

ROMÂNIA
MINISTERUL APĂRĂRII NAȚIONALE
ACADEMIA TEHNICĂ MILITARĂ
„FERDINAND I”

FACULTATEA DE SISTEME INFORMATICE ȘI SECURITATE CIBERNETICĂ
Specializarea: CALCULATOARE ȘI SISTEME INFORMATICE PENTRU APĂRARE ȘI
SECURITATE NAȚIONALĂ



ALGORITM SIMETRIC DE INTERSCHIMBARE A CHEILOR CRIPTOGRAFICE

CONDUCĂTOR ȘTIINȚIFIC:

Lt.col. prof. dr. ing. MIHAI TOGAN

ABSOLVENT:

Sd. Sg. Maj. Claudiu-George PURICE

Conține _____ file

Inventariat sub nr. _____

Poziția din indicator: _____

Termen de păstrare: _____

BUCUREȘTI

2020

NECLASIFICAT

NECLASIFICAT

NECLASIFICAT

NECLASIFICAT

NECLASIFICAT

Abstract

Key exchange schemes are essential for cryptography and for information security in general. There are many mechanisms to achieve this, all of them having both advantages and disadvantages. The password authentication variety presents a very convenient way to exchange keys. Passwords are usually low entropy secrets and thus are easy to remember for humans. It may seem that being able to do a dictionary search renders any password based cryptography insecure, however, password authentication key exchange schemes can be proved to be as secure as any other schemes for key exchange. The basis for the dictionary attack is that the password space being small, each entry can be tried by the attacker until they find the right password. The strength of a secure password authentication protocol comes from forcing the attacker to interact with an honest party for every guess, making the attack obvious, in order to determine if the guess is the right password or not. This paper describes and analysis the Dragonfly protocol, a provably secure password authentication key exchange. An implementation is also provided.

Cuprins

Abstract.....	6
Introducere.....	9
Importanța temei, utilitatea unui astfel de sistem.....	9
Securitatea informatică.....	9
Criptografie.....	9
Scopul și obiectivele lucrării.....	10
Scopul lucrării.....	10
Obiectivele proiectului.....	10
Rezumatul lucrării pe capitole, sumarul rezultatelor obținute.....	11
Capitolul doi.....	11
Capitolul trei.....	12
Capitolul patru.....	12
State of the art.....	13
Problematica managementului cheilor criptografice.....	13
Etapale principale.....	13
Forward secrecy.....	14
Autentificare.....	15
Schimbul de chei autentificat cu parolă.....	16
Certificate digitale.....	17
Schimbări aduse de tehnologia quantum.....	19
Protocoale de schimb de chei.....	21
Diffie-Hellman (DH).....	21
Password-Authenticated Key (PAK).....	22
Simple Password Exponential Key Exchange (SPEKE).....	24
Ring Learning With Errors (RLWE).....	25
Quantum Key Derivation (QKD).....	28
Protocolul Dragonfly.....	30
Premise.....	30
Pași.....	33
Atacuri.....	35
Scenarii de utilizare.....	36
Soluții similare, implementări existente.....	37
Detalii privind implementarea practică.....	38
Definirea cerințelor complete pentru sistemul propus.....	38
Prezentarea cazurilor de utilizare.....	39
Definirea arhitecturii și proiectarea sistemului.....	40
Clasa SecureSocket.....	41

Clasa Peer.....	41
Clasa Element.....	42
Structura ParameterSet.....	42
Tehnologii/API –uri folosite pentru implementarea sistemului.....	42
NTL.....	43
OpenSSL.....	43
Descriere cazuri de testare și elaborare raport de testare.....	43
Concluzii.....	47
Sinteza principalelor idei din lucrare.....	47
Direcții și idei pentru continuarea lucrării.....	47
Bibliografie.....	49

Introducere

Importanța temei, utilitatea unui astfel de sistem

Securitatea informatică

Sistemele informatice au devenit o necesitate în cadrul oricărei organizații, atât militare cât și civile. Sistemele de calcul au evoluat foarte mult și au întrecut capacitățile umane în multe domenii. Deoarece acestea pot juca un rol atât de important, un atac informatic poate avea până la cele mai grave consecințe. Pentru a asigura securitatea informatică trebuie îndeplinite trei principii. Confidențialitatea datelor se referă la garanția că unui atacator îi este imposibil să obțină accesul la date la care nu îi este permis în mod deliberat. Integritatea datelor asigură faptul că nicio informație nu poate fi modificată de potențiali atacatori. Disponibilitatea datelor este principiul care garantează că informațiile trebuie să poată fi accesate atunci când sunt necesare.

Criptografie

Confidențialitatea și integritatea datelor pot fi asigurate de un sistem criptografic bun, evitând multe din atacurile din cadrul sistemelor informatice. Criptografia a existat încă din antichitate, dar singurele metode considerate sigure după evoluția rapidă pe care au avut-o sistemele de calcul se bazează în totalitate pe aceste sisteme. Conceptul de bază a rămas însă același. Datele sunt criptate folosind o informație secretă numită cheie, sunt trimise către destinație, iar acolo sunt decriptate folosind o cheie pentru decriptare. Cheia pentru criptare poate fi una și aceeași cu cheia pentru decriptare, în cazul criptografiei simetrice, sau pot exista două chei diferite care au o legătură între ele astfel încât să îndeplinească o anumită proprietate matematică, în cazul criptografiei asimetrice. Unele din cele mai importante informații de care se folosește un sistem considerat securizat sunt aceste chei criptografice. Un sistem de schimb de chei nu reprezintă altceva decât o metodă prin care cheile criptografice pot fi create în mod sigur.

Scopul și obiectivele lucrării

Scopul lucrării

Această lucrare are ca scop studiul managementului cheilor criptografice și în mod special studiul și analiza protocoalelor pentru schimbul de chei. Se dorește analiza atât din punctul de vedere al criptografiei clasice cât și al criptografiei bazate pe tehnologia cuantum.

În aceasta va fi analizat în mod special protocolul Dragonfly. Vor fi expuse tipurile de atacuri la care protocolul este vulnerabil, dar și atacuri la care este rezistent. Atacurile vor fi prezentate în raport cu alte protocoale asemănătoare, pentru evaluarea realistă a securității protocolului. Pentru o analiză a performanței și în special pentru analiza aplicabilității protocolului va fi folosită o implementare proprie, conform standardului. Implementarea va fi documentată în detaliu în prezenta lucrare.

Obiectivele proiectului

- Analiza comparativă a mai multor protocoale care îndeplinesc rolul de schimb de chei criptografice
- Analiza detaliată a protocolului Dragonfly
- Implementarea practică a protocolului Dragonfly asigurând compatibilitatea pentru mai multe platforme
- Documentarea tuturor detaliilor de implementare în această lucrare
- Realizarea unor teste și măsurători privind performanța implementării și compararea acesteia cu alte implementări similare

Rezumatul lucrării pe capitole, sumarul rezultatelor obținute

Capitolul doi

Acest capitol descrie nivelul dezvoltării științifice din domeniu, la momentul de timp al scrierii lucrării.

La începutul acestui capitol este explicat în detaliu la ce se referă problematica managementului cheilor criptografice. Este definit conceptul de forward secrecy, precum și cele de cheie de sesiune și cheie cu termen lung. Sunt explicate atacurile de tip Man-in-the-Middle. Problema principală este cea a verificării identității participanților, proces numit autentificare. Sunt prezentate posibile soluții la această problemă care se folosesc de certificate digitale precum infrastructura cu chei publice sau rețelele de încredere. Tot în această secțiune sunt prezentate și ideile generale din spatele protocoalelor de schimb de chei cu autentificare pe bază de parolă.

În următoarea secțiune sunt analizate câteva protocoale concrete folosite pentru schimbul de chei criptografice. Pentru fiecare este descris atât algoritmi de generare a mesajelor schimbate cât și atacuri la care protocolul este vulnerabil, dacă au fost găsite astfel de atacuri.

În secțiunea trei se urmărește în particular protocolul Dragonfly din punct de vedere teoretic. Se pleacă de la premisele protocolului așa cum sunt descrise în standard, fiind descrise toate considerațiile necesare implementării acestuia și presupunerile făcute. Este descris algoritmul după care sunt generate mesajele și explicat fiecare pas. Plecând de la o parolă se obține un element într-un grup ales, are loc un schimb de mesaje pentru obținerea unui secret comun și după aceea are loc un al doilea schimb cu rolul de a asigura ambii participanți că sunt în posesia unei chei valide și că cealaltă parte este în posesia aceleiași chei. Tot în această secțiune sunt descrise atacuri ce au putut fi făcute asupra versiunilor mai vechi ale protocolului și scenariile în care a fost implementat în practică acest schimb de chei și anume WPA3.

Cea de a patra secțiune numește implementări open-source deja existente, împreună cu aspecte practice și detalii de implementarea despre acestea. Este găsită o

implementare în Python și analizată din punctul de vedere al performanței și al respectării standardului.

Capitolul trei

Acesta deservește descrierea detaliilor despre implementarea practică a protocolului Dragonfly.

Capitolul începe prin definirea cerințelor pentru sistemul propus. Se dorește implementarea fiecărui pas al protocolului cu respectare strictă a standardului.

În a doua secțiune sunt urmărit cazurilor de utilizare ale fiecărui nivel în parte. Sistemul poate genera chei, face autentificare și criptare și decriptarea datelor comunicate prin rețea.

În secțiunea trei este definită arhitectura și proiectarea sistemului. Împărțirea este făcută în trei clase, pentru calcule matematice necesare protocolului, pentru generarea mesajelor specifice protocolului și pentru folosirea unei chei la criptarea și decriptarea datelor.

Următoarea secțiune descrie tehnologiile și componentele software externe folosite în implementare, precum OpenSSL și biblioteca pentru teoria numerelor (NTL – Number Theory Library) în C++.

Ultima secțiune conține planul și raportul de tastare ale sistemului. Sunt făcute teste calitative pentru verificarea generării chei în scenariul în care nu există erori și în scenariul în care autentificare ar trebui să eșueze. Sunt făcute teste calitative pentru măsurarea timpului de procesare.

Capitolul patru

Al patrulea și cel din urmă capitol reprezintă concluziile lucrării. Prima secțiune sintetizează ideile principale, începând de la securitatea protoalelor de schimb de chei bazate pe autentificare cu parolă și ajungând la implementarea proprie și ceea ce aduce aceasta în plus față de cele existente.

