

Қазақстан Республикасының Ғылым және жоғары білім министрлігі

«Л.Н. Гумилев атындағы Еуразия ұлттық университеті» КеАҚ

Бауыржан Элнұр Нұрболұлы

Веб-сайттың қауіпсіздігін қамтамасыз ету үшін криптографиялық алгоритмді  
зерттеу және әзірлеу

ДИПЛОМДЫҚ ЖҰМЫС

«6B06306 –Ақпараттық қауіпсіздік жүйелері» білім беру бағдарламасы

Астана, 2024

Қазақстан Республикасының Ғылым және жоғары білім министрлігі

«Л.Н. Гумилев атындағы Еуразия ұлттық университеті» КеАҚ

«Қорғауға жіберілді»  
«Ақпараттық қауіпсіздік»  
кафедрасының меңгерушісінің м.а.  
\_\_\_\_\_ Конырханова А.А.  
« \_\_\_\_ » \_\_\_\_\_ 2024ж.

## **ДИПЛОМДЫҚ ЖҰМЫС**

**Тақырыбы: «Веб-сайттың қауіпсіздігін қамтамасыз ету үшін  
криптографиялық алгоритмді зерттеу және әзірлеу»**

«6B06306 –Ақпараттық қауіпсіздік жүйелері» білім беру бағдарламасы  
бойынша

Жұмысты орындаған

Бауыржан Э.Н.

Ғылыми жетекші

Қоржаспаев А.Е.

Астана, 2024

## МАЗМҰНЫ

Кіріспе.....	6
1 Теория бөлімі.....	8
1.1 Заманауи веб қауіпсіздік .....	8
1.2 Веб қауіпсіздік үшін криптографияның рөлі.....	10
1.2.1 Веб қауіпсіздекке қолданылатын алгоритмдерге шолу.....	11
1.2.2 Криптоалгоритмдердің механизмі.....	13
1.3 Криптоалгоритмді бұзу әдістері.....	24
2 Тәжірибе бөлім.....	28
2.1 Криптографиялық алгоритмдердің веб-сайтқа енгізілуі.....	28
2.1.1 SSL/TLS сертификатын орнату.....	33
2.2 Криптографиялық алгоритмді құру.....	37
2.2.1 Алгоритмді жобалау, мүмкін қауіптердің алдын алу.....	37
2.2.2 Алгоритмді код түрінде жүзеге асыру.....	43
2.3 Криптоалгоритм қабілеті, оң және теріс жақтары.....	44
Қорытынды.....	46
Пайдаланылған әдебиеттер тізімі.....	48

«Л.Н. Гумилев атындағы Еуразия ұлттық университеті» КеАҚ

Ақпараттық технологиялар факультеті

«Ақпараттық қауіпсіздік» кафедрасы

«6B06306 – Ақпараттық қауіпсіздік жүйелері» білім беру бағдарламасы

**Бекітемін**

«Ақпараттық қауіпсіздік»  
кафедрасының меңгерушісі м.а.

Конырханова А.А. \_\_\_\_\_

«\_\_» \_\_\_\_\_ 20\_\_ ж.

**Дипломдық жұмысты (жобаны) орындауға  
ТАПСЫРМА**

Бауыржан Элнұр, 4 курс, АҚЖ-43, Ақпараттық қауіпсіздік жүйелері білім беру бағдарламасы, күндізгі оқу бөлімі.

1. Дипломдық жұмыстың (жобаның) тақырыбы: «Веб-сайттың қауіпсіздігін қамтамасыз ету үшін криптографиялық алгоритмді зерттеу және әзірлеу» 11 қаңтар 2024 ж. №40-п университет бұйрығымен бекітілген.

2. Білім алушының аяқталған жұмысты тапсыру мерзімі « » сәуір 2024 ж.

3. Жұмысқа қажетті бастапқы деректер:

- 1) TLS протоколы
- 2) RSA, AES, SHA криптоалгоритмдерінің кезеңдері мен математикалық операциялары
- 3) Жасанды HTTP және HTTPS веб-сайттары

4. Дипломдық жұмыста (жобада) жасалатын тақырыптар тізімі

- 1) Криптоалгоритмдердің веб-сайтқа енгізу жолдарын зерттеу
- 2) Криптоалгоритмдермен деректерді шифрлау
- 3) Жаңа криптоалгоритм әзірлеу

5. Графикалық материалдар тізбесі (сызбалар, кестелер, диаграммалар және т.б.):

Сурет – 19;

Кесте – 5.

## 6. Ұсынылатын негізгі әдебиеттер тізімі

1) Фергюсон Нильс. Практическая криптография. // Шнайер Б. — Москва: Вильямс, 2004. — 432 б.

2) Заурбеков С.С., Отелбаев М. Защита информации и основы криптографии (каз.). — Астана, 2003. — 381 б.

## 7. Жұмыс бойынша консультациялар

Бөлімнің (тараудың) нөмірі, атауы	Ғылыми жетекші, консультант	Тапсырманы алу мерзімі	Тапсырма берілді (қолы)	Тапсырма қабылданды (қолы)
Кіріспе	Қоржаспаев А.Е.	24.01.2023		
Теория бөлімі	Қоржаспаев А.Е.	06.02.2023		
Тәжірибе бөлімі	Қоржаспаев А.Е.	10.02.2023		
Қорытынды	Қоржаспаев А.Е.	01.04.2023		
Қолданылған әдебиеттер тізімі	Қоржаспаев А.Е.	01.04.2023		
Нормабақылау	Тоқсеит Д.Қ.	19.04.2023		

## 8. Дипломдық жұмысты (жобаны) орындау кестесі

№	Жұмыс кезеңдері	Жұмыс кезеңдерін орындау мерзімдері	Ескерту
1	Дипломдық жұмыстың (жобаның) тақырыбын бекіту	26.12.2022	
2	Дипломдық жұмысты (жобаны) дайындау үшін материалдар жинау	24.01.2023	
3	Дипломдық жұмыстың (жобаның) теориялық бөлігін дайындау (1-бөлім)	06.02.2023	Практикаға кеткенге дейін
4	Дипломдық жұмыстың (жобаның) талдамалық бөлігін (2-3-бөлімдер) Дайындау	06.03.2023	Практика кезінде
5	Дипломдық жұмыстың (жобаның) толық мәтінінің жоба нұсқасын аяқтау	01.04.2023	Практика аяқталғаннан кейінгі бірінші аптада
6	Алдын ала қорғауға дипломдық жұмысты (жобаны) ұсыну	13.04.2023	Шолу дәрістері (консультациялар) кезінде

7	Дипломдық жұмысты (жобаны) ұсыну рецензияға	10.05.2023	
8	Ғылыми жетекшінің пікірімен және рецензиясымен дипломдық жұмыстың (жобаның) түпкілікті нұсқасын ұсыну	15.05.2023	
9	Дипломдық жұмысты қорғау (жобаның)	23.05.2023	МАК кестесіне сәйкес

Тапсырманың берілген мерзімі «29» 12 2023 ж.

Ғылыми жетекшісі

Қоржаспаев А.Е.

Тапсырманы орындаған

Бауыржан Э.Н.

## Кіріспе

### Тақырыптың өзектілігін негіздеу

Заманауи технологиялардың дамуы және интернеттің кеңеюі веб-сайттар арқылы берілетін ақпарат көлемінің айтарлықтай өсуіне әкелді. Дегенмен, осы өсіммен қатар, құпия деректерге рұқсатсыз кіру, бұзу және кибершабуылдармен байланысты қауіпсіздік қатерлері де өсті. Қазіргі таңда кәсіпорындар болсын, дүкендер болсын, кез келген веб-сайт үшін қауіпсіздік енгізу жаһандық стандартқа айналды.

Қолданыстағы криптографиялық алгоритмдер және SSL/TLS сияқты қауіпсіздік протоколдары клиенттер мен серверлер арасында тасымалданатын деректерді қорғау үшін пайдаланылады. Дегенмен, шабуыл әдістерінің үнемі дамуына және есептеу қуатының артуына байланысты қолданыстағы алгоритмдер осал болуы мүмкін. Бұл заманауи шабуылдарға тиімді қарсы тұра алатын және веб-сайт қауіпсіздігінің жоғары деңгейін қамтамасыз ететін жаңа криптографиялық алгоритмдерді үздіксіз зерттеу және әзірлеу қажеттілігін көрсетеді.

Интернеттің адамдардың күнделікті өмірінде, соның ішінде онлайн-банкингте, электронды коммерцияда және медициналық деректерді беруде маңыздылығы артып келе жатқандықтан, ақпараттың құпиялылығы мен тұтастығын қамтамасыз ету өзекті мәселеге айналды. Бұл тек сенімді ғана емес, сонымен қатар жоғары жүктеме жағдайында тиімді жұмыс істеуге және өсіп келе жатқан интернет-трафик қажеттіліктерін қанағаттандыру үшін масштабтауға қабілетті криптографиялық алгоритмдерді құру қажеттілігін көрсетеді.

Осылайша, веб-сайттың қауіпсіздігін қамтамасыз ету үшін жаңа криптографиялық алгоритмдерді зерттеу және әзірлеу ақпараттық қауіпсіздікпен және Интернет инфрақұрылымының сенімділігін қамтамасыз етумен тікелей байланысты, өзекті және маңызды тақырып болып табылады. Мұндай алгоритмдерді әзірлеу желідегі қауіпсіздік деңгейін арттырып қана қоймайды, сонымен қатар цифрлық экономиканың одан әрі дамуына және пайдаланушылардың құпиялылығын қамтамасыз етуге ықпал етеді.

### Зерттеудің мақсаты мен міндеттері

Ұсынылып отырған зерттеудің мақсаты – веб-сайттардың қауіпсіздігін қамтамасыз ету үшін арнайы бейімделген жаңа криптографиялық алгоритмді әзірлеу. Осы мақсатқа жету үшін келесі міндеттерді орындау ұсынылады: олардың артықшылықтарын, кемшіліктері мен осалдықтарын анықтау мақсатында қолданыстағы криптографиялық алгоритмдерді талдау; веб-қауіпсіздіктің негізгі аспектілерін ескере отырып, жаңа алгоритмге қойылатын талаптарды анықтау; әзірленген әдістің математикалық моделін және алгоритмдік схемасын жобалау; алгоритмнің бағдарламалық нұсқасын әр түрлі пайдалану сценарийлерінде және әртүрлі шабуыл түрлерінде оның тиімділігі мен сенімділігін кейіннен сынай отырып жүзеге асыру.

Әрі қарай, заманауи стандарттар мен қауіпсіздік талаптарын ескере отырып, әзірленген алгоритмнің қауіпсіздігі мен қолданылуын бағалау, сондай-ақ әзірлеу процесін, алынған нәтижелерді және зерттеу барысында жасалған талдауды сипаттайтын ғылыми мақалалар мен есептер дайындау ұсынылады. Бұл веб-сайттарды заманауи киберқауіптерден тиімді қорғайтын, қауіпсіздіктің жоғары деңгейін және пайдаланушы деректерінің құпиялылығын қамтамасыз ететін инновациялық криптографиялық әдісті жасайды.



## 1 Теория бөлімі

### 1.1 Заманауи веб қауіпсіздік

Ең алғашқы әлемдік сайт 1991 жылы шығарылды. Ол кезде арадағы ақпараттың тұтастығы мен құпиялығы маңызды емес болды, себебі басым назар сайттың қарайымдылығы пен қолжетімділігі болды. Ол кезде желі шабуылдары да әлі дамымаған еді, адамдар мен компаниялар желіде ақпарат тасымалдауда ешқандай қауіп сезінбеді. Алайда ақпарат құндылығы жылдан жылға артты, мысалы компания клиенттерінің жеке деректері немесе құпия сөздер. 1999 жылы TLS протоколы жаһандық пайдалануға шығарылды. Сол мезеттен бастап үлкен компаниялар немесе қауіпсіздікке алаңдаған сайттар TLS протоколын пайдалана бастады. Кейбір факторлар әсерінен көптеген веб-сайттар желідегі қауіпсіз ақпарат тасымалын қажет деп санамады. Ол факторлар: сайт компанияны таныстыру мақсатында құрылған, яғни құпия ақпарат жоқ; артық шығын, яғни сертификат алуға төлемақы қажет; өнімділік төмендеуі, яғни ақпаратты жібері үшін шифрлау қажет, ал оқу үшін дешифрлау қажет. TLS халықаралық қауіпсіздік стандартына айналуы компаниялардың оған көшуіне ықпал етті. Бірақ HTTPS-ке көшу шын мәнінде айтарлықтай өсуі шамамен 2014 жылы басталды және бүгінгі күнге дейін жалғасуда. Себебі Google HTTPS қолданылатын сайттардың рейтингі браузердің іздеу жүйесінде артықшылыққа ие болатындығын жариялады. Қазіргі таңда кез-келген веб-сайт немесе веб-бағдарламалар қауіпсіздікті талап етеді[1].

Клиент пен сервер арасындағы қауіпсіз жалғауды TLS протолы толықтай қамтамасыз етеді (1-кесте).

#### Кесте 1

##### TLS клиент-сервер жалғау механизмі

Сервер клиентке шифрлау параметрлерін жібереді:	Сервер клиентке қолдау көрсетілетін шифрлар тізімін және басқа TLS/SSL параметрлерін жіберу арқылы қол алысуды бастайды.
Клиент шифрлар жиынтығын таңдап, серверге жібереді:	Клиент сервер ұсынған тізімнен сәйкес шифрлар жиынтығын таңдап, оны серверге жібереді.
Сервер кілттерді жасайды және клиентке ашық кілтті жібереді:	Сервер кілттер жұбын жасайды: ашық және жеке кілттер. Сервер клиентке оның сұранысына жауап ретінде сертификатпен бірге ашық кілтті жібереді.

Клиент кілттерді жасайды және өзінің ашық кілтін серверге жібереді:	Клиент сонымен қатар өзінің кілт жұбын жасайды: ашық және жабық кілттер. Клиент өзінің ашық кілтін сервердің ашық кілті арқылы шифрланған түрде жібереді.
Сервер клиенттің ашық кілтін шешеді және сеанс кілтін жасайды:	Сервер клиенттің шифрланған ашық кілтін алады және оның жеке кілті арқылы шифрын ашады. Сервер сеанс кезінде деректерді симметриялы шифрлау үшін пайдаланылатын сеанс кілтін жасайды.
Сервер клиентке шифрланған сеанс кілтін жібереді:	Сервер клиенттің ашық кілтін пайдаланып сеанс кілтін шифрлайды және оны клиентке жібереді.
Клиент сеанс кілтін шешеді:	Клиент шифрланған сеанс кілтін алады және оның жеке кілтін пайдаланып оның шифрын ашады.
Әрі қарай деректер алмасу сеанс кілті арқылы жүзеге асады:	Клиент пен сервер енді сеанс кезінде олардың арасында өткен деректерді симметриялы түрде шифрлау және шифрын ашу үшін сеанс кілтін пайдалана алады.

Демек, кілт алмасу барысында асимметриялық криптоалгоритм қолданылады, ал ақпарат алмасу үшін симметриялық криптоалгоритмдер жеткілікті. Асимметриялық криптоалгоритмдердің қауіпсіздік деңгейі жоғары болғанымен, жылдамдығы симметриялық криптоалгоритмдерден әлдеқайда баяу. Сол себептен ортақ сеанстық кілт алмасу үшін ғана асимметриялық криптоалгоритмді қолданады, екі жақ та кілтті білген соң сол кілт арқылы симметриялық шифрлауды қолданып, еркін ақпарат алмасады.

## 1.2 Веб қауіпсіздік үшін криптографияның рөлі

Криптография веб-қосымшалар мен қызметтердің қауіпсіздігін қамтамасыз етуде маңызды рөл атқарады.

