MANAS Journal of Engineering

MJEN

Volume 3 (Issue 2) (2015) Pages 22-33

Факторизациялоо алгоритмдердин салыштырмалуу анализи

Gulida Kimsanova
Kırgızistan Türkiye Manas Üniversitesi, Bilgisayar Mühendisliği Anabilim Dalı, Bişkek, Kırgızistan
gulida.kms@gmail.com
Rita İsmailova
Kırgızistan Türkiye Manas Üniversitesi, Bilgisayar Mühendisliği Anabilim Dalı, Bişkek, Kırgızistan,
rita.ismailova@manas.edu.kg
Rayımbek Sultanov
Kırgızistan Türkiye Manas Üniversitesi, Bilgisayar Mühendisliği Bölümü, Bişkek, Kırgızistan,
rayimbek.sultanov@manas.edu.kg

Received: 07.07.2015; Accepted: 06.11.2015

Аннотация Көптөгөн ачык ачкычтуу криптосистемалардын негизи катары колдонулган натуралдык санды факторизациялоо проблемасы заманбап компьютерлер үчүн да оор маселе болуп саналат. Биз бул иште C++ программалоо тилинде GMP китепканасын колдонуп, 4 алгоритмди иштеп чыктык жана бул алгоритмдердин иштөө убакыттарын салыштырдык. Алгоритмдер ар кандай өлчөмдөгү сандарды, ошондой эле факторлорунун ортосунда ар кандай аралык болгон сандарды факторизациялоо үчүн колдонулду.Биздин жыйынтыктар 296 биттик санды факторлорго ажыратууда Поллард rho алгоритминин эң бат иштешин, ошол эле учурда, факторлордун ортосундагы аралык кичине болгон учурда Ферма алгоритминин эң бат иштешин көрсөттү. Мындай сандар үчүн Брент алгоритминин жай иштегени байкалды, бирок бул алгоритм Ферма алгоритми натыйжа бере албаган учурларда ийгиликтүү жыйынтык бере алды.

Ачкыч Сөздөр: *Натуралдык сандын факторизациясы, GMP,Тривиалдык бөлүү, Ферма алгоритми, Поллард rho алгоритми, Брент алгоритми*

Comparative analysis of integer factorization algorithms

Abstract Integer factorization problem, which is used as the basis in many public key cryptosystem, is generally thought to be hard problem even on a modern computers. In this work we implement 4 integer factorization algorithms using GMP library on c++ and compare the running time of these algorithms. Algorithms were used to factor numbers of different sizes, as well as for number with different distance between factors. Our results showed that for numbers up to 296 bits the Pollard rho algorithm is the fastest one, while Fermat algorithm is fast when distances between factors are small. Brent algorithms appeared to run slower for this rage of numbers, however it succeeded to factor numbers which Fermat algorithm fail to factor.

Keywords *Integer factorization, GMP, Trial division algorithm,Fermat algorithm, Pollard rho algorithm, Brent algorithm*

КИРИШҮҮ

РША (RSA) Алгоритми

1978 – жылы чыккан [1] макалада RSA Диффи менен Хеллмандын идеясын көрсөткөн чындыгында иштеген системаны сунуштайт. RSA системасы сандар теориясындагы өтө жөнөкөй маселелерге негизделген, азыркы күндө андай көптөгөн ачык ачкычтуу криптосистемалар белгилүү. Азыркы күндө белгилүү болгон, сандар теориясынын маселелерине негизделген ачык ачкычтуу криптосистемаларды жалпысынан үч топко бөлсөк болот:

- Эсептөөдөгү кыйынчылык чектүү Абелиан тобунан тамыр чыгарууга негизделген. Бул топко тиешелүү системалар: RSA, Рабин [2], LUC [3]. . .
- Эсептөөдөгү кыйынчылык дискреттик логарифма маселелерине негизделет. Бул топко кирген системалар: Диффи-Хеллмана схемасы [4], Эль-Гамаль [5]. . .
- Эсептөөдөгү кыйынчылык белгилүү бир топтон калдыктардын классын бөлүп чыгарууга негизделет. Бул топко тиешелүү системалар: Goldwasser-Micali[6], Okamoto - Uchiyama[7].

Алгоритм 1. RSA системасын түзүү
1. p жана q - эки жөнөкөй санды тандап алуу;
2. $n = pq$ амалын аткаруу;
3. $\varphi(n) = (p - 1) \cdot (q - 1)$ амалын аткаруу;
4. $\gcd(d, \varphi(n)) = 1$- функциясына туура келгендей кылып d маанисин тадап алуу ;
5. d маанисинин мультипликативдүү тескериси катары e маанисин тандап алуу;
$ed \equiv 1 \pmod{\varphi(n)}$;
(e, n) – жубу ачык (жалпыга маалым) ачкыч жана (d, n) – жубу жеке (жашыруун) ачкыч

Алгоритм 2. M жөнөкөй текстин шифрленген C текстине өткөрүү:
1. M ди 0, ..., n-1 аралыгындагы M_0, …, M_j сандар катары көрсөтүү;
2. Бардык $M_i \in [M_0, \ldots, M_j]$ үчүн $C_i = M^e \pmod{n}$ амалын аткаруу;
3. Жыйынтыкта алынган шифрленген текст C , бул $C_0, \ldots, Cj$;

Алгоритм 3. C шифрленген текстин M жөнөкөй текстине өткөрүү:
1.C ни 0, ..., n-1 аралыгындагы C_0, …, C_j сандар катары көрсөтүү (бул шифрлөөдө алынган катар менен окшош болушу керек);
2. Бардык $C_i \in [C_0, \ldots, C_j]$ үчүн $M_i = C^d \pmod{n}$ амалын аткаруу;
3. Жыйынтыкта алынган жөнөкөй текст M , бул $M0, \ldots, Mj$;

