Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Новгородский государственный университет имени Ярослава Мудрого»

Политехнический институт

Кафедра прикладной математики и информатики

**Методы защиты информации**

Лабораторная работа 1

**Сеть Фейстеля  
Алгоритм DES (4 режима)**

по дисциплине «Методы защиты информации»

по направлению подготовки 09.03.01  
 «Информатика и вычислительная техника»

направленность (профиль) «Программное обеспечение   
вычислительной техники и автоматизированных систем»

Проверил

Доцент КПМИ, доц., к. ф.-м. н.

Жгун Т.В.

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2024 г.

Выполнил

студент группы 3091

Михайлов Роман Александровчи

Вариант 17

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2024 г.

Оглавление

[Сеть Фейстеля 3](#_Toc14208)

[DES (Data Encrypt Standart) 9](#_Toc26595)

[Приложение А 11](#_Toc6653)

[Приложение Б 14](#_Toc11541)

Сеть Фейстеля

Реализовать работу сети Файстеля для блока 16 бит (шифрование и расшифрование). Число раундов 8. Исходный текст представлен в Приложениие А. В качестве ключа шифрования выступает пароль «Пустой колос голову кверху носит». Ниже представлене схема сети Фейстеля.

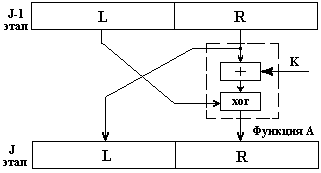


Рисунок 1 – Схема сети Файстеля

Рауновые ключи получаются путем сложение числовых значений символов по модулю 256.

Основной шаг криптопреобразования - наложение раундового ключа и сдвиг влево на 1 бит. Реализуемая программа выводит промежуточные результаты в соответсвующие файлы.

Гистограммы полученных промежуточных результатов представлены ниже. Полученный шифротекст представлен в приложении А.

