**ЎЗБЕКИСТОН РЕСПУБЛИКАСИ ОЛИЙ ВА ЎРТА МАХСУС ТАЪЛИМ ВАЗИРЛИГИ**

**МИРЗО УЛУҒБЕК НОМИДАГИ ЎЗБЕКИСТОН МИЛЛИЙ УНИВЕРСИТЕТИ**

**Р.Д. АЛОЕВ, Ш.Б. Жураев**

# Ахборот ҳимоясининг криптографик усуллари

**ЎҚУВ-УСЛУБИЙ ҚЎЛЛАНМА**

**Тошкент-2018**

Ушбу ўқув-услубий қўлланма Ахборот ҳимоясининг криптографик усулларини ўрганувчиларига мўлжалланган бўлиб, унда Ахборот хавфсизлигини таминловчи бир қанча криптографик алгоритмлар ҳақида қисқача баён этилади. Ушбу қўлланмадан, 5А330302 - Ахборот хавфсизлиги (математик ва дастурий таъминоти) мутахассислиги бўйича таҳсил олаётган талабаларга мўлжалланган «**Ахборот ҳимоясининг криптографик усуллари**» фани учун таянч ўқув қулланма сифатида фойдаланиш мумкин.

**Муаллифлар:**

**Алоев Р.Д.**- ЎзМУ, математик моделлаштириш ва криптоанализ кафедрасининг мудири, физика-математика фанлари доктори, профессор.

**Жураев Ш.Б.** - ЎзМУ, математик моделлаштириш ва криптоанализ кафедраси магистранти.

**Тақризчилар:**

**Мухамедиева Д.Т.** - Тошкент ахборот технологиялари университети қошидаги Дастурий маҳсулотлар ва аппарат-дастурий мажмуалар яратиш маркази етакчи илмий ходими, техника фанлари доктори, профессор

**Музафаров Х.А.** - ЎзМУ, математик моделлаштириш ва криптоанализ кафедрасининг профессори, физика-математика фанлари доктори.

**Масъул муҳаррир:**

Ўқув услубий қўлланма Мирзо Улуғбек номидаги Ўзбекистон Миллий университети илмий кенгаши томонидан нашрга тавсия этилган (  -йил,  -баённома)

## Мундарижа

## 1. Шифрлаш функцияларини қуриш. Бир томонлама функциялар(*1 амалий машғулот*).

Фараз қиламиз очиқ матндаги алфавит ҳарфларининг сонли эквиваленти қуйидаги кўринишда:

$$A = \{1, 2, \dots, n\},$$

ва $m$ узунликдаги $T$ дастлабки очиқ матн эса

$$T = t_1, t_2 \dots t_m, t_m \in A.$$

кўринишда берилган бўлсин.

Шифрлаш функцияси деб аргументнинг $x_1 \neq x_2$ ҳар хил қийматларига функциянинг $f(x_1) \neq f(x_2)$ ҳар хил қийматларини мос қўйадиган функцияга айтилади.

Масалан

$$T = t_1, t_2 \dots t_m, t_m \in A.$$

очиқ матн учун

$$E = f(t_1)f(t_2) \dots f(t_m)$$

шифрматнга эгамиз, бу ерда $f(t_m)$ ихтиёрий бутун бўлмаган ҳақиқий қиймат бўлиши мумкин.

*1 мисол.* Шифрлаш функцияси сифатида

$$Y = \sin(\pi(x-1))/2(n-1)$$

функциясини олсак, унда у

$$A = \{1, 2, \dots, n\}$$

алфавит ҳарфларини [0,1] кесмага акслантиради.

Агарда $T = \text{БАС} = 2\_1\_3$ (рус алифбоси учун $n = 33$) бўлса, унда унинг криптограммаси қуйидаги кўринишда бўлади:

$$E = \sin \pi/64 \_ \sin 0 \_ \sin \pi/32 \ .$$

*2-нчи мисол.* Энди биз тез ўсувчи шифрлаш функцияси ёрдамида шифр мисолини кўриб чиқамиз, айнан $I(u, p)$-($GF(p)$-Галуа майдонида даражаси $u$ бўлган кўпҳадлар сони) функцияси ёрдамида.

Ҳар бир $t \in A$-ни $GF(p)$да даражаси $t$ бўлган оддий кўпҳад коэффициентларидан иборат сўзга мос қўйамиз. Бундай сўзни узунлиги $n+1$ га тенг.

Масалан $p = 2$ бўлганда (1-нчи жадвалга қаранг), агар

$$T = \text{БАС} = 2\_1\_3$$

бўлса, унда қуйидаги тўртта шифрматларни оламиз:

1. $E = 00\dots111\_00\dots01\_00\dots1010$
2. $E = 00\dots111\_00\dots01\_00\dots1110$
3. $E = 00\dots111\_00\dots11\_00\dots1010$
4. $E = 00\dots111\_00\dots11\_00\dots1110$

1- жадвал.

| $t$ | 1 | 2 | 3 | 4 | … | $n$ |
|---|---|---|---|---|---|---|
| $f(x)$ | $x$<br>$x+1$ | $x^2+x+1$ | $x^3+x+1$<br>$x^3+x^2+1$ | $x^4+x^3+x^2+1$ | | |
| $e$ | 00…01<br>00…11 | 00…111 | 00…1011<br>00…1101 | 00…11101 | | |

Таъкидлаш жоизки $T$ – очиқ матнни бу шифрлаш усули унда ҳарфларни учрашув частотасини тарқатиб ташлайди. Ундан ташқари $t \in A$ битта ҳарфини ифодалаш учун тартиби $t$ бўлган $p$ ўсиши билан миқдори тез ўсадиган оддий купҳадлар $I(t, p)$ миқдоридан фойдаланиш мумкин.

Кўрсатилган функцияларни шифрлаш функциялари сифатида ишлатиш мумкинлигига қарамасдан, амалда одатда бир томонлама функция ёки $S$ сирга эга бўлган бир томонлама функциялардан фойдаланишади. Бир томонлама функция тушунчаси Диффи ва Хеллман томонидан 1976 йилда киритилган.

*Бир томонлама* функция деб қуйидаги иккита хоссага эга бўлган $F(x)$ функцияга айтилади:

1. $F(x)$ қийматини ҳисоблаш полиномиал алгоритми мавжуд;
2. $F(x)$ ни тескари қийматини ҳисоблаш, яъни берилган $y$ бўйича $F(x)=y$ тенгламани $x$-га нисбатан ечиш полиномиал алгоритми мавжуд эмас.

$S$ сирга эга бўлган *бир томонлама* функция деб $S$ параметрга боғлиқ бўлган қуйидаги хоссаларга эга бўлган $F_S(x)$ функцияга айтилади:

1. Ҳар қандай $S$ учун $F_S(x)$ қийматини ҳисоблаш полиномиал алгоритми мавжуд;
2. $S$ маълум бўлмаганда $F_S(x)$ ни тескари қийматини ҳисоблаш полиномиал алгоритми мавжуд эмас;
3. $S$ маълум бўлганда $F_S(x)$ ни тескари қийматини ҳисоблаш полиномиал алгоритми мавжуд.

Бир томонлама функцияларни мавжудлиги бардошли криптотизимлар қуришни зарурий ва етарли шартидир. Бунда ушбу функцияларни тескари қийматини ҳисоблаш муайян мураккаб математик масалага эквивалентдир.

## Масала ва машқлар

1. Қуйидаги функциялардан қайси бирлари шифрлаш функциялари бўла олади?
   1) $y = 7x + 13;$
   2) $y = 13x^2 + 6 + 14;$
   3) $y = x - 11 + \cos x.$

2. $T =$ НЕ_ТОРОПИСЬ очиқ матнни $Y = E(x)$ шифлаш функцияси ёрдамида шифрланг.

$$T = \text{АНТИКАНАЛЬЧИК\_ВЕТТЕСС}$$

очиқ матн криптограммасини қуринг.

3. Фараз қиламиз потенциал фойдаланувчи тизимга кириш учун бир замонда $F(x) = x + K \bmod(1001)$ функция қийматини қўллади, бу ерда $x$ таймернинг охирги бешта ишончли рақамлари қиймати, $K$ эса фойдаланувчини ўзини сонли пароли. У тизимга яна бир карра кириб биладими?
4. Сирга эга бўлган ва бўлмаган бир томонлама функцияларга мисоллар келтиринг.
5. Очиқ $T$ матнни аниқланг, агарда шифрматн қуйидаги кўринишда бўлса

$$E = \text{ФВМЖТИВФЮ}$$

6. Фараз қиламиз $x_1, x_2, x_3$

$\psi(x) = x^3 + x - 7$

кўпҳадни илдизлари бўлсин. $T =$ НЕ_ОШИБИСЬ хабарни

$f(x) = 2x^7 + 7x^5 - 14x^4 - 4x^3 - 35x^2 - x + 10$

функция ёрдамида қуйидаги услубда шифрланг: $e = f(x_k) + s$, бу ерда $s$ жорий ҳарфнинг сонли эквиваленти, $x_k$ эса $\psi(x)$ нинг ихтиёрий илдизи.

7. $T =$ ВСТРЕЧА_ОТМЕНЯЕТСЯ_ЯВКА_РАСКРЫТА хабарни 2-нчи жадвал калити билан шифрланг.
8. $Y = E(X) = [X]$ функцияси $T$ очиқ матндаги ҳарфларнинг учрашиш частотасини тарқатиб юборади. Очиқ матнни шифрлаш учун масалан қуйидаги жавдалдан фойдаланамиз:

2-нчи жадвал

| t | 1 | 2 | 3 | …. | n |
|---|---|---|---|---|---|
| e | [0,1) | [1,2) | [2,3) | | [n-1,n) |

Масалан $T =$ БАС $= 2\_1\_3$ бўлса, у ҳолда қуйидаги шифрматнга эга бўламиз:

$E = 1.\alpha\_0.\beta\_2.\gamma$

бу ерда $\alpha, \beta, \gamma$ қийматлари диапазони жуда катта. Ушбу шифр қандай хоссаларга эга?

9. Ихтиёрий $n \neq 2$ учун $GF(2)$ да $n$-нчи тартибил келтирилмайдиган кўпҳадга мос $n$ даврнинг циклик кетма-кетлиги мавжудлигини исботланг. Кўрсатилган циклик кетма-кетликлардан фойдаланиб берилган хабар криптограммасини қуринг.

## 2. Оддий ўрнига қўйиш (алмаштириш) усули, ўрин алмаштириш усули, блокли шифрлар усули, гаммалаштириш усули(*2 амалий машғулот*).

Фараз қиламиз $A=\{t\mid t\in T\}$, $P=\{e\mid e\in E\}$мос равишда хабар ва шифрлаш алифболари бўлсин, бунда умумий ҳолда $n=|A|\geq|P|=m$ бўлади. Бир алфавитли алмаштиришда $A=P$ шарт бажарилганда ҳар бир $t\in T$ ҳарфга битта $e\in E$ ҳарфи мос қўйилади, яъни

$$e=K_1*t+K_2(\mathrm{mod}n),$$

ёки

$$e=t+K_2(\mathrm{mod}n),$$

Бу ерда $K_1$ ва $K_2$- бир вақтда ҳам шифрлаш ва ҳам дешифрлаш махфий калитлари, $n$ - $A$ алифбосидаги белгилари миқдори. $t\pm K(\mathrm{mod}n)$ кўринишдаги ёзув $A$ алифбосидаги $t$ ҳарфдан $\pm K(\mathrm{mod}n)$ позицияга ортда жойлашган ҳарфни англатади. Масалан, агарда

$$T=\text{ ШИФР ЗАМЕНЫ}$$

бўлса, унда $e=t+13(\mathrm{mod}33)$ бўлганда

$$E=\text{ ДЦАЭМФНЩТЪЗ}$$

шифрмаълумотга эга бўламиз. Кўриниб турибтики $T$ ни $E$ бўйича тиклаш учун $t=e-13(\mathrm{mod}33)$ тенгликдан фойдаланиш етарли.

Агарда $K$ калит сифатида муайян аниқ сўз ёки узунлиги $d$ бўлган

$$K=K_1K_2\ldots K_d$$

фраза (*махфий калит*) олинса, унда шу билан биз янада мураккаброқ бўлган ўрнига қўйиш усулини аниқлаймиз, унинг туб моҳияти қуйидаги шифрга (*Вижинер шифри*) олиб келади:

$e_i=t_i+K_i(\mathrm{mod}n)$,

бу ерда

$$K_i=\begin{cases}i(mod\ d), agar\ \ I/d\\ d, \quad\ agar\ i/d\end{cases}$$

$t_i$-очиқ матннинг $i$-нчи белгиси, $e_i$-шифрматннинг $i$-нчи белгиси. Масалан, агарда очиқ маълумот

$$T=\text{ ШИФР ЗАМЕНЫ}$$

ва

$$K=\text{ РРВО}$$

узунлиги 4га тенг бўлган махфий калит бўлса, унда Вижинер шифрматни қуйидаги кўринишда бўлади:

$$E=\text{ ИЩЧЯРШГЫЦЮЮ.}$$

Амалда узунлиги $d$ га тенг бўлган

$$K = K_1 K_2 \dots K_d$$

махфий калит билан Вижинер шифри бўйича $T$ очиқ матнни шифрлаш ва $E$ шифрматнни дешифрлаш учун Вижинер жадвалидан фойдаланиш мумкин:

Вижинер жадвали.