Бул схема түшүнүктүүрөөк болушу үчүн төмөнкү мисалды карап көрсөк:
Алиса Боб менен жашыруун сүйлөшүүнү каалайт, ал Бобдун ачык ачкычын алат (e_B, n_B). Андан соң Алиса $C = M^{eB} \pmod{n_B}$ амалын аткарат, бул жерде M Алиса жөнөткүсү келген жөнөкөй текст. Бирок, Алиса Бобко шифрленген C билдирүүсүн жиберет. Боб Алисадан келген шифрленген C билдирүүсүн алат жана өзүнүн (d_B, n_B) – жеке ачкычын колдонуп, $M = C^{dB} \pmod{n_B}$ амалын аткаруу менен ал билдирүүнүн шифрин чечет. Баштапкы текст калыбына келип, окуганга мүмкүн болгон абалда болуп калат.
Кандайдыр бир коопсуздук механизмдери болбосо криптосистема жараксыз болуп калат. Эң кеңири таралган RSA криптосистемасынын коопсуздук механизми чоң натуралдык сандарды факторизациялоодон көз каранды. Эгерде, Алгоритм 1. де алынган (e, n) – ачык (жалпыга маалым) ачкычтар жубундагы n дин факторлору аныкталып калса, ушул эле алгоритмдин 3-5 – кадамдарында көрсөтүлгөн амалдарды аткаруу менен d жашыруун ачкычын жеңил эле таап алууга жол ачылат.
Бул изилдөөдө биз программалоо тилдеринде чоң сандар менен иштөөдө жакшы мүмкүчүлүктүрдү жараткан GMP китепканасын колдонуп, бир канча факторизациялоо алгоритмдерин C++ тилинде иштеп чыгып, алардын иштөө убакыттарын салыштырабыз.

Бүтүн Сандык Факторизациялоо

Натуралдык санды (бүтүн санды) факторизациялоо – ал санды жөнөкөй көбөйтүүчүлөргө ажыратуу болуп саналат. Факторизация эсептөө боюнча татаал маселе. Аны менен бирге, жалпыга маалым болгон бүтүн санды факторизациялоочу бир дагы алгоритм жок. Андыктан, бул маанилүү бир маселе болуп эсептелет, себеби, учурда дүйнө жүзүндөгү колдонулуп жаткан криптографиялык системалардын коопсуздугу бүтүн сандык факторизацияга негизделген. Ал системалардын катарындагы эң кеңири таралганы - RSA системасы.

Бүгүнкү күндө, бүтүн санды факторизациялоочу бир канча алгоритм белгилүү [8]. Тривиалдык бөлүү алгоритминен тартып, классикалык Ферманын факторизациялоо методуна чейин [9]. Ошондой эле, Эйлердин факторизациялоо методу [10], квадраттык иргеп алуу [11], сандык талаа боюнча иргеп алуу [12] алгоритмдери бар.

Факторизация алгоритмдерин татаалдыгы боюнча эки топко бөлсөк болот: биринчиси – экспоненсиалдык, экинчиси – субэкспоненсиалдык.

Тарыхы

Фермага Чейинки Мезгил

Сандарды жөнөкөй көбөйтүүчүлөргө ажыратуу концепциясы - жөнөкөй сандар жана факторизациялоонун бир маанилүүлүгү арифметика теоремасынын негизин түзгөндүгүн болжол менен Эвклид мезгилинде – б.з.ч. аныкташкан. Алгач математиканын өнүгүүсүнүн себеби негизинен бизнесте же күнүмдүк жашоодо колдонуу болгон жана сандарды факторлорго ажыратуу бизнесте да, күнүмдүк жашоодо да колдонулган эмес, ошондуктан факторизациянын өнүгүүсүнүн себеби - бир гана теориялык кызыкчылык болгон. Бирок, 1970 – жылдардын аягынан тартып ал теориялык кызыкчылык үчүн эмес, коопсуздук талаптары үчүн изилденип, өнүгө баштады. Ферма мезгилине чейин, Тривиалдык бөлүү методунан башка бир дагы факторизациялоо методу тууралуу маалымат болбогону үчүн, факторизациялоо тарыхы Пьер де Фермадан башталат [13], [14] .

Ферма (1640 - жылдар)

1643 – жылы Ферма сандарды факторизациялоо үчүн кээ бир кызыктуу идеяларды көрсөткөн. Ал идеялар менен бүгүнкү күндөгү эң тез иштеген методдорго чейинки жолду көзөмөлдөөгө болот. Ферманын идеясы санды эки сандын квадраттарынын айырмасы катары жазып алуу болгон, б.а. ал $x + y$ жана $x - y$ факторлорун алуу үчүн курама N санын $N = x^2 - y^2$ деп жазып алуу аркылуу табууга аракет кылган. Эгерде, N санынын факторлору бири бирине жакын болсо, Ферма методу өтө тез иштейт, башка учурда эффективдүү эмес болуп калат [13], [14].

Эйлер (1750 - жылдар)

Эң продуктивдүү математиктердин бири болгон Эйлердин да бүтүн сандык факторизациялоого карата өз ойлору болгон. Ал бүтүн сандар үчүн атайын формаларды караган. Анын методдорунун бири – бир эле D санын колдонуп, эки башка жол менен $N = a^2 - Db^2$ катары жазууга мүмкүн болгон бүтүн сандар үчүн гана колдонулат. Бул метод Ферманын жогоруда сүрөттөлгөн методу сыяктуу иштейт. Бул метод ошол убактагы чоң сандарды факторлорго ажыратууда ийгиликтүү колдонулган. Ошондой эле, Эйлер өзүнүн методун Ферманын теоремаларынын бирин жокко чыгаруу үчүн да колдонгон [13], [14].

Лежандр (1798 - жылдар)

Лежандр жыйынтыкта факторизация алгоритмдеринин эволюциясына алып келүүчү идеяны сунуштаган. Лежандрдын туура келүүчү квадраттар теориясы учурдагы жалпы факторизация алгоритмдеринин өзөгү (ядросу) болуп саналат. Азыркы учурда, эсептөө кубаттуулугунун жетишсиздиги факторлорго ажыратылуучу сандын өлчөмүн чектеп койгон, бирок, эсептөө кубаттуулугу ар башка болгон машиналарга бөлүнүп, параллельдик эсептөө мүмкүнчүлүгү колдонулган учурда өзүнүн эң эффективдүү алгоритм экенин далилдей алат [13], [14].

