získat. Pokud je parametr *mergeInputVariables* nastaven na *false*, je každá tato hodnota reprezentována jednou proměnnou. Pokud je nastaven na *true*, jsou všechny tyto znalosti sloučeny do proměnné $x_\bullet$. Výhodou je zmenšení objemu dat na výstupu, což se také může projevit na rychlosti analýzy.

Spuštění analýzy

Analýzu lze spustit pomocí skriptu *run.sml*. Nejprve je potřeba jej v SML/NJ zavést do paměti příkazem

```
- use("run.sml");
```

Poté je možné skript použít ke spuštění analýzy LySa procesu jehož specifikace je uložena v souboru:

```
- run("otway-rees.lysa");
```

Analýzu je také možné spustit s jiným než implicitním nastavením parametrů (viz výše). K tomu slouží funkce *runp*, která očekává parametry jako druhý argument:

```
- runp("otway-rees.lysa", Analysis2.version1parameters);
```

Parametry se nastavují v souboru *Analysis2.sml*. Bližší informace lze nalézt v [37].

Výstup analýzy

Výstupem analýzy je soubor (stejného jména a umístění jako vstupní soubor) ve formátu HTML, ASCII či LATEX. Obsahuje vstupní specifikaci a výsledky analýzy – prostředí proměnných ρ, síťové prostředí κ a komponentu chyb ψ.

Pokud je komponenta chyb neprázdná ($\psi \neq \emptyset$), znamená to, že došlo k porušení podmínek definovaných krypto-pointy, čili k porušení autentizace. Výstup závisí na nastavení parametrů analýzy. Pokud například útočník není modelován jako legitimní účastník, komponenta chyb může obsahovat porušení typu (*CPDY*, s_{01}). Konkrétně tato zpráva znamená, že útočník vystupoval jako subjekt 0, inicioval komunikaci se subjektem 1 a v nějakém bodě musel server dešifrovat zprávu zašifrovanou útočníkem (*CPDY* značí krypto-point útočníka). Toto však nepředstavuje prolomení protokolu a porušení tohoto typu mohou být ve výsledcích analýzy vypuštěna nastavením indexu útočníka (viz. výše).

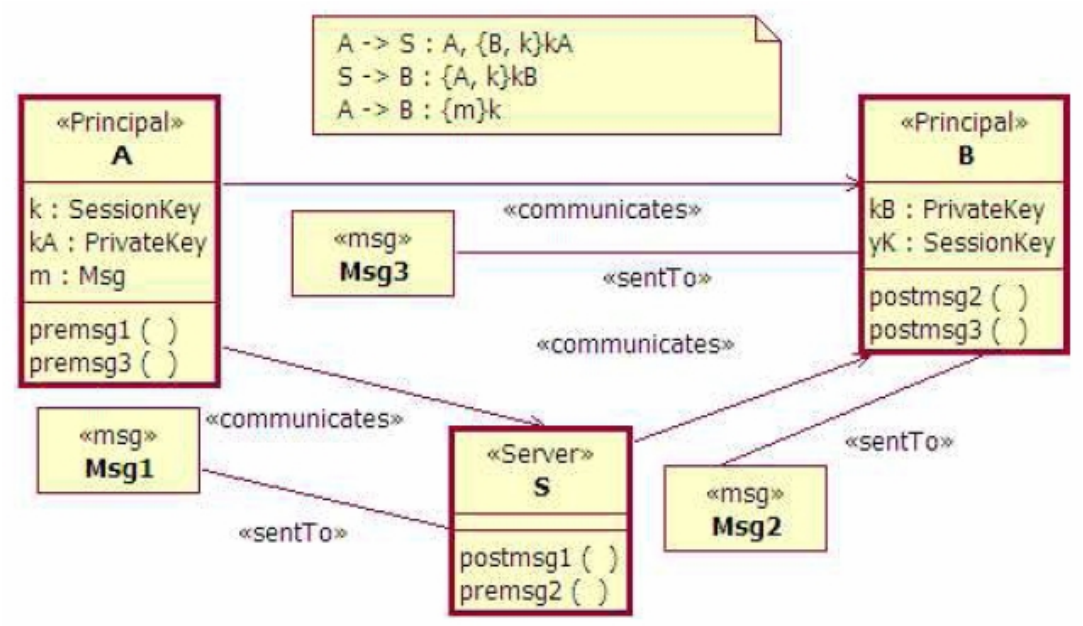
Prověřením prostředí proměnných ρ (zejména pak proměnné $x_\bullet$) lze zjistit, zda protokol splňuje požadavek na důvěrnost. Proměnná $x_\bullet$ představuje veškeré získané znalosti útočníka, měla by tedy obsahovat pouze veřejné hodnoty.

5.1.6 Doplňky

Pro ulehčení specifikace LySa procesu je možné využít k tomu určené nástroje. Jedná se o programy For-LySa [40] a Elyjah [42].

For-LySa

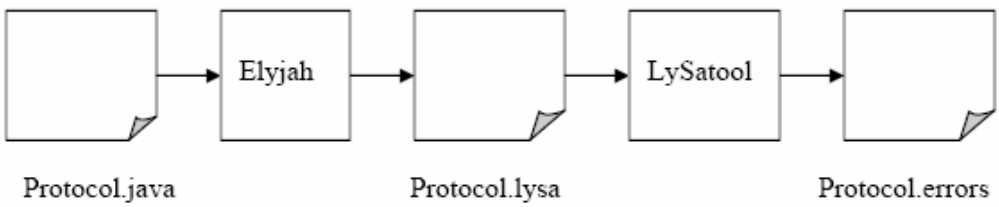
Tento nástroj je schopen převést specifikaci protokolu v UML (Unified Modeling Language) [41] do LySa kalkulu. Na obrázku 7 je ukázka UML specifikace protokolu Wide Mouthed Frog. Podrobnější informace o použití programu jsou v [40].



Obr. 7: Specifikace protokolu Wide Mouthed Frog v UML [40]

Elyjah

Elyjah je další nástroj pro zjednodušení specifikace protokolu. Vstupem je protokol implementovaný v programovacím jazyce Java, výstupem pak LySa specifikace pro LySatool. Tento proces je znázorněn na obrázku 8. Ačkoliv má program různá syntaktická omezení, je poměrně dobře použitelný. Více informací lze nalézt v [42].



Obr. 8: Postup analýzy protokolu s využitím nástroje Elyjah [42]

5.1.7 Shrnutí

Výhody

- Poměrně jednoduchá specifikace LySa procesu.
- Existence podpůrných nástrojů (Elyjah, For-LySa).
- Nástrojem již byla verifikována řada protokolů. Kromě již dříve odhalených chyb byly nalezeny i chyby nové (např. v protokolu MSR).
- Ačkoliv nástroje pracující na podobných principech mají většinou exponenciální časovou složitost, analýza pomocí LySatool je provedena v polynomiálním čase.
- Pro konkrétní účely je možné LySa kalkul poměrně jednoduše upravit / rozšířit ($LySa^{XP}$ [17], $LySa^{NS}$ [21]).

Nevýhody

- Výsledek analýzy může obsahovat chyby, které ve skutečnosti nejsou chybami návrhu protokolu. To je způsobeno použitou technikou aproximace nad chováním protokolu.
- LySatool neumí v případě nalezení chyby vygenerovat popis útoku.

5.1.8 Případová studie

Tato kapitola obsahuje ukázku analýzy protokolu Otway-Rees pomocí nástroje LySatool. Protokol byl již popsán v kapitole 3.1.1, zde je však uvažována jeho opravená verze, která šifruje nonces *Na* a *Nb* a navíc používá nonce *M* pro identifikaci relace.

Formální specifikace protokolu:

1. $A \rightarrow B : M, A, B, \{ Na, M, A, B \}_{Ka}$
2. $B \rightarrow S : M, A, B, \{ Na, M, A, B \}_{Ka}, \{ Nb, M, A, B \}_{Kb}$
3. $S \rightarrow B : M, \{ Na, Kab \}_{Ka}, \{ Nb, Kab \}_{Kb}$
4. $B \rightarrow A : M, \{ Na, Kab \}_{Ka}$

Kvůli porovnávání vzorů je nutné nejprve přeuspořádat některé n-tice tak, aby hodnoty, jež se mají navázat na proměnné, byly až na jejich konci. Dále je vhodné přidat do specifikace další krok vyjadřující vlastní komunikaci mezi subjekty *A* a *B* s nyní již sdíleným relačním klíčem *Kab*.

Upravená specifikace protokolu:

1. A → B : M, A, B, { M, A, B, Na }$_{Ka}$
2. B → S : M, A, B, { M, A, B, Na }$_{Ka}$, { M, A, B, Nb }$_{Kb}$
3. S → B : M, { Na, Kab }$_{Ka}$, { Nb, Kab }$_{Kb}$
4. B → A : M, { Na, Kab }$_{Ka}$
5. B → A : { MSG }$_{Kab}$

Pro snadnější přepis protokolu do LySa kalkulu je vhodné nejprve vytvořit jeho rozšířenou specifikaci (kapitola 5.1.3). Doplňující poznámky *check* v hranatých závorkách značí kontrolu hodnot na vstupech a korespondují s porovnáváním vzorů v LySa specifikaci. Na rozdíl od kontroly krypto-pointů, kde se předkládá globální pohled, kontrolu na vstupech lze provádět pouze na základě lokálních znalostí příjemce zprávy.

Rozšířená specifikace protokolu:

1. A → : A, B, M, A, B, { M, A, B, Na }$^{a1}_{Ka}$ [dest s1]
1′. → B : y_A, y_B, y_M, y_A', y_B', y1 [check y_B = B, y_A = y_A', y_B = y_B']
2. B → : B, S, y_M, y_A, B, y1, { y_M, y_A, B, Nb }$^{b1}_{Kb}$ [dest s2]
2′. → S : z_B, z_S, z_M, z_A, z_B', z1, z2 [check z_S = S, z_B = z_B']
2″. S : decrypt z1 as { z_M', z_A', z_B'', z_{Na} }$^{s1}_{Ka}$ [orig a1]
2‴. S : decrypt z2 as { z_M'', z_A'', z_B''', z_{Nb} }$^{s2}_{Kb}$ [orig b1]
3. S → : S, z_B, z_M, { z_{Na}, Kab }$^{s3}_{Ka}$ [dest a2], { z_{Nb}, Kab }$^{s4}_{Kb}$ [dest b2]
3′. → B : y_S, y_B'', y_M', y2, y3 [check y_B'' = B, y_S = S]
3″. B : decrypt y3 as { y_{Nb}, y_{Kab} }$^{b2}_{Kb}$ [orig s4]
4. B → : B, y_A, y_M', y2
4′. → A : x_B, x_A, x_M, x1 [check x_A = A, x_B = B]
4″. A : decrypt x1 as { x_{Na}, x_{Kab} }$^{a2}_{Ka}$ [orig s3]
5. B → : B, y_A, { MSG }$^{b3}_{y_{Kab}}$
5′. → A : x_B', x_A', x2 [x_A' = A, check x_B' = B]
5″. A : decrypt x2 as { xmsg }$^{a3}_{x_{Kab}}$ [orig b3]

Z rozšířené specifikace lze již poměrně jednoduše vytvořit zápis v LySa kalkulu. Předpokládá se typická situace, kdy se libovolné množství iniciátorů (A_0, A_1, A_2, …) snaží o komunikaci s libovolným množstvím příjemců (B_0, B_1, B_2, …). Každý z těchto subjektů sdílí se serverem tajný klíč. Dále se předpokládá, že útočník může vystupovat jako legitimní účastník komunikace (A_0 nebo

B_0) a navázat spojení s kterýmkoliv dalším subjektem. Pro roli iniciátora se používá jiná adresa a jiný klíč než pro roli příjemce ($A_i \neq B_i$, $Ka_i \neq Kb_i$).

Aby bylo možné modelovat i útoky ze středu (kapitola 2.8.4), subjekty jsou rozděleny do tří skupin, přičemž v každé z nich je libovolné (nekonečné) množství subjektů. Specifikace je rozdělena na bloky podle rolí účastníků v tomto pořadí: iniciátoři, příjemci, server.

LySa specifikace:

let $X \subseteq$ ¥ s.t. $\lfloor$ ¥ $\rfloor$ = { 1, 2, 3 } in

$(\nu_{i\in X}\ Ka_i)\ (\nu_{j\in X}\ Kb_j)$

$|_{i\in X}\ |_{j\in X\cup\{0\}}\ !(\nu\ Na_{ij})$

1. $< A_i, B_j, M_{ij}, A_i, B_j, \{ M_{ij}, A_i, B_j, Na_{ij} \}^{a1_{ij}}_{Ka}\ [\text{dest } s1_{ij}] >.$

4′. $(B_j, A_i, M_{ij} ; x1_{ij}).$

4″. decrypt $x1_{ij}$ as $\{ Na_{ij} ; x_{Kab_{ij}} \}^{a2_{ij}}_{Ka_i}\ [\text{orig } s3_{ij}]$ in

5′. $(B_j, A_i ; x2_{ij}).$

5″. decrypt $x2_{ij}$ as $\{ ; xmsg_{ij} \}^{a3_{ij}}_{x_{Kab_{ij}}}\ [\text{orig } b3_{ij}]$ in 0

|

$|_{i\in X\cup\{0\}}\ |_{j\in X}\ !(\nu\ Nb_{ij})$

1′. $(A_i, B_j, M_{ij}, A_i, B_j ; y1_{ij}).$

2. $< B_j, S, M_{ij}, A_i, B_j, y1_{ij} , \{ M_{ij}, A_i, B_j, Nb_{ij} \}^{b1_{ij}}_{Kb_j}\ [\text{dest } s2_{ij}] >.$

3′. $(S, B_j, M_{ij} ; y2_{ij}, y3_{ij}).$

3″. decrypt $y3_{ij}$ as $\{ Nb_{ij} ; y_{Kab_{ij}} \}^{b2_{ij}}_{Kb_j}\ [\text{orig } s4_{ij}]$ in

4. $< B_j, A_i, M_{ij}, y2_{ij} >.$

5. $(\nu\ MSG_{ij}) < B_j, A_i, \{ MSG_{ij} \}^{b3_{ij}}_{y_{Kab_{ij}}}\ [\text{dest } a3_{ij}] >.0$

|

$|_{i\in X\cup\{0\}}\ |_{j\in X\cup\{0\}}\ !$

2′. $(B_j, S, M_{ij}, A_i, B_j ; z1_{ij}, z2_{ij}).$

2″. decrypt $z1_{ij}$ as $\{ M_{ij}, A_i, B_j ; z_{Na_{ij}} \}^{s1_{ij}}_{Ka_i}\ [\text{orig } a1_{ij}]$ in

2‴. decrypt $z2_{ij}$ as $\{ M_{ij}, A_i, B_j ; z_{Nb_{ij}} \}^{s2_{ij}}_{Kb_i}\ [\text{orig } b1_{ij}]$ in

3. $(\nu\ Kab_{ij}) < S, B_j, M_{ij}, \{ z_{Na_{ij}}, Kab_{ij} \}^{s3_{ij}}_{Ka_i}\ [\text{dest } a2_{ij}], \{ z_{Nb_{ij}}, Kab_{ij} \}^{s4_{ij}}_{Kb_j}\ [\text{dest } b2_{ij}] >.0$

Odpovídající specifikace v ASCII formátu je uvedena v příloze 1.

Výsledky analýzy

V tomto případě, kdy se pro roli iniciátora používá jiná adresa a klíč než pro roli příjemce ($A_i \neq B_i$, $Ka_i \neq Kb_i$) a útočník je modelován jako legitimní účastník komunikace s indexem 0, je komponenta chyb na výstupu prázdná ($\psi = \varnothing$). Podmínky určené krypto-pointy porušeny nebyly a autentizace je tedy zaručena. Množina potenciálních znalostí útočníka (proměnná $x_\bullet$) obsahuje pouze veřejné hodnoty, tzn., že jsou splněny i požadavky na důvěrnost.

5.2 Athena

Athena je plně automatický nástroj pro verifikaci bezpečnostních protokolů založený na kontrole modelem (kapitola 4.1) a dokazování teorému (kapitola 4.3). Kromě požadavku na autentizaci a důvěrnost lze jednoduchou logikou specifikovat například vlastnosti vztahující se k bezpečnosti v elektronickém bankovnictví.

Požadovaná vlastnost se nejprve ověří v počátečním stavu. Poté se vytvoří stromová struktura, která se prochází a v případě nalezení stavu, ve kterém vlastnost neplatí, je proces zastaven. Zároveň je vygenerován popis úspěšného útoku. Pokud není takový stav nalezen, je dokázána korektnost.

Ukončení verifikace není zaručeno, je však možné jej vynutit omezením počtu současných běhů protokolu a délky zpráv.

Athena používá efektivních technik k redukci stavového prostoru. Je založena na rozšířeném modelu SSM (Strand Space Model) [46], který obsahuje relace s přesnými informacemi o původu zprávy. Díky tomuto přístupu a použití zpětného prohledávání (backward search), nedosažitelných hypotéz (unreachability theorems) a dalších technik je možné zamezit tzv. explozi stavového prostoru (state space explosion), kdy počet stavů roste exponenciálně s počtem účastníků. Problém exploze stavového prostoru se objevuje u většiny ostatních nástrojů založených na TBM (Trace Based Model), kde je protokol modelován pomocí procesů a zpráv.

Technika zpětného prohledávání je založena na myšlence dynamického vytváření nových vláken (strands) pouze v případě potřeby. Pomocí nedosažitelných hypotéz lze z množiny procházených stavů odstranit ty, kterých nelze dosáhnout.

Nástroj Athena je implementován v jazyce SML/NJ a je dostupný pro Windows i Linux. Byl použit k verifikaci mnoha bezpečnostních protokolů – Needham-Schroeder, TMN, 1KP, Kerberos, atd. V tabulce 2 je pro srovnání s ostatními nástroji uveden počet procházených stavů při verifikaci protokolu Needham-Schroeder-Lowe.

Počet iniciátorů	Počet příjemců	Murphi	Brutus (1)	Brutus (2)	Athena
1	1	1706	1208	146	19
2	2	514550	------	186340	19

Tab. 2: Srovnání nástrojů – počet procházeným stavů při verifikaci protokolu NSL (1 – Brutus bez použití symetrické redukce, 2 – Brutus s použitím symetrické redukce)

5.2.1 Strand Space Model (SSM)

SSM je prostředkem pro manuální dokazování bezpečnostních požadavků. Athena tento model rozšiřuje a principy pro analýzu protokolů automatizuje. Základem SSM modelu jsou termy, akce, události, vlákna, prostory vláken a balíky.

Termy (Terms)

Množina atomických termů je sjednocením množiny textových termů (text terms) a klíčových termů (key terms):

- Textové termy – např. jména účastníků, nonces, …
- Klíčové termy – množina klíčů; K^{-1} je inverzní klíč ke klíči K.

