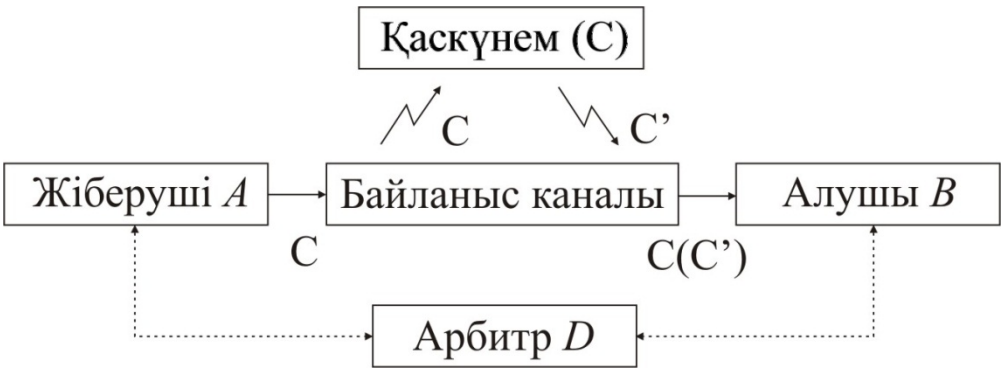

7-сурет. *Ақпараттық айырбас және қауіп төндіру моделі*

Құжаттарды аутентификациялаудың негізгі мақсаты әртүрлі мүмкін қарсылас әрекеттерден қорғау, солардың ішінен бөліп көрсетейік:

1. *Активті ұстап алу* – қарсылас желіге жалғанып алып құжаттарды (файлдарды) ұстап алады және оларды өзгертеді;
2. *Маскарад – С* құжатты *А-ның* атынан жібереді;
3. *Ренегаттылық – А* хабарламаны *В*-ға жібергенімен жіберген жоқпын деп мәлімдейді;
4. *Өзгерту – В* құжатты өзгертеді де, сол құжатты (өзгертілген) *А*-дан алдым деп мәлімдейді;
5. *Ауыстыру – В* құжат (жаңа) жасайды да, оны *А*-дан алдым деп мәлімдейді;
6. *Қайталау* – *А*-ның *В*-ға бұрын жіберген құжатын *С*-ға қайталап жібереді.

Бұл қарсылас әрекеттердің түрлері, өз қызметінде компьютерлік ақпараттық технологиялар қолданатын банктердің, коммерциялық құрылымдардың, мемлекеттік кәсіпорындардың және ұйымдардың, жеке тұлғалардың жұмыстарына маңызды зияндарын тигізеді. Сондықтан қарсылас әрекеттер жасау мүмкіншілігі, компьютерлік технологияларға сенімділікті кеміте түсіреді. Осыған байланысты аутентификациялау мәселесі маңызды болып келеді.

Желідегі хабарламаларды аутентификациялау алгоритмі мен технологияларын таңдаған кезде, жоғарыда көрсетілген қарсылас әрекеттердің барлық түрлерінен қорғайтындай етіп алу керек. Аутентификациялау жүйесінің мұндай қасиеттерімен бірге оның жылдамдығы және іске асыру үшін керек жады көлемі, жоғарыда көрсетілген қауіптерден қорғау дәрежесі (беріктілігі) маңызды болып табылады.

4.2. Электронды цифрлы қолтаңбаны қолдану технологиясы

Электронды цифрлы қолтаңбаны (ЭЦҚ) іске асыру алгоритмдерінде қолданылатын математикалық сызбалар біржақты функцияларда негізделген.

Бұл біржақты функцияларды қолдану келесі мәселелерге негізделген: әрбір жіберуші (жүйенің қолданушысы) алушыға (басқа қолданушыға) *x* құжаттағы өзінің *E(x)* ЭЦҚ-сын тексеру үшін қолданатын *E* процедурасын беруі керек немесе жалпыланған электрондық поштаға салып қою қажет. Өзінің ЭЦҚ-сын қоятын түпнұсқа жүйесін *D* жіберуші құпияда ұстауы қажет.