Гаусс (1800 - жылдар)

Лежандр өзүнүн идеясын жарыкка чыгарып жаткан учурда, Гаусс сандар теориясынын эң маанилүү жумушунун үстүндө иштеп жаткан. 1801 – жылы санды факторлорго ажыратуу боюнча бир канча идеялар жана методдор камтылган "Арифметикалык изилдөөлөр" жарыкка чыгат. Гаусстун методу татаал, бирок аны *N* модулу боюнча бир канча квадраттык калдыкты табуу менен, жөнөкөй сан болгон факторду ирген алган Эратосфендин методу менен салыштырууга болот. Элементтерди ирген алуу методу заманбап факторизация методдорунун маанилүү бөлүгү болуп саналат. Лежандр менен Гаусстан кийин, жылдар бою эч кандай жаңы идеялар пайда болгон эмес. Лежандр менен Гаусстун методдорун колдонуу менен 10-15 орундуу сандарды факторлорго ажыратуу ал учурда татаал болгонуна жана бир канча күндү талап кылганына, көптөгөн барак, боёк сарптаганына карабастан ал убактагы математиктер дагы да болсо, эсептөөлөрдү кол менен жасоодон убактыларын аябай келишкен. Мындай көрүнүш кимдир бирөөнүн ирген алууну кишинин ордуна жасоочу машинаны курууну чечкенге чейин уланып келген жана ал машинанын курулуусу факторизациялоо проблемасынын кайрадан изилденишине себеп болгон [13], [14].

ЭЭМ кирип келиши. . .

19 – кылымдын аягында бир канча киши, бири биринен көз карандысыз, өз алдынча факторизация алгоритмдерине негизделген ирген алуу эсептерин жүргүзүү үчүн ар кандай машиналарды кура баштаган. 1896 – жылы Лоуренс тиштери бар жылып туруучу механизм аркылуу өткөн, жылып туруучу кагаз тасмасы колдонулган машинаны сунуш кылат. Тиштердин саны өзгөчөлөнгөн модулду билдирсе, кагаздын бетиндеги тиштер калтырган чекиттер модул боюнча келип чыккан калдыкты билдирет. Бул машинаны куруу ишке ашпаса да, бул идея ушуга окшогон башка машиналардын курулушуна жол ачкан. 1910 – жылы Лоуренстин макаласы французчага которулуп чыккандан кийин, Бельгиялык математик Морис Крайчик Лоуренстин идеясын колдонуп, өзүнүн машинасын курган. Ошол эле мезгилде, G´erardin жана Carissan бир туугандар да ушуга окшош машинаны курушкан, бирок, бул бир туугандар чын эле иштеген жана жакшы жыйынтыктарды берген, биринчи ирген алуучу машинаны биринчи дүйнөлүк согуштан кийин гана кура алышкан (машина кол менен иштетилген). Эң ийгиликтүү машина куруучу Лехмер болгон. Ал Крайчиктин да, Кариссандардын да машиналары тууралуу маалыматка көп жылдар бою ээ болгон эмес жана ал көптөгөн ирген алуучу түзүлүштөрдү жасаган, алардын кээ бирлери ошол мезгилдин эң алдынкы заманбап технологияларынан болгон. Крайчик жана Лехмер курган машиналар жана алар колдонгон алгоритмдер кийинчерээк иштелип чыккан квадраттык ирген алуу алгоритмдерине өтө окшош болгон [13], [14].

20 – кылымдын акыры

1970 - жылдарга чейин ирген алуучу машиналар - факторизациянын эң тез иштеген методдору болгон, бирок, азыркы учурда компьютерлер жетишээрлик деңгээлдеги эсептөө кубаттуулугу менен камсыз кыла алгандыктан, тадап кылынган нерсе бир гана туура иштеген алгоритмдер болуп калды. 1970 – жылдары көптөгөн факторизациялоо алгоритмдери иштелип чыккан: Даниэл Шанк ишке ашыруу татаал болгон SQUFOF алгоритминде квадраттык формаларды(Гаусс сыяктуу) колдонгон. Факторизация методдорун иштеп чыгууда Джон Поллард активдүү катышкан жана 1974 – жылы $p-1$ методун иштеп чыккан, бир жылдан кийин, *Pollard's rho* чыгары менен, бул эки метод тең бүтүн сандардын өзгөчө классына багытталган.

1975 – жылы Моррисон жана Бриллхарт, алгач Лежандр колдонгон сыяктуу, бөлчөктөргө негизделген, жалпы багыттагы , эң тез иштеген биринчи адгоритмди – CFRAC'ни көрсөтүшкөн. 1980 – жылы Брент Поддарддын *rho* алгоритмин оптимизация кылган. 1982 – жылдары Карл Померанс квадраттык ирген алууну иштеп чыгат. 1984 – Шнорр менен Ленстра Шанкстын идеясын алып, оригинал методду жакшыртышкан. 1987 – жылы Ленстра таптакыр башка жолду тандап, өзүнүн кичинекей факторлордон турган курама сандар үчүн арналган - ECM алгоритмдеринде эллиптикалык кыйшаюуну колдонуу менен жаңы метод иштеп чыккан. 1988 – жылы 31 – августта, Джон Поллард А. М. Одлызкого, жана көчүрмөлөрүн Пю Ричардга, Брентке, Бриллнардка,

Ленстрага, Шноррго жана Суямага алгебралык сандардын талааларынын жардамы менен сандардын өзгөчө классын факторизациялоону сунуштаган кат жеберет. Андан көп өтпөй, сандык талаа боюнча иргеп алуу методу ишке ашырылган жана жалпыга багытталган таанымал алгоритм болуп калат. Сандык талаа боюнча иргеп алуу алгоритми бир кыйла оор алгоритм, бирок, тез иштейт жана 512 биттик курама санды ийгиликтүү факторлорго ажыраткан. Бул алгоритм 1990 – жылдары иштелип чыккан, жана андан бери эч кандай жаңы идеялар сунушталган эмес, жөн гана сандык талаа боюнча иргеп алуу методун оптимизациялап келишүүдө [13], [14].