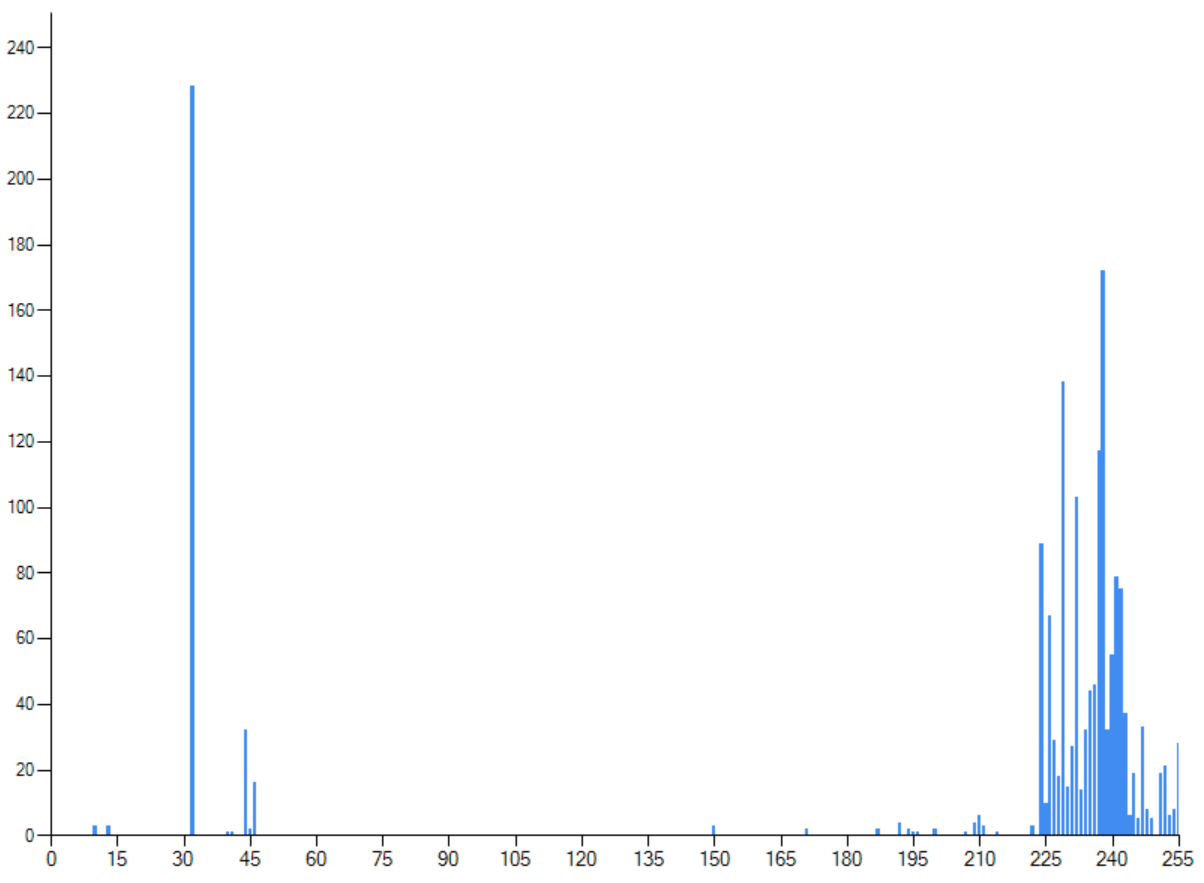


Рисунок 2 – Гистограмма открытого текста, H(x) = 4,607 бит

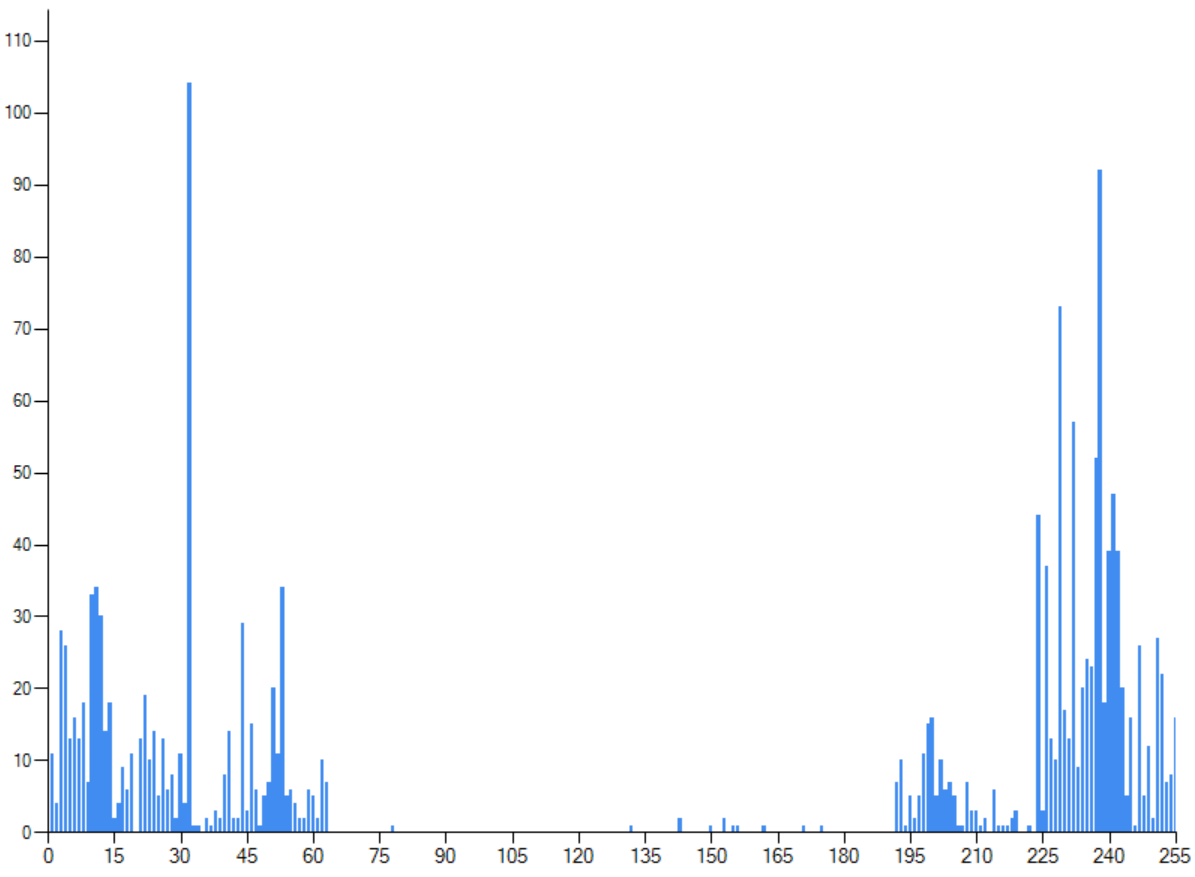


Рисунок 3 – Гистограмма шифротекста 1 раунда, H(x) = 6,201 бит

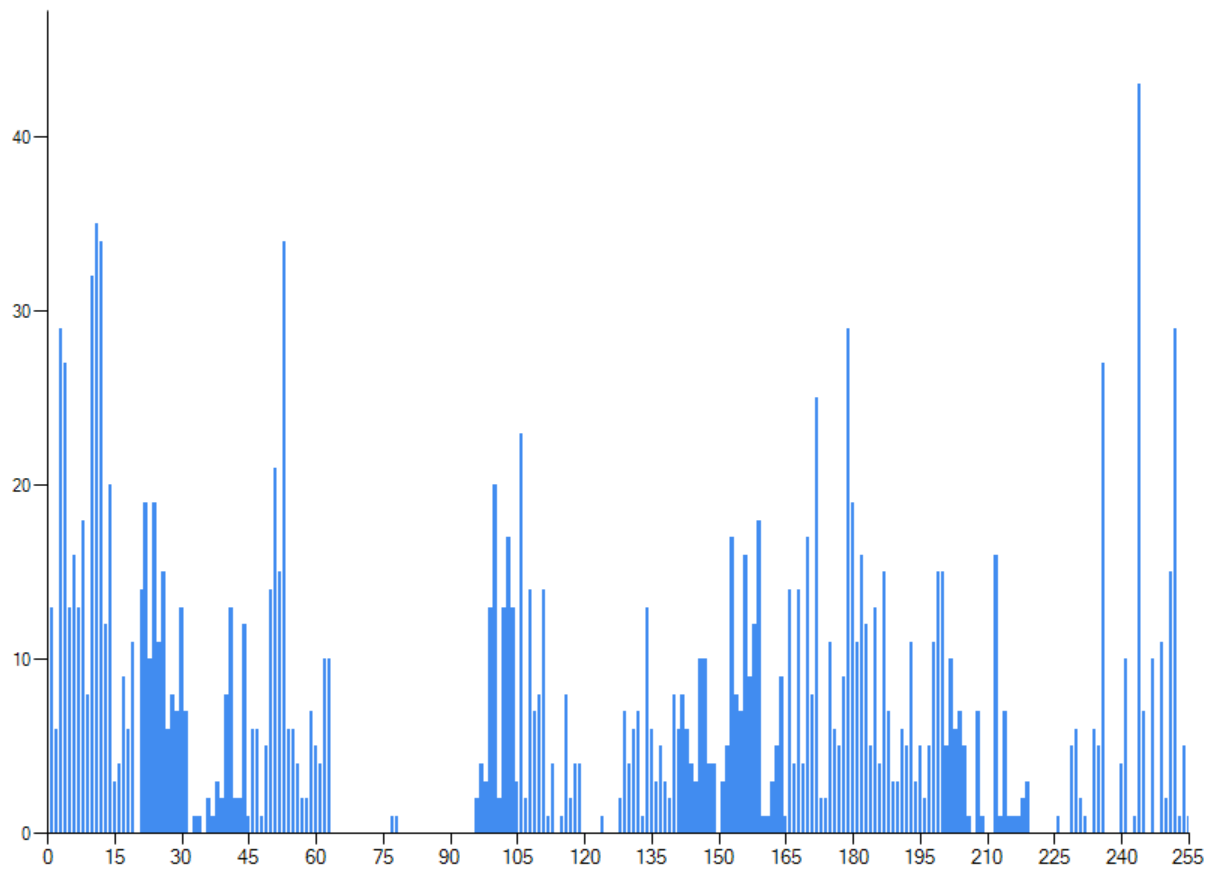


Рисунок 4 – Гистограмма шифротекста 2 раунда, H(x) = 7,115 бит

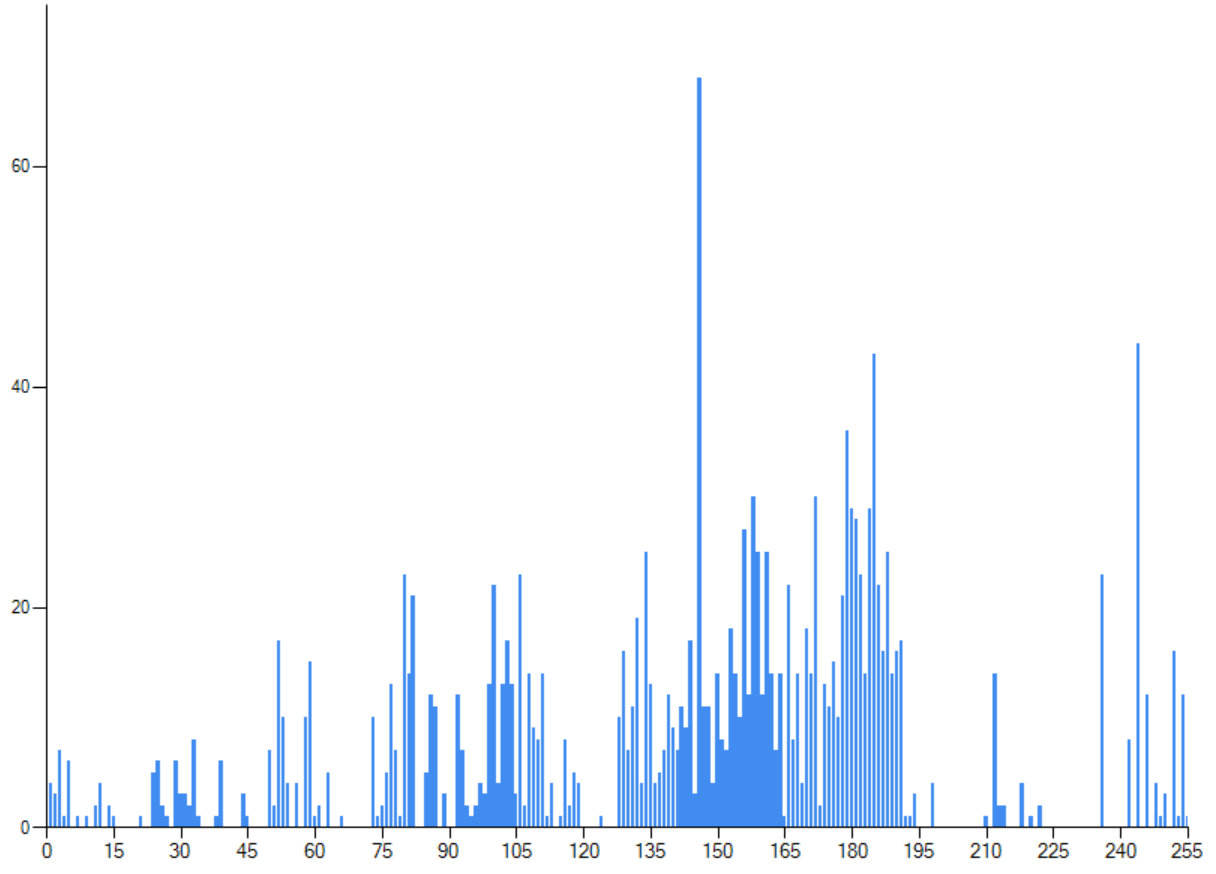


Рисунок 5 – Гистограмма шифротекста 3 раунда, H(x) = 6,806 бит

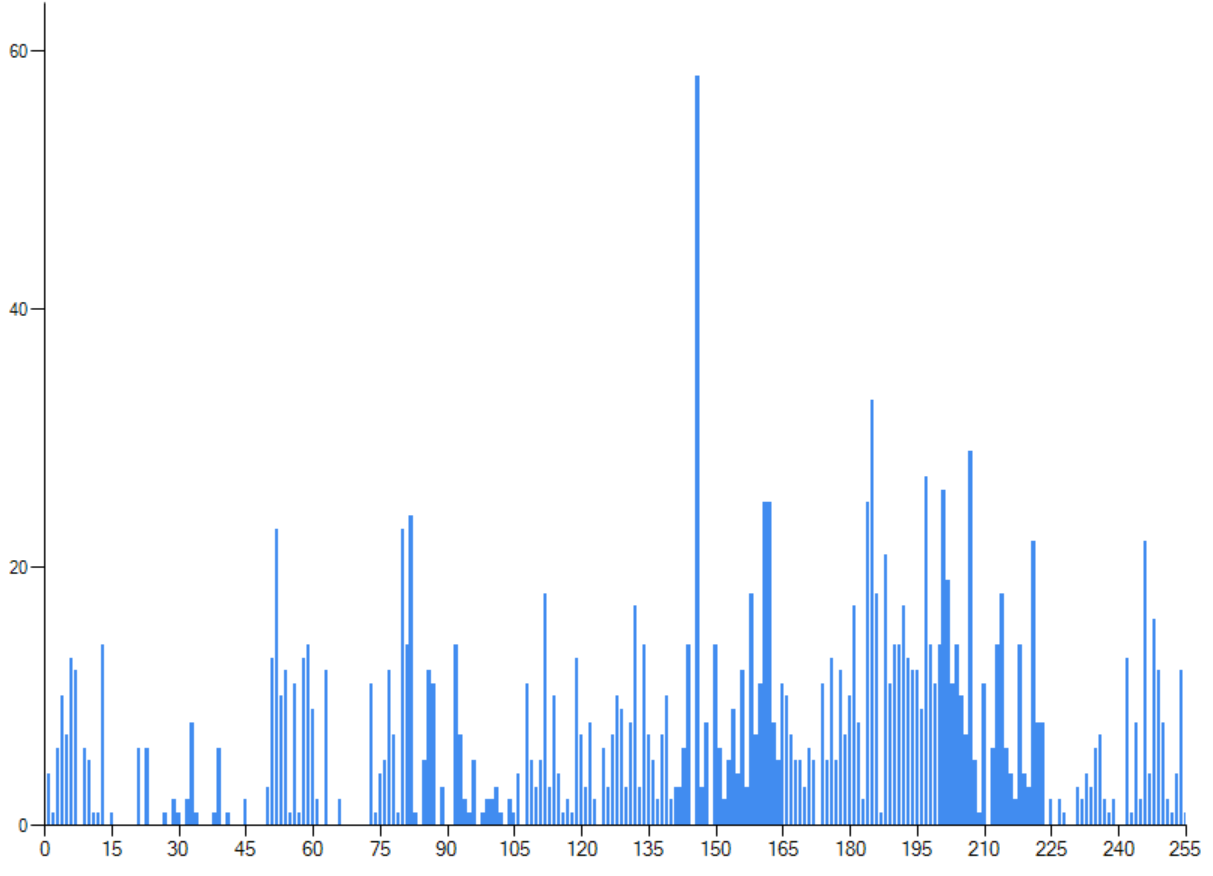


Рисунок 6 – Гистограмма шифротекста 4 раунда, H(x) = 7,185 бит

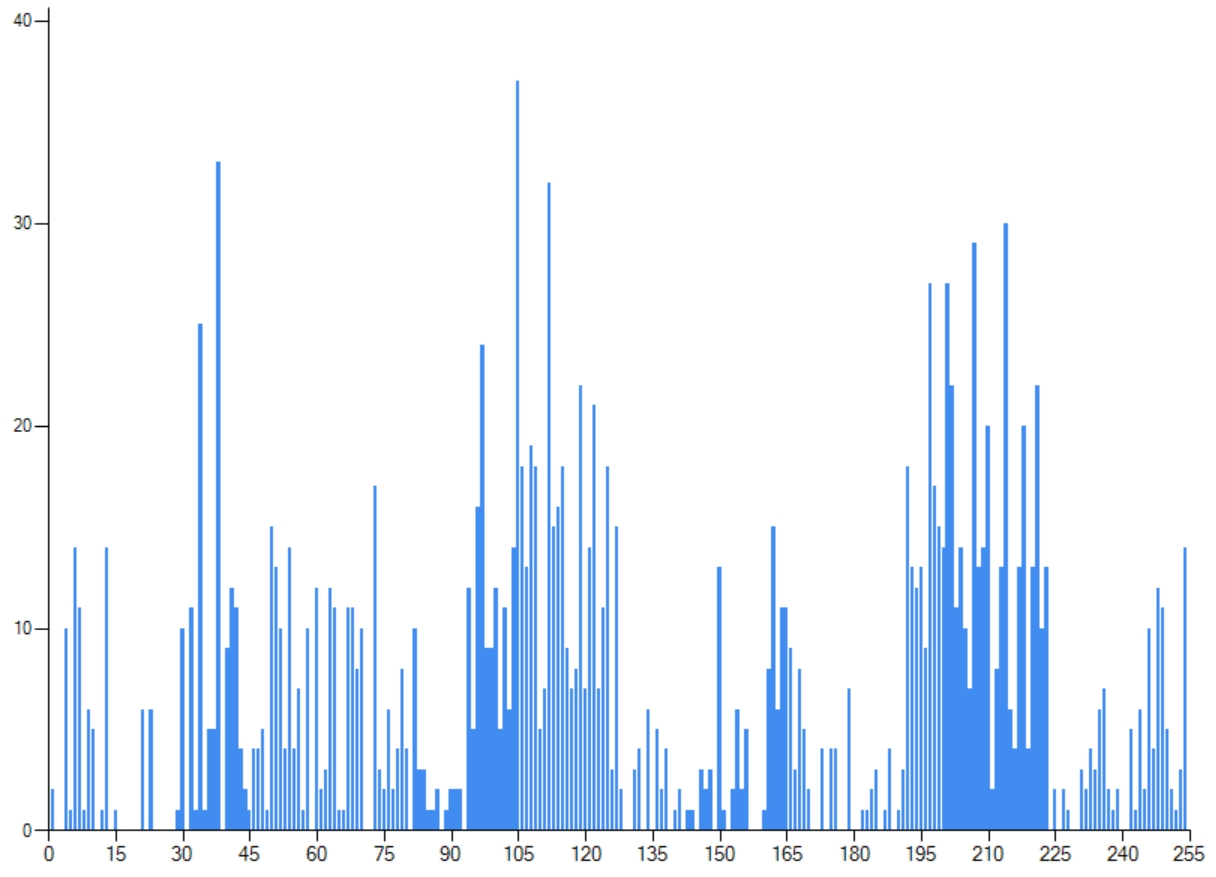


Рисунок 7 – Гистограмма шифротекста 5 раунда, H(x) = 7,183 бит

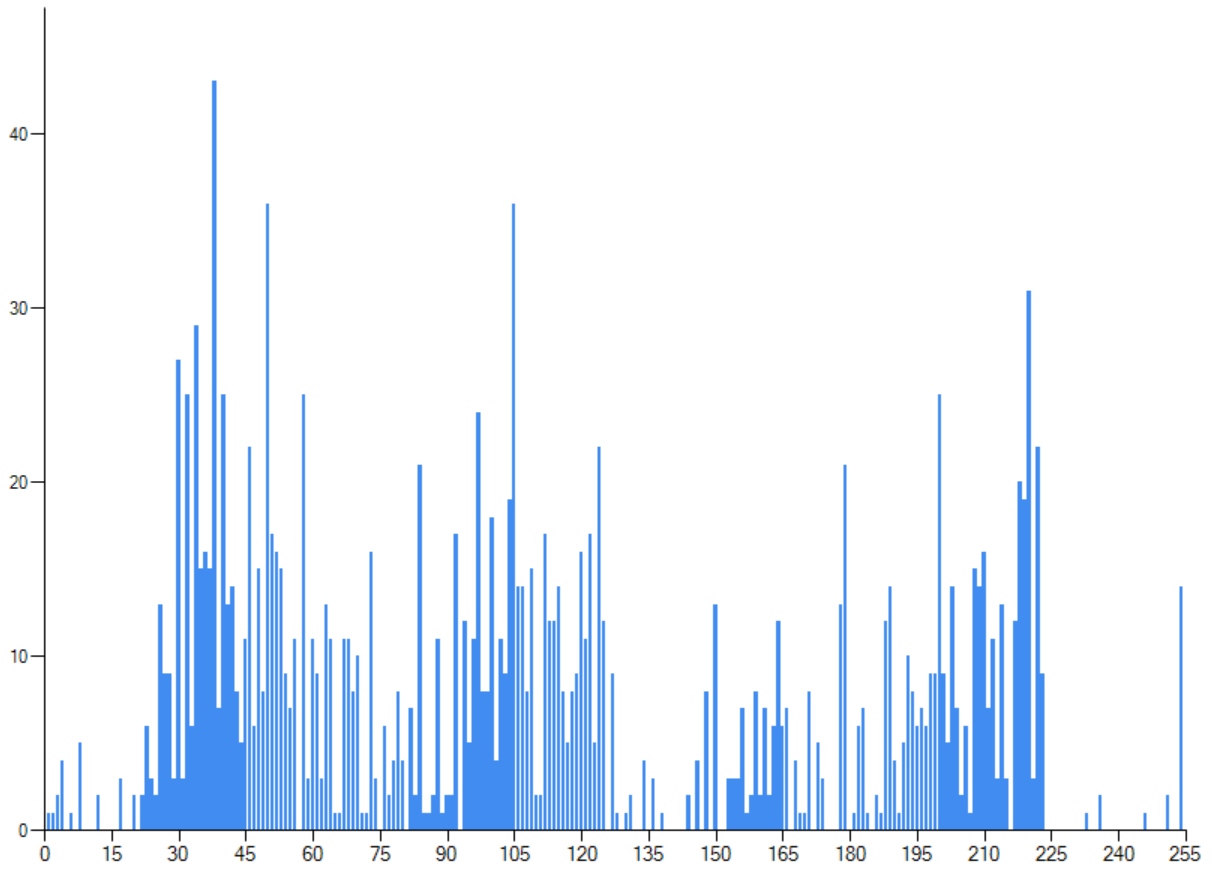


Рисунок 8 – Гистограмма шифротекста 6 раунда, H(x) = 7,075 бит

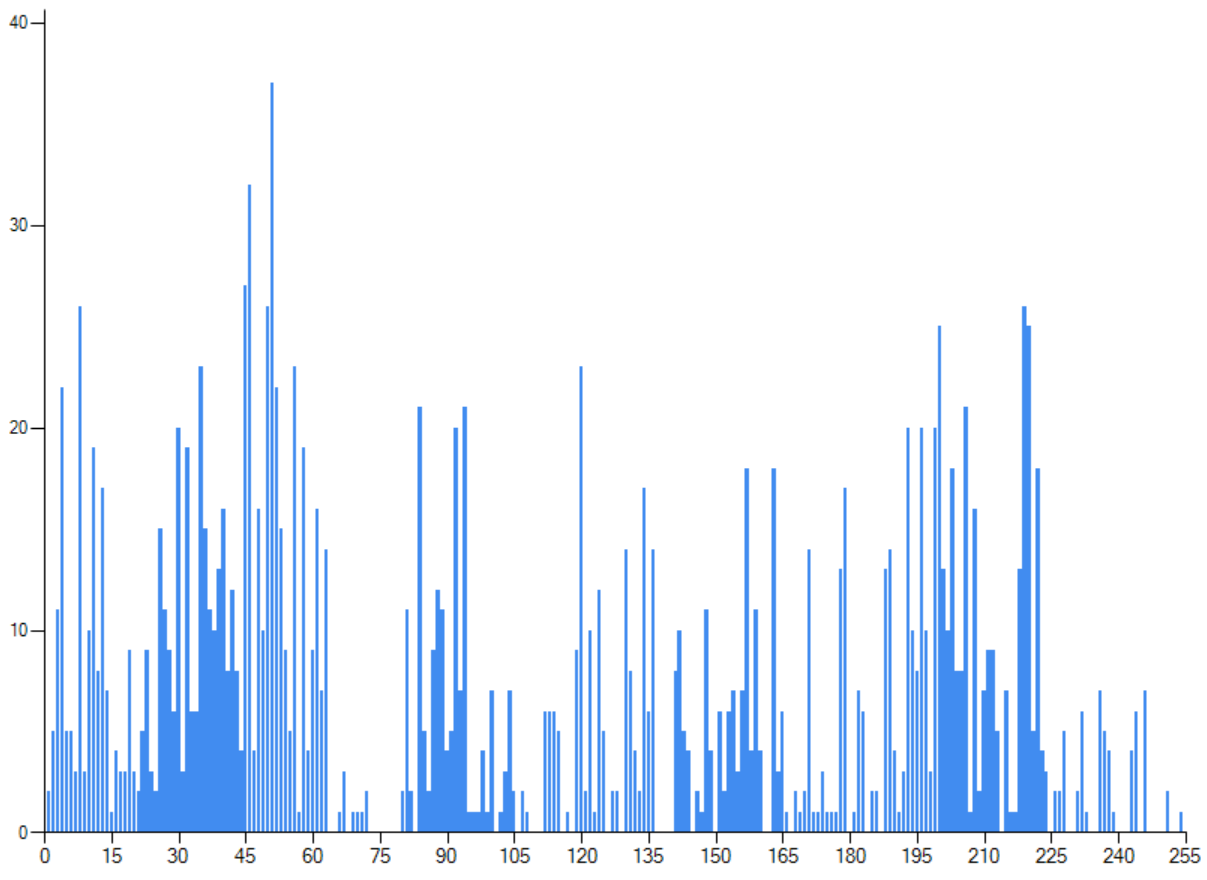


Рисунок 9 – Гистограмма шифротекста 7 раунда, H(x) = 7,175 бит

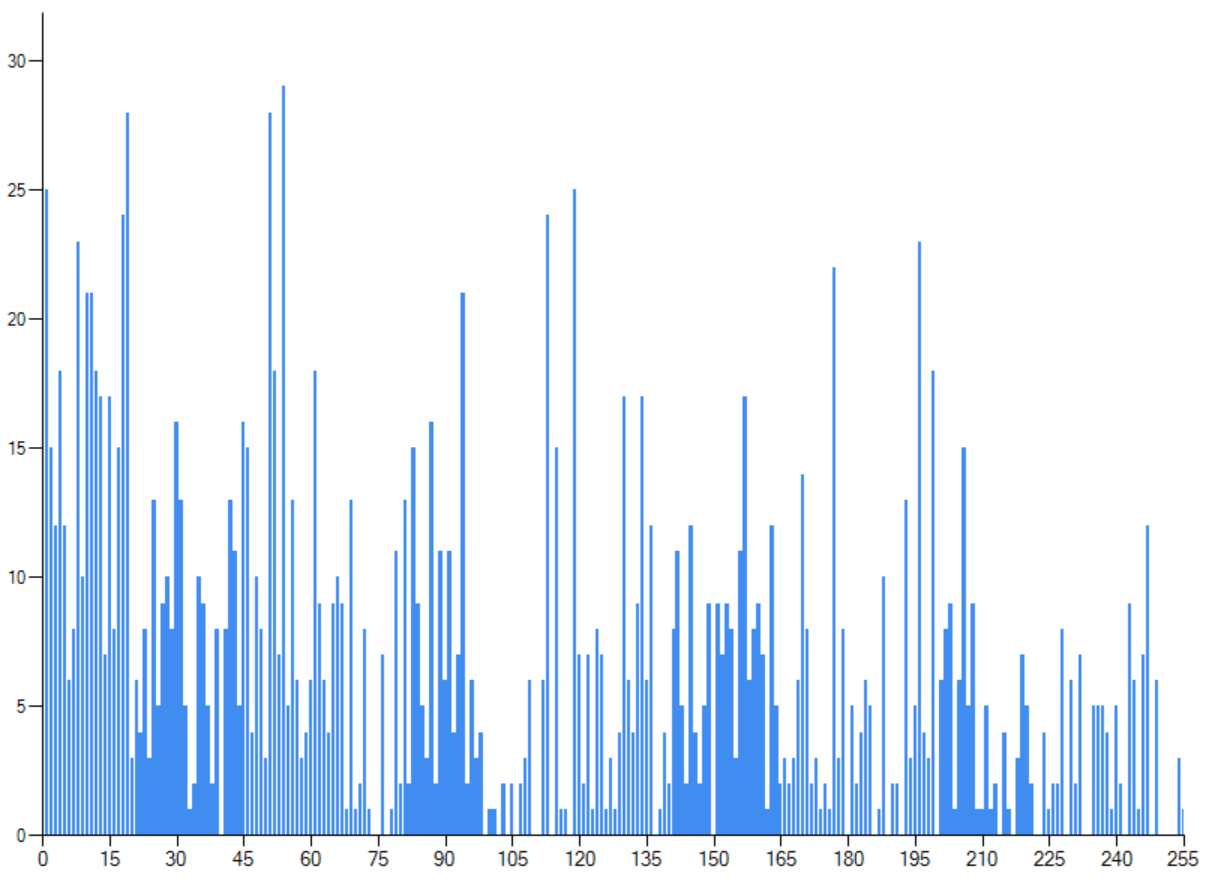


Рисунок 10 – Гистограмма шифротекста 8 раунда, H(x) = 7,369 бит

DES (Data Encrypt Standart)

**Режим простой замены (ECB, Electronic Coding Book)**

Это самый простой режим, в котором 64-битовые блоки шифротекста получают путем зашифрования 64-битовых блоков открытого текста (см. рисунок 4). При этом из одинаковых блоков открытого текста получаются одинаковые блоки шифротекста. Данный режим применяется очень редко, в основном для шифрования ключевых данных, в которых невелика вероятность появления повторяющихся 64-битовых блоков.

Ci = EKi(Pi)

**Режим сцепления блоков шифротекста (CBC, Cipher Block Chaining)**

Шифруемое сообщение разделяется на блоки по 64 бита. Первый блок исходного сообщения складывается по модулю 2 с 64-битным блоком, называемым "вектором инициализации". Полученный после этой операции блок подвергается основному циклу преобразования DES. Блок на выходе преобразования DES является первым блоком шифротекста (см. рисунок 5). Далее этот блок шифротекста складывается по модулю 2 со вторым блоком открытого текста, подвергается циклу преобразования DES и становится вторым блоком шифротекста.

