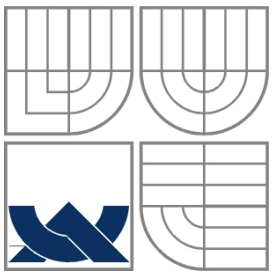
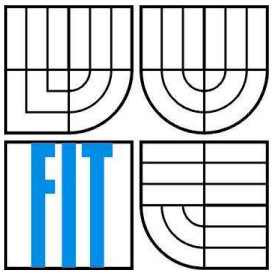


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

ANALÝZA VYBRANÝCH BEZPEČNOSTNÍCH PROTOKOLŮ

ANALYSIS OF SELECTED SECURITY PROTOCOLS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — TOMÁŠ ŘÍHA
AUTHOR

VEDOUCÍ PRÁCE — ING. PAVEL OČENÁŠEK
SUPERVISOR

BRNO 2009

Abstrakt

Tato bakalářská práce se zabývá nástrojem SRI Constraint Solver, určeným pro analýzu bezpečnostních protokolů. Nástroj je v práci stručně charakterizován, a jeho syntaxe předvedena na implementaci protokolu Needham-Schroeder Public Key. Praktická část uvádí příklady analyzovaných protokolů. Pro každý protokol je uvedena jeho specifikace, průběh protokolu v nástroji, publikovaný útok, a v případě nálezu také nalezený útok. Na závěr je uvedena metoda analýzy jednotlivých protokolů a porovnání dosažených výsledků s publikovanými.

Abstract

This bachelor's thesis deals with the SRI Constraint Solver tool used for analysis of security protocols. The tool is shortly characterised, and its syntax is shown on an implementation of the Needham-Schroeder Public Key protocol. The practical part shows some examples of analysed protocols. Every protocol is specified; it's run in the tool, published attack and found attack in case of its presence. At the end of the thesis, a method of analysis of each protocol and a comparison of achieved results with published are described.

Klíčová slova

Bezpečnostní protokol, důvěrnost, autentizace, Needham-Schroeder Public Key, Otway Rees, SRI Constraint Solver.

Keywords

Security protocol, confidentiality, authentication, Needham-Schroeder Public Key, Otway Rees, SRI Constraint Solver.

Citace

Říha Tomáš: Analýza vybraných bezpečnostních protokolů, bakalářská práce, Brno, FIT VUT v Brně, 2009

ANALÝZA VYBRANÝCH BEZPEČNOSTNÍCH PROTOKOLŮ

Prohlášení

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

……
Tomáš Říha
19. května 2009

Poděkování

Rád bych tímto poděkoval panu ing. Pavlu Očenáškovi za veškerou pomoc při vypracování této práce.

Obsah

1 Úvod

První počítačové sítě byly vytvořeny za účelem usnadnění výzkumu, zajišťovaly snadnější výměnu informací a snazší přístup k výpočetním prostředkům. Účastníky komunikace byly především vědecké týmy, a proto se v otázce bezpečnosti spoléhalo především na jejich poctivost a kolegialitu. Pro vzájemnou autentizaci se používala ručně psaná hesla, která byla přenášena v nezašifrované podobě.

V dnešní době tento přístup již není dostačující. Firma AMD odhaduje, že v květnu 2009 využívá služeb internetu více než 1,6 miliardy uživatelů.[4] Vzhledem k tomu se již nelze spoléhat na poctivost jednotlivých klientů, což bylo příčinou vzniku bezpečnostních protokolů a jejich neustálého testování a vývoje.

Bakalářská práce se zabývá právě testováním těchto protokolů. Můžeme ji rozdělit na dvě části. První část se zabývá vysvětlením základních pojmů a principů, které je potřeba znát pro pochopení práce. Součástí je také příklad několika základních protokolů a útoků na ně.

Druhá část popisuje vybraný nástroj a jeho aplikaci na bezpečnostní protokoly. Kapitola SRI Constraint Solver seznamuje čtenáře s tímto nástrojem, vysvětluje jeho syntaxi i návod k použití. Kapitola Implementované protokoly popisuje jednotlivé protokoly analyzované tímto nástrojem. Pro každý protokol je uveden jeho formální popis, průběh protokolu získaný nástrojem a případně také nalezené bezpečnostní riziko.

2 Základní pojmy

2.1 Bezpečnostní protokol

Bezpečnostní protokol předpis pro komunikaci mezi subjekty, zajišťující různé bezpečnostní funkce. Specifikace protokolu by měla obsahovat:

označení subjektů - *A, B, S,* …(Alice – subjekt, Bob – subjekt, Server – třetí strana)
nonces - *Na, Nb,* …(náhodně generované hodnoty)
klíče - *Kab*(symetrický klíč A a B)
K_a,K_b,…, K_a^{-1}, K_b^{-1}(asymetrické klíče pro A a B)
zasílané zprávy a jejich zřetězení - *M, N,*…
zašifrované zprávy - $\{M\}_K$

Bezpečnostní protokol může zajišťovat autentizaci subjektu, kdy prověří zda opravdu komunikuje s kým si myslí, distribuci klíče, pomocí kterého mezi sebou následně subjekty komunikují, popř. obě tyto funkce.

Protokoly lze rozdělit podle šifrovací funkce na protokoly se symetrickým a asymetrickým šifrováním. Symetrické šifrování využívá pro oba subjekty společný klíč, např. K_{AB} je společný klíč pro subjekty *A* a *B*. Při asymetrickém šifrování existuje pro každý subjekt dvojice klíčů – soukromý a veřejný, např. K_a je veřejný klíč subjektu A, k němuž patří párový soukromý klíč K_a^{-1} .

2.2 Vlastnosti bezpečnostních protokolů

Bezpečnostní protokoly by měly splňovat několik základních vlastností, jsou to:

Autentizace

Jedná se o ověření identity účastníků komunikace. Pro zjištění identity se používají čtyři základní způsoby - co uživatel zná(uživatelské jméno a heslo, PIN,…)
co uživatel má(smart card, privátní klíč, USB dongle,…)
čím uživatel je(otisk prstu, snímek duhovky,…)
co uživatel umí(správně odpovědět na náhodne vygenerovaný dotaz)

Distribuce a správa klíčů

Účastnící komunikace by měli být schopni vzájemně se shodnout na klíči, na jehož vytvoření budou mít obě strany svůj podíl. Tento způsob zabraňuje třetí straně donutit účastníky komunikace přijmout falešný klíč. Protokol by také neměl odhalit třetí straně klíč, na kterém se subjekty dohodly.

Důvěrnost

Důvěrnost zajišťuje, že k dané informaci mají přístup pouze autorizovaní uživatelé.

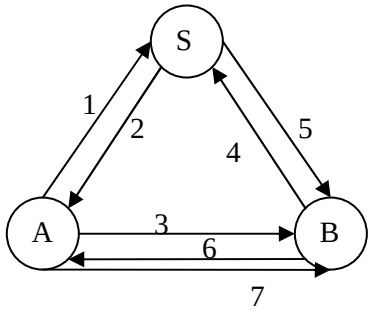
Nepopíratelnost

Zbůsob ujištění se, že skupina počítačů nemůže popřít přijatou platnou zprávu.

3 Příklady protokolů

3.1 Needham-Schroeder Public Key

Tento protokol, jehož autory jsou Roger Needham a Michael Schroeder, byl představen v roce 1978.[14] Protokol zajišťuje vzájemnou autentizaci subjektů za použití důvěryhodného serveru a asymetrické kryptografie. Server poskytne subjektům klíče pro vzájemnou komunikaci, s jejichž pomocí se subjekty A a B vzájemně autentizují.



Obr. 1: Needham-Schroeder Public Key protokol

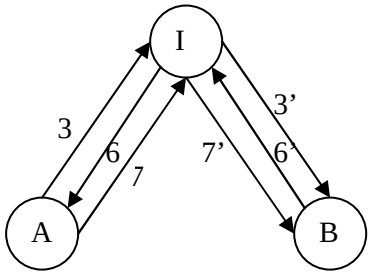
Formální zápis protokolu

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

Alice si vyžádá od serveru klíč pro komunikaci s *Bobem*. *Server Alici* tento klíč poskytne, a ta jej využije pro odeslání zprávy *Bobovi*, která obsahuje nonce *Na* potřebný k autentizaci. *Bob* získá *Na*, ale pro navrácení potřebuje od *Serveru* klíč pro komunikaci s *Alicí*. Po poskytnutí tohoto klíče *Bob* odešle *Alici* zpět nonce *Na*, čímž se jí autentizuje, a nově vygenerovaný nonce *Nb*, který požaduje poslat zpět. *Alice* jej vrátí, a tím je vzájemná autentizace dokončena.