| А | Б | В | Г | Д | Е | Ё | Ж | З | И | Й | К | Л | М | Н | О | П | Р | С | Т | У | Ф | Х | Ц | Ч | Ш | Щ | Ъ | Ы | Ь | Э | Ю | Я | _ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Б | В | Г | Д | Е | Ё | Ж | З | И | Й | К | Л | М | Н | О | П | Р | С | Т | У | Ф | Х | Ц | Ч | Ш | Щ | Ъ | Ы | Ь | Э | Ю | Я | _ | А |
| В | Г | Д | Е | Ё | Ж | З | И | Й | К | Л | М | Н | О | П | Р | С | Т | У | Ф | Х | Ц | Ч | Ш | Щ | Ъ | Ы | Ь | Э | Ю | Я | _ | А | Б |
| Г | Д | Е | Ё | Ж | З | И | Й | К | Л | М | Н | О | П | Р | С | Т | У | Ф | Х | Ц | Ч | Ш | Щ | Ъ | Ы | Ь | Э | Ю | Я | _ | А | Б | В |
| Д | Е | Ё | Ж | З | И | Й | К | Л | М | Н | О | П | Р | С | Т | У | Ф | Х | Ц | Ч | Ш | Щ | Ъ | Ы | Ь | Э | Ю | Я | _ | А | Б | В | Г |
| Е | Ё | Ж | З | И | Й | К | Л | М | Н | О | П | Р | С | Т | У | Ф | Х | Ц | Ч | Ш | Щ | Ъ | Ы | Ь | Э | Ю | Я | _ | А | Б | В | Г | Д |
| Ё | Ж | З | И | Й | К | Л | М | Н | О | П | Р | С | Т | У | Ф | Х | Ц | Ч | Ш | Щ | Ъ | Ы | Ь | Э | Ю | Я | _ | А | Б | В | Г | Д | Е |
| Ж | З | И | Й | К | Л | М | Н | О | П | Р | С | Т | У | Ф | Х | Ц | Ч | Ш | Щ | Ъ | Ы | Ь | Э | Ю | Я | _ | А | Б | В | Г | Д | Е | Ё |
| З | И | Й | К | Л | М | Н | О | П | Р | С | Т | У | Ф | Х | Ц | Ч | Ш | Щ | Ъ | Ы | Ь | Э | Ю | Я | _ | А | Б | В | Г | Д | Е | Ё | Ж |
| И | Й | К | Л | М | Н | О | П | Р | С | Т | У | Ф | Х | Ц | Ч | Ш | Щ | Ъ | Ы | Ь | Э | Ю | Я | _ | А | Б | В | Г | Д | Е | Ё | Ж | З |
| Й | К | Л | М | Н | О | П | Р | С | Т | У | Ф | Х | Ц | Ч | Ш | Щ | Ъ | Ы | Ь | Э | Ю | Я | _ | А | Б | В | Г | Д | Е | Ё | Ж | З | И |
| К | Л | М | Н | О | П | Р | С | Т | У | Ф | Х | Ц | Ч | Ш | Щ | Ъ | Ы | Ь | Э | Ю | Я | _ | А | Б | В | Г | Д | Е | Ё | Ж | З | И | Й |
| Л | М | Н | О | П | Р | С | Т | У | Ф | Х | Ц | Ч | Ш | Щ | Ъ | Ы | Ь | Э | Ю | Я | _ | А | Б | В | Г | Д | Е | Ё | Ж | З | И | Й | К |
| М | Н | О | П | Р | С | Т | У | Ф | Х | Ц | Ч | Ш | Щ | Ъ | Ы | Ь | Э | Ю | Я | _ | А | Б | В | Г | Д | Е | Ё | Ж | З | И | Й | К | Л |
| Н | О | П | Р | С | Т | У | Ф | Х | Ц | Ч | Ш | Щ | Ъ | Ы | Ь | Э | Ю | Я | _ | А | Б | В | Г | Д | Е | Ё | Ж | З | И | Й | К | Л | М |
| О | П | Р | С | Т | У | Ф | Х | Ц | Ч | Ш | Щ | Ъ | Ы | Ь | Э | Ю | Я | _ | А | Б | В | Г | Д | Е | Ё | Ж | З | И | Й | К | Л | М | Н |
| П | Р | С | Т | У | Ф | Х | Ц | Ч | Ш | Щ | Ъ | Ы | Ь | Э | Ю | Я | _ | А | Б | В | Г | Д | Е | Ё | Ж | З | И | Й | К | Л | М | Н | О |
| Р | С | Т | У | Ф | Х | Ц | Ч | Ш | Щ | Ъ | Ы | Ь | Э | Ю | Я | _ | А | Б | В | Г | Д | Е | Ё | Ж | З | И | Й | К | Л | М | Н | О | П |
| С | Т | У | Ф | Х | Ц | Ч | Ш | Щ | Ъ | Ы | Ь | Э | Ю | Я | _ | А | Б | В | Г | Д | Е | Ё | Ж | З | И | Й | К | Л | М | Н | О | П | Р |
| Т | У | Ф | Х | Ц | Ч | Ш | Щ | Ъ | Ы | Ь | Э | Ю | Я | _ | А | Б | В | Г | Д | Е | Ё | Ж | З | И | Й | К | Л | М | Н | О | П | Р | С |
| У | Ф | Х | Ц | Ч | Ш | Щ | Ъ | Ы | Ь | Э | Ю | Я | _ | А | Б | В | Г | Д | Е | Ё | Ж | З | И | Й | К | Л | М | Н | О | П | Р | С | Т |
| Ф | Х | Ц | Ч | Ш | Щ | Ъ | Ы | Ь | Э | Ю | Я | _ | А | Б | В | Г | Д | Е | Ё | Ж | З | И | Й | К | Л | М | Н | О | П | Р | С | Т | У |
| Х | Ц | Ч | Ш | Щ | Ъ | Ы | Ь | Э | Ю | Я | _ | А | Б | В | Г | Д | Е | Ё | Ж | З | И | Й | К | Л | М | Н | О | П | Р | С | Т | У | Ф |
| Ц | Ч | Ш | Щ | Ъ | Ы | Ь | Э | Ю | Я | _ | А | Б | В | Г | Д | Е | Ё | Ж | З | И | Й | К | Л | М | Н | О | П | Р | С | Т | У | Ф | Х |
| Ч | Ш | Щ | Ъ | Ы | Ь | Э | Ю | Я | _ | А | Б | В | Г | Д | Е | Ё | Ж | З | И | Й | К | Л | М | Н | О | П | Р | С | Т | У | Ф | Х | Ц |
| Ш | Щ | Ъ | Ы | Ь | Э | Ю | Я | _ | А | Б | В | Г | Д | Е | Ё | Ж | З | И | Й | К | Л | М | Н | О | П | Р | С | Т | У | Ф | Х | Ц | Ч |
| Щ | Ъ | Ы | Ь | Э | Ю | Я | _ | А | Б | В | Г | Д | Е | Ё | Ж | З | И | Й | К | Л | М | Н | О | П | Р | С | Т | У | Ф | Х | Ц | Ч | Ш |
| Ъ | Ы | Ь | Э | Ю | Я | _ | А | Б | В | Г | Д | Е | Ё | Ж | З | И | Й | К | Л | М | Н | О | П | Р | С | Т | У | Ф | Х | Ц | Ч | Ш | Щ |
| Ы | Ь | Э | Ю | Я | _ | А | Б | В | Г | Д | Е | Ё | Ж | З | И | Й | К | Л | М | Н | О | П | Р | С | Т | У | Ф | Х | Ц | Ч | Ш | Щ | Ъ |
| Ь | Э | Ю | Я | _ | А | Б | В | Г | Д | Е | Ё | Ж | З | И | Й | К | Л | М | Н | О | П | Р | С | Т | У | Ф | Х | Ц | Ч | Ш | Щ | Ъ | Ы |
| Э | Ю | Я | _ | А | Б | В | Г | Д | Е | Ё | Ж | З | И | Й | К | Л | М | Н | О | П | Р | С | Т | У | Ф | Х | Ц | Ч | Ш | Щ | Ъ | Ы | Ь |
| Ю | Я | _ | А | Б | В | Г | Д | Е | Ё | Ж | З | И | Й | К | Л | М | Н | О | П | Р | С | Т | У | Ф | Х | Ц | Ч | Ш | Щ | Ъ | Ы | Ь | Э |
| Я | _ | А | Б | В | Г | Д | Е | Ё | Ж | З | И | Й | К | Л | М | Н | О | П | Р | С | Т | У | Ф | Х | Ц | Ч | Ш | Щ | Ъ | Ы | Ь | Э | Ю |
| _ | А | Б | В | Г | Д | Е | Ё | Ж | З | И | Й | К | Л | М | Н | О | П | Р | С | Т | У | Ф | Х | Ц | Ч | Ш | Щ | Ъ | Ы | Ь | Э | Ю | Я |

## Масалалар ва машқлар

1. ROT13(ОС UNIX) программаси лотин алифбоси ҳарфларини циклик тарзда 13 позицияга ўнг томонга суради. Фақат ROT13 программасидан фойдаланиб қандай қилиб криптограммани дешифрлаш мумкин?
2. Фараз қиламиз ЛЕДЕНЕЦ кодланган матн очиқ матннинг ПОВЕРНУТЬ КЛЮЧ НА 90% ВПРАВО фразасига мос келади.
   ЛЕДЕНЕЦ+ЛЕДЕНЕЦ+ЛЕДЕНЕЦ
   кодланган матнни дешифрланг.
3. Қуйидаги тенгликларни исботланг:
   а) $Y \oplus K = X$;
   б) $Y \oplus Y' = X$;

бу ерда $Y^{'}$ - $K$ калитнинг узунлиги катталигига сурилган шифрматн нусхаси.

4. Модул 2 бўйича қўшиш амали ёрдамида шифрлаш усули кучсиз бардошликка эга эканлигини исботланг.
5. Қандай қилиб шифрматндан келиб чиқиб модул 2 бўйича қўшиш амали ёрдамида шифрлаш усулининг махфий калитини топиш мумкин?
6. Ҳарфлаш учраши статистикасини қўллаб криптотаҳлил ўтказинг ва кириллча ҳарфларни суриш ёрдамида шифрланган

$$E = \text{ЫЛЧУВФНУЮЕГИХВФСЗИУЙГРЛИВХИНФХГ}$$

криптограммани дешифрланг.

7. Фараз қиламиз

$$X = \text{ШИФР\_СКРЫВАЕТ\_СОДЕРЖАНИЕ\_ТЕКСТА}$$

очиқ матн бўлсин ва унга жадвалдаги алмаштириш калити бир марта ва ҳосил бўлган сўзга яна бир марта ва ҳ.к. қўлланилади. Қанча шундай алматиришлардан кейин биринчи маротаба дастлабки $T$ очиқ матн пайдо бўлади?

Жадвал

| А | Б | В | Г | Д | Е | Ж | З | И | Й | К | Л | М | Н |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Т | У | Ф | Х | Ц | Ч | Ш | Щ | Ъ | А | Б | В | Г | Д |

| О | П | Р | С | Т | У | Ф | Х | Ц | Ч | Ш |
|---|---|---|---|---|---|---|---|---|---|---|
| Е | Ж | З | И | Й | К | Л | М | Н | О | П |

| Щ | Ъ | Ы | Ь | Э | Ю | Я | _ |
|---|---|---|---|---|---|---|---|
| Р | С | Ь | Э | Ю | Я | _ | Ы |

8. **(Полибия шифри).** Қуйидаги жадвал асосида лотин алифбоси ҳарфларига биграммларни мос қўйиш усулини кўриб чиқамиз:

| | A | B | C | D | E |
|---|---|---|---|---|---|
| A | A | B | C | D | E |
| B | F | G | H | I | K |
| C | L | M | N | O | P |
| D | Q | R | S | T | U |
| E | V | W | X | Y | Z |

Ҳар бир ҳарфга, ушбу ҳарф жойлашган қатор ва устунни кўрсатувчи жуфт ҳарфларни мос қўйамиз (масалан V-EA, R-DB).

$$T = \text{ТЛЮСТЕН ВАШИ ФАТИМА НАШИ ДРУЗЬЯ}$$

очиқ матнни шифрланг. Ушбу шифрл бир алифболи бўладими?

9. $T$ = ЕС ИМ АНУШ АЙАСТАНИ АРЕВААМ БАРН ЭМ СИРУМ МЕР ЙИН САЗИ ВОГБАНВАГ ЛАЦАКУМАЦ ЛАРН ЭМ СИРУМ криптограммани дешифрланг.
10. $T$ = ЕИЙТДЕФЪЙЭ ФЭКЧЛЫ ДЧФЕЩГЕШДЕ СКЧ ВЯУЬЧ ШЩО МЩЙН_ЧЧ АЧФЫУЧ КИКХКУВ_ЯЙ

матнни тикланг.

11. Фараз қиламиз бирор очиқ матннинг криптограммаси ИЩЧЯРШГЫЦЮЮ кўринишда бўлсин:

$$(25+17)mod33 = 09 => y_1 = \text{И}$$
$$(09+17)mod33 = 26 => y_2 = \text{Щ}$$
$$(21+03)mod33 = 24 => y_3 = \text{Ч}$$
$$(17+15)mod33 = 32 => y_4 = \text{Я}$$
$$(33+17)mod33 = 17 => y_5 = \text{Р}$$
$$(08+17)mod33 = 25 => y_6 = \text{Ш}$$
$$(01+03)mod33 = 04 => y_7 = \text{Г}$$
$$(13+15)mod33 = 28 => y_8 = \text{Ы}$$
$$(06+17)mod33 = 23 => y_9 = \text{Ц}$$
$$(14+17)mod33 = 31 => y_{10} = \text{Ю}$$
$$(28+03)mod33 = 31 => y_{11} = \text{Ю}$$

Криптотаҳлил ўтказинг ва ушбу шифрни очинг.

12. (Вижинер шифри). $T$ очиқ маълумотдан $E$ шифрматнни ҳосил қилиш учун, дастлаб узунлиги $d$ бўлган $K = K_1K_2 \dots K_d$ махфий калитни $T$ очиқ маълумот ҳарфлари устида такрорлаб ёзиб чиқамиз. Бундан кейин очиқ маълумотни биринчи ҳарфидан бошлаб навбатдаги $e_i$ шифрни $t_i$ қатор ($t_i$ ҳарф жойлашган биринчи қатор) ва $K_i$ устун ($K_i$ ҳарф жойлашган биринчи устун) кесишмасида жойлашган ҳарф сифатида топамиз.

$E$ шифрматннинг навбатдаги $e_i$ ҳарфини дешифрлаш учун, уни жадвалдаги $e_i$ қатор ва $K_i$ устун кесишмасида жойлашган $t_i$ ҳарфи сифатида топамиз (бошланғич ҳисоб нуқтаси сифатида Вижинер жадвалининг биринчи сатри ва биринчи устунини танлаш лозим). Масалан $K$ махфий калит битта ЗИМА сўзидан, яъни $K = \text{ЗИМА}$ ва очиқ матн

$$K = \text{ШИФР\_ВИЖЕНЕРА\_Д Л Я\_Х АКЕРОВ}$$

иборат бўлсин.

У ҳолда Вижинер жадвали ёрдамида

$T$ =Ш И Ф Р _ В И Ж Е Н Е Р А _ Д Л Я _ Х А К Е Р О В

$K$ =З И М А З И М А З И М А З И М А З И М А З И М А З

$E$ =А Р Б Р Ж К Ф Ж М Х С Р З З Р Л Ж З В А С Н Э О Й

Ушбу Вижинер шифри қандай хоссаларга эга? Криптотаҳлили учун самарали усул таклиф этинг.

13. Модул 2 бўйича қўшиш амали ёрдамида шифрлаш усулини кўриб чиқамиз. $T$ очиқ матн битларига $K$ калитдаги мос битларини қўшиб чиқамиз (бу ерда белгилар учун ASCII кодидан фойдаланилган).

Масалан, агарда $T = A$ бўлса

$$BAC = 4120424143_{16} =$$

$$0100\ 0001\ 0010\ 0000\ 0100$$

$$0010\ 0100\ 0001\ 0100\ 0011_2$$

Унда $K = 1101$ калитни $T$ очиқ матн тагида бошидан бошлаб 10 маротаба такрорлаб ёзиб 2 модул қўшишни қўллаб қуйидаги криптограммани ҳосил қиламиз

$$E = 1001\ 1100\ 1111\ 1101\ 1001\ 1111\ 1001\ 1100\ 1001\ 1110$$

Ёки

$$E = 9CFD9F9C9E_{16}$$

(кўрсатилган алгоритм америкада ишлаб чиқилган рақамли уяли телефонларда овозли мулоқотларни махфийлаштириш учун ишлатилади). Ушбу шифрни очиш учун қандай самарали усул тавсия этиш мумкин?

14. Алмаштириш усулининг қандай турлари мавжуд? Уларга мос мисоллар келтиринг.

### *2.1.Ўрин алмаштириш усули.*

Ўрин алмаштириш шифрида очиқ матн ҳарфлари бошқаларига алмаштирилмайди, балки уларни келиш тартибининг ўзи ўзгаради. Шу сабабли ихтиёрий ўрин алмаштириш усулини реализация қилиш учун даставвал $T$ очиқ матн узунлиги олдиндан берилган $n$ га тенг бўлган блокларга бўлинади ва ўрин алмаштиришнинг берилган қоидаларига мос равишда блок белгилари ўрнини алмаштириш орқали очиқ матн блоклар бўйича шифрланади.

Ушбу берилган ўрин алмаштириш қоидалари шифрлаш калитини ташкил этади. Бунда кўриниб турибтики очиқ маълумот белгилари ўзгармайди, балки фақат уларни келиш тартиби ўзгаради.

Масалан, дастлабки очиқ матн

$$T = \text{ШИФР ПЕРЕСТАНОВКИ}$$

кўринишда бўлса, унда узунлиги 4га тенг бўлган ҳар бир блокдаги ҳарфларни 1-нчи, 2-нчи, 3-нчи, 4-нчи тартиб рақамларини мос равишда 2-нчи, 3-нчи, 4-нчи, 1-нчи рақамларга ўрнини алматиришдан кейин (бу ерда калит $K$=2341)

$$E = \text{ИФРШ ЕРЕПТАНСВКИО}$$

шифрматнни ҳосил қиламиз.

Умумий ҳолда, блок узунлиги $n$ –га тенг бўлганда, $T$ очиқ матн шифрлашини $n$-нчи даражали $S$ ўрнига қўйиш орқали амалга ошириш лозим:

$$S = \begin{cases} 1\ 2\ 3 \dots n \\ k_1 k_2 \dots k_n \end{cases}$$

бу ерда $k_i$ берилган ўрин алмаштиришда очиқ матннинг $i$-нчи ҳарфи тушадиган криптограммадаги жой рақами, яъни

$$e_i = \begin{cases} t_{ki}, & agar\ i \le n \\ t_{ki-n}, & agar\ i > n \end{cases}$$

$T$ дастлабки матнни ёзиш ва $E$ ни геометрик фигуралар (уларда ҳар хил маршрутларни кўриб чиқиш мумкин) орқали ўқишни бутунлай уникал бўлган усулларини ўйлаб топиш мумкин. Бундай шифр *маршрутли ўрин алмаштириш* деб номланади.

