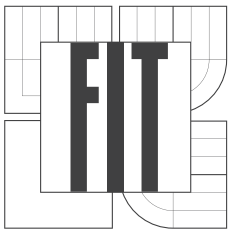
Vysoké učení technické v Brně

Fakulta informačních technologií



Verifikace bezpečnostních protokolů

Diplomová práce

Pavel Očenášek **2003**

Verifikace bezpečnostních protokolů

autor

Pavel Očenášek

Fakulta informačních technologií
Vysoké učení technické v Brně

20. června 2003

Autor prohlašuje, že tuto diplomovou práci vypracoval samostatně pod vedením vedoucího projektu a že uvedl všechny literární prameny a publikace ze kterých čerpal.

Autor tímto dává Fakultě informačních technologií Vysokého učení technického v Brně svolení k použití a úpravám textu i jeho částí pro vědecké a výukové účely při zachování autorských práv. Současně souhlasí s tím, že programový kód, který byl vytvořen jako součást této práce je možné používat v souladu s General Public Licence.

Podpis autora .....
20. června 2003

Podpis vedoucího práce .....
Daniel Cvrček
Fakulta informačních technologií
Vysoké učení technické v Brně

Abstrakt: Bezpečnostní protokoly používají šifrování pro vytvoření privátních komunikačních kanálů v nezabezpečené síti. Pro dokázání jejich bezpečnosti se používá různých přístupů. Při formálním dokazování jsou demonstrovány pokusy útočníka o prolomení protokolu a odposlechnutí komunikace. Diplomová práce se zabývá souhrnem a analýzou jednotlivých formálních metod používaných pro dokazování těchto protokolů. Při demonstraci použití jednotlivých metod jsem se zaměřil na vyzdvižení specifických vlastností bezpečnostních protokolů.

Klíčová slova: komunikace, bezpečnostní protokol, verifikace, zabezpečení, autentizace, veřejný klíč, privátní klíč, Needham-Schroeder, BAN logika, theorem proving

Abstract: In this paper an overview of the state of the art in the application of formal methods of cryptographic protocol analysis is given. Formal methods can be used in various phases of the design, such as specification, construction and verification. Up to now, most of the work concentrated on the formal verification of protocols, and in particular, on the application of modal logics of knowledge and belief for modeling and analyzing protocols. Further approaches represent model checking and inductive methods. Using formal methods in the specification of cryptographic protocols is an emerging area of research, while formal methods for construction of protocols in a systematic way are just appearing.

Key Words: communication, security protocol, verification, protection, authentication, public key, private key, Needham-Schroeder, BAN logic, theorem proving

Obsah

1 Úvod

Počítačové sítě se začaly ve větší míře ve světě rozvíjet na počátku 70. let. Sítě se tehdy používaly převážně pro vědecké či vojenské účely. Uživatelé a správci těchto sítí spoléhali především na kolegialitu a poctivost všech účastníků, zabezpečení se věnovala jen malá pozornost. Neexistovaly ani žádné možnosti, jak by si správci systémů a programátoři mohli vyměňovat informace související s bezpečností.

Protokoly a služby, které v tomto období vznikly, se staly velmi oblíbenými a používají se dodnes. Za všechny lze uvést protokolovou sadu TCP/IP. Zabezpečení komunikace nebylo prakticky žádné, autentizace komunikujících stran probíhala pouze na základě hesla, které se však po síti posílalo nezabezpečené. Bezpečnostní protokoly byly teprve ve vývoji a jejich formální modely byly pro praxi nepoužitelné. K implementaci a nasazení těchto protokolů došlo až později.

V současné době již počítačové sítě prakticky pronikly do běžného života společnosti. Požadavky na zabezpečení elektronické komunikace se zvyšují, původní sada protokolů není dostačující. Vyvíjejí se a implementují nové, bezpečné modely protokolů. Důležitou součástí vývoje bezpečnostních protokolů je jejich formální verifikace. Jedná se vlastně o analýzu protokolu a jeho pravidel po formální stránce, kdy se zjišťuje, zda lze protokol prolomit / napadnout.

Diplomová práce se zabývá souhrnem a analýzou jednotlivých přístupů používaných k formální verifikaci. Vzhledem k tomu, že naprostá většina materiálů, které byly podkladem pro tuto práci, je napsána v anglickém jazyce a odborné termíny často nemají český ekvivalent, pokusil jsem se o přibližný překlad s tím, že za tímto českým termínem uvádím jeho anglický výraz.

2 Základní pojmy

Před vlastní přístupem k jednotlivým verifikačním technikám je nutné vymezit některé základní pojmy.

AUTENTIZACE

Autentizace je proces, při kterém účastník nějakého distribuovaného systému prokazuje svoji identitu. Pokud je v komunikačním protokolu zařazen kvalitní autentizační mechanismus, je pro útočníka velmi obtížné podvrhnout zprávu s falešným původem.

Systém umožňuje silnou autentizaci, pokud je splněna následující podmínka: pokud subjekt A obdrží zprávu, ve které je jako odesílatel uveden subjekt B, pak B odeslal právě tuto zprávu subjektu A.

Pro mnoho aplikací musí být uvedený přístup zjednodušen, vzhledem k parametrům a kvalitě komunikačních kanálů. Systém pak umožňuje slabou autentizaci, pokud je splněna podmínka: pokud subjekt A obdrží zprávu a jako odesílatel je uveden subjekt B, pak buď B poslal právě tuto zprávu subjektu A nebo B uvedenou skutečnost popře.

AUTENTIZAČNÍ PROTOKOL

Soubor pravidel, podle kterých účastníci komunikace zasílají zprávy tak, aby prokázali svoji identitu.

ANONYMITA

Systém pracující s několika subjekty je anonymní, pokud má následující vlastnost: pokud některý ze subjektů zašle zprávu, pak nikdo nemůže tento subjekt identifikovat. V některých systémech se však vyskytuje vlastnost, která pojem anonymity uvádí do jiné roviny. Jedná se o systémy, kdy je požadováno, aby každý subjekt poslal anonymní zprávu pouze jedenkrát. Jako fiktivní příklad můžeme uvést volební server, kdy uživatelé posílají anonymně volební hlasy, avšak žádný uživatel nemůže volit dvakrát.

DŮVĚRNOST

Vzájemně komunikující subjekty mohou požadovat, aby jejich komunikace nemohla být odposlouchávána žádnou jinou neoprávněnou stranou. Nejčastějším prostředkem pro dosažení důvěrnosti je kryptografie (další možností je např. steganografie).

KRYPTOGRAFIE

Jedná se o takové zabezpečení komunikace, kdy vnější útočník nemůže bez znalosti klíčů získat originální zprávy. Kryptografie využívá dvou symetrických procesů: šifrování a dešifrování. Šifrování lze rozdělit podle typu použitých algoritmů a klíčů na symetrické a asymetrické.

Při symetrickém šifrování se používá symetrických klíčů (tedy stejný klíč mají oba komunikující subjekty).

Asymetrické šifrování je založeno na použití dvou párových klíčů. Každý subjekt má svůj veřejný a privátní klíč. V tomto případě je možné nejen zprávy šifrovat, ale také podepisovat: odesílatel data nejdříve dešifruje svým soukromým klíčem a výsledek přidá ke zprávě. Všichni příjemci, kteří mají veřejný klíč odesílatele mohou podpis „zašifrovat“. Pokud získají doručenou zprávu, mají jistotu o jejím původci, protože nikdo kromě majitele veřejného klíče nemohl vytvořit odpovídající šifru.

Asymetrická kryptografie má oproti klasické, symetrické velkou výhodu v tom, že není nutné domlouvat tajný sdílený klíč před zahájením utajené komunikace. Na druhou stranu však hrozí podvržení veřejného klíče (lze volně šířit veřejné klíče s nepravou identitou). Tomu zamezují certifikační autority, které potvrzují pravost veřejných klíčů komunikujících subjektů.

INTEGRITA

Jedná se o nejčastěji požadovanou vlastnost při přenosu dat. Přenášená data nesmí být změněna žádnou neautorizovanou stranou, nesmí být ani uměle zadržována, či opakovaně vysílána po neoprávněném odposlechu. Pro kvalitní zabezpečení integrity dat je nutné použít kryptograficky silných algoritmů, jako jsou např. MD5 nebo SHA-1.

NONCE

Anglický termín „nonce“ je používán přímo ve formální specifikaci bezpečnostních protokolů. Někdy bývá překládán jako „keksík“. Jedná se o náhodně vygenerovaná čísla. Někdy bývají rozlišována jako odhadnutelná (sekvenčně generovaná čísla, časové razítko) nebo neodhadnutelná (např. náhodné 40-bitové číslo).

RELAČNÍ KLÍČ

Tímto pojmem se rozumí klíč jedné komunikační relace („sezení“), který byl ustaven na začátku komunikace za účelem jejího zabezpečení.

3 Síťové protokoly a systémy

Komunikace v síťovém prostředí probíhá v několika vrstvách, podle definice ISO/OSI, od fyzické až po aplikační. Tato kapitola se zabývá souhrnem nejpoužívanějších protokolů, které se v praxi vyskytují, a jejich bezpečnostních vlastností.

3.1 Internet Protocol (IP)

Běžně se používá v prostředí Internetu a patří k nejrozšířenějším protokolům síťové úrovně. Zajišťuje nespolehlivou, nespojovanou komunikaci, není tedy zajištěno doručení zprávy příjemci, ani pořadí jednotlivých zasílaných zpráv.

Základní komunikační jednotkou jsou pakety. Každý paket má vlastní hlavičku, která obsahuje mj. adresu odesílatele a příjemce (tzv. IP adresa). Cestu paketu k cíli určují směrovače (routery).

Podle své definice IPv4 z roku 1981 je bezpečnosti věnovaná jen malá pozornost. Napadení tohoto protokolu je jednoduché: data nejsou uvnitř paketů šifrována a lze jednoduše vyrobit paket s falešnou adresou odesílatele. Novější specifikace IPv6 již se základními zásadami bezpečnosti počítá.

3.2 Transmission Control Protocol (TCP)

Tento protokol je implementován ve vyšší vrstvě, nad protokolem IP. Poskytuje již spolehlivou a spojovanou komunikaci, s detekcí i korekcí chyb. Každý paket má označení pořadí v datovém toku.

Bezpečnostní slabiny TCP protokolu spočívají např. v možnosti „ukradnutí“ relace uživatele. Útočník například sleduje komunikaci mezi uživatelem a serverem a vyčká na autentizaci uživatele. Po dokončení autentizačního procesu převezme komunikaci místo uživatele a uživateli zašle například zprávu o odmítnutí služby. Útočník dále pokračuje v komunikaci se serverem v relaci napadeného uživatele.

3.3 Secure Sockets Layer (SSL)

Tento protokol se provozuje mezi TCP protokolem a aplikační vrstvou SSL. Poskytuje bezpečnou komunikaci mezi klientem a serverem, zajištění integrity, autentizaci serveru a volitelně i klienta. Tento protokol je otevřený a lze jej volně rozšiřovat o další algoritmy a metody.

Na počátku komunikace si klient a server dohodnou typ použitého algoritmu. Dále si ustanoví tajný klíč, který se bude používat pro zabezpečení komunikace. Server pak pošle klientovi své autentizační údaje. Volitelně lze požadovat i autentizaci klienta serveru. Pro autentizaci se používají certifikáty, které bývají vystavené nějakou certifikační autoritou.

3.4 HTTPS, S-HTTP

Pro zabezpečení služby WWW (World-Wide-Web) se používá nejčastěji protokolu HTTPS, což je v podstatě implementace protokolu HTTP nad SSL. Protokol HTTPS zajišťuje komplexní zabezpečení komunikace klienta se serverem. K dalším protokolům patří Secure HTTP (S-HTTP), který přímo rozšiřuje stávající funkce HTTP o bezpečnostní funkce. Tento protokol, na rozdíl od protokolu HTTPS, poskytuje zabezpečení na úrovni dokumentů (soukromé / veřejné dokumenty).

3.5 Kerberos

Nejedná se o jeden síťový protokol, ale o celý bezpečnostní systém. Vznikl v polovině 80. let na Massachusetts Institute of Technology jako součást projektu Athena. Cílem tohoto projektu bylo spojit do funkčního celku několik stovek počítačů propojených do sítě. Úkolem systému Kerberos bylo zajistit spolehlivou autentizaci, kdy všechny citlivé údaje budou uloženy na jednom bezpečném místě a nikdy se nevyskytnou na síti nechráněně.

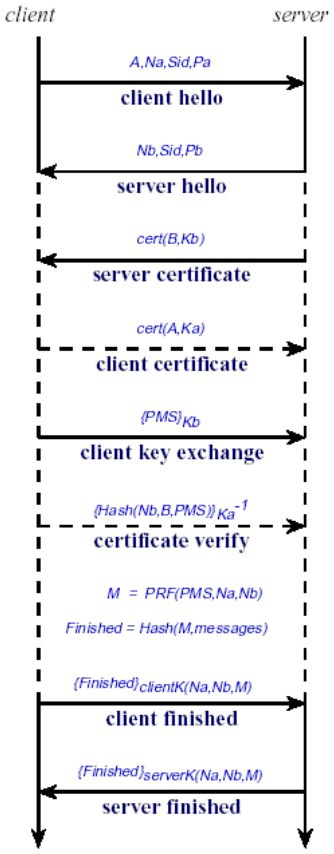
V roce 1991 byla vydána specifikace verze 5 v dokumentu RFC 1510, přičemž zde definovaný standard je poměrně obecný a otevřený a lze snadno dodefinovat další mechanismy, jako je např. použití asymetrické kryptografie.

Kerberos je systém s tzv. důvěryhodnou třetí stranou, založený na modelu Needham-Schroeder. Protože Kerberos používá pouze symetrickou kryptografii, musí znát tajné klíče všech svých klientů. Veškeré citlivé údaje o klientech jsou uloženy v databázi autentizačního serveru. V případě uživatele je klíčem heslo, pro aplikace se obvykle klíč generuje náhodně.

Myšlenka autentizace je jednoduchá: klient, který chce získat přístup k nějaké službě pošle nejdříve žádost autentizačnímu serveru. Server odešle tiket (prokazuje identitu klienta) pro službu a nově vytvořený klíč relace. Klient pak odešle získaný tiket příslušné službě, která jej rozšifruje a porovná, zda se obsažený identifikátor shoduje s proklamovanou identitou klienta.

4 Formální popis protokolů

Jak již bylo uvedeno, pro zabezpečení elektronické komunikace se používají bezpečnostní protokoly. Pojmem „protokol“ je myšlen sled interakcí mezi subjekty (entitami).



Obr. 1: Příklad bezpečnostního protokolu

Aby bylo možné takový protokol pro zabezpečení použít, musí obstát při pokusu o jeho prolomení vnějšími útočníky, kteří se snaží komunikaci odposlechnout nebo jinak napadnout. Obecná zabezpečená komunikace probíhá tak, že se zasílají zprávy šifrované dočasnými klíči, které nejsou známé vnějším pozorovatelům spojení (útočníkům). Zdálo by se, že takto je vše v pořádku a komunikace probíhá v šifrované formě. Mnoho protokolů však obsahuje chyby a jsou popsány metody prolomení definovaných bezpečnostních vlastností protokolů. Samotné tvrzení autorů, že protokol je bezpečný není dostačující. Jednou z možností, jak toto tvrzení důvěryhodným způsobem prokázat, případně nalézt chyby, je použití některých formálních postupů, jak bude popsáno dále.

4.1 Zpráva

Základním prvkem komunikace mezi účastníky A, B je zpráva. Při formálním zápisu zpráv bereme v úvahu její následující charakteristiky:

- označení účastníků A, B, … ;
- nonces Na, Nb, … ;
- klíče Ka, Kb, Kab, … ;
- složená zpráva { X, X′ },
- hašovaná zpráva **Hash** X,
- šifrovaná zpráva klíčem K **Crypt** K X , někdy také označené $\{ X \}_K$.

