O’ZBEKSITON RESPUBLIKASI AXBOROT TEXNOLOGIYALARI VA KOMMUNIKATSIYALARINI RIVOJLANTIRISH VAZIRLIGI

TOSHKENT AXBOROT TEXNOLOGIYALARI UNIVERSITETI

H i m o y a g a
kafedra mudiri
Irgasheva D.Ya._____
«___»_____2015 y.

BAKALAVR BITIRUV ISHI

Mavzu: «Kalit yaratishda va saqlashda xavfsiz qurilmalardan foydalanish»

Bitiruvchi	_____ (imzo)	Onorov A.A.
Rahbar	_____ (imzo)	Kusayev A.
Taqrizchi	_____ (imzo)	Siddiqov I.X. (f.i.sh)
HFX maslahatchisi	_____ (imzo)	Qodirov F.M. (f.i.sh)

Toshkent – 2015

Mundarija

Kirish

Davlat aktivlarini boshqarishni yanada maqbullashtirish, aloqa va axborotlashtirish sohasi tashkilotlariga yuklangan vazifalar samarali bajarilishi uchun qlulay shart-sharoitlar yaratish, ularning moddiy-texnika bazasini mustaxkamlash maqsadida “O’zbekiston Respublikasi Vazirlar Maxkamasi” tomonidan bir qancha qarorlar ishlab chiqilgan bo’lib ushbu qarorlarning maqsadi O’zbekisyon Respublikasi hududida axborotlashtirish sohasini rivojlantirishga xizmat qiladi. Bundan tashqari sohani rivojlantirishga qaratilgan “O’zbekiston Respublikasining Axborotlashtirish to’g’risida” va “Davlat sirlarini saqlash to’g’risida”gi qonunlariga muvofiq, shuningdek axborotni muhofaza qilishning kriptografik vositalari hamda kriptografiya tizimlaridan foydalangan holda maxfiy yoki davlat sirlaridan iborat bo’lgan ma’lumotlar bayon etilgan axborotni muhofaza qilish faoliyatini tartibga solish maqsadida amalga oshiriladi.

Fan-texnika va marketing tadqiqotlari markazining bosh ilmiy xodimi Xasanov P. F. rahbarligida 2002-yilda kriptografiya bo’yicha birinchi patentga talabnoma berildi. Bu patent 2006-yilda O’zbekiston Respublikasi davlat patent idorasi tomonidan ro’yxatga olindi va “Raqamli imzoni shakllantirish va autentifikatsiya usuli” ixtirosi uchun 03070-sonli patent berildi.

Huquqni muhofaza qilish faoliyatining axborot ta’minotini jadallashuvi, tezkor vaziyatni tavfsiflovchi, ishonchli ma’lumotlarni muntazam yig’ib borilishi, uning o’z vaqtida va sifatli tahlil qilinishi hozirgi sharoitda jinoyatchilikka qarshi kurashni tashkil etishning muhim shartlaridan biri hisoblanadi. Axborot jarayonlarini avtomatlashtirish kompyuter texnikasini joriy etishga, ma’lumotlarni yig’ish, saqlash, qayta ishlash va uning asosida ma’lumotlarni berish avtomatlashtirilgan tizimlarini yaratishga bevosita bog’liqdir.

O’zbekiston Respublikasi prezidentining 2015 yildagi “O’zbeksiton Respublikasi Axborot Texnologiyalari va Kommunikatsiyalarini Rivojlantirish Vazirligini yaratish haqida”gi qarorida “Axborot xavfsizligini ta’minlash, mahfiy va sifatli aloqani tashkil etish ...” muammolari ham alohida ko’rib chiqilishi berilgan[2].

«Kriptografiya» atamasi dastlab «yashirish, yozuvni berkitib qo‘ymoq» ma’nosini bildiradi. Birinchi, marta u yozuv paydo bo‘lgan davrlardayoq aytib o‘tilgan. Hozirgi vaqtda kriptografiya deganda har qanday shakldagi, ya’ni diskda saqlanadigan sonlar ko‘rinishida yoki hisoblash tarmoqlarida uzatiladigan xabarlar ko‘rinishidagi axborotni yashirish tushuniladi. Kriptografiyani raqamlar bilan kodlanishi mumkin bo‘lgan har qanday axborotga nisbatan qo‘llash mumkin. Maxfiylikni ta’minlashga qaratilgan kriptografiya kengroq qo‘llanilish doirasiga ega. Aniqroq aytganda, kriptografiyada qo‘llaniladigan usullaning o‘zi axborotni himoyalash bilan bog‘liq bo‘lgan ko‘p jarayonlarda ishlatilishi mumkin.

Kriptografiya axborotni ruxsatsiz kirishdan himoyalab, uning maxfiyligini ta’minlaydi. Masalan, to‘lov varaqlarini elektron pochta orqali uzatishda uning o‘zgartirilishi yoki soxta yozuvlarning qo‘shilishi mumkin. Bunday hollarda axborotning ta’minlash zaruriyati paydo bo‘ladi. Umuman olganda компьyuter tarmog‘iga ruxsatsiz kirishning mutlaqo oldini olish mumkin emas, lekin ularni aniqlash mumkin. Axborotning yaxlitligini tekshirishning bunday jarayoni, ko‘p hollarda, axborotning haqiqiyligini ta’minlash deyiladi. Kriptografiyada qo‘llaniladigan usullar ko‘p bo‘lmagan o‘zgartirishlar bilan axborotlarning haqiqiyligini ta’minlashi mumkin[4].

Nafaqat axborotning kompyuter tarmog‘idan ma’nosi buzilmasdan kelganligini bilish, balki uning muallifdan keganligiga ishonch hosil qilish juda muhim. Axborotni uzatuvchi shaxslarning haqiqiyligini tasdiqlovchi turli usullar ma’lum. Eng universal protsedura parollar bilan almashuvdir, lekin bu juda samarali bo‘lmagan protsedura. Chunki parolni qo‘liga kiritgan har qanday shaxs axborotdan foydalanishi mumkin bo‘ladi. Agar ehtiyotkorlik choralariga rioya qilinsa, u holda parollarning samaradorligini oshirish va ularni kriptografik usullar bilan himoyalash mumkin, lekin kriptografiya bundan kuchliroq parolni uzluksiz o‘zgartirish imkonini beradigan protseduralarni ham ta’minlaydi. Kriptografiya sohasidagi oxirgi yutuqlardan biri – raqamli signatura – maxsus xossa bilan axborotni to‘ldirish yordamida yaxlitlikni ta’minlovchi usul, bunda axborot uning muallifi bergan ochiq kalit ma’lum bo‘lgandagina tekshirilishi

mumkin. Ushbu usul maxfiy kalit yordamida yaxlitlik tekshiriladigan ma’lum usullardan ko‘proq afzalliklarga ega.

Ma’lumotlarni shifrlash va uni mahfiy kanal orqali uzatishda bir martalik parollarning o’rni kata hisoblanadi. Ularni generatsiya qilish va saqlashda maxsus kalit qurilmalarini saqlovchi e-tokenlar, Java simcardlar va boshqalaridan foydalaniladi.

Ishning maqsadi: Kalitlarni xavfsiz saqlash va yaratish tizimini ishlab chiqish.

Ushbu bakalavr bitiruv ishi kirish, 3 ta qism, xulosa va foydalanilgan adabiyotlar to’plamidan iborat.

1-qism nazariy qism bo’lib, kalitlarni yaratishda va saqlashda kriptogafiyaning o’rni, simmetrik va asimmetrik shifrlash algoritmlariga bag’ishlangan.

2-qism amaliy qism, unda kalitlarni yaratuvchi va saqlovchin tizimlar: e-Tokenlar, Java cardlar, USB kalitlar va boshqalari tahlil qilinib, kriptografik algoritmlardan foydalanib kalit yaratish dasturini ishlab chiqilgan.

3-qismda hayot faoliyati xavfsizligiga oid ma’lumotlar keltirilgan.

Xulosada ish bo’yicha asosiy natijalar berilgan.

1. Nazariy qism. Kriptografik algoritmlar yordamida kalitlar yaratish

1.1. Tasodifiy va psevdotasodifiy sonlar asosida kalitlarni yaratish

Jamiyat taraqqiyoti insoniyat tafakkurining mahsuli bo'lgan rivojlangan ilm-fan yutuqlariga asoslangan texnika va texnologiyalar bilan bir qatorda, keng ma'noda, axborotlarning muhim ahamiyatga egaligi orqali ham belgilanadi. Inson tafakkuri rivojining manbai esa ma'lumotlar (axborotlar) majmuidan iboratdir. Shak-shubhasiz o'z vaqtida olingan to'la va ishonchli ma'lumot, shu ma'lumot bilan bog'liq bo'lgan holatdan kelib chiqadigan amaliy faoliyatlarning maqsadli kechishlarini muvofiqlashtirishda muhim ahamiyat kasb etadi. Faoliyat maqsadlarining turlicha bo'lishi tabiiy ravishda axborotlardan turli maqsadlarda foydalanish asoslariga sabab bo'ladi. Shuning uchun bugungi, axborotlarni saqlash va uzatish tizimlari bir tomondan takomillashib murakkablashgan va ikkinchi tomondan axborotlardan foydalanuvchilar uchun keng qulayliklar vujudga kelgan davrda, axborotlarni maqsadli boshqarishning qator muhim masalalari kelib chiqadi. Bunday masalalar qatoriga katta hajmdagi axborotlarning tez va sifatli uzatish hamda qabul qilish, axborotlarni ishonchliligini ta'minlash, axborotlar tizimida axborotlarni begona shaxslardan (keng ma'noda) muhofaza qilish kabi ko'plab boshqa masalalar kiradi. Axborot va axborot tizimidan foydalanish insoniyat faoliyatining barcha sohalariga kirib borib, muhim ahamiyat kasb etib, rivojlanib borayotgan bugungi jamiyatda axborotlarni maqsadli boshqarish faollashmoqda. Kompyuterlar va kompyuter tizimlari axborot tizimining muhim bo'g'imidir. Internet tarmoqlari jamiyat faoliyatining barcha sohalarini qamrab olib, axborotni tez va sifatli almashinuvini ta'minlash texnologiyalarining rivojlanishiga ijobiy manba bo'lib kelmokda. Yuqoridagi keltirilgan asosli mulohazalardan kelib chiqib, axborotlarni asli holidan o'zgartirilgan holda, ya'ni shifrlangan holda, saqlash va uzatish masalalarining muhim ekanligiga shubxa yo'qdir. Axborotlarning muhofazasini ta'minlash masalalari insoniyat jamiyatida qadimdan muxim bo'lib kelgan. Aytish mumkinki, axborotni muhofaza qilish uslublari jamiyatda dastlabki paydo bo'lgan muomila tili va yozuvi bilan uzviy bog'liq hamda tengdoshdir. Haqiqatdan ham, qadimda yozuv muomila vositasidan

faqat ayrim yuqori tabaqadagi jamiyat a’zolarigina foydalanganlar. Qadimiy Misr va Hindistonning ilohiy kitoblari bunga misol bo’la oladi. Eramizdan avvalgi beshinchi asrda yashab o’tgan grek olimi Gerodotning habar berishicha, qadimiy Misrda shifrlangan axborotlar rolini, jreslar, ya’ni yuqori tabaqadagi yetuk fikrli kishilar tomonidan yaratilgan muomila tili bajargan. Bunda uchta alfbo asoslanilgan: yozuv, ilohiy va mahfiy. Yozuv alfbosi oddiy o’zaro muomilada qo’llanilgan, ilohiy alfavit diniy muomila vositasi sifatida qo’llanilgan, mahfiy alfbo esa ma’lumotlarning asl ma’nosini begonalardan muhofaza qilishda va astriologlar tomonidan qo’llanilgan. Turli yozuv alfbolarining vujudga kelishi va rivojlanishi natijasida *kriptografiya* mustaqil yo’nalishda rivojlana bordi.

Fan sifatida *kriptografiya* - axborotni aslidan o’zgartirilgan holatga akslantirish uslublarini topish va takomillashtirish bilan shug’ullanadi. Dastlabki sistemalashgan kriptografik uslublar eramiz boshida, Yuliy Sezarning ish yuritish yozishmalarida uchraydi. U, biror ma’lumotni mahfiy holda biror kishiga yetkazmoqchi bo’lsa, alfboning birinchi harfini to’rtinchi harfi bilan, ikkinchi harfini beshinchisi bilan va hokazo tartibda almashtirib matnni asl holatidan shifrlangan matn holatiga o’tkazgan.

Kriptografiya tizimlari yo’nalishidagi izlanishlar ayniqsa birinchi va ikkinchi jahon urushi yillari davrida muhim ahamiyat kasb etdi va jadal rivojlandi. Urushdan keyingi yillarda, hisoblash texnikalarining yaratilishi, ularning takomillashib, insoniyat faoliyatining barcha sohalariga chuqur va keng ma’noda kirib borishi, kriptografik uslublarni tabiiy ravishda rivojlanib va takomillashib borishini ta’qozo etmoqda.

Kriptografiyaning axborot tizimi muhofazasi masalalarida qo’llanilishi, ayniqsa, hozirgi kunda faollashib bormoqda. Haqiqatan ham, bir tomondan kompter tizimlarida internet tarmoqlaridan foydalangan holda katta hajmdagi davlat va harbiy ahamiyatga ega bo’lgan, hamda, iqtisodiy, shahsiy va boshqa turdagi axborotni tez va sifatli uzatish, qabul qilish kengayib bormoqda. Ikkinchi tomondan esa bunday axborotlarning muhofaza qilinishi ta’minlash masalalari muhimlashib bormoqda.

Axborotni muxofaza qilish masalalari bilan *kriptologiya* (kryptos- mahfiy, logos- ilm) fani shug'ullanadi. Kriptologiya maqsadlari o'zaro qarama-qarshi ikkita yo'nalishiga ega bo'lgan – *kriptografiya* va *kriptoanaliz.*

Kriptografiyaning ochiq ma'lumotlarni shifrlash masalalarining matematik uslublari bilan shug'ullanishi to'g'risida yuqorida aytib o'tildi.

Kriptoanaliz esa shifrlash uslubi (kaliti yoki algoritmi)ni bilmagan holda shifrlangan ma'lumotni asl holatini (mos keluvchi ochiq ma'lumotni) topish masalalarini yechish bilan shug'ullanadi.

Kriptografiya quyidagi to'rtta bo'limni o'z ichiga oladi:

— Simmetrik kriptotizimlar.

— Ochiq uslubga yoki yana boshqacha aytganda ochiq kalitlar algoritmiga asoslangan kriptotizimlar.

— Elektron raqamli imzo kriptografik tizimlari.

— Kriptotizimlar uchun kriptobardoshli kalitlarni ishlab chiqish va ulardan foydalanishni boshqarish.

Uzuliksiz (oqimli) shifrlash algoritmlarini yaratilishi ham tabiiy zaruriyat asosida vujudga kelgan. Nisbatan kichik uzunlikka ega bo'lgan, ya'ni kafolatlangan kriptobardoshlilikni ta'minlovchi uzunlikga ega bo'lgan - bugungi kunda 128 bitdan kam bo'lmagan kalit bilan bir tomonli kriptografik akslantirishlar asosida, yetarli darajada katta uzunlikdagi psevdotasodifiy sonlar ketma-ketlik (PTSKK) gammasi ishlab chiqaruvchi generatorlar negizida uzuliksiz shifrlash algoritmlari yaratiladi. Uzunligi 128 bitdan kam bo'lmagan kalitlarning mumkin bo'lgan barcha variantlari soni 2^{128} tadan kam bo'lmay, ularni hammasini tanlab chiqish jarayonini amalga oshirish, bugungi kun hisoblash texnika va texnologiyalarining mavjud ilg'or imkoniyatlaridan foydalanilganda har doim ham samarali natijalar beravermaydi. Ana shunday generatorlar ishlab chiqargan gamma ketma-ketlikni tashkil etuvchi alfavit belgilarini ochiq ma'lumot mos alfaviti belgilari bilan biror amal bajarish orqali shifr ma'lumot alfaviti belgilariga almashtirish – gammalashtirish amalga oshiriladi. Bunday shifrlash jarayoni ko'p alfavitli o'rniga qo'yish shifrlashni amalga oshirishni samarali usuluni ifodalaydi –

kafolatli kriptobardoshlilikni ta’minlovchi kichik uzunlikdagi kalit bilan, ochiq ma’lumotning chastotaviy xususiyatlarini shifrma’lumotga ko’chirmaydigan yetarli kriptobardoshlilikni ta’minlovchi shifrlashni amalga oshiradi [9].

Oqimli shifrlash algoritmlari asosini PTSKK ishlab chiqaruvchi generatorlar tashkil etadi. Bunday generatorlarning asosiy kriptobardoshlilik xarakteristikasi ushbu generatorlar hosil qilgan ketma-ketlikning tasodifiyligidadir. Hosil qilingan ketma-ketliklar bloklarining tasodifiylik darajasi ma’lum bir kriteriylar orqali baholanadi. Tasodifiylik darajasi yuqori bo’lgan psevdotasodifiy ketma-ketlikni ishlab chiqaruvchi generatorlar zamonaviy kriptotizimlarning ajralmas qismi hisoblanadi. Tasodifiy ketma-ketliklar kriptografiyada quyidagi maqsadlarda qo’laniladi:

— simmetrik kriptotizimlar uchun tasodifiylik darajasi yuqori bo’lgan seans kalitlari va boshqa kalitlarni generasiya qilishda;

— asimmetrik kriptotizimlarda qo’llaniladigan katta qiymatlar qabul qiluvchi parametrlarning tasodifiy boshlang’ich qiymatlari generasiyasida;

— blokli shifrlash algoritmlarining boshlang’ich tasodifiy qiymat talab qiluvchi SVS, OFB va boshqa qo’llanish tartib-qoidalari uchun tasodifiylik darajasi yuqori bo’lgan boshlang’ich vektorlar hosil qilishda;

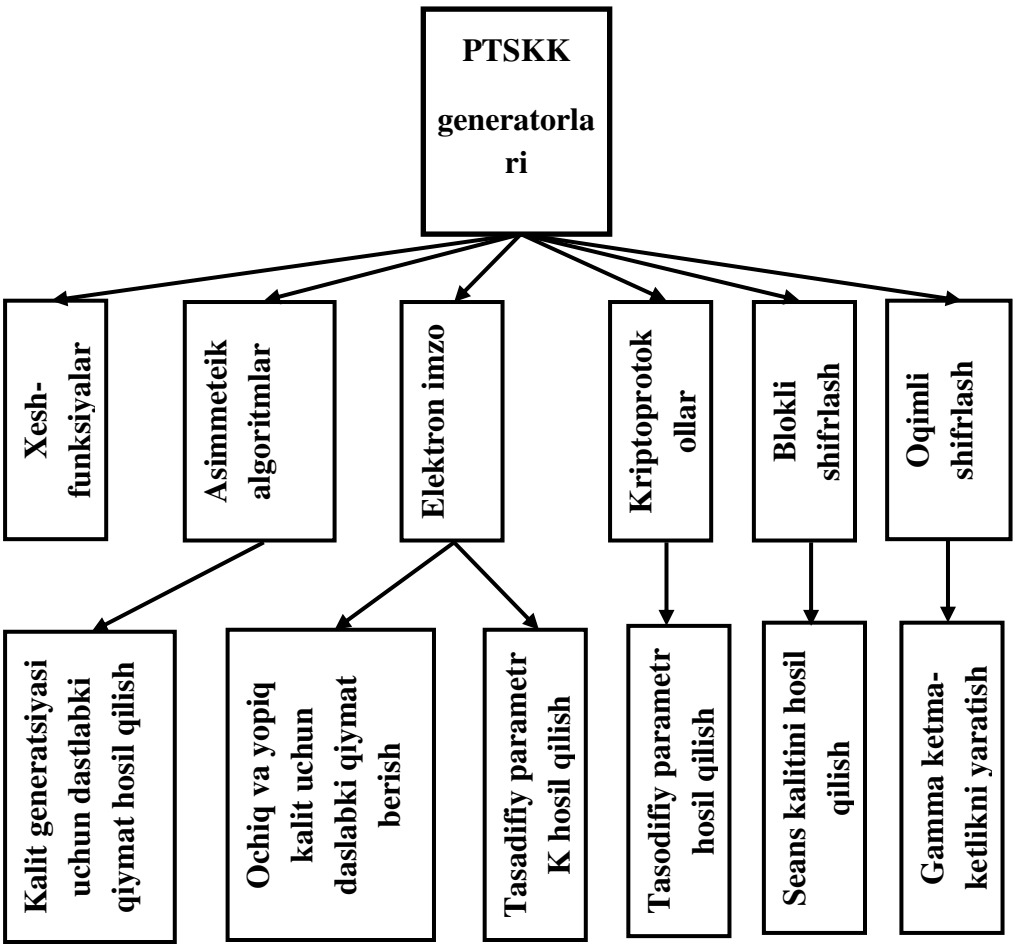
— elektron raqamli imzo tizimlarida katta qiymatga ega parametrlar uchun dastlabki tasodifiy qiymatlarni generasiyasida;

— bitta protokol orqali bir xil ma’lumotlarni har-xil kalitlar qo’llash bilan shifrlab har-xil ko’rinishda uzatish uchun talab qilinadigan holatlarda kalit uchun yetarli uzunlikdagi tasodifiy ketma-ketlik hosil qilishda, masalan SSL va SET protokollarida.

Biror ketma-ketlikni tashkil etuvchi elementlar, shu ketma-ketlikda deyarli teng miqdorda qatnashgan bo’lsa, bu ketma-ketlik tekis taqsimotga ega deyiladi: agar A -ketma-ketlikni tashkil etuvchi $x_t \in A$ -elementlari soni N ta bo’lsa, u holda ihtiyoriy $t \in N$ uchun, A -ketma-ketlikni tashkil etuvchi $x_t \in A$ -elementning, shu ketma-ketlikdagi chastotasi, boshqa elementlarning chastotasi bilan deyarli bir xil bo’ladi, ya’ni har bir $x_t \in A$ –element shu ketmaketlikda deyarli bir xil

ehtimollik bilan qatnashadi. Tasodifiy ketma-ketliklar xaqiqiy tasodifiy ketma-ketliklarga va psevdotasodifiy ketma-ketliklarga bo'linadi.

Tasodifiy ketma-ketlikni: fizik generatorlar va dasturiy generatorlardan foydalanib hosil qilish mumkin.



1.1. Rasm. Psevdotasodifiy sonlar ketma-ketligi generatorlarini qo'llash sohalari

Fizik hodisalarning o'zgarish majmuiga asoslangan generatorlar orqali ishlab chiqilgan ketma-ketlik haqiqiy tasodifiy bo'lib, bu ketma-ketlikni bir martagina ishlab chiqilib, uni keyinchalik biror bir usul yoki vosita bilan xuddi shunday tarzda takrorlanishini boshqarish murakkab hisoblanadi. Shu sababli ma'lumotlarni shifrlash jarayonida bevosita fizik generatorlar bilan ishlab chiqilgan ketma-ketlikni kalitlar gammasi sifatida qo'llash maqsadga muvofiq emas. Chunki, deshifrlash jarayonida qo'llaniladigan fizik generatorning aynan shifrlash jarayonida qo'llanilgan ketma-ketlikni ishlab chiqishi kafolatlanmaydi.

