Лекция 8 АҒЫНДЫ ШИФРЛАР ЖӘНЕ ПСЕВДОКЕЗДЕЙСОҚ САНДАРДЫҢ ГЕНЕРАТОРЛАРЫ

Бұл сабақта регистрге негізделген кері байланысы бар псевдокездейсоқ сандарды генерациялау және RC4 алгоритмдарды қарастырамыз. Одан басқа, блокты шифрлардың OFB және CTR тәртіптерін псевдокездейсоқ сандарды алу үшін қалай пайдалануға болатының көрсетеміз.

Ығысу регистрге негізделген кері байланысы бар псевдокездейсоқ сандар генераторлары

Кодтау теориясында және криптографияда кең қолданылады **кері байланысы бар ығысу регистрлер**. Кері байланысы бар ығысу регистрлер псевдокездейсоқ биттер ағының алуға қолданылу мүмкін.

Кері байланысы бар ығысу регистрлер екі бөліктен тұрады: *n*-битты ығысу регистрден және кері байланыс құрылғыдан (сур. 8.1).

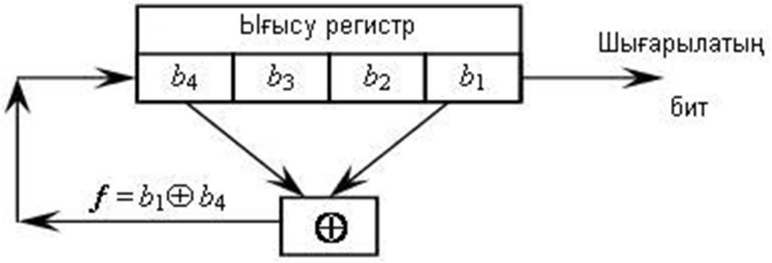
Сурет 8.1. Кері байланысы бар ығысу регистр

Ығысу регистрден биттерді бір бірлеп ғана шығаруға болады. Келесі битты шығару үшін регистрдың барлық биттері оңға қарай 1 разрядқа ығысады. Бұл кезде регистрдың кіруіне сол жақтан жаңа бит түседі, ол кері байланыс құрылғымен құрастырылады және ығысу регистрдің барлық қалған биттеріне тәуелді болады. Осыған сәйкес регистр биттері белгілі бір заң бойынша өзгереді, осы заң ПКС алудың схемасын анықтайды. Әрине, регистр жұмысының кейбір такт санынан кейін биттер тізбегі қайталанады. Қайталауға дейін алынатын тізбектің ұзындығы ығысу регистрдің периоды деп аталады.

Ығысу регистрды пайдаланатын ағынды шифрлар тәжірибеде ұзақ қолданылатын болатын, себебі олар цифрлық аппаратура көмегімен өте жақсы жүзеге асырылады.

Кері байланысы бар ығысу регистрдың қарапайым түрі *кері байланысы бар сызықты ығысу регистры* (linear feedback shift register – LFSR). Бұл құрылғыда кері байланыс регистрдің барлық (немесе кейбіреу) биттердің модулі 2 бойынша қосындысы ретінде жүзеге асырылады. Кері байланыста қатысатын биттер бұруы бар тізбекті құрайды. Кері байланысы бар сызықты ығысу регистрлер немесе олардың модификациялары криптографияда жиі қолданылады.

Кері байланысы бар ығысу регистрдің жұмыс принципін түсіндіру үшін, бірінші және төртінші разрядтан бұруы бар 4-битты LFSR қарастырайық (сур. 8.2).



Сурет 8.2. 4-разрядты сызықты ығысу регистры

Суретте көрсетілген регистрге бастапқы мән 1011 жазайық. Регистрдің ішкі күйлер тізбегін кесте (кесте 8.1) көмегімен есептеу ыңғайлы. Кестеде алғашқы тоғыз регистр күйлері келтірілген.

Әрбір қадамда регистрдің барлық ішіндегісі бір разрядқа оңға қарай ығысады. Мұнда нәтижеде бір битты алуға болады. Сол жақтан босаған орынға кері байланыс функцияның $f = b_1 \oplus b_4$ есептеу нәтижесіне тең болатын бит келіп түседі. Псевдокездейсоқ биттер генератордың шығу тізбегі кестенің соңғы бағанын (алынатын бит) құрайды.

Кесте 8.1. Сызықты ығысу регистрдің жұмыс реті

Күй нөмірі	**Регистрдің ішкі күйі** b_4, b_3, b_2, b_1	**Кері байланыс функцияның есептеу нәтижесі** $f = b_1 \oplus b_4$	**Алынатын бит** (b_1)
0	1 0 1 1	0	1
1	0 1 0 1	1	1
2	1 0 1 0	1	0
3	1 1 0 1	0	1
4	0 1 1 0	0	0
5	0 0 1 1	1	1
6	1 0 0 1	0	1
7	0 1 0 0	0	0
8	0 0 1 0	0	0

n өлшемі бар сызықты ығысу регистры 2^n-1 күйлердің біреуінде болу мүмкін. Сондықтан, осындай регистр максимал периоды 2^n-1 бар псевдокездейсоқ тізбекті жасай алады. Кері байланысы бар сызықты ығысу регистры максимал периодты циклдық биттер тізбегін, тек бұру ретінде белгілі биттер тізбегін таңдағанда, генерациялайды.