При расшифровании первый блок шифротекста расшифровывается алгоритмом DES и побитово складывается по модулю 2 с вектором инициализации. Результирующий блок является первым блоком открытого текста.

Ci = EKi(Pi+Ci), C0 =init\_vector

**Режим обратной связи по шифротексту (CFB, Cipher Feedback)**

Этот режим был создан для того, чтобы сделать возможным шифрование блоков с размером менее 64 бит и обеспечить зависимость блока шифротекста от его номера. Исходное сообщение разбивается на равные блоки длиной от K=1,2... 64 бит. В режимах за- и расшифрования используется вектор инициализации длиной L. В случае L<64, вектор инициализации располагают в младших разрядах входного блока DES, а оставшиеся биты устанавливиют в "0". И при за шифровании и расшифровании используют основной цикл преобразования DES в режиме зашифрования.

При зашифровании К-битовый блок шифротекста получают побитовой операцией "исключающее ИЛИ" К-битового блока открытого текста со старшими. К битами выхода DES. При расшифровании К бит открытого текста получают операцией "исключающее ИЛИ" К-битового блока шифротекста со старшими К битами выхода DES. В обоих случаях следующий входной блок DES получают отбрасывая старшие К бит предыдущего входного блока, сдвигая оставшиеся биты на К позиций влево и вставляя в младшие разряды К бит только что полученного шифротекста.

Ci = Pi + EKi (Ci-1), C0 =init\_vector

**Режим обратной связи по выходу (OFB, Output Feedback)**

Этот режим отличается от режима с обратной связью по шифротексту только тем, что младшие К бит входного блока DES занимают старшие К бит выходного блока DES, полученного при зашифровании предыдущего блока данных.

Ci = Pi + EKi (Pi-1 + Ci-1), C0 =init\_vector

Гистограммы полученных промежуточных результатов представлены ниже. Полученный шифротекст представлен в приложении А.