Факторизациялоо Алгоритмдери

Бүтүн санды факторизациялоо алгоритмдерин төмөнкүдөй эки топко бөлүүгө болот:

- Өзгөчө максатта колдонулган алгоритмдер
- Жалпы максатта колдонулган алгоритмдер

Өзгөчө максатта колдонулган алгоритмдер – сандардын атайын классына арналган алгоритмдер, мисалы: эгер, бир сан жалгыз же бир канча кичинекей факторлорго ээ болсо. Жалпы максатка багытталган алгоритмдер – сандардын атайын классы үчүн багытталбаган, башкача айтканда 100 биттик санды 1 битке жана 99 битке ажыратууда канча убакыт талап кылса, ошол эле санды эки 50 битке ажыратууда да ошончо эле убакыт талап кылган алгоритмдер.

Жогоруда сүрөттөлгөн эки алгоритм тең факторизациялануучу N санынын курама сан болушунан көз каранды, ошондуктан факторлорго ажыратуудан мурда, ал сандын жөнөкөй сан экендигин текшерүү керек.

Кээ бил алгоритмдердин иштөө убактысы субэкспоненсиалдык жана көбүнчө $\ln[\alpha, c]$ жазылышы менен көрсөтүлөт.

Өзгөчө максатта колдонулган алгоритмдер

Өзгөчө максатта колдонулган алгоритмдердин иштөө убактысы – факторлорго ажыратылуучу сандын касиетине же анын белгисиз факторлорунун биринен өлчөмүнө, өзгөчөлүгүнө жараша болот. Айрыкча, алгоритмдердин өзгөчөлүктөрү да алардын иштөө убактылары менен айырмаланат. Өзгөчө максатта колдонулган алгоритмдеринин маанилүү төмөнкү классы болуп биринчи категориядагы алгоритмдер эсептелет. Бул алгоритмдердин иштөө убактылары сандын кичине факторунун өлчөмүнөн көз каранды болушат.

Өзгөчө максатта колдонулган алгоритмдер төмөнкүлөр болуп саналат:

- Тривиалдык бөлүү методу (Trial division)
- Дөңгөлөк факторизация
- Поллард *rho*
- Поллард $p-1$ алгоритми
- Уилям $p+1$ алгоритми
- Ленстра эллиптикалык кыйшаюу факторизациясы
- Ферма факторизациялоо методу
- Эйлер факторизациялоо методу
- Өзгөчө сандык талаа боюнча иргеп алуу

Тривиалдык бөлүү методу

Тривиалдык бөлүү – бүтүн сандык факторизациялоодогу эң жөнөкөй алгоритм.

$$st = N \tag{1}$$

жана $s \leq t$ болгон учурда, s менен t сандарын N санынын тривиалдык эмес факторлору деп элестетебиз. Тривиалдык бөлүү алгоритмин аткаруу үчүн $s = 2, m$, $\lfloor\sqrt{N}\rfloor$ үчүн $s|N$ экендигин текшерүү керек болот. s бөлүүчүсү табылгандан кийин, $t = N/s$ аркылуу экинчи бөлүүчүнү таап алабыз. Ошентип, N саны үчүн факторизация табылды десек болот. $s \leq \lfloor\sqrt{N}\rfloor$ дин жогорку чеги төмөнкү теориянын негизинде келип чыгат.

Теорема: $st = N$ жана $s \leq t$ болгон учурда, эгер s, t сандары N санынын тривиалдык эмес факторлору болсо, анда $s \leq \lfloor \sqrt{N} \rfloor$.
Аныктоо: $s > \sqrt{N}$ деп эсептесек, анда $st = N$ деген божомолго карама-каршы $t \geq s > \sqrt{N}$ болуп калат. Демек, $s \leq \lfloor \sqrt{N} \rfloor$ экендиги аныкталды.
Эгер, бул алгоритмге курама N саны берилсе, алгоритм $s \leq t$ шартына туура келген s, t тривиалдык эмес факторлордун жубун кайтарып берет. $s|N$ амалы $s \equiv 0 (\mathrm{mod}\ N)$ менен эквиваленттүү жана бир канча программалоо тилдеринде модулдук арифметика аркылуу ишке ашырса болот.

Ферма факторизациялоо методу
Бул алгоритм 1600 – жылдарында математик П. Ферма тарабынан ачылган [9]. Ферманын факторизациясы курама N санын эки сандын квадраттарынын айырмасы катары жазып алат :

$$N = x^2 - y^2 \tag{2}$$

Бул квадраттардын айырмасы N санынын факторизациясына алып келет :

$$N = (x + y)(x - y) \tag{3}$$

$s \leq t$ шартында $st = N$ болуп, s менен t сандары N'дин так, тривиалдык эмес факторлору деп алсак, бул жерден x менен e сандарын $s = (x - y); t = (x + y)$ аркылуу тааба алабыз.

Поллард *rho* факторизациялоо методу
Поллард *rho* факторизациялоо алгоритми чоң сандардын бөлүүчүлөрүн табууда эң тез ыктымалдык алгоритмдердин бири. Бул алгоритм Флойд"дун цикл табуу алгоритминде негизделген. Башкача айтканда x жана y сандары $\sqrt[1,177]{p}$ сан тандалса, p модулю боюнча 0,5 ыктымалдуулук менен дал келет [15]. Алгоритмде x_0 саны тандалып,

$$x_{n+1} = x_n^2 + a (\mathrm{mod}\ n) \tag{4}$$

түрүндөгү итерациялоо процесси каалагандай экиx_i жана x_j сандары p модулю боюнча дал келгенде чейин жүргүзүлөт. x_i жана x_j сандары табылгандан кийин $\gcd(|x_j - x_i|, N)$ изделген p саны. Поллардды $j = 2i$ болушун кеңеш берген.

Брент факторизациялоо методу
[16] 1980 Поллард методуна таянып, ылдамдыгы көбүрөөк алгоритмин басмага чыгарган. Ылдамдыкты Флойд"дун цикл табуу алгоритминин ордуна Брент цикл табуу алгоритмин колдонуу мүмкүнчүлүгүн түзгөн. Цикл табууда Поллард алгоритми x_n ди x_{2n} менен салыштырууну сунуштаган болсо, Брент алгоритминде x_n ди x_m менен салыштыруусу сунушталган. Бул алгоритмде m экинин n санынан кичине боло турган эң чоң даражасы.