Množina termů A je definována induktivně takto:

- Jestliže m je textový term nebo klíčový term, pak m je term.
- Jestliže m je term a K je klíč, pak $\{ m \}_K$ (šifrování) je term.
- Jestliže m_1 a m_2 jsou termy, pak $m_1 . m_2$ (konkatenace) je term.

Předpokládá se, že

$$\{ m \}_K = \{ m' \}_K \Leftrightarrow m = m' \wedge K = K' .$$

SSM model obsahuje relaci $\sqsubseteq$ (subterm relation), Athena doplňuje relaci $\Subset$ (interm relation):

- Relace $\sqsubseteq$ (subterm relation) – term a_1 je v relaci $\sqsubseteq$ s termem a_2, pokud se a_1 vyskytuje v a_2.
- Relace $\Subset$ (interm relation) – term a_1 je v relaci $\Subset$ s termem a_2, pokud a_1 může být získán z a_2 bez použití dešifrování.

Akce (Actions)

Množina akcí *Act*, které mohou účastníci komunikace provádět, obsahuje základní akce odeslání (*send*) a příjem (*receive*). Doplňkovými akcemi jsou například *debit*, *credit*, atd. Pro zjednodušení se akce *send* značí symbolem + a akce *receive* symbolem -.

Události (Events)

Událost je dvojice < *akce*, *a* >, kde *akce* $\in$ *Act* a *a* je argument akce z množiny *A* ($a \in A$). V případě použití akcí *send* (+) a *receive* (-), se události zapisují jako označené termy (signed terms) $+a$ a $-a$. Množina událostí (označených termů) se značí $\pm A$ a množina konečných sekvencí označených termů $(\pm A)^*$.

Vlákna a prostory vláken (Strands, Strand Spaces)

Vlákno (strand) je sekvence událostí prováděných subjektem (instancí role). Prostor vláken (strand space) je množina vláken Σ s mapováním *tr*: $\Sigma \rightarrow (\pm A)^*$.

1. Uzel (node) je dvojice < *s*, *i* >, kde *s* $\in$ Σ a *i* je číslo splňující podmínku $1 \leq i \leq$ *délka*(*tr*(*s*)). Množina uzlů se značí *N*.
2. Jestliže *n* = < *s*, *i* > $\in$ *N*, pak *index*(*n*) = *i* a *strand*(*n*) = *s*. Jestliže $(\, tr(s)\,)_i = \sigma\, a$, kde $\sigma \in \{ +, - \}$, pak *term*(*n*) = *a* a *sign*(*n*) = σ. Jinými slovy, uzel je konkrétní událost ve vlákně.
3. Jestliže $n_1, n_2 \in N$, pak $n_1 \rightarrow n_2$ znamená, že $n_1 = +a$ a $n_2 = -a$ pro nějaký term *a*. Toto představuje odeslání a příjem zprávy z n_1 do n_2.
4. Jestliže $n_1, n_2 \in N$, pak $n_1 \Rightarrow n_2$ znamená, že n_1 a n_2 se vyskytují ve stejném vlákně a platí, že *index*(n_2) = *index*(n_1) + 1. Jinými slovy, událost n_1 je bezprostředně následována událostí n_2.
5. Term *t* vzniká (originates) v uzlu *n* $\in$ *N*, když a jen když *sign*(*n*) = + a $t \sqsubseteq term(n)$ a kdykoli *n′* předchází *n* ve stejném vlákně, $t \not\sqsubseteq term(n')$.
6. Term *t* jedinečně vzniká (uniquely originates) v uzlu *n*, když a jen když vzniká v jedinečném uzlu *n*. Takto obvykle vznikají nonces a další čerstvě generované termy.

N značí také orientovaný graf (*N*, *E*), jehož vrcholy jsou uzly a $E = (\rightarrow \cup \Rightarrow)$ je množina hran.

Balík (Bundle)

Balík specifikuje provádění protokolu s určitým nastavením. Balík $C = (N_C, E)$ je podgrafem *N*, kde $E \subseteq (\rightarrow \cup \Rightarrow)$ a $N_C \subseteq N$. Platí, že:

- *C* je neprázdný, konečný a acyklický.
- Jestliže $n_1 \in C$ a *sign*(n_1) = -, pak existuje jedinečné n_2 takové, že $n_2 \rightarrow n_1 \in C$.
- Jestliže $n_1 \in C$ a $n_2 \Rightarrow n_1$, pak $n_2 \Rightarrow n_1 \in C$.

Vlákno *s* patří do balíku *C*, pokud všechny jeho uzly patří do balíku *C*.

Příčinná závislost

Nechť *S* je prostor vláken, uzly $n_1, n_2 \in S$. Uzel n_1 je v relaci příčinné závislosti s n_2 ($n_1 \preceq_S n_2$), když a jen když existuje sekvence nula a více hran typu $\rightarrow$ a $\Rightarrow$ vedoucích z n_1 do n_2 v *S*.

5.2.2 Specifikace protokolu v SSM

Protokol obvykle sestává z několika různých rolí (iniciátor, příjemce, server). Sekvence událostí každé role (vlákno) je podmíněna seznamem parametrů, které se zapisují za jméno role v hranatých závorkách - *role*[*seznam parametrů*]. Provedení protokolu vytváří balík. Vlákna legitimních účastníků se označují jako regulérní vlákna (regular strands). Balík obsahuje také vlákna útočníků (penetrator strands).

Příklad – protokol Needham-Schroeder-Lowe

Formální specifikace protokolu:

1. $A \rightarrow B : \{ Na, A \}_{Kb}$
2. $B \rightarrow A : \{ Na, Nb, B \}_{Ka}$
3. $A \rightarrow B : \{ Nb \}_{Kb}$

Protokol obsahuje dvě role – iniciátora a příjemce. Vlákna těchto rolí vypadají následovně:

Init[A, B, Na, Nb]	Resp[A, B, Na, Nb]
$1 : < + \{ Na . A \}_{Kb} >$	$1 : < - \{ Na . A \}_{Kb} >$
⇓	⇓
$2 : < - \{ Na . Nb . B \}_{Ka} >$	$2 : < + \{ Na . Nb . B \}_{Ka} >$
⇓	⇓
$3 : < + \{ Nb \}_{Kb} >$	$3 : < - \{ Nb \}_{Kb} >$
role iniciátora	role příjemce

Komunikující subjekt je instancí jedné z rolí. Seznamy parametrů obsahují označení účastníků *A* a *B*, nonces *Na* a *Nb*. *Na* jedinečně vzniká v prvním uzlu vlákna iniciátora *Init*[*A*, *B*, *Na*, *Nb*], *Nb* jedinečně vzniká v druhém uzlu vlákna příjemce *Resp*[*A*, *B*, *Na*, *Nb*].

5.2.3 Model útočníka

Útočník *P* má počáteční znalosti *init-info*(*P*). Obvykle sem patří označení účastníků komunikace a klíče, které na počátku komunikace útočník zná (*Kp*). Mohou to být veřejné klíče, soukromé klíče útočníka a symetrické klíče *Kpx* (*Kxp*) sdílené mezi útočníkem a některým z legitimních účastníků komunikace.

Útočník může odposlouchávat zprávy a z těchto odposlechnutých zpráv a počátečních znalostí vytvářet zprávy nové. Tyto akce jsou modelovány množinou vláken útočníka.

Vlákno útočníka může být jedno z následujících (g a h jsou termy):

- $M[t]$ – atomická zpráva (atomic message): $< +t >$, kde $t \in$ *init-info*(P) a $t \in T$.
- $F[g]$ – zachycení zprávy (flushing): $< -g >$.
- $T[g]$ – opětovné odeslání zprávy (tee): $< -g, +g, +g >$.
- $C[g, h]$ – spojení (concatenation): $< -g, -g, +gh >$.
- $R[g, h]$ – rozdělení do komponent (separation): $< -gh, +g, +h >$.
- $K[k]$ – klíč (key): $< +k >$, kde $k \in Kp$.
- $E[k, h]$ – šifrování (encryption): $< -k, -h, +\{ h \}_k >$.
- $D[k, h]$ – dešifrování (decryption): $< -k^{-1}, -\{ h \}_k, +h >$.

Zápis vlákna $< -t_1, \ldots, -t_k , +t_1', \ldots, +t_k' >$ znamená, že termy $t_1, \ldots, t_k$ mohou být obdrženy v libovolném pořadí a termy $t_1', \ldots, t_k'$ mohou být odeslány v libovolném pořadí. Všechny termy t_i však musí být obdrženy před odesláním nějakého termu t_j'.

Výše uvedenou množinu je možné pro konkrétní případy rozšířit.

5.2.4 Logika

Syntaxe

Základními prvky logiky jsou uzlové konstanty – node constants ($n, n_1 , \ldots$), vláknové konstanty – strand constants ($s, s_1 , \ldots$), balíkové konstanty – bundle constants ($c, c_1 , \ldots$) a balíkové proměnné – bundle variables ($C, C_1 , \ldots$).

Výrokové formule jsou definovány následovně:

- $n \in s, n \in c, n \in C, s \in c, s \in C$ jsou (atomické) výrokové formule;
- $\neg f_1$ a $f_1 \wedge f_2$ jsou výrokové formule, jestliže f_1 a f_2 jsou výrokové formule;

WFF formule (Well-formed formula):

- f, $\neg$ F$_1$, F$_1 \wedge$ F$_2$;
- $\forall$ C.f, kde f je výroková formule, jež neobsahuje žádnou jinou volnou proměnnou než C;

F_1 a F_2 jsou WFF formule, f je výroková formule a C je balíková proměnná.

Dále platí:

- $f_1 \vee f_2 \equiv \neg(\neg f_1 \wedge f_2)$;
- $f_1 \Rightarrow f_2 \equiv \neg f_1 \vee f_2$;
- $f_1 \Leftrightarrow f_2 \equiv f_1 \Rightarrow f_2 \wedge f_2 \Rightarrow f_1$;
- $\exists C.f \equiv \neg \forall C. \neg f$;

Sémantika

Nechť *N* je množina uzlů. Pro daný protokol *p*, množina regulérních vláken a vláken útočníků je *Sp*. Provedení protokolu tvoří množinu balíků *Dp*. Pro daný protokol *p* je dán model $M = (\{ N, Sp, Dp \}, I)$, kde *I* je tzv. interpretace. Sémantika logiky je definována následovně:

- $I(n)$ je uzel v *N*, $I(s)$ je vlákno v *Sp* a $I(c)$ je balík v *Dp*.
- $M \models n \in s$, když a jen když $I(n) \in I(s)$. Podobně to platí pro $M \models n \in c$, $M \models s \in c$.
- Jestliže *f* je výroková formule nebo WFF formule, pak $M \models \neg f$, když a jen když $M \not\models f$.
- Jestliže f_1 a f_2 jsou výrokové formule nebo WFF formule, pak $M \models f_1 \wedge f_2$, když a jen když $M \models f_1$ a $M \models f_2$.
- $M \models \forall C.f$, když a jen když $M \models [C/c_0]f$ pro jakýkoliv balík c_0.

Specifikace bezpečnostních požadavků

Pomocí výše specifikované logiky lze vyjádřit řadu bezpečnostních požadavků. Těmi hlavními jsou autentizace a důvěrnost.

Autentizace

Existují dva typy autentizace – slabá a silná (kapitola 2.5.2). Slabou autentizaci lze v logice vyjádřit takto:

$$\forall C.\mathrm{Resp}(\vec{x}) \in C \Rightarrow \mathrm{Init}(\vec{x}) \in C,$$

kde $Resp(\vec{x})$ a $Init(\vec{x})$ jsou vlákna příjemce a odesílatele s parametry $\vec{x}$.

Konkrétně pro protokol Needham-Schroeder-Lowe (kapitola 5.2.2) vypadá specifikace požadavku na autentizaci následovně:

$$\forall C.\mathrm{Resp}[A, B, Na, Nb] \in C \Rightarrow \mathrm{Init}[A, B, Na, Nb] \in C.$$

Silnou autentizaci lze potom obvykle dokázat na základě argumentu, že neexistují dvě vlákna $Init(\vec{x}) \in C$, jelikož nonces v $Init(\vec{x})$ jedinečně vznikají pouze v jednom vlákně. Konkrétně pro protokol Needham-Schroeder-Lowe platí, že nonce *Na* jedinečně vzniká ve vlákně *Init*[*A*, *B*, *Na*, *Nb*].

Důvěrnost

Hodnota *v* je důvěrná v prostoru vláken *S*, jestliže pro každý balík *C*, který obsahuje *S*, platí, že neexistuje uzel $n \in C$ takový, že $term(n) = v$. Například pokud *S* je vlákno příjemce, může být požadavek na důvěrnost specifikován takto:

$\neg\exists C.(\ Resp(\overset{t}{x}) \in C \wedge node(+v) \in C\ )$.

5.2.5 Analýza protokolu

Analýza protokolu je založena na kontrole modelem – zjišťuje se, zda pro daný model protokolu *P* platí formule *F*. Oproti klasickým technikám, patřícím do této kategorie, se však algoritmus značně liší. Pro další výklad je nutné vymezit ještě několik pojmů.

Cíle a závazné cíle

Cíl (goal) je dvojice (t, n), kde $sign(n) = -$, $t \in term(n)$ a *t* není konkatenací jiných dvou termů. Jinými slovy, cíl vyjadřuje původ termu. Množina cílů (goal-set) balíku *C* je množina všech cílů v *C* a značí se $G(C)$.

Cíl je vázán k uzlu *n′*, jestliže *n′* je $\preceq_C$ – minimálním členem množiny

$\psi = \{ m \in C \mid t \Subset term(m) \text{ a } sign(m) = + \text{ a } m \preceq n \}$.

Dvojice $((t, n), n')$ je závazný cíl pro (t, n) a značí se $n' \overset{t}{\twoheadrightarrow} n$. Závazný cíl reprezentuje informaci o uzlu, který jako první odeslal term *t*. Vzhledem k tomu, že útočník může zprávy odposlouchávat a přeposílat, není již důležité, kdo zprávu poslal, ale kdo ji poslal jako první. Závazný cíl tedy nahrazuje relaci $\rightarrow$.

Stavové struktury

Polo-balík (semi-bundle) $H = (N_H, E_{\Rightarrow})$ je podgrafem *N*, kde $E_{\Rightarrow} \subseteq \Rightarrow$ je množina hran, N_H je množina uzlů spojených hranami v $E_{\Rightarrow}$. Platí že:

- *H* je neprázdný, konečný a acyklický.
- Jestliže $n_1 \in N_H$ a $n_2 \Rightarrow n_1$, pak $n_2 \Rightarrow n_1 \in E_{\Rightarrow}$.

Polo-balík je tedy definován stejně jako balík (bundle) s tím rozdílem, že není nutně uzavřený vůči relaci $\Rightarrow$. Podmínka uzavřenosti vůči $\rightarrow$ zůstává.

Stav (state) je definován n-ticí < *S*, *G*, $\twoheadrightarrow$, $\precsim$ >, kde

- *S* je polo-balík obsahující vlákna účastníků komunikace,
- *G* je množina nezávazných cílů z *S*,
- $\twoheadrightarrow$ je relace pro závazné cíle,
- $\precsim = (\twoheadrightarrow \cup \Rightarrow)^*$ je reflexivní a tranzitivní uzávěr relací $\twoheadrightarrow$ a $\Rightarrow$.

Jestliže *G* ve stavu < *S*, *G*, $\twoheadrightarrow$, $\precsim$ > je prázdná množina, pak existuje balík *C* takový, že $S \in C$ a *S* a *C* obsahují stejná vlákna s výjimkou toho, že *C* může navíc obsahovat některá vlákna útočníka typu *R*, *T* nebo *S*.

Množina balíků (bundle set) stavu *l* jsou všechny balíky, které obsahují *l*: $\Psi(l) = \{c{:}bundle \mid l \in c\}$.

Změna stavu (next state transition) je definována jako funkce *F* : *stav* $\rightarrow$ *množina stavů*, která mapuje stav *l* na množinu jeho dalších stavů – $L' = F(l)$. *F* je úplně kompletní, pokud platí, že:

- L' je konečná množina,
- $\Psi(L') = \Psi(l)$, kde $\Psi(L') = \mathbf{U}_{l' \in L'} \Psi(l')$.

Jestliže $F(l)$ je prázdná, pak *l* je nedosažitelný stav (unreachable state). Znamená to, že neexistuje žádný balík, který obsahuje stav *l*.

Verifikační algoritmus

Další výklad je zaměřen na nejběžnější případ, kdy dokazované formule jsou ve tvaru:

$\mu_1 = \forall C.s_1 \in C \Rightarrow s_2 \in C$ a $\mu_2 = \exists C.s_1 \in C$,

kde s_1 je vláknová konstanta a s_2 je regulérní vláknová konstanta. Jedná se o specifikaci výše uvedených požadavků na autentizaci a důvěrnost. Další vlastnosti jsou z těchto základních lehce odvoditelné.

Prvním krokem algoritmu je výpočet počátečního stavu $l_0 = < S_0, G_0, \twoheadrightarrow_0, \precsim_0 >$, kde S_0 je polo-balík, relace $\twoheadrightarrow_0$ je prázdná a $\precsim_0$ je korespondující částečné uspořádání na S_0. Z definice $\Psi(l_0)$ lze odvodit:

$\forall C.s_1 \in C \Rightarrow s_2 \in C \Leftrightarrow \forall c \in \Psi(l_0).s_2 \in c$,

$\exists C.s_1 \in C \Leftrightarrow \exists c.c \in \Psi(l_0)$.

Takto odvozené formule je pak možné použít při redukci nekonečného stavového prostoru $\Psi(l_0)$ do konečného stavového prostoru *L* pomocí analýzy dostupnosti (reachability analysis) a funkce změny stavu (next state transition) *F*. Následující pseudo kód naznačuje proceduru pro formuli vyjadřující vlastnost autentizace, kdy se ověřuje, zda platí podmínky $\forall l \in L.s_2 \in l$ a $\Psi(l_0) = \Psi(L)$:

```
proc searchI(InitialState) {
   L = {InitialState}
   while(¬empty(L)) do {
      l = choose(L);
      L = L \ l;
      L′ = F(l);
      if(¬empty(L′)) then {
         for each l′ = <S′, G′, →′,𝒮′> ∈ L′ {
            if(s2 ∉ l′) then {
               if(empty(G′)) then return false;
               else L = add(l′, L);
            }
         }
      }
   }
   return true;
}
```

Funkce *choose*(*L*) vrací stav z množiny *L*. Jako první se vybírá vždy stav, který obsahuje nejméně vláken. Procedura pro ověření požadavku na důvěrnost vypadá obdobně.

Pokud je procedura úspěšně ukončena (vrací *true*), je vytvořen stavový strom, jehož neprázdné listy představují stavy z množiny *L*, ověřovaná formule platí a korektnost protokolu je tedy dokázána. Pokud procedura vrací *false*, došlo k nalezení stavu, ve kterém formule neplatí a je vygenerován popis úspěšného útoku.

