VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
KOMPIUTERIJOS KATEDRA

Baigiamasis magistro darbas

Lygiagretieji algoritmai kriptoanalizėje

Parallel algorithms in cryptoanalysis

Atliko: 2 kurso, 9 grupės studentas
Aleksandras Voitechovskis (parašas)

Darbo vadovas:
Doc. Tadas Meškauskas (parašas)

Darbo recenzentė:
Dr. Jolita Ignatavičiūtė (parašas)

Vilnius
2007

Turinys

Anotacija....3
Summary....3
Įvadas....4
Tikslas....4
Uždaviniai....4
Rezultatai....4
Aktualumas....4
1. Teorinė dalis....8
1.1. Lygiagretumas....8
1.1.1. Procesai....8
1.1.2. Srautai....8
1.1.3. Semaforai....9
1.2. Kriptosistemos....10
1.2.1. DES....11
1.2.2. Atsparumas....12
1.2.3. RSA....12
1.2.4. Šifravimas....13
1.2.5. Elektroninis parašas....13
1.2.6. Atsparumas....14
1.2.7. Bendras saugumas....15
2. Analitinė dalis....16
2.1. Lygiagretaus algoritmo pavyzdys....16
2.1.1. Kriptografijos sistema DES....16
2.1.2. DES sistemos kritpoanalizė....16
2.2. Programos veikimo rezultatas....17
2.3. Rezultatas....18
3. Praktinė dalis....19
3.1. Techninės galimybės....19
3.1.1. Klasteris....19
3.1.2. LAM/MPI....20
3.1.3. BalticGrid....20
3.2. Kriptografijos bibliotekų naudojimas....21
3.2.1. Biblioteka Crypto++....21
3.3. Vertinimo kriterijai ir metodai....22
3.4. Problemos ir nesklandumai....23
3.5. Efektyvumo analizė....25
3.5.1. Efektyvumas....25
3.6. Eksperimentai....25
3.6.1. Užduočių vykdymas klasteryje....26
3.6.2. Užduočių vykdymas BalticGrid aplinkoje....28
3.7. Tyrimo rezultatas....31
Išvados ir rekomendacijos....32
Literatūros sąrašas....33
Priedas Nr. 1....34
Priedas Nr. 2....34
Priedas Nr. 3....36
Priedas Nr. 4....36

Anotacija

Baigiamajame magistro darbe nagrinėjama simetrinio kodavimo rakto atsparumo nulaužimams problema ir lygiagrečiųjų algoritmų bei paskirstytų skaičiavimų įtaka DES šifro kriptoanalizei. Pagrindinis darbo tikslas yra ištirti lygiagrečiųjų algoritmų ir paskirstytų skaičiavimų naudojimo simetrinio šifro kriptoanalizėje efektyvumą. Siekiant užsibrėžto tikslo yra išnagrinėti populiariausi simetriniai kodavimo algoritmai, išanalizuoti kriptoanalizės metodai ir nustatyti lygiagrečiųjų algoritmų efektyvumo kriterijai. Pasinaudojus MPI bibliotekos ir BalticGrid aplinkos siūlomais lygiagrečiųjų skaičiavimų principais, yra sukurti lygiagretūs kriptoanalizės algoritmai DES kriptosistemai bei ištirta tų algoritmų įtaka kriptoanalizės efektyvumui. Išanalizavus susijusią literatūrą ir atlikus eksperimentus, yra suformuluotos ir pateiktos baigiamojo darbo išvados bei siūlymai.

Summary

The title of this work is “Parallel algorithms in cryptoanalysis”. The main idea of this individual final work of Master studies is to research parallel algorithms and Grid usages effectivity in cryptoanalysis.

In the beginning of the work author provides the information on existing parallel programming methods, existing cryptosistems, symetric encrypt decrypt algoriphm and their hacking possibilities. In the second part of work author carrying out cryptoanalysis of DES cryptographies algorithms and choosing a parallel algorithm, based on MPI protocol to the future researches and experiments.

Further based on cryptoanalysis of DES cryptographies algorithms results author describing experiment which is necessary to achived the main work goal. The received results of work and formulated conclusion finish research and show, that all collected information is analyzed, and the purposes put by the author are executed.

In the closing part of the work author presents the main results of the work and suggests some recommendations.

Įvadas

Tikslas

Darbo tikslas yra ištirti lygiagrečiųjų algoritmų ir paskirstytų skaičiavimų naudojimo simetrinio šifro kriptoanalizėje efektyvumą.

Uždaviniai

1. Išnagrinėti populiariausius simetrinius kriptografijos algoritmus.
2. Pasirinkti kriptoanalizės metodą, kurio pagrindu bus kuriami lygiagretūs, šifravimo rakto nulaužymo galimybių analizės, algoritmai .
3. Išskirti aibę kriterijų, kriptoanalizės lygiagrečiųjų algoritmų efektyvumui įvertinti.
4. Išnagrinėti lygiagretaus programavimo principus klasteriams, naudojant MPI biblioteką.
5. Išnagrinėti paskirstytų skaičiavimų principus BalticGrid aplinkoje.
6. Pagal pasirinktą metodą, sukurti lygiagrečius kriptoanalizės algoritmus kriptosistemai DES.
7. Ištirti lygiagrečiųjų skaičiavimų įtaką kriptoanalizės efektyvumui bei pateikti jų taikymo rekomendacijas.

Rezultatai

1. Išanalizuoti populiariausi simetriniai kriptografijos algoritmai ir tolimesniam nagrinėjimui pasirinktas DES algoritmas.
2. Pasirinktas rakto reikšmių perrinkimo kriptoanalizės metodas.
3. Apibrėžti kriptoanalizės lygiagrečiųjų algoritmų efektyvumo kriterijai.
4. Išnagrinėti MPI bibliotekos lygiagretaus programavimo principai.
5. Išnagrinėti paskirstytų skaičiavimų principai BalticGrid aplinkoje.
6. Sukurti lygiagretūs DES kriptoanalizės algoritmai, atitinkantys pasirinktą kriptoanalizės metodą.
7. Remiantis pasirinktu kriptoanalizės metodu ir apibrėžtais lygiagrečiųjų algoritmų efektyvumo kriterijais, ištirta lygiagrečiųjų algoritmų ir paskirstytų skaičiavimų įtaka kriptoanalizės efektyvumui bei pateiktos jų taikymo rekomendacijos.

Aktualumas

Nuo neatmenamų laikų žmonės turėjo paslapčių – konfidencialios informacijos, kurios neturėtų žinoti kiti, svetimi žmonės ir tuo labiau priešai. Sudėtingiausia ir kelia daugiausiai sunkumų bandymai išlaikyti paslaptį neizoliuotoje žmonių grupėje, kur vyksta komunikacija ir informacijos mainai netik tos grupės ribose, bet ir su kitais žmonėmis. Šiuolaikiniame pasaulyje,

kompiuteriams ir kompiuteriniams tinklams pasiekus didžiausią skvarbos į žmonių gyvenimą lygį, kai kas sekunde yra siunčiami didžiuliai duomenų srautai, yra ypač svarbu apsaugoti privačia ir konfidencialią informaciją. Būtent šiam tikslui užtikrinti yra naudojami sudėtingiausi šifravimo arba kriptografijos algoritmai. [Sta05] [Zel99]

Taip pat negalima pamiršti atsitiktinių informacijos iškraipymų, kurie gali atsirasti informaciją perduodant kanalais, pavyzdžiui tinklais. Koduoti informaciją reiškia taip ją paruošti prieš perduodant į kanalą , kad būtų galima ištaisyti įvykusius iškraipymus, jeigu jų nėra itin daug. Kodavimo teorija pasiūlo konstruktyvius tokio paruošimo (kodavimo) ir iškraipymų ištaisymo (dekodavimo) algoritmus. Šie algoritmai taikomi daugelyje moderniosios komunikacijos sričių. Kitaip sakant informacijos saugumas reiškia ne tik jos konfidencialumo išlaikymą ir apsaugą nuo tyčinio informacijos paviešinimo bei pakeitimo, bet ir galimybę saugiai ją perdavinėti egzistuojančiais kanalais, užtikrinant mechanizmų ir algoritmų, gebančių atstatyti atsitiktinai iškraipytą informaciją, egzistavimą. [Sta02]

Kriptologija – taikomosios matematikos dalis, tyrinėjanti metodus, algoritmus, programines ir aparatines priemones užtikrinančias informacijos apsaugą (kriptografija), o taip pat skirta įvertinti tokios apsaugos efektyvumą (kriptoanalizė). Šiais laikais kriptografija plačiai naudojama slaptos informacijos saugojimui bei jos siuntimui atvirais tinklais tokiais, kaip internetas.[Sta02] [Zel99]

Bet kuri matematikos (informatikos) teorija sprendžia ne tik tiesioginius uždavinius, bet ir priešingas užduotis, kurių sprendimas dažnai yra žymiai sudėtingesnis. Kriptoanalizės tiriami uždaviniai yra priešingi kriptografijos, tačiau dažnai šios sritys yra viena kitą papildančios - geras kriptoanalizės supratimas leidžia kurti saugesnius kriptografijos metodus. Pagrindinis kriptoanalizės tikslas ne įsibrauti į kompiuterinius tinklus ir užvaldyti konfidencialią informaciją, bet įvertinti naudojamų ir projektuojamų kriptosistemų patikimumą. Kriptoanalizės metodai leidžia įvertinti saugumo lygį ir išreikšti kompiuterinių operacijų skaičiumi, kurios yra būtinos kriptoanalitikui, norint iššifruoti slaptažodį arba kitą tekstą. [MOV01]

Kriptoanalitines atakas galima suskirstyti į 4 kategorijas: [Zel99]

- Kriptoanalitinė ataka naudojant kriptogramas;
- Kriptoanalitinė ataka naudojant atvirus tekstus ir juos atitinkančias kriptogramas;
- Kriptoanalitinė ataka naudojant kriptoanalitiko parenkamus atvirus tekstus ir juos atitinkančias kriptogramas;
- Kriptoanalitinė ataka naudojant aktyvų aparatinį poveikį kriptosistemai.

Kaip galima suprasti kriptoanalizė ir kodo dešifravimas užima labai daug resursų, o ypač procesoriaus laiko skaičiavimui. Kai kurių šiuolaikinių šifrų nulaužimas pareikalautų daug

energijos sąnaudų ir užtruktų ne vienerius metus. O to dažniausiai pakanka, kad saugomi duomenys pasentų ir neturėtų jokios praktinės vertės. [Zel99]

Siekiant sumažinti būtino kriptoanalizei laiko sąnaudas yra prasminga dešifravimui naudoti lygiagrečiuosius arba asinchroninius algoritmus. Lygiagretieji algoritmai – užduoties sprendimo algoritmai, kai skaičiavimai yra vykdomi lygiagrečiai (vienu metu) skirtinguose procesoriuose. Tokius algoritmus yra prasminga naudoti superkompiuteriuose bei klasteriuose, kur jie išryškina aparatinės dalies galimybes. [Lev04] Teoriškai lygiagrečių algoritmų vykdymas bei skaičiavimų atlikimas turėtų trukti tiek pat kartų greičiau, lyginant su nuosekliaisiais algoritmais, kiek procesorių turi kompiuteris. Tačiau praktikoje tai toli gražu nėra dėsnis ir lygiagrečių algoritmų veikimo efektyvumas bei taikymo būtinumas priklauso nuo pačio kompiuterio architektūros. [SS02]

Pagrindinis parametras, taikomas klasifikuojant lygiagrečius kompiuterius, yra bendros arba paskirstytos atminties buvimas. Šiais laikais vis populiaresnės tampa kombinuotų, mišrių architektūrų idėjos. Pagal nusakyta kriterijų galima išskirti penkias pagrindinius lygiagrečių kompiuterių architektūras: [Bsu04] [SS02]

- MPP – (Massively Parallel Processing). Sistemą sudaro vienodi skaičiavimo mazgai, turintis vieną arba kelis centrinius procesorius (dažniausiai RISC), lokalią atmintį (priėjimo prie kitų mazgų atminties nėra) ir tinko adapterį arba komunikacinį procesorių. Bendras procesorių skaičius realiose sistemose siekia kalis tūkstančius.
- SMP – (Symmetric Multi-Processing). Sistemą sudaro vienodi centriniai procesoriai ir bendras atminties masyvas (dažniausiai iš kelių nepriklausomų blokų). Visi procesoriai turi vienodą priėjima prie atminties. Procesorius su atmintimi dažniausiai jungia bendra šina arba crossbar-komutatorius. Bendras didžiausias procesorių skaičius realiose sistemose apie 32.
- NUMA - (Nonuniform Memory Access). Sistemą sudaro vienodi baziniai moduliai, susidedantys iš kelių procesorių ir atminties bloko. Moduliai sujungti naudojant greitus komutatorius. Palaikoma bendra adresinė erdvė, aparatiniame lygmenyje palaikomas priėjimas prie nutolusios atminties. Realiose sistemose bendras procesorių skaičius siekia 256.
- PVP - (Parallel Vector Process). Sistemoje naudojami specialūs vektoriškai konvejeriniai procesoriai, kuriuose numatytos komandos, skirtos apdoroti nepriklausomų duomenų vektorius.
- Klasteriai – bendrojo naudojimo darbo stočių arba netgi personalinių kompiuterių rinkinys, naudojamas kaip pigi MPP architektūros realizacija. Mazgai sujungti tarpusavyje naudojant vieną iš egzistuojančių tinklų technologijų (Fast/Gigabit

Ethernet, Myrinet). Heterogeniniai klasteriai – sudaryti iš skirtingo galingumo arba skirtingų architektūrų kompiuterių. Klasterio mazgai gali būti naudojami kaip paprastos darbo stotys.

