ҚАЗАҚСТАН РЕСПУБЛИКАСЫ ҒЫЛЫМ ЖӘНЕ БІЛІМ МИНИСТРЛІГІ

Қ.И.Сәтбаев атындағы Қазақ ұлттық техникалық зерттеу университеті

Автоматика және ақпараттық технологиялар институты

«Программалық инженерия» кафедрасы

Жұманиязұлы Ноян

RSA Деректерді шифрлау жүйесіне арналған программалық құралдар пакетін құру

ТҮСІНДІРМЕ ЖАЗБА

дипломдық жобаға

5В070400 – «Есептеу техникасы және бағдарламалық қамтамасыз ету»

Алматы 2022

ҚАЗАҚСТАН РЕСПУБЛИКАСЫ ҒЫЛЫМ ЖӘНЕ БІЛІМ МИНИСТРЛІГІ

Қ.И.Сәтбаев атындағы Қазақ ұлттық техникалық зерттеу университеті

Автоматика және ақпараттық технологиялар институты

«Программалық инженерия» кафедрасы

ҚОРҒАУҒА ЖІБЕРІЛДІ
ПИ кафедрасының меңгерушісі
физ.-мат. ғыл.канд, профессор
_____А.Н.Молдагулова
"23" 05 2022 ж.

ТҮСІНДІРМЕ ЖАЗБА
дипломдық жобаға

Тақырыбы:

5В070400 – «Есептеу техникасы және бағдарламалық қамтамасыз ету» мамандығы

Орындаған Жұманиязұлы Н.

Рецензент
К.ф.-м.н., старший
преподаватель кафедры
Информатики КазНУ
_____ Уалиева И.М.
"23" 05 2022 ж.

Ғылыми жетекші
Магистр информационных систем,
Сениор-лектор
_____Бекарыстанқызы А.
"16" 05 2022 ж.

Алматы 2022

ҚАЗАҚСТАН РЕСПУБЛИКАСЫ ҒЫЛЫМ ЖӘНЕ БІЛІМ МИНИСТРЛІГІ

Қ.И.Сәтбаев атындағы Қазақ ұлттық техникалық зерттеу университеті

Автоматика және ақпараттық технологиялар институты

«Программалық инженерия» кафедрасы

БЕКІТЕМІН
ПИ кафедрасының меңгерушісі
физ.-мат. ғыл.канд, профессор
_____А.Н.Молдагулова
"23" 05 _____2022 ж.

Дипломдық жобаны орындауға
ТАПСЫРМА

Білім алушы *Жұманиязұлы Ноян*
Тақырыбы: *RSA деректерді шифрлау жүйесіне арналған программалық құралдар пакетін құру*
Академиялық мәселелер жөніндегі проректор бұйрығының
№489-ПВ "24" 12 2021 ж. шешімімен бекітілген.
Орындалған жобаның өткізу мерзімі "25" 05 2022 ж.
Дипломдық жобаға берілген бастапқы мәліметтер: *RSA алгоритмі, программалау тілдері: Python, C++, C#. ER диаграммасы түріндегі ақпаратты сақтауға арналған деректер қорының сипаттамасы.*
Дипломдық жобада әзірленетін сұрақтар тізімі:
а) RSA алгоритмі негізінде шифрлау/шифрды шешу мәселесінің жай-күйіне аналитикалық шолу
б) Python, C++, C# программалау тілдерінде RSA алгоритмін программалау
в) қолданушы интерфейсін жобалау және құру;
д) бағдарламалық пакетті құру, өңдеу, тестілеу.
Графикалық материалдар тізімі (міндетті суреттердің нақты көрсетілуімен):
презентацияның 20 слайдпен берілген құжат түрінде ұсынылған.
Ұсынылған негізгі әдебиеттер: *18 пайдаланылған әдебиеттер тізімінен*

Дипломдық жобаны орындау
КЕСТЕСІ

Бөлімдердің атаулары, зерттелген мәселелердің тізімі	**Ғылыми жетекшіге және кеңесшілерге ұсыну мерзімі**	**Ескерту**
1. RSA алгоритмі негізінде шифрлау/шифрды шешу мәселесінің жай-күйіне аналитикалық шолу	17.01.2022	Жоқ
2. Тапсырманы орындау үшін бағдарламалық құралдарды таңдау	23.01.2022	Жоқ
3. Тапсырманы тұжырымдау, алгоритм мен программаны құру	10.02.2022	Жоқ
4. Программалық өнімді өңдеу және тестілеу	11.03.2022	Жоқ
5. Дипломдық жобаның компьютерлік жинағы және түсініктеме жазбаны дайындау	06.04.2022	Жоқ
6. Дипломдық жобаны қорғауға дайындау	22.04.2022	Жоқ

Дипломдық жұмыс бөлімдерінің кеңесшілерінің аяқталған жұмысқа қойған
қолтаңбалары

Бөлімдер атауы	**Кеңес берушілер (аты-жөні, тегі, ғылыми дәрежесі, атағы)**	**Қолтаңба қойылған мерзімі**	**Қолы**
Нормалық бақылаушы	Жекамбаева М.Н. PhD, ассоциацияланған-профессор	13.05.22	
Бағдарламалық бөлім	Марғұлан Қ. магистр, лектор	16.05.22г.	

Ғылыми жетекші _____ Бекарыстанқызы Ақбаян

Тапсырманы орындауға қабылдап алған студент _____ Жұманиязұлы Ноян

Күні «17» 11 2021 ж.

АНДАТПА

Дипломдық жұмыс RSA алгоритмін жүзеге асыруға арналған және осы алгоритм арқылы деректерді шифрлайтын бағдарламаны құру.

Түсіндірме жазба кіріспеден, теориялық бөлім, практикалық бөлім, қортынды және 20 дереккөз тізімінен тұрады.

Негізгі қарастырылған сұрақтар Интернетте қолданушы лар арасында мәліметтерді тасымалдаудың қауіпсіздігін қамтамасыздандыру. Бұл жұмыста криптографияға, RSA шифрлау жүйесінің алгоритміне ерекше көңіл бөлінген.

RSA шифрлау жүйесі. Бағдарлама коды Python және C++ тілдері негізінде жазылған.

Дипломдық жұмысты келесі логикалық бөліктерге бөлуге болады: алгоритмнің сипаттамасы мен мәні, көмекші шифрлау алгоритмдері, алгоритмді іске асыру және шифрлау бағдарламасын құру, тестілеу.

Дипломдық жұмыста RSA алгоритмі егжей-тегжейлі сипатталған. Алғашқыда оның криптографиядағы және интернет қауіпсіздігіндегі рөлі зерттелген. RSA алгоритмінің қолданылуы және жұмыс істеу жүйесі қарастырылған, ол туралы егжей-тегжейлі сипатталған RSA алгоритмімен шифрлау бағдарламасының жұмыс істеу принципі, неліктен осы платформа таңдалды және бағдарлама қалай жұмыс істейді.

Қолданылған технология деректерді шифрлауға, ол үшін RSA криптографиялық шифрлау жүйесін пайдалануға және оның шифрын шешуге мүмкіндік береді.

АННОТАЦИЯ

Дипломная работа предназначена для реализации алгоритма RSA и создания программы, которая шифрует данные с использованием этого алгоритма.

Пояснительная записка состоит из введения, теоретической части, практической части, заключения и списка из 20 источников.

Основными рассмотренными вопросами являются обеспечение безопасности передачи данных между пользователями в сети Интернет. В данной работе особое внимание уделено криптографии, алгоритму системы шифрования RSA.

Система шифрования RSA. Программный код основан на языках Python и C++.

Дипломную работу можно разделить на следующие логические части: описание и суть алгоритма, вспомогательные алгоритмы шифрования, реализация алгоритма и создание программы шифрования, тестирование.

В дипломной работе дано подробное описание алгоритм шифрования RSA. Первоначально изучалась его роль в криптографии и в интернет-безопасности. Описано применение и работа алгоритма RSA, подробно описан принцип работы программы шифрования RSA, дано обоснование выбора этих платформ и описание работы программы.

Используемая технология позволяет шифровать данные, использовать систему криптографического шифрования данных RSA и дешифровать их.

ANNOTATION

The dissertation is designed to implement the RSA algorithm and create a program that encrypts data using this algorithm.

The explanatory note consists of an introduction, a theoretical part, a practical part, a conclusion and a list of 20 sources.

The main issues under consideration are ensuring the security of data transmission between users on the Internet. In this paper, special attention is paid to cryptography, the algorithm of the RSA encryption system.

The RSA encryption system. The program code is based on Python and C++ languages.

The dissertation can be divided into the following logical parts: description and essence of the algorithm, auxiliary encryption algorithms, implementation of the algorithm and creation of an encryption program, testing.

The dissertation provides a detailed description of the RSA encryption algorithm. Initially, its role in cryptography and Internet security was studied. The application and operation of the RSA algorithm is described, the principle of operation of the RSA encryption program is described in detail, the rationale for choosing these platforms and the description of the program are given.

The technology used allows you to encrypt data, use the RSA cryptographic data encryption system and decrypt them.

МАЗМҰНЫ

	Кіріспе	7
1	Криптографиялық файлдарды қорғауға аналитикалық шолу	9
1.1	Криптографияның міндеттері	9
1.2	Ақпаратты криптографиялық қорғау құралдары	10
1.3	Криптожүйелерге қойылатын талаптар	11
1.4	Шифрлау режимдері	12
2	Симметриялық және асиметриялық криптожүйелер	16
2.1	Симметриялық криптожүйені шифрлау	16
2.2	Асиметриялық криптожүйені шифрлау	17
2.3	Асиметриялық RSA алгоритмін шифрлау және дешифрлау	19
2.4	Ашық кілттегі криптожүйені криптоталдау арқылы шабуыл жасау	22
2.5	RSA шифрлеу алгоритмін UML-да моделдеу	25
3	Шифрлардың алгоритмін құру және оны программалау	30
3.1	RSA криптографиялық алгоритмін Python тілінде программалау	30
3.2	Эйлер функциясы және Евклид алгоритмі	31
3.3	RSA криптографиялық деректерді шифрлау алгоритмі	32
3.4	Dev-C++	34
3.5	Программаның негізгі функциялары мен модульдері	36
3.6	RSA деректерді шифрлау жүйесінің программалық пакетін құру	43
	Қорытынды	46
	Пайдаланылған әдебиеттер тізімі	49
	Қосымшалар	50

КІРІСПЕ

Ақпараттық қоғамда барлық ақпараттық объектілер – мәтін, графика, құжаттар және бейнематериалдар – осы ақпараттың барлығы деректерді сақтау құрылғыларында сақталады. Осылайша, бұл ақпаратты рұқсат етілмеген тұлғалардың енуінен сақтау мәселесі туындайды. Бұл мәселені деректерді шифрлау арқылы шешуге болады.

Тақырыптың өзектілігі көп уақытты қажет етпейтін және ең тиімді криптографиялық жүйе болып табылатын тиімді шифрлау және аутентификация әдістерін пайдалана отырып, программалық өнімді құру қажеттілігімен түсіндіріледі. Ақпараттық технологиялардың дамуымен ақпараттық қауіпсіздіктің ең тиімді жүйелерін құру қажеттілігі туындады.

Криптографиялық программалық өнімдердің арасында eToken RuToken өнімдерін, соның ішінде тегін программалық қамтамасыз етуді – PGP, TrueCrypt пайдалануға болады. Бұл программалардың бірқатар кемшіліктері бар - олар ақылы немесе бетбелгілерді қамтуы мүмкін шетелдік өнімдер. Сонымен қатар, бұл программалық құралдар құжаттамалық қолдаусыз. Сондықтан, моральді тұрғыда ескірген шифрлау алгоритмі бар программалық жасақтаманы пайдалану мәселесі тұрақты жаңартуды қажет ететін неғұрлым жетілдірілген шифрлау алгоритмдерін қолдану арқылы шешіледі. Сонымен қатар, құрылған программалық өнім тұрақты түрде жаңартылуды талап етеді, ал ол үшін құрылған программаны қайтадан құру қажеттілігі туындайды, сондықтан RSA алгоритмімен жұмыс істейтін программаны қолданушының өзі қалаған программалау тілінде құру өзекті мәселе болып табылады.

RSA алгоритмі әртүрлі веб-браузерлерде, электрондық пошталарда, VPN желілерінде, чат бөлмелерінде және басқа байланыс арналарында қолданылуын жалғастыруда. RSA жиі VPN клиенттері мен VPN серверлері арасында қауіпсіз қосылымдар жасау үшін қолданылады. OpenVPN сияқты протоколдарда TLS қол алысулары кілттерді алмасу және қауіпсіз арна құру үшін RSA алгоритмін пайдалана алады.

USB кілті түріндегі көп сатылы қауіпсіздік жүйесін аутентификациялау үшін криптографиялық программалық қамтамасыз ету іс жүзінде қолданылмайды, бұл осы криптографиялық жүйелердің негізгі кемшілігі болып табылады. Аутентификация үшін құпия сөзді пайдалану азайып барады. Сондықтан ақпараттық ресурстарға қол жеткізудің оңай жолы зияткерлік меншіктің қауіпсіздігіне қауіп төндіреді. Құпия сөздің аутентификациясы АТ жүйелеріндегі алғашқы кедергілердің бірі болып табылады. Соңғы жылдары бұл жүйе құпия сөзі аутентификациядағы алғашқы қадам болып табылады. Құпия сөзбен қорғаудың басты артықшылығы - оның қарапайымдылығы. Осылайша, ақпараттық жүйелердегі ақпараттың ағуы әлсіз парольдерді қолданумен байланысты. Құпия сөз жүйесі - зиянкестер белсенді түрде қолданатын ең осал жер.

Құпия сөз жүйесі ақпарат қауіпсіздігін қамтамасыз етуде тығырыққа тіреледі. Бірақ, күрделі құпия сөздерді пайдалану оларды адамның жадында

сақтаудың қиындығымен байланысты. Осылайша, олар электронды ақпарат түрінде сақталады, бірақ бұл жағдайда маңызды емес, қызметкердің жабық жазу кітапшасында Логин/пароль жұбы сақталады.

Қазіргі уақытта көп факторлы аутентификация кеңінен қолданыс табуда. Көп факторлы аутентификацияны пайдалану, ең болмағанда, қауіпсіз және қауіпсіз емес байланыс арналарының ақпараттық жүйелеріне қосылатын немесе дербес жұмыс станциясымен тікелей жұмыс істейтін қолданушы лар үшін ақпаратты қолдану қауіпсіздігін айтарлықтай арттырады.

Жұмыстың мақсаты блоктық шифрлау алгоритмдерін құрудың заманауи әдістерін талдауды және оның негізінде программалық құралды құру үшін шифрлау алгоритмін құруды және сынауды талап ететін ақпаратты криптографиялық қорғауды іске асыратын программалық қамтамасыз етуді әзірлеу болып табылады.

Зерттеу мақсаты. Әртүрлі алгоритмдердің сипаттамаларын зерттеу; Қорғау әдістерін әзірлеу; RSA алгоритмін қолдану мүмкіндігін талдау және таңдау; RSA мәліметтерді шифрлау жүйесі негізінде программалық құралды құру.

Зерттеу пәні. Криптографиялық алгоритмдер: ашық кілтті деректерді шифрлау алгоритмі, RSA шифрлау алгоритмі.

Зерттеу нысаны: программалау тілдері, ақпаратты криптографиялық қорғауды қамтамасыз ететін программалық қамтамасыз етуді әзірлеу құралдары.

Әзірленген жүйе RSA алгоритмдері мен екі сатылы аутентификацияны пайдалана отырып, үшінші тұлғалардың осы деректерге қол жеткізуін шектейді.

1 Криптографиялық файлды қорғауға аналитикалық шолу

1.1 Криптографияның міндеттері

Ақпараттың қауіпсіздігін қамтамасыз ету кезінде рұқсат етілмеген тұлғалардың қол жеткізуін болдырмайтын басты мәселе ақпаратты қорғау және оны түрлендіру тәсілдері болып табылады. Криптографияның тарихы да, адам тілінің тарихы да көне. Бастапқыда жазу криптографиялық жүйе сияқты болды, өйткені ежелгі уақытта оны тек таңдаулылар ғана иеленді. Мысал ретінде Ежелгі Египет, Ежелгі Үндістанның қасиетті кітаптарын келтіруге болады.