5.2.6 Shrnutí

Výhody

- Verifikace pomocí nástroje je rychlá a efektivní.
- Athena využívá efektivních technik k zamezení tzv. exploze stavového prostoru.
- Logika pro specifikaci bezpečnostních požadavků je dostatečně jednoduchá.
- Lze specifikovat například i požadavky vztahující se k elektronickému bankovnictví.
- V případě nalezení chyby v protokolu je vygenerován popis úspěšného útoku.

Nevýhody

- Ukončení procesu analýzy není zaručeno.

5.3 AVISPA

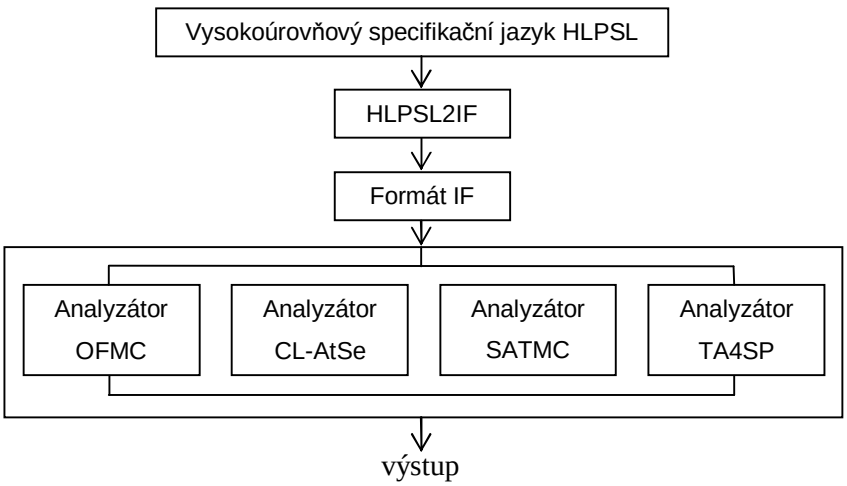
V roce 2003 byl evropskou komisí v rámci programu Information Society Technologies Programme založen projekt AVISPA. Na jeho základě později vznikl stejnojmenný nástroj pro verifikaci bezpečnostních protokolů AVISPA (Automated Validation of Internet Security Protocols and Applications).

AVISPA je plně automatický robustní a výkonný nástroj. Používá formální jazyk HLPSL pro specifikaci protokolů a bezpečnostních požadavků. Sestává ze čtyř různých analyzátorů – OFMC, CL-AtSe, SATMC a TA4SP (viz. dále). Kromě toho že byly nástrojem nalezeny již publikované chyby v protokolech (rodina protokolů ISO-PK, IKEv2, SET, ASW, H.530), v některých případech došlo i k nalezení chyb nových. V rámci projektu byla zveřejněna knihovna protokolů, které byly nástrojem verifikovány, včetně jejich specifikací v HLPSL a nalezených útoků. V případě nalezení chyby v protokolu umí nástroj vygenerovat popis úspěšného útoku.

AVISPA je dostupná jako samostatný programový balík nebo přes webové grafické rozhraní, které podporuje editaci specifikací protokolů a umožňuje vybrat a nastavit nástroje, které mají být pro analýzu použity.

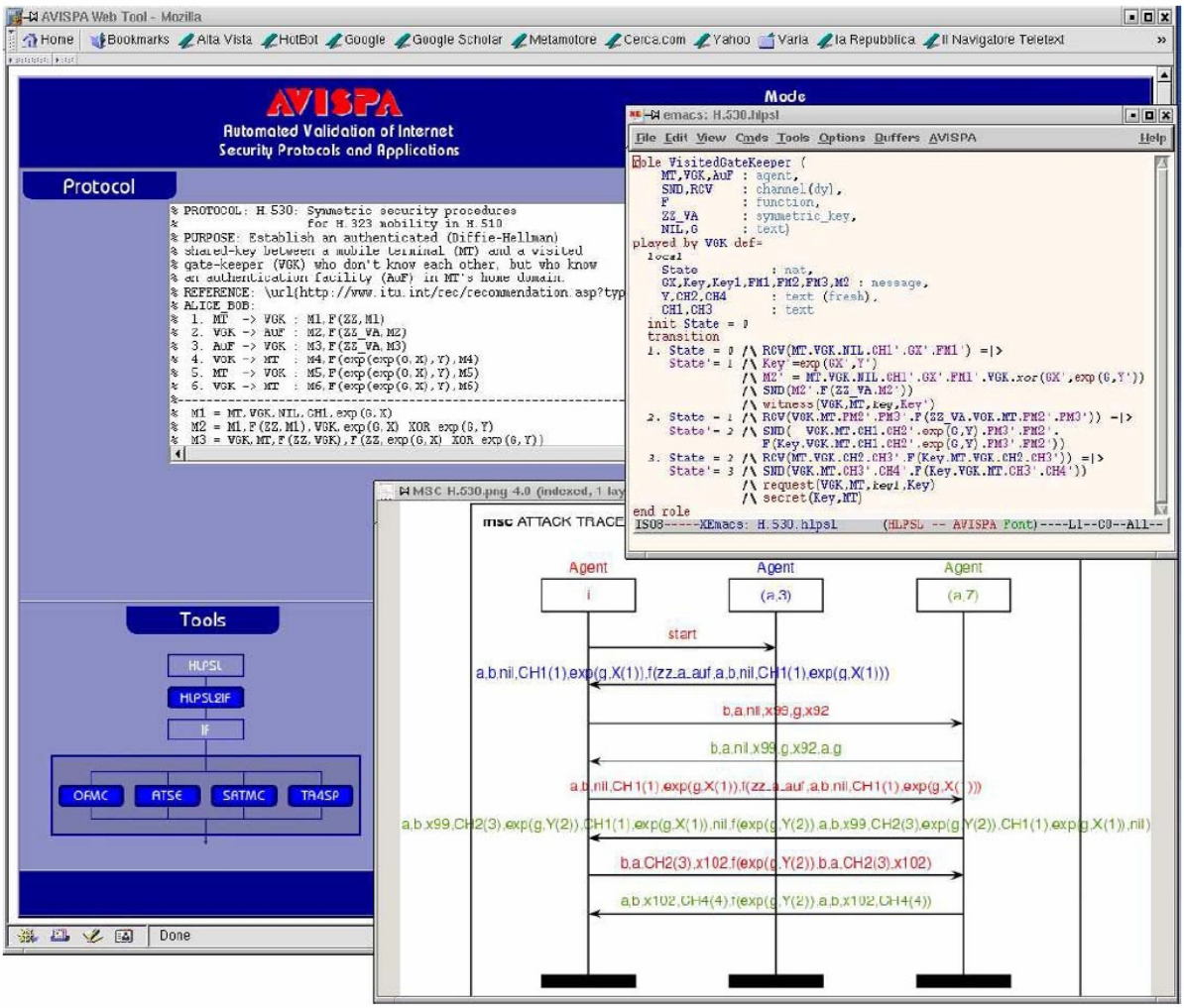
5.3.1 Architektura nástroje

Architektura nástroje je znázorněna na obrázku 9. Nejprve je v jazyce HLPSL specifikován protokol spolu s bezpečnostní vlastností, která se má ověřit. AVISPA poté automaticky (pomocí nástroje HLPSL2IF Translator) přeloží specifikaci do formálního jazyka IF (Intermediate Format). IF specifikace popisuje protokol jako nekonečný stavový systém. Tento popis je pak vstupem pro výše zmíněné analyzátory. Všechny předpokládají perfektní kryptografii (zprávu nelze šifrovat a dešifrovat bez znalosti klíče) a útočníka typu Dolev-Yao [33]. Ten může odposlouchávat zprávy, šifrovat a dešifrovat zprávy jemu známými klíči a iniciovat komunikaci s legitimními účastníky.



Obr. 9: Architektura nástroje AVISPA

Každý z analyzátorů produkuje vlastní výstup s výsledky analýzy. Nalezené útoky jsou popsány v ASCII formátu a znázorněny graficky pomocí Message Sequence Charts. Grafický výstup může být rovněž exportován do souboru ve formátu PostScript. Uživatelské rozhraní poskytuje dva módy – základní a pokročilý. Ukázka uživatelského rozhraní v pokročilém módu je na obrázku 10.



Obr. 10 – Uživatelské rozhraní nástroje AVISPA (pokročilý mód) [47]

5.3.2 HLPSL

V této kapitole jsou popsány pouze základy syntaxe jazyka HLPSL. Bližší podrobnosti o syntaxi a sémantice lze nalézt v [48].

HLPSL (High-Level Protocol Specification Language) je vysokoúrovňový jazyk pro specifikaci protokolů. Je založen na kompozici rolí reprezentujících jednotlivé účastníky komunikace. Každá role je nezávislá a je iniciována parametry. Pro komunikace mezi rolemi se používají kanály.

Syntaxe

Proměnné v HLPSL musí začínat velkým písmenem, konstanty malým písmenem. Komentáře (%[^\n]*) a oddělovače ([\n\r\t]) jsou ignorovány.

Klíčová slova:
accept, agent, authentication_on, bool, channel, composition, cons, const, def=, delete, dy, end, exp, goal, hash, hash_func, iknows, in, init, intruder_knowledge, inv, local, message, nat, new, not, ota, played_by, protocol_id, public_key, request, role, secrecy_of, secret, set, start, symmetric_key, text, transition, weak_authentication_on, witness, wrequest, xor.

Specifikace protokolu se skládá ze čtyř částí – seznam definic rolí, seznam deklarací cílů (nepovinná část) a hlavní role typicky s názvem *environment* (obvykle bez parametrů).

Role jsou nezávislé procesy. Označují se jménem, je možné u nich nastavit parametry a deklarovat lokální proměnné. Mohou být buď základní (basic roles) nebo složené (composition roles).

Ze základních rolí může účastník komunikace vytvářet instance. Parametry a lokální proměnné reprezentují znalosti účastníka. Jeho jméno je předáváno také jako parametr. Základní role obsahují přechody reprezentující změnu stavu na základě nějaké události nebo skutečnosti.

Složené role specifikují konkrétní relaci protokolu. Jsou paralelní či sekvenční kompozicí základních rolí. Na rozdíl od nich neobsahují deklaraci přechodů.

Jméno role a parametry tvoří hlavičku role. Dále mohou být deklarovány lokální proměnné, konstanty, znalosti útočníka a podmínky, za kterých je role považována za ukončenou. Proměnné mohou být rovněž inicializovány.

Deklarace lokálních proměnných jsou uspořádány do skupin podle typu a oddělené čárkou. I když jsou proměnné lokální pro danou roli, stejné proměnné deklarované v jiné roli musí být stejného typu. V deklaraci je možné uvést vlastnictví proměnné.

Konstanty jsou deklarovány uvnitř rolí, ovšem mají globální působnost. Obvykle se deklarují v hlavní roli.

Typy proměnných mohou být buď jednoduché nebo složené. Konstanty mohou být pouze jednoduchého typu.

Znalosti útočníka jsou definovány množinou konstant, parametrů a inicializovaných proměnných. Obvykle se definují pouze v hlavní roli.

Přechody deklarované v základních rolích jsou označeny názvy. Po splnění podmínek definovaných na levé straně pravidla (predikátu) dojde k provedení akce či reakce na pravé straně. Akce může být například přiřazení hodnoty proměnné, odeslání zprávy nebo volání předdefinovaného predikátu cíle.

Lze použít následující předdefinované predikáty cíle:

- *secret*(*E, id, S*) – informace *E* je označena jako tajemství sdílené účastníky z množiny *S*; *id* je identifikátor cíle;
- *witness*(*A, B, id, E*) – používá se pro ověření autentizace; účastník *A* má informaci *E* pro *B*; *id* je identifikátor cíle;
- *request*(*B, A, id, E*) – používá se pro silnou autentizaci; účastník *B* očekává informaci *E* od účastníka *A*; *id* je identifikátor cíle;
- *wrequest*(*B, A, id, E*) – jako request, ale používá se pro slabou autentizaci.

Součástí složené role mohou být základní i jiné složené role. Jejich seznam se uvádí v těle složené role v sekci kompozicí.

Vytvoření instance nějaké role je podobné jako volání procedury s konkrétními parametry. Počet a typ parametrů musí odpovídat deklaraci.

Cíle jsou deklarovány pomocí maker nebo LTL formulí [48]. K dispozici jsou následující předdefinovaná makra:

- *secrecy_of* – takto označená informace má zůstat utajena,
- *authentication_on* – používá se pro ověření silné autentizace,
- *weak_authentication_on* – pro ověření slabé autentizace.

Každý cíl je identifikován konstantou odkazující na předdefinovaný predikát. Pokud dojde k porušení cílů, znamená to, že byl proveden úspěšný útok.

V poslední části specifikace je definována hlavní role *environment*, která typicky obsahuje globální konstanty a kompozici jedné nebo více relací, kde útočník může být instancí některé z legitimních rolí. Obvykle jsou zde také definovány počáteční znalosti útočníka (jména účastníků komunikace, všechny veřejné klíče, soukromý klíč útočníka a klíče sdílené útočníkem a ostatními účastníky).

Příklad specifikace protokolu EKE v HLPSL [25] je uveden v příloze 2.

5.3.3 Modul HLPSL2IF

Modul HLPSL2IF slouží pro převod specifikace protokolu ve vysokoúrovňovém jazyce HLPSL do formálního jazyka IF (Intermediate Format). IF specifikace je založena na přepisovacích pravidlech popisujících nekonečný stavový systém s počátečním stavem, přechodovými pravidly a bezpečnostními požadavky definovanými jako stavy. Cílový predikát (viz. výše) definuje, zda je daný stav stavem útoku či nikoliv. Útok představuje cestu, která vede z počátečního stavu do stavu útoku. Specifikace může být generována jako typovaná nebo netypovaná.

IF je formální jazyk nízké úrovně. Na rozdíl od HLPSL je pro člověka prakticky nečitelný. IF specifikace je vhodná jako vstup pro analyzátory, které jsou součástí nástroje AVISPA. Použití tohoto přístupu umožňuje nezávislý vývoj vysokoúrovňového jazyka a jednotlivých analyzátorů. Poskytuje také rozhraní pro případná budoucí napojení jiných nástrojů.

5.3.4 Integrované analyzátory

Již bylo zmíněno, že nástroj AVISPA používá pro verifikaci protokolů čtyři různé analyzátory – OFMC, CL-AtSe, SATMC a TA4SP. Následující výklad obsahuje jejich stručný popis s odkazy na relevantní literaturu s bližšími informacemi.

OFMC (On-the-fly Model-Checker) [49]

OFMC je analyzátor založený na kontrole modelem. Jako vstup uvažuje jak typované, tak netypované modely specifikované ve formátu IF. Je to efektivní nástroj, který v sobě integruje techniky založené na omezeních (constraint-based techniques). Analýza protokolu probíhá nad konečným počtem relací.

Nástroj je schopen modelovat nekonečný stavový prostor použitím tzv. lazy datových typů. Jde o datové typy, u kterých se při sestavování nevyhodnocují jejich argumenty.

Pro modelování Dolev-Yao útočníka se používají symbolické techniky a optimalizace (lazy intruder) pro redukci procházeného stavového prostoru – zprávy útočníka jsou reprezentovány termy s proměnnými a jsou definována omezení týkající se termů, které mají být generovány a termů, které k tomu mohou být použity. Objem všech možných zpráv útočníka je tak značně redukován, přičemž není vynechán žádný potenciální útok. Další z použitých technik je redukce stavového prostoru na základě částečného uspořádání.

OFMC dále implementuje řadu heuristik pro prohledávání. Poskytuje také možnost modelovat útočníka, který je schopen odhadovat slabá hesla. Rovněž je možné specifikovat algebraické vlastnosti kryptografických operátorů.

CL-AtSe (Constraint-Logic-based Attack Searcher) [50]

CL-AtSe je stejně jako OFMC založen na kontrole modelem a řešení omezení. Zprávy protokolu mohou být typované nebo netypované. Vstupem je IF specifikace protokolu. Analýza je prováděna nad konečným počtem relací.

Nástroj aplikuje různé optimalizační techniky pro urychlení procesu verifikace. Lze je rozdělit do dvou hlavních kategorií. Techniky první kategorie slouží pro zjednodušení specifikace protokolu spojením některých kroků nebo jejich vynecháním, ignorováním nepodstatných zpráv apod.

Do druhé kategorie patří různé optimalizace v odvozovacích pravidlech pro redukci, případně odstranění nadbytečných a neužitečných dat v provádění protokolu. Sem patří například technika lazy intruder zmíněná u popisu nástroje OFMC.

CL-AtSe umožňuje specifikovat algebraické vlastnosti operátoru XOR a umocňování. Jelikož nástroj je modulární, je rovněž možné použít i kryptografické operátory.

SATMC (SAT-based Model-Checker) [51]

Vstupem nástroje je IF specifikace typovaného protokolu. Počet analyzovaných relací je shora omezen.

SATMC používá sofistikovaných technik pro automatické generování výrokových formulí reprezentujících bezpečnostní problém. Formule jsou následně ověřovány pomocí SAT-solverů [52]. Každý nalezený model je reprezentován jako částečně uspořádaná množina přechodů s maximální hloubkou zanoření k, kde k je celočíselná kladná konstanta definovaná na vstupu. Pokud je možné provést linearizaci této částečné uspořádané množiny, znamená to, že byl nalezen útok.

Mezi nástrojem SATMC a SAT-solverem je v podstatě standardní rozhraní ve formátu DIMACS, což umožňuje využití nových SAT-solverů bez nutnosti dalších úprav.

TA4SP (Tree Automata based on Automatic Approximations for the Analysis of Security Protocols) [53]

Počet relací protokolu u nástroje TA4SP není omezen. Verifikace je založena na aproximaci znalostí útočníka pomocí regulárních stromových jazyků.

Použitá metoda je rozšířením aproximační metody pro verifikaci protokolů založené na stromových automatech, která byla představena Genetem a Klayem [54]. Dřívější přístupy tohoto typu vyžadovaly expertní znalosti pro transformaci specifikace protokolu do systému termů a výpočet aproximační funkce. TA4SP tyto postupy zcela automatizuje.

Nástroj dokáže na základě typované specifikace ověřit, zda protokol obsahuje chyby (pomocí pod-aproximace) nebo je bezpečný pro libovolné množství relací (pomocí nad-aproximace).

5.3.5 Shrnutí

Výhody

- Nástroj efektivně kombinuje čtyři různé analyzátory a dokáže tak objevit většinu chyb.
- Verifikace je rychlá, převod specifikace protokolu z HLPSL do IF formátu je proveden řádově v milisekundách.
- V případě nalezení chyby v protokolu umí nástroj vygenerovat popis úspěšného útoku.
- Nalezené útoky jsou graficky znázorněny.
- K dispozici je knihovna protokolů, které byly nástrojem verifikovány, včetně jejich specifikací v HLPSL a nalezených útoků.
- AVISPA je dostupná jako samostatná aplikace nebo vzdáleně přes webové rozhraní.