Útok na protokol

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



Obr. 2: Útok na Needham-Schroeder Public Key protokol

Útočník vytvoří dvě spojení, legitímní spojení s Alicí, a spojení s Bobem ve kterém využívá odposlechnuté zprávy z prvního spojení. Z popisu je vyjmuta komunikace mezi subjekty a Serverem.

Alice právě získala klíč pro komunikaci s *útočníkem*, a odesílá mu zprávu obsahující nonce *Na*. *Útočník* této znalosti využije a ve druhém spojení zprávu přepošle *Bobovi*. *Bob* oplatí odeslání *Na* a přidá nonce *Nb*, který právě vytvořil. Útočník však tuto zprávu nemůže dešifrovat, klíč *Ka* nezná.

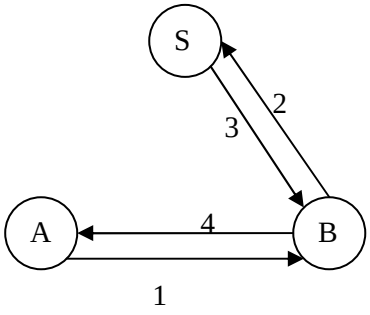
Proto zprávu pouze přepošle *Alici*, která ji dešifrovat dokáže a která mu tento nonce pošle zpět. *Útočník* právě získal nonce *Nb*, kterým se může *Bobovi* autentizovat.

3.2 Otway Rees

Autory tohoto protokolu jsou Dave Otway a Owen Rees, publikován byl roku 1997.[16] Jedná se o symetrický autentizační protokol, jehož cílem je distribuce sdíleného klíče. Server poskytne klíč pro společnou komunikaci, který si subjekty mezi sebou rozešlou.

Formální zápis protokolu

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

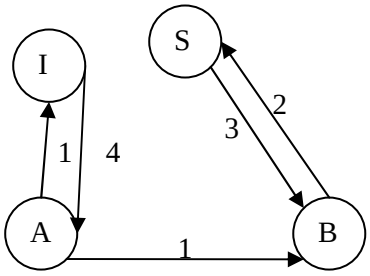


Obr. 3: Otway Rees protokol

Alice odešle *Bobovi* zprávu obsahující čerstvý nonce *M* a informace o subjektech, spolu se zprávou obsahující nonce *Na*, potřebný k autentizaci. Tuto část však dovede dešifrovat pouze *Server*. *Bob* přidá ke zprávě část obsahující *Nb*, kterou opět dovede dešifrovat pouze *Server*. *Server* takto získá *Na* a *Nb*, jejichž přeposláním se subjektům autentizuje, a které spolu s klíčem pro vzájemnou komunikaci odešle zpět mezi subjekty. *Bob* získá klíč pro společnou komunikaci a zbylou část zprávy odešle zpět *Alici*. Klíč byl práve doručen.

Útok na protokol

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



Obr. 4: Útok na Otway Rees protokol

Útok využívá podobnosti první zprávy odeslané Alicí se čtvrtou zprávou, kterou Alice očekává.

Útočník odposlechne první zprávu, obsahující nezašifrované hodnoty *M*, *A* a *B* a hodnotu zprávy $\{Na, M, A, B\}_{Ka}$. Této znalosti využije ve čtvrtém kroku, kdy *Alice* očekává zprávu podobnou první, ve které jsou však hodnoty *M*, *A* a *B* nahrazeny hodnotou klíče pro společnou komunikaci. Přeposláním první zprávy *Alice* přijme hodnotu klíče odpovídající známým *M*, *A* a *B*.

4 SRI Constraint Solver

Tato technika byla poprvé použita v roce 2001 Jonathanem Millenem a Vitalym Shmatikovskym, pracujícími ve firmě SRI International.[19] Svou práci prezentovali na konferenci ACM CCS(Computer and Comunications Security).

Nástroj SRI Constraint Solver využívá pro ověřování protokolů metody stavových automatů, proto u složitějších protokolů může dojít k explozi stavů. Nástroj je implementovaný v prologu, přičemž analýza bezpečností protokolu probíha zcela automaticky. Pro protokol je potřeba nadefinovat pouze pravidla chování a podmínky pro splnění. Po spuštění nástroj automaticky prochází jednotlivé stavy a testuje, zda se komunikace neúčastní také útočník. Pokud projde všechny stavy a nenalezne zprávy pocházející od útočníka, protokol je bezpečný.

Nástroj se vyskytuje v několika verzích, které jsou přiloženy na CD.

csolve

Původní verze nástroje, podporuje neasociativní zřetězení, nejen binární. Předpokládá, že útočník zná všechny veřejné klíče *pk(A)*, a obsahuje oboustrannou vyhledávací funkci *msk(A,B)*, tedy *msk(A,B)= msk(B,A)*. Útočník také zná všechny klíče *msk(e,A)* pro všechna A. Tato verze našla typovou chybu v jedné z verzí Needham-Schroeder-Lowe protokolu.

mysv

Rychlejší verze nástroje kterou vytvořili Ricarod Corin a Sandro Etalle z University of Twente v Nizozemí. Testuje řešitelnost podmínek postupně, jak jednotlivé uzly přibývají. Tato verze byla použita v praktické části.

csolvep

Program csolvep obsahuje oproti ostatním verzím navíc grafické zobrazení diagramů, včetně výpisu zobrazujícímu výslednou cestu. Tato verze využívá čtyři vbudované funkce SWI Prologu – term_to_atom, atom_lenght, gensym, reset_gensym. Použitelná je však pouze na specifikace, kde každá zpráva obsahuje hlavičku - [*odesílatel, příjemce, zpráva*].

4.1 Specifikace protokolu

Každý účastník komunikace protokolu je definován jako sekvence zpráv které přijímá a zpráv které odesílá. Tato sekvence je být specifikována výrokem ve tvaru *strand(role,A,B,…,[zprávy])*, kde *A,B,…* jsou parametry dané role a *zprávy* jsou uvedeny v následující syntaxi.

Jako příklad uveďme Needham Schroeder Public Key(NSPK)protokol, který je formálně popsán takto:

A -> B: [A,Na]pk(B)
B -> A: [Na, Nb]pk(A)
A -> B: [Nb]pk(B)

Role A by v prologu vypadala jako:

```
strand(roleA,A,B,Na,Nb,[
        send([A,Na]*pk(B)),
        recv([Na,Nb]*pk(A)),
        send(Nb*pk(B))
]).
```

Toto je schéma parametrického výroku, protože obsahuje parametry – v prologu značeny velkými písmeny. RoleA je konstantní označení role. Zřetězení znázorňují prologové závorky pro seznam [,] . Operátor * znamená, že na dané zprávy bylo použito asymetrické šifrování, klíčem který následuje. V případě že by protokol využíval symetrické šifrování, použili bychom před klíčem operátor +. Tečka na konci znamená v prologu že daný výraz je deklarovaný jako fakt. Další operátory, které ma program k dispozici jsou *sha()* pro hashovací funkci a znak / reprezentující podpis. Například výraz *X/pk(A)* značí X je podepsáno A. Tento způsob podpisu nelze obrátit.

Pro úplnou specifikaci výroku je potřeba si uvědomit, které z parametrů jsou pro dannou roli proměnné a při každém spuštění se budou lišit, čímž zjistíme jaké počáteční podmínky je potřeba roli přidat. Podmínkami rozumíme neměnné parametry, které vždy přicházejí ve formě *recv* zpráva. Do výroku pro roli A přidáme *recv([e,A,B])*. Nonce Na ani Nb neposíláme jako počáteční podmínky, jejich hodnoty se budou lišit při každém spuštění.

Obě role by tedy měly vypadat takto:

```
strand(roleA,A,B,Na,Nb,[
  recv([e,A,B]),
  send([A,Na]*pk(B)),
  recv([Na,Nb]*pk(A)),
  send(Nb*pk(B))
]).
```

```
strand(roleB,A,B,Na,Nb,[
  recv([A,Na]*pk(B)),
  send([Na,Nb]*pk(A)),
  recv(Nb*pk(B))
]).
```

4.2 Specifikace testovacího prostředí

Testovací prostředí je seznam výroku, formujících balíček. Balíček obsahuje všechny účastníky dané komunikace, které budeme chtít analyzovat.