Šitame magistriniame darbe bus nagrinėjami algoritmai skirti naudoti klasteriuose, o jų veikimo įvertinimui ir efektyvumui nustatyti bus naudojamas VU MIF esantis klasteris bei „BalticGrid“ paskirstytų skaičiavimų tinklas. [Pst05] [Con04] [Bal05] Klasterių programavimas, kaip taisykle, telpa pranešimų perdavimo modelio ribose (dažniausiai MPI). Panašios sistemos pasižymi didelėmis pridėtinėmis išlaidomis lygiagrečių procesorių tarpusavio sąveikai, kas žymiai susiaurina potencialią sprendžiamų užduočių klasę. [Pst05]

MPI (Message Passing Interface) – funkcijų, užtikrinančių duomenų mainus tarp procesų, bibliotekos aprašymas. Kad tokia biblioteka veiktų, turi būti užtikrinta MPI realizacija konkrečioje aplinkoje. MPI realizacija suprantama kaip programinės priemonės, vykdančios visas MPI funkcijas. Šitos procedūros užtikrina porinius ir kolektyvinius mainus, realizuoja komutatorius, topologiją, komunikacijos su vykdomąja aplinka priemones ir daugybę kitų papildomų operacijų, susijusių, pavyzdžiui, su skaičiavimo paleidimu, lygiagrečių skaičiavimų efektyvumo įvertinimu ir t.t. [SS02] [Lev04] [MPI95] [MPI97]

Galimi 2 MPI realizacijos būdai: tiesioginė realizacija konkrečiam kompiuteriui ir realizacija naudojant ADI (Abstract Device Interface). Pirmas būdas nėra pageidautinas, nes realių lygiagrečių sistemų egzistuoja labai daug. Antru atveju, realizuojamas tik vienas ADI, o paskui jo architektūra yra realizuojama konkrečioje sistemoje lygiagrečių įrenginių gamintojų. ADI, siekiant užtikrinti pranešimų mainus, privalo pateikti 4 funkcijų rinkinius: [SS02]

- Gaunamų ir siunčiamų pranešimų aprašymas;
- Duomenų kaita tarp ADI ir perduodančia aparatūra;
- Pakibusių pranešimų (išsiustų ir gaunamų) sąrašo valdymas
- Pagrindinės informacijos apie vykdomąją aplinką ir jos būsena gavimas

Planuojama, kriptoanalizės ir kriptografijos lygiagrečius algoritmus, realizuoti naudojant LAM/MPI (Local Area Multicomputer/Message Passing Interface) programinę įrangą, kuri yra suinstaliuota MIF VU klasteriuose ir leidžia vykdyti bei kompiliuoti aplikacijas, parašytas C bei C++ programavimo kalbomis. Standartas LAM/MPI užtikrina aukštą produktyvumą skirtinguose sistemose: nuo mažų su vienu centriniu procesoriumi, iki didžiulių SMP mašinų su greitai veikiančiais tinklais. Papildomai prie greitaveikos ir stabilumo, LAM siūlo eilę priemonių, padedančių kurti ir vystyti dideles MPI aplikacijas. [Pst05] [LAM05]

1. Teorinė dalis

1.1. Lygiagretumas

1.1.1. Procesai

Kiekvienam programos vykdymui atitinka operacinės sistemos procesas. Kiekvienas procesas turi savo resursus, t.y. vykdomas savo virtualioje atmintyje ir vienas procesas negali nekontroliuojamu būdu skaityti arba rašyti į kito proceso atmintį. Tačiau kontroliuojamas procesų bendradarbiavimas yra leidžiamas sistemos, tai pasiekiama tuo būdu, kad vienas atminties segmentas yra dalinamas tarp kelių procesų virtualios atminties. Kiekvienas procesas gali sugeneruoti naują identišką procesą ir sulaukti savo pavaldžių procesų darbo pabaigos.

Po proceso sukūrimo tėvinis ir vaikinis procesai gali laisvai keisti savo kontekstą. Tai yra kiekvienas iš minėtų procesų gali įvykdyti bet kurį sisteminio kvietimo variantą, kuris visiškai pakeis proceso kontekstą.

1.1.2. Srautai

Srautas arba kitaip vadinama gija – tai nepriklausomas valdymo srautas, turintis savo komandų skaitliuką, vykdomas kurio nors proceso kontekste. Proceso konteksto apibudinimas, faktiškai, keičiamas tokiu būdu. Viskas, kas nepriklauso valdymo srautui (virtuali atmintis, atvirų failų deskriptoriai ir t.d.), lieka bendrame proceso kontekste. Veiksmai būdingi valdymo srautui (registrų kontekstas, skirtingo lygio stekai ir kita) pereina iš proceso konteksto į gijos kontekstą. Tuo pat metu, visų srautų kontekstai lieka priklausomi juos inicializavusio proceso kontekstui.

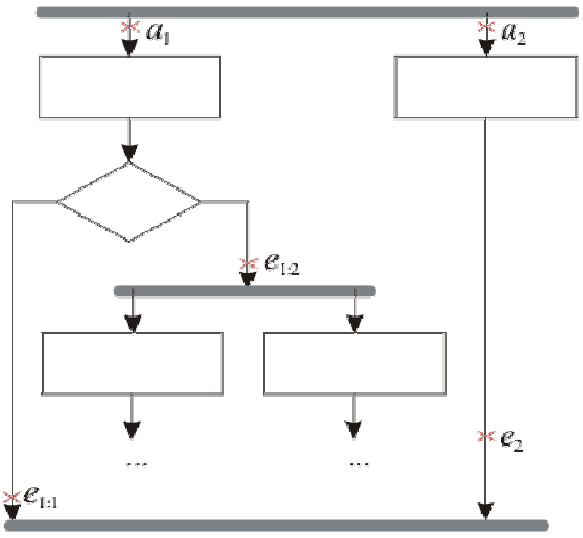
Kadangi vieno proceso srautai yra vykdomi bendroje virtualioje atmintyje, vienos proceso gijos stekas iš esmės nėra apsaugotas nuo savaiminio kreipimosi iš kitų proceso srautų.

Lygiagretumą labai retai nagrinėja neapibrėžiant ryšio su nuosekliu stiliumi. Iš tikrųjų, dažniausiai realiame gyvenime yra sutinkamos lygiagrečiai-nuoseklios sistemos. Egzistuoja rinkinys nuoseklių veiksmų, kurie vienokiu ar kitokiu būdu susikerta tarpusavyje. Yra priimta, šį nuoseklumą vadinti – srautu.

Srautą galima apibūdinti kaip nuosekliai vykdomi veiksmai, kurie yra aprėžti pradine ir galutine arba galutinėmis būsenomis. Galima padaryti prielaidą, kad srautas yra elementaraus nuoseklaus skaitinio automato analogas. Tačiau čia reiktų padaryti išlygą, nes minėto automato pradinė ir galutinė būsenos sujungtos į viena, o srautui tokio sąryšio apibrėžti negalima, nes:

1) Srautas gali pereiti iš vienos būsenos į kitą;
2) Laukti darbo pabaigos – kol užsibaigs kitos lygiagrečios gijos;
3) Laukti darbo pradžios – kol bus galima startuoti.

Galutinės būsenos gali būt kelios. Pav.1 pavaizduotos kelios galimos gijos a_1 būsenos: ji gali inicializuoti naujus srautus, arba užsibaigti kartu su srautu a_2.



Pav.1 Galimos galutinės gijų būsenos

Srautų inicializavimas – kai vienas srautas baigia savo darbą ir paleidžiama daugybė kitų. Yra tikėtina, kad bus paleistas tik vienas naujas srautas, bet tokiu atveju lygiagretumas prarandą prasmę. Pav.1 šis naujų srautų inicializavimas pavaizduotas stora horizontalia linija. Dėl sąlyginių viršūnių yra 3 variantai perėjimo nuo vieno srauto pabaigos į kitų pradžią:

1) Besąlyginis perėjimas;
2) Sąlyginis perėjimas be ciklo;
3) Sąlyginis perėjimas, kaip laukianti viršūnė.

Jei srautas yra galutinėje būsenoje, į pradinę jis gali pereiti tik tuo atveju, kai pasileis visi inicializuojami srautai. Jei toje būsenoje nėra laukiančių srautų, tai iškart įvyksta perėjimas.

1.1.3. Semaforai

Siekiant užkirsti kelią aukščiau aprašytai situacijai, reikia įvesti lygiagrečių procesų sinchronizavimo sistemą. Sprendimo esmė yra, leisti kreiptis į kritinę sekciją (proceso vieta, kai jam reikalingas resursas) tik vienam iš kelių asinchroninių procesų. Šį sprendimą pasiūlė Deikstra semaforų pavidalu. Semaforas tai yra kintamasis S, susijęs, pavyzdžiui, su resursu, ir gaunantis 2 būsenas: 0 – uždraustas kreipimasis ir 1 – kreipimasis leidžiamas. S valdo dvi operacijos V ir P. Operacija V keičia semaforo S reikšmę į S+1. Operacijos P veiksmai yr tokie:

- Jeigu S nelygu 0, tai P mažina reikšmę vienetu;
- Jeigu S lygu 0, tai P nekeičia S reikšmės ir nesibaigia iki to momento, kol koks nors kitas procesas nepakeis S reikšmę naudojant operaciją V.

Operacijos v ir P yra nedalomos, t.y. jos negali būti vykdomos vienu metu.

1.2. Kriptosistemos

Kritposistemos (šifrai) su slaptu raktu, taip pat vadinamos simetrinėmis, sprendžia informacijos slaptumo išsaugojimo problemas. Pagrindinis reikalavimas simetrinėms sistemoms yra atsparumas. Absoliučiai atsparios kriptosistemos yra šifrai su vieną kartą naudojamais raktais. Tokios sistemos yra pakankamai paprastos šifravimo ir dešifravimo procedūrų atžvilgių, tačiau jos reikalauja neaprėpiamai didelio raktinio materialo, o tas daro minėtų sistemų naudojimą labai brangiu.

Raktų paskirstymas saugiais kanalais yra fundamentalus uždavinys. Sprendžiant jį, vienu metu sprendžiama ir rakto identifikavimo užduotis. Iš tikrųjų gavėjas turi ne tik gauti slaptą raktą, bet ir įsitikinti, kad būtent šis raktas buvo išsiustas teisėto siuntėjo. Slaptų raktų paskirstymo problema dažnai nustumia tų raktų identifikavimo problemą į antrą planą. Reiktų pabrėžti, jog antroji problema taipogi yra fundamentali. Maža to, ši problema yra fundamentali ir kriptografijoje su dviem aktais, kuri daugelių atvejų siūlo žymiai ekonomiškesnius ir patogius slaptų raktų skirstymo mechanizmus.

Praktikoje labiausiai yra paplitę šifrai su galutiniu raktu (dažniausiai nuo 64 iki 256 bitų). Tokie šifrai yra sąlyginai atsparus atakoms. Visada egzistuoja teorinė tikimybė nustatyti raktą naudojant pilną rakto intervalo perrinkimą. Tačiau realiame gyvenime, tokia ataka nėra priimtina dėl labai didėlio laiko resursų naudojimo.

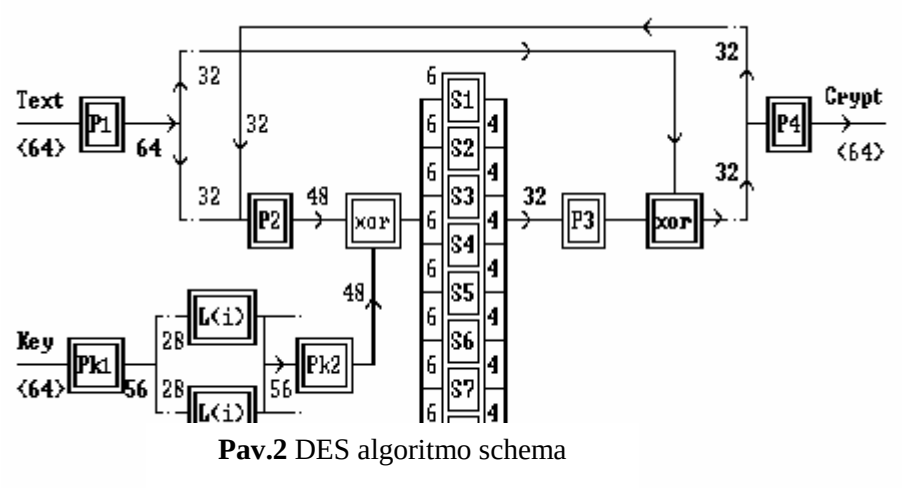
Jei nagrinėti atakas, pagrįstas žinomais arba specialiai parinktais tekstais, tai iš principo yra galimybė nustatyti rakto apskaičiavimo metodus. Šifravimo algoritmai, privalo būti tokie, kad parūpintų aukštą geriausio atakos algoritmo išskaičiavimo sudėtingumą. Dėl šios priežasties šifrai su galutiniu raktu dar vadinamos ir atspariais išskaičiavimams.