Масалан, фараз қиламиз

$$T = \text{ТОЛЬКО\_ДЛЯ\_СТУДЕНТОВ\_ФПМ\_КУБГУ}$$

очиқ матн берилган бўлсин. Дастлаб уни 5х6 ўлчовли тўғрибурчакли жадвалга қаторлар бўйича қуйидагича ёзиб чиқамиз:

| | | | | | |
|---|---|---|---|---|---|
| Т | О | Л | Ь | К | О |
| _ | Д | Л | Я | _ | С |
| Т | У | Д | Е | Н | Т |
| О | В | _ | Ф | М | М |
| _ | Н | У | У | З | _ |

Сўнг $T$ матнни ушбу жадвалдан бошқа услубда, масалан устунлар бўйича қўйидаги кўринишда ёзиб оламиз:

$E$ =Т_ТО_ОДУВКЛЛД_УЬЯЕФБК_НПГОСТМУ

## Масалалар ва машқлар

15. Узунлиги $d$-га тенг бўлган калитли ўрин алмаштириш шифрининг калтилар фазосини аниқланг.
16. $K$ =13 06 14 калитли ўрин алмаштириш шифридан фойдаланиб берилган очиқ матнни шифрланг.
17. Узунлиги $d$-га тенг бўлган ўрин алмаштириш ва ўрнига қўйиш икки калит ёрдамида хосил бўладиган композицион шифрнинг калитлар фазосини аниқланг.
18. Рубик кубикга асосланган ўрин алмаштириш шифрининг ҳар схемаларини аниқланг.
19. Битта алфавит асосидаги барча ўрин алмаштиришлар ҳар хил шифрларининг тўплами кўпайтиришга нисбатан гуруҳни ташкил этишини исботланг.
20. Узунлиги ва даври $d$-га тенг бўлган иккилик ($P$-лик) циклик кетма-кетликлар миқдорини аниқланг.

21. *T* дастлабки матн статистик хоссаларини ўзгартириш учун полиалфавит ўрин алмаштириш усулидан фойдаланамиз, бунда калитлар камида иккита ҳар хил ўрин алмаштиришлар ва алфавитлар ёрдамида аниқланади.
Фараз қиламиз *E* шифрматн мос ўрин алмаштиришларни кўпайтмасини натижаси сифатида ёзилади. Аслида шу тариқа узунлиги шифрланадиган матнга тенг тенг бўлган калитга эга бўламиз.Бундай калитни “чексиз” деб номлаймиз. Ушбу шифр қандай устунлик ва камчиликларга эга бўлади?
22. Фараз қиламиз

$$T = \text{ПРИМЕР НАДЕЖНОГО \_ШИФРА}$$

очиқ матнни чексиз калит билан шифрлаш лозим бўлсин. *K* калит сифатида узунлиги шифрланадиган матн узунлигига тенг бўлган ихтиёрий матнни оламиз:

$$\boldsymbol{K} = \text{МАТЕМАТИКА\_ЦАРИЦА\_НАУК}$$

*E* шифрматнни ўзини тузиш учун қуйидаги жадвалдан фойдаланамиз.

| | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Т** | **:** | **П** | **Р** | **И** | **М** | **Е** | **Р** | **_** | **Н** | **А** | **Д** | **Ё** | **Ж** | **Н** | **О** | **Г** | **О** | **_** | **Ш** | **И** | **Ф** | **Р** | **А** |
| | | **16** | **17** | **09** | **13** | **06** | **17** | **33** | **14** | **10** | **05** | **06** | **07** | **14** | **15** | **04** | **15** | **33** | **35** | **09** | **17** | **01** |
| **К** | **:** | **М** | **А** | **Т** | **Е** | **М** | **А** | **Т** | **И** | **К** | **А** | **_** | **Ц** | **А** | **Р** | **И** | **Ц** | **А** | **_** | **Н** | **А** | **У** | **К** |
| | | **10** | **01** | **10** | **01** | **10** | **01** | **10** | **01** | **10** | **01** | **33** | **22** | **01** | **10** | **01** | **02** | **01** | **33** | **10** | **01** | **02** | **10** |
| | **+** | **31** | **19** | **06** | **31** | **19** | **99** | **11** | **33** | **31** | **33** | **17** | **93** | **11** | **33** | **13** | **44** | **10** | **11** | | | | |
| | | **21** | **12** | **19** | **11** | **19** | **18** | **19** | **32** | **11** | **06** | **06** | **03** | **11** | **32** | **13** | **05** | **01** | **25** | **32** | **22** | **04** | **12** |
| | | **29** | **28** | **28** | **19** | **19** | **18** | **19** | **32** | **12** | **06** | **66** | **05** | **52** | **32** | **35** | **01** | **25** | **32** | **22** | **04** | **12** | |
| **Е** | **:** | **Ь** | **С** | **Ы** | **Т** | **Т** | **С** | **Т** | **Ц** | **Л** | **Е** | **Е** | **Э** | **О** | **Я** | **М** | **Д** | **А** | **Ш** | **Ц** | **Х** | **Г** | **А** |

Ушбу шифр криптобардошлигини баҳоланг ва уни очиш учун самарали алгоритм ишлаб чиқинг.
23. Фараз қиламиз *T* очиқ матн ва *K* калит ҳарфлари ASCII-кодда ифодаланган. У ҳолда калти узунлигига боғлиқ бўлмаган равишда $T \oplus K$ қўшиш амалини байт бўйича бажариб *E* шифрматнни ҳосил қиламиз. Бу эса ASCII-кодда ҳам ўрин алмаштириш натижасидир. Ушбу шифр қандай хоссаларга эга?
24. Амалиётда ўрин алмаштириш ва ўрнига қўйиш шифрлари одатда биргаликда ишлатилади. Маълумотлар ҳимояси симметрик тизимларига мансуб бўлган ахборот ҳимояси воситаларига амалий мисоллар келтиринг.
25. Ўрин алмаштириш усулининг қандай хиллари мавжуд?
26. Ўрин алмаштириш ва ўрнига қўйиш иккита шифрлаш усулларини ўзаро таққосланг. Ҳар бирининг устунлик ва камчилик томонларини кўрсатинг.

### *2.2.Блокли шифрлаш усули.*

Амалда кўпинча берилган узунликка эга бўлган блокли шифрлар қўлланилади, яъни элементар хабарлар битта ҳарфдан иборат бўлмасдан, балки иккита(биграмм), учта(триграмм) ва ундан кўп ҳарфлардан таркиб

топган алоҳида блоклардан иборат. Бундай шифрлар учун шифрлаш ва дешифрлаш блоклар бўйича амалга оширилади. n-ҳарфли алфавитдаги к-ҳарфли блок учун уларни сонли эквиваленти сифатида 0-дан $n^k - 1$-гача сегмент сонларини олиш мумкин, бунда ҳар бир блокни *n*- асосли саноқ системасида k-разрядли бутун сон деб қараш мумкин.

Масалан 26-ҳарфли инглиз алифбосидаги *NO* биграммасига $T = \begin{pmatrix} 13 \\ 14 \end{pmatrix}$ матрица мос келади, шифрловчи матрица эса

$$A = \begin{pmatrix} 2 & 3 \\ 7 & 8 \end{pmatrix}$$

кўринишга эга. У ҳолда *E* криптограммани қуйидаги кўринишда аниқлаймиз:

$$E = A*T = \begin{pmatrix} 2 & 3 \\ 7 & 8 \end{pmatrix}\begin{pmatrix} 13 \\ 14 \end{pmatrix} = \begin{pmatrix} 68 \\ 203 \end{pmatrix} = \begin{pmatrix} 16 \\ 21 \end{pmatrix}$$

бу криптограмма *QV* га мос келади. Очиқ *T* матнини тиклаш учун

$$T = A^{-1} * E = \begin{pmatrix} 14 & 11 \\ 17 & 10 \end{pmatrix}\begin{pmatrix} 16 \\ 21 \end{pmatrix} = \begin{pmatrix} 68 \\ 203 \end{pmatrix} = \begin{pmatrix} 13 \\ 14 \end{pmatrix}$$

матрицавий тенгламани ечиш зарур. Агарда *T* очиқ матни *L* –та биграммлардан (уларга *2xL* ўлчовли матрица мос келади) иборат бўлса, у ҳолда *E* криптограмма олдинги ҳолга ўхшаб иккита матрицани $A*T$ кўпайтмаси ва очиқ матн эса

$$T = A^{-1} * E$$

шаклида аниқланади.

Умумий ҳолда очиқ матнни биграммига нисбатан

$$E = A\ *T + B$$

шифрловчи афин алмаштиришни қўллаш мумкин, бу ерда *B* 2x1 ўлчовли устун шаклидаги матрица. Дешифрлаш учун

$$T = A^{-1} * E + A^{-1} * B$$

тенгликдан фойдаланиш мумкин

**Масала ва машқлар**

27. Қуйидаги таққослаш тенгламалари системасини ечинг
$$\begin{cases} 2x + 3y \equiv 1 \bmod 26, \\ 7x + 8y \equiv 2 \bmod 26. \end{cases}$$
28. *n*-ҳарфли алифбо триграммлари (k-граммлари) учун нечта афин шифрловчи алмаштиришлар мавжуд?
29. *GF(P)* майдони устида $N_p(ax+b)$ чизиқли шифрловчи алмаштиришлар миқдорини аниқланг.
30. *GF(P)* майдони устида $N_p(ax+b)/(cx+d)$ каср-чизиқли шифрловчи алмаштиришлар миқдорини аниқланг.

31. *GF(P)* майдони устида $N_p(A)$ 2-нчи тартибли шифрловчи матрицалар миқдорини аниқланг.

32. *GF(P)* майдони устида $N_p(A*T+)$ чизиқли шифрловчи алмаштиришлар миқдорини аниқланг, агарда матн *n*-ҳарфли алифбо триграммларига (k-граммлари) бўлинган бўлса.

33. 26 ҳарфли *A-Z* алифбо (0-25 сонли эквиваленти) триграммларининг чизиқли шифрловчи алмаштириши ёрдамида шифрланган $T = FBRTLWUGAJQINZTHHXTERHBNXSW$ -хабарни дешифрланг. Охирги учта триграммлар – бу юборувчи *JAMESBOND*нинг имзоси эканлиги маълум. Дешифрловчи матрицани топинг ва хабарни ўқинг.

34. Фараз қиламиз $Y_K$ блокли шифрлаш функцияси, *K*-унинг махфий калити ва

1) $e_I = t_I \oplus Y_K(e_{I-1}), \qquad e_0 = const,$

2) $e_I = t_{I-1} \oplus Y_K(x_I \oplus e_{I-1}),\ t_0 = const,$

бўлсин, бу ерда $\oplus$ модул 2 бўйича қўшиш амали.

$$T = \text{БЛОЧНЫЙ\_ШИФР}$$

матнини шифрланг. Ушбу шифр қандай хоссаларга эга?

35. **(Иккиланган квадрат шифри)** Алифболар тасодифий жойлаштирилган иккита жадвал берилган бўлсин (бу ерда матрицаларнинг ўзи калит ролини ўйнайди). Очиқ матннинг ҳар бир жуфтлигига берилган жадвалларнинг ҳарфлар биграммларини мос қўйямиз. Ушбу шифр қандай хоссаларга эга эканлигини аниқланг.

36. Ўзаро бир қийматли шифрловчи акслантиришларнинг бир нечта оддий ҳолларини кўриб чиқамиз.

1. Ҳар бир *ab* биграммга [0, $n^2-1$] сегментдаги битта сонни мос қўямиз.

2. Ҳар бир *ab* биграммга $ab_n = a*n+b$ битта сонни (– n асосли саноқ ситемасидаги сон) мос қўямиз.

3. Ҳар бир *ab* биграммга $n^2$ модул бўйича битта сонни мос қўямиз.

4. Ҳар бир *ab* биграммга $n$ модул бўйича бир жуфт сонни ёки $\begin{pmatrix} a \\ b \end{pmatrix}$ устунли матрицани мос қўямиз.

Қандай шифрловчи акслантиришда мос шифр бардошлиги қониқарлироқ бўлади?

37. *n* ҳарфли алифболи *T* очиқ матннинг ҳар бир $a_1a_2a_3\cdots a_k$ - *k*-ҳарфли блокига

$(a_1a_2a_3\cdots a_k)_n = a_1n^{k-1} + a_2n^{k-2} + a_3n^{k-3} + \cdots + a_k$

сонни мос қўямиз.

Е криптограмма бўйича очиқ матнни тиклаш самарали алгоритмини ишлаб чиқинг.

### *2.3.Гаммалаштириш усули*.

Шифр гаммаси деганда очиқ маълумотларни шифрлаш ва шифрланган маълумотни дешифрлаш учун берилган алгоритм асосида ишлаб чиқиладиган псевдотасодифий кетма-кетлик тушунилади.

Гаммалаштириш усули билан шифрлаш – бу муайян қоида асосида шифр гаммасини очиқ матн билна қўшишдир.

3. *Псевдотасодифий кетма-кетлик* (ПКК) деб бирор алгоритм ёрдамида яратилган ҳар қандай кетма-кетликка айтилади.

Масалан

$$a_n = [\pi 10^n] - [\pi 10^{n-1}]$$

| $n$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | … |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $a_n$ | 3 | 1 | 4 | 1 | 5 | 9 | 2 | 6 | 5 | 3 | 5 | 8 | 9 | 7 | 9 | 3 | 2 | 3 | … |

тенглик ёрдамида $a_n$ кетма-кетликни аниқлаймиз.

Яхши конгурэнт генераторлардан бири чизиқли конгурэнтли датчик (ЧКД) ҳисобланади. У

$$X_{i+1} = (A * X_i + C) \bmod m$$

муносабат билана ниқланган $X_i$-псевдотасодифий сонлар кетма-кетлигини ишлаб чиқади. Бу ерда *A,C* – ўзгармаслар, *X(0)* – дастлабки бошланғич катталик. Ушбу *A,C*, *X(0)* учта катталиклар ахборот ҳимояси воситаси учун калитни ташкил этади.

Масалан агарда *A=C= X(0)*=7, *m*=10 бўлса, келтирилган муносабатдан узунлиги 4 бўлган қуйидаги даврий кетма-кетликни ҳосил қиламиз:
7,6,9,0,7,6,9,0,…….

Очиқ маълумотни гаммалаштириш усули билан шифрлаш учун очиқ маълумотни тасодифий сонлар генератори ёрдамида ҳосил қилинган шифр гаммаси билан қўшиш (масалан XOR амали ёрдамида) лозим.

Криптограммани дешифрлаш жараёни маълум бўлган калит ёрдамида шифр гаммасини такрорий генерация қилиб шифрланган маълумот билан қўшишга олиб келади (масалан, конгурэнт генераторлар ҳолатида калит *A,C*, *X(0)* лардан иборат ).

Амалда агар гамма даври шифрланган матн узунлигидан ошиб кетса ва очиқ матннинг ҳеч қандай қисми маълум бўлмаса, унда шифрни калитни тўғридан-тўғри «перебор» қилиш орқали очиш мумкин.

**Масала ва машқлар**

1. Фараз қиламиз псевдотасодифий кетма-кетлик (ПКК)

$$F_{n+2} = F_{n+1} + F_n (\bmod m)$$

Фибоначчи тенглиги орқали аниқланган бўлсин.

$m = p (m = p^k)$ бўлганда Фибоначчи кетма-кетлиги даврини аниқланг.

2. Фараз қиламиз псевдотасодифий кетма-кетлик (ПКК)

$$X_{n+2} = (A * X_n^2 + B * X_n + C) \bmod m$$

чизиқсиз тенглик орқали аниқланган бўлсин.

*A,B,C* коэффициентлар ва модул *m* – *GF(P)* майдон элементлари бўлганда ушбу кетма-кетлик даврини аниқланг.

3. Ҳужум натижаси дастлабки матн ва криптограммани фрагменти бўлган ҳолат мисолида шифрлаш усули камчиликларини келтиринг.

4. Фараз қиламиз псевдотасодифий кетма-кетлик (ПКК)

$$X_{n+1} = (A * X_n + B * X_{n-1} + C) \bmod p$$

тенглик орқали аниқланган бўлсин.

Берилган $X_0, X_1, A, B, C$ бутун сонлар учун ушбу ПКК энг катта даврини аниқланг.

5. $S_1, S_2, \ldots, S_L$ сонлар кетма-кетлигини ҳосил қилувчи

$S_1 = AS_0 + C (\bmod m)$,

$S_2 = AS_1 + C (\bmod m)$,

.....

$S_L = AS_{L-1} + C (\bmod m)$

алгоритмни кўриб чиқамиз. $S_0$ қиймат маълум бўлганда *L*-ни аниқланг.

6. Гаммалаштириш усули билан қурилган криптотизимга ҳужум алгоритмини ишлаб чиқинг.

7. Фараз қиламиз ПТС чизиқли конгурэнтли датчики

$$X_{i+1} = (A * X_i + C \bmod m)$$

муносабат билан тавсифланади. Одатда *m*-нинг қиймати $2^n$ ёки $2^{n-1}$ га тенг қилиб ўрнатилади, бу ерда *n* – машина сўзининг битлардаги узунлиги. Бунда *A* ва *C* сонларни шундай танлаш лозимки, кетма-кетлик даври максимал бўлсин.