Nevýhody

- Používání nástroje vyžaduje detailní znalost analyzovaných protokolů.
- Specifikace protokolu v HLPSL je poměrně obtížná. Chyby ve specifikaci mohou způsobit chyby ve výsledcích.
- Analýza výsledků někdy vyžaduje více času.
- Ovládání nástroje je poměrně náročné.

5.4 STA

STA (Symbolic Trace Analyzer) je automatický nástroj pro verifikaci bezpečnostních protokolů. Je založen na kontrole modelem a dokazování teorému. Používá symbolické techniky, jež zamezují explicitní konstrukci celého (pravděpodobně nekonečného) stavového prostoru. Oproti technikám modelujícím konečný (omezený) stavový prostor tento přístup umožňuje přesnější modelování protokolů a větší efektivnost (nižší nároky na paměť, rychlejší verifikace).

Jednotlivé subjekty protokolu jsou modelovány jako souběžné procesy, které mezi sebou komunikují přes nezabezpečenou síť. Podobně jako ostatní nástroje pro analýzu protokolů, STA předpokládá perfektní kryptografii (zprávu nelze šifrovat a dešifrovat bez znalosti klíče) a útočníka typu Dolev-Yao [33]. Útočník typu Dolev-Yao může odposlouchávat zprávy, šifrovat a dešifrovat zprávy jemu známými klíči a komunikovat s legitimními účastníky.

Nástroj na základě specifikovaného protokolu ověřuje platnost bezpečnostního požadavku. Porušení tohoto požadavku znamená nalezení chyby. Proces verifikace je v takovém případě zastaven a je vygenerován popis úspěšného útoku.

STA je implementován v jazyce Moscow ML (zjednodušená verze jazyka Standard ML) a je dostupný pro Windows i Linux. Byl použit k verifikaci několika bezpečnostních protokolů – Needham-Schroeder, Wide Mouthed Frog, Yahalom, Otway-Rees, … V tabulce 3 je pro srovnání uveden počet procházených stavů při verifikaci protokolu Needham-Schroeder nástrojem STA a Murphi.

Počet iniciátorů	Počet příjemců	Murphi	STA
1	1	1706	60
2	1	17277	411
2	2	514550	24655

Tab. 3: Srovnání nástrojů – počet procházeným stavů při verifikaci protokolu NS

5.4.1 Specifikační jazyk

Jazyk pro specifikaci protokolu a bezpečnostních vlastností v STA je odvozen od spi-kalkulu [1]. Hlavním rozdílem je absence komunikačních kanálů. Zde je uvažován pouze jeden kanál přístupný všem subjektům. V této kapitole je stručně popsána syntaxe a sémantika jazyka. Podrobnější popis lze nalézt v [55].

Syntaxe

a, b, …	názvy (labels) L
m, n, …	jména (names) N
a, b, …, h, k, …	lokální jména (local names) LN
$\underline{a}$, $\underline{b}$, …, $\underline{h}$, $\underline{k}$, …	jména prostředí (environmental names) εN
x, y, …	proměnné (variables) V
u, v, …	proměnné nebo jména $V \cup N$

M, N	::=	$u \mid \{ M \}_u \mid < M, N >$	zprávy (messages) M
n, ξ	::=	$u \mid \{ \xi \}_\eta \mid < \xi, \eta >$	termy (terms) Z

A, B	::=		subjekty (agents) A
		0	null
	|	$a (x).A$	vstup
	|	$\overline{a} (\xi).A$	výstup
	|	case ξ of $\{ y \}_\eta$ in A	dešifrování
	|	pair ξ of $< x, y >$ in A	výběr
	|	$[\xi = \eta]A$	porovnávání
	|	A || B	paralelní kompozice

L, N a V jsou spočetné disjunktní množiny. L je množina názvů (labels), N množina jmen (names). N je sjednocením dvou množin – LN (množina lokálních jmen) a εN (množina jmen prostředí). Tyto množiny reprezentují základní data (klíče, nonces, …). V je množina proměnných. Pomocí jmen a proměnných lze vytvářet složené zprávy v M (symetrické šifrování, párování).

Syntaxe výrazů subjektů A je v základě odvozena od spi-kalkulu. Například výraz *case* $\{ M \}_h$ *of* $\{ y \}_k$ *in* A znamená, že se subjekt pokusí dešifrovat zprávu M pomocí klíče k. V případě úspěchu (tedy pokud $k = h$) je dešifrovaná zpráva navázána na proměnnou y a subjekt dále pokračuje jako proces A. V případě neúspěchu výraz není vyhodnocen. Podobně *pair* M *of* $< x, y >$ *in*

A znamená, že se subjekt pokusí zprávu *M* rozdělit na dvě složky *x* a *y*.

Říká se, že výraz subjektu *A* je uzavřený, pokud neobsahuje žádné volné proměnné ($v(A) = \emptyset$, $v(A) \subseteq V$). Lokální jména a jména prostředí vyskytující se v *A* se označují jako $ln(A)$ a $en(A)$. Proces *P* je výchozím (initial) procesem pokud platí, že $en(P) = \emptyset$. Pro každé *M* a *u* výraz [*M*/*u*] značí substituci volných výskytů *u* za *M*.

Jak již bylo zmíněno, na rozdíl od spi-kalkulu je zde uvažován pouze jeden veřejný komunikační kanál. Názvy (labels) u vstupů a výstupů tedy neslouží k označení kanálu, nýbrž jsou to jakési značky pro ulehčení sledování referencí. Výrazy pro výstup a dešifrování mají smysl pouze v případě, že ξ je zpráva a η je jméno.

Operační sémantika

Akce (action) je term ve tvaru $a< M >$ (vstup) nebo $\bar{a} < M >$ (výstup), kde *a* je název a *M* zpráva. *Act* je označení pro množinu akcí, které se značí symboly α , β , … . Act^* je množina řetězců akcí, pro jejichž označení se používají písmena *s, s′*, … Spojení (konkatenace) řetězců se zapisuje pomocí tečky. $Act(s)$ značí množinu akcí v *s*, $msg(s)$ značí množinu zpráv v *s*. Jestliže řetězec *s* neobsahuje žádné volné proměnné ($v(s) = \emptyset$), říká se, že je uzavřený. Pokud platí, že $en(s) = \emptyset$, pak se *s* označuje jako výchozí řetězec.

Sekvence akcí (trace) je uzavřený řetězec $s \in Act^*$, takový, že pro každé s_1, s_2 a $a< M >$ platí, že pokud $s = s_1 \,.\, a< M > \,.\, s_2$, pak $s_1 \vdash M$. Tato sekvence je výsledkem interakce procesu a prostředí, ve kterém proces běží. Každá zpráva obdržená procesem je odvoditelná z předchozích znalostí prostředí.

Konfigurace systému je definována jako dvojice $< s, P >$, kde *P* značí proces a *s* reprezentuje aktuální znalosti prostředí, což je sekvence akcí (trace), které se dosud na síti vyskytly. Pokud platí, že $en(s, P) = \emptyset$, pak se jedná o konfiguraci počáteční. Konfigurace se značí písmeny *C, C′*, … Veškeré znalosti prostředí jsou i znalostmi útočníka. Další znalosti mohou být odvozeny pomocí následujících pravidel:

$$\text{(Ax)}\ \frac{M \in S}{S \vdash M} \qquad \text{(Env)}\ \frac{\underline{a} \in \varepsilon N}{S \vdash \underline{a}}$$

$$\text{(Proj1)}\ \frac{S \vdash < M, N >}{S \vdash M} \qquad \text{(Proj2)}\ \frac{S \vdash < M, N >}{S \vdash N} \qquad \text{(Pair)}\ \frac{S \vdash M \quad S \vdash N}{S \vdash < M, N >}$$

$$\text{(Dec)}\ \frac{S \vdash \{ M \}_u \quad S \vdash u}{S \vdash M} \qquad \text{(Enc)}\ \frac{S \vdash M \quad S \vdash u}{S \vdash \{ M \}_u}$$

Binární relace $S \vdash M$, kde S je podmnožina M ($S \subseteq M$) a M je zpráva, vyjadřuje skutečnost, že z S lze získat M. Je zřejmé, že výsledkem aplikace pravidel *Env*, *Pair* a *Enc* je nekonečná množina zpráv.

Relace přechodů $\rightarrow$ mezi konfiguracemi jsou definovány následujícími pravidly:

(Inp) $< s, a(x) . P > \rightarrow < s . a< M >, P[M/x] >$ $s \vdash M$, M je uzavřená zpráva

(Out) $< s, \overline{a} < M > . P > \rightarrow < s . \overline{a} < M >, P >$

(Case) $< s, \text{case } \{ \xi \}_k \text{ of } \{ y \}_k \text{ in } P > \rightarrow < s, P[\xi/y] >$

(Select) $< s, \text{pair} < \xi, \eta > \text{ of } < x, y > \text{ in } P > \rightarrow < s, P[\xi/x, \eta/y] >$

(Match) $< s, [\xi = \xi]P > \rightarrow < s, P >$

(Par) $$\frac{< s, P > \rightarrow < s', P' >}{< s, P \,||\, Q > \rightarrow < s', P' \,||\, Q >}$$

(+ symetrická verze pravidla Par)

Uvedená pravidla se vztahují pouze k symetrické kryptografii se sdíleným klíčem. Pravidla *Inp* a *Out* vyjadřují příjem a odeslání zprávy. *Case* je pravidlo pro dešifrování, *Select* slouží pro rozdělení zprávy. *Match* vyjadřuje porovnání dvou zpráv. Pravidlo *Par* reprezentuje paralelní zpracování dvou procesů.

Specifikace bezpečnostních vlastností

Konfigurace $< s, P >$ generuje sekvenci akcí (trace) s' ($< s, P >$] s'), jestliže $< s, P > \rightarrow^* < s', P' >$ pro nějaký proces P'. $\rightarrow^*$ je reflexivní a tranzitivní uzávěr $\rightarrow$ (představuje jedno nebo více provedení relace přechodu $\rightarrow$).

Bezpečnostní vlastnosti se specifikují v rámci těchto generovaných sekvencí akcí. Například pro splnění požadavku na autentizaci musí pro každý generovaný trace s platit, že pokud se v sekvenci objeví instance akce β (akce mohou obsahovat volné proměnné), musí jí v té samé sekvenci předcházet instance akce α. Tento požadavek se vyjadřuje zápisem $\alpha \hookleftarrow \beta$. Splnění požadavku vyjadřuje zápis $< s, P > \models \alpha \hookleftarrow \beta$.

Podobně lze vyjádřit požadavek na důvěrnost. K tomuto účelu je vhodné zavést označení akce $\perp$, které není použito v žádném z výrazů komunikujících subjektů. Zápis $\perp \hookleftarrow \alpha$ pak vyjadřuje, že se v sekvenci událostí nikdy nesmí objevit žádná instance akce α. Další možností je rozšířit protokol P o tzv. strážce (guardian), který sleduje, zda se na síti nevyskytne zpráva obsahující důvěrnou hodnotu – $< \varepsilon, P \,||\, g(x).0 > \models \perp \hookleftarrow g(d)$; g je strážce, d důvěrná hodnota a ε prázdný trace.

Symbolická sémantika

Jak již bylo zmíněno, výsledkem aplikace odvozovacích pravidel je nekonečná množina zpráv. Jako opatření proti explozi stavového prostoru STA nahrazuje nekonečně mnoho (konkrétních) relací přechodů $\rightarrow$ jedinou symbolickou relací $\rightarrow_S$. Obdržené zprávy jsou nyní reprezentovány jednoduše pomocí volných proměnných.

Sekvence akcí se symbolickými relacemi (symbolický trace) je řetězec $\sigma \in Act^*$, pro který platí:

- $en(\sigma) = \emptyset$;
- pro každé σ_1 a σ_2, α a x platí, že pokud $\sigma = \sigma_1 . \alpha . \sigma_2$ a $x \in v(\alpha) - v(\sigma_1)$, pak α je vstupní akcí.

Pro symbolickou konfiguraci $< \sigma, A >_S$, kde σ je symbolický trace a A komunikující subjekt, platí:

- $en(A) = \emptyset$;
- $v(A) \subseteq v(\sigma)$

Relace přechodů $\rightarrow_S$ mezi symbolickými konfiguracemi jsou definovány následujícími pravidly:

(Inp_S) $< \sigma, a(x) . A >_S \rightarrow_S < \sigma . a< x >, A >$

(Out_S) $< \sigma, \overline{a} < M > . A >_S \rightarrow_S < \sigma . \overline{a} < M >, A >$

$(Case_S)$ $< \sigma, case \{ \xi \}_v \text{ of } \{ x \}_u \text{ in } A >_S \rightarrow_S < \sigma\theta, A\theta >_S$ $\quad \theta = mgu(\{ \xi \}_v, \{ x \}_u)$

$(Select_S)$ $< \sigma, pair \ \xi \text{ of } < x, y > \text{ in } A >_S \rightarrow_S < \sigma\theta, A\theta >_S$ $\quad \theta = mgu(\xi, < x, y >)$

$(Match_S)$ $< \sigma, [\xi = \eta]A >_S \rightarrow_S < \sigma\theta, A\theta >_S$ $\quad \theta = mgu(\xi, \eta)$

(Par_S) $\dfrac{< \sigma, A >_S \rightarrow_S < \sigma', A' >_S}{< \sigma, A \| B >_S \rightarrow_S < \sigma', A' \| B' >_S}$ $\quad B' = B\theta$, jestliže $\sigma' = \sigma\theta$; $B' = B$, jestliže $\sigma' = \sigma$

(+ symetrická verze pravidla Par_S)

Předpokládá se, že:

- $x = new_v(V)$, kde V je množina volných proměnných v konfiguraci,
- $y = new_v(V \cup \{ x \})$,
- $msg(\sigma)\theta \subseteq M$.

Zápis $t\theta$ vyjadřuje aplikaci substituce θ na nějaký objekt t (proměnná, zpráva, proces, trace, …). Pokud platí, že $t_1\theta = t_2\theta$, pak je substituce θ tzv. unifikátorem t_1 a t_2.

Výraz $mgu(t_1, t_2)$ označuje nejvíce obecný unifikátor t_1 a t_2. Jedná se o unifikátor θ, pro nějž platí, že jakýkoliv jiný unifikátor může být vyjádřen kompozicí substitucí $\theta\,\theta'$ pro nějaké θ'.

Funkce *new*(V) pro nějaké $V \in \mathcal{V}$, označuje proměnnou, která se v množině V nevyskytuje.

Symbolická konfigurace $< \sigma , A >_S$ symbolicky generuje σ' ($< \sigma , A >_S \;]_S \; \sigma'$), jestliže $< \sigma, A >_S \rightarrow_S^* < \sigma', A' >_S$ pro nějaký proces A'. Množina sekvencí akcí se symbolickými relacemi vytváří tzv. symbolický model protokolu. Na rozdíl od standardního modelu s konkrétními relacemi je symbolický model konečný.

Nechť σ je symbolický trace a ρ nějaká substituce. Jestliže $\sigma\rho$ je trace, říká se, že ρ splňuje σ nebo také že $\sigma\rho$ je řešením σ a σ je konzistentní.

5.4.2 Specifikace protokolu

Specifikace protokolu v STA odpovídá syntaxi a sémantice výše popsaného formálního modelu. Sestává z deklarací identifikátorů (názvy, jména a proměnné), procesů, konfigurací a bezpečnostních vlastností. Tyto deklarace mohou být uvedeny v libovolném pořadí. Identifikátory, které jsou použity v deklaraci procesu, musí být deklarovány dříve než tento proces.

Deklarace identifikátorů (názvy, jména a proměnné):

```
DeclLabel $ a1, a2, …, an $;
DeclName $ k1, k2, …, km $;
DeclVar $ x1, x2, …, xr $;
```

Deklarace procesů:

P	::=	stop	přerušený proces
	|	a?x >> P	vstup (*a* je název, *x* je proměnná)
	|	a!M >> P	výstup (*a* je název, *M* je zpráva)
	|	(M is N) >> P	test na shodu a symetrické dešifrování (*M* a *N* jsou zprávy)
	|	(M pkdecr (y, -u)) >> P	asymetrické dešifrování
	|	P1 || P2	paralelní kompozice
	|	P1 & P2	nedeterministický výběr
	|	k new_in P	nové jméno (*k* je jméno)

Symbol >> vyjadřuje sekvenci akcí. Například zápis (*M is N*) >> *P* znamená, že se porovnají zprávy *M* a *N* a v případě shody proces dále pokračuje jako proces *P*. V opačném případě je proces zastaven.

Deklarace zpráv:

M	::=	u	jméno nebo proměnná
	|	+u	veřejný šifrovací klíč
	|	-u	soukromý šifrovací klíč
	|	(M1, M2)	pár
	|	{ M }u	symetrické šifrování zprávy *M* klíčem *u*
	|	{ M }^+u	asymetrické šifrování zprávy *M* klíčem +*u*
	|	H(M)	hash zprávy *M*

Deklarace konfigurací:

val Conf = (L @ Pr);

L je seznam vstupních a výstupních akcí, které představují počáteční znalosti prostředí / útočníka. *Pr* je dříve deklarovaný proces.

Deklarace bezpečnostních vlastností:

val Prop = (A <- - B);

A a *B* jsou vstupní / výstupní akce. Mohou obsahovat proměnné, které by měly být různé od proměnných použitých v deklaraci procesů. Proměnné vyskytující se v akci *A* by se měly objevit i v akci *B*. Pro zápis požadavku na důvěrnost je k dispozici speciální akce *Absurd*, která nemůže být nikdy vykonána.

Příklad – protokol Needham-Schroeder Public Key

Formální specifikace protokolu:

1. A → B : $\{ Na, A \}_{+Kb}$
2. B → A : $\{ Na, Nb \}_{+Ka}$
3. A → B : $\{ Nb \}_{+Kb}$

STA specifikace protokolu:

```
(* Iniciátor a příjemce *)
val InA = a1!(Na, A)^+Kb >> a2?(Na, xNb)^+Ka >>
          a3!(xNb)^+Kb >> stop
        ||
          a'1!(N'a, A)^+Ki >> a'2?(N'a, xNi)^+Ka >>
          a'3!(xNi)^+Ki >> stop;
val Reb = b1?(yNa, A)^+Kb >> b2!(yNa, Nb)^+Ka >>
          b3?(Nb)^+Kb >> stop;
```

```
(* Počáteční konfigurace *)
val Ns = ( [disclose!(Ki, +Ka, +Kb, A, B, I)] @ (InA || ReB) );
(* Vlastnost 1 – autentizace A vůči B *)
val AuthAtoB = (a3!u <-- b3?u);
(* Vlastnost 2 – autentizace B vůči A *)
val AuthBtoA = (b2!u <-- a2?u);
```

5.4.3 Analýza protokolu

Každá konfigurace generuje pouze konečně mnoho symbolických sekvencí akcí. Analýza protokolu spočívá v řešení tzv. STAP problému pro každou z těchto sekvencí. STAP (Symbolic Trace Analysis Problem) je definován dvěmi akcemi α, β ($v(\alpha) \subseteq v(\beta)$) a symbolickou sekvencí σ, kdy se ověřuje, zda je pro každé řešení *s* sekvence σ splněna podmínka $\alpha \hookleftarrow \beta$. Splnění podmínky vyjadřuje zápis $\sigma \models \alpha \hookleftarrow \beta$.