V prologu je balíček reprezentován seznamem výroků, jejichž parametry jsou vyznačeny symbolickými konstantami. Balíček pro dřívě zmíněný NSPK by mohl být nadefinován jako:

```
nspk([Sa,Sb]):-
        strand(roleA,A,B,na,Nb,Sa),
        strand(roleB,a,b,Na,nb,Sb).
```

Konstanty jsou v prologu značeny malými písmeny, zde je *Na* v *roleA* konstanta, a v *roleB* jsou to *A, B* a *Nb*. Velkými písmeny značíme parametry které chceme přijmout z dané komunikace, tedy *A, B* a *Nb* v *roleA*, a *Na* v *roleB*.

Takovýto balíček můžem předložit nástroji. Ten by však zjistil jen prostupnost protokolu bez jakékoliv bezpečnostní kontroly. Z tohoto důvodu je potřeba balíček rozšířit, v závislosti na bezpečnostním testu který chceme provést. Kontrolu můžeme provádět dvěma způsoby, na důvěrnost a na autentizaci.

Důvěrnost

V našem příkladu by měly zůstat nonce utajeny. Pro zajištění důvěrnosti je potřeba nadefinovat nový testovací výrok pro roli *test*.

```
strand(test,X,[
        recv(X),
        send(stop)
]).
```

Tento výrok způsobí, že pokud se útočníkovi podaří zjistit X, pošle zprávu *stop* a tím ukončí analýzu. Nástroj následně vypíše cestu kterou se do tohoto bodu dostal.

Výrok přidáme na konec balíčku pro analýzu. Argument X nahradíme nonce který chceme testovat na utajení, v tomto případě Nb. Balíček pro zjištění důvěrnosti nonce Nb vypadá následovně:

```
nspk([Sa,Sb.St]):-
        strand(roleA,A,B,na,Nb,Sa),
        strand(roleB,a,b,Na,nb,Sb),
        strand(test,nb,St).
```

Autentizace

Způsob autentizace můžeme popsat myšlenou – “pokud se v daném spojení jeden z účastníků dostane na konec své role, pošle zprávu *doneX*, popř. *stop*, pak musí existovat stav, kdy se účastník shoduje na některých parametrech s jiným subjektem.”[2] Pro tyto případy se používá autentizační zpráva kdy se pošle celá daná role, pro roli A to může být:

```
send(roleA(A,B,Nb))
```

Zdrojový kód pro autentizaci by mohl vypadat takto:

```
strand(roleA,A,B,Na,Nb,[
  recv([A,B]),
  send([A,Na]*pk(B)),
  recv([Na,Nb]*pk(A)),
  send(Nb*pk(B)),
  send(roleA(A,B,Na,Nb))
]).

strand(test,X,[
  recv(X),
  send(stop)
]).
```

```
strand(roleB,A,B,Na,Nb,[
  recv([A,Na]*pk(B)),
  send([Na,Nb]*pk(A)),
  recv(Nb*pk(B)),
  send(doneB)
]).

nspk1([Sa,Sb,St],roleA(a,b,na,nb)):-
  strand(roleA,_A,_B,na,_Nb,Sa),
  strand(roleB,a,b,_Na,nb,Sb),
  strand(test,doneB,St).
```

4.3 Spuštění nástroje

Pro spuštění nástroje je potřeba mít nainstalovaný interpret jazyka prolog. Oba soubory, zdrojový soubor nástroje i zdrojový soubor protokolu je potřeba umístit do pracovního adresáře prologu. Po spuštění interpreta zadáme oba tyto soubory do příkazové řádky.

```
1 ?- [mysv,nspk].
% mysv compiled 0.00 sec, 14,384 bytes
% nspk compiled 0.00 sec, 3,084 bytes
true.
```

Příkaz, kterým spustíme analýzu má tvar *search(B,A)*, kde B je balíček a A je autentizační zpráva. V případě, že nás zajímá více test na důvěrnost než test na autentizaci, nahradíme A prazdným řetězcem []. Výsledný příkaz tedy bude mít tvar:

```
2 ?- nspk(B),search(B,[]).
```

Prolog nám následně vrátí *no* v případě že se daný balíček nepovedlo splnit nebo *yes* s výpisem dané cesty.

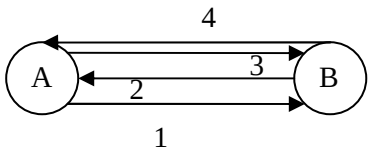
5 Implementované protokoly

5.1 Andrew Secure RPC

autor: Mahadev Satyanarayanan(1987)[8]

Cílem protokolu je vzájemná autentizace subjektů a distribuce čerstvého společného klíče pro komunikaci. Subjekty navzájem potvrdí svou identitu přeposláním upraveného nonce druhého účastníka. Po úspěšném ověření následuje přeposlání nového klíče pro spojení.

1. A › B : A, $\{Na\}_{Kab}$
2. B › A : $\{succNa, Nb\}_{Kab}$
3. A › B : $\{succNb\}_{Kab}$
4. B › A : $\{K'ab, N'b\}_{Kab}$



Obr. 5: Andrew Secure RPC protokol

Průběh protokolu

Trace:
recv([e, a, b])
send([[a, [na]+kab]])
recv([[a, [na]+kab]])
send([[inc(na), nb]+kab])
recv([[inc(na), nb]+kab])
send([[inc(nb)]+kab])
recv([[inc(na), nb]+kab])
recv([[inc(nb)]+kab])
send([[kab2, m]+kab])

Nalezený útok

Trace:
recv([e, _G445, _G448])
send([[_G445, [na]+kab]])
recv([[a, [na]+kab]])
send([[inc(na), nb]+kab])
recv([[inc(na), nb]+kab])
send([[inc(nb)]+kab])
recv([[inc(na), nb]+kab])
send(roleA(_G445, _G448, na, nb, kab, **inc(na), nb**))
recv([[inc(nb)]+kab])
send([[kab2, m]+kab])
send(doneB)

Nástroj nalezl útok při kontrole zda *roleA* přijala již všechny položky. Subjekt *A* odeslal zprávu $\{inc(Nb)\}_{Kab}$ a přešel do stavu očekávání příchodu zprávy $\{K'ab, M\}_{Kab}$. Místo ní však přijal přeposlanou zprávu obsahující $\{inc(Na), Nb\}_{Kab}$, následkem toho nahradí *K'ab* za *inc(Na)* a *M* za *Nb.*

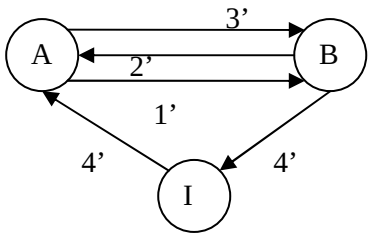
Publikovaný útok

autoři: Michael Burrows, Martin Abadi, Roger Needham(1989)[9]

Čtvrtá zpráva neobsahuje žádnou čerstvou položku, Alice tedy nemá možnost zjistit zda se jedná o nově poslanou zprávu. Útočník může odposlechnout tuto zprávu z některého z předcházejících spojení a později ji přeposlat Alici. Následkem toho Alice přijme neplatný klíč.

1. A › B : A, $\{Na\}_{Kab}$
2. B › A : $\{succNa, Nb\}_{Kab}$
3. A › B : $\{succNb\}_{Kab}$
4. B › A : $\{K'ab, N'b\}_{Kab}$

1'. A › B : A, $\{Ma\}_{Kab}$
2'. B › A : $\{succMa, Mb\}_{Kab}$
3'. A › B : $\{succMb\}_{Kab}$
4'. B › I_A : $\{K''ab, M'b\}_{Kab}$
4'. I_B › A : $\{K'ab, N'b\}_{Kab}$



Obr. 6: Útok na Andrew Secure RPC protokol

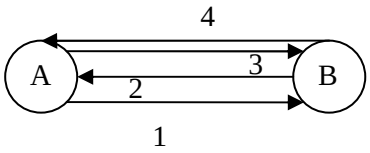
Nástroj nalezl publikovaný útok.

5.2 BAN modified Andrew Secure RPC

autoři: Michael Burrows, Martin Abadi, Roger Needham(1987)[9]

Opravená verze protokolu Andrew Secure RPC. Zajišťuje vzájemnou autentizaci subjektů a přeposlání nového klíče pro komunikaci. Chybu v předchozí verzi opravuje přidáním nonce Na do poslední zprávy, čímž zabraňuje subjektu A přijmout některou z dřívějších zpráv, kterou mu mohl útočník přeposlat.

