Katedra Informatiky
Fakulta Matematiky, Fyziky a Informatiky
Univerzita Komenského, Bratislava

Multicast Security

Diplomová Práca

Dušan Baník

Odbor: Informatika 9.2.1
Vedúci: Doc. RNDr. Martin Stanek, PhD.

Bratislava, 2009

Čestne prehlasujem, že som túto diplomovú prácu vypracoval samostatne s použitím citovaných zdrojov.

. . . . . .

Ďakujem vedúcemu mojej diplomovej práce docentovi Martinovi Stanekovi za pomoc pri hľadaní zaujímavej témy, ako aj za neoceniteľné rady pri jej spracovávaní.

Abstrakt

Multicast je efektívny spôsob komunikácie, ktorý je zameraný na skupinové činnosti, ako sú napr. počítačové hry s viacerými hráčmi, videokonferencie, interaktívne diskusie a vysielanie filmu.

Pri každej komunikácii môže dochádzať k bezpečnostným rizikám, medzi ktoré patria odpočúvanie, manipulácia so správami a falšovanie identity účastníka. V práci prinášame prehľad riešení týchto rizík v multicaste. V prvej časti práce ponúkame prehľad protokolov na správu kľúčov v skupine. V tejto časti popisujeme nami navrhnutý polynomial-based protokol a dokazujeme jeho vlastnosti. Navrhnutý protokol nezabezpečuje dôvernosť budúcich a minulých správ. Navrhli sme modifikovaný polynomial-based protokol. Domnievame sa, že modificated polynomial-based protokol má zabezpečenú dôvernosť minulých aj budúcich správ, pričom efektívnosť kon3trukcie je porovnateľná s už známym protokolom Secure Locks. Všetky prezentované protokoly porovnávame podľa týchto vlastností: dôvernosť minulých a budúcich správ, nezávislosť kľúčov, počet potrebných kľúčov na strane prijímateľa a odosielateľa, počet a dĺžka správ potrebných na zmenu skupinového kľúča.

V druhej časti prinášame prehľad protokolov na autentifikáciu odosielateľa. Dĺžka autentifikačnej informácie bola v jednotlivých protokoloch tažko porovnateľná. V závere práce prinášame porovnanie dĺžky autentifikačnej informácie podľa spoločného parametra.

Kľúčové slová: multicast, bezpečnosť, správa kľúčov v skupine, autentifikácia odosielateľa.

Obsah

1 Úvod **1**

2 Multicast **2**
2.1 Definícia pojmov 2

3 Parametre multicastu **5**
3.1 Charakteristika skupiny v multicaste 5
3.2 Požiadavky na bezpečnosť 6
3.3 Požiadavky na kvalitu služby 8
3.4 Iné požiadavky 9
3.5 Scenáre 9

4 Správa kľúčov v skupine **11**
4.1 Centralizované 14
4.1.1 Párové 15
4.1.2 Vysielanie tajomstva 16
4.1.3 Hierarchia kľúčov 23
4.2 Porovnanie 26

5 Autentifikácia odosielateľa **28**
5.1 Prehľad riešeniami 28
5.2 MAC-BASE 31
5.2.1 Efektívna autentifikácia pomocou MAC 31
5.2.2 TESLA 32
5.3 Hash-BASE 34
5.3.1 Šírenie podpisu 35
5.3.2 Rozptyľovanie podpisu 38
5.3.3 Iné podpisovanie 39
5.4 Porovnanie 40

A Použité vety a definície **43**

6 Záver **46**

Literatúra **47**

Kapitola 1

Úvod

Multicast je efektívny spôsob komunikácie, ktorý je zameraný na skupinové činnosti, ako sú napr. počítačové hry s viacerými hráčmi, videokonferencie, interaktívne diskusie a vysielanie filmu.

Pri každej komunikácii môže dochádzať k bezpečnostným rizikám, medzi ktoré patria odpočúvanie, manipulácia so správami a falšovanie identity účastníka. V práci prinášame prehľad riešení týchto rizík v multicaste. Problém odpočúvania správ je v multicaste riešený šifrovaním správ tak, ako aj v iných typoch komunikácie.

V kapitole 2 sme zadefinovali komunikačný model, ktorý používame v celej práci. V tejto kapitole sa zaoberáme aj inými typmi komunikácie a porovnávame ich efektívnosť s multicastom.

V kapitole 3 uvádzame prehľad dôležitých parametrov, ktoré charakterizujú skupinu prijímateľov v multicaste.

Dôvernosť posielaných správ je v multicaste zabezpečená šifrovaním správ s použitím skupinového kľúča. To, ako sa skupinový kľúč distribuuje členom skupiny, je predmetom protokolu správa kľúčov v skupine. V kapitole 4 prinášame prehľad protokolov na správu kľúčov v skupine. V časti 4.1.2 sme navrhli polynomial-based protokol na správu kľúčov v skupine. V tejto časti skúmame dôvernosť minulých a budúcich správ a popisujeme efektívnosť navrhnutej konštrukcie. Ďalej prezentujeme modifikáciu polynomial-based protokolu. V závere kapitoly porovnávame protokoly podľa týchto vlastností: dôvernosť minulých a budúcich správ, nezávislosť kľúčov, počet potrebných kľúčov na strane prijímateľa a odosielateľa, počet a dĺžka správ potrebných na zmenu skupinového kľúča.

V kapitole 5 sa zaoberáme autentifikáciou odosielateľa. Tu uvádzame prehľad protokolov založených na autentizačných kódoch alebo na digitálnych podpisoch a hešovacích funkciách. V závere kapitoly sme porovnali všetky spomenuté protokoly podľa niektorých kritérií. Pre každý protokol sme dĺžku autentifikačnej informácie vyjadrili pomocou jedného spoločného parametra. Porovnanie je uvedené v tabuľke 5.2.

Za účelom sprehľadnenia práce uvádzame použité matematické tvrdenia a niektoré kryptografické konštrukcie v dodatku A.

Kapitola 2

Multicast

2.1 Definícia pojmov

V tejto práci budeme používať **komunikačný model**, ktorý môžeme reprezentovať grafom. Účastníci komunikácie budú reprezentovaní vrcholmi grafu. Hrana reprezentuje komunikačnú linku medzi 2 účastníkmi. Keď hrana spája dva vrcholy, potom účastníci reprezentovaní týmito vrcholmi môžu medzi sebou priamo komunikovať. Keď nie sú spojení priamo, komunikujú prostredníctvom účastníkov medzi nimi. Komunikácia prebieha posielaním správ. Budeme používať asynchrónny komunikačný model, ktorý charakterizujú nasledujúce požiadavky:

1. účastník môže poslať ľubovoľný počet správ, pričom poradie, v akom boli odoslané, sa nemusí zhodovať s poradím, v akom prišli.

2. správy sa môžu strácať, t.j. môže sa stať, že niektorá odoslaná správa nebude nikdy doručená. Ak sa správa nestratí, bude doručená v konečnom čase.

V práci abstrahujeme od fyzickej implementácie komunikačného modelu.

Účastníka, ktorý odosiela správu, budeme nazývať zdroj alebo odosielateľ. Účastníka, ktorému je správa určená, budeme nazývať prijímateľ. Odosielateľ môže poslať správu naraz viacerým prijímateľom. Množinu týchto prijímateľov budeme nazývať skupina.

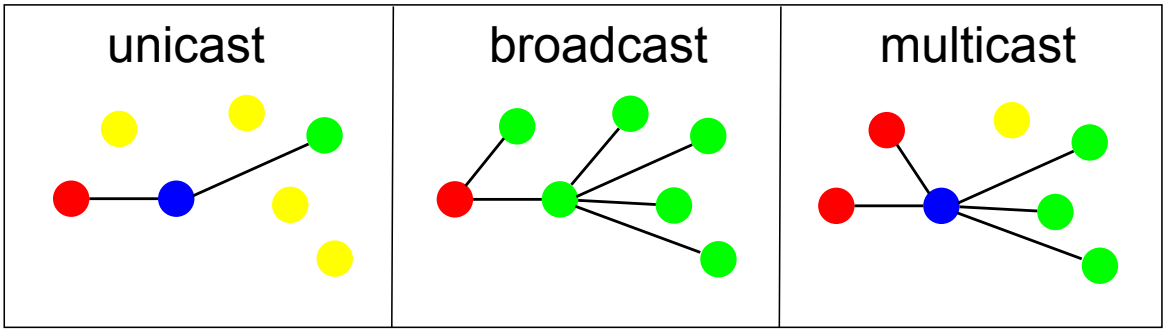
Na základe počtu prijímateľov, ktorým sú správy určené, rozlišujeme medzi tromi rôznymi typmi komunikácie:

1. Unicast – komunikácia medzi jedným odosielateľom a jedným prijímateľom.

2. Broadcast – komunikácia medzi jedným odosielateľom a všetkými účastníkmi.

3. Multicast – komunikácia medzi odosielateľmi a skupinou prijímateľov.

Z obrázku vidíme, že v unicaste je skupina tvorená jedným prijímateľom. V broadcaste skupina zahŕňa všetkých účastníkov okrem zdroja. V multicaste skupinu tvoria len niektorí účastníci. Skupina sa môže meniť počas posielania správ. Účastníci môžu prichádzať/odchádzať zo skupiny nasledovne:

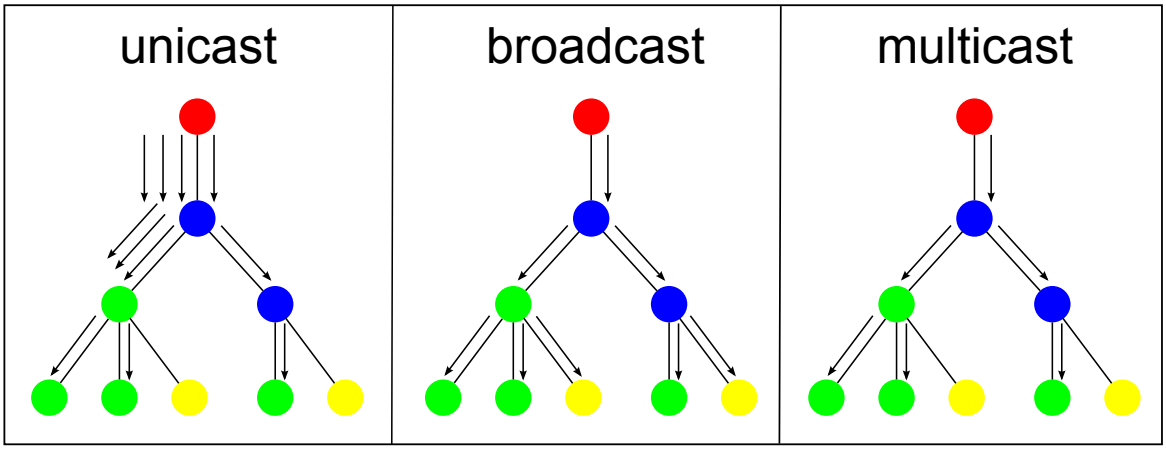


Obr. 2.1: Ilustrácia komunikácie typu unicast, broadcast a multicast

Nech zdroj posiela správy skupine G. Ak má účastník u, ktorý nie je členom skupiny G, záujem o posielané správy, tak sa musí stať členom skupiny G. Potom, ako sa účastník u stane členom tejto skupiny, bude prijímať od zdroja správy určené pre skupinu G. Ak účastník u nechce ďalej prijímať správy určené skupine G, tak opustí skupinu G.

Pri prihlasovaní do skupiny nie je zabezpečené žiadne riadenie prístupu. To znamená, že sa ľubovoľný účastník vie stať členom skupiny, a potom prijímať správy určené celej skupine.

Multicast je najvšeobecnejší spôsob komunikácie z uvedených možností. Multicast je efektívny spôsob komunikácie, ktorý je zameraný na skupinové činnosti, ako sú napr. hry s viacerými hráčmi, videokonferencie, interaktívne diskusie, vysielanie filmu, aktualizácia softvéru a podobne.



Obr. 2.2: Porovnanie efektívnosti komunikácie typu unicast, broadcast a multicast

Ak odosielateľ v unicaste chce poslať správu n prijímateľom, vytvorí n kópií správy a každému prijímateľovi ju pošle zvlášť. Môže sa stať (ako na obrázku 2.2), že po jednej linke bude rovnaká správa poslaná viackrát. Ak je v tomto prípade n veľké, môžeme ľahko zahltiť linky medzi účastníkmi. V broadcaste sa pošle len jedna správa,

ktorá bude doručená všetkým účastníkom. Tento spôsob je nevýhodný, pretože počet účastníkov je zvyčajne väčší, ako počet členov skupiny. Navyše je správa doručená aj účastníkovi, ktorý o ňu nemal záujem. Linka takéhoto účastníka je zbytočne zaťažená nevyžiadanými správami. V multicaste pošle odosielateľ len jednu správu a tá je doručená len členom skupiny. To znamená, že správu dostanú len tí, ktorí o ňu mali záujem. Z hľadiska počtu správ je multicast najefektívnejší spôsob doručovania správ skupine účastníkov.

V závislosti na počte odosielateľov rozlišujeme tieto 2 modely:

1. Jeden zdroj vysielania (Source-Specific Multicast -SSM). Tento model najlepšie vystihuje one-to-many komunikáciu. Príkladom aplikácie s jedným zdrojom je distribúcia burzových správ.

2. Ľubovoľný počet zdrojov vysielania (Any Source Multicast -ASM). Tento model slúži na many-to-many komunikáciu. Model podporuje niekoľko odosielateľov, pričom odosielateľ môže, ale byť nemusí členom skupiny. Príkladom aplikácie s viacerými odosielateľmi je videokonferencia.

IP Multicast [5] je multicast, ktorého implementácia využíva IP protokol. IP Multicast podporuje oba modely SSM aj ASM.

V IP Multicaste je skupina reprezentovaná skupinovou adresou, ktorá slúži na identifikáciu skupiny. Členovia skupiny používajú skupinovú adresu na to, aby sa stali členmi skupiny a tým preukázali záujem o dáta. Zdroj používa skupinovú adresu na to, aby vedel, komu majú ísť dáta. Keď chce zdroj poslať dáta všetkým členom skupiny, pošle správu s dátami na adresu tejto skupiny, t.j. pošle jednu správu.

Účastník, ktorý nie je členom skupiny a má záujem o vysielané dáta, sa prihlási do tejto skupiny pomocou Internet Group Management Protocol (IGMP). Potom, ako sa niekto stane členom skupiny, je vytvorený routovací strom, ktorý slúži na doručovanie vysielaných dát od zdroja ku skupine prijímateľov. Keďže zdroj pošle len jednu správu a tá je určená niekoľkým prijímateľom, je nutné ju vo vhodných uzloch (routroch) replikovať.

IP Multicast má tieto tri hlavné vlastnosti: [2, 11]

1. Všetci členovia skupiny prijímajú rovnaké správy, ktoré boli poslané zdrojom na skupinovú adresu.

2. Dynamická príslušnosť do skupiny, ktorá umožňuje každému členovi skupiny opustiť skupinu a tým prestať prijímať vysielané dáta. Takisto umožňuje účastníkom stať sa členom skupiny, pričom nie sú nutné špecifické práva.

3. Prístup k možnosti posielania správ, každý môže posielať správy všetkým členom skupiny, ktoré budú doručené celej skupine bez ohľadu na pôvod týchto správ.

Kapitola 3

Parametre multicastu

V tejto kapitole sa zameriame na parametre, ktoré definujú prostredie a nastavenia, v ktorých je multicast použitý. Znalosť týchto parametrov hrá kľúčovú úlohu vo výbere a návrhu bezpečnostných mechanizmov (kryptografické konštrukcie, protokoly) v multicaste. Jednotlivé parametre sú bližšie skúmané v [1].

3.1 Charakteristika skupiny v multicaste

V tejto časti poskytneme prehľad parametrov, ktoré charakterizujú skupinu.

1. **Veľkosť skupiny**: Zaoberáme sa otázkou: Aký je predpokladaný počet účastníkov v skupine? Napríklad v malých interaktívnych diskusiách alebo on-line hrách môže počet prijímateľov dosahovať desiatky. Pri vysielaní filmu cez internet počet prijímateľov dosahuje tisíce. Vo veľkých vysielaniach, ktorých obsahom sú zvyčajne najnovšie správy z burzy alebo zo sveta, sa počet prijímateľov šplhá k miliónom.

2. **Charakteristika členov**: Zahŕňa pamäťovú a výpočtovú silu členov skupiny. Majú všetci členovia skupiny približne rovnakú výpočtovú silu, alebo niektorí členovia majú omnoho vyššiu? Koľko pamäte majú k dispozícii? Ak vieme, že niektorí členovia budú mať menšiu výpočtovú silu, je dôležité, aby sme navrhli protokoly tak, že aj títo účastníci mohli byť členmi skupiny. V tomto prípade je dôležité, aby výpočtovo náročné operácie vykonávali výpočtovo silnejší účastníci.

3. **Dynamickosť skupiny**: Zaoberá sa otázkami, ako často sa mení skupina. Ak je skupina dynamická, t.j. členovia môžu kedykoľvek prichádzať/odchádzať zo skupiny, potom sa treba zaoberať nasledujúcimi otázkami: Účastníci sa do skupiny iba prichádzajú? Môžu tiež opúšťať skupinu? Ako často sa mení skupina (príchod/odchod účastníkov)? Vzhľadom na to, že komunikujúca skupina účastníkov sa dynamicky mení, je dôležité, aby navrhnuté protokoly efektívne riešili bezpečnosť pri príchodoch/odchodoch účastníkov zo skupiny. Ak je skupina statická, t.j. členovia sú permanentní, nemusíme sa spomenutými otázkami zaoberať.