În cea de a doua secțiune sunt oferite direcții de studiu pentru continuarea lucrării precum integrarea proiectului în proiecte deja folosite pe scară largă.

State of the art

Problematica managementului cheilor criptografice

Un mesaj poate avea unul sau mai mulți destinatari. Pentru a asigura securitatea informațională trebuie ca numai destinatarii reali să poată să aibă acces la acesta. Criptarea datelor permite acest lucru impunând condiția ca emițătorul și toți destinatarii reali și numai emițătorul și destinatarii reali să aibă acces la cheia criptografică. A folosi un set predefinit de chei pentru fiecare posibilitate devine foarte repede impractic, deoarece numărul scenariilor crește în mod exponențial în funcție de numărul posibilităților destinatari. Politicile și procedurile folosite pentru a soluționa această problemă într-un mod practic definesc managementul cheilor criptografice.

Etapele principale

Managementul cheilor criptografice reprezintă un domeniu mai larg care încorporează generarea, distribuirea, stocarea și ștergerea cheilor. În general problematica poate fi împărțită în 3 etape: schimbul, stocarea și folosirea.

Schimbul de chei criptografice

În cazul cheilor asimetrice, participanți își generează o pereche de chei, o cheie publică și una privată, și o transmit în clar pe cea publică. În situația cheilor simetrice acest lucru nu poate fi realizat pe un canal nesigur, deoarece ambii participanți trebuie să aibă acces la aceeași cheie, iar transmiterea acesteia în clar ar însemna că orice atacator poate să o intercepteze și astfel să descifreze orice mesaj criptat. Aceasta a constituit mult timp o problemă pentru algoritmi simetrici. Cea mai veche soluție este folosirea unui canal securizat, dar în timp au apărut și alte metode care nu se bazează pe existența unui astfel de canal precum criptarea chei cu un algoritm asimetric. Aceasta se întâlnește sub denumirea de criptare hibridă, deoarece folosește atât criptografie asimetrică cât și criptografie simetrică. O altă metodă este reprezentată de protocoale specializate pentru schimbul de chei. Acestea generează un secret comun la care au acces doar utilizatorii protocolului, iar pentru un adversar, sub presupunerea că acesta este un ascultător pasiv, nu este

fezabil din punct de vedere computațional de obținut acest secret folosind numai datele trimise de participanți.

Stocarea de chei criptografice

După generarea cheilor în orice modalitate ar fi fost făcută, acestea trebuie stocate într-un mod sigur altfel un atacator ar putea obține acces la acestea și ar compromite securitatea tuturor comunicațiilor. Soluții pentru aceasta includ utilizarea de module hardware de securitate sau folosirea unei aplicații software care stochează cheile în mod criptat și oferă acces la acestea doar după parcurgerea un pas de autentificare. O altă soluție utilizată de multe ori în completarea primei soluții este stocarea chei în memoria volatilă și distrugerea acesteia imediat după ce a fost folosită.

Folosirea de chei criptografice

Cea mai mare problemă din acest punct de vedere este perioada de timp pentru care o anumită cheie este folosită. Dacă cheile sunt schimbate des, un posibil atacator care reușește să obțină o cheie va avea acces numai la datele care au fost criptate cu respectiva cheie. În situația în care aceeași cheie este folosită mereu, expunerea acesteia ar compromite toate datele acelui sistem. Din acest motiv este bine ca schimbul de chei să aibă loc des.

Forward secrecy

În practică în urma unui schimb de chei se obține ceea ce se numește cheie de sesiune. Aceasta este stocată în memoria volatilă, folosită pentru criptarea mesajelor și distrusă la final. Pe lângă aceasta există, de multe ori, chei cu termen lung necesare pentru autentificare și folosite pentru a genera cheile de sesiune.

Termenul de „forward secrecy” este o proprietate pe care o pot avea sistemele criptografice. Această proprietate garantează faptul că în situația în care o cheie cu termen lung este compromisă, cheile de sesiune generate pe baza acesteia nu vor fi și ele compromise. Cu alte cuvinte mesajele criptate în trecut nu pot fi decriptate de un atacator prin găsirea chei cu termen lung. Acest scenariu reiese din modul în care sunt stocate cheile. Compromiterea unei mașini poate duce la accesul la toate datele stocate în mod persistent de aceasta, însă cheile de sesiune folosite anterior

pentru criptarea datelor nu vor mai fi disponibile după ce se termină schimbul de mesaje.

O variantă mai slabă a acestei proprietăți este definită ca garantând același lucru însă numai pentru sesiunile în care atacatorul nu a trimis sau modificat mesaje.

Autentificare

Necesitatea autentificării și atacul Man-in-the-Middle

Atacurile pot în general să fie împărțite în două categorii pasive și active. Atacurile pasive se referă la ascultare tuturor mesajelor trimise între participanți. Pe de altă parte un atac activ este definit ca un atac în care adversarul are capacitatea de a intercepta, modifica sau trimite orice fel de mesaj. Pe un canal nesecurizat, un atacator poate face mai mult decât să fie un ascultător pasiv. Pe multe canale, precum internetul sau rețelele publice, un atacator poate manipula traficul în orice modalitate dorită.

Atacul de tip Man-in-the-Middle presupune interceptarea tuturor mesajelor încă de la începutul conversației și retransmiterea lor sau trimiterea unor mesaje puțin modificate în locul celor interceptate. Astfel un atacator poate pretinde față de emițător că este destinatarul real al mesajelor, iar față de destinatar pretinde că el este emițătorul original. În contextul în care doi participanți onești nu au niciun fel de cunoștință unul despre celălalt acest tip de atac este imposibil de prevenit. Astfel pentru ca un schimb de mesaje să fie cu adevărat sigur, participanți trebuie să se autentifice.

Modalități de autentificare

Procesul de demonstrare a identității este numit autentificare. Acesta se poate face prin unul sau mai mulți din următorii trei factori. Utilizatorii își pot demonstra identitatea prin ceva ce numai aceștia cunosc, ceva ce numai aceștia au sau ceva ce aceștia sunt. Exemple de cunoștințe care pot demonstra identitatea utilizatorului sunt parole, PIN-uri sau răspunsuri la întrebări de securitate. Posesii care sunt folosite pentru autentificare sunt în general carduri inteligente, module criptografice hardware sau software dedicate sau dispozitive electronice precum telefoane inteligente care conțin module software dedicate. Ceva ce un utilizator este și poate garanta identitatea acestuia se referă de obicei la date biometrice care

pot identifica în mod unic orice om, precum amprente, modelul irisului, secvențe de ADN, fața, vocea sau altele.

Un alt criteriu de clasificare al metodelor de autentificare este cine se autentifică. În timp ce în scenarii simetrice toți participanții trebuie să își demonstreze identitatea, în scenariile de tip client-server autentificarea poate fi atât mutuală cât și unilaterală.

Schimbul de chei autentificat cu parolă

În zilele noastre una din modalitățile predominante de autentificare este bazată pe parole. Multe protocoale folosesc parole sau date derivate pe baza lor pentru a demonstra cunoașterea parolei și prin aceasta dau dovada identității. Aceste protocoale de schimb de chei cu autentificare bazată pe parolă sunt numite în literatura de specialitate în limba engleză PAKE sau „Password Authenticated Key Exchange”.

PAKE echilibrat

Numită „Balanced PAKE” în limba engleză, această categorie se descrie situațiile în care doi sau mai mulți participanți cunosc aceeași parolă pe care o folosesc pentru a deriva o cheie criptografică. Aceasta poate părea un pas înapoi, deoarece pare a reintroduce necesitatea de a avea un secret comun înaintea execuției protocolului. Diferența este însă că din punct de vedere criptografic o parolă este văzută ca un secret cu entropie mult mai mică decât o cheie criptografică. În timp ce o cheie este considerată sigură atunci când dimensiunea acesteia este de ordinul sutelor de biți, pentru o parolă cerințele sunt mult mai mici. Aceasta trebuie să poate fi creată și reținută de un om. De cele mai multe ori poate fi reconstruită cu ușurință folosind un dicționar. Din acest motiv pentru protocoalele de schimb de chei care se bazează pe parolă un criteriu important este ca un atac de tip dicționar să nu se poată face off-line. Dacă un atacator ar strânge anumite date în urma unui atac, fie pasiv fie activ, asupra unui astfel de protocol, acele date nu trebuie să îi fie suficiente să verifice toate parolele posibile. O condiție necesară pentru securitatea unui astfel de protocol este ca o încercare a atacatorului de a ghici parola să nu poată fi invalidată fără a rula protocolul cu un participant onest. Astfel un atac de tip dicționar ar putea fi detectat cu ușurință de participanții onești.

PAKE augmentat

Numit în engleză „Augmented PAKE”, aceasta este o variantă în care se face distincția între participanți care au rolul de client și cei care au rolul de server. Diferența față de un protocol echilibrat apare la nivelul serverului care, în situația unui protocol PAKE augmentat, nu cunoaște parola în clar. Prin această metodă, dacă un server este compromis atacatorul nu poate obține parola pentru a pretinde că acesta este un client onest.

Certificate digitale

În criptografia asimetrică cheile de criptare și decriptare sunt separate, existând o relație matematică între ele. Așadar este posibil ca un mesaj criptat cu o cheie să fie decriptat cu cealaltă. Pentru criptarea unui mesaj, cheia de criptare este făcută publică de către cel ce urmează să îi fie destinatar care însă păstrează cheia pentru decriptare privată. Criptografia asimetrică pe lângă idea de chei diferite pentru criptare și decriptare, a introdus și conceptul de semnătură digitală. O semnătură digitală se folosește în mod invers de cheia privată și de cea publică. După aplicarea unei operații folosind cheia privată, numai posesorul chei private poate pretinde că a trimis un anumit mesaj, iar acesta nu poate repudia trimiterea lui, în timp ce oricine altcineva poate verifica.

Problema autentificării este astfel transformată într-o problemă de asociere a unei chei publice cu o identitate asumată. Un certificat digital reprezintă exact această asociere, între o cheie publică și anumite date care să determine identitatea. Problema devine astfel de natura de a avea sau nu încredere într-un certificat digital.

Infrastructura cu chei publice

Cel mai folosit model care să răspundă la problema încrederii este un model ierarhic. Infrastructura cu chei publice, în engleză „Public Key Infrastructure” sau PKI, se bazează pe un astfel de model. O astfel de infrastructură asigură crearea, distribuirea, stocarea dar și revocarea certificatelor digitale în mod centralizat.

