СИМЕТРИЧНА КРИПТОГРАФІЯ
КОМП'ЮТЕРНИЙ ПРАКТИКУМ №2

Криптоаналіз шифру Віженера

Виконали студенти групи ФІ-94
Костюк Кирило і Панасюк Єгор
Варінт-4

Мета роботи

Засвоєння методів частотного криптоаналізу. Здобуття навичок роботи та аналізу потокових шифрів гамування адитивного типу на прикладі шифру Віженера.

Постановка задачі

Потрібно проаналізувати текст, визначити можливу довжину ключа, і спробувати отримати ключ за допомогою двох методів. Показати, який з методів кращий(точніший).

Варіант

фвоьзтыупдыдксыогыъьжкйюыичшчфнъодтмтаангщинпафктмстлзуэешчкффьцтлзуэеш
чоездфкгдурлкъвитюыргъафешрщехоипиармъыьшндзинющбцжктгацдщргтйоыцэкхабх
одйцщцмцмеыоъвюзаъншцокйоспуюафэмоофммъвуряылтымуфлъргжцлзтвмшфнъвгп
юмьшавеибытншръмжъритжярфрръжжгкхйащомэоятчйлхчжъвсфцюахкоездэтуяуьэшч
учйлснлрюбгцоепхьщпиащъэоуддцшэохфуоъчьучтасвввйхюштсуебчоубшъзэщзчтнгиф
ыущгисрхтаэтгаъфимрзййфешююъутчукзкрнвтйрыхябййскххэчцупзмжбюриэыздмар
хдыренртммпырццьоапхялскызцубднсбьггхоубхжоокмшчащякйфпэооэугишсррйомиж
ющъмкхбжпдцоефьщйыцдэмбэялчэьгоьтукйзхнгяюймхдксбчиегжмрйучепьэкеюхигясп
клаъвюхбпйокбпджодсыкыйнювтмущомяьчйыйсупкэомсйчыоьтузьуадаьдачыэоумъкохр
зэкмынннлпюыкщйуатежкхкушръдльнбььцзвсщфетэрфймсмиэыъэшхошэьчифмрюйъф
зтмбшчиыьоафопеебчомыьдыоцднщумсхэйсэхожксдлзгыцбэкаупмбюриыцэзпыбрмних
ушэчццекхмжмняхъынкгкцчбюллтьыаъусефсфвгыцыймуфуыжммхауойроннхооуурхщй
арзчсьлкгщъмэшштшзусррлгыйояъдъеишыбтэсюэздзмсябьюийъкнхоъмохыщцяфвхте
шохлщиешртехжьуъшрмжкяюзжчэьешгьацаткубеуьшгцлещюкжлъвсфклвкрзхспюыияу
южпчузмнмллбэслптпкнзяклпэъекекздзмсятясяхумеоисшсъяцлээроумфдиаффэкннкжкх
рцъьхжпфвзьбснгъьчачнчфмнимсшэзнкнубфьуодаоючщюэидъеиияуьаоснельшиугызлш
ъвэоьыоомхъэкщвцаиъипаоэмхогрййщыпбьэншнпнйиосичошаощбдмгммифщлъвоетда
сяфмеюййбдрйуснррнгнпыккрйсзгьугопумужььнсуьшычудхддрапхчъмьопуждюьфпцэк
шшроскыоьшэмнжатежжятюзупйзаритзябцишмычбъкжбчинюэзнкъкфппюоерамъфьгап
жмргчьгыъьдесйъвфеюмкчбнеиьоамфооыугврцпьщлжолоыатумзмсяяьяяшппкнбэллтъъ
ьгуоукйъуфвюъгькудукядссысдчофурлзтсзыъзщюзйрбюенюцшъмщбртнидопъсийфзцц
жъенрхсичьзцйачорраьаьцлййиипцвцьйцоъпмймгушрмншызтажфмлъьчабшвсмныуфъо