1. A › B : A, $\{Na\}_{Kab}$
2. B › A : $\{succNa, Nb\}_{Kab}$
3. A › B : $\{succNb\}_{Kab}$
4. B › A : $\{K'ab, N'b, Na\}_{Kab}$



Obr. 7: BAN modified Andrew Secure RPC protokol

Průběh protokolu

Trace:
recv([e, a, b])
send([[a, [na]+kab]])
recv([[a, [na]+kab]])
send([[inc(na), nb]+kab])
recv([[inc(na), nb]+kab])
send([[inc(nb)]+kab])
recv([[inc(nb)]+kab])
send([[kab2, m, na]+kab])
recv([[kab2, m, na]+kab])

Publikovaný útok

Na tento protokol doposud nebyl publikován žádný útok.

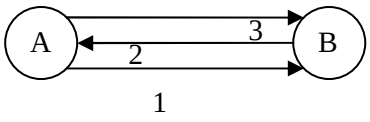
Nástroj nenalezl útok.

5.3 CCITT X.509

autoři: CCITT(1987)[10]

Protokol má za úkol zajistit vzájemnou autentizaci subjektů a postarat se o distribuci bezpečnostního certifikátu. Subjekty si navzájem vymění zprávy obsahující vlastní identitu, časová razítka a informace jako jsou identita účastníků, atributy klíče nebo bezpečnostní zásady. Přijetí těchto informací potvrdí subjekt A zpětným odeslánim nonce Nb.

1. A -> B : A, $\{Ta, Na, B, Xa, \{Ya\}_{Kb}\}_{Ka}^{-1}$
2. B -> A : B, $\{Tb, Nb, A, Na, Xb, \{Yb\}_{Ka}\}_{Kb}^{-1}$
3. A -> B : A, $\{Nb\}_{Ka}^{-1}$



Obr. 8: CCITT X.509 protokol

Průběh protokolu

Trace:

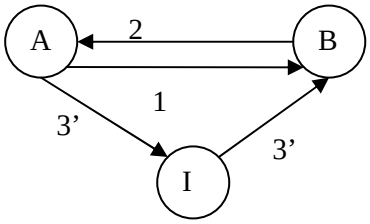
recv([e, a, b])
send([a, b, a, [ta, na, b, xa, [ya]*pk(b)]*pk(a)])
recv([a, b, a, [ta, na, b, xa, [ya]*pk(b)]*pk(a)])
send([b, a, b, [tb, nb, a, na, xb, [yb]*pk(a)]*pk(b)])
recv([b, a, b, [tb, nb, a, na, xb, [yb]*pk(a)]*pk(b)])
send([a, b, a, [nb]*pk(a)])
recv([a, b, a, [nb]*pk(a)])

Publikovaný útok

autor: Michael Burrows, Martin Abadi, Roger Needham (1989)[9]

Nebezpečí u tohoto protokolu nastává v případě, že Bob nezkontroluje časové razítko v první zprávě. Útok je založen na vytvoření dvou paralelních spojení, v prvním spojení útočník odposlouchává a přeposíla komunikaci mezi Alicí a Bobem. V druhém spojení tyto zprávy využívá ke komunikaci s Alicí, čímž ji přesvědčí že kommunikuje se Bobem. Útočníkovi je doručen nonce Nb.

1. A › I_B : A, $\{Ta, Na, B, Xa, \{Ya\}_{Kb}\}_{Ka}^{-1}$
1. I_A › B : A, $\{Ta, Na, B, Xa, \{Ya\}_{Kb}\}_{Ka}^{-1}$
2. B › I_A : B, $\{Tb, Nb, A, Na, Xb, \{Yb\}_{Ka}\}_{Kb}^{-1}$
1'. A › I : A, $\{Ta', Na', C, Xa', \{Ya'\}_{Ki}\}_{Ka}^{-1}$
2'. I › A : I, $\{Ti, Nb, A, N'a, Xi, \{Yi\}_{Ka}\}_{Ki}^{-1}$
3'. A › I : A, $\{Nb\}_{Ka}^{-1}$
3'. I_A › B : A, $\{Nb\}_{Ka}^{-1}$



Obr. 9: Útok na CCITT X.509 protokol

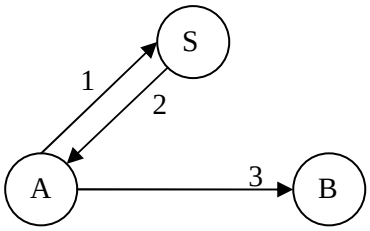
Nástroj nenalezl útok.

5.4 Denning-Sacco shared key

autoři: Dorothy E. Denning, Giovanni Maria Sacco(1981)[11]

Protokol zajišťuje distribuci klíče pro komunikaci, který je poskytnut na požádání důvěryhodným serverem. Komunikaci s ním obstarává pouze subjekt A, který obdrží klíč pro obě zúčastněné strany a ten rozešle.

1. A › S : A, B
2. S › A : {B, Kab, T, {Kab, A, T}Kbs}$_{Ka}$
3. A › B : {Kab, A, T}$_{Kb}$



Obr. 10: Denning-Sacco shared key protokol

Průběh protokolu

Trace:	
recv([e, a, b])	send([[b, kab, t, [kab, a, t]+sk(b)]+sk(a)])
send([a, b])	recv([[b, kab, t, [kab, a, t]+sk(b)]+sk(a)])
recv([a, b])	send([[kab, a, t]+sk(b)])
	recv([[kab, a, t]+sk(b)])

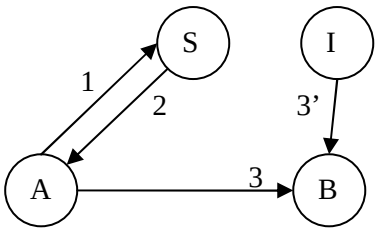
Publikovaný útok

autor: Gavin Lowe(1997)[12]

Útok na tento protokol spočívá v odposlechnutí a přeposlání poslední zprávy. Útočník odposlechne zprávu Alice obsahující klíč pro vzájemnou komunikaci, kterou odešle v době platnosti časového razítka Bobovi. Bob nabude dojmu že se Alice pokouší o vytvoření nového sdíleného klíče a přijme jej.

1. A › S : A, B
2. S › A : {B, Kab, T, {Kab, A, T}Kbs}Kas
3. A › B : {Kab,A, T}Kbs

3‘. I_A › B : {Kab,A, T}Kbs



Obr. 11: Útok na Denning-Sacco shared key protokol

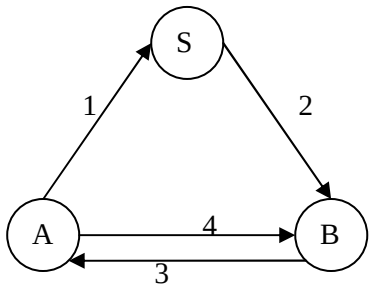
Nástroj nenalezl útok.

5.5 Kao Chow Authentication v.1

autoři: I Long Kao and Randy Chow(1995)[13]

Cílem protokolu je distribuce společného klíče a následná vzájemná autentizace subjektů. Klíč je poskytnut na vyžádání důvěryhodným serverem a následně poslán v zašifrovaných zprávách které jsou určeny jednotlivým subjektům.

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



Obr. 12: Kao Chow Authentication v.1 protokol

Průběh protokolu

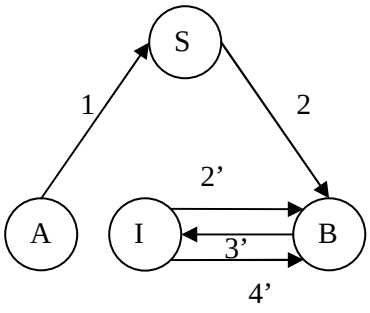
Trace:
recv([e, a, b])
send([a, b, na])
recv([a, b, na])
send([[a, b, na, kab]+sk(a), [a, b, na, kab]+sk(b)])
recv([_G517, [a, b, na, kab]+sk(b)])
send([_G517, [na]+kab, nb])
recv([[a, b, na, kab]+sk(a), [na]+kab, nb])
send([[nb]+kab])
recv([[nb]+kab])

Publikovaný útok

autor: I Lung Kao and Randy Chow(1995)[13]

Pokud se útočníkovi podaří zjistit starší Kab, může této znalosti využít pro vytvoření spojení s Bobem. Útočník, vydávající se za server, přepošle Bobovi druhou zprávu. Pro další komunikaci se útočník vydává za Alici, čímž získá nonce N'b, který zašifruje starým klíčem a odešle zpět Bobovi. Bob věří že navázal spojení s Alicí.