Криптографиялық шифрлау веб-бағдарлама деректерінің құпиялылығын қорғау үшін қолданылады. Бұл қауіпсіз қосылымды қамтамасыз ететін және пайдаланушы браузері мен веб-сервер арасында жіберілген деректерді шифрлайтын HTTPS (SSL/TLS) сияқты протоколдарды пайдаланып клиент пен сервер арасында жіберілетін ақпаратты қорғауды қамтиды.

Құпия сөзді хэштеу дерекқорларда пайдаланушы құпия сөздерін сақтаудың маңызды механизмі болып табылады. Егер хэштеу дұрыс пайдаланылса, дерекқорға шабуылдаушы қол жетімді болса да, ол бастапқы құпия сөздерді таба алмайды. Қауіпсіздікті арттыру үшін тұздарды және баяу хэштеу алгоритмдерін (мысалы, bcrypt, Argon2) пайдалану ұсынылады.

Цифрлық қолтаңбалар аутентификация және деректердің тұтастығын тексеру үшін қолданылады. Олар хабарлама жіберушінің түпнұсқалығын растау мүмкіндігін қамтамасыз етеді және деректердің жіберу кезінде өзгертілмегенін растайды. Сандық қолтаңба әсіресе қаржылық транзакциялар немесе аутентификация деректері сияқты құпия ақпаратты беру және сақтау кезінде маңызды.

Криптографиялық кілттер деректерді шифрлау және дешифрлау үшін қолданылады. Криптографиялық алгоритмдердің қауіпсіздігі кілттерді генерациялау сапасына байланысты. Кілттерді генерациялау кездейсоқ және криптографиялық қауіпсіз болуы керек. Бұған негізгі өмірлік циклды басқару кіреді, соның ішінде кілттерді жаңарту және мерзімді айналдыру.

OAuth сияқты криптографиялық аутентификация протоколы ресурстарға қолжетімділікті қауіпсіз және ыңғайлы басқару және веб-қосымшалардағы пайдаланушыларды анықтау үшін пайдаланылады. Бұл протоколдар арасындағы деректер мен қызметтерге қолжетімділікті қауіпсіз өкілдеуге мүмкіндік береді.

Интернетте сенімділік пен аутентификацияны қамтамасыз етуде цифрлық сертификаттар маңызды рөл атқарады. Олар веб-серверлердің аутентификациясы және SSL/TLS хаттамалары арқылы қауіпсіз байланыс арналарын жасау үшін пайдаланылады. Сертификаттарға енгізілген ашық кілттер деректерді шифрлау және қолтаңбаларды тексеру үшін пайдаланылады.

TLS (Transport Layer Security) — клиент пен сервер арасындағы қауіпсіз қосылымды қамтамасыз ететін қауіпсіздік протоколы. Ол RSA, AES және SHA хэштеу алгоритмдері сияқты криптографиялық алгоритмдерге негізделген. TLS желі арқылы қозғалған кезде деректердің құпиялылығы мен тұтастығын қорғау үшін қолданылады[2-4].

Криптоалгоритмдерді қолданар алдында бірнеше қауіпсіздік талаптары мен аспектілерді ескерген жөн (2-кесте).

Кесте 2  
Криптоалгоритмдер аспектілері

Күш деңгейі	Жіберілетін деректердің құпиялылығы мен сезімталдық деңгейіне байланысты әртүрлі күш деңгейлері бар алгоритмдер қажет болуы мүмкін. Мысалы, қаржылық деректерді беру жалпыға қолжетімді ақпаратты беруден гөрі күштілік деңгейі жоғары алгоритмдерді қажет етуі мүмкін.
Өнімділік	Кейбір криптографиялық алгоритмдер басқаларына қарағанда өнімдірек болуы мүмкін. Алгоритмдерді таңдаған кезде өнімділік қажеттіліктері мен мақсатты платформада қолжетімді ресурстарды ескеру қажет.
Үйлесімділік	Таңдалған алгоритмдердің олар өзара әрекеттесуі керек бар жүйелермен және протоколдармен үйлесімділігін ескеру маңызды. Мысалы, егер жүйе байланыстарды қорғау үшін TLS протоколын бұрыннан пайдаланса, онда осы протокол қолдайтын криптографиялық алгоритмдерді таңдау керек.
Іске асыру және техникалық қызмет көрсету шығындары	Кейбір криптографиялық алгоритмдер күрделірек енгізуді және техникалық қызмет көрсетуді қажет етуі мүмкін, бұл жүйені әзірлеу және техникалық қызмет көрсету шығындарын арттыруы мүмкін.

### 1.2.1 Веб қауіпсіздекке қолданылатын алгоритмдерге шолу

Деректерді қорғау, құпиялылықты, тұтастықты және аутентификацияны қамтамасыз ету үшін веб-қауіпсіздікте қолданылатын көптеген криптографиялық алгоритмдер бар. Сонымен қатар интернет желісіне қауіпсіз байланыс ортану ең маңызды бөлім, ол үшін TLS протоколы қолданылады (3-кесте).

Кесте 3  
 TLS қолдайтын алгоритмдер

TLS протоклы қолдайтын криптографиялық алгоритмдер 4 топқа жіктеуге болады.	
Шифрлау алгоритмдері:	AES (Advanced Encryption Standard)
	ChaCha20
	Camellia
	3DES (Triple DES)
	RC4 (Rivest Cipher 4)
Хэштеу алгоритмдері:	SHA-256, SHA-384, SHA-512
	SHA-3 (Keccak)
	MD5 (Message Digest Algorithm 5)
	SHA-1 (Secure Hash Algorithm 1)
Кілт алмасу алгоритмдері:	RSA (Rivest-Shamir-Adleman)
	Diffie-Hellman
	Elliptic Curve Diffie-Hellman (ECDH)
	DSA (Digital Signature Algorithm)
	ECDSA (Elliptic Curve Digital Signature Algorithm)
Цифрлық қолтаңба алгоритмдері:	RSA
	DSA
	ECDSA

3-кестеде тек TLS протоколы қолдайтын алгоритмдердің бір бөлігі. Қажет криптоалгоритмдерді таңдаған кезде, сол мезетке актуалды алгоритмдер таңдалыданы. Себебі криптоалгоритмдер жылдан жылға жаңарып және ескі алгоритмдердің беріктігі төмендейді[5].

RSA (Ривест-Шамир-Адлеман) - ең танымал асимметриялық алгоритмдердің бірі. Ол шифрлау және цифрлық қолтаңбалау үшін қолданылады. RSA екі кілттің тіркесімін пайдаланады: ашық және жеке (жабық). Ашық кілт

деректерді шифрлау үшін, ал жабық кілт оны ашу үшін қолданылады. Артықшылықтарына келсек, іске асыру және қолдану салыстырмалы түрде оңай. Көптеген криптографиялық кітапханалар мен бағдарламаларда жақсы қолдау көрсетіледі. Кемшілігі өнімділігінің баяулығында. Себебі, жақсы қауіпсіздікке жету үшін ұзын кілттер пайдалану керек болады.

AES (Advanced Encryption Standard) – веб-қауіпсіздікте кеңінен қолданылатын симметриялық шифрлау алгоритмі. Ол бұзуға жоғары қарсылықты қамтамасыз етеді және әртүрлі платформаларда тиімді жұмыс істейді. AES жадтағы, транзакция ақпаратындағы және сеанс кілттеріндегі деректерді шифрлау үшін қолданылады. Артықшылығы заманауи құрылғылар мен процессорлар арқылы жақсы жұмыс жасайды, әрі көп кітапханалар мен протоколдар қолдайды. AES үш түрлі кілт өлшемін қолдайды: 128, 192 және 256 бит. AES алгоритмінің қауіпсіздігі кілттің ұзындығына тікелей байланысты: кілт неғұрлым ұзағырақ болса, оны бұзу қиынырақ болады. Кілттер кездейсоқ түрде жасалады және деректерді шифрлау және шифрды шешу үшін негіз болып табылады. Алайда кілт өлшемі үлкен болғандықтан жылдамдық пен өнімділікке кері әсер етеді.

SHA (Secure Hash Algorithm): SHA-1, SHA-256, SHA-384 және SHA-512 сияқты алгоритмдердің SHA тобы хабарлардың немесе деректердің хэш мәнін есептеу үшін пайдаланылады. Олар деректердің тұтастығын, аутентификацияны және цифрлық қолтаңбаны қамтамасыз ету үшін қолданылады. Артықшылығы үлкен битті таңдаған сайын коллизияға (қайталану) ұшырау қаупі өте төмен.

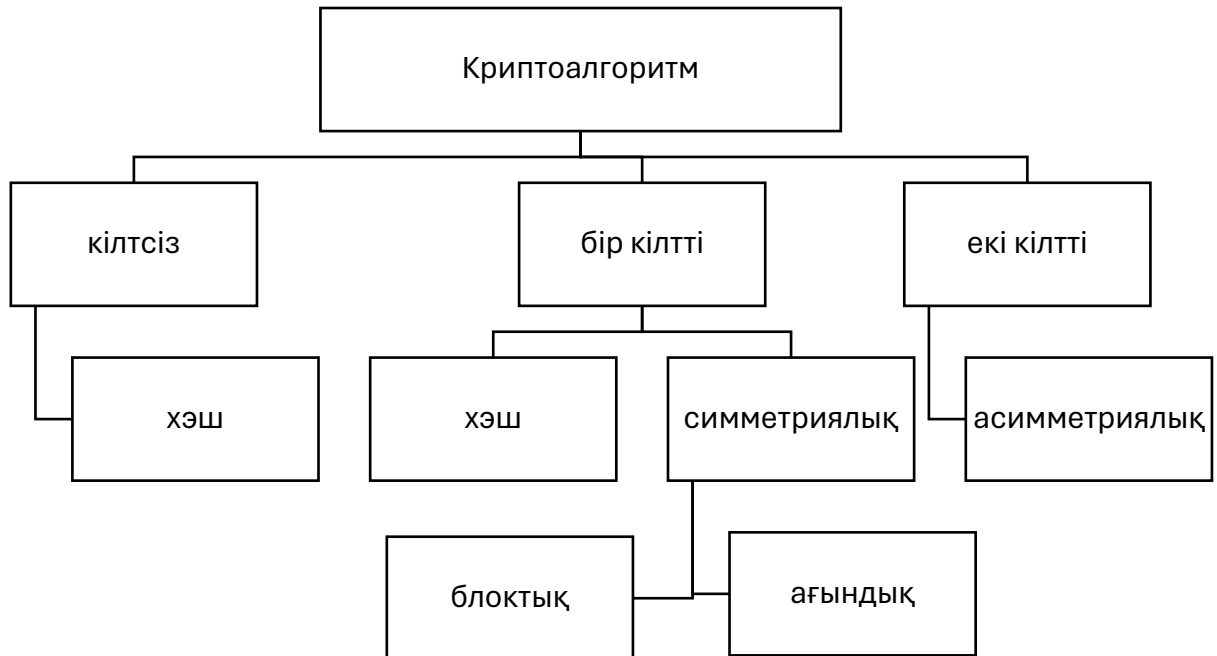
HMAC (Keyed-Hash Message Authentication Code) хэш функциясы мен құпия кілт негізінде MAC есептеу әдісі болып табылады. Ол OAuth, HTTP, IPsec және т.б. сияқты әртүрлі протоколдарда хабарламаның аутентификациясын қамтамасыз ету үшін кеңінен қолданылады[6].

ECC (Elliptic Curve Cryptography) – бүтін сандардың орнына эллиптикалық қисықтарды пайдаланатын балама асимметриялық шифрлау алгоритмі. Ол қысқа кілттермен бірдей қауіпсіздік дәрежесін қамтамасыз етеді, бұл оны RSA-ға қарағанда тиімдірек етеді. Яғни артықшылығы оның жылдамдығы, ал кемшілігі жүйге орнату, енгізу күрделілігі. Сол себепті әзірге көп орталар қолдамауы мүмкін.

### 1.2.2 Криптоалгоритмдердің механизмі

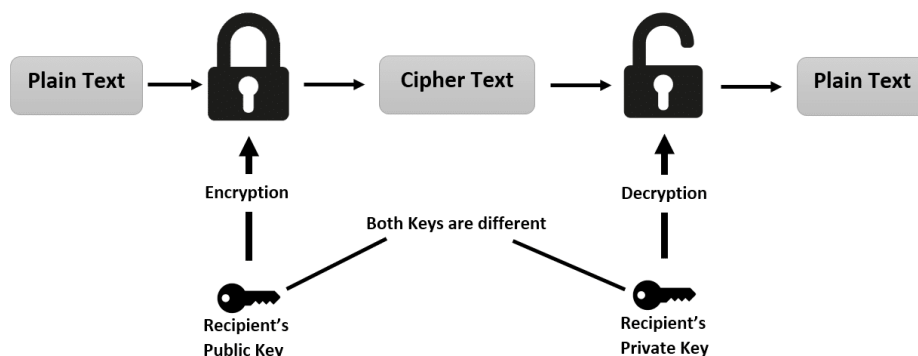
Шифр біздің заманымызға дейін де бар болды. Олардың ең танымалы – Цезарь. Ол кілттік сөзге байланысты әріптерді жылжытып, әріптерді алмастырудан тұрды. Ал криптоаналитикамен ғалымдар IV ғасырдан бастап жұмыстар жасай бастаған. Әлі күнге дейін криптография мен криптоанализ қатар дамып келеді. Яғни, Цезарь шифры қазіргі таңда ақпаратқа ешқандай қауіпсіздік бермейді. Цезарь шифрына brute force пен жиілікті талдау оңай шабуыл жасайды. Brute force барлық мүмкін пернелерді қолданып көруді білдіреді[7]. Жиілік

талдауы статистикаға негізделген: табиғи тілде кейбір әріптер басқаларға қарағанда жиі кездеседі және бұл кодты бұзу үшін пайдаланылуы мүмкін. Криптоалгоритмдерді қолдану аясы мен механизмдеріне байланысты топтарға жіктеуге болады (1-сурет).



**Сурет 1** Криптоалгоритмдердің жіктелуі.

Веб-сайт қауіпсіздігін қамтамасыз ету барысында сеанстық кілт тасымалдау үшін қолданылатын, асимметриялық криптоалгоритмдердің бірі – RSA (2-сурет).



**Сурет 2** RSA.

Мысал:

Ең алдымен екі жай сан “p, q” таңдалып алынады. (2, 3, 5, 7, 11, 13, 17, 19, 23, 29,...)

p - 3,

q - 7 делік. Ал  $n = p \cdot q$ ,

n - 21, бұл болашақ модуль, яғни кілттерді жарты бөлігі:

- Ашық кілт – (e, n);
- Жеке кілт (жабық) – (d, n).

Модуль қаншалықты үлкен сан болса, кілттерді анықтау сәйкесінше қиын болады. Мысалы 21-ді құрайтын жай сандарды табу үшін 21-ден кіші жай сандарға бөлу қажет. 21 саны 2-ге бөлінбейді, 3-ке бөлінеді, осы арқылы p-3, q-7 немесе керісінше екенін анықтау оңай. Ал нағыз бағдарламада кемінде 2024 биттік кілттер пайдаланылады, яғни ондық жүйге келтірсек, 600 таңбадан астам сан қолданылады. Сондықтан оны жай сандарға жіктеп p, q анықтау тіпті компьютер үшін өте ұзақ уақыт алады.

Келесі қадам Эйлер функциясын анықтау:

$$\varphi(N) = (p - 1) \cdot (q - 1) \quad (1)$$

$\varphi = 12$  – таңдалынған мән

Келесі қадам кілттерді анықтау, ашық кілттен бастасақ (e), кілт келесі шарттарды қамту қажет:

1. Жай сан;
2.  $e < \varphi$  ;
3.  $\text{EYOB}(e, \varphi)=1$

(5, 7, 11) – сандары үш шартты да бұзбайды, яғни ашық кілт ретінде бірейін таңдауға болады.

e - 5,

{5, 21} – ашық кілт.

Ал жеке кілтті анықтау үшін:

$$(d * e) \bmod(\varphi) = 1 \quad (2)$$

d-17 делік,  $17 * 5 \bmod(12) = 1$  орындалады,  $85 \% 12 = 1$ .

