РЕФЕРАТ

Пояснительная записка к работе содержит 56 страниц, 17 источников, 3 рисунка, 2 приложения, 11 формул.

Главная цель работы состоит в определении криптостойкости систем, основанных на групповой математике, хаотических преобразованиях и ДНК.

Провести анализ криптосистем, построенных на разных базах, определить недостатки и преимущества каждой.

В работе рассмотрены алгоритмы построения криптосистемы на хаосе и ДНК, также проанализированы базовые методы защиты информации, которые используют групповую математику. Реализован алгоритм RSA, и дополнения, которые позволяют применять его для ДНК-криптографии, с использованием нуклеотидов.

НЕАБЕЛЕВЫ ГРУППЫ, ДНК, ХРАНЕНИЕ ДАННЫХ В ДНК, ДНК-КРИПТОГРАФИЯ, КРИПТОГРАФИЧЕСКИЕ СИСТЕМЫ, ХАОТИЧЕСКИЕ КРИПТОСИСТЕМЫ, ВНЕДРЕНИЕ ХАОТИЧЕСКИХ КРИПТОСИСТЕМ, РЕКОМЕНДАЦИИ ПО РАЗРОБОТКЕ ХАОТИЧЕСКИХ КРИПТОСИСТЕМ.

РЕФЕРАТ

Пояснювальна записка до роботи містить 56 сторінок, 17 посилань, 3 рисунка, 2 додатки, 11 формул.

Головна мета роботи полягає у визначенні криптостійкості систем, заснованих на груповій математиці, хаотичних перетвореннях і ДНК. Провести аналіз криптосистем, побудованих на різних базах, визначити недоліки та переваги кожної.

В роботі розглянуті алгоритми побудови криптосистеми на хаосі і ДНК, також проаналізовані базові методи захисту інформації, які використовують групову математику. Реалізовано алгоритм RSA, та доповнення, які дозволяють застосовувати його для ДНК-криптографії, з використанням нуклеотидів.

НЕАБЕЛЕВІ ГРУППЫ, ДНК, ЗБЕРІГАННЯ ДАННИХ У ДНК, ДНК-КРИПТОГРАФІЯ, КРИПТОГРАФІЧНІ СИСТЕМИ, ХАОТИЧНІ КРИПТОСИСТЕМИ, ВПРОВАДЖЕННЯ ХАОТИЧНИХ КРИПТОСИСТЕМ, РЕКОМЕНДАЦІЇЇ ДО РОЗРОБКИ ХАОТИЧНИХ КРИПТОСИСТЕМ.

ABSTRACT

Explanatory note for the work contains 56 pages, 17 sources, 3 figures, 2 applications, 11 formulas.

The main goal of the paper is to determine the cryptographic strength of systems based on group mathematics, chaotic transformations and DNA. Conduct an analysis of cryptosystems built on different bases, determine the disadvantages and advantages of each.

In work were considered algorithms of construction of a cryptosystem on chaos and DNA, basic methods of information protection that use group mathematics were analyzed. Implemented the RSA algorithm, and additions that allow it to be used for DNA cryptography, using nucleotides.

NON-ABELIANS GROUPS, DNA, STORAGE OF DATA IN DNA, DNA CRYPTOGRAPHY, CRYPTOGRAPHIC SYSTEMS, CHAOTIC CRYPTOSYSTEMS, INTRODUCTION OF CHAOTIC CRYPTOSYSTEMS, RECOMMENDATIONS FOR DEVELOPMENT OF CHAOTIC CRYPTOSYSTEMS.

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

RSA – аббревиатура от фамилий Rivest, Shamir и Adleman

CSP – задач поиска сопряжений (Conjugacy Search Problem)

ДНК – дезоксирибонуклеи́новая кислота́

DH – Diffie-Hellman

A – аденин

T – тимин

C – цитозин

G – гуанин

Base4 – четверичная система счисления

Base3 – троичная система счисления

ПЦР – полимеразная цепная реакция

PCR – polymerase Chain Reaction

XOR – операция, исключающая “ИЛИ”

DES – data encryption standard

IDEA – international Data Encryption Algorithm

RC5 – Rivest's Cipher 5

RC4 – Rivest's Cipher 4

SEAL – software-optimized Encryption Algorithm

ЦП – центральный процессор

ОС – операционная система

ПБайт – петабайт

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 9](#_Toc484633688)

[1 КРИПТОГРАФИЯ В НЕАБЕЛЕВЫХ ГРУППАХ 11](#_Toc484633689)

[1.1 Основы криптографии с открытым ключом 12](#_Toc484633690)

[1.2 Основы криптографии на свободных группах 18](#_Toc484633691)

[1.3 Обмен публичными ключами с помощью неабелевых групп 25](#_Toc484633692)

[1.4 Трехэтапный протокол Шамира и протоколы обмена ключами 29](#_Toc484633693)

[1.5 Криптография в полициклических группах 31](#_Toc484633694)

[2 ИСПОЛЬЗОВАНИЕ ДНК В КРИПТОГРАФИИ 34](#_Toc484633695)

[2.1 Хранение данных с использование ДНК 34](#_Toc484633696)

[2.2 Процесс кодирования информации 36](#_Toc484633697)

[2.3 ДНК-криптография 38](#_Toc484633698)

[2.3.1 Криптосистема ДНК с использованием замены 39](#_Toc484633699)

[2.4 Преимущество и недостатки в использовании ДНК 41](#_Toc484633700)

[3 КРИПТОГРАФИЯ ОСНОВАННАЯ НА ХАОСЕ 42](#_Toc484633701)

[3.1 Внедрение хаотических систем 42](#_Toc484633702)

[3.2 Внедрение криптосистем 43](#_Toc484633703)

[3.3 Анализ безопасности 45](#_Toc484633704)

[3.4 Базовые правила формирования устойчивой криптосистемы 47](#_Toc484633705)

[ВЫВОДЫ 48](#_Toc484633706)

[ПЕРЕЧЕНЬ ССЫЛОК 49](#_Toc484633707)

[ПРИЛОЖЕНИЕ А ВХОДНЫЕ И ПОЛУЧЕННЫЕ ДАННЫЕ 51](#_Toc484633708)

[ПРИЛОЖЕНИЕ Б ИСХОДНЫЕ КОДЫ ПРОГРАММЫ 52](#_Toc484633709)

ВВЕДЕНИЕ

Благодаря междисциплинарному развитию науки об информации, физической науки и биологической науки в области криптографии появилось много новых технологий. Новые области криптографии в основном состоят из квантовой криптографии, хаотической криптографии, ДНК-криптографии и т. д. Безопасность квантовой криптографии основана на принципе неопределенности Гейзенберга. Квантовая криптография - единственная, которая может реализовать безоговорочную безопасность в настоящее время. Мэттьюс впервые применил теорию хаоса в криптографии и предложил хаотическую схему шифрования потока, основанную на пересмотренной логистической карте. С тех пор хаотическая криптография привлекла широкое внимание, большинство исследований в области хаотической криптографии сосредоточено на криптографии с секретным ключом. Криптография ДНК, которая использует вычисления ДНК, является новой ветвью криптографии в последние годы. Используя высокую плотность хранения и высокий параллелизм молекул ДНК, такая криптография может реализовать шифрование, аутентификацию, подпись и т. д.

Между тем, криптографы с нетерпением ждут применения новых трудноразрешимых математических задач в классической криптографии. В настоящее время большинство публичных криптографических примитивов основаны на высокой стойкости некоторых математических задач в очень больших конечных абелевых группах. Выдающиеся трудные задачи состоят из проблемы факторизации больших целых чисел, задачи дискретного логарифмирования над конечным полем или эллиптической кривой и т. д. Однако благодаря квантовым алгоритмам для факторизации целых чисел и решению задачи дискретного логарифмирования большинство известных криптосистем с открытым ключом будут небезопасными, когда квантовые компьютеры станут практичными. Таким образом, это неизбежная работа по разработке эффективных криптографических схем, которые могут противостоять квантовым атакам. Собственно, с 1980-х годов несколько экспертов пытались разработать новые схемы криптографии, основанные на сложных проблемах теории групп. В 1985 году Вагнер и Мадьярик предложили подход к разработке криптосистем с открытым ключом на основе групп и полугрупп с неразрешимой проблемой слов. В 1986 году, Магливерас предложил симметричную криптосистему, основанную на специальном типе факторизации бесконечных групп, названных логарифмическими сигнатурами для конечных групп перестановок. В 2000 году Ko разработал теорию криптографии на основе кос, основанную на жесткости задачи поиска сопряжения (CSP) в группах кос. В 2004 году Эйк и Кахробаи предложили новую криптосистему на основе полициклических групп. В 2005 году Шпильрайн и Ушаков предположили, что группа Томпсона может быть хорошей платформой для построения криптосистем с открытым ключом. Между тем, активная ветвь некоммутативной криптографии, основанная на жесткости проблемы групповой факторизации, добилась больших успехов в течение последних двух десятилетий.