Noma’lum parametrga (kalitga) bog’liq bo’lgan matematik model asosida psevdotasodifiy ketma-ketlik ishlab chiquvchi dasturiy generatorlar hosil qilgan psevdotasodifiy ketma-ketlikni, nomalum parametr qiymatini bilgan holda, xuddi shu matematik model va uning dasturiy ta’minoti asosida ketma-ketlikning qayta takrorlanishini boshqarish mumkin. Bunday holat, ma’lumotlarni shifrlash jarayonida bevosita dasturiy generatorlar bilan ishlab chiqilgan psevdotasodifiy ketma-ketlikni kalitlar gammasi sifatida qo’llash maqsadga muvofiqligini anglatadi va deshifrlash jarayonida qo’llaniladigan dasturiy generatorning aynan shifrlash jarayonida qo’llanilgan psevdotasodifiy ketma-ketlikni ishlab chiqishi kafolatlanadi.

Ko’rsatilgan amaliy masalalarni yechishda xaqiqiy tasodifiy ketma-ketliklar ishlab chiquvchi tasodifiy fizik xodisalarga asoslangan generatorlar oldindan kalitlar bloklari majmuini yaratishda, generatorlarning boshlang’ich parametrlari qiymatlarini o’rnatishda va boshqa shu kabi masalalarni yechishda samarali natijalar beradi.

Yetarli katta davr uzunligiga ega va tasodifiylik darajasi yuqori bo’lgan ketma-ketliklar hosil qiluvchi dasturiy PTSKK generatorini amalda qo’lanishlari samarali va qulay bo’lib, kriptografik vositalarda keng qo’llaniladi. Mavjud dasturiy generatorlar va ular asosidagi uzluksiz shifrlash tizimlari ma’lum bir yondashuvlar asosida yaratilgan.

Algoritmlarni kriptobardoshliligini yetarli darajada taminlanganligini kafolatlash yoki isbotlash asoslari nuqtai - nazaridan mavjud PTSKK generatorlari asosan uchta yo’nalishga ajratish mumkin:

1. Tizimli-nazariy yondashuv asosida qurilgan PTSKK generatorlar;
2. Murakkablikka asoslangan PTSKK generatorlar;
3. Kombinasiyalash yondoshuvi asosida qurilgan PTSKK generatorlar.

Tizimli-nazariy yondashuv asosida yaratilgan PTSKK generatorlarini yaratilish asoslariga ko’ra:

- elementar rekkurent hisoblashlarga;
- siljitish registrlariga;

- bir tomonli funksiyalarga;
- baytlar va bitlar bloklarining o'rnini bog'liqsiz almashtirishga asoslangan generatorlarga ajratish mumkin.

Murakkablikka asoslangan PTSKK generatorlarini yaratilish asosiga ko'ra:

- katta sonlarni tub ko'paytuvchilarga ajratish murakkabligiga asoslangan;
- kvadratik chegirma usuliga asoslangan;
- diskret logarifmlash masalasining murakkabligiga asoslangan generatorlarga ajratish mumkin.

Kombinasiyalash yondoshuvi asosida qurilgan PTSKK generatorlarini yaratilish asoslariga ko'ra:

- polinomial kombinasiyalashga asoslangan;
- tasodifiy parametrli kombinasiyalashga asoslangan;
- maklaren-marsali usuliga asoslangan generatorlarga ajratish mumkin.

1.2. Zamonaviy generatorlarning ishlash asoslari

Bunga misol sifatida chiziqli va chiziqsiz kongruent generatorlar ni keltirish mumkin. Tizimli-nazariy yondashuv asosida yaratilgan uzluksiz shifrlash algoritmlarining bardoshliligi, bu algoritmlarda qo'llanilgan akslantirishlarning nazariy va amaliy bir tomonlilik xususiyatlarining qay darajada ishonchliligini baholash bilan isbotlanadi.

Elementar rekkurent hisoblashlarga asoslangan psevdotasodifiy ketma-ketlik generatorlari ularda qo'llanilgan akslantirishlarga ko'ra *chiziqli, multiplikativ, chiziqsiz* turkumlarga bo'linadi.

Ulardan yana biri *chiziqli va multiplikativ kongruent generatorlardir.* chiziqli kongruent generatorlar umumiy holatda $x_{i+1}=(ax_i+s)\bmod N$ formula bilan aniqlanuvchi rekkurent hisoblashga asoslangan. Dastlabki berilgan kirish parametrlari asosida ketma-ketliklar hosil qilinadi.

Kirish parametrlari:

N – chekli maydon xarakteristikasini ifodalovchi son, a va s - o'zgarmas musbat butun sonlar, x_0 – boshlang'ich butun qiymatli son;

Ketma-ketlikni tashkil etuvchi chiqish qiymatlari :

$$x_{i+1}=(ax_i+s)\bmod N,\quad i=0,1,2,3,\ldots;$$

Chiziqli kongruent generatorning kirish parametri s=0 bo'lsa, ya'ni

$x_{i+1}=(ax_i)\bmod N,\quad i=0,1,2,3,\ldots$; bo'lsa, bu generator chiziqli multiplikativ generator deyiladi.

1-isbot. Ushbu $x_{i+1}=(ax_i+s)\bmod N,\quad i=0,1,2,3,\ldots$; rekurent formula bilan aniqlangan psevdotasodifiy ketma-ketlik maksimal N davrga ega bo'lishi uchun quyidagi:

1) s va N -o'zaro tub sonlar, ya'ni EKUB(s,N)=1;

2) r soni N sonining bo'luvchisi va a-1 soni r soniga karrali;

3) N soni 4 karrali bo'lsa, a-1 soni ham 4 ga karrali;

shartlarning bajarilishi zarur va yetarli [5].

Quyida esa chiziqli va multiplikativ generatorlarning maksimal davrga ega bo'lishi bilan bog'liq ayrim xossalar keltirib o'tiladi [5].

Bevosita hisoblash bilan ishonch hosil qilish mumkinki, ushbu $x_{i+1}=(ax_i+s)\bmod N,\quad i=0,1,2,3,\ldots$; tenglik bilan aniqlanuvchi ketma-ketlik umumiy hadi uchun :

$$x_t=\left(a^t x_0+\frac{a^t-1}{a-1}c\right)\bmod N,\quad t\geq 1$$

formula o'rinli.

Multiplikativ va chiziqli va kongruent generatorlarning kamchiligi shundaki, PTKK biror bigrammasini $(z_1;z_2)$: $z_1=x_t$, $z_2=x_{t+1}$, bilgan holda, uning boshqa tashkil qiluvchilarini topish imkoniyati mavjud . Haqiqatan ham, ketma-ketlikni barcha tashkil qiluvchi qiymatlari $z_2 = a\,z_1 + c - k\,N$, k=0,1,… chiziqlar oilasida yotadi.

Chiziqli kongruyent generator hisoblangan Fishman va More generatorida kiruvchi parametr z_0 quyidagicha olingan [16]:

z_0=*23482349.*

PTSKK esa quyidagi amal orqali hisoblanadi: $z_{i+1}=a*z_i mod(2^{31}-1)$. Bu yerda a holat funksiyasi.

Chiziqsiz kongurent generatorlar

Kirish parametrlari:

N – chekli maydon xarakteristikasini ifodalovchi son;

d, a va s - o'zgarmas musbat butun sonlar, x_0 – boshlang'ich qiymat;

Ketma-ketlikni tashkil etuvchi chiqish qiymatlari :

$x_{i+1}=(dx_i^2+ax_i+c)\text{mod}N$, bu yerda $i=0,1,2,\ldots$.

Bu generator kvadratik generator deb ham ataladi.

2-isboti. Kvadratik generatorlar hosil qilgan PTKK o'zining $T_{\max} = N$ maksimal davriga ega bo'lishi uchun quyidagi shartlarning:

1. s va N –o'zaro tub sonlar;

2. d, a-1 -sonlari biror p –tub songa karrali bo'lib, bu p – soni N ning bo'luvchisi;

3. d – juft son bo'lib,

$$d = \begin{cases} (a-1)\bmod 4, \text{ агар } N \text{ сони } 4 \text{ га каррали бўлса;} \\ (a-1)\bmod 2, \text{ агар } N \text{ сони } 2 \text{ га каррали бўлса;} \end{cases}$$

4. Agarda N soni 9 ga karrali bo'lsa, u holda $d \bmod 9 = 0$, yoki $d \bmod 9 = 1$ va $cd \bmod 9 = 6$; bajarilishi zarur va yetarli.

Shuningdek $N=2^q$ bo'lsa, maksimal davrni ta'minlash uchun d-toq bo'lishi va $a=(d+1)\text{mod}4$ bo'lishi yetarlidir.

Kvadratik kongruyent generator $x_{i+1}=x_i^2 mod\ p(i \geq 0)$ uchun 512 bit uzunlikka ega bo'lgan p va x_0 parametrlar quyidagicha olinishi tavsiya etiladi [21]:

p = 987b6a6bf2c56a97291c445409920032499f9ee7ad128301b5d0
254aa1a9633fdbd378d40149f1e23a13849f3d45992f5c4c6b7104
099bc301f6005f9d8115e1;

x_0 = 3844506a9456c564b8b8538e0cc15aff46c95e69600f084f0657c24
01b3c244734b62ea9bb95be4923b9b7e84eeaf1a224894ef0328d44bc3e
b3e983644da3f5.

Kirish bit uzunligi 512 bit bo'lgan kubik kongruyent generator($x_{i+1}=x_1^3 mod 2^{512}$) uchun yesa kirish bitini

x_0= 7844506a9456c564b8b8538e0cc15aff46c95e69600f084f0657c2401b3c2 44734b62ea9bb95be4923b9b7e84eeaf1a224894ef0328d44bc3eb3e983644 da3f5

ko'rinishda tanlash maqsadga muofiqdir.

Chiziqli va multiplikativ kongruent generatorlar kabi chiziqsiz generatorlar ham kiptotahlil usuliga bardoshsiz.

Siljitish registrlariga asoslangan generatorlar

Xozirgi paytgacha taklif etilgan va muvaffaqiyatli ravishda ishlatilib kelinayotgan uzluksiz shifrlash algoritmlarining asosini siljitish registrlari yoki aniq qilib aytganda chiziqli teskari bog'lanishli siljitish registrlari tashkil qiladi. Bunday teskari bog'lanishli siljitish registrlari Fibbonachi registrlari yoki Galua registrlari ham deb ataladi. Bu xildagi uzluksiz shifrlash algoritmlarining ommaviy qo'llanilishiga ikki hil sababni ko'rsatish mumkin:

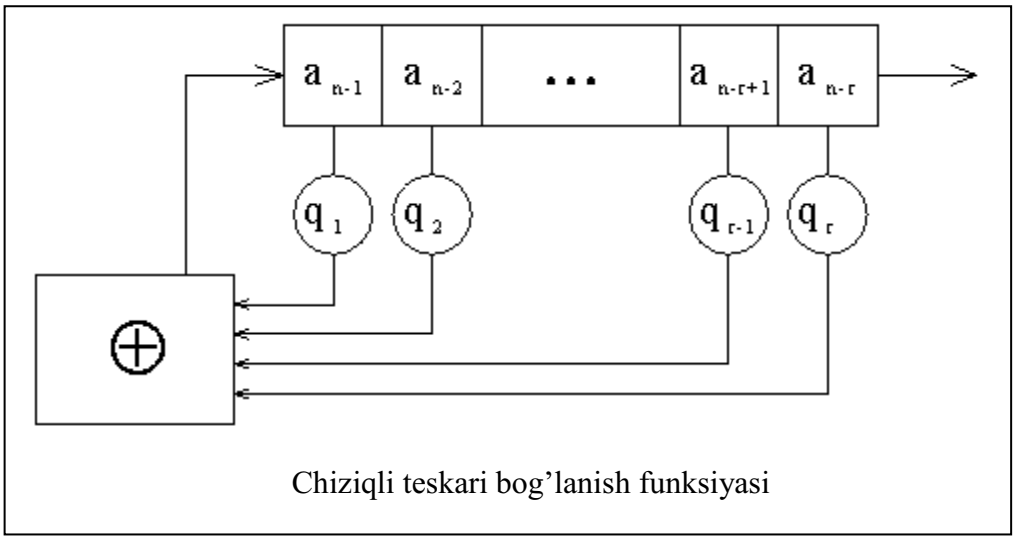
1. Teskari bog'lanishli siljitish registrlariga asoslangan generatorlar hosil qilgan ketma-ketliklar yaxshi tasodifiylik statistik xarakteristikalarini beradi;
2. Siljitish registrlariga asoslangan generatorlarning xususiyatlarini tahlil qilish oson.

Radioelektron elementlarning keng qo'llanilishi boshlangandan keyingi davr uchun siljitish registrlariga asoslangan uzluksiz shifrlash algoritmlari xarbiy sohadagi kriptografik vositalar tizimlarining asosi bo'lib xizmat qilmoqda.

Teskari bog'lanishli siljitish registrlari o'z navbatida chiziqli teskari bog'lanishli va chiziqsiz teskari bog'lanishli siljitish registrlariga bo'linadi.

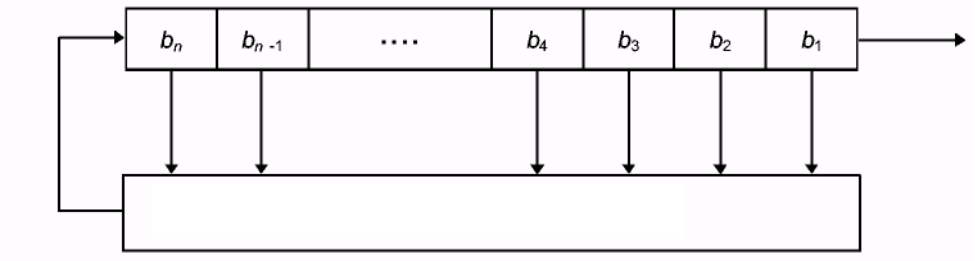
Siljitish registrlariga asoslangan generatorlar ikki qismdan tashkil topgan: birinchi qism bu siljitish registri bo'lsa, ikkinchi qismi teskari bog'lanish funksiyasidir. Siljitish registrlariga asoslangan algoritmlarning dasturiy yoki apparat-dasturiy jihatdan qo'llanishlari qulay va samaralidir.

Quyida siljitish registrlari turlarining funksional sxemalari keltirilgan.



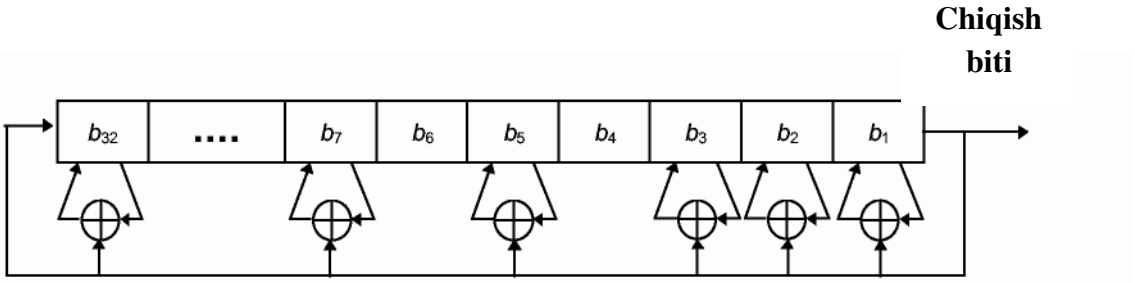
1.2-rasm. Teskari bog'lanishli siljitish registrining umumiy ko'rinishi

Amalda apparat-texnik qurilmalarni yaratishda qulay va tez ishlashni ta'minlash uchun mikroprosessorning registrlari (yacheykalari) soni bilan siltish registrlari (yacheykalari) soni teng qilib olinadi. IBM kompaniyasi tomonidan ishlab chiqilayotgan Intel prosessorlari 64 razryadli registrlarda ishlaganligi sababli dasturiy ta'minotda siljitish registrlariga asoslangan algoritmda registr uzunligini 64 ga teng yoki unga karrali qilib olish maqsadga muvofiqdir. Siljitish registrlarining dastlabki holati va registrlardan teskari bog'lanish funksiyasiga chiqishlar to'g'ri tanlanganda hosil qilingan ketma-ketlik davri maksimal bo'ladi. Siljitish registrlarining ikkinchi qismi bu teskari bog'lanish funksiyasidir. Teskari bog'lanish funksiyasi har bir taktda registrning ko'phad bilan ifodalanuvchi o'rinlaridagi bitlar qiymatini XOR amali bilan qo'shib, hosil bo'lgan qiymatni registrning eng katta razryadi o'rniga siljitish orqali kiritadi. Eng kichik razryad qiymati esa gamma ketma-ketlik elementi sifatida uzatiladi.

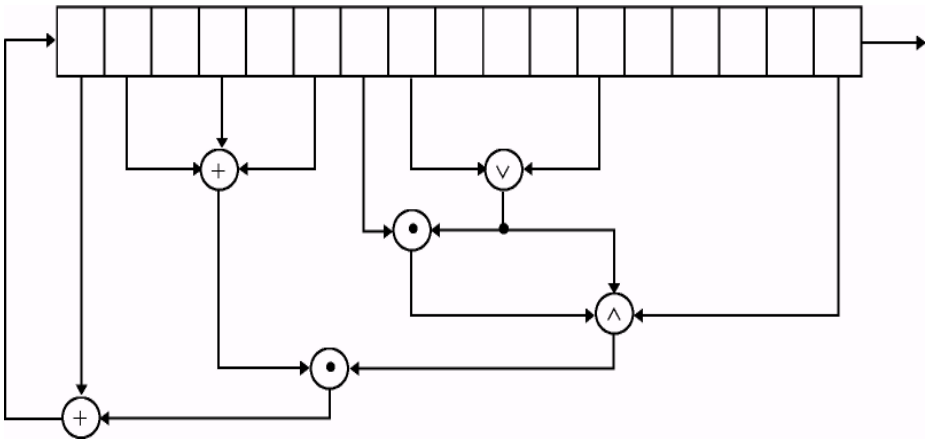


1.3-rasm. Chiziqli teskari bog'lanishli siljitish registri

Chiziqli teskari bog’lanishli siljitish registrlaridan biri bu Galua tuzilmasidir (konfigurasiyasidir). Galua tuzilmasida gamma ketma-ketlik elementlari sifatida uzatiladigan bit qiymati teskari bog’lanish funksiyasida ishtirok etadi. Chiqish biti registrning hamma bitiga XOR amali orqali qo’shiladi va registrning katta biti o’rniga siljitish orqali qo’yiladi. Ushbu registrdan chiquvchi ketma-ketlikning davri maksimal bo’lishi uchun teskari bog’lanish funkiyasi argumentlari registrning keltirilmaydigan ko’phad hosil qiluvchi xadlaridan olinishi lozim.



1.4-rasm. Galua konfigurasiyasiga asoslangan siljitish registri



1.5-rasm. Chiziqsiz teskari bog’lanishli siljitish registri

Ushbu chiziqsiz teskari bog’lanishli siljitish registrlarida teskari bog’lanish funksiyasi bir necha xil chiziqsiz akslantirishlarni qo’llash orqali amalga oshiriladi.

1.5-rasmdagi sxemada teskari bog’lanish funksiyasi XOR, AND, OR amallari orqali amalga oshirilgan. Xozirgacha chiziqsiz siljitish registrlariga asoslangan generatorlar hosil qilgan ketma-ketliklarni yetarlicha tahlil qiluvchi matematik usullar ishlab chiqilmagan. Shu sababli chiziqsiz teskari bog’lanishli

registrlar orqali amalga oshirilgan generatorlar bilan bog'liq bo'lgan quyidagi muammolarni keltirish mumkin:

— hosil qilingan psevdotasodifiy ketma-ketlikda tekis taqsimotdan chetlanish bo'lishi mumkin, ya'ni "0" va "1" belgilarning ishlab chiqilgan gamma ketma-ketlik bloklaridagi miqdori deyarli teng bo'lmasligi mumkin;

— ketma-ketlikning davri kutilganidan qisqa bo'lishi mumkin;

— ketma-ketlik davri har-xil boshlang'ich qiymatlar uchun har-xil bo'lishi mumkin, ya'ni ma'lum bir talabga javob beruvchi parametrlar tanlanganda har qanday ihtiyoriy boshlang'ich qiymat uchun generator hosil qilgan ketma-ketlik davri maksimal bo'ladi deb bo'lmaydi;

— hosil qilingan gamma ketma-ketlik tekshirish hisob-kitoblarisiz tasodifiyga o'xshab ko'rinishi mumkin, lekin registrning ma'lum bir holatidan so'ng, chiziqsizlik amalining mahsuli sifatida, keyingi hosil bo'lgan gamma ketma-ketlik elementlari faqat "0" yoki "1" lardan iborat bo'lib qolishi mumkin.

Chiziqsiz siljitish registrlarining kriptografik samarali tarafi bunday registrlarga asoslangan uzluksiz shifrlarning kriptografik tahlili usullari kamligidir.

Bir tomonli funksiyalarga asoslangan algoritmlar. PTSKK generatorlari yaratishda bir tomonli funksiyalar keng qo'llaniladi. Bir tomonli funksiyalarning xarakterli xossalaridan biri shundan iboratki, bu funksiyaning qiymatini argumentning berilgan qiymati bo'yicha hisoblash poliniomial-murakkablikkga ega bo'lib, funksiyaning berilgan qiymati bo'yicha bu qiymatga mos bo'lgan argument qiymatini hisoblash eksponensial murakkablikga ega yoki hisoblashning rasional algoritmi mavjud emas (yoki noma'lum).

FIPS-186 generatori

FIPS-186 generatoriAQSh milliy standarti sifatida qabul qilingan bo'lib, DSA elektron raqamli imzo algoritmi uchun maxfiy parametr va kalit ishlab chiqish uchun mo'ljallangan. Bu algoritmda bir tomonli funksiyalar sifatida DES va SHA-I algoritmlaridan foydalanilgan.

Kirish parametrlari: m – butun son, q -160 bitli tub son, b=160,
t = 67452301 EFCDAB89 98BADCFE 10325476 C3D2E1F0$_{16}$ – 160 bitli son;

$y_i = 0$;

s -boshlag'ich 160 bitli tasodifiy va mahfiy son.

Chiqish parametrlari:

$x_1, x_2, x_3, \ldots\ldots, x_m$

Algoritm qadamlari:

1. z_i=SHA-1(($s + y_i$)mod2^b);
2. $x_i = G(t, z_i)$;
3. $s = (1+s+x_i) \bmod 2^b$.