Atsparūs išskaičiavimui šifrai praktikoje yra naudojami nepalyginamai plačiau negu teoriškai atsparios kriptosistemos, nes jie neturi aibės praktiniam naudojimui svarbių neigiamų aspektų, susijusių su absoliutaus atsparumo siekimu. Išskaičiavimui atsparių sistemų pasirinkimas leidžia sukurti kokybiškai naujas kriptosistemas – šifrai su dviem raktais, dar vadinami kriptosistemos su atviru raktu. Paskutinis terminas pabrėžia principingą visiems žinomo rakto naudojimo reikšmę. Slapto rakto naudojimo būtinumas yra akivaizdus, kadangi jis reikalingas visų tipų šifruose.

Dviejų raktų šifrai įneša principingai naują tai, jog yra galimybė kurti kriptografinius protokolus, sprendžiančius bendradarbiavimo problemas šalių, nepasitikinčių viena kita. Tokios galimybės atsiradimo priežastis yra ta, kad slaptas raktas lieka žinomas tik vienam naudotojui, kuris jį sugeneravo. Kriptosistemuose su vienų raktų, minėtas slaptas raktas turi būti žinomas mažiausiai dviems šalims, kas reiškia, jog tarp tų šalių turi egzistuoti pasitikėjimas. [MM05]

1.2.1. DES

Vienas iš labiausiai žinomų simetrinio šifravimo algoritmų yra DES. Čia bus nuodugniau ištirtas pats kodavimo algoritmas ir aptartas jo atsparumas atakoms.



Pav.2 DES algoritmo schema

Text	Pradinis tekstas (64 bitų blokas)
Crypt	Užšifruotas blokas
Key	Raktas (64 bitų)
skaičiai	Bitų skaičius algoritmo atšakoje
P, Pk	Perstatymai
S	Perstatymas 6 bitai -> 4 bitai
L(i)	Pastūmimas (i – iteracijos numeris)
xor	Sudėtis pagal modulį 2
	Bitų eilučių konkatenacija, viršutinė priekyje
	Eilučių skaidymas į dvi, priekinė viršuje
	Apribuota taškais vieta kartojasi 16 kartų

Perstatymai vykdomi pagal formulę D[i]=S[P[i]], kur

S	Pradinė eilutė (simbolių masyvas, numeruojamas nuo vieneto)
D	Perstatymo rezultatas (simbolių masyvas, numeruojamas nuo vieneto)
P	Perstatymų lentelė (indeksų masyvas eilutėje S)

S – perstatymas 6 -> 4. Vietoje 6 bitų atitinkamai įrašomi 4. Perstatymas atliekant pagal taisyklę: tarkime pradinė bitų eilutė – /abcdef/, tada /af/ - eilės numeris, o /bcde/ - stulpelio numeris. Eilė ir stulpelis apibūdina rezultato vietą S-lentelėje. Pvz., naudojant lentele S6, skaičius 58 (111010) pervedamas į 13 (1101).

DES operuoja blokais po 64 bitus ir naudoja 56 bitų raktą (8 lyginiai bitai pilno 64 bitų rakto nenaudojami). DES – simetrinė kriptosistema, apibrėžta kaip 16 bitų Feistelio[1] šifras, buvo iš pradžių skirta aparatiniam realizavimui. Tačiau aplinkose su didelių kiekių vartotojų yra sunku organizuoti saugų rakto paskirstymą; tam labiau tinka bendro rakto kriptografija. [Vin98] [MM05]

1.2.2. Atsparumas

Nepaisant ilgamečių tyrinėtojų pastangų, efektyvių DES šifro atakų nėra. Paprasčiausias atakos metodas – pilnas visų galimų raktų perrinkimas. Vėliau Hellman rado būdą kaip patobulinti pilną raktų perrinkimą su sąlyga, kad yra pakankamas atminties kiekis. Taip pat buvo keliami kaltinimai, kad NSA[2] specialiai padarė DES jautriu atakoms. Pirmą DES algoritmo ataką efektyvesnę negu pilnas perrinkimas pasiūlė Biham ir Shamir. Jie naudojo naują metodą, žinomą kaip diferencialinė kriptoanalizė. Šiai atakai reikia 2^{47} atvirų tekstų, parinktų puoliko, šifravimo. Teoriškai esant atskyrimo tašku, ši ataka yra nepraktiška dėl didžiulių reikalavimų duomenų parinkimui ir pačios atakos organizavimo sudėtingumo pagal pasirinktą atvirą tekstą. Patys atakos autoriai pareiškė, kad laiko DES šifrą apsaugotu. Anksčiau Matsui sugalvojo kitą ataką – linijinė kriptoanalizė. Šis metodas leidžia atkurti DES raktą naudojant 2^{43} žinomų atvirų tekstų analizę. Visi šitie pareiškimai leido suabejoti DES saugumu, tačiau per visą laiką nebuvo rastas nei vienas efektyvesnis DES šifro nulaužimo metodas bendru atveju, išskyrus pilną rakto perrinkimą. Reiktų pabrėžti, kad šiais laikais technika pasiekė tą vystymosi lygį, kada viengubas DES algoritmas jau nėra apsaugotas, kadangi 56 bitų ratas yra visiškai neatsparus pilnam perrinkimui. Faktiškai DES jau nebenaudojamas, išskyrus kai kurias sistemas, kur duomenų saugumas nėra kritinis parametras, ir šiuo metu vietoje jo naudojamas Triple DES, Blowfish ir kiti. Standartu dabar yra AES šifras. [MM05]

1.2.3. RSA

RSA – kriptosistema su dviem raktais (vienas yra atviras), naudojama tokiose apsaugos mechanizmuose kaip šifravimas ir elektroninis parašas. RSA sukurta 1977 metais ir pavadinta jos kūrėjų garbei: Ronald Rivest, Adi Shamir ir Leonard Adleman.

RSA algoritmas veikia sekančiu būdu: imami du pakankamai dideli pirminiai skaičiai p ir q, bei apskaičiuojama j sandauga $n = p*q$ (n – vadinamas moduliu). Toliau parenkamas toks skaičius e, kad būtų tenkinama sąlyga $1 < e < (p-1)*(q-1)$ ir nebūtų bendrų daliklių išskyrus 1 su

[1] **Feistel cipher** – Feistelio šifras; speciali blokinių šifrų klasė, kur atviras tekstas koduojamas daug kartų pritaikant kiekvienam blokui vieną ir tą patį pertvarkymą

[2] **NSA (National Security Agency)** – Nacionalinio saugumo agentūra; valstybinis JAV departamentas kontroliuojantis komunikacijas, tame tarpe ir užsienio.

skaičiumi *(p-1)*(q-1*). Paskui randamas skaičius *d*: rezultate turėtų būti tenkinama sąlyga *(e*d-1)* dalinasi iš *(p-1)*(q-1).*

- e – atviras rodiklis
- d – privatus rodiklis
- (n; e) – atviras raktas
- (n; d) – privatus raktas

Daliklius p ir q galima arba sunaikinti, arba išsaugoti kartu su privačiu raktu.

Jei egzistuotų efektyvūs metodai skaidymo į daliklius, tai, išskaidžius n į daugiklius p ir q, galima išskaičiuoti privatų raktą d. Tokiu būdu kriptosistemos RSA atsparumas laikosi ant sunkiai išsprendžiamo – praktiškai neišsprendžiamo – skaidymo n į daugiklius uždavinio (t.y. negalimybės n faktoringo), nes šiuo metu efektyvaus būdo rasti minėtus daugiklius nėra. [Vin98] [MM05]

1.2.4. Šifravimas

Tarkime, siuntėjas nori išsiusti gavėjui pranešimą M. Siuntėjas sudaro užkoduotą tekstą C, keliant pranešimą M laipsniu e ir dauginant iš modulio n: $C = M^{e*(\mathrm{mod}\ n)}$, kur e ir n – atviras gavėjo raktas. Po to siuntėjas išsiunčia tik C (užkoduotą tekstą) gavėjui. Norint iššifruoti gautą tekstą, gavėjas kelia gautą tekstą C laipsniu d ir daugina iš modulio n: $M = C^{d*(\mathrm{mod}\ n)}$. Priklausomybė tarp e ir d garantuoja, kad gavėjas taisyklingai apskaičiuos M. Kadangi tik gavėjas žino d, todėl tik jis turi galimybę iššifruoti gautą pranešimą.

1.2.5. Elektroninis parašas

Tarkime, siuntėjas nori išsiusti gavėjui pranešimą M ir gavėjas turi būti tikras, kad pranešimas nebuvo nulaužtas, ir kad autoriumi iš tikrųjų yra siuntėjas. Siuntėjas generuoja elektroninį parašą S, keliant pranešimą M laipsniu d ir dauginant iš modulio n: $S = M^{d*(\mathrm{mod}\ n)}$, kur d ir n – privatus gavėjo raktas. Po to siuntėjas išsiunčia ir M ir S gavėjui. Norint patikrinti gautą parašą, gavėjas kelia S laipsniu e ir daugina iš modulio n: $M = S^{e*(\mathrm{mod}\ n)}$, kur e ir n – atviras siuntėjo raktas. Tokiu būdu šifravimas ir autoriaus autentifikavimas atliekami be privačių (slaptų) raktų perdavimo: abu korespondentai naudoja tik atvirą raktą savo korespondento arba savo privatų raktą. Išsiusti užšifruotą ir patikrinti pasirašytą pranešimą gali visi, bet iššifruoti arba pasirašyti pranešimą gali tik atitinkamo privataus rakto savininkas.

1.2.6. Atsparumas

Egzistuoja bent keli RSA šifro nulaužimo variantai. Efektyviausia ataka yra rasti privatų (slaptą) raktą atitinkantį reikiamam atviram raktui. Tai leis atakuojančiam skaityti visus pranešimus, užšifruotus atviru raktu ir padirbinėti parašus. Tokią ataką galima įvykdyti radus pagrindinius modulio daugiklius - p ir q. Atakuojantis gali lengvai išskaičiuoti privatų rodiklį d, žinant p,q ir e (bendras rodiklis). Pagrindinis sunkumas – pagrindinių n daugiklių radimas. RSA saugumas priklauso nuo skaidymo į daugiklius, kas yra sunkiai išsprendžiama problema, neturinti efektyvių sprendimo metodų.

Faktiškai, privataus rakto atkūrimo uždavinys ekvivalentus skaidymo į daugiklius uždaviniui: galima naudoti d norint rasti n daugiklius, arba, atvirkščiai, naudoti n norint rasti d. Reiktų pabrėžti, kad skaičiavimo aparatūros tobulėjimas savaime nemažina RSA šifro saugumo, jei raktai bus pakankamai didelio ilgio. Aparatūros tobulėjimas faktiškai padidina kriptosistemos atsparumą.

Kitas RSA algoritmo nulaužimo būdas: rasti metodą apskaičiuoti šaknį laipsnio e is mod n. Kadangi $C = M^{e*(\mathrm{mod}\ n)}$, tai šaknimi bus pranešimas M. Radus minėtą šaknį, galima atidarinėti užšifruotus pranešimus ir padirbinėti parašus, netgi nežinant privataus rakto. Ši ataka neekvivalenti faktoringo, bet yra nežinomi metodai, leidžiantys nulaužti RSA tokiu būdu. Tačiau, atskirais atvejais, kai vieni ir to pačio pakankamai mažo ilgio rodiklio pagrindu yra šifruojama daug susijusių pranėšimų, tai tada galimybė perskaityti pranešimus yra. Minėtos atakos yra vieninteliai metodai iššifruoti visus pranešimus užkoduotus RSA raktu.

Egzistuoja ir kiti atakų tipai, leidžiantys, tarp kitko, nulaužti tik vieną pranešimą ir neleidžiantys skaityti visų kitų, užšifruotų tuo pačiu raktu. Paprasčiausias būdas atakuoti vienintelį pranešimą – ataka pagal numatomą atvirą tekstą. Atakuojantis, turintis užšifruotą tekstą, daro prielaidą, kad pranešimas turi kokį nors spėjamą tekstą, toliau jis užšifruoja minėtą tekstą su atviru gavėjo raktu ir lygina gautą tekstą su turimu užkoduotu. Tokią ataką galima apeiti pridėjus į pranešimą keletą atsitiktinių bitų. Kitas variantas atakuoti vienintelį pranešimą tuo atveju, kai kažkas siunčia vieną ir tą patį pranešimą M trims korespondentams, kiekvienas iš kurių naudoja bendra rodiklį e = 3. Žinant tai, atakuojantis gali perimti ir iššifruoti šituos pranešimus. Tokia ataka irgi bus neefektyvi, jei prieš kiekvieną šifravimą įdėti į pranešimą kelis atsitiktinius bitus. Dar vienas būdas nulaužti RSA, su tikslu padirbti parašą: atakuojantis sudaro kažkurį užkoduotą tekstą ir gauna atitinkantį atvirą tekstą, pavyzdžiui, apgaule priverčiant vartotoją iššifruoti padirbtą pranešimą.