ЭЦҚ жүйесінде *x* құжаты орнына арнайы қасиеттер тізіміне ие болатын, солардың ішінен ең маңыздысы «коллизий» (яғни, хэш-функциясының мәні бірдей болатын, екі өзге құжатты жасауға тәжірибелік мүмкіндік жоқтығы) болдырмайтын оның хэш-функциясын қолданады *H(x)*. ЭЦҚ-ның ең белгілі математикалық схемалары: RSA, OSS (H. Ong, C. P. Schnorr, A. Shamir), Эль-Гамаля, Рабина (M. Rabin), Окамото (T. Okamoto), DSA (Digital Signature Algorithm), Мацумото-Имаи (T. Matsumoto, H. Imai), Микали-Шамир (Mikali, A. Shamir).

Бұл сызбаларындағы ЭЦҚ-ның күрделігі факторизациялау немесе дискретті логарифмдеу есебін есептеуге негізделген. Ұсынылған алгоритмдер ішінен А. А. Грушо (1992 ж.) айрықша сызбасын атап өтуге болады. Оның біржақты функциясының ерекшелігі, жоғарыда айтылғандай теоретикалық-сандық есептің қиындығында емес, ал оның күрделілігі сызықтық емес бульдік теңдеулер жүйесін шешу қиындығында. АҚШ және Ресейде ЭЦҚ-ға (DSS-Digital Signature Standard, ГОСТ 34. 10-94 және Р34.11-94) қабылданған стандарттарда арнайы жасалған алгоритмдер қолданылады. Бұл алгоритмдер Эль-Гамаль және Шнорр дербес алгоритмдеріне негізделген.

ЭЦҚ жүйесін қолдану технологиясы бір-біріне электронды құжаттар жіберетін қолданушылар желісін қарастырады. Бұл қолданушылар, басқалар қолтаңба қойған хабарламаларды тек тексере алады, ал енді бір қолданушылар (оларды қолтаңба қоюға құқығы бар қолданушылар деп атаймыз) қолтаңбаны тексере алады және қолтаңба қоя да алады. Бұл жадайда қолтаңбаны тексере алатын және қолтаңба қоя да алатын, мысалы, директор және бухгалтер болуы мүмкін.

Әрі қарай екі жағдай болуы мүмкін: бұл желіде орталығы бар (айрықша өкілеті бар ерекшеленген қолданушы) немесе барлық қолданушылар қолтаңба құқығымен бірдей. Сонымен бірге орталықтың жұмыстарының бірнеше локальды орталықтарға таратылған варианты да болуы мүмкін. Орталығы бар желілер қолданушылардың орталыққа деген сенімділігінің дәрежесі бойынша классификациялануы мүмкін. Мұндай желілердегі орталықтар қолданушыны толық бақылауы мүмкін немесе желіге жаңа қолданушыларды енгізу сияқты әкімшіліктің тек формалды функцияларын атқаруы мүмкін.

4.3.Электронды цифрлы қолтаңба алгоритмінің архитектурасы

Маңызды құжатқа қойылған қолтаңбадан зиян шегу мүмкіндігін болдырмау үшін, қолтаңба қояр алдында нақты байқаулар жүргізу керек. Хаттаманы іске асыру жағдайында қолтаңба қоюшының құпиялы кілті, көшіруден қорғалған оның жеке дискетінде сақталады. Бірақ бұл жеткіліксіз болады, себебі дискетті ұрлап кетуі немесе жоғалтып алуы да мүмкін. Демек, құпиялы ақпаратты (кілтті) бөгде басқа адамдардың пайдалануларынан қорғау керек. Бұл проблеманы шешу үшін парольды қорғауды қолдану керек. Парольмен тек ЭЦҚ қою және кілт генерациялау жұмыстарын ғана емес, сонымен бірге желідегі қолданушылардың ашық кілттер каталогын өзгертетін және басқа да жұмыстарды жауып тастауға болады.