G: - bir tomonli funksiya sifatida ishlatilgan DES-shifrlash algoritmi

kirish: $t = t_0||t_1||t_2||t_3||t_4$; $z_i = z_0||z_1||z_2||z_3||z_4$;

chiqish: $w = w_0||w_1||w_2||w_3||w_4$.;

Funksiya:

1. $u_i = t_i \oplus z_i$, bu yerda i=0 dan 4 gacha o'zgaradi;
2. $b_1 = z_{(i+4)\bmod 5}$, $b_2 = z_{(i+3)\bmod 5}$, bu yerda $i = 0$ dan 4 gacha o'zgaradi;

$a_1 = u_2$, $a_2 = u_{(i+1)\bmod 5} \oplus u_{(i+4)\bmod 5}$;

$A = a_1||a_2$, $B = b_1||b_2$;

$y_i = E_B(A)$;

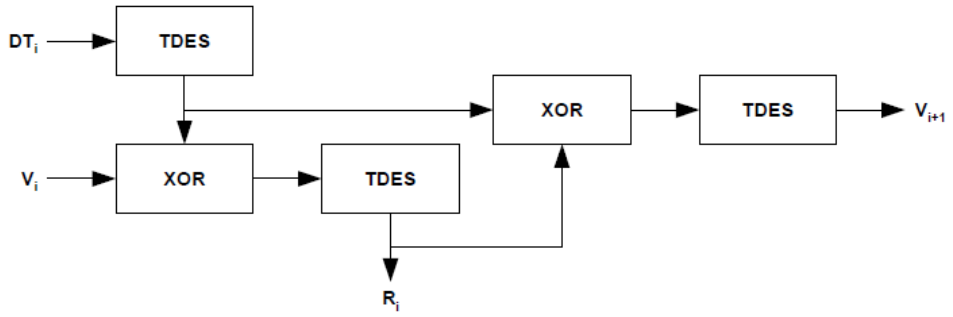
$y_i = L_i||R_i$;

3. $w_i = w_i + L_i \oplus R_{(i+2)\bmod 5} \oplus L_{(i+3)\bmod 5}$, bu yerda $i = 0$ dan 4 gacha o'zgaradi.
4. Natija: $G(t, z_i) = w_0||w_1||w_2||w_3||w_4$.

Ushbu kalit generatorining bardoshliligi unda ishlatilgan bir tomonlama funksiyalar DES va SHA-I ga bog'liq bo'ladi [5].

ANSI X9.17 generatori

ANSI X9.17 generatori AQShda psevdatasodifiy ketma-ketlik ishlab chiquvchi Milliy standart hisoblanib, FIPS (USA Federal Information Processing Standart) tarkibiga kiradi. Algoritmda bir -tomonlama funksiya sifatida uchlik DES ikkita K1, K2$\in$V64 kalit ishlatiladi: DESK1DESK2DESK1(64 bit) . Ushbu generator uchlik DES asosida ishlab chiqilgan bo'lib, uch marta uchlik DES muolajasini qo'llash orqali amalga oshiriladi.



1.6-rasm. ANSI X.9.17 generatori

Bu yerda:

TDES – uchlik DES shifrlash algoritmi bo'lib, shifrlashda ikkita K1, K2∈V64 kalitlardan foydalaniladi;

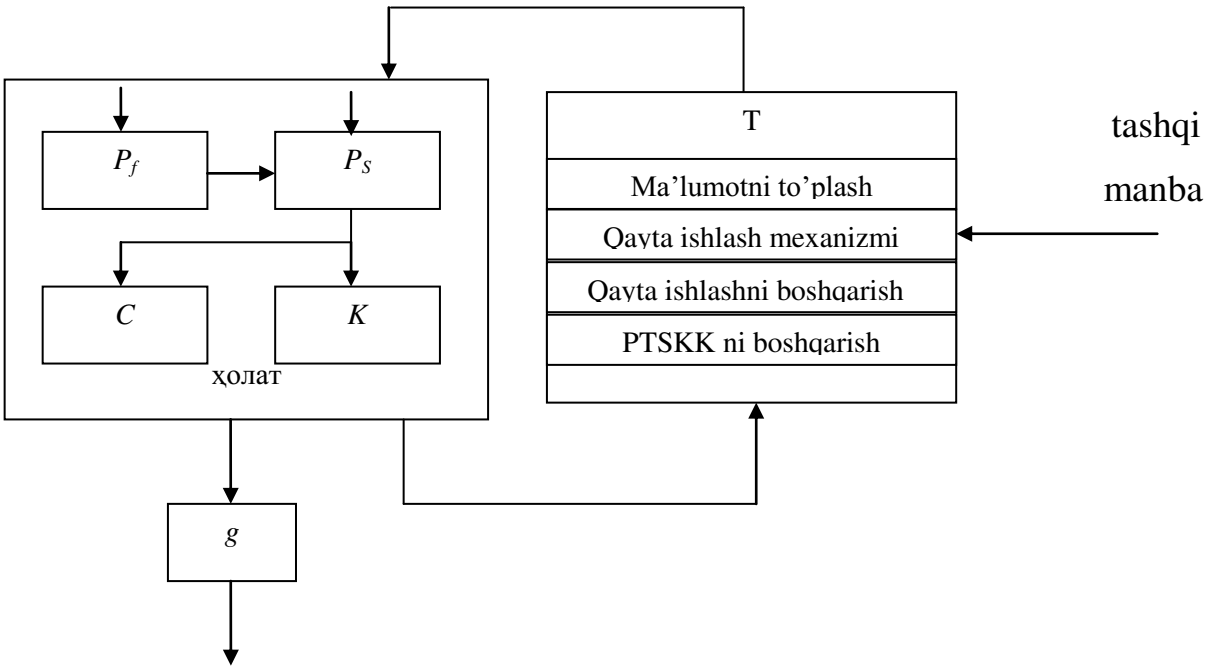
DT_i – kiruvchi vaqt parametri;

V_i - kiruvchi 64-bitli maxfiy kalit bo'lib, u quyidagi quyidagicha hosil qilinadi: V_{i+1}=TDES(TDES(DT_i)XOR(R_i));

R_i – chiquvchi psevdotasodifiy ketma-ketlik bo'lib quyidagicha hosil qilinadi: R_i=TDES(TDES(DT_i)XOR(V_i));

Yarrow-160 generatori

Yarrow-160 psevdatasodifiy ketma-ketlik ishlab chiqaruvchi generatori Kelsi, Shnayer va Fergyuson tomonidan taklif qilingan. Bu yerda uchlik DES va SHA-1 xeshlash algoritmi ishlatilgan.



1.7-rasm. Yarrow algoritmining umumiy strukturasi

Ushbu algoritmda kiruvchi tasodifiy ma'lumotlar ikki, tezkor qism R_f va sekin yig'iluvchi R_s qismlarga bo'linib, xeshlanadi. Kiruvchi ma'lumotlar haqiqiy tasodiyif bo'lgan ma'lumotlar manbalaridan olinadi. Yarrow algoritmining to'rtta tashkil etuvchisi yuqorida berilgan.

Yarrow kalit generatorida xesh funksiya va blokli shifrlashdan foydalanilganligi sababli, kriptohujumlarga bardoshli hisoblanadi. Yarrow-160 algoritmida xesh funksiya sifatida SHA-1 (Secure Hash Algorithm) va blokli shifrlash algoritmi sifatida esa uchlik -DES (Triple Data Encryption Standard) foydalanilgan.

Yuqoridagi simmetrik shifrlash tizimlarida bardoshli kalitlarni ishlab chiqaruvchi PTSKK generatorlari ko'rib chiqildi. Ko'rib o'tilgan generatorlar bardoshliligi tarkibidagi matematik amallar orqali belgilanadi. 1.2 - jadvalda esa psevdotasodifiy sonlar generatorining tarkibidagi amallar, kalit uzunligi va kriptobardoshligi xususyatlari bo'yicha tahlili ko'rsatilgan.

1.2- jadval

PTSKK generatorlari xususiyatlari

Algoritm nomi	**Kalit uzunligi (bit)**	**Tarkibidagi amallar**	**Kripto-bardoshligi**	**Xususyatlari**
Kongruent generatorlar	<64	Ko'paytirish, mod N	$<2^{64}$, kichik	Parametrlar tanlash murakkab, tezligi past
Rekkurent generatorlar	<64	Ko'paytirish, mod N	$<2^{64}$, kichik	Parametrlar tanlash murakkab, tezligi past
Gifford algoritmi	64	Siljitish registri	2^{64}, kichik	Takrorlanmas davri kichik
RC-4, ISAAC	2048 bitgacha	Mod256, o'rin almashtirish	2^{1700}, yuqori	Tezligi yuqori, patentga yega
SEAL, WAKE	128	O'nga surish, XOR	2^{160}, o'rta	Tezligi yuqori
A5, Geffe	64	Siljitish registrlari kombinatsiyasi	2^{64}, kichik	Tezligi yuqori

RSA, BBS, Shamir, Blyum-Mikali	1024-2048	Katta sonlarni darajaga oshirish	2^{1024}, yuqori	Tezligi past, faqat kalit ishlab chiqish uchun
ANSI X9.17 FIPS-186 YARROW-160	64-160	3DES, 2DES, SHA-1 algoritmlari	2^{128}, o'rta	Tezligi past, faqat kalit ishlab chiqish uchun

Ushbu generatorlarda qo'llaniluvchi hesh algoritmlarining asosiy hossalarini keltirish mumkin.

1.3-jadval

Hesh algoritmlar

	Xeshlanadigan matn uzunligi	**Kirish blokining uzunligi**	**Xesh qiymat uzunligi**	**Har bir blokni xeshlash qadamlari soni**
GOST R 34.11-94	Ixtiyoriy	256	256	19
MD 2	Ixtiyoriy	512	128	1598
MD 4	Ixtiyoriy	512	128	72
MD 5	Ixtiyoriy	512	128	88
SHA-1	$<2^{64}$	512	160	80
SHA-256	$<2^{64}$	512	256	64
SHA-384	$<2^{128}$	1024	384	80
SHA-512	$<2^{128}$	1024	512	80
STB 1176.1 – 99	Ixtiyoriy	256	$142 \leq L \leq 256$	77
O'z DSt 1106 : 2006	Ixtiyoriy	128, 256	128, 256	16b+74, 16b+46, Bu yerda b-bloklar soni

1.3. Kriptografik kalitlarning qo'llanilish sohalari

E-token yoki USB kalitlar asosida autentifikatsiyalash tizimi – hozirda parollar asosida autentifikatsiyalash keng tarqalgan ya'ni 60% foydalanuvchilar

foydalanadi. Lekin uning xavfsizlikni ta’minlash imkoniyati unchalik ham yuqori emas, shuning uchun USB kalitlar aosida autentifikatsiyalash keng tarqalmoqda.

1.8-rasm. eToken yoki USB kalit

USB kalitlarni bir ko’rinishi sifatida E-Tokenni keltirsak bo’ladi. U apparatura va dasturlar orqali amalda oshirilib autentifikatsiyalovchi tizimimda ya’ni, USB qurilmada electron raqamli imzo saqlanadi. Uning ko’rinishi quyidagicha bo’lishi mumkin: eToken PRO - USB-kalit.

Ular bir martalik va ko’p martalik kalit generatsiya qilinishi bilan farqlanadi. Masalan: Aladdin firmasi tomonidan tayyorlangan 32k va 64k versiyali USB qurilmalari bir martalik kalit asosida ishlaydi ya’ni har bir foydalanishda yangi kalit hosil qilib turadi.

Uni amalga oshirish uchun dastur yoki apparatura va eToken NG-OTP - gibrid USB-kalit kerak bo’ladi.

Xozirda Alladin firmasi tomonidan ishlab chiqilgan smart kartalar xuddi eTokenlar bajargan ishlarni xam amalga oshiryapti.

Lekin muammoni bir tarafi hal bo’lgani bilan ikkinchi tarafi shundaki, smart kartalarni ishlatish uchun kompyuterda yana boshqa qurilma zarur lekin eToken uchun shart emas.

Bundan tashqari eTokenlar faqat kalitlarni emas ixtiyoriy maxfiy ma’lumotlar, sertifikatlar va boshqa ma’lumotlarni xavfsiz saqlashi mumkin.

Agar eTokenda tizimga kiruvchi PIN(Personal Identification Number) saqalngan bo’lsa, u %systemroot%\system32\etcpass.ini. faylidagi ma’lumotni

o’zgartirib qo’yadi. Va tizimga kirishda faqat shu eToken ichidagi parol orqali kiriladi.

Parol – foydalanuvchi hamda uning axborot almashinuvidagi sherigi biladigan narsa. O`zaro autentifikatsiya uchun foydalanuvchi va uning sherigi o`rtasida parol almashinishi mumkin. Plastik karta va smart-karta egasini autentifikatsiyasida shaxsiy identifikatsiya nomeri PIN sinalgan usul hisoblanadi. PIN – kodning mahfiy qiymati faqat karta egasiga ma`lum bo`lishi shart.

Dinamik – (bir martalik) parol - bir marta ishlatilganidan so`ng boshqa umuman ishlatilmaydigan parol. Amalda odatda doimiy parolga yoki tayanch iboroga asoslanuvchi muntazam o`zgarib turuvchi qiymat ishlatiladi.

**“So`rov-javob” tizimi** - taraflarning biri noyob va oldindan bilib bo`lmaydigan “so`rov” qiymatini ikkinchi tarafga jo`natish orqali autentifikatsiyani boshlab beradi, ikkinchi taraf esa so`rov va sir yordamida hisoblangan javobni jo`natadi. Ikkala tarafga bitta sir ma`lum bo`lgani sababli, birinchi taraf ikkinchi taraf javobini to`g`riligini tekshirishi mumkin.

“So`rov-javob” mexanizmi quyidagicha.Agarb foydalanuvchi A foydalanuvchi V dan oladigan xabari yolg`on emasligiga ishonch xosil qilishni istasa, u foydalanuvchi V uchun yuboradigan xabarga oldindan bilib bo`lmaydigan element – X so`rovini (masalan, qandaydir tasodifiy sonni) qo`shadi. Foydalanuvchi V javob berishda bu amal ustida ma`lum amalni (masalan, qandaydir f(X) funktsiyani hisoblash) bajarishi lozim. Buni oldindan bajarib bo`lmaydi, chunki so`rovda qanday tasodifiy son X kelishi foydalanuvchi Vga ma`lum emas. Foydalanuvchi V harakati natijasini olgan foydalanuvchi A foydalanuvchi V ning xaqiqiy ekanligiga ishonch xosil qilishi mumkin. Ushbu usulning kamchiligi - so`rov va javob o`rtasidagi qonuniyatni aniqlash mumkinligi.

Sertifikatlar va raqamli imzolar - agar autentifikatsiya uchun sertifikatlar ishlatilsa, bu sertifikatlarda raqamli imzoning ishlatilishi talab etiladi. Sertifikatlar foydalanuvchi tashkilotining mas`ul shaxsi,

sertifikatlar serveri yoki tashqi ishonchli tashkilot tomonidan beriladi.

Internet doirasida ochiq kalit sertifikatlarini tarqatish uchun ochiq kalitlarni boshqaruvchi qator tijorat infrastrukturalari PKI (Public Key Infrastrusture) paydo bo`ldi. Foydalanuvchilar turli daraja sertifikatlarini olishlari mumkin.

Autentifikatsiya jaryonlarini ta`minlanuvchi xavfsizlik darajasi bo`yicha ham turkumlash mumkin. Ushbu yondashishga binoan autentifikatsiya jarayonlari quyidagi turlarga bo`linadi:

- parollar va raqamli sertifikatlardan foydalanuvchi autentifikatsiya;
- kriptografik usullar va vositalar asosidagi qatiy autentifikatsiya;
- nullik bilim bilan isbotlash xususiyatiga ega bo`lgan autentifikatsiya jarayonlari (protokollari);
- foydalanuvchilarni biometrik autentifikatsiyasi.

Xavfsizlik nuqtai nazaridan yuqorida keltirilganlarning har biri o`ziga xos masalalarni echishga imkon beradi. SHu sababli autentifikatsiya jarayonlari va protokollari amalda faol ishlatiladi. SHu bilan bir qatorda ta`kidlash lozimki, nullik bilim bilan isbotlash xususiyatiga ega bo`lgan autentifikatsiyaga qiziqish amaliy xarakterga nisbatan ko`proq nazariy xarakterga ega.

Balkim,yaqin kelajakdaulardan axborot almashinuvini himoyalashda faol foydalanishlari mumkin.

Foydalanuvchilarni biometrik parametrlari. Oxirgi vaqtda insonning fiziologik parametrlari va xarakteristikalarini, xulqining xususiyatlarini o`lchash orqali foydalanuvchini ishonchli autentifikatsiyalashga imkon beruvchi biometrik autentifikatsiyalash keng tarqalmoqda.

Biometrik autentifikatsiyalash usullari an`anaviy usullarga nisbatan quyidagi afzalliklarga ega:

— biometrik alomatlarning noyobligi tufayli autentifikatsiyalashning ishonchlilik darajasi yuqori;

— biometrik alomatlarning sog`lom shaxsdan ajratib bo`lmasligi;

— biometrik alomatlarni soxtalashtirishning qiyinligi.

Foydalanuvchini autentifikatsiyalashda faol ishlatiladigan biometrik algoritmlar quyidagilar:

— barmoq izlari;

— qo`l panjasining geometrik shakli;

— yuzning shakli va o`lchamlari;

— ovoz xususiyatlari;

— ko`z yoyi va to`r pardasining naqshi.

**Yuzning tuzilishi va ovoz bo`yicha.** Bu tizimlar arzonligi tufayli eng foydalanuvchan hisoblanadilar, chunki aksariyat zamonaviy kompyuterlar video va audeo vositalariga ega. Bu sinf tizimlari telekommunikatsiya tarmoqlarida masofadagi foydalanuvchi sub`ektni identifikatsiyalash uchun ishlatiladi. YUz tuzilishini skanerlash texnologiyasi boshqa biometrik texnologiyalar yaroqsiz bo`lgan ilovalar uchun to`g`ri keladi. Bu holda shaxsni identifikatsiyalash va verifikatsiyalash uchun ko`z, burun va lab xususiyatlari ishlatiladi. Yuz tuzilishini aniqlovchi qurilmalarni ishlab chiqaruvchilar foydalanuvchini identifikatsiyalashda hususiy matematik algoritmlardan foydalanadilar.

**Ovoz bo`yicha autentifikatsiyalash tizimlari.** Bu tizimlar arzonligi tufayli foydalanuvchan hisoblanadilar. Hususan ularni ko`pgina shaxsiy kompyuterlar standart komplektidagi uskuna (masalan mikrofonlar) bilan birga o`rnatish mumkin. Ovoz bo`yicha autentifikatsiyalash tizimlari har bir odamga noyob bo`lgan balandligi, modulyatsiyasi va tovush chastotasi kabi ovoz xususiyatlariga asoslanadi. Ovozni aniqlash nutqni aniqlashdan farqlanadi. CHunki nutqni aniqlovchi texnologiya abonent so`zini izoxlasa, ovozni aniqlash texnologiyasi so`zlovchining shaxsini tasdiqlaydi.

Barmoq izlari yordamida biometrik autentifikatsiyalash. Biometrik tizimlarning aksariyati identifikatsiyalash parametri sifatida barmoq izlaridan foydalanadi (autentifikatsiyaning daktiloskopik tizimi). Bu tizimlar XX asrning

60-yillarida kirib kelgan bo'lib, jinoyat ishlari bilan shug'ullanuvchi organlar foydalangan.

Bunday tizimlar sodda va qulay, autentifikatsiyalashning yuqori ishonchliligiga ega. Bunday tizimlarning keng tarqalishiga asosiy sabab barmoq izlari bo`yicha katta ma`lumotlar ba`zasining mavjudligidir. Bunday tizimlardan dunyoda asosan politsiya, turli davlat va ba`zi bank tashkilotlari foydalanadi.

Autentifikatsiyaning daktiloskopik tizimi quyidagicha ishlaydi. Avval foydalanuvchi ro`yxatga olinadi. Odatda, skanerda barmoqning turli xolatlarida skanerlashning bir necha variant amalga oshiriladi. Tabiiyki, namunalar bir–biridan biroz farqlanadi va qandaydir umumlashtirilgan namuna, «pasport» shakllantirilishi talab etiladi. Natijalar autentifikatsiyaning ma`lumotlar bazasida xotirlanadi. Autentifikatsiyalashda skanerlangan barmoq izi ma`lumotlar bazasidagi «pasportlar» bilan taqqoslanadi.

Xozzirgi kunda 3xil barmoq izlarini olish texnologiyasi mavjud:

- optik nurlar orqali barmoq izini olish(FTIR ya'ni optik qurilmalar);
- yarim o'tkazgichlar orqali(termoskanerlar);
- ultratovushlar;

bularning barchasi bir xil prinsipda ishlaydi ya'ni bir xil matematik algoritmlar va olingan natijalarni bir biri bilan solishtirish.

Barmoq izlarining skanerlari. Barmoq izlarini skanerlovchi an`anaviy qurilmalarda asosiy element sifatida barmoqning xarakterli rasmini yozuvchi kichkina optik kamera ishlatiladi.

1.9-rasm. Java Card

Java Card yuqori darajada xavfsiz(high - security), multi – ilovali ochiq platformadir. Java Card Smart Card texnologiyasini keying bosqichi hisoblanadi. Java Cardni Smart Card deb atasak ham bo`ladi. Java Card yuqori dasturlash tili hisoblangan Java dasturlash tili yordamida programmalashtiriladi. Tabiiyki bu dasturiy ta’minotni osonlashtiradi. Quyida 1.9-rasmda Java Card tasvirlangan.

Java dasturlash tilini tanlashni bir qancha afzalliklari bor:

1. Ixchamlilik – hisoblash tizimlarida bu juda mashhur va bu loyiha obyektlariga mos.
2. Kam kodlilik – obyektlar mijoz mashinasiga “remote host”dan yuklab olishga mo’ljallangan. Kichkina kod bu ishni osonlashtiradi.
3. Mukammalik – smart card uchun mukammal hisoblanadi.

Java Card SIM karta va bankomatlarda keng foydalaniladi. Birinchi Java Card 1996 – yil, 29 – oktabrda Schlumberger firmasi tomonidan ishlab chiqarilib ommaga namoyish qilingan. Hozirda bu kompaniya Gemalto deb ataladi.

Java Card ning qo’llanilish sohasi:

- SIM karta sifatida telefonlarda qo’llaniladi.Shu o’rinda aytib o’tish kerakki ko’proq simsiz tarmoqlarda foydalaniladi.
- Plastik (Moliyaviy) karta sifatida hozirda keng qo’llanilib kelinmoqda.
- Identifikatsiya kartasi sifatida.
- Mantiqiy va fizik ruxsat beruvchi (muassasa resurslariga) karta sifdatida ham qo’llanilib kelinmoqda.
- Transport biletlari sifatida qo’llaniladi.

Java Card texnologiyasi ilovalarni xavfsiz muhit bilan ta’minlaydi qaysiki smart card va boshqa qurilmalarda juda kichkina xotira bilan ishlaydigan va qayta ishlash imkoniyatiga ega bo’lagan. Murakkab ilovalar bitta Java Cadrni o’zida ishlashi mumkin, ya’ni o’zgarishlar foydalanuvchiga berilgandan keyin ham qo’shish mumkin. Java Card hozirda smart kartalar ichida o’zining imkoniyatlari ko’pligi, xavfsizlik darajasi yuqoriligi va murakkab ilovalarni ham qo’llab quvvatlashi bilan birinchi o’rinda kelmoqda. Java Card yangi mahsulotlarga ishlov berish uchun o’z ichiga mukammal qurilmalarni jamlagan.

Java Card boshida smart kartadagi konfidensial ma’lumotlarni xavfsizligini ta’minlash maqsadida ishlab chiqarilgan. Xavfsizlik bu texnologiyani turli nuqtai nazarlar orqali belgilaydi:

1. Ma’lumotlar inkapsulyatsiyasi (Data Encapsulation). Ma’lumotlar ilovalar ichida saqlanadi va Java Card ilovalar izolyatsiyalangan muhitda (JCVM Java Card Virtual Machine) ishlaydi va boshqarish sistemasi qurilmaga ruxsatga ega bo`lishi mumkin.