Криптографиялық төзімділік - шифрдың сипаттамаларына берілген атау, оның шифрды шешу кілтін білмей-ақ ашуға негізгі қарсылығы.

Криптология (kryptos – құпия, logos – ғылым) ақпаратты түрлендіру арқылы қорғау мәселесімен айналысады. Криптология криптография және криптоталдау болып екіге бөлінеді. Криптография мен криптоталдау дің мақсаттары тікелей қарама-қарсы.

Криптография ақпаратты түрлендірудің математикалық әдістерін іздеумен және зерттеумен айналысады.

Криптоталдау – кілттерді білмей-ақ ақпаратты шифрдан шығару мүмкіндігін зерттеу.

Алфавит - ақпаратты кодтауға арналған белгілердің шектеулі жиынтығы. Мәтін – алфавит элементтерінің жиынтығы. Шифрлау – бастапқы кодты түрлендіру, яғни. ашық мәтін шифрланған мәтінмен ауыстырылады [1].

Қазіргі криптография төрт негізгі бөлімді қамтиды:

- симметриялық криптожүйелер;
- ашық кілті бар криптожүйелер;
- электрондық қолтаңба жүйелері;
- негізгі басқару.

Криптографиялық әдістерді қолданудың негізгі бағыттары байланыс арналары (мысалы, электрондық пошта) арқылы құпия ақпаратты беру, жіберілген хабарламалардың аутентификациясы, ақпаратты тасымалдаушыларда (құжаттар, мәліметтер базасы) шифрланған түрде сақтау болып табылады.

Криптография ақпаратты тек кілт қол жетімді болған жағдайда оқуға (қалпына келтіруге) болатындай түрлендіруге мүмкіндік береді.

Шифрланатын және шифрын шешетін ақпарат ретінде кейбір алфавит негізінде құрастырылған мәтіндер қарастырылады [2].

1.2 Ақпаратты криптографиялық қорғау құралдары

Криптография деректерді қорғаудың ең сенімді әдістерінің бірі болып саналады, өйткені ол ақпаратқа қол жеткізуді емес, ақпараттың өзін қорғайды. Криптографиялық түрлендірілген ақпарат жоғары құпияны сақтай отырып, жоғарырақ қорғау дәрежесіне ие.

Ақпаратты криптографиялық қорғау құралдары (CIPF) - ақпаратты криптографиялық түрлендіру алгоритмдерін іске асыратын және байланыс арналары бойынша беру кезінде ақпаратты қорғауға және (немесе) ақпаратты рұқсатсыз кіруден қорғауға арналған аппараттық, программалық және аппараттық-программалық құралдар, жүйелер мен кешендер. оны өңдеу және сақтау.

Бұл жүйенің жалпы құрылымы 1.1-суретте көрсетілген.

1.1 - Сурет - Шифрлау схемасы

Шифрды шешу – шифрлаудың кері процесі. Кілт негізінде шифрланған мәтін түпнұсқаға түрлендіріледі (1.2-сурет).

1.2 - Сурет - Шифрды шешу схемасы

Кілт – мәтіндерді тікелей шифрлау және дешифрлау үшін қажетті ақпарат. Криптографиялық беріктік – кілтті білмей-ақ (яғни криптоталдау) шифрдың шешуге төзімділігін анықтайтын шифрдың сипаттамасы.

Алфавит - ақпаратты кодтау үшін қолданылатын белгілердің ақырлы жиынтығы.

Мәтін – алфавиттік элементтердің реттелген жиынтығы.

Шифрлау – трансформациялық процесс: ашық мәтін деп те аталатын бастапқы мәтін шифрланған мәтінмен ауыстырылады [4].

Криптографиялық түрлендіру сәйкес алгоритммен және негізгі параметрдің мәнімен анықталады. Ақпаратты қорғау үшін шифрлаудың тиімділігі кілттің құпиясының сақталуына және шифрдың криптографиялық беріктігіне байланысты.

1.3 Криптожүйелерге қойылатын талаптар

Шифрлау және кодтау процесіндегі деректер аппараттық және программалық құрал болуы мүмкін. Программалық жасақтаманы енгізу кең таралған және біршама икемділікке ие.

Криптографиялық ақпаратты қорғау жүйелері үшін келесі талаптар орындалады:

- шифрланған хабарлама кілт болған жағдайда ғана талап етілуі керек;
- кілтті анықтау операцияларының саны, шифрланған хабарламаның және мәтіннің пайдаланылған фрагменттері мүмкін болатын кілттердің жалпы санынан кем болмауы керек;
- кілттердің барлық мүмкін түрлері бойынша ақпараттың шифрын ашу үшін қажетті операциялардың санының төменгі шегі болуы және қазіргі заманғы компьютердің мүмкіндіктерінен шығуы керек (желілік есептеулерді қолдану мүмкіндігімен байланысты);
- шифрлау алгоритмін білу қорғаныс сенімділігіне әсер етпеуі керек;
- кілттің азғантай өзгеруі, тіпті бір кілтті пайдаланған кезде де шифрланған хабарлама пішінінің айтарлықтай өзгеруіне әкелуі керек;
- шифрлау алгоритмінде құрылымның элементтері бірдей болуы керек;
- шифрлау процесі кезінде хабарламаға қосымша разрядтар енгізіледі, шифрлық мәтінде толық және сенімді түрде жасыртылуы керек;
- шифрланған мәтіннің ұзындығы түпнұсқа мәтіннің ұзындығына тең болуы керек;
- қарапайым болуы және шифрлау процесінде ретімен қолданылатын кілттер арасында тәуелділікті орнату мүмкіндігі болуы керек;
- кез келген кілт, көптеген мүмкіндіктер ақпаратты сенімді қорғауды қамтамасыз етуі керек;
- алгоритм программалық және аппараттық қамтамасыз етуді жүзеге асыруға мүмкіндік беруі керек.

Сонымен, кілттің ұзындығын өзгерту, шифрлау алгоритмінің сапасының нашарлауына әкелмеуі керек.

1.4 Шифрлау режимдері

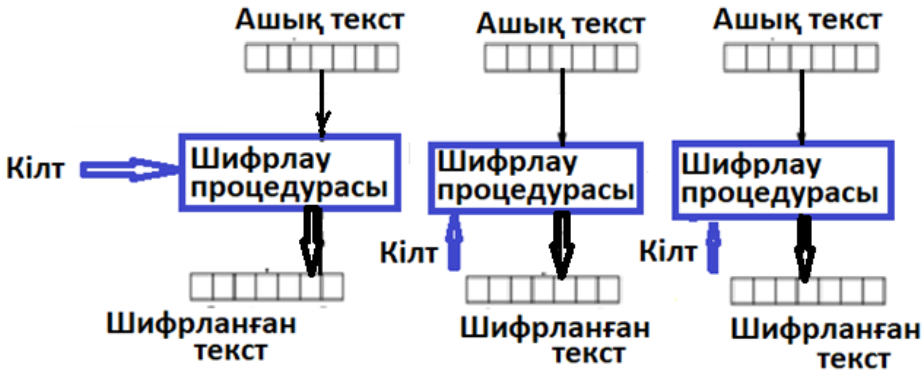
Төменде бес шифрлау режимі АҚШ NIST жариялаған және «Блоктық шифрмен шифрлау режимдері бойынша ұсыныстар» деп аталатын құжатта [5] ұсынылып талқыланған. Бұл режимдер:

- ECB (Electronic Code Book) – электронды код кітабы;
- CBC (Cipher Block Chaining) – блоктардың шифрлық мәтінді тізбегі;
- CFB (Cipher Feed Back) – шифрлы мәтінді кері жүктеу;
- OFB (Output Feed Back) – шығыс деректерін кері жүктеу;
- CTR (Counter) – санауышпен шифрлау.

Бұл жағдайда шифрлаудың алғашқы төрт режимі негізгі болып табылады. Соңғы режим (CTR) кейінірек пайда болды - бірінші шифрлау стандарты қабылданғаннан кейін үш жылдан кейін [1]. Айта кету керек, ГОСТ 28147-89-да OFB режимі жоқ және режимдердің атаулары NIST классификациясымен келесідей корреляциялануы мүмкін [1]:

- ECB (Electronic Code Book) – қарапайым ауыстыру режимі;
- CBC (Cipher Block Chaining) – имитация енгізуді генерациялау режимі;
- CFB (Cipher Feed Back) – кері байланысы бар гамма режимі;
- CTR (Counter) – гамма режимі.

Бұл режим 20-ғасырдың басында шифрланған хабарламаны жіберу үшін агенттер пайдаланатын режимнің электронды аналогы болып табылады. Агент жазу кітапшасын алды, оның әрбір бетінде бірегей реттілік – хабарлама шифрланған код бар. Қолданғаннан кейін мұндай парақ дәптерден жұлып алынып, жойылды. Қажет болса, жыртылған беттерде пайдаланылмаған код қалмауы үшін хабарлама толықтырылады. Қабылдаушы тарапта блокноттың көшірмесі болады, сондықтан беттер синхронды түрде пайдаланылған жағдайда, бұл шифрлау режимі хабарламалардың шифрлануын да, шифрын ашуды да қамтамасыз етеді. ECB-де шифрлау үшін код кітапшасының бір бетін пайдалану CIPHK функциясымен енгізілген деректерге түрлендіруді қолданумен сәйкес келеді, ал шифрды шешу үшін - CIPH-1K функциясын қолданған. Синхрондау үшін екі тарап K құпия кілтінің мәнін келісіп алу керек [5]. Жалпы алғанда, бұл режим ең қарапайым және хабарламаларды шифрлау үшін блоктық шифрды қолданудың бірінші жолы болып табылады. Ол 1.3-суретте көрсетілген.



1.3 - Сурет - Электрондық код кітапшасының шифрлау режимі

1.3-суреттен көрініп тұрғандай, бүкіл ECB алгоритмі CIPHK және CIPH-1K функцияларын сәйкесінше шифрлау және дешифрлау үшін хабарға және шифрлық мәтінге тікелей қолданудан тұрады. Шифрлауды келесідей көрсетуге болады:

Cj = CIPHK(Pj),

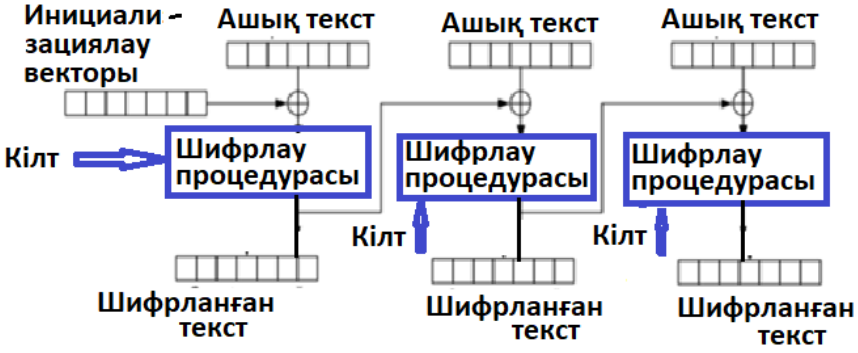
мұндағы: j=1…n;

Pj келесі, j-ші ашық мәтіндік блок;

Cj – шифрланған мәтіннің келесі j-ші блогы.

Осылайша, шифрлау CIPHK және = CIPH-1K функциялары үшін кіріс/шығыс деректерінің өлшеміне сәйкес блоктарда орын алады. Блоктар бөлек және бір-бірінен тәуелсіз шифрланады, бұл оны параллель орындауға мүмкіндік береді. ECB режимінің бұл артықшылығы және оның қарапайымдылығы екі маңызды кемшіліктермен жасырылады. Біріншісі, хабарламаның ұзындығы блоктық шифрдің кіріс блогының ұзындығының еселігі болуы керек, яғни бүкіл хабарлама не мұндай блоктардың бүтін санына бөлінуі мүмкін немесе соңғы блокты қандай да бір жолмен толықтыру қажет. Екінші кемшілік одан да маңызды - егер бірдей реттілік енгізілсе, шифрлаудан кейін шығу да бірдей болады және бұл хабарламаның мазмұнын талдауға түсінік бере алады.

CBC шифрлау режимінде хабарламаның барлық блоктары шифрлық мәтінге сәйкес біріктіріледі. Бұған қалай қол жеткізілгенін осы шифрлау режимін суреттейтін 1.4-суреттен көруге болады.



1.4 - Сурет - Шифрлау режимі шифр-мәтінді блок тізбегі

1.4-суреттен көрініп тұрғандай, CIPHK енгізу функциясы үшін шифрлау алгоритмі әр уақытта келесі ашық хабарлама деректер блогы функциясының 2-модулінің қосындысы және алдыңғы блок үшін CIPHK шығыс ретінде қызмет етеді. Келесі CIPHK-дегі функционалдық блоктың шығысы тікелей CBC алгоритмінің шығысына өтетіндіктен, яғни бұл келесі блокты кодтау үшін бір уақытта бір функцияға енгізілген шифрлы мәтін блогы, бұл шифрлық мәтін блогында орын алады. Бірінші ашық деректер блогы инициализация векторына қосылады [6]. Бұл ретте бұл инициализация векторы сеанстың басында жіберушіге де, қабылдаушыға да белгілі болатынын түсіну жеткілікті (сондықтан оны жиі жай ғана инициализация векторы, шифр мәтіні деп атайды). Шифрды шешу сәйкесінше кері тәртіпте жүреді – бірінші пайдаланылған шифрланған мәтін CIPHK функциясын орындауы керек, содан кейін келесі ашық мәтін блогының алгоритмін шығару үшін алдыңғы шифрлық мәтін блогымен жинақталады. Ашық мәтіннің бірінші блогы қайта инициализациялау векторымен қайта құрылады. Осылайша, барлық алгоритмді теңдеулер түрінде келесідей көрсетуге болады:

$$C1 = CIPHK(P1 \text{ xor } IV);$$

$Cj = CIPHK(Pj \text{ xor } Cj\text{-}1)$,

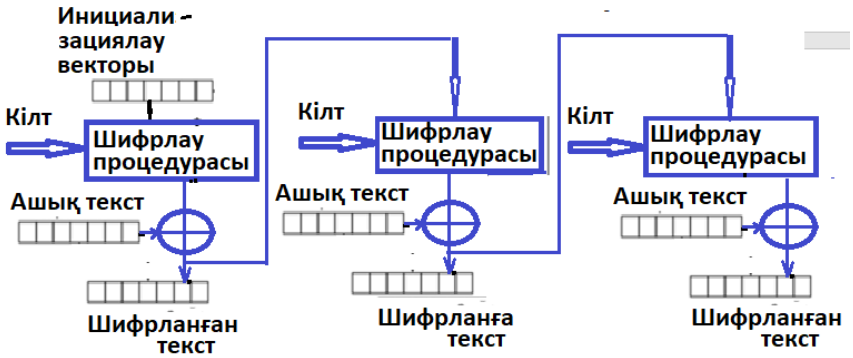
мұндағы: $j = 1 \ldots n$;

IV – инициализация векторы;

Pj келесі, j-ші ашық мәтіндік блок;

Cj келесі, j-ші шифрланған мәтін блогы.

CFB режимі кіріс CIPHK деректерін кодтау функциясы алгоритмнің алдыңғы итерациясынан шифрланған мәтін негізінде құрылған кезде іске қосылады, сондықтан алгоритм итерацияларының параллель орындалуы мүмкін емес. Дегенмен, CFB шифрын шешу режимінде, толық шифрлық мәтін қолжетімді болса, толық шифрлық мәтіннің барлық бөліктерін бір уақытта алуға болады (1.5-сурет).

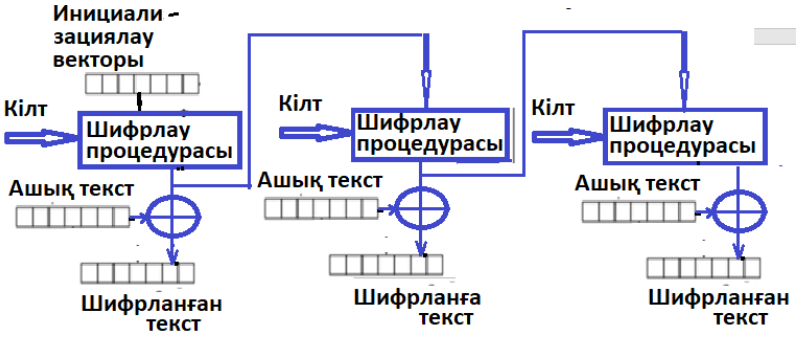


1.5 - Сурет - Шифрлық мәтінді кері жүктеуді шифрлау режимі

OFB режимі, CFB, ағындық хабарлама ретінде, яғни CIPHK функциясы мәтіннің бір бөлігін қорытындылайтын алдын-ала жинақтау алгоритмінде

шақырылады. Дегенмен, бұл жолы CIPHK кірісі алдыңғы итерациядағы шифрланбаған мәтін болып табылады және бұл жай ғана шығыс.