Жүйеде «криптовирустар» жоқ екендігін тексеру қажет. Мысалы, қолтаңба қою кезінде криптовирустар құпиялы кілтті ұстап алуы мүмкін (оны қажетті жерге көшіреді). Сонымен бірге қолтаңбаны тексеру кезінде олар қолтаңба дұрыс болмаса да жүйеге оны дұрыс деп «айтқызуы» мүмкін. Кейбір криптовирустар жүйеге кілт генерациялау кезінде бір рет ғана түсіп, жүйеге әлсіз кілттер генерациялауға «көмектесуі» мүмкін. Мысалы, егер кілттер, енгізілген таймерді қолданатын кездейсоқ сандар датчигі негізінде генерацияланатын болса, онда вирус таймер көрсеткіштерін өзгерте алады, одан соң «статус-квоны» қалпына келтіреді. Нәтижесінде бұл кілттер қарсыластармен оңай ашылуы мүмкін. Мұндай криптовирустардан тек бір ғана қорғаныс бар – таза жүйелік дискетіні және таза бағдарламаны қолдану.

4.4. Қолтаңбаны қою және тексеру

ЭЦҚ-ны нақты бір құжатқа қою үшін, үлкен көлемді есептеу жұмысын жасау керек. Бұл есептеулер екі бөлікке бөлінеді: кілттерді генерациялау және құжатқа қолтаңба қою.

4.4.1. Кілттерді генерациялау

Бұл сатыда әрбір қолданушыда екі кілт генерацияланады: құпиялы d және ашық e. Құпиялы кілт қолданушымен жасырын сақталады. Ол қолтаңбаны қалыптастыру үшін қолданылады. Ашық кілт құпиялы кілтпен байланысты, яғни $ed \equiv 1(\mathrm{mod}\,\varphi(n))$. Ашық кілт желідегі басқа барлық қолданушыларға белгілі және қолтаңбаны тексеруге арналған. Оны қолтаңбаның иесін (авторын) және электронды құжаттың дұрыстығын анықтауға мүмкіндік беретін, бірақ құпиялы кілтті есептеп шығаруға мүмкіндік бермейтін керекті құрал ретінде қарастыру керек.

Бұл этапты жүргізудің екі жағдайы бар. Қолданушының өзі кілттерді генерациялай алатын жағдайы тиімді болады. Бірақ белгілі бір жағдайларда бұл жұмысты құпиялы кілттерді әрбір қолданушыға генерациялап оларды тиімді таратумен айналысатын орталыққа берген орынды болады. Екінші варианттың көптеген әкімшілік артықшылықтары бар, бірақ кемшілігі де жоқ емес - қолданушыда оның жеке құпиялы кілті құпиялы екендігіне кепілдігі жоқ. Басқа сөзбен айтқанда барлық қолданушылар орталық «бөркінің астында» болады және орталық кез келген қолтаңбаны жасап алуы мүмкін.

4.4.2. Хэш-фукнциясы

Құжатты хэш-функциясының көмегімен бірнеше ондық немесе жүздік байтқа дейін «сығады». Бұл жерде «сығу» термині «архивация» терминімен ұқсас емес, хэш-функцияның мәні тек күрделі бейнесі арқылы құжатпен байланысты және ол арқылы құжатты қалпына келтіруге мүмкіндік жоқ. Бұл хэш-функция келесі шарттар тізімін қанағаттандыру керек:

- қыстырулар, алыптастаулар, алмастырулар және тағы сол сияқты мәтіндегі әртүрлі мүмкін өзгерістерге сезімді болуы керек;

- қалпына келтіру мүмкіндігінің жоқтығы, яғни хэш-функцияның қажетті мәні бар құжатты табуға болмайтындығы;
- екі әртүрлі құжаттың (олардың ұзындығына қарамастан) хэш-функция мәндері бірдей болу ықтималдылығы жоққа жақын болуы керек.