Componentele necesare unei PKI sunt următoarele

- o autoritate de certificare asigură stocarea, eliberarea și semnarea certificatelor;

- o autoritatea de înregistrare verifică identitatea entităților ale căror certificate sunt stocate la autoritate de certificare;
- un directorul central reprezintă o modalitate sigură de stocare și indexare pentru cheile criptografice;
- un sistem de management al certificatelor stabilește accesul la certificate stocate sau livrarea certificatelor eliberate;
- o politică de certificare definește cerințele pentru procedurile infrastructurii pentru a permite părților externe să analizeze și să determine nivelul de încredere în respectiva infrastructură.

Încrederea într-o anumită autoritate de certificare este în general oferită printr-un certificat semnat de o altă autoritate mai înaltă. Astfel se construiește modelul ierarhic care are în vârf o autoritate de certificare care își semnează singură propriile certificate și este denumită autoritate în limba engleză „Root certificate authority”. Aceste certificate auto-semnate vin preinstalate de fabricanții sistemului de calcul, de dezvoltatorii aplicației software sau pot fi instalate manual de utilizator. Calea în ierarhie până la certificatul auto-semnat se numește lanț de încredere și trebuie verificat la verificarea fiecărei semnături.

Rețele de încredere

O soluție diferită pentru problema încrederii este reprezentată de un model descentralizat. În literatura de specialitate termenul în limba engleză folosit este „Web of trust”. O rețea de încredere, asemănător cu o infrastructură cu chei publice, funcționează folosind aceleași principii ale certificatelor digitale care să ateste legătura între o cheie publică și o entitate. Diferența este dată de locul din care vine încrederea în semnatarul certificatului. În timp de în PKI un utilizator se încrede într-un certificat numai după ce verifică lanțul de încredere, într-o rețea de încredere utilizatorul va verifica dacă certificatul este semnat de un alt membru al rețelei care prezintă deja încredere. Astfel fiecare utilizator poate să aleagă cine prezintă încredere. În timp orice utilizator va acumula și distribui împreună cu cheia sa publică semnături de la alți utilizatori. [1]

Schimbări aduse de tehnologia quantum

În domeniul mecanici cuantice multe fenomene par a se opune înțelegerii noastre clasice asupra fenomenelor fizice. Acestea duc la posibilități noi, care ar fi greu de imaginat, folosind numai cunoștințe clasice. Una din cele mai bine cunoscute aplicații a mecanici cuantice în criptografie este schimbul de chei. O altă schimbare este impusă în mod indirect algoritmilor criptografici, prin posibilitatea utilizării proprietăților cuantice în criptanaliză.

Avantaje aduse criptografiei

Nedeterminismul cuantic contrazice ideea clasică conform căreia starea unui sistem fizic este determinată în mod unic de totalitatea valorilor proprietăților măsurabile ale respectivului sistem și proprietățile sunt, în mod similar, determinate în mod unic de starea în care se află sistemul.

La nivel cuantic toate proprietățile măsurabile sunt văzute ca niște distribuții de probabilități, nu ca valori exacte. Deoarece proprietățile cuantice sunt definite numai de o distribuție de probabilități, măsurarea oricărei valori duce la modificarea distribuției și deci schimbarea stării sistemului cuantic. Aceasta este baza teoremei imposibilității clonării informației cuantice.

Un exemplu de proprietate măsurabilă este polarizarea unui foton. Alegerea unei măsurători va perturba starea fotonului și va schimba astfel rezultatul pe care alte măsurători l-ar da. Dacă polarizarea unui foton este măsurată în aceeași bază de mai multe ori la rând atunci toate măsurătorile vor arăta același rezultat, deoarece după prima măsurătoare polarizarea este cunoscută în acea bază cu probabilitate 1. Dacă baza este însă schimbată, a doua măsurătoare va avea probabilități diferite de zero pentru multiple valori posibile.

Principiul incertitudinii impune o limită superioară asupra informațiilor pe care le putem obține dintr-un sistem cuantic. Considerând cazul în care observăm polarizarea unui foton individual, principiul incertitudinii interzice obținerea a mai mult de un bit de informație. Măsurarea se poate face în mai multe moduri. Un exemplu este la în baza rectilinie, verificând unghiurile 0° și 90° sau altul este baza diagonală, sub unghiurile 45° și 135° . Dacă polarizarea unui foton este măsurată în baza diagonală pe direcția de 45° , când aceasta va fi măsurată în baza rectilinie probabilitățile celor două valori posibile, 0° și 90° , vor fi egale și nici un fel de informație despre nu poate fi extrasă.

Prin transmiterea informației codificate în stări cuantice se poate elimina un tip de atac inevitabil în mod clasic, atacul pasiv. Deoarece informația cuantică nu poate fi clonată, dacă un atacator încearcă să asculte un mesaj, mesajul își va schimba starea cuantică și atacul va putea fi detectat. Pentru aceasta nu este necesar un computer cuantic ci un canal de comunicație care să păstreze informația codificată în proprietăți cuantice și dispozitive care să facă codificarea și decodificarea datelor trimise.

În practică există deja implementări atât experimentale cât și comerciale ale schimbului de chei pe baza tehnologiei quantum, însă mediul atât de specific care este necesar pentru conservarea stărilor cuantice limitează din punct de vedere geografic implementările existente.

Avantaje aduse criptanalizei

Dezvoltarea sistemelor de calcul cuantice a creat o nouă viziune în ceea ce era considerat posibil folosind numai sisteme clasice. În 1994 matematicianul american Peter Shor a creat un algoritm [2] care poate pune în pericol securitatea celor mai multe sisteme criptografice existente.

Algoritmul poate rezolva problema factorizării numerelor întregi într-o unitate de timp care crește polinomial în funcție de dimensiunea numărului dat. Problema logaritmului discret poate și ea să fie rezolvată prin factorizare. În mod clasic rezolvarea unei astfel de probleme ar avea o durată de timp care ar crește mult mai rapid, aproape exponențial, ca funcție de dimensiunea numărului ce trebuie factorizat. Alegând chei criptografice suficient de mari astfel încât să poată fi considerat imposibil de rezolvat în mod clasic, aceasta problemă matematică reprezintă una din cele care stau la baza celor mai mulți algoritmi asimetrice.

Singurul impediment existent în momentul de față de la un atac care să folosească acest algoritm de factorizare este puterea de procesare actuală a sistemelor cuantice existente. Pentru chei de dimensiuni mari precum cele folosite în practică algoritmul necesită un sistem de calcul cuantic mult mai puternic decât cele mai puternice disponibile la momentul de față. Cu toate acestea, construirea unui sistem suficient de mare pare inevitabilă în viitor.

Din această cauză sunt căutate criptosisteme care se bazează pe alte probleme matematice asupra cărora nu au fost găsite soluții suficient de eficiente pentru a

putea fi implementate în viitor folosind avantajele procesării cuantice. Studiul schemelor criptografice în care este luată în calcul posibilitatea ca atacatorul să poate avea acces la un calculator cuantic se numește criptografie post-cuantică.

Protocoale de schimb de chei

În general fiecare protocol are anumite cerințe, premise sau se bazează pe anumite presupuneri. Spre exemplu folosirea unor primitive criptografice cu anumite proprietăți sau imposibilitatea rezolvării anumitor probleme matematice. Unele au demonstrații formale ale securității, pentru care sunt necesare astfel de presupuneri, în timp ce pentru altele au fost găsite atacuri. În această secțiune sunt prezentate câteva protocoale, cerințele și presupunerile necesare, pași ce trebuie urmați, și în anumite cazuri sunt descrise atacuri împotriva acestora.

Diffie-Hellman (DH)

Primul și posibil cel mai important demers publicat până la acest moment este protocolul de schimbare de chei Diffie-Hellman [3]. Acesta a permis schimbul de chei fără existența unui canal securizat și a deschis calea către criptografia asimetrică.

Parametri necesari înainte de rularea protocolului sunt

- un grup ciclic G de ordin n și
- un element generator g al grupului G .

Baza matematică este problema logaritmului discret. Dacă este cunoscut un element x din grupul G , este considerat dificil din punct de vedere computațional să se afle h astfel încât relația (1) să fie adevărată, unde $\square \cdot \square$ reprezintă operația multiplicativă a grupului G . Cel mai comun exemplu de grup folosit este grupul numerelor întregi, nenule modulo p , unde p este un număr prim foarte mare, dar în afară de acest tip de grup se pot folosi și curbe eliptice.

$$x = \underbrace{g \cdot g \cdot \dots \cdot g}_h \quad (1)$$

Pași

Presupunând că cei doi participanți sunt Alice și Bob, și Alice dorește să realizeze un schimb de chei folosind protocolul Diffie-Hellman aceștia sunt pași pe care cei doi în vor urma

1. Cei doi se decid asupra unui număr prim p , și un generator g al grupului ciclic modulo p , aceștia fiind parametri publici

2. Alice alege un număr secret a , calculează și trimite A conform (2)

$$A = g^a \mod p \quad (2)$$

3. Bob alege un număr secret b , calculează și trimite B conform (3)

$$B = g^b \mod p \quad (3)$$

4. Alice calculează cheia secretă K conform (4)

$$K = B^a \mod p \quad (4)$$

5. Bob calculează cheia secretă K conform (5)

$$K = A^b \mod p \quad (5)$$

6. Cei doi au ajuns în posesia unei chei comune deoarece (6).

$$K = g^{a \cdot b} \mod p \quad (6)$$

Atacuri

Deoarece nu are nicio formă de autentificare, schimbul de chei Diffie-Hellman este vulnerabil la atacurile de tip Man-in-the-Middle.

O parte din calcule necesare rezolvării problemei logaritmului discret pot fi făcute pe baza parametrilor, grupul ciclic și elementul generator, fără a necesita un schimb de chei. În situația în care acești parametri sunt refolosiți în mod foarte des, este posibil ca un atacator să fi precalculat rezultatul acestei prime părți și în momentul în care se desfășoară un schimb de chei acesta va putea termina calculele folosind mesajele transmise într-un timp foarte scurt. [4]

Password-Authenticated Key (PAK)

Prin introducerea unei componente de autentificare bazată pe parole, protocolul PAK încearcă să rezolve problema atacului de tip Man-in-the-Middle plecând de la

schimbul de chei Diffie-Hellman. În [5] sunt oferite atât pași pentru implementarea protocolului cât și o demonstrație formală a securității acestuia.

Pentru rularea protocolului PAK este necesară cunoașterea unei parole de către cele două părți participante și de folosirea a patru funcții aleatoare independente: H_1 , H_{2a} , H_{2b} și H_3 .