МАТЕРИАЛ ЖАНА МЕТОДДОР

Материалдар
Бул изилдөөдө колдонулган компьютерлер:

1. Персоналдык компьютер

Процессор: Intel(R) Core(TM) 2 Quad CPU Q8300 @ 2.50GHz
Оперативдик эс (RAM) : 4,00 GB
Операциондук система: 64-bit Operating System

2. Ноутбук

Процессор: Intel(R) Core(TM) i3-2310M CPU @ 2.10GHz

Оперативдик эс (RAM) : 4,00 GB
Операциондук система: 64-bit Operating System

Microsoft Visual Studio — Microsoft компаниясынын продуктусу. Өз ичине программдык жабдыкты иштеп чыгуудагы айкалышкан чөйрөнү жана бир катар башка аспаптарды камтыйт. Бул продукттар консольдук тиркемелерди да, графикалык интерфейстүү тиркемелерди да иштеп чыгууга мүмкүндүк берет.
Visual Studio 2010 (коддук аталышы *Hawaii*, Ultimate үчүн — *Rosario*; ички версиясы 10.0) — 2010-жылдын 12 - апрелинде чыгарылган. Буга чейинки версиялардан өзгөчөлөнүп, бул версияда C# 4.0 жана Visual Basic .NET 10.0, ошондой эле, F# тилдери кошулган.

Microsoft Visual C++ (MSVC) — C++ тилинде тиркемелерди иштеп чыккан айкалышкан чөйрө. Microsoft фирмасы тарабынан иштелип чыккан жана Microsoft Visual Studio комплектинин бир бөлүгү катары орнотулган.

C++ тили
C ++ - типтелген статистикалык компиляцалануучу программалоо тили. Процедуралык программалоону, абстракциялык берилиштерди, типтерди (объектер), виртуалдык функцияларды, объектке багытталган программалоону, жалпыланган программалоону, контейнерлер жана алгоритмдерди, жогорку деңгээлдеги жана төмөнкү деңгээлдеги тилдерди айкалыштырып өз ичине камтыйт.
Эң популярдуу программалоо тилдеринен бири болуп, C++ программдык жабдыктарды иштеп чыгууда колдонулат. Операционндук сисемаларды, компьютерик оюндарды, компьютердин жабдууларына драйверлерди түзүү областарында эч бири менен алмаштырылгыз түрдө колдонулат. C++ тилинин ар түрдүү платформалар үчүн акылуу жана акысыз көптөгөн реализациясы бар. Мисалы: x86 платформасы үчүн - GCC, Visual C++, Intel C++ Compiler, Embarcadero (Borland) C++ Builder жана башкалар. Бул тил “Java” жана “C#” сыяктуу башка программалоо тилдеринин өнүгүүсүнө чоң салымын кошкон .

C++ тилинде китепканалар
C++ тилинде китепканалар – бул базалык тилде жазылган класстардын жана функциялардын жыйындысы. C++ тилинде стандартка кирбеген көп сандагы китепканалар бар. C++ тилиндеги программаларда C тилиндеги көптөгөн китепканаларды колдонсо болот. Биз өзүбүздүн ишибизде чоң сандар менен иштөөдө колдонулган gmp – 6.0.0 китепканасын колдондук.

Ар түрдүү тактыктагы арифметикалык китепканасы (GNU Multiple Precision Arithmetic Library)
GMP китепканасынын максатталган программалары - криптографиялык программалар жана изилдөөлөр, интернет коопсуздугунун программалары жана компьютердик алгебра системалары.
GMP китепканасы бардык операнддардын өлчөмү боюнча чоң сандар менен иштеген башка китепканалардан тезирээк иштөөнү көздөйт. Ал үчүн жасалган кээ бир маанилүү факторлор төмөнкүлөр:

- Бүтүн сөздөрдү жөнөкөй арифметикалык тип катары колдонуусу;
- Операнддардын ар кайсы өлчөмдөрүнө ар башка алгоритмдерди колдонуусу;
- Чоң сандар менен тез иштеген алгоритмдер негизинен кичине сандар менен иштегенде жай болуп калат;
- Атайын ар башка процессорлор үчүн көптөгөн маанилүү ички циклдар үчүн жогору оптимизацияланган ассемблер тилинде жазылган коддун колдонулушу;

МЕТОДДОР

HP ProBook 4530s ноутбугуна Windows 7 Ultimate операциондук системасында, Microsoft компаниясынын Visual Studio 2010 пакетин орнотуп, [17] көрсөтүлгөн инструкция боюнча gmp – 6.0.0 китепканасын кошуу менен алгоритмдердин программдык кодун жазуучу чөйрөнү даярдадык. C++ тилинди 4 алгоритмдин коду жазылды (Trial Division, Fermat Factorization, Pollard Rho, Brent Factorization). Ар бир алгоритм үчүн өзүнчө Project түзүлүп, Debug аркылуу .exe файлдары алынды. Алынган .exe файлдарды талап кылынган бир канча .dll файлдар менен чогуу колдонуу менен, материалдар бөлүмүндө көрсөтүлгөн 2 персоналдык компьютерде жана бир ноутбукта иштетип, анализ үчүн жыйынтыктар алынды.

ЖЫЙЫНТЫКТАР

Бул иште C++ тилинде GMP китепканасы колдонулуп 4 алгоритмдин программасы түзүлдү. Ал алгоритмдер - Тривиалдык бөлүү, Ферма факторизациясы, Поллард *rho* жана Брент факторизациясы алгоритмдери. Жыйытыктар Жадыбал 4-1 жана Жадыбал 4-2де берилген.
Алгач, бири-бирине жакын жөнөкөй сандардын көбөйтүндүсү болгон татаал санды факторизациялоо үчүн алгоритмдердин каржаган убактысын эсептедик. Жыйынтыгын Жадыбал 4-1 ден көрсөк болот.