чыыкжуубьезэухжэшкмдэфвгпяизпфжшхоьаршдмзтэхъпкпотшяизкшрчтмъевфьчбчогап
ьорцзцючщьдкпдюеоотъюпрэнокюоуюуябпъктчоыяхмшмеыооужкчьрюрэеъйнеумфпсь
ьегцоенмйстуюыяээрзцмнюомяугыьцпежбьееюецачовртиоофъуънбуфрмюъпуюяюощо
нуцофсняуьмьыбчфдщазжоючъбпнубьетыуюыизкохыэдуршъишъзймйърсурвачаткцпю
мсшхмийакдпдюеураялшчжнузъмгвцсдтсзтйьчожшухюъбгуумсерщфйбупбзмябспурэ
кбуфйзмпсфгоыьцфбпдэтншэъкшщйэмборгчызаыархтйзрсьодекызнхльаешъмъыогуцьт
нлжоуыобоьмюъжиточыэхжшемлцгпфсжпхрсжоъвухълекшпкляймксьйхгмуубрьыозюхь
гьунбсчхтляьнлшяъкнймрццыъмьцжкркщсхаоюгмырфтоьяфрщыаъужртфмдлэьхзшюж
уннммсфуччйоефэмливьшмнюбмскхсхаучйуъпузкьакюръюшцуфтзизстощгйавпыоъни
яьщъыыржязъксуьъиыьгнфьстфпсьылцфбрбщялыцьйцоъпчыырляьрийщвцаймсхаушачо
чцмйюеухяъзъоуцрьюрлтолвкюиежзснрлчщыфьпкыйфквэуроцоаьцкйтсбошйпбжеыйъе
пхяюйощбчтмпоеыцмиьцмздмфахюърймутнцбчьрюяъйлзкрдыщекэоцйцдхзтоосхрюшг
чухзднныиуьэлэшвсмкюоуточгмуттйбътугпфжтхпейосллвошйбупбсбчюдпаьыспросдцо
еаьшывшвуейкхгыомллонкцльекацюахкпушчькргщмгчосхтуиижэььгъббифьниххщлкш
абтчзмсяъоыкпакчскхзиыущгцоннсйфкгшхоцыьмуняящлъьюйъьеуцкчьнюеюймьйтмжч
оавьгооьдрдлюяшяюсмьбстяейлиьущнсоокцйъспгафжынпеокщеъэццвкаткубюаоъоцфп
ыбьмебвсбафеэрэйтммснрнщгйцсуоьлнлппзжшуыкиьченюхмьхбжэдыъвщрбыйъяфзтю
млгтьплпгзцфбнрсымнгйопшцжмжлтьэжпхвохжънвфуоекнйаоьюавъьдахумнпдтикэкры
щьъшхоьембщзутжюдмгъбурхфжкбунпбщнцоюмьшахсянчиортщйэпзймзцоыхщсещочт
чзъокюыъзхцшузммблвучъ
щуткибьнцыьэчсювьодьафицьгхцубззмзгюяюафщзтйзфисэъсяхупттkыуфвцоыъпврйоът
епхмжтпзцумсксбщъэчьсуиьчмрхаъфтчокькцбсыамэвцлцгпыовбоънцуъпдммзлыьцйвш
всмдцуухръзшянеедовиауоеэяфдздоаеибкюшюнуждщувяюъкэяфнпьэкеюхиккщюыяйсс
нющиешъзродбмсэуюзкщрздоъдсаьпйуърорьчоераъмхупытеьадюамавкгклкпормшмэщ
юрьйиюангффьеаюзтгимцтшмтпяфзйьоарбоъмбоучпдоиорнплкпкчйзлшелюлъпъхфтащ
ожшэямььлсйфъкляюсяээткидчавзуьасэвхчащемнаъдружуфкпомэоуыоэркипюангфноок
жуемыофвоьыуябсййлниаоуйботужьюьпгффечурчфчоавюашачнйездынсвцугъьдесйъгс
тхпжйтвсйачерэьщыныхрхыыозэнчзпжрпмгмсхлщэншщгццкчбсьысэыцнещфсвиыкщоь
кудебумсхсъэптрсджмкхющиешсрхйяьадшасжжямхязэялчрфяэнжкджюиешюъэкщвмд
жчиррыфатэрмжкчиорюъзкжмкубьемвцэюкщрфдймлбсцшоиуачфпэццяцэчцьйекьюоит
ьолпъбфтаэчиворачуешрбщяпхфрофьнксыширхъщйньсурычофмцянъофнпкюоерющцу
ркйжльоъхъьыыяхймзмяжыооишупктрърпыущгпоууибьжгэцсйчьзлйжъмтхыэгъьдепкщ
езюяюыьшкхйлсйрюсъжтяфемныоомхъэкыпузкоунаьшишъокхосажррънчомусбвиссьш
ипэвхднюрсъхмятвкхдинапшхмюктрьскшугаюмефаххчльотгяюгвкктйгммивцдчсодхйи
лодгеситндзяндемишъжпевхтасееяцагуцфхдющищртскьшвзтзихгяюъъмцнехбнякэокйб
упбузъхсэуямттщнжьщеххюыахцдехюъееющиикхпеокшавяуэийщажмоулюхжянфцктйк
рнсъюмнчьскфбщофшафрыррцудюарэццймывзтныихрасжпчячткщпуюрсяъьщчтзфдгсй
чйштпужбреуъьрймьнбскъбэхъфмььзццкллсбуначогепжснгфтоаъкастбхксьымнелжеодх
сгььрахкпанжмпнрыщыыьукибхпсишъжжырсчкнщиццюгитвжяйспсторишпэртэсзфяюъ
мэьжепвфвгвкязпыбптйэпиъазгфоучфъчифэооыъуптяищархтсжипрхэкшмсцуюобцфкп
шюансйщаъойулзфлфпуэжтпэзйаьоебуюыфцрффкциргщъмжопкчлмлсхъяиксрртюясхю
чшъмъзццбэвсфьхйьщкьдбюсвауретъкцукэодэьэнйъкпуммкхшесмфьлнцбъривцгаышро
рьпилбцччьтелюфтсщцщнэцшнйеныфвюъыоосчммяехбнципяфесамрхэьхмтахеъдфхгь
ьтаьнзюййсбичымяпфпесбырлыцгикнмеоьлтсуюмитдвшикпеелбйжжврзжчэоншуюлкрэ