Иллюстрация 1.6-суретте көрсетілген.



1.6 - Сурет - OFB шифрлау режимі - шығыс кері жүктемесі

Төменде OFB режимінде шифрды шешу және дешифрлау теңдеулері берілген:

OFB шифрлауы:

I1 = IV,

Ij = Oj-1, мұндағы: j = 2…n.

Oj = CIPHK(Ij), мұндағы: j = 1,2…n.

Cj = Pj xor Oj, мұндағы: j = 1,2..n-1.

Cj = Pj xor MSBr(Oj), мұндағы: IV – инициализация векторы;

Pj келесі, j-ші ашық мәтіндік блок;

Cj – келесі, j-ші шифрланған мәтін блогы;

MSBr (X) – X екілік санының r ең маңызды биттері [5].

Хабардың соңғы, мүмкін толық емес блогы үшін CIPHK функциясы шығысының дәл осы блокта қанша бит болса, сонша бит пайдаланылады. Бұл режимде, алдыңғылардан айырмашылығы, хабарламаның ұзындығы шифрлау кезінде және ең бастысы, жіберу кезінде өзгеріссіз қалады [6].

2 Симметриялық және асиметриялық криптожүйелер

2.1 Симметриялық криптожүйені шифрлау

Шифрлаудың негізгі екі түрі бар: және асиметриялы (ашық кілт түрі) шифрлау.

Симметриялы шифрлау құпия кілтті шифрлар, себебі шифрлау мен шифрды ашу үшін бір ортақ кілт қолданылады.

2.1 - сурет – Шифрлау сұлбасы

Бұл шифрлау түрі пайдалану кезінде ақпараттың құпиялылығын және шифрын ашу үшін тұтынушыда кілт болу керек. Бұл оңай, әрі тез іске асатын шифрлау тәсілі. Жіберілген ашық мәтін жабық кілттің көмегімен шифрланған мәтінге айналады (2.2-сурет).

2.2 - Сурет – Симметриялық шифрлау

Симметриялық шифрлаудың жұмыс істеу үрдісі:

- генерациялау кілті. E,D (шифрлау мен шифрды ашу функциялары) алгоритмдері мен d шифрлау кілті таңдалады;

- шифрлау және ақпаратты жіберу. Бірінші қолданушы d кілтінің көмегімен екінші қолданушыға c(E(m, d)=c) шифромәтін ретінде жіберіледі;

- ақпараттың шифрын ашу. Екінші тұтынушы шифромәтінді c(D(c, d)=m) d кілті арқылы ашады.

Симметриялы криптографиялық алгоритмдердің ерекшеліктері:

- қайта орнату оңай, ең жеңіл шифрлаудың бірі болып саналады. Барлық хабарламаларды баған түрінде кестеге жазу керек. Жіберуші мен қабылдаушы алдын-ала кестенің көлемін ақылдасып алуы керек, ортақ кілт арқылы қайта орнату негіізінде;

- бірінші реттік кілт арқылы қайтадан орнату. Алдыңғы тәсілге ұқсас болады, тек бір ерекшелігі кестенің бағандары кілттік сөзбен өзгертіледі;

- екінші реттік қайтадан орнату. Құпиялылығын сақтау үшін шифрланған ақпаратты қайтадан шифрлауға болады. Мұны іске асыру үшін, екінші кестенің көлемі бағандар мен жолдардың ұзындығынан басқаша болатындай етіп таңдалады. Кестеледің өз ара қарапайым болғандары дұрыс. Сонымен қатар, кестені әртүрлі тәсілдермен толтыруға болады.

Симметриялы шифрдың артықшылығы мен кемшіліктері.

Артықшылығы:

- жоғары жылдамдық;
- іске асыру оңай;
- жақсы зерттелген.
- сәйкес келетін ұзақтыққа кілттің қысқа ұзындығы.

Кемшіліктері:

- үлкен желіде басқару қиын. Тез жылдамдықта кілттерді жіберу, сақтау, жаңартуды қажет етеді.

- кілттермен алмасу қиынға соғады. Екі жаққа ақпарат жіберген кезде құпия кілтті сұрау.

2.2 Асиметриялық криптожүйені шифрлау

Жіберуші мен қабылдаушының кілттері бар. Бұл кілттер бір-бірімен тығыз байланыста болады, сонда да әртүрлі. Мысалы шифрлаған ақпарат К1 кілт арқылы іске асса, шифрын ашу үшін К2 кілті қажет екені мәлім соның дәлелі 2.4 суретте көрсетілген. Немесе керісінше, бұл кілттердің бірі ашық, келесі құпия деп аталады.

2.3 - Сурет – Шифрлау сұлбасы

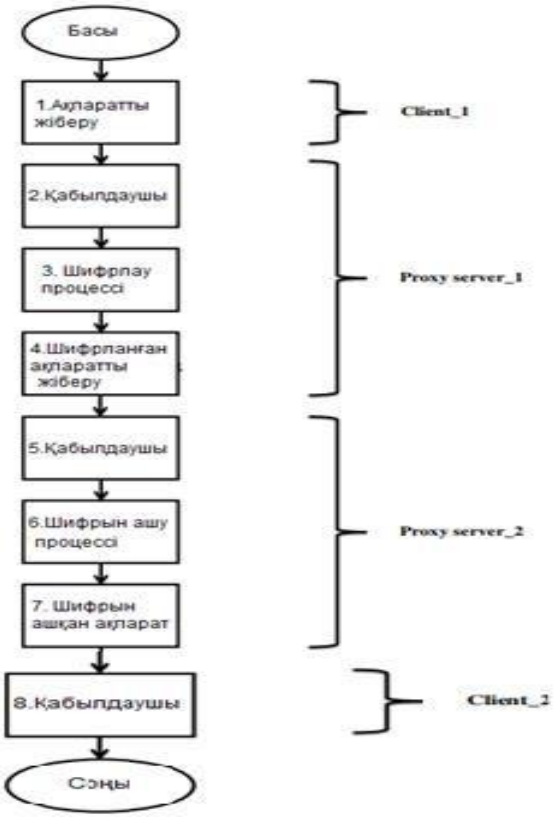
RSA деректерді шифрлау жүйесінің жұмыс істеу процесіне сипаттама:

- негізгі генерация жұпты кілттер. Бірінші тұтынушы (e,d) алгоримдері арқылы ашық кілт, (e,d) жабық кілт түрінде. Ашық арна арқылы екінші тұтынушыға е ашық кілтін жібереді;
- ақпаратты шифрлау мен жіберу. Екінші тұтынушы ақпаратты шифрлауды ашық е кілті арқылы іске асырады. E(m,c)=c бірінші тұтынушыға с шифромәтінді жібереді;
- хабарламаның шифрын ашу. Бірінші тұтынушы d жабық кілтінің көмегімен, D(c,d)=m шифромәтіннің шифрын ашады. Жұмысты іске асыру үшін алдымен екі жақта алдыңғы екі операцияны орындаулары қажет. Жабық кілт қорғалмаған арна арқылы жіберіле алмайды, сол себекті құпиялылығын сақтайды.
- View Report Card
- Courses

Артықшылығы:

- сенімді арна арқылы құпия кілтті қажет етпеуі;
- құпия кілтті тек бір жақ біле алады (симметриялы шифрлау кезінде екі жақта да белгілі болады);
- ашық және құпия кілттер белгілі уақытқа дейін жарамды (симметриялы шифрлау кезінде әр жіберген сайын жаңартып отыруы шарт);

-үлкен желілерде кілттер саны маңызды емес. Арна байланысын қорғау алгоритмы (2.4-сурет).



2.4 - Сурет - Арна байланысын қорғау алгоритмы

Асиметриялы шифрлеудің негізгі түрлері:

- RSA (Rivest-Shamir-Adleman) - Ривест-Шамир-Адлеман;
- DSA (Digital Signature Algorithm) - Шифрлық қолтанбамен алгоритм;
- EGSA (El-Gaman Signature Algorithm) - Эль-Гамальдің цифрлыэлектронды қолтаңба (ЭЦП) алгоритмі;
- ECC (Eliptic Curve Cryptography) - Эллипстік қисық криптографиясы;
- ГОСТ Р 34.10-94 - DSA-ге ұқсас Ресейлік стандарт;
- ГОСТ Р 34.10-2001 - ECC-ге ұқсас Ресейлік стандарт.

RSA деректерді шифрлеу жүйесінің кемшіліктері:

- алгоритмді өзгерту қиындық туғызады;
- кілттер ұзындығы өспелі;

Тәжірибе жүзінде асимметриялық шифрлеу басқа, себебі RSA деректерді шифрлеу алгоритмдері көптеген есептеу ресурстарды қажет етеді.

2.3 Асиметриялық RSA алгоритмін шифрлау және дешифрлау

Ақпаратты қорғау ақпараттық қауіпсіздіктің негізгі бөлігі болып табылады. Өнімділік, байланыс, транспорт, банктік істерде, барлық дерлік ақпарат алмасулар ақпарат алмасудың қарқындылығына, ақпараттың толықтығына, уақытылығына, сенімділігі мен қауіпсіздігіне байланысты.

Есептеу техникаларының қарқынды дамуына байланысты адамзат алдында ақпаратты автоматизациялау қажеттіліг туындады.

Ғаламтор бұл ғаламдық желі, дүние жүзіне таратылған мыңдаған компьютерлерді байланыстырушы желі. Екі компьютер өзара ақпарат алмасу кезінде жіберуші мен қабылдаушы арасында әртүрлі құрылғылар (маршрутизатор, коммутатор, т.б.) арқылы өтеді. Бірақ, мұндай құрылғылық желілердің қауіпсіздігіне ешқандай кепілділік жоқ, себебі, ақпараттың қай десте арқылы қабылдаушыға жететінін алдын-ала болжау мүмкін емес.

Десте жолында кездескен кедергілер негізінде олардың мазмұнын көруге немесе оларды өзгертуге болады. Мұның бәрі маңызды мәселелерді туғызады, маңыздылығы немесе берілетін деректердің құпиялығы жоғары болады.

Мұндай мәселерді шешу үшін қорғалған ақпараттық арналар қолданылады. Оларды туннель деп есептеуге болады. Туннель бір жағынан ақпаратты жинаса, екінші жағынан оны оқуға мүмкін болады.

Жіберілетін ақпарат жеткізілу барысында өзгеріске ұшырайды, сондықтан оны өзгертуге немесе қарауға болмайды. Бұл механизмді қолдану кезінде жабу қарастырылған, оның бүкіл жолында ақпарат және оны ауыстырудың мүмкін еместігі.

«Ақпаратты қорғау» шифрлау алгоритмінің негізгі бөлімі болып табылады. Бүгінгі таңда шифрлау алгоритмін екі негізгі классқа бөлуге болады: симметриялы және асиметриялы (ашық кілтпен қолданылатын алгоритм). Бағдарламаны шифрлаудың ең ыңғайлы түрі асиметриялы алгоритм. Бұл алгоритмдердің негізінде тек шифрлау жүйелері ғана емес, сонымен қатар электронды қолтаңба құрылатындығына байланысты. Электронды қолтаңба қазіргі таңда ақпараттық процесс болып саналады.

Асиметриялы алгоритмның ішінен RSA алгоритмы таңдалды. Үлкен бүтін санды факторизациялау есебі, есептеу күрделілігіне негізделген, бұл ашық криптографиялық алгоритм. RSA криптожүйесі шифрлауға қолайлы және цифрлы қолтаңбаға ыңғайлы ең алғашқы жүйе. Бұл алгоритм криптографиялық PGP, S/MIME, TLS/SSL, IPSEC/IKE және басқа да бағдарламаларда қолданылады.

Асиметриялы алгоритмды шифрлауда RSA қажет. Бұл бағдарлама кезең-кезеңмен және көрсетілген алгоритмның жұмысын нақты көрсетуі керек.

Функционалды бағдарламаны іске асыру үшін келесі мүмкіндіктер қамтамасыз етілуі қажет:

- алгоритм көбінесе үлкен бүтін сандармен жұмыс істейтіндіктен, қолданушының қалауы бойынша параметрлер мен кілт деректерін енгізу немесе оларды автоматты түрде құру мүмкіндігін қамтамасыз етеді;
- шифрлайтын/шифрын ашатын ақпарат міндетті түрде қажет;
- RSA алгоритмінде әрбір жұмыс атқарған деңгейі үшін қысқаша тоқталу қажет, орындалған әрекет туралы мәлімет қажет болған кітабына кіруге рұқсат болуы керек (анықтама бөлімінде);
- әр кезең үшін аралық ақпараттың шығуын ұйымдастыру ондық сан, он алтылық сан, байттар жиыны түрінде, олардың саны жадыда сақталады;
- уақыт интервалымен салыстыратын ақпаратты шифрлау мен шифрын ашу үшін уақытты есептеу жүйесі қажет болады;
- зертханалық жұмыс бойынша есеп материалдарын әзірлеп кою, есептерді сақтау мүмкіндігін қамтамасыз ету;
- мәтіндік хабарламаларды ғана емес, еркін түрдегі файлдардың да шифрлау мен шифрын ашуға арналған операцияларға мүмкіндік беру;
- RSA алгоритмі және оны жүзеге асыру кезінде қолданылатын қосымша алгоритмдер туралы барлық ақпараттарды қамтитын онлайн анықтамалық жүйені құру.

RSA алгоритмін жүзеге асыру. Жұмыстың нәтижесінде «Виртуалды тәжірибе: RSA асиметриялы алгоритмін шифрлау іске асырылады.

Хабар жіберуші мен қабылдаушы арасында операцияны жүзеге асыру үшін алдымен кілтті орналастыру қажет:

- екі p мен q кездейсоқ жай сандары таңдалып, $|p| \approx |q|$ шарты орындалуы керек.
- $N = p \cdot q$ есептеу.
- $\varphi(n) = (p-1) \cdot (q-1)$ есептеу.
- кездейсоқ бүтін $e < \varphi(N)$ санын таңдау және $(ged(e, \varphi)) = 1$ шартын қанағаттандырып, d бүтін санын есептеу,

$$ed = 1(\mathrm{mod}(N)), \quad (2.5)$$

Шифрлау үрдісі. Құпиялы хабарлама жіберу үшін, c шифрланған мәтін $m < N$ ұзындығын пайдалану.

$$c \leftarrow m^e(mod\ N), \quad (2.6)$$

Хабарлама жіберуші жағынан N санынан кіші кіріс сандар көрсетіледі.

Шифрды ашу

Шифрланған мәтіннің c шифрын ашу үшін, келесі формула шығады:

k-кейбір бүтін сан болғандықтан, шифрын ашу келесі формула арқылы жүзеге асады:

Ескере кететін жағдай, ол $m < N$, ол $m \in Z^*$ (N-нен кіші барлық сандар, мультикативты топтың бүтін сандарына жатады). $m \in Z^*$ шарты бұзылады, тек қана $m = up$ немесе $m = uq$ мұндағы $u < qu < p$ тең. Мұндай жағдайда хабарлама жіберуші $ged(m,$ N)-ның мағынасында N жай сандарға келтіруге болады.

Егер $m \in Z^*$ Лагранж теоремасы бойынша жіктелсе, онда функция келесідей болады:

Мысалы: Хабарлама жіберуші $N = 7 \cdot 13 = 91$, e=5 деп таңдады. Содан $\varphi(\mathrm{N}) = 6 \cdot 12 = 72$ шықты. (a,b)=(72, 5) жұбы деп ала отырып, келесідей сандарды алады:

$$72 \cdot (-2) + 5 \cdot 29 = 1$$

Басқаша айтқанда $5 \cdot 29 \equiv 1(mod\ 72)$. Хабарлама жіберуші құпия шифрлау ретінде 29 саның алып отыр. Хабарлама жіберуші $(N,$ e$) = (91, 5)$ жұбын RSA криптожүйесінде ашық кілт параметрі ретінде орнатады.