# КРИПТОГРАФИЯ В НЕАБЕЛЕВЫХ ГРУППАХ

Традиционно, криптография является наукой и искусством разработки и реализации секретных алгоритмов шифрования или криптосистем. Криптоанализ – это наука о разрушении криптосистем, а криптология – все поле криптографии плюс криптоанализ. В современной литературе, криптография используется параллельно с криптологией. В настоящее время растет потребность в безопасных криптосистемах благодаря использованию интернет-магазинов, электронных финансовых переводов и так далее.

Большинство используемых в настоящее время криптосистем с открытым ключом и протоколов обмена публичными ключами, таких как алгоритм RSA, алгоритм и Диффи-Хеллмана, основаны на теории чисел и, следовательно, теоретически зависят от структуры абелевых групп. Хотя успешных атак на стандартные протоколы не было, возникает ощущение, что сила вычислительной техники сделала эти методы менее безопасными. В результате этого был проведен ряд исследований для разработки и анализа новых криптосистем и протоколов обмена ключами на основе некоммутативных криптографических платформ. Эта ряд исследований получил широкое название некоммутативной алгебраической криптографии[1].

До сих пор основными источниками некоммутативных криптографических платформ были неабелевы группы. В криптосистемах, основанных на этих объектах, алгебраические свойства платформ широко используются как в разработке криптосистем, так и в криптоанализе. В частности, сложность, в смысле сложности, некоторых алгоритмических задач в конечно-заданных группах, таких как проблема поиска сопряжения, сыграла решающую роль в шифровании и расшифровке

Основными источниками неабелевых групп являются комбинаторная теория групп и теория линейных групп. Криптография группы кос, где шифрование выполняется в классических группах кос, является одним из ярких примеров[2]. Однонаправленные функции в системах групп кос основаны на сложности решения задач теории групп, таких как сопряженность.

Хотя криптография группы кос имела первоначальный впечатляющий успех, были выявлены различные потенциальные атаки. Боровик, Мясников, Шпильрайн и другие исследовали статистические аспекты этих атак и идентифицировали, так называемые черные дыры, в группах платформ, вне которых присутствуют криптографические проблемы. Баумслаг Файн и Ксу предложили потенциальные криптосистемы, используя комбинацию теории комбинаторных групп и линейных групп, и дали общую схему для этих типов криптосистем.

Исследование и криптоанализ потенциальных групп-платформ оказали сильное положительное влияние как на теорию групп, так и на теорию сложности.

* 1. Основы криптографии с открытым ключом

В этом разделе описывается стандартная терминология, используемая в криптографии, а затем будет рассмотрено два наиболее распространенных алгоритма передачи открытых ключей, протокол Diffie-Hellman и протокол RSA.

Как правило, как текстовое сообщение (не зашифрованое сообщение), так и сообщение зашифрованного текста (зашифрованое сообщение) записываются в некоторый алфавит N-букв, который обычно одинаковый и для открытого текста, и для зашифрованого. Тогда метод шифрования или алгоритм шифрования является преобразованием N-букв. Наиболее распространенным способом выполнения этого преобразования является рассмотрение N букв как N целых чисел по модулю N, а затем выполнение теоретико-числовой функции на них. Поэтому большинство алгоритмов кодирования используют модульную арифметику, следовательно, криптография тесно связана с теорией чисел.

Современная криптография обычно разделяется на криптографию классическую (симметричную) и криптографию с открытым ключом (не симметричную). В первом алгоритмы шифрования и дешифрования предположительно известны только отправителю и получателю, обычно называемому Бобом и Алисой. В последнем случае метод шифрования является общедоступным, но только приемная сторона знает, как дешифровать.

Процесс превращения текстового сообщения в шифротекст называется шифрованием, тогда как обратный процесс называется расшифровкой. Алгоритмы шифрования разделяют сообщение открытого текста и зашифрованного текста на единицы сообщений. Это одиночные буквы или пары букв или более общие k-векторы букв. Преобразования выполняются на этих блоках сообщений, и алгоритм шифрования представляет собой отображение из набора блоков сообщения открытого текста в набор блоков сообщения зашифрованного текста. Вводя это в математическую формулировку, ми получим = {набор всех блоков сообщения открытого текста} в качестве открытого текста и = {набор всех блоков сообщения зашифрованного текста} в качестве шифротекста.

Алгоритм шифрования является тогда применением обратимой функции вида *.* Функция является алгоритмом шифрования. Обратная функция является расшифровкой или функцией расшифрования. Тройка , состоящая из набора блоков текстовых сообщений, набора блоков зашифрованного текста и функции шифрования, называется криптосистемой.

Взлом зашифрованного текста называется криптоанализом. Попытка взлома этого текста называется атакой. Большая часть криптоанализа начинается со статистического и частотного анализа используемого языка открытого текста. Криптоанализ зависит также от знания алгоритма формирования зашифрованного текста, то есть типа используемой криптосистемы.

Большинство классических криптосистем являются криптосистемами, полученными теоретически. При применении криптосистемы к *N*-буквенному алфавиту мы рассматриваем буквы как целые числа по модулю *N*. Алгоритмы шифрования затем применяют теоретико-числовые функции и используют модульную арифметику для этих целых чисел.

Обычно мы используем не одну букву за раз, а последовательность из *k* букв. Тогда таких букв являются единицей сообщения. Алгоритм шифрования является функцией (1.1)

(1.1)

В настоящее время существует много случаев, когда защищенная информация должна передаваться по открытым коммуникационным линиям. К ним относятся, например, банковские и финансовые операции, покупка предметов через кредитные карты через Интернет и тому подобное. Это привело к развитию криптографии с открытым ключом. В классической криптографии только отправитель и получатель знают методы зашифрования и расшифрования. Кроме того, особенностью таких криптосистем является то, что, если известен способ шифрования, может быть выполнено дешифрирование.

Основная идея криптосистемы с открытым ключом – наличие односторонней функции. Это функция, которую легко реализовать, но ее очень сложно инвертировать. Следовательно, шифрование сообщения становится простым, а расшифровка очень сложной.

Стандартная модель криптосистемы с открытым ключом заключается в следующем. Алиса хочет отправить сообщение Бобу. Алгоритм шифрования для Алисы является общедоступной информацией, также как и для Боба. С другой стороны, алгоритмы дешифрования и являются секретными и известны только Алисе и Бобу соответственно. Пусть *P* – сообщение, которое Алиса хочет отправить Бобу. Она посылает. Для декодирования Боб применяет первый , который только он знает. Это дает ему ( = . Затем он использует , который общедоступен, и применяет это ( для получения сообщения.

Алиса отправляет , а не только для проверки подлинности, Боб убеждается в том, сообщение действительно пришло от Алисы. Предположим что – код подтверждения Алисы; подпись, номер социального страхования и т.д. Если Боб получает , он может быть отправлен любым лицом, поскольку является общедоступным. С другой стороны, поскольку только Алиса предположительно знает, получает имеющее смысл сообщение , что позволяет убедиться, что это от Алисы.

Получение, имеющей смысл, односторонней функции может быть трудной задачей. Наиболее широко используемые (в настоящее время) системы открытых ключей основаны на различии в обращении некоторых теоретико-числовых функций. Первый настоящий протокол с открытым ключом был разработан в 1976 году Диффи и Хеллманом, используя проблему нахождения дискретного логарифма.

В модульной арифметике легко возвести элемент до степени, но трудно определить, степень какого элемента это. В частности, если *G* – конечная группа, такая как циклическая мультипликативная группа , где – простое число, и для некоторого , то дискретный логарифм является любым целым *t* при условии *h =* . Грубая форма системы открытых ключей Diffe-Helman заключается в следующем.

Боб и Алиса будут использовать классическую криптосистему, основанную на ключе который лежит в приделах, где – простое число. Это ключ , который Алиса должна отправить Бобу. Пусть – мультипликативная образующая . Алиса выбирает *∈* , которое находится в пределах . Она публикует . Боб выбирает  *∈* и делает общедоступным Секретный ключ – . И Боб и Алиса, могут знать этот ключ но, никто другой, не может открыть его. Алиса знает свою секретную экспоненту, значение является общедоступным от Боба. Следовательно, она может вычислить ключ = (.Аналогичная ситуация имеет место и для Боба. Однако злоумышленник знает только, , и . Если злоумышленник не может решить проблему с дискретным логарифмом, то есть найти или , обмен ключами безопасен. Обратите внимание, что это зависит от (= (.

В 1997 году стало известно, что идеи криптографии с открытым ключом были разработаны британскими разведывательными службами до Диффи и Хеллмана.

В 1977 году Ривест, Адельман и Шамир разработали Алгоритм RSA, который в настоящее время является наиболее широко используемым криптосистемами с открытым ключом. Он основан на различии факторинга больших целых чисел и, в частности, на том, что легче проверять на простоту, чем на фактор. В базовых схемах на простейшем уровне это работает следующим образом.

Алиса выбирает два больших числа , и целое , взаимно простые с = () (). Предполагается, что эти целые числа выбираются случайным образом, чтобы минимизировать атаки. Числа, которые она выбирает, должны быть достаточно большими. Первоначально RSA использовало простые числа, состоящие приблизительно из 100 десятичных цифр, но, поскольку, вычисления и атаки стали более сложными, пришлось использовать большие числа. Как только Алиса получила , , , она формирует = и вычисляет , мультипликативно обратный элемент для по модулю . То есть удовлетворяет . Она обнародует ключ шифрования = (, ) и известный всем алгоритм шифрования где ∈ – единица сообщения.