шзйъмстяцыхйхздачанюрплэтэзяэьоыхыщйуывфтюсршэьэнжхзспдядушоряэпэфташехр
мегпьгрджмузабфяшрмербщнюхшьонцхрвасйссщякхуптябэтиэйцычонахгмтшыьхэштр
офьудиыущиеусефкхтуючцйуэрйюбшнюеъъмььдйдззсюхяюлкфпнодырлэцбьоцфкпшщ
озаушжизккчлфргпяиикюиежчсаохпмлойвмибэьсбцщцтхжкжьюеомяюэшшвптбюосбьн
ачыркхзфуъхяючбарлмосллфьпамйдерлфрюеььйвцзчджфесбцщцыуткемфсхлуюлпэзюч
хфекрьэгчоонэбцоъхащальрццмъвиагфдпщрзяецоехюэлпткптоърсуьцйпсжкумъгоптзэк
мфмдоьаеыущиеугкфбуклшяъмпымнхьшайхтъюпрврйвфтефьчгчумеолчюыъощыоызхд
ныифэьхктонайнчифььюлпаюцщкчмгъюмъщцвряюфдиэпсжечржтаъклчьэгчрфуфкхпсв
ьчфпэрчйищеишхсжпчвзъбщтухжфлшшрклбчерюуришямхдлъчнръфмкъвйтясвгтшъжж
дззтгреорыщцяэррйефодыоцыяахсдймпсфьхяпжюзузътргмцяпюззаышнгбамэхннсййлу
ъшасжжцсуьзсшяьйэжпыпаннфгррщоюхбгбпбэаоопъдцьикьшрупраийбуошзкрнсыцдчл
йущтмшиуюрмнжърюспктиыуыыьцщкхжяюеющщокшрсчищеихымкъоднцшщассзсдбо
учтоскхюыфьэтюнлнюргцухюъопнъчясолжфнйяошэбщнсъоджйыхрдзячыхпзацбячтяън
ибщзукаютйкнякрасжжырффкруплрмнжьщкфбрнцоэмешяфблймюкэокнеывтмсэвиагом
йшрорбьыжююмвоопусьэемаоукшяфефьхсвтьуяпюиецшэясвьщцяьщкрщюовргрышррр
сбоапяцьцшезтаюъстацфзбцдаоослияоюгтцухдгяохаумютюхгцймеияюфмэаучждсхсвнф
ипяоузпюдсбшъикщютттммсышвьймофбцьцчхюымктювосьцоошесмьрресбщанресмьрр
зтоюанпяшйьюатдазмвйбкькыатдиърьшрчйльфшхбьдлпцщыцьазтзщухгатаъувиммяюхя
нючтщкэйнюьгфуыншрмуцкьыиюрсъчыэкуььоптйнаноачмгрвиаухгмйсхяяфвгоафвшшр
тдомкстощиеудтйгмпрюьтгмжксмюснлъошгъбмнепруьшгцйхщзильщяхлжмдфоонумфк
улрмщгцонтооершшъэыооуйнъкуюрсичьзшыдюеомбэзтюбкцюснлъцъячцойждтершуш
ьъьднскозжынрцкттккхкоарэйсуэапнъбщезюъзнеъкптзчежръфипяхргощиъхсършюрэйяъ
яфъоуыхэрйчимтярыснъопцпдьоераивкшннсьщивцоччбармдяъдюяюъэптзтсчсвавцбмьу
пещозтгъцщачохьмзмсяфбшыькщчтврупрмэншмсуикэирючщыцьтюмючодшюъзкжмечв
счхюаъбэуфмйзснгпячщчоууыдюсвшхгаъыэъчяълнюрныъгвчмнюышубнлкхжкпбщнюе
шъжждвсмприовучащофнфкоахмхщыбцфцтгщаххщиишгзрмоисвьйэжйъкцобшнлсбрчо
юхимиреоояикшянкийфмлчщсмкабтйоырсещмяшчуниыеавэооаьшзнжкчгггтфэхупттлъ
кгзцоыпачзтнюоптьюэпаперкфупбаоачыэкукюужшщфьшхвтгофесъоуыхьубрйснияьуол
оормэъоырюмупыпайнюргццплкыкыялъпгфочгчоокхоссурфсичйзстбхмсйыъдобоуниъи
фвоьычцбшжбщчгыцэчэяскщкшмлэбютпюзмхкьоншхмшььдхйиляцбэсжкэзхйаямгвкнй
хкъкыбшзгяюсяетхюмбоофхъщыодвчазстгтьюгъуйшпшюахрсмкштзохоооершъгхцртум
ъхсфцчзтърцппгамэьлэщутзябявлфучымоофммвсчъъбчьзцпднысэръвчцмнлйфохъбры
лйнжпбреръоцмцутчадщезтбэзеянксйъснрщфжшштужьолиэызасбгагэзежюцэкэсврдник
гъяъыадксйъитыощгъьдепьшолчымитндуезмсхшрмзщцтужбрершннысцтужьчифмымт
пщрзйуссншгчанупйдсфпухтмитнчуцзфчгтжфрынткижъсхэйшкпяунрдумсьшюйцбикж
нсгуррклешвхцдщыжюпсжпыэтднцаомымърцдуудэьлчьнлкфвгцюммтшюъэтямрвуфти
ыкщйчьщбвдотиьыънпштапшлзцсйцйнакзхбйтсъотаьябчджфмфвюмучйонтяьопэйшнш
щюптлътсьбгншаррокшнлзупйчунблыьакущляпаюйсбонсщяоаьювмжбллсауъжэфвцдю
ыушшэыыяэтнхкчнизыъзырзчйймфсэунаыште
нйфхтющсдтрэцтжфхзхюсэжудздиыиуяьцысонцгищееюхшоьцфкпшщопхщцгццгклкни
зэйшкъодйдысщиэуэкпжкрдниатацджшяюссбпаорыюишэткизъжлыцьофбдухлълячьоы
ькудокюоуючокъьщкрыщеушяциэщвздиыиугщзьчбнцглъкгчоояхцптябцлюъцчщльлшпч
ннцыяльебднибокгънэкшщйрднжешрщмкшштшцкшууюъмуфцпурпнонгэслпшктчюыое
ютиъбуткляълстбчйвожнгюзжлфокфпбучдюфлгбкщымоофммсхаотюъопнъчъкгчочмй