Pași

Presupunând

- că cei doi participanți, Alice și Bob, se află în posesia aceleiași parole secrete π ;
- parametri publici sunt numărul prim p , și generatorul g ;
- șirurile A și B se reprezintă identitățile lui Alice și Bob;

Aceștia sunt pași pe care cei doi în vor urma pentru a realiza un schimb de chei folosind protocolul PAK:

1. Alice alege un număr secret x , calculează și trimite m conform (7)

$$m = g^x \cdot H_1(A, B, \pi) \mod p \quad (7)$$

2. Bob verifică valoarea primită pentru m și dacă este 0 se oprește
3. Bob alege un număr secret y , calculează și trimite μ și k conform (8), (9) și (10)

$$\mu = g^y \mod p \quad (8)$$

$$k = H_{2a}(A, B, m, \mu, \sigma, \pi) \quad (9)$$

$$\sigma = \left(\frac{m}{(H_1(A, B, \pi))} \right)^b \mod p \quad (10)$$

4. Alice calculează σ conform (11) și verifică valoarea primită pentru k , dacă este diferită de valoarea obținută prin calcul propriu se oprește

$$\sigma = \mu^x \mod p \quad (11)$$

5. Alice calculează și trimite k' conform (12)

$$k' = H_{2b}(A, B, m, \mu, \sigma, \pi) \quad (12)$$

6. Bob verifică valoarea primită pentru k' , dacă este diferită de valoarea obținută prin calcul propriu se oprește
7. Alice și Bob calculează cheia comună K conform (13)

$$K = H_3(A, B, m, \mu, \sigma, \pi) \quad (13)$$

Simple Password Exponential Key Exchange (SPEKE)

Protocolul SPEKE [6] reprezintă o altă modalitate bine cunoscută de a incorpora autentificare prin parolă în schimbul de chei Diffie-Hellman. Se pot delimita clar două etape, schimbul efectiv și verificarea. Schimbul efectiv este foarte asemănător cu Diffie-Hellman, dar are ca punct de plecare, generatorul grupului, obținut pe baza parolei în loc să fie un parametru public. În cea de a doua fază, cea de verificare, este folosită o funcție de hash asupra cheii obținute.

Următoarele cerințe sunt necesare pentru implementarea protocolului:

- o funcție de hash H
- un număr prim p care să respecte proprietatea (14), unde q este tot un număr prim

$$p = 2q + 1 \quad (14)$$

- o funcție f care să primească la intrare o parolă și să returneze un element dintr-un grup ciclic modulo p de ordin q

Un exemplu de astfel de funcție este (15)

$$f(\pi) = (H(\pi))^2 \bmod p \quad (15)$$

Pași

Presupunând că cei doi participanți sunt Alice și Bob, notațiile sunt cele definite anterior, parola este π , aceștia sunt pași pe care cei doi în vor urma pentru realizarea schimbului de chei folosind protocolul SPEKE:

1. Alice alege un număr secret x , calculează și trimite Q_A conform (16)

$$Q_A = (f(\pi))^x \bmod p \quad (16)$$

2. Bob alege un număr secret y , calculează și trimite Q_B conform (17)

$$Q_B = (f(\pi))^y \bmod p \quad (17)$$

3. Alice calculează cheia secretă K conform (18)

$$K = (H(Q_B))^x \bmod p \quad (18)$$

4. Bob calculează cheia secretă K conform (19)

$$K = (H(Q_A))^y \bmod p \quad (19)$$

5. Alice trimite $H(H(K))$ drept dovadă a posesiei chei
 6. Bob verifică valoarea primită pentru $H(H(K))$, dacă este diferită de valoarea obținută prin calcul propriu se oprește
 7. Bob trimite $H(K)$ drept dovadă a posesiei chei
 8. Alice verifică valoarea primită pentru $H(K)$, dacă este diferită de valoarea obținută prin calcul propriu se oprește

Ring Learning With Errors (RLWE)

Cele mai recente dezvoltări în industria criptografică, sunt cele care iau în calcul posibilitățile tehnologiei quantum. Deoarece problema factorizării numerelor întregi poate fi rezolvată eficient în [7] este propus un alt model matematic. Pentru acest nou model nu există algoritmi cuantici care să rezolve problema matematică mult mai eficient decât cei clasici. O variantă simplă de schimb de chei folosind RLWE, însoțită de demonstrația formală a securității, este găsită în [8]. Protocolul încearcă să se asemene cu PAK păstrând simplitatea, dar schimbând fundamentul matematic.

Corectitudinea protocolului Diffie-Hellman se bazează pe relația (20), iar securitatea se bazează pe dificultatea rezolvării problemei logaritmului discret de către calculatoarele clasice. În timp ce corectitudinea protocoalelor de tipul RLWE se bazează pe asociativitatea înmulțirii polinoamelor într-un inel. Deoarece împărțirea polinoamelor nu reprezintă o problemă dificilă din punct de vedere computațional, este adăugat un element de zgomot, ales conform unei distribuții de probabilitate stabilite dinainte. Acest zgomot garantează securitatea.

$$(g^a)^b = (g^b)^a \mod p \quad (20)$$

Pentru implementarea protocolului RLWE-PAK sunt necesare

- un număr n putere a lui 2
- un număr prim q de forma (21)

$$\begin{aligned} q &= 2^{\omega \cdot \log(n)} + 1 \\ q \mod 2n &= 1 \end{aligned} \quad (21)$$

- o funcție de hash H_1 al cărei rezultat poate fi considerat ca aparținând $\mathbb{Z}_q[x]/(x^n+1)$, mulțimea polinoamelor reduse modulo x^n+1 cu coeficienți întregi modulo q
- două funcții de hash H_2 și H_3 pentru verificare comunicațiilor și o funcție de derivare de chei H_4
- un parametru public a ales aleator din $\mathbb{Z}_q[x]/(x^n+1)$
- o distribuție discretă de tip Gauss χ_b cu parametrul β real, pozitiv și nenul
- funcția $Mod_2 : \mathbb{Z}_q \times \{0, 1\} \rightarrow \{0, 1\}$ definită conform (22)

$$Mod_2(v, b) = \left(v + b \cdot \frac{q-1}{2} \right) \mod q \mod 2 \quad (22)$$

- funcția $Cha : \mathbb{Z}_q \rightarrow \{0, 1\}$ definită astfel: folosind reprezentarea simetrică a $\mathbb{Z}_q = \left\{ -\frac{q-1}{2}, \dots, \frac{q-1}{2} \right\}$, funcția Cha întoarce valoarea 0, dacă intrarea este în mulțimea $\left\{ -\left\lfloor \frac{q}{4} \right\rfloor, \dots, \left\lfloor \frac{q}{4} \right\rfloor \right\}$, și 1 altfel

Pași

Presupunând că cei doi participanți sunt Alice și Bob, notațiile sunt cele definite anterior, șirurile A și B se reprezintă identitățile lui Alice și Bob, parola este

π , aceştia sunt paşi pe care cei doi în vor urma pentru realizarea schimbului de chei folosind protocolul RLWE-PAK

1. Alice alege două polinoame s_A şi e_A aleator, conform distribuţiei χ_β
2. Alice calculează şi trimite A şi m conform (23)

$$m = a \cdot s_A + 2e_A + H_1(\pi) \quad (23)$$

3. Bob verifică valoarea primită pentru m şi dacă nu aparţine $\mathbb{Z}_q[x]/(x^n+1)$ se opreşte
4. Bob alege două polinoame s_B şi e_B aleator, conform distribuţiei χ_β
5. Bob calculează şi trimite μ , w şi k conform (24) - (29)

$$\mu = a \cdot s_B + 2e_B \quad (24)$$

$$\gamma' = -H_1(\pi) \quad (25)$$

$$k_B = (m + \gamma') \cdot s_b \quad (26)$$

$$w = Cha(k_B) \quad (27)$$

$$\sigma = Mod_2(k_B, w) \quad (28)$$

$$k = H_2(A, B, m, \mu, \sigma, \gamma') \quad (29)$$

6. Alice verifică valoarea primită pentru μ şi dacă nu aparţine $\mathbb{Z}_q[x]/(x^n+1)$ se opreşte
7. Alice calculează σ conform (30) şi verifică valoarea primită pentru k , dacă este diferită de valoarea obţinută prin calcul propriu se opreşte

$$\sigma = Mod_2(k_A, w) \quad (30)$$

8. Alice calculează şi trimite k' conform (31)

$$k' = H_3(A, B, m, \mu, \sigma, \gamma') \quad (31)$$

9. Bob verifică valoarea primită pentru k' , dacă este diferită de valoarea obţinută prin calcul propriu se opreşte
10. Alice şi Bob calculează cheia comună K conform (32)

$$K = H_4(A, B, m, \mu, \sigma, \gamma') \quad (32)$$

Quantum Key Derivation (QKD)

Primul protocol de schimb de chei care să folosească tehnologia quantum propus de Bennett și Brassard în 1984 a fost numit BB84 [9]. Cel mai mare avantaj al acestui protocol este posibilitatea de detecție a atacurilor pasive. Pentru acesta este într-adevăr necesar un canal de comunicație special, denumit în continuare canal cuantic, pe lângă existența canalului clasic pe care sunt trimise mesajele obișnuite, dar și cele necesare pentru negocierea chei. Protocolul poate detecta încercări de atacuri, atât pasive cât și active, de pe canalul cuantic, dar cel clasic trebuie să fie autentificat.