1. A › S : A, B, Na
2. S › B : $\{A, B, Na, Kab\}_{Ka}$, $\{A, B, Na, Kab\}_{Kb}$
2'. I_S › B : $\{A, B, Na, Kab\}_{Ka}$, $\{A, B, Na, Kab\}_{Kb}$
3'. B › I_A : $\{A, B, Na, Kab\}_{Ka}$, $\{Na\}_{Kab}$, N'b
4'. I_A › B : $\{N'b\}_{Kab}$



Obr. 13: Útok na Kao Chow Authentication v.1 protokol

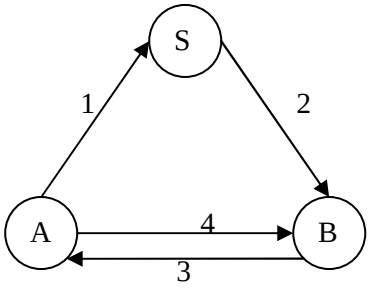
Nástroj nenalezl útok.

5.6 Kao Chow Authentication v.2

autoři: I Long Kao and Randy Chow(1995)[13]

Jedná se o modifikovanou verzi protokolu Kao Chow Authentication v.1. Cílem protokolu je distribuce společného klíče a následná autentizace subjeků. Chybu svého předchůdce opravuje přidáním jednorázového klíče Kt, jehož jedinou úlohou je zajistit bezpečnější distribuci společného klíče.

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



Obr. 14: Kao Chow Authentication v.2 protokol

Průběh protokolu

Trace:

recv([e, a, b])
send([a, b, na])
recv([a, b, na])
send([[a, b, na, kab, kt]+sk(a), [a, b, na, kab, kt]+sk(b)])
recv([_G844, [a, b, na, kab, kt]+sk(b)])
send([b, _G844, [na, kab]+kt, nb])
recv([b, [a, b, na, kab, kt]+sk(a), [na, kab]+kt, nb])
send([[nb, kab]+kt])

Publikovaný útok

Na tento protokol nebyl doposud publikován žádný útok.

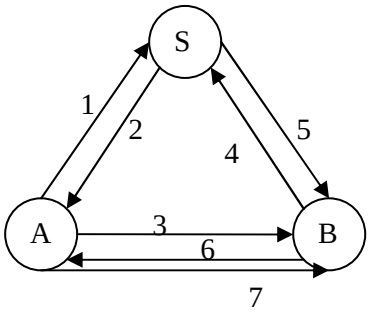
Nástroj nenalezl útok.

5.7 Needham-Schroeder Public Key

autoři: Roger Needham, Michael Schroeder(1978)[14]

Protokol zajišťuje vzájemnou autentizaci jednotlivých subjektů, k čemuž využívá důvěryhodný server a asymetrické šifrování. Účastníci komunikace vyžádají od serveru klíče pro komunikaci, s jejichž pomocí se vzájemně autentizují.

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



Obr. 15: Needham-Schroeder Public Key protocol

Průběh protokolu

Trace:
recv([e, a, e])
send([a, b])
recv([a, b])
send([pk(b), b]*pk(s))
recv([pk(b), b]*pk(s))
send([na, a]*pk(b))
recv([na, a]*pk(b))
send([b, a])
recv([b, a])
send([pk(a), a]*pk(s))
recv([pk(a), a]*pk(s))
send([na, nb]*pk(a))
recv([na, nb]*pk(a))
send([nb]*pk(b))
recv([nb]*pk(b))

Nalezený útok

Trace:
recv([e, a, e])
send([a, e])
recv([pk(e), e]*pk(_G444))
send([na, a]*pk(e))
recv([na, a]*pk(b))
send([b, a])
recv([a, b])
send([pk(b), b]*pk(s))
recv([b, a])
send([pk(a), a]*pk(s))
recv([pk(a), a]*pk(s))
send([na, nb]*pk(a))
recv([na, nb]*pk(a))
send([nb]*pk(e))
recv([nb]*pk(b))
recv(nb)

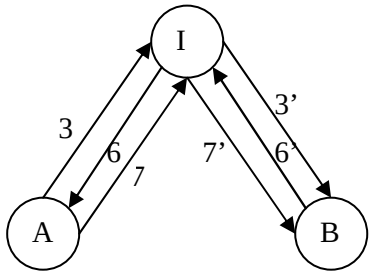
Nalezený útok spočívá v nálezu odesílaných zpráv zašifrovaných klíčem *pk(e)*, čili zpráv určených pro útočníka, které jsou následně přijímány Bobem. Z toho můžeme soudit, že do komunikace se zapojila třetí strana, které byl prozrazen nonce *Nb* nutný pro autentizaci.

Publikovaný útok

autor: Gavin Lowe(1995)[15]

Útočník vytvoří paralelní spojení. V prvním spojení útočník legitimně komunikuje se subjektem A, ve druhém se vydává za subjekt A. Útočník přijímá zprávy od obou subjektů, které následně přeposílá ve druhém spojení. Na konci spojení získá útočník nonce Nb, který přepošle subjektu B a tím se mu autentizuje.

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



Obr. 16: Útok na Needham-Schroeder Public Key protokol

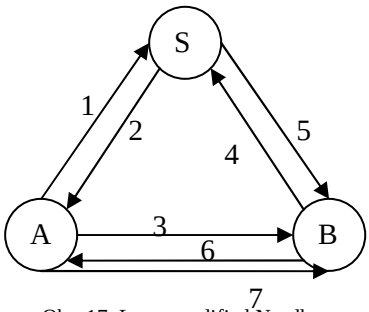
Nástroj nalezl publikovaný útok.

5.8 Lowe modified Needham-Schroeder Public Key

autor: Gavin Lowe(1995)[15]

Verze protokolu Needham-Schroeder Public Key, opravující předchozí útok. Stará se o vzájemnou autentizaci jednotlivých subjektů za využití důveryhodného serveru a asymetrické kryptografie. Pro zabránění útoku byla do šesté zprávy přidána identita subjektu B.

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



Obr. 17: Lowe modified Needham-Schroeder Public Key protocol

Průběh protokolu

Trace:	send([b, a])
recv([e, a, b])	recv([b, a])
send([a, b])	send([pk(a), a]*pk(s))
recv([a, b])	recv([pk(a), a]*pk(s))
send([pk(b), b]*pk(s))	send([na, nb, b]*pk(a))
recv([pk(b), b]*pk(s))	recv([na, nb, b]*pk(a))
send([na, a]*pk(b))	send([nb]*pk(b))
recv([na, a]*pk(b))	recv([nb]*pk(b))

Publikovaný útok

Na tento protokol doposud nebyl publikován žádný útok.

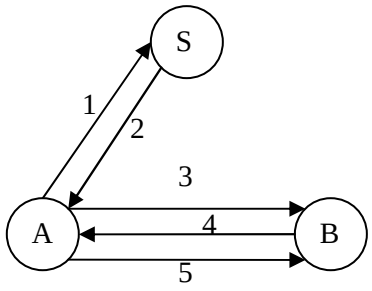
Nástroj nenalezl útok.

5.9 Needham-Schroeder Symetric Key

autoři: Roger Needham, Michael Schroeder(1978)[14]

Cílem protokolu je vzájemná autentizace jednotlivých subjektů. Subjekt dostane na vyžádání od serveru klíč pro vzájemnou komunikaci, který následně přepošle druhému subjektu. Ten mu vrátí hodnotu nonce Nb zašifrovanou jeho klíčem, jejíž upravenou hodnotu očekává jako odpověd a důkaz autentizace.

1. A › S : A, B, Na
2. S › A : {Na, B, Kab, {Kab, A}$_{Kb}$}$_{Ka}$
3. A › B : {Kab,A}$_{Kb}$
4. B › A : {Nb}$_{Kab}$
5. A › B : {dec(Nb)}$_{Kab}$



Obr. 18: Needham-Schroeder Symmetric Key protocol

Průběh protokolu

Trace:	send([[kab, a]+sk(b)])
recv([e, a, b])	recv([[kab, a]+sk(b)])
send([a, b, na])	send([[nb]+kab])
recv([a, b, na])	recv([[nb]+kab])
send([[na, b, kab, [kab, a]+sk(b)]+sk(a)])	send([[dec(nb)]+kab])
recv([[na, b, kab, [kab, a]+sk(b)]+sk(a)])	

Publikovaný útok

autoři: Dorothy E. Denning, Giovanni M. Sacco(1981)[11]

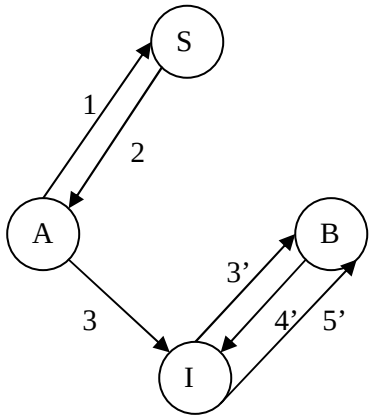
Útok na protokol spočívá ve stejném principu jako u protokolu Kao Chow Authentication v.1. V případě, že útočník zjistí klíč Kab, vytvoří spojení s Bobem, kterému přepošle zprávu dřívě odpolechnutou od Alice. Bob pošle zprávu obsahující nonce Nb, kterou útočník za použití známého Kab dešifruje a upravenou hodnotu odešle zpět.