Кері байланысы бар сызықты ығысу регистрлер бұрын да, қазір де деректер ағындарын шифрлауға жиі пайдаланады. Криптоберіктігін күшейту үшін бірнеше кері байланысы бар ығысу регистрлердің комбинациясы қолданылады, және қосымша араластыру операциялар енгізіледі. Осындай электронды схемалар мысалы, A5 алгоритмда қолданылаған болатын, оны Еуропада GSM стандарты ұялы цифрлық байланыс арналарды шифрлауға пайдаланды.

Сызықты ығысу регистрлер негізіндегі псевдокездейсоқ сандар генераторлардың негізгі кемшілігі бағдарламалық жүзеге асыру күрделілігі. Ығысулар мен биттік операциялар электронды аппаратурада оңай және жылдам орындалады, сондықтан түрлі елде микросхемалар және ағынды шифрлау құрылғылар шығарылады.

8.2 Псевдокездейсоқ сандарды алу үшін блокты шифрлардың OFB және CTR тәртіптерін пайдалану

Ақпаратты ағынды шифрлау үшін OFB пен CTR тәртіптерді қолданып, түрлі блокты алгоритмды пайдалануға болады, мысалы AES немесе ГОСТ 28147-89.

OFB (Output FeedBack) тәртіп аты былай аударылады «шығу арқылы кері байланыс».

Жіберуге пайдаланатын минимал деректер блогы *j* биттен тұрсын; әдетте *j*=8 (яғни жіберілетін деректердің минимал порциясы 1 байт). OFB тәртіпте құпиялы кілт *K* және кейбір инициализациялау мәні Y_0 негізіндегі блокты шифр *f* псевдокездейсоқ *j*-битты $z_1,z_2,...,z_k$ сандар тізбегін құрастырады, ол сосын хабарды шифрлауда гамма ретінде

пайдалану мүмкін. Шифрлау нәтижесі бастапқы хабардың келесі блокты шифрлау процедурасының кіруі болып табылады. Шифрлаудың әрбір қадамында шифрланған блоктан Y_i кіші j биттер таңдап алынады.

Сонымен, псевдокездейсоқ тізбекті алу үшін мына схема пайдаланылады:

$$Y_i = f(Y_{i-1}, K),$$

Y_i-ың $z_i = j$ үлкен биттері, $1 < i < k$

Егер шифр блогының мөлшері N битке тең болса, онда j параметрі 1 ден N-ға дейін мән қабылдай алады. Y_0 мәні инициализациялау векторы деп аталады.

z_i сандар тізбегін x_i символдан тұратын бастапқы мәліметтер ағының шифрлау үшін гамма ретінде пайдалануға болады:

$$y_i = x_i \oplus z_i$$

нәтижесінде шифрланған y_i символдар ағыны алынады.

y_i мәні ашық мәтінге x_i тәуелсіз болғандықтан, бірдей параметрлерді K мен Y_0 пайдаланып, біз бірдей гамма z_i тізбегін аламыз. Сондықтан әрбір жаңа хабарды жібергенде кілт K мәнін ауыстыру қажет.

Бейнеленген тәртібі үшін хабарды ашып оқу тізбек басынан ғана жүргізіледі, себебі алдыңғыларды есептемей кез келген тізбек элементің z_i алу мүмкін емес.

OFB тәртібінің негізгі артықшылығы – түскен кезде ағынды хабарларды жылдам шифрлау мен дешифрлау үшін z тізбегі алдын ала құрастырылу мүмкін. Бұл нақты уақытта деректерді өңдейтін жүйелерге өте өзекті болу мүмкін.

OFB тәртібінің тағы бір маңызды артықшылығы – егер деректерді берген кезде қателік пайда болса, онда ол келесі шифрланған блоктарға таралмайды, сонымен келесі блоктарды ашып оқуға мүмкіндік қалады. Мысалы, шуылдаған байланыс арнада y_i блокта қателік бит пайда болса, онда тек осы блокты ғана ашып оқуға болмайды.

CTR тәртібінің аты «CounTeR» – «есептеуіш» сөзден шығады. Бұл тәртіп OFB тәртібінің модификациясы. Бір ғана айырмашылығы - CTR тәртіпте шифрдың бұрыңғы шығысы емес, әрбір қадамда 1-ге көбейтілетін есептеуіш шифрланады. Есептеуіштің бастапқы мәні кейбір инициализациялау Y_0 мәнімен анықталады. Жалпы формула келесі түрде жазылады:

$$Y_i = f(Y_{i-1}+1, K),$$

Y_i-ың $z_i = j$ үлкен биттері

CTR тәртібінің артықшылығы – z тізбегінің әрбір элементі тікелей есептелу мүмкін. Өйткені әрбір қадамда Y_i бірге көбейтіледі, демек, егер біз қадам нөмірін i білсек, онда Y_i мәнің Y_0 мен i біле отырып, мына формула бойынша тікелей есептеуге болады:

$$Y_i = f(Y_0+i, K).$$

Бұл хабардың кез келген фрагменттерін бір бірінен тәуелсіз шифрлауға және дешифрлауға мүмкіндік береді.