Aišku, egzistuoja atakos ne pačios kriptosistemos, bet visos sistemos bendrai. Tačiau tokių atakų negalima nagrinėti kaip RSA nulaužimas, nes jos byloja ne apie RSA algoritmo, bet apie jo konkrečios realizacijos, silpnumą. [MM05]

1.2.7. Bendras saugumas

Praktikoje RSA ir DES kriptosistemos dažnai naudojamus kartu, dėl pranešimo šifravimo su RSA raktu, naudojantis skaitmeniniu voku. Tarkime, siuntėjas siunčia gavėjui užkoduotą pranešimą. Iš pradžių siuntėjas šifruoja pranešimą su DES šifru, atsitiktinai parinkus kažkurį DES raktą, ir vėliau koduoja tą DES raktą, naudojantis atviru gavėjo RSA raktu. Pranešimas, užkoduotas DES raktu, ir paskui pats DES raktas, užšifruotas RSA raktu, bendrai formuoja elektroninį voką RSA ir išsiunčiami gavėjui. Gavęs elektroninį voką, gavėjas dekoduoja DES raktą, savo slapto RSA rakto pagalba ir paskui naudoja rastą DES raktą, kad iššifruotų patį pranešimą. Toks metodas leidžia sujungti DES algoritmo didelio greičio privalumus su RSA kriptosistemos patikimumu. [MM05]

2. Analitinė dalis

2.1. Lygiagretaus algoritmo pavyzdys

2.1.1. Kriptografijos sistema DES

DES (Data Encryption Standart) – simetrinis šifravimo algoritmas, kuriame vienas raktas naudojamas kaip duomenų šifravimui taip ir dešifravimui. Šis algoritmas sukurtas firmos IBM ir JAV vyriausybės priimtas kaip oficialus kodavimo standartas 1977 metais. Buvo pakeistas tik 2001 metais, nes neatitiko šiuolaikinių saugumo reikalavimų. DES naudoja blokus po 64 bitų ir 16 ciklų Feistelio tinklo struktūrą. Šifravimui naudojamas raktas yra 56 bitų. Tokiu būdu, kai turimas dvejetainis tekstas (N = 2), šifruojama blokais po 64 bitus ir naudojamas 64 bitų raktas. Todėl $|\Theta| = 2^{56}$, $W = 2^{56}$ $q\omega(64)$. [SS02]

2.1.2. DES sistemos kritpoanalizė

Naudosime paprasčiausią ir bendru atveju efektyviausią DES šifro atsparumo nulaužymams tyrimo metodą - šifravimo rakto reikšmių perrinkimą. Reikės atlikti pilną rakto reikšmių perrinkimą nurodytame diapazone. Taip pat reikės kiekvienai rakto reikšmei kviesti dešifravimo procedūrą su šia rakto reikšme. Paprastumo dėlei laikysim, kad raktas rastas, kai dešifruotas tekstas sutaps su pradiniu.

Procedūra DESshifr (in, out); realizuoja dvejetainio teksto šifravimą pagal aukščiau aprašytą DES algoritmą. Čia in – pradinio pranešimo blokas, šifravimo bloko didis yra 64 bitai; out – užkoduotas tekstas. Procedūra DESdeshifr (out, outin, key); dešifruoja dvejetainį tekstą. Čia out – užkoduotas tekstas; outin – dešifravimo rezultatas, naudojant raktą – key. Kadangi aktuali sritis yra kriptoanalizės algoritmų išlygiagretinimas, pats kodavimo/dekodavimo algoritmas smulkiai nagrinėjamas nebus.

DES algoritmo raktas yra 64 bitų, todėl jį galima išreikšti dviem 32 bitų kodais, keturiais 16 bitų arba aštuoniais 8 bitų. Pasirinkime pirmą rakto variantą: unsigned long key_data[2].
Nuoseklus rakto perrinkimo metodo algoritmas – [Priedas Nr 1].

Lygiagretaus algoritmo [Priedas Nr 2] principas yra tas, kad galima padalinti tarp procesų visų perrenkamų raktų reikšmių diapazoną. Paminėtina, kad šis lygiagretaus algoritmo pavyzdys nėra vienintelis, tačiau šiame darbo etape yra siekiama tik susipažinti su lygiagrečių algoritmų naudojimų, o šis variantas yra natūraliausias būdas išlygiagretinti rakto perrinkimo algoritmą. Tarkime, min – apatinė, max – viršutinė rakto reikšmių riba; myid – proceso numeris, numproces – procesų kiekis; n1 – apatinė, n2 – viršutinė rakto riba duotam procesui. Tada galimas darbo paskirstymas tarp procesorių:

```
h   = (max – min  + 1) / numproces;
ost = (max – min  + 1) % numproces;
n1  =  min + myid * h;
n2  =  n1  + h;
if ((ost != 0) && (myid == numproces)) n2 = n2 + ost;
```

Kadangi bendras rakto variantų kiekis daug didesnis už procesų kiekį, galima teigti, kad bendras skaičiavimų kiekis bus lygiai paskirstytas tarp procesų. Norint sužinoti, ar rastas raktas, patogu naudoti kolektyvinę funkciją:

```
MPI_Allgather (&num, MPI_INT, buf, 1, MPI_INT, MPI_COMM_WORLD).
```

Visas rakto perrinkimo lygiagretus algoritmas yra pateiktas [Priedas Nr 2].

2.2. Programos veikimo rezultatas

Aukščiau paminėtos programos esmė yra parodyti lygiagrečiųjų algoritmų taikymo kriptoanalizės uždaviniuose prasmingumą ir efektyvumą, o taipogi praktikoje pritaikyti sukauptas lygiagretaus programavimo principų bei algoritmų išlygiagretinimo žinias.

Paprastumo dėlei šiame etape apsiribojama vienu algoritmo efektyvumo matu – laiku, reikalingu nustatyti kodavimo raktą. Laikas yra matuojamas naudojant standartinę MPI funkciją MPI_Wtime().

Žemiau yra pateikta lentelė, pavaizduojanti rakto suradimo laiko priklausomybę no rakto reikšmės, naudojant lygiagretųjį (aštuonių mazgų paskirstytų skaičiavimų tinklas) ir nuoseklų algoritmus.

Rakto reikšmė (long key_data[2])	Lygiagretus algoritmas	Nuoseklus algoritmas
0x0 0x68c	0,1 s	1 s
0x1 0x68c	4,2 min	1,6 h
0x08 0x68c	37 min	-

Akivaizdu, kad lygiagretus algoritmas šiuo atveju yra žymiai pranašesnis už nuoseklų, be to, kuo daugiau šifravimo rakto perrinkimų tenka vykdyti, tuo akivaizdesnis pranašumas. Tačiau, norint įvykdyti perrinkimą visame rakto reikšmių diapazone, ir šio algoritmo veikimas užtruks nemažai dienų. Kas realiame taikyme nėra priimtina. Norint pagreitinti algoritmą, reikia atlikti nuodugnesnę jo analizę ir kitus išlygiagretinimo būdus. Tai yra vienas iš tolimesniojo šio darbo tyrimo punktų.

2.3. Rezultatas

Sprendžiant iškeltą problemą – lygiagrečių algoritmų taikymas kriptografijoje ir kriptoanalizėje bei taikymo prasmingumas – šiame darbo etape pakako anksčiau išnagrinėtų ir susistemintų žinių, t.y. papildoma sprendimų paieška buvo nereikalinga. Tačiau tęsiant pradėtą darbą – efektyvesnių kriptoanalizės algoritmų išlygiagretinimo būdų paiešką, bus reikalinga nuodugnesnė susijusių egzistuojančių sprendimų analizė.

Rengiant analitinę darbo dalį, buvo padaryta:

1. Išnagrinėtas šifravimo algoritmas DES;
2. Parašyta programa, realizuojanti DES algoritmą, C kalba;
3. Parašyta programa, realizuojanti nuoseklų DES šifravimo rakto suradimo algoritmą, C kalba;
4. Parašyta programa, realizuojanti lygiagretų DES šifravimo rakto suradimo algoritmą, C kalba;
5. Pritaikytos įgytos lygiagretaus programavimo žinios;
6. Panaudota MPI biblioteka, užtikrinanti duomenų mainus tarp procesų;
7. Įvykdyta programa VU MIF paskirstytų skaičiavimų tinkle - klasteryje;
8. Atliktas bandymas, įrodantis lygiagretaus algoritmo veikimo pranašumus prieš nuoseklu nagrinėjamam atvejui;
9. Uždokumentuoti DES kodavimo rakto radimo metodas ir jį pritaikančio lygiagretaus algoritmo realizacija.

3. Praktinė dalis

Tiriant kriptoanalizės algoritmo išlygiagretinimo galimybes yra pasitelktos dvi skaičiavimo paskirstymo metodikos: MPI bibliotekos funkcijų naudojimas ir BalticGrid siūlomos skaičiavimų išlygiagretinimų galimybės, naudojant JDL[3] failus, o taipogi šių dviejų metodikų bendras naudojimas. Yra išnagrinėtas minėtų metodikų naudojimo efektyvumas skirtingomis vykdymo sąlygomis: klasteryje su mažų mazgų skaičiumi, klasteryje su didelių mazgų skaičiumi, BalticGrid aplinkos klasteryje su mažu mazgų skaičiumi, BalticGrid aplinkos klasteryje su dideliu mazgų skaičiumi bei BalticGrid aplinkos keliuose klasteriuose su dideliu mazgų skaičiumi vienu metu.

Vykdant kriptografijos rakto parinkimo algoritmą, tyrimo tikslais yra taikomas supaprastintas DES kriptoanalizės modelis, kai koduojamą tekstą sudaro 8 simbolių blokas ir yra daroma prielaida, kad įeities tekstas yra žinomas. Pasirinktas modelis tinka užsibrėžtiems tikslams pasiekti, nes jis nesupaprastina patį rakto radimo metodą, bet sumažina procesoriaus darbo laiką, reikalingą ieškomam raktui identifikuoti. Kodavimo raktas yra atsitiktinai generuojamas ir naudojamas tiriant kiekvieną išlygiagretinimo metodiką bei su kiekviena aparatinės dalies architektūra.

Bandymai atliekami kelis kartus, siekiant apskaičiuoti vidutinį užduoties – kodavimo rakto nustatymo, vykdymo laiką. Būtent jis yra labiausiai tikėtinas užduoties vykdymo laikas. Tačiau pakankamai informatyvus ir įdomus yra mažiausias vykdymo laikas, kuris visada apribuotas iš apačios aparatūros parametrais, kadangi jis leidžia apibūdinti aparatūros galimybes.

3.1. Techninės galimybės

3.1.1. Klasteris

MPI išlygiagretinimo metodais pagrįstų kriptoanalizės algoritmų efektyvumas yra tiriamas naudojant Vilniaus universiteto Matematikos ir informatikos fakulteto Informacinių technologijų centro paskirstytų skaičiavimų tinklą – klasterį. Tokį pasirinkimą lėmė pakankamai didelis našumas, prieinamumas ir sąlyginai nedidelės duomenų mainų sąnaudos. Iš dviejų MIF ITC esančių klasterių buvo pasirinktas našesnis, realizuotas Linux operacinės sistemos aplinkoje – Linux PST. Prie pasirinkto klasterio buvo jungiamasi naudojant **ssh** protokolą realizuojančią programą bei atsižvelgiant į MIF klasterio specifiką.

[3] **Job Description Language** – užduoties failas Grid aplinkoje (*.jdl)

3.1.2. LAM/MPI

Linux PST yra suinstaliuota LAM/MPI programinė įranga. Tai pramoninio standarto MPI-1.2 (iš dalies MPI-2) Open Source[4] realizacija. Šis paketas apibrėžia pranešimų mainų interfeisą tarp klasterio mazgų, užtikrindamas lygiagrečių programų vykdymą. LAM/MPI komplektą sudaro API programų kūrimui, pranešimų mainų serveris, monitoringo ir konfigūracijos priemonės.

3.1.3. BalticGrid

Grid[5] tinklas - tai geografiškai paskirstytų kompiuterinių resursų visuma, apjungtų į virtualų superkompiuterį. Grid resursus dažniausiai sudaro klasteriai, priklausantys Grid tinklo partneriams, kurie skiria visus arba dalį klasterio mazgų Grid vartotojų uždaviniams skaičiuoti. Natūralu, kad tokiame tinkle ypač aktuali saugumo problema, kuri leistų lokalių resursų administratoriams pasitikėti Grid vartotojais bei jų grupėmis (virtualiomis organizacijomis) ir užtikrintų saugų vartotojų prisijungimą bei tinkamą resursų naudojimą.

Grid tinklo vartotojai neturi lokalių resursų prisijungimo vardų ir slaptažodžių, tačiau jie turi skaitmeninius sertifikatus, išduotus sertifikavimo tarnybos[6]. Sertifikatas autentifikuoja arba kitaip sakant identifikuoja vartotoją, patvirtindamas, kad sertifikato turėtojas tikrai yra tas žmogus, kuris yra užregistruotas vartotojų registre.