Әрі қарай хэш-функциядан алынған мәнге, таңдалған ЭЦҚ алгоритміне байланысты математикалық түрлендіру қолдану арқылы құжаттың қолтаңбасын алады. Бұл қолтаңба әбден оқылмалы «әріпті» түрде де болуы мүмкін, бірақ оны негізінен кез келген «оқылмайтын» символдар қатары ретінде көрсетеді. ЭЦҚ құжатпен бірге сақталуы мүмкін, мысалы, оның басында немесе соңында сақталады, бірақ басқа файлда да сақталуы мүмкін. Соңғы жағдайда қолтаңба тексеру кезінде құжаттың өзі де оның қолтаңбасы бар файлда болуы керек.

Енді көп қолданатын хэш-функцияларға тоқталып кетейік:

1. Р. Л. Ривесттің хэш-функцияны есептеу MD (Message Digest Algorithm) алгоритмі жиынтығы.

Бұл алгоритмдердегі хэш-функциялар кірістегі кез келген ұзындықтағы хабарламадан 128-биттік сығылған бейнесін алады.

Мұндағы негізгі амалдар - 2^{32} модулі бойынша қосу, цикілдік жылжыту, $\oplus, \wedge, \vee$ логикалық амалдары.

MD алгоритмі негізінде келесі хэш-функциялар жасалған: MD2, MD5, SHA1 (Secure Hash Standard) және RIPEMD-160 (Race Integrity Primitives Evaluation).

Олар 1994 жылғы MD4 нұсқасында жасалған. MD4-те бір блокты өңдеу, әрбіреуі 16 амалдардан тұратын 3 цикілде орындалады. Әр цикілде өзінің f_j цикілдік функциясы қолданылады, $1 \le j \le 3$, олардың әрбіреуі кірісінде 32 биттік X, Y, Z сөздерін алады, ал шығысында 32 биттік сөзді береді:

$$f_1 = (X \wedge Y) \vee (X \wedge Z),$$
$$f_1 = (X \wedge Y) \vee (X \wedge Z)(Y \wedge Z),$$
$$f_3 = X \oplus Y \oplus Z.$$

2. 1991 жылы К. Шнорр ұсынған және бір жыл өткеннен кейін өзі жетілдірген Фурье жылдам түрлендіруі (ФЖТ) негізіндегі хэш-функция, кез келген ұзындықтағы хабарламаға 128-биттік қысылған бейнесін сәйкестендіреді.

Бұл алгоритмдегі негізгі амалдар – Дискретті Фурье түрлендіруі (ДФТ) және ақырғы өрістердегі полиномдар рекурсиясы.

m хабарламасы *t*-битті жолмен берілсін $m_1\, m_2 \ldots\, m_t$.

m хабарламасы жолының ұзындығы 128-ге еселі болғанша толықтырылады, яғни хабарламаға «1», бұдан кейін нөлдердің керек мөлшері және *t* санының екілік көрінісі тіркеледі. Толықтырылған хабарлама $M = M_1\, M_2 \ldots\, M_n$ әрбіреуінің ұзындығы 128 бит болатын *n* блоктардан тұрсын, яғни 16 биттік сегіз сөзден тұрады.

128 биттік FFT_0 бастапқы хэш-функциясы сегіз 16 биттік сөздердің конкатенациясын көрсетеді $fft_0 \| fft_1 \| \ldots \| fft_7$, мұндағы

$$fft_0 = 0123,\ fft_1 = 4567,\ fft_2 = 89ab,\ fft_3 = cdef,$$
$$fft_4 = fedc,\ fft_5 = ba98,\ fft_6 = 7654,\ fft_7 = 3210.$$

h хэш-функциясын есептеу:

Кіріс: $M = M_1\, M_2 \ldots\, M_n$ (128 биттен *n* блоктар).

Алгоритм: Барлық $i=0,\ldots,n$ үшін $FFT_i = g(FFT_{i-1}, M_i)$.

Шығыс: $h(M) = FFT_n$.