8.3 RC4 алгоритмы

RC4 алгоритмды Р.Ривест айнымалы ұзындығы бар кілтімен кілттік ақпарат ағын генераторы ретінде арнайы жасап шығарған болатын. Осындай алгоритмы көмегімен құрылған псевдокездейсоқ сандар генераторы блокты шифрларға негізделгеннен бірталай жылдам. RC4 алгоритмы ақпаратты қорғау жүйелерде, компьютерлік желілерде (мысалы, SSL протоколда, Windows NT-да парольді шифрлауда және т.б.) кең пайдаланылады.

RC4 — бұл блок немесе сөз мөлшерімен (n параметры) анықталатын алгоритмдар класы. Әдетте n = 8, бірақ басқа мәндерді де пайдалануға болады. Алгоритм талдауын оңайлату үшін n=4 алайық. RC4-ң ішкі күйі мөлшері 2^n сөзді массивтан және бір сөзді екі

есептеуіштен тұрады. Екі есептеуішті (екеуі де 4-битты) *i* және *j* деп атайық. Барлық есептер модулі 2^n бойынша жүргізіледі.

Массив *S*-бокс деп аталатын ауыстыру кесте ретінде қолданылады, оны әрі қарай *S* деп белгілейміз. Әрбір уақытта *S* кестесі барлық мүмкін *n*-битты араластырылған түрдегі сандардан (біздің жағдайда 4-битты) тұрады. Кестедегі мәндердің нақты орын ауыстыруы кілтпен анықталады. Кестедегі әрбір элемент 0 ден 15-ге дейін мән алатындықтан, оны екі түрлі түсіндіруге болады: не сан ретінде, не кестедегі басқа элементің нөмірі ретінде.

RC4 алгоритмы екі кезеңнен тұрады. Бірінші, дайындау кезеңде орын ауыстыру *S* кесте инициализацияланады. Екінші, негізгі кезеңде псевдокездейсоқ сандар есептеледі.

S кестенің инициализациялауын қарастырайық. Алдымен ол рет-ретімен 0 ден 15-ге дейін сандармен толтырылады. Кілт 4-битты сөздер тізбегі ретінде беріледі, олармен басқа K массивы толтырылады, оның мөлшері *S* массивтың мөлшеріне тең. Егер кілт қысқа болып қалса, онда ол қажетті рет қайталанады. Сосын келесі амалдар орындалады (алгоритм 1):

1. $j = 0;\ i = 0;$
2. $j = (j + S_i + K_i) \bmod 16;$
3. S_i мен S_j орынмен ауыстыру;
4. $i = i + 1;$
5. егер $i < 16$, онда п.2 қайтып келу

Осы алгоритмды орындау нәтижесінде орын ауыстыру *S* кестенің бастапқы толтырылуы жасалынады.

S кестені дайындағаннан кейін, кездейсоқ *n*-битты сөздердің генерациялауын бастауға болады. Ол үшін *i* мен *j* есептеуіштерге бастапқы мән 0 беріледі. Сосын әрбір жаңа z_i мәнің алу үшін келесі амалдар орындалады (алгоритм 2):

$i = (i + 1) \bmod 16;$
$j = (j + S_i) \bmod 16;$
S_i мен S_j орынмен ауыстыру;
$a = (S_i + S_j) \bmod 16;$
$z_i = S_a.$

Алынған 4-битты z_i мәні келесі кіру деректер ағынның 4-битты блокты шифрлау үшін кілт ретінде пайдалану мүмкін.

Мысалы, құпиялы кілт алты 4-битты мәннен тұрсын (оларды ондық түрде көрсетейік): 1, 2, 3, 4, 5, 6. RC4 алгоритмы бойынша сандар тізбегін генерациялап көрейік.

S кестені рет-ретімен 0 ден 15-ге дейін сандармен толтырайық.

Элемент нөмірі	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Мәні	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Сосын *K* кестені дайындаймыз, оған кілтті қажетті рет жазамыз:

Элемент нөмірі	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Мәні	1	2	3	4	5	6	1	2	3	4	5	6	1	2	3	4

Сосын *S* кестенің ішіндегісін араластырамыз. Ол үшін алгоритм 1 пайдаланамыз. Орындау процесін трассировка кестесі түрінде көрсетеміз (кесте 8.2), онда барлық амалдарды белгілейміз. Есептеуді орындағанда еске сақтау керек, барлық қосу операциялар модулі 16 бойынша жүргізіледі.