2. Aplet ekrani (Applet Firewall). Boshqalardan farqli o’laroq Java Virtual Machine, Java Card Virtual Machine har biri konfidensial ma’lumotlarni boshqaradigan bir qancha ilovalarni boshqaradi.Ilovalarni bir- biridan applet firewall bilan ajratilishiga sabab bir appletdan ikkinchisiga ruxsatsiz kirishni taqiqlash.

3. Kriptografiya (Cryptography). Simmetrik DES, Triple DES, AES va assimmetrik RSA, elliptic curve cryptography algoritmlarini madadlaydi (qo’llab quvvatlaydi). Shuningdek bir qancha kriptografik xizmatlarni qo’llab quvvatlaydi. Bular: elektron raqamli imzo, Kalitlar gneratori va kalitlar ayriboshlash.

4. Applet. Applet bu holat mashinasi hisoblanadi qaysiki so’rov va javob mexanizmi asosida ishlaydi ya’ni kiruvchi buyruqlarga javob beradi, ma’lumot yuborish yoki interfeys qurilmasiga javob status so’zlari yuborish orqali.

Hozirda Java Card juda ham keng miqyosida qo`llanilmoqda Bunga bir necha sabablarni keltirishimiz mumkin, masalan: Java kartaning yuqori xavfsizlik darajasiga egali, murakkab ilovalarni o’zida ishlata olishi, bir vaqtning o’zida bir nechta ilovani ishlata olishi va foydalanuvchilar uchun qulayligi va hokazo. Buning isboti sifatida 2001- yildan beri sotilgan 3 milliard kartaning 700 millioni Java Card ligini keltirsak ham shuning o’zi kifoya. Java cardning ham har xil parametrli turlari mavjud. Paramertlari 1K RAM, 16K EEPROM va 24K ROM bo’lgan Java Cardda to`liq funksiyali Java tizimini joriy qilib bo’lmaydi ya’ni Java dasturlash tilining to’liq xususiyatlaridan foydalanib bo’lmaydi. Ammo to’liq bo’lmagan versiyasini qo’llashimiz mumkin, ya’ni qisman foydalanishimiz mumkin va bu haqda quyida gapirib o’tiladi.

Unda 2 ta asosiy qism bor:

1. Java Card Virtual Machine (JVM) – Programmist oson aloqa qila oladigan mantiqiy mashinani aniqlaydi.

2. Java Card Runtime Environment (JCRE) – bu aslini olganda sodda qilib aytadigan bo'lsak Java Cardning operatsion tizimidir.

Java Card Virtual Machine (JCVM). Java Cardga Java dasturlash tilida programma yozilishini yuqorida aytib o'tgan edik. Lekin Java dasturlash tilining hamma imkoniyatlaridan foydalanib bo'lmaydi ya'ni Java Cardning konfiguratsiyasidan kelib chiqib aytadigan bo'lsak Java dasturlash tilining hamma imkoniyatlarini namoyon qilishga kuchi yetmaydi . Chunki Java Card Java dasturlash tilining barcha imkoniyatlarini madadlamaydi (ya'ni qo'llab quvvatlamaydi) va Java tili obyektga yo'naltirilgan dasturlash tili hisoblanadi. Lekin Java Card Java dasturlash tilining qisman imkoniyatlarini qo'llab quvvatlay oladi. Quyida Java Cardni Java dasturlash tilining qaysi xususiyatlarini qo'llab quvvatlashi va qaysilarini qo'llab quvvatlamasligini keltirib o'tamiz.

Demak qo'llab quvvatlaydiganlari:

— boolean, byte, short, int.

— Bir o'lchovli array.

— Java class, package, interface, exception lar.

— Javaning object-oriented ga xos xususiyatlaridan: inheritance, virtual methods, overloading va dynamic object creation, access scope, va binding rules

Qo'llab quvvatlamaydiganlari:

— double, float, long, character, string.

— Multi-dimensional arrays

— Dynamic Class Loading

— Security Manager

— Garbage Collection

— Threads

— Java Card Virtual Machine 2 qismga bo'linadi:

— Converter – Java Card Virtual Machine ning off – card qismida ishlaydi.

Interpreter – Java Card Virtual Machine ning on – card qismida ishlaydi.

Xulosa qilib shuni aytish mumkinki xar bir autentifikatsiyalash vositalarining o'ziga yarasha kamchilik va avzalliklari mavjud. Quyidagi jadvalda ularning kamchili va avzalliklari keltirilgan.

1.4-jadval.

Generator qo'llanilishi turlarining asosiy afzalliklari va kamchiliklari

Generator qo'llanilishi turi	**Kamchiligi**	**Avfzalligi**
USB kalitlar	Begona shaxslar qo'liga tushishi va qimmatligi	Bir martalik parollar generatsiyasi mavjudligi va kompyuterga USB port orqali foydalanish mumkinligi
Smart kartalar	USB kalitlarga o'xshash va kompyuterda foydalanish uchun alohida qurilma talab qiladi	O'lchami kichik onlayn xolatda yo'qotib qo'ysa avtomatik o'chirish imkoniyati
Parol tizimlari	Uni buzish yoki topish oson, eslab qolish qiyin, sedan ham chiqib qolishi mumkin	Xarajatsiz va o'zgartirish va foydalanish oson
Biometrik tizimlar	Vaqt o'tishi bilan va vaziyatga qarab o'zgarishi, qurilmalar qimmatligi.	Faqat insonning o'ziga xos va usiz amalga oshirib bo'maydi.
So'rov-javob tizimi	O'rtada turib ma'lumotlarni ushlab qolishi mumkin	Qo'shimcha qurilmalarsiz va yetarlicha aniqlik mavjud
Java Card tizimlari	Foydalanuvchi faqat karta bo'lgandagina ulanishi mumkin. Agar u kartani yo'qotsa tizimga ulanishdan mahrum bo'ladi	Katta tezlikda va xavfsizlik vositalarini o'zida biriktirgan holda ishlaydi

2.Asosiy qism. Kalitlarni xavfsiz saqlash va yaratish tizimi

2.1. E-Tokenlardan kalitlarni yaratish va saqlashda qo’llash

Hozirgi kunda USB kalitlarda kalitlarni saqlash va ulardan foydalanish keng tarqalmoqda.

2.1-rasm. E-tokenlarning umumiy ko’rinishlari

USB kalitlarni bir ko’rinishi sifatida E-Tokenni keltirsak bo’ladi. U apparatura va dasturlar orqali amalda oshirilib autentifikatsiyalovchi tizimimda ya’ni, USB qurilmada electron raqamli imzo saqlanadi. Uning ko’rinishi quyidagicha bo’lishi mumkin: eToken PRO - USB-kalit.

Ular bir martalik va ko’p martalik kalit generatsiya qilinishi bilan farqlanadi. Masalan: Aladdin firmasi tomonidan tayyorlangan 32k va 64k versiyali USB qurilmalari bir martalik kalit asosida ishlaydi ya’ni har bir foydalanishda yangi kalit hosil qilib turadi.

Uni amalga oshirish uchun dastur yoki apparatura va eToken NG-OTP - gibrid USB-kalit kerak bo’ladi .

Xozzirda Alladin firmasi tomonidan ishlab chiqilgan smart kartalar xuddi eTokenlar bajargan ishlarni xam amalga oshiryapti.

Lekin muammoni bir tarafi hal bo’lgani bilan ikkinchi tarafi shundaki, smart kartalarni ishlatish uchun kompyuterda yana boshqa qurilma zarur lekin eToken uchun shart emas.

Bundan tashqari eTokenlar faqat kalitlarni emas ixtiyoriy maxfiy ma’lumotlar, sertifikatlar va boshqa ma’lumotlarni xavfsiz saqlashi mumkin.

Agar eTokenda tizimga kiruvchi PIN(Personal Identification Number) saqalngan bo’lsa, u %systemroot%\system32\etcpass.ini. faylidagi ma’lumotni o’zgartirib qo’yadi. Va tizimga kirishda faqat shu eToken ichidagi parol orqali kiriladi.

E-TOKEN – bu smart kartalar va USB qurilmalar orqali kalitlarni olib yurish va undan foydalanish uchun ishlab chiqilgan.

Uning quyidagi ko’rinishlari mavjud:

— eToken PRO и eToken PRO (Java) — smart kartalar va USB kalitlarning umumiy vazifalarini 1 ta smart artada birlashtirilgan

— eToken GT — unda kalitlarni saqlashdan tashqari ma’lumotlarni saqlovchi qismi ham mavjud;

— eToken NG-FLASH и eToken NG-FLASH (Java) — bu yuqoridagi etokenning keying ko’rinishi bo’lib unda saqlanadigan ma’lumotlar xajmi yanada kengaytirilgan;

— eToken NG-OTP и eToken NG-OTP (Java) — bu bir martalik kalitlarni hosil qilib beradi;

— eToken PASS — OTP-bir martalik kalitlarni almashtirishni o’zi avtomatik amalga oshiradi;

— eToken Virtual — smart kartalarning dasturiy moduli;

— MobilePASS — bir martalik kalitlarni dasturiy apparati.

2.1-jadval

Quyida bir necha keng tarqalgan USB qurilmalarni klassifikatsiyasi keltirilgan

Ishlatilishi bo'yicha turlari	Modeli	Ko'rinishi
USB kalitlar	eToken GT eToken NG-FLASH eToken NG-FLASH (Java) eToken PRO eToken PRO (Java)	
Bir martalik parollar asosidagi USB	eToken NG-OTP eToken NG-OTP (Java)	
OTP-token	eToken PASS	
Smart karta	eToken PRO eToken PRO (Java)	

USB kalitlar quiyidagi OT larda ishlashi mumkin;

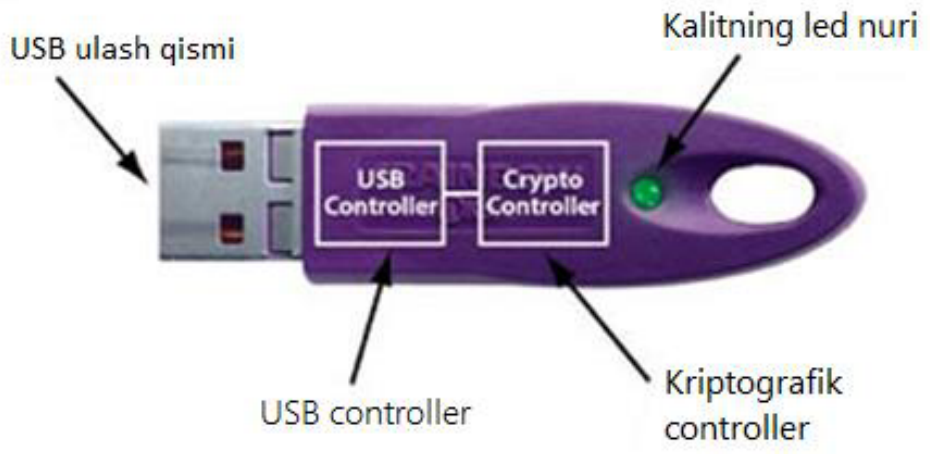
— BlackBerry;

— GNU/Linux;

— Mac OS;

— Microsoft Windows.

Quyida esa USB kalitlar qanday tuzilishga ega va qanday ishlashi bilan tanishib chiqamiz:



2.2-rasm. USB kalitning umumiy tuzilishi

Bunda USB kontrollerini keltirish mumkin. Ya'ni u orqali kompyuterning USB poritga ulanadi va etoken va port o'rtasida aloqa o'rnatiladi;

USB controller – kriptografik kodlash, USB konnektor va operatsiyalarni amalga oshirish bo'limi bilan aloqani ta'minlaydi;

Kriptografik controller – kriptografik amallarni bajarilishi va kalitlari almashish funksiyalarini bajaradi;

Kalitning led nuri – aloqa bo'layotgani yoki USB kalitning ishlayotgani yoki ishlamayotgnini aniqlaydi.

Bu USB kalitlar asosida autentifikatsiyalash texnologiyasida ikki faktorli autentifikatsiya usuli qo'llanlishini ko'ramiz.

eToken tizimini 4 ta turga bo'lishimiz mumkin:

1. Mexmon xotiraning ochiq maydonida obyektlarini ko'risih mumkin;

2.eToken bilan bog'liq bo'lgan umumiy ma'lumotlarni tizimning xotira maydonidan olish mimkin. eToken nomini, identifikatori va boshqa parametrlarini olish mimkin. Mexmon foydalanuvchi pin kodni bilishi shart emas;

3.Foydalanuvchilar yopiq, ochiq va xotiraning bo'sh xotirasidagi obyektlarni o'chirishi, o'zgartirishi va ko'rishi mumkin.

4.Pin kod va eTokenni to'gri o'zgartirishi lozim.

USB kalitlarni bir ko’rinishi sifatida E-Tokenni keltirsak bo’ladi. U apparatura va dasturlar orqali amalda oshirilib autentifikatsiyalovchi tizimimda ya’ni, USB qurilmada electron raqamli imzo saqlanadi. Uning ko’rinishi quyidagicha bo’lishi mumkin: eToken PRO - USB-kalit.

Ular bir martalik va ko’p martalik kalit generatsiya qilinishi bilan farqlanadi. Masalan: Aladdin firmasi tomonidan tayyorlangan 32k va 64k versiyali USB qurilmalari bir martalik kalit asosida ishlaydi ya’ni har bir foydalanishda yangi kalit hosil qilib turadi.

Uni amalga oshirish uchun dastur yoki apparatura va eToken NG-OTP - gibrid USB-kalit kerak bo’ladi .

Xozzirda Alladin firmasi tomonidan ishlab chiqilgan smart kartalar xuddi eTokenlar bajargan ishlarni xam amalga oshiryapti.

Lekin muammoni bir tarafi hal bo’lgani bilan ikkinchi tarafi shundaki, smart kartalarni ishlatish uchun kompyuterda yana boshqa qurilma zarur lekin eToken uchun shart emas.

Bundan tashqari eTokenlar faqat kalitlarni emas ixtiyoriy maxfiy ma’lumotlar, sertifikatlar va boshqa ma’lumotlarni xavfsiz saqlashi mumkin.

Agar eTokenda tizimga kiruvchi PIN(Personal Identification Number) saqalngan bo’lsa, u %systemroot%\system32\etcpass.ini. faylidagi ma’lumotni o’zgartirib qo’yadi. Va tizimga kirishda faqat shu eToken ichidagi parol orqali kiriladi.

O`zaro autentifikatsiya uchun foydalanuvchi va uning sherigi o`rtasida parol almashinishi mumkin. Plastik karta va smart-karta egasini autentifikatsiyasida shaxsiy identifikatsiya nomeri PIN sinalgan usul hisoblanadi. PIN – kodning mahfiy qiymati faqat karta egasiga ma`lum bo`lishi shart.

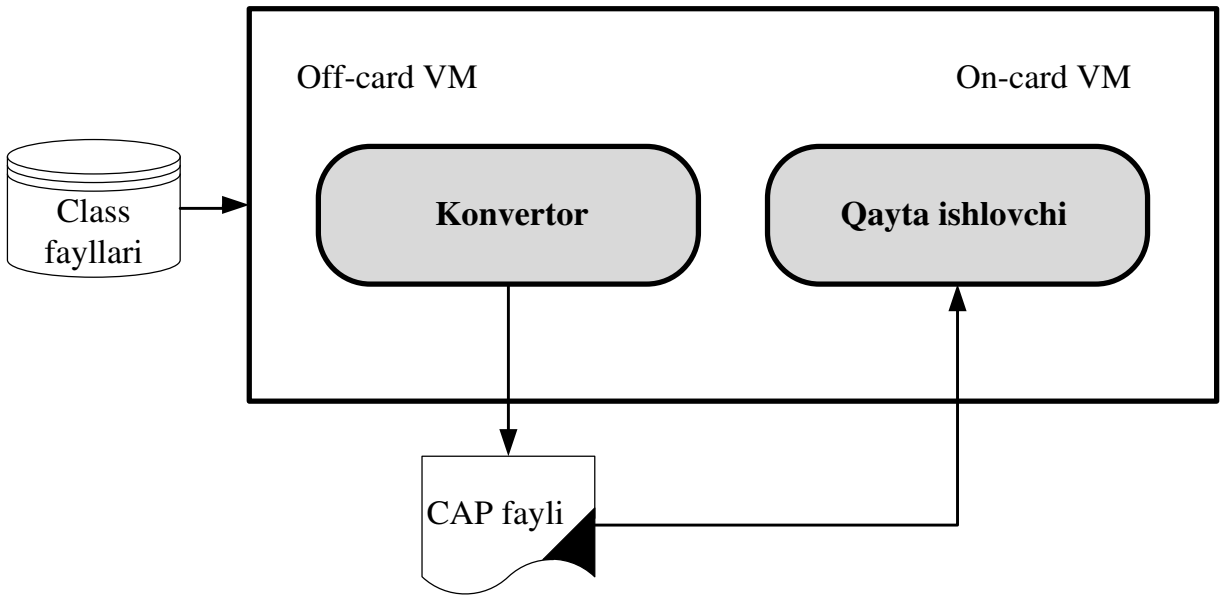
Dinamik – (bir martalik) parol - bir marta ishlatilganidan so`ng boshqa umuman ishlatilmaydigan parol. Amalda odatda doimiy parolga yoki tayanch iboroga asoslanuvchi muntazam o`zgarib turuvchi qiymat ishlatiladi.

2.2. Java Card virtual mashinalarida kalitlarni saqlash hususiyatlari

Java Card Virtual Machine 2 qismga bo'linadi:

— Converter – Java Card Virtual Machine ning off – card qismida ishlaydi.

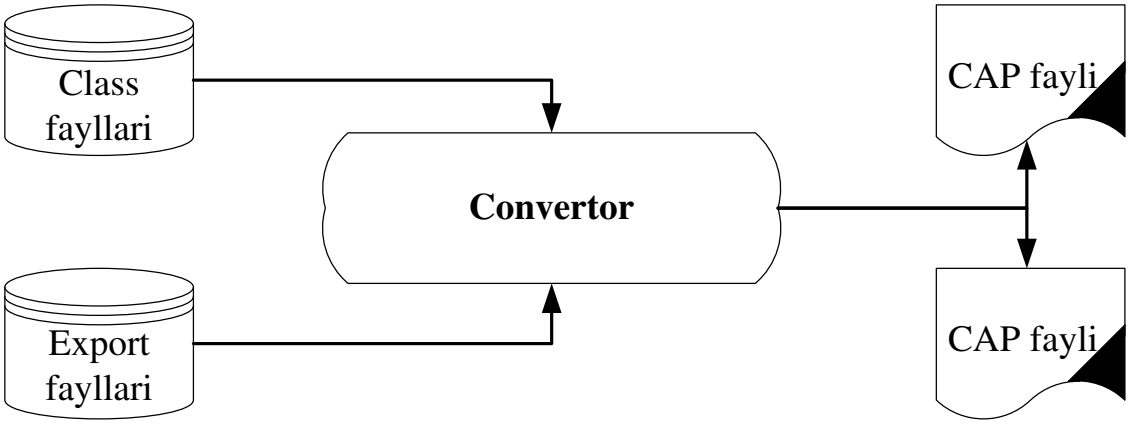
— Interpreter – Java Card Virtual Machine ning on – card qismida ishlaydi. (2.3 – rasmga qarang).



2.3 – rasm. Java Card Virtual Machine qismlari

Berilgan class fayllari va eksport qilingan fayllar Java kompilyatorning 3 – qismida ishlab chiqariladi. Konverter quyidagi vazifalarni bajaradi:

- Java Card standartiga nisbatan yo'l qo'yilgan xatoliklarni tekshiradi.
- Byte kodni to'g'rililigni tekshiradi.
- Statik o'zgaruvchilarni inisilizatsiya qiladi.
- Simvolli ulanishlarni ancha qulayroq formaga keltiradi, qaysiki bu Java Cardlarda ancha muvaffaqiyatli bo'ladigan.
- Class yuklashda va bog'lanish vaqtida ma'lumotlarni qulay olish orqali byte kodni optimizatsiyalashtiradi.
- Xotiradan joy ajratadi va class larni e'lon qilish uchun virtual mashina yarati.
- Interpretatorda ishlatib bo'lishi uchun binar CAP fayl yaratadi va faylni eksport qiladi.



2.4 – rasm. Konvertorning ishlash sxemasi

Konverter java cardning tashqi qismi hisoblanadi. Bu Java ilova foydalanuvchi kompyuterida ishlaydi. Konverter odatiy .class fayl (standart bayt kodli) ni maxsus formatli CAP(Converted applet) faylga o'giradi. CAP fayl "converted applet" degan ma'noni anglatadi va fayl formatlarining yangi turi hisoblanadi. Bu format Java Card uchun maxsus loyihalashtirilgan bo'lib, CAP fayl Java Card ichiga yuklanadi. Lekin Export faylda bunday bo'lmaydi. Export fayllarni C da sarlavha sifatida ko'rish mumkin. Barcha class paketlari uchun ular API(Application Programming Interface) jamiyat ma'lumotlarini saqlaydi. Ular maqsadlarni tekshirish va bog'lash uchun ishlatiladi, shuningdek foydalanuvchilarga applet ishlab chiqaruvchilar tomonidan taqsimlanishi mumkin. CAP fayl Java cardga on - chip va off - chip o'rnatgichlari orqali yuklanadi. On – chip installer CAP faylni java card xotirasiga yozadi, yana CAP faylni java carddagi allaqachon joylashgan class larga bog'laydi va Java Card ishlash muhitida foydalaniladigan biror bir ma'lumot strukturasini yaratadi va boshlang'ich qiymat beradi. Interpretator va CAP fayl o'rnatgichi funksiyalari o'rtasidagi bo'lak o'rnatishni amalga oshirish uchun kichkina interpretatorni va moslashuvchanlikni ta'minlaydi.

Interpretator CAP faylni qabul qilgandan keyin quyidagilarni o'z zimmasiga oladi.

— Bayt kodni bajarish

— Xotiradan joy ajratish va obyektlarni yaratishni nazorat qiladi.

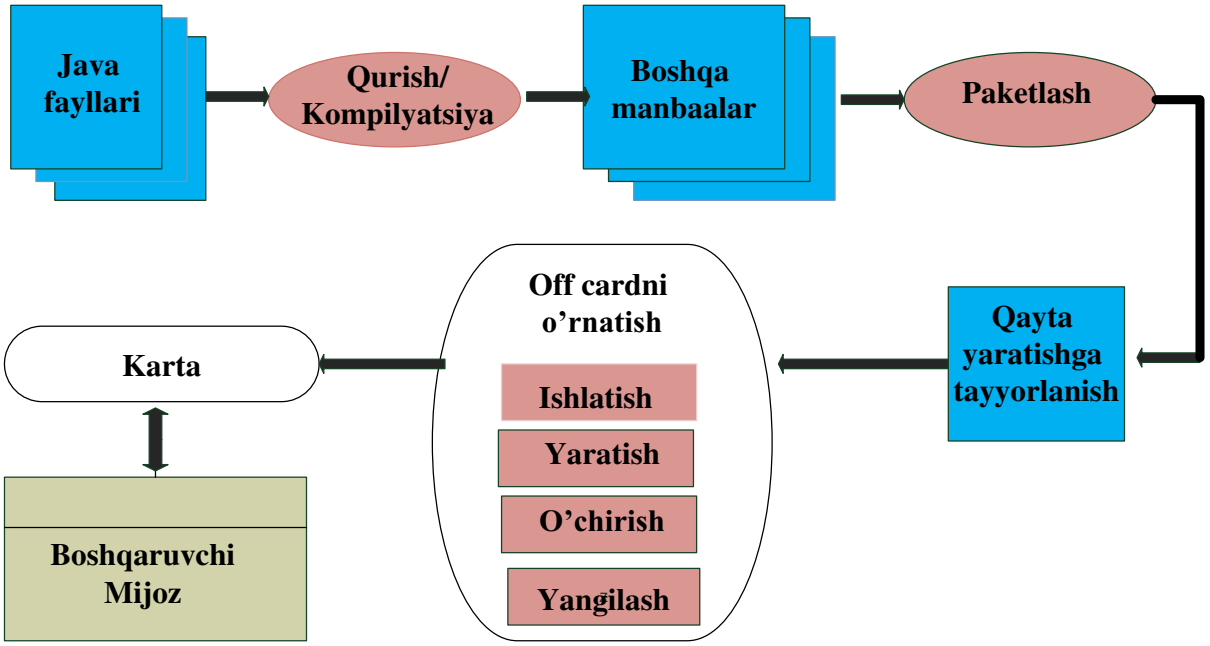
— Ishlash payti xavfsizlik ta’minlaydi.