Vartotojų ir resursų autentifikavimui Grid'e naudojama viešo rakto infrastruktūra. Tai asimetrinio rakto šifravimo būdas, kai naudojama šifravimo raktų pora - viešas ir privatus raktai. Taigi, jei gautą informaciją galima iššifruoti turint konkretų viešąjį raktą, tai vienareikšmiškai galima tvirtinti, kad ji buvo užkoduota naudojant šios raktų poros privačiuoju raktu, kurį gali turėti tik konkretus vartotojas. Tokiu būdu yra identifikuojamas vartotojas.

Svarbu žinoti, kad sugeneruota raktų pora, automatiškai nesuteikia priėjimo prie Grid resursų. Jūsų turimą raktų porą turi pasirašyti sertifikavimo tarnyba, t.y. BalticGrid CA, patvirtindama jūsų asmenybę. Sertifikavimo tarnyba pasirašymo procedūra prilyginama sertifikato išdavimui, todėl tai atsakinga saugumą Grid tinke užtikrinanti procedūra. BalticGris CA išduotas skaitmeninis sertifikatas atitinka ITU-T X.509 standartą. Gavus sertifikatą, vartotojui turi būti leista naudotis tam tikru Grid resursų rinkiniu arba kitaip sakant, Grid vartotojas, o tiksliau jų grupė turi būti susieta su klasterių lokaliais vartotojais. Šis procesas vadinama autorizavimu.

Virtuali organizacija - tai vartotojų grupė, kuriai suteikta teisė leisti darbus tam tikrame Grid resursų rinkinyje t.y. tam tikruose Grid tinkle esančiuose klasteriuose. Pavyzdžiui viename moksliniame projekte dalyvaujantys vartotojai gali priklausyti vienai virtualiai organizacijai ir

[4] **Open Source** – programinė įranga su (atviru) išeities kodu
[5] **Grid** – geografiškai paskirstytų kompiuterinių resursų tinklas
[6] **Certificate Authority** - sertifikavimo tarnyba arba sutrumpintai CA

naudotis vienodais Grid resursais. Į virtualias organizacijas vartotojai gali būti grupuojami ir pagal skaičiavimų aplinkas. Kiekviena virtuali organizacija turi administratorių, kuris tvarko vartotojų sąrašą bei atlieka reikiamos programinės įrangos instaliavimą į lokalius klasterius, kontaktuoja su klasterių administratoriais dėl tinkamos skaičiavimo aplinkos sukūrimo Grid naudotojams. Kiekvieno klasterio administratorius nusprendžia ar leisti tam tikrai virtualiai organizacijai naudotis jo resursais, ar ne. [Lie06] [Bal05]

3.2. Kriptografijos bibliotekų naudojimas

Darbo pradžioje viena iš užduočių buvo realizuoti funkcijas koduojančias bei dekoduojančias tekstą pagal DES algoritmą su tolimesniu jų taikymu nagrinėjamos kriptosistemos analizei. Tačiau tolimesnė ir nuodugnesnė susijusios literatūros bei egzistuojančių atitinkamų algoritmų analizė, suteikė pagrindą egzistuojančių kriptografijos bibliotekų naudojimui. Po kriptografijos bibliotekų informacijos nagrinėjimo buvo priimtas sprendimas naudoti biblioteką Crypto++, kurios autorius Wen Dai didelį dėmesį skyrė algoritmų optimizavimui greičio atžvilgiu, naudojant prekompiliatoriaus direktyvas, konkrečios operacinės sistemos savybes (biblioteka adaptuota Unix/Linux, Windows ir MacOS operacinėms sistemos) bei kitas priemones pagreitinančias kriptografinius skaičiavimus. [Cry06]

Minėtoje bibliotekoje realizuotas DES kodavimo ir dekodavimo algoritmas praktikoje veikia greičiau už darbo autoriaus algoritmo realizaciją su tais pačiais įėjimo duomenimis: šifravimo raktas, kuoduojamas tekstas, ir su pagrindinėmis nagrinėjamomis kompiuterių architektūromis: PC ir Cluster. Atsižvelgiant į tai, kad darbo užduotis yra lygiagrečiųjų algoritmų taikymas kriptoanalizėje, o ne optimalaus kodavimo/dekodavimo algoritmo konkrečiai operacinei sistemai bei kompiuterio architektūrai rašymas, ir kadangi blokinio šifro bendru atveju efektyvaus išlygiagretinimo galimybės yra abejotinos dėl nuoseklaus kodavimo principo ir, palyginus, nedidelių keitimo matricų naudojimo, buvo nutarta naudoti Crypto++ bibliotekos siūlomą DES šifravimo algoritmą. Optimalesnis ir tuo pačiu greitesnis dekodavimo algoritmas sąlygoja mažesnę įtaką cryptoanalizės metodams. Ši savybė yra aktuali ir tiriamuoju atveju labai svarbi, kadangi blogiausiu DES kriptoanalizės atveju dešifravimo metodas yra kviečiamas 2^{64} kartų ir tuo pačiu turi didelę įtaką, lyginant su naujų raktų generavimu, kriptoanalizės metodų efektyvumui.

3.2.1. Biblioteka Crypto++

Informacijos šifravimui yra naudojamos funkcijos iš atviro kodo bibliotekos Crypto++ 5.4 versijos. Ši biblioteka yra „galinga“ realizacija daugelio populiariausių kodavimo algoritmų. Taigi, Crypto++ tai nemokama C++ kriptografinių schemų klasių biblioteka. API klasės

hierarchija apibrėžiama kitomis abstrakčiomis klasėmis AES: RC6, MAPC, Rijndael, Twofish, Serpen; kiti simetriniai blokiniai šifrai: IDEA, DES, Triple DES (DES-EDE2 ir DES-EDE3), DESX (DES-XEX3), RC2, RC5, Blowfish, Diamond2, TEA, SAFER, 3-WAY, GOST, SHARK, CAST-128; universalūs kvadratiniai šifro režimai: CBC, CBC, CTS,CFB, OFB, srautų šifrai: ARC4 ARC4, SEAL, WAKE, Sapphire, BlumBlumShub; šifravimas su viešu raktu: RSA, ElGamal, Nyberg-Rueppel (NR), BlumGoldwasser, Rabin, Rabin-Williams (RW), LUC, LUCELG, elipsinės kriptosistemos: PKCS#1 v2.0, OAEP, PSSR, IEEE P1363 EMSA2; hash-funkcijos: SHA-1, MD2, MD5, HAVAL, RIPEMD-160, pranešimų identifikavimo Tiger kodai: MD5-MAC, HMAC, XOR-MAC, CBC-MAC, DMAC; atsitiktinių skaičių pseudogeneratoriai (PRNG): ANSI X9.17, bei kiti mažiau svarbus arba populiarus kriptografijos algoritmai. Crypto++ palaiko šias platformas: Linux, Solaris, UNIX, Windows (priklausomai nuo versijos).

Atsižvelgiant į palaikomų šifravimo sistemų gausą, pačios bibliotekos populiarumą bei nuolatinį jos palaikymą, galima teigti, kad mus dominantys, vieni populiariausių kodavimo algoritmų, bus gerai ir tinkamai realizuoti. Kas, kartu su laisvu bibliotekos kodo naudojimu, leidžia pasinaudoti čia realizuotais kodavimo algoritmais, pagrindinį tyrimo dėmesį skiriant ne kodavimo algoritmų rašymui, bet efektyvios (lygiagretumo atžvilgiu) kriptosistemos kūrimui. [Cry06]

3.3. Vertinimo kriterijai ir metodai

Kadangi darbo užduotis - sumažinti kriptoanalizės vykdymo laiką ir tuo pačiu išskirti efektyviausią kodavimo rakto atsparumo analizės metodą bei algoritmą, sutampa su visų lygiagrečių skaičiavimų pagrindiniu tikslu - greitaveikos didinimu, buvo įvesti algoritmų ir metodų vertinimo kriterijai bei metodikos, leidusios nustatyti efektyvesnį lygiagrečios kriptoanalizės būdą. Po susijusios literatūros analizės ir kitų autorių panašios tematikos darbų nagrinėjimo buvo apibrėžta aibė kriterijų leidžiančių apibūdinti kriptoanalizės algoritmo bei skaičiavimų paskirstymo efektyvumus.

Pagrindinis lygiagrečiųjų skaičiavimų uždavinys ir tikslas yra sutrumpinti laiką, sugaištamą teisingam sprendiniui rasti, arba nustatyti, kad tokio sprandinio nėra. Todėl pagrindinis kriterijus, pagal kurį bus vertinami tiriami lygiagrečios kriptoanalizės algoritmai – laikas, - laikas nuo programos vykdymo pradžios su pradinių duomenų įvedimu iki vykdymo pabaigos, kai yra surandamas kodavimo/dekodavimo raktas.

Efektyvumas yra charakterizuojamas tokiais parametrais:

- Vykdymo laikas – maksimalus programos vykdymo kiekviename procesoriuje laikas;
- Naudojamumo koeficientas;

- Viršytas laikas – skirtumas tarp vykdymo laiko, padauginto iš procesorių skaičiaus, ir nuoseklaus vykdymo laiko.

Skaičiuojant algorimto išlygiagretinimo efektyvumo koeficientą, reikia rasti 2 laikus:

- Laiką, reikalingą programos vykdymui kompiuteryje su vienu procesoriumi;
- Suminį lygiagretaus vykdymo laiką, kuris apskaičiuojamas kaip vykdymo laiko ir procesorių skaičiaus sandauga.

Suminį lygiagretaus vykdymo laiką svarbu apskaičiuoti būtent tokiu būdu (o ne kaip vykdymo laikų visuose procesoriuose sumą), nes visi procesoriai yra pateikiami programai vykdymo pradžios momentu ir atlaisvinami jos pabaigos momentu. Tada nuoseklaus vykdymo laiko santykį su suminiu lygiagretaus vykdymo laiku atitinką naudojamumo koeficientas.

Tokio tipo vertinimo kriterijų pasirinkimą lėmė naudojimas, iš esmės, tarpusavyje labai skirtingų išlygiagretinimo metodų, algoritmų, aparatinių architektūrų bei jų apjungimo technologijų. Akivaizdu, kad kriptoanalizės, naudojant paskirstytus, išlygiagretintus skaičiavimus, efektyvumo kriterijai ir jų nustatymo būdas remiasi nuosekliaisiais kriptoanalizės algoritmais, vykdomais ant vieno personalinio kompiuterio, kurių efektyvumas paimtas kaip pagrindas, kaip atskaitos taškas, vertinant tyrinėjamus efektyvesnius kriptoanalizės algoritmų išlygiagretinimo būdus. Tai yra, kiekvieno nagrinėjamo algoritmo ir skaičiavimų paskirstymo metodo veikimas yra lyginamas su nuoseklaus algoritmo veikimu.

Reikėtų paminėti, kad išvardinti vertinimo kriterijai ir metodai nėra tobuli, bet jie kuo geriausiai tinka lygiagrečiųjų algoritmų efektyvumui nustatyti. Tačiau, vertinant efektyvumą, programų, vykdomų BalticGrid aplinkoje, tenka konstatuoti, kad praktiškai nustatyti, ant kiek algoritmas ir skaičiavimai efektyvūs, yra nelengvas darbas, o kartais to neįmanoma padaryti, nes BalticGrid nėra homogeninė struktūra, jos struktūros nėra tobulos, nes tai sąlyginai naują, bet sparčiai besivystanti ir perspektyvi technologija. Tai reiškia, kad pasirinkti kriterijai tinka vertinti BalticGrid programas tik su sąlyga, kad jos vykdomos viename mazge ir atitinkamos architektūros kaip ir programos, naudojančios MPI realizaciją. [Lev04] [SS02]

3.4. Problemos ir nesklandumai

Yra prasminga paminėti visas problemas ir nesklandumus, atsirandančius vykdant užduočių realizavimą ir siekiant užsibrėžto tikslo. Kadangi yra naudojamos palyginus naujos metodikos ir technologijos, kurios yra dar besivystymo stadijoje ir dalis jų funkcionalumo yra eksperimentinio pobūdžio, t.y. jos nėra pilnai ištestuotos ir visa jų naudojimo atsakomybė tenka pačiam vartotojui, praktiškai, neįmanoma išvengti, nors ir mažiausių, problemų.

Šiame skyriuje, visų pirma, bus minimos problemos susijusios su BalticGrid naudojimu, nes būtent šis tinklas yra mažiausiai išplėtota technologija lyginant su klasteriais ir MPI

bibliotekomis, kurie irgi naudojami kriptoanalizės uždavinių išlygiagretinimui. Pagrindinė problema, trukdanti plačiai naudoti BalticGrid resusrsus, yra atitinkamos dokumentacijos ir veikiančių sudėtingesnių pavyzdžių trukumas. Čia reiktų pabrėžti, kad yra daug literatūros, aprašančios Grid technologiją, tame tarpe ir BalticGrit bei LitGrid, tačiau mažai, pilnai aprašančios visas jos galimybes, visus reikalavimus bei nurodančios kaip pasiekti didesnį efektyvumą su turimais resursais. Kaip paprasčiausia pavyzdį galima pateikti, BlticGrid programos vykdymą su tipu „Collection“, kai jis yra vykdomas tik su glite-wms-job-submit komandą, nors paprastai programos vykdymui yra naudojama glite-job-submit.