Қадамдық *g* хэш-функциясын есептеу кезінде 16 биттік сөздер векторы $\{e_j\},\ 0 \le j \le 15$ және 16 биттік сегіз сөзден тұратын $E_{0/7}$ және $E_{8/15}$ екі жинағыш қолданылады. Сөздер бірінші жинағышта $e_0, \ldots e_7$, екіншісінде $e_8, \ldots e_{15}$ түрінде белгіленген. $E_{0/7}$ жинағышының бастапқы толтырылуы FFT_0 бастапқы хэш-функциясы сөздерін 2^{16} бойынша модульдеуі болып табылады.

Әрбір 128 биттік $M_1\, M_2 \ldots\, M_n$ блоктарын өңдеу келесі алгоритіммен іске асырылады:

1. M_i блогы 16 биттен $W_0, \ldots, W_7$ 8 сөзге бөлінеді де $E_{8/15}$ екінші жинағышқа жазылады, яғни $j = 0,\ldots,7$ үшін $e_{j+8} = W_j$.

2. Барлық $j = 0,\ldots,15$ үшін

$e_j = e_j + e'_{j-1} + e'_{j-2} + e'_{j-3} + 2^j \pmod p$, мұндағы

$p = 2^{16} + 1;\ e' = e,$ егер $e \ne 0$ және $e' = 1$, егер $e = 0$ ($j,\ j-1,\ j-2,\ j-3$ төменгі индекстері 16 модулі бойнша есептеледі).

3. $(e_0, e_2, \ldots, e_{14}) = FT_8(e_0, e_2, \ldots, e_{14})$;

4. $(e_1, e_3, \ldots, e_{15}) = FT_8(e_1, e_3, \ldots, e_{15})$;

5. мұндағы $(b_0, \ldots, b_7) = FT_8(a_0, \ldots, a_7)$ векторы - $(a_0, \ldots, a_7)$ векторын Z/pZ өрісіндегі ДФТ нәтижесі, $b_i = \sum_{j=0}^{7} 2^{4ij} a_j \pmod p,\ 0 \le i \le 7$.

6. Барлық $j = 0,\ldots,15$ үшін

$e_j = e_j + e'_{j-1} + e'_{j-2} + e'_{j-3} + 2^j \pmod p$

7. $E_{8/15}$ екінші жинағышының мәндері 2^{16} модулі бойынша келтіріледі де $E_{0/7}$ бірінші жинағышқа жазылады, яғни $j = 0,\ldots,7$ үшін $e_j = e_{j+8} \pmod{2^{16}}$.

Әрбір қадам үшін FFT_i өзінен $E_{0/7}$ жинағышының ағындағы мәндерінің сегіз сөзінің конкатенациясын көрсетеді. M_n блогін өңдегеннен кейін хабарламаның сығылған қорытынды бейнесі болып сегіз сөзден $FFT_n = e_0 \| e_1 \| \ldots \| e_7$ тұратын 128 биттік жол табылады.

g қадамдық хэш-функциясын есептеу алгоритімінде 2 және 4 қадамдарды келесімен алмастыруға болатынын айтып кеткен жөн:

$$e_j = e_j + f(e_0,\ldots,e_{j-1},e_{j+1},\ldots,e_{15},j) \pmod p,\ 0 \le j \le 15,$$

мұндағы, f - кез келген функция, оны таңдау нақты қосымшаның қауіпсіздігін қамтамасыз ету керек талаптармен анықталады.

Хэш-функцияға қойылған жоғарыда көрсетілген шарттардың біреуі орындалмаған жағдайда қандай шабуылдар мүмкін болатыны төменде қарастырылады.

4.4.3. Қолтаңбаны тексеру

Қолтаңбаны тексеру үшін тексерушіде қолтаңба қойған қолданушының ашық кілті болуы керек. Бұл кілт аутентификацияланған болуы керек, яғни тексеруші бұл ашық кілттің иесіне сенімді болуы қажет. Қолданушылар өздері кілттермен айырбас жасайтын жағдайда, бұл сенімділік телефондық, жеке байланыс немесе басқа кез келген байланыс түрлерімен бекітілуі мүмкін. Қолданушылар орталығы бар желіде жұмыс жасағанда, қолданушылардың ашық кілттеріне орталық қолтаңба қояды (сертификацияланады), бұл кезде қолданушылардың арасындағы байланыстар (кілтті беруде немесе дұрыстығын растау кезінде), әрбір жеке орталық байланыспен алмастырылады.