Jelikož je třeba vzít v úvahu všechna řešení dané sekvence akcí, kterých může být i nekonečně mnoho, není řešení STAP problému zcela triviální záležitostí. Detailní informace lze nalézt v [55].

Příklad – výstup analýzy protokolu Needham-Schroeder Public Key

```
An attack was found:
   disclose!(Ki, +Ka, +Kb, A, B, I) . a‘1!(N’a, A)^+Ki .
   a1!(Na, A)^+Kb . b1?(N’a, A)^+Kb . b2!(N’a, Nb)^+Ka .
   a‘2?(N’a, Nb)^+Ka . a‘3!(Nb)^+Ki . b3?(Nb)^+Kb.
26 symbolic configurations reached.
```

Výstupem analýzy je informace o nalezení chyby v protokolu po ověření 26 symbolických konfigurací. Dále je uvedena sekvence akcí, které vedou k porušení specifikované bezpečnostní vlastnosti *AuthAtoB* (autentizace *A* vůči *B*).

5.4.4 Shrnutí

Výhody

- Verifikace pomocí STA je rychlá a efektivní.
- Nástroj využívá symbolických technik k zamezení exploze stavového prostoru.
- Specifikační jazyk je poměrně jednoduchý.
- V případě nalezení chyby v protokolu je vygenerován popis úspěšného útoku.

Nevýhody

- Nízkoúrovňové uživatelské rozhraní.

5.5 Srovnání nástrojů

Ohledně srovnání nástrojů pro verifikaci / analýzu protokolů dosud bohužel mnoho informací publikováno nebylo. Obvykle jsou sice k dispozici experimentální výsledky daného nástroje, ovšem množina testovaných protokolů se u jednotlivých nástrojů značně liší. Přímé srovnání výsledků tedy není možné.

Stejně tak je tomu v případě výše popsaných verifikačních nástrojů. V praktické části této práce byly protokoly verifikovány pouze nástrojem LySatool. Většina protokolů pochází z databáze protokolů SPORE (Security Protocols Open Repository) [9]. Výsledky analýzy těchto protokolů zbývajícími nástroji ovšem k dispozici nejsou. Z těchto důvodů je následující srovnání omezeno pouze na vlastnosti jednotlivých nástrojů.

Hlavní výhody a nevýhody jsou u každého nástroje vždy uvedeny v závěrečném shrnutí. Výhodou všech nástrojů je plně automatická verifikace. Specifikace protokolu v nástrojích LySatool, Athena a STA je dostatečně jednoduchá. Pro ulehčení specifikace v LySa kalkulu dokonce existují podpůrné aplikace jako Elyjah a For-LySa. Specifikace protokolu v HLPSL pro nástroj AVISPA je poměrně obtížná. Je nutná detailní znalost analyzovaného protokolu.

V LySatool standardně není možné specifikovat protokoly, které jsou založeny na použití algebraických operátorů jako XOR či umocňování. K tomuto účelu však lze LySa kalkul i nástroj LySatool vhodně rozšířit. Algebraické operátory rovněž nelze použít ve specifikaci STA. Ostatní nástroje použití algebraických operátorů umožňují.

Ve standardní verzi LySatool a v nástroji STA rovněž nelze specifikovat jiné bezpečnostní vlastnosti než autentizaci a důvěrnost. LySatool je však opět možné poměrně jednoduše rozšířit. Athena a AVISPA umožňují ověřit i jiné bezpečnostní vlastnosti. Například Athena dokáže ověřit požadavky týkající se protokolů v elektronickém bankovnictví.

Athena, AVISPA i STA dokáží v případě nalezení chyby v protokolu vygenerovat popis úspěšného útoku. LySatool toto neumí. Výsledkem analýzy je pouze komponenta chyb a prostředí proměnných. Tyto informace je nutné projít a případný útok sestavit manuálně. Zároveň je nutné vyloučit případné chyby vzniklé vlivem použité aproximace, které ve skutečnosti nejsou chybami v protokolu.

Athena, LySatool i STA poskytují standardně pouze textové uživatelské rozhraní. AVISPA disponuje pokročilejším grafickým rozhraním. Případné nalezené útoky jsou v nástroji graficky znázorněny. Kromě toho, že je AVISPA dostupná jako samostatná aplikace, je možné využít i vzdálený přístup přes webové rozhraní.

Výhodou nástroje AVISPA je také fakt, že kombinuje čtyři různé analyzátory s odlišnými přístupy a teoreticky tak dokáže objevit více chyb.

6 Praktická část

6.1 Bezpečnostní protokoly

Tato kapitola obsahuje seznam bezpečnostních protokolů, které byly v rámci práce verifikovány nástrojem LySatool. Protokoly jsou řazeny abecedně podle názvu původní verze protokolu. U každého protokolu je stručně popsána jeho funkce, formální specifikace, výsledky analýzy a jejich interpretace – nalezené útoky. Rovněž jsou uvedeny publikované útoky, které nástroj nenalezl.

Uvedené formální specifikace se mohou od publikovaných specifikací lišit – kvůli analýze musí být některé zprávy jinak uspořádány. V případě protokolů pro distribuci relačního klíče je do specifikace zároveň přidán další krok vyjadřující vlastní komunikaci mezi subjekty (viz. kapitola 5.1.8).

Obsah komponenty chyb ve výsledcích analýzy je zapsán pomocí indexů (*i*, *j*, *r*, *s*), přičemž každý z nich nabývá hodnot od 1 do 3. Pokud je někde uvedeno, že porušení nepředstavuje proveditelný útok bez vysvětlení, obvykle jde pouze o důsledek použité techniky aproximace. Z množiny potenciálních znalostí útočníka jsou uváděny pouze relevantní informace.

Za útok není považována situace, kdy útočník vystupuje jako legitimní účastník, naváže komunikaci se dvěma jinými účastníky a v obou relacích se mu podaří ustavit stejný relační klíč. Také nejsou uvažovány útoky založené na kompromitovaném klíči starší relace a útoky založené na předpokladu, že operace šifrování je komutativní. Tyto útoky jsou u protokolů pouze zmíněny (pokud byly publikovány).

LySa specifikace analyzovaných protokolů jsou k dispozici na přiloženém CD.

6.1.1 BAN concrete Andrew Secure RPC

Funkce protokolu

Distribuce čerstvého relačního klíče s použitím symetrické kryptografie.

Formální specifikace protokolu

1. $A \rightarrow B : A, Na$
2. $B \rightarrow A : \{ Na, Kab2 \}_{Kab}$
3. $A \rightarrow B : \{ Na \}_{Kab2}$
4. $B \rightarrow A : Nb$
5. $B \rightarrow A : \{ MSG \}_{Kab2}$

Výsledky analýzy

ψ ={ ($a2_{ij}$, $a3_{ij}$), ($b3_{ij}$, $b2_{ij}$), ($b3_{0j}$, $b2_{0j}$), ($a2_{i0}$, $a3_{r0}$), ($a2_{i0}$, $b2_{0s}$), ($b3_{0j}$, $a3_{r0}$) }

$LMSG_{ij}$, … $\in \rho(x_{\bullet})$

$\rho(xMSG_{ij}) = LNa_{ij}$, $LMSG_{ij}$

Nalezené útoky

Porušení ($a2_{ij}$, $a3_{ij}$) znamená, že útočník může v kroku 5 přehrát zprávu z kroku 3 ve stejné relaci. Tím dojde k dešifrování zprávy { *Na* }$_{Kab2}$ jako *MSG*. Nonce *Na* je veřejná hodnota (přenáší se v kroku 1 nezašifrovaně) a je tedy známa i útočníkovi. Z toho důvodu množina potenciálních znalostí útočníka obsahuje $LMSG_{ij}$. Nejde však o prozrazení důvěrné informace a porušení nepředstavuje proveditelný útok.

Žádné z porušení ($b3_{ij}$, $b2_{ij}$) a ($b3_{0j}$, $b2_{0j}$) nepředstavuje proveditelný útok.

Porušení ($a2_{i0}$, $a3_{r0}$), ($a2_{i0}$, $b2_{0s}$) a ($b3_{0j}$, $a3_{r0}$) také nepředstavují útok. Znamenají pouze, že pokud útočník vystupuje jako legitimní účastník komunikace a naváže dvě relace se dvěma jinými subjekty, mohou být zprávy zašifrované v jedné relaci dešifrovány ve druhé relaci. Tzn., že v obou relacích je použit stejný relační klíč.

Nenalezené útoky

Nástroj nenalezl útok publikovaný v [58]:

1. A → I_B : A, Na
1'. I_B → A : B, Na
2'. A → I_B : { Na, Kab2 }$_{Kab}$
2. I_B → A : { Na, Kab2 }$_{Kab}$
3. A → I_B : { Na }$_{Kab2}$
3'. I_B → A : { Na }$_{Kab2}$
4. I_B → A : Ni
4'. A → I_B : Na2
5. I_B → A : { MSG }$_{Kab2}$
5'. A → I_B : { MSG }$_{Kab2}$

Subjekt *A* naváže komunikaci s útočníkem, který se vydává za subjekt *B*. Ten následně iniciuje komunikaci s *A*. Útočník v kroku 2 v první relaci přehraje zprávu zašifrovanou v kroku 2' v druhé relaci. V obou relacích je takto použit stejný relační klíč *Kab*2. Po provedení útoku subjekt *A* věří, že navázal relaci s *B* a že subjekt *B* s ním navázal druhou relaci.

6.1.2 Lowe modified BAN concrete Andrew Secure RPC

Funkce protokolu

Distribuce čerstvého relačního klíče s použitím symetrické kryptografie. Opravená verze protokolu BAN concrete Andrew Secure RPC.

Formální specifikace protokolu

1. A → B : A, Na
2. B → A : { Na, B, Kab2 }$_{Kab}$
3. A → B : { Na }$_{Kab2}$
4. B → A : Nb
5. B → A : { MSG }$_{Kab2}$

Pro zamezení útoku proveditelného na původní verzi protokolu je do zprávy 2 přidána identita příjemce *B*.

Výsledky analýzy

Stejné jako u původní verze protokolu (BAN concrete Andrew Secure RPC).

Nalezené útoky

Nebyly nalezeny žádné útoky (viz. BAN concrete Andrew Secure RPC).

Nenalezené útoky

Nebyly publikovány žádné útoky.

6.1.3 Denning-Sacco shared key

Funkce protokolu

Vzájemná autentizace a distribuce relačního klíče s použitím symetrické kryptografie a důvěryhodného serveru.

Formální specifikace protokolu

1. A → S : A, B
2. S → A : { B, Kab, T, { A, Kab, T }$_{Kb}$ }$_{Ka}$
3. A → B : { A, Kab, T }$_{Kb}$
4. A → B : { MSG }$_{Kab}$

Jelikož LySatool neumožňuje modelovat časová razítka, timestamp *T* je v LySa specifikaci

modelováno jako nonce.

Výsledky analýzy

$\psi = \{ \varnothing \}$

Množina potenciálních znalostí útočníka neobsahuje žádné důvěrné informace.

Nalezené útoky

Nebyly nalezeny žádné útoky.

Nenalezené útoky

Nástroj nenalezl útok publikovaný v [60]:

1. $A \rightarrow S : A, B$
2. $S \rightarrow A : \{ B, Kab, T, \{ A, Kab, T \}_{Kb} \}_{Ka}$
3. $A \rightarrow B : \{ A, Kab, T \}_{Kb}$
3′. $I_A \rightarrow B : \{ A, Kab, T \}_{Kb}$

První tři kroky představují normální průběh protokolu. V kroku 3′ se útočník vydává za subjekt *A* a subjektu *B* znovu odešle zprávu z kroku 3. Subjekt *B* si nyní myslí, že *A* s ním chce navázat dvě relace.

6.1.4 Lowe modified Denning-Sacco shared key

Funkce protokolu

Vzájemná autentizace a distribuce relačního klíče s použitím symetrické kryptografie a důvěryhodného serveru. Opravený protokol Denning-Sacco shared key.

Formální specifikace protokolu

1. $A \rightarrow S : A, B$
2. $S \rightarrow A : \{ B, Kab, T, \{ A, Kab, T \}_{Kb} \}_{Ka}$
3. $A \rightarrow B : \{ A, Kab, T \}_{Kb}$
4. $B \rightarrow A : \{ Nb, Nb2 \}_{Kab}$
5. $A \rightarrow B : \{ Nb2 \}_{Kab}$
6. $A \rightarrow B : \{ MSG \}_{Kab}$

Pro zamezení útoku proveditelného na původní verzi protokolu je zde přidán tzv. nonce handshake (zprávy 4 a 5), který je kvůli omezením nástroje LySatool upraven (liší se od nonce handshake v publikovaných specifikacích).

Výsledky analýzy

$\psi = \{ \varnothing \}$

Množina potenciálních znalostí útočníka neobsahuje žádné důvěrné informace.

Nalezené útoky

Nebyly nalezeny žádné útoky.

Nenalezené útoky

Nebyly publikovány žádné útoky.

6.1.5 Encrypted Key Exchange (EKE)

Funkce protokolu

Vzájemná autentizace a distribuce relačního klíče s použitím symetrické a asymetrické kryptografie.

Formální specifikace protokolu

1. $A \rightarrow B : \{ Ka \}_{Kab}$
2. $B \rightarrow A : \{ \{ K \}_{Ka} \}_{Kab}$
3. $A \rightarrow B : \{ Na \}_{K}$
4. $B \rightarrow A : \{ Na, Nb \}_{K}$
5. $A \rightarrow B : \{ Nb \}_{K}$
6. $A \rightarrow B : \{ MSG \}_{K}$

Předpokládá se, že komunikující subjekty na počátku sdílí „slabé“ (odhadnutelné) heslo / klíč *Kab*, který je použit pro výměnu „silného“ relačního klíče *K*.

Výsledky analýzy

$\psi = \{ (a1_{ij}, a2_{ij}), (a1_{i0}, a2_{i0}), (b3_{ij}, b1_{ij}), (b3_{0j}, b1_{0j}), (a4_{ij}, b7_{ij}), (a6_{ij}, b4_{ij}), (a6_{ij}, b7_{ij}), (a7_{ij}, b4_{ij}), (a4_{i0}, b7_{0s}), (a6_{i0}, b4_{0s}), (a6_{i0}, b7_{0s}), (a7_{i0}, b4_{0s}), (b2_{0j}, a3_{r0}), (a4_{i0}, b4_{0s}), (a4_{i0}, b1_{0s}), (a4_{i0}, a2_{r0}), (b5_{0j}, a5_{r0}), (a6_{i0}, b6_{0s}), (a6_{i0}, b1_{0s}), (a6_{i0}, a2_{r0}), (a7_{i0}, b7_{0s}), (a7_{i0}, b1_{0s}), (a7_{i0}, a2_{r0}) \}$

Množina potenciálních znalostí útočníka neobsahuje žádné důvěrné informace.

Nalezené útoky

Žádné z porušení $(a1_{ij}, a2_{ij})$, $(a1_{i0}, a2_{i0})$, $(b3_{ij}, b1_{ij})$, $(b3_{0j}, b1_{0j})$, $(a4_{ij}, b7_{ij})$, $(a6_{ij}, b4_{ij})$, $(a6_{ij}, b7_{ij})$ a $(a7_{ij}, b4_{ij})$ nepředstavuje proveditelný útok.

Ostatní porušení také nepředstavují útok. Znamenají pouze, že pokud útočník vystupuje jako legitimní účastník komunikace a naváže dvě relace se dvěma jinými subjekty, mohou být zprávy zašifrované v jedné relaci dešifrovány ve druhé relaci. Tzn., že v obou relacích je použit stejný relační klíč.

Nenalezené útoky

Nástroj nenalezl útok publikovaný v [43]:

1. $A \rightarrow I_B : \{ Ka \}_{Kab}$
1′. $I_B \rightarrow A : \{ Ka \}_{Kab}$
2′. $A \rightarrow I_B : \{ \{ K \}_{Ka} \}_{Kab}$
2. $I_B \rightarrow A : \{ \{ K \}_{Ka} \}_{Kab}$
3. $A \rightarrow I_B : \{ Na \}_{K}$
3′. $I_B \rightarrow A : \{ Na \}_{K}$
4′. $A \rightarrow I_B : \{ Na, Nb \}_{K}$
4. $I_B \rightarrow A : \{ Na, Nb \}_{K}$
5. $A \rightarrow B : \{ Nb \}_{K}$
5′. $I_B \rightarrow A : \{ Nb \}_{K}$
6. $A \rightarrow I_B : \{ MSG \}_{K}$
6′. $I_B \rightarrow A : \{ MSG \}_{K}$

Subjekt *A* iniciuje komunikaci s útočníkem *I* vydávajícím se za *B*, který následně naváže další relaci s *A*. Zprávy z jedné relace jsou přehrávány ve druhé relaci tak, že po provedení útoku je v obou relacích akceptován stejný relační klíč *K*, subjekt *A* věří, že navázal relaci s *B* a že subjekt *B* s ním navázal druhou relaci.

6.1.6 Kao Chow Authentication v.1

Funkce protokolu

Vzájemná autentizace a distribuce relačního klíče s použitím symetrické kryptografie a důvěryhodného serveru.

Formální specifikace protokolu

1. $A \rightarrow S : A, B, Na$
2. $S \rightarrow B : \{ A, B, Na, Kab \}_{Ka} , \{ A, B, Na, Kab \}_{Kb}$
3. $B \rightarrow A : \{ A, B, Na, Kab \}_{Ka} , \{ Na \}_{Kab} , Nb$
4. $A \rightarrow B : \{ Nb \}_{Kab}$
5. $A \rightarrow B : \{ MSG \}_{Kab}$

Výsledky analýzy

ψ = { ($b2_{ij}$, $b3_{ij}$), ($b2_{ij}$, $b4_{ij}$), ($a3_{ij}$, $b4_{ij}$), ($a3_{ij}$, $a2_{ij}$), ($b2_{0j}$, $b3_{0j}$), ($b2_{0j}$, $b4_{0j}$), ($a3_{i0}$, $a2_{i0}$) }

Množina potenciálních znalostí útočníka neobsahuje žádné důvěrné informace.

Nalezené útoky

Žádné z porušení nepředstavuje proveditelný útok.

Nenalezené útoky

Publikován byl pouze útok založený na kompromitovaném klíči starší relace [59].

6.1.7 Kao Chow Authentication v.2

Funkce protokolu

Vzájemná autentizace a distribuce relačního klíče s použitím symetrické kryptografie a důvěryhodného serveru. Opravená verze protokolu Kao Chow Authentication v.1.

Formální specifikace protokolu

1. A → S : A, B, Na
2. S → B : { A, B, Na, Kab, Kt }$_{Ka}$, { A, B, Na, Kab, Kt }$_{Kb}$
3. B → A : B, { A, B, Na, Kab, Kt }$_{Ka}$, { Na, Kab }$_{Kt}$, Nb
4. A → B : { Nb, Kab }$_{Kt}$
5. A → B : { MSG }$_{Kab}$

Pro zamezení útoku proveditelného na původní verzi protokolu je zde navíc použit symetrický klíč *Kt*.