*C* тоқ ва $A \bmod 4 = 1$ бўлгандагина ПТС чизиқли конгурэнтли датчики даври максимал *m* узунликка эга бўлишини исботланг.

8. Коэффициентлари $GF(p)$ майдондан олинган

$$f(x) = x^k - a_1 x^{k-1} - a_2 x^{k-2} - \cdots - a_k$$

нормалланган кўпҳад примитив бўлгандагина

$$X_n = (a_1 X_{n-1} + a_2 X_{n-2} + \cdots + a_k X_{n-k}) \bmod p$$

тенглик орқали аниқланган тасодифий сонлар кетма-кетлиги $GF(p^k)$ чекли майдонда узунлиги $p^k - 1$ бўлган даврга эга бўлишини исботланг.

9. Ҳар қандай $GF(p^k)$ майдонда ихтиёрий мусбат даражали примитив кўпҳад мавжуд бўлишини исботланг.

10. $GF(p^k)$ майдоннинг примитив элементлари сонини аниқланг.

11. $GF(p)$ майдон устида $\alpha$ элементнинг минимал кўпҳади деб коэффициентлари $GF(p)$ дан бўлган $M(\alpha) = 0$ шартни қаноатлантирувчи мумкин бўлган энг кичик даражали $M(x)$ нормалланган кўпҳадга айтилади.

$M(x)$ кўпҳади қандай хоссаларга эга?

12. $p=2$ бўлган ҳолда кафолатли тарзда олдиндан берилган узунликка эга бўлган тасодифий сонлар кетма-кетлигини қуриш мумкинлиги исботланг.
13. Артин гипотезасини исботланг ёки рад этинг: 2 элементи $GF(p)$ даги чексиз содда майдонлар учун примитив бўлади.

### 3. DES-АҚШ криптоҳимоя стандарти(*3 амалий машғулот*)

Маълумотларни шифрлаш симметрик криптизимлардан энг кенг тарқалган ва маълум бўлган – DES (Data Encryption Standart, 1977й.) стандарти - АҚШ федерал стандарти ҳисобланади. DES-алгоритми махсус электрон қурилмалар учун маълумотларни шифрлаш ва дешифрлаш учун ишлаб чиқилган. DES-алгоритмига биноан шифрлаш ва дешифрлаш ҳар бирининг узунлиги 48 битдан иборат $K=\{k_1,k_2,\dots,k_{16}\}$ битта калитлар тизими ёрдамида амлага оширилади.

Шифрлаш жараёни 64-битли $T$ кириш блокини бошланғич IP ўрин алмаштиришларидан, 16 цикл шифрлаш ва охирида $IP^{-1}$ ўрин алмаштиришлардан иборат. Ушбу алгоритм мукаммал тафсилотини маълум манбалардан топиш мумкин. Бу ерда биз фақат алгоритмнинг блок схемасини келтирамиз (3-нчи расмга қаранг).

Қурилган $L_{n-1}$ ва $R_{n-1}$ $(1\le n\le 16)$ эга бўлиб учун $L_n$ ва $R_n$ қуйидагича аниқлаймиз

$$L_n=R_{n-1},\quad R_n=L_{n-1}\oplus f(R_{n-1},K_n)$$

бу ерда $\oplus$ 2 модул бўйича битма-бит қўшишни англатади, $f$ –шифрлаш функцияси.

Дешифрлаш тескари тартибда амалга оширилади. Ҳақиқатан ҳам бу жараён жуда содда. Кўрсатилган тенгликларни

$$R_{n-1}=L_n,\quad L_{n-1}=R_n\oplus f(L_n,K_n)$$

кўринишда ёзиш мумкин ва шундай қилиб, $L_{16}$ ва $R_{16}$ дан бошлаб $L_0$ ва $R_0$ гача пастга тушиш мумкин, унда дешифрлаш жараёни аён бўлиб қолади.

DES-алгоритми тез ишлаши ва махфийлик нуқтаи назаридан керакли хусусиятга эга эканлигини эътироф этамиз: дастлабки хабар ёки калитни озгина ўзгариш криптоматннинг катта ўзгаришига олиб келади. Бундан ташқари криптотаҳлил кўпсонли чизиқсиз тенгламалар системасига олиб келади, яъни бунда вужудга келадиган масалалар камида $NP$ –тўлиқ масалалар бўлади. Яна шуни таъкидлаш жоизки АҚШ янги стандарти AES ҳисобланади.

Энди мазкур алгоритмнинг жава дастурлаш тилидаги дастурини кўрамиз.

```
import java.io.UnsupportedEncodingException;
import java.security.spec.AlgorithmParameterSpec;
import java.security.spec.KeySpec;
import javax.crypto.Cipher;
import javax.crypto.IllegalBlockSizeException;
```

```
import javax.crypto.SecretKey;
import javax.crypto.SecretKeyFactory;
import javax.crypto.spec.PBEKeySpec;
import javax.crypto.spec.PBEParameterSpec;
import java.io.*;
import java.security.KeyPair;
import java.security.KeyPairGenerator;
import java.security.NoSuchAlgorithmException;
import java.security.PrivateKey;
import java.security.PublicKey;
import javax.crypto.Cipher;
import java.io.IOException;

/**
 * @author ginanjar
 *
 * To change the template for this generated type comment go to
 * Window - Preferences - Java - Code Generation - Code and Comments
 */
public class DesEncrypter {
    Cipher ecipher;
    Cipher dcipher;

    // 8-byte Salt
    byte[] salt = {
        (byte)0xA9, (byte)0x9B, (byte)0xC8, (byte)0x32,
        (byte)0x56, (byte)0x35, (byte)0xE3, (byte)0x03
    };

    // Iteration count
    int iterationCount = 19;
    public static final DesEncrypter NAGASAKTI = new DesEncrypter("NagaSakti");

    private DesEncrypter(String passPhrase) {
        try {
            // Create the key
            KeySpec keySpec = new PBEKeySpec(passPhrase.toCharArray(), salt,
iterationCount);
            SecretKey key = SecretKeyFactory.getInstance(
                "PBEWithMD5AndDES").generateSecret(keySpec);
            ecipher = Cipher.getInstance(key.getAlgorithm());
            dcipher = Cipher.getInstance(key.getAlgorithm());

            // Prepare the parameter to the ciphers
            AlgorithmParameterSpec paramSpec = new PBEParameterSpec(salt,
iterationCount);

            // Create the ciphers
            ecipher.init(Cipher.ENCRYPT_MODE, key, paramSpec);
            dcipher.init(Cipher.DECRYPT_MODE, key, paramSpec);
        } catch (java.security.InvalidAlgorithmParameterException e) {
        } catch (java.security.spec.InvalidKeySpecException e) {
        } catch (javax.crypto.NoSuchPaddingException e) {
        } catch (java.security.NoSuchAlgorithmException e) {
        } catch (java.security.InvalidKeyException e) {
        }
    }

    public String encrypt(String str) {
        try {
            // Encode the string into bytes using utf-8
            byte[] utf8 = str.getBytes("UTF8");
```

```
            // Encrypt
            byte[] enc = ecipher.doFinal(utf8);

            // Encode bytes to base64 to get a string
            return new sun.misc.BASE64Encoder().encode(enc);
        } catch (javax.crypto.BadPaddingException e) {
        } catch (IllegalBlockSizeException e) {
        } catch (UnsupportedEncodingException e) {
        }
        return null;
    }

    public String decrypt(String str) {
        try {
            // Decode base64 to get bytes
            byte[] dec = new sun.misc.BASE64Decoder().decodeBuffer(str);

            // Decrypt
            byte[] utf8 = dcipher.doFinal(dec);

            // Decode using utf-8
            return new String(utf8, "UTF8");
        } catch (javax.crypto.BadPaddingException e) {
        } catch (IllegalBlockSizeException e) {
        } catch (UnsupportedEncodingException e) {
        } catch (java.io.IOException e) {
        }
        return null;
    }

    public static void main(String args[]) throws Exception{
        String text = "text.txt";
        BufferedReader br1 = new BufferedReader(new InputStreamReader(new
FileInputStream(text)));
        String matn="";
        String words;
        while ((words = br1.readLine()) != null) {
                        matn += words + " ";
        }
        if (matn == "") {
            return ;
        }

        String encrypted = DesEncrypter.NAGASAKTI.encrypt(matn);

        System.out.println(" Shifrlangan malumot --->>" + encrypted);
        String decrypted = DesEncrypter.NAGASAKTI.decrypt(encrypted);
        System.out.println(" Deshifrlangan malumot --->>"+ decrypted);
    }
}
```

## Масала ва машқлар

14. DES – ахборотни ҳимоялаш воситаси учун $K_E$ ва $K_D$ калитларни келтиринг.
15. DES криптобардошлигига $IP$ ва $IP^{-1}$ ўрин алмаштиришлар таъсир қиладими?
16. DES тизимини криптобардошлигини баҳоланг. DES криптотизимининг заиф ва кучли жойлари нималардан иборат?

17. DES криптотизимини реализация қилувчи программани ишлаб чиқинг.
18. DES тизими ёрдамида навбатдаги $T$ блокни шифрлашда унинг битлари устида $IP$ ўрин алмаштиришлар бажарилади:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 58 | 50 | 42 | 34 | 26 | 18 | 10 | 2 |
| 60 | 52 | 44 | 36 | 28 | 20 | 12 | 4 |
| 62 | 54 | 46 | 38 | 30 | 22 | 14 | 6 |
| 64 | 56 | 48 | 40 | 32 | 24 | 16 | 8 |
| 57 | 49 | 41 | 33 | 25 | 17 | 9 | 1 |
| 59 | 51 | 43 | 35 | 27 | 19 | 11 | 3 |
| 61 | 53 | 45 | 37 | 29 | 21 | 13 | 5 |
| 63 | 55 | 47 | 39 | 31 | 23 | 15 | 7 |

Охирги $IP^{-1}$ ўрин алмаштиришни аниқланг.

## 4. ГОСТ 28147-89 Россия криптотизими стандарти(*4 амалий машғулот*)

Криптографик алмаштиришлар алгоритмини батафсил кўриш учун ГОСТ 28147-89га мурожаат қилиш лозим. Биз эса криптографик алмаштиришни асосий принципларини кўриб чиқамиз, чунки улар кўп жиҳатдан DES алгоритмига ўхшашдир. Шифрлаш алгоритмининг схемаси қуйидаги кўринишга эга:

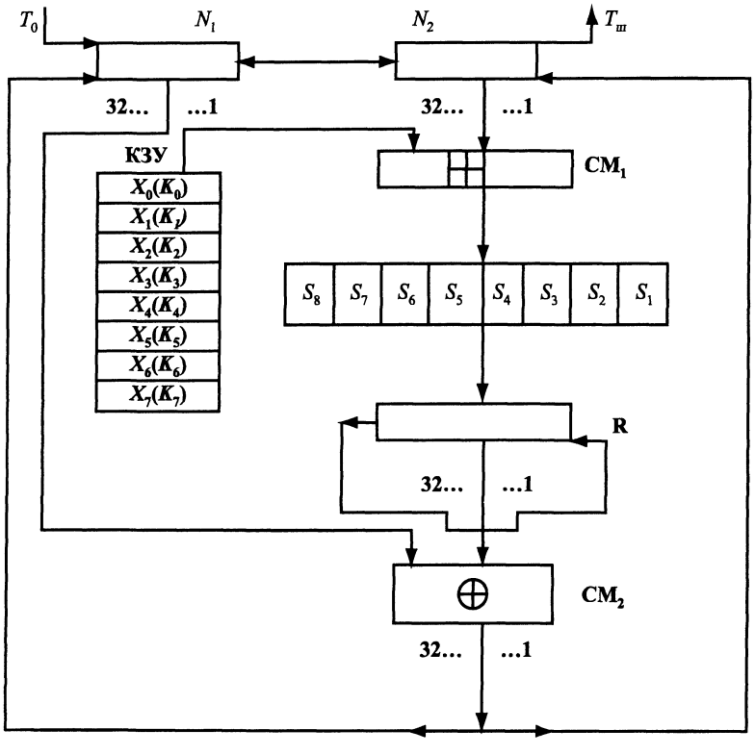


*Рис. 4. Схема алгоритма ГОСТа 28147-89*

Алгоритмни тавсифлашда қуйидаги белгилашлар ишлатилади:

$N_1, N_2$ – 32 разрядли йиғувчилар;

$CM_1$ ва $CM_2$ - иккилик саноқ системасидаги иккита сонларни мос равишда модул $2^{32}$ ва 2 бўйича сумматорлар;

КЭҚ –8та 32 разрядли йиғувчилардан иборат 256 бит ҳажмга эга бўлган калитларни эслаб қолувчи қурилма;

$R$ – 32 разрядли циклик суриш регистри.

Криптографик алмаштириш алгоритми бир нечта ишлаш режимига. Аммо лекин ҳар қандай ҳолда ҳам маълумотларни шифрлаш учун ўлчови 256 бит бўлган калит ишлатилади ва у 8та 32 разрядли $X(i)$ сонлар кўринишида ифодаланади. Агарда калитни $K$ орқали белгиласак, унда

$K = X(7)X(6)X(5)X(4)X(3)X(2)X(1)X(0)$.

Алгоритмни биринчи ва энг содда ишлаш режими – алмаштириш ҳисобланади. Шифрланиши лозим бўлган очиқ маълумотлар ҳар 64 битдан иборат $T(j)$ блокларга бўлинади.

Навбатдаги $T(j)$ битлар кетма-кетлиги ҳар бири 32 битни ўз ичига олган икки қисмктма-кетликларга $B(0)$ (чап ёки катта битлар) ва $A(0)$ (ўнг ёки кичик битлар) бўлинади. Сўнг қуйидаги формулалар билан ифодаланадиган шифрлашни итератив жараёни бажарилади.

1. $\begin{cases} A(I) = f\big(A(I-1)[+]X(j) \oplus B(I-1)\big), \\ B(I) = A(I-1), agar\ I = 1..24, j = (I-1) mod 8 \end{cases}$

2. $\begin{cases} A(I) = f(A(I-1)[+]X(j) \oplus B(I-1)), \\ B(I) = A(I-1), agar\ I = 25, 26, \dots, 31,\ j = 32 - i. \end{cases}$

3. $\begin{cases} A(32) = A(31) \\ B(32) = f(A(31)[+]X(0) \oplus B(31)),\ agar\ \ I = 32 \end{cases}$

бу ерда *i*, *i*=1…32 итерация номерини белгилайди, *f* – шифрлаш функцияси. Унинг аргументи эса олдинги итерация қадамида ҳосил қилинган A(i) ва калитнинг X(j) сонларининг $2^{32}$ модул бўйича йиғиндиси.

Шифрлаш функцияси 32 разрядли йиғинди устида иккита амални ўз ичига олади. Биринчи амал *S* ўрнига қўйиш дейилади.

Иккинчи амал- *S* ўрнига қўйиш натижасида ҳосил қилинган 32 разрядли векторни 11 позицияга чапга циклик суриш.

$T_0$ блокни шифрлаш натижаси $T_ø$ бўлиб қуйидаги кўринишда ифодаланади.

$$T_ø = A(32)B(32).$$

Навбатдаги шифрлаш режими-гаммалаштириш режими ҳисобланади. 64 разрядли *T(i)*, (*i=1…m*, бу ерда *m* шифрланадиган маълумотлар ҳажми билан аниқланади) блокларга бўлинган очиқ маълумотлар 2 модул бўйича 64 блоклар шаклида ишлаб чиқиладиган $\tilde{A}_ø$ шифр гаммаси билан разрядма-разряд қўшиш ёрдамида гаммалаштириш режимида шифрланади. Бу ерда

$$Гш = (Г(1), Г(2), \ldots, Г(г), \ldots, Г(т)).$$

Маълумотларни шифрлаш тенгламаси гаммалаштириш режимида қуйидаги кўринишда ифодаланиши мумкин:

$$Ш(I) = A(Y(I — 1)[+]C2$$
$$Z(I - 1)\{+\}C1 \oplus T(1) = T(1) \oplus T(Г)$$
$$Г_{ш} = (Г(1), Г(2),...,Г(I),...,Г(m))$$

ГОСТ 28147-89 да барча режимлар учун бир хил бўлган имитовставка ишлаб чиқиш жараёни аниқланади. Калит узунлиги етарлича катта бўлганлиги сабабли унинг бардошлиги DES алгоритмига қараганда анча юқори ҳисобланади.