ЭЦҚ-ы тексеру жұмыстары екі сатыдан тұрады: құжаттың хэш-функциясын есептеу және бұл қолтаңба қою алгоритмінде қарастырылған математикалық есептеулерді орындау, яғни хэш-функция мен құжат арасындағы қатынасты тексеру. Егер

қарастырылып отырған қатынас орындалса, онда қолтаңба дұрыс, ал құжаттың өзі өзгертілмеген деп есептелінеді, кері жағдайда құжат өзгертілген, ал қолтаңба – дұрыс емес.

4.5. Электронды цифрлы қолтаңбаға қолданушылардың қоятын талаптары

Қолданушыларды әртүрлі сызбалардың математикалық модельдері қызықтырмайды, ал ЭЦҚ функцияларын орындайтын бағдарламалық немесе аппараттық кешендерде болатын қасиеттер қызықтырады, яғни:

- цифрлік қолтаңбаның криптомықтылығы қолтаңба қойған адамның құпиялы кілтіне рұқсаты жоқ кез келген тұлғаның қолтаңбаны жасауына жол бермеу керек;
- қиындық бөгде адаммен ұсталған қолтаңба қойылған құжаттардың санына байланысты болмауы керек;
- сақтаулы тұрған құпиялы «қолтаңба түпнұсқасына» санкцияланбаған пайдаланудан қорғауы керек;
- хэш-функцияны және бағдарламалық кешенді санкцияланбаған пайдаланудан қорғау жүйесін дұрыс таңдау.

ЭЦҚ-ға шабуылды келесі классификацияға бөлуге болады:

- түрлері бойынша: қарапайым (өзгерту, қайталау) және күрделі (хабарламаны сәйкестендіру, тексерушінің ашық каталогына шабуыл жасау);
- зияндары бойынша (қолданушының бір хабарламасына немесе барлық хабарламаларына, желінің барлық қолданушыларының барлық хабарламасына);
- қарсыластың мүмкіншіліктері бойынша (байланыс арнасына, қолданушы компьютеріне немесе қолтаңба жүйесін жасауға мүмкіншілік алу);
- қарсыласқа қажетті ресурстар бойынша (уақыт, ЭЕМ).

Теориялық жағынан алдын алу қиын болатын «өмірлік» шабуылдың үш түрін қарастырайық.

«Маңдайлық» шабуыл. Мұны ең көп қолданылатын шабуыл деп атауға болады, бұдан негізінен бәрі қорғанады. Бұл кезде қарсыласқа қолтаңба қою және хэш-функцияны есептеу алгоритмі белгілі, сонымен қатар қуатты есептеу ресурстары да бар деп есептелінеді.

Сіздің хатшыңыздың қатысуымен өтетін шабуыл. Сіздің қарсыластарыңыздың мүддесінде жұмыс істейтін хатшы, сізге қолтаңба қоюға құжаттарды дайындайды деп есептейік. Сіз хатшыға тағы да бір керек құжатты жасауға нұсқау бердіңіз дейік. Ол құжатты сіздің қарсыластарыңызға апарып береді, олар құжатты өзгерткен кезде бір жағынан ол өзінің мағынасын сақтап қалатындай етіп, ал екінші жағынан оның хэш-функциясының мәні сіздің қарсыластарыңыздың жасаған құжаттың хэш-функциясының мәнімен сәйкес келетіндей етеді. Әрі қарай сіз өзгертілген құжатқа қолтаңба қоясыз, ал сіздің қарсыластарыңыз ол құжаттағы қолтаңбаны «кесіп» алады да, басқа құжатқа «тіркейді», егер енді жаңа құжаттың хэш-функциясы мәні ескі құжаттың хэш-функциясы мәнімен сәйкес келсе, онда қолтаңбаны дұрыс деп мойындайды. Бұл тәрізді қиындықтарды шешудің өзіндік таза алгоритмдік әдіс бар, оны ағылшын әдебиеттерінде «ортада кездесу әдісі» деп атайды. Бұл әдістің мағынасы, хэш-функцияның бір мәніне екі құжатты сәйкестендіруге болады, біріншісі сіз қолтаңба қоятын, екіншісі сіздің қолтаңбаны «тіркеу».