{17, 21} - Жеке кілт.

Шифрлау:

$$C = m^e \bmod(N) \quad (3)$$

m – хабарлама. “h” латын әріпі, Ascii-код бойынша 104 сандық таңбасымен белгіленген. Яғни m=104;

C – шифр мәні.

$$C = 104^5 \bmod(21) = 16$$

Дешифрлау:

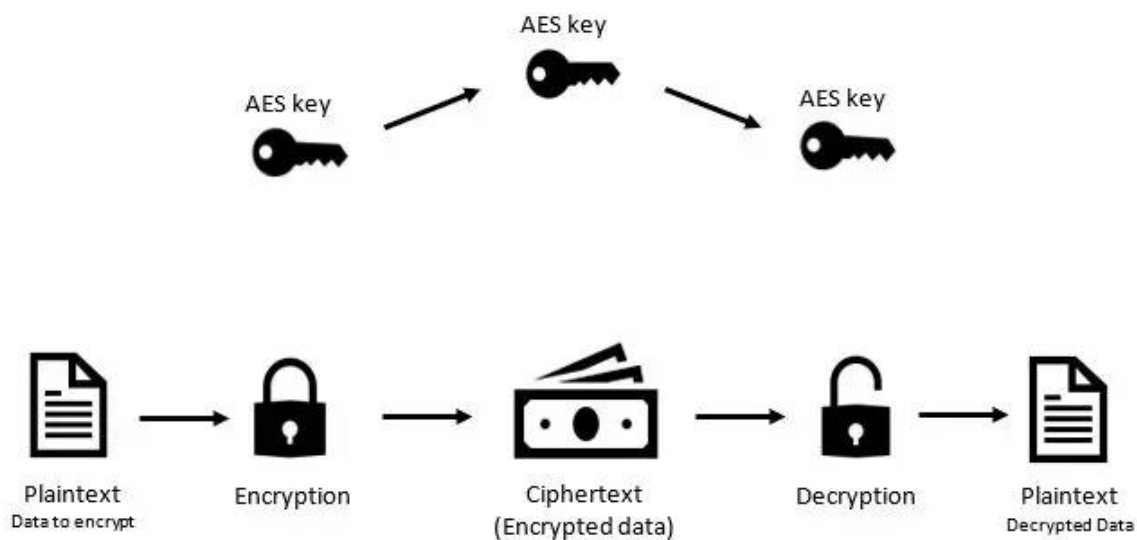
$$m = C^d \bmod(N) \quad (4)$$



$$m = 16^{17} \bmod(21) = 104$$

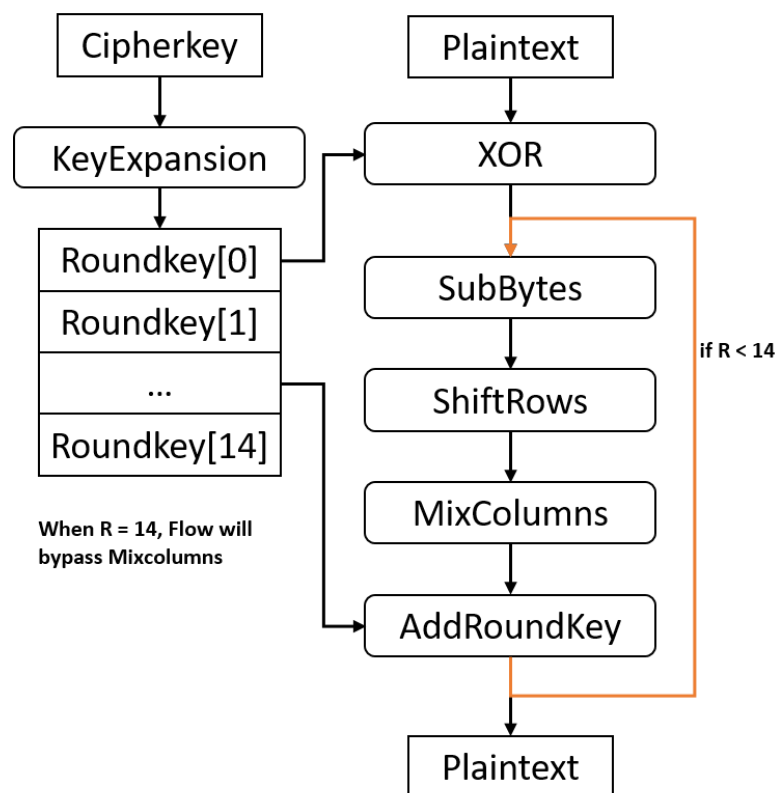
Ал егер жеке кілт болмаса, дешифрлау өте қиын[8].

Веб-сайт қауіпсіздігін қамтамасыз ету барысында сеанстық кілтті пайдаланып ақпаратты шифрлауға және дешифрлауға арналған симметриялық криптоалгоритмдердің бірі – AES (3-сурет)[9].



Сурет 3 AES.

AES өте күрделі және көп раундтардан тұратын криптоалгоритм. Әрбір раунд бірнеше кезеңнен тұрады: байтты ауыстыру, жолды ауыстыру, бағандарды араластыру және раундтық кілтті қосу. Байтты ауыстыру блокағы әрбір байтты басқасына ауыстыратын сызықты емес S-бокс көмегімен орындалады[10]. Жолды ауыстыру әрбір жолдағы байттарды белгіленген позициялар санын солға жылжытады. Бағандарды араластыру блок бағандарын бекітілген матрицаға көбейту арқылы орындалады. Раундтық кілтті қосу кілт пен деректер блогын разряд бойынша қосу арқылы орындалады. Әрбір раунд негізгі шифрлау кілтінен алынған өзінің бірегей дөңгелек кілтін пайдаланады. Процестің соңында барлық айналымдар аяқталғаннан кейін алынған шифрлық мәтінді беруге немесе сақтауға дайын болады. Кілт 128 бит болғандық 16 әріптен тұратын сөздер жиыны кілт етіп таңдалады. Әрбір әріп ол бір байтпен өлшенеді. Шифрлатын текст кілт өлшемімен бір яғни 16 байт, 4\*4 матрицаларға бөлінеді. Толмаған жағдайда бос орындарға арнаулы әдістемелер арқылы белгілі бір байт өлшемдерімен толықтырылады (4-сурет).



Сурет 4 AES механизмі.

Мысал:

Құпия кілт: “Thats my Kung Fu”

Құпия кілттің байт түрінде матрицаға орналасуы:

```

[84 104 97 116]
[115 32 109 121]
[32 75 117 110]
[103 32 70 117]

```

Ашық текст: “Elnur today...”

Ашық текст өлшемі 16 байтқа жетпейді, сондықтан танымал толтыру әдісі PKCS#7 padding арқылы берілген 14 байтқа “2” мәнімен 2 байт қосылады:

```

[69 108 110 117 114 32 116 111 100 97 121 46 46 46 2 2]

```

Алгоритмды бастау үшін S-box, Rcon және Mixcolumns кестелері құрылады. Кесте кездейсоқ байттармен (нағыз қолданыста оналтылық сандар жүйесінде) толтырылмайды. S-box – SubBytes қадамында қолданылатын сызықты емес ауыстыру кестесі. Ол шифрлау процесінде криптографиялық беріктік және диффузияны қамтамасыз ету үшін арнайы әзірленген. S-box шифрға шабуыл жасауды қиындату үшін байтты ауыстырулардың кездейсоқ және сызықты емес екендігін қамтамасыз ететін математикалық түрлендірулер негізінде жасалған. Бұл кезеңдер әртүрлі шабуылдар мен ықтимал қауіпсіздік қатерлерін ескере отырып, AES шифрлау алгоритмінің қауіпсіздігі мен

тиімділігін қамтамасыз ететіндей етіп құрылды. Олар сенімділігі мен ұзақ мерзімділігін қамтамасыз ету үшін мұқият талдау мен сынақтан кейін AES стандартында анықталған.

S-box (16\*16 матрица, 256 байт, ыңғайлылық үшін ондық жүйеде):

{82, 9, 16, ..., 251}  
{124, 227, 57, ..., 203}  
...  
{23, 43, 4, ..., 125}

Mixcolumns:

{02, 03, 01, 01}  
{01, 02, 03, 01}  
{01, 01, 02, 03}  
{03, 01, 01, 02}

Rcon (Round Constant):

{0x00, 0x00, 0x00, 0x00}  
{0x01, 0x00, 0x00, 0x00}  
{0x02, 0x00, 0x00, 0x00}  
{0x04, 0x00, 0x00, 0x00}  
...  
{0x1b, 0x00, 0x00, 0x00}  
{0x36, 0x00, 0x00, 0x00}

Шифрлау механизмі 10 раундтан тұратындықтан, 10 түрлі раундық кілттер әзірлеу қажет. Алғашқы құпия кілт негізінде қалған кілттер жасалынады:

[84 104 97 116]  
[115 32 109 121]  
[32 75 117 110]  
[103 32 70 117]

1-ші раунд кілтін әзірлеу механизмі. Соңғы бағанды [103 32 70 117] s-box кестесімен сәйкесінше алмасады, яғни

103 -> 133,  
32 -> 183 ,  
70 -> 90 ,  
117 ->157

Әрі қарай XOR операциясын қолданамыз:

84 (алдыңғы кілттің бірінші бағанынан) ^ (XOR) 133 (алдыңғы кілттің соңғы бағанынан s-box мәнінен) ^ (XOR) 1 (раунд константасынан) = 208 ,

104 ^ 183 ^ 0 = 223  
97 ^ 90 ^ 0 = 59  
116 ^ 157 = 223

Бұл бірінші кілттің бірінші бағаны болады:

```
[208 223 59 233]
[0   0   0   0   ]
[0   0   0   0   ]
[0   0   0   0   ]
```

Екінші бағанды анықтау үшін бірінші баған мен алдыңғы раунд кілтінің екінші бағанына XOR операциясын қолданылады. Дәл солай үшінші бағанды анықтау үшін екінші баған мен алдыңғы кілттің үшінші бағанын XOR-лаймыз. Сонда,  $163 = 208 \wedge 115$ ,

$$255 = 223 \wedge 32,$$

$$86 = 59 \wedge 109,$$

$144 = 233 \wedge 121$ , екінші баған анықталды, дәл осылай үшінші мен төртінші бағандар да анықталады. Шыққан мән:

```
[208 223 59 233]
[163 255 86 144]
[131 180 35 254]
[228 148 101 139],
```

бірінші раундтық кілт, осы алгоритм бойынша тағы 9 раундтық кілттер анықталу қажет.

Шифрды жүзеге асыру үшін 10 раунд жүргізіледі. Әр раунд subBytes, shiftRows, mixColumns, addRoundKey қадамдарынан тұрады, тек 10-шы раундта mixColumns қадамы қажет етілмейді. Хабарлама күйі (деректер блогы):

```
[69 114 100 46]
[108 32 97 46]
[110 116 121 2]
[117 111 46 2]
```

Ең алдымен ашық текст құпия кілтпен XOR-ланады, себебі addRoundKey әр турдың бірінші және соңғы кезеңі. Деректер блогының әрбір байты сәйкес дөңгелек кілт байтымен XOR-ға сәйкестендірілген. Бұл әрбір раундқа бірегейлік қосады және деректер блогына құпия кілтті енгізеді.

$69(\text{ашық текст алғашқы элементі}) \wedge (\text{xor}) 84(\text{құпия кілттің бірінші элементі}) = 17$ , осылайша қазіргі күйі:

```
[17 26 5 90]
[31 0 12 87]
[78 63 12 108]
[18 79 104 119]
```

Раунд механизмі (мысал ретінде 1-ші раунд қаралады):

1) SubBytes - бұл операция деректер блогының әрбір байтын S-Box ішіндегі сәйкес байтпен ауыстырады. S-Box – шифрға сызықтық еместікті енгізетін ауыстыру кестесі. Ол шифрланған деректерді болжауды және шабуылдарға төзімді етуге көмектеседі. Қазіргі күйі:

[130 162 107 190]  
[192 99 254 91]  
[47 117 254 80]  
[201 132 69 245]

2) ShiftRows - бұл деректердегі үлгілерді бұлдыратуға көмектеседі және шифрланған деректерді кездейсоқ етеді. Бірінші қатар өзгермейді, екінші қатар бір орын солға жылжижа, үшінші қатар екі орын, төртінші қатар үш орыннан солға қарай жылжиды.

[192 99 254 91] бір қадам солға жылжығаннан соң: [99 254 91 192]

ShiftRows операциясынан кейінгі деректер күйі:

[130 162 107 190]  
[99 254 91 192]  
[254 80 47 117]  
[245 201 132 69]]

3) MixColumns - бұл операцияда күй матрицасының әрбір бағаны бекітілген MixColumns матрицасымен көбейтіледі. Бұл деректер биттерінің диффузиясына мүмкіндік беретін және шифрға шабуылдардың күрделілігін арттыратын сызықтық түрлендіру. Деректер күйі:

[130 162 107 190]  
[99 254 91 192]  
[254 80 47 117]  
[245 201 132 69 ]

MixColumns кестесі:

{02, 03, 01, 01}  
{01, 02, 03, 01}  
{01, 01, 02, 03}  
{03, 01, 01, 02}

Бірінші элементті анықтау үшін деректер матрицасының бірінші бағанын mixcolumns кестесінің бірінші қатарына көбейту қажет.

[130 99 254 245] x {02, 03, 01, 01} – әрбір байттарды көпмүшелікке келтіріліп, сәйкес көпмүшелікпен Галуа өрісінде көбейтіліп, шыққан мәндерді XOR операциясымен біріктіріледі.

GF(256): 130 \* 02:

- Екілік жүйеде 130 саны 10000010;
- Көпмүшелік күй:

$$x^7 \cdot 1 + x^6 \cdot 0 + x^5 \cdot 0 + x^4 \cdot 0 + x^3 \cdot 0 + x^2 \cdot 0 + x^1 \cdot 1 + x^0 \cdot 0;$$
$$x^7 + x;$$

- 02 саны екілік жүйеде: 00000010;
- Көпмүшелік күй:

$$x^7 \cdot 0 + x^6 \cdot 0 + x^5 \cdot 0 + x^4 \cdot 0 + x^3 \cdot 0 + x^2 \cdot 0 + x^1 \cdot 1 + x^0 \cdot 0 = x ;$$

- Көпмүшелерді көбейту:  $x(x^7 + x) = x^8 + x^2$  ;
- Шыққан көпмүшені қайта екілік жүйеге айналдыру қажет, алайда мән 8 биттен асып кетті, сол себептен келтірілмейтін көпмүшеге хог орындалады. GF(2<sup>8</sup>) өрісі үшін  $x^8 + x^4 + x^3 + x + 1$  келтірілмейтін көпмүшесі қолданылады;

- $(x^8 + x^2) \wedge (x^8 + x^4 + x^3 + x + 1) = (x^4 + x^3 + x^2 + x + 1)$ ;
- Екілік күйде: 00011111;
- Ондық жүйеде: 31.

GF(256):

$$130 * 02 = 31$$

$$3 * 99 = 165$$

$$1 * 254 = 254$$

$$1 * 245 = 245$$

$31 \wedge 165 \wedge 254 \wedge 245 = 177$ , яғни деректер күйінің бірінші элементі 177.

- $1 * 130 \wedge 2 * 99 \wedge 3 * 254 \wedge 1 * 245 = 168$ ;
- $1 * 130 \wedge 1 * 99 \wedge 2 * 254 \wedge 3 * 245 = 2$ ;
- $3 * 130 \wedge 1 * 99 \wedge 1 * 254 \wedge 2 * 245 = 241$ , деректер күйінің бірінші

бағаны анықталды: (177, 168, 2, 241).