ГОСТ 28147-89 алгоритмининг дастури

```
(function (root, factory) {


    if (typeof define === 'function' && define.amd) {
        define(['gostCrypto'], factory);
    } else if (typeof exports === 'object') {
        module.exports = factory(require('gostCrypto'));
    } else {
        root.GostCoding = factory(root.gostCrypto);
    }


}(this, function (gostCrypto) {


    var root = this;
    var DataError = root.DataError || root.Error;
    var CryptoOperationData = root.ArrayBuffer;
    var Date = root.Date;

    function buffer(d) {
        if (d instanceof CryptoOperationData)
            return d;
        else if (d && d.buffer && d.buffer instanceof CryptoOperationData)
            return d.byteOffset === 0 && d.byteLength === d.buffer.byteLength ?
                    d.buffer : new Uint8Array(new Uint8Array(d, d.byteOffset,
d.byteLength)).buffer;
        else
            throw new DataError('CryptoOperationData required');
    }

    function GostCoding() {
    }


    var Base64 = {

        decode: function (s) {
            s = s.replace(/[^A-Za-z0-9\+\/]/g, '');
            var n = s.length,
                    k = n * 3 + 1 >> 2, r = new Uint8Array(k);

            for (var m3, m4, u24 = 0, j = 0, i = 0; i < n; i++) {
                m4 = i & 3;
                var c = s.charCodeAt(i);
```

```
                c = c > 64 && c < 91 ?
                        c - 65 : c > 96 && c < 123 ?
                        c - 71 : c > 47 && c < 58 ?
                        c + 4 : c === 43 ?
                        62 : c === 47 ?
                        63 : 0;

                u24 |= c << 18 - 6 * m4;
                if (m4 === 3 || n - i === 1) {
                    for (m3 = 0; m3 < 3 && j < k; m3++, j++) {
                        r[j] = u24 >>> (16 >>> m3 & 24) & 255;
                    }
                    u24 = 0;

                }
            }
            return r.buffer;
        },

        encode: function (data) {
            var slen = 8, d = new Uint8Array(buffer(data));
            var m3 = 2, s = '';
            for (var n = d.length, u24 = 0, i = 0; i < n; i++) {
                m3 = i % 3;
                if (i > 0 && (i * 4 / 3) % (12 * slen) === 0)
                    s += '\r\n';
                u24 |= d[i] << (16 >>> m3 & 24);
                if (m3 === 2 || n - i === 1) {
                    for (var j = 18; j >= 0; j -= 6) {
                        var c = u24 >>> j & 63;
                        c = c < 26 ? c + 65 : c < 52 ? c + 71 : c < 62 ? c - 4 :
                                c === 62 ? 43 : c === 63 ? 47 : 65;
                        s += String.fromCharCode(c);
                    }
                    u24 = 0;
                }
            }
            return s.substr(0, s.length - 2 + m3) + (m3 === 2 ? '' : m3 === 1 ? '=' :
'==');
        }
    };


    GostCoding.prototype.Base64 = Base64;

    var Chars = (function () {

        var _win1251_ = {
            0x402: 0x80, 0x403: 0x81, 0x201A: 0x82, 0x453: 0x83, 0x201E: 0x84, 0x2026:
0x85, 0x2020: 0x86, 0x2021: 0x87,
            0x20AC: 0x88, 0x2030: 0x89, 0x409: 0x8A, 0x2039: 0x8B, 0x40A: 0x8C, 0x40C:
0x8D, 0x40B: 0x8E, 0x40f: 0x8f,
            0x452: 0x90, 0x2018: 0x91, 0x2019: 0x92, 0x201C: 0x93, 0x201D: 0x94, 0x2022:
0x95, 0x2013: 0x96, 0x2014: 0x97,
            0x2122: 0x99, 0x459: 0x9A, 0x203A: 0x9B, 0x45A: 0x9C, 0x45C: 0x9D, 0x45B:
0x9E, 0x45f: 0x9f,
            0xA0: 0xA0, 0x40E: 0xA1, 0x45E: 0xA2, 0x408: 0xA3, 0xA4: 0xA4, 0x490: 0xA5,
0xA6: 0xA6, 0xA7: 0xA7,
            0x401: 0xA8, 0xA9: 0xA9, 0x404: 0xAA, 0xAB: 0xAB, 0xAC: 0xAC, 0xAD: 0xAD,
0xAE: 0xAE, 0x407: 0xAf,
            0xB0: 0xB0, 0xB1: 0xB1, 0x406: 0xB2, 0x456: 0xB3, 0x491: 0xB4, 0xB5: 0xB5,
0xB6: 0xB6, 0xB7: 0xB7,
```

```
            0x451: 0xB8, 0x2116: 0xB9, 0x454: 0xBA, 0xBB: 0xBB, 0x458: 0xBC, 0x405: 0xBD,
0x455: 0xBE, 0x457: 0xBf
        };
        var _win1251back_ = {};
        for (var from in _win1251_) {
            var to = _win1251_[from];
            _win1251back_[to] = from;
        }

        return {

            decode: function (s, charset) {
                charset = (charset || 'win1251').toLowerCase().replace('-', '');
                var r = [];
                for (var i = 0, j = s.length; i < j; i++) {
                    var c = s.charCodeAt(i);
                    if (charset === 'utf8') {
                        if (c < 0x80) {
                            r.push(c);
                        } else if (c < 0x800) {
                            r.push(0xc0 + (c >>> 6));
                            r.push(0x80 + (c & 63));
                        } else if (c < 0x10000) {
                            r.push(0xe0 + (c >>> 12));
                            r.push(0x80 + (c >>> 6 & 63));
                            r.push(0x80 + (c & 63));
                        } else if (c < 0x200000) {
                            r.push(0xf0 + (c >>> 18));
                            r.push(0x80 + (c >>> 12 & 63));
                            r.push(0x80 + (c >>> 6 & 63));
                            r.push(0x80 + (c & 63));
                        } else if (c < 0x4000000) {
                            r.push(0xf8 + (c >>> 24));
                            r.push(0x80 + (c >>> 18 & 63));
                            r.push(0x80 + (c >>> 12 & 63));
                            r.push(0x80 + (c >>> 6 & 63));
                            r.push(0x80 + (c & 63));
                        } else {
                            r.push(0xfc + (c >>> 30));
                            r.push(0x80 + (c >>> 24 & 63));
                            r.push(0x80 + (c >>> 18 & 63));
                            r.push(0x80 + (c >>> 12 & 63));
                            r.push(0x80 + (c >>> 6 & 63));
                            r.push(0x80 + (c & 63));
                        }
                    } else if (charset === 'unicode' || charset === 'ucs2' || charset ===
'utf16') {
                        if (c < 0xD800 || (c >= 0xE000 && c <= 0x10000)) {
                            r.push(c >>> 8);
                            r.push(c & 0xff);
                        } else if (c >= 0x10000 && c < 0x110000) {
                            c -= 0x10000;
                            var first = ((0xffc00 & c) >> 10) + 0xD800;
                            var second = (0x3ff & c) + 0xDC00;
                            r.push(first >>> 8);
                            r.push(first & 0xff);
                            r.push(second >>> 8);
                            r.push(second & 0xff);
                        }
                    } else if (charset === 'utf32' || charset === 'ucs4') {
                        r.push(c >>> 24 & 0xff);
                        r.push(c >>> 16 & 0xff);
                        r.push(c >>> 8 & 0xff);
```

```
                        r.push(c & 0xff);
                    } else if (charset === 'win1251') {
                        if (c >= 0x80) {
                            if (c >= 0x410 && c < 0x450) // A..Яа..я
                                c -= 0x350;
                            else
                                c = _win1251_[c] || 0;
                        }
                        r.push(c);
                    } else
                        r.push(c & 0xff);
                }
                return new Uint8Array(r).buffer;
            },

            encode: function (data, charset) {
                charset = (charset || 'win1251').toLowerCase().replace('-', '');
                var r = [], d = new Uint8Array(buffer(data));
                for (var i = 0, n = d.length; i < n; i++) {
                    var c = d[i];
                    if (charset === 'utf8') {
                        c = c >= 0xfc && c < 0xfe && i + 5 < n ? // six bytes
                                (c - 0xfc) * 1073741824 + (d[++i] - 0x80 << 24) + (d[++i]
- 0x80 << 18) + (d[++i] - 0x80 << 12) + (d[++i] - 0x80 << 6) + d[++i] - 0x80
                                : c >> 0xf8 && c < 0xfc && i + 4 < n ? // five bytes
                                (c - 0xf8 << 24) + (d[++i] - 0x80 << 18) + (d[++i] - 0x80
<< 12) + (d[++i] - 0x80 << 6) + d[++i] - 0x80
                                : c >> 0xf0 && c < 0xf8 && i + 3 < n ? // four bytes
                                (c - 0xf0 << 18) + (d[++i] - 0x80 << 12) + (d[++i] - 0x80
<< 6) + d[++i] - 0x80
                                : c >= 0xe0 && c < 0xf0 && i + 2 < n ? // three bytes
                                (c - 0xe0 << 12) + (d[++i] - 0x80 << 6) + d[++i] - 0x80
                                : c >= 0xc0 && c < 0xe0 && i + 1 < n ? // two bytes
                                (c - 0xc0 << 6) + d[++i] - 0x80
                                : c; // one byte
                    } else if (charset === 'unicode' || charset === 'ucs2' || charset ===
'utf16') {
                        c = (c << 8) + d[++i];
                        if (c >= 0xD800 && c < 0xE000) {
                            var first = (c - 0xD800) << 10;
                            c = d[++i];
                            c = (c << 8) + d[++i];
                            var second = c - 0xDC00;
                            c = first + second + 0x10000;
                        }
                    } else if (charset === 'utf32' || charset === 'ucs4') {
                        c = (c << 8) + d[++i];
                        c = (c << 8) + d[++i];
                        c = (c << 8) + d[++i];
                    } else if (charset === 'win1251') {
                        if (c >= 0x80) {
                            if (c >= 0xC0 && c < 0x100)
                                c += 0x350; // A..Яа..я
                            else
                                c = _win1251back_[c] || 0;
                        }
                    }
                    r.push(String.fromCharCode(c));
                }
                return r.join('');
            }
        };
    })();
```

```
GostCoding.prototype.Chars = Chars;


var Hex = {

    decode: function (s, endean) {
        s = s.replace(/[^A-fa-f0-9]/g, '');
        var n = Math.ceil(s.length / 2), r = new Uint8Array(n);
        s = (s.length % 2 > 0 ? '0' : '') + s;
        if (endean && ((typeof endean !== 'string') ||
                (endean.toLowerCase().indexOf('little') < 0)))
            for (var i = 0; i < n; i++)
                r[i] = parseInt(s.substr((n - i - 1) * 2, 2), 16);
        else
            for (var i = 0; i < n; i++)
                r[i] = parseInt(s.substr(i * 2, 2), 16);
        return r.buffer;
    },

    encode: function (data, endean) {
        var s = [], d = new Uint8Array(buffer(data)), n = d.length;
        if (endean && ((typeof endean !== 'string') ||
                (endean.toLowerCase().indexOf('little') < 0)))
            for (var i = 0; i < n; i++) {
                var j = n - i - 1;
                s[j] = (j > 0 && j % 32 === 0 ? '\r\n' : '') +
                        ('00' + d[i].toString(16)).slice(-2);
            }
        else
            for (var i = 0; i < n; i++)
                s[i] = (i > 0 && i % 32 === 0 ? '\r\n' : '') +
                        ('00' + d[i].toString(16)).slice(-2);
        return s.join('');
    }
};


GostCoding.prototype.Hex = Hex;


var Int16 = {

    decode: function (s) {
        s = (s || '').replace(/[^\-A-fa-f0-9]/g, '');
        if (s.length === 0)
            s = '0';

        var neg = false;
        if (s.charAt(0) === '-') {
            neg = true;
            s = s.substring(1);
        }

        while (s.charAt(0) === '0' && s.length > 1)
            s = s.substring(1);
        s = (s.length % 2 > 0 ? '0' : '') + s;

        if ((!neg && !/^[0-7]/.test(s)) ||
                (neg && !/^[0-7]|8[0]+$/.test(s)))
            s = '00' + s;
```

```
        var n = s.length / 2, r = new Uint8Array(n), t = 0;
        for (var i = n - 1; i >= 0; --i) {
            var c = parseInt(s.substr(i * 2, 2), 16);
            if (neg && (c + t > 0)) {
                c = 256 - c - t;
                t = 1;
            }
            r[i] = c;
        }
        return r.buffer;
    },

    encode: function (data) {
        var d = new Uint8Array(buffer(data)), n = d.length;
        if (d.length === 0)
            return '0x00';
        var s = [], neg = d[0] > 0x7f, t = 0;
        for (var i = n - 1; i >= 0; --i) {
            var v = d[i];
            if (neg && (v + t > 0)) {
                v = 256 - v - t;
                t = 1;
            }
            s[i] = ('00' + v.toString(16)).slice(-2);
        }
        s = s.join('');
        while (s.charAt(0) === '0')
            s = s.substring(1);
        return (neg ? '-' : '') + '0x' + s;
    }
};


GostCoding.prototype.Int16 = Int16;

var BER = (function () {

    function encodeBER(source, format, onlyContent) {
        // Correct primitive type
        var object = source.object;
        if (object === undefined)
            object = source;

        // Determinate tagClass
        var tagClass = source.tagClass = source.tagClass || 0; // Universial default

        // Determinate tagNumber. Use only for Universal class
        if (tagClass === 0) {
            var tagNumber = source.tagNumber;
            if (typeof tagNumber === 'undefined') {
                if (typeof object === 'string') {
                    if (object === '')   // NULL
                        tagNumber = 0x05;
                    else if (/^\-?0x[0-9a-fA-F]+$/.test(object)) // INTEGER
                        tagNumber = 0x02;
                    else if (/^(\d+\.)+\d+$/.test(object)) // OID
                        tagNumber = 0x06;
                    else if (/^[01]+$/.test(object)) // BIT STRING
                        tagNumber = 0x03;
                    else if (/^(true|false)$/.test(object)) // BOOLEAN
                        tagNumber = 0x01;
                    else if (/^[0-9a-fA-F]+$/.test(object)) // OCTET STRING
                        tagNumber = 0x04;
```

```
                        else
                            tagNumber = 0x13; // Printable string (later can be changed
to UTF8String)
                    } else if (typeof object === 'number') { // INTEGER
                        tagNumber = 0x02;
                    } else if (typeof object === 'boolean') { // BOOLEAN
                        tagNumber = 0x01;
                    } else if (object instanceof Array) { // SEQUENCE
                        tagNumber = 0x10;
                    } else if (object instanceof Date) { // GeneralizedTime
                        tagNumber = 0x18;
                    } else if (object instanceof CryptoOperationData || (object &&
object.buffer instanceof CryptoOperationData)) {
                        tagNumber = 0x04;
                    } else
                        throw new DataError('Unrecognized type for ' + object);
                }
            }

            // Determinate constructed
            var tagConstructed = source.tagConstructed;
            if (typeof tagConstructed === 'undefined')
                tagConstructed = source.tagConstructed = object instanceof Array;

            // Create content
            var content;
            if (object instanceof CryptoOperationData || (object && object.buffer
instanceof CryptoOperationData)) { // Direct
                content = new Uint8Array(buffer(object));
                if (tagNumber === 0x03) { // BITSTRING
                    // Set unused bits
                    var a = new Uint8Array(buffer(content));
                    content = new Uint8Array(a.length + 1);
                    content[0] = 0; // No unused bits
                    content.set(a, 1);
                }
            } else if (tagConstructed) { // Sub items coding
                if (object instanceof Array) {
                    var bytelen = 0, ba = [], offset = 0;
                    for (var i = 0, n = object.length; i < n; i++) {
                        ba[i] = encodeBER(object[i], format);
                        bytelen += ba[i].length;
                    }
                    if (tagNumber === 0x11)
                        ba.sort(function (a, b) { // Sort order for SET components
                            for (var i = 0, n = Math.min(a.length, b.length); i < n; i++)
{
                                var r = a[i] - b[i];
                                if (r !== 0)
                                    return r;
                            }
                            return a.length - b.length;
                        });
                    if (format === 'CER') { // final for CER 00 00
                        ba[n] = new Uint8Array(2);
                        bytelen += 2;
                    }
                    content = new Uint8Array(bytelen);
                    for (var i = 0, n = ba.length; i < n; i++) {
                        content.set(ba[i], offset);
                        offset = offset + ba[i].length;
                    }
                } else
```

```
                    throw new DataError('Constracted block can\'t be primitive');
                } else {
                    switch (tagNumber) {
                        // 0x00: // EOC
                        case 0x01: // BOOLEAN
                            content = new Uint8Array(1);
                            content[0] = object ? 0xff : 0;
                            break;
                        case 0x02: // INTEGER
                        case 0x0a: // ENUMIRATED
                            content = Int16.decode(
                                    typeof object === 'number' ? object.toString(16) :
object);
                            break;
                        case 0x03: // BIT STRING
                            if (typeof object === 'string') {
                                var unusedBits = 7 - (object.length + 7) % 8;
                                var n = Math.ceil(object.length / 8);
                                content = new Uint8Array(n + 1);
                                content[0] = unusedBits;
                                for (var i = 0; i < n; i++) {
                                    var c = 0;
                                    for (var j = 0; j < 8; j++) {
                                        var k = i * 8 + j;
                                        c = (c << 1) + (k < object.length ? (object.charAt(k)
=== '1' ? 1 : 0) : 0);
                                    }
                                    content[i + 1] = c;
                                }
                            }
                            break;
                        case 0x04:
                            content = Hex.decode(
                                    typeof object === 'number' ? object.toString(16) :
object);
                            break;
                            // case 0x05: // NULL
                        case 0x06: // OBJECT IDENTIFIER
                            var a = object.match(/\d+/g), r = [];
                            for (var i = 1; i < a.length; i++) {
                                var n = +a[i], r1 = [];
                                if (i === 1)
                                    n = n + a[0] * 40;
                                do {
                                    r1.push(n & 0x7F);
                                    n = n >>> 7;
                                } while (n);
                                // reverse order
                                for (j = r1.length - 1; j >= 0; --j)
                                    r.push(r1[j] + (j === 0 ? 0x00 : 0x80));
                            }
                            content = new Uint8Array(r);
                            break;
                            // case 0x07: // ObjectDescriptor
                            // case 0x08: // EXTERNAL
                            // case 0x09: // REAL
                            // case 0x0A: // ENUMERATED
                            // case 0x0B: // EMBEDDED PDV
                        case 0x0C: // UTF8String
                            content = Chars.decode(object, 'utf8');
                            break;
                            // case 0x10: // SEQUENCE
                            // case 0x11: // SET
```

```
                    case 0x12: // NumericString
                    case 0x16: // IA5String // ASCII
                    case 0x13: // PrintableString // ASCII subset
                    case 0x14: // TeletexString // aka T61String
                    case 0x15: // VideotexString
                    case 0x19: // GraphicString
                    case 0x1A: // VisibleString // ASCII subset
                    case 0x1B: // GeneralString
                        // Reflect on character encoding