Хэш-функциясының мәні бірдей болатын екі құжаттың болу жағдайын ағылшын тілінде *«коллизий»* деп атау қабылданған.

*Тексерушіге шабуыл.*Орталығы жоқ желіде, әрбір қолданушы хабарлама алатын адамдардың, ашық кілттері каталогін өз компьютерінде сақтайды. Бұл жағдай орталығы бар желіде де болуы мүмкін, егер әрбір қолтаңба қойылған құжат сертификатпен жөнетілсе, яғни орталық қолтаңба қойған тағы да бір хабарламамен, оның ішінде жіберушінің аты мен ашық кілті жазылған. Мұндай жағдайда тексеруші жаққа шабуыл жасауға болады. Тексерушінің компьютерін қолдануға (уақытша болса да) рұқсаты бар қарсылас, каталогтағы сәйкес жазбаларды жәй ғана өзгертуі мүмкін, өзінің тегінің орнына сіздің тегіңізді жазып қояды. Егер ол өзі қолтаңба қойған хабарламаны жіберсе, онда өзгертілген қаталогі бар компьютерде тексеру бағдарламасы, ол хабарламаға сіз қолтаңба қойғандығыңызды көрсетеді. Сипатталған ашық кілттер каталогына шабуыл, ондағы ақпарат шифрланған болса да, кейде мүмкін болады. Алайда қарсыласқа сипатталған шабуылды орындау үшін шифрді ашу міндетті емес! Қарсылас басқа деректерді өзгертпей, өзінің және сіздің шифрланған ашық кілттер жазбаларын өзара ауыстыруы мүмкін.

4.6. Электронды цифрлы қолтаңбалар және олардың салыстырмалы талдауы

Қазіргі уақытта цифрлы қолтаңбаның көптеген түрлері белгілі. Олардың ішінен көп таралғандары: RSA, Эль-Гамаль, DSA, Окамото, Шнорра, Микали-Шамира, ГОСТ Р-34.10-94 және басқалары. Енді осы аталғанцифрлы қолтаңбаларға қысқаша тоқталып өтейік.

Алгоритмдерді есептеу уақыты, кілттің және қолтаңбаның ұзындығы және практикалық қолданулар бойынша салыстырулары келтіріледі.

Қарастырылатын цифрлы қолтаңбаның сызбаларын сипаттау үшін келесі белгілеулер қолданылады:

Z_n - 0 ден $(n-1)$ дейінгі сандар жиынтығы;

$\lceil M \rceil$ - M -ға тең немесе үлкен ең кіші бүтін сан;

$\lfloor M \rfloor$ - M -нан кіші немесе тең ең үлкен бүтін сан;

$|X| = [\log_2 X]$ - X санының екілік жүйедегі разрядының өлшемі;

$JM(|n|)$ - ұзындығы $|n|$ бит болатын модуль бойынша J көбейтуін орындау үшін керек амалдар санының қосындысы;

$JI(|n|)$ - ұзындығы $|n|$ бит болатын модуль бойынша J кері амалдарын орындау үшін керек амалдар санының қосындысы.

Қолтаңба қоюдың барлық сызбаларды, қолтаңба генерациялау алгоритмдерінің параметрлер жиынтығы және оның верификациясы түрінде сипатталады.

4.6.1. Цифрлы қолтаңбаның RSA алгоритмі

Параметрлер:

p және q екі үлкен жай сандары;
$n = pq$ оң бүтін саны;
құпиялы кілт d;
ашық кілт e, $ed = 1(\mathrm{mod}(p-1)(q-1))$;
H - біржақты хэш-функция;
m хабарламасы.