рээйиснвэоыйхсрртюзшлаьцэщщзьдсфвибкшычужшфбщсссьхяхцптябюепэйсэшщрцяк
нргьщлжтбйптбуажююсжшуннъкэлсцущиеуйехлфнсщшьцхкбффдраерщфэкъснфпщен
юаълшууътаэтеюяшйъьзсибшорюъыйыщвтсдцопбъсльцжкхыюигажфичобццьюувъьюй
ъьеучжьаырщмыфарзяеуаотйэупчъптззчиаызащюртлдюеомызаббфбфьэксбйсюхойежь
шплаофвэекрмиъюпрщъкъцдрйжмтиыкщоирпкьйъсхмыънкшктйнямиыцыыссвйдоичхю
хмипчууомзтс

Хід роботи(тупо код з cpp)

main.cpp

```cpp
#include <iostream>
#include "VijinerEncoder.h"
#include "VijinerDecoder.h"



int main()
{
   setlocale(LC_ALL, "");
   #ifdef _WIN32
   system("chcp 1251"); // настраиваем кодировку консоли
   #endif

   std::wstring_convert<std::codecvt_utf8<wchar_t>> converter;
   std::wstring input;
   std::string line;
   std::ifstream fin;
   fin.open( "small copy.txt", std::ifstream::in );
   while (std::getline(fin, line, '\n') && !line.empty())
   {
       input += converter.from_bytes(line);
   }
   fin.close( );

   input = lower_case( input );

   std::wcout << input << std::endl;

   std::wcout << L"I" << 0 << " = " <<
VijinerDecoder::compliance_index( input ) << std::endl;

   for (size_t r = 2; r < 6; r++)
   {
       std::wcout << L"I" << r << " = " <<
VijinerDecoder::compliance_index( input, r ) << std::endl;
```

```
    }

    std::vector<size_t> r = VijinerDecoder::get_period_v( input );

    std::wcout << L"Method 1" << std::endl;
    for ( const auto& el : r)
    {
        std::wcout << el << std::endl;
        auto maybe_key = VijinerDecoder::get_key_variant1( input, L'o',
el );
        std::wcout << maybe_key << std::endl;
    }

    std::wcout << L"Method 2" << std::endl;
    for ( const auto& el : r)
    {
        std::wcout << el << std::endl;
        auto maybe_key = VijinerDecoder::get_key_variant2( input, el );
        std::wcout << maybe_key << std::endl;
    }


    return 0;
}
```

VijinerEncoder.h

```
#include "algorithmics.h"
class VijinerEncoder
{
public:
    static void set_key( std::wstring&& );
    static std::wstring encode( const std::wstring& );
private:
    static std::wstring key;
};

std::wstring VijinerEncoder::key = L"";

void
VijinerEncoder::set_key( std::wstring&& key_in )
{
    key = key_in;
}
```

```
std::wstring
VijinerEncoder::encode( const std::wstring& original_text )
{
   std::wstring encoded_text;
   std::wstring clear_original_text = lower_case( original_text );

   encoded_text.reserve( clear_original_text.size() );

   const static auto alphabet_size = alphabet.size( );

   for (size_t i = 0; i < clear_original_text.size(); i++)
   {
       if ( clear_original_text[i] == L' ' )
       {
           continue;
       }

       auto letter_numeric = numeric_value_of_letter(
clear_original_text[i] );
       auto key_letter_numeric = numeric_value_of_letter(key[i %
key.size()] );
       auto numeric_new_letter = ( letter_numeric + key_letter_numeric
) % alphabet_size;
       wchar_t new_letter = alphabet[ numeric_new_letter ];
       encoded_text.push_back( new_letter );
   }

   return encoded_text;

}
```