Можно показать, что если ( и   
, то . Поэтому алгоритм дешифрования представлен формулой (1.2)

= (1.2)

Заметим, что (, поэтому он является обратимым.

Теперь Боб делает те же действия, чтобы получить , , . Он представляет = и обнародует свой ключ = ( ).

Если Алиса хочет отправить сообщение Бобу, которое может аутентифицировать ее, она посылает (()). Атака требует факторизацииили , задача эта гораздо более сложная, чем получение простых чисел , , .

Было много расширений и улучшений основных протоколов открытых ключей. Криптография на эллиптической кривой использует задачу дискретного логарифма в группе эллиптической кривой. Эта группа является конечной абелевой группой и имеет определенные преимущества над циклическими группами, используемыми в стандартном протоколе Диффи-Хеллмана.

Криптосистема Эль-Гамаля – это использования метода обмена ключами Диффи-Хеллмана для шифрования. Метод работает следующим образом. Предположим, что Боб и Элиса хотят открыто общаться. Они обменялись секретным ключом *k*, который, по общему мнению, они знают. Пусть – функция шифрования или алгоритм шифрования, основанный на ключе *.* Алиса хочет отправить сообщение Бобу, а – а двоичная битовая строка: где – битовая строка для ключа , и – сложение по модулю 2.

Боб знает ключ и, следовательно, может вычислить его как бинарную строку. Теперь он вычисляет . Поскольку сложение по модулю 2 имеет порядок 2, имеем алгоритм декодирования (1.3)

(1.3)

Теперь Боб применяет алгоритм дешифрования для декодирования сообщения.

* 1. Основы криптографии на свободных группах

Расширение всех этих идей до некоммутативных платформ является предметом некоммутативной алгебраической криптографии. Это предполагает следующие идеи:

* общие алгебраические приемы построения криптосистем;
* потенциальные алгебраические платформы (конкретные группы, кольца и т.д.) Для реализации методов;
* криптоанализ и анализ безопасности результирующих систем.

Основным источником некоммутативных платформ являются неабелевы группы, и основным методом обработки неабелевых групп в криптографии является комбинаторная теория групп. Это относится к ветви теории групп, которая изучает группы с помощью групповых представлений, то есть множеств генераторов и отношений между ними. Основная идея использования комбинаторной теории групп для криптографии состоит в том, что элементы групп могут быть выражены как слова в некотором алфавите. Если есть простой способ переписать элементы группы в пределах этих слов и далее техника, используемая в этом процессе перезаписи, может быть предоставлена секретным ключом, тогда может быть создана криптосистема. Простейшим примером является, возможно, сводная криптосистема. Её можно описать следующим образом.

Рассмотрим свободную группу на свободных образующих . Тогда каждый элемент из имеет единственное выражение в виде слова . Пусть при условии, что = – множество слов в генераторах свободной группы . На самом базовом уровне для построения криптосистемы предположим, что у нас есть алфавит с открытым текстом. Например, предположим, что являются символами, необходимыми для построения осмысленных сообщений на английском языке. Для шифрования используем подстановочный шифр (1.4).

(1.4)

То есть символы алфавита переходят в значения нашей группы (1.4)

Тогда заданному слову в алфавитном формате открытого текста будет присвоено свободное групповое слово . Это представляет элемент в . Тогда может быть отправлено в качестве секретного сообщения.

Для реализации этой схемы нам нужно конкретное представление , а затем для расшифровки – способ переписать обратно в терминах . Это конкретное представление является идеей, лежащей в основе гомоморфных криптосистем.

Алгоритм дешифрования в криптосистеме свободной группы зависит от процесса перезаписи Рейдемайстера-Шрейера. Это метод перезаписи элементов подгруппы свободной группы в пределах генераторов этой подгруппы. Примерно это работает следующим образом. Предположим, что являются свободными образующими для некоторой подгруппы свободной группы *F* над . Каждое является сокращенным словом в генераторах . Трансверсаль Шрейера есть множество . (левых) смежных представителей для в *F* специального вида[3]. Любая подгруппа свободной группы имеет трансверсаль Шрейера. Процесс Райдемайстера-Шрайера позволяет построить набор образующих для с использованием трансверсали Шрейера. Далее, с учетом трансверсальности Шрейера, из которой был построен набор генераторов для , процесс переписывания Райдемайстера-Шрайера позволяет нам алгоритмически переписать элемент из . Для такого элемента, выраженного словом . В генераторах *F* этот алгоритм переписывает как слово в генераторах .

Знание трансверсальности Шрейера и использование перезаписи Рейдемайстера-Шрейера облегчает процесс декодирования в случае свободной группы, но не является необходимым. Учитывая известный набор генераторов для подгруппы, метод сгибания Столлингса для разработки графа подгруппы также может быть использован для переписывания в условиях заданных генераторов.

Чистые криптосистемы свободной группы подвержены различным атакам и могут быть легко взломаны. Наиболее успешные атаки на криптосистемы свободной группы называются атаками на основе длины.

Баумслег, Файн и Ксу описали следующую общую схему шифрования с использованием криптографии свободных групп[4].

Начнем с конечной определенной группы где и точное представление где может быть любым из нескольких различных типов объектов: линейной группой, группой перестановок, кольцом степенных рядов и т.д.

Предположим, что существует алгоритм повторной экспрессии элемента в в пределах генераторов. То есть, если = . ∈ , где *W* – это слово в этих генераторах и мы получаем ∈ , алгоритмически можем найти и его выражение как слово .

Как только мы получим , предположим, что у нас есть две свободные подгруппы , учитывая, что

Предположим, что мы установили трансверсали Шрейера для в и для

в , которые держаться в секрете Бобом и Алисой. Теперь, основываясь на фиксированных трансверсалях Шрейера, мы имеем множества генераторов Шрейера, построенных по процессу Рейдемейстера-Шрейера для (1.5) и для (1.6)

для (1.5)

а также

для (1.6)

Заметим, что генераторы для будут заданы как слова в , образующие , также как и генераторы для будут заданы как слова в генераторах для . Заметим, что и могут совпадать и что и в общем случае не должны быть свободными, а должны только иметь уникальный набор нормальных форм, так что бы представление элементов в пределах данных генераторов Шрейера являлись уникальными.

Мы будем шифровать внутри , точнее, внутри . Предположим, что число генераторов для больше, чем множество символов в нашем алфавите с открытым текстом. Пусть – наш алфавит с открытым текстом. На низшем уровне мы выбираем начальную точку в генераторах и начинаем шифровать согласно формуле (1.7)

(1.7)

Предположим, что Боб хочет передать сообщение Алисе, где – слово в алфавите с открытым текстом. Напомним, что как Боб, так и Алиса знают различные поперечные пересечения Шрейера, которые держатся в секрете между ними. Затем Боб кодирует ( и вычисляет в элемент который он посылает Алисе. Сообщение отправляется как матрица, если – линейная группа или как перестановка, если – группа подстановок и т. д.

Алиса использует алгоритм для относительно , чтобы переписать как слово в генераторах . Затем она использует трансверсаль Шрейера для в что бы перезаписать, используя процесс Райдемайстера-Шрейера, как слово в образующих . Поскольку свободен или имеет единственную нормальную образующую этого выражения для уникального элемента из . Как только у нее есть слово, написанное на генераторах , она использует трансверсаль для в , чтобы переписать снова, используя процесс Рейдемайстера-Шрайера, в пределах генераторов для . Тогда она имеет слово и используя расшифровывает сообщение.

В реальной реализации добавляется дополнительный случайный коэффициент шума.

Также была представлена реализация этого процесса, который использовал для базовой группы G классическую модулярную группу. Кроме того, это был надежный полиалфавитный шифр.

Система в модульной группе была представлена следующим образом. Список конечно порожденных свободных подгрупп группы является открытым и представлен как система генераторов (представленных в виде матриц). При полной практической реализации предполагается, что велико. Для каждого имеем трансверсаль Шрейера. И соответствующий упорядоченный набор образующих построенные из трансверсали Шрейера процессом Райдемайстера-Шрейера.

Считается, что каждый , где – размер алфавита открытого текста, то есть каждая подгруппа имеет гораздо больше генераторов, чем размер алфавита открытого текста.

Подгруппы и их соответствующие трансверсали Шрейера могут быть выбраны множеством способов. Например, подгруппа комутаторов модулярной группы свободна от ранга 2, и некоторые из подгрупп могут быть определены из гомоморфизмов этой подгруппы на множество конечных групп.

Предположим, что Боб хочет отправить сообщение Алисе. Сначала Боб выбирает три целых числа, где = выбор подгруппы , = начальная точка среди генераторов для замены алфавита с открытым текстом, = размер блока сообщений.

Проясним значения и . Как только Боб выбирает m, чтобы уточнить значение , он делает замену .