Pași

Presupunând că

- cei doi participanți sunt Alice și Bob,
- cei doi dispun de un canal cuantic pe care pot fi trimiși fotoni în mod secvențial, și de un canal clasic autentificat
- polarizarea fotonilor poate fi măsurată atât în baza rectilinie, 0° sau 90° , cât și în baza diagonală, 45° sau 135° , atât Alice, cât și de Bob
- dacă valoarea măsurată polarizării este 0° sau 45° , aceasta este codificată ca un 0 logic, dacă valoarea polarizării măsurată este 90° sau 135° aceasta este codificată cu 1 logic
- i este un contor

aceștia sunt pași pe care cei doi în vor urma pentru realizarea schimbului de chei folosind protocolul BB84:

1. Alice alege un bit b_i în mod aleator
2. Alice alege o baze B_i în mod aleator
3. Bob alege o baze B'_i în mod aleator

4. Alice codifică bitul b_i în baza B_i și trimite fotonul rezultat pe canalul cuantic
5. Bob decodifică bitul b'_i prin măsurarea în baza B'_i a fotonului receptat pe canalul cuantic
6. Alice și Bob salvează valorile obținute, incrementează contorul i și dacă acesta nu a ajuns la o lungime suficientă repetă întreg procesul de la pasul 1
7. Alice trimite secvența de baze aleatorii B_i pe canalul clasic
8. Bob trimite secvența de baze aleatorii B'_i pe canalul clasic
9. Bob verifică baza B_i și dacă este diferită de baza B'_i ignoră bitul b'_i , pentru fiecare i
10. Alice verifică baza B'_i și dacă este diferită de baza B_i ignoră bitul b_i , pentru fiecare i
11. Alice alege o submulțime aleatoare S din mulțimea perechilor de indecși și biți care nu au fost ignorați conform (33)

$$S \subset \{(i, b_i) \mid B_i = B'_i\} \quad (33)$$

12. Alice trimite submulțimea S pe canalul clasic
13. Bob verifică submulțimea S și dacă nu respectă (34), înseamnă că a avut loc o interceptare

$$S \subset \{(i, b'_i) \mid B_i = B'_i\} \quad (34)$$

Folosind metoda aceasta, cu o probabilitate de 0.5, cei doi nu vor alege aceeași bază și nu vor obține un bit util. În medie jumătate din biți transmiși ar trebui să fie vor fi cunoscuți numai de cei doi participanți până la ultimii pași.

Atacuri

În cazul în care un atacator încearcă să obțină informații despre polarizarea fotonilor trimiși pe canalul cuantic acesta va trebui să execute măsurători într-o bază sau în cealaltă.

Jumătate din fotoni sunt ignorați deoarece alegerea bazei a fost distinctă între părțile oneste. Din jumătatea rămasă un sfert sunt mășurați de ambii participanți în

baza rectilinie și celălalt sfert în baza diagonală. Atacatorul nu poate obține nicio informație asupra bazei în care au fost făcute măsurătorile niciunuia dintre participanți. Putem presupune că în medie în jumătate din cazuri va folosi aceeași bază, iar în cealaltă jumătate baza greșită.

Dacă observația atacatorului este în aceeași bază, acesta poate obține un bit corect. Dacă măsurătoarea atacatorului este, însă, în bază opusă față de măsurătoarea participanților onești, informația originală se va pierde. La destinație informația receptată va fi aleatoare, fără să fie corelată nici cu observația atacatorului, nici cu cea a emițătorului.

Ultimi pași ai protocolului trebuie să asigure un eșantion aleator suficient de mare astfel încât, dacă atacatorul ar fi interceptat și încercat să recreeze fotonii, o parte din aceștia ar ajunge modificați la destinație, iar din aceea parte sunt regăsite inclusiv în eșantionul făcut public în ultimii pași ai protocolului.

Protocolul Dragonfly

Protocolul Dragonfly este un protocol de schimb de chei criptografice care folosește autentificare pe bază de parolă care a fost standardizat în [10]. Este un protocol simetric ce poate fi rulat între doi participanți care execută aceiași pași și cunosc aceeași parolă.

Premise

Protocolul se bazează pe problema logaritmului discret. În standard este specificat că se pot folosi atât câmpuri finite cât și curbe eliptice. Pentru o descriere cât mai generală se vor presupune cunoscute următoarele notații legate de grupuri ciclice:

- Un grup ciclic este fie un câmp finit, fie o curbă eliptică;
- Un câmp finit este definit de
 - un număr prim mare, notat p ,
 - un element ce generează un subgrup suficient de mare și
 - ordinul q al subgrupului generat, unde q trebuie să fie prim și să respecte (35);

$$q \mid \frac{p-1}{2} \quad (35)$$

- O curbă eliptică este definită de
 - un număr prim mare, notat p ,
 - două numere a și b care să definească dacă un punct de coordonate x și y aparține curbei conform (36),
 - un element ce generează un subgrup suficient de mare și
 - ordinul q al subgrupului generat, unde q trebuie să fie prim;

$$y^2 - x^3 - a \cdot x - b = 0 \mod p \quad (36)$$

- Elementele grupului sunt notate cu majusculă;
- $element-op(X, Y)$ reprezintă aplicarea operației grupului asupra elementelor X și Y . Într-un câmp finit acesta este $X \cdot Y \mod p$, iar pe o curbă eliptică $X + Y$;
- $scalar-op(x, Y)$ reprezintă aplicarea operației grupului asupra elementului Y în mod repetat de x ori. Într-un câmp finit acesta este $Y^x \mod p$, iar pe o curbă eliptică $x \cdot Y$;
- $inverse(X)$ este definită astfel încât $element-op(X, inverse(X))$ să rezulte în elementul neutru al grupului respectiv.

Conform standardului se fac următoarele presupuneri:

- Participanții au acces la funcție aleatoare sigură H , care să satisfacă proprietățile modelului unui oracol aleator [11]. Aceasta are ca intrare o șiruri de biți de dimensiuni variabile și ca ieșire un șir aleator de dimensiune fixă.
- Funcția $F(X)$ are ca intrare elementul X al grupului ales și ca ieșire un număr întreg. În cazul unui câmp finit F este funcția identitate, deoarece elementele câmpului sunt reprezentate prin numere întregi. În cazul unei curbe eliptice, ieșirea funcției va fi prima coordonată a punctului de pe curbă ce reprezintă elementul dat ca intrare în F .
- Funcția $KDF(n, k, label)$ este o funcție de derivare de chei, care respectă standardul NIST [12]. Primul parametru reprezintă dimensiunea dorită a

cheii rezultate, al doilea este cheia de intrare, din care se va face derivarea, iar cel de al treilea este o etichetă care este legată de cheie.

- Parola folosită poate fi văzută ca făcând parte dintr-o mulțime finită de parole posibile, cum ar fi cuvinte luate din dicționar. Atacatorul are acces la mulțimea din care a fost aleasă parola, dar nu știe decât că toate posibilitățile au probabilitate egală de a fi alese.
- Problema logaritmului discret, fie în câmpuri finite, fie pe curbe eliptice, este dificil de rezolvat din punct de vedere computațional.
- Participanții trebuie să poată genera numere aleatoare, calitative din punct de vedere criptografic.

Alte notații folosite

- A și B reprezintă două șiruri ce conțin identitățile celor doi participanți;
- π reprezintă parola, un cuvânt, frază sau cod cu entropie scăzută, secret cunoscut doar de participanți onești;
- $lsb(a)$ este o funcție care returnează cel mai puțin semnificativ bit din șirului a primit la intrare;
- $len(a)$ este o funcție care întoarce dimensiunea în biți a șirului a primit la intrare;
- $lgr(a, b)$ este o funcție care are la intrare două numere, a și b , al doilea fiind prim, și întoarce valoarea simbolului Legendre $\left(\frac{a}{b}\right)$ conform (37);

$$lgr(a, b) = \left(\frac{a}{b}\right) = a^{\frac{b-1}{2}} \mod p \quad (37)$$

$$\left(\frac{a}{b}\right) \in \{-1, 0, 1\}$$

- $min(a, b)$ este o funcție care are la intrare două șiruri, a și b , și întoarce valoarea minimă din punct de vedere lexicografic dintre cele două, sau zero (0) dacă cele două sunt egale;

- $\max(a, b)$ este o funcție care are la intrare două șiruri, a și b , și întoarce valoarea maximă din punct de vedere lexicografic dintre cele două, sau zero (0) dacă cele două sunt egale;

Pași

Protocolul este unul simetric în sensul că nu există roluri specifice. O descriere se poate face după următorii patru pași. Ambii participanți

- se folosesc de parolă pentru a obține același element secret aparținând grupului ales;
- se folosesc de elementul secret pentru a genera și trimite un mesaj de angajare, „commit message” în limba engleză;
- se folosesc de mesajul de angajare receptat pentru a genera și trimite un mesaj de confirmare, „confirm message” în limba engleză;
- verifică mesajul de confirmare receptat pentru a-și autentifica partenerul.

Alegerea unui element pe baza parolei

Pentru a obține un element al grupului ales folosind parola standardul nu impune o anumite modalitate, dar este propusă una numită în engleză „Hunting and Pecking”. În modalitatea descrisă este creată o bază comună după care în funcție de tipul grupul folosit, câmp finit sau curbă eliptică, se aplică operații diferite.

Se alege un contor *counter* pe opt (8) biți, inițializat cu valoarea unu (1). Este calculat un șir *base* folosind identitățile celor doi participanți, parola și contorul conform (38). Funcția *KDF* este aplicată șirului *base*, pentru a se obține un nou șir, *seed*, de dimensiune $\text{len}(p)+64$. Se verifică, în funcție de tipul grupului folosit, dacă noul șir, *seed*, poate duce la un element al grupului ales. Dacă nu, se reia procesul cu incrementarea contorului. Dacă da, elementul obținut este numit element bazat pe parolă, *PE*.

$$\text{base} = H(\max(A, B), \min(A, B), \pi, \text{counter}) \quad (38)$$

Este posibil să fie mai multe sau mai puține iterații până se găsește un șir care să ducă la un element ce aparține grupului ales. Pentru a evita atacurile ce obțin informații prin măsurarea timpului de execuție, trebuie să existe un număr minim

de iterații, k , suficient de mare astfel încât numărul iterațiilor executate să difere foarte puțin între rulări diferite.

Mesajul de angajare

După obținerea unui element al grupului derivat din parola secretă, are loc primul schimb de mesaje. Fiecare participant generează în mod aleator două numere întregi *private* și *mask* conform (39) și (40). După aceea generează trimite un număr s și un element E conform (41) și (42).

$$1 < private < q \quad (39)$$

$$1 < mask < q \quad (40)$$

$$s = (private + mask) \bmod q \quad (41)$$

$$E = inverse(scalar - op(mask, PE)) \quad (42)$$

Mesajul primit de la partener este parcurs și împărțit într-un număr s_{peer} și un element E_{peer} . Acestea sunt verificate, s_{peer} trebuie să fie mai mare strict decât unu (1) și mai mic strict decât q , iar E_{peer} trebuie să fie un element valid în grupul ales. Pe lângă aceste verificări s_{peer} trebuie să fie diferit de s , iar E_{peer} să fie diferit de E .