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

3'. $I_A \rightarrow B$: $\{Kab,A\}_{Kb}$

4'. $B \rightarrow I_A$: $\{Nb\}_{Kab}$

5'. $I_A \rightarrow B$: $\{dec(Nb)\}_{Kab}$



Obr. 19: Útok na Needham-Schroeder Symmetric Key protocol

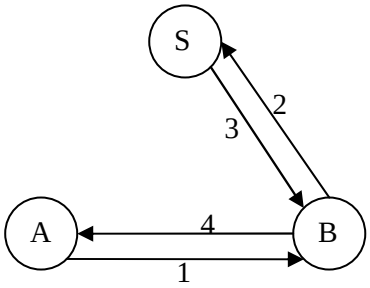
Nástroj nenalezl útok.

5.10 Otway Rees

autoři: Dave Otway, Owen Rees (1997)[16]

Tento protokol má za úkol distribuci sdíleného klíče, pocházejícího od důvěryhodného serveru. Na žádost subjektů server vytvoří klíč pro vzájemnou komunikaci, který pošle ve zprávě, kterou si subjekty přepošlou.

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



Obr. 20: Otway Rees protocol

Průběh protokolu

Trace:
recv([e, a, b])
send([a, b, [m, a, b, [na, m, a, b]+sk(a)]])
recv([a, b, [m, a, b, _G501]])
send([b, s, [m, a, b, _G501, [nb, m, a, b]+sk(b)]])
recv([b, s, [m, a, b, [na, m, a, b]+sk(a), [nb, m, a, b]+sk(b)]])
send([s, b, [m, [na, kab]+sk(a), [nb, kab]+sk(b)]])
recv([b, a, [m, [na, kab]+sk(a)]])
recv([s, b, [m, _G567, [nb, kab]+sk(b)]])
send([b, a, [m, _G567]])

Nalezený útok

Trace:
recv([e, a, b])
send([a, b, [m, a, b, [na, m, a, b]+sk(a)]])
recv([a, b, [m, a, b, _G539]])
send([b, s, [m, a, b, _G539, [nb, m, a, b]+sk(b)]])
recv([b, s, [m, a, b, [na, m, a, b]+sk(a), [nb, m, a, b]+sk(b)]])
send([s, b, [m, [na, kab]+sk(a), [nb, kab]+sk(b)]])
recv([b, a, [m, [na, kab]+sk(a)]])
send(doneA)

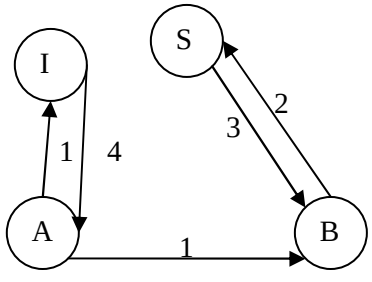
Subjekt *A* odeslal zprávu pro *B* a přešel do stavu očekávání zprávy $M, \{Na, Kab\}_{Ka.}$ Než však subjekt *B* tuto zprávu odeslal, *A* ji přijalo od někoho, kdo se za subjekt *B* vydával, tedy útočníka.

Publikovaný útok

autor: John Clark and Jeremy Jacob (1997)[17]

Na tento protokol lze zaútočit přeposláním první zprávy. Útočník odpolechne první zprávu komunikace, ze které získá nezašifrované hodnoty M, A a B. S touto znalostí může ve čtvrtém kroku přeposlat Alici část první zprávy. Alice však očekává Kab, hodnoty M, A a B si subjekt tedy přeloží jako nový klíč Kab.

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



Obr. 21: Útok na Otway Rees protokol

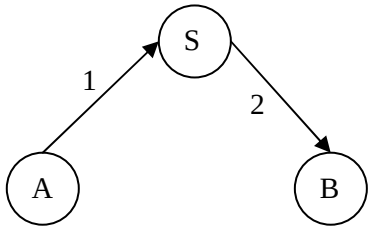
Nástroj nalezl publikovaný útok.

5.11 Wide Mouthed Frog

autor: Michael Burrows (1989)[9]

Protokol zajišťuje distribuci klíče pro společnou komunikaci. Tvůrcem klíče je jeden ze subjektů a distribuce probíhá přes důvěryhodný server. Každý klíč má omezenou dobu platnosti.

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



Obr. 22: Wide Mouthed Frog protocol

Průběh protokolu

Trace:
recv([e, a, b])
send([a, s, [a, [ta, b, kab]+sk(a)]])
recv([a, s, [a, [ta, b, kab]+sk(a)]])
send([s, b, [ts, a, kab]+sk(b)])
recv([s, b, [ts, a, kab]+sk(b)])

Nalezený útok

Trace:
recv([e, b, a])
send([b, s, [b, [ta, a, _G472]+sk(b)]])
recv([s, b, [ta, a, _G472]+sk(b)])
send(doneB)

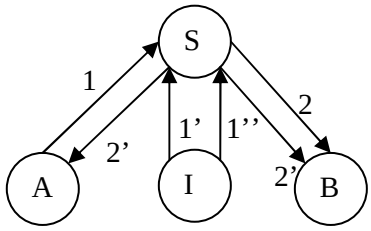
Subjekt B poslal zprávu serveru, tato komunikace však není součástí daného protokolu, byla provedena neznámou třetí stranou – útočníkem.

Publikovaný útok

autor: Gavin Lowe (1997)[12]

Útočník odposlechne komunikaci mezi subjekty Alicí a Bobem. Ná základě této znalosti může přeposílat serveru přijaté zprávy, vydávajíce se za jeden ze subjektů. Po přijetí takovéto zprávy server prodlouží platnost časového razítka i daného klíče.

1. A › S : A, $\{Ta, B, Kab\}_{Ka}$
2. S › B : $\{Ts, A, Kab\}_{Kb}$
1'. I_B › S : B, $\{Ts, A, Kab\}_{Kb}$
2'. S › A : $\{T's, B, Kab\}_{Ka}$
1''. I_A › S : A, $\{T's, B, Kab\}_{Kb}$
2''. S › B : $\{T''s, A, Kab\}_{Kb}$
....



Obr. 23: Útok na Wide Mouthed Frog protocol

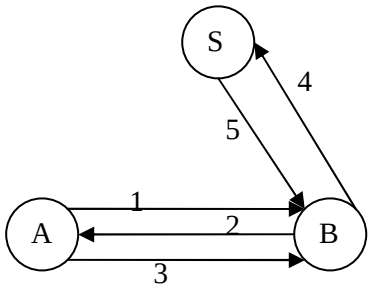
Nástroj nalezl publikovaný útok.

5.12 Woo Lam Pi

autoři: Simon S. Lam, Thomas Y. C. Woo(1994)[18]

Cílem protokolu je vzájemná autentizace účastníků komunikace. Ověření identity obou účastníků zajišťuje třetí strana, důvěryhodný server.

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



Obr. 24: Woo Lam Pi protocol

Průběh protokolu

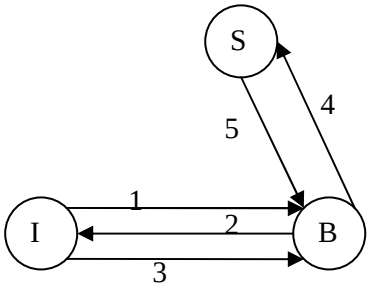
Trace:	send([a, b, [[nb]+sk(a)]])
recv([e, a, b])	recv([a, b, [[[nb]+sk(a)]]])
send([a, b, [a]])	send([b, s, [[a, [[nb]+sk(a)]]+sk(b)]])
recv([a, b, [a]])	recv([b, s, [[a, [[nb]+sk(a)]]+sk(b)]])
send([b, a, [nb]])	send([s, b, [[nb]+sk(b)]])
recv([b, a, [nb]])	recv([s, b, [[nb]+sk(b)]])

Publikovaný útok

autoři: John Clark, Jeremy Jacob(1997)[17]

Útočník se vytvoří dvě na sobě nezávislé relace. V první relaci útočník pouze odposlouchává, aby zjistil typ zprávy který je od něj v druhé relaci očekáván. Takto postupuje do třetího kroku, kdy odešle Bobovi hodnotu G, která odpovídá hodnotě $\{Nbi\}_{Ki}$. Tím docílí toho, že hodnota Nb se bude rovnat hodnotě $\{Nbi\}_{Ki}$.