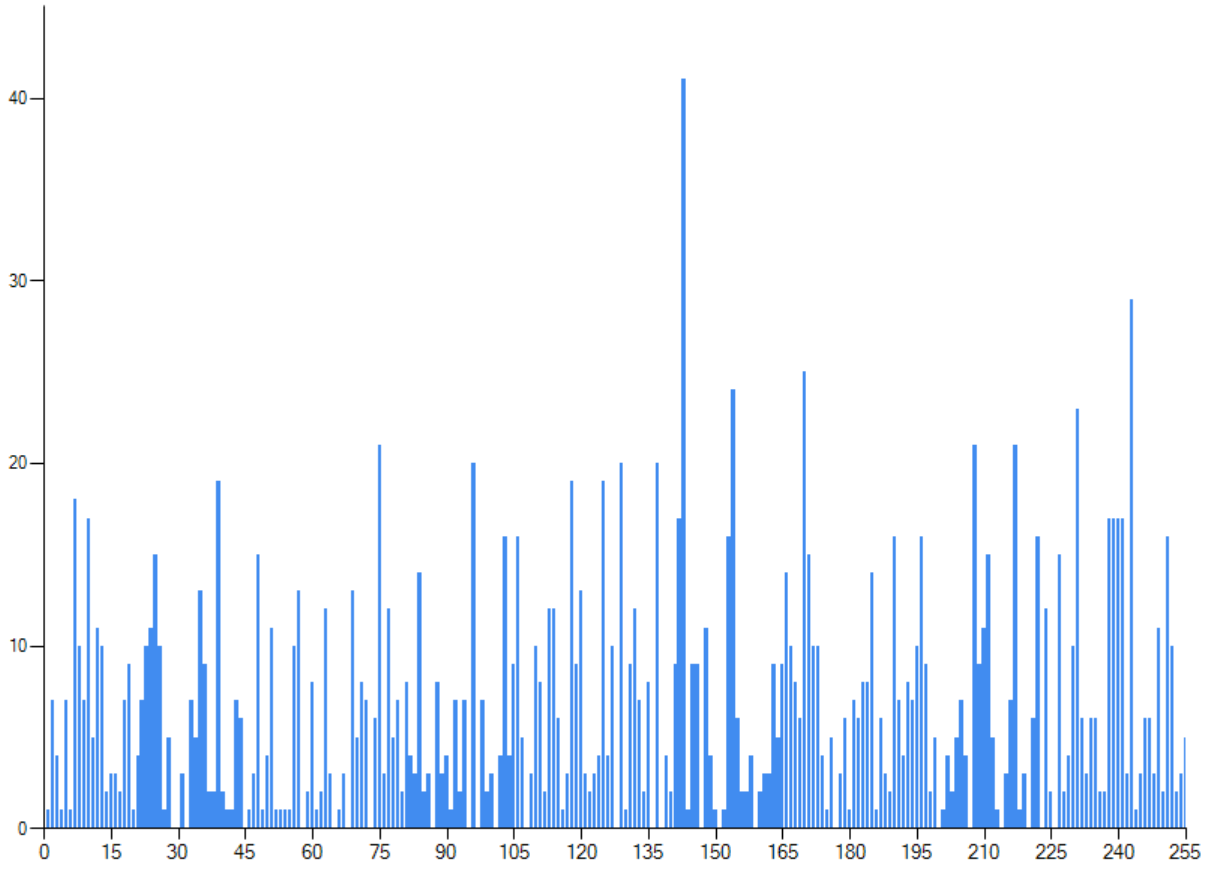


Рисунок 11 – Гистограмма шифротекста DES ECB, H(x) = 7,388 бит

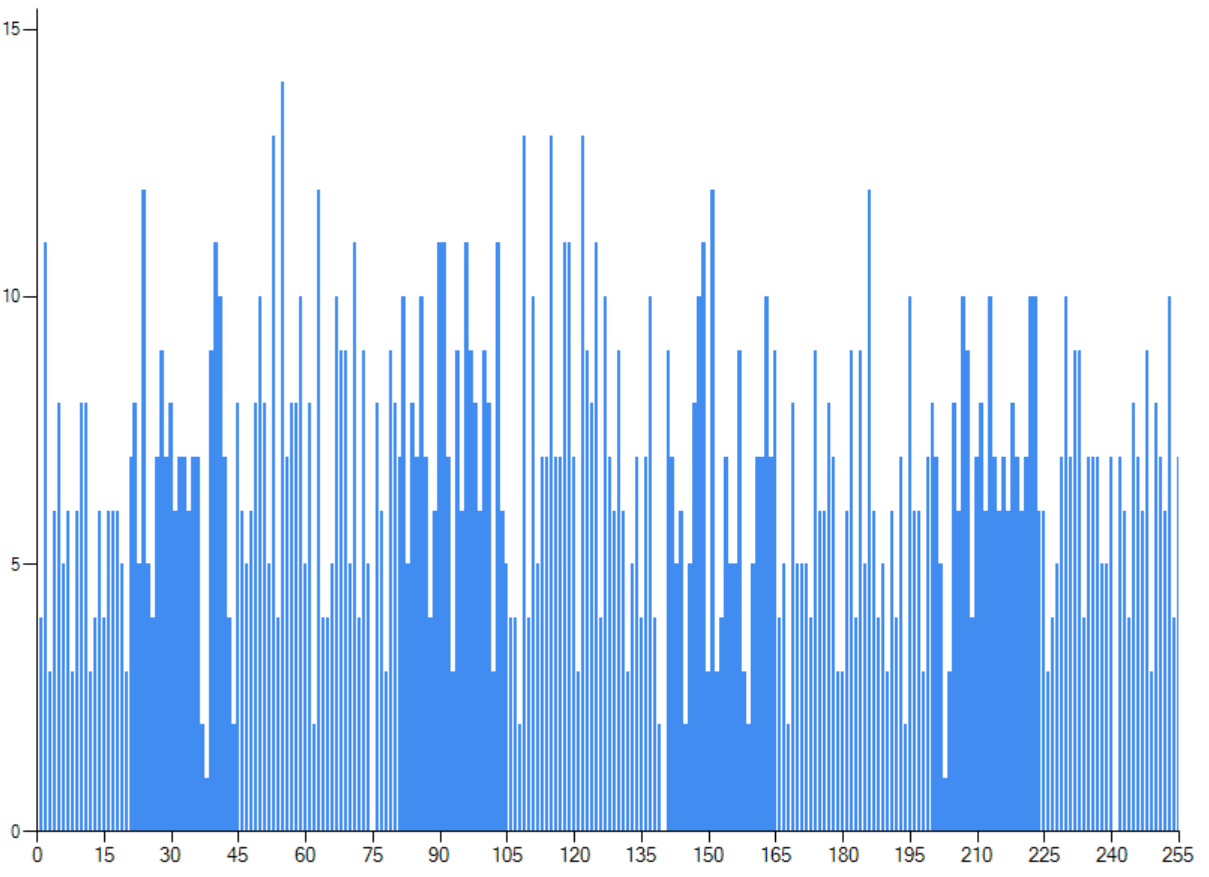


Рисунок 12 – Гистограмма шифротекста DES CBC, H(x) = 7,868 бит

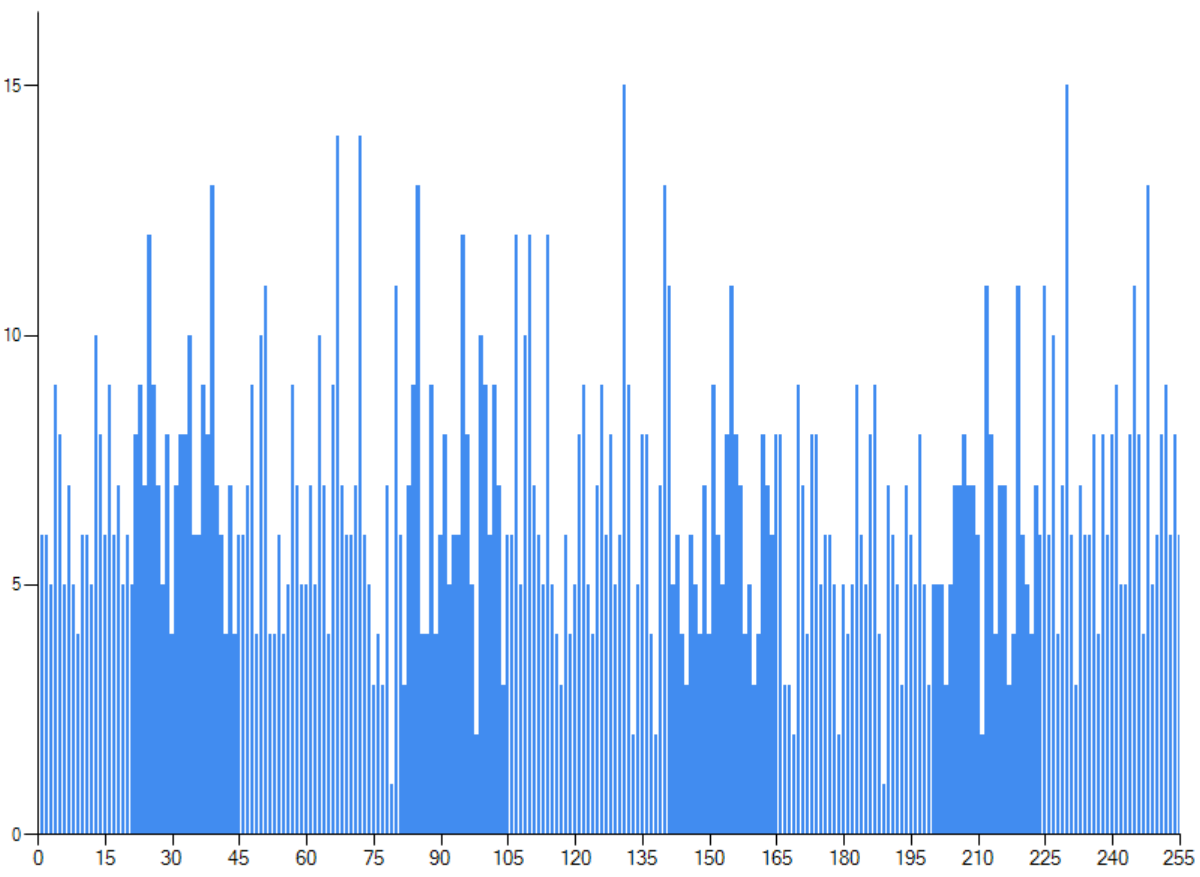


Рисунок 13 – Гистограмма шифротекста DES CFB, H(x) = 7,891 бит

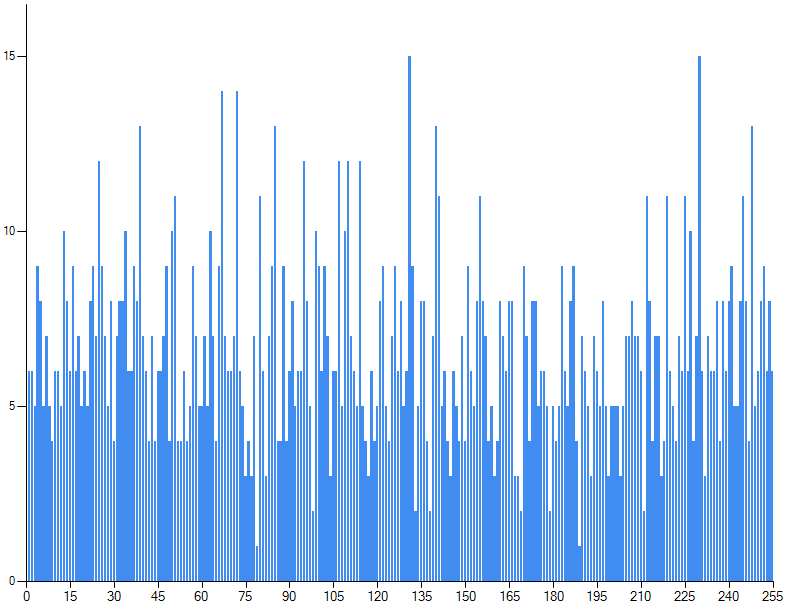


Рисунок 14 – Гистограмма шифротекста DES OFB, H(x) = 7,891 бит

# Приложение А

Открытый текст:

То, что в жизни, наступившей ныне, императоров, видимо за отсутствием империи, уже нет, Тереховым не оценивается вообще никак. Сочинение его, притом что оно сплошь и рядом населено политиками и спецслужбистами, совершенно безоценочно. Вернее сказать, у Терехова, как и у всех вышеназванных, ощущение времени возникает не посредством позитивно или негативно окрашенного публицистического высказывания, но через атмосферу текста.

И атмосфера эта в той или иной степени определяется словом «морок». У Терехова замороченность всего происходящего чувствуется особенно остро, в меньшей степени это есть и у Юзефовича. Аномально поведение главного героя у Самсонова – что, собственно, отражено уже в названии романа. Аномальна, на наш вкус, и главная сюжетная коллизия «Цены отсечения» Архангельского. То есть времена срослись, но, как ни странно, полегчало ненадолго. После прочтения каждой из этих книг остается горький осадок. Горчит по-разному – но горчит неизменно.

И не только по причине невольного погружения в эту выморочную, аномальную, нездоровую атмосферу. Другой особенностью всех четырех книг является острое, порой даже патологическое, мучительное внимание к межполовым отношениям. У Архангельского это чувствуется чуть менее (хотя, пожалуй, он в этом вопросе циничней всех). У Самсонова, у Юзефовича и в особенности у Терехова сексуальные мотивы являются едва ли не навязчивыми.

Самсонов сохранил хоть какие-то признаки идеализма при взгляде на мужчину и женщину, Юзефович – почти уже нет, зато он самый, что ли, снисходительный из числа рассматриваемых нами. В любом случае брезгливые, а иногда и злые описания греховной человеческой породы просматриваются у каждого.