VijinerDecoder.h

```
#include "algorithmics.h"
#include <fstream>
class VijinerDecoder
{
public:
   static std::wstring get_key_variant2( const std::wstring& Y, size_t
r );
   static std::wstring decode( const std::wstring& Y, const
std::wstring& key );
```

```
    //I(Y)
    static double compliance_index( const std::wstring& );
    //I_r(Y_r)
    static double compliance_index( const std::wstring& Y, size_t r );
    //D_r
    static double same_pares_counter( const std::wstring& Y, size_t r );
    static size_t get_period( const std::wstring& );
    static std::vector<size_t> get_period_v( const std::wstring& );
    static std::wstring get_key_variant1( const std::wstring& Y, wchar_t
x, size_t r );
    static std::vector<std::wstring> get_Y_blocks( const std::wstring&
Y, size_t r );
};

std::vector<std::wstring>
VijinerDecoder::get_Y_blocks( const std::wstring& Y, size_t r )
{
    std::vector<std::wstring> blocks;
    blocks.reserve( r );
    std::wstring block_of_r;
    block_of_r.reserve( ( Y.size( ) / r )  + 1 );

    size_t letter_for_read_index = 0;
    for (size_t i = 0; i < r; i++)
    {
        letter_for_read_index = i;
        while ( letter_for_read_index <= Y.size() )
        {
            block_of_r.push_back( Y[letter_for_read_index] );
            letter_for_read_index += r;
        }
        blocks.push_back( block_of_r );
        block_of_r.clear();
    }
    return blocks;
}

std::wstring
VijinerDecoder::get_key_variant1( const std::wstring& Y, wchar_t x,
size_t r )
{
    const auto blocks = get_Y_blocks( Y, r );
    std::wstring key;
```

```
    key.reserve( r );
    const auto numeric_x = numeric_value_of_letter( x );

    for ( const auto& Y_i : blocks )
    {
        size_t max_counted_letter = 0;
        wchar_t most_common_letter = ' ';

        for ( const auto& letter : alphabet )
        {
            size_t letter_count = std::count( std::begin( Y_i ),
                                  std::end( Y_i ), letter );
            if ( letter_count > max_counted_letter )
            {
                max_counted_letter = letter_count;
                most_common_letter = letter;
            }
        }

        auto numeric_most_common_letter = numeric_value_of_letter(
most_common_letter );
        auto numeric_key_letter = numeric_difference_between_letters(
numeric_most_common_letter, numeric_x );
        wchar_t key_letter = alphabet[ numeric_key_letter ];
        key.push_back( key_letter );
    }

    return key;
}

double
M_i( const std::wstring& Y_i, const std::map<wchar_t, size_t>&
letters_freqs,
     wchar_t maybe_key_letter, size_t text_size )
{
    auto numeric_maybe_key_letter = numeric_value_of_letter(
maybe_key_letter );
    double m_i = 0;
    for ( const auto& l : alphabet )
    {
        auto letter_prob = letter_probability( text_size,
letters_freqs.at( l ) );
        auto numeric_l = numeric_value_of_letter( l );
```

```
        auto x = alphabet[ ( numeric_maybe_key_letter + numeric_1 ) %
alphabet.size() ];
        m_i += letter_prob * std::count( std::begin( Y_i ), std::end(
Y_i ), x );
    }


    return m_i;
}

std::wstring
VijinerDecoder::get_key_variant2( const std::wstring& Y, size_t r )
{
    const auto blocks = get_Y_blocks( Y, r );
    std::wstring key;
    key.reserve( r );

    std::wstring_convert<std::codecvt_utf8<wchar_t>> converter;
    std::wstring input;
    std::string line;
    std::ifstream fin;
    fin.open( "dead_souls.txt", std::ifstream::in );
    while (std::getline(fin, line, '\n') && !line.empty())
    {
        input += converter.from_bytes(line);
    }
    fin.close( );

    input = lower_case( input );

    auto letters_freq = letter_frequency( input );

    for ( const auto& block : blocks )
    {
        std::vector<double> m_i;
        m_i.reserve( alphabet.size( ) );
        for ( const auto& g : alphabet )
        {
            auto m_i_for_g = M_i( block, letters_freq, g, input.size( )
);
            m_i.push_back( m_i_for_g );
        }
```

```
        auto max_m_i_g = std::max_element( std::begin( m_i ), std::end(
m_i ) );
        auto numeric_g_value = std::distance( std::begin( m_i ),
max_m_i_g );
        auto g_letter = alphabet[ numeric_g_value ];
        key.push_back( g_letter );
    }

    return key;
}

std::vector<size_t>
VijinerDecoder::get_period_v( const std::wstring&  Y )
{
    size_t r = 2;
    std::vector<double> D_r;
    D_r.reserve( 28 );
    for (; r < 30; r++)
    {
        D_r.push_back( VijinerDecoder::same_pares_counter( Y, r ) );
    }

    auto max_el = std::max_element( std::begin( D_r ), std::end( D_r )
);
    double max = *max_el;

    r = std::distance( std::begin( D_r ), max_el ) + 2;

    return dividers( r );
}

size_t
VijinerDecoder::get_period( const std::wstring&  Y )
{
    size_t r = 2;
    std::vector<double> D_r;
    D_r.reserve( 28 );
    for (; r < 30; r++)
    {
        D_r.push_back( VijinerDecoder::same_pares_counter( Y, r ) );
    }

    // for ( const auto& el : D_r )
```

```
    // {
    //     std::wcout << el << std::endl;
    // }

    auto max_el = std::max_element( std::begin( D_r ), std::end( D_r )
);
    double max = *max_el;

    r = std::distance( std::begin( D_r ), max_el ) + 2;

    return r;
}

double
VijinerDecoder::compliance_index( const std::wstring& Y )
{
    double comp_index = 0;
    for ( const auto& c : alphabet )
    {
        auto letter_count_in_text = std::count( std::begin( Y ),
std::end( Y ), c );
        comp_index += letter_count_in_text * ( letter_count_in_text - 1
);
    }
    double n = Y.size( );
    n -= std::count( std::begin( Y ), std::end( Y ), L' ' );
    comp_index /= n * ( n - 1 );

    return comp_index;
}

double
VijinerDecoder::compliance_index( const std::wstring& Y, size_t r )
{
    double comp_index = 0;

    const auto blocks = VijinerDecoder::get_Y_blocks( Y, r );
    for ( const auto& Y_i : blocks )
    {
        comp_index += VijinerDecoder::compliance_index( Y_i );
    }
```

```
    return comp_index;
}

double
VijinerDecoder::same_pares_counter( const std::wstring& Y, size_t r )
{
    double d = 0;
    for (size_t i = 0; i < Y.size( ) - r; i++)
    {
        if ( Y[i] == Y[i + r] )
        {
            ++d;
        }

    }
    return d;
}

std::wstring
VijinerDecoder::decode( const std::wstring& Y, const std::wstring& key
)
{
    std::wstring decoded_text;
    decoded_text.reserve( Y.size() );

    for (size_t i = 0; i < Y.size(); i++)
    {
        auto numeric_decoded_letter =
numeric_difference_between_letters(

numeric_value_of_letter( Y[i] ),

numeric_value_of_letter( key[ i % key.size() ] )
                                        );
        decoded_text.push_back( alphabet[ numeric_decoded_letter ] );
    }

    return decoded_text;
}
```