Кесте 8.2. RC4 алгоритмның дайындау кезеңі (орын ауыстыру кестені инициализациялау)

Алг. пунктінің	**Орындалатын амал (mod 16 бойынша)**	***i* жаңа мәні**	***j* жаңа мәні**

нөмірі			
1	$j = 0$; $i = 0$	0	
2	$j = j + S_i + K_i = 0 + 0 + 1 = 1$		1
3	S_i және S_i орнымен ауыстыру, яғни S_0 және S_1		
4	$i = i + 1$	1	
5	$i < 16$, сондықтан п.2 қайтып келу		
2	$j = j + S_i + K_i = 1 + 1 + 1 = 3$		3
3	S_i және S_i, орнымен ауыстыру, яғни S_1 және S_3		
4	$i = i + 1$	2	
5	$i < 16$, сондықтан п.2 қайтып келу		
2	$j = (j + S_i + K_i) \bmod 16 = (3 + 2 + 3) \bmod 16 = 8$		8
3	S_i және S_i, орнымен ауыстыру, яғни S_2 және S_8		
4	$i = i + 1$	3	
5	$i < 16$, сондықтан п.2 қайтып келу		
2	$j = (j + S_i + K_i) \bmod 16 = (8 + 0 + 4) \bmod 16 = 12$		12
3	S_i және S_i, орнымен ауыстыру, яғни S_3 және S_{12}		
4	$i = i + 1$	4	
5	$i < 16$, сондықтан п.2 қайтып келу		
2	$j = (j + S_i + K_i) \bmod 16 = (12 + 4 + 5) \bmod 16 = 5$		5
3	S_i және S_i, орнымен ауыстыру, яғни S_4 және S_5		
4	$i = i + 1$	5	
5	$i < 16$, сондықтан п.2 қайтып келу		
2	$j = (j + S_i + K_i) \bmod 16 = (5 + 4 + 6) \bmod 16 = 15$		15
3	S_i және S_i, орнымен ауыстыру, яғни S_5 және S_{15}		
4	$i = i + 1$	6	
5	$i < 16$, сондықтан п.2 қайтып келу		
2	$j = (j + S_i + K_i) \bmod 16 = (15 + 6 + 1) \bmod 16 = 6$		6
3	S_i және S_i, орнымен ауыстыру, яғни S_6 және S_6		
4	$i = i + 1$	7	
5	$i < 16$, сондықтан п.2 қайтып келу		
2	$j = (j + S_i + K_i) \bmod 16 = (6 + 7 + 2) \bmod 16 = 15$		15
3	S_i және S_i, орнымен ауыстыру, яғни S_7 және S_{15}		
4	$i = i + 1$	8	
5	$i < 16$, сондықтан п.2 қайтып келу		
2	$j = (j + S_i + K_i) \bmod 16 = (15 + 2 + 3) \bmod 16 = 4$		4
3	S_i және S_i, орнымен ауыстыру, яғни S_8 және S_4		
4	$i = i + 1$	9	
5	$i < 16$, сондықтан п.2 қайтып келу		
2	$j = (j + S_i + K_i) \bmod 16 = (4 + 9 + 4) \bmod 16 = 1$		1

3	S_i және S_j, орнымен ауыстыру, яғни S_9 және S_1		
4	$i = i + 1$	10	
5	$i < 16$, сондықтан п.2 қайтып келу		
2	$j = (j + S_i + K_i) \bmod 16 = (1 + 10 + 5) \bmod 16 = 0$		0
3	S_i және S_j, орнымен ауыстыру, яғни S_{10} және S_0		
4	$i = i + 1$	11	
5	$i < 16$, сондықтан п.2 қайтып келу		
2	$j = (j + S_i + K_i) \bmod 16 = (0 + 11 + 6) \bmod 16 = 1$		1
3	S_i және S_j, орнымен ауыстыру, яғни S_{11} және S_1		
4	$i = i + 1$	12	
5	$i < 16$, сондықтан п.2 қайтып келу		
2	$j = (j + S_i + K_i) \bmod 16 = (1 + 0 + 1) \bmod 16 = 2$		2
3	S_i және S_j, орнымен ауыстыру, яғни S_{12} және S_2		
4	$i = i + 1$	13	
5	$i < 16$, сондықтан п.2 қайтып келу		
2	$j = (j + S_i + K_i) \bmod 16 = (2 + 13 + 2) \bmod 16 = 1$		1
3	S_i және S_j, орнымен ауыстыру, яғни S_{13} және S_1		
4	$i = i + 1$	14	
5	$i < 16$, сондықтан п.2 қайтып келу		
2	$j = (j + S_i + K_i) \bmod 16 = (1 + 14 + 3) \bmod 16 = 2$		2
3	S_i және S_j, орнымен ауыстыру, яғни S_{14} және S_2		
4	$i = i + 1$	15	
5	$i < 16$, сондықтан п.2 қайтып келу		
2	$j = (j + S_i + K_i) \bmod 16 = (2 + 7 + 4) \bmod 16 = 13$		13
3	S_i және S_j, орнымен ауыстыру, яғни S_{15} және S_{13}		
4	$i = i + 1$	16	
5	$i < 16$ – дұрыс емес, сондықтан аяқтау		