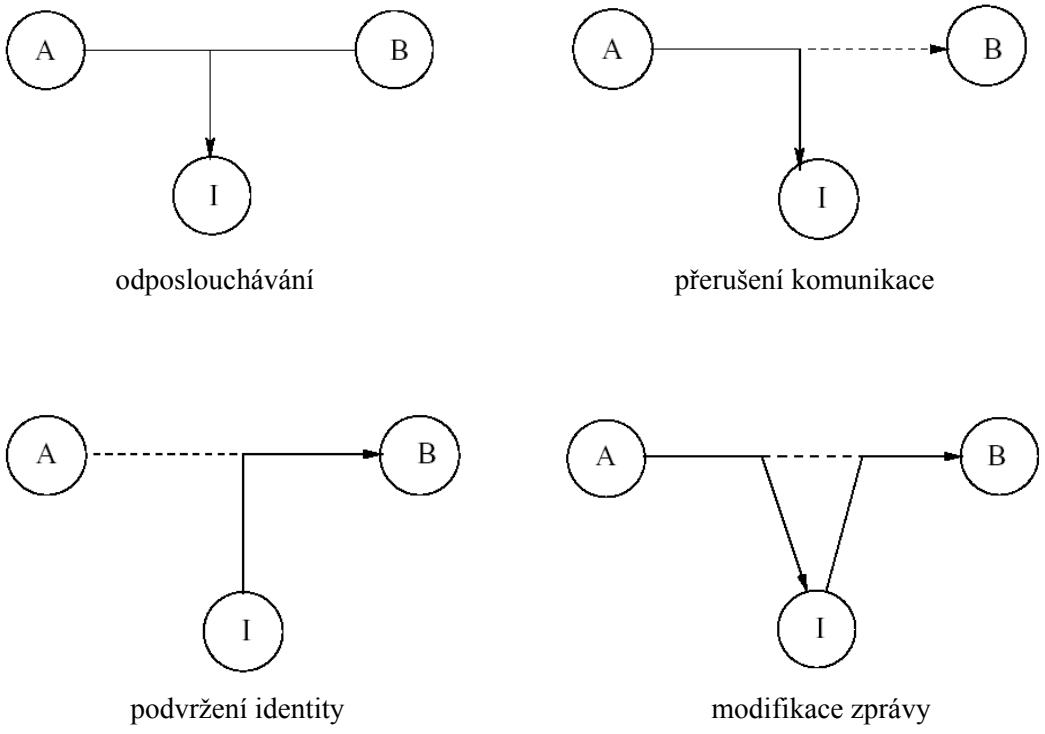
Při použití šifrování metodou veřejného klíče předpokládáme, že K^{-1} je inverzní ke klíči K. V případě rovnosti $K^{-1} = K$ se jedná o symetrické klíče. Dále předpokládáme $(K^{-1})^{-1} = K$ pro všechny klíče K. Předpokládáme, že šifrovaná zpráva nemůže být nikdy přečtena bez znalosti příslušných klíčů.

4.2 Útok na zprávu

Bezpečnostní protokol musí odolat všem formám pokusů o jeho prolomení. Aktivní útočník, tedy ten který může zasílat zprávy, je mnohem nebezpečnější než pasivní sledovatel komunikace. Vzhledem k tomu, že typů útoků je velké množství, mělo by být tvrzení že „protokol je bezpečný“ doplněno o poznámku, proti kterým typům útoku je tato ochrana zaručena.

4.2.1 Základní metody útoků

Možnosti útoku na komunikaci subjektů A, B; útočník je zde označen symbolem I (*intruder*):



Obr. 2: Možnosti útoků na komunikaci

Odposlouchávání (zadržení zprávy): situace, kdy se útočníkovi povede získat neautorizovaný přístup ke komunikaci. Do této kategorie patří také většina programů, které se snaží odposlechnout heslo, kterým se klient autentizuje ke vzdálenému serveru. Tato hesla se často posílají zcela nešifrována a jsou tedy poměrně lehce zjistitelná.

Přerušení komunikace: útočníkovi se podaří zničit či jinak znepřístupnit přenášenou zprávu. Příkladem může být přerušení komunikačního spoje.

Podvržení identity: útočník se může pokusit o vytvoření zprávy s falešnou identitou odesílatele.

Modifikace zprávy: útočník, který získá přenášenou zprávu, může pozměnit její obsah. Příjemce tak dostane zprávu, kterou sice původně vyslala autorizovaná strana, ale během přenosu došlo k nějakému zásahu do jejího obsahu.

4.2.2 Útok přehráváním

Tento typ útoku je založen na principu, že komunikace mezi zúčastněnými subjekty je odposlouchávána a útočník si tuto komunikaci ukládá za účelem její další analýzy. Poté co jsou zachycená data analyzována, mohou být použita pro podvrhnutí identity útočníka vůči některému ze subjektů. Vzdálený subjekt totiž nemá žádnou možnost ověřit aktuálnost zprávy (*freshness*).

Jako obrana proti tomuto typu útoku se používá časové razítkování zprávy, čímž se zabrání možnosti pozdějšího použití odposlechnuté informace.

4.2.3 Útok ze středu

Další z možností jak protokol napadnout je použití „útoku ze středu“ (*man-in-the-middle attack, middle-person attack*). V tomto případě existuje kromě dvou komunikujících stran A a B ještě aktivní útočník. Tento naváže komunikaci jak s A, tak s B, přičemž se vždy vydává za regulérního partnera. S využitím informací odposlechnutých od druhé strany se jeví pro A jako B a naopak. Útočník má tedy možnost monitorovat veškerý přenos.

4.3 Rozdělení protokolů

4.3.1 Symetrické protokoly

V tradiční kryptografii je základem aplikace symetrických šifrovacích algoritmů (např. DES), kdy se pro zašifrování a dešifrování zpráv používá stejného sdíleného klíče. Před odesláním je zpráva zašifrována pomocí klíče, který zná jak odesílatel, tak příjemce zprávy. Tato zpráva je pak příjemcem standardně dešifrována. Při uvedené komunikaci se často vyskytuje třetí subjekt – KDC (*Key Distribution Center*), který zajišťuje výměnu klíče mezi dvěma hlavními subjekty bezpečnou formou, obvykle po zvláštním zabezpečeném komunikačním kanále. Příkladem symetrického protokolu je Otway-Rees protokol.

4.3.2 Asymetrické protokoly

Alternativou je použití asymetrických protokolů, které jsou založeny na šifrovacích algoritmech používajících dvojici veřejný a privátní klíč. Privátní klíč reprezentuje identitu jeho vlastníka, veřejný klíč je volně k dispozici všem ostatním subjektům, kteří se na komunikaci podílejí.

Do této skupiny patří mnoho protokolů, například IKE (*Internet Key Exchange protocol*) používaný v IPv6 protokolu, SET (*Secure Electronic Transaction protocol*) používaný v bankovnictví společnostmi VISA a MasterCard, TMN protokol a dále také známý Needham-Schroeder autentizační protokol.

4.4 Příklady protokolů

V této kapitole budou popsány některé základní protokoly. Na těchto protokolech bude demonstrován formální zápis a také případně možnosti jejich napadení.

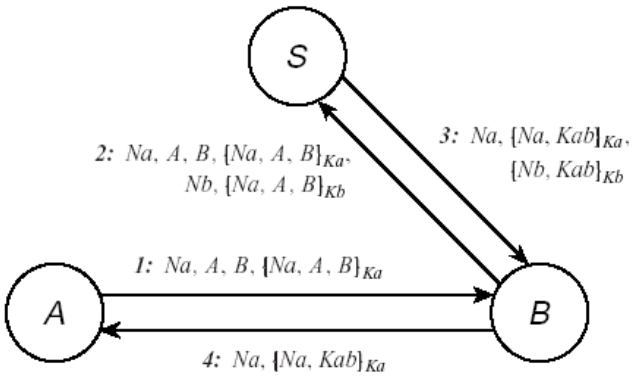
4.4.1 Otway-Rees protokol

Tento protokol je určen k tomu, aby mohl jakýkoliv uživatel vytvořit bezpečné spojení s kýmkoliv jiným. Předpokládejme pokus o navázání komunikace mezi subjekty A a B, přičemž S je důvěryhodný server (*trusted key server*).

Formální zápis:

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

Zde Ka znamená klíč uživatele A a Kb klíč uživatele B. $\{ X \}_K$ značí zprávu X šifrovanou klíčem K. Přičemž předpokládáme, že nikdo nemůže získat originál zprávy X nebo vytvořit novou šifrovanou zprávu $\{ X' \}_K$ bez znalosti klíče K.



Obr. 3: Neformální popis Otway-Rees protokolu (s chybou)

Popis komunikace:

1. A kontaktuje B, vytvoří nonce Na pro identifikaci spojení.
2. B kontaktuje S, vytvoří nonce Nb. Nemůže ale přečíst zprávu $\{ Na, A, B \}_{Ka}$ protože je zašifrována klíčem uživatele A, takže ji pouze přepošle S. Navíc také zašifruje přijatou zprávu svým klíčem a přepošle ji S.
3. Server S zná klíče všech zúčastněných stran, takže obě zprávy může dešifrovat. Vytvoří nový relační klíč Kab. Pro subjekt A zašifruje zprávu skládající se z Kab a Na klíčem Ka. Taková šifrovaná zpráva bývá často označována jako certifikát.

4. B převezme svůj certifikát. Zkontroluje zda přijatá nonce Nb je stejná jako jím odeslaná v kroku 2. Pokud toto souhlasí, pak přepošle druhý certifikát subjektu A, který podobně přijatý certifikát ověří.

Pak následuje šifrovaná komunikace s použitím klíče Kab jako relačního klíče.

Uvedený protokol lze však napadnout útočníkem C, a to z těchto důvodů:

- oba certifikáty zaslané v kroku 3 mají stejný formát
- v kroku 2 se nonce Nb posílá v nezabezpečené formě

Útok pak bude mít následující formu (C_A značí, že se subjekt C vydává za subjekt A apod.):

1. $A \rightarrow C_B : Na, A, B, \{ Na, A, B \}_{Ka}$

1′. $C \rightarrow A : Nc, C, A, \{ Nc, C, A \}_{Kc}$

Subjekt A se snaží kontaktovat B, ale C zachytí uvedenou zprávu a kontaktuje A

2′. $A \rightarrow C_S : Nc, C, A, \{ Nc, C, A \}_{Kc}, Na', \{ Nc, C, A \}_{Ka}$

2″. $C_A \rightarrow S : Nc, C, A, \{ Nc, C, A \}_{Kc}, Na, \{ Nc, C, A \}_{Ka}$

Subjekt A odpovídá na zprávu od C podle pravidel protokolu, snaží se kontaktovat S. Ale C modifikuje tuto zprávu a nahradí původní nonce Na′ novým nonce Na.

3′. $S \rightarrow C_A : Nc, \{ Nc, Kca \}_{Kc}, \{ Na, Kca \}_{Ka}$

Jako odpověď posílá S subjektu A relační klíč, který se pak bude sdílet mezi A a C.

4. $C_B \rightarrow A : Na, \{ Na, Kca \}_{Ka}$

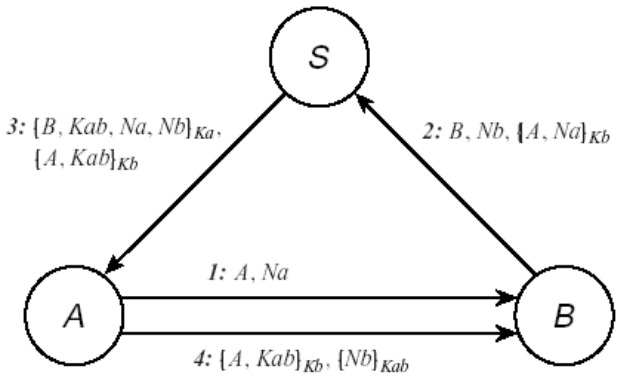
Útočník C získává oba certifikáty a posílá subjektu A ten špatný. Tento obsahuje Na, tedy A akceptuje Kca jako relační klíč zaslaný od B. Ale tento klíč je již sdílený s C.

4.4.2 Yahalom protokol

Protokol má následující specifikaci:

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

Trojúhelníková konfigurace předurčuje subjektům A i B možnost získat informaci o tom, zda druhá strana je aktivní:



Obr. 4: Yahalom protokol (s chybou)

Protokol má následující chybu - certifikát subjektu B $\{ A, Kab \}_{Kb}$ neobsahuje žádné nonces, proto nepodává žádnou časovou informaci. Pokud bude K starým relačním klíčem, který již jednou byl sdílen subjekty A a B, pak pokud C nějakým způsobem získá K a uloží si starý certifikát $\{ A, K \}_{Kb}$ obsahující tento klíč, pak může provést následující konverzaci s B:

1. $C_A \rightarrow B : A, Nc$
2. $B \rightarrow C_S : B, Nb, \{ A, Nc \}_{Kb}$
4. $C_A \rightarrow B : \{ A, K \}_{Kb}, \{ Nb \}_K$

Nyní B přijme starý kompromitovaný relační klíč.

Podobně jako u Otway-Rees protokolu, správná verze protokolu zašifruje nonce Nb v kroku 2.

Poznámky k bezpečnosti obou protokolů:

- Otway-Rees protokol vyžaduje ochranu Nb před modifikací. Útok je proveden zevnitř systému. Oběť útoku získá klíč, který je bohužel sdílen s nesprávnou osobou.
- Yahalom vyžaduje zašifrování Nb. Útok je jednoduchý a může být proveden zvenčí. Oběť útoku získá klíč který je později zkompromitován.

4.4.3 Needham-Schroeder protokol

Tento protokol patří obecně mezi asymetrické autentizační protokoly. Je znám svojí jednoduchostí a vzhledem k tomu, že jsou na něm v této práci demonstrovány některé verifikační techniky, je zde popsán podrobněji, než ostatní protokoly.

Formální specifikace protokolu:

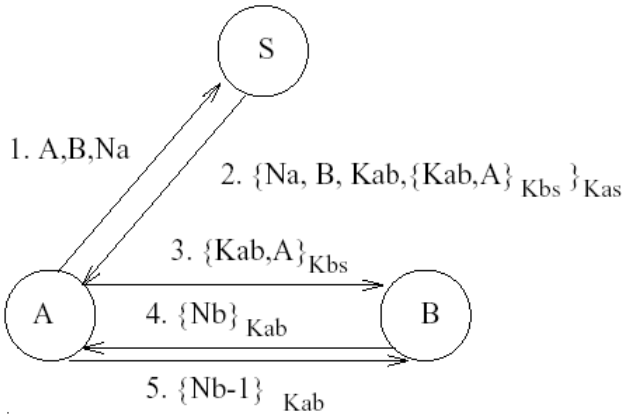
1. $A \rightarrow S$: A, B
2. $S \rightarrow A$: $\{ K_B \}_{Ks}^{-1}$
3. $A \rightarrow B$: $\{ Na, A \}_{Kb}$
4. $B \rightarrow S$: B, A
5. $S \rightarrow B$: $\{ Ka, A \}_{Ks}^{-1}$
6. $B \rightarrow A$: $\{ Na, Nb \}_{Ka}$
7. $A \rightarrow B$: $\{ Nb \}_{Kb}$

Symboly Ka, Kb, Ks jsou veřejné klíče subjektů A, B, S. Inverzní klíče jsou privátní klíče, které jsou známé pouze svým vlastníkům. Přepokládá se, že každý subjekt zná veřejný klíč Ks, tedy každý si může přečíst zprávu odeslanou serverem S a zašifrovanou jeho privátním klíčem Ks^{-1}. V případě, že některý subjekt potřebuje veřejný klíč jiného subjektu, pošle svůj požadavek serveru S.