algorithmics.h

```
#pragma once
#include <string>
```

```
#include <vector>
#include <cctype>
#include <algorithm>
#include <cwctype>
#include <map>
#include <cmath>
#include <sstream>
#include <locale>
#include <codecvt>

const std::vector<wchar_t> alphabet =
{

L'а',L'б',L'в',L'г',L'д',L'е',L'ж',L'з',L'и',L'й',L'к',L'л',L'м',L'н',L
'о',L'п',L'р',L'с',L'т',L'у',L'ф',L'х',L'ц',L'ч',L'ш',L'щ',L'ъ',L'ы',L'
ь',L'э',L'ю',L'я'
};

const std::map<wchar_t, wchar_t> lowercase =
{
   {L'А',L'а'},
   {L'Б',L'б'},
   {L'В',L'в'},
   {L'Г',L'г'},
   {L'Д',L'д'},
   {L'Е',L'е'},
   {L'Ё',L'е'},
   {L'Ж',L'ж'},
   {L'З',L'з'},
   {L'И',L'и'},
   {L'Й',L'й'},
   {L'К',L'к'},
   {L'Л',L'л'},
   {L'М',L'м'},
   {L'Н',L'н'},
   {L'О',L'о'},
   {L'П',L'п'},
   {L'Р',L'р'},
   {L'С',L'с'},
   {L'Т',L'т'},
   {L'У',L'у'},
   {L'Ф',L'ф'},
   {L'Х',L'х'},
```

```
  {L'Ц',L'ц'},
  {L'Ч',L'ч'},
  {L'Ш',L'ш'},
  {L'Щ',L'щ'},
  {L'Ъ',L'Ъ'},
  {L'Ы',L'ы'},
  {L'Ь',L'ь'},
  {L'Э',L'э'},
  {L'Ю',L'ю'},
  {L'Я',L'я'},
  {L'а',L'а'},
  {L'б',L'б'},
  {L'в',L'в'},
  {L'г',L'г'},
  {L'д',L'д'},
  {L'е',L'е'},
  {L'ё',L'е'},
  {L'ж',L'ж'},
  {L'з',L'з'},
  {L'и',L'и'},
  {L'й',L'й'},
  {L'к',L'к'},
  {L'л',L'л'},
  {L'м',L'м'},
  {L'н',L'н'},
  {L'о',L'о'},
  {L'п',L'п'},
  {L'р',L'р'},
  {L'с',L'с'},
  {L'т',L'т'},
  {L'у',L'у'},
  {L'ф',L'ф'},
  {L'х',L'х'},
  {L'ц',L'ц'},
  {L'ч',L'ч'},
  {L'ш',L'ш'},
  {L'щ',L'щ'},
  {L'ъ',L'ъ'},
  {L'ы',L'ы'},
  {L'ь',L'ь'},
  {L'э',L'э'},
  {L'ю',L'ю'},
  {L'я',L'я'},
```

```
};


std::wstring
delete_rubish(const std::wstring& input) {
   std::wstring result;
   for (const auto& c : input) {
       if (lowercase.find(c) != lowercase.end()) {
           result.push_back(c);
       }
   }
   return result;
}

std::wstring
lower_case(std::wstring input) {
   input = delete_rubish(input);
   for_each(input.begin(), input.end(), [&](wchar_t& c) {c =
lowercase.at(c); });
   return input;
}


size_t
numeric_difference_between_letters( size_t y, size_t x )
{
   if ( y >= x )
   {
       return y - x;
   }
   else
   {
       auto buf = x - y;
       return alphabet.size( ) - buf;
   }
}

size_t
modulo_substitute( size_t y, size_t x, size_t mod )
{
   if ( y >= x )
   {
       return ( y - x ) % mod;
```

```
    }
    else
    {
        auto buf = ( x - y ) % mod;
        return mod - buf;
    }
}


auto
bigram_count(const std::wstring& input) {
    std::map <std::wstring, size_t> bigram_counter;

    for (int i = 0; i < input.length() - 1; i += 2) {

        ++bigram_counter[input.substr(i, 2)];
    }

    return bigram_counter;
}

auto
top_bigrams( const std::wstring& input )
{
    const auto bigram_counter = bigram_count( input );
    std::multimap< size_t, std::wstring > top_of_bigrams;
    for ( const auto& p : bigram_counter )
    {
        top_of_bigrams.insert( { p.second, p.first } );
    }

    return top_of_bigrams;
}

std::vector<size_t>
dividers( size_t r )
{
    std::vector<size_t> divs;
    divs.reserve( std::sqrt( r ) );
    for (size_t i = 2; i <= r; i++)
    {
        if ( r % i == 0 )
        {
```

```
            divs.push_back( i );
        }

    }

    return divs;
}

size_t
numeric_value_of_letter( wchar_t l )
{
    auto alphabet_position =  std::find( std::begin( alphabet ),
std::end( alphabet ), l );
    auto val = std::distance(  std::begin( alphabet ), alphabet_position
);
    return val;
}

size_t
numeric_value_of_letter( wchar_t l, const std::vector<wchar_t>&
custom_alphabet )
{
    auto alphabet_position =  std::find( std::begin( custom_alphabet ),
std::end( custom_alphabet ), l );
    auto val = std::distance(  std::begin( custom_alphabet ),
alphabet_position );
    return val;
}

auto letter_frequency(std::wstring input) {
    std::map <wchar_t, size_t> let_counter;
    for (const auto& c : input) {
        ++let_counter[c];
    }

    return let_counter;
}

double letter_probability( double text_size, double letter_count )
{
    return letter_count / text_size;
}
```

```
int gcd(long long a, long long b) {//Алгоритм Стейна
   int d = 1;
   while ((a % 2 == 0) && (b % 2 == 0)) {
       b /= 2;
       a /= 2;
       d *= 2;
   }
   while (a % 2 == 0) {
       a /= 2;
   }
   while (b != 0) {
       while (b % 2 == 0) {
           b /= 2;
       }
       if (a <= b) {
           a = a;
           b = b - a;
       }
       else {
           int b1 = b;
           b = a - b;
           a = b1;
       }
   }
   d *= a;
   return d;
}

//Написанно по псевдокоду из Википедии
long long reverse(long long a, long long n) {
   if (a >= n)a %= n;
   long long t = 0, r = n, newt = 1, newr = a;
   while (newr != 0) {
       long long quotient = r / newr;
       long long newt1 = t;
       t = newt;
       newt = newt1 - quotient * newt;
       long long newr1 = r;
       r = newr;
       newr = newr1 - quotient * newr;
   }

   if( t < 0 )
```

```
    {
        t += n;
    }

    return t;
}

size_t
get_bigram_number( const std::wstring& bigram, std::vector<wchar_t>
custom_alphabet )
{
    if ( bigram.size( ) != 2)
    {
        throw( "error value" );
    }

    auto x_i = numeric_value_of_letter( bigram[0], custom_alphabet );
         x_i *= custom_alphabet.size( );
         x_i += numeric_value_of_letter( bigram[1], custom_alphabet );

    return x_i;
}
```