                        for (var i = 0, n = object.length; i < n; i++)
                            if (object.charCodeAt(i) > 255)
                                tagNumber = 0x0C;
                        if (tagNumber === 0x0C)
                            content = Chars.decode(object, 'utf8');
                        else
                            content = Chars.decode(object, 'ascii');
                        break;
                    case 0x17: // UTCTime
                    case 0x18: // GeneralizedTime
                        var result = object.original;
                        if (!result) {
                            var date = new Date(object);
                            date.setMinutes(date.getMinutes() +
date.getTimezoneOffset()); // to UTC
                            var ms = tagNumber === 0x18 ?
date.getMilliseconds().toString() : ''; // Milliseconds, remove trailing zeros
                            while (ms.length > 0 && ms.charAt(ms.length - 1) === '0')
                                ms = ms.substring(0, ms.length - 1);
                            if (ms.length > 0)
                                ms = '.' + ms;
                            result = (tagNumber === 0x17 ?
date.getYear().toString().slice(-2) : date.getFullYear().toString()) +
                                    ('00' + (date.getMonth() + 1)).slice(-2) +
                                    ('00' + date.getDate()).slice(-2) +
                                    ('00' + date.getHours()).slice(-2) +
                                    ('00' + date.getMinutes()).slice(-2) +
                                    ('00' + date.getSeconds()).slice(-2) + ms + 'Z';
                        }
                        content = Chars.decode(result, 'ascii');
                        break;
                    case 0x1C: // UniversalString
                        content = Chars.decode(object, 'utf32');
                        break;
                    case 0x1E: // BMPString
                        content = Chars.decode(object, 'utf16');
                        break;
                }
            }

            if (!content)
                content = new Uint8Array(0);
            if (content instanceof CryptoOperationData)
                content = new Uint8Array(content);

            if (!tagConstructed && format === 'CER') {
                // Encoding CER-form for string types
                var k;
                switch (tagNumber) {
                    case 0x03: // BIT_STRING
                        k = 1; // ingnore unused bit for bit string
                    case 0x04: // OCTET_STRING
                    case 0x0C: // UTF8String
                    case 0x12: // NumericString
```

```
                    case 0x13: // PrintableString
                    case 0x14: // TeletexString
                    case 0x15: // VideotexString
                    case 0x16: // IA5String
                    case 0x19: // GraphicString
                    case 0x1A: // VisibleString
                    case 0x1B: // GeneralString
                    case 0x1C: // UniversalString
                    case 0x1E: // BMPString
                        k = k || 0;
                        // Split content on 1000 octet len parts
                        var size = 1000;
                        var bytelen = 0, ba = [], offset = 0;
                        for (var i = k, n = content.length; i < n; i += size - k) {
                            ba[i] = encodeBER({
                                object: new Uint8Array(content.buffer, i, Math.min(size -
k, n - i)),
                                tagNumber: tagNumber,
                                tagClass: 0,
                                tagConstructed: false
                            }, format);
                            bytelen += ba[i].length;
                        }
                        ba[n] = new Uint8Array(2); // final for CER 00 00
                        bytelen += 2;
                        content = new Uint8Array(bytelen);
                        for (var i = 0, n = ba.length; i < n; i++) {
                            content.set(ba[i], offset);
                            offset = offset + ba[i].length;
                        }
                }
            }

            // Restore tagNumber for all classes
            if (tagClass === 0)
                source.tagNumber = tagNumber;
            else
                source.tagNumber = tagNumber = source.tagNumber || 0;
            source.content = content;

            if (onlyContent)
                return content;

            // Create header
            // tagNumber
            var ha = [], first = tagClass === 3 ? 0xC0 : tagClass === 2 ? 0x80 :
                    tagClass === 1 ? 0x40 : 0x00;
            if (tagConstructed)
                first |= 0x20;
            if (tagNumber < 0x1F) {
                first |= tagNumber & 0x1F;
                ha.push(first);
            } else {
                first |= 0x1F;
                ha.push(first);
                var n = tagNumber, ha1 = [];
                do {
                    ha1.push(n & 0x7F);
                    n = n >>> 7;
                } while (n)
                // reverse order
                for (var j = ha1.length - 1; j >= 0; --j)
                    ha.push(ha1[j] + (j === 0 ? 0x00 : 0x80));
```

```
            }
            // Length
            if (tagConstructed && format === 'CER') {
                ha.push(0x80);
            } else {
                var len = content.length;
                if (len > 0x7F) {
                    var l2 = len, ha2 = [];
                    do {
                        ha2.push(l2 & 0xff);
                        l2 = l2 >>> 8;
                    } while (l2);
                    ha.push(ha2.length + 0x80); // reverse order
                    for (var j = ha2.length - 1; j >= 0; --j)
                        ha.push(ha2[j]);
                } else {
                    // simple len
                    ha.push(len);
                }
            }
            var header = source.header = new Uint8Array(ha);

            // Result - complete buffer
            var block = new Uint8Array(header.length + content.length);
            block.set(header, 0);
            block.set(content, header.length);
            return block;
        }

        function decodeBER(source, offset) {

            // start pos
            var pos = offset || 0, start = pos;
            var tagNumber, tagClass, tagConstructed,
                    content, header, buffer, sub, len;

            if (source.object) {
                // Ready from source
                tagNumber = source.tagNumber;
                tagClass = source.tagClass;
                tagConstructed = source.tagConstructed;
                content = source.content;
                header = source.header;
                buffer = source.object instanceof CryptoOperationData ?
                        new Uint8Array(source.object) : null;
                sub = source.object instanceof Array ? source.object : null;
                len = buffer && buffer.length || null;
            } else {
                // Decode header
                var d = source;

                // Read tag
                var buf = d[pos++];
                tagNumber = buf & 0x1f;
                tagClass = buf >> 6;
                tagConstructed = (buf & 0x20) !== 0;
                if (tagNumber === 0x1f) { // long tag
                    tagNumber = 0;
                    do {
                        if (tagNumber > 0x1fffffffffff80)
                            throw new DataError('Convertor not supported tag number more
then (2^53 - 1) at position ' + offset);
                        buf = d[pos++];
```

```
                    tagNumber = (tagNumber << 7) + (buf & 0x7f);
                } while (buf & 0x80);
            }

            // Read len
            buf = d[pos++];
            len = buf & 0x7f;
            if (len !== buf) {
                if (len > 6) // no reason to use Int10, as it would be a huge buffer
anyways
                    throw new DataError('Length over 48 bits not supported at
position ' + offset);
                if (len === 0)
                    len = null; // undefined
                else {
                    buf = 0;
                    for (var i = 0; i < len; ++i)
                        buf = (buf << 8) + d[pos++];
                    len = buf;
                }
            }

            start = pos;
            sub = null;

            if (tagConstructed) {
                // must have valid content
                sub = [];
                if (len !== null) {
                    // definite length
                    var end = start + len;
                    while (pos < end) {
                        var s = decodeBER(d, pos);
                        sub.push(s);
                        pos += s.header.length + s.content.length;
                    }
                    if (pos !== end)
                        throw new DataError('Content size is not correct for
container starting at offset ' + start);
                } else {
                    // undefined length
                    try {
                        for (; ; ) {
                            var s = decodeBER(d, pos);
                            pos += s.header.length + s.content.length;
                            if (s.tagClass === 0x00 && s.tagNumber === 0x00)
                                break;
                            sub.push(s);
                        }
                        len = pos - start;
                    } catch (e) {
                        throw new DataError('Exception ' + e + ' while decoding
undefined length content at offset ' + start);
                    }
                }
            }

            // Header and content
            header = new Uint8Array(d.buffer, offset, start - offset);
            content = new Uint8Array(d.buffer, start, len);
            buffer = content;
        }
```

```
            // Constructed types - check for string concationation
            if (sub !== null && tagClass === 0) {
                var k;
                switch (tagNumber) {
                    case 0x03: // BIT_STRING
                        k = 1; // ingnore unused bit for bit string
                    case 0x04: // OCTET_STRING
                    case 0x0C: // UTF8String
                    case 0x12: // NumericString
                    case 0x13: // PrintableString
                    case 0x14: // TeletexString
                    case 0x15: // VideotexString
                    case 0x16: // IA5String
                    case 0x19: // GraphicString
                    case 0x1A: // VisibleString
                    case 0x1B: // GeneralString
                    case 0x1C: // UniversalString
                    case 0x1E: // BMPString
                        k = k || 0;
                        // Concatination
                        if (sub.length === 0)
                            throw new DataError('No constructed encoding content of
string type at offset ' + start);
                        len = k;
                        for (var i = 0, n = sub.length; i < n; i++) {
                            var s = sub[i];
                            if (s.tagClass !== tagClass || s.tagNumber !== tagNumber ||
s.tagConstructed)
                                throw new DataError('Invalid constructed encoding of
string type at offset ' + start);
                            len += s.content.length - k;
                        }
                        buffer = new Uint8Array(len);
                        for (var i = 0, n = sub.length, j = k; i < n; i++) {
                            var s = sub[i];
                            if (k > 0)
                                buffer.set(s.content.subarray(1), j);
                            else
                                buffer.set(s.content, j);
                            j += s.content.length - k;
                        }
                        tagConstructed = false; // follow not required
                        sub = null;
                        break;
                }
            }
            // Primitive types
            var object = '';
            if (sub === null) {
                if (len === null)
                    throw new DataError('Invalid tag with undefined length at offset ' +
start);

                if (tagClass === 0) {
                    switch (tagNumber) {
                        case 0x01: // BOOLEAN
                            object = buffer[0] !== 0;
                            break;
                        case 0x02: // INTEGER
                        case 0x0a: // ENUMIRATED
                            if (len > 6) {
                                object = Int16.encode(buffer);
                            } else {
```

```
                        var v = buffer[0];
                        if (buffer[0] > 0x7f)
                            v = v - 256;
                        for (var i = 1; i < len; i++)
                            v = v * 256 + buffer[i];
                        object = v;
                    }
                    break;
                case 0x03: // BIT_STRING
                    if (len > 5) { // Content buffer
                        object = new Uint8Array(buffer.subarray(1)).buffer;
                    } else { // Max bit mask only for 32 bit
                        var unusedBit = buffer[0],
                                skip = unusedBit, s = [];
                        for (var i = len - 1; i >= 1; --i) {
                            var b = buffer[i];
                            for (var j = skip; j < 8; ++j)
                                s.push((b >> j) & 1 ? '1' : '0');
                            skip = 0;
                        }
                        object = s.reverse().join('');
                    }
                    break;
                case 0x04: // OCTET_STRING
                    object = new Uint8Array(buffer).buffer;
                    break;
                    //  case 0x05: // NULL
                case 0x06: // OBJECT_IDENTIFIER
                    var s = '',
                            n = 0,
                            bits = 0;
                    for (var i = 0; i < len; ++i) {
                        var v = buffer[i];
                        n = (n << 7) + (v & 0x7F);
                        bits += 7;
                        if (!(v & 0x80)) { // finished
                            if (s === '') {
                                var m = n < 80 ? n < 40 ? 0 : 1 : 2;
                                s = m + "." + (n - m * 40);
                            } else
                                s += "." + n.toString();
                            n = 0;
                            bits = 0;
                        }
                    }
                    if (bits > 0)
                        throw new DataError('Incompleted OID at offset ' +
start);
                    object = s;
                    break;

                case 0x10: // SEQUENCE
                case 0x11: // SET
                    object = [];
                    break;
                case 0x0C: // UTF8String
                    object = Chars.encode(buffer, 'utf8');
                    break;
                case 0x12: // NumericString
                case 0x13: // PrintableString
                case 0x14: // TeletexString
                case 0x15: // VideotexString
                case 0x16: // IA5String
```

```
                        case 0x19: // GraphicString
                        case 0x1A: // VisibleString
                        case 0x1B: // GeneralString
                            object = Chars.encode(buffer, 'ascii');
                            break;
                        case 0x1C: // UniversalString
                            object = Chars.encode(buffer, 'utf32');
                            break;
                        case 0x1E: // BMPString
                            object = Chars.encode(buffer, 'utf16');
                            break;
                        case 0x17: // UTCTime
                        case 0x18: // GeneralizedTime
                            var shortYear = tagNumber === 0x17;
                            var s = Chars.encode(buffer, 'ascii'),
                                    m = (shortYear ?
                                            /^(\d\d)(0[1-9]|1[0-2])(0[1-
9]|[12]\d|3[01])([01]\d|2[0-3])(?:([0-5]\d)(?:([0-5]\d)(?:[.,](\d{1,3}))?)?)?(Z|[-
+](?:[0]\d|1[0-2])([0-5]\d)?)?$/ :
                                            /^(\d\d\d\d)(0[1-9]|1[0-2])(0[1-
9]|[12]\d|3[01])([01]\d|2[0-3])(?:([0-5]\d)(?:([0-5]\d)(?:[.,](\d{1,3}))?)?)?(Z|[-
+](?:[0]\d|1[0-2])([0-5]\d)?)?$/).exec(s);
                            if (!m)
                                throw new DataError('Unrecognized time format "' + s + '"
at offset ' + start);
                            if (shortYear) {
                                // Where YY is greater than or equal to 50, the year
SHALL be interpreted as 19YY; and
                                // Where YY is less than 50, the year SHALL be
interpreted as 20YY
                                m[1] = +m[1];
                                m[1] += (m[1] < 50) ? 2000 : 1900;
                            }
                            var dt = new Date(m[1], +m[2] - 1, +m[3], +(m[4] || '0'),
+(m[5] || '0'), +(m[6] || '0'), +(m[7] || '0')),
                                    tz = dt.getTimezoneOffset();
                            if (m[8] || tagNumber === 0x17) {
                                if (m[8].toUpperCase() !== 'Z' && m[9]) {
                                    tz = tz + parseInt(m[9]);
                                }
                                dt.setMinutes(dt.getMinutes() - tz);
                            }
                            dt.original = s;
                            object = dt;
                            break;
                    }
                } else // OCTET_STRING
                    object = new Uint8Array(buffer).buffer;
            } else
                object = sub;

            // result
            return {
                tagConstructed: tagConstructed,
                tagClass: tagClass,
                tagNumber: tagNumber,
                header: header,
                content: content,
                object: object
            };
        }

        return {
```

```
            encode: function (object, format, onlyContent) {
                return encodeBER(object, format, onlyContent).buffer;
            },
            decode: function (data) {
                return decodeBER(data.object ? data : new Uint8Array(buffer(data)), 0);
            }
        }; // </editor-fold>
    })();


    GostCoding.prototype.BER = BER;


    var PEM = {// <editor-fold defaultstate="collapsed">

        encode: function (data, name) {
            return (name ? '-----BEGIN ' + name.toUpperCase() + '-----\r\n' : '') +
                    Base64.encode(data instanceof CryptoOperationData ? data :
BER.encode(data)) +
                    (name ? '\r\n-----END ' + name.toUpperCase() + '-----' : '');
        },
        decode: function (s, name, deep, index) {
            // Try clear base64
            var re1 = /([A-Za-z0-9\+\/\s\=]+)/g,
                    valid = re1.exec(s);
            if (valid[1].length !== s.length)
                valid = false;
            if (!valid && name) {
                // Try with the name
                var re2 = new RegExp(
                        '-----\\s?BEGIN ' + name.toUpperCase() +
                        '-----([A-Za-z0-9\\+\\/\\s\\=]+)-----\\s?END ' +
                        name.toUpperCase() + '-----', 'g');
                valid = re2.exec(s);
            }
            if (!valid) {
                // Try with some name
                var re3 = new RegExp(
                        '-----\\s?BEGIN [A-Z0-9\\s]+' +
                        '-----([A-Za-z0-9\\+\\/\\s\\=]+)-----\\s?END ' +
                        '[A-Z0-9\\s]+-----', 'g');
                valid = re3.exec(s);
            }
            var r = valid && valid[1 + (index || 0)];
            if (!r)
                throw new DataError('Not valid PEM format');
            var out = Base64.decode(r);
            if (deep)
                out = BER.decode(out);
            return out;
        } // </editor-fold>
    };


    GostCoding.prototype.PEM = PEM;

    if (gostCrypto)

        gostCrypto.coding = new GostCoding();

    return GostCoding;
```

```
}));
```