Жадыбал 1 Жакын сандардын факторизациялоо убакыттары

N	***p* × *q* биттердин саны**	**Тривиалдык бөлүү**	**Ферма**	**Поллард *rho***	**Брент**
20-30 биттик сандар	14 × 14	0.0219	0.0015	0.0013	0.1153
30-40 битик сандар	18 × 18	0.1134	0.005	0.0052	1.7978
40-50 битик сандар	22 × 22	6.983	7.6445	0.01	89.097
50-60 битик сандар	26 × 26	25.0054	0.3982	0.0274	446.1792

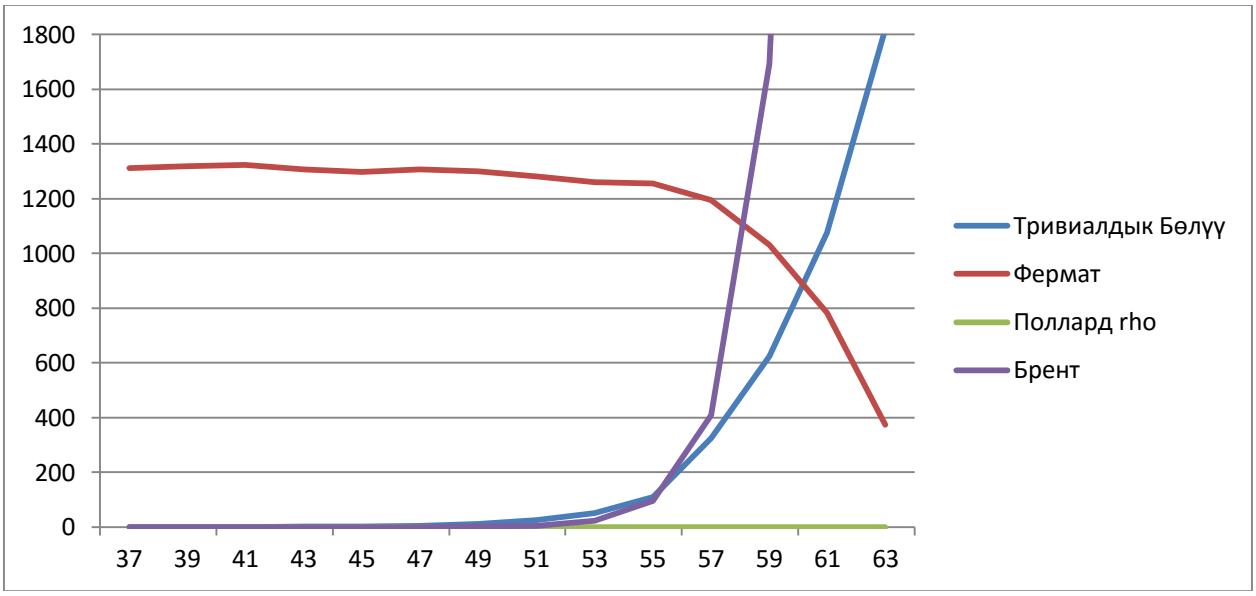
Алынган жыйынтыктан Поллард *rho* алгоритминин эң бат аткаргандыгын көрө алабыз. Бул алгоритмдин иштөө убактысы сан чоңойгон сайын өсүп келет. Экинчи орунда Ферма алгоритми көрүнүп турат, бирок 40-50 биттик сандар үчүн бул алгоритм Тривиалдык алгоритминден жайыраак иштегени байкалат. Ошол эле учурда, сандар чоңойгон сайын алгоритмдин иштөө убактысы кайрадан азайганын көрө алабыз. Тривиалдык бөлүү жана Брент алгоритмдери сандардан чоңоюшу менен иштөө убактылары да өсүп жатат. Таң калтырган көрүнүш, теорияда көрсөтүлгөн бул 4 алгоритмдин эң тез иштеген Брент алгоритминин кичине сандар үчүн абдан жай иштегенин көрдүк. Экинчиден, RSA шифрлөө системасында FIPS тарабындан p жана q сандарынын 1025 жана 180 бит болушу, башкача айтканда ара байланыш 100×15 болушу керектиги сунушталган. Бул түр ара мамилеси болгон сандардын көбөйтүүчүсүн факторизациялоо абдан оор деп эсептелинет. Андыктан, бул иште биз татаал санды түзгөн жөнөкөй сандардын ара байланышы өзгөргөндө факторизациялоого кеткен убакыттын өзгөрүлүшүн эсептеп чыктык. Жыйытыгы Жадыбал 4-2 жана Сүрөт 4-1де көрсөк болот.

Жадыбал 2 Сандардын ара байланышына карата факторизациялоо убакыттары

N	***p* × *q* биттердин саны**	**Тривиалдык бөлүү**		**Ферма**		**Поллард *rho***		**Брент**	
37 бит	32 × 5	0 ,	203	1312 ,	27	0 ,	001	0 ,	001
39 бит	32 × 7	0 ,	38	1319 ,	40	0 ,	001	0 ,	001
41 бит	32 × 9	0 ,	7	1324 ,	18	0 ,	001	0 ,	001
43 бит	32 × 11	1 ,	197	1306 ,	41	0 ,	001	0 ,	01
45 бит	32 × 13	2 ,	754	1298 ,	50	0 ,	001	0 ,	06
47 бит	32 × 15	5 ,	245	1306 ,	07	0 ,	001	0 ,	231

49 бит	32×17	11 ,	743	1300 ,	76	0 ,	001	1 ,	113
51 бит	32×19	25 ,	893	1281 ,	98	0 ,	001	5 ,	178
53 бит	32×21	52 ,	365	1260 ,	23	0 ,	01	22 ,	27
55 бит	32×23	109 ,	721	1256 ,	78	0 ,	051	95 ,	516
57 бит	32×25	324 ,	827	1193 ,	85	0 ,	117	410 ,	167
59 бит	32×27	625 ,	139	1030 ,	10	0 ,	09	1693 ,	11
61 бит	32×29	1075 ,	44	784 ,	49	0 ,	211	6142 ,	29
63 бит	32×31	1819 ,	63	373 ,	91	0 ,	523	23838 ,	3

Татаал сандын факторлору болгон жөнөкөй сандардын ара байланышы өзгөргөндө Тривиалдык бөлүү алгоритминин иштөө убактысынын сандардын чоңдугуна түздөн-түз байланышта экендигин көрсөк болот. Ал эми, Ферма алгоритминде тескеринче сан чоңойгон сайын анын иштөө убактысынын азайгандыгын көрсөк болот. Жадыбал 4-2 деги берилген факторлорго карап жыйынтык чыгара турган болсок, алгоримдин иштөө убактысынын p жана q сандарынын бири-бирине жакындаган сайын азайганын байкайбыз. Кайрадан эле, Поллард rho алгоритминин эң тез алгоритм болгонун көрсөк болот.