4. **Počet odosielateľov**: Zaoberá sa počtom odosielateľov pri multicaste. Príkladom multicastu s jedným odosielateľom je vysielanie televízie. Pri interaktívnych videokonferenciách, čatoch je viacero odosielateľov. Zaujíma nás otázka: Je predpoklad, že aj niekto iný ako člen skupiny bude posielať správy?

5. **Dĺžka vysielania**: Očakávaný čas vysielania rozhoduje o tom, aký protokol je najvýhodnejšie použiť. Napríklad pri vysielaní filmu je doba vysielania presne daná dĺžkou vysielaného filmu. Niektoré aplikácie, ako televízia cez internet má časovo neohraničenú dĺžku vysielania.

6. **Objem dát**: Zaoberáme sa otázkami ako: Aký je predpokladaný objem dát? Aká môže byť maximálna prípustná odozva (latencia) aplikácie? Napríklad pri videokonferenciách je objem vysielaných dát veľký. Navyše požadujeme, aby správy prichádzali v reálnom čase. Pri interaktívnych čatoch je objem relatívne malý a požadovaná latencia môže byť maximálne niekoľko sekúnd.

Medzi ďalšie parametre patrí napr. smerovací algoritmus, ktorý rieši spôsob doručovania správ od zdroja ku skupine prijímateľov.

3.2 Požiadavky na bezpečnosť

Keďže v komunikačnom modeli sme nekládli žiadne požiadavky na bezpečnosť, môže dochádzať k týmto bezpečnostným rizikám:

1. **Odpočúvanie** – v komunikačnom modeli nie sú linky medzi účastníkmi zabezpečené. Navyše odosielateľ väčšinou nemá priamu linku so všetkými členmi skupiny. Môže sa stať, že účastník, ktorý preposiela správy, nie je členom skupiny. V tomto prípade mu v modeli nevieme zabrániť, aby neodpočúval správu, ktorá je určená pre iného účastníka.

2. **Impersonate** - komunikačný model nemá zabezpečenú autentifikáciu odosielateľa. To znamená, že ľubovoľný účastník sa môže pri odosielaní správy vydávať za iného účastníka.

3. **Manipulácia so správami** - správy posielané v modeli môžu prechádzať cez viacerých účastníkov. Títo účastníci môžu reprezentovať útočníka, ktorý správu zmení. Potom sa môže stať, že člen skupiny dostane zmenené správy.

Každá aplikácia si vyžaduje vlastné požiadavky na bezpečnosť. V tejto časti poskytneme prehľad základných požiadaviek na bezpečnosť. V časti 3.5 uvedieme príklady aplikácií, na ktorých budeme bližšie skúmať jednotlivé požiadavky. Medzi základné požiadavky patrí [1, 2]:

1. **Dôvernosť správ** – zabrániť účastníkom, ktorí nepatria do skupiny prístup k správam, ktoré sú posielané v skupine. Rozoznávame 2 druhy dôvernosti:

(a) Krátkodobá – zabezpečuje dôvernosť správ počas vysielania. Na zabezpečenie dôvernosti správ môžeme použiť mechanizmus, ktorý oddiali prístup k správam, alebo zabráni prístup ku kritickým častiam správy.

(b) Dlhodobá – zabezpečuje, že správy ostanú dôverné aj po ukončení vysielania. Príkladom aplikácie, ktorá si vyžaduje tento druh dôvernosti, je napr. vysielanie filmu.

2. **Dôvernosť budúcich správ** (Forward secrecy) – požadujeme, aby účastník, ktorý opustí skupinu, už nemal prístup k správam, ktoré budú posielané v rámci skupiny po jeho odchode.

3. **Dôvernosť minulých správ** (Backward secrecy) – požadujeme, aby nový člen skupiny nemal prístup k predchádzajúcim správam, ktoré boli posielané v skupine pred jeho príchodom do skupiny.

4. **Odolnosť voči dohodám** (Collusion freedom) – požadujeme, aby ľubovoľná množina účastníkov nebola schopná rozbiť schému.

5. **Nezávislosť** – požadujeme, aby protokoly používali nezávislé mechanizmy. To znamená, že pri prelomení jednej kryptografickej konštrukcie budú zvyšné stále fungovať.

6. **Minimal trust** – požadujeme, aby protokol nevyžadoval veľa dôveryhodných subjektov. Inak sa môže stať, že nasadenie protokolu bude príliš zložité a teda protokol bude v praxi nepoužiteľný.

7. **Autentickosť skupiny** – všetci členovia skupiny vedia overiť, či odosielateľ správy patrí do skupiny.

8. **Autentickosť odosielateľa (zdroja)** – prijímateľ musí byť schopný overiť identitu odosielateľa. V prvom rade musí overiť, či je odosielateľ členom skupiny. Pri schémach, kde nemôžu všetci členovia posielať správy, prijímateľ musí byť schopný overiť, či odosielateľ patrí do skupiny, ktorá má právo posielať správy. To znamená, že ak odosielateľ A pošle správu, tak prijímateľ musí vedieť overiť, že správa bola naozaj poslaná účastníkom A.

9. **Riadenie prístupu** – znamená, že iba registrovaní účastníci majú prístup k správam. Riadenie prístupu je značne obtiažnejšie, ak je skupina dynamická.

10. **Integrita správ** – chrániť správy pred nežiadúcimi zmenami. Zmeny správy môžu byť spôsobené chybnou linkou alebo zámerne nejakým účastníkom. Požadujeme, aby bol prijímateľ správy schopný overiť, či správa nebola modifikovaná.

11. **Nemožnosť popretia** – odosielateľ nevie poprieť správu, ktorú poslal.

12. **Anonymita** – niektoré aplikácie si vyžadujú rôzne typy anonymity. Väčšinou sa jedná o aplikácie s citlivým obsahom. Rozoznávame 2 druhy anonymity:

(a) Anonymita členov skupiny – požadujeme, aby identita členov skupiny bola držaná v tajnosti alebo v anonymite pred ostatnými účastníkmi.

(b) Anonymita odosielateľa – požadujeme, aby identita odosielateľa správy bola držaná v tajnosti alebo dokonca v anonymite.

13. **Replay resistant** – požadujeme, aby zdroj aj členovia skupiny boli odolní voči replay attack. Táto požiadavka je extrémne dôležitá hlavne v aplikáciách, kde vyžadujeme, aby všetky odoslané správy boli doručené. Ak správa nie je doručená, tak ju odosielateľ musí znovu poslať. Príkladom aplikácie, kde požadujeme, aby všetky správy boli doručené je aktualizácia softvéru.

Dôvera v multicaste s jedným zdrojom vysielania je zvyčajne riešená tak, že predpokladáme, že zdroju vysielania môžeme veriť. Typické funkcie, ktoré zdroj zabezpečuje sú: riadenie prístupu, správa kľúčov, logovanie. V iných prípadoch zastávame myšlienku, že nie je dobré, aby jeden subjekt vykonával všetky tieto funkcie. Dôvodom je väčšia pravdepodobnosť zlyhania, prípadne kompromitácie jedného subjektu ako viacerých subjektov naraz. Aplikácie, ktoré sú postavené na dôvere k jednej entite sú zraniteľnejšie. Preto sa vo všeobecnosti snažíme distribuovať úlohy týkajúce sa bezpečnosti.

3.3 Požiadavky na kvalitu služby

Kvalita služby úzko súvisí s dosiahnutou bezpečnosťou. Čím viac požiadaviek z bezpečnosti chceme splniť, tým menšiu kvalitu služby vieme dosiahnuť. Pretože na zabezpečenie týchto požiadaviek potrebujeme splniť protokoly, ktoré zvyčajne vyžadujú extra prácu na strane odosielateľa aj prijímateľa. Najdôležitejšie parametre, ktoré ovplyvňujú kvalitu služby sú:

1. **Latencia** – je čas od momentu, kedy odosielateľ pošle správu, po moment, kedy prijímateľ dostane správu. V real-time aplikáciách je dôležitá nízka latencia. Napríklad obchodník na burze potrebuje vedieť aktuálne ceny akcií, aby mohol promptne zareagovať na vývoj situácie na trhu. Ak by bola latencia príliš vysoká, (niekoľko minút) jeho rozhodnutia by nemuseli mať želaný efekt.

2. **Inicializácia a terminácia** – práca, ktorá je potrebná pri inicializácii a terminácii skupiny. V tejto práci je zahrnuté napr. riadenie prístupu.

3. **Príchod/odchod** – práca, ktorú potrebujeme pri pridávaní/mazaní členov zo skupiny. Napríklad pri vymazaní člena zo skupiny potrebujeme zabezpečiť, aby vymazaný člen nemal prístup k ďalšej komunikácii.

4. **Zahltenie** – nastáva pri centralizovanom riadení, ak sa veľa členov skupiny pripojí tesne pred začatím vysielania alebo odpojí tesne po ukončení vysielania. V tomto prípade sa môže stať, že linky budú preťažené a pre niektorého člena skupiny môže byť služba nedostupná. Riešením môže byť riadenie viacerých liniek tak, aby bolo zaistené, že dostupná priepustnosť nie je prekročená.

5. **Dostupnosť služby** – požadujeme, aby služba bola dostupná pre všetkých účastníkov. Dostupnosť služby je možné znížiť pomocou útoku zahltením. Tento útok je jednoduchý na implementáciu a má veľký dopad. Je preto dobré chrániť proti zahlteniu nielen členov skupiny, ale aj účastníkov, ktorí preposielajú správy.

3.4 Iné požiadavky

1. **Pamäťové nároky** – pri vykonávaní protokolov často potrebujeme ukladať pomocné informácie na strane odosielateľa aj prijímateľa. Preto sa snažíme navrhnúť protokoly aj kryptografické konštrukcie tak, aby spotrebovaná pamäťová kapacita bola čo najmenšia.

2. **Výpočtová sila** – pri vykonávaní protokolov často potrebujeme vykonávať výpočtovo náročné operácie na strane odosielateľa aj prijímateľa. Preto sa snažíme protokoly aj kryptografické konštrukcie navrhnúť tak, aby potrebná výpočtová sila bola čo najmenšia.

3.5 Scenáre

V predchádzajúcich častiach sme sa zaoberali parametrami, ktoré charakterizujú rôzne scenáre v multicaste. V tejto časti sa budeme zaoberať aplikáciami s jedným a viacerými odosielateľmi v multicaste. Ilustrujeme a vysvetlíme parametre, ktoré charakterizujú tieto aplikácie, pričom niektoré z týchto parametrov boli prezentované v [1].

Príklad - burzové správy

Nasledujúce parametre sú charakteristické pre aplikáciu, ktorá posiela členom skupiny aktuálne burzové správy. Správy posiela len jeden účastník, pričom ostatní členovia skupiny sú prijímatelia.

- Počet členov v skupine môže byť od tisícov do miliónov, pričom účastníci môžu byť na rôznych geografických miestach.
- Zdroj zvyčajne disponuje väčšou výpočtovou silou a pamäťovou kapacitou. Je vhodné, aby protokoly a kryptografické konštrukcie boli optimalizované hlavne na strane prijímateľov.
- Skupina je dynamická, účastníci prichádzajú do skupiny a odchádzajú zo skupiny veľmi často.
- Objem dát je závislý od formy, akou sú burzové správy reprezentované. Ak sa jedná o text, objem je relatívne malý.
- Dĺžka vysielania zvyčajne nie je presne stanovená.

- Požadujeme, aby aplikácia mala latenciu maximálne niekoľko sekúnd. Po tomto čase burzové informácie v správach stratia svoju hodnotu.
- V týchto aplikáciách je dôležité, aby bola zabezpečená autentickosť odosielateľa a integrita správ. Člen skupiny nesmie nikdy považovať falošné burzové správy za platné. Protokoly a kryptografické konštrukcie, ktoré zabezpečujú autentifikáciu odosielateľa, budeme bližšie skúmať v kapitole 5.
- Účastník sa stane členom skupiny, iba ak si za službu zaplatí. Ďalej požadujeme, aby účastníci, ktorí nie sú členmi skupiny, nemali prístup k tejto službe.
- V týchto aplikáciách obsahujú správy informácie s vysokou hodnotou pre prijímateľa. Počas vysielania strácajú správy aktuálnosť a s odstupom času už nemajú žiadnu hodnotu. Väčšina burzových správ je s časovým oneskorením zverejnená. V týchto prípadoch môže byť dôvernosť správ riešená použitím šifrovania alebo mechanizmov, ktoré na nejaký čas zabránia prístup k správam. Zvyčajne požadujeme, aby boli splnené aj nasledovné požiadavky: dôvernosť budúcich správ, dôvernosť minulých správ, odolnosť voči dohodám.

Príklad - videokonferencia

V tejto časti uvádzame parametre aplikácie s viacerými zdrojmi vysielania. Vo väčšine prípadov všetci členovia skupiny majú záujem posielať správy. Medzi takéto aplikácie patrí: videokonferencia, interaktívne prednášky a iné.

- Počet členov v skupine je zvyčajne od desiatok do stoviek účastníkov.
- Všetci členovia skupiny majú približne porovnateľnú výpočtovú silu.
- Skupina je väčšinou statická a formuje sa pred začatím konferencie.
- V rámci konferencie sa prenášajú dáta s veľkým objemom, keďže sa jedná o prenos obrazu a zvuku.
- Dĺžka vysielania je od niekoľko minút do pár hodín.
- Konferencie prebiehajú v reálnom čase, preto sa snažíme protokoly a kryptografické konštrukcie optimalizovať tak, aby latencia bola minimálna.
- Je dôležité, aby bola zabezpečená autentickosť odosielateľa a integrita správy. Protokoly a kryptografické konštrukcie, ktoré zabezpečujú autentifikáciu odosielateľa, budeme bližšie skúmať v kapitole 5.
- Dôvernosť správ je zväčša závislá na obsahu konferencie. Ak sa jedná o videokonferenciu, na ktorej budú prezentované dôverné informácie, tak požadujeme dôvernosť správ. Dôvernosť správ je riešená symetrickým šifrovaním. Väčšinou v týchto typoch aplikáciách požadujeme dlhodobú dôvernosť správ, ktorú sme definovali v časti 3.2. Týmto problémom sa budeme bližšie zaoberať v kapitole 4.

Kapitola 4

Správa kľúčov v skupine

Keďže správy v multicaste môžu byť posielané cez nezabezpečené linky, vzniká riziko možného odpočúvania. To znamená, že neoprávnený účastník môže získať správy, ktoré boli určené pre iného účastníka. V multicaste môže byť bezpečná skupinová komunikácia zabezpečená šifrovaním správ s použitím skupinového kľúča. Tento kľúč je zvyčajne symetrický, pretože symetrické šifrovanie je výpočtovo efektívnejšie ako asymetrické šifrovanie. Do popredia sa dostáva otázka, ako efektívne generovať a distribuovať skupinový kľúč všetkým členom skupiny? Týmto problémom sa zaoberá protokol na správu kľúčov v skupine.

Protokol na správu kľúčov v skupine by mal spĺňať nasledujúce bezpečnostné požiadavky:

1. Skupinový kľúč by mal byť známy len členom skupiny.
2. Dôvernosť budúcich správ – definované v časti 3.2
3. Dôvernosť minulých správ – definované v časti 3.2
4. Odolnosť voči dohodám – definované v časti 3.2
5. Použité skupinové kľúče by mali byť od seba nezávislé.

Na prezentovanie protokolov použijeme komunikačný model definovaný v časti 2.1, v ktorom požadujeme existenciu týchto dvoch subjektov:

1. Kontrolór skupiny (Group Controller GC) - účastník, ktorý je zodpovedný za autentifikáciu a riadenie prístupu pre účastníka, ktorý sa chce stať členom skupiny.
2. Účastník s kľúčmi (Key Server KS) - účastník, ktorý je zodpovedný za generovanie a distribúciu skupinového kľúča.

Obe funkcie môže mať na starosti jeden účastník. V tomto modeli účastníci môžu vykonávať nasledujúce operácie:

1. Pridanie do skupiny – účastník u sa stane členom skupiny. Potom účastník s kľúčmi mu musí bezpečne doručiť skupinový kľúč. Keďže požadujeme dôvernosť minulých správ, tak účastník s kľúčmi vygeneruje nový skupinový kľúč, ktorý doručí všetkým členom skupiny.

2. Opustenie skupiny – účastník u opustí skupinu. Keďže požadujeme dôvernosť budúcich správ, tak účastník s kľúčmi musí zmeniť skupinový kľúč a distribuovať ho všetkým členom skupiny.

Ďalšie predpoklady na model:

1. V tejto kapitole predpokladáme, že pred pridaním učastníka u do skupiny vždy prebehne autentifikácia a riadenie prístupu pre účastníka u. Takto zabezpečíme, že iba registrovaní a autentifikovaní účastníci budú môcť byť pridaní do skupiny.

2. Predpokladáme, že každý prijímateľ po prijatí správy so skupinovým kľúčom vie overiť identitu odosielateľa a integritu správy.

3. Predpokladáme, že keď sa účastník u stane členom skupiny, tak KS a u zdieľajú symetrický kľúč K_u, ktorý slúži na bezpečnú komunikáciu medzi KS a u. (Komunikácia medzi KS a u bude šifrovaná s použitím symetrického kľúča K_u.)

Označenie:

1. K_u – symetrický kľúč, ktorý je známy len účastníkom u a KS.

2. K_G^{old} – skupinový kľúč, ktorý bol známy členom skupiny pred zmenou skupinového kľúča. Napríklad pred odchodom účastníka u zo skupiny.

3. K_G^{new} – nový skupinový kľúč, tento kľúč je vygenerovaný pri príchode/odchode účastníka u do skupiny G.

Uvedieme si príklad, ako vie kontrolór skupiny zistiť, či má účastník právo stať sa členom skupiny. Ak sa nový autentifikovaný účastník u chce stať členom skupiny, musíme zistiť, či je registrovaný a či má právo používať túto službu. Tento proces sa nazýva riadenie prístupu. Zodpovedný účastník je kontrolór skupiny. Vo väčšine prípadov má kontrolór skupiny k dispozícii zoznam registrovaných účastníkov (Acces Controll List ACL). Účastník u sa stane členom skupiny, len ak sa nachádza na zozname registrovaných účastníkov.