MixColumns операциясынан кейінгі деректер блогының күйі:

[177 223 144 12]

[168 124 40 255]

[2 188 249 91]

[241 218 218 230]

4) AddRoundKey – деректер блогы өз раунд кілтімен хог операциясын орындайды, бірінші раунд кілті - [[84 104 97 116] [115 32 109 121] [32 75 117 110] [103 32 70 117]]. Деректер күйі:

[97 0 171 229]

[11 131 126 111]

[129 8 218 165]

[21 78 191 109]

Бұл бірінші раунд нәтижесі. Осылай 10 раунд қайталағандағы деректер блогы қарапайым 16-лық жүйеге келтіріледі. Шифр мәні:

d1146ca84f5e9371c2c2863c3f0a303d

Дешифр үшін раундтар, кестелер мен операциялар кері бағытта қолданылады. Бұл 10 раундтағы AES операциялары машина үшін өте жылдам орындалады. Қазіргі таңдағы веб-сайттар үшін ең берік симметриялық криптоалгоритм болып саналады[11].

Веб-қауіпсіздікте деректер тұтастығын қамтамасыз ету криптографиялық хэш алгоритмдердің қызметі. Таңдалған криптографиялық хэш функциясы және хабардың артына қосылған ортақ кілт арқылы жасалады. Мәліметтерді жібермес бұрын жіберуші ол үшін MAC (Message Authentication Code) есептейді, ал қабылдаушы өңдеу алдында қабылданған хабарлама үшін MAC есептейді және

оны сол қабылданған хабарламаның MAC-мен салыстырады. Яғни жіберу кезінде хабардың өзгертілмегенін тексеру үшін жасалған. SHA-1 криптографиялық хэш алгоритмін зерттейміз. Бұл хэш-функция коллизияға байланысты әлқашан нағыз қолданыстан шектетілген. Қазіргі интернетте беріктігі жоғары SHA-256 хэш-функциясын пайдалануға кеңес беріледі[12].

Мысал:

Ұлттық стандарттар институтымен ұсынылған бес 32 биттік айнымалылар қажет (4-кесте).

Кесте 4

32-биттік тұрақты айнымалылар

16-лық жүйеде	10-дық жүйеде
A = 0x67452301	$h_0 = 1732584193$
B = 0xEFCDAB89	$h_1 = 4023233417$
C = 0x98BADCFE	$h_2 = 2562383102$
D = 0x10325476	$h_3 = 271733878$
E = 0xC3D2E1F0	$h_4 = 3382297650$

Негізгі циклдер үшін төрт тұрақты мән, сызықты емес операциялар қажет (5-кесте).

Кесте 5

SHA-1 тұрақтылары

$F_t(b, c, d)$	Константтар	Диапазон
$(b \text{ and } c) \text{ or } (\text{not}(b) \text{ and } d)$	K = 0x5A827999	$0 \leq t \leq 19$
$b \text{ xor } c \text{ xor } d$	K = 0x6ED9EBA1	$20 \leq t \leq 39$
$(b \text{ and } c) \text{ or } (b \text{ and } d) \text{ or } (c \text{ and } d)$	K = 0x8F1BBCDC	$40 \leq t \leq 59$
$b \text{ xor } c \text{ xor } d$	K = 0xCA62C1D6	$60 \leq t \leq 79$

"Hi, Elnur!" хабарламасын хэш мәнін анықтау байт түріне келтірілуі қажет: [72 105 44 32 69 108 110 117 114 33]. Егер кілттік сөз пайдалану қажет жағдайда, хабарға кілттік сөзді жалғау ретінде қосуға болады. Хабарға 0x80 байты қосылады. Бұл деректердің аяқталуын және толтырудың басын көрсету үшін жасалады: [72 105 44 32 69 108 110 117 114 33 128]. Хабарлама ұзындығы 64 байтка жетпеді, толықтыру үшін нольдік байттарды қосамыз. Соңғы 8 байт ол





Шыққан соңғы мәндерді ондық жүйеден он алтылық жүйеге келтіріп, біріктіру қажет. Сонда 40 байттан тұратын хэш код пайда болады: 1c2628f6a35911d7131b69ce05a0e68bb7de20e1. Кіріс ақпарат қандай өлшемде болмасын, шығыс хэш әрқашан 40 байттан тұрады және ақпараттағы аз ғана өзгеріс хэш кодтың толықтай жаңаруына алып келеді[13].

Кіріс сөз: Hi, Elnur!

Хэш мән: 1c2628f6a35911d7131b69ce05a0e68bb7de20e1

Өзгертілген кіріс сөз: Hi, elnur!

Хэш мән: f15b5ce7af61da8ab6f82f32d22127618c7659e9

### 1.3 Криптоалгоритмді бұзу әдістері

Шифрлаудың жаңа әдістері дамыған сайын математиканың маңызы арта түсті. Мысалы, жиілікті талдауда криптоаналитиктің лингвистикадан да, статистикадан да білімі болуы керек. Математиканың арқасында криптография осындай дамуға жетті, сондықтан бұзу үшін қажетті қарапайым математикалық операциялардың саны тым үлкен мәндерге жете бастады.

Қазіргі криптография бұзу үшін ескі замандағыдай қағаз бен қаламды алып есептеу нәтиже бермейді. Қазіргі шифрларды бұзуда криптоанализдың теориялық операцияларымен бұзу мүмкін емес көрінеді. Бірақ компьютердің дамуы, криптоаналитикаға жаңа дем берді. Компьютерлер өте көп операцияларды циклды түрде жылдам орындай алады, ал криптоаналитиктің мақсаты сол операциялар санын барынша азайту[14].

Жаңа криптографиялық алгоритмдердің пайда болуы оларды бұзу әдістерінің дамуына әкеледі. Әрбір жаңа криптоталдау әдісінің пайда болуы, шифрлардың қауіпсіздікті бағалауды қайта қарауға, жаңартуға, күделендіруге әкеледі. Осылайша, криптография мен криптоанализдің даму кезеңдері бір-бірімен тығыз байланысты.

Криптоаналитикалық шабуылдардың төрт негізгі түрі белгілі. Әрбір жағдайда криптоаналитик қолданылатын шифрлау алгоритмін біледі деп болжанады.

- Тек шифртекст негізінде шабуыл (Ciphertext Only Attack) - криптографиядағы ең қиын шабуылдардың бірі, себебі шабуылдаушыға сәйкес ашық мәтін немесе кілт туралы ешбір ақпаратсыз тек шифрлық мәтін қол жетімді.

- Белгілі ашық мәтіндік шабуыл (Known Plaintext Attack) - шабуылдың бұл түрінде шабуылдаушы ашық мәтіннің бірнеше жұптарына және оларға сәйкес шифрлық мәтінге қол жеткізе алады. Шабуыл жасаушы бұл жұптарды шифрлау алгоритміндегі немесе кілттегі осалдықтарды талдау және анықтау үшін пайдалана алады. Мысалы, егер шабуылдаушы «hi» сөзі «abcde» шифрланғанын

білсе, ол бұл ақпаратты кілтті немесе басқа шифрлау мәліметтерін кері есептеу үшін пайдалана алады.

- Таңдалған ашық мәтіндік шабуыл (Chosen Plaintext Attack) - бұл жағдайда шабуылдаушы ашық мәтіндерді таңдап, сәйкес шифрлық мәтіндерді ала алады. Мұндай шабуылдың мақсаты жүйенің әртүрлі кірістерге жауап беруін талдау және осалдықтарды анықтау болып табылады. Мысалы, шабуылдаушы бірнеше арнайы таңдалған ашық мәтіндерді шифрлай алады және шифрлау кілті немесе алгоритмі туралы ақпаратты алу үшін алынған шифрлық мәтіндерді талдай алады.

- Бейімделетін таңдалған ашық мәтіндік шабуыл (Adaptive Chosen Plaintext Attack) - шабуылдың бұл түрі таңдалған ашық мәтіндік шабуылға ұқсайды, тек шабуылдаушы өзінің сұрауларын алдыңғы жауаптар негізінде бейімдей алады. Бұл шабуылды икемді және күшті етеді, өйткені шабуылдаушы жүйеден алынған ақпаратты барынша арттыру үшін өз сұрауларын бейімдей алады. Мысалы, шабуылдаушы жүйенің алдыңғы таңдалған мәтінге жауабы негізінде келесі таңдалған ашық мәтінді бейімдей алады, бұл оған жүйедегі осалдықтарды тиімдірек талдауға мүмкіндік береді.

Криптоталдаудың әмбебап әдістері:

1) Дөрекі күш әдісі (Brute Force)

Жоғары өнімді есептеуіш технологиялардың пайда болуымен криптоаналитиктер негізгі күшті қолдану арқылы шифрларды бұзу мүмкіндігіне ие болды. Шифрды шешу үшін шифрлау кілтінің барлық ықтимал нұсқаларын кезекті түрде сынау арқылы жүзеге асырылады. Бұл әдіс шабуылдаушы шифр кілтін білмеген жағдайда қолданылады.

Мысалы, криптоаналитик немесе шабуылшы шифр түрін біледі. Бірнеше ашық хабарлама пен сәйкес шифрланған хабарлама бар. Барлық мүмкін кілт нұсқаларын даярлап алып, толық бір операция үшін бір кілтті қолданады. Әрекет шифртекст түсінікті немесе ашық хабарламаға сәйкес нәтиже бергенше қайталанады.

Қазіргі заманғы криптографиялық алгоритмдер әдетте дөрекі күштердің шабуылдарынан қорғалғанымен, әлсіз кілттерге немесе басқа осалдықтарға байланысты осы әдісті қолдану арқылы шифрлар сәтті бұзылған жағдайлар болды. Міне, кейбір мысалдар:

- DES қазіргі заманғы алгоритмдер әзірленгенге дейін шифрлау стандарты болды. 1999 жылы Electronic Frontier Foundation (EFF) командасы 56-биттік DES кілтіне дөрекі күшпен шабуыл жасау үшін бөлінген компьютерлердің желісін пайдаланды. DES Cracker деп аталатын бұл шабуыл шифрланған хабарламаларды бұза алды, бұл DES әлсіздігін көрсетті.

- MD5 алгоритмі де соқтығыстарды табу үшін дөрекі күш қолдану арқылы бұзылды. Бұл оны криптографиялық мақсаттар үшін жарамсыз етті, өйткені соқтығыстар деректердің тұтастығын бұзатын жалған хэштерді жасауға мүмкіндік береді.

2) Жиілікті талдау

Криптоталдауда жиілікті талдау (әріптерді санау деп те аталады) шифрленген мәтіндегі әріптердің немесе әріптер тобының жиілігін зерттеу болып табылады. Бұл әдіс классикалық шифрларды бұзуға көмекші құрал ретінде қолданылады. Жиілік талдауы жазба тілдің кез келген берілген кеңістігінде белгілі бір әріптер мен әріптер тіркесімі әртүрлі жиілікте болатынына негізделген. Сонымен қатар, сол тілдің барлық дерлік үлгілері үшін шамамен бірдей әріптердің тән таралу ықтималдылығы бар. Мысалы, ағылшын тілінің бөлімінде Е, Т, А және О жиі кездеседі, ал Z, Q, X және J сирек кездеседі. Сол сияқты TH, ER, ON және AN әріптердің ең көп тараған жұптары, ал SS, EE, TT және FF - ең көп қайталанатындар[15].

Кейбір шифрларда табиғи тілдің ашық мәтінінің мұндай қасиеттері шифрленген мәтінде сақталады және бұл үлгілерді “тек шифртекст негізінде шабуылда” да қолдануға болады.

Мысал:

Шабуылшы қарапайым ауыстыру шифрымен шифрланған мәтінді ұстап алды:

*“LIVITCSWPIYVEWHEVSRIQMXLEYVEOIEWHRXEXIPFEMVEWHKVSTYLXZIX  
LIKIIXPJVSZEYPERRGERIMWQLMGLMXQERIWGPSRIHMXQEREKIETXMJTP  
RGEVEKEITREWHEXXLEXXMZITWAWSQWXSWEEXTVEPMRXRSJGSTVRIEYVIE  
XCVMUIMWERGMIWXMJMGCSMWXSJOMIQXLIQIVIXQSVSTWHKPEGARCS  
XRWIEVSWIIBXVIZMXFSJXLIKEGAEWHEPSWYSWIWIEVXLISXLIVXLIRGEPIR  
QIVIBGIIHMWYPFLEVHEWHYPSRRFQMXLEPPXLI ECCIEVEWGISJKTVWMRL  
IHYS PHXLIQIMYLSXJXLIMWRIGXQEROIVFVIZEVAEKPIEWHXEAMWYEPPL  
MWYRMWXSGSWRMHIVEXMSWMGSTPHLEVHPFKPEZINTCMXIVJSVLMRSC  
MWMSWVIRCIGXMWYMX”*

Мәтінде үштік жұп “XLI” екені анық, ал жиілік бойынша ағылшын тілінде “the” жұбы жиі кездеседі. “E” әріпінің жиілігі “a” әріпіне жақын, осы 4 әріпті ауыстырып қарайды:

*“heVeTCSWP eYVaWHaVSReQMthaYVaOeaWHRtatePFaMVaWHKVSTYhtZetheKeet  
PeJVSZaYPaRRGaReMWQhMGhMtQaReWGPSReHMTQaRaKeaTtMJTPRGaVaKae  
TRaWHatthattMZeTWAWSQWtSWatTVaPMRtRSJGSTVReaYVeatCVMUeMWaRGMe  
WtMJMGCSMWtSJOMeQtheVeQeVetQSVSTWHKPaGARCS tRWeaVSWeeBtVeZMtF  
SJtheKaGAaWHaPSWYSWeWeaVtheStheVtheRGaPeRQeVeeBGeeHMWYPFhaVHa  
WHYPSRRFQMthaPPtheaCCeaVaWGeSJKTVMWRheHYSPhtheQeMYhtSJtheMWR  
eGtQaROeVFVeZaVAaKPeaWHtaAMWYaPPthMWYRMWtSGSWRMHeVatMSWMG  
STPHhaVHPFKPaZeNTCMteVJSVhMRSCMWMSWVeRCeGtMWYMt”*

Осы болжамды пайдаланып “Rtate” – “state”, “heVe” – “here” сөздерді байқауға болады, әр ауыстырудан кейін мүмкін сөздері қалпына келтіру арқылы шифрдың ашық мәтінін көруге болады:

*“Hereupon Legrand arose, with a grave and stately air, and brought me the beetle from a glass case in which it was enclosed. It was a beautiful scarabaeus, and, at that time, unknown to naturalists—of course a great prize in a scientific point of view. There were two round black spots near one extremity of the back, and a long one near the other. The scales were exceedingly hard and glossy, with all the appearance of burnished gold. The weight of the insect was very remarkable, and, taking all things into consideration, I could hardly blame Jupiter for his opinion respecting it.”*

3) Дифференциалды криптоталдау әртүрлі шифрлау раундтарындағы шифрланған мәндер арасындағы айырмашылықтарды түрлендіруді зерттеуге негізделген. Бұл статистикалық шабуыл. 90-жылдардың басына дейін әзірленген DES, FEAL және кейбір басқа шифрларды бұзу үшін қолайлы. Заманауи шифрлардың (AES, Camellia және т.б.) раундтарының саны дифференциалды криптоталдауды қоса алғанда, қауіпсіздікті ескере отырып есептелді[16].

4) Сызықтық криптоталдау DES (Деректерді шифрлау стандарты) немесе AES (Жетілдірілген шифрлау стандарты) сияқты блоктық шифрларды бұзу үшін қолданылатын криптоталдау әдісі болып табылады. Ол 1980 жылдардың соңында әзірленді және шифрларға шабуыл жасау әдістерінің бірі болып табылады.

Сызықтық криптоталдаудың негізгі идеясы шифрлаудың әрбір айналымының кірісі мен шығысы арасында сызықтық жуықтауларды жасау болып табылады. Бұл сызықтық жуықтаулар белгілі бір шифрлау кілтінің болуы ықтималдығын анықтау үшін қолданылады.