Kroky 1,2,4,5 se týkají zjištění veřejných klíčů, přičemž v 3,6,7 se pak provádí autentizace A a B. V kroku 3 posílá subjekt A svůj nonce a svoji identitu zašifrovanou klíčem Kb. Protože pouze B vlastní privátní klíč Kb^{-1}, může A předpokládat, že takto zašifrovanou zprávu nepřečte nikdo jiný než B. Subjekt B pak vrací tento nonce spolu se svým Nb, zašifrované veřejným klíčem Ka.

V některých částech práce uvažujeme také jednodušší symetrickou variantu protokolu:

1. $A \rightarrow S$: A, B, Na
2. $S \rightarrow A$: $\{ Na, B, Kab, \{ Kab, A \}_{Kbs} \}_{Kas}$
3. $A \rightarrow B$: $\{ Kab, A \}_{Kbs}$
4. $B \rightarrow A$: $\{ Nb \}_{Kab}$
5. $A \rightarrow B$: $\{ Nb - 1 \}_{Kab}$.



Obr. 5: Needham-Schroederův autentizační protokol

Právě na této symetrické variantě bude níže vysvětlena chyba, která je v protokolu obsažena.

Prolomení protokolu:

První slabinu v tomto protokolu objevili Dennig a Sacco v roce 1981. Předpokládá se, že relační klíč je určen pro jedno použití a pak je zapomenut. Pokud nyní vezmeme v úvahu útočníka C, který sledoval celou komunikaci protokolu, pak je možný následující útok:

Pro ilustraci předpokládejme, že starý relační klíč byl kompromitován. Pokud C zaznamenal komunikaci protokolu, kdy byl ustaven uvedený (starý) relační klíč (označme jej CK), pak může C odpovědět zprávou:

$C \rightarrow B$: $\{ CK, A \}_{Kab}$

Subjekt B si myslí že A inicializovalo novou konverzaci a zasílá A zprávu:

$B \rightarrow A$: $\{ Nb \}_{CK}$

Útočník C však zachycuje uvedenou zprávu, dešifruje ji klíčem CK a napodobuje odpověď A .

$Z \rightarrow B$: $\{ Nb - 1 \}_{CK}$

Útočník takto zasílá subjektu B zprávu, která vypadá jako by pocházela od A . Subjekt B nemá žádnou možnost rozpoznat, že nekomunikuje s A ale s útočníkem.

Ošetření protokolu:

Denning a Sacco navrhují přidat do protokolu do zpráv 2 a 3 časová razítka (označena T), čímž by se problém vyřešil. Pak specifikace vypadala následovně:

$S \rightarrow A$: $\{ T, Na, B, Kab, \{ Kab, A, T \}_{Kab} , \}_{Kas}$,

$A \rightarrow B$: $\{ Kab, A, T \}_{Kbs}$.

Tedy správná odpověď na zprávu 3 by byla rozpoznána a neplatná (stará) odpověď by byla ignorována.

Další možnost, kterou navrhují Needham a Schroeder, je založena na použití nonces. Je faktem, že jedna z komunikujících stran bude v konverzaci vyžadovat kontrolu časové platnosti odpovědi. Touto stranou by měla být právě ta strana, která vygeneruje platný nonce.

Toho je dosaženo následujícím způsobem:

$A \rightarrow B$: A

$B \rightarrow A$: $\{ A, J \}_{Kbs}$, kde J je nonce, který bude uchován subjektem B .

Nyní J může být přiloženo ke zprávě zaslané A, která bude předána B. Tím se zajistí, že subjekt B si bude jistý časovou platností relačního klíče.

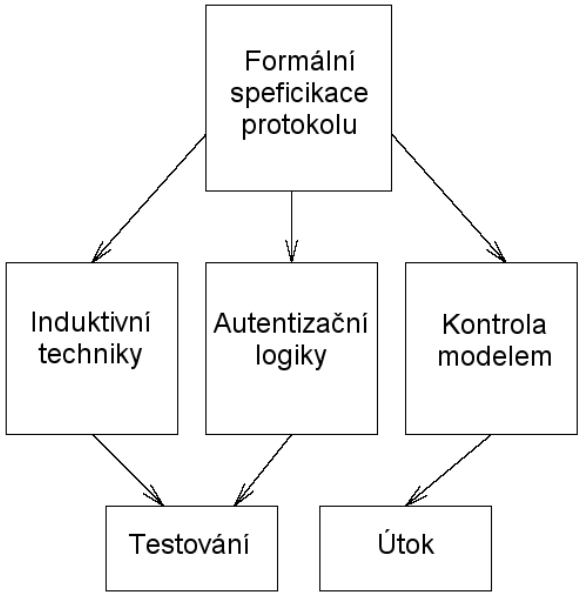
Chyba v Needham-Schroeder protokolu je založena na předpokladu, že každý relační klíč je určen právě pro jednu relaci (komunikaci). Pokud se útočník dostane ke staršímu klíči, může vnutit jeho

použití v další relaci. Jak Denning a Sacco, tak i Needham a Schroeder zajišťují opravu tohoto problému požadavkem, že přeposlaná zpráva od A k B vytváří novou relaci.

5 Úvod do formální verifikace

Proces formální kontroly modelu systému, jejímž cílem je určit, zda model splňuje specifikované požadavky, se nazývá formální verifikace. Tato technika hraje v oblasti bezpečnosti významnou roli. V zásadě můžeme říci, že vývoj formální technik se ubírá třemi hlavními směry:

- **Kontrola modelem** (*model checking*) je založena na konstrukci pravděpodobných množin útoků, které vychází z algebraických vlastností protokolů. Jejich hlavní nevýhoda spočívá ve velkém množství událostí, které je nutné zkontrolovat.
- **Autentizační logiky** (*authentication logics*) jedná se o modální logiky podobné těm, které byly vytvořeny pro analýzu a vývoj znalostí a předpokladů (*logics of knowledge and belief*).
- **Induktivní techniky** (*inductive proofs*) tyto metody nahrazují náročné prohledávání teoretickými poznatky (teorémy) o tomto prohledávání. Tyto techniky jsou komplementární k první skupině technik, protože jsou také založeny na formalizaci problémů, při které se používají hypotézy a autentizační vlastnosti. Tyto techniky formálně modelují aktuální operace v protokolu a ověřují teorémy těchto operací (*theorem proving*)



Obr. 6: Metody formální analýzy protokolů

6 Kontrola modelem

V tomto případě se protokol modeluje jako systém, obvykle s konečným počtem stavů. Pak se prochází tento stavový prostor a kontroluje se, zda všechny stavy odpovídají zadaným požadavkům. Pokud se jedná o systém s konečným počtem stavů, celý proces verifikace může probíhat automaticky.

6.1 Specifikační jazyky

Při použití formálních metod je třeba dosáhnout dvou cílů. Prvním z nich je formální verifikace, kdy ověříme, zda protokol splňuje určené bezpečnostní požadavky. Druhým úkolem je najít v jeho specifikaci slabá místa.

Techniky zařazené do této kategorie používají pro analýzu a verifikaci bezpečnostních protokolů specifikační jazyky. Hlavní myšlenkou je přistupovat k bezpečnostnímu protokolu jako k běžnému programu a ověřit jeho správnost. Kritický pohled na tento přístup říká, že ověření správnosti nemusí vždy znamenat ověření bezpečnosti.

Přestože v počátcích vývoje formálních technik doznala tato kategorie velké pozornosti, nyní se vývoj orientuje spíše na autentizační logiky a induktivní metody.

6.1.1 Kemmer a jazyk „Ina Jo“

Kemmer popisuje [21] formální model, který používá stavový automat. Při použití tohoto modelu se systém nachází v různých stavech, které se od sebe vzájemně liší hodnotami stavových proměnných. Hodnoty těchto proměnných mohou být modifikovány pouze na základě přesně definovaných pravidel. Kemmer používá rozšířený kalkul (*first-order calculus*), který nazval „Ina Jo“. Tento neprocedurální výrokový jazyk nebyl původně určen pro specifikaci bezpečnostních protokolů. Byl vytvořen jako obecný nástroj pro vývoj softwarových technik a ověřování jejich správnosti.

Ina Jo používá následující symboly pro logické operace:

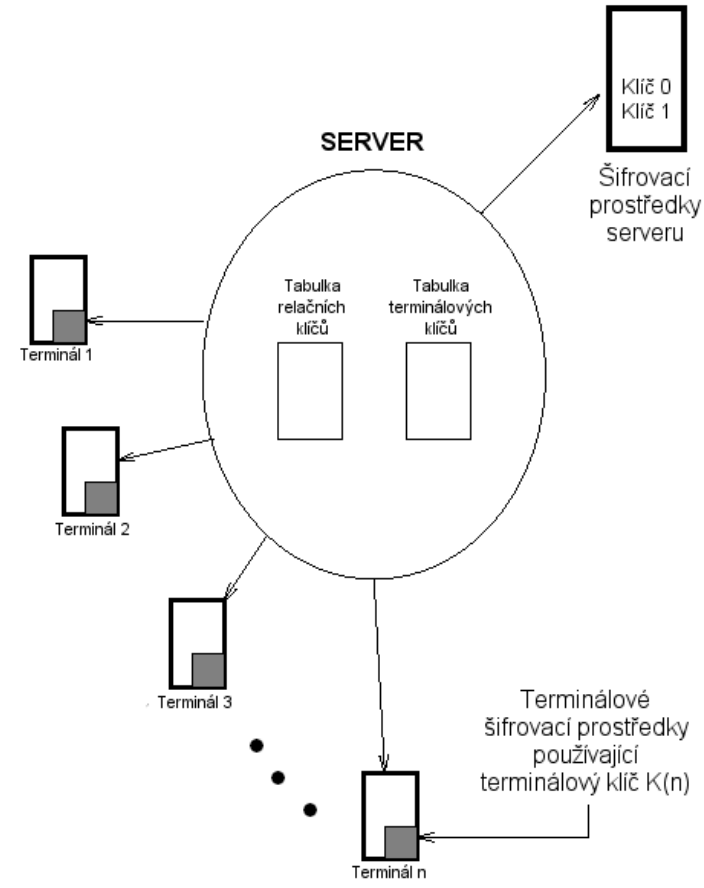
&	logické AND
→	logická implikace
(if A then B else C)	podmínka, kde A je predikát a B, C jsou definované výrazy
∈	…je prvkem…
∪	sjednocení množin
{ a, b, …, c }	množina obsahující prvky a, b, …, c
{ popis množiny }	množina explicitně popsaná

Dále se používají následující kvantifikátory:

∀	pro každé
∃	existuje
N″	nová hodnota proměnné, např. N″v1 znamená nová hodnota proměnné v1

T″ určuje podtyp daného typu, T

Kemmer popisuje příklad systému a přikládá jeho specifikaci v jazyce Ina Jo. Jedná se o demonstrační systém, který však nebyl implementován. V tomto systému je připojeno *n* terminálů k hlavnímu serveru. Každý terminál je vybaven kryptografickou ochranou pro uchování příslušného terminálového klíče. Server obsahuje dvě tabulky klíčů. V první tabulce je uložen seznam relačních klíčů používaných v systému, druhá tabulka obsahuje terminálové klíče. Centrální server zde hraje roli autentizačního serveru.



Obr. 7: Architektura demonstračního systému

Jazyk Ina Jo obsahuje konstanty, proměnné a klauzule. Protože každý terminál má svůj konstantní klíč, můžeme jako příklad konstanty uvést:

```
Terminal_key(Terminal_num):Key
```

Naproti tomu relační klíče jsou rozdílné pro různé relace, jako příklad proměnné můžeme uvést:

```
Session_Key(Terminal_num):Key
```

Klauzule se užívají pro změnu stavu při analýze, například pro generování relačního klíče:

```
Generate_Session_Key
```

Axiomy v jazyce Ina Jo reprezentují vlastnosti, které předpokládáme. Pro vyjádření komutativnosti šifrování a dešifrování použijeme následující axiom:

```
AXIOM  ∀t:TEXT,  k1, k2: Key
(Encrypt(k1, Decrypt(k2, t)) = Decrypt(k2, Encrypt(k1,t)))
```

Kritéria v tomto jazyce udávají mezní podmínky, které musí být v každém stavu splněny. Pokud chceme například vyjádřit, že se útočník nedostane k žádnému klíči, můžeme napsat:

```
CRITERION  ∀k:Key  (k ∈ Intruder_info → k ∉ Keys_Used)
```

Jakmile je specifikace systému hotova, Ina Jo z ní vytvoří soubor pravidel, které mohou být použity k verifikaci. V průběhu této verifikace se ověří, zda v každém stavu jsou splněny mezní podmínky.

Nevýhodou této metody je, že úspěšné otestování mezních podmínek při analýze nemusí nutně znamenat bezpečnost systému. Při specifikaci požadavků, které zabezpečí systém před aktivními útoky, je také třeba znát druhy těchto útoků.

6.1.2 LOTOS

Varadharajan vytváří jazyk LOTOS (Language of Temporal Ordering Specification) [8] pro specifikaci autentizačních protokolů.

Vlastnosti systému LOTOS:

- výkonnost – schopnost postihnout široké spektrum vlastností služeb a protokolů
- dobrá definovanost – syntax a sémantika umožňují jednoduché zacházení a validaci
- dobrá strukturovanost – přehlednost, jednoduché porozumění specifikace
- abstrakce – umožňuje vyšší abstrakci bez specifikace implementačních detailů

Jazyk LOTOS byl vyvinut v rámci Open Systems Interconnection (OSI) a je založen na procesní algebře (*process algebra*), jež však nepoužívá temporální logiku (přestože by se podle názvu mohlo předpokládat).

Systémy jsou v jazyce LOTOS modelovány jako seznamy procesů, u kterých je specifikováno pořadí událostí. Tento jazyk tak může být použit i k modelování zpráv zasílaných v autentizačních protokolech.

6.1.3 Sindhu a Varadharajan

Sidhu a Varadharajan popisují [9] [10], jak specifikovat protokol pomocí konečných stavových diagramů. Pro každého účastníka komunikace je použit orientovaný graf. Nejprve je specifikován počáteční stav, pak jsou do grafu přidávány další stavy podle zpráv, které mohou být zaslány nebo přijaty.