Na nasledujúcom príklade si ilustrujeme problém distribúcie skupinového kľúča pri zmene členov skupiny. Predpokladajme, že účastník u sa stal členom skupiny. Keďže komunikácia v skupine je šifrovaná, musíme členovi u bezpečne poslať skupinový kľúč. Ak požadujeme, aby člen u nemal prístup k predchádzajúcej komunikácii, tak v skupine musí byť zmenený skupinový kľúč. Účastník s kľúčmi vygeneruje nový skupinový kľúč K_G^{new}. Tento kľúč musí bezpečne doručiť všetkým členom skupiny. Jedným možným riešením je zašifrovať skupinový kľúč K_G^{new} s použitím starého skupinového kľúča K_G^{old}. Účastník s kľúčmi pošle všetkým členom skupiny správu:

$$M = E_{K_G^{new}}(K_G^{old})$$

Nový člen u však nepozná starý skupinový kľúč K_G^{old} a teda nevie dešifrovať správu M. Preto členovi u je skupinový kľúč zašifrovaný s kľúčom K_u. Účastník s kľúčmi pošle členovi u správu:

$$M' = E_{K_u}(K_G^{new})$$

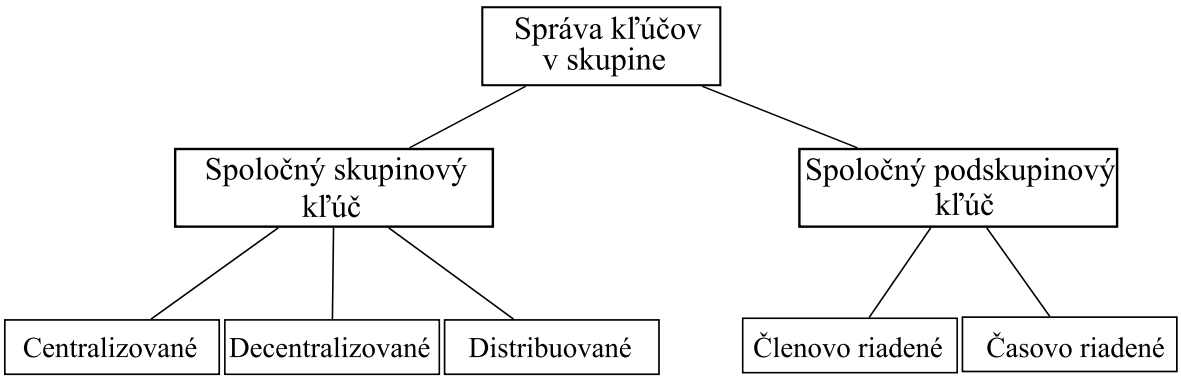
Po doručení u dešifruje správu M' a tak získa skupinový kľúč K_G^{new}. Pri príchode nového člena do skupiny účastník s kľúčmi pošle 2 správy. Prvá správa M je poslaná všetkým členom skupiny a druhá správa M' je poslaná len členovi u. Tieto dve správy zabezpečia, že všetci členovia skupiny poznajú skupinový kľúč K_G^{new}.

Vážnejší problém nastáva, ak účastník u opustí skupinu. Ak chceme zabezpečiť dôvernosť budúcich správ, tak v skupine musí byť zmenený skupinový kľúč. KS, ktorý je zodpovedný za generovanie skupinového kľúča vygeneruje nový skupinový kľúč K_G^{new}. Tento kľúč musí byť doručený všetkým členom skupiny bez šifrovania starým skupinovým kľúčom K_G^{old}, lebo účastník u, ktorý odišiel, by sa dozvedel nový skupinový kľúč K_G^{new}. Jednoduchým riešením problému je každému členovi i poslať nový skupinový kľúč, ktorý bude zašifrovaný kľúčom K_i. Účastník s kľúčmi pošle každému členovi i správu:

$$M = E_{K_i}(K_G^{new})$$

V tomto prípade KS musí poslať $n-1$ rôznych správ, pričom každá správa obsahuje zašifrovaný skupinový kľúč K_G^{new}. Toto riešenie je neefektívne a vyžaduje pri každom odchode účastníka zo skupiny $O(n)$ správ na distribúciu skupinového kľúča K_G^{new}, kde n je počet členov skupiny. Ďalej účastník, ktorý je zodpovedný za generovanie a distribúciu kľúčov, potrebuje mať uložených $n+1$ kľúčov. Jeden kľúč je skupinový a zvyšných n kľúčov zodpovedá členom skupiny.

V tejto kapitole poskytneme prehľad protokolov, ktoré sa zaoberajú správou kľúčov v multicaste. V závere kapitoly sú všetky prezentované protokoly porovnané podľa relevantných kritérií.



Obr. 4.1: Prehľad riešení

Existujúce riešenia môžeme rozdeliť na dve skupiny [3]:

1. **Spoločný skupinový kľúč** – všetci členovia skupiny zdieľajú jeden skupinový kľúč. Aby sme zabezpečili dôvernosť minulých správ a dôvernosť budúcich správ,

tak pri akejkoľvek zmene členov skupiny je dôležité zmeniť skupinový kľúč. V dynamických skupinách s veľkým počtom účastníkov je dôležité, aby počet správ pri zmene skupinového kľúča bol minimálny. Riešenia, ktoré sú postavené na jednom skupinovom kľúči, môžeme rozdeliť do týchto kategórií:

(a) **Centralizované protokoly** – za generovanie a distribúciu kľúčov je zodpovedný jeden účastník. Prehľad protokolov, ktoré patria do tejto skupiny, poskytneme v časti 4.1.

(b) **Decentralizované protokoly** – za generovanie a distribúciu kľúčov zodpovedá viacero účastníkov.

(c) **Distribuované protokoly** – členovia skupiny kooperujú a spoločne sa dohodnú na skupinovom kľúči. V tejto skupine protokolov nie je potrebný účastník s kľúčmi, lebo členovia skupiny sú zodpovední za generovanie skupinového kľúča.

2. **Spoločný podskupinový kľúč** – členovia skupiny sú organizovaní do podskupín, pričom každá podskupina má svoj vlastný podskupinový kľúč. V tomto prípade, ak člen u opustí skupinu, tak stačí vygenerovať nový podskupinový kľúč, iba pre podskupinu, v ktorej sa člen u nachádzal. Organizovanie členov do podskupín redukuje počet správ potrebných pri zmene kľúča. Riešenie však vyžaduje dodatočnú transformáciu správ pre členov danej podskupiny. Riešenia, ktoré sú postavené na spoločnom podskupinovom kľúči, môžeme rozdeliť do dvoch skupín v závislosti od toho, ako sa mení podskupinový kľúč.

   (a) **Členovo riadené** (Membership-Driven Re-keying) – kľúč, ktorý prislúcha podskupine, sa mení pri akejkoľvek zmene členov danej podskupiny.

   (b) **Časovo riadené** (Time-Driven Re-keying) – kľúč, ktorý prislúcha podskupine, sa mení periodicky. To znamená, že ak sa účastník stane členom skupiny, správy vie dešifrovať až po najbližšej zmene podskupinového kľúča. Ak účastník opustí skupinu, nestratí schopnosť dešifrovať správy. Túto schopnosť stráca až po zmene podskupinového kľúča. Dôvernosť minulých aj budúcich správ je zabezpečená až po zmene podskupinového kľúča.

4.1 Centralizované

Pri centralizovanom riešení zodpovedá za generovanie a distribúciu jeden subjekt. V závislosti od toho, ako je nový skupinový kľúč K_G^{new} distribuovaný, môžeme centralizované protokoly rozdeliť na tri skupiny [3]:

- Párové (Pairwise)
- Vysielanie tajomstva (Broadcast Secrets)
- Hierarchia kľúčov (Key Hierarchy)

4.1.1 Párové

Pre protokoly, ktoré patria do skupiny párové (pairwise), je pri distribúcii nového skupinového kľúča charakteristický nasledujúci postup. Účastník s kľúčmi vytvorí bezpečné komunikačné linky medzi ním a účastníkom i pomocou zdieľaného kľúča K_i. (t.j. účastník s kľúčmi bude šifrovať správy pre účastníka i s kľúčom K_i)

Group Key Management Protocol (GKMP)

V tomto protokole [3, 9, 10] účastník s kľúčmi pri akejkoľvek zmene členov skupiny posiela správu, ktorá obsahuje dva kľúče $M = \langle K_G^{new}, GK \rangle$, kde K_G^{new} je nový skupinový kľúč a GK je kľúč, ktorý sa používa pri periodickej zmene skupinového kľúča. Účastník s kľúčmi periodicky mení skupinový kľúč, pričom na bezpečné doručenie skupinového kľúča používa kľúč GK.

1. Pridanie do skupiny – ak sa účastník u stane členom skupiny, tak KS vygeneruje nový skupinový kľúč K_G^{new} a kľúč GK. Kľúč GK sa používa na šifrovanie skupinového kľúča pri periodickej zmene skupinového kľúča. Oba kľúče $\langle K_G^{new}, GK \rangle$ potrebuje bezpečne doručiť všetkým členom skupiny. Členovi u pošle správu:

   $$M = E_{K_u}(\langle K_G^{new}, GK \rangle)$$

   Zvyšným členom, ktorí poznajú predchádzajúci starý skupinový kľúč K_G^{old}, pošle KS správu:

   $$M' = E_{K_G^{old}}(\langle K_G^{new}, GK \rangle)$$

   Tieto dve správy M, M' zabezpečia, že všetci členovia skupiny budú poznať nový skupinový kľúč K_G^{new} a nový kľúč GK.

2. Opustenie skupiny – keďže požadujeme dôvernosť budúcich správ, tak pri odchode učastníka u zo skupiny KS vygeneruje novú dvojicu kľúčov $\langle K_G^{new}, GK \rangle$. Tieto kľúče potrebuje bezpečne poslať všetkým členom skupiny.

   $$M_i = E_{K_i}(\langle K_G^{new}, GK \rangle) \text{ pre } \forall i \in G$$

   Keď účastník i dostane správu M_i tak ju dešifruje kľúčom K_i a tak získa skupinový kľúč K_G^{new}. Účastník u, ktorý opustil skupinu, sa v množine G už nenachádza a teda nedostane správu s novým skupinovým kľúčom.

3. Periodická zmena skupinového kľúča – KS vygeneruje novú dvojicu $\langle K_G^{new}, GK \rangle$ a pošle správu:

   $$M = E_{GK'}(\langle K_G^{new}, GK \rangle)$$

   Všetci členovia skupiny G poznajú kľúč GK' a teda vedia dešifrovať správu M a zistiť nový skupinový kľúč K_G^{new}.

Efektívnosť konštrukcie

Keďže skupinový kľúč je zmenený pri príchode/odchode účastníka zo skupiny, protokol má zabezpečenú dôvernosť minulých aj budúcich správ. Z navrhnutého protokolu je jasné, že pri každom odchode účastníka zo skupiny je potrebných $O(n)$ správ, na zmenu skupinového kľúča. Protokol si vyžaduje, aby účastník s kľúčmi mal uložených $n+2$ kľúčov (K_i pre $i = 1, 2..., n$, GK, K_G), kde K_G je aktuálne používaný skupinový kľúč. Pamäťová náročnosť pre prijímateľa i je pomerne malá. Prijímateľ i potrebuje mať uložené len 3 kľúče (K_i, GK, K_G). Keďže účastník s kľúčmi musí mať uložených $O(n)$ kľúčov a navyše pri každom odchode účastníka zo skupiny potrebuje unicastom poslať $O(n)$ správ, je navrhnutý protokol veľmi neefektívny pre dynamické skupiny s veľkým počtom účastníkov.

4.1.2 Vysielanie tajomstva

Hlavnou myšlienkou protokolov v tejto skupine je, že účastník s kľúčmi posiela nový skupinový kľúč v jednej správe pre všetkých členov skupiny.

Triviálne riešenie

Pre účely porovnania si ilustrujeme nasledujúci jednoduchý protokol:

1. Pridanie do skupiny/Opustenie skupiny – účastník s kľúčmi vygeneruje nový skupinový kľúč K_G^{new}. Nech G je množina členov skupiny, ktorým účastník s kľúčmi potrebuje doručiť skupinový kľúč K_G^{new}. Pre každého člena i zo skupiny G KS zašifruje skupinový kľúč K_G^{new} nasledovne:

   $$M_i = E_{K_i}(K_G^{new})$$

   Účastník s kľúčmi pošle jednu správu, v ktorej sa nachádzajú všetky šifrové texty M_i. Správa M, v ktorej je zašifrovaný skupinový kľúč, má tvar:

   $$M = \langle M_1 | M_2 | M_3 | ... | M_n \rangle$$

   Keď prijímateľ i dostane správu M, dešifruje časť M_i pomocou kľúča K_i a získa skupinový kľúč. Formálne pre prijímateľa i:

   $$K_G^{new} = D_{K_i}(M_i)$$

Efektívnosť konštrukcie

Keďže skupinový kľúč je zmenený pri príchode/odchode účastníka zo skupiny, protokol má zabezpečenú dôvernosť minulých aj budúcich správ. Z navrhnutého protokolu je zrejmé, že pri každom príchode/odchode účastníka zo skupiny je potrebná jedna správa na zmenu skupinového kľúča. Protokol vyžaduje, aby mal účastník s kľúčmi uložených n kľúčov. Prijímateľ i musí mať uložený jeden symetrický kľúč K_i. Hlavnou nevýhodou protokolu je, že účastník s kľúčmi pri zmene skupinového kľúča posiela správu M, ktorej veľkosť je $(n \cdot s)$, kde n je počet členov v skupine a s je veľkosť výstupu použitého šifrovacieho algoritmu.

Secure Locks

Skupinový kľúč sa mení po každej správe, pričom iba členovia skupiny ho vedia zo správy dešifrovať. V protokole secure locks autor predpokladá, že odosielateľ je účastník s kľúčmi. Hlavnou myšlienkou protokolu secure locks [4] je, že odosielateľ vytvorí zámok, ktorý je závislý na skupinovom kľúči a na kľúčoch, ktoré prislúchajú členom skupiny. Zámok je navrhnutý tak, aby ho boli schopní otvoriť iba členovia skupiny . Účastníci po otvorení zámku získajú skupinový kľúč, ktorý bude slúžiť na dešifrovanie správy. Konštrukcia zámku pre správu so skupinovým kľúčom je postavená na čínskej zvyškovej vete, ktorá je uvedená v dodatku. Teraz si vysvetlíme, ako sa konštruuje a používa zámok v tomto protokole. Predpokladáme, že každý člen i skupiny pozná dvojicu $\langle N_i, K_i \rangle$, ktorá je tiež známa účastníkovi s kľúčmi. Keď účastník s kľúčmi chce poslať správu M, vygeneruje si nový skupinový kľúč K_G^{new}, ktorý slúži na zašifrovanie správy M. Zdroj postupne zašifruje skupinový kľúč K_G^{new} všetkými kľúčmi K_i, kde K_i reprezentuje kľúč, ktorý je zdieľaný medzi účastníkom s kľúčmi a účastníkom i. To znamená $C_i = E_{K_i}(K_G^{new})$. Účastník s kľúčmi vypočíta zámok X, ktorý je riešením nasledujúcich kongruencií:

$$\begin{aligned} X &\equiv C_1 \pmod{N_1} \\ X &\equiv C_2 \pmod{N_2} \\ &\dots \\ X &\equiv C_n \pmod{N_n} \end{aligned}$$

Účastník s kľúčmi pošle správu:

$$M' = \langle X, E_{K_G^{new}}(M) \rangle$$

Každý člen i po prijatí správy M' vie získať skupinový kľúč $K_G^{new} = D_{K_i}(X \pmod{N_i})$. Pomocou skupinového kľúča K_G^{new} vie dešifrovať správu $D_{K_G^{new}}(E_{K_G^{new}}(M)) = M$. Správu M' však vedia dešifrovať iba členovia skupiny, ktorých kľúče K_i sú použité na zašifrovanie kľúča K_G^{new}.

Efektívnosť konštrukcie

Protokol má zabezpečenú dôvernosť minulých aj dôvernosť budúcich správ, pretože odosielateľ pri poslaní každej správy vygeneruje nový skupinový kľúč. Hlavnou výhodou navrhnutého protokolu je, že keď účastník A opustí skupinu, odosielateľ potrebuje jednu správu na zmenu kľúča. Skupinový kľúč sa mení po každej správe, pričom iba členovia skupiny ho vedia zo správy dešifrovať. Prijímateľ potrebuje mať uložené 2 kľúče $\langle K_i, N_i \rangle$. Kľúč K_G^{new}, ktorým dešifruje správu, sa dozvie v správe M'. Protokol vyžaduje, aby odosielateľ mal pre $\forall\ i \in G$ uloženú dvojicu kľúčov $\langle K_i, N_i \rangle$. Hlavnou nevýhodou protokolu je, že odosielateľ pri každom poslaní správy potrebuje nájsť riešenie X, čo je výpočtovo náročný proces. Navyše odosielateľ musí mať uložených $2n$ kľúčov. Preto sa dá protokol secure lock použiť len v skupinách s malým počtom členov.