Nemalonus yra tas faktas, kad užduoties nusiuntimas į BalticGrid mazgą negarantuoja jos greito startavimo, atvirkščiai, programa bus patalpinta į eilę su statusu „wait“ ir lauks savo vykdymo. Po „wait“ statuso seka „sheduled“ statusas. Tokiu būdu iki programos vykdymo, „runnig“ statusas, paprastai praeina daug laiko, kuris atskirais atvejais matuojamas valandomis.

Opi problema yra duomenų siuntimo greitis ir rezultato grąžinimas. Nereikėtų pamiršti to fakto, kad Grid yra geografiškai nutolusių paskirstytų kompiuterinių resursų tinklas, todėl dėl galimų techninių nesklandumų rezultato galima nesulaukti. Šis faktorius taipogi trukdo paskirstytų skaičiavimų algoritmų efektyvumo vertinimui. Dėl vidinės tinklo struktūros dažnas yra variantas, kad rezultatas yra pasiekiamas tik po tam tikro laiko po programos veikimo pabaigos. Ši savybė blogina bendra BalticGrid efektyvumą, o atskirais atvejais rezultato sugrąžinimas trunka ilgiau negu jo paieška programoje. Bet, atsižvelgiant į tą situaciją, kad vidutiniu atveju visgi rakto radimas užtrunka žymiai ilgiau negu rezultato grąžinimas ir kad galima pasiekti trumpesnio bendro programos veikimo BalticGrid aplinkoje laiko negu naudojant kitus išlygiagretinimo metodus, daro šia technologiją patrauklią tyrinėjimui, kadangi pagrindinis tikslas yra surasti kriptografijos raktą per trumpesnį laikotarpį.

Atliekant eksperimentus MIF Linux PST klasteryje, būtina turėti omenyje, kad visiems prieinamas katalogas, kuris klasterio naudojimo metu yra padaromas namų katalogų, yra trinamas kiekvienos dienos 6:20 h ryto. Tai yra programos, kurios rezultatas grąžinamas į failą, esantį minėtame kataloge, pranešimo apie rastą kriptografijos raktą arba klaidos pranešimo galima nesulaukti.

Taip pat reiktų paminėti faktą, lietinantį Grid resursų naudojimo pradžią. Kaip ir daugelyje kitų stambių organizacijų, negalima išvengti „biurokratinių“ problemų susijusių su sertifikatų gavimu ir registravimusi virtualiose organizacijos. Tai yra, minėti veiksmai reikalavo kažkiek laiko vykdymui (iki kelių dienų), nors tam vėl gi yra priežastis – saugumas.

3.5. Efektyvumo analizė

Svarbus uždavinys yra identifikavimas priežasčių, turinčių įtakos lygiagretaus algoritmo efektyvumui, naudojant eksperimentinius duomenis. Siūlomas standartinis efektyvumo vertinimo metodas – profiliavimas. Efektyvumas gali būti charakterizuojamas programos vykdymo laiku, naudojamumo koeficientu, pagreičiu. Pabrėžtina, kad efektyvumas priklauso nuo pasirinkto užduoties sprendimo metodo, šio metodo realizavimo, naudojamos aparatūros bei jos konfiguravimo. MPI standarto bibliotekoms naudojama profiliavimo biblioteka MPE (Multi-Processing Envirounment). [Mug04]

3.5.1. Efektyvumas

Kadangi visų lygiagrečių skaičiavimų pagrindinis tikslas yra greitaveikos didinimas, tai tobulinimo procesas turi būti irgi nukreiptas efektyvumo didinimo linkme. Profiliavimas leidžia pateikti informaciją apie perėjimus tarp būsenų patogiu pavidalu. Būsena apibūdinama dviem parametrais: perėjimas į nagrinėjama būseną ir išėjimas iš jos.

Šio uždavinio sprendimas MPE – visų įvykių sekos išsaugojimas faile, o vėliau šios informacijos pateikimas norimu pavidalu. Biblioteka funkcionuoja su viena sąlyga, yra daroma prielaida, kad paskirstytos sistemos mazguose esantys laikrodžiai yra sinchronizuoti. [Mug04]

Atlikti bandymai leido pamatyti nelabai efektyvų aparatinių resursų naudojimą. Galima išskirti tris nepriklausomus prarasto laiko komponentus:

- Sąnaudos dėl lygiagretumo trukumo, priveda prie skaičiavimų dubliavimo keliuose procesoriuose;
- Sąnaudos dėl tarp procesorinių mainų;
- Sąnaudos dėl procesorių, kur vykdymas pasibaigė anksčiau negu kituose, prastovų.

Procesorių apkrovos išbalansavimas skirtinguose lygiagrečios programos vykdymo etapuose, gali padidinti laiko sąnaudas, reikalingas tų procesorių sinchronizavimui. Vis gi, skaičiavimo sistemos efektyvumas labiausia priklauso nuo naudojamų procesorių našumo, nes laikas sunaudotas skaičiavimams vyrauja lyginant su siuntimo greičiu. [Mug04]

3.6. Eksperimentai

Eksperimentai buvo atliekami MIF Linux PST klasteryje ir BalticGrid aplinkoje. Skirtingų simetrinio kodo išlygiagretinimo metodikų analizė yra atliekama, vykdant DES šifro dekodavimo programą. Programos veikimo principas buvo aptartas darbo antroje dalyje. Šiuo atveju programos kodas buvo pritaikytas vykdymui paskirstytų skaičiavimų kompiuteriniuose tinkluose: Linux PST klasteryje ir BalticGrid tinkle. Taip pat buvo panaudotos C++ atviro kodo bibliotekos Crypto++ teksto šifravimo ir iššifravimo DES algoritmu funkcijos.

Kadangi, išanalizavus DES algoritmo atsparumą nulaužimams, buvo konstatuota, jog nėra, praktiškai, jokio efektyvesnio kriptoanalizės bendru atveju būdo, negu pilnas galimų raktų intervalo perrinkimas, darbe yra atliekamas skirtingų algoritmų išlygiagretinimo ir skaičiavimų paskirstymo metodų, bei priklausomybės nuo aparatinės dalies architektūros tyrimas. Siekiant išsiaiškinti, kas labiausiai įtakoja lygiagrečios kriptoanalizės efektyvų dabą.

Buvo skaičiuojamas vidutinis vienos metodo iteracijos laikas. Paskui buvo atrenkami vidutinis, mažiausias ir didžiausias bandymo laikai. Labiau informatyvus tokioms charakteristikoms yra mažiausias laikas, kadangi laikas visada apribuotas iš apačios aparatūros parametrais. Iš kitos pusės, labiau tikėtinas užduoties vykdymo laikas yra vidutinis. Lygiagrečiųjų algoritmų vykdymo laikui nustatyti yra naudojamos standartinės MPI priemonės – funkcija `MPI_Wtime();`

Iš tikrųjų, programos vykdymo laikas kiekvienu atveju yra atsitiktinis didis. Tą sąlygoja tai, kad lygiagrečių programų procesai yra leidžiami kartu su kitomis programomis ir naudoja bendrus aparatinius resursus. Lygiagrečių skaičiavimų sistemų našumas, pirmiausiai, priklauso nuo mazgų našumo ir jų skaičiaus, kuo jis didesnis, tuo mažesnis programos vykdymo laikas. Tinklo sandara taipogi įtakoja skaičiavimo laiko priklausomybę nuo procesorių skaičiaus, nors DES šifro kriptoanalizė, kuriai buvo vykdomi bandymai, turi mažą duomenų perdavimo operacijų įtaką.

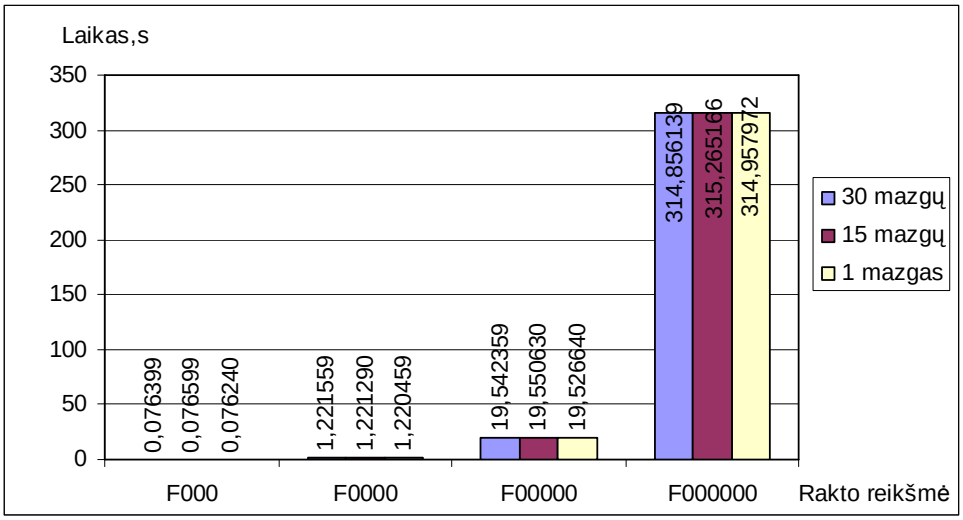
3.6.1. Užduočių vykdymas klasteryje

Vykdant lygiagrečias programas MIF Linux PST klasteryje, būtina susipažinti su šio klasterio specifika, technine ir programine įranga. Minėtu klasteriu gali naudotis tik registruoti VU MIF kompiuterių tinklo naudotojai, kurie privalo susikurti laikiną katalogą, pasiekiamą visuose mazguose, kuriuose bus vykdoma programa, ir padaryti jį namų direktorija. Šis laikinai sukurtas katalogas kasryt 06:20 yra ištrinamas. Tai yra vykdant programą būtina atsižvelgti į tą faktą, kad, jeigu naudojamas rezultatų ir klaidų išvedimas į failą, duomenys gali būti prarasti kai programos vykdymas užtrunka pakankamai ilgai.

Lygiagretaus DES šifro kriptonalizės algoritmo bandymai su Linux PST buvo atliekami skirtingam mazgų skaičiui. Maksimalus klasterio mazgų skaičius - 30, leidžia nustatyti kriptoanalizės algoritmo veikimo greitį esant didžiausiam našumui. Atliekant eksperimentą su penkiolika mazgų, galima padaryti prielaidą, kad programa veiks dvigubai ilgiau. Taip pat buvo atliktas bandymas, vykdant DES kriptoanalize su vienu mazgu ir ant to paties klasterio mazgo vykdant nuoseklų DES dešifravimo algoritmą. Paskutiniai bandymas neturi didelės praktinės vertės, tačiau yra informatyvus MPI sąsajos efektyvumo prasme. Tai yra leidžia įvertinti, kaip

pasikeičia programos veikimo laikas naudojant lygiagretaus programavimo technologijas, bet kai nėra duomenų tarp mazgų apsikeitimo.

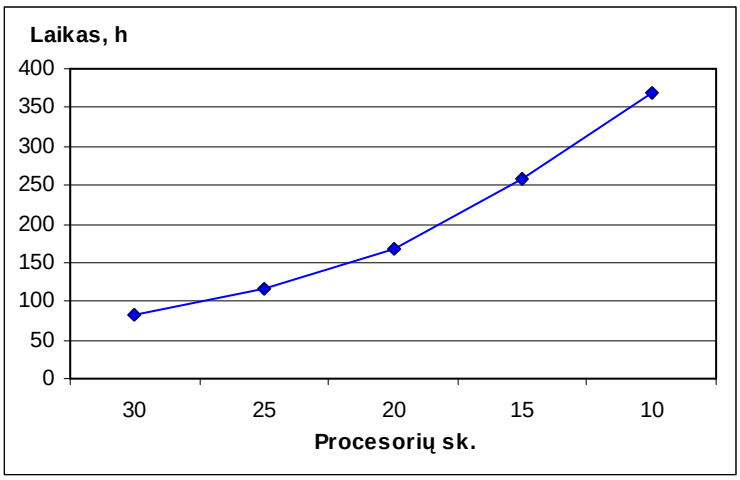
Žemiau pateiktame grafike matyti, kaip priklauso programos vykdymo laikas nuo naudojamų klasterio mazgų skaičiaus ir šifravimo rakto reikšmės. Vaizdavimo paprastumo dėlei, rakto reikšmė tolygiai didinama, siekiant ištirti, kaip kis kriptoanalizės programos vykdymo laikas.