Java Card Runtime Environment. Interpretator va Installer Java Cardda ishlaydigan sistemaning yagona tarkibiy qismi emas. Boshqa tarkibiy qismlar ham bor va ular bilan birgalikda Java Card Runtime Environment (JCRE) ni tashkil qilishadi.

Java Card Virtual Machine(JCVM) xotiradan joy ajratish, bayt kodni ishlatish, obyektlarni boshqarishini va xavfsizlikni ta’minlashni o’z zimmasiga oladi. Mahalliy (Java cardning o’zini rodnoy metodlari) metodlar past sathdagi kommunikatsiya protokollar, xotirani boshqarish, kriptografik amallarni va boshqalarni o’z zimmasiga oladi. Tizim klaslari operatsion tizim yadrosi vazifalarni bajaradi va buni bajarish uchun mahalliy metodlarga murojaat etadi. API(Application Programming Interface) klaslari zich joylashgan kutubxona hisoblanib, ular appletlarni osonroq yaratishni ta’minlayd. Bu dasturlovchi uchun juda ham qulay hisoblanadi. O’rnatgichning vazifalaridan biri appletlarni Java cardga yuklash. Boshqa maxsus kutubxonalar masalan: Open Platform JCRE(Java Card Runtime Environment) xizmatlarini kengaytiradi, xususan xavfsizlikka oid talablarini. JCRE Java Cardga fabrikada(ya’ni java card ishlab chiqarilayorganda paytda) yuklanadi va Java card ichida Java card qayta qurilmaguncha qoladi[14].

Qachonki Java card Card Accepting Devise (CAD Java card ishga tushiriladigan muhit, bunga terminallarni, Java SIM kartani qo’llab quvvatlaydigan telefonlarni, Card readerlarni, autentifikatsiyadan o’tkazuvchi qurilmalarni misol qilib keltirishimiz mumkin) ichiga joylashtirilsa u aktiv holatda bo’ladi va ma’lumotlar, programmalar EEPROM va ROMlardan tezkor xotira RAM ga ko’chirila boshlanadi. Operatsiya davomida(ya’ni ma’lumotlar, programmalar EEPROM va ROMlardan tezkor xotira RAM ga ko’chirilayotganda) saqlanishi kerak bo’lgan ma’lumotlar va obyektlarni nusxasi RAM dan EEPROM ga ko’chiriladi.Bu yerda EEPROM kompyuterning doimiy xotirasi bajaradigan ishni bajaradi va o’zidagi ma’lumotlarni o’chirib yubormaganimizgacha saqlaydi. EEPROM quvvat uzilgan vaqtda ma’lumotni saqlaydi qaysiki karta pitaniyadan uzilgan payti ma’lumotlar EEPROM turadi. Qachonki quvvat yo’qolsa card uyqu

rejimiga(hibernation) o'tadi. Java Cardga programma yozish qadamlarini umumiy qilib quyidagi 2.6- rasmda keltirishimiz mumkin.



2.5-rasm. Java Cardga dastur yozish qadamlari.

2.1- jadval.

Java Card turlari

Java card nomi	Model nomeri	Foydalanish sohasi	EEPROM	Java card haqidagi fikr
Java SIM	ELC - JS - 32	GSM, CDMA	32K	Sifatli
	ELC - JS - 64	GSM, CDMA	64K	Sifatli
	ELC - JS – 128	GSM, CDMA	128K	Sifatli
Java 3G	ELC - J3G – 64	3G	64K	Sifatli
	ELC - J3G- 128	3G	128K	Sifatli
Java EMV	ELC – JEMV	EMV2000	16K	Sifatli
Java PKI	ELC – JPK	CA	32K	Sifatli

Java SIM Card

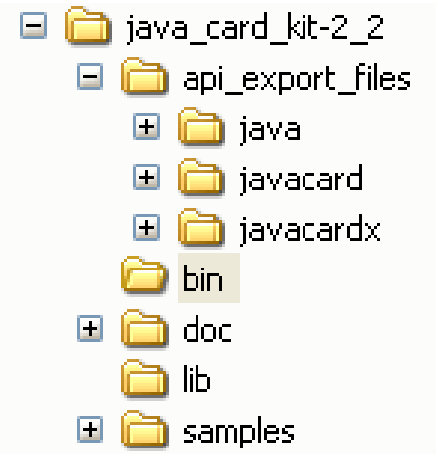
Java SIM (Subscriber Identity Module) card GSM tarmog'i uchun maxsus SIM karta hisoblanadi. Phease 2 + STK(SIM application Toolkit) kartalarning barcha funksiyalari qatorida SIM Java cardga STK ning ishlab chiqarilishi endilikda juda qulay va standart SIM kartalarga nisbatan kamroq vaqtni talab qiladi. Bu Java tili himoyasining kuchli tomonlari va ochiq platformaning o'zgachaligi bilan bog'liq. RFM(Remote Management File) Java SIM tayanchi, ma'lumotlarning xavfsizli, tezlikni oshirish va funksiyalarning nusxa ko'chirilishiga qarshi. Mahsulot juda xavfsiz, foydali, boshqarishda oddiy va har xil ilovalarga ko'maklashish uchun ko'pgina imkoniyatlarga ega. Java SIM kartalarning seriyalari shuningdek xotiraning har xil konfiguratsiyalari uchun qo'llanilsa bo'ladi.

Java SIM karta quyidagilarni o'z ichiga mujassamlaydi: GSM, PHACE2, STK barcha funksiyalarini va IOD(Information-On Demand) ning keng spektorini va STK ilovaning kitobi, SMS gruppa, Multi – IMCI telefon kitobining kengayishi, menyu tuzish, mobil bank ishi va boshqalar.RFM COMP128 – v1, v2, v3, v0, XOR o'xshash autentifikatsiyalash algoritmlarini madadlaydi. Operator tomonidan aniqlangan algoritmlar,shuningdek ACS xavfsizlik texnologiyasi unikal kartasida taminlangan bo'lishi mumkin. Autentifikatsiya hisoblagichi, anti nusxa ko'chirish va nusxa ko'chirish xususiyatlari instrumentlardan nusxa ko'chirib buzish uchun juda katta qiyinchilik tug'diradi. Har xil mashhur shifrlash algoritmlari va ma'lumotlarning butunligini saqlash mexanizmlari DES, 3DES, PBOK, va MAC bor.

Java SIM GSM operatorlari uchun mukammaldir. GSM da operatorlar ehtiyojining o'zaro ta'siri bilan Java SIM yuqori sifatli Sim kartalar bozorida standart bo'lib xizmat qilyapti. Java Powered logotipi Sim belgisi ishlashida Sim Aliance ta'siriga mos keladi, xuddi boshqa standartlarga o'xshab. Java Powered Sim logotipi yuqori unumdorlikka asoslangan, Java Card Virtual Machine ning 100% o'zi ishlab chiqargan. Java Powered Sim logotipi o'zining Sim logotipi bilan Virtual machine deb hisoblaydi. Natijada kuchli, sinab ko'rilgan mahsulot, Sim

card ning dasturiy ta’minotini 10 yillik ish malakasiga asos bo’ladi. Smard kartalar uchun logotiplar Java kartalarni kelajakdagi mahsulotlar uchun muhim texnologiya deb biladi va Java Powered Sim kartalar bozoridagi ulushini doimiy oshirib boradi. Java kartalarning butun litsenziyasi logotipi Java Card jamiyatining bo’lagi bo’lib, Java kartalar standartlarining kelajagini aniqlash jarayoni bo’lib xizmat qiladi. 3GPP T3 ning ishchi gruppasininh faoliyatiga qarab logotiplar Logos va boshqa ishlab chiqaruvchilar mahsuloti bilan birga keyingi o’zaro aloqalarni yaratib beradi. SIM logotiplari Java cardning ishlab chiqargan har qanday muhitida foydalansa bo’ladi.Java SIM Card quyidagi 2.12 – rasmda keltirilgan[18].

2.6 – rasm. Java SIM card



2.7 – rasm. Sun Java Card Development Kit ning strukturasi

Java SIM toolkit yordamida

— SIM cardga dastur;

— Telefonga SIM card application (klavishalardan, ekrandan foydalanish huquqini deradigan va hokazo) tuzish mumkin.

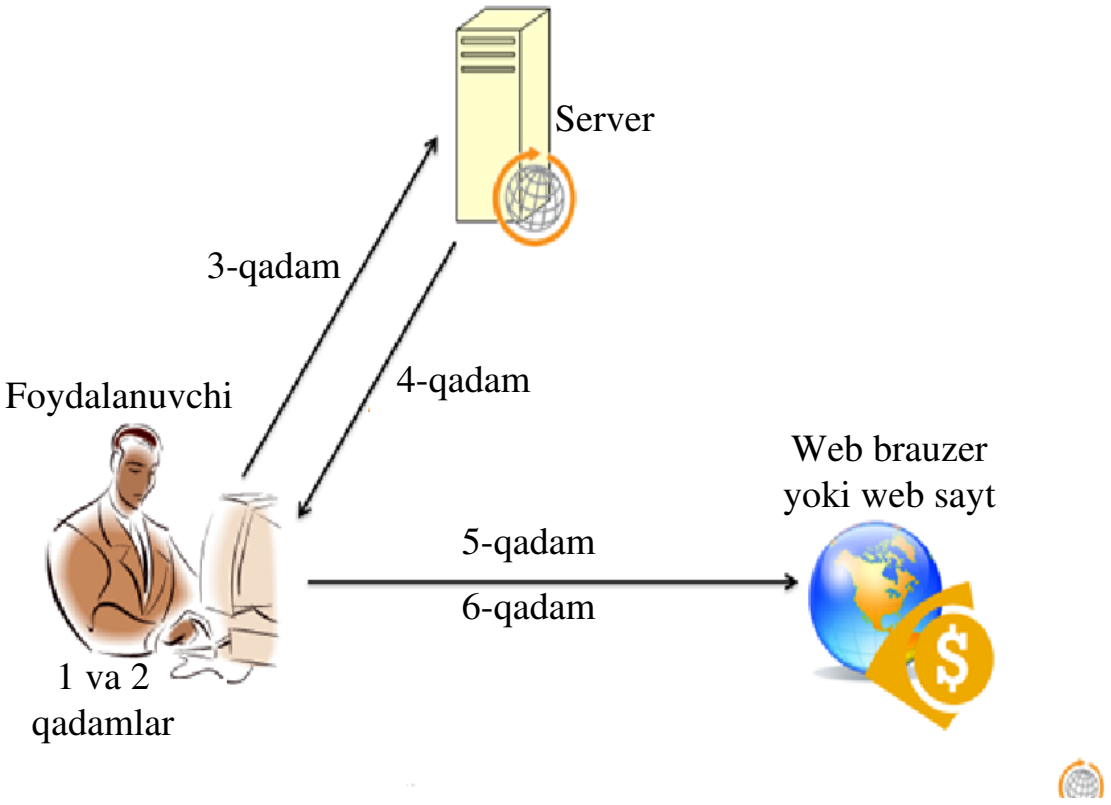
Maxsus protokol orqali SIM card telefon bilan munosobat o'rnatadi.

SIM cardga dastur yozishimiz uchun Sun Java Card Development Kit ni kompyuterimizga o'rnatamiz va editor sifatida NetBeans IDE dan foydalanamiz. Sun Java Card Development Kit ning strukturasi quyidagicha (2.7 - rasm).

2.3. USB kalitlarni kalitlarni generatsiyalash jarayonlariga tadbiqi

USB kalitlarni bir ko'rinishi sifatida E-Tokenni keltirsak bo'ladi. U apparatura va dasturlar orqali amalda oshirilib autentifikatsiyalovchi tizimimda ya'ni, USB qurilmada electron raqamli imzo saqlanadi. Uning ko'rinishi quyidagicha bo'lishi mumkin: eToken PRO - USB-kalit.

Ular bir martalik va ko'p martalik kalit generatsiya qilinishi bilan farqlanadi. Masalan: Aladdin firmasi tomonidan tayyorlangan 32k va 64k versiyali USB qurilmalari bir martalik kalit asosida ishlaydi ya'ni har bir foydalanishda yangi kalit hosil qilib turadi.



2.8-rasm. E-tokenning ishlash sxemasi

Uni amalga oshirish uchun dastur yoki apparatura va eToken NG-OTP - gibrid USB-kalit kerak bo'ladi.

eToken RTE 3.65 ni o'rnatish o'chirish quyidagicha amalga oshiriladi.

O'rnatish: eToken Run Time Environment 3.65 Setup ishga tushiriladi

Bunda eTokenlarning ishlashini ko'rishimiz mumkin:

Ushbu sxemaning ishlash sxemasi quyidagi qadamlardan iboratdir.

1. Tokkenga ulanish. Qurilmani komyuternig USB portiga ulaymiz.

2. CD-ROMdan dastur ishga tushadi.

3. Token o'zining ma'lumotlarini Serverga junatadi va serverdan javobni kutadi.

4. Xavfsizlik dasturlaridan biri Anywhere bundle dasturini qo'llab unga javob qaytaradi.

5. Web brauzer ishga tushadi va unga ulanishni talab qiladi va foydalanuvchi web brauzerdan foydalanish uchun ulanadi.

6. Saytda autentifikatsiyadan o'tadi va xavfsiz ishlashga ruxsat oladi.

Biosmart texnologiyasi orqali eTokenlar bilan bir qatorda biometrik apparat vositalar orqali ham autentifikatsiyani amalga oshirish mumkin.

Bunda esa Weigand yordamida biometrik autentifikatsiyadan o'tamiz va u muvafaqqiyatli amalga oshirilsa, so'ng eToken orqali tekshiruv amalga oshiriladi.

2.9-rasm orqali biz biometrik va USB kalitli tizimni bir biriga bog'liqligini ko'ramiz. Bunda:

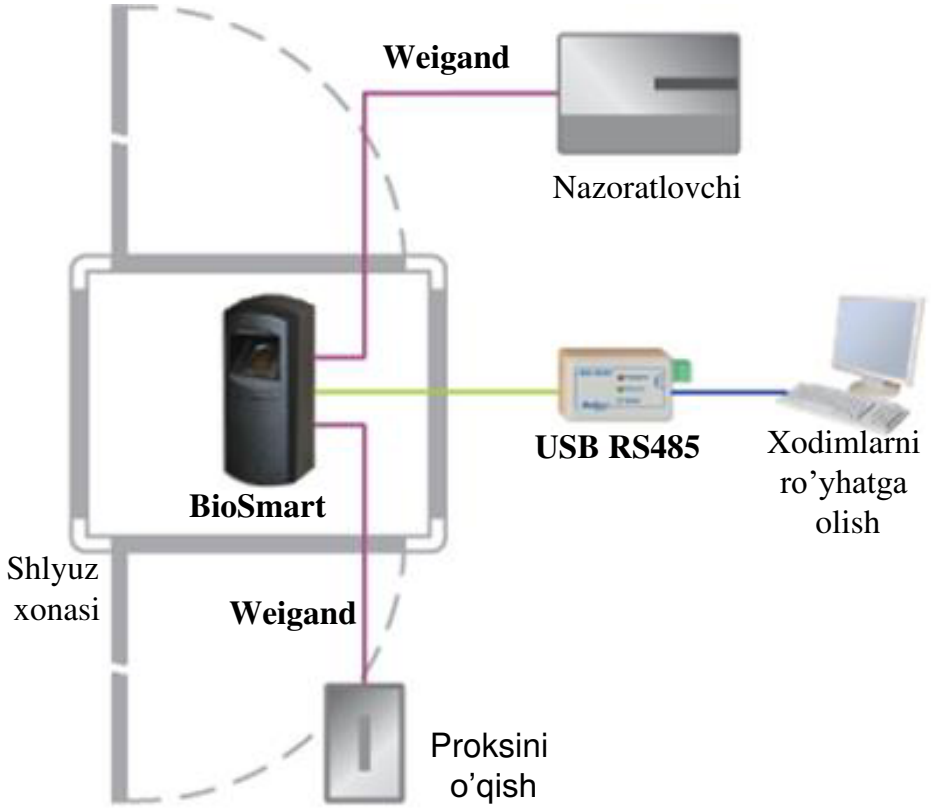
— kirishda biotizim amalga oshirildi;

— biotekshiruvdan o'tgandan keyin elektron qulf orqali xonasiga kiradi va ishini amalga oshiradi;

— serverning ma'lumotlar bazasi, buxgalteriya hisoboti va byuro uchun USB va biotizim orqali amalga oshiriladi.

Umuman olgan biotizim va USB kalitlarni ishlashi juda oson. Chunki 10 000 barmoq/sek tezlik bilan amalga oshiriladi.

USB tizimlar bilan ishlaganda GPRS-RS485 magistralidan foydalanish maqsadga muvofiq. Chunki serverga ulanish va bir vaqtning o’zida turli xizmatlardan foydalanish arzon xisoblanadi.



2.9-rasm. USB kalit va biotizimning birgalikda ishlash sxemasi

Java SIM cardga parol generatsiya qilib beradigan dastur yozishda quyidagi qadamlar bajariladi:

— Parol generatsiya qilish uchun algoritmini ishlab chiqish.

— Java SIM kartadagi programma ishga tushiriladi.

— Programma hosil qilib bergan parolni saqlab olamiz.

— Java Mobil application ishga tushiriladi.

— Autentifikatsiyadan o’tish sahifasidagi parol kiritiladigan joyga hosil qilingan parol kiritiladi.

— Autentifikatisiya tugmasi bosish orqali autentifikatsiyadan o’tamiz.

Serverda quyidagi amallar bajariladi:

— Yuborilgan parol qabul qilib olinadi.

— Bu parol kelishib olingan parol uzunligi bilan tekshiriladi.

— Keyingi qadamda parol kelishib olingan algoritm bo'yicha generatsiya qilinganligi yoki qilinmaganligi tekshiriladi va shu orqali bu foydalanuvchi aniq biz aloqa qilmoqchi bo'lgan foydalanuvchi ekanligini bilib olamiz.

— Tizimda bir martali parollardan foydalanayotganligimiz sababli bu parol orqali oldin autentifikatsiyadan o'tilgan yoki o'tilmaganligi bazadagi ma'lumotlar yordamida tekshiriladi.

— Oxirgi qadamda barcha tekshiruvlardan ijobiy o'tgan shaxs tizimga kirishga ruxsat etiladi.

Ushbu dasturlarda bir martali parollardan fodalanildi. Bir martali parollar asosida autentifikatsiyalashning usullaridan diri ikki tomon o'zaro kelishgan holda yaratilgan maxsus algoritmdan foydalanadi. Autentifikatsiyadan o'tishda mijoz shu algoritm bo'yicha parol hosil qiladi, serverga yuboradi va bu parol kelishib olingan algoritm bo'yicha tuzilganligi tekshiriladi. Bu dasturda yuqoridagiga amal qilingan holda quyidagi algoritm ishlab chiqildi:

1. Kelishilgan holda alphabet = "A,E<GL.MN)P;SZ!*bUcYde~Bfge} VjkTl-m|Jno>p[qrstQuv_Rw:xCyDza/? YXKy {@E= (^F$WO" to'plami yaratildi.

2. Tasodifiy sonlar generatoridan foydalanib 5 xonali son yaratiladi.

*Math.random() * 100000*

3. To'plamdagi shu 5 xonali songa teng bo'lgan o'rindagi simvol ajratib olinadi. Bu jarayon 5 marta davom etadi. Ajratib olingan simvollar shu sikl davomida to'lamdan o'chirib turiladi.

```
for (int i = 0; i < 5; i++) {
k = (int) ((n - k) % length);
if (k == 0) { k = length;}
password = password + alphabet.charAt(k - 1);
alphabet = alphabet.substring(0, (k - 1)) + alphabet.substring(k, length);
k = length - k;
length--;}
```

4. Hosil bo’lgan 5ta simvol va 5 xonali son aralashtirilgan holda 10 xonali parol hosil qilanadi.

```
for (int i = 0; i < password.length(); i++) {
j += 1;
pass = pass + password.charAt(i) + str.substring(l, j);
l = j;}
```

Autentifikatsiyadan o’tishda serverda quyidagi amallar bajariladi:

1. Qabul qilib olingan paroldan simvollar va tasodifiy hosil qilib olingan son ajratib olinadi.

2. Ajratib olingan 5 xonali tasofiy son yordamida alphabet = "A,E<GL. MN)P;SZ!*bUcYde~Bfge} VjkTl-m|Jno>p[qrstQuv_Rw:xCyDza/? YXKy {@E= (^F$WO" to’plamidagi shu 5 xonali songa teng bo’lgan o’rindagi simvol ajratib olinadi. Bu jarayon ham 5 marta davom etadi. Ajratib olingan simvollar shu sikl davomida to’lamdan o’chirib turiladi.

3. Hosil bo’lgan 5ta simvol va ajratib olingan 5 xonali son aralashtirilgan holda 10 xonali parol hosil qilanadi.

4. Mijozdan qabul qilib olingan parol P_1 hamda serverda hosil qilingan P_2 parol tengligi tekshirib ko’riladi. Agar foydalanuchi tomonidan yuborilgan parol serverda hosil qilingan parolga teng bo’lsa bu paroldan oldin foydalanilgan yoki yo’qligi tekshiriladi. Barcha tekshiruvdan muvaffaqiyatli o’tgandan keyin ma’lumot almashinuvi boshlanadi.

Bu dasturda bir martali parollardan fodalanildi. Quyida bir martali parollar haqida ma’lumot berib o’tamiz:

Bir martali parol. Bir martali parol(One Time Password) bu autentifikatsiyaning bitta seansi uchun haqiqiy bo’lgan parol, ya’ni bu paroldan faqat bir marta fodalanish mumkin. Bir martali parolning ishlashi ma’lum bir vaqt oralig’i bilan cheklangan bo’lishi ham mumkin.Bir martali parol static paroldan ustun hisoblanadi. Bir martali parolning statik paroldan ustunligi shundaki ushbu parolni qayta ishlatib bo’maydi. Shunday qilib autentifikatsiyaning bir marta muvaffaqiyatli seansidan ma’lumotni(masalan parolni ko’chirib olish) o’g’irlagan

jinoyatchi himoyalangan tizimga kirish uchun ko’chirib olingan bu paroldan qayta foydalana olmaydi. Bir martali paroldan foydalanish autentifikatsiya uchun ishlatiladigan aloqa kanaliga aralashishga asoslangan hujumlardan(men of the middle) himoya qila olmaydi. Bir martali parollar ham “men of the middle” hujumiga qarshilik qila olmaydi. Bu hujumni oldini olish uchun boshqa metodlardan foydalanish kerak.