Needham-Schroeder protokol jako konečný stavový automat

Tuto techniku demonstrujeme na Needham-Schroeder protokolu. Jeho specifikace je následující:

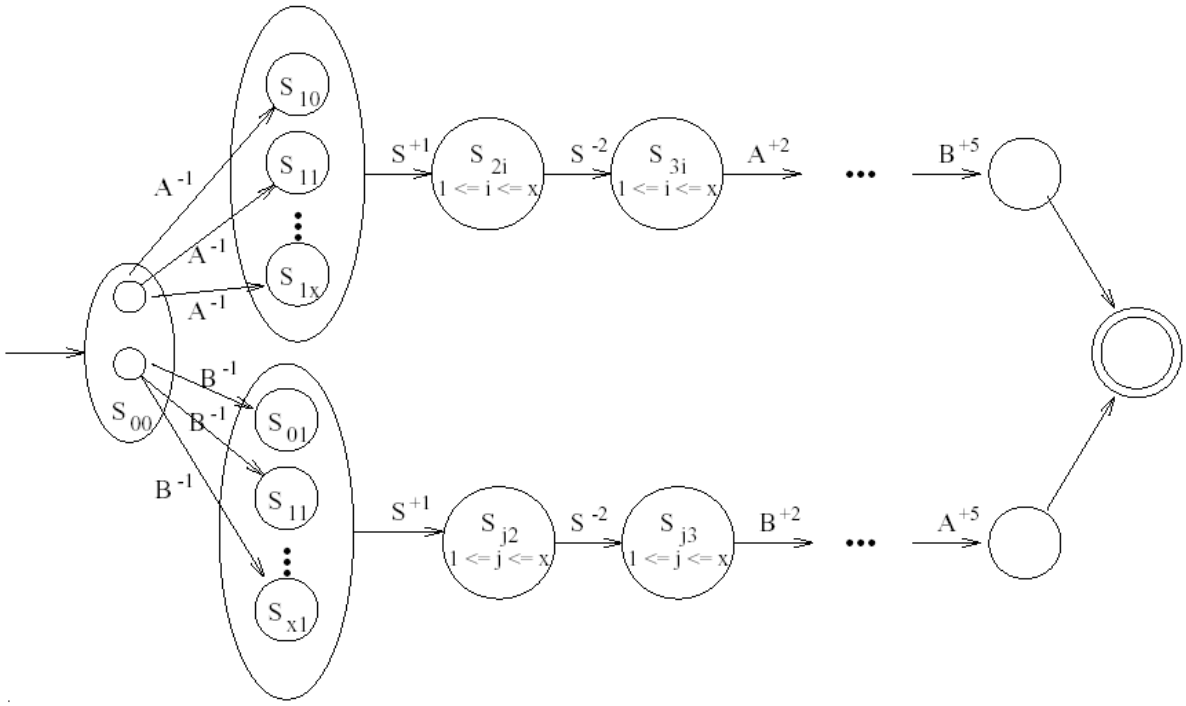
1. $A \rightarrow S$: A, B, Na
2. $S \rightarrow A$: { Na, B, Kab, { Kab, A }$_{Kbs}$ }$_{Kas}$
3. $A \rightarrow B$: { Kab, A }$_{Kab}$
4. $B \rightarrow A$: { Nb }$_{Kab}$
5. $A \rightarrow B$: { Nb – 1 }$_{Kab}$

Zároveň uvažujme následující notaci:

P^{-1} – subjekt P odeslal zprávu 1

P^{+1} – subjekt P přijal zprávu 1

Varadharajan vytváří stavový diagram pro každou entitu: A, B a S. Úkolem je sledovat chování jednotlivých subjektů. Protože je pak tento příklad dosti složitý, budeme uvažovat protokol jako kartézský součin jednotlivých grafů:



Obr. 8: Konečný automat reprezentující bezpečnostní protokol

Tento nedeterministický konečný stavový automat je konstruován z dílčích automatů subjektů A a B. Tyto dílčí automaty jsou sestaveny jako sekvence stavů, kdy jednotlivé přechody reprezentují odeslání nebo přijetí zprávy. Stavy jsou označeny P^{-n} a P^{+n}, což znamená, že subjekt P právě odeslal nebo přijal zprávu n.

Pokud je dosaženo koncového stavu, znamená to správný průběh protokolu. Pokud se dílčí automat pro daný subjekt sestává z *x* stavů, pak kartézský součin těchto automatů má x^2+1 stavů, kdy $+1$ znamená přidání koncového stavu. Všechny ostatní stavy reprezentují špatný průběh protokolu.

Tato verifikační technika je vhodná pro analýzu protokolů, kdy kontrolujeme, zda v jednotlivých stavech mají zúčastněné subjekty správné informace, očekávané podle specifikace protokolu. Nevýhodou této techniky je to, že pouze kontroluje správnost protokolu a ne jeho bezpečnost proti aktivnímu útočníkovi.

6.2 Algebraické zjednodušování teoretických modelů

V této kategorii přístupu k analýze protokolů se vyvíjí formální model založený na algebraickém přepisování (*algebraic term-rewriting*) vlastností kryptografických systémů. Součástí je analýza dosažitelnosti specifikovaných stavů protokolu. Tento přístup se snaží dokázat, že nezabezpečený stav protokolu není dosažitelný, ve srovnání s jinými přístupy, kdy začínáme analýzu v nezabezpečeném stavu a dokazujeme, že neexistuje cesta vedoucí do toho stavu, která by začínala v počátečním stavu.

6.2.1 Dolev a Yao

Dolev a Yao navrhli [13] první algebraický model pro bezpečnostní protokoly. Jejich protokoly se více zaměřovaly na distribuci tajemství než na autentizaci, přestože jsou tyto dvě věci spolu úzce spojeny. Hlavní rozdíl je v absenci třetí komunikující strany, tedy autentizačního serveru. Jsou tedy uvažovány pouze dvě strany.

Autoři definují některé třídy protokolů. Zabývají se více těmito třídami, na úkor studie protokolů samotných a vyzdvihují některé zajímavé vlastnosti těchto tříd. Příkladem takových tříd mohou být kaskádní protokoly (*cascade protocols*) nebo protokoly se jmenným razítkem (*name-stamp protocols*). Kaskádní protokoly jsou takové, v nichž uživatel může na zprávy aplikovat šifrovací/dešifrovací operace v různých vrstvách.

Podle autorů je protokol bezpečný, právě když splňuje následující podmínky:

1. Zpráva zaslána mezi subjekty X a Y vždy obsahuje nějaké úrovně šifrovacích funkci E_X, E_Y .
2. Při generování odpovědi na zprávu neaplikuje nikdy žádný subjekt A (A = X/Y) funkci D_A bez aplikace E_A .

Ve svých pracích autoři také prezentují algoritmus (s polynomiální časovou složitostí), který rozhoduje bezpečnost protokolu se jmenným razítkem.

6.2.2 Merittův model

Merrit [20] zobecňuje techniku rozpracovanou Dolev&Yao tak, aby ji bylo možné použít pro modelování odlišných kryptografických systémů a pro účely formální specifikace jiných bezpečnostních vlastností než je utajení. Základem jeho přístupu je použití zpráv, které útočník zná, a také vztahy mezi těmito zprávami pro modelování znalosti útočníka.

Je definována parciální algebra (*partial algebra*), $A = \langle D, R_1, \ldots, R_k \rangle$, kde D je množina a $R_1 \ldots R_k$ jsou operace nad touto množinou. Podalgebrou A je libovolná algebra A' s $D' \subseteq D$, přičemž její operace jsou ty z algebry A, které jsou omezené nad množinou D'. Algebry jako $B = \langle D, R_1 \rangle$ obsahující podmnožinu relací z A jsou nazývány redukcemi (*reducts*).

Mějme parciální algebru $I = \langle M, C, E_A, E_B, e_A, e_B, e_A^{-1}, e_B^{-1} \rangle$, kde M je množina zpráv, $C \subseteq M$ je množina originálních (nezašifrovaných) zpráv, E_X je predikát subjektu X, který je *true* pouze pro zprávy, které byly úspěšně zašifrovány, e_X je šifrovací funkce subjektu X a e_A^{-1} je dešifrovací funkce subjektu X. Pak redukce $I = \langle M, C, E_A\ e_A, e_A^{-1}, e_B \rangle$ a $I = \langle M, C, E_B, e_B, e_B^{-1}, e_A \rangle$ reprezentují kryptografické schopnosti subjektů A, B.

Merrit ukazuje, jak může být algebraický model použitelný k analýze kryptografických protokolů, přičemž aplikujeme algebraické operace výše uvedené algebry.

6.2.3 Toussantova technika

Toussant prezentuje techniku pro formální verifikaci pravděpodobnostních vlastností kryptografických protokolů. Prvním krokem je modelování stavů znalostí (*states of knowledge*) v systému. Je popisována technika pro kompletní odvození znalostí zúčastněných subjektů, založená na algebraickém modelu, který uvádí Merrit.

Stav znalosti subjektů v protokolu je definován jako parciální znalost isomorfismu mezi volnou a krypto-algebrou. Stavy znalostí jsou rozděleny do třech množin označených S, V a SV, které jsou definovány následovně:

F (*fixed*)	obsahuje dvojice (a, b) které reprezentují vzájemné izomorfní mapování z podmnožiny F do podmnožiny krypto-algebry C. Tyto odpovídají prvkům, které subjekt viděl nebo zná na počátku protokolu.
V (*variables*)	obsahuje generátory krypto-algebry, kterých si je subjekt vědom ale nemá je dostupné.
SV (*semi-variables*)	obsahuje prvky, které má subjekt dostupné, ale nemůže je pojmenovat.

Uzávěr CL(X) je definován jako sjednocení X a toho, co může být z X odvozeno za použití operací šifrování a dešifrování. Subjekty provádějí výpočty nad svými aktuálními stavy znalostí a vytvářejí tak nové stavy znalostí. Toussant dále popisuje stavy víry (*states of belief*), které jsou svázány se stavy znalostí (*states of knowledge*) ostatních subjektů.

Různé globální stavy jsou zaneseny do stromu reprezentujícího možné průchody protokolem (*protocol execution tree*). Uzly stromu odpovídají různým volbám hodnot proměnných, které určují účastníci komunikace. Každý průchod stromem pak znamená příslušný průběh protokolu.

Autor ukazuje, že strom může být redukován na konečnou délku a jednotlivým větvím mohou být přiřazeny pravděpodobnostní parametry. Použitím této techniky mohou být verifikovány pravděpodobnostní vlastnosti protokolu.

Nevýhodou této techniky je to, že ve stromu jsou modelovány pouze akce specifikované protokolem. Pokud jsou tedy při analýze zprávy neobsažené v originálním protokolu zaslány útočníkem, pak se nezachovávají předpokládané pravděpodobnostní vlastnosti. Proto se tato metoda používá pro analýzu zaměřenou proti odposlouchávání a útoku přehráváním (*reply attack*). Není však vhodná pro analýzu zaměřenou proti aktivnímu útočníkovi, který dokáže zasílat nové zprávy.

6.2.4 Algoritmus NARROWER

Myšlenkou tohoto algoritmu je tvrzení Catherine Meadows, která uvádí, že na kryptografický protokol může být částečně nahlíženo jako na množinu pravidel pro generování slov v určitém

formálním jazyce. Můžeme taktéž definovat algebraické operace šifrování a dešifrování. Bezpečnost protokolu pak může být založena na možnosti útočníka vygenerovat příslušná slova.

Operacemi v tomto přepisovacím systému jsou redukce termů za použití odmazávacích vlastností slov v systému. Příkladem může být:

`d(X, e(X, Y))` → `Y`

`e(X, d(X, Y))` → `Y`

což reprezentuje symetrické vlastnosti šifrování a dešifrování. Útočník se pokouší zjistit, zda mohou být některá slova derivována na nějaká utajené slovo, například relační klíč.

Meadows používá tento algoritmus implementovaný v jazyce Prolog. Tento algoritmus vychází z triviální (eventuálně prázdné) množiny slov dostupné útočníkovi, a počátečního stavu. Mimo to je definována množina tajemství, která nesmí být útočníkovi dostupná. Algoritmus má ukázat, že neexistuje průchod protokolem, který má začátek v počátečním stavu a vede do stavu, ve kterém se útočník může dozvědět nějaká slova z množiny tajemství.

Algoritmus pracuje induktivně na délce cesty. Začíná s triviální množinou a pokračuje, dokud není možné generovat nový průchod (cestu) protokolem. Pro určitou hodnotu *m* pak můžeme prohlásit, že neexistuje „nebezpečný“ průchod délky menší nebo rovné *m*. Uživatel může program ovlivňovat a změnou počáteční množiny a množiny dostupných pravidel tak vylepšit analyzovatelnost problému.

V průběhu analýz různých protokolů se ukázalo, že jistá tajemství nejsou pro útočníka dostupná, dokud není kompromitován relační klíč. Taková verifikace však není zkouškou bezpečnosti protokolu. To by vyžadovalo otestování, zda je přepisovací systém kompletní, tzn. každé platné slovo je generováno. Bohužel tento systém však kompletní není. Dalším požadavkem na otestování bezpečnosti protokolu je to, že samotná metoda na formalizaci protokolu musí být formální, což není.

Metoda používaná v algoritmu NARROWER tedy sice může být použita k nalezení nezabezpečených míst v protokolu, ale nepředstavuje zkoušku bezpečnosti analyzovaného protokolu.

6.2.5 NRL protocol analyzer

Syverson a Meadows použili některé uvedené techniky, především přepisovacích vlastností (*term-rewriting properties*), k vývoji nástroje NRL protocol analyzer [12] (někdy se používá zkratka NPA). Tento systém je používán k analýze tříd protokolů a není svázán s žádnými předpoklady protokolů. Za pomocí jednoduché množiny požadavků je specifikována třída protokolů. Používají se následující symboly:

→	standardní předpoklad
∧	konjunkce
▽	temporální operátor, znamenající „někdy v minulosti“ (*at some point in the past*).

Každý subjekt si pamatuje lokální číslo průchodu zprávy protokolu (*local round number for a protocol*). Jsou definovány následující operace:

`accept(B, A, Mes, N)`	B přijal zprávu Mes od A, v průběhu lokálního čísla průchodu zprávy N, jenž specifikuje B.

`learn(Z, Mes)`	útočník Z se dozvěděl zprávu Mes.
`send(A, B, (Query, Mes))`	A zaslal B zprávu Mes jako odpověď na Query. Parametr Query je volitelný.
`request(B, A, Query, N)`	B posílá Query subjektu A, v průběhu identifikátoru N jenž specifikuje B. Parametr Query je volitelný.

Za pomocí těchto symbolů a operací mohou být specifikovány požadavky kladené na protokol. To se uvádí jako konjunkce jednotlivých příkazů.

POŽADAVEK 1:

- $\neg(\Diamond$ `accept(B, A, Mes, N)` $\wedge \Diamond$ `learn(Z, Mes)`$)$
- `accept(B, A, Mes, N)` $\rightarrow$
  `(send(A, B, (Query, Mes))` $\wedge$
  `request(B, A, Query, N))`

Tento požadavek obsahuje dvě podmínky, které musí být splněny. První podmínka znamená, že B musel „někdy v minulosti“ přijmout zprávu Mes od A a také že se útočník Z „někdy v minulosti“ nedozvěděl zprávu Mes. Druhá podmínka vyjadřuje, že pokud B přijme zprávu Mes od A v rámci lokálního čísla zprávy N subjektu B, pak A zašle jako odpověď zprávu Mess subjektu B, při identifikátoru N specifikovaném B.

Analýza nástrojem NRL protocol analyzer spočívá ve vytvoření stavového prostoru a jeho průchodu, přičemž průchod protokolem znamená sekvenci jednotlivých stavů. Cílem tohoto přístupu je zjistit, že nezabezpečený stav není přístupný z počátečního stavu.

Tento nástroj je bohužel omezen pouze na použití v rámci Spojených států.

6.2.6 CSP model checker FDR

Autorem tohoto nástroje je Roscoe [13]. Příklad verifikace je možné demonstrovat na protokolu Netbill. Tento protokol zajišťuje bezpečné objednávání zboží a služeb po Internetu. Netbill počítá se 3 typy subjektů: zákazník (C), obchodník (M) a banka (B). Jeho formální specifikace je následující:

1. C $\rightarrow$ M: poptávka zboží
2. M $\rightarrow$ C: zboží, šifrované klíčem K
3. C $\rightarrow$ M: podepsaný příkaz k úhradě
4. M $\rightarrow$ B: schválený, podepsaný příkaz k úhradě
5. B $\rightarrow$ M: podepsané potvrzení
6. M $\rightarrow$ C: podepsané potvrzení
7. C $\rightarrow$ B: dotaz na proběhlou transakci
8. B $\rightarrow$ C: podepsané potvrzení

Protokol lze v modelu CSP zapsat jako 3 různé procesy (pro každý subjekt jeden).

Příklad - proces „zákazník“:

```
CONSUMER = ABORT || coutm ! goodsReq -> GOODS_REQ_SENT\\
GOODS_REQ_SENT = ABORT ||cinm ?x ->
         (if x==encryptedGoods then ENCRYPTED_GOODS_REC
          else ERROR)
ENCRYPTED_GOODS_REC = ABORT || coutm !epo -> EPO_SENT
EPO_SENT = (cinm ?x -> (if (x==paymentSlip) then SUCCESS
                    else if (x==noPayment) then NO_FUNDS
                           else ERROR)) []
    (timeoutEvent -> coutb !transactionEnquiry -> BANK_QUERIED)
BANK_QUERIED = cinb ?x -> (if (x==paymentSlip) then SUCCESS
                  else   if (x==noPayment) then NO_FUNDS
                    else (if (x==noRecord) then NO_TRANSACTION
                             else ERROR))
```

Metoda kontroly modelem obecně předpokládá systémy s konečným počtem stavů. Bohužel existuje řada aspektů, které při modelování protokolu mohou vést k nekonečnému stavovému prostoru. Lze zmínit například výběr klíčů a nonces nebo problém při pamatování si rozsahu procesu útočníka. Postupy používané při analýze nekonečných stavových prostorů jsou nastíněny v kapitole 9.