Опять же предполагается, что , чтобы такая подстановка могла начаться почти в любой точке последовательности генераторов . Размер единицы сообщения – это количество закодированных букв, которые Боб поместит в каждую зашифрованую интегральную матрицу.

Как только Боб сделал выбор он берет свое текстовое сообщение и группирует блоки из букв. Затем он делает указанную замену выше для формирования соответствующих матриц в модульной группе: (1.8)

(1.8)

Введем случайный коэффициент шума. После формирования (1.8) Боб умножает справа каждый элемент на случайную матрицу , и получает (разный для каждого). Единственное ограничение на эту матрицу состоит в том, что при формировании произведения не должно равняться нулю. Это может быть легко проверено и гарантирует, что свободно приведенная форма для является просто конкатенацией выражений для . Затем он посылает Алисе интегральный ключ с помощью некоторого алгоритма обмена ключами (RSA, DH). Затем он отправляет сообщение в виде случайных матриц

Поэтому то, что на самом деле посылается, не является элементами выбранной подгруппы , а элементами случайных правых смежных классов в . Цель отправки смежных элементов двоякая. Первая цель – помешать любой геометрической атаке, маскируя подгруппу. Во-вторых, это делает слова в генераторах модульной группы более длинными, что эффективно препятствует атаке грубой силы.

Чтобы декодировать сообщение, Алиса сначала использует дешифрование с помощью открытого ключа, чтобы получить целые ключи . Так как она знает подгруппу , подстановку зашифрованного текста из генераторов и сколько букв кодирует каждая матрица. Она использует описанные выше алгоритмы для выражения каждого в пределах генераторов . Она знает границу трансверсали Шрейера, которая тайно хранится Бобом и Алисой, поэтому теперь использует процесс переписывания Рейдемайстера-Шрейера, чтобы начать извлекать это слово в пределах генераторов . Переписывание Рейдемайстера-Шрейера выполняется буквами слева направо. Следовательно, когда она достигает свободных генераторов, она останавливается. Стоит обратить внимание, что строка, которую она переписывает, длиннее, чем она должна быть, это сделано, чтобы декодировать случайную матрицу . Это связано с тем, что она фактически переписывает не элемент подгруппы, а элемент из правого смежного класса. Это вносит дополнительные сложности атакующему. Так как они являются случайными правыми смежными классами, это заставляет подобрать статистические шаблоны в генераторах, даже если перехвачено более одного сообщения. На практике, подгруппы должны меняться после каждого сообщения.

Первоначальный ключ часто изменяется. Следовательно, как упомянуто выше, этот метод становится типом полиалфавитного шифра. Исторически сложилось, что полиалфавитные шифры очень трудно декодировать.

* 1. Обмен публичными ключами с помощью неабелевых групп

Среди первых попыток использования неабелевых групп в криптографии были схемы Аншеля-Аншела-Голдфелда и Ко-Ли. Оба набора авторов примерно в то же время предложили использовать неабелевы группы и комбинаторную теорию групп для обмена открытыми ключами. Безопасность этих систем зависела от различий в решении некоторых «жестких» теоретико-групповых задач.

Протокол Аншеля-Аншела-Голдфелда и протокол Ко-Ли начинаются с группы платформ , заданной групповым представлением. Главным предположением в обоих протоколах является то, что элементы имеют удовлетворяющие уникальные формы, которые легко вычислить для заданных элементов группы. Однако далее предполагается, что при нормальных формах для нормальная форма для произведения не раскрывая или .

Сначала опишем протокол обмена открытыми ключами Аншеля-Аншеля-Гольдфелда. Пусть – группа платформы, заданная конечным представлением и с нормальными формами, как описано выше.

Алиса и Боб хотят передать общий секрет. Во-первых, Алиса и Боб выбирают случайные конечно порожденные подгруппы в (1.9), задавая для каждой из них множество образующих.

(1.9)

И делают их общедоступными. Подгруппа (1.9) является подгруппой Алисы, а подгруппа (1.9) – подгруппой Боба.

Алиса выбирает в своей подгруппе слово секретной группы , а Боб выбирает в своей подгруппе секретное групповое слово . Для элемента ∈ обозначим через нормальную форму для. Алиса знает свое секретное слово и знает генераторы подгруппы Боба. Она публикует нормальные формы сопряжений

Боб знает свое секретное слово и генераторы подгруппы Алисы и публикует нормальные формы сопряжений . Общий   
секрет – это коммутатор вида (1.10)

(1.10)

Заметьте, что Алиса знает , так как она знает в пределах генераторов ее подгруппы, и она знает сопряжение с b, так как Боб сделал сопряженные с генераторами c общедоступным. Поскольку Алиса знает , она знает .

Аналогичным образом Боб знает . Злоумышленник должен знать соответствующий конъюгатор, то есть элемент, который сопрягает каждый из генераторов. Для элементов в группе , где известно, что = , задача поиска сопряжения заключается в определении сопряжения . Известно, что эта проблема неразрешима, в общем, то есть в группах, где сопряжение не может быть определено алгоритмически. С другой стороны, существуют группы, в которых задача поиска сопряжений разрешима, но «сложна», т. е. Сложность решения задачи поиска сопряжения трудна. Такие группы становятся идеальными платформами для протокола Аншеля-Аншеля-Голдфелда.

Таким образом, безопасность в этой системе связана с проблемой поиска сопряжнеия. Аншель, Аншель, Голдфелд предложили группы кос в качестве потенциальных платформ и использовали, с 12 и более генераторами в подгруппах. Их предложение и предположения Ко и Ли привели к развитию криптографии в группах кос. Были различные атаки на такие группы. Однако некоторые из них были основаны на изменения параметров. В общем идеи остаются в силе, несмотря на атаки.

Ко и Ли разработали аналогичную систему, которая является прямым переводом протокола Диффи-Хеллмана на неабелеву групповую теоретическую установку. Его безопасность основана на различии проблемы сопряженности. Мы снова предполагаем, что группа платформы имеет удовлетворяющие уникальные формы, которые легко вычислить для данного элемента группы, но трудно восстановить элемент группы. Напомним еще раз, что означает сопряжение через [5].

В протоколе Ко-Ли, Алиса и Боб выбирают коммутирующие подгруппы и группы платформ . Группа является подгруппой Алиса, а подгруппа Боб – , и они являются секретными. Теперь они полностью имитируют классическую технику Дифини-Хеллмана. Существует открытый элемент   
, Алиса выбирает случайный секретный элемент и делает открытым , Боб выбирает случайный секретный элемент и делает открытым . Секретный общий ключ – . Заметим, что , так как подгруппы коммутируют. Отсюда следует, что =. Следовательно, и Боб и Алиса могут определить общий секрет. Сложность заключается в решении проблемы сопряженности.

Задача сопряженности для группы , точнее для группового представления для , для заданных , является алгоритмически определять, сопряжены ли эти элементы или нет. Как и в случае с задачей поиска сопряжения, известно, что сопряженность, вообще говоря, неразрешима, но есть группы, в которых она очень трудна. Эти группы затем становятся основными группами платформы для протокола Ко и Ли. Как и в случае с протоколом Аншеля-Аншеля-Голдфелда, Ко и Ли предлагают использовать группы кос.

Проблема сопряженности и проблема поиска сопряжения являются лишь двумя из групповых теоретических проблем поиска и решения, которые были использованы для построения односторонних функций в криптографической установке. Мы вспоминаем несколько других важных таких проблем, а затем повторно используем их для шифрования и обмена общедоступными ключами.

Первое определение (проблема с текстом). Дана, конечно, определенная группа , существует ли алгоритм для определения того, является ли слово в генераторах тривиальным словом?

Второе определение (Проблема сопряженности решений). Существует ли в группе с конечным представлением какой-либо алгоритм для определения того, сопряжена ли произвольная пара слов и в порождающих ? То есть, существует ли такой, что ?

Третье определение (Проблема разложения). Пусть – конечно определенная группа с подгруппами. Для двух элементов и из существует алгоритм для нахождения двух элементов и таких, что ?

Четвертое определение (Проблема одновременного поиска). Пусть – конечно определенная группа. Для учитывая, что для каждого существует ли алгоритм для нахождения, для каждого ?

Закрывая этот раздел, можно описать некоммутативный аналог системы обмена открытыми ключами Эль-Гамаля на основе проблемы сопряженности поиска. Это было предложено Кахробаем и Ханом [6]. Как и в протоколах Ко-Ли и Аншеля-Аншеля-Голдфелда, мы начинаем с конечно определенной группы платформ , заданной групповым представлением. Как и ранее, основные предположения состоят в том, что элементы имеют, удовлетворяющие нас, уникальные формы, которые легко вычислить для данных элементов группы. Однако далее предполагается, что при нормальных формах для нормальная форма для произведения не раскрывая или .

Далее содержит две коммутирующие конечно порожденные собственные подгруппы и . Криптографическая цель заключается в том, чтобы Алиса и Боб установили ключ сеанса по незащищенной сети.

Боб выбирает секретный элемент и произвольный элемент . Боб раскрывает и = . Предположим, Алиса хочет отправить в качестве сеансового ключа Бобу. Тогда,

1. Алиса выбирает случайное и посылает Бобу вместе с .