Diskusia

Skúsme porovnať protokol Secure Lock s predchádzajúcim protokolom. Nech sú v oboch porovnaných protokoloch použité rovnaké dĺžky kľúčov a rovnaký šifrovací algoritmus, ktorého dĺžka výstupu je s. Správa M v nami navrhnutom protokole má dĺžku $n \cdot s$. Z protokolu Secure Lock je zrejmé, že $N_i > s$. To však znamená, že súčin $L = N_1 \cdot N_2 \cdot ... \cdot N_n > n \cdot s$. Riešenie X je síce z intervalu $\langle 0, L-1 \rangle$, ale s veľkou pravdepodobnosťou je dĺžka X väčšia alebo rovná ako $n \cdot s$. V oboch protokoloch treba postupne zašifrovať skupinový kľúč všetkými kľúčmi členov skupiny. Navyše v protokole Secure Lock treba pri výpočte zámku X vykonávať zložité aritmetické operácie. Z tohto porovnania je zrejmé, že nami prezentovaný protokol je efektívnejší, ako protokol Secure Lock.

Polynomial-Based Protocol (PBP)

Protokol má riešiť bezpečnú distribúciu skupinového kľúča členom skupiny G. Od protokolu požadujeme, aby účastníci, ktorí nepatria do skupiny G, nepoznali skupinový kľúč. V navrhnutom protokole predpokladáme, že účastník s kľúčmi pozná kľúče K_i, kde K_i reprezentuje symetrický kľúč, ktorý je známy účastníkovi i a účastníkovi s kľúčmi. V protokole sa nezaoberáme spôsobom, ako sa účastník s kľúčmi a účastník i dohodnú na symetrickom kľúči K_i. Dohoda na kľúči K_i môže prebiehať pri autentifikácii učastníka i. V navrhnutom protokole nerozlišujeme medzi operáciami pridanie do skupiny/opustenie skupiny, pretože v protokole prebieha rovnaký algoritmus pre obe spomínané operácie.

1. Pridanie do skupiny/Opustenie skupiny – účastník s kľúčmi vygeneruje nový skupinový kľúč K_G^{new}. Tento kľúč musí bezpečne poslať všetkým členom skupiny. Účastník s kľúčmi vygeneruje polynóm:

   $$p(x) = a_n x^n + a_{n-1} x^{n-1} + ... + a_2 x^2 + a_1 x + a_0$$

   Polynóm $p(x)$ je stupňa najviac n pričom koeficienty $a_n, a_{n-1}, ..., a_2, a_1, a_0$ sú z poľa Z_q, kde q je dostatočne veľké prvočíslo. Pre polynóm $p(x)$ platia nasledujúce podmienky:

   - $\forall i \in G \; p(K_i) = K_G^{new}$, kde $|G| = n$
   - $p(0) = a_0$, kde $a_0 \neq K_G^{new}$ je generované náhodne z poľa Z_q

   Účastník s kľúčmi pošle správu M, v ktorej sa nachádzajú všetky koeficienty polynómu $p(x)$.

   $$M = \langle a_n, a_{n-1}, ..., a_2, a_1, a_0 \rangle$$

   Keď prijímateľ i dostane správu M, zrekonštruuje si polynóm $p(x)$. Do polynómu dosadí svoju hodnotu K_i a tak získa skupinový kľúč ($p(K_i) = K_G^{new}$).

Pre korektnosť konštrukcie treba dokázať, že vždy existuje polynóm $p(x)$, pre ktorý platia horeuvedené podmienky. Polynóm $p(x)$ vieme skonštruovať podľa Lagrangeovej interpolácie.

Efektívnosť konštrukcie

Účastník s kľúčmi musí skonštruovať polynóm $p(x)$. Ak použijeme na konštrukciu $p(x)$ Lagrangeovu interpoláciu, tak časová náročnosť výpočtu koeficientov $(a_n, a_{n-1}, ..., a_1, a_0)$ polynómu $p(x)$ bude $O(n^2)$. Správa M na zmenu skupinového kľúča obsahuje koeficienty $(a_n, a_{n-1}, ..., a_1, a_0)$. Každý koeficient polynómu $p(x)$ je z poľa Z_q a ich veľkosť môžeme zhora ohraničiť hodnotou q. Správa M bude mať veľkosť $n \cdot \log q$. Prijímateľ i po prijatí správy M potrebuje zistiť hodnotu polynómu v bode K_i. Hodnota $p(K_i)$ sa dá vypočítať v čase $O(n)$. Účastník s kľúčmi potrebuje mať uložených n kľúčov. Každý prijímateľ má uložený jeden kľúč.

Bezpečnosť konštrukcie

Nech má skupina 3 účastníkov. Pri distribúcii skupinového kľúča vygeneruje účastník s kľúčmi nasledujúci polynóm:

$$p(x) = A(x - K_1)(x - K_2)(x - K_3) + K_G$$

Naprogramovali sme program, ktorý postupne vyskúša všetky kombinácie kľúčov (K_1, K_2, K_3) nad poľom Z_q a vypočíta, koľko existuje rôznych trojíc (x_1, x_2, x_3) takých, že $p(x_1) = p(x_2) = p(x_3)$. Dospeli sme k výsledku, že pre polia, ktoré majú $q \pmod 3 = 1$ prvkov, existuje kombinácia kľúčov K_1, K_2, K_3 taká, že počet rôznych trojíc je $\frac{q-1}{3}$. Každá trojica (x_1, x_2, x_3) reprezentuje potenciálne kľúče účastníkov a hodnota $p(x_1)$ reprezentuje potenciálny skupinový kľúč.

Po dlhšom skúmaní sme medzi kľúčmi K_1, K_2, K_3 objavili závislosť $K_2 = \omega K_1$ a $K_3 = \omega^2 K_1$, kde $\omega^3 = 1$ a $\omega \neq 1$. Výsledky nám umožnili sformulovať nasledujúce tvrdenia, kde sme spomínanú hypotézu všeobecne dokázali.

Definícia 4.1.1. *Nech q je prvočíslo také, že $q \pmod n = 1$ a Z_q je pole s q prvkami. Polynóm $p(x)$ stupňa n nad poľom Z_q budeme nazývať bezpečný polynóm, ak existuje $\frac{q-1}{n}$ rôznych n-tíc $(x_1, x_2, ..., x_n)$ takých, že:*

$$p(x_1) = p(x_2) = ... = p(x_{n-1}) = p(x_n) = h$$

A navyše pre každú hodnotu h existuje najviac jedna taká n-tica.

Veta 4.1.1. *Nech má skupina G n účastníkov. Potom účastník s kľúčmi vie zostrojiť bezpečný polynóm.*

Dôkaz. Účastník s kľúčmi vygeneruje polynóm:

$$p(x) = A(x - K_1)(x - K_2)...(x - K_n) + K_G$$

Je zrejmé, že hodnota A, ani K_G neovplyvnia počet kandidátov na skupinový kľúč. Počet kandidátov ovplyvňuje len výber hodnôt $(K_1, K_2, ..., K_n)$. Zvoľme si prvok ω z poľa Z_q tak, aby $\omega^n = 1$, pričom $\omega \neq 1$. Musíme ukázať, že také ω vždy existuje. Nech g je generátor grupy $(Z_q - \{0\}, \cdot)$, potom

$$\omega = g^{\frac{q-1}{n}}$$

$$\omega^n = (g^{\frac{q-1}{n}})^n = g^{q-1} = 1$$

Dôkaz bude uplný ak ukážeme, že pre hodnoty kľúčov

$$K_i = \omega^{i-1} K_1,\ i \in \langle 0, 1, ...n \rangle$$

má polynóm $p(x)$ presne $\frac{q-1}{n}$ kandidátov na skupinový kľúč.

$$p(x) = A(x - K_1)(x - \omega K_1)...(x - \omega^{i-1} K_1)...(x - \omega^{n-1} K_1) + K_G$$

po roznásobení nám vnútorné členy, ktoré obsahujú x^i pre $1 \leq i \leq n-1$ vypadnú pretože:

$$\omega^n - 1 = (\omega - 1)(\omega^{n-1} + \omega^{n-2} + ... + \omega^2 + \omega + 1) = 0$$

$$(\omega^{n-1} + \omega^{n-2} + ... + \omega^2 + \omega + 1) = 0$$

Rovnica $p(x) - K_G = 0$ má n rôznych riešení, ktorými sú $(K_1, \omega K_1, \ldots, \omega^{n-1} K_1)$ Po úprave môžeme polynóm $p(x)$ zapísať v tvare:

$$p(x) = A(x^n - K_1^n) + K_G$$

Zvoľme si ľubovoľný prvok t z poľa Z_q. Tento prvok reprezentuje nasledujúcu n-ticu

$$(t, t\omega, t\omega^2, ..., t\omega^i, ...t\omega^{n-1})$$

$$p(t) = p(t\omega) = p(t\omega^2) = ...p(t\omega^i) = ...p(t\omega^{n-2}) = p(t\omega^{n-1})$$

Rôznych n-tíc existuje $\frac{q-1}{n}$.

Ak si zoberieme 2 prvky t_1, t_2, ktoré reprezentujú 2 rôzne n-tice, tak hodnota $p(t_1) \neq p(t_2)$. Nech $p(t_1) = p(t_2)$, potom

$$A(t_1^n - K_1^n) + K_G = A(t_2^n - K_1^n) + K_G$$

$$t_1^n = t_2^n$$

$$\left(\frac{t_1}{t_2}\right)^n = 1$$

Oba prvky t_1, t_2 by patrili do rovnakej n-tici, čo je spor s predpokladom. □

V nasledujúcej vete dokazujeme, že nami navrhnutý protokol nemá zabezpečenú dôvernosť minulých, ani budúcich správ.

Veta 4.1.2. *Navrhnutý protokol nemá zabezpečenú dôvernosť minulých a budúcich správ.*

Dôkaz. Bez ujmy na všeobecnosti predpokladajme dôvernosť minulých správ. Nech skupina G má $n-1$ účastníkov, ktorí poznajú polynóm

$$p(x) = a_{n-1} x^{n-1} + ... + a_2 x^2 + a_1 x + a_0$$

Polynóm $p(x)$ je stupňa najviac $n-1$, pričom platí $\forall i \in G : \ p(K_i) = K_G$. Keď sa účastník u stane členom skupiny, tak podľa navrhnutého protokolu vygeneruje účastník s kľúčmi nový polynóm

$$q(x) = b_n x^n + b_{n-1}x^{n-1} + ... + b_2 x^2 + b_1 x + b_0$$

Polynóm $q(x)$ je stupňa najviac n, pričom platí $\forall i \in G \cup \{u\} : \ q(K_i) = K_G^{new}$. Polynóm $q(x)$ bude doručený všetkým členom skupiny a teda aj účastníkovi u. Predpokladajme, že účastník u pozná polynóm $p(x)$. Teraz ukážeme, ako účastník u zo znalosti polynómov $p(x)$, $q(x)$ a nového skupinového kľúča K_G^{new} dokáže určiť skupinový kľúč K_G.

$$q(x) - K_G^{new} = 0$$

Horeuvedená rovnica má n rôznych koreňov, pričom účastník u pozná jeden koreň, ktorý je K_u.

$$r(x) = \frac{q(x) - K_G^{new}}{x - K_u}$$

Rovnica $r(x) = 0$ má $n-1$ rôznych koreňov, ktoré sú $\{K_i \mid i \in G\}$.

$$p(x) \pmod{r(x)} = K_G$$

Účastník u sa dozvie predchádzajúci skupinový kľúč K_G. Pomocou skupinového kľúča K_G vie účastník u dešifrovať správy, ktoré boli posielané v skupine pred jeho príchodom do skupiny. Preto navrhnutý protokol nemá zabezpečenú dôvernosť minulých správ. Dôkaz dôvernosti budúcich správ je podobný.

□

Ľubovoľný člen skupiny vie zistiť všetky K_i. Stačí nájsť korene rovnice $p(x) - K_G = 0$. Použitím Berlekampovho algoritmu vieme efektívne faktorizovať $p(x) - K_G$ a tak vypočítať všetky K_i.

Modificated Polynomial-Based Protocol (MPBP)

Rovnako, ako v polynomial-based protokole, účastník s kľúčmi pošle členom skupiny polynóm. Člen i po dosadení kľúča K_i do polynómu dostane hodnotu Y_i, ktorá je nezávislá od ostatných hodnôt Y_j. Pomocou druhého kľúča K_i' a hodnoty Y_i vypočíta účastník i nový skupinový kľúč.

1. Pridanie do skupiny/Opustenie skupiny – účastník s kľúčmi vygeneruje nový skupinový kľúč K_G^{new}. Tento kľúč musí bezpečne poslať všetkým členom skupiny. Účastník s kľúčmi vygeneruje polynóm:

   $$p(x) = a_n x^n + a_{n-1}x^{n-1} + ... + a_2 x^2 + a_1 x + a_0$$

   Polynóm $p(x)$ je stupňa najviac n pričom koeficienty $a_n, a_{n-1}, ..., a_2, a_1, a_0$ sú z poľa Z_q, kde q je dostatočne veľké prvočíslo. Pre polynóm $p(x)$ platia nasledujúce podmienky:

- $\forall i \in G\ p(K_i) = Y_i$ kde $Y_i \cdot K'_i = K_G^{new}$.
- $p(0) = a_0$, kde a_0 je generované náhodne z poľa Z_q

Účastník s kľúčmi pošle správu M, v ktorej sa nachádzajú všetky koeficienty polynómu $p(x)$.

$$M = \langle a_n, a_{n-1}, ..., a_2, a_1, a_0 \rangle$$

Keď prijímateľ i dostane správu M, zrekonštruuje si polynóm $p(x)$. Do polynómu dosadí svoju hodnotu K_i, čím získa hodnotu Y_i a po vynásobení hodnotou K'_i dostane skupinový kľúč. Po každej distribúcii skupinového kľúča sa oba kľúče deterministicky posunú:

$$K_i = g^{K_i}$$
$$K'_i = g^{K'_i}$$

Počet kľúčov na strane odosielateľa sa oproti polynomial-based protokolu zvýši na $2n$. Každý prijímateľ má uložené dva kľúče. Dĺžka a počet správ na zmenu skupinového kľúča ostáva rovnaká ako v PBP.

Bezpečnosť konštrukcie

Predpokladajme, že účastník, ktorý nie je členom skupiny, získa polynóm $p(x)$. V súčasnosti neexistuje efektívny algoritmus, ktorý zo znalosti koeficientov polynómu $p(x)$ zistí aspoň jeden z kľúčov K_i, K'_i, K_G pre $1 \leq i \leq n$. Zo znalosti polynómu $p(x)$ nie je možné vylúčiť, že ľubovoľný prvok z poľa je potenciálny skupinový kľúč. Pre každý prvok $K_G \in Z_q$ existujú dvojice (K_i, K'_i) pre $1 \leq i \leq n$ také, že

$$p(K_1)K'_1 = \ldots = p(K_i)K'_i = \ldots = p(K_n)K'_n = K_G$$

Ak účastník pozná k rôznych polynómov, tak predpokladáme, že zo znalosti týchto polynómov nevie zistiť viac, ako keby poznal len jeden polynóm. Dôvodom je, že všetky kľúče, ktoré prislúchajú členom skupiny, sa po každej distribúcii menia. Pre účastníka, ktorý nepozná ani jednu z dvojíc (K_i, K'_i), kde $1 \leq i \leq n$, je táto zmena náhodná. Nech $p(x)$ a $r(x)$ sú polynómy, ktoré si účastník odchytil pri distribúcii členom skupiny. Nenašli sme žiadnu závislosť medzi polynómami $p(x)$ a $r(x)$. Preto sa domnievame, že aj keď účastník pozná viac rôznych polynómov, nevie o kľúčoch K_i, K'_i, K_G pre $1 \leq i \leq n$ povedať nič viac, ako keby poznal len jeden polynóm.

Predpokladajme, že účastník u, ktorý nebol členom skupiny, sa stal členom skupiny, t.j. účastník u pozná K_u, K'_u. Ďalej predpokladajme, že účastník u si odchytil aj polynóm $r(x)$, ktorý slúžil na vypočítanie skupinového kľúča pred príchodom účastníka u do skupiny. Účastník s kľúčmi vygeneroval a poslal členom skupiny polynóm $p(x)$ taký, že po dosadení $p(K_u) \cdot K'_u = K_G$. Účastník u nevie zo znalosti polynómu $r(x)$ vypočítať starý skupinový kľúč, a preto sa domnievame, že protokol má zabezpečenú dôvernosť minulých správ. Navyše účastník u nevie zo znalosti skupinového kľúča K_G a $p(x)$ určiť niektorú dvojicu kľúčov (K_i, K'_i) kde $K_i \neq K_u$ a $K'_i \neq K'_u$. Ku každej možnej hodnote kľúča $K_l \in Z_q$ existuje práve jedna hodnota K'_l taká, že: $K'_l = K_G \cdot p(K_l)^{-1}$, kde $p(K_l)^{-1}$ je inverzný prvok k $p(K_l)$ vzhľadom na násobenie.

Ak si účastník u spraví zoznam všetkých možných dvojíc kľúčov (K_l, K'_l), tak pri novej distribúcii polynómu $p'(x)$ vie overiť, či kľúče $(g^{K_l}, g^{K'_l})$ stále spĺňajú nasledujúcu rovnicu.

$$p'(g^{K_u})g^{K'_u} = p'(g^{K_l})g^{K'_l}$$

Ak tvrdenie neplatí, tak dvojica (K_l, K'_l) nezodpovedá žiadnemu členovi skupiny a účastník u ju môže vylúčiť. Na vypočítanie všetkých možných dvojíc potrebujeme prejsť celé pole Z_q, čo je pre dostatočne veľké q $(2^{128} + 17)$ prakticky nemožné.

Nech člen u opustí skupinu bez toho, aby sa dozvedel nejakú dvojicu kľúčov (K_i, K'_i), ktorá prislúcha inému členovi skupiny. Účastník s kľúčmi vygeneruje nový polynóm $p(x)$ pre nový skupinový kľúč K_G. Domnievame sa, že tento účastník je v rovnakej situácii, ako účastník, ktorý nikdy nebol členom skupiny, pretože z jeho pohľadu sú polynómy a k nim prislúchajúce skupinové kľúče na sebe nezávislé.