Сүрөт 1. Алгоритмдердин иштөө убакыты

Ошондой эле, 100×15 биттик катышта болгон чоң сандар үчүн факторизациялоо жүргүзүлдү. Сандардын эң кичинеси 141 биттик сан, эң чоңу 296 биттик болду. Жыйынтыгын Жадыбал 4-3 төн көрсөк болот.

Жадыбал 3. Факторлору 100×15 биттик катышта болгон чоң сандарды факторизациялоо убакыттары

N	***p* × *q* биттердин саны**	**Тривиалдык бөлүү**	**Ферма**	**Поллард *rho***		**Брент**	
141 бит	122×19	>432000	>432000	0 ,	04	5 ,	404
144 бит	125×19	>432000	>432000	0 ,	3	7 ,	23
145 бит	126×20	>432000	>432000	0 ,	01	8 ,	771
148 бит	128×20	>432000	>432000	0 ,	06	11 ,	17
218 бит	190×28	>432000	>432000	0 ,	348	3322 ,	44
295 бит	256×40	>432000	>432000	42 ,	747	>432000	
296 бит	256×40	>432000	>432000	72 ,	001	>432000	

Татаал N санын чоң болгондо жана факторлор 100×15 биттик катышта болгондо Тривиалдык бөлүү жана Ферма алгоритмдеринин иштөө убакыты 432000 секундадан ашык, башкача айтканда 5 күндөн ашык болгонун көрдүк. Эң тез Поллард *rho* алгоритми жыйынтык чыгарды. Брент алгоритми факторлору жакын сандарды факторизациялоодо жай иштегени менен, ара байланышы 100×15 болгондо чоң сандардын (218 битке чейинки сандардын) факторлорун таба алганын көрдүк. А бирок, факторлор p саны 256 биттик сан жана q саны 40 биттик сан болгондо жыйынтык ала албадык.

ТАЛКУЛООЛОР ЖАНА АЛДЫДАГЫ ИЗИЛДӨӨЛӨР

Тривиалдык бөлүү алгоритми факторлорго ажыратууда мүмкүн болгон бардык варианттарды карап чыгат, ошондуктан бул алгоритм детерминисттик алгоритмдердин катарына кирет. Себеби, бул алгоритм сөзсүз түрдө жыйынтык берет. Алынган жыйынтыктарга карап, алгоритмдин иштөө убактысынын сандар чоңойгон сайын өсүп жатканын байкайбыз. Анын себеби – алгоритмдин иштөө кадамы 2ге барабар жана ал мүмкүн болгон бүт вариантты текшерип чыгышы керек. Тривиалдык бөлүү алгоритминин иштөө убакыты факторлорунун ара мамилесине байланышта эмес, көбүрөөк факторлордун бирөөсүнүн канчалык кичинекей болгонундан таасирленет. Ошондуктан, RSA системасында колдонулган чоң сандар үчүн бул алгоритмди колдонуу абдан эле ыңгайсыз болуп калат. Биринчиден, факторлорду таап берүү үчүн узак убакыт талап кылса, экинчиден алгоритм иштеп жаткан машинанын үзгүлтүксүз иштөөсү талап кылынат. Биз өзүбүздүн практикабызда колдонгон кээ бир чоң сандар үчүн 3-4 сутка коротуп, бирок жыйынтык ала албаган учурлар болду. Кээ бир учурларда алгоритм иштеп жаткан компьютерлердин белгилүү бир убакыттан кийин өчүп калган учурларын байкадык.

Ферма факторизациясы алгоритми факторлорду издөөдө тривиалдык бөлүү алгоритминен айырмаланып, факторлорго ажыратылып жаткан татаал сандын квадраттык тамырына жакын болгон сандан баштап текшерип баштайт. Ошондуктан, биз алган жыйынтыктардан текшерилип жаткан сандардын чоң же кичинелигинен эмес, ал сандардын факторлорунун ортосундагы аралыктын чоң же кичинелигинен алгоритмдин иштөө убактысынын өзгөрүп жаткандыгын байкайбыз. N дин факторлору жакын болгондо (p жана q нун бит саны тең болгондо) жана N өзү 40-50 биттик сан болсо, Ферма алгоритминин иштөө убактысы 7.6445 секунданы түзүп, ал эми p саны 32 биттик сан жана q саны 7 ден 17 битке чейинки сан болгондо ошол эле 40-50 биттик N дин факторизациялоосу орто эсеп менен 1307.184 секунда алды. Бул жыйынтыкты Формула (2) де көргөнүбүздөй факторлордун жакын болгондо аларды табуу аз убакыт алганы, башкача айтканда

$$N = (x + y)(x - y)$$

жана x менен y тез табуу менен түшүндүрүлөт. Алгоритмдин мындай иштөө принцибинен, анын да тривиалдык бөлүү алгоритми сыяктуу детерминистик алгоритмдердин катарында экендигин билүүгө болот. А бирок, бул алгоритмди колдонуу да, RSA системасында колдонулган чоң сандар үчүн ыңгайсыз болуп саналат. Анын эң негизги себеби - RSA системасында 100×15 биттик катыштагы жөнөкөй сандардын колдонулушу. Мындай катыштагы кичинекей сандар үчүн да Ферма алгоритми эң жаман чечим болуп калат.