2. Затем Боб вычисляет.

3. Теперь Боб может вычислить , что позволит ему расшифровать ключ сеанса, поскольку

Возможность реализации этой схемы основана на предположении, что порождающие и обратные элементы в могут быть вычислены эффективно. Определение закрытых ключей Боба влечет за собой решение проблемы сопряженности поиска для Это определено элементами и = , так как они определяют . Следовательно, безопасность этой схемы основана на предположении, что нет практического алгоритма для решения проблемы поиска сопряженности для .

* 1. Трехэтапный протокол Шамира и протоколы обмена ключами

Протокол обмена ключами – это алгоритм, который позволяет отправить ключ (например, информацию о том, какую систему шифрования использовать) от одного пользователя к другому по не защищенной среде. Групповой протокол обмена ключами, основанный на схеме Диффии-Хеллмана, может быть реализован следующим образом. Предположим, что мы имеем, конечно, определенную группу с теми же условиями, что и в протоколах Аншеля-Ансель-Гольдфелда и Ко-Ли. То есть задается , а элементы имеют удовлетворяющие на формы. Далее предполагается, что имеет две большие подгруппы , , которые коммутируют поэлементно. В качестве альтернативы можно использовать одну большую абелеву подгруппу из .

Теперь предположим, что Боб хочет общаться с Алисой через незащищенный канал. Секретный ключ дает понять какую кодирующую систему использовать, кодируется внутри конечно-порожденной группы G со свойствами, приведенными выше. Две подгруппы , , которые коммутируют поэлементно, хранятся в секрете Бобом и Алисой. является подгруппой для Боба и – подгруппой для Алисы. Боб хочет послать ключ , Алисе. Он выбирает два случайных элемента , и посылает Алисе сообщение (в зашифрованном виде) . Теперь Алиса выбирает два случайных элемента , и отправляет обратно Бобу. Так как коммутирует элементно с мы имеем .

Кроме того, поскольку Боб знает свои выбранные элементы и , он может умножать их на обратные элементы, чтобы получить , который затем он отправляет Алисе. Так как Алиса знает ее выбранные элементы и, она может умножать их на обратные значения, чтобы получить ключ . Предполагается, что для каждого сообщения Боб и Алиса будут выбирать разные пары случайных элементов из или . Этот метод известен как трехступенчатый алгоритм Шамир обмена ключами, который был введен Шамиром для общих алгебраических объектов.

В настоящей схеме секретный ключ полностью определяется Бобом, который затем передает его Алисе. Затем схема попадает в класс ключевых транспортных протоколов, а не в протоколы обмена общедоступными ключами. В большинстве случаев протоколы передачи ключей разрабатываются исходя из предположения, что имеется базовая система шифрования (и обычно также система проверки подписей). Безопасность протокола передачи ключей будет зависеть от безопасности этих вспомогательных схем. В теоретическом предложении группы предлагается, чтобы схема шифрования выполнялась в той же группе, что и протокол передачи ключей, хотя это не является существенным. В групповом транспортном протоколе злоумышленник знает всю группу и вид зашифрованных сообщений. Безопасность кроется в сложности определения поэлементно коммутирующих подгрупп , , которые держатся в секрете Бобом и Алисой, а также безопасностью реальной схемы шифрования.

Группа является платформой-кандидатом для данного типа протокола передачи ключей, если у нее есть либо хорошее конечное представление с работоспособными удовлетворяющими формами и большой абелевой подгруппой , либо две большие подгруппы , , которые коммутируют поэлементно. Хотя слово “большое” здесь неоднозначно, мы подразумеваем достаточно большое, чтобы из них можно было сделать произвольный выбор. В частности, например, циклические подгруппы являются неуместными. Также должна быть связь между группой, используемой для обмена ключами и методом шифрования, хотя это не является существенным. Стандартные группы кос имеют несколько возможностей для формирования удовлетворяющих форм и имеют большие коммутирующие подгруппы. К ним относятся полная группа автоморфизмов конечно порожденной свободной группы, матричная группа и поверхностной группы косы. Шпильрейн и Ушаков [7] использовали этот метод, используя группу Томпсона в качестве платформы.

* 1. Криптография в полициклических группах

В этом разделе мы вкратце рассмотрим криптографию, основанную полициклических группах, которая не изучена до конца, но имеет много существенных особенностей для идеальных групповых платформ.

Группа G называется полициклической, если она имеет ряд   
, в котором каждый является нормальной подгруппой группы и циклична для Ряд этого типа называется полициклическим рядом. Полициклические группы являются естественным некоммутативным обобщением циклических групп.

Каждая полициклическая группа имеет представление вида (1.11)

(1.11)

 для , где , , если и – слова в образующих . Если для каждого , то такое представление (1.11) называется последовательным полициклическим представлением. Каждый элемент в группе, определяемый этим согласованным полициклическим представлением, может быть записан однозначно в виде если и и ∈.   
Это единственное представление каждого элемента называется нормальной формой . Известно, что каждая полициклическая группа обладает последовательным полициклическим представлением. Следовательно, каждая полициклическая группа имеет нормальный вид. Это используется как основа для вычислений в полициклических группах.

Каждая полициклическая группа может быть вложена в , что показывает важные свойства полициклических групп. Так как матричное умножение разрешимо за полиномиальное время, групповое умножение в полициклических группах является эффективным. Доказано, что проблема поиска сопряженности в любой подгруппе общей линейной группы разрешима. Поскольку всякую полициклическую группу можно вложить как подгруппу в , проблема поиска сопряженности в полициклических группах разрешима. Сложность поиска проблемы сопряженности в полициклических группах неизвестна, но широко распространена гипотеза о экспоненциальном времени.

1. ИСПОЛЬЗОВАНИЕ ДНК В КРИПТОГРАФИИ

Размеры «цифровой вселенной» должны были превысить 16 зеттабайт к 2017 году. Значительная доля этих данных хранится в виде архивов. К примеру, компания Facebook недавно построила отдельный дата-центр для «холодного» хранения 1 эксабайта данных. Такое же количество информации способно уместиться в 1 мм3 ДНК. Сохранение данных в ДНК проходит в три этапа: преобразование цифровых данных в последовательность нуклеотидов ДНК, синтез молекул ДНК и, непосредственно, хранение данных. Чтобы данные считать, необходимо выделить требуемую последовательность из молекулы ДНК и преобразовать её в первоначальный вид. Время на проведение синтеза и секвенирования уменьшается экспоненциально как показано на рисунке 2.1 [8].

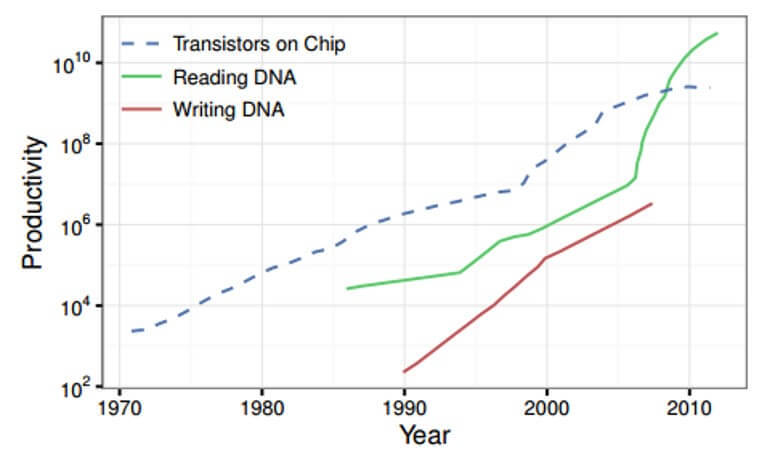


Рисунок 2.1 – Рост продуктивности использования ДНК

## 2.1 Хранение данных с использование ДНК

Молекула ДНК хранит информацию в четверичной системе счисления, по количеству нуклеотидов. Это компактный контейнер с плотностью записи в тысячи раз больше, чем у существующих носителей. Однако, чтобы технология перешла от научных испытаний к коммерческому использованию, требуется решить ряд проблем. Одна из них — специфика цифровой информации, в которой одни и те же биты могут многократно повторяться. Если многократно повторять один и тот же нуклеотид в молекуле ДНК, то это негативно влияет на стабильность кластера и информация может быть потеряна, даже при использовании избыточного дублирования и коррекции ошибок.  
 Исследователи из Европейского института биоинформатики  опубликовали работу с описанием способа, как можно существенно повысить стабильность ДНК. Попросту, они предлагают отказаться от четверичной системы (Base-4) в пользу троичной (Base-3), а четвёртый нуклеотид использовать в служебных целях для разбиения длинных цепочек согласно рисунка 2.2 [9].