Pav.3 Kriptoanalizės laiko priklausomybė nuo rakto reikšmės

Iš grafiko matyti, kad laiko, reikalingo programos vykdymui, sąnaudos auga proporcingai šifravimo rakto reikšmės augimui, t.y. maždaug F kartų šešioliktainėje skaičiavimų sistemoje. Iš čia išplaukia, kad šifravimo raktų diapazono perrinkimas, naudojant lygiagrečiuosius algoritmus ir klasterio aparatinę architektūrą, iki tam tikros nagrinėjamo rakto reikšmės užtrunka tiek pat laiko kaip ir nuoseklaus algoritmo vykdomo ant vieno iš klasterio mazgų. Nesunku apskaičiuoti, kad šį rakto reikšmė priklauso nuo procesorių skaičiaus ir yra lygi maždaug k ~ U/n, kur U – rakto reikšmių diapazonas, o n – procesorių skaičius. Pranašumas yra tame, kad visi klasterio procesoriai tokį rakto reikšmių diapazoną apdoroja vienu metu, kiekvienas savo reikšmių intervale. Reikėtų pabrėžti, jog šis eksperimentas yra tiriamojo pobūdžio ir iliustruoja tiesioginę lygiagrečiųjų algoritmų vykdymo efektyvumo priklausomybę nuo aparatinės dalies ir konkrečiai klasterio mazgų skaičiaus bei jų našumo. Praktikoje rakto reikšmių diapazonas dažniausiai būna žymiai siauresnis negu atliktame bandyme, kadangi paprastai raktą sudaro tik raidžių, skaitmenų ir kai kurių specialių simbolių aibė.

Atsiktinai sugeneruoto DES šifro rakto parinkimo programos, pagrįstos lygiagrečiųjų algoritmų naudojimu, veikimo laiko priklausomybės nuo procesorių skaičiaus grafikas įrodo, kad negalimas teiginys, jog lygiagretus algoritmas tiek kartų greitesnis, kiek procesorių vienu metu jį vykdo. Kiekvienu bandymu rakto reikšmė nesikeitė.



Pav.4 Kriptoanalizės laiko priklausomybė nuo procesorių skaičiaus

Iš grafiko matyti, kad laikas kinta netolygiai ir linija, jungianti rezultatų taškus, yra kreivė, kas nusako tą faktorių, kad didesnis procesorių skaičius reikalauja daugiau tarpusavio pranešimų mainų, o tai sulėtina visos sistemos ir, konkrečiai, lygiagretaus DES kriptoanalizės algoritmo darbą.

Galima padaryti išvadą, kad klasterio architektūros ir MPI sąsajos tandemas, yra pakankamai gera priemonė lygiagrečiosios DES kriptoanalizės vykdymui. Praktikoje, santykinai, didelis paskirstytų skaičiavimų tinklas – klasteris, leidžia efektyviai, su tenkinančiomis laiko sąnaudomis, „nulaužti“ DES šifro raktą, tačiau su sąlyga, kad rakto reikšmių diapazonas nėra didelis ir yra bent apitiksliai žinomas prieš programos veikimo pradžią. Tai yra bendruoju atveju ir rimtuose moksliniuose simetrinių šifrų kriptoanalizės tyrimuose, tik vieno klasterio naudojimo pakankamumas yra abejotinas, dėl jo aparatinių galimybių ribotumo.

3.6.2. Užduočių vykdymas BalticGrid aplinkoje

Grid – geografiški nutolusių kompiuterinių resursų tinklas. Paprastai Grid tinklo mazgus sudaro didelio našumo klasteriai, neretai turintis po kelis šimtus savo mazgų. BalticGrid technologija siūlo dar vieną skaičiavimų paskirstymo galimybę. Aplinkos JDL užduoties faile galima paskirstyti užduoties vykdymą tarp kelių skirtingų tinklo mazgų. [Priedas Nr 3] nurodytas paprastas būdas kaip paleisti nuoseklų algoritmo vykdymą BalticGrid mazge, kuris bus automatiškai parinktas. Tačiau BalticGrid užduočių faile yra kelios galimybės nurodyti, kad tas pats nuoseklus algoritmas turi būti vykdomas iškart keliuose tinklo mazguose su skirtingais parametrais. Paprasčiausias būdas yra naudoti atributą: `JobType = "Parametric"`, apibrėžti parametrų kitimo rėžius i žingsnį bei nurodyti, kad vykdomoji programa turi startuoti su jai paduodamais parametrais. Kai yra pasirenkamas `Job="DAG"` arba `Job="Collection"`, yra galimybė kiekvienam atskirai naudojamam BalticGrid tinklo mazgui nurodyti kokią programą su kokiais parametrais turi būti vykdoma. Visi minėti skaičiavimų paskirstymo variantai gali būti

traktuojami, kaip lygiagrečios užduoties vykdymo klasteryje, naudojant MPI sąsają, analogas. Tačiau toks išlygiagretinimo būdas, DES kriptosistemos rakto atsparumo analizėje, atsižvelgiant į tikslą parinkti efektyvesnį lygiagrečios kriptoanalizės algoritmą, nėra tinkamas dėl kelių priežasčių:

- Statinis parametrų nurodymas (išskyrus `JobType = "Parametric"` atvejį);
- Neišnaudojamos pilnai Grid tinklo mazgų galimybės;
- Didelės laiko sąnaudos dėl duomenų mainų ir sistemos valdymo mechanizmo.

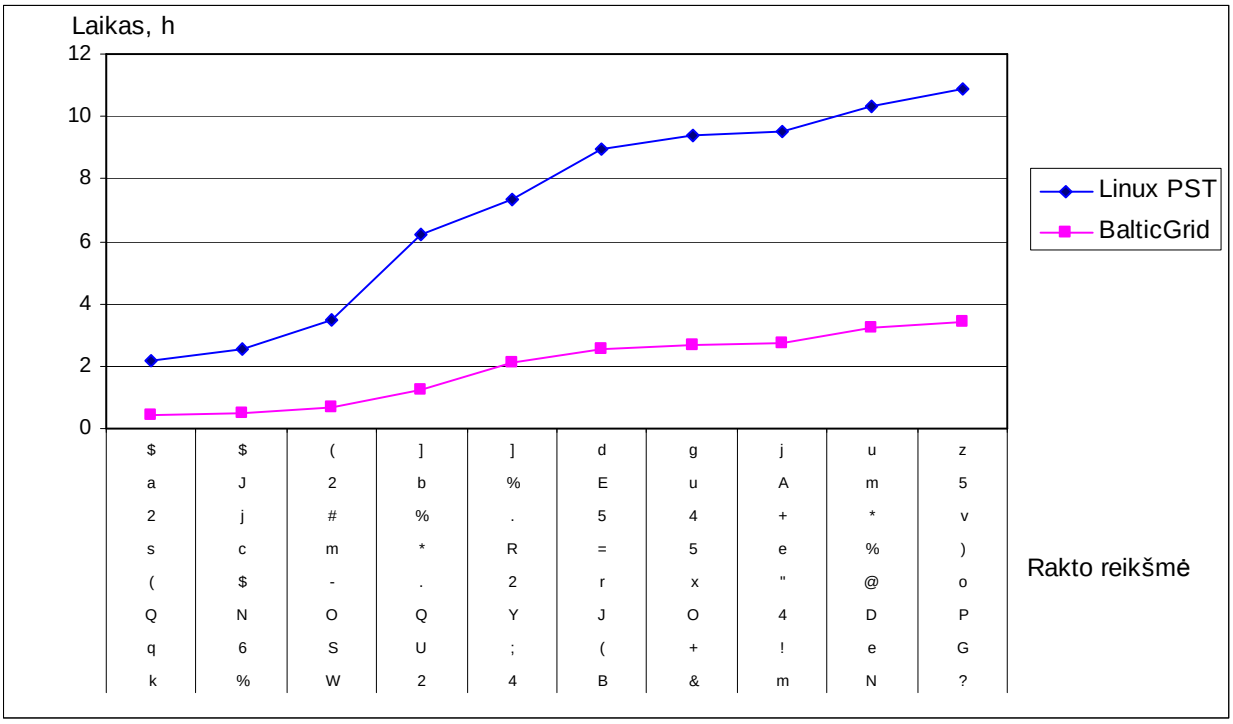
BalticGrid yra realizuota galimybė į tinklo mazgą perduoti vykdyti lygiagretų algoritmą, naudojanti MPI technologiją. `JobType="MPICH"` atributas JDL užduoties faile nurodo, kad bus vykdoma lygiagreti programa, kuriai reikalinga MPI programinė įranga tinklo mazguose. Kaip nurodyti, kad BalticGrid parinktų tik tinkamus mazgus yra nurodyta [Priedas Nr 4]. gLite[7] tik rezervuoja klasterio mazgus užduoties vykdymui, o šių mazgų valdymas paliekamas vartotojui. Iš esmės šis metodas atitinka lygiagretaus algoritmo vykdymą paprastame klasteryj su MPI programine įranga, tik su didesnėmis laiko sąnaudomis duomenų apsikeitimui netik tarp klasterio mazgų, bet ir BalticGrid tinklo mazgų. Tačiau Grid tinklas suteikia galimybę pasirinkti našesnį (klasterio mazgų kokybės ir kiekybės prasme) mazgą – klasterį iš gana plataus sąrašo. DES kriptoanalizės naudojant lygiagrečiuosius algoritmus, pagrįstus MPI naudojimu, vykdymo laiko priklausomybė nuo vienu metu paleistų procesų skaičiaus viename BalticGrid mazge. Skirti šiam eksperimentui daugiau laiko nėra prasminga, nes uždavinys susiveda iki MPI ir klasterio technologijų bendro naudojimo, kuris jau buvo išnagrinėtas aukščiau.

Galima padaryti prielaida, kad šio darbo tikslo rėmuose įdomiausias tyrimo objektas būtų bendras BalticGrid skaičiavimų paskirstymo ir MPI algoritmų išlygiagretinimo technologijų naudojimas. Turima omenyje, kad naudojant jau minėtus Grid tinklo užduočių tarp mazgų paskirstymo priemones: `Job="DAG"` ir `Job="Collection"`, galima kiekvienam pasirinktam mazgui pavesti vykdyti MPI lygiagretų algoritmą. Kas rezultate garantuoja šifravimo rakto reikšmių perrinkimo diapazono paskirstymo padalinimą tarp `n*m` procesorių, kur n – BalticGrid mazgų (klasterių) skaičius, m – vidutinis pasirinktų tinklo mazgų procesorių skaičius.

Tačiau šio tyrimo vertinimas nėra trivialus. BalticGrid sudaro skirtingi savo struktūra, organizacija ir našumu klasteriai. Todėl vienareikšmiškai gali būti nusakytas tik vienas DES kriptoanalizės efektyvumo vertinimo kriterijus, t.y. laikas, kuris reikalingas kriptoanalizės užduočiai atlikti. Visi kiti algoritmo išlygiagretinimo efektyvumo kriterijai negali būti taikomi dėl to, kad visi Grid tinklo klasteriai nėra homogeniniai, neįmanoma pilnai valdyti ir kontroluoti užduočių vykdymo atskirame tinklo mazge (ši problema plačiau aprašyta 3.4. punkte) bei skirtingo mazgų apkrovimo.

[7] **gLite** – vartotojo sąsaja Grid aplinkoje

Atliktas eksperimentas, kurio esmė ištirti BalticGrid ir MPI technologijų apjungimo, DES kriptoanalizės tikslams pasiekti, pranašumą lyginant su lygiagrečiais šifravimo rakto reikšmių diapazono perrinkimo algoritmais adaptuotais Linux PST klasteriui. Bandymų užduotis yra suemulioti vidutinį DES kriptoanalizės atvejį, generuojant atsitiktinius šifravimo raktus, ir ištirti BalticGrid ir Linux PST šio rakto nulaužimo galimybes. Eksperimento tikslais raktas bus generuojamas iš sąlyginai siauro reikšmių intervalo, artimo dažniausiai naudojamam praktikoje, t.y. raktą sudaro 8 simboliai, kurių ASCII lentelės reikšmės yra intervale [20 - 7A] šešioliktainėje skaičiavimų sistemoje.



Pav.5 Kriptoanalizės Linux PST ir BalticGrid aplinkuose laiko priklausomybė nuo rakto reikšmės

Bandymas buvo atliekamas su MIF Linux PST klasteriu, naudojant 30 mazgų ir BalticGrid aplinkos penkiais klasteriais, kurių procesorių bendras kiekis yra 250, tai yra JDL užduoties faile nurodytas parametras NodeNumber = 50. Matyti, kad tiriamu atveju bendras DES kriptoanalizės vykdymo laikas BalticGrid aplinkoje yra mažesnis. Tačiau, taip pat pastebima, kad Grid technologijos naudingumo koeficentas mažėja, didėjant kodavimo rakto reikšmėm. Tą paaiškina šio tinklo architektūra, t.y. augant rakto reikšmėm netik padidėja pranešimų apsikeitimo tarp mazgų srautas, kas yra būdinga klasterių technologijai, bet ir padidėja proceso užlaikymo mazgo vykdymo eilėje tikimybė. Būtent ši Grid technologijos savybė sumažina kriptoanalizės, naudojančios lygiagrečių skaičiavimų paskirstymą užduoties faile tarp skirtingų BalticGrid mazgų, santykinį efektyvumą.