Având mesajul de angajare al partenerului, se poate calcula un secret comun ss conform (43) și pe baza lui cheia key conform (44). Pentru o securitate mai bună se împarte cheia key în două părți egale, kck și mk . kck este o cheie folosită în mesajul de confirmare, iar mk este cheia finală.

$$ss = F(scalar - op(private, element - op(E_{peer}, scalar - op(s_{peer}, PE))) \quad (43)$$

$$key = KDF(2 \cdot len(p), ss, 'Dragonfly Key Derivation') \quad (44)$$

Folosind notațiile comune pentru $scalar - op$ și $element - op$ în contextul câmpurilor finite modulo p relația (43) se poate scrie ca (45), iar folosind notații comune criptografiei pe curbe eliptice aceasta devine (46)

$$ss = F((E_{peer} \cdot PE^{s_{peer}})^{private}) \bmod p \quad (45)$$

$$ss = F(private \cdot (E_{peer} + s_{peer} \cdot PE)) \quad (46)$$

Mesajul de confirmare

În mesajul de confirmare fiecare participant se folosește de valoarea obținută pentru cheia de confirmare comună, kck , informațiile trimise în cele două mesaje de angajare și trebuie să folosească șirul ce conține acestuia. Șirul care conține identitatea participantului ce execută operațiile, fie acesta A sau B , este notat aici ca id_{sender} , iar celălalt este notat cu id_{peer} . Mesajul de confirmare este calculat conform (47).

$$confirm = H(kck, s, s_{peer}, E, E_{peer}, id_{sender}) \quad (47)$$

Verificarea mesajului de confirmare

Singura informație privată care intră în alcătuirea mesajului de confirmare este kck . Deoarece aceasta este obținută dintr-un secret comun, ambii participanți onești, și nimeni altcineva, au acces la kck . Astfel pentru verificare trebuie comparat mesajul de confirmare primit cu cel obținut prin calcul conform (48).

$$confirm_{peer} = H(kck, s_{peer}, s, E_{peer}, E, id_{peer}) \quad (48)$$

Atacuri

După ce prima versiune a fost propusă pentru standardizare asupra acesteia au fost descoperite modalități de criptanaliză în [13]. Inițial, protocolul nu impunea verificări asupra mesajului de angajare. Aceasta a dus la două tipuri de atacuri, atacuri folosind subgrupuri mici și atacuri prin reflexie. Atacul folosind un grup mic se bazează pe alegerea unui element care nu aparține subgrupului ales. Elementul ales și trimis în mesajul de angajare are un ordin mult mai mic decât q . Operațiile aplicate asupra elementului vor avea rezultate în subgrupul mic ales de atacator. Deoarece numărul de posibile chei generate de participantul onest este redus considerabil fără ca acesta să observe atacatorul poate încerca fiecare combinație de parolă cu un element din subgrupul mic ales. Este garantat să ajungă la secretul comun ss , și folosind mesajul de confirmare primit să decidă care este cel bun. Astfel atacatorul nu mai poate fi distins de un participant onest, deoarece se află în posesia secretului ss .

Atacurile prin reflexie sunt atacuri în care atacatorul așteaptă mesajul de angajare al participantului onest și trimite aceleași valori. În această situație mesajul de confirmare al atacatorului va arăta identic cu cel al părții oneste și poate fi trimis, în

mod similar, același mesaj. Atacatorul va fi considerat autentificat chiar dacă nu cunoaște nici parola și nici cheia ce este obținută la final.

O altă modificare făcută față de versiunea originală se referă la adăugarea șirului ce se referă la identitatea emițătorului în mesajul de confirmare. Aceasta previne atacurile prin reflexie cât și atacuri asemănătoare cu unul de tip Man-in-the-Middle, dar în care atacatorul doar interceptează și retrimite mesajele exact așa cum le primește fără să înțeleagă sensul lor deoarece nu are parola.

O variantă apropiată a protocolului a fost creată pentru a demonstra în mod formal securitatea acestuia în [14]. Diferențele observate sunt trimiterea identității încă din mesajul de angajare și împărțirea cheii în trei, câte una folosită de fiecare din cei doi participanți în mesajul de confirmare și cea finală, precum și utilizarea ambelor identități ale participanților în construcția mesajelor de confirmare.

Scenarii de utilizare

Unul din scenariile în care este nevoie de autentificare mutuală pe bază de parolă este conectarea a două dispozitive prin Wi-Fi. Wi-Fi Alliance a publicat Wi-Fi Protected Access 3 sau WPA3 în iunie 2018. Unu din cele mai importante aspecte ale WPA3 este că acesta impune implementarea protocolului Dragonfly pentru autentificarea simultană a punctului de acces față de client și a clientului față de punctul de acces. Predecesorul, WPA2, folosește pentru autentificare un protocol vulnerabil la atacuri de dicționar și care nu prezintă proprietatea de „forward secrecy”.

În ciuda demonstrației formale a securității protocolului Dragonfly din [15], integrarea acestuia într-un sistem precum WPA3 nu a fost lipsită de probleme. În [16] sunt descrise mai multe atacuri utilizabile în practică. Protocolu Dragonfly oferă autentificare, într-un mod rezistent la atacuri de tip dicționar off-line. Cu toate acestea, un atacator poate pretinde că este punctul de acces și să convingă dispozitivul client să înceapă un schimb procesul de autentificare folosind WPA2. Chiar dacă atacatorul care nu deține parola nu poate termina cu succes protocolul folosit de WPA2, datele trimise până în momentul când clientul detectează atacul sunt suficiente pentru atacul de tip dicționar.

Soluții similare, implementări existente

În termeni generali există foarte multe implementări de protocoale de schimb de chei. Cele mai utilizate sunt, cel mai probabil, cele care sunt integrate în implementările protocoalelor SSL („Secure Sockets Layer”) și TLS („Transport Layer Security”).

În contextul mai restrâns al protocolului Dragonfly, foarte puține implementări au fost făcute open-source. Mai exact o singură implementare a fost găsită la momentul scrierii lucrării. Aceasta poate fi accesată la linkul de Github <https://github.com/NikolaiT/Dragonfly-SAE> [17]. Implementarea este făcută în limbajul Python și folosește criptografia bazată pe curbe eliptice.

Din punctul de vedere al standardului sunt urmărite toate normele impuse, dar nu și toate cele recomandate. Secțiunea 3.2.1 din standardul protocolului Dragonfly [10], urmărește derivarea unui element secret pe baza parolei în modul de lucru cu curbe eliptice. În criptografia cu curbe eliptice sunt folosite reziduri pătratice și non-reziduri pătratice. Este menționat că verificarea dacă o valoare este rezidu sau non-rezidu pătratic poate costa unități diferite de timp și astfel pot fi dezvăluite informații despre respectiva valoare. Din această cauză standardul recomandă mascarea valorii înainte de verificarea acesteia. Masca care este aplicată trebuie să fie aleasă ca un rezidu pătratic sau un non-rezidu pătratic în mod aleator și înmulțită cu valoarea ce trebuie verificată. Astfel fiind introdus un efect aleator cu probabilități egale, informația dezvăluită devine nulă.

Din punctul de vedere al performanțelor, implementarea a fost testată pentru măsurarea timpului de execuție pe un procesor i7-8750 rulând Microsoft Windows 10 și folosind Python 3.7, iar timpul mediu rezultat a fost de 44.5 milisecunde.

Detalii privind implementarea practică

Acest capitol detaliază modul în care a fost implementat protocolul în practică. În următoarele secțiuni sunt prezentate cerințele software, cazurile de utilizare, arhitectura și modalitățile de testare ale unei biblioteci informatice care implementează protocolul Dragonfly pentru schimbul de chei criptografice.

Definirea cerințelor complete pentru sistemul propus

Se dorește implementarea unei biblioteci software care să expună programelor ce o accesează metodele necesare generării unei chei criptografice de tip simetric și a comunicării în mod criptat folosind respectiva cheie. Această cheie este generată pe baza unei parole conform protocolului Dragonfly. Sunt expuse primitivele necesare generării mesajelor conform protocolului, cât funcționalitatea de criptare, trimitere, primire și decriptare de date.

Biblioteca trebuie să permită celui ce o utilizează să execute fiecare din pașii pe care un participant la protocolul Dragonfly îi poate executa.

- Trebuie să permită selectarea setului de parametri ce va fi folosit.
 - Trebuie să ofere mai multe posibilități de seturi de parametri.
 - Trebuie să limiteze alegerea setului de parametri la unul predefinit.
 - Trebuie să conțină atât seturi de parametri predefinite în care se folosesc curbe eliptice cât și seturi de parametri predefinite în care se folosesc câmpuri finite.
- Trebuie să permită inițializarea și eliberarea memoriei.
 - Trebuie să permită inițializarea la cerere.
 - Trebuie să permită eliberarea memoriei la cerere.
 - Trebuie să permită eliberarea memoriei în mod automat în momentul în care există certitudinea că nicio altă metodă nu va mai fi apelată.
 - Trebuie să șteargă datele secrete din memorie înainte de eliberarea memoriei.

- Trebuie să permită generarea mesajelor de angajare și mesajelor de confirmare.
 - Trebuie ca procesul prin care sunt generate mesajele să respecte standardul.
 - Trebuie să semnaleze dacă se încearcă generarea unor mesaje în afara ordini descrise conform standardului.
 - Trebuie să semnaleze dacă există erori în timpul generării mesajelor de angajare.
 - Trebuie să semnaleze dacă există erori în timpul generării mesajelor de confirmare.
- Trebuie să permită verificarea mesajelor de angajare și mesajelor de confirmare generate de un alt participant.
 - Trebuie să semnaleze dacă mesajul primit nu este valabil.
- Trebuie să permită generarea cheilor.
 - Trebuie să stocheze cheile în mod sigur.
- Trebuie să permită crearea unei conexiuni securizate cu o altă mașină într-o rețea.
 - Trebuie să permită comunicarea în rețea.
 - Trebuie să permită rularea tuturor pașilor cu o altă mașină din rețea.
 - Trebuie să folosească cheia obținută pentru criptarea traficului transmis.
 - Trebuie să folosească cheia obținută pentru decriptarea traficului primit.

Prezentarea cazurilor de utilizare

Cazurile de utilizare sunt foarte numeroase, existând o varietate foarte mare de aplicații transmit și primesc date și care pot beneficia de avantajul securității prin criptarea datelor. De asemenea există aplicații care folosesc scheme de criptare și decriptarea, dar ar putea folosi implementarea protocolului Dragonfly pentru o

autentificare eficientă sau pentru un schimb de chei cu proprietatea de forward secrecy și rezistență la atacuri de tip dicționar. Un programator poate folosi această bibliotecă pentru a executa la nivelul aplicației dezvoltate de acesta oricare din aceste operații, Autentificare unui partener, trimiterea și primirea de date criptate, cu posibilitatea de criptare și decriptare bazată pe o cheie generată. Figura 1: UML usecase reprezintă o diagramă UML pentru cazurile de utilizare a bibliotecii implementate. Actorii, sau utilizatorii bibliotecii, sunt programatorii care își dezvoltă propriile aplicații apelând metodele acestei biblioteci.