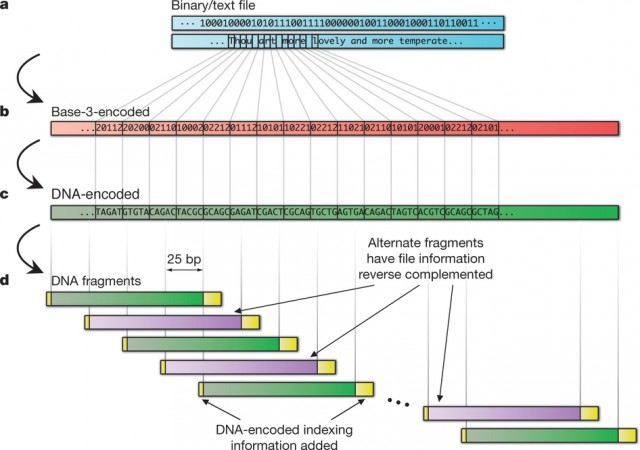
. 

Рисунок 2.2 – Процесс записи данных в ДНК

Во время эксперимента исследователи записали в ДНК почти мегабайт информации, в том числе все 154 сонета Шекспира в формате .txt, видеоролик с записью выступления Мартина Лютера Кинга продолжительностью 26 секунд, обложку журнала Bioinformatics Institute в формате .jpeg, научную работу с описанием структуры ДНК в формате .pdf, а также ещё один файл с описанием процесса кодирования. В общей сложности всё уместилось в 739 килобайт.  
При переходе с Base-4 на Base-3 мы теряем 25% информационной ёмкости, но даже в таком варианте учёные сообщают об информационной плотности записи 2,2 петабайта на 1 грамм биологического материала. Эксперимент показал надёжность считывания информации 100%. Теоретически, эта схема способна масштабироваться в пределах, превышающих объёмы всей существующей цифровой информации, пишут авторы исследования.

Исходя из нынешнего технологического прогресса в области синтеза и секвенирования, носители ДНК для записи информации должны появиться в открытой продаже в течение десяти лет. Хотя ДНК позволяет хранить информацию тысячелетиями, первые коммерческие носители будут продаваться с гарантией до 50-ти лет, считают исследователи.

## 2.2 Процесс кодирования информации

ДНК содержит четыре типа нуклеотидов: аденин (A), цитозин (С), гуанин (G) и тимин (T). Нить ДНК представляет собой линейную последовательность этих нуклеотидов. Таким образом, у нас есть четыре кодовых   
символа (A, C, G и T), поэтому очевидным подходом к хранению двоичных данных будет их кодирование в четверичной системе счисления, например, 0=A, 1=C, 2=G, и 3=T, на основе этих данных, в приложении А, представлен реализованный алгоритм шифрования основанный на RSA с использованием нуклеотидов. Однако стоит учитывать, что синтез и секвенирование подвержены ошибкам. Вероятность ошибок можно снизить, если закодировать двоичную информацию не в четверичной, а в троичной   
системе счисления, как показано на рисунке ниже. Чтобы избежать неэффективного преобразования исходных двоичных данных в троичную систему счисления, используется код Хаффмана. Пример такого преобразования показан на рисунке 2.3.

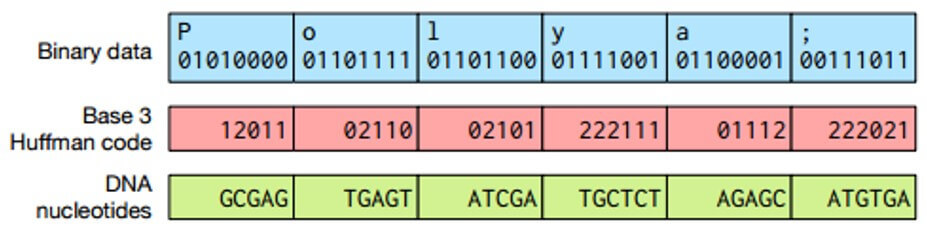


Рисунок 2.3 – Процесс преобразование слова в нуклеотиды ДНК

Каждая из трех цифр соотносится с нуклеотидом ДНК в соответствии с рисунком 2.4, причем нуклеотиды в цепочке не повторяются, что приводит к снижению ошибки секвенирования.

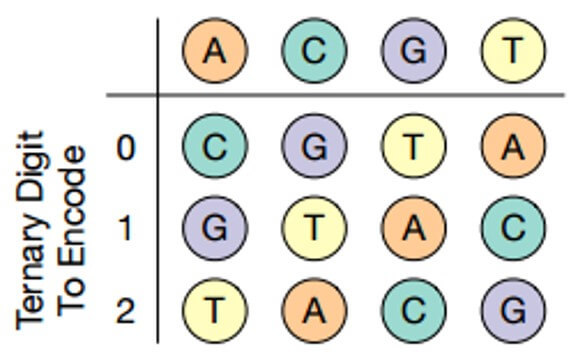


Рисунок 2.4 – Кодирования нуклеотидов

Чтобы обеспечить возможность произвольного доступа к данным, ученые организовали перевод ключей в уникальные последовательности праймеров. Праймеры – это короткие синтетические нити, определяющие начало и конец области, которую необходимо амплифицировать. Праймеры обеспечивают произвольный доступ с помощью полимеразной цепной реакции, которая генерирует множество копий ДНК. Цепи конкретного объекта имеют общий праймер, а разные цепи с одним и тем же праймером различаются по адресам.  «Контролируя последовательности, которые используются как праймеры для полимеразной цепной реакции (ПЦР), мы можем указать, какие нити в решении будут проходить амплификацию. Для того чтобы считать значение ключа в решении, мы просто проводим ПЦР, используя соответствующий этому ключу праймер», – говорят ученые.

## 2.3 ДНК-криптография

ДНК криптография – это новое поле, основанное на исследованиях в ДНК вычислений и новых технологиях, таких как PCR (Polymerase Chain Reaction),. ДНК-вычисления обладают высоким уровнем вычислительной способности и способны хранить огромное количество данные. Грамм ДНК содержит 1021 основание ДНК, что эквивалентно 108 терабайтам данных. В криптографии ДНК мы используем существующую биологическую информацию из общедоступных баз данных ДНК для кодирования открытого текста. Криптографический процесс может использовать разные методы. В методах одноразовых блокнотов описаны наиболее эффективные алгоритмы безопасности. В случае одноразового блокнота, открытый текст комбинируется с секретным случайным ключом или нашим блокнотом, который используется только один раз. Блокнот сочетается с открытым текстом, используя типичное модульное сложение или операцию XOR. Скорость алгоритма должна быть довольно высокой. Биологический фон: ДНК является аббревиатурой для дезоксирибонуклеиновой кислоты, которая является зародышевой плазмой всех стилей жизни. ДНК является своего рода биологической макромолекулой и состоит из нуклеотидов. Как уже выше упоминалось, каждый нуклеотид содержит одно основание и существует четыре типа оснований: аденин (А) и тимин (Т) или цитозин (С) и гуанин (G), соответствующий четырем типам нуклеотидов. Одноцепочная ДНК построена с ориентацией: один конец называется 5 ', а другой конец называется 3'. Обычно ДНК существует как двухцепочные молекулы в природе. Две взаимодополняющие нити ДНК удерживаются вместе, образуя структуру двойной спирали. Структура двойной спирали была обнаружена Уотсоном и Криком; Таким образом, комплементарная структура называется комплементарностью Уотсона Крика. Их открытие является одним из величайших научных открытий 20-го века. Развитие ДНК-криптографии, выигрывает от развития ДНК-вычислений (также называемых молекулярными вычислениями или биологическими вычислениями). С одной стороны, криптография всегда имеет некоторые отношения с соответствующей вычислительной моделью более или менее. С другой стороны, некоторые биологические технологии используются в ДНК-криптографии.

### 2.3.1 Криптосистема ДНК с использованием замены

В одноразовой системе замещения используется группа двоичного сообщения открытого текста и группа таблиц, определяющая случайное отображение на зашифрованный текст. Входная цепочка имеет длину и разбивается на незашифрованные слова фиксированной длины. Таблица отображает все возможные строки незашифрованного текста с фиксированной длиной в соответствующие строки зашифрованного текста, так что существует уникальное обратное отображение.

Шифрование происходит путем замены каждого слова открытого текста ДНК соответствующим шифрованным словом ДНК. Отображение реализуется с использованием длинного ДНК-блокнота, состоящего из множества сегментов, каждый из которых указывает на одно слово с открытого текста для шифрования словосочетаний. Слово с открытым текстом действует связывания праймера, который затем удлиняется. Это приводит к формированию текстовой пары открытого текста и зашифрованного текста.

Идеальная одноразовая библиотека будет содержать огромное количество блокнотов, и каждый из них обеспечит совершенно уникальное, случайное отображение из незащищенных слов в зашифрованные слова.

Повторяющийся блок состоит из: одного слова последовательности , из набора слов, соответствующих шифру или кодовой книге и префикса . Приметим, что включает в себя уникальную подпоследовательность, которая предотвращает атаки частотного анализа путем сопоставления нескольких экземпляров одного и того же открытого текста сообщения с различными зашифрованными словами. Кроме того, этот префикс может быть необязательно использован для кодирования положение слова в сообщении.

Каждая пара последовательностей однозначно связывает текстовое слово с шифровым словом.

Последовательность стопоров запрещает расширение растущей цепи ДНК за границу парного шифрованного слова. Используя эту тему, создается библиотека уникальных цепочек кодовых книг. Каждая отдельная цепочка из этой библиотеки кодовой книги задает конкретный уникальный набор парных слов[10].