Bir martali parol tuzish va tarqatish usullari. Bir martali parol tuzish algoritmlarida odatda tasodifiy sonlar ishlatiladi. Bu kerak. Chunki oldingi parollarning asoslarini bilib keying parollarni osongina taxmin qilish mumkin. Bir martali parollarning aniq algoritmlari detallarda kuchli farq qiladi. Bir martali parollarni tuzish usullari quyida keltirib o’tilgan:

— Yangi parol tuzish uchun oldingi parollar asosida matematik algoritmlardan foydalanish orqali(parollar zanjirini tashkil etadi va aniq bir tartibda ishlatiladi).

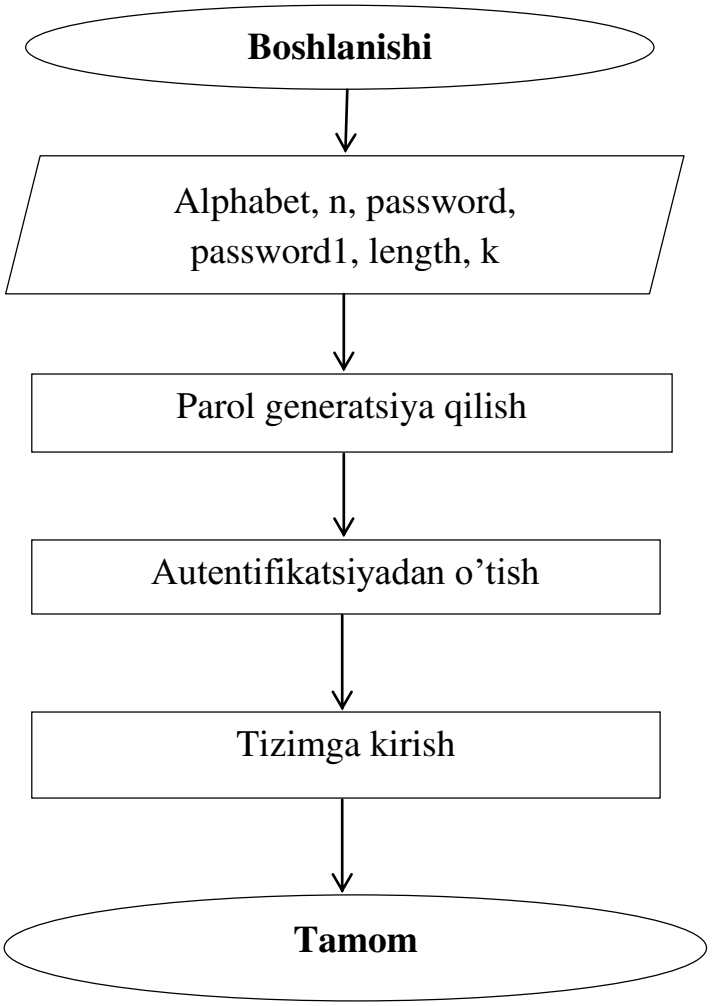
— Server va mijoz o’rtasidagi vaqtning sinxronligiga asoslangan ta’minlovchi parollar(qisqa vaqt ichida haqiqiy bo’lgan parollar).

— Matematik parollarni ishlatuvchi so’rovga asoslangan parollar(masalan taxminiy sonlar, server tomonidan tanlanadigan yoki kiruvchi xabarning bir qismi)

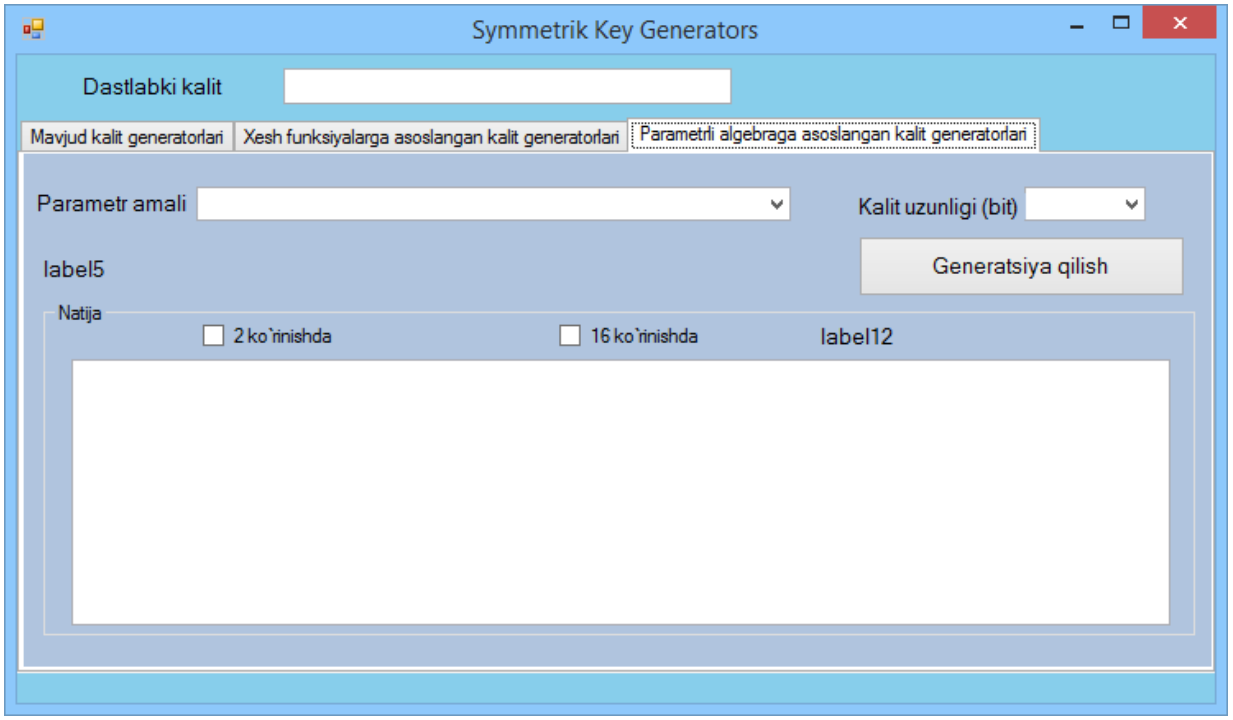
Foydalanuvchiga keyingi parolni aytish uchun har xil usullar mavjud. Ba’zi bir tizimlar foydalanuvchi o’zi bilan olib yuradigan, bir martali parol yaratadigan va kichkina ekranga chiqaradigan maxsus elektron tokenlardan fodalanadi.Boshqa tizimlar foydalanuvchi mobil telefonda ishga tushiradigan programmadan tashkil topgan.Yana boshqa tizimlar serverdagi bir martali parollarni generatsiyalaydi va keyin boshqa kanallardan foydalanib foydalanuvchiga jo’natadi, masalan sms orqali.

2.4. Kriptografik algoritmlardan foydalanib kalit yaratish dasturini ishlab chiqish

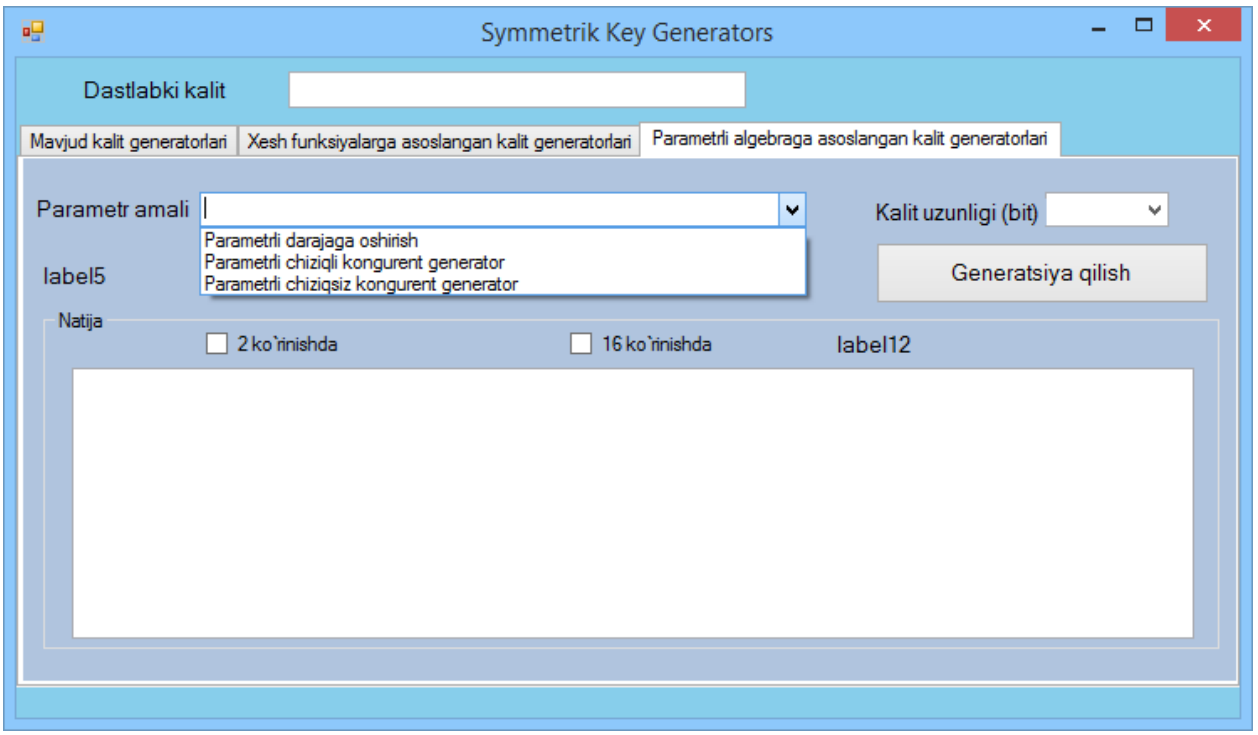
Dasturni parol generatsiya qilish va autentifikatsiyadan o’tish qismi blok sxemasi quyidagi keltirilgan(1 – blok sxema).



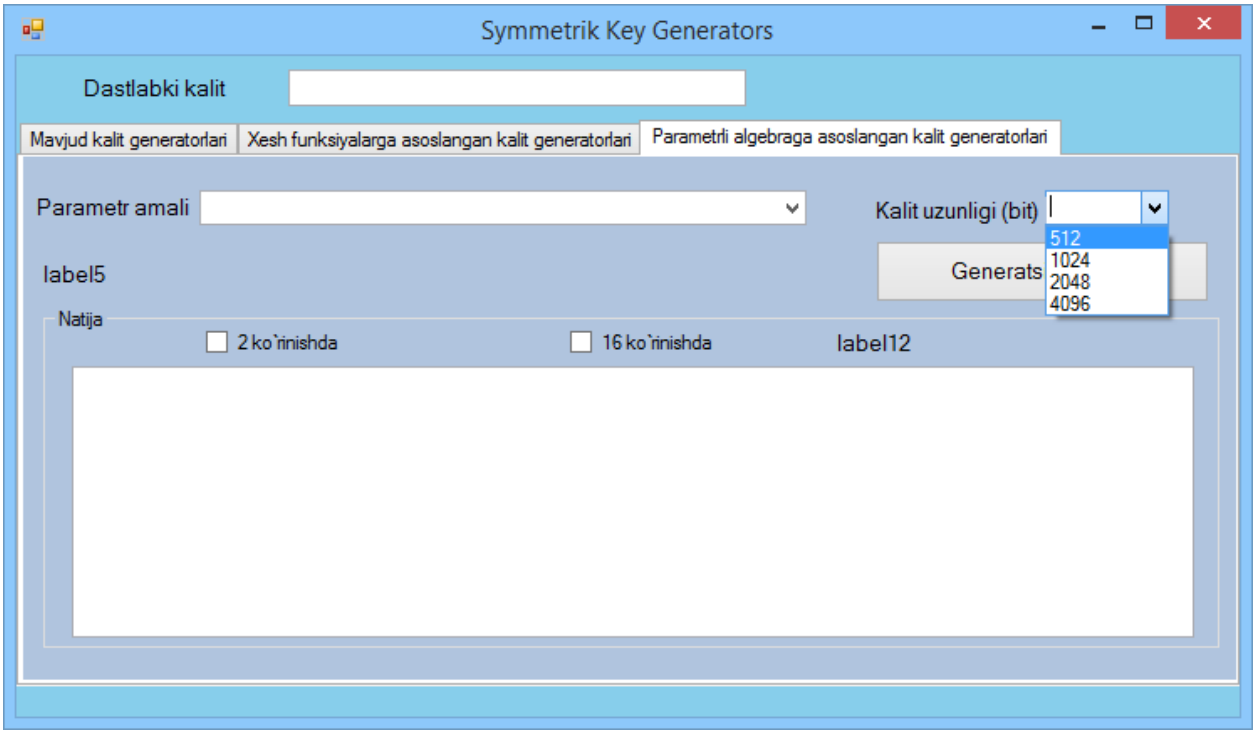
2.1-blok sxema. Parol generatsiya qilish va autentifikatsiyadan o’tish



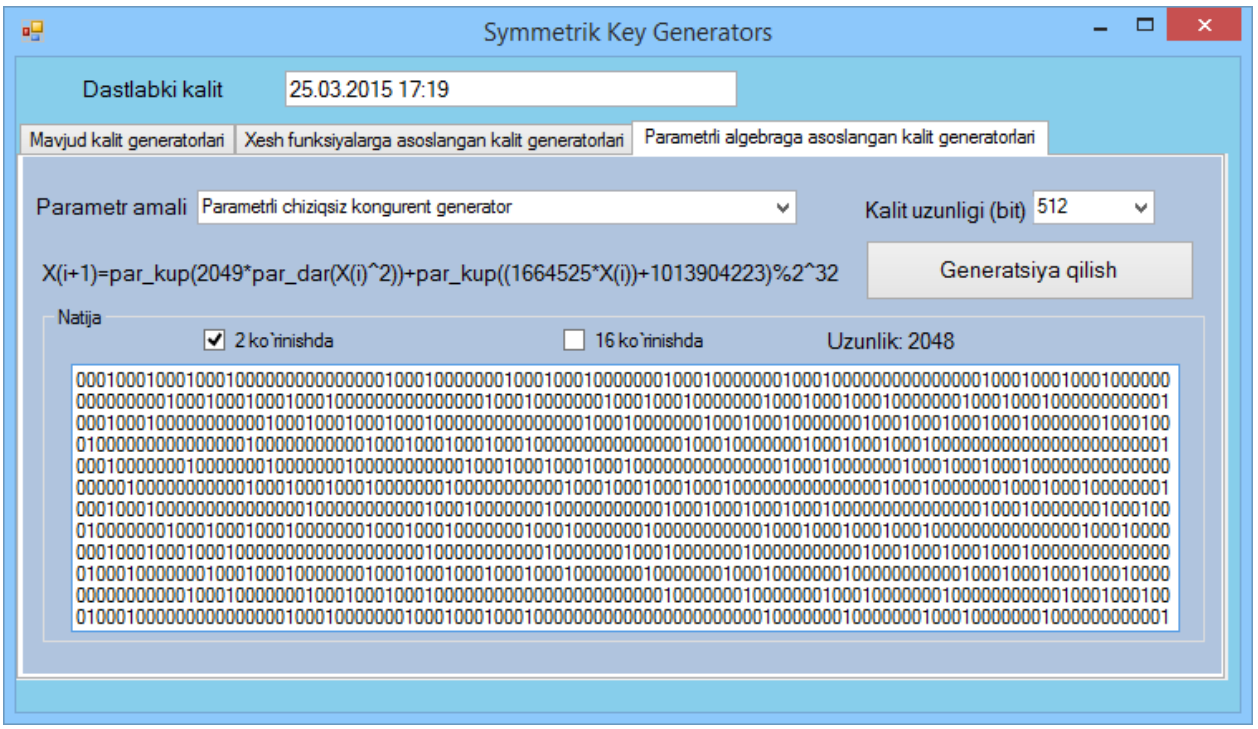
2.8-rasm. Asosiy ko’rinishi



2.9-rasm. Parametrli algebraning 3ta turi bo’yicha ishlaydi



2.10-rasm. Kalitning 4 xil o’lchamlari berilgan



2.11-rasm. Kalitning 2 lik sanoq sistemadagi ko’rinishi

3. Hayot faoliyati xavfsizligi

Insonni hayot faoliyati uni urab turgan atrof-muhitda mavjud bo'lib, u har xil omillar ta'sirida kechadi. Bu omillar kelib chiqish mohiyatidan ko'ra insonga ko'rsatadigan ta'siri, xarakteriga ko'ra nihoyatda turlicha bo'lib, ularning ba'zilari inson hayot faoliyati davomida o'ta ta'sir etadi. Bu omillarga mehnat predmetlari, mehnat vositalari, energiya, mehnat mahsullari, texnologiya, flora (o'simlik), fauna (hayvonot), tabiiy ofatlar, urush-mojarolar, ijtimoiy, iqtisodiy munosabatlar va xokazolar kiradi.

Hayot faoliyati xavfsizligi **-** insonni ishlab chiqarish bilan bog'liq bo'lgan va bog'liq bulmagan faoliyatda uning atrof-muhitga antropologik ta'sirini xisobga olgan xolda xavfsizligini ta'minlovchi bilimlar tizimini tushunamiz**.** Hayot faoliyati xavfsizligihar qanday yo'nalish buyicha o'zini izlanish obyektiga maksad va vazifasiga hamda metodologik yo'liga bog'liq. Xavfsizlik deganda biz inson hayot faoliyati davomida mavjud bo'lgan salbiy omillarni ta'sir extimolini ma'lum darajada yoki butkul bartaraf qilinganini tushunamiz.

Tashqi muhitni muhofaza qilish muammosi bugungi kunning muammosi emas. Insoniyat taraqqiyotining turli bosqichlarida bu muammolar har turli qirralari bilan ko'rinish berib kelgan. Masalan, o'rta asr boshlarida jahonning katta shaharlarida isinish uchun va boshqa maqsadalar uchun tosh ko'mirdan foydalanish boshlangan kezlarda bu shaharlar tutunning ko'payib ketishi natijasida odamlar tutunga qarshi ko'rash e'lon qilgani haqida ma'lumot bor.

Asrimizning 50-yillaridan boshlab avtomobilsozlikning rivojlanishi tufayli avtomobil dvigatellarida yonishdan hosil bo'lgan gaz dunyo miqiyosida eng xavfli ekologik muvozanning buzilishiga olib keladigan omilga aylandi.

3.1. Insonni mehnat faoliyati samaradorligini oshirishning xususiyatlari

Mehnat faoliyati samaradorligini oshirish muammosi har doim ilmiy-tadqiqotchilarning e'tibor markazida bo'lib kelgan va natijada quyidagilar aniqlangan:

1) Mehnat faoliyati samaradorligiga ta’sir etuvchi yuqori mehnat qobiliyatining muhim sharti ishga asta-sekinlik bilan kirishish hisoblanadi. Tadqiqotlarda insonni aqliy mehnatga qaraganda, jismoniy mehnatga tez kirishishi aniqlangan.

2) Mehnat faoliyati unumdorligini maqsadli oshirish uchun ishlovchi mehnat qobiliyati dinamikasi va uning har xil fazalarini bilishi kerak. Birinchidan tayyorlanish vaqti fazasini hisobga olish lozim. Bu faza davomida quyidagilar sodir bo’ladi:

a) ishga tayyorlanish, inson organizmini mehnat rejimiga ko’nikishi;

b) harakat aniqligi va tezligi muvofiqligini yaxshilash;

v) optimal ish holatiga kirishish;

g) ishchi ritmga mos holda (jismoniy yoki aqliy mehnatda) nafas olish va qon aylanish normal rejimini o’rnatish. Ikkinchidan navbatdagi ish davrida eng yuqori mehnat unumdorligini ta’minlovchi ish holatining muvozanat fazasi boshlanadi.

Dam olish va ish jarayoni to’g’ri takrorlanib turishi shikastlanishlar oldini olishning birdan bir asosiy shartidir. Kishining ish qobiliyati uning sezgirligiga, ishlab chiqarishdagi har xil xavfli va zararli omillarga ta’sirchanligiga, ish jarayonining uzluksizligiga bog’liqdir.

Agar kishi kun davomida uzluksiz me’yorda ko’rsatilgan vaqtdan ortiqcha ishlasa, unda jismoniy charchash bilan bir qatorda ruhiy charchash ham paydo bo’lishi mumkin. Buning ustiga agar ishchiga uzoq vaqt mobaynida juda ko’p qarorlar qabul qilish yoki juda ko’p asboblarning ko’rsatkichlariga qarash to’g’ri kelsa, unda ruhiy charchash jismoniy charchashdan oldin kelishi mumkin. Ish joyida shovqin, titrash, gaz, chang va nurlanishning bo’lishi ruhiy charchashni tezlashtiradi va kishining noto’g’ri harakat qilishiga, shikastlanishiga yoki avariya holatining vujudga kelishiga olib kelishi mumkin. Shuning uchun ma’muriyat ish va dam olish tartibiga qat’iyan rioya qilishi kerak.

Odatda ish boshlangandan 3-4 soat o’tib mehnat qobiliyati pasayadi, shu sababli ish smenasining o’rtasida ovqatlanish uchun tanaffus belgilanadi.

Tanaffusdan so’ng mehnat unumdorligi yana o’zining eng yuqori fazasiga ko’tariladi, lekin uning davomiyligi ish smenasining boshlanishidagi kabi uzoq bo’lmaydi. Mos ravishda ishchi holatning muvozanati qisqa vaqt davom etib organizmni ma’lum resurslarini safarbar etishni talab etadigan qisman toliqish fazasi kuzatiladi. Mehnat unumdorligini shirish uchun mehnat ritmiga silliq kirishish katta ahamiyatga ega. Bunga erishish uchun ortiqcha shoshilish va intilishga chek qo’yish lozim. Joydan tez harakatga kelish jismoniy mehnatda ham aqliy mehnatda ham zararlidir.

Ko’psonli tadqiqotlar natijasi mehnat unumdorligini oshirishda quyidagi omillarni (holatlar) ham katta ahamiyatga ega ekanligini ko’rsatadi:

1) yuqori samaradorlikka (mehnatni yaxshi natijalari ko’rinishida) yordam beruvchi mehnat faoliyatini (birinchi navbatda har kunlik) rejalashtirish;

2) mehnatdagi ma’lum tizim, yaxshi ishlangan va o’ylangan ketma-ketlik;

3) ilmiy mutaxassislarni - mehnat gigiyenistlari va fiziologlarining mehnat faoliyatlari bo’yicha tavsiyalaridan foydalanish. Agar ishlar ma’lum ketma-ketlikda bajarilsa har qanday ishlar ham mahsuldor va kam charchashli bo’lishi olimlar tomonidan maxsus tadqiqotlar orqali aniqlangan; ishchi ritm, yuqori mehnat qobiliyatini va muvozanatini o’zlashtirish juda muhimdir; mehnat faoliyati unumdorligini oshirishda bajariladigan ish bo’yicha tajriba, mashk qilinganlik katta ahamiyatga ega bo’lib, bunda ishchi aqliy va jismoniy mehnatni kam sarflab ishni avtomatik tarzda bajarishi mumkin. Mehnat qobiliyatini va birinchi navbatda sog’likni saqlashning muhim omillaridan biri organizm tomonidan o’zlashtirilgan ma’lum mehnat faoliyati tezligi va ritmiga rioya qilish muhim ahamiyatga ega. Bunda ayniqsa ishning optimal ritmda bajarilishi yuqori mehnat unumdorligining asosi hisoblanadi.