Опис труднощів: літаючі жілізякі над головою, розв’язком стало ППО. Також неможливість підібрати ключ першим методом, тож прийшлося використати другий.

Ir (r = 0-5)

I0 = 0.0326304
I2 = 0.0651931
I3 = 0.0977066
I4 = 0.130571
I5 = 0.162637

Dr(r=2-30)
D2 = 261
D3 = 241
D4 = 260
D5 = 249
D6 = 240
D7 = 230
D8 = 256
D9 = 253

D10 = 248
D11 = 263
D12 = 264
D13 = 456
D14 = 249
D15 = 267
D16 = 256
D17 = 270
D18 = 236
D19 = 252
D20 = 238
D21 = 278
D22 = 242
D23 = 235
D24 = 271
D25 = 238
D26 = 416
D27 = 235
D28 = 278
D29 = 254
D30 = 237

Перший метод отримання ключа(прирівнювати найчастіші літери у блоці до найчастішої літери у мові)

Результат: громыкавьдума

Другий метод отримання ключа(значення ключа, одержане із використанням функціїM(g)i)

Результат: громыковедьма

Розшифрування тексту варіанту(шифротекст є вище)

старминскаяшколачародеевпифийитравницфакультеттеоретическойипрактическоймаги икафедрамаговпрактиковчастьперваясоциальныйукладбытинравывампирьейобщиныви качтовычтотоимеетепротиввампировраспринкорпорациямифкурсоваяработаадепткивос ьмогокурсавольхиреднойнаучныйруководительмагистрпервойстепениархимагксанперл овдевятьсотдевяностодевятыйгодпобелорскомулетосчислениюгородстарминведениехо рошийсегоднявыдалсяденектеплыйбезветренныйвтораядекадасеноставамесяцанеспешн осочилаьсквозьклепсидрусолнечноголетаиголосазябликовдоносившиесяизпридорожн ыхкустовзвенеливушахяехаласквозьихгнездовыеугодьякаквдольпограничнойполосыпо лосойбыладорогазаброшенныйпроклевывающийсяпыльнойтравойкривойбольшакзябли кипопеременновозмущалисьвторжениемчеловеканабелойлошадивихчастныевладенияза лихватскиетрелисменялисьхриплымчириканьемптахисуетливоперепархивалиповеточка мтревожалиствуразноцветнаякаймавокругчерныхподсыхающихлужвзрываласьсотнями