Výsledky analýzy

ψ = { ($b2_{ij}$, $b3_{ij}$), ($a3_{ij}$, $a2_{ij}$), ($b2_{0j}$, $b3_{0j}$), ($a3_{i0}$, $a2_{i0}$) }

Množina potenciálních znalostí útočníka neobsahuje žádné důvěrné informace.

Nalezené útoky

Žádné z porušení nepředstavuje proveditelný útok.

Nenalezené útoky

Nebyly publikovány žádné útoky.

6.1.8 Needham-Schroeder Public Key

Funkce protokolu

Vzájemná autentizace s použitím asymetrické kryptografie a důvěryhodného serveru.

Formální specifikace protokolu

1. $A \rightarrow S : A, B$
2. $S \rightarrow A : \{ B, Kb \}_{Ks}^{-1}$
3. $A \rightarrow B : \{ A, Na \}_{Kb}$
4. $B \rightarrow S : B, A$
5. $S \rightarrow B : \{ A, Ka \}_{Ks}^{-1}$
6. $B \rightarrow A : \{ Na, Nb \}_{Ka}$
7. $A \rightarrow B : \{ Nb \}_{Kb}$

Výsledky analýzy

$\psi = \{ (s1_{ij}, a1_{rj}), (s2_{ij}, b2_{is}), (s1_{i0}, a1_{r0}), (s1_{00}, a1_{r0}), (s1_{0j}, a1_{rj}), (s2_{i0}, b2_{is}) \}$

Množina potenciálních znalostí útočníka neobsahuje žádné důvěrné informace.

Nalezené útoky

Žádné z porušení nepředstavuje proveditelný útok.

Nenalezené útoky

Nástroj nenalezl útok publikovaný v [27]:

3. $A \rightarrow I : \{ A, Na \}_{Ki}$
3′. $I_A \rightarrow B : \{ A, Na \}_{Kb}$
6′. $B \rightarrow I_A : \{ Na, Nb \}_{Ka}$
6. $I \rightarrow A : \{ Na, Nb \}_{Ka}$
7. $A \rightarrow I : \{ Nb \}_{Ki}$
7′. $I_A \rightarrow B : \{ Nb \}_{Kb}$

V popisu je vynechána komunikace se serverem *S*. *A* iniciuje komunikaci s útočníkem *I*, který vystupuje jako legitimní účastník. Útočník se následně vydává za subjekt *A* a zahájí komunikaci se subjektem *B*. Získá tak zprávu $\{ Na, Nb \}_{Ka}$, kterou využije v první relaci, aby získal nonce *Nb*. To nakonec ve druhé relaci pošle subjektu *B* ve zprávě zašifrované jeho veřejným klíčem. *B* nyní věří, že s ním navázal relaci subjekt *A*.

6.1.9 Lowe's fixed version of Needham-Schroeder Public Key

Funkce protokolu

Vzájemná autentizace s použitím asymetrické kryptografie a důvěryhodného serveru. Opravená verze protokolu Needham-Schroeder Public Key.

Formální specifikace protokolu

1. $A \rightarrow S : A, B$
2. $S \rightarrow A : \{ B, Kb \}_{Ks}^{-1}$
3. $A \rightarrow B : \{ A, Na \}_{Kb}$
4. $B \rightarrow S : B, A$
5. $S \rightarrow B : \{ A, Ka \}_{Ks}^{-1}$
6. $B \rightarrow A : \{ B, Na, Nb \}_{Ka}$
7. $A \rightarrow B : \{ Nb \}_{Kb}$

Oproti původní verzi protokolu je zde přidána identita příjemce *B* do zprávy 6.

Výsledky analýzy

$\psi = \{ (s1_{ij}, a1_{rj}), (s2_{ij}, b2_{is}), (s1_{i0}, a1_{r0}), (s1_{00}, a1_{r0}), (s1_{0j}, a1_{rj}), (s2_{i0}, b2_{is}) \}$

Množina potenciálních znalostí útočníka neobsahuje žádné důvěrné informace.

Nalezené útoky

Žádné z porušení nepředstavuje proveditelný útok.

Nenalezené útoky

Nebyly publikovány žádné útoky.

6.1.10 Needham-Schroeder Symmetric Key

Funkce protokolu

Vzájemná autentizace a distribuce relačního klíče s použitím symetrických klíčů a důvěryhodného serveru.

Formální specifikace protokolu

1. A → S : A, B, Na
2. S → A : { Na, B, Kab, { A, Kab }$_{Kb}$ }$_{Ka}$
3. A → B : { A, Kab }$_{Kb}$
4. B → A : { Nb, Nb2 }$_{Kab}$
5. A → B : { Nb2 }$_{Kab}$
6. A → B : { MSG }$_{Kab}$

Nonce handshake (zprávy 4 a 5) je zde kvůli omezením nástroje LySatool upraven.

Výsledky analýzy

ψ = { (a3$_{ij}$, b4$_{ij}$) }

Množina potenciálních znalostí útočníka neobsahuje žádné důvěrné informace.

Nalezené útoky

Porušení ($a3_{ij}$, $b4_{ij}$) nepředstavuje proveditelný útok.

Nenalezené útoky

Publikován byl pouze útok založený na kompromitovaném klíči starší relace [61].

6.1.11 Lowe modified Wide Mouthed Frog

Funkce protokolu

Vzájemná autentizace a distribuce relačního klíče s použitím symetrické kryptografie a důvěryhodného serveru. Opravená verze protokolu Wide Mouthed Frog.

Formální specifikace protokolu

1. A → S : A, { B, Kab }$_{Ka}$
2. S → B : { A, Kab }$_{Kb}$
3. B → A : { Nb, Nb2 }$_{Kab}$
4. A → B : { Nb2 }$_{Kab}$
5. A → B : { MSG }$_{Kab}$

Oproti specifikaci protokolu uvedené v [9], výše uvedená specifikace postrádá časová razítka, která v LySatool modelovat nelze. Ze stejných důvodů je zde pozměněn tzv. nonce handshake (zprávy 3 a 4).

Výsledky analýzy

ψ = { ($a3_{ij}$, $b4_{ij}$), ($b2_{0j}$, $a2_{r0}$), ($a3_{i0}$, $b3_{0s}$), ($a3_{i0}$, $b4_{0s}$), ($a4_{i0}$, $b4_{0s}$) }

Množina potenciálních znalostí útočníka neobsahuje žádné důvěrné informace.

Nalezené útoky

Porušení ($a3_{ij}$, $b4_{ij}$) nepředstavuje proveditelný útok.

Porušení ($b2_{0j}$, $a2_{r0}$), ($a3_{i0}$, $b3_{0s}$), ($a3_{i0}$, $b4_{0s}$), ($a4_{i0}$, $b4_{0s}$) také nepředstavují útok. Znamenají pouze, že pokud útočník vystupuje jako legitimní účastník komunikace a naváže dvě relace se dvěma jinými subjekty, mohou být zprávy zašifrované v jedné relaci dešifrovány ve druhé relaci. Tzn., že v obou relacích je použit stejný relační klíč.

Nenalezené útoky

Nebyly publikovány žádné útoky.

6.1.12 Woo and Lam Mutual Authentication

Funkce protokolu

Vzájemná autentizace a distribuce relačního klíče s použitím symetrických klíčů a důvěryhodného serveru.

Formální specifikace protokolu

1. A → B : A, Na
2. B → A : B, Nb
3. A → B : { A, B, Na, Nb }$_{Ka}$
4. B → S : { A, B, Na, Nb }$_{Ka}$, { A, B, Na, Nb }$_{Kb}$
5. S → B : { B, Na, Nb, Kab }$_{Ka}$, { A, Na, Nb, Kab }$_{Kb}$
6. B → A : { B, Na, Nb, Kab }$_{Ka}$, { Na, Nb }$_{Kab}$
7. A → B : { Nb }$_{Kab}$
8. A → B : { MSG }$_{Kab}$

Výsledky analýzy

ψ = { ($b1_{ij}$, $b2_{ij}$), ($b3_{ij}$, CPDY), (CPDY, $b4_{ij}$), (CPDY, $b5_{ij}$), ($s4_{ij}$, $s2_{ij}$), ($a4_{ij}$, $b5_{ij}$), ($b1_{0j}$, $b2_{0j}$), ($s4_{0j}$, $s2_{0j}$), ($s4_{i0}$, $s2_{i0}$), ($s4_{00}$, $s2_{00}$) }

Množina potenciálních znalostí útočníka neobsahuje žádné důvěrné informace.

Nalezené útoky

Porušení ($b1_{ij}$, $b2_{ij}$), ($b3_{ij}$, *CPDY*), (*CPDY*, $b4_{ij}$) a (*CPDY*, $b5_{ij}$) představují útok publikovaný v [58]:

1. $I_A \rightarrow B : A, B$
2. $B \rightarrow I_A : B, Nb$
3. $I_A \rightarrow B : G$
4. $B \rightarrow I_S : G, \{ A, B, B, Nb \}_{Kb}$

1′. $I_A \rightarrow B : A, Nb$

2′. $B \rightarrow I_A : B, Nb2$

3′. $I_A \rightarrow B : G2$

4′. $B \rightarrow I_S : G2, \{ A, B, Nb, Nb2 \}_{Kb}$

5. $I_S \rightarrow B : G3, \{ A, B, Nb, Nb2 \}_{Kb}$
6. $B \rightarrow I_A : G3, \{ B, Nb \}_{Nb2}$
7. $I_A \rightarrow B : \{ Nb \}_{Nb2}$
8. $I_A \rightarrow B : \{ MSG \}_{Nb2}$

Útočník *I* se vydává za *A* a iniciuje komunikaci se subjektem *B*, přičemž místo nonce *Na* v kroku 1 použije identitu příjemce *B*. V kroku 3 odešle subjektu *B* libovolnou hodnotu *G* vyhovující formátu očekávané zprávy. Aby mohl útočník v kroku 5 odpovědět zprávou podle pravidel, iniciuje druhou relaci se subjektem *B*. V první zprávě použije nonce *Nb* z první relace. Zprávu $\{ A, B, Nb, Nb2 \}_{Kb}$ z druhé relace pak použije v kroku 5 v první relaci, čímž dojde k tomu, že *B* akceptuje nonce *Nb*2 jako relační klíč.

Žádné z porušení ($s4_{ij}$, $s2_{ij}$), ($a4_{ij}$, $b5_{ij}$), ($b1_{0j}$, $b2_{0j}$), ($s4_{0j}$, $s2_{0j}$), ($s4_{i0}$, $s2_{i0}$) a ($s4_{00}$, $s2_{00}$) nepředstavuje proveditelný útok.

Nenalezené útoky

V [43] je publikován postup, který vede k použití stejného relačního klíče ve dvou relacích mezi subjektem *A* a útočníkem *I* vystupujícím jako legitimní účastník komunikace. Toto však nepředstavuje skutečný útok na protokol.

6.1.13 Woo and Lam Pi

Funkce protokolu

Jednosměrná autentizace s použitím symetrických klíčů a důvěryhodného serveru. Zjednodušení protokolu Woo and Lam Pi 3, Woo and Lam Pi 2, Woo and Lam Pi 1 a Woo and Lam Pi f.

Formální specifikace protokolu

1. $A \rightarrow B : A$
2. $B \rightarrow A : Nb$
3. $A \rightarrow B : \{ Nb \}_{Ka}$
4. $B \rightarrow S : \{ A, \{ Nb \}_{Ka} \}_{Kb}$
5. $S \rightarrow B : \{ Nb \}_{Kb}$

Výsledky analýzy

$\psi = \{ (a1_{ij}, s2_{is}), (a1_{i0}, s2_{is}), (s3_{0j}, b2_{rj}), (s3_{ij}, b2_{rj}), (a1_{ij}, s2_{i0}), (s3_{ij}, b2_{0j}) \}$

Množina potenciálních znalostí útočníka neobsahuje žádné důvěrné informace.

Nalezené útoky

Porušení ($a1_{ij}$, $s2_{is}$) představuje útok, který pravděpodobně nebyl publikován:

1. $I_A \rightarrow B : A$
1′. $A \rightarrow I_C : A$
2. $B \rightarrow I_A : Nb$
2′. $I_C \rightarrow A : Nb$
3′. $A \rightarrow I_C : \{ Nb \}_{Ka}$
3. $I_A \rightarrow B : \{ Nb \}_{Ka}$
4. $B \rightarrow S : \{ A, \{ Nb \}_{Ka} \}_{Kb}$
5. $S \rightarrow B : \{ Nb \}_{Kb}$

Útočník *I* se vydává za *A* a iniciuje komunikaci se subjektem *B*. Aby mohl v kroku 3 odpovědět zprávou $\{ Nb \}_{Ka}$, vydává se za subjekt *C*, se kterým se *A* pokusí navázat jinou relaci. V této druhé relaci pak útočník použije nonce *Nb* z první relace a v kroku 3′ pak získá požadovanou zprávu. Po provedení útoku *B* věří, že s ním navázal relaci subjekt *A*. Porušení ($a1_{i0}$, $s2_{is}$) představuje stejný útok s tím rozdílem, že se útočník ve druhé relaci nevydává za subjekt *C*, ale vystupuje jako legitimní účastník.

Porušení ($s3_{0j}$, $b2_{rj}$) představuje útok publikovaný v [43]:

1. $I_A \rightarrow B : A$
1′. $I \rightarrow B : I$
2. $B \rightarrow I_A : Nb1$
2′. $B \rightarrow I : Nb2$
3. $I_A \rightarrow B : G$
3′. $I \rightarrow B : \{ Nb1 \}_{Ki}$
4. $B \rightarrow S : \{ A, G \}_{Kb}$
4′. $B \rightarrow S : \{ I, \{ Nb1 \}_{Ki} \}_{Kb}$
5. $S \rightarrow B : \{ Nb1 \}_{Kb}$

Útočník *I* se vydává za *A* a iniciuje komunikaci se subjektem *B*. Zároveň s *B* naváže další relaci, ve které vystupuje jako legitimní účastník. V kroku 3 odpoví útočník v první relaci zprávou *G*, což může být jakákoliv hodnota odpovídající očekávanému formátu. Ve druhé relaci odešle subjektu *B* nonce *Nb1* z první relace zašifrované svým klíčem *Ki*. *B* tuto zprávu přepošle serveru, který pak v kroku 5 odpoví zprávou obsahující nonce *Nb1* zašifrované klíčem *Kb*. *B* nyní věří, že relaci s ním navázal *A*.

Žádné z porušení ($s3_{ij}$, $b2_{rj}$), ($a1_{ij}$, $s2_{i0}$), ($s3_{ij}$, $b2_{0j}$) nepředstavuje proveditelný útok.

Nenalezené útoky

Nástroj nenalezl útok publikovaný v [43]:

1. $I_A \rightarrow B : A$
2. $B \rightarrow I_A : Nb$
3. $I_A \rightarrow B : G$
4. $B \rightarrow I_S : \{ A, G \}_{Kb}$
1′. $B \rightarrow I_C : B$
2′. $I_C \rightarrow B : I, \{ Nb \}_{Ki}$
3′. $B \rightarrow I_C : \{ I, \{ Nb \}_{Ki} \}_{Kb}$
4′. $I_B \rightarrow S : \{ I, \{ Nb \}_{Ki} \}_{Kb}$
5′. $S \rightarrow I_B : \{ Nb \}_{Kb}$
5. $I_S \rightarrow B : \{ Nb \}_{Kb}$

Útočník *I* se vydává za *A* a iniciuje komunikaci se subjektem *B*. Aby mohl v kroku 5 odpovědět zprávou podle pravidel, vydává se za subjekt *C*, se kterým se *B* pokusí navázat jinou relaci. V této druhé relaci pak ve druhém kroku místo nonce *Nb* odešle zprávu *I*, $\{ Nb \}_{Ki}$, kterou *B* zašifruje svým klíčem *Kb*. Výslednou zprávu pošle útočník (vydávajíc se za *B*) serveru *S*, který mu podle pravidel vrátí nonce *Nb* zašifrované klíčem subjektu *B*. Tuto zprávu útočník přehraje v kroku 5 v první relaci.

Po provedení útoku *B* věří, že s ním navázal relaci subjekt *A*.

V [43] je dále publikován útok založený na předpokladu, že operace šifrování je komutativní.

6.1.14 Woo and Lam Pi 1

Funkce protokolu

Jednosměrná autentizace s použitím symetrických klíčů a důvěryhodného serveru. Zjednodušení protokolu Woo and Lam Pi f.

Formální specifikace protokolu

1. $A \rightarrow B : A$
2. $B \rightarrow A : Nb$
3. $A \rightarrow B : \{ A, B, Nb \}_{Ka}$
4. $B \rightarrow S : \{ A, B, \{ A, B, Nb \}_{Ka} \}_{Kb}$
5. $S \rightarrow B : \{ A, B, Nb \}_{Kb}$

Výsledky analýzy

$\psi = \{ (b1_{ij}, b2_{ij}), (b1_{0j}, b2_{0j}), (s3_{ij}, s1_{ij}), (s3_{i0}, s1_{i0}), (s3_{0j}, s1_{0j}), (s3_{00}, s1_{00}) \}$

Množina potenciálních znalostí útočníka neobsahuje žádné důvěrné informace.

Nalezené útoky

Porušení ($b1_{ij}$, $b2_{ij}$) představuje útok, který je variantou útoku publikovaného v [43]:

1. $I_A \rightarrow B : A$
2. $B \rightarrow I_A : Nb$
3. $I_A \rightarrow B : Nb$
4. $B \rightarrow I_S : \{ A, B, Nb \}_{Kb}$
5. $I_S \rightarrow B : \{ A, B, Nb \}_{Kb}$

Útočník *I* se vydává za *A* a iniciuje komunikaci s *B*. V kroku 3 pak místo zprávy $\{ A, B, Nb \}_{Ka}$ odešle pouze nonce *Nb*. Ve 4. kroku se vydává za *S* a od *B* získá zprávu, kterou přehraje v kroku 5. Subjekt *B* nyní věří, že s ním navázal relaci subjekt *A*.

Žádné z porušení ($b1_{0j}$, $b2_{0j}$), ($s3_{ij}$, $s1_{ij}$), ($s3_{i0}$, $s1_{i0}$), ($s3_{0j}$, $s1_{0j}$) a ($s3_{00}$, $s1_{00}$) nepředstavuje proveditelný útok.

Nenalezené útoky

Nebyly publikovány žádné další útoky.

6.1.15 Woo and Lam Pi 2

Funkce protokolu

Jednosměrná autentizace s použitím symetrických klíčů a důvěryhodného serveru. Zjednodušení protokolu Woo and Lam Pi 1 a Woo and Lam Pi f.