Алгоритм 1 орындағаннан кейін инициализацияланған және негізгі кезеңге дайындалған *S* кестені аламыз:

Элемент нөмірі	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Мәні	10	13	14	12	2	15	6	4	5	3	1	9	8	7	0	11

S кестені дайындағаннан кейін кездейсоқ 4-битты сөздердің генеарциялауын бастауға болады. Алгоритм 2 пайдаланып, псевдокездейсоқ тізбектін алдыңғы 5 санын есептейік. Есептеу нәтижесін кесте түрінде келтірейік (кесте 8.3).

Кесте 8.3. RC4 алгоритмның негізгі кезеңі
(псевдокездейсоқ тізбектін элементтерін есептеу)

	Орындалатын амал (mod 16 бойынша)	***i* жаңа мәні**	***j* жаңа мәні**	***a* жаңа мәні**

z_1 есептеу	1. $i = (i+1) = 0+1 = 1$	1		
	2. $j = (j + S_i) \bmod 16 = (0+13) \bmod 16 = 13$		13	
	3. S_1 және S_{13} орнымен ауыстыру			
	4. $a = (S_i + S_j) \bmod 16 = (7+13) \bmod 16 = 4$			4
	5. $z_1 = S_4 = 2$			
z_2 есептеу	1. $i = (i+1) = 1+1 = 2$	2		
	2. $j = (j + S_i) \bmod 16 = (13+14) \bmod 16 = 11$		11	
	3. S_2 және S_{11} орнымен ауыстыру			
	4. $a = (S_i + S_j) \bmod 16 = (9+14) \bmod 16 = 7$			7
	5. $z_2 = S_7 = 4$			
z_3 есептеу	1. $i = (i + 1) = 2+1 = 3$	3		
	2. $j = (j + S_i) \bmod 16 = (11+12) \bmod 16 = 7$		7	
	3. S_3 және S_7 орнымен ауыстыру			
	4. $a = (S_i + S_j) \bmod 16 = (4+12) \bmod 16 = 0$			0
	5. $z_3 = S_0 = 10$			
z_4 есептеу	1. $i = (i + 1) = 3+1 = 4$	4		
	2. $j = (j + S_i) \bmod 16 = (7+2) \bmod 16 = 9$		9	
	3. S_4 және S_9 орнымен ауыстыру			
	4. $a = (S_i + S_j) \bmod 16 = (3+2) \bmod 16 = 5$			5
	5. $z_4 = S_5 = 15$			
z_5 есептеу	1. $i = (i+1) = 4+1 = 5$	5		
	2. $j = (j + S_i) \bmod 16 = (9+15) \bmod 16 = 8$		8	
	3. S_5 және S_8 орнымен ауыстыру			
	4. $a = (S_i + S_j) \bmod 16 = (5+15) \bmod 16 = 4$			4
	5. $z_5 = S_4 = 3$			

Нәтижесінде алғашқы бес мәні келесі болып шықты: 2, 4, 10, 15, 3. Егер одан көп кездейсоқ саны керек болса есептеуді жалғастыруға болады. n=4 болғанда генерацияланған сандардың мөлшері 4 бит болады, яғни 0 ден 15-ке дейін мән алады.

Қарастырылған мысалда сөз немесе алгоритм блоғының n мөлшері төртке тең алынған болатын. Бұл мән басқа да болу мүмкін, мысалы 8 немесе 16. Егер n=8 пайдалансақ, онда орын ауыстыру S кесте 2^8=256 мәннен тұрады, ал орын ауыстыру кестенің элементтері 0 ден 255-ке дейін сандар. i мен j есептеуіштің мөлшері де сегіз битке дейін өзгертілу керек (максимал мәні - 255). Одан басқа, n=8 болғанда есепті модулі 256 бойынша жүргізу қажет.

RC4 алгоритмды криптоталдаушылар өте мұқият зерттеген болатын. Қандай да осал жерлері табылмады. Жоғары криптоберіктігінен басқа, бұл алгоритм өте жылдам және ағынды шифрлауда кілттік тізбекті генерациялау үшін пайдалану мүмкін.

8.4 Криптографиядағы нағыз кездейсоқ сандар генераторлары

ПКС генераторлары крпитографияда кең қолданылады, мысалы, ағынды шифрлауда. Бірақ кейде алдын ала болжамбайтын немесе абсолют кездейсоқ сандардың генерациялауы қажет болу мүмкін. Осындай генераторлар *кездейсоқ сандар*

генераторлары (random number generator) немесе КСГ (RNG) деп аталады. Нағыз кездейсоқ сандар генераторы кейін қайталанбайтын тізбекті жасап шығарады.