Сызықтық криптоталдау процесінде шабуылдаушы шифрдың кіріс және шығыс разрядтары арасындағы байланыстардың статистикасын зерттейді. Осы статистиканы пайдалана отырып, шабуылдаушы белгілі бір ықтималдықпен кіріс биттерінің негізінде шығыс биттерінің мәндерін болжай алатын сызықтық жуықтауларды жасайды.

## 2 Тәжірибе бөлім

### 2.1 Криптографиялық алгоритмдердің веб-сайтқа енгізілуі

Құпия сөзді хэштеу веб-сайт қауіпсіздігі үшін аса маңызды. Ол үшін криптографиялық хэштеу алгоритмдері қолданылады. Пайдаланушы құпия сөздерін сақтаудың орнына сервер олардың хэш мәндерін сақтайды. Хэш функциясы - бұл құпия сөзді қабылдайтын және оны хэш деп аталатын тұрақты ұзындықтағы жолға түрлендіретін алгоритм. Хэш мәні хэш функциясының нәтижесі болып табылады. Құпия сөз хэш мәнін бастапқы құпия сөзге қайта түрлендіру мүмкін емес. Яғни, шабуылдаушы пароль хэштерінің дерекқорына қол жеткізсе де, ол құпия сөздерді қайта қалпына келтіре алмайды[17].

Іске асыру үшін 2 функция қажет. CreateUser функциясында (5-сурет) пайдаланушыдан серверге келіп түскен тіркелу ақпараттары өңделінеді. Бұл функция пайдаланушы құпия сөзінің хэш мәнін анықтайды және оны пайдаланушы құпия сөзіне теңестіреді. Яғни құпия сөз деректер қорына жетпей хэш мәнін қабылдап алады.

```
func (a *AuthService) CreateUser(user models.User) error {  
    if err := validUser(user); err != nil { ...  
    }  
  
    uniq, err := a.storage.CheckUserByNameEmail(user.Email, user.Username)  
    if err != nil { ...  
    }  
    if uniq { ...  
    }  
  
    user.Password, err := generateHashPassword(user.Password)  
    if err != nil {  
        return err  
    }  
  
    return a.storage.Auth.CreateUser(user)  
}
```

Сурет 5 CreateUser функциясы.

Келесі функция CheckUser (6-сурет) авторизация үшін қажет. Пайдаланушы жүйеге кіру үшін құпия сөзін енгізеді. Сервер келіп түскен ақпаратты өңдейді, пайдаланушының логины арқылы деректер қорынан пайланушының хэш құпия сөзін (7-сурет) алып шығады. Пайдаланушыдан келіп түскен құпия сөздің хэш мәнін қайта анықтап деректер қорындағы хэш мәнмен салыстырады, сәйкес келген жағдайда пайдаланушы жүйеге кіреді.

```

54 func (a *AuthService) CheckUser(user models.User) (string, time.Time, error) {
55     password, err := a.storage.GetPasswordByUsername(user.Username)
56 >     if err != nil {...
57     }
58     if err := compareHashAndPassword(password, user.Password); err != nil {
59         return "", time.Time{}, err
60     }
61     token := uuid.NewGen()
62     d, err := token.NewV4()
63 >     if err != nil {...
64     }
65     expired := time.Now().Add(time.Hour * 12)
66 >     if err := a.storage.SaveToken(d.String(), expired, user.Username); err != nil {...
67     }
68     return d.String(), expired, nil
69 }

```

Сурет 6 CheckUser функциясы.

Пайдаланушының құпия сөзін пайдаланушының өзінен басқа ешкім білмейді. Дерекқор тек құпия сөздің хэш мәнін сақтайды, ол арқылы құпия сөздің алғашқы мәнін анықтау мүмкін емес. Сол себепті құпия сөзді ұмытқан жағдайда тіпті сервер құпия сөзді пайдаланушыға анықтап бере алмайды.

username	password	sess
Amanda	\$2a\$10\$ZxJJKWVhZeP8Y46j6fbAJe8Q4.PJjGBrZeu8L...	
Elnursccc	\$2a\$10\$W/3pR.YP/yrzmhivFfZ.6epdxMvU0VEevUP...	
David777	\$2a\$10\$xl/JXr1eLHvXM1MdcDpZk.rsAwD6ZGCujv8...	

Сурет 7 Деректер қоры.

Кез келген қауіпсіз веб-сайтқа кіріп сертификат бөлімінен қандай шифрлау алгоритмдер жиыны таңдалғанын білуге болады. TLS 1.2 протоколы үшін ең көп кездесетін жиын -“TLS\_ECDHE\_RSA\_WITH\_AES\_128\_GCM\_SHA256”. Бұл алгоритм сеанс барысында құпиялық пен тұтастықты қамтасыз етеді.

ECDHE (эллиптикалық қисық Диффи-Хеллман эфемерлі) - бұл сервер мен клиентке ұсталмайтын ортақ құпияға негізделген құпия кілт туралы келісуге мүмкіндік беретін кілт алмасу әдісі.

RSA - сервер деректерді клиентке жібермес бұрын қол қою және түпнұсқалығын растау үшін RSA жеке кілтін пайдаланады. Ал клиент шифрді ашық кілт арқылы шешіп сервердің жеке кілті бар екеніне, яғни шынайылығына көз жеткізеді.

AES\_128 (Advanced Encryption Standard) - бұл деректердің құпиялығын қорғау үшін қолданылатын симметриялық шифрлау алгоритмі. Ол деректерді шифрлау және шифрын шешу үшін бірдей кілтті пайдаланады.

GCM (Galois/Counter Mode): Бұл аутентификация мен жалғандыққа қарсы қорғауды, сондай-ақ құпиялылықты қамтамасыз ететін AES жұмыс режимі. GCM қосымша деректер аутентификациясын қамтамасыз етеді, яғни алушы деректерді транзит кезінде шабуылдаушы бұзбағанына сенімді болуы үшін.

SHA256: Бұл деректер тұтастығын қамтамасыз ету үшін аутентификация кодын есептеу үшін пайдаланылатын хэштеу алгоритмі. TLS контекстінде SHA256 деректер хэштерін жасау үшін пайдаланылады, олар кейін сервер мен клиент арасында жіберілген хабарлардың түпнұсқалығын растау және тұтастығын қамтамасыз ету үшін пайдаланылады.

Сервер қолдайтын шифрлар (5-сурет) жиындарының көп болу себебі, таңдау мүмкіндігінің болуы үшін. Клиент пен сервер қолдай алатын шифрлар жиыны табылуы қажет. Сервер өз максималды және минималды TLS протокол бөлімін көрсете алады, эллиптикалық қисықтарды ұсынады және міндетті түрде сенімді лицензиясы бар сертификат ұсынады.

```
15 func (s *Server) Run(port string, handler http.Handler) error {
16     cert, err := tls.LoadX509KeyPair("secure/server.crt", "secure/server.key")
17     if err != nil {
18         log.Fatal("Ошибка загрузки сертификата: ", err)
19     }
20     tlsConfig := &tls.Config{
21         MinVersion:         tls.VersionTLS13,
22         MaxVersion:         tls.VersionTLS13,
23         PreferServerCipherSuites: true,
24         CipherSuites: []uint16{
25             tls.TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384,
26             tls.TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384,
27             tls.TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305,
28             tls.TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305,
29             tls.TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256,
30             tls.TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,
31         },
32         CurvePreferences: []tls.CurveID{tls.X25519, tls.CurveP256},
33         Certificates:     []tls.Certificate{cert},
34     }
```

Сурет 8 Golang тіліндегі сервер tls-конфигурациясы.

Клиенттің қолдайтын шифрлау алгоритмдері (6-сурет) және басқа да параметрлері көп жағдайда қолданылып жатқан браузерге байланысты болады. TLS қолалысу кезеңдері протокол версиясына байланысты өзгеруі мүмкін. Бірақ кез келген TLS қолалысу ClientHello пакетінен басталады. Бұл пакет ашық жүзде серверге жіберіледі.

```

▼ Cipher Suites (17 suites)
  Cipher Suite: TLS_AES_128_GCM_SHA256 (0x1301)
  Cipher Suite: TLS_CHACHA20_POLY1305_SHA256 (0x1303)
  Cipher Suite: TLS_AES_256_GCM_SHA384 (0x1302)
  Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (0xc02b)
  Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)
  Cipher Suite: TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256 (0xcca9)
  Cipher Suite: TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 (0xcca8)
  Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 (0xc02c)
  Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030)
  Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA (0xc00a)
  Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA (0xc009)
  Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xc013)
  Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014)
  Cipher Suite: TLS_RSA_WITH_AES_128_GCM_SHA256 (0x009c)
  Cipher Suite: TLS_RSA_WITH_AES_256_GCM_SHA384 (0x009d)
  Cipher Suite: TLS_RSA_WITH_AES_128_CBC_SHA (0x002f)
  Cipher Suite: TLS_RSA_WITH_AES_256_CBC_SHA (0x0035)

```

**Сурет 9** Клиент қолдайтын шифрлау алгоритм жиындарының тізімі (Wireshark көмегімен түсірілген трафик).

Сервер жауап ретінде 3 пакет қайтарады, ServerHello (7-сурет), Certificate және ServerHelloDone. TLS версияларына байлынысты немесе арнайы әзіргенген конфигурацияларға байланысты өзгеруі мүмкін. TLS 1.3 бір ғана рет барып-қайту арқылы қолалмасуды аяқтай алады.

```

▼ Handshake Protocol: Server Hello
  Handshake Type: Server Hello (2)
  Length: 118
  Version: TLS 1.2 (0x0303)
  Random: 54d925c917fccdd905c3e22ad0b6bb94158dec01c7c8ac5cda2c6a77eeb8db52
  Session ID Length: 32
  Session ID: 819acde268f3f5022d021c5a328cec01dc70a15598256121fda1dafd3299398e
  Cipher Suite: TLS_AES_128_GCM_SHA256 (0x1301)
  Compression Method: null (0)
  Extensions Length: 46
  ▶ Extension: supported_versions (len=2)
  ▶ Extension: key_share (len=36)
    [JA3S Fullstring: 771,4865,43-51]
    [JA3S: f4febcb55ea12b31ae17cfb7e614afda8]

```

**Сурет 10** ServerHello.

Асимметриялық шифрлау алгоритмі арқылы ортақ құпия сеанстық кілт анықталады. Сеанстық кілт симметриялық шифрлау алгоритмінің кілті болып табылады, ол серверге де клиентке де берілуі міндетті, ал ортадағы адамға деректер (8-сурет) анықтау мүмкін еместей болуы қажет.

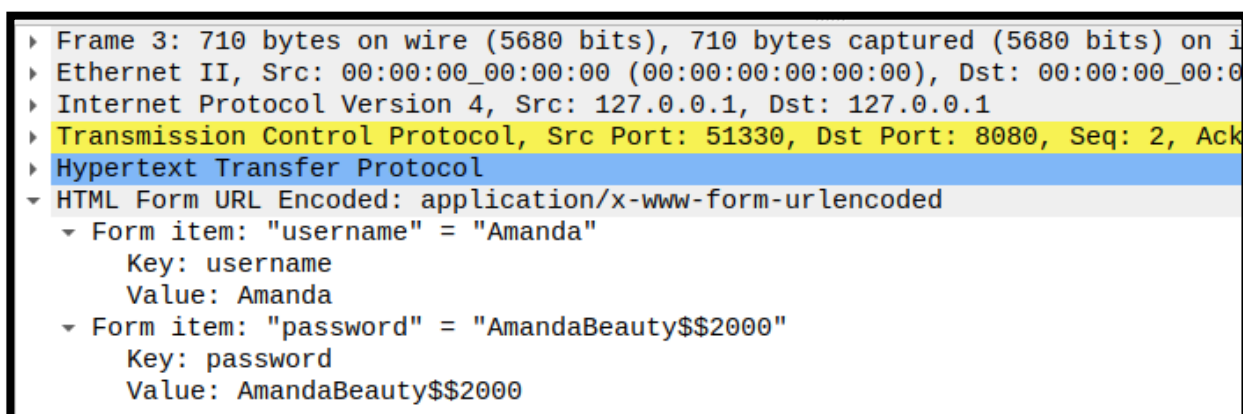




Сурет 11 Шифрланған дерек.

2009 жылдарда шамамен 90% дан астам веб-сайттар ақпаратты ашық түрде (9-сурет) алмасты. Жіберілетін деректер шифрланбады. Қазіргі таңда HTTP протоколымен жұмыс жасайтын веб-сайттарды табу өте қиын, браузерлер олардың қауіпті екендігін ескертеді.

Тексеру үшін веб-сайт әзірлеп, локалды хост 8080 портты веб-сервер тыңдау қажет немесе кез келген бос портты пайдалануға болады. Біздің жағдайда сайт үшін аутентификация және авторизация қарастырылған. Аманда есімді қыз сайттың тұрақты пайдаланушысы. Ол жүйеге кіру үшін әдеттегідей логині мен құпия сөзін толтырады. Ал ортадағы адам кез келген трафикты бақылау бағдарламасы арқылы клиенттің деректерін қарай алады. Шабуылшы бұл ақпаратты өз мақсаттарына жаратады.



Сурет 12 Шифрланбаған дерек.

### 2.1.1 SSL/TLS сертификатын орнату

Сертификатсыз қауіпсіз жалғану орындалмайды. Сертификат веб-сайт домендік атауы, ашық кілт, цифрлық қолтаңба, сертификаттау орталығы, т.б. ақпараттарды көрсетеді.

Домендік атау - веб-сайттың домендік атауын көрсететін негізгі өріс (мысалы, [www.wiki.com](http://www.wiki.com)). Бұл клиенттерге дұрыс веб-сайтқа қосылғанына көз жеткізуге мүмкіндік береді.

Ашық кілт - серверге тасымалданатын деректерді шифрлау үшін пайдаланылатын кілт. Ашық кілт сервердің аутентификациясы үшін де қолданылады.

Сертификаттау орталығы (CA) ақпараты - сертификатты берген сертификаттау орталығы туралы ақпарат. Ол CA атауын, сертификаттың берілген күнін, жарамдылық мерзімін және басқа мета ақпаратын қамтиды.

Цифрлық қолтаңба - сертификаттың түпнұсқалығын растайтын криптографиялық қолтаңба. Ол сертификаттың шығарылғаннан кейін өзгертілмегенін және оны уәкілетті CA бергенін қамтамасыз етеді.

Серверде сертификатпен бірге жеке кілт болады. Жеке кілтті сертификаттың ешкімге жіберілмейтін бір бөлігі деуге болады. Сертификат ашық түрде жіберілгендіктен ұрлануы мүмкін. Зиянкес сертификат ақпараттарын өзгеркен жағдайда, цифрлық қолтаңбамен сәйкестенуін тоқтатады. Себебі кез келген өзгеріс мүлде жаңа хэш-мән тудырады. Зиянкес цифрлық қолтаңбаны да ауыстырды делік. Ондай жағдайда клиент серверге сенбейді. Қолалмасу соңында сервер клиентке бар ақпараттың келісілген хэш алгоритм бойынша хэш-мәнін тауып, жеке кілтпен шифрлайды. Клиент шифрланған ақпаратты сервердің ашық кілтімен шеше алуы қажет, сонда серверде жеке кілттің бар екеніне көз жеткізеді, яғни сервер өзінің шын мәнінде сертификат иесі екендігін дәлелдейді. RSA әмбебап алгоритм, деректі ашық кілтпен шифрлап, жеке кілтпен шешуге және жеке кілтпен шифрлап, ашық кілт арқылы шешуге болады.

Пайдаланушының веб-сайтқа күдігі болған жағдайда, сертификаттау центрінің базасынан (<https://crt.sh/>) тексеруге мүмкіндігі бар. Веб-сайтыңыз үшін TLS сертификаттарын (SSL сертификаттары) алудың бірнеше жолы бар. Бірақ өз серверіміз үшін өз сертификатымызды жасай аламыз, тек ол браузерлер үшін қауіпсіз сайт ретінде қаралмайды.