Хабарламаны қабылдап алушы кіріс хабарламаны $m = 3$ арқылы шифрлайды, келесі формуланы пайдалану арқылы:

$$\mathrm{c} = 3^5 = 243 \equiv 61(mod\ 91).$$

Шифрланған хабарлама 61 санын білдіреді.

Шифрын ашу үшін қабылдап алушы 61 санының шифрын ашады.

2.4 Ашық кілттегі криптожүйеге криптоталдау арқылы шабуыл жасау

X криптожүйесі Y шабылуына төтеп бере алады, ал Z шабылуына керісінше төтеп бере алмайды. Белсенді шабуылдар үш түрге бөлінеді:

- ашық мәтінге шабуыл (chosen-plaintext attack-CPA). Шабуылдаушы кіріс хабарламаны алып, шифрлаушы арқылы шифрланған мәтінді алады. Шабуылдаушының мақсаты бірнеше ашық кілт және шифрланған мәтін арқылы криптожүйені бұзу;

- шифрланған мәтін арқылы шабуылдау (chosen-ciphertext attack-CCA)/ Шабуылдаушы шифрланған мәтіннің шифрын ашу арқылы кіріс хабарламаға қол жеткізеді. Шабуылдаушының мақсаты бірнеше ашық және шифрланған мәтін арқылы криптожүйені бұзу;

- шифрланған мәтін арқылы адаптивті мәтінге шабуыл (adaptive chosen-ciphertext attack-CCA2) жасалады. Шифрды ашу қызметтері барлық шифрланған мәтіндерге ашық берілгенінен басқасына.

CCA шабуыл түрі.

Бұл шабуыл түрлерін келесідей сипаттауға болады:

- шабуылдаушы ашық мәтін арқылы шифрлау блогын иемденеді;

- шифрланған мәтін арқылы шабуылдаушы шифрлау блогынан алыстай береді, себебі бірнеше әрекеттен кейін блокты иемденуге жол жабылады, шабуылдаушы оның көмегінсіз керек мәтіннің шифрын мәтінсіз ашуы керек болады;

- адаптивті шабуыл кезінде шабуылдаушы шифрлау блогын иемденгенмен, алдыңғы жағдай секілді мәтіннің шифрын өзі ашуы керек болады. Бұл талап орынды, себебі криптожүйесіне шабуылдаудың керегі жоқ.

CPA мен CCA шабуылдары бастапқыдан құпия кілттегі белсенді криптоталдау криптожүйесіне арналған. Криптоталдау дың мақсаты шабуыл барысында алынған ашық және шифрланған мәтін арқылы шабуыл жасау. Кейін олар ашық кілтті криптоталдау криптожүйесіне бейімделді. Ашық кілтті криптожүйесінің үш негізгі ерекшеліктері бар, ол:

- ашық кілтті криптожүйеде шифрлау қызметтері оны кез келген қалаушы адам үшін қол жетімді етеді, өйткені ашық кілтке иелік ету бақылауды қамтамасыз етеді. Басқаша айтқанда, ашық кілті бар криптожүйелерге қарсы, кілттің криптожүйесіне жасалған кез-келген шабуылды CPA шабуылы деп атауға болады. Ашық кілттердің кез-келген криптожүйесі таңдалған қарапайым мәтінге негізделген шабуылдарға төтеп беруі керек, десекте ол аз болады;

- әдетте, ашық кілттегі криптожүйелер жабылу, ассоциация, гомоморфизм және т.б. қасиеттерімен үйлесімді алгебралық құрылымға ие.

Шабуылдаушы осы қасиеттерді қолдану арқылы және есептеулерді қолданып шифрланған мәтінді жойып жібере алады. Егер шабуылдаушы шифрлау блогына қол жеткізетін болса, оның есептеулері туралы түсінік беріп, тіпті бүкіл криптожүйені бұзуы мүмкін. Демек, ашық кілттік криптожүйелер CCA және CCA2 шабуылдарына осал болады;

- бір қарағанда, CCA шабуылы шабуылшының мүмкіндіктерін тым көп шектейді. Қолданбаларда шабуылға ұшыраған қолданушы (яғни, хабарламаны шифрлау үшін байланысқа түскен қолданушы) бұл туралы нақты білмейді. Сондықтан, қолданушы шифрды шешуді қашан тоқтату керектігін білмейді. Көбінесе, тұтынушы қателіген, хабарламаларды әрдайым шифрлауы қажет.

Екінші жағынан, кез-келген ашық төтеп беруі керек, өйткені қаскүнем таңдалған жалпы хабарламаларды өзі шифрлай алады. Осы себепті, біз CCA2 шабуылынан қорғану қарастырылады.

RSA шешуі келесідей болады. RSA криптожүйесі n композиттік бүтін сан модулін қолдана отырып, c шифрмәтінінің e дәрежесінің түбірін есептеу қиындықтарына негізделген CPA шабуылдарынан қорғайды.

$c \in Z^*$.

Нәтиже

сі:

$m \in Z^*$ жалғыз бүтін сан, шартын қанағаттандырушы.

Басқада алгоримдер секілді ашық кілттегі криптожүйенің төтеп бере алуын қамтамасыз ету, RSA шешімі қиындығы да дұрыс параметр табуына байланысты.

Шешуші алгоритм RSA ықтималды полиномиялық алгоритм A деп аталатын $\epsilon > 0$ шартты:

мұндағы, A - мәліметтерді енгізу алгоритмы.

Сонымен қатар, RSA мәселесі шешілмеген жағдайда, шифрлау кезінде пайдаланылатын экспонент ұсынылған алгоритмге енеді.

Күшті RSA проблемасы деп аталатын RSA (strong RSA problem). Мәселесінің балама нұсқасы бар. Оның мақсаты e>1 тақ экспонентін табу және осы экспонент үшін RSA есептерін шешу. Күрделі RSA мәселесін шешу e-дисплейі бекітілген қарапайым RSA мәселесінен оңайырақ. Күрделі RSA шақыруы мүмкін емес деп саналады. Күшті RSA есебінің шешілетіндігі кейбір шифрлау алгоритмдері мен криптографиялық хаттамалардың негізін құрайды.

Егер параметрлер үшін ашық кілттегі (N, e) $m < N^{1/e}$ шарты орындалып, хабарламаны шифрлау $\mathrm{c} = m^e (mod\ N)$ операциясын қолданбайды, сондықтан m мәні тиімді болуы мүмкін e-дәреженің түбірін бүтін сандардан алу арқылы есептеу, бұл e нұсқасын таңдамаудың бір себебі.

Ендігі кезекте $m < (N_1 N_2 N_3)^{1}$ экспонциалдауда осылай орындалады. Сонымен текшенің түбірін бүтін сандардан алу үшін C хабарламасының шифры шешіледі, сандары бар және тиімді жүзеге асырылуы мүмкін.

Бағдарламаларда көбінесе кіріс мәтінінің жартысы белгілі кезде, RSA алгоритмында шифрлау кезінде e мәнінің ең кіші көрсеткіштерін қолданбау керек. Әдеттегідей, шифрлау ықтималдылығы көрсеткіш дәрежесінде жай сандар болып келетін $e = 2^{16} + 1 = 65537$ қолданылады. Бұл көрсеткіш шифрлауда жоғары нәтиже берумен және шабуылға төтеп беру арқылы сипатталады.

Егер шифрын ашу кезінде d көрсеткіші қолданылса, кіші нәтиже береді, RSA алгоритмі таңдалған қарапайым мәтінге негізделген шабуылдарға төзімді. Винер (Wiener) ойлап тапқан әдіс e/N үздіксіз. Бұл алгоритм d санды

табуға көмектеседі, егер $d < N^{1/4}$. Ары қарай бұл нәтиже $d < N^{0,292}$ сандар үшін жақсартылды.

2.5 RSA шифрлеу алгоритмін UML-да моделдеу

RSA криптожүйесіндегі қауіпсіздік екі математикалық мәселеге сүйенеді: RSA мәселесі мен үлкен сандардың факторизациясының мәселесі. RSA дағы шифромәтіннің шифрын ашу осы екі мәселе бойынша мүмкін емес, яғни алгоритм тиімсіз. Ішінара шифрлауды қамтамасыз ету қауіпсіз толықтырулар сұлбасын қосуды қажет етуі мүмкін.

мұндағы (П, e) RSA ашық кілті болып табылады және RSA дағы шифромәтінін білдіреді.

Бүгінгі күнде RSA мәселесін шешуде П факторлы модулі ерекше орын алады. Шабуылдаушы (П, e) ашық кілт арқылы D құпия көрсеткішті біле алады, ары қарай стандартты процесстер ақылы шифрын аша береді.

Көпмүшелікті квадраттық (MPQS) көпшілікті n факторлау үшін қолдануға болады. Жұмыс үстеліндегі компьютерде 128 биттік биттік факторларға жұмсалған уақыт (процессор: Intel i7-4500U екі ядролы 1,80Гц) сәйкесінше 2 секунд және 35 минутты құрайды.

MPQS биттері мен уақыттары және жады көлемі 2.2, 2.1-кестеде көрсетілген.

2.1-кесте – MPQS биттері мен уақыттары

Биттер	Уақыты
128	2 секундқа дейін
192	16 секунд
256	35 минут
260	1 сағат

t мәнін арқылы $c = m'^{e}(mod\ N)$ әсерлі есептеуге болады.

Бағдарламаларда көбінесе кіріс мәтіннің жартысы белгілі кезде, RSA алгоритмында шифрлау кезінде e мәнінің ең кіші көрсеткіштерін қолданбау керек. Әдеттегідей, шифрлау ықтималдылығы көрсеткіш дәрежесінде жай

сандар болып келетін $e = 2^{16} + 1 = 65537$ қолданылады. Бұл көрсеткіш шифрлауда жоғары нәтиже берумен, ал шабуылға төтеп беру арқылы сипатталады.

Егер шифрын ашу кезінде d көрсеткіші қолданылса, кіші нәтиже береді, RSA алгоритмі таңдалған қарапайым.

Винер (Wiener) ойлап тапқан әдіс e/N үздіксіз бөлшек ретінде негізделген. Бұл алгоритм d санды табуға көмектеседі, егер $d < N^{1/4}$. Ары қарай бұл нәтиже сандар үшін жақсартылды $d < N^{0,292}$.

RSA криптожүйесіндегі қауіпсіздік екі математикалық мәселеге сүйенеді: RSA мәселесі мен үлкен сандардың факторизациясының мәселесі. RSA дағы шифромәтіннің шифрын ашу осы екі мәселе бойынша мүмкін емес, яғни алгоритм тиімсіз. Ішінара шифрлауды қамтамасыз ету қауіпсіз толықтырулар сұлбасын қосуды қажет етуі мүмкін.

мұндағы (П, e) RSA ашық кілті болып табылады және RSA дағы шифромәтінін білдіреді. Бүгінгі күнде RSA мәселесін шешуде П факторлы модулі. Шабуылдаушы (П, e) ашық кілт арқылы D құпия көрсеткішті біле алады, ары қарай стандартты процесстер ақылы шифрын аша береді.

Көпмүшелікті квадраттық (MPQS) көпшілікті n факторлау үшін қолдануға болады. Жұмыс үстеліндегі компьютерде 128 факторларға жұмсалған уақыт (процессор: Intel i7-4500U екі ядролы 1,80Гц) сәйкесінше 2 секунд және 35 минутты құрайды.

MPQS биттері мен уақыттары және жады көлемі 2.2, 2.2-кестеде көрсетілген.

MPQS биттері мен уақыттары

Биттер	Уақыты
128	2 секундқа дейін
192	16 секунд
256	35 минут
260	1 сағат

Осы үрдісті іске асыратын құрылғы Yafu деп аталады. 5720s фактор 320bit-N сол компьютерге қажет болды (2.3-ксте)-.

2.3-кесте - MPQS биттері, уақыттары және жады көлемі

Биттер	Уақыты	Жады көлемі
128	0,4886 секунд	0,1 MiB
192	3,9979 секунд	0,5 MiB

256	103,1746 секунд	2 MiB
300	1175,7826 секунд	10,9 MiB

2009 жылы Бенджамин RSA-512 биттік кілтін кәдімгі бағдарламалық жасақтаманы және компьютерді (Althlon64 екі ядролы 1900 МГц процессоры) пайдалана отырып, 73 кілтке (GGNFS) бөлді. Процессорға кемінде бес гигабайт дискілік кеңістік және 2,5 гигабайт жедел жады қажет болды. 1999 жылы бірінші RSA-512 факторизациясы 8400 MIPS баламасын талап етті.

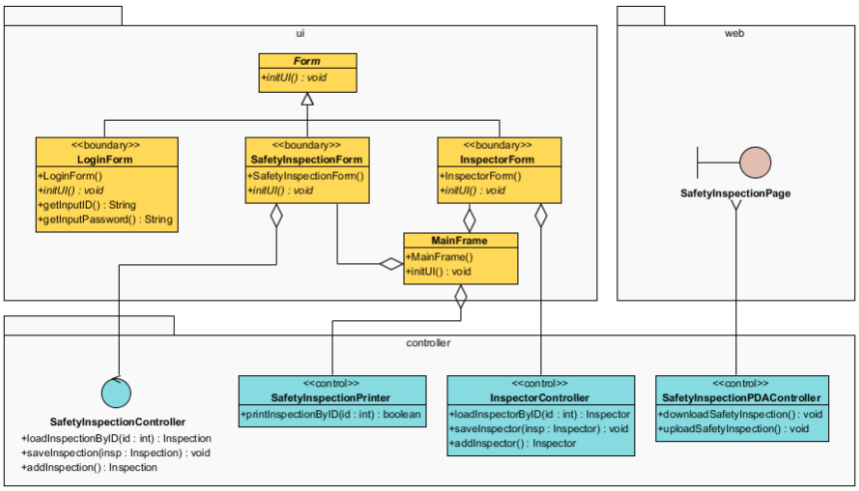
Ривест, Шамир және Адлеман, Миллер егер кеңейтілген Риман гипотезасы дұрыс болса, n және e-ден q табуды n, r және q-ға (уақыт айырмашылығына дейін) ыдырату қиын екенін дәлелдеді. Дегенмен, Ривест, Шамир және Адлеман өз құжаттарының IX/D бөлімінде RSA инверсиясымен факторингтің соншалықты қиын екендігіне дәлел таба алмағанын атап өтті. 2010 Бұл жағдайда ең үлкен ыдырайтын RSA саны 768 бит болды (232 ондық таңба, RSA-768 қараңыз). Оның таралуы бөлінген кірістер арқылы жүзеге асырылады, процессор шамамен бір мың бес жүз жылға созылады (нақты уақытта екі жыл, көптеген жүздеген компьютерлерде). Белгілі RSA кілттері есепке алынбайды. Шын мәнінде, RSA кілттерінің ұзындығы әдетте 1024 және 4096 бит аралығында болады. Кейбір сарапшылардың пікірінше, 1024 биттік кілт жақсы қаржыландырылған шабуылдаушы үшін жақын арада нәзік немесе әлсіз болуы мүмкін, бірақ бұл даулы мәселе. 4096 биттік кілттер болашақта бүлінуі мүмкін екенін аз адамдар көреді. Сондықтан, егер n жеткілікті үлкен болса, RSA қауіпсіз болып саналады.

Егер n 300 бит немесе одан аз болса, оны бірнеше сағатта дербес компьютерде тегін бағдарламалық құрал арқылы ыдыратуға болады. 1999 жылы 512-биттік кілттерді бұзу мүмкін емес екені көрсетілді, бұл кезде RSA-155 бірнеше жүздеген компьютерлермен өзгертілген және қазір кәдімгі жабдықта бірнеше апта ішінде есептелетін болды.

Есептелген 512 биттік сертификатқа қол қою кодының ақаулары 2011 жылы хабарланды. 2003 жылы Шамира мен Тромер сипаттаған TWIRL деп аталатын теориялық жабдық 1024 биттік кілттердің қауіпсіздігіне күмән келтіреді. Қазіргі уақытта n ұзындығы кемінде 2048 бит болуы ұсынылады.

1994 жылы Питер Шор кванттық компьютер - егер оны іс жүзінде құрастыруға болатын болса - RSA-ны жойып, мультиполимерлі уақыт факторына айналатынын көрсетті.

UML (Unified Modeling Language) – объектілі-бағытталған жүйелерді талдау және жобалау процесінде сипаттауға, визуализациялауға және құжаттауға арналған бірыңғай модельдеу тілі (2.5-сурет).