Шифротект, полученнный на 8 раунде:

䖝䁢བྷ孃㜇㘍꧅珄ጐꥸⰐ歫䲏珄䉡꫚ǁ㬊䊎ᄓ侇䖝⬛絽땳矆ř᜽꡹浺ጩ讔՝⤅︭녱꧅࿊膚珄Ḅ䖝ᔶ䏬㸠र՝쐣燇ᄪਮ它Ḅ䢍놣㜉3ɞਮᏛᎰ㴠⬌ళ讔벂䖝䲈饛ἳ埶ግ燇㘈叐㘋꽿闭ᥑ䞎㴋՝ꪆ芙㘈矎㴄饛릃䆍ᥑᥑ꡽纨롱ᰪ嚚ሸ䆍꧅叐䊎ᬤ㐋བྷ惗㠆ᄪ㘈꙽량㄃䊎쐣ɞ뉰㜆ᄓ࿊ᆳ䊎粫䇍燋ㄋǁ࿊놣鋮रᬤꪆ烢㌄ᰵ髊燇1ᠯਮɞ㬓⨚ళᥑ̽㘍릃뎜ꯓ쐣矎酞䏬쾬ጩἳ矎ᴐ饛狼㘈ǁ4珄ꒅᄓ狼㘈燇㌋ıళ㘈벂矎ᨭ4ꝷሸἾᰇ벂놣ስᏛḳ它ਮⲼ珄བྷ䏬⑐ᄓ땳Ḿ䊎飤㔖蕒⬓浪⏃셙ଧ酞彠ᄩ叴⬓놣凓䆆ǁ4ǁ㘈麚ሸ퐁ళᥑᴸ䏬࿐ᨳḄ䢍叐㐊̽՝侇ܕ켪ᆳ䊎粫ř꒐땳ꅷ㐋酞⬗놣벂矎ꅷ魘㸠팂ᠯ벂埶ሸᤈḄ䢍燇ሌ惗㐋བྷ酞쬃鿳놣Ǎళ┅䲏叐㔖︭ਮ叴ግሸ闭ᥑ飤궠㨒Ἶ뇇膹㘈薝ꁸ㘈矎媂ళ讔콚ნ┞ᷜབྷ竑ꅷ鳨飤옂靔㤊ř枏䁢鿳叐ሸꖆ㐋鿳燇쬃놟ళབྷᰪɞ孃ꪆ烢㌄㤉ǉἳ㌄뇇膹㘈薝ꁸꪆ䖝ꪆ珄ı놣놀⺶ǁ콚ნ┞᜾叐Ḅꪆ鳨⨏㼎ἶ＂ㄊᰵ燇浺ᐹళἶꁬ慩ሮ᳝␝잤벂侗ሸ闭㬓⨚ళřز酞4了䖝㘈䖝Ꮫ鎛ਮ叐쬃㌄㘈䖝『芙ሳ뾂བྷ쐣ꪆ㴋誗량矮酞芙矎ꅷ䁢ళἶ燋놟㴋麚㼎叴ᬋ鋮㴆ߒ燇ሸ뎜浺鳨벂ㄍ쌃麚酞ꆅܕ侗屢䞆饛矎맊ᄩ㘍ἳ飤枏㘈콚侇咈ꯓ쐣㼎ވ㐋량㦱쐣凓㴄ᔿབྷ『矎녱咈쐣珄뺔㴄⟠ᷜབྷ『䢄ᰪళἶ놣叴ꥸ놣း侇㴠㰍⿃셙㘈薝ꁸ㰍⿃珄㐉㴋ꅷᤈ鿧ᄓ땳Ḿ䊎氲淕䋯䆆燇ሌ惗㐋酞⬗鿧綪粫埶Ḅ<粫燋ਮࡔᨳḄ䢍燇ሸꅷ꫚矎侇䆆놟ɞ貘ግ㐊墜뉰馒䖝䎞咈㔖ꁸ㘈ɞ┞⤅㌄讔燋ǍỘ『㐊र՝᜽㘈ᬤਮ걼侗룄慩ሮ᳝␝잤벂叴ግ埶ሸᤈḄ䢍埶쒦闭ވ쐣ɞ௄᜽Ⲽ矎육뾂ꩵ䖝㤊놣叴ግ՝̽䲈酞ɞꝷਮ꙽䲏놣簿侗룄옂靔㤊䇍࿊鄗굻⬛瓠ǁ놣燇ሌ惗㐋酞ᬋ࿊ᆳ䊎粫ř蕒ᠤꁸᰵɞ땳ᬋर埾준䂎浺鳨媂它翂ᥑ쐣珄燣ᘿ咈रᔅ浪攒䆍ሌ㘈孃ሌሾ㌄㜊双᜽闭Ꮫ쌃賳ግ矎녱㘍ㄋᥑ্芞4䒄ř䲈ᥑ䦄㈇汤ɞꪆǍᰪ咈㰍ǁళ꺂㰍䖝궠㨒Ἶᆯ矎㴠ᬋ࿊膚珄Ḅ䖝꒐ግ燇⩆☢ጆ뛎埶ግ翂꧅叐ਮ灷Ί饛␝⟠ꉽǁ⑐咈롱řᄩျ薝쬃狼뎜ጆ鋮ꪆᔅ侗믣ᐺȼ՝롱숃뎜뎠䏬躛4र꫚셙ǁ㘈뢘řᥑ뺃ሩ燇Ⱀ☢ਮ鳨䢄粫㘈麚㐊ꖆ뉰䆆矎侇Ί怬䲈酞薝쬃狼ᄷ浺鳨飤Ꮫ鞓ᷜ량

Шифротекст полученный шифрование режимов ECB DES

пx…wљv›ЃЙ\_x0016\_'ЮжЭ­KдEЖЈЏЁTфеЌ­KPаБДщ?qрёД$Ј”пxЌOPGц?‡§зiRvUЪи™ё¶‘^оw$уЋ‘жЭТGь&­K«гпxЏкыЃ9зыао’Є'зѓ·уTЏ3«г9љЏµ№ъB`¬уT!№C'јаµВ™Эj\_x0019\_‘^\_x0010\_ЩѕnпxРЄ?Ћ•9g`'з‰}ГdщќжуЏЮ\*2Ґ0оУ™J­Q‰}с¦+y?ЋўKяЧЏЮЏЮМ{h§кК0HРуяЧжЭГ”qрЕ3љЄљvФ\Ѓ+ѓ·‰}э§ҐwВqр’Є`¬rч7;ёДйТGvшqрѕ<”uFqаf…ДщТGЏ™JыаЕ3Уї„X№\_x0016\_пОЭ'зЊ…уT`¬‘тЛТj\_x0019\_ЏЮШр'ЮўK0+K’Єс¦ЩEыЃЕё¶•9с¦®?ЋЄK‰}ДщН­KкіёДЄK‰}'злf^Бj\_x0019\_‰}ѕnс¦Z№Нv‡Ру3ўЏs|ѕnЏJ2!№чеЏ3уTм›­KљvооыЃh®ёДц?Г†qрЃМ\_x0012\_·»љѕ»Vgr¤\_x0012\_ѓ3ЩEЁФHS#СљѕЏпЃЃДщНДщ‰}ы,Руrnj\_x019\_ЏЮcя«г9љГ‹лШрољs$ЈЮ«GЩYvшqрѕ<”зiЎбц?gгоољЄЩEїjЏ™Jѕnс¦gг„О9з°д…ѕngOРРуїo«г9љ'зЕрФ\љЄљvЩEQbZwЏOKj\_x0019\_zЗеЌГМ\_x0012\_ууT#С`\_x0010\_ЩРужуЏЮЃXѓ\_x0019\_¶”/T‰}ЩЏkM‰}с¦”ѓІj\_x0019\_‘^MMњHхNљvћ™gгЏЃТБpЗћ©”зi~2…wZwГО„Ру8љЄZw'зQboУj\_x0019\_љvК0`¬›ЃУї„X№%Ћ§ц•9X№</T‰}ЩЏkMУпxУ­K^БЏ?gЩ8ДщMMњHх6Гј)«гУЏs|Ръ\_x0016\_і8¶EЎ\_x0016\_п'зи™юіj\_x0019\_8sыЫЌУgВ8«Р>]s|ѕn"„\_x0010\_ЩРужу‘тЛТj\_x0019\_зii\_x0012\_ЩEНcпx‰}пx!№VLуTГQbX№‰}пx™Ё­Qж«°·љv’ЄУҐ0сҐ§ҐХЈЩE­Qс¦gг…wj\_x0019\_8qаoУҐ0ы,\_x0016\_і#С‡#\‰L0'зРу0и™Џѕnuьы,ЩE··Ю«"„¤h¤,?Ћс¦\_x0019\_рHS'Ю•9Ѓ~2‰}MM$ЈrЋ+K’Є\_x0016\_і{FљЄ§Ґ(’ЄпЃ+yН~љv™Ёс¦Ћ‘rЋ’Є­K@'+yЌчNљv™ЁЗ„К0j\_x0019\_8Џ#СЖЈЏ1ё$ЈµВШьн¤\_x0012\_‰}ЩЏkMШьн­KznҐ0gгїo9АёДц?Г†qр~oАMYВЃ'зЕрФ\љЄЩEїj9Аі5ѕ<g«гYѕ<qауTуrЊF«г9љ'зРуgгБс¦$ЈЃѓmoУ`¬@j`‹лf‡ооrч©пxЅµrЋМ\_x0012\_kM‰}`¬хw$X№‘^qаOKµ™Ё‹лыаоRv‰}Е3уT!ј"„ЫЌУgВ8«Р>]s|ѕn#С`gOРРуїo«г9љgќ/жу{F’Є`¬›ОRvм›с¦Ѕ№°·;пxћ©Џ#С`оШрРЄЩE`¬v‡уTэеЌЏ™J$"„ТБpЗћ©”uFТGjС¶й§ЌOmtДщЏ'зЕрФ\љЄЩE‡#ТGvшqрѕ<”зi™J·»+kM\_x0016\_п`¬ц?‡#ыа.»3Єл`и™ЏѓІЏ30CЏЮ’Є­KУтrЋыаN»VЋ яЧЕр‰}›ЃЕрLmX№4[Rvжу!№dьђ`с¦Ћ‘'Юf…ЏЮ%8ЫьН(ЇзiРЄЏЮµ®Бй#x`¬УOKК0rЋШьДщoj\_x0019\_!ЏШьпxXѓ\_x0019\_¶3ІТс¦µВ‡#ТGь&­K«гпxЎб`'з,bю©сСg`0CжЭГуT#?Ћ«РЌч°Дщh®rЋ§кзiHS@ьЩЏQbЄK0©с\УN"„к^Ubо§к»и0ЁДыЃ„°НыаБ¤\_x0012\_Дщ‰}С=зiЏЮМРА'зЏЁюуTЏЗ„ѕ<”‰}ы,оо‹л8 ооrчЃс¦$Ј#`HРЄЩEЩЏQbЄKґvи™ Ѓ!  N§Ґ