Одноразовый блокнот состоит из цепочки ДНК длиной , содержащей копииповторяющегося шаблона: шифрованное слово длиной , слово открытого текста длиной и стопорной последовательности длины . Заметим, что длина слова растет логарифмически по всей длине блокнота. Определенно ; и = , где фиксированные целочисленные константы и . Каждый повторитель задает единую пару сопоставления, и ни одно слово из кодовой книги или текстовое слово не будет использоваться более одного раза в любом блокноте. Поэтому, учитывая шифрованное слово , мы уверены, что оно отображает только одно слово с открытым текстом и наоборот. Последовательность стоппера действует как «пунктуация» между повторяющимися звеньями, так что ДНК-полимераза не сможет продолжать копирование нити матрицы (блокнота). Стоппер – последовательности состоят из последовательности идентичных нуклеотидов, которые действуют для прекращения копирования цепи ДНК-полимеразой, учитывая отсутствие комплементарного нуклеотидтрифосфата в пробирке. Например, последовательность TTTT будет действовать в качестве точки остановки, если полимеризационной смеси не хватает своего комплементарного основания.

Экспериментальная осуществимость зависит от следующих факторов: размер лексикона, который представляет собой число пар слов открытого текста-зашифрованного текста, размер каждого слова, количество одноразовых ДНК, блокноты, которые могут быть созданы в цикле синтеза, и длина каждого сообщения, которое должно быть зашифровано. Если бы используемый лексикон состоял из слов английского языка, его размер находился бы в диапазоне от 10 000 до 25 000 словных пар. Если по экспериментальным причинам требуется меньший лексикон, тогда используемые слова могут представлять собой более малый набор, такой как ASCII-символы, что приводит к размеру словаря 128.

## 2.4 Преимущество и недостатки в использовании ДНК

В будущем такие системы потенциально позволят сохранить огромное количество данных на микроскопических носителях. Представьте себе «флешку» объемом 100 мм3, способную хранить порядка 100 000 ПБайт данных. Однако пока что самым крупным препятствием к внедрению подобных технологий остается время. Расшифровка и чтение молекулы ДНК занимает многие часы. Поэтому такой тип хранилищ вряд ли подойдет для содержания часто используемых данных, однако способен перевернуть наше представление о долговременных хранилищах в дата-центрах. Также на сегодняшний день стоимость кодирования информации в ДНК оценивается примерно в [$12400 за мегабайт](http://mashable.com/2013/01/23/dna-replace-hard-drive/), стоимость считывания — $220 за 1 МБ, что является явным недостатком. Еще одним минусом является то, что необходимы специальные емкости для хранения ДНК, во избежание ее распада и потери данных.

1. КРИПТОГРАФИЯ ОСНОВАННАЯ НА ХАОСЕ

Для многих публикаций по криптосистемам на основе хаоса описаны только базовые понятия. Однако, вообще говоря, детали реализации очень важны для криптоаналитиков, чтобы оценить безопасность криптосистемы. Кроме того, шифрование. Таким образом, отсутствие деталей реализации в целом не позволяет изучить надежность и значимость предлагаемой криптосистемы посредством анализа безопасности и оценки производительности.

* 1. Внедрение хаотических систем

Существуют два основных подхода к разработке криптосистем, основанных на хаосе: аналоговые и цифровые. Первые, как правило, основаны на синхронизации хаоса, и связанные с ним хаотические системы реализуются в аналоговой форме. Вторые не зависят от синхронизации хаоса и хаотические системы полностью реализованы в цифровой форме.

Для аналоговой реализации схемы (по крайней мере явный вид системы дифференциальных уравнений) должны быть приведены достаточное количество деталей; Тогда как для цифровой реализации должны быть представлены следующие детали: конечная вычислительная точность, принятая цифровая арифметика (с фиксированной точкой или с плавающей запятой), конфигурация аппаратного / программного обеспечения.

Когда хаотические системы полностью или частично реализуются в цифровой форме, произойдет динамическое ухудшение, то есть динамические свойства цифровых хаотических систем могут стать неидеальными. Наиболее известной проблемой является существование многих хаотичных орбит короткой длины, которые могут ослабить желаемые статистические свойства цифровых хаотических шифров, а затем снизить безопасность шифров. Эта проблема широко изучалась в последние два десятилетия, и было обнаружено, что такая динамическая деградация может действительно вызывать нарушения безопасности в некоторых криптосистемах, основанных на хаосе [11]. Для преодоления этой проблемы необходимо использовать некоторые методы для уменьшения динамической деградации цифровых хаотических систем. Эффективной встречной мерой является своевременное внедрение в основу хаотической системы небольшого псевдослучайного сигнала[12].

* 1. Внедрение криптосистем

В криптографическом сообществе есть два хорошо известных высказывания: «довольно легко разработать безопасный, но очень медленный шифр» и «довольно просто разработать безопасный, но очень большой шифр». Если безопасность цифрового хаотического шифрования достигается без эффективной работы, его значимость будет тривиальной и не будет принята как практиками, так и криптоаналитиками, потому что в реальном мире производительность и стоимость внедрения являются важными проблемами, помимо безопасности. Необходимо изучить и объяснить стоимость, связанную с реализацией и исполнением шифра, включая такие аспекты, как вычислительная эффективность, размер программы и требования к рабочей памяти в программных реализациях на различных общих платформах и область микросхемы в специализированных аппаратных реализациях. Таким образом, уровень безопасности, производительности и простоты реализации являются тремя основными критериями оценки новых криптосистем.

Как раньше было упомянуто, криптосистемы обычно делятся на два общих типа: симметричные и асимметричные ключи. Первая группа криптосистем использует один и тот же секретный ключ как для шифрования, так и для дешифрования и очень быстрая, поэтому подходит для обработки больших объемов данных на высокой скорости, таких как шифрование видео. Существует еще одно разделение между алгоритмами с симметричными ключами: блочными и потоковыми шифрами. Блочные шифры шифруют исходное сообщение, группируя символы в блоках из двух или более элементов, так что каждый блок зашифрован / дешифрован всегда одним и тем же способом. Блочные шифры обычно состоят из начального преобразования, криптографической функции, повторяемой определённое количество раз (или «раундов») и окончательного преобразования. Секретный ключ расширяется с использованием некоторого алгоритма, чтобы иметь достаточный ключевой материал для использования в каждом раунде шифрования. Среди наиболее распространенных блочных шифров AES, TripleDES, IDEA, DES, RC5. С другой стороны, потоковые шифры генерируют псевдослучайный поток символов, используя детерминированный общественный алгоритм, управляемый секретным ключом. Сообщение смешивается с этой последовательностью, также известной как «ключевой поток», обычно через сумму по модулю 2, что приводит к шифрованному тексту. Среди наиболее распространенных потоковых шифров A5, E0, RC4, SEAL. Длина ключа симметричных шифров обычно составляет от 128 до 256 бит.

Когда для шифрования и дешифрования используются два разных ключа, криптосистема является ассиметричной. Обычно один ключ пары общеизвестен, а другой – закрыт. Эти алгоритмы намного медленнее, потому что в целом они связаны с тяжелыми арифметическими операциями с большими целыми числами, такими как дискретный логарифм или экспоненцирование по модулю. Как следствие, они используются для задач, предполагающих шифрование небольшого объема данных, таких как соглашение о секретном ключе, цифровые подписи, аутентификация и т. Д. Наиболее широко используемым алгоритмом с открытым ключом является RSA. Длина ключа алгоритмов с открытым ключом обычно составляет от 1024 до 4096 бит[13,14,15].

Естественно, можно использовать такие широко известные традиционные шифры, как AES, DES, IDEA, RC5, RSA, как хорошие рекомендации для оценки приемлемости стоимости внедрения и скорости шифрования новой криптосистемы на основе хаоса. Обратите внимание, что, как упоминалось выше, скорость шифрования алгоритмов с открытым ключом, как правило, гораздо медленнее, чем скорость шифрования секретным ключом, поэтому следует сравнивать хаотический шифр с тем же типом традиционных шифров. Поскольку большинство хаотических криптосистем основаны на секретных ключах, мы будем в основном обращать внимание на скорость шифрования хаотических шифров с секретным ключом. На скорость шифрования оказывает наибольшее воздействие тактовая частота процессора. Помимо основной частоты процессора, скорость шифрования программной реализации сильно зависит от многих других проблем, таких как структура ЦП, размер памяти, лежащая в основе платформа ОС, язык разработки и все опции компилятора, и поэтому необходимо указать скорость шифрования новой криптосистемы на основе хаоса с такими деталями. Кроме того, хорошо известно, что оптимизация кода очень важна для резкого увеличения скорости алгоритма, например, использование умножений вместо деления полезно для увеличения скорости в несколько раз (как для фиксированной, так и с плавающей запятой, точечной арифметики)[16]. Таким образом, несколько бессмысленно сравнивать скорости шифрования двух шифров без использования тех же самых сред разработки и методов оптимизации.