## Масала ва машқлар

1. ГОСТ 28147-89 – ахборотни ҳимоялаш воситаси учун $K_E$ ва $K_D$ калитларни келтиринг.
2. ГОСТ 28147-89 алгоритмини биринчи ва энг содда режими – алмаштириш ҳисобланади. Алмаштириш алгоритмини тавсифлаб беринг.
3. ГОСТ 28147-89 тизимини криптобардошлигини баҳоланг. DES криптотизимининг заиф ва кучли жойлари нималардан иборат?
4. ГОСТ 28147-89 криптотизимини реализация қилувчи программани ишлаб чиқинг.
5. ГОСТ 28147-89 ва DES тизимларини қиёсий таҳлил қилинг.
6. Очиқ маълумотларни гаммалаштириш режимида шифрлашни реализация қилиш учун алгоритм ишлаб чиқинг.

# 5. RSA алгоритми(*5 амалий машғулот*)

### *5.1.RSA ҳимоя тизимини сонли варианти*

RSA тизими 1978 йилда ишлаб чиқилган. Ишлаб чиқувчилар томонидан жуда ҳам қийин масала ҳисобланган содда кўпайтувчилари олдиндан маълум бўлмаган катта бутун сонни тўлиқ факторизациялашга асосланган шифрлаш функцияси мисоли таклиф этилди. Бунда қуйидаги факт инобатга олинди: агарда $p$ сони $\varphi(n)$га нисбатан ўзаро туб бўлса, унда шундай $d$ бутун сони топиладики

$$p * d = 1(\mathrm{mod}\varphi(n))$$

бўлади. Ушбу математик фактларга машҳур бўлган RSA алгоритми асосланган.

RSA ҳимоя тизимини сонли вариантини амалда қўллаш учун, даставвал қуйидаги қадамларни бажариб очиқ ва махфий калитларни генерация қилиш лозим:

1. Иккита $p$ ва $q$ жуда катта туб сонларни танланг;
2. $p$ сонини ва $q$-сонига кўпайтирш натижаси сифатида $n$ сонини аниқланг;
3. $p$ ва $\varphi(n)$ орасидан шундай катта $L$ сонини танлангки
$$\big((p-1)*(q-1), L\big) = 1 \; yoki \; (\varphi(n), L) = 1$$
бўлсин
4. Шундай S сонини танлангки
$$L * S = 1\big((p-1)*(q-1)\big), yoki \; S = L^{-1} mod(\varphi(n))$$
муносабат ўринли бўлсин
5. $K_E = (L, n)$ очиқ калит сифатида ва $K_D = (S, n)$ махфий калит сифатида ўрнатинг.

Шундай қилиб RSA ҳимоя тизимини сонли варианти учун қуйидаги шифрлаш функциясига эга бўламиз:

$$F(t) = t^L \bmod(n)$$

ва мос

$$F^{-1}(c) = c^S \bmod(n)$$

дешифрлаш функциясига эга бўламиз.

### *5.2.RSA ҳимоя тизимининг кўпфойдаланувчили варианти*

Соддалик учун $p$ сони учта туб сонлар кўпаймаси бўлган ҳолни кўриб чиқамиз. Масалан:

$$n = P_1 * P_2 * P_3,\ \ m = (P_1 - 1) * (P_2 - 1) * (P_3 - 1)$$

бўлсин.

Олдинги ҳолатда бўлгани каби, битта очиқ калитни $K_E = \{L, n\}$ бўлса ва иккита махфий калит $K_{D1} = \{S_1, n\}$ , $K_{D2} = \{S_2, n\}$ танлаймиз, қуйидаги шарт бажарилади:

$$L * S_1 * S_2 = 1 mod(m)$$

Бундай ҳолатда биз умумий сир $S$ ни иккита қисмга $S_1, S_2$ каби ажратамиз, бу ГИС иштирокчилари бир-бирига ишонмайдиган ҳолларда зарур.

Масалан: $P_1 = 3$ , $P_2 = 7$, $P_3 = 11$ бўлса:

$$n = P_1 * P_2 * P_3 = 231,\ \ m = \phi(231) = 120,$$

Ва очиқ калит сифатида ушбу $K_E = \{13{,}231\}$ ва текширувчи иккита махфий калитлар сифатида $K_{D1} = \{11{,}231\}$ $K_{D2} = \{47{,}231\}$ лар қабул қилинади.

Айтайли

$T$= BAC AB AB

Кирувчи очиқ текс бўлсин,очиқ калит ярдамида шифрлангандан кейинг ҳолат қуйидагича(бунда шифрлаш функсияси $F(t) = t^L mod(n)$ko'rinishida ):

$$F(1) = 1^{13} mod(231) = 1$$

$$F(2) = 2^{13} mod(231) = 107$$

$$F(3) = 3^{13} mod(231) = 192$$

Ва биринчи криптограмма қуйидаги кўринишда:

$$E_1 = 107\ 1\ 192\ 1\ 107\ 1\ 107$$

Энди икки фойдаланувчи ўзининг $K_{D1} = \{11,231\}$ калитидан фойдаланиб $E_1 = 107\ 1\ 192\ 1\ 107\ 1\ 107$ ни қайта ишлайди ва ўзининг криптограммасини яратади.

$$E_2 = 74\ 1\ 159\ 1\ 74\ 1\ 74$$

Бу қандай амалга оширилади:

$$F(1) = 1^{11} mod(231) = 1$$
$$F(107) = 107^{11} mod(231) = 74$$
$$F(192) = 192^{11} mod(231) = 159$$

Натижада $E_2 = 74\ 1\ 159\ 1\ 74\ 1\ 74$ ҳосил қилинади, Учинчи фойдаланувчи сўнгги $K_{D2} = \{47,231\}$ калитдан фойдаланган ҳолда кирувчи текстни қайта тиклаб олади:

$T_S = 2131212$ ёки $T = BAC\ AB\ AB$ ни, қандай қилиб:

$$F(74) = 74^{47} mod(231) = 2$$
$$F(1) = 1^{47} mod(231) = 1$$
$$F(159) = 159^{47} mod(231) = 3$$

Худди шундай усулдан фойдаланиб, иштирокчилар сони учдан ортиқ бўлса, РСА тизимининг янада мураккаб версиясини қўллашингиз мумкин, ва ба'зи кўп алфабетли криптосистемларда ҳарфлар частоталарини камайтириш учун ишлатилади.

***5.3.RSA ҳимоя тизимининг polinominal варианти***.

RSA РСА тизимининг полином версиясини амалда қо'ллаш учун, қуйидаги тарзда, қуйидаги босқичларни бажариб, очиқ ва махфий калитни яратишингиз керак:

1. $GF(P)$ фазодан иккита ката туб $p(x)$ ва $q(x)$ ларни $u$ ва $v$ ларга мос ҳолда танлаб оламиз.

2. $n(x)$ ни $p(x)$ ва $q(x)$ ни бир бирига кўпайтириш орқали аниқлаймиз, яъни: $n(x) = p(x) * q(x)$.
3. Катта $L(x)$ туб сони танланади, яъни қуйидаги шартларни қаноатлантирувчи:

$$\big((p(x) - 1) * (q(x) - 1), L(x)\big) = 1;$$

4. Қуйидаги тенгликни қаноатлантирувчи полином $S(x)$ ни аниқлаймиз:

$$S(x) * L(x) = 1 mod\big((p(x) - 1) * (q(x) - 1)\big);$$

5. Очиқ калит $K_E = \{L(x), n\}$ ва махфий калит $K_D = \{S(x), n\}$ лар аниқлаб олинади. $GF(P)$ полином фазосида туб сонларнинг аниқ мавжудлиги аниқланади - $I_p(n)$ $GF(P)$ фазода $n$ дан катта ёки тенг бўлган ягона туб сон.

Очиқ матнни шифрлаш ва криптограмманинг парчаланиши олдинги иккита вариантнинг босқичларига кўрилди, шунинг учун уларни ко'риб чиқмаймиз.

RSA алгоритмининг дастури.

```
import javax.swing.*;
import java.io.FileNotFoundException;
import java.math.BigInteger;
import java.util.Random;
import java.util.Scanner;


public class EncryptionRSA {
    public static void main(String[] args) throws FileNotFoundException {

        int p,q;
        Scanner input = new Scanner(System.in);

        System.out.print("Ochiq Matn = ");
        String matn = input.nextLine();

        System.out.print(" p = ");
        p=input.nextInt();
        System.out.print(" q = ");
        q=input.nextInt();

        boolean h =true;
        if(tublikka_tekshirish(p, q)){
            int n = y3(p,q);
            int fn = y4(p,q);
            int a = y5(fn);
            int e = y6();
            int k=y7();
```

```
            int b=y8();
            while (X4(k)) {
                while (X5(a, b)) {
                    if (X6(a, b)) {
                        a = y9(a, b);
                    }else{
                        b = y10(b, a);
                    }
                }
                if (a == 1) {
                    k = y14();
                }
                else{
                    e = y11(e);
                    a = y12(fn);
                    b = y13(e);
                }
            }
            int d=y15();
            while (X8(d,e,fn)) {
                d=y16(d);
            }

            char []belgilar = matn.toCharArray();

            String []shifrMatn =new String[belgilar.length];

            for (int i = 0;i < belgilar.length; i++) {
                shifrMatn[i] = y17((int)belgilar[i], e, n).toString();
            }
            BigInteger sum = BigInteger.valueOf(0);
            String deshifr_matn[] = new String[belgilar.length];
            System.out.print("Shifrlangan ma'lumot: ");
            int index=0;
            for (int i = 0; i < shifrMatn.length; i++) {
                sum = BigInteger.valueOf(Long.parseLong(shifrMatn[i]));
                index = sum.intValue();
                System.out.print((char)index);
                deshifr_matn[i] = (y18(sum, d, n)).toString();
            }

            System.out.print("\nDeshifrlangan malumot:");
            for(int i=0; i < belgilar.length; i++){
                System.out.print((char)Integer.parseInt(deshifr_matn[i]));
            }
            System.out.println();
        }
        else{
            JOptionPane.showMessageDialog(null, "Kiritilgan son tub son emas!!!",
                    "Xatolik", JOptionPane.CLOSED_OPTION);
            h = false;
        }
    }

    public static BigInteger y17(int data,int ochiq_kalit,int n){

        BigInteger data_big = BigInteger.valueOf(data);
        BigInteger ochiq_big = BigInteger.valueOf(ochiq_kalit);
        BigInteger n_big = BigInteger.valueOf(n);
        return data_big.modPow(ochiq_big, n_big);
    }

    public static BigInteger y18(BigInteger shifr,int yopiq_kalit,int n){
```

```
    BigInteger yopiq_big = BigInteger.valueOf(yopiq_kalit);
    BigInteger n_big = BigInteger.valueOf(n);
    return shifr.modPow(yopiq_big, n_big);
}
public static boolean X8(int d,int e,int fn){
    if((d*e)%fn != 1) return true;
    else return false;
}
public static int y16(int d){
    return d+1;
}
public static int y15(){
    return 2;
}
public static int y14(){
    return 1;
}
public static int y11(int e){
    return e+1;
}
public static int y12(int e){
    return e;
}
public static int y13(int e){
    return e;
}
public static int y9(int a, int b){
    return a-b;
}
public static int y10(int b, int a){
    return b-a;
}
public static boolean X6(int a, int b){
    if(a > b) return true;
    else return false;
}
public static boolean X5(int a, int b){
    if(a!=b) return true;
    else return false;
}
public static int y8(){
    return 2;
}

public static boolean X4(int a){
    if (a == 0) return true;
    else return false;
}

public static int y6(){
    return 2;
}
public static int y7(){
    return 0;
}
public static int y5(int a){
    return a;
}
public static int y4(int p,int q){
    return (p-1)*(q-1);
}
public static int y3(int p,int q){
    return p*q;
```

```
    }
    public static boolean tublikka_tekshirish(int p,int q){

        if (!tub(q) && !tub(p)) {
            return true;
        }
        else{
            return false;
        }
    }
    public static boolean tub(int number){
        int i=y1();
        boolean status = true;
        if (number < 2)
            return true;
        while (x1(i,number)) {
            i++;
            if (x2(number,i)) {
                status = false;
                break;
            }
        }
        if (status)
            return false;
        else
            return true;
    }

    public static boolean x2(int p,int i){
        return p%i==0;
    }

    public static int y2(int i){
        return i =i+1;
    }

    public static boolean x1(int i,int number){
        return i < number/2.0;
    }
    public static int y1(){
        return 1;
    }
}
```

## Масала ва машқлар

1. GF (2) майдонида полиномий вариант учун RSA тизимининг намунасини яратинг.
2. Агар $n = p * q$ бўлса ( ва $q$ ўзаро тенг бўлмаган туб сонлар) $\phi(n) = (p-1) * (q-1)$ еканини исботланг.
3. Агар $m$ туб сон бўлса $(m-1)! + 1$ ни $m$ га бўлинишини исботланг.
4. Қуйидаги техстни RSA нинг кўрсатилган учта усулларида шифрланг.

$T$ =ПАУК_НА_СУРАНЕН_КЕРЦЕ

5. $n$ дан катта ёки тенг бўлган $I_p(n)$ катта туб сонини $GF(P)$ фазодан аниқлаб берувчи фойдали алгоритмни яратинг.

## 6. Диффи Хеллман – ахборот ҳимояси тизими(*6 амалий машғулот*)

Айтайли $GF(q)$ – Галуа майдонидаги тартиби $q$. У ҳолда ихтиёрий $GF^*(q)$ мултипликатив груҳидан олинган иккита $a$ ва $b$ лар учун қуйидаги таққослама $x$ га нисбатан доимо ўринлидир:

$$a^x = b(modp)$$

$x$ $GF(q)$ дан олинган бутун сон, таққосламадаги $x$ ни бошқача қилиб логарифм $a$ асосига кўра $b$ деб айтилади ва одатда қуйидагича ёзилади: $x = \log_a b$.

Масалан оддий Галуа $GF(5) = \{0,1,2,3,4\}$ майдонини таққосламаси:

$$3^x = 4(mod5)$$

Буни ечими еса қуйидагича: $x = \log_3 4 = 2$.