6.3 Expertní systémy

V této kategorii technik se používají expertní systémy, které analyzují různé průběhy a scénáře bezpečnostních protokolů. Princip spočívá v tom, že se vychází z nežádoucího stavu a kontroluje se, zda je možné tohoto stavu dosáhnout ze stavu výchozího.

Přestože tyto techniky umožňují lépe identifikovat a nalézat chyby v protokolu, nezaručují bezpečnost protokolu, ani nemohou sloužit jako automatický nástroj na vyhledávání chyb. Jinými slovy, je možné zkontrolovat zda protokol neobsahuje danou chybu, ale není možné v něm objevit neznámé typy chyb.

Podle shrnutí Longleyho a Rigbyho poskytují expertní systémy:

- nový pohled na autentizační systém;
- techniku vytváření modelů schopných plynulých zpřesňování;
- metodu interakce s modelem, která poskytuje lepší pohled na operace celého systému;
- model odpovídající na otázky typu „co když…“;
- metodu testování návrhů na modifikaci systému

Expertní systémy je vhodné používat v kombinaci s ostatními technikami.

6.3.1 Interrogator

Interrogator [14] (autor Millen) je program v Prologu, jehož vstupem je specifikace analyzovaného protokolu a kompromitovaná data. Výstupem je historie zpráv ukazující, jak útočník mohl kompromitovaná data získat. Původní verze byly plně automatické, bez možnosti zásahu uživatele v průběhu analýzy. V posledních verzích Interrogatoru je již možná průběžná interakce s uživatelem, přestože některé části programu stále pracují plně automaticky.

V tomto systému je protokol viděn jako soubor komunikujících procesů, vždy jeden proces pro každý zúčastněný subjekt. Systém je založen na konečném stavovém automatu. Každý proces je popsán seznamem možných stavů. Zaslání zprávy způsobí přechod mezi těmito stavy. Každý proces si pamatuje vlastní stav a po vyslání zprávy jiným procesům způsobí změnu stavu u těchto procesů.

Interrogator generuje velké množství cest protokolem, které končí ve specifikovaném nezabezpečeném stavu. Pokud do tohoto stavu existuje cesta z počátečního stavu, pak byla nalezena v protokolu bezpečnostní chyba. Proto je velmi důležité správně specifikovat koncový stav.

V případě, že pro procházení stavy existuje více cest, může být uživatel dotázán, kterou cestu má program zvolit. Tímto se také snižuje pravděpodobnost, že program zůstane v nekonečné smyčce.

Útočník je specifikován relací:

`p_knows(x, H, q)`

kde x jsou položky dat, které byly získány útočníkem, H je historie zpráv ukazující jak k tomu došlo a q je stav systému dosažitelný z počátečního stavu.

Definice této relace je následující:

`p_knows(x, H, q)` ⇔

`x je známo již v počátečním stavu`

∨ (H=H′sent(m) ∧ sent(m): $q' \rightarrow q$ ∧ H′: $q_0 \rightarrow q'$ ∧ p_gets(x, m, H′, q′))

∨ (H=H′e ∧ e: $q' \rightarrow q$ ∧ p_knows(x, H′, q′))

∨ (H: $q_0 \rightarrow q'$ ∧ p_modifies(q′, q, H) ∧ p_knows(x, H, q′))

Podobně je p_gets definováno:

`p_gets(x, m, H, q)` ⇔

`x je součástí m`

∨ ($\{m'\}_k$ je součástí m ∧ p_knows(k, H, q) ∧ p_gets(x, m′, H, q))

Definice `p_knows` ukazuje tři cesty, jak může útočník získat informaci x (s historií zpráv H, stav q):

- získá ji z poslední přečtené zprávy
- může ji znát již z posledního stavu systému
- získá ji pomocí `p_modifies`

Definice `p_gets` říká, že útočník může číst libovolnou zprávu, pokud je však tato zpráva šifrována, nemůže být originál získán bez znalosti správného klíče. Klauzule `p_modifies(q′, q, H)` popisuje, jakým způsobem může útočník získat informaci x.

Autor Interrogratoru Milen tvrdí, že pomocí tohoto nástroje je možné ověřit bezpečnostní chybu v Needham-Schroeder protokolu. Musí být ale na počátku analýzy specifikováno, že útočník již vlastní starý relační klíč.

6.3.2 Systém založený na pravidlech

Tento systém popisují Longley a Rigby. Je používán na otestování, zda je možné specifickými útoky získat klíč v zadaném bezpečnostním schématu. Tento expertní systém používá důkladné prohledávání pro zjištění, zda byl útok úspěšný. Jakmile se systém při analýze zastaví, pak v ukládané historii můžeme najít seznam pravidel vedoucích k tomuto útoku. V některých případech se stává, že prohledávaný prostor je nekonečný a prohledávání se neukončí.

Longley v další svých pracích uvádí systém OPL5, který využívá pravidel pro transformaci cílů na dílčí podcíle, přičemž cílem se zde rozumí útok na protokol. Variantou je systém implementovaný v jazyce PROLOG. Útočník je zde modelován pomocí prohledávacího stromu. Kořen stromu reprezentuje informace, které útočník potřebuje k tomu, aby vznikla „bezpečnostní díra“ (například šifrovací klíče). Ostatní uzly stromu obsahují informace, které jsou útočníkovi k dispozici nebo ty, které vyžaduje. Listy stromu pak reprezentují informace potřebné ke znalosti kořenové položky.

6.4 Shrnutí

V případě, že protokol není bezpečný a nalezne se v něm chyba, okamžitě se může vypsat stav, který požadavkům nevyhovuje. Toto je velká výhoda tohoto způsobu modelování. Tato technika je vhodná pro modelování protokolů, které nemají příliš velký záběr (tedy nejsou příliš obecné) a nemají velký stavový prostor.

Obecně mají tyto typy přístupu pouze omezenou funkci. Expertní systémy se používají zpravidla pro ověření již známých útoků. Nelze je však uplatňovat při obecném dokazování bezpečnosti autentizačních protokolů.

7 Autentizační logiky

Obecně se dá říci, že použití modální logiky při verifikaci protokolů je deduktivní proces. Většina technik tohoto typu je založena na axiomizaci znalostí a předpokladů (*knowledge and belief*) jednotlivých zúčastněných subjektů.

Před vlastním procesem musí být specifikovány počáteční parametry. Vlastní proces pak proběhne v následujících krocích:

- Formální specifikace kroků protokolu v jazyce logiky
- Formální specifikace požadovaných cílů
- Spuštění procesu s počátečními předpoklady a opakovaná kontrola, zda průběžné výsledky odpovídají požadovaným cílům

7.1 BAN logika

Jedná se o nejrozšířenější logiku [15], která se používá pro analýzu autentizačních protokolů. Přestože se vyvinulo velké množství variant této logiky, zaznamenává se mnoho návratů k originální formě. Autory této logiky jsou Burrow, Abadi, Needham.

BAN logika se spíše než pro přímé dokazování bezpečnosti protokolů používá pro zapsání autentizačních protokolů do formálních pravidel za účelem reprezentace vztahů mezi jednotlivými subjekty v systému.

Cílem tvůrců této logiky bylo vytvořit nástroj, který by umožňoval zodpovězení následujících otázek:

1 Co je výsledkem použití konkrétního protokolu?

2 Jaké předpoklady vyžaduje tento protokol ve srovnání s jinými?

3 Nastane v průběhu protokolu nějaký neočekávaný krok, který by mohl být vynechán aniž by tím byla ohrožena bezpečnost protokolu?

4 Šifruje tento protokol zprávu, která by mohla být zaslána nešifrovaně, aniž by byla ohrožena bezpečnost?

7.1.1 Základy BAN logiky

Jediným logickým spojením mezi dílčími výrazy oddělenými čárkou je konjunkce. Podobně jako v běžných logikách i zde platí asociační a komutativní pravidla.

Elementární výrazy:

`P |≡ X, P believes X`	„P předpokládá X“. To znamená že P předpokládá, že X je pravda.
`P ◁ X, P received X, P seen X`	„P přijal X“. Někdo zaslal zprávu obsahující X subjektu P, který tuto zprávu může přečíst (obvykle po dešifrování).
`P |~ X, P said X`	P zaslal zprávu, která obsahovala X. Nebereme v úvahu čas, kdy byla zpráva zaslána ani zda byla zaslána právě v aktuálním průběhu protokolu. Jediné co můžeme říci je, že P |≡ X v době odeslání zprávy.
`P |⇒ X, P controls X`	P má moc (*jurisdiction, authority*) nad X a tak by k němu mělo být přistupováno (mělo by mu být důvěřováno). Tento obrat se používá zejména tehdy, kdy P předává autoritu nad X někomu jinému.
`#(X)`	X je aktuální (*fresh*). To znamená že výraz X nebyl zaslán v žádné zprávě, která předcházela aktuálnímu průběhu protokolu. Definujme platnost tohoto výrazu na *true* pro výrazy používající nonces.

$P \leftrightarrow^{K} Q$	P a Q mohou pro komunikaci používat sdílený klíč K. Tento klíč považujeme za vhodný v tom smyslu, že nebude získán žádným jiným subjektem.
$\mapsto^{K} P$	P disponuje veřejným klíčem K. Příslušný soukromý klíč K^{-1} nebude nikdy získán žádným jiným subjektem než P.
$P \leftrightharpoons^{X} Q$	Zpráva X je utajena a je známa pouze subjektům P a Q nebo těm subjektům, kterým ji P a Q sdělí. Subjekty P a Q mohou využít této zprávy k vzájemnému prokázání identity.

Všechny další logické výrazy jsou složeny z výše uvedených základních výrazů. Před převodem protokolu do BAN logiky musí být provedena jeho idealizace, podle definovaného postupu autorů.

7.1.2 Pravidla usuzování v BAN logice

Základním pravidlem používaným v BAN logice je vyjádření významu zprávy (*message meaning rule*). Používá se k odvození „víry“ subjektu z originálu zprávy:

$$\frac{P \mid\equiv Q \overset{K}{\longleftrightarrow} P, P\,sees\,\{X\}_K}{P \mid\equiv Q\,said\,X}$$

Tento výraz můžeme přeložit jako: „Pokud P předpokládá, že Q a P sdílejí tajný klíč K a P má přístupné X zašifrované klíčem K, a P nezašifrovalo zprávu X klíčem K, POTOM P předpokládá, že Q zaslal zprávu X“.

Podobné pravidlo existuje pro veřejné klíče a sdílené tajemství (*nonce-verification rule*):

$$\frac{P \mid\equiv \#(X),\ P \mid\equiv Q\,said\,X}{P \mid\equiv Q \mid\equiv X}$$

„Pokud P předpokládá, že X je aktuální a nemohl být zaslán v nějaké předchozí relaci a že Q zaslal X, pak P předpokládá, že Q předpokládá X“. Z důvodu jednoduchosti autoři uvažují, že X je jednoduchý text a neobsahuje žádnou podformuli typu $\{Y\}_K$. Tento výraz je důležitý v protokolech, které použitím nonces předchází útokům založených na přehrávání (*replay attacks*).

Dalším výrazem je kompetentní pravidlo (*jurisdiction rule*), které se používá k předávání kompetencí:

$$\frac{P \mid\equiv Q\,controls\,X,\ P \mid\equiv Q \mid\equiv X}{P \mid\equiv X}$$

„Pokud P předpokládá, že Q má kompetenci nad X, a P předpokládá, že Q předpokládá X, POTOM P předpokládá X“.

Burrows dále uvádí ve své práci mnoho dalších pravidel a jejich vzájemných kombinací.

7.1.3 Idealizovaný protokol v BAN logice

Před tím, než je protokol v BAN logice analyzován, musí být převeden do idealizované formy. Typicky je krok v protokolu zapisován jako:

$P \rightarrow Q:\ message$

Tento zápis znamená, že subjekt P poslal subjektu Q zprávu (*message*) a že subjekt Q tuto zprávu přijal. Tento pohled na komunikaci bývá ale často nejednoznačný a složitě se uplatňuje při formální analýze. To můžeme demonstrovat na příkladě, kdy některý ze subjektů použil šifrování relačním klíčem a není přesně jasné, která část zprávy je aktuální (*fresh*), která je zašifrovaná a kdo zná příslušné klíče. Proto je každý krok protokolu transformován do idealizované formy. Zprávu v idealizovaném protokolu pak nazýváme formulí.

$A \rightarrow B:\ \{ A,\ K_{ab} \}_{Kbs}$

V tomto kroku A sděluje B, jenž zná klíč K_{bs}, že K_{ab} je klíč pro komunikaci s A. Můžeme prohlásit, že A není autorem této zprávy, protože nezná klíč K_{bs}. Zpráva tedy pochází ze serveru S. Tento krok je idealizován takto:

$A \rightarrow B:\ \{ A \rightarrow [K_{ab}]\ B \}_{Kbs}$.

Jakmile je tato zpráva odeslána subjektu B, můžeme uvažovat formuli:

$B \lhd \{ A \rightarrow [K_{ab}]\ B \}_{Kbs}$,

která znamená, že přijímací subjekt B si je vědom zaslané zprávy a může na ni reagovat.

V idealizované formě se části zprávy, které u příjemce nijak nepřispívají k rozšíření předpokladů, neberou v úvahu. Originální (nezašifrované) texty zasílané subjekty se tedy neuvažují, protože mohou být čteny nebo zapomínány kýmkoliv. Idealizované zprávy jsou ve formátu $\{X_1\}_{K1}, \ldots, \{X_n\}_{Kn}$, přičemž se k jednotlivým zašifrovaným částem zpráv přistupuje odděleně.

7.1.4 Analýza protokolu BAN logikou

Autoři této logiky shrnují postup analýzy protokolu do následujících kroků:

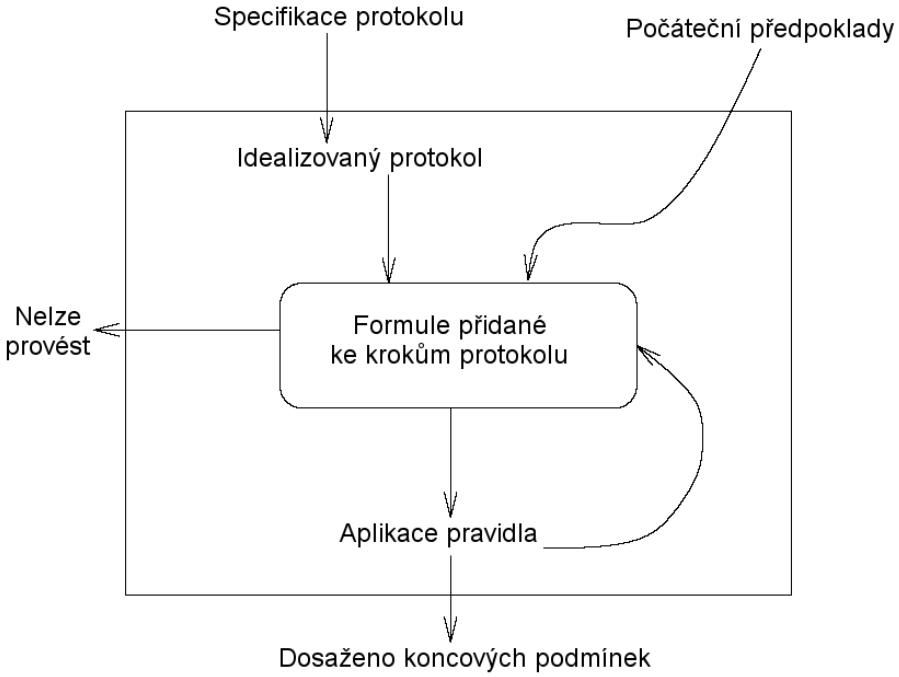
1. Odvození idealizovaného protokolu z původní verze.
2. Specifikace předpokladů v počátečním stavu.
3. Jednotlivým výrazům v protokolu jsou přiřazeny logické formule, podobně jako „tvrzení“(*assertions*) o stavu systému po každém výrazu.
4. Logické formule jsou aplikovány na specifikovaná tvrzení, za účelem nalezení předpokladů, které získají jednotlivé komunikující strany.