Formální specifikace protokolu

1. $A \rightarrow B : A$
2. $B \rightarrow A : Nb$
3. $A \rightarrow B : \{ A, Nb \}_{Ka}$
4. $B \rightarrow S : \{ A, \{ A, Nb \}_{Ka} \}_{Kb}$
5. $S \rightarrow B : \{ A, Nb \}_{Kb}$

Výsledky analýzy

$\psi = \{ (a1_{ij}, s2_{is}), (a1_{i0}, s2_{is}), (b1_{ij}, b2_{ij}), (a1_{ij}, s2_{i0}), (b1_{0j}, b2_{0j}), (s3_{ij}, s1_{ij}), (s3_{i0}, s1_{i0}), (s3_{0j}, s1_{0j}), (s3_{00}, s1_{00}) \}$

Množina potenciálních znalostí útočníka neobsahuje žádné důvěrné informace.

Nalezené útoky

Porušení ($a1_{ij}$, $s2_{is}$) představuje útok, který pravděpodobně nebyl publikován:

1. $I_A \rightarrow B : A$
1′. $A \rightarrow I_C : A$
2. $B \rightarrow I_A : Nb$
2′. $I_C \rightarrow A : Nb$
3′. $A \rightarrow I_C : \{ A, Nb \}_{Ka}$
3. $I_A \rightarrow B : \{ A, Nb \}_{Ka}$
4. $B \rightarrow S : \{ A, \{ A, Nb \}_{Ka} \}_{Kb}$
5. $S \rightarrow B : \{ A, Nb \}_{Kb}$

Útočník *I* se vydává za *A* a iniciuje komunikaci se subjektem *B*. Aby mohl v kroku 3 odpovědět zprávou $\{ A, Nb \}_{Ka}$, vydává se za subjekt *C*, se kterým se *A* pokusí navázat jinou relaci. V této druhé relaci pak útočník použije nonce *Nb* z první relace a v kroku 3′ pak získá požadovanou zprávu. Po provedení útoku *B* věří, že s ním navázal relaci subjekt *A*. Porušení ($a1_{i0}$, $s2_{is}$) představuje stejný útok s tím rozdílem, že se útočník ve druhé relaci nevydává za subjekt *C*, ale vystupuje jako legitimní účastník.

Porušení ($b1_{ij}$, $b2_{ij}$) představuje útok publikovaný v [43]:

1. $I_A \rightarrow B : A$
2. $B \rightarrow I_A : Nb$
3. $I_A \rightarrow B : Nb$
4. $B \rightarrow I_S : \{ A, Nb \}_{Kb}$
5. $I_S \rightarrow B : \{ A, Nb \}_{Kb}$

Útočník *I* se vydává za *A* a iniciuje komunikaci s *B*. V kroku 3 pak místo zprávy $\{ A, Nb \}_{Ka}$ odešle pouze nonce *Nb*. Ve 4. kroku se vydává za *S* a od *B* získá zprávu, kterou přehraje v kroku 5. Subjekt *B* nyní věří, že s ním navázal relaci subjekt *A*.

Žádné z porušení ($a1_{ij}$, $s2_{i0}$), ($b1_{0j}$, $b2_{0j}$), ($s3_{ij}$, $s1_{ij}$), ($s3_{i0}$, $s1_{i0}$), ($s3_{0j}$, $s1_{0j}$) a ($s3_{00}$, $s1_{00}$) nepředstavuje proveditelný útok.

Nenalezené útoky

Nebyly publikovány žádné další útoky.

6.1.16 Woo and Lam Pi 3

Funkce protokolu

Jednosměrná autentizace s použitím symetrických klíčů a důvěryhodného serveru. Zjednodušení protokolu Woo and Lam Pi 2, Woo and Lam Pi 1 a Woo and Lam Pi f.

Formální specifikace protokolu

1. $A \rightarrow B : A$
2. $B \rightarrow A : Nb$
3. $A \rightarrow B : \{ Nb \}_{Ka}$
4. $B \rightarrow S : \{ A, \{ Nb \}_{Ka} \}_{Kb}$
5. $S \rightarrow B : \{ A, Nb \}_{Kb}$

Výsledky analýzy

ψ = { ($a1_{ij}$, $s2_{is}$), ($a1_{i0}$, $s2_{is}$), ($b1_{ij}$, $b2_{ij}$), ($a1_{ij}$, $s2_{i0}$), ($b1_{0j}$, $b2_{0j}$), ($s3_{ij}$, $s1_{ij}$), ($s3_{i0}$, $s1_{i0}$), ($s3_{0j}$, $s1_{0j}$), ($s3_{00}$, $s1_{00}$) }

Množina potenciálních znalostí útočníka neobsahuje žádné důvěrné informace.

Nalezené útoky

Porušení ($a1_{ij}$, $s2_{is}$) představuje útok, který pravděpodobně nebyl publikován:

1. $I_A \rightarrow B : A$
1′. $A \rightarrow I_C : A$
2. $B \rightarrow I_A : Nb$
2′. $I_C \rightarrow A : Nb$
3′. $A \rightarrow I_C : \{ Nb \}_{Ka}$
3. $I_A \rightarrow B : \{ Nb \}_{Ka}$
4. $B \rightarrow S : \{ A, \{ Nb \}_{Ka} \}_{Kb}$
5. $S \rightarrow B : \{ A, Nb \}_{Kb}$

Útočník *I* se vydává za *A* a iniciuje komunikaci se subjektem *B*. Aby mohl v kroku 3 odpovědět zprávou $\{ Nb \}_{Ka}$, vydává se za subjekt *C*, se kterým se *A* pokusí navázat jinou relaci. V této druhé relaci pak útočník použije nonce *Nb* z první relace a v kroku 3′ pak získá požadovanou zprávu. Po provedení útoku *B* věří, že s ním navázal relaci subjekt *A*. Porušení ($a1_{i0}$, $s2_{is}$) představuje stejný útok s tím rozdílem, že útočník ve druhé relaci vystupuje jako legitimní účastník.

Porušení ($b1_{ij}$, $b2_{ij}$) představuje útok publikovaný v [43]:

1. $I_A \rightarrow B : A$
2. $B \rightarrow I_A : Nb$
3. $I_A \rightarrow B : Nb$
4. $B \rightarrow I_S : \{ A, Nb \}_{Kb}$
5. $I_S \rightarrow B : \{ A, Nb \}_{Kb}$

Útočník *I* se vydává za *A* a iniciuje komunikaci s *B*. V kroku 3 pak místo zprávy $\{ Nb \}_{Ka}$ odešle nezašifrované nonce *Nb*. Ve 4. kroku se vydává za *S* a od *B* získá zprávu, kterou přehraje v kroku 5. Subjekt *B* nyní věří, že s ním navázal relaci subjekt *A*.

Žádné z porušení ($a1_{ij}$, $s2_{i0}$), ($b1_{0j}$, $b2_{0j}$), ($s3_{ij}$, $s1_{ij}$), ($s3_{i0}$, $s1_{i0}$), ($s3_{0j}$, $s1_{0j}$), ($s3_{00}$, $s1_{00}$) nepředstavuje proveditelný útok.

Nenalezené útoky

V [43] je publikován útok založený na předpokladu, že operace šifrování je komutativní.

6.1.17 Yahalom

Funkce protokolu

Vzájemná autentizace a distribuce relačního klíče s použitím symetrických klíčů a důvěryhodného serveru.

Formální specifikace protokolu

1. $A \rightarrow B : A, Na$
2. $B \rightarrow S : B, \{ A, Na, Nb \}_{Kb}$
3. $S \rightarrow A : \{ B, Na, Kab, Nb \}_{Ka}, \{ A, Kab \}_{Kb}$
4. $A \rightarrow B : \{ A, Kab \}_{Kb}, \{ Nb \}_{Kab}$
5. $A \rightarrow B : \{ MSG \}_{Kab}$

Výsledky analýzy

$\psi = \{ (a2_{ij}, b4_{ij}) \}$

Množina potenciálních znalostí útočníka neobsahuje žádné důvěrné informace.

Nalezené útoky

Porušení ($a2_{ij}$, $b4_{ij}$) nepředstavuje proveditelný útok.

Nenalezené útoky

Nástroj nenalezl útok publikovaný v [43]:

1. $I_A \rightarrow B : A, Na$
2. $B \rightarrow I_S : B, \{ A, Na, Nb \}_{Kb}$
3. -
4. $A \rightarrow B : \{ A, Na, Nb \}_{Kb}, \{ Nb \}_{(Na, Nb)}$
5. $A \rightarrow B : \{ MSG \}_{(Na, Nb)}$

Útočník *I* se vydává za subjekt *A* a naváže relaci s *B*. V kroku 4 přehraje zprávu z kroku 2, čímž dojde k tomu, že subjekt *B* akceptuje (*Na*, *Nb*) jako relační klíč.

6.1.18 BAN simplified version of Yahalom

Funkce protokolu

Vzájemná autentizace a distribuce relačního klíče s použitím symetrických klíčů a důvěryhodného serveru. Upravená verze protokolu Yahalom.

Formální specifikace protokolu

1. $A \rightarrow B : A, Na$
2. $B \rightarrow S : B, Nb, \{ A, Na \}_{Kb}$
3. $S \rightarrow A : Nb, \{ B, Na, Kab \}_{Ka}, \{ A, Nb, Kab \}_{Kb}$
4. $A \rightarrow B : \{ A, Nb, Kab \}_{Kb}, \{ Nb \}_{Kab}$
5. $A \rightarrow B : \{ MSG \}_{Kab}$

Výsledky analýzy

$\psi = \{ (a2_{ij}, b4_{ij}) \}$

Množina potenciálních znalostí útočníka neobsahuje žádné důvěrné informace.

Nalezené útoky

Porušení ($a2_{ij}$, $b4_{ij}$) nepředstavuje proveditelný útok.

Nenalezené útoky

Nástroj nenalezl dva útoky publikované v [29]:

1. $I_A \rightarrow B : A, Na$
2. $B \rightarrow S : B, Nb, \{ A, Na \}_{Kb}$

1′. $I_A \rightarrow B : A, (Nb, Na)$

2′. $B \rightarrow I_S : B, Nb2, \{ A, Nb, Na \}_{Kb}$

3. -
4. $I_A \rightarrow B : \{ A, Nb, Na \}_{Kb}, \{ Nb \}_{Na}$
5. $I_A \rightarrow B : \{ MSG \}_{Na}$

Útočník *I* se vydává za subjekt *A* a naváže dvě relace se subjektem *B*. Zprávu $\{ A, Nb, Na \}_{Kb}$ získanou ve druhé relaci použije v první relaci tak, že subjekt *B* akceptuje nonce *Na* jako relační klíč.

Druhý útok nepředstavuje tak velké riziko jako první útok:

1. $A \rightarrow I_B : A, Na$
1′. $I_B \rightarrow A : B, Na$
2′. $A \rightarrow I_S : A, Na2, \{ B, Na \}_{Ka}$
2″. $I_A \rightarrow S : A, Na, \{ B, Na \}_{Ka}$
3′. $S \rightarrow I_B : Na, \{ A, Na, Kab \}_{Kb}, \{ B, Na, Kab \}_{Ka}$
2. -
3. $I_S \rightarrow A : Ni, \{ B, Na, Kab \}_{Ka}, \{ A, Na, Kab \}_{Kb}$
4. $A \rightarrow I_B : \{ A, Na, Kab \}_{Kb}, \{ Ni \}_{Kab}$
5. $A \rightarrow I_B : \{ MSG \}_{Kab}$

Subjekt *A* naváže relaci s útočníkem *I*, který se vydává za subjekt *B*. Útočník naváže druhou relaci se subjektem *A*. Zprávy $\{ A, Na, Kab \}_{Kb}$ a $\{ B, Na, Kab \}_{Ka}$ z této relace pak použije v první relaci. Po provedení útoku subjekt *A* věří, že sdílí relační klíč se subjektem *B*.

6.1.19 Lowe's modified version of Yahalom

Funkce protokolu

Vzájemná autentizace a distribuce relačního klíče s použitím symetrických klíčů a důvěryhodného serveru. Opravená verze protokolu Yahalom.

Formální specifikace protokolu

1. $A \rightarrow B : A, Na$
2. $B \rightarrow S : \{ A, Na, Nb \}_{Kb}$
3. $S \rightarrow A : \{ B, Na, Kab, Nb \}_{Ka}$
4. $S \rightarrow B : \{ A, Kab \}_{Kb}$
5. $A \rightarrow B : \{ A, B, S, Nb \}_{Kab}$
6. $A \rightarrow B : \{ MSG \}_{Kab}$

Výsledky analýzy

$\psi = \{ \varnothing \}$

Množina potenciálních znalostí útočníka neobsahuje žádné důvěrné informace.

Nalezené útoky

Nástroj nenalezl žádné útoky.

Nenalezené útoky

Nebyly publikovány žádné útoky.

6.1.20 Paulson's strengthened version of Yahalom

Funkce protokolu

Vzájemná autentizace a distribuce relačního klíče s použitím symetrických klíčů a důvěryhodného serveru. Opravená verze protokolu Yahalom.

Formální specifikace protokolu

1. A → B : A, Na
2. B → S : B, Nb, { A, Na }$_{Kb}$
3. S → A : Nb, { B, Na, Kab }$_{Ka}$, { A, B, Nb, Kab }$_{Kb}$
4. A → B : { A, B, Nb, Kab }$_{Kb}$, { Nb }$_{Kab}$
5. A → B : { MSG }$_{Kab}$

Výsledky analýzy

ψ = { (a2$_{ij}$, b4$_{ij}$) }

Množina potenciálních znalostí útočníka neobsahuje žádné důvěrné informace.

Nalezené útoky

Porušení ($a2_{ij}$, $b4_{ij}$) nepředstavuje proveditelný útok.

Nenalezené útoky

Nebyly publikovány žádné útoky.

6.2 Srovnání protokolů

Níže uvedená tabulka 4 obsahuje seznam analyzovaných protokolů, přičemž u každého z nich je pro srovnání uveden typ protokolu, počet komunikujících subjektů, počet pravidel a počet nalezených (publikovaných i nepublikovaných) útoků.

Jak již bylo zmíněno, publikované útoky založené na kompromitovaném klíči starší relace a útoky založené na předpokladu, že operace šifrování je komutativní, nebyly uvažovány a proto v tabulce nejsou zahrnuty.

NÁZEV	T	S	R	PÚ	NÚ		
					P	N	Σ
BAN concrete Andrew Secure RPC	S	2	4	1	0	0	0
Lowe modified BAN concrete Andrew Secure RPC	S	2	4	0	0	0	0
Denning-Sacco shared key	S	3	3	1	0	0	0
Lowe modified Denning-Sacco shared key	S	3	5	0	0	0	0
Encrypted Key Exchange	S/A	2	5	1	0	0	0
Kao Chow Authentication v.1	S	3	4	0	0	0	0
Kao Chow Authentication v.2	S	3	4	0	0	0	0
Needham-Schroeder Public Key	A	3	7	1	0	0	0
Lowe's fixed version of Needham-Schroeder PK	A	3	7	0	0	0	0
Needham-Schroeder SK	S	3	5	0	0	0	0
Lowe modified Wide Mouthed Frog	S	3	4	0	0	0	0
Woo and Lam Mutual Authentication	S	3	7	1	1	0	1
Woo and Lam Pi	S	3	5	2	1	1	2
Woo and Lam Pi 1	S	3	5	1	1	0	1
Woo and Lam Pi 2	S	3	5	1	1	1	2
Woo and Lam Pi 3	S	3	5	1	1	1	2
Yahalom	S	3	4	1	0	0	0
BAN simplified version of Yahalom	S	3	4	2	0	0	0
Lowe's modified version of Yahalom	S	3	5	0	0	0	0
Paulson's strengthened version of Yahalom	S	3	4	0	0	0	0

Tab. 4: Srovnání protokolů (T – typ: S – symetrický, A – asymetrický; S – počet subjektů; R – počet pravidel; PÚ – počet publikovaných útoků; NÚ – počet nalezených útoků: P – publikovaných, N – nepublikovaných)

7 Závěr

Cílem diplomové práce bylo prostudování a zpracování materiálů z oblasti verifikace bezpečnostních protokolů, zejména pak studium vybraných nástrojů pro automatickou verifikaci a praktická implementace a ověření bezpečnostních protokolů v jednom z nich.

V úvodní části práce byly uvedeny některé základní pojmy z oblasti síťové bezpečnosti a příklady bezpečnostních protokolů včetně jejich formálních specifikací a proveditelných útoků. Dále byly stručně popsány základní přístupy analýzy protokolů. Stěžejní část práce tvoří kapitola, ve které byly představeny a detailně popsány čtyři nástroje pro automatickou formální verifikaci. Praktická část práce obsahuje výsledky analýzy dvaceti vybraných bezpečnostních protokolů nástrojem LySatool. Databáze specifikací těchto protokolů je k dispozici na přiloženém CD. Větší rozsah práce je dán jejím charakterem a požadavkem na prezentaci praktických výsledků přímo v práci.

Ve srovnávací tabulce protokolů v kapitole 6.2 je dobře viditelné, jakých výsledků bylo v praktické části dosaženo. Nástroj LySatool objevil téměř 50% publikovaných útoků a v protokolech Woo and Lam Pi objevil i útoky, které pravděpodobně dosud publikovány nebyly.

Při analýze se ukázalo, že největší nevýhodou nástroje je fakt, že v případě nalezení chyby v protokolu nedokáže vygenerovat popis útoku. Komponenta chyb ve výsledcích analýzy také často obsahuje velké množství porušení, která jsou pouze výsledkem použité techniky aproximace a ve skutečnosti chybu nepředstavují. Analýza je tak velmi pracná a časově náročná. Navíc je nutná detailní znalost protokolu a určité zkušenosti s analýzou. Velmi snadno totiž může dojít k chybné interpretaci výsledků.

Hlavním přínosem teoretické části práce je ucelený a detailní popis verifikačních nástrojů. Zejména informace o nástroji LySa / LySatool pochází z několika různých technických zpráv v anglickém jazyce. V současné době existuje mnoho nástrojů pro verifikaci protokolů, ovšem co se týče jejich vzájemného srovnání, mnoho informací zatím publikováno nebylo. Důvodem je pravděpodobně fakt, že případné experimentální výsledky každého nástroje se vztahují k jiné množině testovaných protokolů. Přímé srovnání by bylo možné pouze v případě shodné databáze. Zde bych viděl hlavní přínos praktické části práce a zároveň možnost dalšího pokračování. Převážná většina ověřovaných protokolů pochází z dobře zpracované a dostupné databáze SPORE (Security Protocols Open Repository) [9], kterou by bylo možné použít jako referenční a obsažené protokoly verifikovat dalšími nástroji.

Literatura

[1] Očenášek, P.: Verifikace bezpečnostních protokolů, diplomová práce, Brno, CZ, FIT VUT, 2003.

[2] Ptáček, M.: Specifikační jazyky a nástroje pro analýzu a verifikaci bezpečnostních protokolů, diplomová práce, Brno, CZ, FIT VUT, 2006.