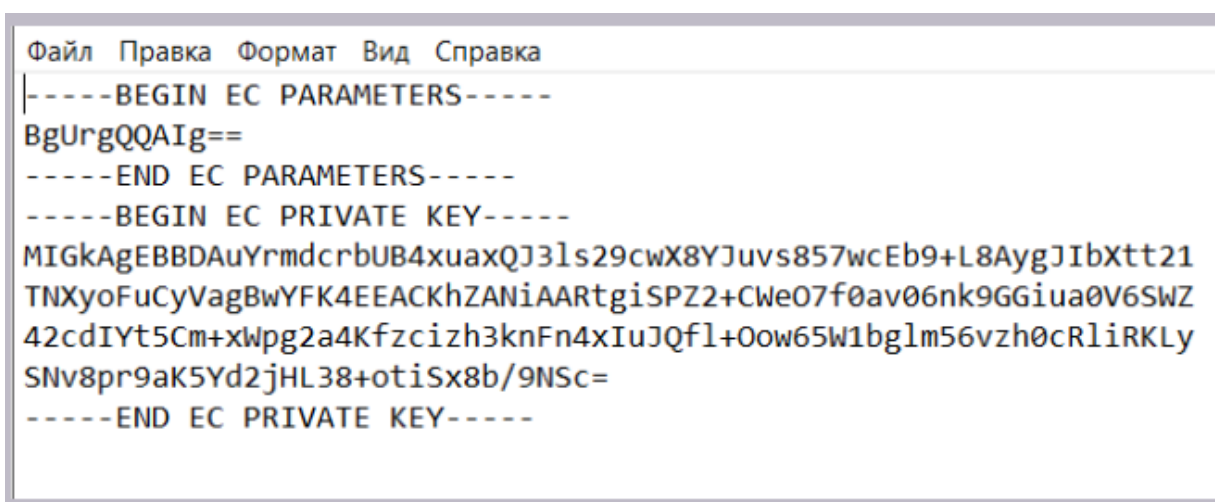
OpenSSL утилитасын қолдану арқылы өз ssl-сертификатымызды жасауға болады. Ең алдымен жеке кілт құрылады. Жеке кілт қандай криптоалгоритмге арналатынын алдын ала шешу қажет. Ең көп таралғаны RSA және ECDSA.

- “openssl genrsa -out server.key 2048” - бұл команда ұзындығы 2048 бит болатын RSA жеке кілтін жасайды. Осы пәрмен арқылы жасалған жеке кілт server.key файлында сақталады.

- “openssl ecparam -genkey -name secp384r1 -out server.key” - команда да жеке кілтті (10-сурет) жасауды қамтиды, бірақ secp384r1 параметрлері бар эллиптикалық қисық сызықта.

Эллиптикалық қисық RSA сияқты классикалық алгоритмдермен салыстырғанда ресурстарды тиімдірек пайдалануды ұсынады. Алайда, ECDSA пайдалану үшін клиенттерден (браузерлерден, операциялық жүйелерден және т.б.) қолдау қажет екенін ескеру қажет, себебі енгізілу күрделілігіне байланысты сертификат кейбір құрылғылар немесе бағдарламалар үшін қол жетімді болмауы мүмкін.

- “openssl req -new -x509 -sha256 -key server.key -out server.crt -days 365” - бұл пәрмен жеке кілт (11-сурет) негізінде өзіндік қол қойылған сертификатты (server.crt) жасайды . “-new” жалаушасы жаңа сертификат сұрауын (CSR) жасауды көрсетеді, ол өздігінен қол қойылған куәлікті (-x509) жасау үшін пайдаланылады. –“sha256” жалаушасы сертификатқа қол қою үшін SHA-256 хэштеу алгоритмін көрсетеді. “-days 365” жалаушасы сертификаттың жарамдылық мерзімін күндермен көрсетеді (365 күн).



```
Файл  Правка  Формат  Вид  Справка
|-----BEGIN EC PARAMETERS-----
BgUrgQQAig==
-----END EC PARAMETERS-----
-----BEGIN EC PRIVATE KEY-----
MIGkAgEBBDAuYrmdcrbUB4xuaxQJ3ls29cwX8YJuvs857wcEb9+L8AygJIbXtt21
TNXyoFuCyVagBwYFK4EEACKhZANiAARTgiSPZ2+CWe07f0av06nk9GGiua0V6SWZ
42cdIYt5Cm+xWpg2a4Kfzcizh3knFn4xIuJQfl+0ow65W1bglm56vzh0cRliRKLy
SNv8pr9aK5Yd2jHL38+otiSx8b/9NSc=
-----END EC PRIVATE KEY-----
```

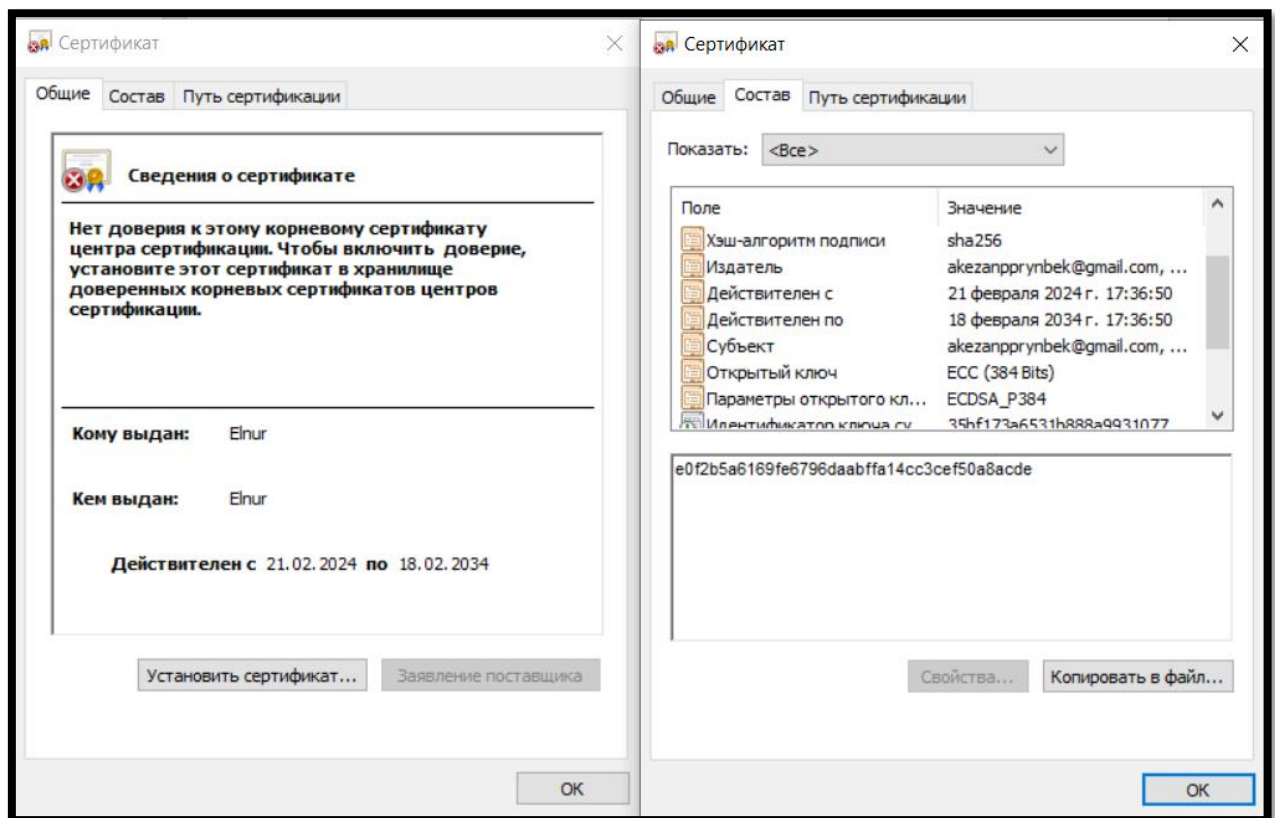
Сурет 13 Жеке кілт.

```
Файл  Правка  Формат  Вид  Справка
|-----BEGIN CERTIFICATE-----
MIICrDCCAjkGAWIBAgIUaHo2+eaSpVQWiM0G/0+9wi9TTA0wCgYIKoZIZj0EAwIw
gYwxCzAJBgNVBAYTAmt6MQ8wDQYDVQQIDAZBbG1hdHkxDzANBgNVBACMBkFsZW0x
eTENMA5GA1UECgwEYX1lBTETMBEGA1UECwwKYXN0YW5hIGh1YjEOMAwGA1UEAwwF
RWxudXIXJzAlBgkqhkiG9w0BCQEWGGFrZXphbnBwcnluYmVrQGdtYWlsLmNvbTAe
Fw0yNDAYMjExMjM2NTBaFw0zNDAYMTgxMjM2NTBaMIGMMQswCQYDVQQGEwJreJEP
MA0GA1UECAwGQWxtYXR5MQ8wDQYDVQQHDAZBbG1hdHkxDTALBgNVBAoMBGFsZW0x
EzARBGNVBASMCmFzdGFuYSBodWIXdJAMBGNVBAMMBUvsnVyMScwJQYJKoZIhvcN
AQkBFhhha2V6YW5wCHJ5bmJla0BnbWFPbC5jb20wdjAQBgcqhkiG9w0BQIBBgUrgQQA
IgniAARTgisPZ2+Cwe07f0av06nk9GGiua0V6SWZ42cdIYt5Cm+xwpg2a4Kfzciz
h3knFn4xIuJQfl+Oow65W1bglm56vzh0cRliRKLySNv8pr9aK5Yd2jHL38+otiSx
8b/9NSejUzBRMB0GA1UdDgQWBQ1vxc6ZTG4iKmTEHemd/j8d82aijAfbGNVHSME
GDAWgBQ1vxc6ZTG4iKmTEHemd/j8d82aijAPBgNVHRMBAf8EBTADAQH/MAoGCCqG
SM49BAMCA2gAMGUQMCCQqOzzwGlp5iOzTOLYr1fexs3ae+jKEq4Vk0z5os/h06Yi
j/3hNkME1jemdkhptHUCMBEjyl5+ya20D5yPgWT2mrdwyokzQjT2a3g62X9CX3i0
CO/rGU81WJYvM33mIC7ulg==
-----END CERTIFICATE-----
```

Сурет 14 Сертификат.

Сертификатты арнайы оқып қарау бағдарламалары арқылы оқуға болады. Яғни байттар түрінен адамға оқуға ыңғайлы етіп жасау. Мысалы, Windows операциялық жүйесінің кіріктірілген бағдарламасы сертификаттарды ашып оқуға, тексеруге, зерттеп қарауға мүмкіндік береді (12-сурет). SSL сертификаттары криптоалгоритмдермен қатысты ақпаратты қамтиды:

- Ашық кілт - SSL сертификатында клиент пен сервер арасында қауіпсіз байланыс орнату үшін пайдаланылатын сервердің ашық кілті бар. Бұл кілт әдетте аутентификация процесі кезінде клиент серверге жіберген деректерді шифрлау және қауіпсіз байланыс арнасын орнату үшін пайдаланылады.
- Шифрлау алгоритмі - сертификатта қолданылатын шифрлау алгоритмі туралы ақпарат бар. Бұл, мысалы, RSA шифрлау алгоритмі немесе эллиптикалық қисық шифрлау алгоритмі (ECDSA) болуы мүмкін.
- Цифрлық қолтаңба - SSL сертификатына сертификаттың түпнұсқалығы мен жарамдылығын растау үшін сертификаттау орталығының (CA) жабық кілтімен қол қойылады. ЭЦҚ арқылы клиент сертификатты жалған серверден емес, шын мәнінде сертификаттау органы бергенін тексере алады.



Сурет 15 Сертификат.

Бұл сертификаттарды (11,12-сурет) веб-серверге орнатамыз. Run әдісінің ішінде SSL сертификаты мен жеке кілт `tls.LoadX509KeyPair` функциясы арқылы жүктеледі. Бұл файлдар қауіпсіз қалтада орналасуы және тиісінше `server.crt` және `server.key` деп аталатын болуы керек.

Содан кейін қолдау көрсетілетін TLS нұсқалары, пайдаланылатын шифрлау алгоритмдері және ECC қисық параметрлері сияқты TLS параметрлерін қамтитын `tls.Config` нысаны жасалады. Сервер параметрлері портты, сұрау өңдеушісін, тақырыптың ең үлкен өлшемін және оқу және жазу күту уақыттарын көрсетеді. Осыдан кейін серверді іске қосу хабары көрсетіледі және сервер `ListenAndServeTLS` әдісі арқылы іске қосылады (13-сурет). Сервер <https://localhost:8080> адресін тыңдайды.



```

11 type Server struct {
12     | httpServer *http.Server
13 }
14
15 func (s *Server) Run(port string, handler http.Handler) error {
16     cert, err := tls.LoadX509KeyPair("secure/server.crt", "secure/server.key")
17 > if err != nil { ...
18 }
19
20     tlsConfig := &tls.Config{
21         | MinVersion:          tls.VersionTLS13,
22         | MaxVersion:          tls.VersionTLS13,
23         | PreferServerCipherSuites: true,
24 > | CipherSuites: []uint16{ ...
31     | },
32     | CurvePreferences: []tls.CurveID{tls.X25519, tls.CurveP256},
33     | Certificates:      []tls.Certificate{cert},
34     }
35     s.httpServer = &http.Server{
36         | Addr:          port,
37         | Handler:       handler,
38         | MaxHeaderBytes: 1 << 20,
39         | ReadTimeout:   10 * time.Second,
40         | WriteTimeout:  10 * time.Second,
41         | TLSConfig:     tlsConfig,
42     }
43     models.InfoLog.Printf("Server run on https://localhost%s", port)
44     return s.httpServer.ListenAndServeTLS("", "")
45 }

```

**Сурет 16** Golang тіліндегі TLS сервер қосу.

## 2.2 Криптографиялық алгоритмді құру

Криптографиялық алгоритмді құру – ақпаратты рұқсатсыз кіруден немесе өзгертуден қорғайтын математикалық шифрлау немесе хэштеу әдісін әзірлеу процесі. Криптографиялық алгоритмдерді құру математиканы, ақпараттық қауіпсіздікті және бағдарламалауды терең білуді, сондай-ақ ықтимал осалдықтарды болдырмау үшін егжей-тегжейге назар аударуды және сақтықты қажет ететін күрделі және көп уақытты қажет ететін процесс екенін ескеру маңызды[18].

### 2.2.1 Алгоритмді жобалау, мүмкін қауіптердің алдын алу

Криптоалгоритмнің симметриялық түрін әзірлейтін боламыз. Ең алдымен симметриялық криптоалгоритмді бұзатын криптоаналитика әдістерін білу қажет.

Симметриялық криптографиялық алгоритмдерге жасалған шабуылдар қандай осалдықтарды пайдаланатынына және шабуылдаушы қандай деректерді ала алатынына байланысты бірнеше түрге бөлуге болады. Міне, шабуылдардың негізгі түрлері:

- Man-in-the-Middle шабуылдары: шабуылдаушы өзін жіберуші мен алушының арасына енгізеді және жіберіліп жатқан хабарламаларды ұстайды, кейде өзгертеді. Бұл шабуылдаушыға деректерді көруге және өзгертуге, сондай-ақ байланыстарға жалған хабарламаларды енгізуге мүмкіндік береді.

- Белгілі ашық мәтіндік шабуылдар: шабуылдаушы сәйкес ашық мәтінмен бірге шифрланған хабарларға қол жеткізе алады. Ол бұл ақпаратты шифрлау алгоритмін талдау және кілтті қалпына келтіру немесе тіпті басқа хабарларға қол жеткізу үшін пайдаланады.

- Таңдалған шифрлық мәтіндік шабуылдар: шабуылдаушы шифрды шешу үшін шифрланған хабарламаларды жіберу және нәтижелерді алу мүмкіндігіне ие. Ол бұл деректерді шифрлау алгоритмін талдау және кілтті қалпына келтіру немесе тіпті басқа хабарларға қол жеткізу үшін пайдаланады.