$GF(9)$ Галуа майдонининг таркиби учун $a$ елемент квадрат тенгламанинг илдизлари бўлади:

$$X^2 + 2X + 2 = 0 \text{ бўлса } a^2 + 2a + 2 = 0,$$

Натижа

$$\log_a(-1) = 4 \text{ va } a^4 = -1.$$

Бу ҳолда криптографик о'згаришларнинг қайтарилмаслиги $q$ елементларидан ташкил топган сонли Галуа майдонида експонент функсиясини яъни $y = a^x$ ни ҳисоблаш учун осонлик билан та'минланади. Аммо $x = \log_a y$ каби логарифмларни ҳисоблаш ушбу усулнинг криптографик кучини аниқлайдиган ва жуда ко'п вақт сарфлайдиган операция ҳисобланади.

Айтайлик $q$(майдондан олинган) ва $\alpha$ ( $F(q)$ майдонидан аниқланган) елементлари барча учун очиқ.

Қуйидаги кетма-кетлик орқали Диффи-Хеллман тизими учун умумий махфий калитни яратамиз: Ixtiyoriy $a$ (1< $a$< $q$)soni tanlanadi.

1. $T = \alpha^a mod q$ ҳисобланади ва қабул қилувчига жўнатилади.
2. Қабул қилувчи ихтиёрий $b$ (1<$b$<$q$) сонини танлайди ва $H = a^b (mod q)$ ни ҳисоблайди кейин жўнатувчига юборади.
3. Қабул қилувчи $K_1 = T^b = \alpha^{ab} (mod q)$ ни ҳисоблайди.
4. Жўнатувчи $K_2 = H^a = \alpha^{ab} (mod q) = K_1$ ни ҳисоблайди.

**Масала ва машқлар**

1. $GF(8)$ ва$GF(9)$ майдонлари елементлари учун лагарифм жадвалини қуриш.
2. Қайта очилган текст ($T =$ МАРАНД — ЭТО АХБЮУР) ва ихтиёрий туб $GF(101)$ майдонидан фойдаланган ҳолда икки фойдаланувчи ўртасидаги умумий махфий калитни қуринг.
3. Диффие-Ҳеллман шифрлаш тизимини ўзгартиринг, агар иштирокчилар сони учдан катта ёки тенг бўлса.
4. Яширин калитни яратиш учун Диффие-Ҳеллман шифрлаш тизимидаги ҳужум алгоритмини ишлаб чиқинг.
5. Оддий Галуа майдонининг чексиз синфлари учун иккита оддий елемент маълум бўлса. Биринчи юзта шундай майдонларни аниқланг.

## 7. Электрон рақамли имзо(*7 амалий машғулот*)

Мунтазам хат ёки ҳужжатнинг охирида ижрочи ёки мас'ул шахс одатда иккита мақсадга еришади. Биринчидан, қабул қилувчининг хатни ҳақиқатлигини текшириш имконияти мавжуд, у имзо учун ўзи мавжуд бўлган намуна билан бириктирилади. Иккинчидан, шахсий имзолаш - муаллифлик ҳужжатининг ҳуқуқий кафолати бўлиб, у шартнома тузишда, ишончнома тузишда, мажбуриятларда ва бошқаларда.

Шу каби вазифалар рақамли имзони ҳал қилишга имкон беради.

Айтайлик $A$ фойдаланувчи $T$ маълумотни имзолаши керак. У махфий $K$ калитни ва махфий $F_K(p)$ функсияни қўллашни билади. $F_K(P) = T$ ни қаноатлантирувчи $P$ мавжуд ва $T$ маълумотга ўзининг ҳақиқий $P$ имзосини қўйиб $B$ манзилга жўнатади. Рақамли имзо орқали имзоланган малимот қандай текширилади, $(T, P)$ жуфтлик сифатида ифодаланиши мумкин, бу ерда $T$ ва $P$ юборувчининг хабарлари ва имзоси.

Бу ҳолда $(T, P)$ жуфтликлар мавжуд,жунатиладиган манзил аниқ, кейинг вазифаларни параллел ҳолда бажариш мумкин.

1. $T$ маълумот имзоланади, $F_K(P) = T$ тенглама ечилади фақат манзилида рухсат етилган, имзолаш учун хавфсиз вақтдан кам бўлиши мумкин эмас.
2. $F_K$ билувчи қонуний фойдаланувчилар томонидан ҳар қандай имзо ҳақиқийлиги текширилиши мумкин.
3. Қачонки имзо қалбакилиги аниқланса рад етилади. $(T, P)$ Имзоланган хабарга зарар етказилишидан қўрқмасдан тегишли алоқа каналлари орқали жўнатилиши мумкин. Амалда, очиқ калитларга ега алгоритмлар кўпинча ҳужжатларни имзолаш учун камдан-кам ҳолларда қўлланилади.

Шу сабабли, қулайлик учун рақамли имзо протоколлари одатда бир томонлама хаш функциялари билан биргаликда ишлатилади ва жўнатувчи ҳужжатни ўзи имзоламайди, лекин хеш функсия тушунчаси мазкур ҳужжат учун. АҚШ ва Россиянинг қабул қилинган стандартларида рақамли имзоланган алгоритмлар (DSA ва ГОСТ Р 34.10-94) махсус яратилган алгоритмлардан фойдаланади, хусусан, Диффие-Ҳеллман, Эл-Гамал ва Шнорrнинг катта туб сонлар учун алгоритмлари.

Диффие-Ҳеллман алгоритмининг дастури.

```
/*  this program calculates the Key for two persons  using the Diffie Hellman Key
exchange algorithm */
#include<stdio.h>
long int power(int a,int b,int mod)
 {
  long long int t;
```

```
  if(b==1)
  return a;
  t=power(a,b/2,mod);
  if(b%2==0)
   return (t*t)%mod;
  else
   return (((t*t)%mod)*a)%mod;
 }
 long long int calculateKey(int a,int x,int n)
 {
  return power(a,x,n);
 }
 int main()
 {
  int n,g,x,a,y,b;
// both the persons will be agreed upon the common n and g
  printf("Enter the value of n and g : ");
  scanf("%d%d",&n,&g);
// first person will choose the x
  printf("Enter the value of x for the first person : ");
  scanf("%d",&x);  a=power(g,x,n);
// second person will choose the y
  printf("Enter the value of y for the second person : ");
  scanf("%d",&y);  b=power(g,y,n);
  printf("key for the first person is : %lld\n",power(b,x,n));
  printf("key for the second person is : %lld\n",power(a,y,n));
  return 0;
 }
```

## Масала ва машқлар

1. Фойдаланувчилар бир-бирига ишонмаса, худди шу ҳужжатни бир вақтнинг ўзида иккита фойдаланувчи томонидан хаш функцияларини қўллаш учун протоколни имзоланг.
2. Рақамли имзоларни амалга оширадиган барча криптосистемаларнинг бир томонлама функцияларга асосланганлиги, яни ҳар қандай ассиметрик тизим рақамли имзони қўллаш учун асос бўлиши мумкинлиги маълум. Маълум криптографик тизимлардан фойдаланган ҳолда рақамли имзони қўллаш учун расмий схемани беринг.
3. Сизга маълум бўлган криптосистемалар учун рақамли имзони қўллаш учун алгоритмни ишлаб чиқиш.
4. Шифрланган калитларга асосланган криптографик тизимларда шифрлаш ва рақамли имзолаш учун бир хил калитларни ишлата оласизми?

5. Умумий ва рақамли имзоларнинг умумийлиги нима? Улар қандай фарқ қилади?
6. Рақамли имзо тизимларини амалий қўллашда асосий мураккаблик нимадан иборат?
7. Маълум криптографик тизимлардан фойдаланган ҳолда рақамли имзони қўллаш учун расмий схемани истемол қилинг.

## 8. Эль-Гамал критотизими асосида рақамли имзони жорий қилиш(*8 амалий машғулот*)

Айтайлик GF (p) сонли майдон катта хусусиятли $p, (k, p-1) = 1$ ва $g$ примитив елементдан ташкил топган бўлсин. Биринчидан, хабарларни шифрлаш / дешифрлаш учун Эл-Гамал криптосистемасининг зарур параметрларини ўрнатамиз. Улар:

***Очиқ калит***

$p$ – оддий сон(ҳамма учун умумий бўлиши мумкин);

$g < p$ (ҳамма учун умумий бўлиши мумкин);

$y = g^x modp$ ($x$ – ихтиёрий сон);

***Махфий калит***

$$x < p;$$

***Шифрлаш***

$k$ – ихтиёрий усулда танланади, $(k, p-1) = 1$;

Шифртекстнинг биринчи қисмида:

$$a = g^K (modp);$$

Шифртекстнинг иккинчи қисмида:

$b = M * g^K (modp\ )$ ($M$ – маълумот)

Натижада $M$ ни шифрлашдан ҳосил бўлган ушбу жуфтлик $(a, b) = E$ очиқ малумотдан икки баробар узундир.

***Дешифрлаш:***

$$M = b/g^x(modp).$$

Энди, бу криптосистеманинг модификациясини електрон рақамли имзо учун ишлатиш имконини беради. Имзонинг иккинчи қисмини истисно қилишдан ташқари, криптография тизимининг параметрлари билан ҳисоб-китоб қилиш ва бутунликни текшириш параметрлари мос келади:

$b$ қуйидагича, $M = (xa + kb)\big(mod(p1)\big)$.

имзони текширамиз

Агар $y^a a^b(modp) = g^M(modp)$ бўлса имзо тўғри.

*Мисол.* Айтайлик $p = 13$, $g = 2$ ва $M = 7$ , махфий калит $x = 6$.

у ҳолда $y = g^x(modp) = 2^6(mod13) = 12$

Ва биз кейин очиқ калитларни аниқлаймиз булар: $y = 12, g = 2, p = 13$.

Энди $M = 7$ маълумот имзоланади. Ихтиёрий $k$ сони танланади $p - 1$ га боғлиқ ҳолда, масалан $k = 11$. Кейинг белгилаш $E = (a, b)$ бу ерда:

$$a = g^K(modp) = 2^{11}(mod13) = 7.$$

$a, b$ га боғлиқлиги учун тенгламани ечамиз

$M = (xa + kb)(mod(p - 1))$, $7 = 6 * 7 + 11 * b\ mod12$ демак $b = 11$ екан ва имзони қўямиз: $E = (a, b) = (7, 11)$.

Имзонинг тўғрилигини текшириш учун $y^a a^b(modp) = g^M(modp)$ ни текширамиз.

Ечим: $12^7 7^{11}(mod13) = 35831808 * 1977326743\ mod13 = 12 * 2\ mod\ 13 = 11;\ 2^7(mod13) = 128mod\ 13 = 11$ демак имзо ҳақиқий.

Эл-Гамал алгоритми ёрдамида имзо қўйиш дастури.

```
Imzo quyish
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Unit1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
int qoldiq1(int a,int n)
```

```
{
int r;
r=a%n;
return r;
}
//---------------------------------------------------------------------------
int qoldiq(int a,int b, int c) //b-daraja,c-bo'luvchi
{
int p;//qoldiq
p=qoldiq1(a,c);
for(int i=2;i<=b;i++)
p=qoldiq1(p*a,c);
return p;
}
int tub(int a)
{
int b=1;
for(int i=1;i<=a;i++)
if(a%i==0) b++;
if(b==2) b=0;
else { if(b>3) b=0; else b=1;}
return b;
}
//---------------------------------------------------------------------------
int teskari(int a,int n)
{
int n1,i1,t=1,a1[100000],b[100000], fi;
if(tub(n)==1){ fi=n-2; return qoldiq(a,fi,n);}
if(tub(n)==0)
{
n1=n;
for(int i=2;i<=n;i++)
{
i1=tub(i);
if(i1) { if(n%i==0) a1[t++]=i;}
}
for(int i=1;i<t;i++)  b[i]=0;
while(n!=1) { for(int i=1;i<t;i++) if(n%a1[i]==0) { b[i]++; n=n/a1[i];}    }
int fi1=1,fi2=1;
for(int i=1;i<t;i++) {fi1*=a1[i]; fi2*=(a1[i]-1); }
fi=n1*fi2/fi1; fi--;
return qoldiq(a,fi,n1);
}
}
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button1Click(TObject *Sender)
{
Edit4->Text=qoldiq(StrToInt(Edit2->Text),StrToInt(Edit3->Text),StrToInt(Edit1->Text));
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button2Click(TObject *Sender)
{ int n;
Edit6->Text=qoldiq(StrToInt(Edit2->Text),StrToInt(Edit5->Text),StrToInt(Edit1->Text));
n=teskari(StrToInt(Edit5->Text),StrToInt(Edit1->Text)-1);
Edit7->Text=(n*(StrToInt(Edit8->Text)-StrToInt(Edit3->Text)*StrToInt(Edit6-
>Text)))%(StrToInt(Edit1->Text)-1);
```

```
}
//---------------------------------------------------------------------------
```

Қўйилган имзони текшириш дастури.

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Unit1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
int qoldiq1(int a,int n)
{
int r;
r=a%n;
return r;
}
//---------------------------------------------------------------------------
int qoldiq(int a,int b, int c) //b-daraja,c-bo'luvchi
{
int p;//qoldiq
p=qoldiq1(a,c);
for(int i=2;i<=b;i++)
p=qoldiq1(p*a,c);
return p;
}
int tub(int a)
{
int b=1;
for(int i=1;i<=a;i++)
if(a%i==0) b++;
if(b==2) b=0;
else { if(b>3) b=0; else b=1;}
return b;
}
//---------------------------------------------------------------------------
int teskari(int a,int n)
{
int n1,i1,t=1,a1[100000],b[100000], fi;
if(tub(n)==1){ fi=n-2; return qoldiq(a,fi,n);}
if(tub(n)==0)
{
n1=n;
for(int i=2;i<=n;i++)
{
i1=tub(i);
if(i1) { if(n%i==0) a1[t++]=i;}
}
for(int i=1;i<t;i++)  b[i]=0;
while(n!=1) { for(int i=1;i<t;i++) if(n%a1[i]==0) { b[i]++; n=n/a1[i];}    }
int fi1=1,fi2=1;
for(int i=1;i<t;i++) {fi1*=a1[i]; fi2*=(a1[i]-1); }
fi=n1*fi2/fi1; fi--;
return qoldiq(a,fi,n1);
}
}
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
```

```
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button1Click(TObject *Sender)
{
int n=StrToInt(Edit5->Text);
StringGrid1->ColCount=n;
StringGrid2->ColCount=n;
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button2Click(TObject *Sender)
{
int g,n=StrToInt(Edit5->Text),y,y1,k;
g=qoldiq(StrToInt(Edit3->Text),StrToInt(Edit1->Text),StrToInt(Edit2->Text));
for(int i=0;i<n;i++)
{
for(int j=0;j<n;j++)
{
y=qoldiq(StrToInt(Edit4->Text),StrToInt(StringGrid1->Cells[i][0]),StrToInt(Edit2->Text));
y1=qoldiq(StrToInt(StringGrid1->Cells[i][0]),StrToInt(StringGrid2-
>Cells[j][0]),StrToInt(Edit2->Text));
k=(y*y1)%StrToInt(Edit2->Text);
if(k==g)
{
Edit6->Text=StrToInt(StringGrid1->Cells[i][0]);
Edit7->Text=StrToInt(StringGrid2->Cells[j][0]);
}


}

}
}
//---------------------------------------------------------------------------
```

## Масала ва машқлар

1. Эл-Гамал ва Диффие-Ҳеллман криптосистемалари ўртасидаги фарқ нима? Ҳар доим $q(q < p)$ рақами сифатида $GF(p)$ нинг примитив елементини танлаш керакми?
2. Эл-Гамал шифрлаш тизимини $p = 19$ учун қуриб, $M =$ ВЕРНИСЬ_В_АРЦАХ маълумотни имзоланг.

## Адабиётлар


1. *Алферов А. П., Зубов А. Ю., Кузьмин А. С., ЧеремушкинА. В. Основы криптографии. — М.: Гелиос АРВ, 2002.*

2. Введение в криптографию/Под ред. *В. В. Ященко.* — М., 1998.

3. Защита информации в персональных ЭВМ/*А. В. Спесивцев*, *В. А. Вегнер*, *А. Ю. Крутиков*, *В. В. Серегин*, Б. Л. *Сидоров.* — М., 1993.

4. *Коблиц Н.* Курс теории чисел и криптографии. — М., 2001.

5. *Нечаев В. И.* Элементы криптографии. Основы теории защиты информации. — М., 1999.

6. *Саломаа А.* Криптография с открытым ключом. — М., 1995.

7. *Хоффман Л. Дж.* Современные методы защиты информации. — М., 1980.

8. *Чмора А. Л.* Современная прикладная криптография. — М.: Гёлиос АРВ, 2001.

9. *Шеннон К.* Теория связи в секретных системах. — М., 1949.