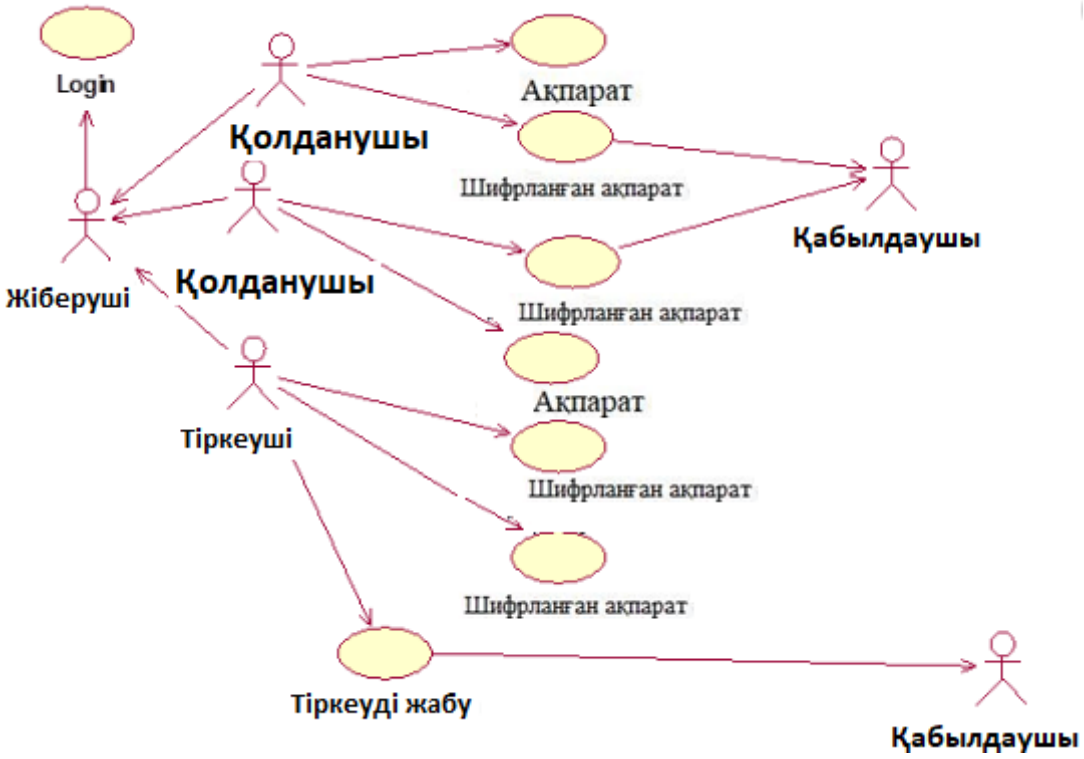
2.5 - Сурет – Реттілік диаграммасы

UML жүйесі бизнес-процестер мен жүйелік функциялар сияқты концептуалды аспектілерді, сондай-ақ бағдарламалау тілінің өрнектері, дерекқор схемалары және қайта пайдалануға болатын бағдарламалық құрал құрамдастары сияқты нақты аспектілерді қамтитын жүйені жобалау құжаттамасын жазудың стандартты әдісін ұсынады.

Өзара әрекеттесудегі нысандар немесе рөлдер арасындағы хабар алмасу арқылы пайдаланушыларды, жүйелерді және ішкі жүйелерді визуализациялау. Сынып диаграммасы олардың қасиеттері мен әдістерін көрсету арқылы сыныптың қаңқасын көрсетсе, UML реттілік диаграммасы толтырылатын бағдарламалау логикасын көрсету арқылы сыныпты аяқтайды. әдіс денесі Класс және объект диаграммалары статикалық үлгінің көрінісі болып табылады.

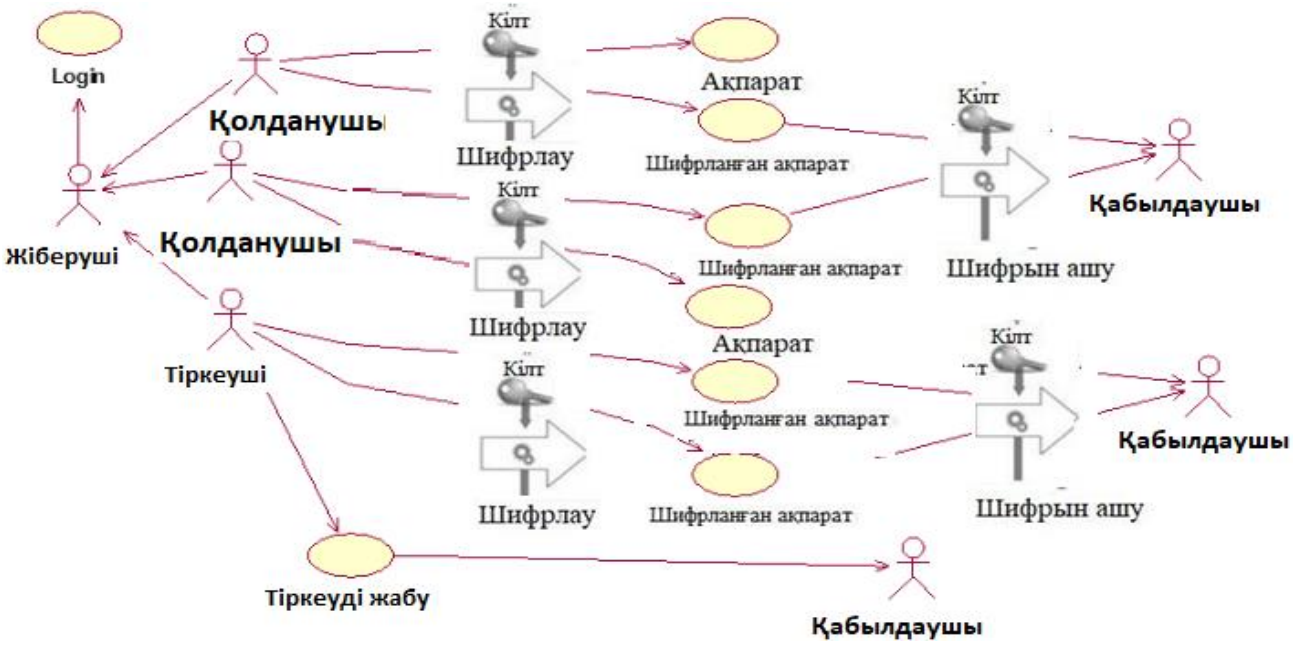
Динамикалық өзара әрекеттесу диаграммалары. Олар объектілердің өзара әрекеттесуін сипаттайды:

- субъектімен әрекеттесетін субъект атқаратын рөл түрі (мысалы, сигналдар мен деректер алмасу арқылы)
- субъектінің сыртқы (яғни, актер данасы оның сәйкес субъектінің данасына кірмейді деген мағынада).
- адам пайдаланушылары, сыртқы жабдық немесе басқа нысандар ойнайтын рөлдерді білдіреді (2.6-2.7-суреттер).



2.6 - Сурет – Реттілік диаграммасы

RSA алгоритмін жүзеге асыру, UML реттілік диаграммасы негізінде. Жұмыстың нәтижесінде «Виртуалды тәжірибе: RSA асиметриялы алгоритмін шифрлау» іске асырылады.



2.7 - Сурет – RSA алгоритмін модельдеу диаграммасы

Хабар жіберуші мен қабылдаушы арасында операцияны жүзеге асыру үшін алдымен кілтті орналастыру қажет. Реттілік диаграммасы бойынша жіберуші қажетті ақпаратты қолданушылардың сұранысы бойынша жеткізеді, жеткізілген ақпараттар қабылдаушыға жіберу кезінде шифрланады да, қабылдаушы жақ ақпаратты қабылдар алдында шифры шешіледі.

3 Шифрлардың алгоритмін құру және оны программалау

3.1 RSA криптографиялық алгоритмін Python тілінде программалау

Python программалау тілінде жазылған программа танымал RSA шифрлау және дешифрлау криптографиялық алгоритмін жүзеге асыру болып табылады. Жоба тапсырмасын сәтті орындау үшін Эйлер функциясын, GCD (ең үлкен ортақ бөлгіш), бүтін сандардың бар-жоғын тексеру функциясын автоматтандыруды, кері сандарды, шифрлауды және шифрды шешуді, сондай-ақ олардың барлығын жүзеге асыруды зерттеу қажет. Python программалау тіліндегі программа кодында, программа кодындағы әртүрлі функцияларды біріктіруді, сондай-ақ оларды Python программалау тіліне бейімдеу қарастырылған. Басқа көздерден алынған дайын программалау кодтарын пайдалану, сондай-ақ оған өзгерістер кірістіру жабық интерпретация мен идеяларды қажет етеді, сонымен қатар, модельдерді жаңартуға, жаңаларын құруға мүмкіндік береді [7].

Python – әзірлеушілердің өнімділігін, кодты оқу мүмкіндігін және сапасын жақсартуға және онда жазылған программалардың тасымалдануын қамтамасыз етуге бағытталған динамикалық күшті теру және автоматты жадыда басқаруы бар жоғары деңгейлі, жалпы мақсаттағы программалау тілі. Тіл толығымен объектіге бағытталған – бәрі объект болып табылады. Тілдің әдеттен тыс ерекшелігі - бос орындар шегінісі бар код блоктарына бөлуге мүмкіндік береді. Негізгі тілдің синтаксисі минималистік болып табылады, сондықтан іс жүзінде құжаттамаға сілтеме жасау қажеттілігі сирек кездеседі [8]. Тілдің өзі интерпретация ретінде белгілі және басқалармен қатар сценарий жазу үшін қолданылады. Тілдің кемшіліктері C немесе C++ сияқты құрастырылған тілдерде жазылған ұқсас кодпен салыстырғанда жиі төмен жылдамдыққа ие және онда жазылған программалардың жадыны көп қажет ететіні болып табылады.

Python тілінің негізгі ерекшеліктері:

- функционалдық және құрылымдық бағдарламалау мен әдістерді, сонымен қатар объектіге бағытталған бағдарламалау әдістерін қамтиды.
- Python тілін сценарий тілі немесе бағдарламалау тілі ретінде қолдануға болады.
- Автоматты түрде қажетсіз заттарды жинауды қамтиды.
- Python тілі жоғары деңгейлі динамикалық деректер түрлерін қамтиды және әртүрлі динамикалық типті тексерулерді қолдайды.
- Python C, C++ және Java сияқты тілдермен интеграцияны қамтиды.

Ол функционалдық және құрылымдық бағдарламалау мен әдістерді, сонымен қатар объектіге бағытталған бағдарламалау әдістерін қамтиды.

3.2 Эйлер функциясы және Евклид алгоритмі

Эйлер функциясы мультипликативті арифметикалық функция болып табылады, оның мәні аспайтын натурал сандар санына тең және онымен салыстырылады.

Мысалы, 36 саны үшін 12 кіші және онымен бірге қарапайым сандар (1, 5, 7, 11, 13, 17, 19, 23, 25, 29, 31, 35) бар.

Оны алғаш рет 1763 жылы сандар теориясы жөніндегі жұмысында Ферманың кіші теоремасын, кейінірек жалпылама тұжырымды, Эйлер теоремасын дәлелдеу үшін пайдаланған Эйлердің атымен аталған. Функцияны кейінірек Гаусс 1801 жылы жарияланған «Арифметикалық зерттеулерінде» пайдаланды. Гаусс қазіргі стандартты белгілерді енгізді [9].

Эйлер функциясы бөлінгіштік пен қалдық теориясына (модульдік салыстыру), сандар теориясына және криптографияға қатысты мәселелерде қолданыс табады. RSA алгоритмінде Эйлер функциясы негізгі рөл атқарады.

Евклид алгоритмі – екі бүтін санның ең үлкен ортақ бөлгішін (немесе екі сегменттің ортақ өлшемін) табудың тиімді алгоритмі. Алгоритм «Бастаудың» VII және X кітаптарында алғаш рет сипаттаған грек математигі Евклидтің (б.з.д. III ғ.) құрметіне аталған. Бұл қазіргі кезде қолданылатын ең көне сандық алгоритмдердің бірі [10].

Ең қарапайым түрде Евклид алгоритмі оң бүтін сандар жұбына қолданылады және кіші сан мен үлкен және кіші сандар арасындағы айырмашылықтан тұратын жаңа жұпты жасайды. Процесс сандар тең болғанша қайталанады. Табылған сан бастапқы жұптың ең үлкен ортақ бөлгіші болып табылады. Евклид тек натурал сандар мен геометриялық шамалар (ұзындықтар, аудандар, көлемдер) алгоритмін ұсынды. Алайда 19 ғасырда ол математикалық объектілердің басқа түрлеріне, соның ішінде Гаусс бүтін сандары мен бір айнымалыдағы көпмүшеліктерге жалпыланды. Бұл қазіргі жалпы алгебрада Евклид сақинасы сияқты ұғымның пайда болуына әкелді. Кейінірек Евклид алгоритмі түйіндер мен көпөлшемді көпмүшеліктер сияқты басқа математикалық құрылымдарға жалпыланды.

Бұл алгоритм үшін көптеген теориялық және практикалық қолданбалар бар. Атап айтқанда, ол электронды коммерцияда кеңінен қолданылатын RSA ашық кілті, криптографиялық алгоритмнің негізі болып табылады. Алгоритм сонымен қатар сызықтық диофантиндік теңдеулерді шешуде, жалғастырылған бөлшектерді құруда, Штурм әдісінде қолданылады. Евклид алгоритмі қазіргі сандар теориясындағы Лагранждың төрт квадраттық теоремасы және арифметиканың негізгі теоремасы сияқты теоремаларды дәлелдеудің негізгі құралы болып табылады [11].

3.3 RSA криптографиялық деректерді шифрлау алгоритмі

RSA (Rivest, Shamir және Adleman сөздерінің аббревиатурасы) үлкен бүтін сандарды көбейту мәселесінің есептеу күрделілігіне негізделген ашық кілтті криптографиялық алгоритм болып табылады.

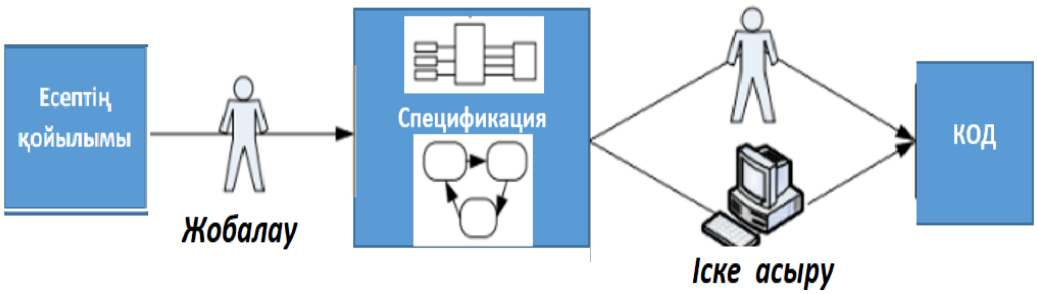
RSA криптожүйесі шифрлау үшін де, цифрлық қолтаңба үшін де қолайлы бірінші жүйе болды. Алгоритм көптеген криптографиялық қолданбаларда, соның ішінде PGP, S/MIME, TLS/SSL, IPSEC/IKE және т.б. қолданылады.

Қолданылатын кілттердің құрылымына байланысты шифрлау әдістері келесіге бөлінеді:

- симметриялық шифрлау: үшінші тұлғалар шифрлау алгоритмін білуі мүмкін, бірақ құпия ақпараттың шағын бөлігі белгісіз – кілт хабарламаны жіберуші мен алушы үшін бірдей; Мысалдар: DES, 3DES, AES, Blowfish, Twofish, ГОСТ 28147-89
- асимметриялық шифрлау: үшінші тараптар шифрлау алгоритмін және мүмкін ашық кілтті білуі мүмкін, бірақ тек алушыға белгілі жабық кілт емес. Ашық кілтті криптографиялық жүйелер қазіргі уақытта әртүрлі желілік протоколдарда, атап айтқанда, TLS протоколдарында және оның алдындағы SSL (негізгі HTTPS), сонымен қатар SSH, PGP, S/MIME және т.б. кеңінен қолданылады. Асимметриялық шифрлауды қолданатын стандарт – ГОСТ Р 34.10-2001 ж.