Přesněji řečeno, protokol v BAN logice je uspořádanou posloupností výrazů „*send*“ (zaslání zprávy) $S_1, \ldots, S_n$, každý ve formě $P \rightarrow Q:\ X$, přičemž $P \neq Q$. Jednotlivá tvrzení jsou pak přidána na začátek protokolu a za každý jeho výraz. Tato tvrzení jsou kombinací formulí „$P \mid\equiv X$“ (P předpokládá X) a „$P \lhd X$“ (P přijal X). První tvrzení obsahuje počáteční předpoklady (*assumptions*), konečné tvrzení pak důsledky (*conclusions*).

Formule jsou zapisovány ve tvaru:

$$\begin{matrix}[\text{assumptions}] \\ S_1[\text{assertion 1}]S_2...[\text{assertion n}-1]S_n \\ [\text{conclusions}]\end{matrix} .$$

Princip analýzy protokolu v BAN logice je zřejmý z následujícího schématu. Vstupem je specifikace protokolu a počáteční předpoklady. V každém kroku jsou formule přidruženy ke zprávám protokolu a buď je formule aplikována nebo se musí logika přerušit. Pokud je to možné, je dosaženo požadovaného koncového stavu. Protokoly nepoužívají žádné časové specifikace, pouze se rozlišuje, zda již bylo něco vyřčeno v předchozím nebo aktuálním běhu protokolu.



Obr. 9: Princip analýzy protokolu v BAN logice

7.1.5 Cíle autentizace v BAN logice

Otázkou zůstává jak vlastně specifikovat cíle autentizace. Jedno z tvrzení říká, že autentizace mezi A a B je ukončena, pokud existuje K, pro které platí:

$A \mid\equiv A \overset{K}{\leftrightarrow} B$

$B \mid\equiv A \overset{K}{\leftrightarrow} B$

Tuto množinu nazvěme „předpoklad prvního stupně" (*first-level belief*).

Jiné tvrzení říká, že autentizační protokol by měl zabezpečit:

$A \mid\equiv B \mid\equiv A \overset{K}{\leftrightarrow} B$

$B \mid\equiv A \mid\equiv A \overset{K}{\leftrightarrow} B$

Tuto množinu nazvěme „předpoklad druhého stupně“ (*second-level belief*).

Podle Syversona [7] se jednotlivé stupně úrovní liší v závislosti na potřebách jednotlivých aplikací. Tyto úrovně by pak měly být specifikovány spolu s protokolem.

Cheng a Glyor tvrdí, že pro BAN logiku by měly být splněny na konci běhu protokolu následující podmínky:

1. Jak A tak i B jsou přesvědčeny že K_{ab} je tajný klíč sdílený exkluzivně mezi A a B.
2. Oba subjekty A i B předpokládají, že druhá strana má „předpoklad prvního stupně“. Toto je „předpoklad druhého stupně“. Pokud jedna ze stran má „předpoklad druhého stupně“, pak má za to, že byl ustanoven bezpečný kanál.
3. Mezi oběma stupni předpokladů platí příčinná souvislost. To znamená, že „předpoklad prvního stupně“ musí být ustanoven před „předpokladem druhého stupně“.
4. K_{ab} musí být rozeslané stranám A a B a žádný jiný subjekt nesmí K_{ab} předpokládat.

Použití BAN logiky je mnohostranné v tom smyslu, že její funkce mohou být interpretovány různými způsoby. Také cíle BAN logiky se mohou lišit individuálně v závislosti na jejím použití.

Při návrhu BAN logiky bylo postupováno tak, aby logika byla otevřená dalším rozšířením. To znamená možnost přidání dalších konstrukcí a pravidel. Některé takto odvozené logiky budou popsány v dalších kapitolách.

7.1.6 Analýza Needham-Schroeder protokolu v BAN logice

Prvním krokem je převedení uvedeného protokolu do idealizované formy:

$2.\ S \rightarrow A: \{ Na, A \overset{Kab}{\leftrightarrow} B, \#(Kab), \{ A \overset{Kab}{\leftrightarrow} B \}_{Kbs} \}_{Kas}$	from S
$3.\ S \rightarrow B: \{ A \overset{Kab}{\leftrightarrow} B \}_{Kbs}$	from S
$4.\ B \rightarrow A: \{ Nb, A \overset{Kab}{\leftrightarrow} B \}_{Kab}$	from B
$5.\ A \rightarrow B: \{ Nb, A \overset{Kab}{\leftrightarrow} B \}_{Kab}$	from A

První zpráva je v idealizované formě vynechána. Také jsou vynechány nezašifrované části zpráv. Vždy se předpokládá, že subjekty rozeznají svoje zprávy. Příjemce tedy může po dešifrování zprávy říci, od koho zpráva pochází. Změnil se také obsah zašifrovaných zpráv. Především klíč Kab je nahrazen tvrzeními o tomto klíči. Například idealizovaná zpráva 2 obsahuje následující informace: nonce Na, tvrzení že Kab je „dobrý“ pro komunikaci s B, tvrzení o aktuálnosti Kab a zašifrovanou zprávu, která má být zaslána subjektu B. Další změnou je nahrazení Nb-1 na Nb v poslední zprávě. Toto je proto, že účelem -1 je odlišení od zprávy č. 4. Toto je však v idealizovaném protokolu zohledněno v položce from .

Jakmile je hotová idealizace protokolu, můžeme ustavit předpoklady:

P1. $A \mid\equiv A \overset{Kas}{\leftrightarrow} S$

P2. $B \mid\equiv B \overset{Kbb}{\leftrightarrow} S$

P3. $A \mid\equiv S \mid\Rightarrow A \overset{K}{\leftrightarrow} B$

P4. $B \mid\equiv S \mid\Rightarrow A \overset{K}{\leftrightarrow} B$

P5. $A \mid\equiv S \mid\Rightarrow \#(A \overset{K}{\leftrightarrow} B)$

P6. $A \mid\equiv \#(Na)$

P7. $B \mid\equiv \#(Nb)$

V předpokladech P1 a P2 subjekty A a B věří v kvalitu dlouhodobých klíčů. P3 až P5 představuje tvrzení že subjekty A a B předpokládají kompetentnost serveru. V P6 a P7 se tvrdí, že subjekty A a B věří v aktuálnost vygenerovaných nonces. Pro doplnění předpokladů musíme také uvažovat tvrzení převzaté z idealizované formy protokolu.

Doplňující tvrzení pak mají tvar:

P8. $A \lhd \{ Na, A \overset{Kab}{\leftrightarrow} B, \#(Kab), \{ A \overset{Kab}{\leftrightarrow} B \}_{Kbs} \}_{Kas}$

P9. $B \lhd \{ A \overset{Kab}{\leftrightarrow} B \}_{Kbs}$ from S

P10. $A \lhd \{ Nb, A \overset{Kab}{\leftrightarrow} B \}_{Kab}$ from B

P11. $B \lhd \{ Nb-1, A \overset{Kab}{\leftrightarrow} B \}_{Kab}$ from A

Odvozené specifikace předpokladů (*beliefs*) subjektu A:

1. $A \mid\equiv S \mid\sim (Na, A \overset{Kab}{\leftrightarrow} B, \#(A \overset{Kab}{\leftrightarrow} B), \{ A \overset{Kab}{\leftrightarrow} B \}_{Kbs})$

2. $A \mid\equiv \#(Na, A \overset{Kab}{\leftrightarrow} B, \#(A \overset{Kab}{\leftrightarrow} B), \{ A \overset{Kab}{\leftrightarrow} B \}_{Kbs})$

3. $A \mid\equiv S \mid\equiv (Na, A \overset{Kab}{\leftrightarrow} B, \#(A \overset{Kab}{\leftrightarrow} B), \{ A \overset{Kab}{\leftrightarrow} B \}_{Kbs})$

4. $A \mid\equiv S \mid\equiv (A \overset{Kab}{\leftrightarrow} B)$

5. $A \mid\equiv S \mid\equiv (\# A \overset{Kab}{\leftrightarrow} B)$

6. $A \mid\equiv (A \overset{Kab}{\leftrightarrow} B)$

7. $A \mid\equiv \#(A \overset{Kab}{\leftrightarrow} B)$

Nyní přejděme k subjektu B:

8. $B \mid\equiv S \mid\sim A \overset{Kab}{\leftrightarrow} B$

To je vše co můžeme o subjektu B tvrdit vzhledem k Kab. Na rozdíl od A neposílá B žádné nonce. Jediné co bychom mohli dále specifikovat je, že B věří, že to co přijal je aktuální (*fresh*):

P12. $B \mid\equiv \#(A \overset{Kab}{\leftrightarrow} B)$

Můžeme odvozovat další tvrzení:

9. $B \mid\equiv S \mid\equiv A \overset{Kab}{\leftrightarrow} B$

10. $B \mid\equiv A \overset{Kab}{\leftrightarrow} B$

Na rozdíl od A jsme vnutili tvrzení: $B \mid\equiv \#(A \leftrightarrow^{Kab} B)$ protože jsme jej nemohli odvodit. Tímto jsme odvodili „předpoklady prvního stupně“ pro subjekty A a B a aktuálnost Kab. Nyní odvodíme jejich „předpoklady druhého stupně“:

11. $A \mid\equiv B \mid\sim (Nb, A \leftrightarrow^{Kab} B)$
12. $A \mid\equiv \#(Nb, A \leftrightarrow^{Kab} B)$
13. $A \mid\equiv B \mid\equiv (Nb, A \leftrightarrow^{Kab} B)$
14. $A \mid\equiv B \mid\equiv A \leftrightarrow^{Kab} B$

Podobně můžeme odvodit:

$B \mid\equiv A \mid\equiv A \leftrightarrow^{Kab} B$,

ale s tím rozdílem, že již

$B \mid\equiv \#(A \leftrightarrow^{Kab} B)$,

není tento předpoklad nutný pro Nb (tj. aktuálnost Nb pro B). Jedinou rolí, kterou zde Nb má, je odlišení zprávy 4 od zprávy 5. Nb ale nemusí být aktuální.

7.2 GNY logika

Autory této logiky jsou Gong, Needham a Yahalom [17]. Popisují nové způsoby eliminace požadavků původní BAN logiky. Pro tuto logiku je specifické, že nepočítá s tím, že v zašifrovaných zprávách existuje redundance. Místo toho zavádí nový pojen „rozpoznatelnost“ (*recognizability*) pro reprezentaci faktu, že subjekt očekává jistý formát zprávy, kterou přijme.

Subjekt zúčastněný v protokolu očekává od zprávy, kterou přijímá, určité vlastnosti. Tato skutečnost je zohledněna v analýze. Pokud tedy krok v protokolu určuje, že A přijme nonces N_a a N_b , pak následující dvě hodnoty budou interpretovány jako nonces. Pro reprezentaci této vlastnosti byla zavedena pravidla, která jsou definována níže.

Jedním z principů GNY logiky je definice rozdílu mezi vírou (*belief*) a vlastnictvím (*possession*). V této rozšířené logice jsou tedy každému subjektu přiřazeny dvě množiny: množina předpokladů (*belief set*) a množina vlastnictví (*possession set*).

GNY logika definuje následující základní pravidla:

$P \lhd X$ P přijme X, obvykle po provedení nějaké operace jako je např. dešifrování. Formule, která je sdělena, může být samotná zpráva X nebo cokoliv co může z X subjekt P vytvořit.

$P \ni X$ P vlastní nebo je schopné vlastnit formuli X. V průběhu protokolu vlastní subjekt P formule, které mu byly sděleny, se kterými začal relaci nebo ty, které může ze získaných formulí odvodit.

$\phi(X)$ Formule X je rozpoznatelná (*recognizable*). Pokud P předpokládá $\phi(X)$, pak P rozpozná X, jestliže existuje určité očekávání vlastností, které by mělo X mít.

Na formule v GNY logice může být také pohlíženo jako na vnější (*non-originated-here*), což znamená, že nebyly vytvořené žádným subjektem v aktuálním průběhu protokolu. Takové formule se pak označují hvězdičkou (*).

Jsou definovány následující postuláty:

$$\frac{P \lhd X}{P \ni X}$$

který znamená, že subjekt vlastní to, co mu bylo sděleno

$$\frac{P \ni X, P \ni Y}{P \ni (X,Y), P \ni F(X,Y)}$$

který znamená, že pokud subjekt P vlastní X a Y, pak také vlastní konkatenaci X a Y a také libovolnou vypočitatelnou funkci s argumenty X, Y.

Podobně pro rozpoznatelnost:

$$\frac{P \mid\equiv \phi(X)}{P \mid\equiv \phi(X,Y), P \mid\equiv \phi(F(X))}$$

což znamená, že pokud P předpokládá, že X je rozpoznatelné, pak také předpokládá, že X v konkatenaci s čímkoliv je rozpoznatelné, stejně jako jakákoliv spočitatelná funkce aplikovaná na X je rozpoznatelná.

Další významný postulát v GNY logice říká, že pokud uvažujeme

$$\frac{C1}{C2},$$

pak pro každý subjekt P platí také postulát:

$$\frac{P \mid\equiv C1}{P \mid\equiv C2}.$$

Toto se nazývá rozumové pravidlo (*rationality rule*) a umožňuje subjektům usuzovat o stavu ostatních subjektů.

Výsledky analýzy touto logikou jsou závislé na úrovní důvěry (*trust level*) projevené mezi jednotlivými subjekty. Originální BAN logika vůbec neuvažovala s různými úrovněmi důvěry. GNY

logika tímto rozšiřuje třídu protokolů, které mohou být analyzovány. Bohužel bývá méně používaná pro svoji těžkopádnost a rozsáhlost pravidel. Ve srovnání s ní se více používá právě jednodušší BAN logika.

Autoři GNY logiku dále rozšířili tak, aby bylo možné zacházet s neproveditelnými specifikacemi protokolu. Jádro problému je v tom, že specifikace, která nemůže reprezentovat situaci reálného světa, může být přesto verifikována a uznána za správnou.

Zavádí se nový pojem „oprávněnost" (*eligible*), jenž je definován:

$P \propto X$ Subjekt P je oprávněný odeslat formuli X. Subjekt je oprávněný zaslat jen to, co vlastní (*possesses*) nebo dokáže vytvořit (*construct*).

Jednoduchým příkladem neproveditelné specifikace může být protokol, ve kterém subjekt P posílá subjektu Q heslo dalšího subjektu (označme jej např. R). Pokud toto heslo označíme X, potom je krok v protokolu zapsán jako:

$P \rightarrow Q: \{ X \}_{Kpq}$.

Vzhledem k tomu, že ani P ani Q neznají heslo subjektu R, je tento protokol neproveditelný.

Dalším příkladem neproveditelné specifikace, kterou původní GNY logika ani BAN logika neuměly detekovat, je nepodchycení příčinných vztahů předpokladů.

Mějme protokol, v němž jedním krokem je:

$P \rightarrow Q: \{ P \mid\equiv P \leftrightarrow^{S} Q \}_{Kpq}$.

Toto způsobí $Q \mid\equiv P \mid\equiv P \leftrightarrow S\ Q$. Pokud však již v předchozích krocích nebylo specifikováno $P \mid\equiv P \leftrightarrow S\ Q$, pak nejsou zachovány příčinné vztahy mezi jednotlivými předpoklady. To znamená, že byl ustanoven „předpoklad druhého stupně", avšak bez ustanovení „předpokladu prvního stupně". Příčinný (kauzální) řetězec je přerušen vždy, když se subjekt P snaží zaslat zprávu obsahující předpoklad, který doopravdy nevlastní.