Кездейсоқ сандар генераторлардың басты қолдану облысы шифрлау үшін бірегей кілттерді құрастыру. Қандай да болса құпиялы деректерді беру жүйесінде барлық пайдаланушыларға көп кілттер қажет болады. Әдетте шифрлау кілттерді RC4 алгоритмды немесе OFB тәртіпте блокты шифрды пайдалана отырып алуға болады. Бірақ, егер қарсылас кілтті біле алса, онда ол тура осындай кілттерді жасап хабарларды ашып оқи алады. Демек, құпиялы кілттер шынында кездейсоқ болу керек. Сондықтан, нағыз кездейсоқ сандарды генерациялау есебі өте маңызды.

Ең жақсы сипаттамасы нақты әлемнің табиғи кездейсоғына негізделген кездейсоқ сандар генераторында болады. Мысалы, келесі мәліметтерге негізделген КСГ жасауға болады:

- уақыт бірлігінде, мысалы бір секундта, Гейгер есептеуіштің импульстар саны;
- еркін лақтыруда шығатын ойын кубиктің үстінгі қырындаға сандар;
- уақыт бірлігінде, мысалы бір айда, белгілі бір район арқылы ұшып өтетің ұшақтар саны.

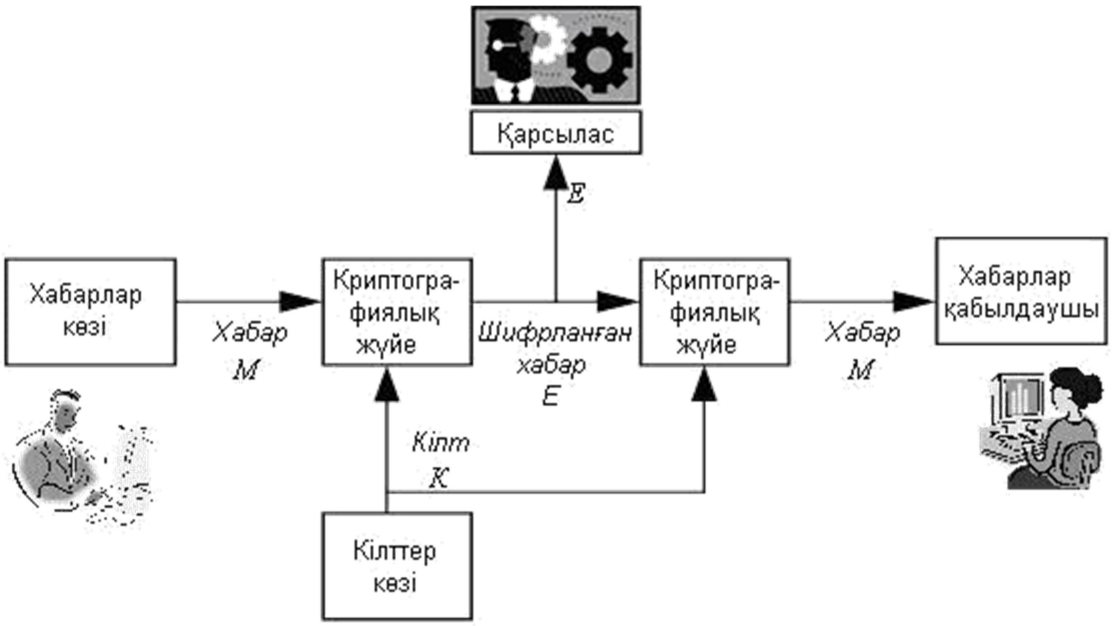
Одан басқа, түрлі физикалық құбылыстардың параметрлері КСГ негізіне қойылу мүмкін. Өкінішке орай, нағыз кездейсоқ сандарды алудың көп әдістемелері тәжірибеде жүзеге асырылмайды, себебі генератор жинақты, шапшаң (сандарды бір секундтан жылдам генерациялау), сыртқы факторларға және қоршаған ортаның жағдайына тәуелсіз болу керек.

Оған қарамастан, түрлі принциптерге негізделген аппараттық КСГ жобалап жасалынады. Мысалы, екі «метал – диэлектрик – жартылай өткізгіш» типті конденсаторды пайдаланып әдістеме құрастырылған. Кездейсоқ мәні осы конденсаторлардың заряд айырмасының функциясы болып табылады. Басқа құрылғыда жартылай өткізгіш диодтің температуралық шу мәні өңделеді және пайдаланады.

Кездейсоқ сандарды алу үшін бағдарламалық-аппараттық әдістер де ұсынылады. Мысалы, дербес компьютердің дыбыс картасының шуына, процессор тактінің есептеуіш мәніне, қатқыл дискінің айналу жылдамдығына, жүйелік таймер мәніне, перне тақтаның жеке пернелерін басу жылдамдығына немесе маус қозғалуына негізделген әдістер. Қандай да болсын әдістеме арқылы алынған кездейсоқ мәліметтер криптоберікті ПКС генераторымен өңделеді және осыдан кейін ғана пайдаланылады.