Қазіргі уақытта RSA ашық кілтіне негізделген асимметриялық шифрлау ақпараттық қауіпсіздік нарығындағы өнімдердің көпшілігінде қолданылады [12].

Оның криптографиялық күші үлкен сандарды факторингке бөлудің қиындығына, атап айтқанда ашық кілтке негізделген құпия кілтті анықтау тапсырмасының ерекше қиындығына негізделген, өйткені бұл бүтін санның бөлгіштерінің бар екендігі туралы мәселені шешуді талап етеді. Криптографиялық тұрғыдан ең күшті жүйелер 1024 биттік және одан үлкен сандарды пайдаланады (3.1-сурет).



3.1 - Сурет - Күрделі сипаттамадағы бағдарламалық жүйені әзірлеу кезеңдері

RSA алгоритмін практикалық тұрғыдан қарастыру.

Алдымен ашық және жабық кілттерді құру керек:

- Екі үлкен p және q жай сандары алынады.
- q-ға p көбейту нәтижесі ретінде n-ді анықтау керек (n= p*q).
- Кездейсоқ сан таңдалады, ол d деп аталады. Бұл сан көбейту нәтижесімен (p-1)*(q-1) қос жай (1-ден басқа ортақ бөлгіші жоқ) болуы керек.
- Осындай e саны анықталады, ол үшін келесі (e*d) mod ((p-1)*(q-1))=1 қатынасы ақиқат.
- Ашық кілт e және n сандарынан, ал құпия – d және n сандары деп аталады.

{e,n} ашық кілтінің көмегімен деректерді шифрлау үшін қолданушыға мыналар қажет:

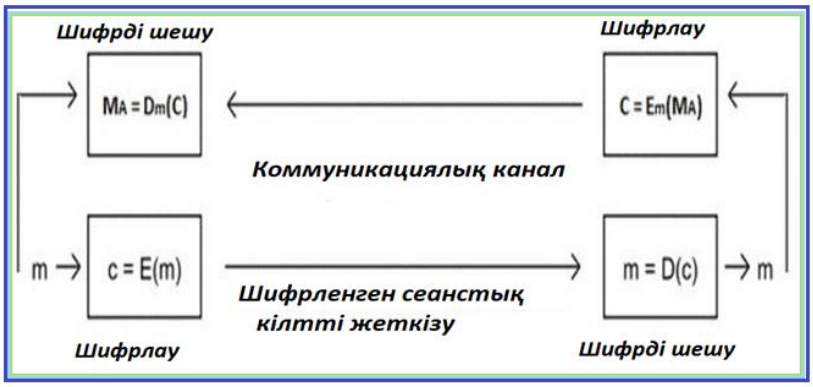
- шифрланған мәтіндер блоктарға бөлінеді, олардың әрқайсысы M(i)=0,1,2..., n-1 (яғни тек n-1-ге дейін) саны ретінде ұсынылуы мүмкін.
- C(i)=(M(I)^e)mod n формуласы бойынша M(i) сандар тізбегі ретінде қарастырылатын мәтінді шифрлау керек [13].

{d,n} құпия кілтін пайдаланып бұл деректердің шифрын ашу үшін келесі есептеулерді орындау қажет: M(i) = (C(i)^d) mod n. Нәтижесінде түпнұсқа мәтінді білдіретін M(i) сандар жинағы алынады (3.2-сурет).

3.2 - Сурет – Шифрланған деректерді жеткізу

Келесі мысал RSA шифрлау алгоритмін анық көрсетеді:

RSA алгоритмі арқылы «Сәлем» хабарламасын шифрлап, шифрын ашу орындалады (3.3-сурет).



3.3 - Сурет – Шифрланған сеанстық кілтті жеткізу

Қарапайымдылық үшін шағын сандар алынады - бұл барлық есептеулерді қысқартады.

- p=3 және q=11 таңдалады.
- n= 3*11=33 мәні анықталады.
- (p-1)*(q-1)=20 табу керек. Демек, d, мысалы, 3-ке тең болады: (d=3).
- Мына формула бойынша e саны таңдалады: (e*3) mod 20=1. Сонымен e, мысалы, 7: (e=7) тең болады.
- Шифрланған хабарлама 0-ден 32-ге дейінгі диапазондағы сандар тізбегі ретінде көрсетіледі (ол n-1-мен аяқталатынын ұмытпау керек). Әріп A=1, B=2, C=3.

Енді хабар {7,33} ашық кілт арқылы шифрланады.
C1 = (3^7) мод 33 = 2187 мод 33 = 9;
C2 = (1^7) мод 33 = 1 мод 33 = 1;
C3 = (2^7) мод 33 = 128 мод 33 = 29;

Енді {3,33} жабық кілтін пайдаланып деректердің шифрын ашу керек.
M1=(9^3) мод 33 =729 мод 33 = 3(C);
M2=(1^3) мод 33 =1 мод 33 = 1(A);
M3=(29^3) мод 33 = 24389 мод 33 = 2(B);
Сонымен, шифрлау және шифрды ашу іске асырылды.

3.4 Dev-C++

Dev-C++ - C/C++бағдарламалау тілдеріне арналған қосымшаларды әзірлеудің еркін интеграцияланған ортасы. Дистрибуцияға MinGW компиляторы кіреді. Dev-c++ өзі Delphi-де жазылған. GPL сәйкес таратылады. Жобаның авторы Колин Лаплас, bloodshed Software. 2001 жылдың 30 шілдесіне дейін Linux порты (0.7.0 нұсқасы) жасалды. Алайда, әзірлеушілер Windows үшін өнімді жасаумен

шектелуді шешті. Түпнұсқа Dev-c++ соңғы нұсқасы 2005 жылдың ақпан айында шықты.

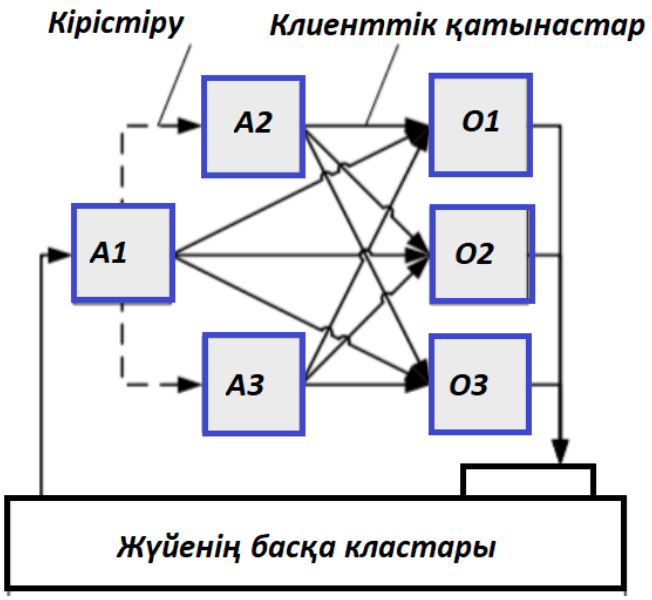
Orwell Dev-c++ деп аталатын шанышқыны 2015 жылдың сәуіріне дейін Johan Mes (Orwell) жасады.

Wxwidgets-wxDev — C++ тілінде Dev-c++интерфейсінің порты да жасалды. Жоба қараусыз қалып.

Қазіргі уақытта дамуды Embarcadero Dev-c++ шанышқысы түрінде жалғастыруда, бұл Bloodshed Software веб-сайтына сілтеме арқылы расталады.

Embarcadero Dev-c++-бұл bloodshed Dev-C++ және Orwell Dev-c++ жаңа және жетілдірілген шанышқы. Бұл толық функционалды IDE және C/C++бағдарламалау тіліне арналған код редакторы. Компилятор ретінде ол GCC (GNU Compiler Collection) үшін MinGW портын пайдаланады. Embarcadero Dev-c++ - ті Cygwin немесе GCC негізіндегі кез-келген басқа компилятормен бірге қолдануға болады. Бұл шанышқыны әзірлеу Embarcadero Delphi-дің қазіргі нұсқасын қолдану арқылы жүзеге асырылады.

Ол тегін таратылады, ашық бастапқы программалық қамтамасыз ету ретінде әзірленген, бірақ дайын жинақтар меншік лицензиясы бойынша таратылады 3.4-сурет).



3.4 - Сурет – Күрделі жүйе архитектурасы

Visual Studio коды Electron негізінде жасалған және Visual Studio Online үшін жасалған Монако веб-редакторы арқылы жүзеге асырылады.

3.5 Программаның негізгі функциялары мен модульдері

Программаның негізгі принципі деректерді шифрлау және шифрын ашу болып табылады. Сондай-ақ программада жай сандардың бар-жоғын автоматты түрде тексеру, RSA модулін, Эйлер функциясын, сондай-ақ GCD (ең үлкен ортақ бөлгіш) есептеу функциясы бар [16].

Алдымен код толық қарастырылады. Программаның мақсаты көрсетіледі. Бірден кейін функциялар мен алгоритмдердегі мәндерді есептеу үшін жай сандарды енгізуді қамтитын программа коды құрылады (3.5-сурет).

```
'''
ДОБРО ПОЖАЛОВАТЬ В ШИФРАТОР И ДЕШИФРАТОР, ОСНОВАННЫЙ НА АЛГОРИТМЕ RSA.

'''

import math

print("ШИФРАТОР/ДЕШИФРАТОР RSA")
print("**************************************************")

# Введение  простых числ
print("ВНЕСИТЕ 'p' И 'q' ЗНАЧЕНИЯ НИЖЕ:")
p = int(input("Внесите простое число для p: "))
q = int(input("Внесите простое число для q: "))
print("**************************************************")
```

3.5 - Сурет - Python тіліндегі программалық код

Енгізілген деректерді тексеру функциясы бар программа коды, атап айтқанда, жай сандардың болуы немесе болмауы (3.6-сурет). Бұл функциялар мен алгоритмдерді дұрыс және оңтайлы есептеу үшін жасалады.

```
# Проверьте, являются ли входные данные простыми
'''ЭТА ФУНКЦИЯ И КОД АВТОМАТИЧЕСКИ ПРОВЕРЯЮТ, ПРОСТОЕ ЛИ ЧИСЛО БЫЛО ВНЕСЕНО'''


def prime_check(a):
    if (a == 2):
        return True
    elif ((a < 2) or ((a % 2) == 0)):
        return False
    elif (a > 2):
        for i in range(2, a):
            if not (a % i):
                return False
    return True


check_p = prime_check(p)
check_q = prime_check(q)
while (((check_p == False) or (check_q == False))):
    p = int(input("Внесите простое число для p: "))
    q = int(input("Внесите простое число для q: "))
    check_p = prime_check(p)
    check_q = prime_check(q)
```

3.6 - Сурет - Енгізілген деректерді тексеру

Содан кейін «n» мәні бар RSA модулін, «r» мәні бар Эйлер функциясын, сондай-ақ «e» мәні бар GCD (ең үлкен ортақ бөлгіш) есебін (3.7-сурет) қабылдайды.

```
# RSA Modulus
'''ВЫЧИСЛЕНИЕ RSA МОДУЛЯ 'n'.'''
n = p * q
print("RSA Модуль(n) равен:", n)

# Eulers Toitent
'''ВЫЧИСЛЕНИЕ ФУНКЦИИ ЭЙЛЕРА 'r'.'''
r = (p - 1) * (q - 1)
print("Функция Эйлера(r) равна:", r)
print("**************************************************")

# GCD
'''ВЫЧИСЛЕНИЕ НОД(наибольший общий делитель) ДЛЯ 'e'.'''


def egcd(e, r):
    while (r != 0):
        e, r = r, e % r
    return e
```

3.7 - Сурет - Ең үлкен ортақ бөлгішті анықтау

Осыдан кейін Евклид алгоритмімен, оның кеңейтілген нұсқасымен және өзара әрекеттестігімен жұмыс атқарылады (3.8-сурет).

```python
# Euclid's Algorithm
def eugcd(e, r):
    for i in range(1, r):
        while (e != 0):
            a, b = r // e, r % e
            if (b != 0):
                print("%d = %d*(%d) + %d" % (r, a, e, b))
            r = e
            e = b


# Extended Euclidean Algorithm
def eea(a, b):
    if (a % b == 0):
        return (b, 0, 1)
    else:
        gcd, s, t = eea(b, a % b)
        s = s - ((a // b) * t)
        print("%d = %d*(%d) + (%d)*(%d)" % (gcd, a, t, s, b))
        return (gcd, t, s)

    # Multiplicative Inverse


def mult_inv(e, r):
    gcd, s, _ = eea(e, r)
    if (gcd != 1):
        return None
    else:
        if (s < 0):
            print("s=%d. Так как %d меньше 0, s = s(modr), т.е., s=%d." % (s, s, s % r))
        elif (s > 0):
            print("s=%d." % (s))
        return s % r
```

3.8 - Сурет - Евклид алгоритмі

Соңында, программа 1 мен 1000 арасындағы ең үлкен gcd (ең үлкен ортақ бөлгішті) табады, ол gcd мен Эйлер функциясының мәнін құрайды. Сондай-ақ Евклид алгоритмі жабық және ашық кілттерді есептейді (3.9-сурет).

```python
# e Value Calculation
'''НАХОДИТ НАИБОЛЬШЕЕ ЗНАЧЕНИЕ 'e' МЕЖДУ 1 И 1000 ЧТО ДЕЛАЕТ (e,r) ВЗАИМНО ПРОСТЫМИ ЧИСЛАМИ.'''
for i in range(1, 1000):
    if (egcd(i, r) == 1):
        e = i
print("Значение e равно:", e)
print("****************************************************")

# d, Private and Public Keys
'''ВЫЧИСЛЕНИЕ 'd', ЗАКРЫТЫЙ КЛЮЧ И ОТКРЫТЫЙ КЛЮЧ.'''
print("АЛГОРИТМ ЕВКЛИДА:")
eugcd(e, r)
print("ПОСЛЕДНИЕ ШАГИ ДЛЯ ВЫЧИСЛЕНИЯ АЛГОРИТМА ЕВКЛИДА.")
print("****************************************************")
print("РАСШИРЕННЫЙ АЛГОРИТМ ЕВКЛИДА:")
d = mult_inv(e, r)
print("ПОСЛЕДНИЕ ШАГИ ДЛЯ ВЫЧИСЛЕНИЯ ЗНАЧЕНИЯ 'd'.")
print("Значение d равно:", d)
print("****************************************************")
public = (e, n)
private = (d, n)
print("Закрытый ключ:", private)
print("Открытый ключ:", public)
print("****************************************************")
```

3.9 - Сурет - Жабық және ашық кілттерді есептеу

Шифрлау және дешифрлау алгоритмдері бар программа коды (3.10-сурет).

Алдымен, қолданушы шифрлайтын немесе шифрын ашатын деректерді енгізуі керек. Содан кейін тапсырма негізінде деректерді шифрлау үшін «1» немесе шифрын шешу үшін «2» мәнін енгізеді.

```
# Encryption
'''АЛГОРИТМ ШИФРОВАНИЯ.'''
def encrypt(pub_key, n_text):
    e, n = pub_key
    x = []
    m = 0
    for i in n_text:
        if (i.isupper()):
            m = ord(i) - 65
            c = (m ** e) % n
            x.append(c)
        elif (i.islower()):
            m = ord(i) - 97
            c = (m ** e) % n
            x.append(c)
        elif (i.isspace()):
            spc = 400
            x.append(400)
    return x
# Decryption
'''АЛГОРИТМ ДЕШИФРОВАНИЯ.'''
def decrypt(priv_key, c_text):
    d, n = priv_key
    txt = c_text.split(',')
    x = ''
    m = 0
    for i in txt:
        if (i == '400'):
            x += ' '
        else:
            m = (int(i) ** d) % n
            m += 65
            c = chr(m)
            x += c
    return x
```

3.10 - Сурет -. Шифрлау және дешифрлау алгоритмдері бар программа коды

Содан кейін қолданушы RSA криптографиялық алгоритмі арқылы өңделген деректерді алады. Мән қате енгізілсе, қате орын алады (3.11-сурет).