1’. I_A › B : I
1. A › B : A
2’. B › I_A : Nbi
2. B › A : Nb
3’. I_A › B : G
3. A › B : $\{Nbi\}_{Ki}$
4’. B › S : $\{I, G\}_{Kb}$
4. B › S : $\{I, \{Nbi\}_{Ki}\}_{Kb}$
5. S › B : $\{Nb\}_{Kb}$



Obr. 25: Útok na Woo Lam Pi protocol

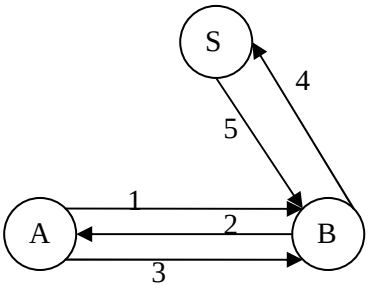
Nástroj nenalezl útok.

5.13 Woo Lam Pi 1

autoři: Simon S. Lam, Thomas Y. C. Woo(1994)[18]

Protokol zajišťuje vzájemnou autentizaci subjektů za použití třetí, důvěryhodné strany. Součástí komunikace je i výměna identit subjektu A a B.

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



Obr. 26: Woo Lam Pi 1 protocol

Průběh protokolu

Trace:
recv([e, a, b])
send([a, b, [a]])
recv([a, b, [a]])
send([b, a, [nb]])
recv([b, a, [nb]])
send([a, b, [[a, b, nb]+sk(a)]])
recv([a, b, [[[a, b, nb]+sk(a)]]])
send([b, s, [[a, b, [[a, b, nb]+sk(a)]]+sk(b)]])
recv([b, s, [[a, b, [[a, b, nb]+sk(a)]]+sk(b)]])
send([s, b, [[a, b, nb]+sk(b)]])
recv([s, b, [[a, b, nb]+sk(b)]])

Nalezený útok

Trace:
recv([e, _G434, _G437])
send([a, b, [_G434]])
recv([b, a, [_G471]])
send([a, b, [[_G434, _G437,
_G471]+sk(_G434)]])
send([roleA(_G434, _G471)])
recv([a, b, [a]])
send([b, a, [nb]])
recv([a, b, [nb]])
send([b, s, [[a, b, nb]+sk(b)]])
recv([s, b, [[a, b, nb]+sk(b)]])
send(doneB)

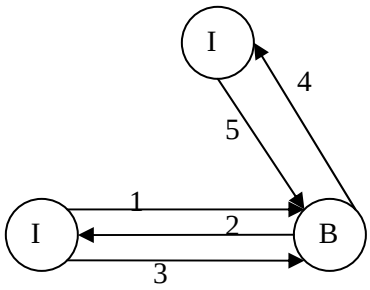
Poté, co byla *roleA* dokončena, probíhá další komunikace označující *A* jako příjemce i odesílatele, jedná se o práci *útočníka*.

Publikovaný útok

autoři: John Clark, Jeremy Jacob(1997)[17]

Útočník, vydávající se za Alici vytvoří spojení s Bobem. Ve třetím kroku místo $\{A, B, Nb\}_{Ka}$ odešle pouze odposlechnutý Nb, čímž přiměje Boba odeslat serveru zprávu, kterou bude sám v následujícím kroku očekávat. Útočník vydávající se za server tuto zprávu odposlechne.

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



Obr. 27: Útok na Woo Lam Pi 1 protocol

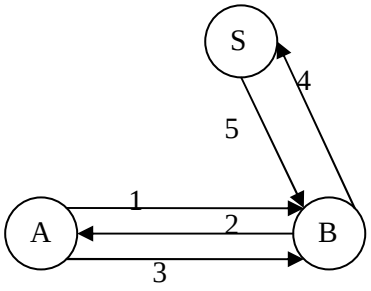
Nástroj nalezl publikovaný útok.

5.14 Woo Lam Pi 2

autoři: Simon S. Lam, Thomas Y. C. Woo(1994)[18]

Protokol zajišťuje vzájemnou autentizaci subjektů za použití třetí, důvěryhodné strany. Součástí komunikace je i poslání identity subjektu A.

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



Obr. 28: Woo Lam Pi 2 protocol

Průběh protokolu

Trace:
recv([e, a, b])
send([a, b, [a]])
recv([a, b, [a]])
send([b, a, [nb]])
recv([b, a, [nb]])
send([a, b, [[a, nb]+sk(a)]])
recv([a, b, [[[a, nb]+sk(a)]]])
send([b, s, [[a, [[a, nb]+sk(a)]]+sk(b)]])
recv([b, s, [[a, [[a, nb]+sk(a)]]+sk(b)]])
send([s, b, [[a, nb]+sk(b)]])
recv([s, b, [[a, nb]+sk(b)]])

Nalezený útok

Trace:
recv([e, _G434, _G437])
send([a, b, [_G434]])
recv([b, a, [_G471]])
send([a, b, [[_G434, _G471]+sk(_G434)]])
send([roleA(_G434, _G471)])
recv([a, b, [a]])
send([b, a, [nb]])
recv([a, b, [nb]])
send([b, s, [[a, nb]+sk(b)]])
recv([s, b, [[a, nb]+sk(b)]])
send(doneB)

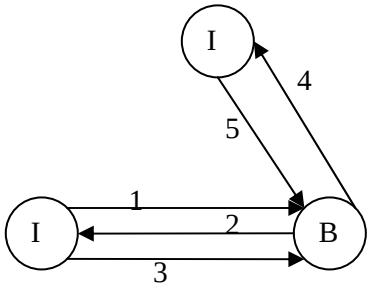
Stejný druh útoku jako v předchozím případě. Poté, co byla *roleA* dokončena, probíhá další komunikace označující *A* jako příjemce i odesílatele, jedná se o práci *útočníka*.

Publikovaný útok

autoři: John Clark, Jeremy Jacob(1997)[17]

Útočník, vydávající se za Alici vytvoří spojení s Bobem. Ve třetím kroku místo $\{A, Nb\}_{Ka}$ odešle pouze odposlechnutý Nb, čímž přiměje Boba odeslat serveru zprávu, kterou bude sám v následujícím kroku očekávat. Útočník vydávající se za server tuto zprávu odposlechne.

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



Obr. 29: Útok na Woo Lam Pi 1 protocol

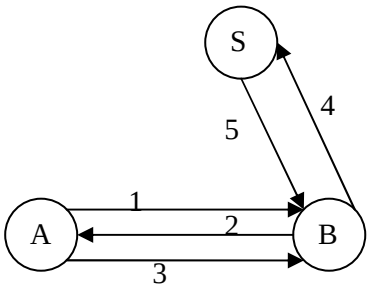
Nástroj nalezl publikovaný útok.

5.15 Woo Lam Pi 3

autoři: Simon S. Lam, Thomas Y. C. Woo(1994)[18]

Protokol zajišťuje vzájemnou autentizaci subjektů za použití třetí, důvěryhodné strany. Součástí posílaných informací jsou i identita subjektu A.

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



Obr. 30: Woo Lam Pi 3 protocol

Průběh protokolu

Trace:
recv([e, a, b])
send([a, b, [a]])
recv([a, b, [a]])
send([b, a, [nb]])
recv([b, a, [nb]])
send([a, b, [[nb]+sk(a)]])
recv([a, b, [[[nb]+sk(a)]]])
send([b, s, [[a, [[nb]+sk(a)]]+sk(b)]])
recv([b, s, [[a, [[nb]+sk(a)]]+sk(b)]])
send([s, b, [[a, nb]+sk(b)]])
recv([s, b, [[a, nb]+sk(b)]])

Nalezený útok

Trace:
recv([e, _G434, _G437])
send([a, b, [_G434]])
recv([b, a, [_G471]])
send([a, b, [[_G471]+sk(_G434)]])
send([roleA(_G434, _G471)])
recv([a, b, [a]])
send([b, a, [nb]])
recv([a, b, [nb]])
send([b, s, [[a, nb]+sk(b)]])
recv([s, b, [[a, nb]+sk(b)]])
send(doneB)

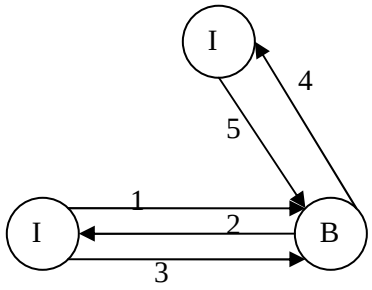
Útok jako v předchozích případech. Poté, co byla *roleA* dokončena, probíhá další komunikace označující *A* jako příjemce i odesílatele, jedná se o práci *útočníka*.