Z uvedených dôvodov sa domnievame, že modifikovaný polynomial-based protokol má zabezpečenú dôvernosť minulých aj budúcich správ.

Predpokladajme, že účastník u sa stal členom skupiny a dozvedel sa jednu dvojicu kľúčov (K_i, K'_i) prislúchajúcu inému členovi i. Ďalej predpokladajme, že pozná polynóm $r(x)$, ktorý prislúcha predchádzajúcemu skupinovému kľúču. Na to, aby z kľúčov K_i, K'_i zistil predchádzajúcu dvojicu kľúčov člena i, by musel vedieť riešiť problém diskrétneho logaritmu.

Navyše útok, ktorý sme prezentovali v časti 4.1.2 na zistenie minulých a budúcich správ, nie je možné aplikovať na modifikovaný polynomial-based protokol.

4.1.3 Hierarchia kľúčov

Protokoly, ktoré patria do skupiny "párové" (Pairwise), potrebujú pri odchode člena zo skupiny $O(n)$ správ na zmenu skupinového kľúča, kde n je počet členov skupiny. Dôvodom je, že KS pri zmene skupinového kľúča posiela nový skupinový kľúč každému členovi skupiny zvlášť, čo vyžaduje $O(n)$ správ.

Hlavnou myšlienkou protokolov, ktoré sú postavené na hierarchii kľúčov je skutočnosť, že účastník s kľúčmi zdieľa kľúče s rôznymi podskupinami. Keď účastník u opustí skupinu, účastník s kľúčmi na doručenie nového skupinového kľúča K_G používa podskupinové kľúče, ktoré sú neznáme účastníkovi u. Tento prístup redukuje počet potrebných správ na zmenu skupinového kľúča K_G. Do tejto kategórie protokolov patrí Local Key Hierarchy (LKH), ktorému sa budeme bližšie venovať v nasledujúcej časti.

Local Key Hierarchy LKH

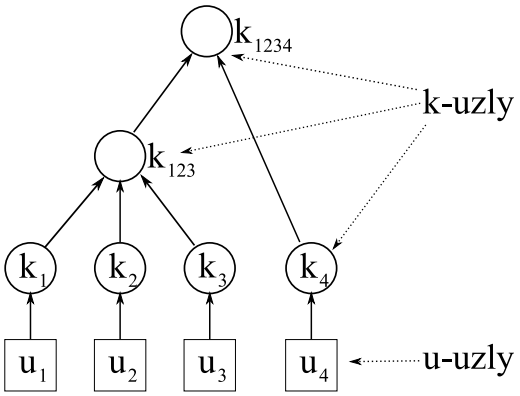
V protokole Local Key Hierarchy [15] má účastník s kľúčmi pre každého účastníka u zo skupiny uložený symetrický kľúč K_u, ktorý je známy účastníkovi u. Účastníci zo skupiny sú organizovaní do podskupín, pričom každej podskupine je priradený symetrický kľúč známy členom podskupiny a KS. Tieto informácie sú uložené v štruktúre, ktorá sa nazýva bezpečná skupina.

Definícia 4.1.2. *[15] Bezpečná skupina je trojica* (U, K, R)*, kde*

1. *U je neprázdna konečná množina členov skupiny.*

2. *K je neprázdna konečná množina kľúčov.*

3. *R je binárna relácia $R \subset U \times K$. Člen u pozná kľúč $k \Leftrightarrow (u, k) \in R$*



Obr. 4.2: Strom s kľúčmi

Na obrázku 4.2 je príklad hierarchie kľúčov, ktorá reprezentuje nasledujúcu bezpečnú skupinu:

- $U = \{u_1, u_2, u_3, u_4\}$
- $K = \{k_1, k_2, k_3, k_4, k_{123}, k_{1234}\}$
- $R = \{(u_1, k_1), (u_1, k_{123}), (u_1, k_{1234}), (u_2, k_2), (u_2, k_{123}), (u_2, k_{1234}), (u_3, k_3), (u_3, k_{123}), (u_3, k_{1234}), (u_4, k_4), (u_4, k_{1234})\}$

1. $keyset(u) = \{k|(u, k) \in R\}$ množina kľúčov, ktorá je známa členovi u.

2. $userset(k) = \{u|(u, k) \in R)\}$ množina členov, ktorá pozná kľúč k.

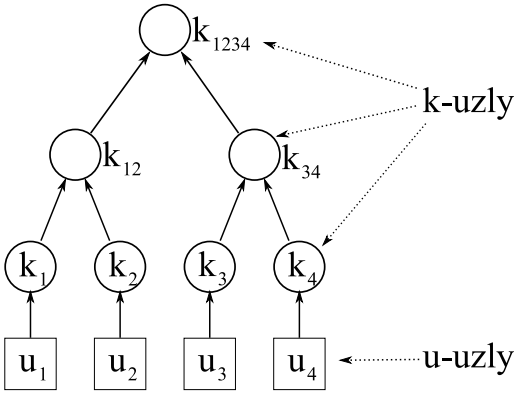
Keď člen u opustí bezpečnú skupinu (U, K, R) tak každý kľúč, ktorý bol známy účastníkovi u a inému účastníkovi z množiny U, musí byť zmenený.

1. Príchod do skupiny – predpokladajme, že účastník u dostal povolenie na prístup do skupiny (úspešne prebehla autentifikácia a riadenie prístupu pre u). Účastník s kľúčmi vytvorí pre nového člena u-uzol u a k-uzol k_u. Kľúč k_u je známy len KS a účastníkovi u. Účastník s kľúčmi nájde vhodný k-uzol k_j, ku ktorému pripojí uzol k_u (k_j je rodič k_u). Ak nechceme, aby mal účastník u prístup k predchádzajúcej komunikácií, účastník s kľúčmi musí zmeniť všetky kľúče na ceste od k_j po koreň stromu. Účastník s kľúčmi pošle zmenené kľúče všetkým potrebným členom skupiny.

2. Opustenie skupiny – predpokladajme, že účastník u dostal povolenie na opustenie skupiny. Účastník s kľúčmi vymaže u-uzol u a k-uzol k_u. Rodič k_l uzla k_u sa nazýva uzol odchodu. Aby účastník u nemal prístup k budúcej komunikácií, účastník s kľúčmi musí zmeniť všetky kľúče, ktoré sú reprezentované vrcholmi na ceste od k_j po koreň stromu. Po zmene ich doručí všetkým potrebným členom skupiny.

Pre zrozumiteľnosť vysvetlíme protokol na tomto príklade. Predpokladajme, že skupinu tvoria štyria členovia $\{u_1, u_2, u_3, u_4\}$. Účastník s kľúčmi vytvorí hierarchiu kľúčov, ako na obrázku 4.3. Skupinový kľúč k_{1234} je známy všetkým členom. Z obrázku je jasné, že:

- $keyset(u_1) = \{k_1, k_{12}, k_{1234}\}$
- $keyset(u_2) = \{k_2, k_{12}, k_{1234}\}$
- $keyset(u_3) = \{k_3, k_{34}, k_{1234}\}$
- $keyset(u_4) = \{k_4, k_{34}, k_{1234}\}$



Obr. 4.3: Strom s kľúčmi

Predpokladajme, že účastník u_2 dostal právo na opustenie skupiny. Účastník s kľúčmi vymaže u-uzol u_2 a k-uzol k_2. Účastník s kľúčmi musí zmeniť všetky kľúče, ktoré poznal účastník u_2 a ešte aspoň jeden člen skupiny. V našom prípade musí účastník s kľúčmi zmeniť kľúče $\{k_{12}, k_{1234}\}$. Aby nemal u_2 prístup k budúcim správam, KS vygeneruje nový skupinový kľúč k'_{1234} a ten doručí členom $\{u_1, u_2, u_3\}$. KS pošle správu:

$$M_{34} = E_{k_{34}}(k'_{1234}) \text{ pre členov } \{u_3, u_4\}$$

$$M_1 = \langle E_{k_1}(k'_{1234}), E_{k_1}(k'_{12}) \rangle \text{ pre člena } \{u_1\}$$

V správe M_1 posiela účastník s kľúčmi aj zmenený kľúč k_{12}.

Efektívnosť konštrukcie

Protokol má zabezpečenú dôvernosť minulých aj dôvernosť budúcich správ, pretože účastník s kľúčmi pri zmene členov skupiny zmení skupinový kľúč. Hlavnou výhodou navrhnutého protokolu je, že keď sa účastník u stane členom skupiny alebo skupinu opustí, odosielateľ (alebo KS) potrebuje len $O(\log n)$ správ na doručenie nového skupinového kľúča. Každý člen skupiny má uložených $\log_k n + 1$, kde k je stupeň vrchola v strome. Účastník s kľúčmi musí mať uložených $O(n)$ kľúčov. LHK je efektívnejší ako pairwise pre skupiny s veľkým počtom účastníkov. Dôvodom je, že počet správ pri zmene skupinového kľúča je logaritmický na rozdiel od lineárneho pri protokole pairwise.

4.2 Porovnanie

V tejto časti porovnávame efektívnosť protokolov na správu kľúčov, ktoré boli prezentované v tejto kapitole. Zameriame sa hlavne na tieto parametre:

1. Počet uložených kľúčov na strane prijímateľa.
2. Počet uložených kľúčov na strane odosielateľa.
3. Počet správ pri príchode – počet potrebných správ na distribúciu nového skupinového kľúča po príchode nového člena do skupiny.
4. Počet správ pri odchode – počet potrebných správ na distribúciu nového skupinového kľúča po odchode člena zo skupiny.

Protokol	Dôvernosť správ		Nezávislosť skupinových kľúčov
	minulých	budúcich	
GKMP	Áno	Áno	Áno
Secure Lock	Áno	Áno	Áno
LHK	Áno	Áno	Áno
PBP	Nie	Nie	Áno
MPBP	Áno*	Áno*	Áno

Tabuľka 4.1: Porovnanie protokolov z hľadiska bezpečnosti. Vlastnosti označené * sú hypotézy zdôvodnené v časti 4.1.2.

V predchádzajúcej tabuľke sme porovnali protokoly z hľadiska počtu správ pri zmene skupinového kľúča a z hľadiska počtu uložených kľúčov. V tabuľke 4.3 sú prezentované protokoly porovnané z hľadiska dĺžky správy, v ktorej je nachádza nový skupinový kľuč.

Protokol	Počet uložených kľúčov		Počet správ pri	
	odosielateľ	prijímateľ	príchode	odchode
GKMP	$n+2$	3	2	$O(n)$
Secure Lock	$2n$	2	1	1
LHK	$O(n)$	$O(\log n)$	$O(\log n)$	$O(\log n)$
PBP	n	1	1	1
MPBP	$2n$	2	1	1

Tabuľka 4.2: Porovnanie protokolov z hľadiska počtu správ a počtu uložených kľúčov.

Protokol	Dĺžka správy
GKMP	s
Secure Lock	$n \cdot s$
LHK	$\log n \cdot s$
PBP	$(n+1) \cdot \log q$
MPBP	$(n+1) \cdot \log q$

Tabuľka 4.3: Porovnanie dĺžky správ pri zmene skupinového kľúča. Parameter q vyjadruje veľkosť použitého poľa.

Kapitola 5

Autentifikácia odosielateľa

Autentifikácia odosielateľa je schopnosť členov skupiny overiť autentickosť odosielateľa správy a integritu správy. Od schémy, ktorá bude zabezpečovať autentifikáciu odosielateľa požadujeme, aby spĺňala nasledujúce bezpečnostné požiadavky:

- Integritu správy – definované v časti 3.2
- Odolnosť voči dohodám – definované v časti 3.2
- Autentickosť zdroja/odosielateľa – definované v časti 3.2

V unicaste je vo väčšine prípadov autentifikácia odosielateľa riešená pomocou MAC alebo digitálnych podpisov. Tieto kryptografické konštrukcie boli však navrhnuté pre unicast a použitie v multicaste je zvyčajne neefektívnym a neadekvátnym riešením. Hlavnou príčinou neefektívnosti je, že v multicaste je počet komunikujúcich účastníkov značne vyšší. V tejto kapitole poskytneme prehľad protokolov, ktoré autentifikujú odosielateľa v multicaste. V závere kapitoly sú všetky prezentované protokoly porovnané podľa relevantných kritérií.

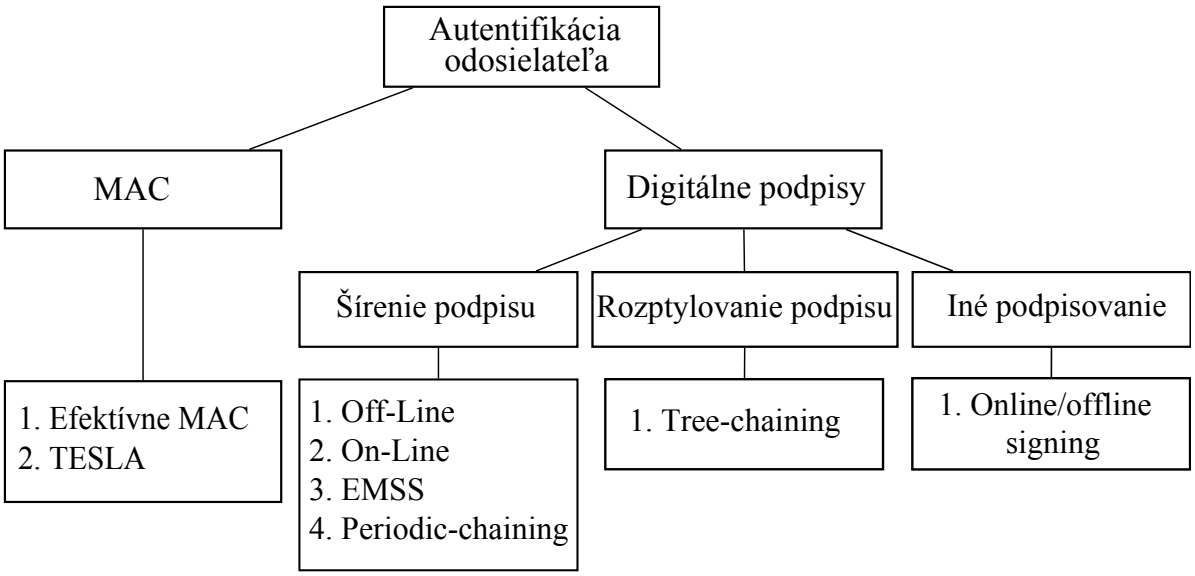
5.1 Prehľad riešeniami

Na obrázku 5.1 sú znázornené dva prístupy k autentifikácií odosielateľa.

1. MAC – v tejto skupine budeme prezentovať autentifikačné schémy, ktoré sú postavené na MAC. Prehľad protokolov, ktoré patria do tejto skupiny, poskytneme v časti 5.2.

   Predpokladajme, že odosielateľ aj prijímateľ poznajú kľúč K a algoritmus na výpočet MAC. Potom autentifikácia odosielateľa pomocou MAC prebieha nasledovne:

   (a) Keď odosielateľ chce poslať správu M, vypočíta hodnotu $M_O = MAC(K, M)$, ktorú pošle spolu so správou M. Odosielateľ teda pošle správu $M' = \langle M, MAC(K, M)\rangle$.



Obr. 5.1: Prehľad autentifikačných schém

(b) Autentifikácia na strane prijímateľa prebieha nasledovne: Prijímateľ dostane správu $M' = \langle M, MAC(K, M)\rangle$. Keďže prijímateľ pozná kľúč K, algoritmus na výpočet MAC a správu M, tak vie vypočítať hodnotu $M_P = MAC(K, M)$.

(c) Prijímateľ porovná hodnoty M_O, M_P. Ak sa nerovnajú, tak správu zamietne.

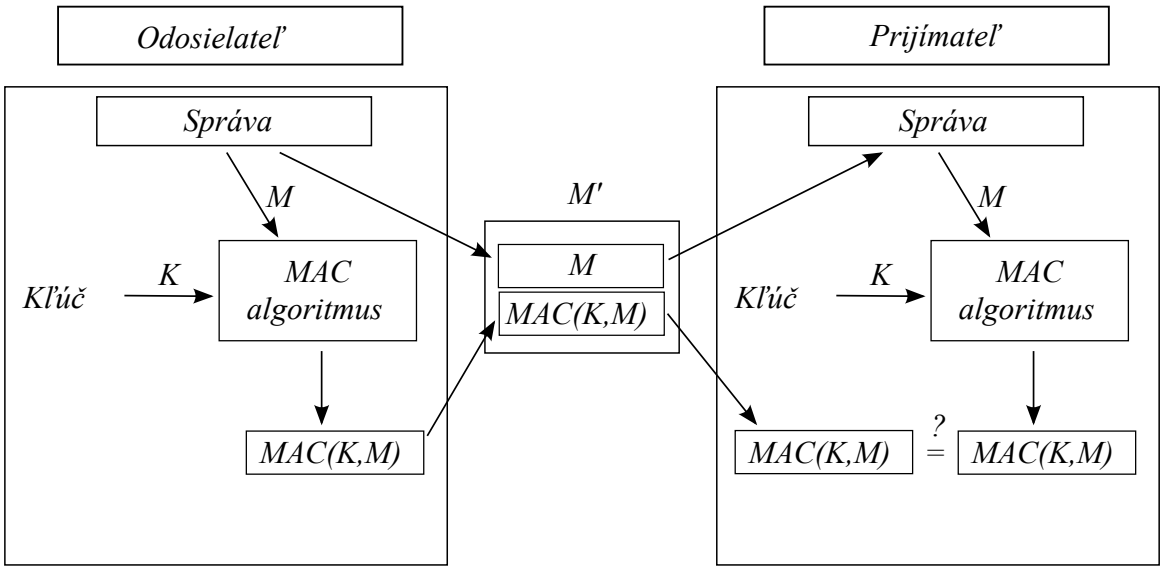
Tento spôsob autentifikácie odosielateľa je znázornený na obrázku 5.2.