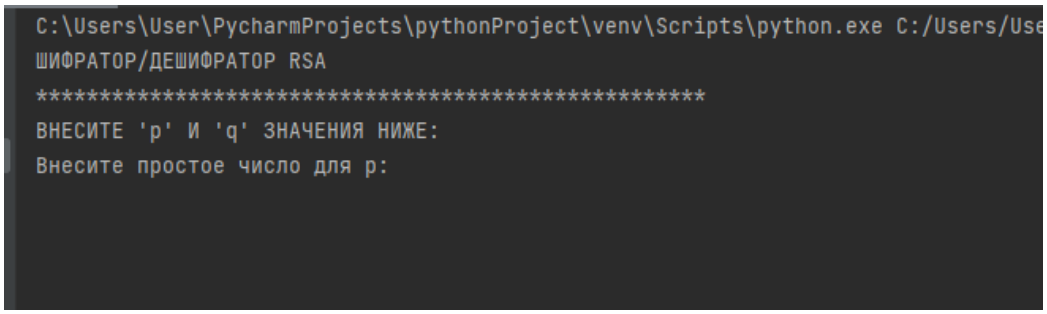
```
# Message
message = input("Внесите сообщение для шифрования или дешифрования.(Числа раздельно с ',' для дешифрования):")
print("Ваше сообщение:", message)

# Choose Encrypt or Decrypt and Print
choose = input("Напишите '1' for шифрования и '2' для дешифрования:")
if (choose == '1'):
    enc_msg = encrypt(public, message)
    print("Ваше шифрованное сообщение:", enc_msg)
    print("Благодарим за пользование шифратором RSA!")
elif (choose == '2'):
    print("Ваше дешифрованное сообщение:", decrypt(private, message))
    print("Благодарим за пользование дешифратором RSA!")
else:
    print("Вы внесли неправильное значение.")
    print("Ошибка!")
```

3.11 - Сурет - Енгізілген мәнді тексеру

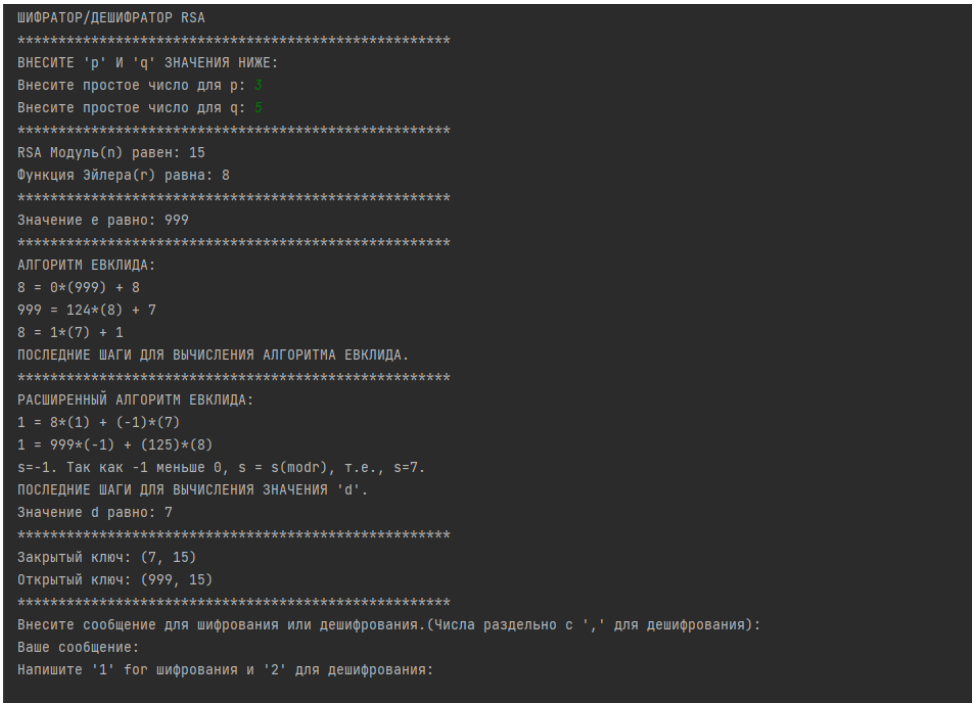
Егер енгізілген мән қата болатын болса, онда қолданушыға қата туралы ақпарат беріледі, ал, дұрыс енгізілген болса, онда шифрлау немесе дешифрлау іске асырылады.

Көрнекі мысалды пайдалана отырып, пәрмен жолында іске қосылған программаның жұмысы қарастырылады (3.12-сурет):

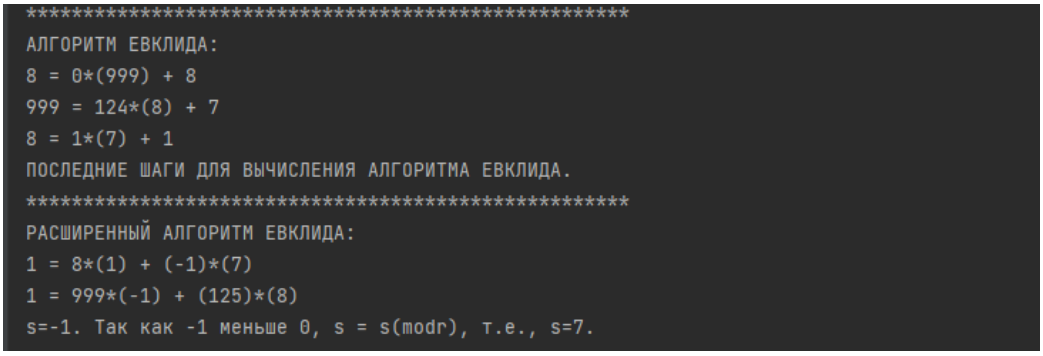


3.12 - Сурет - Программаның жұмысы

Программа кодын сәтті құрастырғаннан кейін программа пәрмен жолында іске қосылады. Алдымен 'p' және 'q' мәндері үшін жай сандарды енгізу керек (3.13-3.14-суреттер).

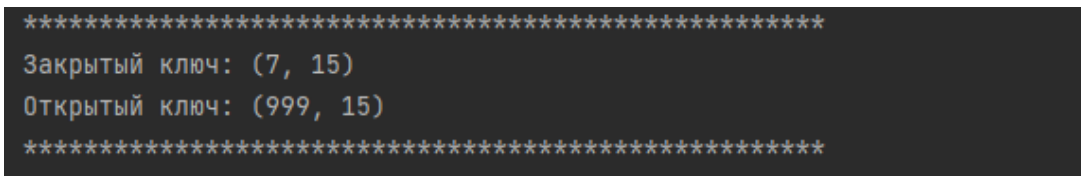


3.13 - Сурет - 'p' және 'q' мәндері үшін жай сандарды енгізу



3.14 - Сурет - Евклид алгоритмін есептеу

Мәндерді енгізгеннен кейін программа олардың дұрыстығын автоматты түрде тексереді. Сәтті тексеруден кейін ол RSA модулі мен Эйлер функциясын есептейді. Евклид алгоритмінде және оның кеңейтілген нұсқасында есептеулерді орындайды. Соңында жабық және ашық кілт мәндерін ашу орындалады (3.15-сурет).



3.15 - Сурет - Жабық және ашық кілт мәндерін ашу

Содан кейін шифрланатын немесе шифрын шешетін деректерді енгізу керек. «СӘЛЕМ» сөзін шифрлау керек. Деректерді шифрлау үшін «1» мәні енгізіледі. Деректер шифрланған және «13, 4, 11, 11, 14» шифрланған хабарлама алынады (3.16-сурет).

```
**************************************************
Закрытый ключ: (7, 15)
Открытый ключ: (999, 15)
**************************************************
Внесите сообщение для шифрования или дешифрования.(Числа раздельно с ',' для дешифрования):Hello
Ваше сообщение: Hello
Напишите '1' for шифрования и '2' для дешифрования:1
Ваше шифрованное сообщение: [13, 4, 11, 11, 14]
Благодарим за пользование шифратором RSA!
```

3.16 - Сурет - Шифрланған хабарлама

Енді бұрын шифрланған «13, 4, 11, 11, 14» хабарламасының шифрын ашуға тырысу керек. Деректердің шифрын ашу үшін «2» мәні енгізіледі. Деректер шифры шешілді және шифры шешілген «СӘЛЕМ» хабарламасы алынады (3.17-сурет).

```
**************************************************
Внесите сообщение для шифрования или дешифрования.(Числа раздельно с ',' для дешифрования):13,4,11,11,14
Ваше сообщение: 13,4,11,11,14
Напишите '1' for шифрования и '2' для дешифрования:2
Ваше дешифрованное сообщение: HELLO
Благодарим за пользование дешифратором RSA!
```

3.17 - Сурет - Шифрды ашу

Программаның жұмыс қабілеттілігін тексерілді және тапсырма орындалды деп сенімді түрде айтуға болады. Қарастырылған мысалдар программада ақпаратты қолданушы нұсқаулығы бар екенін көрсетеді. Программа ақпаратты қорғауды қамтамасыз ететін деректерді шифрлау/шифрды шешу функцияларын орындайды [18]. Программа Windows 10, Linux және MacOS операциялық жүйелерінде тұрақты жұмыс істейді (3.18-3.20-сурет).

3.6 RSA деректерді шифрлау жүйесінің программалық пакетін құру

RSA шифрлау криптожүйесі симметриялық жүйеден ерекше болатын ашық кілтті ұсынады. Оның жұмыс істеу принципі мынада: екі түрлі кілт қолданылады - жеке (шифрланған) және жалпыға ортақ. Біріншісі ЭСҚ құру үшін пайдаланылады және кейіннен мәтіннің шифрын шеше алады. Екіншісі – нақты шифрлеуге және ЭЦҚ-ны тексеруге арналған. Қолтаңбаны пайдалану RSA шифрлауын жақсырақ түсінуге мүмкіндік береді, оның мысалын қарапайым құпия ретінде «көзге көрінбейтін» құжат ретінде келтіруге болады.

RSA алгоритмі төрт қадамнан тұрады: кілттерді генерациялау, кілттерді тарату, шифрлау және шифрды шешу. RSA шифрлауы ашық және жабық кілтті қамтиды. Ашық кілт барлығына белгілі болуы мүмкін және ол хабарламаларды шифрлау үшін қолданылады. Оның мәні мынада: ашық кілт арқылы шифрланған хабарламалар құпия кілттің көмегімен белгілі бір уақыт аралығында ғана шифрдан шығарылуы мүмкін.

Алдымен ашық және жабық кілттерді құру керек:

- Екі үлкен p және q жай сандары алынады: p =17; q = 31

Тексеру жүргізілді – екі санның екеуі де жай сандар.

- q-ға p көбейту нәтижесі ретінде n-ді анықталады (n= p*q =17*31=527).
- Кездейсоқ сан таңдалады, ол d деп аталады. Бұл сан көбейту нәтижесі бойынша (p-1)*(q-1) көбейтіндісі қос жай сан (1-ден басқа ортақ бөлгіші жоқ) болуы керек

$$f(n)= (p-1)*(q-1) = 16*30 = 480$$

$$\varphi(n)=\varphi(p*q)=\varphi(17x31) = \varphi(17)\varphi(31)=16x30=480$$

- Осындай e санын анықталады, ол үшін келесі (e*d) mod ((p-1)*(q-1))=1 қатынасы ақиқат болу керек.
- Ашық кілт e және n сандары, ал құпия – d және n деп аталуы керек.

{e,n} ашық кілтінің көмегімен деректерді шифрлау үшін қолданушыға мыналар қажет:

- шифрланған мәтінді блоктарға бөліп, олардың әрқайсысы M(i)=0,1,2..., n-1 (яғни тек n-1-ге дейін) саны ретінде ұсынылады.
- C(i)=(M(I)^e)mod n формуласы бойынша M(i) сандар тізбегі ретінде қарастырылатын мәтін шифрланады.

{d,n} құпия кілтті пайдаланып бұл деректердің шифрын ашу үшін келесі есептеулерді орындау қажет:

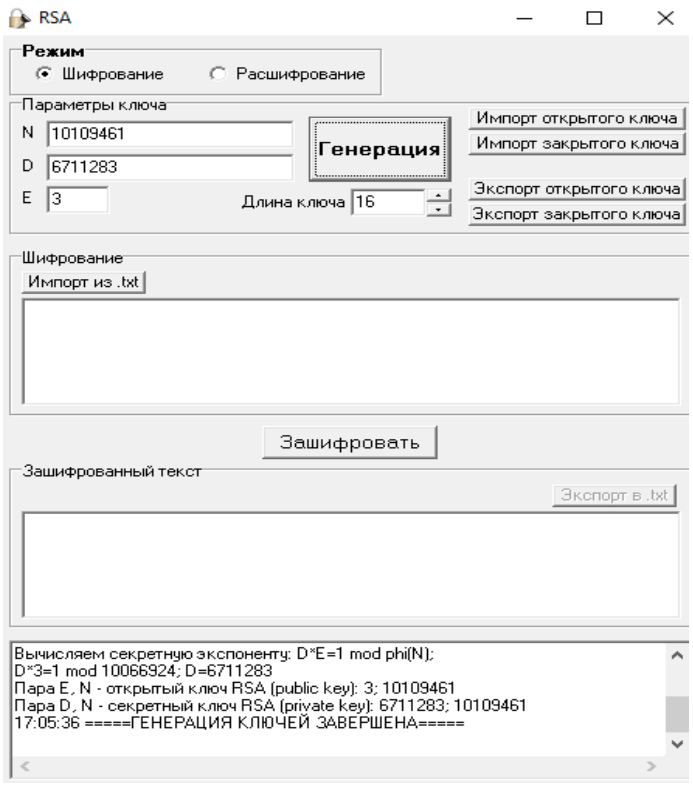
M(i) = (C(i)^d) mod n.

Құрылған программа ашылатын 3 қосымшадан тұрады: Кілттерді құру, Шифрлеу, Дешифрлеу. Бастапқы терезеде, 1-қосымша ашылған кезде p және q мәндері енгізіледі, Проверка түймешігін шертіп, бұл екі санның жәй сан екендігі

тексеріледі. Бұл сандар жәй болған жағдайда e=3 енгізіледі, Эйлер функциясын есептеу үшін f(n), d және n мәндері есептеледі (3.18-3.19-суреттер).

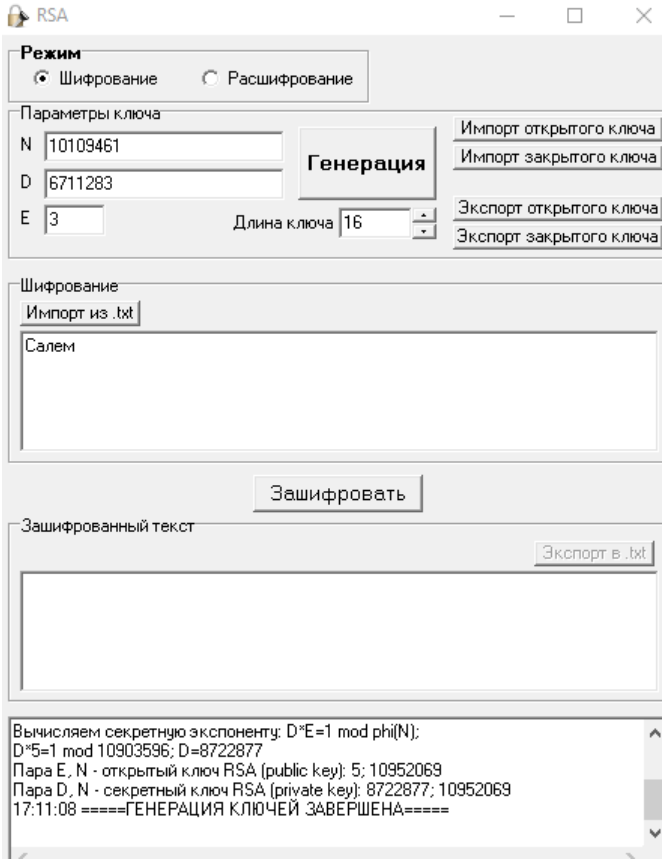
3.18 - Сурет – Бастапқы бет

Жабық кілтті құру, ол үшін N,D,E –енгізіледі.