8.5 Құпиялы кілттерді басқару

Тағы да құпиялы жүйенің жалпы құрылымын қарап шығайық. Осындай жүйенің құрылымы 8.3 суретте бейнеленген.



Сурет. 8.3. Симметриялық шифрлауды пайдаланатын құпиялы жүйенің жалпы құрылымы

Жіберуші (хабарлар көзі) және алушы (шифрланған хабарларды қабылдаушы) шифр мен кілтті таңдау туралы келіседі. Сосын жіберуші таңдап алынған шифрлау алгоритмды пайдаланып өз хабарын шифрлайды және алынған шифрмәтінді (ашық) байланыс арна арқылы жібереді. Алушы шифр мен кілтті пайдаланып оны ашып оқиды.

Қарсылас шифрланған хабарды әрине, ұстап алу мүмкін, өйткені ол ашық байланыс арна арқылы жіберіліп отыр. Бұл жағдайда қарсыластың криптоталдаушысы шифрмәтінді ашуға тырысады. Хабарды жіберуші мен алушы жеткілікті сенімді шифрды пайдалансын, және оны ашып оқу ықтималдығы аса ұлкен емес болсын. Бұл жағдайда шифрлаудың қауіпсіздігі кілттің қауіпсіздігіне толық тәуелді болады. Кілттің ашылуы берілетін деректерді ашуға келтіріде. Сонымен, кілт жасырынып сақталыну керек. Сондықтан кілттерді бастапқы үлестіру үшін сенімді байланыс арна қажет.

Кілттерді бастапқы үлестірудің ең сенімді тәсілі - абоненттер жеке кездескенде кілттермен алмасу. Кілттерді жеткізу үшін арнайы курьерлерді де пайдалануға болады. Егер құпиялы хабарлармен алмасуда көп емес, мысалы, екі немесе үш қатысушылар болса, онда айтылған тәсілдерді әбден пайдалануға болады. Егер де абоненттер саны көп болса, онда кілттерді үлестіру есебі өте күрделі болып шығады.

Құпиялы кілттерді пайдаланғанда басқа да қиыншылақтар бар. Мысалы, кілттер анда-санда ауыстырылу қажет. Өйткені кілт неғұрлым көп пайдаланса, соғұрлым оны ашу ықтималдығы көп болады. Кілт ашылмаса да, қарсыласқа криптоталдауды жүргізу оңай болады, себебі оның қолында бірдей кілтпен шифрланған жеткілікті көп хабарлар болады. Әрбір шифрланған хабарлармен алмасу сеансында ең жақсы өз бірегей кілтті (сеанстық кілт) пайдалану. Үлкен телекоммуникациялық желі үшін сонша көп кілттерді қайдан алуға болады және оларды үлестіруге қалай болады?

Сонымен, өзара әрекеттесу жақтар саны көп болғанда, едәуір көп кілттерді алдын ала тарату қажет және оларды сақтау мен ауыстыру керек. Жергілікті желіде 100 пайдаланушы болсын. Пайдаланушылар құпиялы деректермен «әрбір әрбіреумен» принципы бойынша бір-бірімен алмасқысы келсін. Бұл жағдайда әрбір жұп пайдаланушылар үшін өз құпиялы кілт қажетті. Жүз пайдаланушылардан $100 \cdot 99/2=4950$ жұп құрастыруға болады, демек деректерді беру жүйеде 4950 әртүрлі құпиялы кілт пайдаланатын болады. Осы кілттердің барлығы сенімді түрде жасалынып үлестірілу керек. Одан басқа, жүз пайдаланушылардың әрбіреуі 99 әртүрлі кілтті есте сақтау керек. Егер де хабарлармен алмасуда жүз емес мың адам қатысатын болса, онда кілттерді басқару есебі тым күрделі болып шығады.

Айтылған қиындықтарға байланысты тәжірибеде арнайы автоматтандырылған кілттерді басқару жүйелер қолданылады. Осындай жүйелер кілттерді генерациялауға, сақтауға, архивтеуге және жоғалған кілттерді қалпына келтіруге, ауыстыруға немесе ескі кілттерді жоюға мүмкіндік береді. Кілттерді басқару жүйенің ең маңызды бөлігі *кілттерді үлестіру орталығы* (Key Distribution Center – KDC), оның міндеті кілттерді генерациялау, үлестіру және беру.

Мамандар арнайы процедуралар (немесе протоколдар) жасап шығарған, олар кілттерді үлестіру орталығына пайдаланушыларға жеке байланыс сеансты жүргізу үшін кілттерді (сеанстық кілттер) жеткізуге мүмкіндік береді. Өкінішке орай, симметриялық шифрлауды пайдаланатын барлық протоколдарда кемшіліктер бар. Мүмкін болатын кілттермен алмасу протоколдын біреуін қарап шығайық.