V multicaste existuje skupinový kľúč, ktorý sa používa na šifrovanie komunikácie v skupine. Ak tento kľúč použijeme na autentifikáciu odosielateľa, tak prijímateľ bude vedieť overiť, či je odosielateľ členom skupiny. Použitím skupinového kľúča nemožno autentifikovať odosielateľa. Dôvodom je, že skupinový kľúč poznajú všetci členovia skupiny a tak ľubovoľný člen skupiny sa vie vydávať za platného odosielateľa.

Tento problém sa dá jednoducho vyriešiť pomocou tejto schémy: Nech odosielateľ S chce posielať správy skupine, ktorá má n účastníkov. Predpokladajme, že S má k dispozícii kľúče $K_1, K_2, \ldots, K_n$, kde K_i je kľúč, ktorý je známy len odosielateľovi S a účastníkovi i. Protokol na autentifikáciu odosielateľa môže prebiehať nasledovne:

(a) Keď chce S poslať správu M, autentifikuje ju s použitím kľúčov $K_1, ...K_n$. Potom so správou M pošle aj $MAC(K_1, M), MAC(K_2, M), ..., MAC(K_n, M)$.

(b) Keď prijímateľ i dostane správu, tak spočíta hodnotu $MAC(K_i, M)$ a porovná s hodnotou, ktorú dostal. Ak sa nezhodujú, správu odmietne.



Obr. 5.2: Autentifikácia odosielateľa pomocou MAC

Teraz bližšie preskúmame efektívnosť tohto riešenia. Odosielateľ na úspešné vypočítanie všetkých MAC-ov, musí mať uložených n kľúčov. Nech T_{MAC} je čas potrebný na výpočet jedného MAC-u, potom odosielateľ pred poslaním správy M spotrebuje čas $n \cdot T_{MAC}$ na výpočet všetkých potrebných MAC-ov. Navyše v každej správe bude mať informácia potrebná na autentifikáciu veľkosť $n \cdot |MAC|$. Efektívnosť prezentovaného riešenia klesá s rastúcim počtom členov skupiny. To znamená, že toto riešenie je prakticky nepoužiteľné pri väčšine multicastových aplikácií s veľkým počtom účastníkov.

Konštrukcie, ktoré budú prezentované v časti 5.2, sa snažia minimalizovať autentifikačnú informáciu, ktorá je posielaná spolu so správou.

2. Digitálne podpisy – autentifikačné schémy patriace do tejto skupiny sú postavené na digitálnych podpisoch. Digitálne podpisy sú riešením, ktoré zabezpečuje autentifikáciu odosielateľa. Ak chce odosielateľ poslať správu, podpíše ju svojím súkromným kľúčom. Prijímateľ dostane podpísanú správu. Na základe podpisu správy vie overiť identitu odosielateľa a integritu správy. Podpisovanie a následné overovanie každej správy je výpočtovo náročné, preto sa toto riešenie prakticky nepoužíva pri väčšine multicastových aplikácií, ktoré majú prebiehať v reálnom čase.

   Autentifikačné protokoly prezentované v tejto skupine sa snažia minimalizovať počet podpísaných správ. Prehľad protokolov, ktoré patria do tejto skupiny, uvedieme v časti 5.3.

5.2 MAC-BASE

5.2.1 Efektívna autentifikácia pomocou MAC

Je známe, že MAC sú efektívnejšie pri generovaní aj overovaní, ako digitálne podpisy. Vyžadujú však, aby všetci overovatelia mali prístup ku kľúču. V schéme navrhnutej v [1] pozná odosielateľ l kľúčov a každý člen skupiny má vlastnú podmnožinu týchto kľúčov. Aby sa útočník úspešne vydával za platného odosielateľa, potrebuje nadobudnúť všetkých l kľúčov.

Autentifikačná schéma dostane na vstup dva parametre w a q, kde w je maximálny počet účastníkov, ktorí spoločne spolupracujú a q je pravdepodobnosť, s akou má množina s w účastníkmi k dispozícií všetky kľúče účastníka u. Nech S je zdroj vysielania a u je prijímateľ v skupine. Základný protokol funguje nasledovne:

1. S vytvorí množinu kľúčov $R = K_1, ..., K_l$, kde $l = e(w+1)\ln(1/q)$.

2. Každý člen skupiny dostane podmnožinu kľúčov R. Pre prijímateľa u: $R_u \subset R$. Pričom každý kľúč K_i je pridaný do R_u s pravdepodobnosťou $1/(w+1)$, nezávisle pre každé i a u.

3. Keď chce S poslať správu M, autentifikuje ju s použitím kľúčov $K_1, ...K_l$. Potom so správou M pošle aj $MAC(K_1, M)$, $MAC(K_2, M)$, ..., $MAC(K_l, M)$.

4. Každý člen skupiny si vypočíta hodnoty MAC prislúchajúce k jeho množine kľúčov a porovná ich s MAC-ami, ktoré dostal. Ak sa niektorý z nich nezhoduje s MAC-om, ktorý dostal, správu odmietne.

Efektívnosť navrhnutej schémy je prezentovaná v nasledujúcich bodoch [1]:

1. Pamäťová zložitosť – v navrhnutej schéme musí mať zdroj uložených $P_S = l = e(w+1)\ln(1/q)$ MAC kľúčov. Každý prijímateľ musí mať uložených $P_P = e\ln(1/q)$ MAC kľúčov.

2. Časová zložitosť – ak zdroj S chce poslať správu M, musí spočítať všetky prislúchajúce MAC, čo bude trvať $T_S = P_S \cdot T_{MAC}$, kde T_{MAC} je čas potrebný na výpočet jednej hodnoty MAC. Čas potrebný na overenie doručenej správy bude $T_P = P_P \cdot T_{MAC}$, kde T_{MAC} je čas potrebný na výpočet a porovnanie hodnoty jedného MAC-u.

3. Dĺžka autentifikačnej informácie – zdroj S pridá do každej správy informáciu dĺžky $C = P_S \cdot |MAC|$, kde $|MAC|$ je dĺžka hodnoty MAC.

Sila tejto schémy závisí od pravdepodobnosti, s akou má množina účastníkov H s mohutnosťou w všetky kľúče R_u. Ak má množina účastníkov H všetky kľúče R_u, potom vedia predstierať odosielateľa pre prijímateľa u. Autor tejto schémy dokázal, že množina H vie autentifikovať správu M pre u s pravdepodobnosťou najviac $q + q'$, kde q' je pravdepodobnosť vypočítania hodnoty MAC bez poznania kľúča a q je pravdepodobnosť, s akou má množina s w účastníkmi k dispozícií všetky kľúče R_u, ktoré

prislúchajú účastníkovi u. Schéma je podrobnejšie skúmaná v [1], kde autor uvádza aj modifikáciu schémy, ktorá vyžaduje menšie náklady na komunikáciu. Hlavnou myšlienkou je, že autor používa MAC s jednobitovým výstupom (t.j MAC vracia hodnoty $0, 1$).

Efektívnosť konštrukcie

Toto riešenie nám ponúka efektívne generovanie a overovanie autentifikácie. Medzi ďalšie výhody patrí, že schéma je odolná voči stratám správ, pretože každá správa je overená samostatne. Hlavnou nevýhodou riešenia je neodolnosť voči spolupráci členov skupiny, ktorí vedia poskladať celú množinu kľúčov R, a tým vedia odoslať ľubovoľnú správu, ktorej budú veriť všetci členovia skupiny.

5.2.2 TESLA

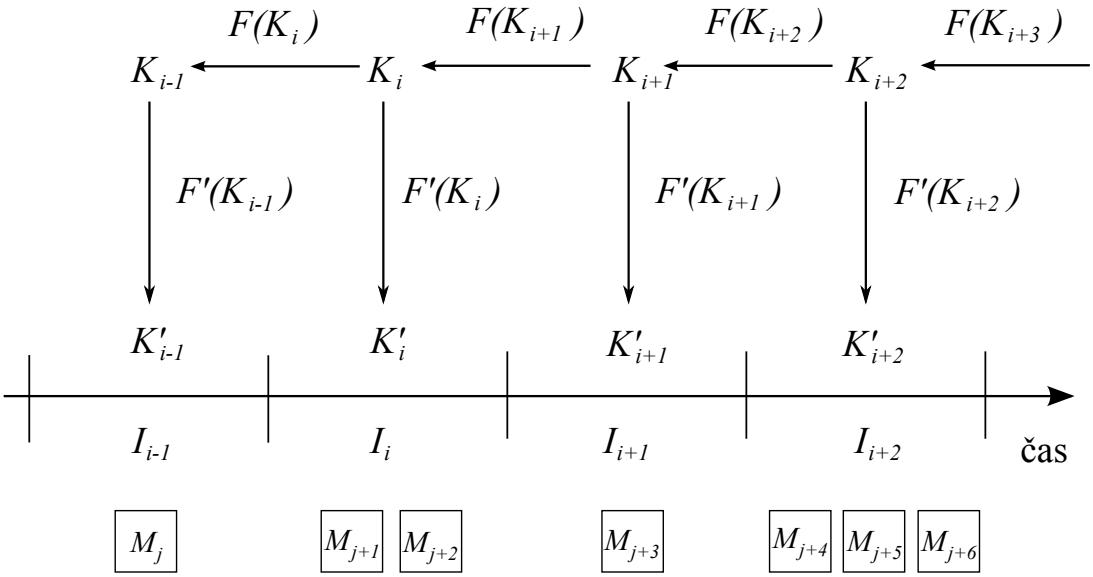
Druhou schémou, ktorá je založená na MAC, je TESLA (Timed Efficient Stream Loss-tolerant Authentication) [13, 14]. Hlavnou myšlienkou TESLA je, že účastník používa v každom časovom intervale na autentifikáciu správy iný kľúč. TESLA používa jednosmerné funkcie na vygenerovanie MAC kľúčov. Kľúč je použitý na autentifikáciu správy, tento kľúč zatiaľ ostáva odosielateľovi, aby sa potenciálny útočník nemohol vydávať za odosielateľa. Keď uplynie platnosť kľúča, odosielateľ zverejní kľúč. Prijímatelia poznajú tento kľúč a vedia overiť správy, ktoré dostali v časovom okamihu, keď kľúč platil. Ak útočník použije tento kľúč na to, aby sa vydával za odosielateľa, tak prijímatelia odmietnú túto správu, pretože odosielateľ a prijímatelia majú synchronizované hodiny a vedia, že je nový časový úsek a teda kľúč už nie je platný.

Predpokladajme, že odosielateľ a prijímatelia majú čiastočne synchronizovaný čas. Protokol na synchronizovanie času je prezentovaný v [13, 14].

Základný protokol môžeme rozdeliť do týchto krokov [13, 14]:

1. **Inicializácia odosielateľa** – Odosielateľ rozdelí čas vysielania na rovnaké časové intervaly I_i. Nech dĺžka každého intervalu je $T_{interval}$ a nech interval I_i začína v čase T_i, potom pre $T_i = T_0 + i \cdot T_{interval}$.
   Pred poslaním prvej správy si odosielateľ stanoví dĺžku vysielania a N, ktoré reprezentuje počet kľúčov v reťazi. Odosielateľ si zoberie náhodnú hodnotu K_N a pomocou jednosmernej funkcie F vygeneruje reťaz kľúčov. Kľúč $K_i = F(K_{i+1})$, tento kľúč je možné vygenerovať aj z kľúča K_N ($K_i = F^{N-i}(K_N)$, kde $F^j(k) = F^{j-1}(F(k))$, $F^0(k) = k$). Na výpočet autentifikačnej informácie je použitý kľúč $K' = F'(K)$, kde F' je pseudonáhodná funkcia. Každý kľúč je použitý na autentifikáciu správ v jednom časovom intervale. Napríklad kľuč K'_i je použitý pre interval I_i.

2. **Inicializácia nového prijímateľa** – Keď sa účastník stane členom skupiny, musí si synchronizovať čas s odosielateľom. Protokol na synchronizáciu času medzi odosielateľom a prijímateľom je podrobne vysvetlený v [14, 13]. Aby prijímateľ mohol autentifikovať správy, ktoré budú poslané odosielateľom, potrebuje poznať:



Obr. 5.3: autentifikácia odosielateľa pomocou MAC

- Začiatočný čas intervalu T_j, v ktorom sa stal účastník členom skupiny. Spolu s T_j pošle odosielateľ aj číslo intervalu I_j.
- Dĺžku trvania intervalu $T_{interval}$.
- Oneskorenie d je počet intervalov, po ktorých sa zverejní kľúč.
- Potvrdenie pre reťaz kľúčov K_i, kde $i < j - d$, kde j je aktuálny časový interval.

Z týchto informácií vyrobí odosielateľ správu M_0, ktorú pošle novému prijímateľovi. Je však nutné, aby prijímateľ vedel autentifikovať správu M_0.

3. **Posielanie správ** – Odosielateľ môže poslať ľubovoľne veľa správ v každom časovom intervale. Na autentifikáciu týchto správ bude použitý kľúč, ktorý prislúcha danému časovému intervalu. Predpokladajme, že odosielateľ chce poslať správu M_j v časovom intervale I_i. Odosielateľ pošle správu

$$M'_j = \langle M_j, MAC(K'_i, M_j), K_{i-d} \rangle$$

kde d je parameter oneskorenia a znamená, že kľúč K_i bude známy až v časovom intervale I_{i+d}.

4. **Overovanie na strane prijímateľa** – Keď prijímateľ dostane správu

$$M'_j = \langle M_j, MAC(K'_i, M_j), K_{i-d} \rangle$$

overí, či správa prišla bezpečne. Správa prišla bezpečne vtedy, keď si je prijímateľ istý, že odosielateľ nemôže byť v časovom intervale, v ktorom kľúč K_i je

zverejnený. Prijímateľ zatiaľ nevie autentifikovať správu M_j, pretože nepozná kľúč K_i a teda nepozná ani K'_i. Preto si uloží trojicu $(I_i, M_j, MAC(K'_i, M_j))$. Keď prijímateľ dostane správu, v ktorej sa nachádza kľúč K_i, overí, či nepozná kľúč K_j pre $j \geq i$. Ak je K_i najnovší kľúč, prijímateľ overí platnosť K_i pomocou $K_v = F^{i-v}(K_i)$ kde K_v je nejaký predchádzajúci už overený kľúč. Prijímateľ potom vypočíta $K'_i = F'(K_i)$ a pomocou kľúča K_i overí všetky správy, ktoré boli poslané v intervale I_i a všetkých predchádzajúcich intervalov, ku ktorým prijímateľ ešte nemá kľúč (kľúč môže byť odvodený z K_i).

Pre lepšie porozumenie protokolu ilustrujeme tento príklad: Na obrázku 5.3 vidíme reťaz kľúčov, kde F je jednosmerná funkcia. MAC kľúče sú odvodené s použitím jednosmernej funkcie F'. Z obrázku tiež vidíme správy, ktoré boli poslané v jednotlivých časových úsekoch. Pre každú správu použije odosielateľ kľúč zodpovedajúci danému časovému intervalu. Napríklad pre správu M_{j+2} vypočíta odosielateľ hodnotu MAC s použitím kľúča K'_i. Keď prijímateľ obdrží správu, uchová si ju. V tejto správe je uložený kľúč na autentifikáciu nejakej predchádzajúcej správy. Ak predpokladáme oneskorenie $d = 1$, tak správa M_{j+2} ponesie kľúč K_{i-1}, ktorý slúži na autentifikáciu správy M_j.

Efektívnosť konštrukcie

Na autentifikovanie správy potrebuje prijímateľ spočítať a porovnať iba jednu hodnotu MAC. TESLA je odolná voči stratám správ za predpokladu, že prijímateľ dostane nejakú neskoršiu správu. V tejto správe sa nachádza kľúč, z ktorého sa dajú odvodiť všetky predchádzajúce kľúče. Veľkou nevýhodou TESLA je synchronizácia medzi odosielateľom a prijímateľmi, pri ktorej môžu vznikať bezpečnostné riziká a dlhá odozva. Prijímateľ po prijatí správy nevie autentifikovať doručenú správu, pretože zatiaľ nepozná kľúč, ktorým by bol schopný autentifikovať doručenú správu. TESLA teda nemôže byť použitá v aplikáciách, ktoré majú bežať v reálnom čase. Niektoré z uvedených nevýhod sú vyriešené v [13].

5.3 Hash-BASE

Z obrázku číslo 5.1 môžeme vidieť, že existujúce riešenia vieme rozdeliť do týchto skupín:

1. Šírenie podpisu (signature propagation) – Hlavnou myšlienkou schém v tejto skupine je zreťazovanie správ. Správy vytvárajú reťaz, pričom v protokoloch Off-line, On-line je správa M'_i použitá na autentifikáciu správy M'_{i+1}. V protokole Periodic-chaining je správa M'_{i+1} použitá na autentifikáciu správy M'_i. V časti 5.3.1 poskytneme detailný prehľad autentifikačných schém, ktoré sú postavené na šírení podpisu.

2. Rozptyľovanie podpisu (Signature dispersal) – Hlavnou myšlienkou schém v tejto skupine je rozdelenie správ do blokov. Každý blok správ je potom podpí-

saný a podpis je pridaný do každej správy. V časti 5.3.2 poskytneme prehľad autentifikačných schém, ktoré sú založené na rozptyľovaní podpisu.