истомленныхжароймотыльковраскручиваласьввысьвихремтрепещущихкрыльевповодья
завернутыепетлейсвисалипереднейлукияпокачиваласьвседлекакмешокскрупойпридер
живаялевойрукойлежавшеенаколеняхписьмоипытаясьразобратьпрыгающиепередглазам
ируныромашкапользоваласьмоимрасслабленнымсостояниемвсезамедляяизамедляяшагн
адеясьчтояувлеченнаячтениемнезамечуеековарногоманевраидамейостановитьсяиспоко
йнопощипатьтравкутычегоэтоголубушкаанушевеликопытамиплутоватаякобылкаразоча
рованновсхрапнуладавайдавайхалтурщицаяустроиласьпоудобнейесливообщеможноуст
роитьсяпоудобнейнатомпыточномпредметекоимявлялосьдляменяжесткоеказенноеседло
натретийденьпутиромашкинагриватоненькимиколечкамиспускалаьдопереднейлукизаб
иваясьмеждустраницамипухлогописьмакотороеядолжнабылавручитьповелителюдогевы
икотороеужеминутпятькаксамовольновскрылаприпомощимагиинетронувувесистойпеча
тинаверевочкенааломвоскеотчетливопроступалоттискперстнятринадцатьруниперепле та
ющийсяздракономединорогвцентретутмоизанятиялитературойдипломатиейигенеалоги
ейгрубопрервалиоченьгрубояедвауспелаподхватитьлисткипоползшиевразныестороныр
омашканеисправимаясаботажницазадумчивожевалауздубряцаяжелезомвтовремякакнез
накомыйивесьмаподозрительныйтипобросшейнаружностидемонстративнопотрясалпер
едлошадиноймордойсамодельнымарбалетомсгрязнойстрелоймногоразовогоиспользова
ниятакчтонепонятнобылокогоонсобираетсяграбитьменяилиромашкуяприподняласьнаст
ременахсинтересомрассматриваязаржавленныйнаконечникянедумаючтоэтосамоеудачно
еместодляторговлиантиквариатомдоверительносообщилаянезнакомцувотвстарминеувас
быегосрукамиоторваливернееотрубилизнаетелитамоченьнелюбятразбойниковромашка
обнюхалаарбалетпрезрительнофыркнулаинапрочьигнорируяграбителяпотянуласькаппе
титнойзеленималинникаизвысокойгущикотороготолькочтовозниклоэточудовлаптяхпрес
тупныйэлементзаметносмутилсянаконечникзатрепеталкакщенячийхвостикувыдораская
нияипокаяниябылоещедалекозаблудшаяовцаупорствовалавогрехесеребролюбияануткаж
ивослезайсконядевкаязыкатаякошелекилижизньдапошустрейслышишьяизобразилаусил
еннуюработумыслиладноубедилкошелекпахнулоозономлицограбителяпередернулосьзр
ачкирасширилисьглазаостекленелиионмедленноопустиварбалетотвязалибеспрекословн
оподалмнетощиймешокболтавшийсяупоясаотмешкаразилокошкамиикуревомослабивве
ревкустягивавшуюгорловинуяпропустиласквозьпальцынесколькомелкихмонетмаловато
дорогоймоймаловатосленцойработаешьбезогонькавпрочемтакужибытьвозьмувкачестве
авансаосчастливилаяграбителяшвыряяемуподногипустоймешокипредупредилаячерезпа
руднейэтойжедорогойназадпоедутакужбудьдобрпостарайсяменянеразочароватьмужикн
еотрываяотменязагипнотизированноговзглядамедленнонагнулсяподнялмешокизастылс
толбстолбомневсилахшевельнутьсябезмоеговедомакактолькогореграбительскрылсяизв
идуядеактивировалазаклинаниеипозволиларомашкеперейтисгалопаналюбимуюеютрус
цуписьмозажатоевовремяподсчетаденегуменямеждуколеняминемногопомялосьиутрати
лотоварныйвидвпрочемрассудилаяглавноенеоформлениеасодержаниеоноежекомпенсир
овалонедостаткирепейноголистаиспользованноговукромномместеагавотнаконециобомн
епарастрокзадифирамбамизагадочномуарракктурупропустишьинезаметишьзавремяобу
чениявысшейшколечародеевпифийитравницадептквольхапроявиласебязнаюоченьпло
хонеусидчиванетерпеливасвоевольназнакомаяпеснялюбитзлыешуткиинеоднократнопер
еноситихсвоспитанниковнавоспитателейэтоонпроведрочтолидабылоодноведеркодоволь
ноообъемистоестоялосебенабалкенаддверьюмоейкомнатыэдакийсамодельныйкапканнас

оседейпошкольномуобщежитиюдабынеповаднобылобезспросуодалживатьуменяконспе
ктыикастрюлиснавареннымнанеделюборщомможетучительтакбынеразозлилсяеслибыве
дровсетакиопрокинулосьанеупалоемунаголовустоймявместесводойотличаетсяредкимис
пособностямикпрактическойитеоретическоймагиисильноразвитойинтуициейбыстроада
птируетсякнестандартнойситуациихаможетяещенебезнадежнанеприличнаякакаятогран
ицаудогевыуэльфоввысокиетравыугномовскалыувадлаковгрудывыброшеннойнаповерх
ностьземлиудриаддубыподметающиеоблакаудруидовкаменныекругиулюдейоблупленн
ыестеныканалысзатхлойводойразделенныепаройтройкойподъемныхмостовдалысыестр
ажникипринихбдительнодремлющиеупираясьнаржавыеалебардыаздесьосиныиздевател
ьствокакоетоособенноеслиучестьчтожителидогевывампирыхорошиетакиеосинысеребр
истыетрепещущиезаосинамищекочетнебоостроверхийеловыйковерсредикоторогокоегд
епроглядываютзатравленныеберезкиисосенкисамажедогевалежитвдолинекакплюшкана
днерасписнойпиалыеслисмотретьсхолмакраяпиалывиденбелыйободокизосинвторойпот
олщепотемнееизелейавцентрешироКоезеленоедноскрапочкамисамадогевавкольцевозде
ланныхполейиоблакахтуманаподойдешьвплотнуюкдеревьямнаставлялменяучительипо
шлешьмысленныйсигналвглубьлесалюбойможешьдуматьочемугоднолишьбысформиро
ватьмощнуютелепатическуюволнуакомумнееенаправитьнаобщейчастотектонибудьизст
ражейграницыуслышитясмущеннокашлянулалучшебыемуэтогонеслышатьнеобязательн
опродумыватьочереднуюпакостьзнаюзнаютынанихсверхвсякоймерыгоразданонасейраз
постарайсявоздержатьсяотоныхочемэтояхдаоволневампирыоченьвосприимчивыктеле
патииисразуотреагируютнаееприсутствиехотяинесмогутдоскональнорасшифроватьтакч
тонапирайнаколичествоаненакачествоттакясмотрюнадымящуюбанюнаморщивлоботу
сердияинамоюволнутутжереагируютпятьилишестьадептовкоторыеовеянныепаромвыбе
гаютиздверейивыпрыгиваютизоконатакованныевнезапноожившимивеникамирукибуду
щихколлегзанятышайкамиприкрывающимиотвениковсамоесокровенноеучительусмиря
етвеникиоднимдвижениембровиновзгляды

Висновок

Засвоїли методи частотного криптоаналізу. Здобули навички роботи та аналізу потокових шифрів гамування адитивного типу на прикладі шифру Віженера.
Лаба цікава, як і остання, але більш легка.