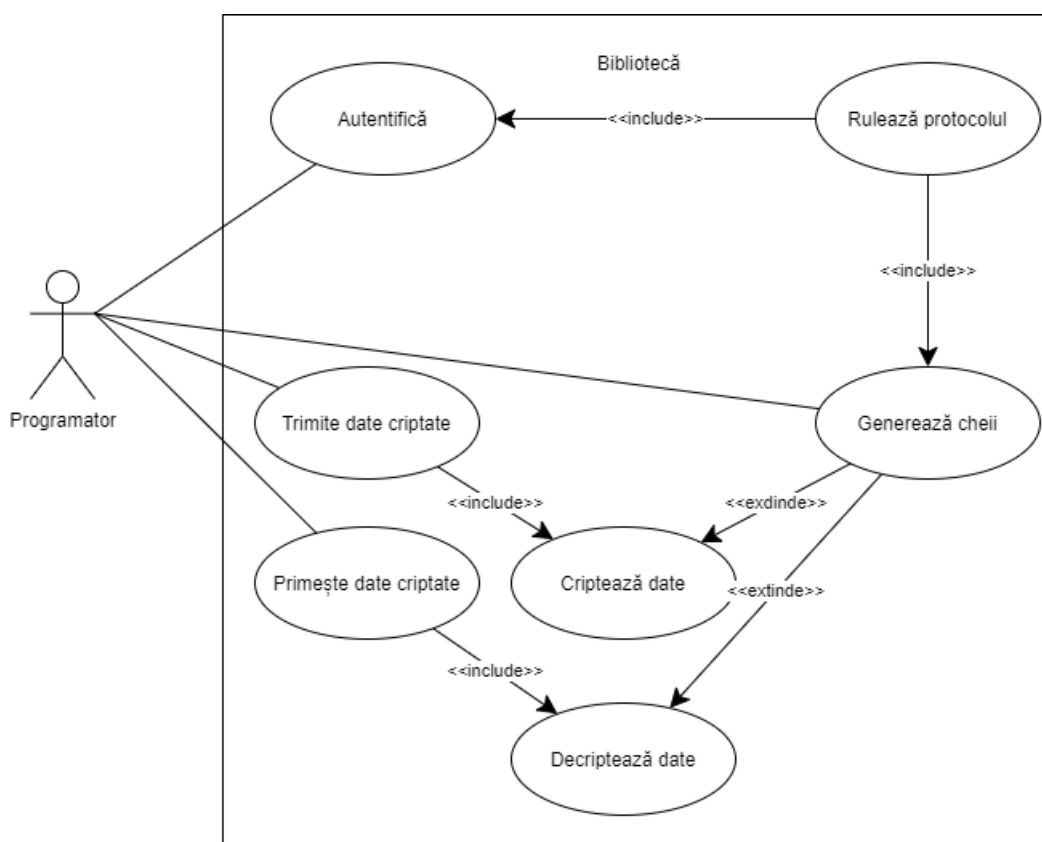


Figura 1: UML usecase

Definirea arhitecturii și proiectarea sistemului

Biblioteca este scrisă în limbajul C++ folosind paradigma de programare orientată pe obiecte. Clasa SecureSocket modelează un socket de rețea și oferă funcționalitățile de autentificare pe bază de parolă și criptare, trimitere, recepție și decriptare a datelor la nivel rețea. Clasa Peer modelează un participant la schimbul de chei. La nivelul acestei clase sunt implementate metodele executate de un participant la fiecare pas. Prin clasa Element sunt implementate operațiile matematice de baza de lucru pe curbe eliptice sau pe câmpuri finite. Diferențierea între cele două moduri de lucru matematic se face printr-un context static care definește setul de parametri de domeniu ce vor fi folosiți. Acest context este reprezentat de o instanță statică a structurii ParameterSet.

Clasa SecureSocket

Aceasta conține un descriptor socket de rețea și folosește un obiect de tip Peer pentru a putea rula protocolul și a obține o cheie criptografică în momentul conectării cu alt proces din rețea care a creat și folosește un obiect SecureSocket. Această clasă poate fi văzută ca un decorator al obiectului de tip socket deoarece folosește aceeași interfață, dar adaugă funcționalitatea de criptare și decriptare în mod simetric a datelor trimise folosind un cifru oferit de biblioteca openssl și o cheie oferită de implementarea protocolului Dragonfly de la nivelul clasei Peer.

Metodele clasei SecureSocket sunt bazate pe funcțiile de lucru cu socket-uri din bibliotecile <winsock2.h> de pe platformele Microsoft Windows și <sys/socket.h> de pe sistemele compatibile standardului POSIX. Diferența este dată de metodele connect și accept care în plus față de funcțiile pe care sunt bazate au ca parametru de intrare un șir de caractere care reprezintă parola pentru protocolul Dragonfly.

Clasa Peer

Instanțierea clasei Peer se va face pe baza unui șir de caractere care va reprezenta identitatea aceluși participant. Nu are sens ca un participant să nu aibă o identitate și de asemenea, nu are sens să își schimbe identitatea fără a fi distrus și a fi creat un alt participant. Pentru a rula protocolul este nevoie de 2 participanți și o parolă comună. Un obiect de tip Peer poate rula protocolul de mai multe ori, iar la fiecare rulare este necesar să aibă setate parola și șirul care reprezintă identitatea celui alt participant, deoarece aceste se pot schimba de la o rulare a protocolului la alta.

Aceasta se realizează printr-o metodă de inițializare a protocolului care ia ca parametri două șiruri, parola și identitatea partenerului. Mesajul de angajare și mesajul de confirmare sunt generate intern la apelarea metodelor `commitExchange` și `confirmExchange`. După generarea internă conținutul fiecărui tip de mesaj poate fi obținut folosind metodele `getCommitMessage` și `getConfirmMessage`. Mesajele primite de la partener sunt înregistrate și verificate cu ajutorul metodelor `verifyPeerCommit` și `verifyPeerConfirm`. Generarea internă a chei se face în același timp cu mesajul de confirmare, iar expunerea chei se face prin metoda `getKey`.

Clasa Element

La nivelul acestei clase sunt implementate operațiile matematice necesare atât pentru lucru pe curbe eliptice cât și pentru lucru pe câmpuri finite. În standard sunt descrise trei operații, operația între elemente, operația între un element și un scalar, și operația care determină inversul unui element. Clasa `Element` implementează aceste trei operații și în plus implementează metodele necesare pentru conversia obiectului de tip `Element` în șir de caractere și conversia inversă. Aceste funcționalități sunt utile la trimiterea obiectelor de tip `Element`.

Structura ParameterSet

Această structură conține toți parametrii care pot influența rularea protocolului, precum modul de lucru și valorile care definesc grupul în care se lucrează. Pentru criptografia pe curbe eliptice acestea sunt patru numere p , q , a și b , iar pentru criptografia bazată pe câmpuri finite, numai două: p și q . Această structură conține de asemenea un membru static constant de tip vector de `ParameterSet`. Acest vector este inițializat cu valori predefinite și orice rulare a protocolului trebuie să aleagă din acest vector setul de parametri care va fi folosit prin intermediul unui indice. Indicele este și el un membru static, dar nu și constant, al structurii `ParameterSet`.

Tehnologii/API –uri folosite pentru implementarea sistemului

În implementarea claselor care efectuează calculele matematice a fost folosită biblioteca de C++ pentru teoria numerelor, numită NTL („Number Theory Library”). Pentru primitive criptografice precum criptarea și decriptarea cu algoritmi simetrici sau funcții de hash a fost folosit proiectul openssl.

NTL

Din cadrul bibliotecii pentru teoria numerelor au fost folosite clasele `ZZ` pentru numere întregi de dimensiune arbitrară și `ZZ_p` pentru clase de resturi modulo p . NTL oferă limbajului C++ posibilitatea de a executa operațiile matematice de bază precum adunări, înmulțiri, sau ridicări la exponent într-o manieră eficientă atât pentru numere întregi, dar și în special pentru clase de resturi modulo p . Un obiect de tip `Element` conține în compoziția sa obiecte de tip `ZZ_p`, iar toate operațiile între instanțe ale clasei `Element` sunt bazate pe o serie de operații matematice cu întregi sau cu clase de resturi.

OpenSSL

Este un proiect open-source care oferă implementări software în limbajul C pentru protocoalele SSL („Secure Sockets Layer”) și TLS („Transport Layer Security”), dar și pentru alte diverse operații criptografice. Funcțiile din OpenSSL folosite în cadrul proiectului pot fi grupate în două categorii: funcții pentru criptarea și decriptare, și funcții pentru calcularea unui hash.

Criptarea și decriptarea sunt folosite în cadrul clasei `SecureSocket` la apelul metodelor `send` și `receive` utilizând interfața `EVP` din OpenSSL. Funcția de hash este calculată la nivelul clasei `Peer` pentru implementarea oracolului aleator menționat în standardul protocolului Dragonfly.