3. Iné podpisovanie – V časti 5.3.3 poskytneme prehľad autentifikačných schém, ktoré sú postavené na one-time podpisoch.

5.3.1 Šírenie podpisu

Off-Line [7] – predpokladáme, že zdroj pozná všetky správy ešte pred začatím vysielania. Ďalej predpokladáme, že zdroj má k dispozícii podpisovú schému a dva kľúče (súkromný SK, verejný PK). Nech H je hešovacia funkcia, ktorá spĺňa definíciu A.0.3. Protokol funguje nasledovne:

1. Zdroj chce poslať správy $M_1, M_2, M_3, \ldots, M_k$.

2. Zdroj transformuje správy na $M'_0, M'_1, M'_2, M'_3, \ldots, M'_k$, ktoré pošle skupine. Kde

$$M'_k = \langle M_k, 00\ldots0 \rangle$$

$$M'_i = \langle M_i, H(M'_{i+1}) \rangle \text{ pre } i = 1, \ldots, k-1$$

$$M'_0 = \langle H(M'_1, k), S(PK, H(M'_1, k)) \rangle$$

   V správe M'_0 zdroj posiela aj k, ktoré reprezentuje počet správ. Do poslednej správy M'_k, zdroj pridá $00\ldots0$, ktoré jednoznačne identifikujú poslednú správu. Počet núl je rovný veľkosti výstupu použitej hešovacej funkcie H.

3. Prijímateľ po prijatí správy M'_0 overí podpis správy verejným kľúčom. Ak je platný zo správy zistí k. Ak prijímateľ dostane správu $M'_i = \langle M_i, A_i \rangle$ pre $i = 1, \ldots, k-1$, tak M_i akceptuje len vtedy, ak platí $H(M'_i) = A_{i-1}$. Na autentifikovanie správ prijímateľ musí overiť podpis a vypočítať heš pre každú správu.

Offline protokol navrhnutý v [7] nie je funkčný. Prijímateľ po prijatí správy M'_0 nevie zistiť k, ani heš nasledujúcej správy M'_1. Pre korektné fungovanie protokolu stačí, aby odosielateľ do správy M'_0 pridal aj počet správ. Odosielateľ teda pošle správu:

$$M^*_0 = \langle k, H(M'_1, k), S(PK, H(M'_1, k)) \rangle$$

Prijímateľ po prijatí správy $M*_0$ overí podpis správy verejným kľúčom SK. Ak je platný, prijímateľ potom pozná počet správ, ktoré sa mu chystá odosielateľ poslať.

Prijímateľ však znalosť počtu správ nepotrebuje a preto je offline protokol možné vylepšiť tak, že odosielateľ pošle správu:

$$M^{**}_0 = \langle H(M'_1), S(PK, H(M'_1)) \rangle$$

Efektívnosť konštrukcie

Overovanie správ je efektívne, keďže stačí overiť podpis správy M'_0 a v zvyšných správach stačí spočítať a porovnať heše. Prijímateľ môže overiť správu hneď po prijatí. Hlavnou nevýhodou riešenia je strata správy. Ak sa stratí správa M'_i, tak reťaz je porušená a ďalšie správy M'_{i+1}, $M'_{i+2}, \ldots, M'_k$ nemôžu byť autentifikované. Zdroj pridá do každej správy autentifikačnú informáciu dĺžky $|d|$, kde $|d|$ reprezentuje dĺžku hešu. Aby bol zdroj schopný vytvoriť reťaz, musí poznať všetky správy ešte pred začatím vysielania. V niektorých prípadoch však zdroj nemá k dispozícií všetky správy ešte pred začatím vysielania. V týchto prípadoch sa protokol nedá použiť a preto bol navrhnutý protokol On-line, ktorý nevyžaduje znalosť všetkých správ ešte pred začatím vysielania.

On-Line [7] – v tomto protokole zdroj nepozná všetky správy pred začatím vysielania. Ďalej predpokladáme, že zdroj má k dispozícií podpisovú schému a dva kľúče (súkromný SK, verejný PK). Nech H je hešovacia funkcia, ktorá spĺňa definíciu A.0.3. Riešenie je postavené na one-time podpisoch. Rozdiel medzi digitálnym podpisom a one-time je v tom, že jednou inštanciou one-time podpisu sa dá podpísať len jedna správa. Predpokladajme, že zdroj má k dispozícií one-time podpisovú schému. Zdroj si vygeneruje niekoľko dvojíc one-time kľúčov (sk_i, pk_i) pre $i \geq 0$. Protokol funguje nasledovne:

1. Zdroj chce poslať správy $M_1, M_2, M_3, \ldots$.
2. Zdroj transformuje správy na $M'_0, M'_1, M'_2, M'_3, \ldots$, ktoré pošle skupine. Kde

   $$M'_0 = \langle pk_0, S(SK, pk_0)\rangle \text{ – digitálny podpis prvej správy}$$

   $$M'_i = \langle M_i, pk_i, S(sk_{i-1}(H(M_i, pk_i)))\rangle \text{ – one-time podpis}$$

3. Prijímateľ po prijatí správy M'_0 overí podpis správy. Ak je platný, tak dostane správny one-time kľúč pk_0. Ak sa prijímateľovi nepodarí overiť one-time podpis ľubovoľnej $M'_i, i \geq 1$, tak zamietne správu M'_i.

Efektívnosť konštrukcie

Na autentifikáciu správy sú použité one-time podpisy, ktoré sa dajú efektívne generovať a overovať. Prijímateľ vie autentifikovať správu hneď po prijatí, pretože v predchádzajúcej správe mu bol doručený one-time kľúč, ktorým môže autentifikovať správu. Hlavnou nevýhodou riešenia je strata správy. Ak sa stratí správa M'_i, tak reťaz je porušená a ďalšie správy M'_{i+1}, $M'_{i+2}, \ldots M'_k$ nemôžu byť autentifikované, pretože v správe M'_i je obsiahnutý kľúč, ktorý slúži na autentifikáciu M'_{i+1}. Zdroj pridá do každej správy informáciu dĺžky $|pk| + |\sigma|$, kde $|pk|$ je veľkosť one-time kľúča a $|\sigma|$ je veľkosť one-time podpisu.

EMSS Efficient Multi-chained Stream Signature– Perrig ponúkol riešenie EMSS, ktoré je čiastočne odolné voči strate správ. Keď chce zdroj poslať správu,

tak k tejto správe pridá heše niektorých ďalších správ. Z tejto správy vypočíta heš, ktorý bude pridaný do p správ, ktoré zdroj náhodne vyberie. Pri strate správy môžu byť ostatné autentifikované, len ak niektorá z doručených správ obsahuje heš správy, ktorú sa snažíme overiť. Autor tohto riešenia ukázal, že ak $p = 6$, tak 90% doručených správ môže byť autentifikovaných. Ak heš správy je pridaný do p správ, tak dĺžka autentifikačnej informácie, ktorá je pridaná do jednej správy je $p|d|$, kde $|d|$ je dĺžka hešu. V našom prípade uvažujeme $p = 6$, teda dostávame $6|d|$.

Periodic-chaining[8] – Autentifikačná schéma funguje podobne ako EMSS s rozdielom, že cieľové správy sú vyberané deterministicky a nie náhodne. Autor sa snaží navrhnúť protokol tak, aby po dlhodobej jednej strate správ $M_k, M_{k+1}, \ldots M_l$, bolo možné autentifikovať doručené správy. Predpokladáme, že zdroj má k dispozícií podpisovú schému a dva kľúče (súkromný SK, verejný PK). Nech H je hešovacia funkcia, ktorá spĺňa definíciu A.0.3. Hlavným princípom riešenia je:

1. Zdroj chce poslať správy $M_1, M_2, \ldots M_k$, pričom nie je nutné, aby ich poznal ešte pred začatím vysielania.

2. Zdroj transformuje správy na $M'_1, M'_2, \ldots M'_k$, kde

$$M'_1 = M_1$$

$$M'_i = \langle M_i, H(M'_{i-1})\rangle \text{ pre } i = 2 \ldots k-1$$

$$M'_k = \langle M_k, H(M'_{k-1}), S(SK, H(M_k, H(M'_{k-1}))\rangle$$

   Správa M'_k je podpísaná kľúčom SK.

3. Prijímateľ vie po prijatí správy $M'_i = \langle M_i, H(M'_{i-1})\rangle$ overiť platnosť správy M'_{i-1}. Ak sa heše nerovnajú, správu odmietne. Keď dostane správu, v ktorej sa nachádza digitálny podpis, overí platnosť prislúchajúcim kľúčom PK. Ak je platný, všetky správy sú autentifikované.

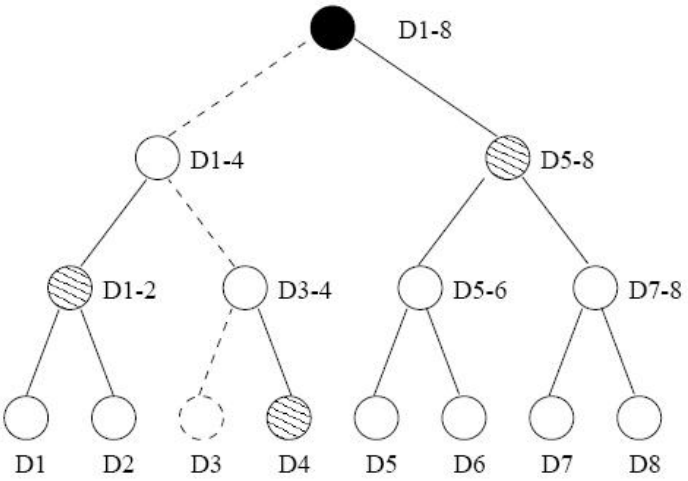
Toto riešenie je však len čiastočne odolné voči dlhodobej strate správ. Ak sa stratili správy $M'_i, M'_{i+1}, \ldots M'_l$, tak správy, ktoré vieme autentifikovať, sú $M'_{l+2}, \ldots M'_k$. V závislosti na veľkosti bufra (p) na strane odosielateľa prezentuje autor 2 schémy. Bez bufrovania to znamená $p = 1$. V tomto prípade ide o schému C_a, kde a je parameter schémy. C_a je periodická autentifikačná schéma definovaná takto [8]: heš správy M_i je pridaný do správ M_{i+1} a M_{i+a}, pričom posledná správa je podpísaná. Pre $p > 1$ vysvetľuje autor schému $C_{a,p}$, ktorej sa bližšie venuje v [8].

Efektívnosť konštrukcie

Schéma C_a je odolná voči jednej dlhodobej strate správ. (počet stratených správ však nesmie prekročiť $a-1$). Informácia na autentifikáciu správy je redukovaná na 2 heše, ktoré sú pridané do správy. Nevýhodou riešenia je, že schéma nie je optimalizovaná na viacej dlhodobých strát správ. Môže sa stať, že pri overovaní digitálneho podpisu bude podpis neplatný a tým aj všetky prijaté správy. To znamená, že prijímateľ vie autentifikovať správy až po overení podpisu, čo vytvára latenciu na strane prijímateľa.

5.3.2 Rozptyľovanie podpisu

Tree-chaining [16] – Wong a Lam ponúkli autentifikačnú schému, ktorej podstatou je podpísanie bloku správ. Každá správa nesie so sebou autentifikačnú informáciu a teda správa môže byť overená nezávisle od ostatných. Prúd dát je digitálne podpísaný blok po bloku, pričom blok tvorí k správ, kde k je závislé od aplikácie. Predpokladajme, že správy vytvoria binárny strom, pričom listy stromu sú heše správ. Každý vnútorný uzol reprezentuje heš svojich dvoch synov. Koreň stromu je tiež heš svojich dvoch synov. Každá odoslaná správa obsahuje digitálny podpis koreňa stromu, pozíciu správy v bloku a súrodencov všetkých uzlov na ceste od listu až ku koreňu. Po prijatí správy vie prijímateľ overiť cestu od listu ku koreňu, pričom vypočítava hodnoty vo vnútorných uzloch. Nakoniec porovná vypočítanú hodnotu koreňa s hodnotou, ktorú dostal. Ak sa nezhodujú, správu odmietne.



Obr. 5.4: Strom pre blok 8 správ

Pre zrozumiteľnosť vysvetlíme protokol na tomto príklade: Predpokladajme, že blok tvorí 8 správ $M_1, M_2, \ldots M_8$, ku ktorým zodpovedajúce heše sú $D_1, D_2, \ldots D_8$. Heše správ tvoria listy stromu ako na obrázku 5.4. Predpokladajme autentifikáciu správy 3, ktorej prislúchajúca cesta je znázornená čiarkovanou čiarou na obrázku 5.4. Prijímateľ musí vypočítať hodnotu každého vrcholu na tejto ceste. Prijímateľ vie spočítať heš správy D'_3, ktorú dostal. Postupne vypočíta hodnoty v uzloch $D'_{3-4} = H(D'_3|D_4)$, $D'_{1-4} = H(D_{1-2}|D'_{3-4})$. Nakoniec vypočíta heš koreňa stromu $D'_{1-8} = H(D'_{1-4}|D_{5-8}$. Pričom D_4, D_{1-2}, D_{5-8} sú heše, ktoré boli doručené so správou. Prijímateľ overí podpis, ktorý je posielaný so správou. Ak $D'_{1-8} = D_{1-8}$, správa je autentifikovaná. Inak je správa zamietnutá. Autor kvôli efektívnosti schémy odporúča zapamätať si vypočítané hodnoty vo vnútorných uzloch.

Efektívnosť konštrukcie

Výhodou riešenia je, že správy môžu byť overené nezávisle. Schéma je teda odolná voči strate správ. Hlavnou nevýhodou riešenia je, že autentifikačná informácia, ktorá je pridaná do každej správy, je príliš veľká. Ak blok obsahuje k správ, tak správa obsahuje $\log_d(k) - 1$ hešov (ktoré slúžia pri výpočte koreňa, kde d je stupeň vrchola v strome), digitálny podpis koreňa stromu a pozíciu správy v bloku. Toto množstvo informácie je dosť veľké a vytvára veľké náklady na komunikáciu. Navyše riešenie vyžaduje podpisovanie, ktoré je výpočtovo a časovo náročné pre obe strany (podpisovateľa aj overovateľa). Podrobnejšie o tejto schéme nájdeme v [16], kde je vysvetlený aj star-chain, ktorý je špeciálny prípad tree-chain.

5.3.3 Iné podpisovanie

Online/Offline podpisovanie [6]– Nech $D = (G, S, V)$ je podpisovacia schéma definovaná v A.0.4 a $o = (g, s, v)$ je one-time podpisovacia schéma definovaná v [6]. Ďalej predpokladajme, že zdroj má schémou D vygenerovanú dvojicu kľúčov $\langle PK, SK\rangle$. Kľúč PK je zverejnený a známy všetkým členom skupiny. Autentifikačná schéma je postavená na digitálnych podpisoch a na one-time podpisoch. Kvôli prehľadnosti budú značky pri one-time podpisoch malými písmenami a značky pre digitálny podpis veľkými písmenami. Autentifikačná schéma je rozdelená do dvoch fáz [6]:

- off-line predspracovanie – v tejto fáze odosielateľ nechá vygenerovať generátorom g (zo schémy o) dvojicu one-time kľúčov $\langle sk, pk\rangle$. Táto dvojica kľúčov je uložená a bude použitá v druhej fáze. Odosielateľ podpíše kľúč pk podpisovacím algoritmom S zo schémy D s použitím kľúča SK.

  $$\Sigma = S(SK, pk)$$

  Táto fáza prebieha nezávisle na správe. To znamená, že odosielateľ môže, ale nemusí poznať správu, ktorú neskôr pošle. Pre efektívnosť si môže odosielateľ vygenerovať a podpísať viacero dvojíc kľúčov, ktoré použije v druhej fáze.

- on-line podpisovanie – v tejto fáze prebieha samotné podpisovanie správy M schémou o. Odosielateľ pošle správu M', ktorá obsahuje správu M, one-time kľúč, digitálny podpis one-time kľúča a one-time podpis správy M. Formálnejšie:

  $$M' = \langle M, pk, \Sigma, \sigma\rangle \text{ kde } \sigma = s(sk, M),\ \Sigma = S(SK, pk)$$

Proces overovania na strane prijímateľa prebieha nasledovne:

- Prijímateľ prijme správu $M' = \langle M, pk, \Sigma, \sigma\rangle$.
- Overí platnosť one-time kľúča overovacím algoritmom V zo schémy D. Ak $V(PK, \Sigma, pk) = 1$, prijímateľ si uloží one-time kľúč pk, ktorý použije na overenie platnosti one-time podpisu. Ak bol digitálny podpis neplatný, prijímateľ správu zamietne.

- Prijímateľ overí platnosť one-time podpisu algoritmom v zo schémy o. To znamená, že overí, či platí $v(pk, \sigma, M) = 1$. Ak je one-time podpis neplatný, prijímateľ správu zamietne.

Správa je autentifikovaná len vtedy, keď nadobúda nasledujúci výraz hodnotu 1.

$$V(PK, \Sigma, pk) \wedge v(pk, \sigma, M)$$

Efektívnosť konštrukcie

Každá správa je podpísaná one-time podpisovacou schémou, nezávisle na ostatných správach. Každá prijatá správa môže byť nezávisle autentifikovaná. To znamená, že schéma je odolná voči stratám správ. Hlavnou nevýhodou tejto schémy je, že overovateľ (prijímateľ) musí overiť digitálny podpis one-time kľúča a one-time podpis správy. Veľkosť autentifikačnej informácie, ktorá je posielaná s každou správou, je $|pk| + |\Sigma| + |sigma|$, kde $|\Sigma|$ je dĺžka digitálneho podpisu kľúča pk, $|\sigma|$ je dĺžka one-time podpisu správy M a $|pk|$ je veľkosť one-time kľúča, ktorý bol vygenerovaný generátorom g (zo schémy o). Je známe, že veľkosť one-time kľúčov môže byť veľmi veľká, čo zväčšuje autentifikačnú informáciu, ktorá je posielaná spolu so správou.