Поллард *rho* алгоритминин бардык учурда эң бат иштегени жыйынтыктардан көрүнүп турат. Поллард *rho* алгоритми N саны 40-50 биттик сан жана факторлору бири бирине жакын болгондо 0,01 секундана факторлорду таба алса, факторлору алыс болгон учурда 0,001 секундада иштеген. Ферма алгоритми факторлору жакындаган сайын тез иштесе, Поллард *rho* алгоритми факторлордун ара байланыш 100×15 ден 100×59 болгон сандарды тез ажырата алганын көрдүк. Бул алгоритмдин чын эле эң оптималдуу алгоритм экендигин тастыктоо үчүн жүргүзүлгөн ар түрдүү текшерүүлөрдүн натыйжасына көнүл буруу керек. Себеби, теория боюнча бүтүн сандардын факторизациясы чоң сандар үчүн жогорку татаалдыктагы маселе болуп эсептелет жана аны кыска убакытта чечүүчү белгилүү бир да жол жок. Демек, Поллард rho алгоритми белгилүү болгонуна карабастан, теорияда ал факторизация маселесин чечүүчү ийгиликтүү чечим катары саналбайт. Анын себептеринин бири – алгоритм ыктымалдуу алгоритмдердин катарын толуктайт. Себеби,

алгоритм ар дайым эле жыйынтык чыгарып бере бербейт. Кээ бир учурларда факторлорго ажыратып жаткан сандын чын эле татаал сан болушуна карабастан, алгоритм анын факторлорун таба албайт. Жыйынтык катары берилген татаал сандын өзүн кайтарып берип коёт. Натыйжада ал сандын чын эле жөнөкөй сан же татаал сан экендиги аныкталбай калат. Биздин практикада 21 санынын факторлорун таба алган жок. Ал эми 21 санынын эки жөнөкөй сандын (3 жана 7) көбөйтүндүсү экендиги баарыбызга белгилүү.

Брент факторизациялоо алгоритми Поллард *rho* алгоритминин оптимизацияланган варианты экендиги теорияда белгиленген. Бирок, Жадыбал 4-1 жана Жадыбал 4-2де көрсөтүлгөн жыйынтыктарыбыздан бул алгоритмдин дээрлик колдонулган бүт алгоритмдерден жай иштегени байкалган. Ал эми, Жадыбал 4-3тө Тривиалдык бөлүү жана Ферма алгоритмдери жыйынтык бере албаган учурларда Брент алгоритминин факторлорду тапкандыгын көрө алабыз. Албетте анын себептерини бири – бул алгоритм Поллард *rho* алгоритминин оптимизацияланган варианты катары ойлоп табылгандыгынан ыктымалдуу алгоритм болуп саналат.

Изилдөөдөнүн материал чектөөлөр болгондуктан өтө чоң сандар үчүн алгоритмдердин иштөө убакытын эсептей ала албадык. Андыктан чыныгы RSA системаларында колдонула турган 1024×180 сандарын текшере алганыбыз жок. Ошондуктан адабиятта Поллард *rho* нун модификациясы болгон Брент алгоритминин чоң сандар үчүн чындыгынды кандай иштешин текшере албадык. Келечекте бул алгоритмдерди кубаттуулугу жогору болгон компьютерлерде жана параллель иштөө методдорун колдонуп текшерүүнү каалайбыз.

АДАБИЯТТАР

[1] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems", *Communications of the ACM* 21.2 ,pp. 120-126, 1978.

[2] O. M. Rabin, “Digitalized signatures and public-key functions as intractable as factorization”, *MIT Technical Report* TR-212, 1979.

[3] P. Smith and M. J. J. Lennon, “LUC: A new public key system”, Tech. report, 1993.

[4] D. Whitfield and M. E. Hellman, "New directions in cryptography", *Information Theory, IEEE Transactions on* 22.6, pp. 644-654, 1976.

[5] T. ElGamal, “A public-key cryptosystem and a signature scheme based on discrete logarithms”, IEEE Transactions on Information Theory IT-31, pp.10–18, 1985.

[6] D. R. Stinson, “Cryptography: theory and practice”. CRC press, 1995.

[7] T. Okamoto and S. Uchiyama, “A new public-key cryptosystem as secure as factoring”, Lecture Notes in Computer Science 1403, pp. 308–318, 1998.

[8] D. Knuth, “The Art of Computer Programming”, Volume 2: Seminumerical Algorithms, Third Edition. Addison-Wesley. ISBN 0-201-89684-2. Section 4.5.4: Factoring into Primes, pp. 379_417, 1997.

[9] I. Kleiner, "Fermat: The founder of modern number theory." *Mathematics Magazine*, pp. 3-14, 2005.

[10] Mc. James, “Turning Euler's Factoring Method into a Factoring Algorithm", in Bulletin of the London Mathematical Society; issue 28 (volume 4), pp. 351-355, 1996.

[11] C. Pomerance, ”The quadratic sieve factoring algorithm”, In Proc. of the EUROCRYPT 84 Workshop on Advances in Cryptology: theory and Application of Cryptographic Techniques. Springer-Verlag New York, pp. 169-182, 1985.

[12] A. K. Lenstra, H.W. Lenstra, M.S. Manasse and J.M. Pollard, “The number field sieve”, In Proceedings of the Twenty-Second Annual ACM Symposium on theory of Computing. ACM, New York, NY,pp. 564-572, 1990.

[13] H. Riesel, “ Prime numbers and computer methods for factorization” (2nd ed.), Birkhauser Verlag, Basel, Switzerland, Switzerland, 1994.

[14] H. C.Williams and J. O. Shallit, “Factoring Integers Before Computers, Mathematics of Computation 1943–1993: a half-century of computational mathematics” (Providence) (Walter Gautschi, ed.), American Mathematical Society, pp. 481–531, 1991.
[15] J. M. Pollard, "Monte Carlo methods for index computation ($mod\ p$)."*Mathematics of computation* 32.143, pp. 918-924, 1978.
[16] R. P. Brent, and J. M. Pollard, "Factorization of the eighth Fermat number", *Mathematics of Computation* 36.154, pp. 627-630, 1981.
[17] Z. Du, GMP Install Instruction for Windows Platform, 2006. [E-documentation] Available: http://cs.nyu.edu/~exact/core/gmp/index.html.