3.2. Ekologik xavfsizlik va barqaror rivojlanish

Ekologik xavfsizlik deganda atrof tabiiy muhit holatini organizmlar hayoti uchun extiyojlariga javob beradigan yoki inson uchun sog’lom, toza va qulay tabiiy sharoitga ega atrof muhit tushiniladi. Ekologik xavfsizlikni ta’minlash uchun

har bir alohida davlat ma’lum ekologik siyosat olib borishi kerak.

Ekologik taxdidlar deganda atrof-muhit holati va insonlarning hayot faoliyatiga bevosita yoki bilvosita zarar yetkazadigan tabiiy va texnogen xarkterdagi hodisalar tushiniladi. Ekologik taxdidlar global, mintaqaviy, milliy va maxalliy darajalarga ajraladi.

O’zbekiston Respublikasida ekologik xavfsizlikni ta’minlash strategiyasi ekologiya sohasidagi shaxs, jamiyat va davlatning O’zbekiston Respublikasining milliy xavfsizlik konsepsiyasi va konstitusiyasida berilgan hayotiy zarur manfaatlardan kelib chiqadi.

Shaxsning hayotiy zarur manfaatlariga:

-insonning hayot faoliyati uchun optimal ekologik sharoitlarni ta’minlash, aholii salomatligini himoya qilish kiradi ;

Jamiyatning hayoti zarur manfaatlariga :

- barqaror ekologik vaziyatni qaror toptirish, aholi salomatligini ta’minlash, sog’lom avlodni shakllantirish kiradi ;

Davlatning hayoti zarur manfaatlariga:

- barqaror rivojalantirish regionda ekologik vaziyatlarning barqarorligi, sog’lom turmush tarzini shakllantirish ;

- iqtisodiyotning ustuvor tarmoqlarida ilmiy-texnik rivojlantirishning yuqori darajasini ta’minlash ;

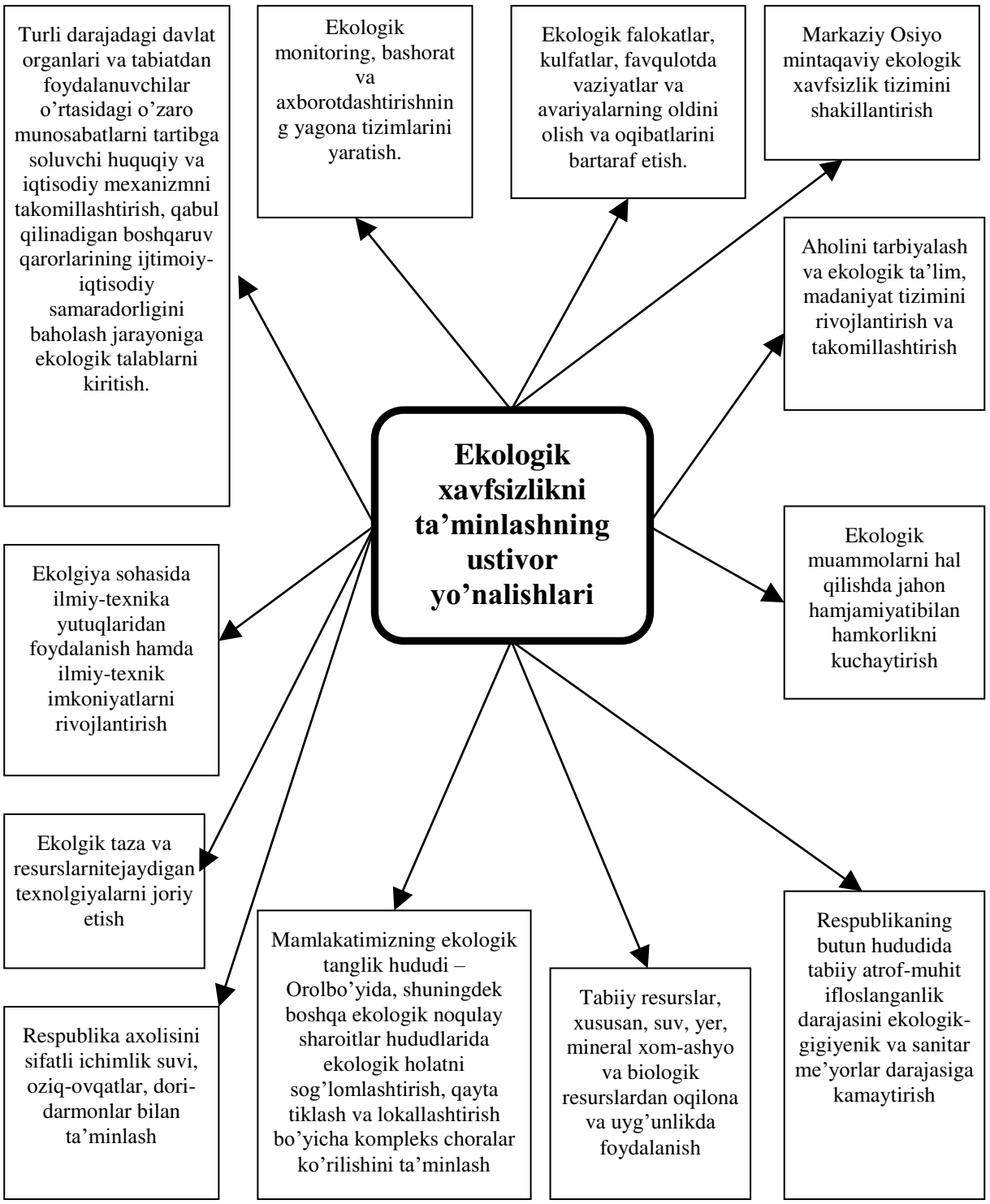
- milliy xavfsizlikning samarali tizimini yaratish, O’zbekistonning kollektiv xavfsizlik va hamkorlikning regional va global tizimlarini tarkibiy va tabiiy qo’shilishini ta’minlash kiradi.

Ekologik xavfsizlikni ta’minlash va ekologik taxdidlarni oldini olish uchun O’zbekistonda birinchi navbatda quyidagi tadbirlarni amalga oshirish maqsadga muvofiqdir:

1. Tabiiyresurslardan, shu jumladan,suv, yer,mineral hom-ashyo va biologikresurslardan kompleks foydalanish.

2.Respublika hududidaatrof – muhitifloslanishiniekologik –gigiyenik va sanitar me’yorlarigachakamaytirish.

3. Ekologik falokat zonasi – Orolbuyida, shuningdek mamlakatning boshqa ekologik nomaqbul hududlarida ekologik holatlarni tiklash va sog'lomlashtirish bo'yicha kompleks tadbirlarni amalga oshirish.



3.1-rasm. Ekologik xavfsizlikni ta'minlashning ustivor yo'nalishlari

4.Respublika aholisinisifatliichimliksuvi, ozuqamahsulotlari, dori-

darmonlarbilanta’minlash.

5.Ekologiktoza va kamchiqitlitexnalogiyalarnijoriyetish.

6.Ekologiya sohasidailmiy-texnik salohiyatnioshirish, fan va texnika yutuqlaridanfoydalanish.

7.Aholiningekologikta’limi, madaniyati, tarbiyasitiziminirivojlantirishi va takomillashtirish.

8. Ekologik halokatlar, ofatlar, favqulotda vaziyatlar, avariyalarni oldini olish va oqibatlarini tugatish.

9. Ekologik muammolarni hal qilishda jahon hamjamiyati bilan hamkorlikni chuqurlatish va boshqalar.

Mamlakatning tashqi va ichki ekologik siyosatini jahon talablari doirasida olib borishda qonuniy hujjatlar hal qiluvchi rol o’ynaydi. Mustaqillik yillarida O’zbekistonda 120 dan ortiq qonun va qonun osti hujjatlari qabul qilingan. Ekologik qonunchilikning maqsadi insonlarning salomatligi, mehnat va maishiy sharoitlari to’g’risida g’amho’rlik qilish hisoblanadi.

Tabiatni muhofaza qilishda qilingan ishlardan biri zarur qonunlar, qarorlar va boshqa me’yoriy hujjatlar tuzishdan iborat. Ekologik huquqning asosiy manbai bo’lib O’zbekiston Respubilikasining Konstitusiyasi hisoblaniladi. Unda tabiat-jamiyat tizimidagi o’zaro munosabatlarni tartibga solish mahsus qoida talablar ham belgilangan bo’lib, ekologik huquqning manbalarini asosini tashkil etadi.

Asosiy qonunimizda fuqarolarning burchilari bu atrof-muhitga ehtiyotkorona munosabatda bo’lishga majburdirlar (50 modda), jamiyatning iqtisodiy negizlaridan biri bu mulkdor mulkni o’z hohishicha egalik qiladi, undan foydalanadi. Mulkdor foydalanish jarayonida ekologik muhitga zarar yetkazmasligi, fuqarolar yuridik shaxslar va davlatning huquqlarini hamda qonun bilan qo’riqlanadigan manfaatlarini buzmasligi shart (54 modda), va yer, yer osti boyliklari, suv, o’simlik va hayvonot dunyosi va boshqa tabiiy zahiralar qator umummilliy boylikdir, undan oqilona foydalanish zarur va ular davlat muhofazasidadir (55 modda).

Huquqni saqlash chegaralariichki va tashqiga bo’linadi. Huquqiy

himoyalashning ichki chegaralari tabiiy dunyoda ijtimoiy dunyoga o'tgan tabiat elementlariga: foydali qazilmalar, suv havzalaridan olingan suv, qazilgan tuproq, otilgan hayvonlar, kushlar va boshqalar. Shu obyektlar uchun insonning tabiat bilan aloqasi uziladi, ular tovar-moddiy boyliklarga aylanadi. Huquqiy himoyaning tashqi chegaralari odamlar yashaydigan yer tabiati, shu jumladan o'zida yerning ta'sirini sezadigan va odamning yashash muhiti holatiga ta'sir ko'rsatadigan (masalan, yerning sun'iy yo'ldoshlarini uchirilishi paytidagi hodisalar) yer atrofidagi bo'shliqni tashkil etadi. Huquqiy himoyalashning tabiiy obyektlari milliy, xalqaro, regional va globalgabo'linadi. Tabiatni saqlash qonunchiligiga asosan caqlashning tabiiy obyektlariga yer, uning boyliklari, suv, o'rmon, hayvonot dunyosi, atmosfera havosi kiradi. Bularning hammasi inson yashashi uchun tabiiy muhit bo'lgan biosferani tashkil etadi.

O'zbekistonda atrof-muhitni himoya qilishning huquqiy asoslari tabiatni saqlash huquqiy normalaridan, ya'ni qonunlardan va qonun mohiyatiga ega bo'lgan aktlardan iborat. Atrof-muhitni saqlash va tabiiy resurslardan unumli foydalanish qonunchiligi keyingi 20 yilda jadal rivojlandi. Keng ko'lamli munosabatlarni tartibga soluvchi qonunlar qabul qilindi: yer qonunchiligi asoslari, sog'liqni saqlash to'g'risidagi qonunchilik asoslari, suv qonunchiligi asoslari, yer osti boyliklari to'g'risidagi qonunchilik asoslari, o'rmon qonunchiligi asoslari, hayvonot dunyosini saqlash va undan foydalanish to'g'risidagi qonun, atmosfera havosini saqlash to'g'risidagi qonun va boshqalar. Qonunlar orqali korxonalarga tabiatni saqlash qonunchiligiga rioya qilish, tabiiy resurslardan samarali foydalanish va qayta ishlab chiqarish, atrof-muhitni ifloslanishdan saqlash, energiya tejovchi, kam chiqit chiqaradigan va chiqitsiz texnologiyalarni joriy etish, shuningdek, tabiiy hom ashyoni kompleks qayta ishlash, atrof-muhit holatini nazorat qiladigan avtomatlashtirilgan tizimlar va asboblar ishlab chiqish yuklatilgan. Atrof-muhit holati yangi texnologiyalar va mashinalar yaratuvchilardan ekologiya masalalariga e'tiborni talab qiladi. Ular qanday texnik yechim texnik va iqtisodiy shartlarnigina emas, balki ekologik aspektlarni ham hisobga olgan holda qabul qilinadi. Loyihaviy yechimlar albatta ekologik

ekspertizadan o’tkazilishi kerak, yangi yaratilayotgan texnologik jarayonlar, mashina-uskunalar va materiallar ularni joriy etishda xalq xo’jalik samarasi bilan bir qatorda yuqori ekologik xavfsizlik darajasini ta’minlashi kerak.

Ekologik qonunchilik bir necha darajalarni o’z ichiga oladi. O’zbekiston Respublikasi Konstitusiyasi normalari ekologik qonunchilikning asosini tashkil kiladi. 1992 yil 8 dekabrda qabul qilingan O’zbekiston Respublikasi Konstitusiyasi asosiy qonun hisoblanib, hamma uchun majburiy va oliy yuridik kuchga egadir. Atrof muhitni muhofaza qilish masalalari Konstitusiyaning 50, 54, 55 va 100-moddalarida berilgan. Konstitusiyaning 50-moddasida “Fuqarolar atrof-muhitga ehtiyotkorona munosabatda bo’lishga majburdirlar” deb ta’kidlanadi. Ushbu talabga ko’ra O’zbekistonning har bir fuqarosi atrof tabiiy muhitni muhofaza qilishi va tabiiy boyliklardan oqilona foydalanishi talablariga to’la amal qilishi shartdir. Asosiy qonunning 54-moddasiga ko’ra, jamiyatning iqtisodiy negizlaridan biri bo’lgan mulkiy munosabatlar bozor iqtisodiyoti qonuniyatlariga mos ravishda e’tirof etiladi. Lekin mulkdor o’z hoxishicha egalik qilishi, foydalanishi va uni tasarruf etishi hyech qachon ekologik muhitga, ya’ni atrof-muhit holatiga zarar yetkazmaslik kerak.

Xulosa

Ushbu bitiruv malakaviy ishini bajarish natijasida quyidagi natijalarga ega bo’ldindi:

- Tasodifiy va psevdotasodifiy sonlar asosida kalitlarni yaratish, simmetrik va asimmetrik shifrlash algoritmlarining ishlash asoslari yoritildi;
- Zamonaviy bir martali kalitlarni yaratish generatorlari, Fips va boshqalarining ishash sxemalari, platformalari, kalitning uzunligi va mustahkamliklari tahlil qilindi;
- Kriptografik kalitlarni qo’llanilish sohalari E-savdo, kompyuter texnologiyalari bilan ishlash va electron xujjat aylanishi tizimlari bayon etilgan;
- Kalitlarni xavfsiz yaratish va saqlash qurilmalari: E-Tokenlar, Java card qurilmalari, USB kalitlar va Java SIM carddan foydalanib kalitlarni yaratish, qayta ishlash saqlash usullari berilgan;
- Kriptografik algoritmlar:parametr algebradi, xesh funksiyalardan foydalngan holda kalit yaratish algoritmi ishlab chiqilgan va u asosida dasturiy modul yaratilgan.