Descriere cazuri de testare și elaborare raport de testare

Pentru testarea funcționalității este verificat local întreg procesul. Sunt generate două obiecte de tip Peer, folosind ca șir pentru identitate „Alice” pentru primul și „Bob” pentru cel de al doilea. Ambele instanțe apelează metoda de inițiere a protocolului folosind o parolă comună și identitatea partenerului. Sunt executați toți pași în ordinea corectă, iar rezultatul așteptat este ca la final ambele instanțe să obțină aceiași cheie și niciun mesaj de eroare să nu fie semnalat. Codul folosit pentru acest scenariu este prezentat în Figura 2: Cod scenariu 1, iar rezultatul obținut este cel așteptat, ieșirea din program fără semnalarea unei erori.

```

20 void scenariu1()
21 {
22     Peer::selectParameterSet(0);
23     Peer alice("Alice"), bob("Bob");
24     alice.initiate("Bob", "1234");
25     bob.initiate("Alice", "1234");
26
27     alice.commitExchange();
28     bob.commitExchange();
29     alice.getCommitMessage(buffer1, 256);
30     bob.getCommitMessage(buffer2, 256);
31
32     if (!alice.verifyPeerCommit(buffer2) || !bob.verifyPeerCommit(buffer1)) {
33         std::cerr << "Commit error" << std::endl;
34         return;
35     }
36     alice.confirmExchange();
37     bob.confirmExchange();
38     alice.getConfirmMessage(buffer1, 256);
39     bob.getConfirmMessage(buffer2, 256);
40
41     if (!alice.verifyPeerConfirm(buffer2) || !bob.verifyPeerConfirm(buffer1)) {
42         std::cerr << "Confirm error" << std::endl;
43         return;
44     }
45     alice.getKey(buffer1, 256);
46     bob.getKey(buffer2, 256);
47     if (memcmp(buffer1, buffer2, alice.getKeySize()) != 0) {
48         std::cerr << "Keys differ" << std::endl;
49         return;
50     }
51     alice.destroy();
52     bob.destroy();
53 }

```

Figura 2: Cod scenariu 1

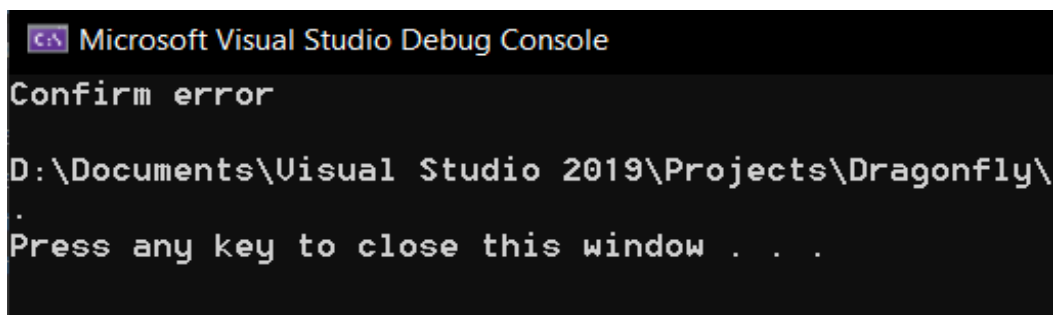
Pentru a verifica corectitudinea este refăcut testul anterior cu diferența că în acest scenariu sunt folosite parole diferite. Acest test trebuie să rezulte în semnalarea erori de autentificare. Codul este prezentat în Figura 3: Cod scenariu 2, iar rezultatul în Figura 4: Rezultat scenariu 2

```

60 void scenariu2()
61 {
62     Peer::selectParameterSet(0);
63     Peer alice("Alice"), bob("Bob");
64     alice.initiate("Bob", "1234");
65     bob.initiate("Alice", "0000");
66
67     alice.commitExchange();
68     bob.commitExchange();
69     alice.getCommitMessage(buffer1, 256);
70     bob.getCommitMessage(buffer2, 256);
71
72     if (!alice.verifyPeerCommit(buffer2) || !bob.verifyPeerCommit(buffer1)) {
73         std::cerr << "Commit error" << std::endl;
74         return;
75     }
76     alice.confirmExchange();
77     bob.confirmExchange();
78     alice.getConfirmMessage(buffer1, 256);
79     bob.getConfirmMessage(buffer2, 256);
80
81     if (!alice.verifyPeerConfirm(buffer2) || !bob.verifyPeerConfirm(buffer1)) {
82         std::cerr << "Confirm error" << std::endl;
83         return;
84     }
85     alice.getKey(buffer1, 256);
86     bob.getKey(buffer2, 256);
87     if (memcmp(buffer1, buffer2, alice.getKeySize()) != 0) {
88         std::cerr << "Keys differ" << std::endl;
89         return;
90     }
91     alice.destroy();
92     bob.destroy();
93 }

```

Figura 3: Cod scenariu 2



```

Microsoft Visual Studio Debug Console
Confirm error
D:\Documents\Visual Studio 2019\Projects\Dragonfly\
Press any key to close this window . . .

```

Figura 4: Rezultat scenariu 2

Pentru a testa performanța, primul scenariu va fi executat într-o buclă de 20 de ori și va fi măsurat timpul fiecărui ciclu. Figura 5: Cod scenariu 3 și Figura 6: Rezultate scenariu 3 ilustrează acest test. Timpul mediu după 20 de rulări este 43.1 milisecunde. Testul a fost rulat pe un procesor i7-8750H, rulând pe platforma Microsoft Windows 10. Pentru comparație în aceleași condiții implementarea open-source descrisă într-un capitol anterior a obținut un timp mediu de 44.5 ms.

```
98 void scenariu3()
99 {
100     for (int i = 0; i < 20; i++) {
101         clock_t start = clock(), diff;
102         scenariu1();
103         diff = clock() - start;
104
105         int msec = diff * 1000 / CLOCKS_PER_SEC;
106         cout << "Time taken: " << msec << " milliseconds\n";
107     }
108 }
```

Figura 5: Cod scenariu 3

```
Time taken: 46 milliseconds
Time taken: 43 milliseconds
Time taken: 43 milliseconds
Time taken: 44 milliseconds
Time taken: 44 milliseconds
Time taken: 43 milliseconds
Time taken: 43 milliseconds
Time taken: 43 milliseconds
Time taken: 43 milliseconds
Time taken: 43 milliseconds
Time taken: 43 milliseconds
Time taken: 43 milliseconds
Time taken: 43 milliseconds
Time taken: 43 milliseconds
Time taken: 43 milliseconds
Time taken: 43 milliseconds
Time taken: 43 milliseconds
Time taken: 42 milliseconds
Time taken: 42 milliseconds
Time taken: 42 milliseconds
Time taken: 43 milliseconds
Time taken: 43 milliseconds
```

Figura 6: Rezultate scenariu 3

Concluzii

Sinteza principalelor idei din lucrare

- Schemele de schimb de chei bazate pe autentificare cu parolă, PAKE, reprezintă o alternativă nu numai convenabilă, dar și sigură pentru criptografia bazată pe certificate digitale.
- Un atacator trebuie să ruleze protocolul cu un participant care cunoaște parola corectă pentru a verifica dacă o încercare de parolă este bună sau nu, ceea ce face atacurile de tip dicționar ușor de detectat și la fel de ușor de prevenit.
- Protocolul Dragonfly este un protocol PAKE care deși este standardizat recent, beneficiază de o demonstrație formală a securității.
- În acest proiect este prezentată o implementarea proprie a protocolului Dragonfly.
- Comparativ cu singura implementare open-source găsită, implementarea proprie este mai flexibilă permițând selecția modului de lucru atât pentru folosirea curbelor eliptice cât și pentru folosirea câmpurilor finite.

Direcții și idei pentru continuarea lucrării

- Studiul, verificarea și eventual extinderea seturilor de parametri pe care lucrează aplicația constituie o posibilă direcție de îmbunătățire. Institutul Național de Standarde și Tehnologie din Statele Unite oferă o listă de curbe eliptice considerate sigure, iar acestea pot fi implementate pentru mai multă flexibilitate.
- Utilitatea practică a acestei lucrări poate fi de asemenea sporită prin integrarea implementării practice într-un pachet deja existent și utilizat pe scară largă precum OpenSSL. Încadrarea protocolului într-o suită deja existentă și folosită ar facilita utilizarea acestuia fără efort în plus din partea utilizatorului final.

- O altă idee pe care se poate continua dezvoltarea acestui proiect este lucrul cu mai multe fire de execuție. Implementarea curentă nu permite folosirea mai multor fire de execuție decât cu limitarea la un singur set de parametri. Pentru o paralelizare completă este necesar mai multă cercetare în algoritmi matematici folosiți.

Bibliografie

- 1: Zimmermann, Philip R., and Philip R. Zimmermann. The official PGP user's guide. Vol. 5. Cambridge: MIT press, 1995.
- 2: Shor, Peter W. "Algorithms for quantum computation: discrete logarithms and factoring." Proceedings 35th annual symposium on foundations of computer science. Ieee, 1994.
- 3: Diffie, Whitfield, and Martin Hellman. "New directions in cryptography." IEEE transactions on Information Theory 22.6 (1976): 644-654.
- 4: Adrian, David, et al. "Imperfect forward secrecy: How Diffie-Hellman fails in practice." Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security. 2015.
- 5: Boyko, Victor, Philip MacKenzie, and Sarvar Patel. "Provably secure password-authenticated key exchange using Diffie-Hellman." International Conference on the Theory and Applications of Cryptographic Techniques. Springer, Berlin, Heidelberg, 2000.
- 6: Jablon, David P. "Strong password-only authenticated key exchange." ACM SIGCOMM Computer Communication Review 26.5 (1996): 5-26.
- 7: Regev, Oded. "On lattices, learning with errors, random linear codes, and cryptography." Journal of the ACM (JACM) 56.6 (2009): 1-40.
- 8: Ding, Jintai, et al. "Provably secure password authenticated key exchange based on RLWE for the post-quantum world." Cryptographers' Track at the RSA Conference. Springer, Cham, 2017.
- 9: Bennett, Charles H., and Gilles Brassard. "Quantum cryptography: Public key distribution and coin tossing." arXiv preprint arXiv:2003.06557 (2020).
- 10: Harkins, Dan. "Dragonfly key exchange." Internet Requests for Comments, RFC Editor, RFC 7664 (2015).
- 11: Bellare, Mihir, and Phillip Rogaway. "Random oracles are practical: A paradigm for designing efficient protocols." Proceedings of the 1st ACM conference on Computer and communications security. 1993.
- 12: Chen, Lily. Recommendation for key derivation using pseudorandom functions. No. NIST Special Publication (SP) 800-108 (Withdrawn). National Institute of Standards and Technology, 2008.
- 13: Clarke, Dylan, and Feng Hao. "Cryptanalysis of the dragonfly key exchange protocol." IET Information Security 8.6 (2014): 283-289.
- 14: Lancrenon, Jean, and Marjan Škrobot. "On the Provable Security of the Dragonfly protocol." International Conference on Information Security. Springer, Cham, 2015.
- 15: Lancrenon, Jean, and Marjan Škrobot. "On the Provable Security of the Dragonfly protocol." International Conference on Information Security. Springer, Cham, 2015.
- 16: Vanhoef, Mathy, and Eyal Ronen. "Dragonblood: A Security Analysis of WPA3's SAE Handshake." IACR Cryptology ePrint Archive 2019 (2019): 383.

NECLASIFICAT

17: Tschacher, Nikolai "Dragonfly-SAE", 2018, GitHub repository,
github.com/NikolaiT/Dragonfly-SAE

NECLASIFICAT
50 din 50