Кілттерді үлестіру орталығын (қысқаша Орталық деп атайық) пайдаланып, екі абоненттер арасындағы байланыс сеансты жүргізу үшін құпиялы кілттерді үлестіру процедурасы былайша болу мүмкін:

1. Абонент *А* абонент *Б*-мен байланысу үшін Орталықтан сеанстық кілт сұрайды.
2. Орталықта кездейсоқ сеанстық кілт жасалынады. Осы сеанстық кілттің екі көшірмесі шифрланады – біреуі абонент *А*-ның құпиялы кілтің пайдаланып, басқасы - абонент *Б*-ның құпиялы кілтің пайдаланып. Сосын екі шифрланған көшірмелер Орталықтан абонент *А*-ға жіберіледі.
3. Абонент *А* сеанстық кілттің өз көшірмесін ашады және екінші шифрланған көшірмесін абонент *Б*-ға жібереді.
4. Абонент *Б* сеанстық кілттің өз көшірмесін ашады.
5. *А* мен *Б* абоненттер алынған сеанстық кілтті ақпаратпен құпиялы алмасуда пайдаланады.

Көрсетілген протокол қарапайым және мысалы, деректерді беру программа арқылы автоматтандырылу мүмкін. Бірақ келтірілген сеанстық кілтті үлестіру процедурасында бірнеше анық кемшіліктері бар.

Берілген жүйенің бірінші кемшілігі – Орталық барлық алмасуға қатысады. Егер Орталық жұмысында қателік болса онда барлық жүйенін де жұмысы бұзылады.

Екінші кемшілік - кілтті үлестіру орталығы желі абонентердің құпиялы кілттерін қандай да болса түрде сақтау керек. Егер қаскүнем жүйе пайдаланушылардың құпиялы кілттеріне қол жеткізсе (жүйені «бұзып ашу», әкімшіні сатып алу және т.б.), онда берілетің хабарларды ол оқып өзгерте алады.

Ақырында, пайдаланушы желігі кіргенде құпиялы кілттін бастапқы үлестіру проблемасы қалады. Бастапқы құпиялы кілт абсолютті сенімді байланыс арна бойынша жеткізілу керек, әйтпесе барлық протоколдың мағынасы жоғалады.

Симметриялық шифрлау алгоритмдардың осындай және басқа кемшіліктері XX ғасырдың 70-ші жылдары анықталған болатын. Кілттерді үлестіру проблеманың (және кейбір басқа маңызды проблемалар) шешімі симметриялық емес шифрлау алгоритмды пайдалануы болып шықты.

Негізгі ұғымдар

CTR – блокты шифрдың жұмыс тәртібі, ол ақпаратты ағынды шифрлауда кілттерді генерациялауға мүмкіндік береді.

LFSR (linear feedback shift register) – кері байланысы бар сызықтық ығысу регистры.

OFB – блокты шифрдың жұмыс тәртібі, ол ақпаратты ағынды шифрлауда кілттерді генерациялауға мүмкіндік береді.

RC4 алгоритмы – псевдокездейсоқ сандарды генерациялау алгоритмы. Ағынды шифрлауда кілттерді генерациялау үшін пайдалану мүмкін.

Кері байланысы бар сызықтық ығысу регистры (linear feedback shift register – LFSR) – кері байланысы бар ығысу регистрдың варианты. Осындай регистрде кері байланыс барлық (немесе кейбір) биттердің модулі 2 бойынша қосындысы ретінде жүзеге асырылады.

Кері байланысы бар ығысу регистры *n*-битты ығысу регистрден және кері байланыс құрылғыдан тұрады. Битты алған кезде, регистрдың барлық биттері оңға қарай бір позицияға ығысады. Сол жақтағы жаңа бит басқа биттерден кері байланыс функциямен анықталады. Кері байланысы бар ығысу регистрлер псевдокездейсоқ биттер ағының алуға пайдалану мүмкін.

Сұрақтар

1. Псевдокездейсоқ сандар генераторлардың негізгі сипаттамаларын, артықшылықтарын және кемшіліктерін айтып беріңіз.
2. Псевдокездейсоқ сандарды алу үшін кері байланысы бар ығысу регистрлер қалай пайдалану мүмкін? Олардың жұмыс принципын түсіндіріп беріңіз.
3. Деректерді ағынды шифрлау үшін OFB тәртіпте блокты шифр қалай пайдалану
мүмкін?
4. Блокты шифрдың CTR тәртібі қалай орындалады?
5. Кездейсоқ және псевдокездейсоқ сандар генераторлар арасында қандай айырмашылық бар?
6. Ағынды шифрлауда гамманы алу үшін нағыз кездейсоқ сандар генераторын пайдалануға болама? Неліктен?
7. Нағыз кездейсоқ сандар генераторы қандай криптографиялық мақсаты үшін пайдалану мүмкін?
8. Шифрланған хабарлармен алмасу жүйелерде құпиялы кілттерді басқару кезінде қандай проблемалар болу мүмкін?