[3] Stallings, W.: Cryptography and Network Security: Principles and Practice, Second Edition, New Jersey, Prentice-Hall, 1999.

[4] IT University of Copenhagen: Verification of Protocols for Security and Mobility [online] <http://www.first.dk/VPSM> [cit. 2008-03-20]

[5] Bodei, C., aj.: Automatic Validation of Protocol Narration [online] <http://www.di.unipi.it/~chiara/publ-40/BBDNN03.ps> [cit. 2008-03-20]

[6] Cheminod, M., aj.: Experimental Comparison of Automatic Tools for the Formal Analysis of Cryptographic Protocols [online] <http://csdl.computer.org/dl/proceedings/depcos-relcomex/2007/2850/00/28500153.pdf> [cit. 2008-03-20]

[7] Očenášek, P.: On Inductive Approach in Security Protocol Verification [online] <http://www.feec.vutbr.cz/EEICT/2004/sbornik/03-Doktorske_projekty/07-Informacni_systemy/08-xocena01.pdf> [cit. 2008-03-20]

[8] Matoušek, P.: Tools for Verification of Security Protocols <http://www.fit.vutbr.cz/~matousp/doc/2006/sec_protocols.pdf> [cit. 2008-03-20]

[9] Laboratoire Spécification et Vérification: Security Protocols Open Repository [online] <http://www.lsv.ens-cachan.fr/spore/index.html> [cit. 2008-03-20]

[10] Design and Analysis of Security Protocols [online] <http://www.cs.utexas.edu/~shmat/courses/cs395t_fall04/cs395t_home.html> [cit. 2008-03-20]

[11] Allen, M.: CTDP Security Reference Manual Version 0.3.0 [online] <http://www.comptechdoc.org/independent/security/guide/index.html> [cit. 2008-03-20]

[12] Paulson, C. L.: Proving Security Protocols Correct [online] <http://www.cl.cam.ac.uk/~lp15/papers/Auth/lics.pdf> [cit. 2008-03-20]

[13] Perrig, A.: SPINS: security protocols for sensor networks [online] <http://portal.acm.org/ft_gateway.cfm?id=582464&type=pdf&coll=ACM&dl=ACM&CFID=9292230&CFTOKEN=17163698> [cit. 2008-03-20]

[14] Cervesato, I.: Security Protocol Specification Languages [online] <http://www.qatar.cmu.edu/iliano/slides/fosad01.ppt> [cit. 2008-03-20]

[15] Gao, H., Nielson, H. R.: Analysis of LYSA-calculus with explicit confidentiality annotations [online] <http://csdl.computer.org/dl/proceedings/aina/2006/2466/02/246620039.pdf> [cit. 2008-03-20]

[16] Bodei, C., aj.: Static Validation Of Security Protocols [online] <http://www2.imm.dtu.dk/pubdb/views/edoc_download.php/3199/pdf/imm3199.pdf> [cit. 2008-03-20]

[17] Andersen, E. H., Nielsen, C. R.: Static Validation of Voting Protocols [online] <http://www2.imm.dtu.dk/pubdb/views/edoc_download.php/3968/pdf/imm3968.pdf> [cit. 2008-03-20]

[18] Bugliesi, M., aj.: Compositional Analysis of Authentication Protocols [online] <http://citeseer.ist.psu.edu/rd/71259186%2C665724%2C1%2C0.25%2CDownload/http%3AqSqqSqwww.dsi.unive.itqSq~maffeiqSqpublicationsqSqBFMesop04.ps.gz> [cit. 2008-03-20]

[19] Bugliesi, M., aj.: Authenticity by Tagging and Typing [online] <http://portal.acm.org/ft_gateway.cfm?id=1029135&type=pdf&coll=GUIDE&dl=GUIDE&CFID=21036721&CFTOKEN=10470458> [cit. 2008-03-20]

[20] Backes, M., aj.: A Calculus of Challenges and Responses [online] <http://portal.acm.org/ft_gateway.cfm?id=1314444&type=pdf&coll=&dl=ACM&CFID=21042076&CFTOKEN=81792235> [cit. 2008-03-20]

[21] Buchholtz, M., aj.: A Calculus for Control Flow Analysis of Security Protocols [online] <http://www2.imm.dtu.dk/pubdb/views/edoc_download.php/2841/pdf/imm2841.pdf> [cit. 2008-03-20]

[22] Hansen, M., aj.: Using Static Analysis to Validate the SAML Single Sign-on Protocol [online] <http://portal.acm.org/ft_gateway.cfm?id=1045409&type=pdf&coll=GUIDE&dl=GUIDE&CFID=21042763&CFTOKEN=35304208> [cit. 2008-03-20]

[23] Paulson, L. C.: Seven Years of Verifying Security Protocols [online] <http://www.cl.cam.ac.uk/~lp15/papers/Auth/dagstuhl2003-slides.pdf> [cit. 2008-03-20]

[24] von Oheimb, D.: The AVISPA Library [online] <http://david.von-oheimb.de/cs/talks/AVISPA_Library.pdf> [cit. 2008-03-20]

[25] The AVISPA Library of Protocols [online] <http://www.avispa-project.org/library/> [cit. 2008-03-20]

[26] Paulson, C. L.: The Inductive Approach to Verifying Protocols [online] <http://www.cl.cam.ac.uk/~lp15/papers/Auth/jcs.pdf> [cit. 2008-03-20]

[27] Lowe, G.: An Attack on the Needham-Schroeder Public-Key Authentication Protocol [online] <http://web.comlab.ox.ac.uk/oucl/work/gavin.lowe/Security/Papers/NSPKP.ps> [cit. 2008-03-20]

[28] Relations Between Secrets: Two Formal Analyses of the Yahalom Protocol [online] <http://www.cl.cam.ac.uk/~lp15/papers/Auth/yahalom.pdf> [cit. 2008-03-20]

[29] Syverson, P.: A Taxonomy of Replay Attacks [online] <http://chacs.nrl.navy.mil/publications/CHACS/1994syverson-foundations.pdf> [cit. 2008-03-20]

[30] Mitchel, J.: Protocol Verification by the Inductive Method [online] <http://www.stanford.edu/class/cs259/WWW04/lectures/08-Inductive%20Method.ppt> [cit. 2008-03-20]

[31] Boreale, M., Gorla, D.: Process Calculi and the Verification of Security Protocols [online] <http://www.dsi.uniroma1.it/~gorla/papers/BG-JTIT02.pdf> [cit. 2008-03-20]

[32] Wikipedia: Pi-calculus [online] <http://en.wikipedia.org/wiki/Pi_calculus> [cit. 2008-03-20]

[33] Dolev, D., Yao, A. C.: On the Security of Public Key Protocols [online] <ftp://reports.stanford.edu/pub/cstr/reports/cs/tr/81/854/CS-TR-81-854.pdf> [cit. 2008-03-20]

[34] Buchholtz, M.: Automated Analysis of Infinite Scenarios [online] <http://www2.imm.dtu.dk/cs_LySa/buchholtz05.pdf> [cit. 2008-03-20]

[35] Bauer, R. K., aj.: A Key Distribution Protocol Using Event Markers [online] <http://portal.acm.org/ft_gateway.cfm?id=357373&type=pdf&coll=&dl=GUIDE&CFID=57058036&CFTOKEN=42396053> [cit. 2008-03-20]

[36] Wikipedia: Horn clause [online] <http://en.wikipedia.org/wiki/Horn_clause> [cit. 2008-03-20]

[37] Buchholtz, M.: User's Guide for the LySatool version 2.01 [online] <http://www2.imm.dtu.dk/cs_LySa/lysatool/lysatool-2.01.pdf> [cit. 2008-03-20]

[38] Sun, H.: User's Guide for the Succinct Solver (V2.0) [online] <http://www.imm.dtu.dk/cs_SuccinctSolver/userGuide.pdf> [cit. 2008-03-20]

[39] Nielson, F.: A Succinct Solver for ALFP [online] <http://citeseer.ist.psu.edu/rd/55202534%2C591161%2C1%2C0.25%2CDownload/http://citeseer.ist.psu.edu/cache/papers/cs/29140/http:zSzzSzwww.informatik.uni-trier.dezSz~seidlzSzpaperszSzsuccinct.pdf/nielson02succinct.pdf> [cit. 2008-03-20]

[40] Buchholtz, M., aj.: For-LySa: UML for Authentication Analysis <http://www2.imm.dtu.dk/cs_LySa/bmps04.pdf> [cit. 2008-03-20]

[41] Wikipedia: Unified Modeling Language [online] <http://en.wikipedia.org/wiki/Unified_Modeling_Language> [cit. 2008-03-20]

[42] O'Shea, N.: Elyjah: A security analyzer for Java implementations of communications protocols <http://homepages.inf.ed.ac.uk/s0237477/report.pdf> [cit. 2008-03-20]

[43] Clark, J., Jacob, J.: A Survey of Authentication Protocol Literature: Version 1.0 [online] <http://www.cs.york.ac.uk/~jac/papers/drareview.ps.gz> [cit. 2008-03-20]

[44] Song, D., aj.: Athena: a Novel Approach to Efficient Automatic Security Protocol Analysis [online] <http://www.cs.berkeley.edu/~dawnsong/papers/athena-jcs.pdf> [cit. 2008-04-08]

[45] Song, D.: Athena: a New Efficient Automatic Checker for Security Protocol Analysis [online] <http://csdl.computer.org/dl/proceedings/csfw/1999/0201/00/02010192.pdf> [cit. 2008-05-15]

[46] Fábrega, F. J. T., aj.: Strand Spaces: Why is a Security Protocol Correct? [online] <http://www.mitre.org/work/tech_papers/tech_papers_00/guttman_strands/guttman_strands.pdf> [cit. 2008-04-08]

[47] Viganò, L.: Automated Security Protocol Analysis With the AVISPA Tool [online] <http://www.avispa-project.org/papers/avispa-mfps21.pdf> [cit. 2008-04-09]

[48] AVISPA v1.1 User Manual [online] <http://www.avispa-project.org/package/user-manual.pdf> [cit. 2008-04-10]

[49] Basin, D., aj.: OFMC: A Symbolic Model Checker for Security Protocols [online] <http://www.avispa-project.org/papers/ofmc-jis05.pdf> [cit. 2008-04-10]

[50] Turuani, M.: The CL-AtSe Protocol Analyzer [online] <http://hal.inria.fr/docs/00/10/35/73/PDF/RTA06_16_Turuani.pdf> [cit. 2008-04-10]

[51] Armando, A., Compagna, L.: SATMC: a SAT-based Model Checker for Security Protocols [online] <http://www.avispa-project.org/papers/satmc-sd-jelia04.ps> [cit. 2008-04-10]

[52] Wikipedia: Boolean Satisfiability Problem [online] <http://en.wikipedia.org/wiki/Boolean_satisfiability_problem> [cit. 2008-04-10]

[53] Boichut, Y., aj.: Automatic Verification of Security Protocols Using Approximations [online] <http://hal.inria.fr/docs/00/07/02/91/PDF/RR-5727.pdf> [cit. 2008-04-10]

[54] Genet, T., Klay, F.: Rewriting for Cryptographic Protocol Verification [online] <ftp://ftp.inria.fr/INRIA/publication/publi-pdf/RR/RR-3921.pdf> [cit. 2008-04-10]

[55] Boreale, M.: Symbolic Trace Analysis of Cryptographic Protocols [online] <http://www.dsi.unifi.it/~boreale/symbspi.pdf> [cit. 2008-04-16]

[56] Boreale, M., Buscemi, M. G.: Experimenting with STA, a Tool for Automatic Analysis of Security Protocols [online] <http://www.dsi.unifi.it/~boreale/STAKer.pdf> [cit. 2008-04-16]

[57] Boreale, M., Buscemi, M. G.: A Framework for the Analysis of Security Protocol [online] <http://www.dsi.unifi.it/~boreale/FrameConcur.ps.gz> [cit. 2008-04-16]

[58] Lowe, G.: Some New Attacks upon Security Protocols [online] <http://citeseer.ist.psu.edu/rd/17390356%2C16582%2C1%2C0.25%2CDownload/http://citeseer.ist.psu.edu/cache/papers/cs/908/http:zSzzSzwww.mcs.le.ac.ukzSz~gl7zSzSecurityzSzPaperszSzattacks.pdf/lowe96some.pdf> [cit. 2008-05-11]

[59] Kao, I.-Lung, Chow R.: An Efficient and Secure Authentication Protocol Using Uncertified Keys [online] <http://portal.acm.org/ft_gateway.cfm?id=206832&type=pdf&coll=portal&dl=ACM&CFID=27631250&CFTOKEN=17932348> [cit. 2008-05-13]

[60] Lowe, G.: A Family of Attacks upon Authentication Protocols [online] <http://citeseer.ist.psu.edu/rd/5307030%2C149578%2C1%2C0.25%2CDownload/http://citeseer.ist.psu.edu/cache/papers/cs/908/http:zSzzSzwww.mcs.le.ac.ukzSz~gl7zSzSecurityzSzPaperszSzmultiplicityTR.pdf/lowe97family.pdf> [cit. 2008-05-14]

[61] Denning, D. E., Sacco, G. M.: Timestamps in Key Distribution Protocols [online] <http://portal.acm.org/ft_gateway.cfm?id=358740&type=pdf&coll=GUIDE&dl=GUIDE&CFID=68192273&CFTOKEN=69988211> [cit. 2008-05-15]

Seznam příloh

Příloha 1. LySa specifikace protokolu Otway-Rees v ASCII formátu

```
let X subset NATURAL3 in
( new_{i in X} Ka_{i} )
( new_{j in X} Kb_{j} ) (

   /* Initiators – A_{i} */
   ( |_{i in X} |_{j in  X union ZERO} !( new Na_{i, j} )
      < A_{i}, B_{j}, M_{i, j}, A_{i}, B_{j},
      { A_{i}, B_{j}, M_{i, j}, Na_{i, j} } : Ka_{i}
      [at a1_{i, j} dest {s1_{i, j}}] >.
      ( B_{j}, A_{i}, M_{i, j}; x1_{i, j} ).
      decrypt x1_{i, j} as { Na_{i, j}; xk_{i, j} } : Ka_{i}
      [at a2_{i, j} orig {s3_{i, j}}] in
      ( B_{j}, A_{i}; x2_{i, j} ).
      decrypt x2_{i, j} as { ; xmsg_{i, j} } : xk_{i, j}
      [at a3_{i, j} orig {b3_{i, j}}] in 0
   )

   |

   /* Responders – B_{j} */
   ( |_{j in X} |_{i in X union ZERO} !( new Nb_{i, j} )
      ( A_{i}, B_{j}, M_{i, j}, A_{i}, B_{j}; y1_{i, j} ).
      < B_{j}, S, M_{i, j}, A_{i}, B_{j}, y1_{i, j},
      { A_{i}, B_{j}, M_{i, j}, Nb_{i, j} } : Kb_{j}
      [at b1_{i, j} dest {s2_{i, j}}] >.
      ( S, B_{j}, M_{i, j}; y2_{i, j}, y3_{i, j} ).
      decrypt y3_{i, j} as { Nb_{i, j}; yk_{i, j} } : Kb_{j}
      [at b2_{i, j} orig {s4_{i, j}}] in
      < B_{j}, A_{i}, M_{i, j}, y2_{i, j} >.
      ( new MSG_{i, j} )
      < B_{j}, A_{i}, { MSG_{i, j} } : yk_{i, j}
      [at b3_{i, j} dest {a3_{i, j}}] >.0
   )
```

```
  |

  /* Server - S */
  ( |_{i in X union ZERO} |_{j in X union ZERO} !
    ( B_{j}, S, M_{i, j}, A_{i}, B_{j}; z1_{i, j}, z2_{i, j} ).
    decrypt z1_{i, j} as
    { A_{i}, B_{j}, M_{i, j}; zNa_{i, j} } : Ka_{i}
    [at s1_{i, j} orig {a1_{i, j}}] in
    decrypt z2_{i, j} as
    { A_{i}, B_{j}, M_{i, j}; zNb_{i, j} } : Kb_{j}
    [at s2_{i, j} orig {b1_{i, j}}] in
    ( new K_{i, j} )
    < S, B_{j}, M_{i, j}, { zNa_{i, j}, K_{i, j} } : Ka_{i}
    [at s3_{i, j} dest {a2_{i, j}}],
    { zNb_{i, j}, K_{i, j} } : Kb_{j}
    [at s4_{i, j} dest {b2_{i, j}}]>.0
  )

)
```

Příloha 2. Specifikace protokolu EKE v HLPSL [25]

```
role eke_Init (A,B: agent,
               Kab: symmetric_key,
               Snd,Rcv: channel(dy))
played_by A
def=
  local State   : nat,
        Ea      : public_key,
        Na,Nb,K : text
  const sec_k1 : protocol_id
  init  State := 0

  transition
   1. State = 0
      /\ Rcv(start)
      =|>
      State' := 1
      /\ Ea' := new()
      /\ Snd({Ea'}_Kab)
   2. State = 1
      /\ Rcv({{K'}_Ea}_Kab)
      =|>
      State' := 2
      /\ Na' := new()
      /\ Snd({Na'}_K')
      /\ secret(K',sec_k1,{A,B})
      /\ witness(A,B,na,Na')
   3. State = 2
      /\ Rcv({Na.Nb'}_K)
      =|>
      State' := 3
      /\ Snd({Nb'}_K)
      /\ request(A,B,nb,Nb')
end role
```

```
role eke_Resp (B,A: agent,
               Kab: symmetric_key,
               Snd,Rcv: channel(dy))
played_by B
def=
  local State   : nat,
        Na,Nb,K : text,
        Ea      : public_key
  const sec_k2 : protocol_id
  init  State := 0

  transition
   1. State = 0 /\ Rcv({Ea'}_Kab)
      =|>
      State' := 1
      /\ K' := new()
      /\ Snd({{K'}_Ea'}_Kab)
      /\ secret(K',sec_k2,{A,B})
   2. State = 1 /\ Rcv({Na'}_K)
      =|>
      State' := 2
      /\ Nb' := new()
      /\ Snd({Na'.Nb'}_K)
      /\ witness(B,A,nb,Nb')
   3. State = 2
      /\ Rcv({Nb}_K)
      =|>
      State' := 3
      /\ request(B,A,na,Na)
end role
```

```
role session(A,B: agent,
             Kab: symmetric_key)
def=
  local SA, RA, SB, RB: channel (dy)
  composition
     eke_Init(A,B,Kab,SA,RA)
  /\ eke_Resp(B,A,Kab,SB,RB)
end role

role environment()
def=
  const a, b   : agent,
        kab    : symmetric_key,
        na, nb : protocol_id
  intruder_knowledge={a,b}
  composition
      session(a,b,kab)
   /\ session(b,a,kab)
end role

goal
 % Confidentiality (G12)
 secrecy_of sec_k1, sec_k2

 % Message authentication (G2)
 % EKE_Init authenticates EKE_Resp on nb
 authentication_on nb

 % Message authentication (G2)
 % EKE_Resp authenticates EKE_Init on na
 authentication_on na
end goal
```