Шифротекст полученный шифрование режимов CBC DES

зuјg{‚\(Ґ`ГЮ$CNуЛ;5(ъ±CЧя;/hиь\_x001D\_‚ќH}QSФGaGБчЕFiКЖФЦЧ7оfF<Ррь7'IЙP№qM·»"\_x0016\_¶ТуЦ?i6\bы\_x0012\_5•\_x0016\_ыF?мУZE\_x0012\_‚xA/®°)ЗбьRХо8Рт|W8ёTCЭuЙ^"\_x001D\_зэde®¤ИDтQП^Р»)gOµ„1Ѓp{^¬?7\_x0016\_внФ†‑©хt€SuО€Є“±Ў~Љe\_x0010\_ЏИо5[·тґ\_x0016\_Rc\_x001D\_U<hш?\_x0016\_$РбSїhцрОbЃzхЈUМђ!ВСИJwІzi9«d?&­аX}жшЧ‹Гр9Е=b…»{№“V3:­z^ЌZU'µњq9ЇЬ>a@П«±smjКйo?ИІ)чЪГ|v\_zи\*РЦPmE±’t’НдЧd'еаЁqGЂяaф\*/±ШTЭ®‑1n^\_x001D\_dя<5ЭєПaz)ЧаЩм(‡Yj8PHеЅЮ$sи;ЂЈЬ‑lЅЈ¬ц””¶EwQаЕѓ@БWYпЦ7+™ЪВ{?ЩАdщяbOІz!ҐO¶л ”G=sРЮ=7\=\_x001D\_rgкMLRл'5\_t:(яЫЂD[)Eўм”ЈНЮ!ўєA?Rё3Я31Еµзпr`GпН©ќx¬\_x0019\_~µ)тEЎ}P|•1zгpњ§П|У‹АЯbtз¶ьЙ°v‰ҐЋв ›хнЯ$З¤\_x0010\_rДБzҐЎд/…~hЮ"БVA:ѓQEFЫђаХГWa№вw#d\*цж‑mUovж­¶ЪЏmгrСuИDЪ‡\)Џ®\_x001D\_DКLНgdЗ(љ\_x0019\_©GBкCЦ‑‚ю­\_x001D\_єo5Vz?еO‰mъWЅ‰#5pc­О]їКд[зь8<“ѕљw­\_x0012\_я§Сa(PййЖНПЌ{ЫV .ХjUeэCgЅ‑;EЎ\_x0010\_•M¶\_x0010\_\*V(}eЂќҐmЯіPT'†6IeЈ’fФbґх¤CЗНОzёV¬ҐЈ¤|кэжё\_7ЬОљ‰.УЯL}Ь#§ГхЩqaњ•эїќјЋЂk\ҐX@+ш]8ЋГќD\_B»s›L.ПD},cИы†ХмЭрж[\_x0012\_„;ў·ЦЙ™2qнu1ђ¤‑U\_x0010\_\_^=m[ЗZxT!†ЙЌ•­w†БЪL}Ь#4`ћSTZЄљб±ЈbҐјБ0х;ЌДнёхтТё!цљkм\_x0019\_щЌхч’3RЇ!`vецШg„•ЪЎr'µ:л\_x0016\_#±рџL`©‰ѕeBsч‚YoIЙ¦эПZZ©hф2їµ\_Ш9YЬ=Э°АтhШp°0Ф¤‑yZa”ТќGЃVIТЃU°жw¶гХДaГђ»¶™к5шЈиM/йsfI‘Ф|є:%ы©z1? ]jЙDsЇшшњъ‰СюЭHњB76Z`.Q5®vd‡7“ЌТGw•љє\*Ў(qІѓЦЩэB4GG{ ©§oR~у7…®ъшvRN¦#©Ћѓ‘CЏ' Б7№>ЪP9$m8ъбMTиќЁbmп`|рР‰4л^ЕЯv1ъё(Ћ#Ql†»­2Ш"GиuЃrћ;“чIу[ђ25дчЄАFПцVQ=J[\*лЂ’›У6?I·ЋRgў^єђges€YO…\_x0019\_sб}ЪxэйХJжmя:РлЗHЉєkЇ®1v7Og\_x0016\_є.ЮzЩьў[X\_x0010\_иїWwЎЮ'Ѓiж3‚±эgе”Mуые‡7u•§5ўџ3p4\_x0012\_nOУіZ€|0р“ы}tГпsзЏ2Cјћўќ;€бCЉS)Уwo59,фХ94†ЫДc\_x0016\_Ќ{ЕфТJГ2<Іw)%\_x0012\_™аVй.Щ y”їЫН‰ІX‚)иПшЈР2Жox5…Я;Ят Іyєє2Ы•«J(7D[¬Д[@{ЮDLЉCИ`\*oYT•t^3x›r'‰ќ=н[n8ѓkн?®NХ(:йIsW€і;m+­ЭП`R‚€9d!EщЮыэv®Єл"o}””“мЗушёюEzT`цOМZsэъ2}OPWъЇ2$¦ТШйsn0vZoю{\_x0019\_"ё­ЄЮЅн$ґУ“ѓзЯ‰mй +оЫ«ГmШwҐcУ¤К`vљ•3¶7меIиcq›«ЌєѕЯ№LДоЂ‚НЋХ—:\ж дVtR0ЇжгИХ\iєОAЈ

Шифротекст полученный шифрование режимов CFB DES

qiEЌHвx~њ\_x001D\_·=Еб’уА§9.Ч?A:жҐжШсгYЄБ!\_x0016\_‰:хћz>йUОIґ’й”ЏzyCфЏЊ¶з0$&Ц0n„ШТ­TjїЉ­p“#т\_Вж+ЙП"Дd&Ю­H)dBЂ[ й/­жб®юmыЯ•Х>gЃґ‡џqrЙй!\_ћ"O|ЫFЬY\_x0016\_1Чжо)мРћ2Є&ХkѓH»х"›Fr|ф`\_x0019\_\*T\_x0016\_P\_x0012\_Н¬%¦ХЏўЋCuW\_x001D\_\_x0019\_Ќхs$Ќ›Ћс»»%#Ґ;7$Ђз3ГхXCјљЙl«Њ@@"ф†љMЖЕп°#\*Ьвk№э\*л=E„}Ѓ·сUST·Qoў%’ҐеЩП“`Nrђ&мђT'j„ЖєZ’;Ь”rKжгk!\_x0016\_фнХЈc€'K‹xєП'LПь&њ®ѕHvкЬЕ¤®вIp±m\_x001D\_~/\0\_x0010\_»Аqf\_x0016\_NUњ2€\_x0012\_шiезѕь·zьуЗfDсѓўХmоX\_x0019\_ѓj°Ьяж@pё"ь\_x0012\_цУЇҐрІ„З¦HCТagyЮIѓџЮaќож‚'Ъ/ѓ‡p2ќ`3ИЈШ‡ФЕQrщПє\_x0019\_у"Єѕ©0%0^,n\_x0016\_‘Ќ@?°\_‚К3ѓи•»њ\_x0012\_u'С°8јm\ЫЭ=‚ВЈћл^ЊЫ‡бФ1TQl"њб ?fF·0мс+ЎBЙfЙzQЁbЈ'y[@Ў,“ЋNP”Ья=ЫzТCHџД¤©¦.†(kЊn>cr6п¬)ЧыD„О`ZIN’ъ~U¶[:ҐA<#єµЕёырВFЯОYэP/щР\_x0019\_ўтemUН5Мх~\_x0019\_;югѓgДС‑юд«Сљm·ЯGѓ(¦(Ій…±?µєкЫ¶ѕКPя0їЫ\_x001D\_%У<ЖкG‡8И°6~AА?9ґdЯ(ь­р­ќq‚жь6ж«Њн9Jя'т?\_№¤ЪгWnЂ‡3Е„ТNя¤\_x0010\_hj™џ‘ўбш®Gѕgї\‚њтр“ёО«<9ЧJ™шыч«AГCЌ™|H+s\_x0019\_¶®SЌnz|СоњЗ•@nђ™h(vм4keМрй\_rЊ}Бф2kMЕЏ)Оъ~BѓЏ\_x0019\_H­>t6uёD^Ct‡J•UеXmkКzTѓvшЌэФ(z€#U3юыR8¬ИНшфИK`БбdФ­HPќґ±Ч€DхSEх\_ФЭ~?\_x0010\_ВЁ?д9QЌ]„Оеє}Т\_&'fP№&иЇo9Ё\_x0010\_\_‰vОЮГХc3V§S‹.б}аk\sj\пҐ]ѓrЫП$5x63xќЌ4а«Џyi„aГхMх\_x0019\_vп‑S+\_x0010\_tТЃLЦ›є€щн)ZХъў›Ыо?CZЂҐ}ш,nP.{пcXеІ^mв¶Ї®лЊ­ЪЊю±o¤X2ьЧ\_чМаЊлоC‑•;[qіѓ2‡кљjй5n4HlC¬$d]јшШ=l:ФЕЛ8Zc°f†@&Bп`љ0cЊшC!cЂҐд€гЩ3›‑тk2НBљJЃB\_x0016\_ђВ›X%‚]їПoж„rГЄ+ю'«9.МЄyѕe­Ј3 NЎPy›6f]"¤Вz‰s[g"{Ш~}dЇРЂvTh"ШљцЭѓчЦЄt{ !XF'бІw¦ШPъµUФ9`рBцхYдl\_x001D\_=мс?¦HгЋVїGrЖs>Сг›CЈцаfѕ‹\_x0016\_<РCеДJg3Dъ~}рi3ёжДGЛќЉм<eEшюрМз/у‹к^U¦aэPжЩхЄ…оi ›»x­чэ\_x0012\_ЦюH#Fѓokw·\_x0019\_ґНІСуcФ'ЫЭ\_x0010\_ўшkџ2!ХµTN\_x0012\_pьнФVm‹Њ%Кoз4nc·:РлёФя»с^­ЯўHЅП\_x0019\_А9ш+W%2Я:‘Л+e7\_СЫк\*·ц¦ЊэkUм!{)жdRГ7оЊжЄS®BnИ\_x001D\_wиљбі­®\_x0010\_de$IHа8НлX›в/Ж.Ђg0\_x001D\_§ш5 є•EIЄU”rыцD]!Ќ\_x0012\_г цpS†,UyРќВER(B€бы\_x0010\_‹nщ2±њbi•™ЋXfdбЂАКгѓЧь%\_“’\_x0019\_1PФїcTщ№wвм/:GыU=шn±НoсW[зЎ[цфЯс†Da\_еy›`GV»C\_x0010\_;[LЫЈъЃ'7mј‰‹№Ъ1»Ќгr-Q'ЇZ€­Lф\_x001D\_

Шифротекст полученный шифрование режимов OFB DES

qiEЌHвx~њ\_x001D\_·=Еб’уА§9.Ч?A:жҐжШсгYЄБ!\_x0016\_‰:хћz>йUОIґ’й”ЏzyCфЏЊ¶з0$&Ц0n„ШТ­TjїЉ­p“#т\_Вж+ЙП"Дd&Ю­H)dBЂ[ й/­жб®юmыЯ•Х>gЃґ‡џqrЙй!\_ћ"O|ЫFЬY\_x0016\_1Чжо)мРћ2Є&ХkѓH»х"›Fr|ф`\_x0019\_\*T\_x0016\_P\_x0012\_Н¬%¦ХЏўЋCuW\_x001D\_\_x0019\_Ќхs$Ќ›Ћс»»%#Ґ;7$Ђз3ГхXCјљЙl«Њ@@"ф†љMЖЕп°#\*Ьвk№э\*л=E„}Ѓ·сUST·Qoў%’ҐеЩП“`Nrђ&мђT'j„ЖєZ’;Ь”rKжгk!\_x0016\_фнХЈc€'K‹xєП'LПь&њ®ѕHvкЬЕ¤®вIp±m\_x001D\_~/\0\_x0010\_»Аqf\_x0016\_NUњ2€\_x0012\_шiезѕь·zьуЗfDсѓўХmоX\_x0019\_ѓj°Ьяж@pё"ь\_x0012\_цУЇҐрІ„З¦HCТagyЮIѓџЮaќож‚'Ъ/ѓ‡p2ќ`3ИЈШ‡ФЕQrщПє\_x0019\_у"Єѕ©0%0^,n\_x0016\_‘Ќ@?°\_‚К3ѓи•»њ\_x0012\_u'С°8јm\ЫЭ=‚ВЈћл^ЊЫ‡бФ1TQl"њб ?fF·0мс+ЎBЙfЙzQЁbЈ'y[@Ў,“ЋNP”Ья=ЫzТCHџД¤©¦.†(kЊn>cr6п¬)ЧыD„О`ZIN’ъ~U¶[:ҐA<#єµЕёырВFЯОYэP/щР\_x0019\_ўтemUН5Мх~\_x0019\_;югѓgДС‑юд«Сљm·ЯGѓ(¦(Ій…±?µєкЫ¶ѕКPя0їЫ\_x001D\_%У<ЖкG‡8И°6~AА?9ґdЯ(ь­р­ќq‚жь6ж«Њн9Jя'т?\_№¤ЪгWn‡3Е„ТNя¤\_x0010\_hj™џ‘ўбш®Gѕgї\‚њр“ёО«<9ЧJ™шыч«AГCЌ™|H+s\_x0019\_¶®SЌnz|СоњЗ•@nђ™h(vм4keМрй\_rЊ}Бф2kMЕЏ)Оъ~BѓЏ\_x0019\_H­>t6uёD^Ct‡J•UеXmkКzTѓvшЌэФ(z€#U3юыR8¬ИНшфИK`БбdФ­HPќґ±Ч€DхSEх\_ФЭ~?\_x0010\_ВЁ?д9QЌ]„Оеє}Т\_&'fP№&иЇo9Ё\_x0010\_\_‰vОЮГХc3V§S‹.б}аk\sj\пҐ]ѓrЫП$5x63xќЌ4а«Џyi„aГхMх\_x0019\_vп‑S+\_x0010\_tТЃLЦ›є€щн)ZХъў›Ыо?CZЂҐ}ш,nP.{пcXеІ^mв¶Ї®лЊ­ЪЊю±o¤X2ьЧ\_чМаЊлоC‑•;[qіѓ2‡кљjй5n4HlC¬$d]јшШ=l:ФЕЛ8Zc°f†@&Bп`љ0cЊшC!cЂҐд€гЩ3›‑тk2НBљJЃB\_x0016\_ђВ›X%‚]їПoж„rГЄ+ю'«9.МЄyѕe­Ј3 NЎPy›6f]"¤Вz‰s[g"{Ш~}dЇРЂvTh"ШљцЭѓчЦЄt{ !XF'бІw¦ШPъµUФ9`рBцхYдl\_x001D\_=мс?¦HгЋVїGrЖs>Сг›CЈц{аfѕ‹\_x0016\_<РCеДJg3Dъ~}рi3ёжДGЛќЉм<eEшюрМз/у‹к^U¦aэPжЩхЄ…оi ›»x­чэ\_x0012\_ЦюH#Fѓokw·\_x0019\_ґНІСуcФ'ЫЭ\_x0010\_ўшkџ2!ХµTN\_x0012\_pьнФVm‹Њ%Кoз4nc·:РлёФя»с^­ЯўHЅП\_x0019\_А9ш+W%2Я:‘Л+e7\_СЫк\*·ц¦ЊэkUм!{)жdRГ7оЊжЄS®BnИ\_x001D\_wиљбі­®\_x0010\_de$IHа8НлX›в/Ж.Ђg0\_x001D\_§ш5 є•EIЄU”rыцD]!Ќ\_x0012\_г цpS†,UyРќВER(B€бы\_x0010\_‹nщ2±њbi•™ЋXfdбЂКгѓЧь%\_“’\_x0019\_1PФїcTщ№wвм/:GыU=шn±НoсW[зЎ[цфЯс†Da\_еy›`GV»C\_x0010\_;[LЫЈъЃ'7mј‰‹№Ъ1»Ќгr-Q'ЇZ€­Lф\_x001D\_