Publikovaný útok

autoři: John Clark, Jeremy Jacob(1997)[17]

Útočník, vydávající se za Alici vytvoří spojení s Bobem. Ve třetím kroku místo $\{Nb\}_{Ka}$ odešle pouze odposlechnutý Nb, čímž přiměje Boba odeslat serveru zprávu, kterou bude sám v následujícím kroku očekávat. Útočník vydávající se za server tuto zprávu odposlechne.

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



Obr. 31: Útok na Woo Lam Pi 1 protocol

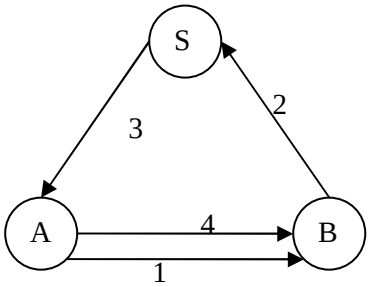
Nástroj nalezl publikovaný útok.

5.16 Yahalom

autor: Raphael Yahalom(1988)[9]

Protokol Yahalom zajišťuje distribuci klíče pro společnou komunikaci a následnou autentizaci subjektů. Klíč je poskytnut důvěryhodným serverem jednomu ze subjektů, který jej rozešle.

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



Obr. 32: Yahalom protocol

Průběh protokolu

Trace:

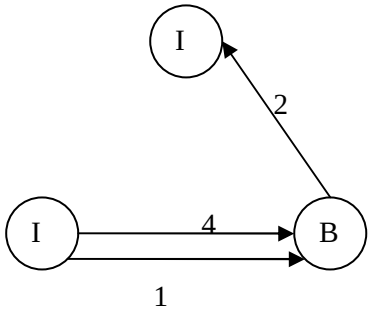
recv([e, a, b])
send([a, b, [a, na]])
recv([a, b, [a, na]])
send([b, s, [b, [a, na, nb]+sk(b)]])
recv([b, s, [b, [a, na, nb]+sk(b)]])
send([s, a, [[b, kab, na, nb]+sk(a), [a, kab]+sk(b)]])
recv([s, a, [[b, kab, na, nb]+sk(a), _G459]])
send([a, b, [_G459, [nb]+kab]])
recv([a, b, [[a, kab]+sk(b), [nb]+kab]])

Publikovaný útok

autoři: John Clark, Jeremy Jacob (1997)[17]

Útočník, vydávající se za subjekt A pošle subjektu B zprávu obsahující známé hodnoty A a Na. B na ni zareaguje odesláním zprávy serveru, který v tomto případě představuje také útočník. Z této zprávy útočník získá hodnoty B a Nb. V posledním kroku, kdy subjekt B očekává hodnotu klíče Kab mu útočník prepošle část druhé zprávy, ze které B

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



Nástroj nenalezl útok.

Obr. 33: Útok na Yahalom protocol

6 Porovnání výsledků

Níže uvedená tabulka obsahuje seznam analyzovaných protokolů, způsob jakým byl každý protokol testován, zda byl na protokol publikován útok a jestli nástroj tento útok dokázal nalézt.

Název protokolu	Testována		Publikovaný útok	Nalezený útok
	Autentizace	Důvěrnost		
Andrew Secure RPC	✓		✓	✓
BAN modified Andrew Secure RPC	✓		✗	✗
CCITT X.509		✓	✓	✗
Denning-Sacco shared key	✓		✓	✗
Kao Chow Authentication v.1	✓		✓	✗
Kao Chow Authentication v.2	✓		✗	✗
Needham-Schroeder Public Key		✓	✓	✓
Lowe modified Needham-Schroeder Public Key		✓	✗	✗
Needham-Schroeder Symmetic Key	✓		✓	✗
Otway Rees	✓		✓	✓
Wide Mouthed Frog	✓		✓	✓
Woo Lam Pi	✓		✓	✗
Woo Lam Pi 1	✓		✓	✓
Woo Lam Pi 2	✓		✓	✓
Woo Lam Pi 3	✓		✓	✓
Yahalom	✓		✓	✗

Tab. 1: Srovnání výsledků

7 Závěr

Cílem práce bylo prostudování a zpracování materiálů zabývajicích se problematikou bezpečnosti v počítačové síti.

Bylo potřeba nastudovat jednotlivé bezpečnostní protokoly, jakým způsobem fungují, k čemu se využívají, jaké mají slabiny a silné stránky. Protokoly zvolené pro tuto práci jsem vybíral především podle využití a útoků, které na ně existují. Mnohdy se jedná o více verzí daného protokolu, kde novější verze opravuje bezpečnostní riziko verze starší.

Dalším krokem byl výběr nástroje pro analýzu. Po prostudování materiálů k několika nástrojům, a jejich odzkoušení v praxi, jsem vybral nástroj SRI Constraint Solver. Nástroj mě zaujal svou jednoduchostí a efektivností. Jazyk prolog, kterého nástroj využívá, pro mě nebyl neznámý, a popis protokolů v něm je intuitivní. Problém byl pouze v dokumentaci k danému nástroji, která je velice strohá a některé postupy je potřeba si domyslet.

V praktické části jsou uvedeny výstupy přímo z tohoto programu pro jednotlivé protokoly. Nástroj nenašel všechny publikované útoky, což se dalo předpokládat. Tento projekt by se dal rozšířit přidáním dalších bezpečnostních protokolů k analýze.

Literatura

[1] Ptáček M. *Specifikační jazyky a nástroje pro analýzu a verifikaci bezpečnostních protokolů*, diplomová práce, Brno, FIT VUT v Brně, 2007.

[2] *Constraint Solving for cryptographic protocol analysis* [online]. 2009 [cit. 2009-05-15]. < http://www.homepages.dsu.edu/malladis/research/ConSolv/Webpage/ > .

[3] *Constraint Solving in Prolog* [online]. 2003 [cit. 2009-05-15]. < http://www.csl.sri.com/users/millen/capsl/constraints.html >

[4] AMD, *World internet usage* [online]. 2009 [cit. 2009-05-15]. < http://www.50x15.com/en-us/internet_usage.aspx >

[5] *Design and Analysis of Security Protocols* [online]. 2004 [cit. 2009-05-15]. < http://www.cs.utexas.edu/~shmat/courses/cs395t_fall04/cs395t_ref.html >

[6] Millen J. *Using the Constraint Solver*, [online]. 2003 [cit. 2009-05-15]. < http://www.zisc.ethz.ch/events/FS2003PDFs/M4.pdf >

[7] *Security Protocols Open Repository* [online]. 2003 [cit. 2009-05-15]. < http://www.lsv.ens-cachan.fr/Software/spore/table.html >

[8] Satyanarayanan M., *Integrating security in a large distributed system.* ACM Transactions on Computer Systems, 7(3):247--280, 1989

[9] Burrows M, Abadi M., Needham R. *A logic of authentication.* Technical Report 39, Digital Systems Research Center, february 1989.

[10] CCITT. *The directory authentification framework.* Draft Recommendation X.509, 1987. Version 7.

[11] Denning D., Sacco G., *Timestamps in key distributed protocols.* Communication of the ACM, 24(8):533--535, 1981.

[12] Lowe G., *A family of attacks upon authentication protocols.* Technical Report 1997/5, Department of Mathematics and Computer Science, University of Leicester, 1997.

[13] Kao I. L., Chow R., *An efficient and secure authentication protocol using uncertified keys.* Operating Systems Review, 29(3):14--21, 1995.

[14] Needham R., Schroeder M., *Using encryption for authentification in large networks of computers*. Communications of the ACM, 21(12), December 1978.

[15] Lowe G., *An attack on the Needham-Schroeder public key authentication protocol*. Information Processing Letters, 56(3):131--136, November 1995.

[16] Otway D., Rees O., *Efficient and timely mutual authentication.* Operating Systems Review, 21(1):8--10, 1987.

[17] Clark J., Jacob J., *A survey of authentication protocol literature : Version 1.0.*, November 1997.

[18] Woo T. Y. C., Lam S. S., *A lesson on authentication protocol design.* Operating Systems Review, 1994.

[19] Millen J., Shmatikov V. *Constraint Solving for Bounded-Process Cryptographic Protocol Analysis*, SRI International, 2001.

[20] Wikipedia [online] [cit. 2009-05-15]. < http://en.wikipedia.org/wiki/ >

Přílohy

Příloha 1. CD se zdrojovými soubory