BAN logika předpokládá, že subjekty automaticky uvažují to, co říkají a tedy tato situace v příčinných vztazích se nebere jako problém.

Gong zavádí do GNY logiky další postuláty

$$\frac{P \rightarrow Q: X, P \propto X}{Q \triangleleft X}$$

Tento postulát říká, že pokud P zašle zprávu X subjektu Q a zároveň P je oprávněný k zaslání X, pak Q přijme zprávu X.

Podobně pak také:

$$\frac{P \ni X}{P \propto X},$$

což znamená, že pokud P vlastní X, pak P je oprávněno poslat X.

Dalšími pravidly bychom mohli specifikovat, kdy je subjekt oprávněn šifrovat klíčem K nebo provádět hašovací funkci.

7.3 KPL logika

V již zmíněných logikách lze pozorovat možnost znázornění faktu, že P ví (*knows*) to, že K_{PQ} je tajným klíčem mezi P a Q. Nebyla ale ukázána žádná možnost, jak znázornit jednoduchý fakt, že útočník Z zná klíč subjektu P. Syverson toto nazývá bezvýhradnost klíče (*key simpliciter*) a zavádí logiku KPL, ve které lze tento fakt znázornit.

Jedná se o výrokovou logiku znalostí (*propositional logic of knowledge*), Je to kvantifikovaná modální logika (*quantified modal logic*) s odpovídající interpretací pravděpodobných slov (*possible-words interpretation*). V této logice zná útočník Z klíč subjektu P, pokud je tento klíč přítomný ve všech slovech dostupných útočníkovi. Přičemž máme na mysli všechna slova, která je možné odvodit z útočníkovy aktuální množiny slov.

Syverson tvrdí, že spolehlivost a úplnost KPL logiky nemusí být zárukou k tomu, že neexistuje chyba v úvahách daného bezpečnostního protokolu. Dostačuje však pro důkaz toho, že v protokolu není zavedena žádná formální chyba. Jakmile byl jednou protokol formálně specifikován, logické odvození jakéhokoliv výsledku vztahujícího se ke specifikaci je správné. Podobně cokoliv, na co bude pohlíženo jako na sémantický význam specifikace, může být logikou otestováno.

7.4 Mao a Boyd

Mao a Boyd [18] popisují čtyři slabiny BAN logiky a navrhují novou logiku založenou na BAN, která nabízí některá vylepšení. Ve svých pracích například demonstrují nalezení chyby v Otway-Rees protokolu, přestože tato chyba nebyla BAN logikou nalezena. Zároveň navrhují novou verzi tohoto protokolu a ukazují jeho správnost.

Podle autorů Mao a Boyda má původní BAN logika nedostatky v následujících částech:

1. Idealizace protokolu
2. Víry subjektů
3. Předpoklady protokolu
4. Utajení (*confidentiality*)

Nedostatky idealizace protokolu spočívají podle autorů především v přílišné flexibilností. Nové termy a konstrukce mohou být volně zvoleny z nekonečné abecedy, kterou původní BAN logika disponuje. Chybu v idealizaci protokolu lépe pochopíme, pokud si představíme specifikaci protokolu jako deklaraci procedury. Jakmile jsou formální parametry nahrazeny reálnými hodnotami, je chování podle autorů nepředvídatelné. Autoři využili této analogie pro demonstraci chyby v Otway-Rees protokolu.

V BAN logice je pravidlo pro verifikaci nonce následující:

$$\frac{P \mid\equiv \#(X),\ P \mid\equiv Q\,said\,X}{P \mid\equiv Q \mid\equiv X}.$$

Důsledkem tohoto pravidla je to, že Q předpokládá X, jenž je nonce. Mao ukazuje [18], že „předpokládat nonce“ je nesmyslné. Buď můžeme předpokládat že X je nonce nebo že nonce X je aktuální (*fresh*), ale vůbec také nemusíme nonce předpokládat.

Autoři dále ukazují, že metoda popisující předpoklady je v BAN logice chybná. Malá modifikace předpokladů může z nepoužitelného protokolu udělat použitelný a naopak.

Poslední výtkou autorů BAN logice je oblast utajení. Na příkladech ukazují, jak BAN logika selhala při analýze protokolu, který poskytnul útočníkovi utajené informace. Informace, která má zůstat utajena, se tedy musí explicitně definovat.

Nová logika prezentovaná autory vyžaduje striktní psaní formulí a zpráv. Nahrazuje některé přístupy ve srovnání s původní BAN logikou. Přestože přináší nové zlepšení ve srovnání s původní logikou, nelze ji stále použít jako nástroj na verifikaci správnosti protokolu.

7.5 Gaarder a Snekkenes

Autoři Gaarder a Snekkenes vytváří rozšíření BAN logiky pro analýzu systému veřejných klíčů (PKCS, *public key crypto system*), CCITT X.509 Strong Two-way autentizačního protokolu. Přidávají konstrukce zabývající se časem.

Jsou zavedeny následující konstrukce:

$\mathcal{PK}$ (K, U)	Entita U přidružila „dobrý“ veřejný klíč K.
Π (U)	Entita U přidružila privátní klíč. Tento klíč je znám pouze entitě U.
σ (X, U)	Formule X je podepsána privátním klíčem patřícím U.
(Θ(t_1, t_2), X)	X mezi intervaly t_1, t_2 . Autor, který zaslal zprávu s časovým razítkem X tvrdí, že „je dobrá“ v intervalu t_1, t_2,
Δ(t_1, t_2)	Lokální unikátní RTC čas v intervalu t_1, t_2 .

Podle pravidel veřejného šifrování pak specifikujeme:

$$\frac{U_i\ sees\ \sigma(X, U_j)}{U_i\ sees\ X}$$

Toto reprezentuje tvrzení, že kterýkoliv subjekt, jenž má přístupnou formuli podepsanou privátním klíčem jiného subjektu, má také přístupnou tuto formuli samotnou.

Zajímavým příkladem z této logiky je následující pravidlo

$$\frac{P \mid\equiv Q \mid\equiv \Delta(t_1, t_2),\quad P \mid\equiv Q\, said\, (\Theta(t_1, t_2), X)}{P \mid\equiv Q \mid\equiv X}$$

Toto pravidlo počítá s dobou platnosti časových razítek. To znamená, že formule „je dobrá“ v určitém časovém intervalu. Toho se využívá při analýze X.509 protokolu, který bere v úvahu reálný čas.

Další rozšířenou konstrukcí je:

$\mathcal{R}$(P, X) ,

která znamená: „subjekt P je zamýšlený příjemce zprávy X“.

Typické použití je v následujícím případě:

P $\rightarrow$ Q: $\mathcal{R}$(Q, X), X

kde $\mathcal{R}$(Q, X) označuje, že Q je zamýšleným příjemcem zprávy X.

7.6 VO logika

Paul C. Oorschot rozšiřuje BAN a GNY logiky [15] tak, aby byly schopné pracovat s klíči. Tato logika bývá označována jako VO logika.

Autor představuje sadu nových konstrukcí:

$A \leftrightarrow^{K-} B$	K je nepotvrzené tajemství subjektu A vhodné pro subjekt B. Pouze subjekty A a B znají nebo mohou odvodit K. Avšak B nemusí K znát.
$A \leftrightarrow^{K+} B$	K je potvrzené tajemství subjektu A vhodné pro subjekt B. V tomto případě A získalo potvrzující informaci, že B zná K.
$PK_\sigma(A, K)$	K je veřejný podpisový verifikační klíč patřící subjektu A.
$PK_\sigma^{-1}(A)$	K je „dobrý“ privátní klíč subjektu A. Klíč K^{-1} odpovídá veřejnému klíči K z konstrukce $PK_\sigma(A, K)$.
$PK_\delta(A, K)$	Dohoda na veřejném klíči (*key-agreement*) subjektu A. Pokud neuvažujeme specifickou hodnotu klíče, pak můžeme použít zápis $PK_\delta(A)$.
$PK_\delta^{-1}(A)$	Dohoda na soukromém klíči subjektu A. Klíč K^{-1} odpovídá veřejnému klíči K z konstrukce $PK_\delta(A, K)$.
confirm(K)	Byla demonstrována aktuální znalost K (bez kompromitace K).

Přičemž pojmem „dobrý“ klíč zde máme na mysli nezkompromitovaný klíč příslušného subjektu.

Přestože existují ještě další konstrukce, výše uvedené patří mezi ty nejdůležitější. Za použití těchto základních konstrukcí se definují další pravidla.

Jako příklad pravidla můžeme uvést: subjekt A může získat společný klíč K na základě vlastního privátního klíče a veřejného klíče nějakého dalšího subjektu.

$$\frac{A \text{ has } PK_{\delta}^{-1}(A), A \text{ has } PK_{\delta}(U)}{A \text{ has } K}$$

kde `K = f(PKδ-1(A), PKδ(U))`. K je tedy nějakou funkcí privátního klíče A a veřejného klíče U.

V této logice se poprvé definovala pravidla, která nastínila možnost svázání veřejného klíče s identitou. To je důležité pro předcházení útokům ze středu (*man-in-the-middle attacks*). Pomocí této logiky se také snáze porovnávají formální předpoklady a cíle různých protokolů.

Postupně tato logika doznala dalších vylepšení, autor dále spolupracoval se Syversonem a později společně představují logiku označovanou jako SVO [15].

7.7 Shrnutí

Vzhledem k tomu, že technika modální logiky není příliš prostorově ani časově náročná, je často používaná. Nevýhodou je nutnost přepsat protokol do množiny logických formulí, což se obvykle provádí manuálně – toto je právě potenciální zdroj chyb.

8 Induktivní metody

Tento přístup [1] [2] je založen na logických teoriích. Proces vychází z dedukce konečného modelu protokolu, na rozdíl od jeho generalizace.

8.1 Theorem proving

8.1.1 Základní metoda

Základem je myšlenka, že logika reprezentující model protokolu má konečný počet pravidel, který během kontroly narůstá (pravidla se odvozují).

Principem této verifikační techniky je sled následujících kroků:

- Vytvoření konečné množiny axiomů a pravidel. Používaný nástroj na kontrolu automaticky vytváří samostatnou kontrolní utilitu (*checker*) C.
- Protokol P, jehož vlastnosti *phi* bude kontrolována: počáteční předpoklady a zprávy protokolu P jsou interpretovány jako pravidla, množinu těchto pravidel označme T^0.
- Použitím kontrolního nástroje C se prohledají všechny pravidla (vzorce), T^*, jenž jsou součástí množiny pravidel, které jsou odvoditelné z počáteční množiny T^0.
- Kontrola zda pravidla reprezentující *phi* jsou uložena v množině T^*.

8.1.2 Induktivní techniky

Princip

Lawrencova induktivní metoda nebere, na rozdíl od původní základní metody, ohled na konečnost modelu protokolu. Protokol je induktivně reprezentován jako množina interakcí (zpráv, *traces*) mezi subjekty. V porovnání s modální logikou, vyžaduje tato technika delší a mnohem detailnější testování. Na verifikaci bývá používán nástroj Isabelle. Základní myšlenka vychází z následující matematické definice:

$$P(0) \ldots \text{počáteční podmínky}$$

$$P(x) \Rightarrow P(\text{Suc } x)$$

Při ověření, zda vlastnost P(evs) je splněna pro každou zprávu evs, je požadováno splnění následujících podmínek:

(1) P reprezentuje prázdnou zprávu

(2) Pro každé pravidlo se provede verifikace $P(evs) \Rightarrow P(ev\#evs)$, kde ev#evs je rozšířená zpráva evs o vlastnost ev.

Operátory parts, analz a synth

Tyto tři operátory jsou definovány jako nekonečné množiny zpráv. Každý je definován induktivně jako nejmenší množina uzavřená nad specifickým rozšířením. Každý rozšiřuje množinu zpráv H o ostatní prvky, které jsou z této množiny odvozené. Například H obsahuje množinu počátečních subjektových „znalostí“ a celou historii poslaných zpráv mezi subjekty komunikace.

Množinu parts H získáme z H opakovaným přidáváním komponent složených zpráv. Reprezentuje množinu komponent H, které lze potenciálně (i za cenu použití dalších klíčů) ze zprávy odvodit. Platí následující definice:

$$\text{Crypt } K\ X \in \text{parts } H \Rightarrow X \in \text{parts } H$$

$$\text{parts } G \cup \text{parts } H = \text{parts } (G \cup H)\ .$$

Množina analz H obsahuje, podobně jako u předchozího operátoru, všechny dílčí složky zprávy, avšak s tím rozdílem, že zašifrované zprávy se při znalosti klíče okamžitě dešifrují:

$$\text{Crypt } K\ X \in \text{analz } H,\ K^{-1} \in \text{analz } H \Rightarrow X \in \text{analz } H$$

$$\text{analz } G \cup \text{analz } H \subseteq \text{analz } (G \cup H)$$

$$\text{analz } H \subseteq \text{parts } H\ .$$

Množina synth H představuje všechny možné zprávy, které může útočník získat odposlechnutím konverzace.

$$X \in \text{synth } H,\ K \in H \Rightarrow \text{Crypt } K\ X \in \text{synth } H$$

$$K \in \text{synth } H \Rightarrow K \in H\ .$$

8.1.3 Isabelle

V systému Isabelle jsou účastníci komunikace a zprávy modelovány následujícím způsobem:

```
datatype agent = Server | Friend nat | Spy
datatype msg = Agent agent
        | Nonce     nat
        | Key key
        | MPair     msg msg
        | Crypt     key msg
```

V systému jsou rozlišovány 3 typy účastníků (*agents*): Server S, běžní účastníci komunikace (*friend*), kteří jdou indexování přirozenými čísly, a aktivní útočník (*spy*), který se chová jako běžný účastník komunikace.

Pro ilustraci by pak byl operátor analz v systému Isabelle definován následujícím způsobem:

```
const analz :: msg set => msg set
inductive "analz H"
intrs
Inj  "X ∈ H ⇒ X ∈ analz H"
Fst  "{|X,Y|}∈analz H ⇒ X∈analz H"
Snd "{|X,Y|}∈analz H ⇒ Y∈analz H"
Decrypt "[ | Crypt K X∈analz H;  Key(invKey K)∈analz H | ] ⇒ X∈analz H"
```

Na základě této definice testuje nástroj Isabelle požadovaná pravidla. Tato pravidla zahrnují základní pravidla reprezentující vlastní definici , dále pak případové analýzy (*case analysis*) a vlastní indukci.

8.1.4 Modelování Needham-Schroeder protokolu

V případě, že uvažujeme asymetrickou verzi protokolu, subjekt A vlastní veřejný klíč pubK A, který znají všechny zúčastněné subjekty. Dále privátní klíč priK A. Útočník zná „špatný“ privátní klíč.

Induktivní definice Needham-Schroeder protokolu vypadá následovně:

```
Nil [] ∈ ns_public
Fake [| evs ∈ ns_public; B≠Spy;
   X ∈ synth (analz (spies evs)) |]
   ⇒ Says Spy B X  # evs ∈ ns_public
NS1 [| evs1 ∈ ns_public; A≠B; Nonce NA ∉ used evs1 |]
   ⇒ Says A B (Crypt (pubK B) {| Nonce NA, Agent A |})
         # evs1 ∈ ns_public
NS2 [| evs2 ∈ ns_public; A≠B; Nonce NB ∉ used evs2;
```

```
   Says A' B (Crypt (pubK B) {| Nonce NA, Agent A |}) ∉ set evs2 |]
   ⇒ Says B A (Crypt (pubK A) {| Nonce NA, Nonce NB|})
         # evs2 ∈ ns_public
NS3 [| evs3 ∈ ns_public;
   Says A B (Crypt (pubK B) {| Nonce NA, Agent A |}) ∈ set evs3;
   Says B' A (Crypt (pubK A) {| Nonce NA, Nonce NB |}) ∈ set evs3 |]
   ⇒ Says A B (Crypt (pubK B) (Nonce NB))
         # evs3 ∈ ns_public
```

Uvažujeme 3 pravidla specifická pro tento protokol.