Adabiyotlar ro’yhati

1. «O’zbekiston Respublikasida axborotni kriptografik muhofaza qilishni tashkil etish chora-tadbirlari» to’g’risida O’zbekiston Respublikasi prezidentining qarori.
2. «O’zbeksiton Respublikasi Axborot Texnologiyalari va Kommunikatsiyalarini Rivojlantirish Vazirligini yaratish haqida» to’g’risida O’zbekiston Respublikasi prezidentining qarori. 2015y.
3. «Milliy Axborot-kommunikasiya tizimlarining kompyuter xavfsizligini ta’minlash borasidagi qo’shimcha chora-tadbirlar to’g’risida»gi O’zbekiston Respublikasi Prezidentining qarori, 5 - sentabr 2005 – yil.
4. «Axborotning kriptografik himoya vositalarini loyihalashtirish, tayyorlash, ishlab chiqarish, realizasiya qilish, ta’mirlash va ulardan foydalanish faoliyatini lisenziyalash to’g’risidagi nizomni tasdiqlash haqida»gi O’zbekiston Respublikasi Vazirlar mahkamasining qarori, 21 - noyabr 2007 – yil.
5. Ganiyev S.K., Karimov M.M., Tashev K.A. Axborot xavfsizligi. Oliy o’quv yurtlari uchun. Aloqachi. 2008.
6. Akbarov D. Ye. «Axborot xavfsizligini ta’minlashning kriptografik usullari va ularning qo’llanilishi» – Toshkent, 2008 – 394 bet.
7. Легезо, Денис Безопасность финансовых услуг он-лайн. Intelligent Enterprise. 17 марта 2009 года.
8. Ференец, Вадим ИБ в ТЭКе невозможна без управления доступом (часть 2-я). CIO: руководитель информационной службы 8 октября 2010.
9. Экология и безопасность жизнедеятельности: Учебное пособие для студентов ВУЗов/ ред. Л. А. Муравий, 2002.
10. Белов С.В. Безопасность жизнедеятельности М.: Высшая школа. 2003.
11. ЁрматовҒ.Ё., Исамухамедов Ё.У. Меҳнатни мухофаза қилиш. Дарслик. Ўзбекистан нашриёти. Тошкент 2002.

```
namespace Symmetrik_Key_Generation
{
    partial class Form1
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be disposed;
otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.textBox4 = new System.Windows.Forms.TextBox();
            this.label9 = new System.Windows.Forms.Label();
            this.tabPage3 = new System.Windows.Forms.TabPage();
            this.comboBox4 = new System.Windows.Forms.ComboBox();
            this.label4 = new System.Windows.Forms.Label();
            this.comboBox3 = new System.Windows.Forms.ComboBox();
            this.label3 = new System.Windows.Forms.Label();
            this.groupBox2 = new System.Windows.Forms.GroupBox();
            this.textBox2 = new System.Windows.Forms.TextBox();
            this.checkBox4 = new System.Windows.Forms.CheckBox();
            this.checkBox3 = new System.Windows.Forms.CheckBox();
            this.button2 = new System.Windows.Forms.Button();
            this.tabPage2 = new System.Windows.Forms.TabPage();
            this.comboBox1 = new System.Windows.Forms.ComboBox();
            this.label1 = new System.Windows.Forms.Label();
            this.comboBox2 = new System.Windows.Forms.ComboBox();
            this.label2 = new System.Windows.Forms.Label();
            this.groupBox1 = new System.Windows.Forms.GroupBox();
            this.textBox1 = new System.Windows.Forms.TextBox();
            this.checkBox1 = new System.Windows.Forms.CheckBox();
            this.checkBox2 = new System.Windows.Forms.CheckBox();
            this.label11 = new System.Windows.Forms.Label();
            this.button1 = new System.Windows.Forms.Button();
            this.tabPage1 = new System.Windows.Forms.TabPage();
            this.comboBox6 = new System.Windows.Forms.ComboBox();
            this.label7 = new System.Windows.Forms.Label();
            this.comboBox5 = new System.Windows.Forms.ComboBox();
            this.label6 = new System.Windows.Forms.Label();
            this.groupBox3 = new System.Windows.Forms.GroupBox();
            this.textBox3 = new System.Windows.Forms.TextBox();
            this.checkBox6 = new System.Windows.Forms.CheckBox();
            this.checkBox5 = new System.Windows.Forms.CheckBox();
```

```
            this.label10 = new System.Windows.Forms.Label();
            this.button3 = new System.Windows.Forms.Button();
            this.label8 = new System.Windows.Forms.Label();
            this.tabControl1 = new System.Windows.Forms.TabControl();
            this.tabPage3.SuspendLayout();
            this.groupBox2.SuspendLayout();
            this.tabPage2.SuspendLayout();
            this.groupBox1.SuspendLayout();
            this.tabPage1.SuspendLayout();
            this.groupBox3.SuspendLayout();
            this.tabControl1.SuspendLayout();
            this.SuspendLayout();
            //
            // textBox4
            //
            this.textBox4.Font = new System.Drawing.Font("Microsoft Sans Serif", 9.75F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));
            this.textBox4.Location = new System.Drawing.Point(162, 8);
            this.textBox4.Name = "textBox4";
            this.textBox4.Size = new System.Drawing.Size(273, 22);
            this.textBox4.TabIndex = 1;
            //
            // label9
            //
            this.label9.AutoSize = true;
            this.label9.Font = new System.Drawing.Font("Microsoft Sans Serif", 9.75F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));
            this.label9.Location = new System.Drawing.Point(37, 11);
            this.label9.Name = "label9";
            this.label9.Size = new System.Drawing.Size(92, 16);
            this.label9.TabIndex = 14;
            this.label9.Text = "Dastlabki kalit";
            //
            // tabPage3
            //
            this.tabPage3.BackColor = System.Drawing.Color.SpringGreen;
            this.tabPage3.Controls.Add(this.button2);
            this.tabPage3.Controls.Add(this.groupBox2);
            this.tabPage3.Controls.Add(this.label3);
            this.tabPage3.Controls.Add(this.comboBox3);
            this.tabPage3.Controls.Add(this.label4);
            this.tabPage3.Controls.Add(this.comboBox4);
            this.tabPage3.Location = new System.Drawing.Point(4, 22);
            this.tabPage3.Name = "tabPage3";
            this.tabPage3.Size = new System.Drawing.Size(723, 312);
            this.tabPage3.TabIndex = 2;
            this.tabPage3.Text = "Parametrli algebraga asoslangan kalit generatorlari";
            //
            // comboBox4
            //
            this.comboBox4.FormattingEnabled = true;
            this.comboBox4.Items.AddRange(new object[] {
            "Parametrli darajaga oshirish",
            "Parametrli chiziqli kongurent generator",
            "Parametrli chiziqsiz kongurent generator"});
            this.comboBox4.Location = new System.Drawing.Point(105, 20);
            this.comboBox4.Name = "comboBox4";
            this.comboBox4.Size = new System.Drawing.Size(362, 21);
            this.comboBox4.TabIndex = 7;
            this.comboBox4.SelectedIndexChanged += new
System.EventHandler(this.comboBox4_SelectedIndexChanged);
            //
            // label4
            //
            this.label4.AutoSize = true;
```

```
            this.label4.Font = new System.Drawing.Font("Microsoft Sans Serif", 9.75F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));
            this.label4.Location = new System.Drawing.Point(5, 22);
            this.label4.Name = "label4";
            this.label4.Size = new System.Drawing.Size(99, 16);
            this.label4.TabIndex = 8;
            this.label4.Text = "Parametr amali";
            //
            // comboBox3
            //
            this.comboBox3.FormattingEnabled = true;
            this.comboBox3.Items.AddRange(new object[] {
            "512",
            "1024",
            "2048",
            "4096"});
            this.comboBox3.Location = new System.Drawing.Point(609, 20);
            this.comboBox3.Name = "comboBox3";
            this.comboBox3.Size = new System.Drawing.Size(74, 21);
            this.comboBox3.TabIndex = 9;
            //
            // label3
            //
            this.label3.AutoSize = true;
            this.label3.Font = new System.Drawing.Font("Microsoft Sans Serif", 9.75F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));
            this.label3.Location = new System.Drawing.Point(505, 24);
            this.label3.Name = "label3";
            this.label3.Size = new System.Drawing.Size(105, 16);
            this.label3.TabIndex = 10;
            this.label3.Text = "Kalit uzunligi (bit)";
            //
            // groupBox2
            //
            this.groupBox2.Controls.Add(this.checkBox3);
            this.groupBox2.Controls.Add(this.checkBox4);
            this.groupBox2.Controls.Add(this.textBox2);
            this.groupBox2.Location = new System.Drawing.Point(12, 90);
            this.groupBox2.Name = "groupBox2";
            this.groupBox2.Size = new System.Drawing.Size(701, 203);
            this.groupBox2.TabIndex = 11;
            this.groupBox2.TabStop = false;
            this.groupBox2.Text = "Natija";
            //
            // textBox2
            //
            this.textBox2.Location = new System.Drawing.Point(17, 35);
            this.textBox2.Multiline = true;
            this.textBox2.Name = "textBox2";
            this.textBox2.Size = new System.Drawing.Size(669, 162);
            this.textBox2.TabIndex = 4;
            this.textBox2.TextChanged += new
System.EventHandler(this.textBox2_TextChanged);
            //
            // checkBox4
            //
            this.checkBox4.AutoSize = true;
            this.checkBox4.Location = new System.Drawing.Point(97, 13);
            this.checkBox4.Name = "checkBox4";
            this.checkBox4.Size = new System.Drawing.Size(86, 17);
            this.checkBox4.TabIndex = 5;
            this.checkBox4.Text = "2 ko`rinishda";
            this.checkBox4.UseVisualStyleBackColor = true;
            this.checkBox4.CheckedChanged += new
System.EventHandler(this.checkBox4_CheckedChanged);
```

```
            //
            // checkBox3
            //
            this.checkBox3.AutoSize = true;
            this.checkBox3.Location = new System.Drawing.Point(314, 13);
            this.checkBox3.Name = "checkBox3";
            this.checkBox3.Size = new System.Drawing.Size(92, 17);
            this.checkBox3.TabIndex = 6;
            this.checkBox3.Text = "16 ko`rinishda";
            this.checkBox3.UseVisualStyleBackColor = true;
            this.checkBox3.CheckedChanged += new
System.EventHandler(this.checkBox3_CheckedChanged);
            //
            // button2
            //
            this.button2.Font = new System.Drawing.Font("Microsoft Sans Serif", 9.75F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));
            this.button2.Location = new System.Drawing.Point(508, 50);
            this.button2.Name = "button2";
            this.button2.Size = new System.Drawing.Size(199, 37);
            this.button2.TabIndex = 12;
            this.button2.Text = "Generatsiya qilish";
            this.button2.UseVisualStyleBackColor = true;
            this.button2.Click += new System.EventHandler(this.button2_Click);
            //
            // tabPage2
            //
            this.tabPage2.BackColor = System.Drawing.SystemColors.Desktop;
            this.tabPage2.Controls.Add(this.button1);
            this.tabPage2.Controls.Add(this.groupBox1);
            this.tabPage2.Controls.Add(this.label2);
            this.tabPage2.Controls.Add(this.comboBox2);
            this.tabPage2.Controls.Add(this.label1);
            this.tabPage2.Controls.Add(this.comboBox1);
            this.tabPage2.Location = new System.Drawing.Point(4, 22);
            this.tabPage2.Name = "tabPage2";
            this.tabPage2.Padding = new System.Windows.Forms.Padding(3);
            this.tabPage2.Size = new System.Drawing.Size(723, 312);
            this.tabPage2.TabIndex = 1;
            this.tabPage2.Text = "Xesh funksiyalarga asoslangan kalit generatorlari";
            //
            // comboBox1
            //
            this.comboBox1.FormattingEnabled = true;
            this.comboBox1.Items.AddRange(new object[] {
            "MD5",
            "SHA1",
            "SHA256",
            "SHA512"});
            this.comboBox1.Location = new System.Drawing.Point(105, 33);
            this.comboBox1.Name = "comboBox1";
            this.comboBox1.Size = new System.Drawing.Size(362, 21);
            this.comboBox1.TabIndex = 0;
            //
            // label1
            //
            this.label1.AutoSize = true;
            this.label1.Font = new System.Drawing.Font("Microsoft Sans Serif", 9.75F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));
            this.label1.Location = new System.Drawing.Point(9, 35);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(90, 16);
            this.label1.TabIndex = 1;
            this.label1.Text = "Xesh funksiya";
            //
```

```
            // comboBox2
            //
            this.comboBox2.FormattingEnabled = true;
            this.comboBox2.Items.AddRange(new object[] {
            "512",
            "1024",
            "2048",
            "4096"});
            this.comboBox2.Location = new System.Drawing.Point(609, 33);
            this.comboBox2.Name = "comboBox2";
            this.comboBox2.Size = new System.Drawing.Size(74, 21);
            this.comboBox2.TabIndex = 2;
            //
            // label2
            //
            this.label2.AutoSize = true;
            this.label2.Font = new System.Drawing.Font("Microsoft Sans Serif", 9.75F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));
            this.label2.Location = new System.Drawing.Point(505, 37);
            this.label2.Name = "label2";
            this.label2.Size = new System.Drawing.Size(105, 16);
            this.label2.TabIndex = 3;
            this.label2.Text = "Kalit uzunligi (bit)";
            //
            // groupBox1
            //
            this.groupBox1.Controls.Add(this.label11);
            this.groupBox1.Controls.Add(this.checkBox2);
            this.groupBox1.Controls.Add(this.checkBox1);
            this.groupBox1.Controls.Add(this.textBox1);
            this.groupBox1.Location = new System.Drawing.Point(12, 103);
            this.groupBox1.Name = "groupBox1";
            this.groupBox1.Size = new System.Drawing.Size(701, 203);
            this.groupBox1.TabIndex = 5;
            this.groupBox1.TabStop = false;
            this.groupBox1.Text = "Natija";
            //
            // textBox1
            //
            this.textBox1.Location = new System.Drawing.Point(17, 35);
            this.textBox1.Multiline = true;
            this.textBox1.Name = "textBox1";
            this.textBox1.Size = new System.Drawing.Size(669, 162);
            this.textBox1.TabIndex = 4;
            this.textBox1.TextChanged += new
System.EventHandler(this.textBox1_TextChanged);
            //
            // checkBox1
            //
            this.checkBox1.AutoSize = true;
            this.checkBox1.Location = new System.Drawing.Point(97, 13);
            this.checkBox1.Name = "checkBox1";
            this.checkBox1.Size = new System.Drawing.Size(86, 17);
            this.checkBox1.TabIndex = 5;
            this.checkBox1.Text = "2 ko`rinishda";
            this.checkBox1.UseVisualStyleBackColor = true;
            this.checkBox1.CheckedChanged += new
System.EventHandler(this.checkBox1_CheckedChanged);
            //
            // checkBox2
            //
            this.checkBox2.AutoSize = true;
            this.checkBox2.Location = new System.Drawing.Point(314, 13);
            this.checkBox2.Name = "checkBox2";
            this.checkBox2.Size = new System.Drawing.Size(92, 17);
```

```
            this.checkBox2.TabIndex = 6;
            this.checkBox2.Text = "16 ko`rinishda";
            this.checkBox2.UseVisualStyleBackColor = true;
            this.checkBox2.CheckedChanged += new
System.EventHandler(this.checkBox2_CheckedChanged);
            //
            // label11
            //
            this.label11.AutoSize = true;
            this.label11.Font = new System.Drawing.Font("Microsoft Sans Serif", 9.75F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));
            this.label11.Location = new System.Drawing.Point(464, 14);
            this.label11.Name = "label11";
            this.label11.Size = new System.Drawing.Size(0, 16);
            this.label11.TabIndex = 7;
            //
            // button1
            //
            this.button1.Font = new System.Drawing.Font("Microsoft Sans Serif", 9.75F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));
            this.button1.Location = new System.Drawing.Point(242, 60);
            this.button1.Name = "button1";
            this.button1.Size = new System.Drawing.Size(225, 37);
            this.button1.TabIndex = 6;
            this.button1.Text = "Generatsiya qilish";
            this.button1.UseVisualStyleBackColor = true;
            this.button1.Click += new System.EventHandler(this.button1_Click);
            //
            // tabPage1
            //
            this.tabPage1.BackColor = System.Drawing.Color.Green;
            this.tabPage1.Controls.Add(this.label8);
            this.tabPage1.Controls.Add(this.button3);
            this.tabPage1.Controls.Add(this.groupBox3);
            this.tabPage1.Controls.Add(this.label6);
            this.tabPage1.Controls.Add(this.comboBox5);
            this.tabPage1.Controls.Add(this.label7);
            this.tabPage1.Controls.Add(this.comboBox6);
            this.tabPage1.Location = new System.Drawing.Point(4, 22);
            this.tabPage1.Name = "tabPage1";
            this.tabPage1.Padding = new System.Windows.Forms.Padding(3);
            this.tabPage1.Size = new System.Drawing.Size(723, 312);
            this.tabPage1.TabIndex = 0;
            this.tabPage1.Text = "Mavjud kalit generatorlari";
            //
            // comboBox6
            //
            this.comboBox6.FormattingEnabled = true;
            this.comboBox6.Items.AddRange(new object[] {
            "Chiziqli kongurent generator",
            "Chiziqsiz kongurent generator",
            "RC4 generatori",
            "AES kalit generatori"});
            this.comboBox6.Location = new System.Drawing.Point(105, 20);
            this.comboBox6.Name = "comboBox6";
            this.comboBox6.Size = new System.Drawing.Size(362, 21);
            this.comboBox6.TabIndex = 7;
            this.comboBox6.SelectedIndexChanged += new
System.EventHandler(this.comboBox6_SelectedIndexChanged);
            //
            // label7
            //
            this.label7.AutoSize = true;
            this.label7.Font = new System.Drawing.Font("Microsoft Sans Serif", 9.75F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));
```

```
            this.label7.Location = new System.Drawing.Point(3, 22);
            this.label7.Name = "label7";
            this.label7.Size = new System.Drawing.Size(100, 16);
            this.label7.TabIndex = 8;
            this.label7.Text = "Generator nomi";
            //
            // comboBox5
            //
            this.comboBox5.FormattingEnabled = true;
            this.comboBox5.Items.AddRange(new object[] {
            "512",
            "1024",
            "2048",
            "4096"});
            this.comboBox5.Location = new System.Drawing.Point(609, 20);
            this.comboBox5.Name = "comboBox5";
            this.comboBox5.Size = new System.Drawing.Size(74, 21);
            this.comboBox5.TabIndex = 9;
            //
            // label6
            //
            this.label6.AutoSize = true;
            this.label6.Font = new System.Drawing.Font("Microsoft Sans Serif", 9.75F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));
            this.label6.Location = new System.Drawing.Point(505, 24);
            this.label6.Name = "label6";
            this.label6.Size = new System.Drawing.Size(105, 16);
            this.label6.TabIndex = 10;
            this.label6.Text = "Kalit uzunligi (bit)";
            //
            // groupBox3
            //
            this.groupBox3.Controls.Add(this.label10);
            this.groupBox3.Controls.Add(this.checkBox5);
            this.groupBox3.Controls.Add(this.checkBox6);
            this.groupBox3.Controls.Add(this.textBox3);
            this.groupBox3.Location = new System.Drawing.Point(12, 90);
            this.groupBox3.Name = "groupBox3";
            this.groupBox3.Size = new System.Drawing.Size(701, 203);
            this.groupBox3.TabIndex = 11;
            this.groupBox3.TabStop = false;
            this.groupBox3.Text = "Natija";
            //
            // textBox3
            //
            this.textBox3.Location = new System.Drawing.Point(17, 35);
            this.textBox3.Multiline = true;
            this.textBox3.Name = "textBox3";
            this.textBox3.Size = new System.Drawing.Size(669, 162);
            this.textBox3.TabIndex = 4;
            this.textBox3.TextChanged += new
System.EventHandler(this.textBox3_TextChanged);
            //
            // checkBox6
            //
            this.checkBox6.AutoSize = true;
            this.checkBox6.Location = new System.Drawing.Point(97, 13);
            this.checkBox6.Name = "checkBox6";
            this.checkBox6.Size = new System.Drawing.Size(86, 17);
            this.checkBox6.TabIndex = 5;
            this.checkBox6.Text = "2 ko`rinishda";
            this.checkBox6.UseVisualStyleBackColor = true;
            this.checkBox6.CheckedChanged += new
System.EventHandler(this.checkBox6_CheckedChanged);
            //
```

```
            // checkBox5
            //
            this.checkBox5.AutoSize = true;
            this.checkBox5.Location = new System.Drawing.Point(314, 13);
            this.checkBox5.Name = "checkBox5";
            this.checkBox5.Size = new System.Drawing.Size(92, 17);
            this.checkBox5.TabIndex = 6;
            this.checkBox5.Text = "16 ko`rinishda";
            this.checkBox5.UseVisualStyleBackColor = true;
            this.checkBox5.CheckedChanged += new
System.EventHandler(this.checkBox5_CheckedChanged);
            //
            // label10
            //
            this.label10.AutoSize = true;
            this.label10.Location = new System.Drawing.Point(452, 16);
            this.label10.Name = "label10";
            this.label10.Size = new System.Drawing.Size(0, 13);
            this.label10.TabIndex = 7;
            //
            // button3
            //
            this.button3.Font = new System.Drawing.Font("Microsoft Sans Serif", 9.75F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));
            this.button3.Location = new System.Drawing.Point(488, 47);
            this.button3.Name = "button3";
            this.button3.Size = new System.Drawing.Size(225, 37);
            this.button3.TabIndex = 12;
            this.button3.Text = "Generatsiya qilish";
            this.button3.UseVisualStyleBackColor = true;
            this.button3.Click += new System.EventHandler(this.button3_Click);
            //
            // label8
            //
            this.label8.AutoSize = true;
            this.label8.Font = new System.Drawing.Font("Microsoft Sans Serif", 9.75F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));
            this.label8.Location = new System.Drawing.Point(33, 61);
            this.label8.Name = "label8";
            this.label8.Size = new System.Drawing.Size(45, 16);
            this.label8.TabIndex = 13;
            this.label8.Text = "label8";
            //
            // tabControl1
            //
            this.tabControl1.Controls.Add(this.tabPage1);
            this.tabControl1.Controls.Add(this.tabPage2);
            this.tabControl1.Controls.Add(this.tabPage3);
            this.tabControl1.Location = new System.Drawing.Point(0, 38);
            this.tabControl1.Name = "tabControl1";
            this.tabControl1.SelectedIndex = 0;
            this.tabControl1.Size = new System.Drawing.Size(731, 338);
            this.tabControl1.TabIndex = 0;
            //
            // Form1
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.BackColor = System.Drawing.Color.SkyBlue;
            this.ClientSize = new System.Drawing.Size(729, 394);
            this.Controls.Add(this.label9);
            this.Controls.Add(this.textBox4);
            this.Controls.Add(this.tabControl1);
            this.Name = "Form1";
            this.Text = "Symmetrik Key Generators";
```

```
            this.Load += new System.EventHandler(this.Form1_Load);
            this.tabPage3.ResumeLayout(false);
            this.tabPage3.PerformLayout();
            this.groupBox2.ResumeLayout(false);
            this.groupBox2.PerformLayout();
            this.tabPage2.ResumeLayout(false);
            this.tabPage2.PerformLayout();
            this.groupBox1.ResumeLayout(false);
            this.groupBox1.PerformLayout();
            this.tabPage1.ResumeLayout(false);
            this.tabPage1.PerformLayout();
            this.groupBox3.ResumeLayout(false);
            this.groupBox3.PerformLayout();
            this.tabControl1.ResumeLayout(false);
            this.ResumeLayout(false);
            this.PerformLayout();

        }

        #endregion

        private System.Windows.Forms.TextBox textBox4;
        private System.Windows.Forms.Label label9;
        private System.Windows.Forms.TabPage tabPage3;
        private System.Windows.Forms.Button button2;
        private System.Windows.Forms.GroupBox groupBox2;
        private System.Windows.Forms.CheckBox checkBox3;
        private System.Windows.Forms.CheckBox checkBox4;
        private System.Windows.Forms.TextBox textBox2;
        private System.Windows.Forms.Label label3;
        private System.Windows.Forms.ComboBox comboBox3;
        private System.Windows.Forms.Label label4;
        private System.Windows.Forms.ComboBox comboBox4;
        private System.Windows.Forms.TabPage tabPage2;
        private System.Windows.Forms.Button button1;
        private System.Windows.Forms.GroupBox groupBox1;
        private System.Windows.Forms.Label label11;
        private System.Windows.Forms.CheckBox checkBox2;
        private System.Windows.Forms.CheckBox checkBox1;
        private System.Windows.Forms.TextBox textBox1;
        private System.Windows.Forms.Label label2;
        private System.Windows.Forms.ComboBox comboBox2;
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.ComboBox comboBox1;
        private System.Windows.Forms.TabPage tabPage1;
        private System.Windows.Forms.Label label8;
        private System.Windows.Forms.Button button3;
        private System.Windows.Forms.GroupBox groupBox3;
        private System.Windows.Forms.Label label10;
        private System.Windows.Forms.CheckBox checkBox5;
        private System.Windows.Forms.CheckBox checkBox6;
        private System.Windows.Forms.TextBox textBox3;
        private System.Windows.Forms.Label label6;
        private System.Windows.Forms.ComboBox comboBox5;
        private System.Windows.Forms.Label label7;
        private System.Windows.Forms.ComboBox comboBox6;
        private System.Windows.Forms.TabControl tabControl1;
    }
}


using System;
using System.Collections.Generic;
using System.Text;
using System.Security.Cryptography;
```

```
namespace Symmetrik_Key_Generation
{
    class hash
    {
        public string sha1_xeshing(string str, int size)
        {
            string result=String.Empty;
            for (int i = 0; i < 32; i++)
            {
                byte[] bytes = Encoding.Unicode.GetBytes(str);
                SHA1Managed hashstring = new SHA1Managed();
                byte[] hash = hashstring.ComputeHash(bytes);
                string hashString = string.Empty;
                foreach (byte x in hash)
                {
                    hashString += String.Format("{0:x2}", x);
                }
                result += hex2binary(hashString);
                if (result.Length >= size)
                {
                    break;
                }
            }
            return result.Substring(0,size);
        }
        public string sha256_xeshing(string str, int size)
        {
            string result = String.Empty;
            for (int i = 0; i < 16; i++)
            {
                byte[] bytes = Encoding.Unicode.GetBytes(str);
                SHA256Managed hashstring = new SHA256Managed();
                byte[] hash = hashstring.ComputeHash(bytes);
                string hashString = string.Empty;
                foreach (byte x in hash)
                {
                    hashString += String.Format("{0:x2}", x);
                }
                result +=hex2binary(hashString);
                if (result.Length >= size)
                {
                    break;
                }

            }
            return result;
        }
        public string sha512_xeshing(string str, int size)
        {
            string result = String.Empty;
            for (int i = 0; i < 8; i++)
            {
                byte[] bytes = Encoding.Unicode.GetBytes(str);
                SHA512Managed hashstring = new SHA512Managed();
                byte[] hash = hashstring.ComputeHash(bytes);
                string hashString = string.Empty;
                foreach (byte x in hash)
                {
                    hashString += String.Format("{0:x2}", x);
                }
                result += hex2binary(hashString);
                if (result.Length >= size)
                {
                    break;
```

```
                }
            }
            return result;
        }
        public string md5_xeshing(string str, int size)
        {
            string result = String.Empty;
            for (int i = 0; i < 32; i++)
            {
                byte[] bytes = Encoding.Unicode.GetBytes(str);
                MD5CryptoServiceProvider hashstring = new MD5CryptoServiceProvider();
                byte[] hash = hashstring.ComputeHash(bytes);
                string hashString = string.Empty;
                foreach (byte x in hash)
                {
                    hashString += String.Format("{0:x2}", x);
                }
                result += hex2binary(hashString);
                if (result.Length >= size)
                {
                    break;
                }
            }
            return result;
        }
        private string hex2binary(string hex)
        {
            string binar = "";
            string result = "";
            for (int i = 0; i < hex.Length; i++)
            {
                string rest = "";
                rest = Convert.ToString(Convert.ToInt16(hex[i].ToString(), 16), 2);
                int uzun = rest.Length;
                switch (uzun)
                {
                    case 4: binar = rest;
                        break;
                    case 3: binar = "0" + rest;
                        break;
                    case 2: binar = "00" + rest;
                        break;
                    case 1: binar = "000" + rest;
                        break;
                }
                result += binar;
            }
            return result;
        }
    }
}

using System;
using System.Collections.Generic;
using System.Text;

namespace Symmetrik_Key_Generation
{
    class parameter
    {
        private int parametrli_kupaytirish(int x, int y, int p, int r)
        {
            int yigindi = (x + y * (1 + r * x)) % p;
            return yigindi;
        }
```

```
private int parametrli_daraja_oshirish(int x, int r, int d, int p)
{
    int bir=0;
    int rest=0;
    while (rest != 1)
    {
        int i = 1;
        int sum = 1;
        while (sum <= d)
        {
            sum = (Int16)Math.Pow(2, i);
            i++;
        }
        rest = d - (Int16)Math.Pow(2,i-2);
        if (bir > 0)
        {
            int ikki = daraja_qism(i-2, x, r, p);
            bir = parametrli_kupaytirish(bir,ikki, p, r);
        }
        else
        {
            bir = daraja_qism(i-2, x, r, p);
        }
        d = rest;
    }
    if(rest==1)
    {
        bir = parametrli_kupaytirish(bir, x, p, r);
    }
    return bir;
 }
private int daraja_qism(int n, int x, int r, int p)
{
    int s = x;
    for (int i = 0; i < n; i++)
    {
        s = s * (2 + s * r) % p;
    }
    return s;
}
public string param_daraja(long son, int size)
{
    string result = String.Empty;
    int sum = (int) son;
    for (int i = 0; i < 200; i++)
    {
        sum = parametrli_daraja_oshirish(sum, 5, 37, 4096);
        result += ikkilik(sum);
        if (result.Length > size)
        {
            result = result.Substring(0, size);
            break;
        }
    }
    return result;
}
public string ikkilik(int k)
{
    string binary = Convert.ToString(k, 2);
    return binary;
}
public string chiziqli(long son, int size)
{
    long son1 = son;
    long sonlar;
```

```
            string binValue = string.Empty;
            for (int i = 0; i < 1000; i++)
            {
                int param_kup = parametrli_kupaytirish(1664525, (int)son1, 4096, 7);
                sonlar = ( param_kup+ 1013904223) % (long)Math.Pow(2, 32);
                son1 = sonlar;
                binValue += Convert.ToString(sonlar, 2);
                if (binValue.Length >= size)
                {
                    break;
                }
            }
            return binValue.Substring(0, size);
        }
        public string chiziqsiz(long son, int size)
        {
            long son1 = son;
            long sonlar;
            string binValue = string.Empty;
            for (int i = 0; i < 1000; i++)
            {
                int param_dar = daraja_qism(1,(int)son1, 5, 4096);
                int param_kupay = parametrli_kupaytirish(2049, param_dar, 4096, 5);
                int param_kup = parametrli_kupaytirish(1664525, (int) son1,4096,5);
                sonlar = (param_kupay + param_kup + 1013904223) % (long)Math.Pow(2, 32);
                son1 = sonlar;
                binValue += Convert.ToString(sonlar, 2);
                if (binValue.Length >= size)
                {
                    break;
                }
            }
            return binValue.Substring(0, size);
        }
    }
}
```