3.7. Tyrimo rezultatas

Rezultate galima konstatuoti, kad lygiagretieji algoritmai ir skaičiavimų paskirstymas, taisyklingai juos taikant bei atsižvelgiant į poreikius konkrečioje situacijoje, gali žymiai supaprastinti kriptoanalizės užduotis ir pasiūlyti efektyvesnį kriptografijos algoritmų analizės metodą. Kaip matyti iš atliktų eksperimentų, išlygiagretinti kriptoanalizės algoritmai bendruoju atveju yra kelis kartus (priklausomai nuo procesorių skaičiaus) greitesni už nuosekliuosius. Tačiau neatsargus naudojimas, neįvertinus kodavimo rakto reikšmių diapazono kitimą, gali sąlygoti panašius lygiagrečių ir nuoseklių kriptoanalizės metodų rezultatus.

Buvo įrodyta, kad bendruoju atveju simetrinės kriptoanalizės uždaviniuose yra priimtinesnis kombinuotas Grid ir MPI technologijų naudojimas. Tai yra JDL užduočių failo pagalba paskirstyti šifro rakto reikšmių rėžius tarp atskyrų procesų, kurie naudodami MPI bibliotekos pranešimų technologiją bus lygiagrečiai vykdomi kiekviename atskiro klasterio mazge. Kadangi BalticGrid tinklą sudaro skirtingi, kokybine ir kiekybine prasme, klasteriai, prieš paskirstant rakto reikšmių diapazoną tinklo mazgams, būtina atlikti išankstine naudotinų Grid aplinkos klasterių analizę, siekiant įvertinti jų našumą, ir kriptoanalizės ieškomo rakto reikšmių rėžius nustatyti kiekvienam mazgui proporcingai jo pajėgumui kitų atžvilgiu.

Išvados ir rekomendacijos

Iš ankstesnių testavimo rezultatų yra aiškiai matomas taisyklingai naudojamų lygiagrečiųjų algoritmų pranašumas prieš nuosekliuosius, netgi esant žemam išlygiagretinimo lygiui. Taip pat galima teigti, kad lygiagrečių DES kriptoanalizės algoritmų efektyvumas, didele dalimi priklauso nuo ieškomo rakto perrinkimo reikšmių rėžių. Kuo daugiau tenka perrinkti raktų, tuo labiau išryškėja lygiagretaus algoritmo sprendimo efektyvumas prieš nuoseklų algoritmą.

Šiais laikais sparčiai besivystant Grid tinklų technologijoms, pakankamai greitu laiku, kažkada laikyti nenulaužomi, kriptografijos šifrai taps „lengvai įkandamu riešutu“ visiems, kas turi galimybę naudotis besiplečiančiomis lygiagrečiųjų algoritmų ir paskirstytų skaičiavimų tinklų galimybėmis. Paskirstytų skaičiavimų technologijos atsiradimas ir spartus vystymasis verčia kriptografijos ekspertus ieškoti naujus, efektyvesnius duomenų kodavimo būdus.

Bandymai parodė, kad kompleksinis naudojimas BalticGrid paskirstytų skaičiavimų ir MPI bibliotekos teikiamų lygiagretaus programavimo principų įgalina pasiekti aukštą kriptoanalizės lygį, nepasiekiamą atskiriems klasteriams ir tuo labiau paprastiems kompiuteriams. Tačiau šiuo atveju yra būtina išankstinė Grid tinklo analizė, siekiant išskirti mazgus, pasižyminčius didesniu našumu, bei paskirstyti DES kriptografijos rakto reikšmių diapazoną tarp pasirinktų tinklo mazgų atsižvelgiant į jų pajėgumą.

Tai pat galima pasiekti didesnio lygiagrečios DES kriptoanalizės užduoties efektyvumo kiekviename BalticGrid mazge – klasteryje. Iš gautų eksperimentų rezultatų, galima teigti, kad klasterio procesorių apkrovos perskirstymas gali sąlygoti efektyvesnį skaičiavimo resursų panaudojimą, dėl netiesioginio kreipimosi į duomenų erdvę eiliškumo reglamentavimo. Tuo atveju, jeigu pavyktų suderinti duomenų mainus su skaičiavimais, galima išgauti programų efektyvumo augimą, sąlygojamą tuo, kad bus perskirstytas apkrovimas atsižvelgiant į mainų operacijų realizavimą. Bandymai leidžia analizuoti lygiagrečių kriptoanalizės algoritmų vykdymą MPI pranešimų siuntimo lygmeniu.

Literatūros sąrašas

[MM05] Н. Молдовян, А. Молдовян. Введение в криптосистемы с открытым ключом. БХВ-Петербург, Санкт-Петербург, 2005, 288 с.

[Vin98] А. Винокуров. Задачи, решаемые криптографическими методами. Шифрование и шифры. URL:http://www.enlight.ru/ib/tech/crypto/part2.htm

[Sta05] Vilius Stakėnas. Šifrų istorijos. TEV, Vilnius, 2005

[Sta02] Vilius Stakėnas. Kodavimo teorija Paskaitų kursas. URL:http://www.mif.vu.lt/matinf/asm/vs/pask/cdth/cdth.htm, Москва, 2002, 238 Kb

[Zel99] Владимир Жельников. Криптография от папируса до компьютера. 2002

[MOV01] A. Menezes, P. Van Oorschot S. Vanstone. Handbook of Applied Cryptography. CRC Press, 1996, 816 pages

[SS02] Шпаковский Г.И., Серикова Н.В. Программирование для многопроцессорных систем в стандарте MPI. URL:http://www.cluster.bsu.by/download/book_PDF.pdf , БГУ, 2002. 323 с.

[Mug04] С. Муглярчик. Разработка параллельных алгоритмов обучения нейронных сетей. БГУ, Минск, 2004

[Lev04] Вадим Левченко. Асинхронные параллельные алгоритмы как способ достижения 100% эффективности вычислений. Институт прикладной математики им. М.В.Келдыша РАН, Москва, 2004.

[Lie06] Lietuvos akademinių institucijų lygiagrečiųjų ir paskirstytų skaičiavimų tinklas. URL:http://www.litgrid.lt/naudotojams/, 2006

[Bal05] BalticGrid Web Site. URL:http://www.balticgrid.org/, 2005,150Kb

[Con04] “Baltic Grid” Conference. URL:http://www.mif.vu.lt/grid/, 10Kb

[Pst05] VU MIF paskirstytų skaičiavimų tinklas. URL:http://kedras.mif.vu.lt/cluster/, 40Kb

[Cry06] Crypto++® Library 5.4. URL:http://www.cryptopp.com/, 2006, 30Kb

[Bsu04] Основные классы современных параллельных компьютеров. URL:http://www.cluster.bsu.by/classes.htm 30 Kb

[MPI95] MPI: A Message-Passing Interface Standard. URL: http://www.cluster.bsu.by/MPI-1_angl.pdf, 1995, 1,2 Mb

[MPI97] MPI-2: Extensions to the Message-Passing Interface. URL: http://www.cluster.bsu.by/MPI-2_angl.PDF, 1997, 1,9 Mb

[LAM05] LAM/MPI Parallel Computing**.** http://www.lam-mpi.org/, 2005, 100Kb

Priedas Nr. 1

Nuoseklus rakto perrinkimo algoritmas

```
int main(int argc, char *argv[])
{
   int i,j,err=0;
   unsigned char in[8],out[8],outin[8];    /* 64 bitų pranešimai  */
   unsigned long key_data[2];              /* 64 bitų raktas         */
   unsigned long i1=0,i2=0;
   unsigned long imin[2]={0,0};            /* apatinė rakto riba  */
   unsigned long imax[2]={0xffff,0xffff};  /* viršutinė rakto riba   */
   DESshifr(in,out);                              /* šifravimo procedūra */
   i1=0;i2=0;
   for (i1=imin[0]; i1<imax[0]; i1++)
   {
      key_data[i][0]=i1;
      for (i2=imin[1]; i2<imax[1]; i2++)
      {
         key_data[i][1]=i2;
         /* dešifravimo procedūra su raktu key_data */
         DESdeshifr(out,outin,key_data);
         if (memcmp(in,outin,8) == 0)
         {
            printf("raktas: 0x%x 0x%x \n",i1,i2); /* raktas rastas */
            return 0;
         }
      }
   }
   return 0;
}
```

Priedas Nr. 2

Lygiagretus rakto perrinkimo algoritmas

```
int main(int argc, char *argv[])
{
   int i,numprocs,myid,PR;
   unsigned long key_data[2];              /* 64 bitų raktas         */
   unsigned char in[8],out[8],outin[8];    /* 64 bitų pranešimai  */
   unsigned long i1=0,i2=0;
   int h,n1,n2,ost,num,*buf;
   unsigned long imin[2]={0,0};               /* apatinė rakto riba  */
   unsigned long imax[2]={0xffff,0xffff};  /* viršutinė rakto riba*/
```

```
double t1,t2;
MPI_Init(&argc,&argv);
MPI_Comm_size(MPI_COMM_WORLD,&numprocs);
MPI_Comm_rank(MPI_COMM_WORLD,&myid);
DESshifr(in,out);                          /* šifravimo procedūra */
/* rakto reikšmių diapazono nustatymas kiekvienam procesui */
h=(imax[0]-imin[0]+1)/numprocs;
ost=(imax[0]-imin[0]+1)%numprocs;
n1=imin[0]+myid*h;
n2=n1+h;
if ((ost!=0)&&(myid==numprocs-1))
   n2=n2+ost;
num=0;
buf= (int *)malloc(numprocs*sizeof(int));
for (i=0; i<numprocs; i++) buf[i]=0;
t1 = MPI_Wtime();
for (i1=n1; i1<n2; i1++)
{
   key_data[0]=i1;
   for (i2=imin[1]; i2<imax[1]; i2++)
   {
      key_data[1]=i2;
      /* dešifravimo procedūra su raktu key_data */
      DESdeshifr(out,outin,key_data);
      if (memcmp(in,outin,8) == 0)          /* raktas surastas */
      {
         t2 = MPI_Wtime();
         printf("raktas: 0x%x 0x%x \n",i1,i2);
         printf("procesas %d %d \n",myid,i1);
         printf("laikas %f \n",t2-t1);
         num=0xFF;
      }
   }
   /* procesų užklausimas, ar rastas raktas */
   MPI_Allgather( &num,1,MPI_INT,buf,1,MPI_INT,MPI_COMM_WORLD);
   for (i=0; i<numprocs; i++)
      if (buf[i]==0xFF)
      {
         printf("procesas %d sustabdytas \n",myid);
         MPI_Finalize();                        /* procesų darbo nutraukimas
      */
         return 0;
         }
```

```
   }
   if (num==0) printf(" nerastas raktas procese %d \n",myid);
   num=myid;                                        /* procesas, baigęs
perrinkimą */
   do
      { /* procesų užklausimas, ar rastas raktas */
      MPI_Allgather(&num,1,MPI_INT,buf,1,MPI_INT,MPI_COMM_WORLD);
      for (i=0; i<numprocs; i++)
         if (buf[i]==0xFF)
         {
            printf("procesas %d sustabdytas \n",myid);
            MPI_Finalize();                     /* procesų darbo nutraukimas */
            return 0;
         }
      PR=0;
      for (i=0; i<numprocs; i++)
         if (buf[i]!=i) { PR=0xFF; break; }
   }while (PR!=0);
   /* visi procesai baigė darbą, raktas nerastas */
   MPI_Finalize();
   return 0;
}
```

Priedas Nr. 3

Paprasto JDL užduočių failo pavyzdys.

```
Executable = "/bin/sh";
Arguments = "des.sh";
StdOutput = "stdout.log";
StdError = "stderr.log";
InputSandbox = {"des.sh", "crypt.c", "des.h"};
OutputSandbox = {"stdout.log", "stderr.log"};
```

Priedas Nr. 4

JDL užduočių failo pavyzdys. Atributo „`Requirements`“ reikšmė – naudoti tik tinklo magus, kuriose realizuotas MPI palaikymas.

```
[
Type = "collection";
Requirements =   Member("VO-balticgrid-ktu/ENV/MPI/MPICH2/1.0.5",
                 other.GlueHostApplicationSoftwareRunTimeEnvironment) &&
                 Member("VO-balticgrid-ktu",
                 other.GlueHostApplicationSoftwareRunTimeEnvironment) && ! (
```

```
        other.GlueCEUniqueID == "grid5.mif.vu.lt:2119/blah-pbs-sdj" ||
        other.GlueCEUniqueID == "grid2.mif.vu.lt:2119/jobmanager-lcgpbs-sdj"
                );
nodes = {
[
JobType = "MPICH";
NodeNumber = 20;
Executable = "MPItest.sh";
Arguments = "crypt 00";
StdOutput = "test.out";
StdError = "test.err";
InputSandbox = {"MPItest.sh","crypt.c","des.h"};
OutputSandbox = {"test.err","test.out","mpiexec.out"};
Requirements = (other.GlueCEInfoLRMSType == "PBS");
RetryCount = 0;
Lrms_Type = "PBS";
],
[
JobType = "MPICH";
NodeNumber = 20;
Executable = "MPItest.sh";
Arguments = "crypt 02";
StdOutput = "test.out";
StdError = "test.err";
InputSandbox = {"MPItest.sh","crypt.c","des.h"};
OutputSandbox = {"test.err","test.out","mpiexec.out"};
Requirements = (other.GlueCEInfoLRMSType == "PBS");
RetryCount = 0;
Lrms_Type = "PBS";
]
};
]
```