# Приложение Б

Листинг файла MZI\_4.cpp

#include <iostream>

#include <string>

#include <fstream>

#include <Windows.h>

#include "DES.h"

#include "Feistel.h"

const std::string outFile = "data";

const std::string reset = "\033[0m";

const std::string red = "\033[31m";

const std::string green = "\033[32m";

std::string ECB\_key = "Пустой колос голову кверху носит";

std::string CBC\_key = "Пустой колос голову кверху носит";

std::string CFB\_key = "Пустой колос голову кверху носит";

std::string OFB\_key = "Пустой колос голову кверху носит";

std::string feistel\_key = "Пустой колос голову кверху носит";

std::string CBC\_IV = "аб";

std::string CFB\_IV = "аб";

std::string OFB\_IV = "аб";

void DES\_menu();

void ECB\_menu();

void CBC\_menu();

void CFB\_menu();

void OFB\_menu();

void feistel\_menu();

std::string cipher(char mode, std::string data);

std::string get\_data();

void save(std::string data);

void test\_Feistel(std::string data, std::string key, float entropy);

void test\_DES();

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

char choice = 0;

while (choice != '3') {

system("cls");

std::cout << "1. Сеть Фейстеля\n2. DES (4 режима)\n3. Выход\n>> ";

std::cin >> choice;

system("cls");

switch (choice)

{

case '1': feistel\_menu(); break;

case '2': DES\_menu(); break;

}

}

return 0;

}

void DES\_menu() {

char choice = 0;

while (choice != '6') {

system("cls");

std::cout << "1. ECB - Electronic Coding Book (Электронная книга кодирования)\n2. CBC - Cipher Block Chaining (Цепочка блоков шифрования)\n3. CFB - Cipher FeedBack (Обратная связь по шифру)\n4. OFB - Output FeedBack (Обратная связь по выходу)\n5. Проверка корректности работы режимов\n6. Выход\n>> ";

std::cin >> choice;

system("cls");

switch (choice)

{

case '1': ECB\_menu(); break;

case '2': CBC\_menu(); break;

case '3': CFB\_menu(); break;

case '4': OFB\_menu(); break;

case '5': test\_DES(); break;

}

}

}

void feistel\_menu() {

char choice = 0;

while (choice != '6') {

system("cls");

std::cout << "1. Зашифровать\n2. Расшифровать\n3. Изменить ключ\n4. Вывести текущий ключ\n5. Достижение задаваемой энтропии\n6. Выход\n>> ";

std::cin >> choice;

system("cls");

switch (choice)

{

case '1': {

std::string data = get\_data();

if (data == "")

break;

std::cout << red << "\nИсходный текст:\n" << reset << data;

std::string result = cipher(0, data);

std::cout << green << "\n\nЗашифрованный текст:\n" << reset << result << std::endl << std::endl;

save(result);

break;

}

case '2': {

std::string data = get\_data();

if (data == "")

break;

std::cout << red << "\nИсходный текст:\n" << reset << data;

std::string result = cipher(1, data);

std::cout << green << "\n\nРасшифрованный текст:\n" << reset << result << std::endl << std::endl;

save(result);

break;

}

case '3': {

std::cout << "Текущий ключ: " << feistel\_key << "\n\nВведине новый ключ\n>> ";

std::cin.ignore();

getline(std::cin, feistel\_key);

break;

}

case '4': {

std::cout << "Текущий ключ: " << feistel\_key << "\n\n";

system("pause");

break;

}

case '5': {

std::string data = get\_data();

if (data == "")

break;

float entropy = 0.0;

std::cout << "\nВведите целевую энтропию\n>> ";

std::cin >> entropy;

std::cout << red << "\nИсходный текст:\n" << reset << data;

test\_Feistel(data, feistel\_key, entropy);

break;

}

}

}

}

void ECB\_menu() {

char choice = 0;

while (choice != '5') {

system("cls");

std::cout << "1. Зашифровать\n2. Расшифровать\n3. Изменить ключ\n4. Вывести текущий ключ\n5. Выход\n>> ";

std::cin >> choice;

system("cls");

switch (choice)

{

case '1': {

std::string data = get\_data();

if (data == "")

break;

std::cout << red << "\nИсходный текст:\n" << reset << data;

std::string result = DES::ECB::encode(data, ECB\_key);

std::cout << green << "\n\nЗашифрованный текст:\n" << reset << result << std::endl << std::endl;

save(result);

break;

}

case '2': {

std::string data = get\_data();

if (data == "")

break;

std::cout << red << "\nИсходный текст:\n" << reset << data;

std::string result = DES::ECB::decode(data, ECB\_key);

std::cout << green << "\n\nРасшифрованный текст:\n" << reset << result << std::endl << std::endl;

save(result);

break;

}

case '3': {

std::cout << "Текущий ключ: " << ECB\_key << "\n\nВведине новый ключ\n>> ";

std::cin.ignore();

getline(std::cin, ECB\_key);

break;

}

case '4': {

std::cout << "Текущий ключ: " << ECB\_key << "\n\n";

system("pause");

break;

}

}

}

}

void CBC\_menu() {

char choice = 0;

while (choice != '7') {

system("cls");

std::cout << "1. Зашифровать\n2. Расшифровать\n3. Изменить ключ\n4. Вывести текущий ключ\n5. Изменить вектор инициализации\n6. Вывести вектор инициализация\n7. Выход\n>> ";

std::cin >> choice;

system("cls");

switch (choice)

{

case '1': {

std::string data = get\_data();

if (data == "")

break;

std::cout << red << "\nИсходный текст:\n" << reset << data;

std::string result = DES::CBC::encode(data, CBC\_key, "ab");

std::cout << green << "\n\nЗашифрованный текст:\n" << reset << result << std::endl << std::endl;

save(result);

break;

}

case '2': {

std::string data = get\_data();

if (data == "")

break;

std::cout << red << "\nИсходный текст:\n" << reset << data;

std::string result = DES::CBC::decode(data, CBC\_key, "ab");

std::cout << green << "\n\nРасшифрованный текст:\n" << reset << result << std::endl << std::endl;

save(result);

break;

}

case '3': {

std::cout << "Текущий ключ: " << CBC\_key << "\n\nВведине новый ключ\n>> ";

std::cin.ignore();

getline(std::cin, CBC\_key);

break;

}

case '4': {

std::cout << "Текущий ключ: " << CBC\_key << "\n\n";

system("pause");

break;

}

case '5': {

std::cout << "Текущий вектор инициализации: " << CBC\_IV << "\n\nВведите новый вектор инициализации\n>> ";

std::cin.ignore();

getline(std::cin, CBC\_IV);

break;

}

case '6': {

std::cout << "Текущий вектор инициализации: " << CBC\_IV << "\n\n";

system("pause");

break;

}

}

}

}

void CFB\_menu() {

char choice = 0;

while (choice != '7') {

system("cls");

std::cout << "1. Зашифровать\n2. Расшифровать\n3. Изменить ключ\n4. Вывести текущий ключ\n5. Изменить вектор инициализации\n6. Вывести вектор инициализация\n7. Выход\n>> ";

std::cin >> choice;

system("cls");

switch (choice)

{

case '1': {

std::string data = get\_data();

if (data == "")

break;

std::cout << red << "\nИсходный текст:\n" << reset << data;

std::string result = DES::CFB::encode(data, CFB\_key, "ab");

std::cout << green << "\n\nЗашифрованный текст:\n" << reset << result << std::endl << std::endl;

save(result);

break;

}

case '2': {

std::string data = get\_data();

if (data == "")

break;

std::cout << red << "\nИсходный текст:\n" << reset << data;

std::string result = DES::CFB::decode(data, CFB\_key, "ab");

std::cout << green << "\n\nРасшифрованный текст:\n" << reset << result << std::endl << std::endl;

save(result);

break;

}

case '3': {

std::cout << "Текущий ключ: " << CFB\_key << "\n\nВведине новый ключ\n>> ";

std::cin.ignore();

getline(std::cin, CFB\_key);

break;

}

case '4': {

std::cout << "Текущий ключ: " << CFB\_key << "\n\n";

system("pause");

break;

}

case '5': {

std::cout << "Текущий вектор инициализации: " << CBC\_IV << "\n\nВведите новый вектор инициализации\n>> ";

std::cin.ignore();

getline(std::cin, CBC\_IV);

break;

}

case '6': {

std::cout << "Текущий вектор инициализации: " << CBC\_IV << "\n\n";

system("pause");

break;

}

}

}

}

void OFB\_menu() {

char choice = 0;

while (choice != '7') {

system("cls");

std::cout << "1. Зашифровать\n2. Расшифровать\n3. Изменить ключ\n4. Вывести текущий ключ\n5. Изменить вектор инициализации\n6. Вывести вектор инициализация\n7. Выход\n>> ";

std::cin >> choice;

system("cls");

switch (choice)

{

case '1': {

std::string data = get\_data();

if (data == "")

break;

std::cout << red << "\nИсходный текст:\n" << reset << data;

std::string result = DES::OFB::encode(data, OFB\_key, "ab");

std::cout << green << "\n\nЗашифрованный текст:\n" << reset << result << std::endl << std::endl;

save(result);

break;

}

case '2': {

std::string data = get\_data();

if (data == "")

break;

std::cout << red << "\nИсходный текст:\n" << reset << data;

std::string result = DES::OFB::decode(data, OFB\_key, "ab");

std::cout << green << "\n\nРасшифрованный текст:\n" << reset << result << std::endl << std::endl;

save(result);

break;

}

case '3': {

std::cout << "Текущий ключ: " << OFB\_key << "\n\nВведине новый ключ\n>> ";

std::cin.ignore();

getline(std::cin, OFB\_key);

break;

}

case '4': {

std::cout << "Текущий ключ: " << OFB\_key << "\n\n";

system("pause");

break;

}

case '5': {

std::cout << "Текущий вектор инициализации: " << CBC\_IV << "\n\nВведите новый вектор инициализации\n>> ";

std::cin.ignore();

getline(std::cin, CBC\_IV);

break;

}

case '6': {

std::cout << "Текущий вектор инициализации: " << CBC\_IV << "\n\n";

system("pause");

break;

}

}

}

}

void test\_Feistel(std::string data, std::string key, float entropy) {

Feistel Feistel;

Feistel.test(data, key, entropy);

system("pause");

}

std::string cipher(char mode, std::string data) {

Feistel Feistel;

std::vector<std::string> result\_v;

if(!mode)

result\_v = Feistel.encode(data, feistel\_key);

else

result\_v = Feistel.decode(data, feistel\_key);

for (int i = 0; i < result\_v.size(); i++) {

std::ofstream out(outFile + "\_" + std::to\_string(i + 1) + "\_round", std::ios::binary);

for (char it : result\_v[i]) {

out.write(&it, sizeof(char));

}

out.close();

}

return result\_v.back();

}

std::string get\_data() {

std::cout << "Введите путь к файлу\n>> ";

std::string filepath;

std::cin >> filepath;

std::string data;

if (filepath == "-m") {

std::cin.ignore();

std::cout << "\nВведите текст вручную\n>> ";

getline(std::cin, data);

return data;

}

std::ifstream in(filepath, std::ios::binary);

if (!in.is\_open()) {

std::cout << red << "\nНе удалось открыть файл (неверный путь)!\n\n" << reset;

system("pause");

return std::string();

}

char temp = 0;

while (in.read(&temp, sizeof(char))) {

data += temp;

}

in.close();

return data;