- Жиілікті талдау шабуылдары: шабуылдаушы шифрланған хабарламалардағы таңбалар немесе биттердің жиілігін талдайды. Мысалы, егер шифрленген мәтін табиғи тіл болса, онда жиі кездесетін таңбалар шифр мәтінінің бастапқы таңбаларға сәйкес келетінін көрсете алады.

- Уақытша шабуылдар: шабуылдаушы шифрлау немесе шифрды шешу әрекеттерін орындауға кеткен уақытты талдайды. Орындау уақытындағы шағын айырмашылықтар шабуылдаушыға кілт немесе шифрлау алгоритмінің өзі туралы ақпарат бере алады.

- Сөздік шабуылдары: шабуылдаушы шифрлауды бұзу үшін алдын ала дайындалған шифрланған хабарламалардың сөздігін және олардың ашық мәтіндерін пайдаланады. Бұл әдіс әсіресе әлсіз құпия сөздерді немесе кілттерді пайдаланған кезде тиімді.

- Қосымша шабуылдар: шабуылдаушы хабарламалардағы ақпараттың артықтығын талдайды және бұл ақпаратты шифрлауды бұзу үшін пайдаланады. Мысалы, хабардың бір бөлігі әрқашан бірдей мағынаға немесе құрылымға ие болса, бұл шабуылға негіз болуы мүмкін.

Бұл шабуылдар, егер шифрлау алгоритмінде немесе оны жүзеге асыруда шифрланған деректерге қол жеткізу немесе шифрлау кілтін қалпына келтіру үшін шабуылдаушы пайдалана алатын осалдықтар болса, сәтті болуы мүмкін. Сондықтан әзірленетін криптоалгоритм аталған шабуыл түрлеріне төзімді болуы қажет.

Криптографиялық алгоритмдер әдетте шифрлау процесі кезінде орындалатын бірқатар операцияларды немесе қадамдарды қамтиды. Криптоалгоритм әзірлеу үшін қажетті математикалық операциялар мен қадамдарды белгілеп алу қажет. Маңызды қадамдар мен операциялар:

- Деректер блогын шифрлау;
- Ауыстыру (Substitution);

- Орын ауыстыру (Permutation);
- Итерациялар (Iterations);
- Кері операция.

*Деректер блогын шифрлау* қазіргі криптографиялық алгоритмдердің негізгі қадамы болып табылады, ол берілетін ақпараттың құпиялылығын қамтамасыз етеді. Бұл процесс кілтсіз шифрын шешуге болмайтын деректердің шифрланған блогын жасау үшін белгілі бір кілттің көмегімен деректер бөлігін (әдетте белгіленген өлшемді) түрлендіру арқылы орындалады.

Ең алдымен, шифрлау алдында деректер бекітілген блоктарға бөлінеді. Блок өлшемі арнайы криптографиялық алгоритммен анықталады. Мысалы, AES алгоритмінде блок өлшемі 128 бит.

Келесі қадам деректердің әрбір блогына шифрлау алгоритмін қолдану болып табылады. Бұл алгоритм жіберуші мен алушы арасында ортақ құпия параметр болып табылатын шифрлау кілтін пайдаланады. Деректер блогын шифрлау әртүрлі операцияларды қамтиды, мысалы, ауыстыру, ауыстыру, арифметикалық операциялар және кілтті қолдану.

Мәліметтер блогының тиімді шифрлануы қайтымды болуы керек, яғни шифрланған блокты шифрлау үшін қолданылатын сол кілттің көмегімен дешифрлеу мүмкіндігі болуы керек екенін ескеру маңызды. Бұл бастапқы ақпаратты алушының қауіпсіз түрде алуына кепілдік береді.

Блоктық шифрлау хабарды шифрлау және аутентификация режимдері сияқты көптеген криптографиялық примитивтер үшін негізгі құрылымдық блок болып табылады. Деректер блогының қауіпсіз шифрлануын қамтамасыз ету ақпараттың ашық немесе сенімсіз желілер арқылы тасымалдануы кезінде оның құпиялылығын қорғау үшін өте маңызды. Заманауи шифрлау жүйелерінде бұл қадамды қамтамасыз ету деректерді рұқсатсыз кіруден және шабуылдардан қорғау үшін өте маңызды.

*Ауыстыру (Substitution)* – деректерді шифрлау үшін қолданылатын криптографиядағы негізгі операциялардың бірі. Бұл операция белгілі бір ережеге немесе пернеге сәйкес енгізілген мәтіндегі таңбаларды немесе таңбалар тобын басқа таңбалармен немесе таңбалар тобымен ауыстыруды қамтиды.

Шифрлауға қолданылғанда ауыстыру операциясы ашық мәтіннен шифрлық мәтін жасау үшін қолданылады. Мұны алфавитті ауыстыру, биттерді ауыстыру және сызықты емес математикалық функцияларды пайдалану сияқты әртүрлі әдістер арқылы жасауға болады.

Ауыстырудың ең қарапайым мысалы Цезарь шифры болып табылады, мұнда кіріс мәтінінің әрбір әрпі алфавитте бекітілген ығысудағы әріппен ауыстырылады. Мысалы, 3 таңбаға жылжитқанда «А» әрпі «D», «В» әрпі «Е» және т.б.

Неғұрлым күрделі ауыстыру әдістері кіріс мәтініндегі әрбір таңба белгілі бір пернеге сәйкес басқа таңбамен ауыстырылатын ауыстыру кестелерін қолдануы мүмкін. Бұл ашық мәтін мен шифр мәтін таңбалары арасында екіұшты сәйкестікті тудырады, бұл криптоанализді қиындатады.



Ауыстыру операциясы жиі ауыстыру операциясымен біріктіріледі, мұнда символдардың реті де өзгереді. Бұл шифрлау алгоритмінің күрделілігін арттырады және оны криптоталдауға төзімді етеді.

Қазіргі криптографияда ауыстыру операциясы шифрлау қадамдарының негізгі құрамдастарының бірі болып табылатын AES (Advanced Encryption Standard) сияқты блоктық шифрларда кеңінен қолданылады. Ол сонымен қатар құпиялылық пен деректерді қорғауды қамтамасыз ету үшін ағынды шифрлау алгоритмдерінде және басқа криптографиялық примитивтерде қолданылады.

*Орын ауыстыру (Permutation)* - деректерді шифрлау үшін қолданылатын криптографиядағы маңызды операция. Бұл операция белгілі бір ережеге немесе кілтке сәйкес енгізілген мәтіннің таңбаларының немесе биттерінің ретін өзгертуді қамтиды.

Шифрлау контекстінде ауыстыру әдетте шифрлау алгоритміне қауіпсіздік пен күрделіліктің қосымша деңгейін қамтамасыз ету үшін Ауыстыру операциясымен бірге қолданылады. Ол деректер құрылымын өзгертеді, криптоталдау тапсырмасын қиындатады және шифрдың қауіпсіздігін арттырады.

Орын ауыстырудың ең қарапайым мысалы - ауыстыру шифры. Бұл әдісте енгізілген мәтіннің символдарының реті белгілі бір пернеге немесе ережеге сәйкес өзгертіледі. Мысалы, «231» пернесін пайдаланған кезде енгізілген мәтіннің таңбалары келесідей қайта реттеледі: бірінші символ үшіншіге, екінші таңба біріншіге, үшінші символ екіншіге айналады. Бұл түпнұсқадан өзгеше таңбалардың жаңа ретін жасайды, бұл шифрды талдауды қиындатады.

DES (Деректерді шифрлау стандарты) немесе AES (Жетілдірілген шифрлау стандарты) сияқты күрделі криптографиялық алгоритмдерде ауыстыру шифрлау айналымдарында қолданылады. Мысалы, AES алгоритмінде ауыстыру кезеңінен кейін әрбір деректер блогы белгілі бір ережелерге сәйкес блоктағы байттар араласатын ауыстыру кезеңінен өтеді.

Шифрлауда ауыстыруды қолдану деректер құрылымын өзгерту және криптоталдауды қиындату арқылы қауіпсіздіктің қосымша деңгейін қамтамасыз етеді. Ол сондай-ақ деректерді біркелкі таратуға көмектеседі, бұл осалдықтар мен шабуылдардың ықтималдығын азайтады.

Жалпы, ауыстыру құпиялылықты қамтамасыз етуге және деректерді рұқсатсыз кіруден қорғауға көмектесетін заманауи криптографиялық алгоритмдердің маңызды элементі болып табылады. Ол криптографияның әртүрлі салаларында кең ауқымды қосымшаларға ие және көптеген шифрлау әдістерінің құрамдас бөлігі болып табылады.

*Кілттерді араластыру* қазіргі криптографиялық алгоритмдердегі деректердің құпиялылығын және рұқсатсыз кіруден қорғауды қамтамасыз ететін маңызды операция болып табылады. Бұл операция шифрлау кілтін шифрлау процесінің алдында немесе оның барысында деректермен біріктіруді қамтиды, бұл шифрды құпия кілтке тәуелді етеді.

Криптографияда кілт — деректерді шифрлау және шифрын ашу үшін пайдаланылатын құпия ақпарат. Кілттік қолданба әдетте шифрлаудың нақты алгоритміне байланысты биттік XOR, арифметикалық операциялар немесе

логикалық операциялар сияқты әртүрлі операцияларды қолдану арқылы орындалады.

Ең көп тараған пернелеу операцияларының бірі биттік XOR. Бұл операцияда әрбір деректер биті сәйкес кілт битімен біріктіріледі. Егер биттер сәйкес келсе, нәтиже 0 болады, әйтпесе ол 1 болады. Бұл деректер биттерін кілт биттерімен «араластырып», шифрды кілтке тәуелді етіп, деректердің құпиялылығын қамтамасыз етуге мүмкіндік береді.

Оған қоса, кілтті қосу, алу, көбейту немесе бөлу сияқты басқа математикалық операцияларды орындау үшін пайдалануға болады. Мысалы, кілттің көмегімен деректер блогын шифрлау кезінде блоктың әрбір байты кілттің сәйкес байтына қосылуы немесе көбейтілуі мүмкін. Бұл сонымен қатар шифрдың кілтке тәуелді екендігін және деректер қауіпсіздігін қамтамасыз етеді.

Кілтті қолдану деректерді шифрлау және дешифрлеу процесіндегі маңызды қадам болып табылады. Егер кілт шабуылдаушыға белгісіз болса, ол кілтті білмей шифрланған деректердің шифрын шеше алмайды. Дегенмен, шифрдың қауіпсіздігі толығымен кілттің қауіпсіздігіне байланысты және егер ол ағып кетсе немесе бұзылса, деректердің құпиялылығы бұзылуы мүмкін.

*Итерациялар* заманауи криптографиялық алгоритмдерде маңызды рөл атқарады, қауіпсіздіктің жоғары деңгейін және әртүрлі шабуылдарға қарсы тұруды қамтамасыз етеді. Бұл тұжырымдама күшті қорғанысты қамтамасыз ету үшін деректерге нақты операцияларды немесе айналымдарды қайталап қолдануды білдіреді.

AES (Advanced Encryption Standard) сияқты блоктық шифрларда шифрлау процесі әдетте деректер блогына арнайы операцияларды қолданатын бірнеше айналымнан тұрады. Әрбір раунд барысында деректер ауыстыру, ауыстыру, негізгі қолдану және т.б. сияқты түрлендірулердің тұтас жиынтығынан өтеді. Барлық айналымдар аяқталғаннан кейін деректер шифрланады.

Итерацияларды қолдану шифрдың қауіпсіздігін күшейтуге мүмкіндік береді, өйткені әрбір қосымша раунд криптоталдау тапсырмасын қиындатады. Әрбір раунд бір бағытта қайтымды, бірақ құпия кілтті білмей орындау қиын операцияларды қолдануды қамтиды. Бұл шифрдың әртүрлі шабуылдарға, соның ішінде дифференциалды криптоталдау, сызықтық криптоталдау, кілттік кеңістік шабуылдары және т.б. төзімділігін қамтамасыз етеді.

Сонымен қатар, итерацияларды пайдалану шифрланған деректерге өзгерістерді біркелкі таратуға көмектеседі, бұл алгоритмді статистикалық сипаттамаларға негізделген шабуылдарға азырақ сезімтал етеді. Бұл сонымен қатар қауіпсіздікті жақсартады және осалдықтардың ықтималдығын азайтады.

Дегенмен, тым көп итерациялар, әсіресе деректердің үлкен көлемімен жұмыс істегенде, алгоритм жұмысына әсер етуі мүмкін екенін ескеру қажет. Сондықтан қауіпсіздік пен өнімділік арасындағы теңгерім криптографиялық алгоритмдерді әзірлеудің маңызды аспектісі болып табылады.

Криптографиялық алгоритмдерде итерацияларды қолдану қауіпсіздіктің жоғары деңгейін, әртүрлі шабуылдарға қарсы тұруды және шифрланған деректердегі өзгерістердің біркелкі таралуын қамтамасыз етуге мүмкіндік береді.

*Кері операция* – шифрланған мәліметтердің шифрын ашу және бастапқы мәтінді қалпына келтіру үшін қолданылатын шифрлаудың кері процесі. Криптографиялық алгоритмдерде кері операция әдетте шифрлау процесі кезінде қолданылатын түрлендірулердің инверсиясы болып табылады.

Симметриялық шифрлау контекстінде кері операция құпия кілтті білмей-ақ қайтымды және орындалатын болуы керек. Бұл шифрланған деректерді алушыға жіберуші шифрлау үшін пайдаланған кілтті пайдаланып хабарламаның шифрын сәтті шешуге мүмкіндік береді. Кері операция бастапқы ақпаратқа қол жеткізуге мүмкіндік беретін шифрленген мәтіннен деректердің бастапқы күйін қалпына келтіреді.

Заманауи криптографиялық алгоритмдерде кері операция жиі маңызды құрамдас болып табылады, себебі ол құпия ақпаратты қауіпсіз алмасуға мүмкіндік береді. Сондай-ақ ол ашық немесе сенімсіз байланыс арналары арқылы берілген деректердің тұтастығы мен түпнұсқалығын тексеруге мүмкіндік береді.

Кері операцияның мысалы ретінде AES алгоритмі арқылы шифрланған хабарламаның шифрын ашу процесі болуы мүмкін. Бұл жағдайда шифрленген мәтін шифрлау әсерін жойатын кері түрлендірулер сериясынан өтеді, нәтижесінде түпнұсқа мәтін пайда болады.

Кері операция криптографиялық алгоритмдердің қауіпсіздігі мен функционалдығы үшін өте маңызды. Ол дұрыс іске асырылуы және дұрыс кілт берілген жағдайда деректердің сәтті шифрын шешуге болатынына көз жеткізу үшін қайтымдылықты қамтамасыз етуі керек.

Әзірленетін криптоалгоритм деректердің құпиялығын қорғау үшін қолданылатын симметриялық шифрлау принциптеріне негізделген. Бұл бекітілген өлшемді деректер блоктарында жұмыс істейтін блоктық шифр. Алгоритмнің негізгі кезеңдері шифрлау және шифрды шешуді қамтиды.

Шифрлау деректер блогының күйін инициализациялаудан басталады, содан кейін ол бірнеше ретті түрлендірулерден өтеді. Бірінші кезең - байтты ауыстыру, мұнда әрбір байт ауыстыру кестесіндегі алдын ала анықталған мәнмен ауыстырылады. Содан кейін блоктың әрбір жолындағы байттарды белгіленген позициялар санын солға жылжытатын циклдік жолды ауыстыру орын алады. Осыдан кейін бағанды араластыру операциясы қолданылады, мұнда блоктың әрбір бағанасы Галуа өрісіндегі тіркелген факторларға көбейтіледі. Соңында, нәтиже раундтық кілтті қосу арқылы нәтижеге қолданылады, мұнда әрбір күй байты сәйкес раундтық кілт байтымен XOR-ға өзгертіледі.