1. V aktuálním průběhu protokolu je Na aktuální (*fresh*) nonce a subjekt B není totožný s A.
   Potom můžeme uvažovat:
   `Says A B (Crypt(pubK B) { Na, A })`

2. Pokud aktuální průběh obsahuje také událost:
   `Says A' B (Crypt(pubK B) { Na, A })`

   a Nb je aktuální nonce, A ≠ B, pak můžeme přidat událost:
   `Says B A (Crypt(pubK B) { Na, Nb })`
   Pokud zapisujeme odesílatele jako A′, pak tím máme na mysli skutečnost, že B neví kdo je autorem zprávy.

3. Pokud aktuální průběh následující dvě události:
   `Says A B (Crypt(pubK B) { Na, A })`
   `Says B' A (Crypt(pubK A) { Na, Nb })`

   a můžeme přidat událost
   `Says A B (Crypt(pubK B) { Nb })`

   Subjekt A dešifruje zprávu a zkontroluje, zda Na souhlasí s tím nonce, které předtím poslal subjektu B. Dále pak odpovídá na výzvu subjektu B tak, že odešle zpět nonce Nb.

Mohli bychom sice modelovat čtyři implicitní kroky, ve který subjekt B kontroluje zprávu přicházející od subjektu A, ale stačí pouze prověřit vzorce (*theorems*) říkající, co B může z uvedené kontroly odvodit, jako například přítomnost subjektu A.
Modelování by dále pokračovalo prověřením záruk bezpečnosti pro jednotlivé komunikující strany.

8.1.5 Shrnutí

Formální metody obecně nemohou zaručit bezpečnost protokolu. Další nevýhodou je, že teorémy mohou být jednoduše špatně interpretovány.

Při verifikaci těmito metodami je vhodné přistupovat k protokolu na různých úrovních abstrakce. Jednotlivé části, např. zprávy protokolu, kryptografické algoritmy a transportní protokoly by měly být ověřovány odděleně.

8.2 spi calculus

Spi kalkul (*spi calculus*) [19] je rozšířením pi kalkulu o podporu kryptografických funkcí. Byl přímo navržen pro popis a analýzu bezpečnostních protokolů. Hlavními funkcemi spi kalkulu jsou kryptografické operace a možnost specifikování komunikace po ustavených kanálech.

Samotný pi kalkul se používá pro popis protokolů na abstraktní úrovni. Funkce pro práci s kanály jsou jednoduché, ale efektivní. Kanály se mohou vytvářet například mezi autentizačním serverem a klienty. Pravidla pi kalkulu zajišťují, že prostředí protokolu (např. útočník) nemá přístup ke kanálu, který mu není explicitně přidělen. Bohužel však pi kalkul nedisponuje žádnými kryptografickými funkcemi, které se běžně používají při implementaci kanálů v distribuovaných systémech. Právě za tímto účelem byl vytvořen spi kalkul, ve kterém lze již požadované funkce specifikovat.

Předpokládejme nekonečnou množinu jmen používanou pro komunikační kanály a nekonečnou množinu proměnných. Nechť *m, n, p, q* a *r* nabývají všechny hodnoty jmen a *x, y* všechny proměnné.

Pak množina termů je definována následující gramatikou:

`L, M, N ::=`	termy
`n`	jméno
`(M, N)`	dvojice
`0`	nula
`suc(M)`	následník
`x`	proměnná

Množina procesů je pak definována gramatikou:

`P, Q, R ::=`	procesy
`M<N>.P`	výstup
`M(x).P`	vstup
`P | Q`	kompozice
`(νn)P`	omezení
`!P`	replikace
`[M is N] P`	totožnost
`0`	nil
`let (x, y) = M in P`	rozdělení dvojice
`case M of 0: P suc (x): Q`	case-příkaz

Základním výpočetním krokem a synchronizačním mechanismem v pi kalkulu je interakce, probíhá komunikace pojmenovaným kanálem *m*, který je ustaven mezi výstupním a vstupním procesem a předává se v něm term *N*.

Kompozici `P | Q` definujeme tak, že P a Q běží paralelně.

Omezení `(νn)P` definujeme tak, že je to proces, který vytvoří nové privátní jméno *n* a pak se chová jako P.

Replikaci `!P` definujeme jako nekonečný počet kopií P, běžících paralelně.

Totožnost A match `[M is N]` P definujeme tak, že pokud M a N jsou totožné, pak se provede process P.

Ve formální podobě zapisujeme protokol sledem následujících příkazů:

$A \rightarrow B: M \text{ on } c_{AB}$

Zápis zaslání zprávy by pak v pi kalkulu vypadal následovně:

$A(M) \triangleq c_{AB}\langle M\rangle$

$B \triangleq c_{AB}(x).\mathbf{0}$

$Inst(M) \triangleq (\nu c_{AB})(A(M) \mid B)$

Pokud vezmeme komunikační protokol:

$A \rightarrow S: c_{AB} \text{ on } c_{AS}$

$S \rightarrow B: c_{AB} \text{on } c_{AB}$

$A \rightarrow B: M \text{ on } c_{AB}$

Pak odpovídající zápis v pi je:

$A(M) \triangleq (\nu c_{AB})c_{AS}\langle c_{AB}\rangle.c_{AB}\langle M\rangle$

$S \triangleq c_{AS}(x).c_{SB}\langle x\rangle$

$B \triangleq c_{SB}(x).x(y).F(y)$

$Inst(M) \triangleq (\nu c_{AS})(\nu c_{SB})(A(M) \mid S \mid B)$

Kde F(y) znamená činnost, kterou provede subjekt B se zprávou y .

Spi kalkul rozšiřuje původní pi kalkul o možnost použití kryptografických funkcí, například operace se sdílenými klíči.

Definujme nový term: $\{M\}_N$ reprezentující zašifrovanou sdíleným klíčem

a přidejme novou klauzuli: `case L of` $\{x\}_N$ `in P` což znamená dešifrování

Krok v komunikačním protokolu, kdy subjekt A zasílá subjektu B zašifrovanou zprávu:

$A \rightarrow B: \{M\}_{Kab}$ on c_{AB}

pak můžeme zapsat ve spi kalkulu takto:

$A(M) \triangleq c_{AB}\langle\{M\}_{Kab}\rangle$

$B \triangleq c_{AB}(x).\text{case } x \text{ of } \{y\}_{Kab} \text{ in } F(y)$

$Inst(M) \triangleq (\nu K_{AB})(A(M) \mid B)$

Autoři pak definují některé složitější kryptografické operace, jako například ustavení klíčů mezi subjekty.

Zápis protokolu ve spi kalkulu se zdá poněkud složitější, než v klasické formální formě. Na druhou stranu bychom mohli říci, že spi kalkul je detailnější. Nebere se v úvahu pouze jaké zprávy jsou zasílány, ale také jak jsou tyto zprávy vytvářeny a ověřovány.

8.3 Shrnutí

Induktivní metody jsou jednoduché a obecné. Představují symbolické prošetřování protokolu. V současné době se klade důraz právě na analýzu induktivními metodami. Mnoho problémů týkajících se induktivních metod stále zůstává nevyřešených.

9 Ostatní přístupy

V následující kapitole budou zmíněny některé přístupy, které nelze přímo zařadit do některé z hlavních kategorií verifikačních metod.

9.1 Hybridní verifikační techniky

Tyto techniky kombinují výhody obou rozdílných přístupů, tedy testování konečného modelu protokolu na bázi procházení stavového prostoru vs. verifikace teoretických pravidel reprezentujících daný protokol.

9.2 Metody řešící problém nekonečného stavového prostoru

Během procesu generování stavového prostoru z počátečního stavu protokolu se může vyskytnout problém nekonečnosti tohoto stavového prostoru, především pak pokud není systém nějak přesněji omezený. Toto může nastat v některém z následujících případů:

- Některým účastníkům komunikace je dovoleno provést předem neodhadnutelné kroky v protokolu
- Někteří účastníci generují nekonečný počet stavů
- Počet účastníků je nekonečný
- Množina zpráv přijatých účastníkem je nekonečná

Atti vytváří novou metodu a nástroj HPA1 (Huima's Protocol Analyzer 1) založený na symbolickém prohledávání stavového prostoru, který počítá s analýzou nekonečného počtu stavů v případě, že tato nekonečnost je způsobena posledním zmiňovaným typem.

Někdy se může stát, že nekonečný stavový prostor je generován i přesto, že samotný model je konečný. Pokud rozměr systému narůstá lineárně, pak mohutnost stavového prostoru roste exponenciálně. V takových případech se využívá kompozionálních verifikačních nástrojů, které problém redukují a vytváří ekvivalentní stavový prostor mnohem menších rozměrů, s ohledem na původní parametry systému. Mezi takové nástroje patří například IOTA (Eric, Tsai).

10 Závěr

Tato diplomová práce měla za úkol shrnutí formálních metod používaných při dokazování a verifikaci bezpečnostních protokolů.

V úvodní části byla představena problematika bezpečnostních protokolů a popsány některé známé protokoly. Ve stěžejní části této práce byly ukázány základní přístupy k analýze bezpečnostních protokolů. Byly vždy představeny hlavní techniky, které uvedený přístup reprezentují. Technik pro formální verifikaci je velké množství, proto byly vybrány pouze některé. Účelem totiž nebylo obsáhnout všechny existující techniky spadající do příslušné kategorie, ale spíše na vybraných technikách příslušný přístup demonstrovat.

V průběhu popisu a demonstrace jednotlivých technik jsem vyzdvihl specifické vlastnosti bezpečnostních protokolů, které jsou pro příslušný přístup důležité. Ukázal jsem, jak se uvedené vlastnosti převedou do jazyka nebo do prostředí uvedené techniky. Tam kde to bylo účelné a vhodné, jsem analýzu demonstroval na protokolu Needham-Schroeder. Tento protokol jsem si vybral pro jeho jednoduchost a všeobecnou známost.

Úkolem diplomové práce bylo mj. také modifikovat vybraný existující nástroj, který se používá pro verifikaci protokolů, tak, aby jej bylo možné použít přímo pro bezpečnostní protokoly. Na začátku tohoto úkolu jsem se zabýval problematikou a možnostmi již existujících nástrojů pro formální verifikaci protokolů, které nejsou přímo určené pro bezpečnostní protokoly. V průběhu studie jednotlivých nástrojů a přístupů jsem pak, na základě konzultace s vedoucím práce, od modifikace již existujícího nástroje upustil. K tomu mě přivedl fakt, že existující nástroje určené pro verifikaci bezpečnostních protokolů považuji za dostatečně univerzální. Propracovaných existujících nástrojů a různých přístupů je velmi mnoho, samotná práce se zabývá pouze výběrem těch stěžejních. Proto jsem se spíše než na návrh nového nástroje zaměřil na analýzu společných rysů existujících nástrojů a na metodiku jejich použití.

Nezanedbatelným přínosem práce je také fakt, že může sloužit jako souhrnný přehled existujících přístupů v českém jazyce, ve srovnání se dostupnými materiály, které jsou k dispozici většinou pouze v angličtině.

Literatura

[1] Paulson, C. L.: The Inductive Approach to Verifying Cryptographic Protocols, Computer Laboratory at University of Cambridge, prosinec 1998

[2] Paulson, C. L.: Proving Security Protocols Correct, Computer Laboratory at University of Cambridge, 1998

[3] Pitt, Martin: Modeling and verification of security protocols, Dresden University of Technology, November 2002

[4] Ma L., Tsai Jeffrey: Formal verification techniques for computer communication security protocols, University of Illinois, Chicago, 2000

[5] Očenášek P.: Verifying Security protocols, In: Proceedings of 9th conference, Student EEICT 2003, Brno, CZ, p. 211-213

[6] Huima A., Efficient Infnite-State Analysis of Security Protocols, `http://netlib.bell-labs.com/who/nch/fmsp99/program.html`

[7] Leathrum J.F., Morsi R., Leathrum T.E.: Formal Verification of Communication Protocols, `http://www.ece.odu.edu/~leathrum/`

[8] Bolognesi T., Bringsma E.:Introduction to the ISO Specification Language LOTOS, Computer Network and ISDN Systems, Vol. 14, 1987, p. 25-29

[9] Sindhu D.: Authentication Protocols for Computer Networks, Computer Network and ISDN Systems, Vol. 11, 1986, p. 297-310

[10] Varadharajan V.: Verification of Network Security Protocols, Computer and Security, vol. 8, 1989, p. 693-708, Elsevier Advanced Technology

[11] Dolev D., Yao A.: On the Security of Public Key Protocols, IEEE Transaction of Information Theory, 29(2), 1983, p. 198-208

[12] Meadows C.: Language Generation and Verification in the NRL Protocol Analyzer, In: Proceedings of the 1996 IEEE Computer Society Foundation Workshop IX, 1996, p. 48-61, IEEE Computer Society Press

[13] Roscoe A.W.: Modeling and verifying key-exchange protocols using CSP & FDR, In: Proceedings of the 1995 IEEE Computer Security Foundations Workshop IX, 1995, p. 98-107, IEEE Computer Society Press

[14] Millen J.: The Interrogator Model, In: Proceedings of the 1995 IEEE Symposium on Security and Privacy, 1995, p. 251-260, IEEE Computer Society Press

[15] Burrows M., Abadi M., Needham R.: A Logic of Authentication, ACM Transactions on Computer Systems, 8(1), 1990, p. 18-36

[16] Syverson P., van Oorschot P.C.: On Unifying some Cryptographic Protocol Logics, In: Proceedings of the 1994 IEEE Computer Security Foundations Workshop VII, 1994, p. 14-29, IEEE Computer Society Press

[17] Gong L., Needham R., Yahalom R.: Reasoning about Belief in Cryptographic Protocols, In: Proceedings of the 1990 IEEE Symposium on Security and Privacy, 1990, p. 234-248, IEEE Computer Society Press

[18] Mao W., Boyd C.: Towards formal analysis of security protocols, In: Proceedings of the 1993 IEEE Computer Society Foundations Workshop VI, 1993, p. 147-158, IEEE Computer Society Press

[19] Abadi M., Gordon A.D.: A Calculus for Cryptographic Protocols. The Spi Calculus, 1998, Digital System Research Center

[20] DeMillo R.A., Lunch N.A., Merrit M.: Cryptographic protocols, In: Proccedings of the 14th Annual ACM Symposium on Theory of Computing, 1982

[21] Kemmer R., Meadows C., Millen J.: Three system for cryptographic protocol analysis, Journal of Cryptology, 7(2), 1994, p.79-130

[22] Kouřil D.: Bezpečnostní protokoly na počítačových sítích. Diplomová práce, 1999, FI MU Brno, CZ

[23] Kohl J., Neuman C.: The Kerberos Network Authentication Service (V54), 1993, Network Working Group, RFC-1510.

[24] Lowe G.: Breaking and fixing the Needham-Schroeder public-key protocol usány FDR, Tools and Algorithms for the Construction and Analysis of Systems, vol. 1055, Lecture Notes in Computer Science, 1996, p. 147-166, Springer Verlag

[25] Rubin A.D., Honeyman P.: Formal Methods for the Analysis of Authentication Protocols, Center for Information Technology Intrgration, University of Michigan, 1996

[26] Volpano D., Smith G.: Verifying secrets and relative secrecy. In: Proceedings of the 27th ACM Symposium on Principles of Programming Languages, 2000, p. 268-276

[27] Fábrega F.J., Herzog J.C., Guttman J.D.: Strand spaces: Why is a security protocol correct? In: Proccedings 1998 IEEE Symposium on Security and Privacy, 1998, p. 160-171, IEEE Computer Society Press

[28] Meadows C.: A system for the specification and analysis of key management protocols. In: Proceedings of the 1991 IEEE Symposium on Research in Security and Privacy, 1991, p. 182-195, Computer Society Press