5.4 Porovnanie

V tejto časti porovnáme efektívnosť autentifikačných schém, ktoré boli prezentované v tejto kapitole. Zameriame sa hlavne na tieto parametre:

1. Latencia odosielateľa – znamená, že si odosielateľ na autentifikovanie správy M_i potrebuje uložiť správu M_j, kde $j > i$. Zo správy M_j sa väčšinou vypočíta autentifikačná informácia, ktorá bude pridaná do M_i. To znamená, že odosielateľ pošle správu M_i až potom, ako spracuje správu M_j. Príkladom autentifikačnej schémy, ktorá má latenciu na strane odosielateľa, je protokol off-line, kde odosielateľ spracováva správy od poslednej smerom k prvej.

2. Latencia prijímateľa – znamená, že prijímateľ po prijatí správy M_i nevie autentifikovať odosielateľa tejto správy. Autentifikačná informácia pre správu M_i je posielaná v niektorej z neskorších správ M_j, $i < j$. To znamená, že prijímateľ si musí správy ukladať a M_i vie autentifikovať až po prijatí M_j.

3. Odolná voči stratám správ – znamená, že prijímateľ vie autentifikovať odosielateľa aj v prípade strate správ. V tabuľke pod pojmom "Áno" budeme rozumieť, že keď odosielateľ pošle n správ, z ktorých sa stratí $n - 1$ správ, tak prijímateľ vie prijatú správu autentifikovať. V protokole TESLA čiastočne znamená, že vieme autentifikovať doručenú správu za predpokladu, že po strate správy bude doručená nejaká neskoršia správa. V protokole EMSS autor dokázal, že ak je heš správy pridaný do $k = 6$ cieľových správ, tak 90 % doručených správ môže byť autentifikovaných. Protokol periodic chaining je odolný voči jednej dlhodobej strate správ. Ak je počet posebe idúcich stratených správ menší ako $a-1$, kde a je parameter protokolu, tak všetky doručené správy môžu byť autentifikované.

4. Dĺžka autentifikačnej informácie – dĺžka informácie, ktorú odosielateľ pridá do správy, aby prijímateľ vedel autentifikovať odosielateľa.

5. Nemožnosť popretia (Non-repudation) – odosielateľ nevie poprieť správu, ktorú poslal. Autentifikačné schémy, ktoré sú postavené na digitálnych podpisoch majú zabezpečenú túto vlastnosť. To znamená, že všetky autentifikačné schémy, ktoré boli prezentované v časti digitálne podpisy majú v tabuľke "Áno". Autentifikačné schémy, ktoré sú prezentované v časti autentifikácia pomocou MAC nemajú zabezpečenú nemožnosť popretia.

Prehľad symbolov použitých v tabuľke 5.1:

- $|MAC|$– dĺžka výstupu MAC funkcie
- $|K|$ – dĺžka kľúča
- $|d|$ – dĺžka hešu
- $|pk|$ – dĺžka one-time verejného kľúča
- $|\sigma|$ – dĺžka one-time podpisu
- $|\Sigma|$ – dĺžka digitálneho podpisu
- n – počet správ v bloku
- k – stupeň vrchola v strome

Protokol	Latencia odosielateľa	Latencia prijímateľa	Odolná voči stratám správ	Nemožnosť popretia	Dĺžka autentifikačnej informácie
Efficient-MAC	Nie	Nie	Áno	Nie	$l\|MAC\|$
TESLA	Nie	Áno	Čiastočne	Nie	$\|MAC\| + \|K\|$
off-line	Áno	Nie	Nie	Áno	$\|d\|$
on-line	Nie	Nie	Nie	Áno	$\|pk\| + \|\sigma\|$
EMSS	Nie	Áno	Čiastočne	Áno	$6\|d\|$
Periodic chaining C_a	Nie	Áno	Čiastočne	Áno	$2\|d\|$
Tree-chaining	Áno	Nie	Áno	Áno	$\|\Sigma\| + (\log_k(n) - 1)\|d\|$
on/off-line podpisovanie	Nie	Nie	Áno	Áno	$\|\Sigma\| + \|pk\| + \|\sigma\|$

Tabuľka 5.1: Porovnanie protokolov

Zjednotenie autentifikačnej informácie

Ako vidno z tabuľky 5.1, dĺžka autentifikačnej informácie je v tomto tvare len ťažko porovnateľná. Na zjednotenie dĺžky autentifikačnej informácie použijeme kryptografický parameter $s = 128$ bit.

1. Ak požadujeme bezpečnosť 2^s hešovacej funkcie, tak dĺžka výstupu hešovacej funkcie musí byť $2s$. Dôvodom je možnosť implementácie narodeninového útoku na hešovacie funkcie.

2. Nech funkcia MAC dáva výstup dĺžky s, pričom je použitý kľúč K, ktorého dĺžka je s.

3. Ak predpokladáme, že digitálne podpisy sú vytvárané pomocou RSA podpisovej schémy, tak na dosiahnutie bezpečnosti 2^s treba použiť RSA-3072. To znamená, že digitálny podpis bude mať dĺžku 3072 bitov.

4. Ak predpokladáme, že one-time podpisy sú vytvárané pomocou Lamportovej podpisovej schémy, tak veľkosť verejného kľúča je $1024 \cdot s$ a veľkosť one-time podpisu je $512 \cdot s$. V one-time podpisových schémach je závislosť medzi dĺžkou verejného kľúča pk a dĺžkou one-time podpisu σ. Čím menší verejný kľúč pk použijeme, tým väčší bude one-time podpis.

5. V protokole Efficient-MAC [1] používa autor parametre $w = 10$ a $q = 10^{-3}$. Potom môžeme vypočítať $l = e(w + 1)\ln(1/q) = e \cdot 11 \cdot \ln(1000) = 207$. To znamená, že odosielateľ pridá do každej správy 207 hodnôt MAC.

6. V protokole Tree-chaining [16] predpokladáme, že blok tvorí 16 správ, pričom v protokole je použitý binárny strom. To znamená, že odosielateľ pridá do každej správy $\log_2(16) - 1 = 3$ heše. To zodpovedá dĺžke autentifikačnej informácie $6 \cdot s$.

Protokol	Dĺžka autentifikačnej informácie v bitoch
Efficient-MAC	$207 \cdot s$
TESLA	$2 \cdot s$
off-line	$2 \cdot s$
on-line	$1536 \cdot s$
EMSS	$12 \cdot s$
Periodic chaining C_a	$4 \cdot s$
Tree-chaining	$3072 + 6 \cdot s$
on/off-line podpisovanie	$3072 + 1536 \cdot s$

Tabuľka 5.2: Porovnanie protokolov

Dodatok A

Použité vety a definície

Kvôli definícií šifrovania a dešifrovania si zadefinujme nasledujúce množiny: [12]

1. A reprezentuje konečnú množinu, ktorú budeme nazývať abeceda.
2. M je množina, ktorá reprezentuje priestor správ. M obsahuje slová nad abecedou A. Jeden prvok z M sa nazýva otvorený text.
3. C je množina, ktorá reprezentuje šifrový priestor. Jeden element z C sa nazýva šifrový text.

Definícia A.0.1. [12] *Šifrovanie a dešifrovanie*

1. *K reprezentuje množinu kľúčov.*
2. *Každý prvok $e \in K$ definuje bijektívnu funkciu $E_e : M \to C$. Funkcia E_e sa nazýva šifrovacia funkcia alebo šifrovacia transformácia.*
3. *Každý prvok $d \in K$ definuje bijektívnu funkciu $D_d : C \to M$. Funkcia D_d sa nazýva dešifrovacia funkcia alebo dešifrovacia transformácia.*
4. *Šifrovacia schéma pozostáva z množiny $\{E_e : e \in K\}$ a prislúchajúcej množiny $\{D_d : d \in K\}$, takže $\forall e \in K\ \exists d \in K$ taký, že $\forall m \in M$ platí:*

$$D_d(E_e(m)) = m$$

5. *Dvojica kľúčov $\langle e, d\rangle$ tvorí kľúčový pár. Kľúče e, d sa môžu rovnať. Ak sa $e = d$, tak ide o symetrické šifrovanie. Ak $e \neq d$, tak sa jedná o asymetrické šifrovanie.*

Veta A.0.1. [4] *Čínska zvyšková veta Nech $N_1, N_2, \ldots, N_n$ sú po dvojiciach nesúdeliteľné čísla, t.j.*

$$\forall i, j\ (1 \leq i < j \leq n) :\ nsd(N_i, N_j) = 1$$

nech $m_1, m_2, \ldots, m_n$ sú celé čísla a $L = N_1 \cdot N_2 \cdot \ldots \cdot N_n$. Potom sústava kongruencií:

$$X \equiv m_1 \pmod{N_1}$$

$$X \equiv m_2 \pmod{N_2}$$

...

$$X \equiv m_n \pmod{N_n}$$

má práve jedno riešenie $X \in \langle 0, L-1 \rangle$*, ktoré je:*

$$X = \left(\sum_{i=1}^{n} \frac{N_i}{L} \cdot m_i \cdot f_i \right) \pmod{L}, \text{ kde } f_i \cdot \frac{L}{N_i} \equiv 1 \pmod{N_i}$$

Definícia A.0.2. *Message Authentification Code (MAC)[12] je trieda funkcií, ktoré sú parametrizované kľúčom k, pričom sú splnené nasledujúce vlastnosti:*

- *ľahko vypočítateľná* (*easy of computation*)
  Pre známu funkciu h_k, hodnotu k a vstup x je jednoduché spočítať $h_k(x)$. Hodnotu $h_k(x)$ budeme väčšinou označovať $MAC(k, x)$.

- *kompresia* (*compression*)
  Funkcia h_k dostane na vstupe x, kde x je reťazec ľubovoľnej konečnej dĺžky a hodnota $h_k(x)$ je pevnej dĺžky.
  Navyše pre danú triedu funkcií h a pre každý platný kľúč k (neznámy útočníkovi) platí nasledujúca vlastnosť:

- *výpočtovo odolná*(*computation-resistance*)
  Nech $\langle x_i, h_k(x_i) \rangle$ je konečný počet dvojíc $\langle$ text - MAC $\rangle$ pre $i \geq 0$,. Potom pre $x \neq x_i$ nie je možné vypočítať hodnotu $h_k(x)$ (ani za predpokladu, že $h_k(x) = h_k(x_i)$ pre nejaké i).

Definícia A.0.3. *Hešovacia funkcia h je efektívne vypočítateľná funkcia.*

$$h : X \to Y$$

kde Y je konečná množina, X môže, ale nemusí byť konečná množina.

Hodnotu $x \in X$ nazývame správou alebo vzorom, hodnotu $h(x)$ najčastejšie hešom alebo odtlačkom. V práci používame hešovacie funkcie, ktoré navyše spĺňajú tieto požiadavky [12]:

- *jednosmernosť* (*preimage resistance*)
  Pre dané $y \in Y$ nevieme efektívne nájsť jeho vzor $x \in X$ tak, aby platilo $h(x) = y$.

- *slabú odolnosť voči kolíziám* (*second preimage resistance*)
  K vzoru $x \in X$ nevieme efektívne nájsť $x' \in X \setminus x$ s takým istým obrazom $(f(x) = f(x'))$.

- *silnú odolnosť voči kolíziám*
  Nevieme efektívne nájsť dva vzory $x_1, x_2 \in X, x_1 \neq x_2$ také, pre ktoré platí $h(x_1) = h(x_2)$.

Definícia A.0.4. [7] *Podpisovacia schéma je trojica* (G, S, V)*, kde:*

- G *je generátor kľúčov, ktorý na vstupe* 1^n *vráti dvojicu* $(PK, SK) \in \{0,1\}^{2n}$*, kde* SK *sa nazýva súkromný (podpisovací) kľúč a* PK *sa nazýva verejný (overovací) kľúč.*
- S *je podpisovací algoritmus, ktorý dostane na vstupe súkromný kľúč* SK *so správou* M *a na výstupe vráti podpis správy* $\sigma = S(SK, M)$*.*
- V *je overovací algoritmus. Pre každú dvojicu* $(PK, SK) = G(1^n)$ *a* $\sigma = S(SK, M)$ *vráti hodnotu* $V(PK, \sigma, M) = 1$*.*

Od podpisovej schémy požadujeme: Nech $\langle x_i, S(SK, x_i)\rangle$ *je konečný počet dvojíc* $\langle$ *text - podpis* $\rangle$ *pre* $i \geq 0$*. Potom pre útočníka, ktorý nepozná sukromný kľúč* SK*, nie je možné vypočítať hodnotu* $S(SK, x)$ *pre* $x \neq x_i$ *tak, aby platilo*

$$V(PK, S(SK, x), x) = 1$$

.

Kapitola 6

Záver

V práci sme riešili otázku bezpečnej komunikácie v multicaste.

Medzi protokoly, ktoré sme prezentovali patria: GKMP [3, 9, 10], Secure Locks [4], LKH [15]. Ku každému z týchto protokolov sme uviedli spôsob, akým sa distribuuje skupinový kľúč členom skupiny. O protokole Secure Lock sme v diskusii uviedli, že veľkosť správy je vo väčšine prípadov väčšia ako v triviálnom riešení, ktoré sme prezentovali v kapitole 4. V časti 4.1.2 sme navrhli protokol na správu kľúčov v skupine. Hlavnou myšlienkou polynomial-based protokolu je, že pri zmene skupinového kľúča účastník s kľúčmi zostrojí nový polynóm $p(x)$ taký, že $p(K_1) = p(K_2) = ... = p(K_n) = K_G^{new}$. Dokázali sme, že tento protokol nemá zabezpečenú dôvernosť minulých ani budúcich správ. Tiež sme ukázali, že každý člen skupiny vie pomocou Berlekampovho algoritmu zistiť zvyšné kľúče ostatných členov skupiny. Na odstránenie týchto vlastností sme navrhli modifikovať polynomial-based protokol tak, ako je to v časti 4.1.2. Domnievame sa, že modificated polynomial-based protokol má zabezpečenú dôvernosť minulých aj budúcich správ, pričom efektívnosť je porovnateľná s už známym protokolom Secure Locks.

Prezentované protokoly sme porovnali podľa relevantných kritérií. Porovnávali sme tieto vlastnosti: dôvernosť minulých a budúcich správ, nezávislosť kľúčov, počet potrebných kľúčov na strane prijímateľa a odosielateľa, počet a dĺžka správ potrebných na zmenu skupinového kľúča.

V kapitole 5 sme sa venovali autentifikácii odosielateľa. Medzi existujúce protokoly, ktoré zabezpečujú autentifikáciu odosielateľa, patria: Efficient-MAC [1], TESLA [14], Off-line [7], On-line [7], Periodic-chaining [8], Tree-chaining [16],Online/Offline podpisovanie [6]. Súčasťou popisu protokolu Off-line je oprava prvej posielanej správy. Pred opravou, ktorú sme vykonali, bol tento protokol len ťažko použiteľný. Prijímateľ by musel po prijatí prvej správy uhádnuť počet posielaných správ, aby mohol druhú správu autentifikovať. Zvolili sme kryptografický parameter s, ktorý reprezentuje požadovanú bezpečnosť. Pre každý protokol sme určili dĺžku autentifikačnej informácie, ktorú protokol vyžaduje na dosiahnutie bezpečnosti s. Porovnanie týchto dĺžok sme uviedli v tabuľke 5.2. Ďalej sme tieto protokoly porovnali z hľadiska latencie na strane prijímateľa a odosielateľa, odolnosti voči stratám správ, nemožnosti popretia. Celkové porovnanie je v tabuľke 5.1.

Literatúra

[1] Ran Canetti, Juan Garay, Gene Itkis, Daniele Micciancio, Moni Naor, and Benny Pinkas. Multicast security: A taxonomy and some efficient constructions. In *INFOCOMM'99*, 1999.

[2] Y. Challal, H. Bettahar, and A. Bouabdallah. A taxonomy of multicast data origin authentication: issues and solutions, 2004.

[3] Y. Challal and H. Seba. Group key management protocols: A novel taxonomy. *International Journal of Information Technology*, 2(2):105–118, july 2005.

[4] Guang-Huei Chiou and Wen-Tsuen Chen. Secure broadcasting using the secure lock. *IEEE Transactions on Software Engineering*, 15(8):929–934, 1989.

[5] S. Deering. *Host Extensions for IP Multicasting*, August 1989. Internet Engineering Task Force, RFC 1112, STD 5.

[6] Shimon Even, Oded Goldreich, and Silvio Micali. On-line/off-line digital signatures. *Journal of Cryptology*, 9(1):35–67, 1996.

[7] Rosario Gennaro and Pankaj Rohatgi. How to sign digital streams. In *CRYPTO '97: Proceedings of the 17th Annual International Cryptology Conference on Advances in Cryptology*, pages 180–197, London, UK, 1997. Springer-Verlag.

[8] P. Golle and N. Modadugu. Authenticating streamed data in the presence of random packet loss, 2001.

[9] H. Harney and C. Muckenhirn. Group key management protocol (gkmp) architecture, 1997.

[10] H. Harney and C. Muckenhirn. Group key management protocol (gkmp) specification, 1997.

[11] Paul Judge and Mostafa Ammar. Security issues and solutions in multicast content distribution:a survey. *IEEE Network*, pages 30–36, 2003.

[12] Alfred J. Menezes, Scott A. Vanstone, and Paul C. Van Oorschot. *Handbook of Applied Cryptography*. CRC Press, Inc., Boca Raton, FL, USA, 1996.

[13] A. Perrig, R. Canetti, D. Tygar, and D. Song. The TESLA broadcast authentication protocol, 2002.

[14] Adrian Perrig, Ran Canetti, Dawn Song, and J. D. Tygar. Efficient and secure source authentication for multicast. In *In Network and Distributed System Security Symposium, NDSS ?1*, pages 35–46, 2001.

[15] Chung Kei Wong, Mohamed Gouda, and Simon S. Lam. Secure group communications using key graphs. *IEEE/ACM Trans. Netw.*, 8(1):16–30, 2000.

[16] Chung Kei Wong and Simon S. Lam. Digital signatures for flows and multicasts. *IEEE/ACM Trans. Netw.*, 7(4):502–513, 1999.