Шифрды шешу кері тәртіпте жүзеге асырылады. Біріншіден, соңғы айналым кілтін қосу операциясы қолданылады. Содан кейін жолды ауыстырудың, байттарды ауыстырудың және бағандарды араластырудың кері операциялары қарама-қарсы бағытта қолданылады. Шифрды шешудің соңында бастапқы хабарлама алынады.

Осылайша, криптографиялық алгоритм ақпарат блоктарын кілттің көмегімен түрлендіру арқылы деректерді қорғауды қамтамасыз етеді, сәйкес шифрды шешу кілтінсіз оны оқылмайтын етеді[19].

### 2.2.2 Алгоритмді код түрінде жүзеге асыру

Әзірленген криптографиялық алгоритмнің әрбір құрамдас бөлігі деректерді шифрлау және шифрды шешудің жалпы жұмысына ықпал ететін нақты функцияларды орындайды. Әр компоненттің рөлін қарастырайық, алгоритмде қолданылатын математикалық формулалар мен алгоритмдер: байттарды ауыстыру, қатарларды жылжыту, бағандарды араластыру.

Байттарды ауыстыру (13-сурет):

- Рөл: күйдегі әрбір байтты ауыстыру кестесіндегі (S-Box) алдын ала анықталған мәнмен ауыстырады.
- Математикалық формула:  $\text{күй}[i] = \text{sbox}[\text{күй}[i] \gg 4][\text{күй}[i] \& 0x0f]$

```
// Байттарды ауыстыру
func (sc *SimpleCrypto) subBytes(state []byte) {
    for i := 0; i < 16; i++ {
        state[i] = sbox[state[i]>>4][state[i]&0x0f]
    }
}
```

Сурет 17 Байттарды ауыстыру (GO программалау тілі).

Қатарды жылжыту (14-сурет):

- Рөл: деректер блогының әрбір күй жолындағы байттарды циклдік түрде ауыстырады.
- Математикалық формула: Жоқ, бұл байт жылжыту операциясы.

$$a_1, a_2, a_3, a_4 \rightarrow a_1, a_2, a_3, a_4$$

$$b_1, b_2, b_3, b_4 \rightarrow b_2, b_3, b_4, b_1$$

$$c_1, c_2, c_3, c_4 \rightarrow c_3, c_4, c_1, c_2$$

$$d_1, d_2, d_3, d_4 \rightarrow d_4, d_1, d_2, d_3$$

```
// Қатарды жылжыту
func (sc *SimpleCrypto) shiftRows(state []byte) {
    for row := 1; row < 4; row++ {
        sc.rotateLeft(state[row*4 : row*4+4])
    }
}
```

Сурет 18 Қатарды жылжыту (GO программалау тілі).

Бағандарды араластыру (15-сурет):

- Рөл: әрбір күй бағанын Галуа өрісіндегі тіркелген факторларға көбейтеді.
- Математикалық формула:  $\text{kүй}[i] = \text{gmul}(0x02, \text{temp}[0]) \wedge \text{gmul}(0x03, \text{temp}[1]) \wedge \text{temp}[2] \wedge \text{temp}[3]$

```
// Бағандарды матрицалық араластыру
func (sc *SimpleCrypto) mixColumns(state []byte) {
    temp := make([]byte, 4)
    copy(temp, state)

    state[0] = sc.gmul(0x02, temp[0]) ^ sc.gmul(0x03, temp[1]) ^ temp[2] ^ temp[3]
    state[1] = temp[0] ^ sc.gmul(0x02, temp[1]) ^ sc.gmul(0x03, temp[2]) ^ temp[3]
    state[2] = temp[0] ^ temp[1] ^ sc.gmul(0x02, temp[2]) ^ sc.gmul(0x03, temp[3])
    state[3] = sc.gmul(0x03, temp[0]) ^ temp[1] ^ temp[2] ^ sc.gmul(0x02, temp[3])
}
```

**Сурет 19** Бағандарды араластыру (GO программалау тілі).

Мысал:

Кілт: aon4v9dmv2uf8bx0

Хабарлама: Hello Jupiter!:) )

Шифрлау нәтижесі: f26fba624586bd6a43f66446759423e8

Шифрды шешу нәтижесі: Hello Jupiter!:) )

### 2.3 Криптоалгоритм қабілеті, оң және теріс жақтары

Өзірленген криптографиялық алгоритмнің әртүрлі криптоталдау әдістеріне қарсы тұруды жеңілдететін бірнеше сипаттамаларға ие. Алгоритм қандай криптоталдау әдістеріне қарсы тұра алатынын, сондай-ақ олардың оң және теріс жақтарын қарастырайық:

#### 1. Сызықтық және дифференциалды криптоталдау:

- Қарсылық: Сызықты емес байтты ауыстыру әрекетін және араластыру бағандарын пайдалану алгоритмді сызықтық және дифференциалды шабуылдарға төзімді етеді.

- Артықшылықтары: криптоталдау әдістерінің кейбіріне тиімді қарсы тұрады.

- Кемшіліктері: шабуылдарға кейбір өзгертулер алгоритмнің дизайнына байланысты тиімді болуы мүмкін.

#### 2. Негізгі дәрекі күш шабуылдары:

- Қарсылық: 128 бит кілт ұзындығы негізгі дәрекі күш шабуылдарына жоғары қарсылықты қамтамасыз етеді, бұл оларды іс жүзінде мүмкін емес етеді.

- Артықшылықтары: Ұзын кілт ұзындығы негізгі шабуылдарды әлдеқайда қиындатады.

- Кемшіліктері: ұзағырақ пернелерді пайдалану мүмкіндігі шабуылдарға төзімділікті арттыруы мүмкін, бірақ сонымен бірге алгоритм жұмысына әсер етуі мүмкін.

### 3. Басқа криптоталдау әдістері:

- Қарсыласу: сызықты емес операциялардың және жақсы жобаланған алгоритм құрылымының болуы белгілі мәтіндік шабуылдар немесе таңдалған мәтіндік шабуылдар сияқты басқа криптоталдау әдістерін қолдануды қиындатады.

- Артықшылықтары: криптографиялық алгоритмнің белгілі құрылымына негізделген кең ауқымды шабуылдарға жоғары қарсылық.

- Кемшіліктері: Алгоритмді жүзеге асыруға байланысты ықтимал осалдықтар оның қауіпсіздігінің бұзылуына әкелуі мүмкін.

### 4. Блок өлшемі және жолды ауыстыру операциясы:

- Қарсы шаралар: 128 биттік деректер блогын және жолдарды ауыстыру әрекетін пайдалану статистикалық талдау шабуылдарының күрделілігін арттырады.

- Артықшылықтары: жиілік пен статистикалық талдау негізінде сәтті шабуылдардың ықтималдығын азайтады.

- Кемшіліктері: өнімділікке әсер етуі мүмкін деректерді өңдеу үшін қажетті жад көлемін арттырады.

Әзірленген криптографиялық алгоритмнің әртүрлі криптоталдау әдістеріне төзімді ететін бірқатар сипаттамаларға ие. Дегенмен, толық қауіпсіздікті қамтамасыз ету үшін мұқият тестілеу және осалдықты талдау, сондай-ақ криптографиялық әдістер мен шабуылдардың дамуын бақылау маңызды. Алайда, әзірленген криптоалгоритм нағыз веб-сайт қорғау жарамасыз. Бұл жай ғана алгоритм жасауға қажетті бірқатар әдістерді қамтиды және алгоритмнің жасалу және жұмыс жасау принциптерін көрсетеді[20]. TLS 1.3 протоколы қатысу үшін алгоритм криптография стандарттары қауымдастығымен мақұлдап, TLS 1.3 протоколы белгілеген қауіпсіздік талаптарына сай болуы керек, сонымен бірге оны кең қауымдастықтың қабылдауы мен сенімін де талап етеді.

## Қорытынды

Қазіргі цифрлық әлемде веб-сайттың қауіпсіздігі, әсіресе кибершабуылдар мен деректерді бұзу қаупінің өсуі жағдайында маңызды рөл атқарады. Осы дипломдық жұмыс шеңберінде веб-сайттардың қауіпсіздігін қамтамасыз етуге бағытталған криптографиялық алгоритмді зерттеу және әзірлеу жүргізілді. Бұл процесс қолданыстағы криптографиялық әдістерді талдауды және жаңа алгоритмге қойылатын талаптарды анықтауды қамтыды.

Бұл дипломдық жұмыстың мақсаты веб-сайттың қауіпсіздігін қамтамасыз ету үшін криптографиялық алгоритмді зерттеу және әзірлеу болды. Жұмыс барысында келесі міндеттер орындалды:

- Қолданыстағы криптографиялық алгоритмдерді талдау: Қолданыстағы шифрлау және хэштеу әдістеріне шолу жүргізілді, олардың артықшылықтары мен кемшіліктері анықталды.

- Алгоритм талаптарын анықтау: Шабуылдарға төзімділік, тиімділік және әртүрлі сценарийлерде пайдалану мүмкіндігі сияқты веб-сайттың қауіпсіздігін қамтамасыз ету үшін криптографиялық алгоритмге қойылатын негізгі талаптар ерекшеленген.

- Криптографиялық алгоритмді құру: Талдау негізінде және талаптарды ескере отырып, қажетті қауіпсіздік қасиеттеріне ие өзіміздің криптографиялық алгоритміміз жасалды. Веб-сайт көп жағдайда мақаларды бөлісу, компаниялардың өзін таныстыруы, жалпыға қолжетімді пайдалы ақпараттарды тарату немесе жарнамалау үшін құрылады. Осы тұрғыда веб-бағдарламалар веб-сайттарға қарағанда әлдеқайда маңызды, құпия ақпараттармен алмасады. Сол себепті, веб-сайт үшін криптоалгоритмнің жылдам, өнімді жұмыс жасауы күделі, көп операциялар мен көп уақыт талап ететін алгоритмдерге қарағанда бірінші орынға қойылады. Өзерленген криптоалгоритм де өнімділік пен жылдамдыққа негізделген.

Зерттеу нәтижелері әзірленген криптографиялық алгоритмнің әртүрлі шабуылдарға жоғары төзімділігін және веб-сайттарда қолдануда тиімді екенін көрсетті. Ол негізгі қауіпсіздік талаптарына жауап береді және пайдаланушылардың құпия ақпаратын қорғау үшін сәтті пайдаланылуы мүмкін.

Жұмыстың болашақ әзірлемелеріне криптографияны тереңірек зерттеу, шабуылдың жаңа әдістерін есепке алу үшін алгоритмді жетілдіру және ақпараттық қауіпсіздіктің басқа салаларында қосымшаларды табу кіреді.

Дегенмен, криптографиялық алгоритмдерді әзірлеу үздіксіз процесс және одан әрі зерттеулер мен жетілдірулер қажет екенін атап өткен жөн. Болашақта алгоритмнің функционалдығын кеңейту, оны қауіптердің жаңа түрлеріне бейімдеу және қолданыстағы қауіпсіздік стандарттарымен біріктіру жоспарлануда.

Бұл дипломдық жұмыс веб-сайттарды киберқауіптерден тиімді қорғай алатын жаңа алгоритмді ұсына отырып, криптография және ақпараттық

қауіпсіздік саласына елеулі үлес қосады. Зерттеу нәтижелерін осы саладағы одан әрі ғылыми және практикалық жұмыстарға негіз ретінде пайдалануға болады.



## Пайдаланылган әдебиеттер тізімі

- 1 Фергюсон Нильс. Практическая криптография. // Шнайер Б. — Москва: Вильямс, 2004. — 432 б.
- 2 Заурбеков С.С., Отелбаев М. Защита информации и основы криптографии (каз.). — Астана, 2003. — 381 б.
- 3 Шнайер Б. Applied Cryptography. — Нью-Йорк: John Wiley & Sons, 1996. — 784 б.
- 4 Стил Т. Black Hat Go: Программирование для хакеров и пентестеров. // Паттен К., Коттманн Д. — Питер, 2022. — 384 б.
- 5 Козлов В.А. Математические основы криптологии. Пятигорск: ПГЛУ, 2012. — 129 б.
- 6 Моллин, Р. А. An Introduction to Cryptography. — CRC Press, 2007. — P. 80. — 413 б.
- 7 Калужнин Л. А., Сущанский В. И. Преобразования и перестановки: Пер. с укр. — 2-е изд. — Москва: Наука, 1985. — 160 б.
- 8 Паар Х., Пеллц Я. Understanding Cryptography: A Textbook for Students and Practitioners. — Берлин: Springer, 2010. — 412 б.
- 9 Менезес А. Дж., ван Ооршот, Пол С., Ванстоун, Скотт А. Handbook of Applied Cryptography. — Бока-Ратон, Флорида: CRC Press, 1996. — 820 б.
- 10 Спиричева, Н. Р. Алгоритмы блочной криптографии учебно-методическое пособие. — Екатеринбург : Изд-во Урал, 2013,-78, б.
- 11 Залевский М. The Tangled Web: A Guide to Securing Modern Web Applications. — Себастьян, Калифорния: No Starch Press, 2011. — 320 б.
- 12 Петров А.А. Компьютерная безопасность. Криптографические методы — М.: ДМК, 2000. — 448 б.
- 13 Patrick N. Taking a Closer Look at the SSL/TLS Handshake — <https://www.thesslstore.com/blog/explaining-ssl-handshake/#the-tls-12-handshake-step-by-step> – интернеттегі мақала.
- 14 Бабаш А. В. Криптографические методы защиты информации: учебник для вузов / А. В. Бабаш, Е. К. Баранова. - Москва : КноРус, 2016. — 189 б.
- 15 AES (стандарт шифрования) - [https://ru.wikipedia.org/wiki/AES\\_\(%D1%81%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D1%82\\_%D1%88%D0%B8%D1%84%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F\)](https://ru.wikipedia.org/wiki/AES_(%D1%81%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D1%82_%D1%88%D0%B8%D1%84%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)) – интернеттегі мақала.
- 16 SHA-1 - <https://ru.wikipedia.org/wiki/SHA-1> - интернеттегі мақала.
- 17 Денис Г. Криптоалгоритмы. Классификация с точки зрения количества ключей - <https://habr.com/ru/articles/336578/> - интернеттегі мақала.
- 18 Шифрование - <https://ru.wikipedia.org/wiki/%D0%A8%D0%B8%D1%84%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5> - интернеттегі мақала.
- 19 Капалова Н.А. Разработка и исследование алгоритма шифрования и создание программно-аппаратного комплекса для его реализации -

<https://iict.kz/ru/razrabotka-i-issledovanie-algoritma-shifrovaniya-i-sozdanie-programmno-apparatnogo-kompleksa-dlya-ego-realizacii/> - интернеттегі мақала.

20 Оловянишников, А. Р. Разработка алгоритма и программного обеспечения для шифрования данных / А. Р. Оловянишников, Е. Е. Симаков. — интернеттегі мақала — 2021. — № 2 (43). — б. 46-52.



Дипломдық жұмыс/жоба, магистрлік диссертация/жоба туралы ақпарат			
Оқу жылы:	2023 - 2024	Факультеті:	Ақпараттық технологиялар
Мамандығы:	B058	Ақпараттық қауіпсіздік	
	(шифрі)	(атауы)	
Ғылыми жетекшісі:	Коржаспаев Арман Ермакович		
	(толық Т.А.Ә.)		
Бітіруші:	Бауыржан Әлнұр Нұрболұлы		
	(толық Т.А.Ә.)		
Оқу тілі:	Қазақ тілі	Жұмыстың жазылған негізгі тілі:	Қазақ тілі
Жұмыстың тақырыбы:	Веб-сайттың қауіпсіздігін қамтамасыз ету үшін криптографиялық алгоритмді зерттеу және әзірлеу		
Жұмыстың түрі:	Дипломдық жұмыс		
	(дипломдық жұмыс/жоба, магистрлік диссертация/жоба)		

Анықтама берген күн: 07.05.2024



8504094409