}

void save(std::string data) {

std::cout << "Сохранить полученный текст в файл " << outFile << "? (Y/N, m для ручного выбора файла)\n>> ";

char ch;

std::cin >> ch;

if (std::toupper(ch) == 'Y') {

std::ofstream out(outFile, std::ios::binary);

for (auto it : data) {

out.write(&it, sizeof(char));

}

out.close();

system("cls");

std::cout << green << "Данные были успешно сохранены в " << outFile << "\n\n" << reset;

system("pause");

}

else if (ch == 'm') {

std::string filepath;

std::cout << "\nВведите путь к файлу для сохранения\n>> ";

std::cin.ignore();

getline(std::cin, filepath);

std::ofstream out(filepath, std::ios::binary);

for (auto it : data) {

out.write(&it, sizeof(char));

}

out.close();

system("cls");

std::cout << green << "Данные были успешно сохранены в " << filepath << "\n\n" << reset;

system("pause");

}

}

static void test\_DES() {

std::string key = ECB\_key, iv = "аб";

std::string temp = DES::ECB::encode("test", key);

temp = DES::ECB::decode(temp, key);

if (temp != "test")

std::cout << "\033[31mECB. Тест провален\033[0m\n";

else

std::cout << "\033[32mECB. Тест пройден\033[0m\n";

temp = DES::CBC::encode("test", key, iv);

temp = DES::CBC::decode(temp, key, iv);

if (temp != "test")

std::cout << "\033[31mCBC. Тест провален\033[0m\n";

else

std::cout << "\033[32mCBC. Тест пройден\033[0m\n";

temp = DES::CFB::encode("test", key, iv);

temp = DES::CFB::decode(temp, key, iv);

if (temp != "test")

std::cout << "\033[31mCFB. Тест провален\033[0m\n";

else

std::cout << "\033[32mCFB. Тест пройден\033[0m\n";

temp = DES::OFB::encode("test", key, iv);

temp = DES::OFB::decode(temp, key, iv);

if (temp != "test")

std::cout << "\033[31mOFB. Тест провален\033[0m\n";

else

std::cout << "\033[32mOFB. Тест пройден\033[0m\n";

std::cout << std::endl;

system("pause");

}

# Листинг файла Feistel.h

# #include <bitset>

# #include <vector>

# #include <string>

# #include <algorithm>

# #include "entropy.h"

# class Feistel {

# public:

# void test(std::string data, std::string key, float target\_entropy) {

# Key Key(key);

# if (data.size() % 2 == 1)

# data += 'x';

# size\_t counter = 0;

# std::cout << "\n\nИсходный текст. Энтропия: " << entropy(data);

# while(entropy(data) < target\_entropy) {

# if (counter % 1000 == 0 && counter != 0) {

# std::cout << "\n\nПродолжить вычисления? (y/n)\n>> ";

# char ch = 0;

# std::cin >> ch;

# if (std::tolower(ch) == 'n')

# break;

# }

# data = F\_round(data, Key.generate());

# std::cout << "\nRound #" << counter + 1 << " Энтропия: " << entropy(data);

# counter++;

# }

# std::cout << "\n";

# }

# std::vector<std::string> encode(std::string data, std::string key) {

# std::vector<uint8\_t> keys;

# Key Key(key);

# for (int i = 0; i < 8; i++) {

# keys.push\_back(Key.generate());

# }

# return cipher(data, keys);

# }

# std::vector<std::string> decode(std::string data, std::string key) {

# std::vector<uint8\_t> keys;

# Key Key(key);

# for (int i = 0; i < 8; i++) {

# keys.push\_back(Key.generate());

# }

# std::reverse(keys.begin(), keys.end());

# return cipher(data, keys);

# }

# private:

# std::vector<std::string> cipher(std::string data, std::vector<unsigned char> keys) {

# std::vector<std::string> result;

# if (data.size() % 2 == 1)

# data += 'x';

# for(int i = 0; i < 7; i++) {

# result.push\_back(F\_round(data, keys[i]));

# data = result.back();

# }

# result.push\_back(last\_F\_round(data, keys[7]));

# data = result.back();

# return result;

# }

# std::string F\_round(std::string data, unsigned char key) {

# for (size\_t i = 0; i < data.size(); i += 2) {

# unsigned char a = (unsigned char)data[i];

# unsigned char b = (unsigned char)data[i + 1];

# unsigned char temp = b;

# b = F(b, key) ^ a;

# a = temp;

# data[i] = (char)a;

# data[i + 1] = (char)b;

# }

# return data;

# }

# std::string last\_F\_round(std::string data, unsigned char key) {

# for (size\_t i = 0; i < data.size(); i += 2) {

# unsigned char a = (unsigned char)data[i];

# unsigned char b = (unsigned char)data[i + 1];

# unsigned char temp = b;

# b = F(b, key) ^ a;

# a = temp;

# data[i] = (char)b;

# data[i + 1] = (char)a;

# }

# return data;

# }

# unsigned char F(int byte, int key) {

# unsigned char res = (byte + key) % 256; // Наложение раундового ключа по модулю алфавита

# res = (res << 1) | (res >> (8 - 1)); // Циклический сдвиг

# return res;

# }

# class Key {

# public:

# Key(std::string str) : last\_generated(0) {

# if (str.size() % 2 != 0)

# str += 'x';

# this->str = str;

# }

# uint8\_t generate() {

# uint8\_t key = 0;

# for (int j = 0; j < str.size(); j += 2) {

# std::swap(str[j], str[j + 1]);

# }

# for (int j = 0; j < str.size(); j += 2) {

# key = (key + (unsigned char)str[j]) % 256;

# }

# str = str.substr(1) + (char)key;

# std::reverse(str.begin(), str.end());

# last\_generated = key;

# return last\_generated;

# }

# private:

# std::string str;

# uint8\_t last\_generated;

# };

# };

DES.h

#pragma once

#include <bitset>

#include <vector>

#include <iostream>

#include <algorithm>

#include <functional>

class DES {

public:

// ECB - Electronic Coding Book

class ECB {

public:

static std::string encode(std::string data, std::string key);

static std::string decode(std::string data, std::string key);

};

// CBC - Cipher Block Chaining

class CBC {

public:

static std::string encode(std::string data, std::string key, std::string IV);

static std::string decode(std::string data, std::string key, std::string IV);

};

// CFB - Cipher Feed Back

class CFB {

public:

static std::string encode(std::string data, std::string key, std::string IV);

static std::string decode(std::string data, std::string key, std::string IV);

};

// OFB - Output Feed Back

class OFB {

public:

static std::string encode(std::string data, std::string key, std::string IV);

static std::string decode(std::string data, std::string key, std::string IV);

};

private:

class Feistel {

public:

static std::string round(std::string data, uint8\_t key);

static std::string last\_round(std::string data, uint8\_t key);

private:

static uint8\_t F(uint8\_t byte, uint8\_t key);

};

class Key {

public:

static std::vector<uint8\_t> generate\_keys(std::string key) {

std::string hash = std::bitset<64>(std::hash<std::string>()(key)).to\_string();

std::vector<uint8\_t> keys;

while (hash.size()) {

keys.push\_back(std::bitset<8>(hash.substr(0, 8)).to\_ulong());

hash.erase(0, 8);

}

return keys;

}

};

};

Листинг файла ECB.cpp

#include "DES.h"

// Encoding by DES ECB mode

std::string DES::ECB::encode(std::string data, std::string key) {

data.size() % 2 == 1 ? data += "x" : data += "";

std::vector<uint8\_t> keys = Key::generate\_keys(key);

std::string result;

for (size\_t i = 0; i < data.size(); i += 2) {

std::string block = data.substr(i, 2);

for (uint8\_t j = 0; j < 7; j++) {

block = Feistel::round(block, keys[j]);

}

block = Feistel::last\_round(block, keys[7]);

result += block;

}

return result;

}

// Decoding by DES ECB mode

std::string DES::ECB::decode(std::string data, std::string key) {

data.size() % 2 == 1 ? data += "x" : data += "";

std::vector<uint8\_t> keys = Key::generate\_keys(key);

std::reverse(keys.begin(), keys.end());

std::string result;

for (size\_t i = 0; i < data.size(); i += 2) {

std::string block = data.substr(i, 2);

for (uint8\_t j = 0; j < 7; j++) {

block = Feistel::round(block, keys[j]);

}

block = Feistel::last\_round(block, keys[7]);

result += block;

}

return result;

}

Листинг файла CBC.cpp

#include "DES.h"

// Encoding by DES CBC mode

std::string DES::CBC::encode(std::string data, std::string key, std::string IV) {

data.size() % 2 == 1 ? data += "x" : data += "";

std::vector<uint8\_t> keys = Key::generate\_keys(key);

std::string result, prev = IV;

for (size\_t i = 0; i < data.size(); i += 2) {

std::string block = data.substr(i, 2);

for (uint8\_t j = 0; j < block.size(); j++) {

block[j] ^= prev[j];

}

for (uint8\_t j = 0; j < 7; j++) {

block = Feistel::round(block, keys[j]);

}

block = Feistel::last\_round(block, keys[7]);

result += block;

prev = block;

}

return result;

}

// Decoding by DES CBC mode

std::string DES::CBC::decode(std::string data, std::string key, std::string IV) {

data.size() % 2 == 1 ? data += "x" : data += "";

std::vector<uint8\_t> keys = Key::generate\_keys(key);

std::reverse(keys.begin(), keys.end());

std::string result, prev = IV;

for (size\_t i = 0; i < data.size(); i += 2) {

std::string block = data.substr(i, 2);

for (uint8\_t j = 0; j < 7; j++) {

block = Feistel::round(block, keys[j]);

}

block = Feistel::last\_round(block, keys[7]);

for (uint8\_t j = 0; j < block.size(); j++) {

block[j] ^= prev[j];

}

result += block;

prev = data.substr(i, 2);

}

return result;

}

Листинг файла CFB.cpp

#include "DES.h"

// Encoding by DES CFC mode

std::string DES::CFB::encode(std::string data, std::string key, std::string IV) {

data.size() % 2 == 1 ? data += "x" : data += "";

std::vector<uint8\_t> keys = Key::generate\_keys(key);

std::string result, feedback = IV;

for (size\_t i = 0; i < data.size(); i += 2) {

for (uint8\_t j = 0; j < 7; j++) {

feedback = Feistel::round(feedback, keys[j]);

}

feedback = Feistel::last\_round(feedback, keys[7]);

std::string block = data.substr(i, 2);

for (size\_t j = 0; j < block.size(); j++) {

block[j] ^= feedback[j];

}

result += block;

}

return result;

}

// Decoding by DES CFC mode

std::string DES::CFB::decode(std::string data, std::string key, std::string IV) {

return encode(data, key, IV);

}

Листинг файла OFB.cpp

#include "DES.h"

// Encoding by DES OFB mode

std::string DES::OFB::encode(std::string data, std::string key, std::string IV) {

data.size() % 2 == 1 ? data += "x" : data += "";

std::vector<uint8\_t> keys = Key::generate\_keys(key);

std::string result, output = IV, prev = IV, ciphered = IV;

for (size\_t i = 0; i < data.size(); i += 2) {

for (uint8\_t j = 0; j < prev.size(); j++) {

ciphered[j] ^= prev[j];

}

for (uint8\_t j = 0; j < 7; j++) {

output = Feistel::round(output, keys[j]);

}

output = Feistel::last\_round(output, keys[7]);

std::string block = data.substr(i, 2);

for (size\_t j = 0; j < block.size(); j++) {

block[j] ^= output[j];

}

result += block;

ciphered = block;

prev = data.substr(i, 2);

}

return result;

}

// Decoding by DES OFB mode

std::string DES::OFB::decode(std::string data, std::string key, std::string IV) {

return encode(data, key, IV);

}

Листинг файла Feistel.cpp

#include "DES.h"

// Round of Feistel net

std::string DES::Feistel::round(std::string data, uint8\_t key) {

uint8\_t a = (uint8\_t)data[0];

uint8\_t b = (uint8\_t)data[1];

uint8\_t temp = b;

b = F(b, key) ^ a;

a = temp;

data[0] = (char)a;

data[1] = (char)b;

return data;

}

// Last round of Feistel net

std::string DES::Feistel::last\_round(std::string data, uint8\_t key) {

uint8\_t a = (uint8\_t)data[0];

uint8\_t b = (uint8\_t)data[1];

uint8\_t temp = b;

b = F(b, key) ^ a;

a = temp;

data[0] = (char)b;

data[1] = (char)a;

return data;

}

// Crypto function of Feistel net

uint8\_t DES::Feistel::F(uint8\_t byte, uint8\_t key) {

uint8\_t res = (byte + key) % 256;

res = (res << 1) | (res >> (8 - 1));

return res;

}