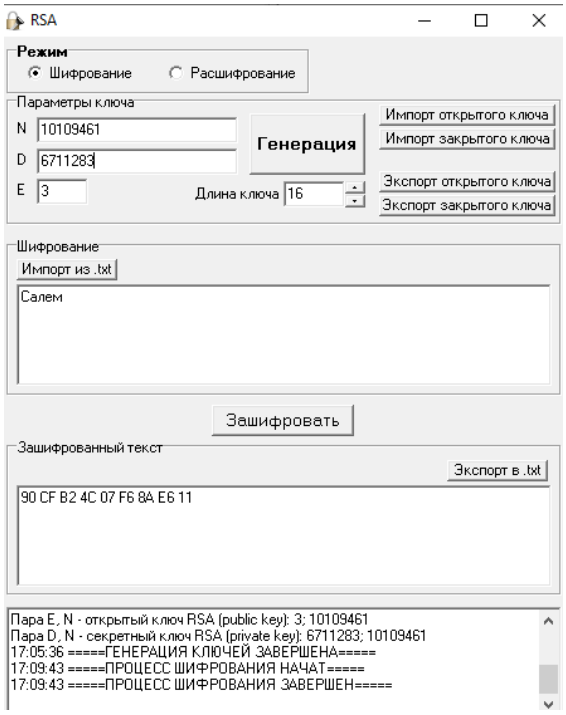
3.19 - Сурет – Шифрлау процесі

2-қосымшаны Шифрование ашып, ашық мәтін «Салем» енгізіледі. Шифрограмма терезесінен 90 CF B2 4C 07 F6 8A E6 11 – шифрланған мәтін шығады, Осы терезеде құпия кілтті көрсетуге боладығ ол d=6711283.



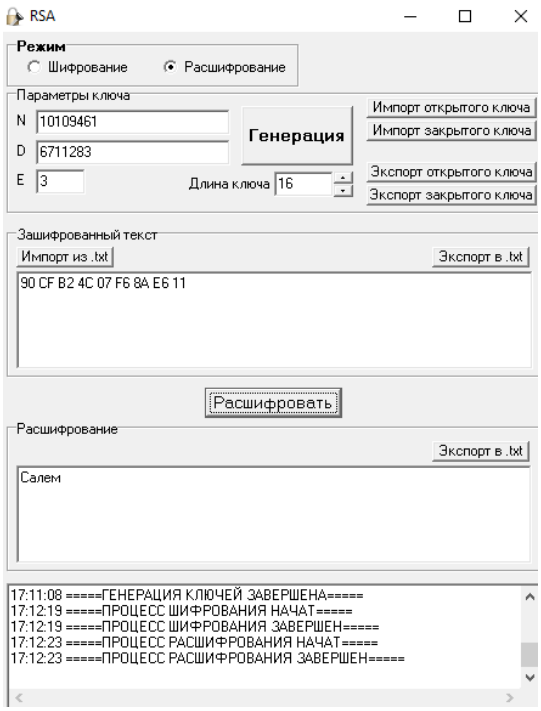
3.20 - Сурет – Шифрланатын хабарлама

Шифрование қосымшасында Шифрограмма жолында шифрленген мәтін шығады (3.21-сурет).



3.21 - Сурет – Шифрланған хабарлама

Дешифрование қосымшасында Открытый текст жолында «Салем» хабарлама шығады. Ашық кілт e=3.



3.22 - Сурет – Дешифрланған хабарлама

Нәтижесінде түпнұсқа мәтінді білдіретін M(i) сандар жинағы алынады.

Қарапайымдылық үшін шағын сандар алынады - бұл шифрлау барысын және есептеулерді қысқартады.

- p=3 және q=11 таңдалады.
- n= 3*11=33 мәні анықталады.
- (p-1)*(q-1)=20 табу керек. Демек, d, мысалы, 3-ке тең болады: (d=3).
- Мына формула бойынша e саны таңдалады: (e*3) mod 20=1. Сонымен e, мысалы, 7: (e=7) тең болады.
- Шифрланған хабарлама 0-ден 32-ге дейінгі диапазондағы сандар тізбегі ретінде көрсетіледі (ол n-1-мен аяқталады). Әріп A=1, B=2, C=3.

Енді хабар {7,33} ашық кілт арқылы шифрланады.

C1 = (3^7) мод 33 = 2187 мод 33 = 9;

C2 = (1^7) мод 33 = 1 мод 33 = 1;

C3 = (2^7) мод 33 = 128 мод 33 = 29;

Енді {3,33} жабық кілтті пайдаланып деректердің шифрын ашу керек.

M1=(9^3) мод 33 =729 мод 33 = 3(C);

M2=(1^3) мод 33 =1 мод 33 = 1(A);

M3=(29^3) мод 33 = 24389 мод 33 = 2(B);

Құрылған программалық пакет берлық қойылған тапсырмаларға сәйкес құрылған, барлық терезелер мен қосымшалар нақты, қатесіз жұмыс істейді.

ҚОРЫТЫНДЫ

RSA жүйесі программалық қамтамасыз етуді қорғау үшін және цифрлық қолтаңба схемаларында қолданылады.

Ол сондай-ақ PGP ашық шифрлау жүйесінде және басқа шифрлау жүйелерінде (мысалы, және xdc пішімі) симметриялық алгоритмдермен бірге қолданылады.

Шифрлау жылдамдығы төмен болғандықтан, хабарламалар әдетте кездейсоқ сеанс кілтімен (мысалы, AES, IDEA, Serpent, Twofish) тиімдірек симметриялық алгоритмдер арқылы шифрланады және тек осы кілт RSA көмегімен шифрланады, осылайша гибридті криптожүйе жүзеге асырылады. Кездейсоқ симметриялық шифрлау сеансының кілтін жасау үшін криптографиялық күшті псевдокездейсоқ сандар генераторын пайдалану қажеттілігіне байланысты мұндай механизм ықтимал осалдықтарға ие.

RSA алгоритмі AES және симметриялық блоктық шифрларды пайдаланатын басқа алгоритмдерге қарағанда әлдеқайда баяу.

Егер іске асыру дұрыс емес немесе оңтайлы емес болса немесе алгоритм пайдаланылса, кішігірім құпия көрсеткіші бар схемаларға немесе жалпы таңдалған модуль мәні бар схемаларға шабуылдар сияқты арнайы криптографиялық шабуылдар мүмкін.

Шифрлау жылдамдығы төмен болғандықтан (22 ГГц процессорында 512512 биттік кілтпен шамамен 3030 кбит) хабарламалар әдетте кездейсоқ сеанс кілті (мысалы, IDEA, Serpent, Twofish) бар жылдамырақ симметриялы алгоритмдер арқылы шифрланады және RSA шифрланады, тек осы кілт, осылайша гибридті криптожүйені жүзеге асырады.

Шын мәнінде, сипатталған шифрлау әдісі өте әлсіз және ешқашан пайдаланылмайды. Себебі, әріпті шифрлау қарапайым әдіс. Сол әріп бірдей нөмірмен шифрланады. Егер шабуылдаушы жеткілікті түрде үлкен хабарламаны ұстаса, оның мазмұнын болжай алады. Біріншіден, ол жиі бос орын кодтарына назар аударады және шифрлауды сөздерге бөледі. Содан кейін ол бір әріпті сөздерді байқап, «а», «және», «о», «с», «к» әріптерінің қалай кодталғанын болжайды. Қысқа іздеу арқылы ол қысқа сөздерден «бірақ», «емес», «by» сияқты қосымша әріптерді есептейді. Ал ұзағырақ сөздер үшін ол барлық қалған әріптерді оңай қалпына келтіреді.

Осылайша, шабуылдаушы жіберушінің құпия кілттерін болжамайды. Ол жіберілген хабарламаны білмей тұрып бұзуы мүмкін.

Бұған жол бермеу үшін арнайы қосымша алгоритмдер қолданылады, олардың мәні хабарламаның әрбір алдыңғы бөлігі келесіге әсер ететін бола бастайды.

Бұл жұмыста файлдарды криптографиялық қорғау әдістері мен құралдарына шолу және талдау жүргізілді; файлдармен жұмыс істеудің тікелей алгоритмдері мен схемаларын көрсететін алгоритмдік және программалық жасақтаманы жобалауды және оларды компьютерде программалық қамтамасыз ету жүзеге асырылды.

Бұл жұмыстың мақсаты криптографиялық деректерді қорғаудың программалық құралын құру және блоктық шифрлау алгоритмдерін құрудың заманауи тәсілдерін талдау, зерттелетін алгоритм негізінде программа құрылып, оны сынақтан өткізу іске асырылды.

Бұл жұмыста келесі әрекеттер орындалды:

- Python программалау тілінде деректерді шифрлау және дешифрлау алгоритмы құрыылып, оның негізінде программа құрылды;
- RSA модулі егізінде интегралды функциялар, Эйлер функциялары, GCD есептелді;
- жабық және ашық кілттерді идентификациялау және криптографиялық шифрлау алгоритмдері қолданылды;
- RSA шифрлау алгоритмі ұсынылып, сыналды.

Бұл программаны кез-келген қолданушылар да, кәсіпорын жұмыс станциялары да қолдана алады.

Дипломдық жұмысқа берілген тапсырма толығымен орындалды. Қойылған тапсырманы қанағаттандыратын программа құрылды. Программаның интерфейсі қажетті талапқа сай орындалған. Программа дизайны барлық эргономикалық шарттарға сәйкес құрылған. Программа сынақтан өтті және Windows 10, Linux және MacOS операциялық жүйелерінде жұмыс істейді. Программа деректерді шифрлауға және шифрын шешуге мүмкіндік береді, бұл олардың қауіпсіз пайдаланылуын қамтамасыз етеді және маңызды ақпаратты тасымалдау үшін қызмет ете алады.

ПАЙДАЛАНЫЛҒАН ӘДЕБИЕТТЕР ТІЗІМІ

1 https://github.com
2 https://habr.com
3 https://www.comparitech.com
4 https://brilliant.org
5 Вендров А.М. Проектирование программного обеспечения экономических информационных систем. – М.: Финансы и статистика, 2000. – 352с.
6 Канер С., Фолк Д., Кен Нгуен Е. Тестирование программного обеспечения: Пер. с англ. – Киев: ДиаСофт, 2000. – 544с.
7 С. Панасенко "Алгоритмы шифрования. Специальный справочник", 2009.
8 А.В. Аграновский, Р.А. Хади "Практическая криптография", 2009.
9 Н. Смарт "Криптография", 2005.
10 Б.Д. Кудряшов "Основы теории кодирования", 2016.
11 Дэвид Кан. Взломщики кодов. Централполиграф, 2000.
12 Венбо Мао. Современная криптография. Теория и практика. Уильямс, 2005.
13 Дэн браун. Цифровая крепость.2014.
14 А. В. Бабаш, Г. П. Шанкин. Криптография.2002.

Қосымшасы А

```
'''
ДОБРО ПОЖАЛОВАТЬ В ШИФРАТОР И ДЕШИФРАТОР, ОСНОВАННЫЙ НА АЛГОРИТМЕ RSA.

'''


import math

print("ШИФРАТОР/ДЕШИФРАТОР RSA")
print("**********************************************************")

#Input Prime Numbers
print("ВНЕСИТЕ 'p' И 'q' ЗНАЧЕНИЯ НИЖЕ:")
p = int(input("Внесите простое число для p: "))
q = int(input("Внесите простое число для q: "))
print("**********************************************************")

#Check if Input's are Prime
'''ЭТА ФУНКЦИЯ И КОД АВТОМАТИЧЕСКИ ПРОВЕРЯЮТ, ПРОСТОЕ ЛИ ЧИСЛО БЫЛО ВНЕСЕНО'''
def prime_check(a):
    if(a==2):
        return True
    elif((a<2) or ((a%2)==0)):
        return False
    elif(a>2):
        for i in range(2,a):
            if not(a%i):
                return False
    return True

check_p = prime_check(p)
check_q = prime_check(q)
while(((check_p==False)or(check_q==False))):
    p = int(input("Внесите простое число для p: "))
    q = int(input("Внесите простое число для q: "))
    check_p = prime_check(p)
    check_q = prime_check(q)
```

```
#RSA Modulus
'''ВЫЧИСЛЕНИЕ RSA МОДУЛЯ 'n'.'''
n = p * q
print("RSA Модуль(n) равен:",n)

#Eulers Toitent
'''ВЫЧИСЛЕНИЕ ФУНКЦИИ ЭЙЛЕРА 'r'.'''
r= (p-1)*(q-1)
print("Функция Эйлера(r) равна:",r)
print("**********************************************************")

#GCD
'''ВЫЧИСЛЕНИЕ НОД(наибольший общий делитель) ДЛЯ 'e'.'''
def egcd(e,r):
    while(r!=0):
        e,r=r,e%r
    return e

#Euclid's Algorithm
def eugcd(e,r):
    for i in range(1,r):
        while(e!=0):
            a,b=r//e,r%e
            if(b!=0):
                print("%d = %d*(%d) + %d"%(r,a,e,b))
            r=e
            e=b

#Extended Euclidean Algorithm
def eea(a,b):
    if(a%b==0):
        return(b,0,1)
    else:
        gcd,s,t = eea(b,a%b)
        s = s-((a//b) * t)
        print("%d = %d*(%d) + (%d)*(%d)"%(gcd,a,t,s,b))
        return(gcd,t,s)

#Multiplicative Inverse
def mult_inv(e,r):
```

```
        gcd,s,_=eea(e,r)
        if(gcd!=1):
            return None
        else:
            if(s<0):
                print("s=%d. Так как %d меньше 0, s = s(modr), т.е., s=%d."%(s,s,s%r))
            elif(s>0):
                print("s=%d."%(s))
            return s%r

    #e Value Calculation
    '"НАХОДИТ НАИБОЛЬШЕЕ ЗНАЧЕНИЕ 'e' МЕЖДУ 1 И 1000 ЧТО ДЕЛАЕТ (e,r) ВЗАИМНО ПРОСТЫМИ ЧИСЛАМИ."'
    for i in range(1,1000):
        if(egcd(i,r)==1):
            e=i
    print("Значение e равно:",e)
    print("**********************************************************")

    #d, Private and Public Keys
    '"ВЫЧИСЛЕНИЕ 'd', ЗАКРЫТЫЙ КЛЮЧ И ОТКРЫТЫЙ КЛЮЧ."'
    print("АЛГОРИТМ ЕВКЛИДА:")
    eugcd(e,r)
    print("ПОСЛЕДНИЕ ШАГИ ДЛЯ ВЫЧИСЛЕНИЯ АЛГОРИТМА ЕВКЛИДА.")
    print("**********************************************************")
    print("РАСШИРЕННЫЙ АЛГОРИТМ ЕВКЛИДА:")
    d = mult_inv(e,r)
    print("ПОСЛЕДНИЕ ШАГИ ДЛЯ ВЫЧИСЛЕНИЯ ЗНАЧЕНИЯ 'd'.")
    print("Значение d равно:",d)
    print("**********************************************************")
    public = (e,n)
    private = (d,n)
    print("Закрытый ключ:",private)
    print("Открытый ключ:",public)
    print("**********************************************************")

    #Encryption
    '"АЛГОРИТМ ШИФРОВАНИЯ."'
    def encrypt(pub_key,n_text):
        e,n=pub_key
```

```
    x=[]
    m=0
    for i in n_text:
        if(i.isupper()):
            m = ord(i)-65
            c=(m**e)%n
            x.append(c)
        elif(i.islower()):
            m= ord(i)-97
            c=(m**e)%n
            x.append(c)
        elif(i.isspace()):
            spc=400
            x.append(400)
    return x


#Decryption
'''АЛГОРИТМ ДЕШИФРОВАНИЯ.'''
def decrypt(priv_key,c_text):
    d,n=priv_key
    txt=c_text.split(',')
    x=''
    m=0
    for i in txt:
        if(i=='400'):
            x+=' '
        else:
            m=(int(i)**d)%n
            m+=65
            c=chr(m)
            x+=c
    return x

#Message
message = input("Внесите сообщение для шифрования или дешифрования.(Числа раздельно с ',' для дешифрования):")
print("Ваше сообщение:",message)

#Choose Encrypt or Decrypt and Print
choose = input("Напишите '1' for шифрования и '2' для дешифрования:")
```

```
if(choose=='1'):
    enc_msg=encrypt(public,message)
    print("Ваше шифрованное сообщение:",enc_msg)
    print("Благодарим за пользование шифратором RSA!")
elif(choose=='2'):
    print("Ваше дешифрованное сообщение:",decrypt(private,message))
    print("Благодарим за пользование дешифратором RSA!")
else:
    print("Вы внесли неправильное значение.")
    print("Ошибка!")
```

Қосымшасы Б

https://github.com/Noyan12341/RSA