Можно сформировать некоторые базовые предложения для проектирования быстрых и недорогих цифровых хаотических шифров: избегайте использования нескольких итераций для каждого шага шифрования, избегайте использования сложной арифметики с плавающей запятой, выбирайте простейшие хаотические системы с точки зрения реализации.

* 1. Анализ безопасности

Здесь идет основа криптоанализа, при условии, что безопасность является главной заботой в криптосистеме, хотя обычно это самая сложная оценка. После того, как была разработана новая криптосистема, ее всегда следует оценивать с помощью некоторого базового анализа безопасности. Хотя этот анализ не может содержать все возможные атаки на новый шифр, он должен охватывать по крайней мере некоторые наиболее известные атаки, чтобы проверить, сможет ли он пройти базовые тесты. Этот анализ помогает выявить и исправить дефекты и недостатки до публикации новой схемы.

Прежде всего, чтобы противостоять обычным атакам, спроектированная криптосистема должна иметь следующие два основных криптографических свойства: путаницу и различие. Первое свойство предназначено для того, чтобы сделать связь между ключом и зашифрованным текстом настолько сложной, насколько это возможно, тем самым срывая попытки изучить зашифрованный текст, ищущий повторения и статистические шаблоны. Второе свойство относится к перераспределению или расширению битов в сообщении, так что влияние отдельных текстовых или ключевых битов распространяется по большей части зашифрованного текста. Здесь нужны некоторые правила, соответствующие этим двум свойствам. Для достижения путаницы статистические свойства зашифрованного текста, такие как распределение, корреляция и разностная вероятность, должны быть независимыми от точного значения ключа и открытого текста. Как отметил Фридрих в [17], среди многих других желательных свойств с точки зрения безопасности хорошая криптосистема (основанная на симметричном или асимметричном шифровании) должна быть чувствительной к ключам (замена одного бита в ключе создает совершенно разный зашифрованный текст при применении к одному и тому же открытому тексту), быть чувствительной в отношении открытого текста (переворот одного бита в открытом тексте создает совершенно разный зашифрованный текст).

* 1. Базовые правила формирования устойчивой криптосистемы

Подводя итоги этого раздела, можно сформировать ряд рекомендаций, которые не являются необходимыми, но могут значительно повысить безопасность криптосистемы:

Предлагаемое правило 1. Должно быть обеспечено подробное описание реализации хаотических систем.

Предлагаемое правило 2. Для хаотических систем, реализованных в цифровой форме, следует учитывать отрицательные эффекты динамической деградации при тщательной оценке.

Предлагаемое правило 3. Без потери безопасности криптосистема должна быть легко реализована с приемлемыми затратами и скоростью.

Предлагаемое правило 4. Для двух ключей с малейшей разницей, никакой разброс между соответствующими шифротекстами не может быть обнаружен никаким из известных статистических анализов.

Предлагаемое правило 5. Шифротекст должен быть статистически неразличим после применения случайной функции и должен быть статистически одинаковым для всех ключей.

# ВЫВОДЫ

В данной работе были проанализированы криптосистемы, основанные на неабелевых группах, были рассмотрены алгоритмы обмена ключами реализованные на модулярной арифметике, так же эти алгоритмы были представлены на абелевых группах, что дает большую стойкость, за счет сложности нахождения сопряжений между группами и подгруппами.

Анализ ДНК-криптографии показал, что использование ее на данном этапе развития не целесообразно за счет потребление больших вычислительных средств и дороговизны всей процедуры, также большим минусом является то, что процесс записи и извлечения занимает большое время, примерно в 5 раз больше, чем с использованием алгоритмах в групповой математике. Положительной стороной есть то, что технологии развиваются чрезвычайно быстро и через несколько лет данный вид защиты информации может быть реализован без больших финансовых и вычислительных трудностей.

Криптосистемы, основанные на хаосе, являются чрезвычайно чувствительными к выбору параметров, и для их создания был сформирован ряд правил, которые не являются необходимыми, но есть желательными. Для построения успешной такой криптосистемы были определены два свойства: путаницу и различие. Путаница используется для усложнения понимания алгоритма связи ключа с шифротекстом, различие – должно минимизировать количество шаблонов в самой криптосистеме.

# ПЕРЕЧЕНЬ ССЫЛОК

1. A.G. Myasnikov, V.Shpilrain and A. Ushakov, Group-Based Cryptography Advanced Courses in Mathematics, CRM Barcelona, 415, 2007.
2. P.Dehornoy, Braid-Based Cryptography Cont. Math., 360, 2004, 5–34.
3. W. Magnus, A. Karass and D. Solitar Combinatorial Group Theory, Wiley Interscience,New York, 527, 1968.
4. G. Baumslag, B.Fine, and X.Xu, Cryptosystems Using Linear Groups Appl. Alg. in Engineering, Communication and Computing 17, 290, 2006, 205-217.
5. K.Ko, J.Lee, J.H. Cheon, J.W. Han, J.Kang, C.Park, New Public-Key Cryptosystem Using Braid Groups , Advances in Cryptology - CRYPTO 2000 Santa Barbara CA , Lecture Notes in Computer Science, Springer 1880, 2000, 166-183.
6. D.Kahrobaei and B.Khan, A Non-Commutative Generalization of the ElGamal Key Exchange using Polycyclic Groups Proceeding of IEEE, GLOBECOM, 2006.
7. V.Shpilrain and A. Ushakov, The Conjugacy Search Problem in Public Key Cryptography; Unnecessary and Insufficient Applicable Algebra in Engineering, Communication and computing, 17, 2006 285-289.
8. «Заложено природой»: Система хранения данных на основе ДНК [Электронный ресурс] // URL: <https://m.habrahabr.ru/company/1cloud/blog/306656/> (дата обращения: 11.05.2017).
9. Надёжное хранение информации в ДНК (2,2 петабайта на грамм) [Электронный ресурс] // URL: <https://geektimes.ru/post/166889/> (дата обращения: 21.05.2017).
10. DNA Data cryptography [Электронный ресурс] // URL: <https://www.slideshare.net/mayukhmaitra/dna-cryptography> (дата обращения: 25.05.2017).
11. Alvarez, G. & Li, S. “Breaking cryptography with chaos at the physical level,” arXiv:nlin.CD/0403029, 215, 2004.
12. Cerm´ak, J. “Digital generators of chaos,” Phys. Lett. 525, 1996, 151–160.
13. Stinson, D. R. [1995] Cryptography: Theory and Practice (CRC Press).
14. Schneier, B. [1996] Applied Cryptography – Protocols, algorithms, and souce code in C second edn. (John Wiley & Sons, Inc., New York, USA).
15. Menezes, A. J., van Oorschot, P. C. & Vanstone, S. A. Handbook of Applied Cryptography (CRC Press), 1997.
16. Fog, A. [2000] “How to optimize for the Pentium family of microprocessors,” Online document, <http://www.codingnow.com/2000/download/pentopt.htm>.
17. Fridrich, J. “Symmetric ciphers based on two-dimensional chaotic maps,” Int. J. Bifurc. Chaos 8, 1465, 1998, 1259–1284.

# ПРИЛОЖЕНИЕ А

Входные и полученные данные

За основу брался алгоритм RSA, входными данными для него были:

p = 3557, q = 2579, N = 9173503, e = 3, m = 111111, c = 4051753, где p и q –простые числа для генерации модуля, N – сгенерированный модуль, e – открытая константа, m – наше сообщение, которое шифруется, c – сама криптограмма. Потом криптограмма наша подавалась на функцию преобразования десятичного числа в четверичное и мы получали следующий результат – 33131030221, потом это число подавалось на функцию преобразования четверичного числа в ДНК-нуклеотиды и мы получали – GGTGTAGACCT. Процедура расшифрования подобна, только используются другие функции, а именно – функция преобразования из ДНК-нуклеотидов в четверичную систему, затем функция преобразования с четверичной системы исчисления в десятичную, в конце мы получаем нашу криптограмму – 4051753, и при использовании заданных ранее параметров в RSA алгоритме, восстанавливаем наше сообщение m – 111111.

# ПРИЛОЖЕНИЕ Б

# Исходные коды программы

*# rsa.py - RSA algorithm***def** main():  
 n = e = d = 0  
 **while** 1:  
 print(**"""  
 1. Set Public Key  
 2. Encode  
 3. Decode  
 0. Quit  
 Your choice? """**, end=**""**)  
 choice = int(input())  
 **if not** choice:  
 **return  
 if** choice == 1:  
 n, e, d = set\_keys()  
 **if** choice == 2:  
 **if not** n:  
 n = int(input(**"Public Key: "**))  
 e = int(input(**"e: "**))  
 encode(n, e)  
 **if** choice == 3:  
 **if not** d:  
 n, e, d = set\_keys()  
 decode(d, n)  
  
**def** set\_keys():  
 *"""This fuction asks for 2 primes.  
 It sets a public key and an encoding number, 'e'."""* p = int(input(**"p: "**))  
 q = int(input(**"q: "**))  
 n = p \* q  
 m = (p - 1) \* (q - 1)  
 e = get\_e(m)  
 print(**"N = "**, n, **"\ne = "**, e)  
 d = get\_d(e, m)  
 **while** d < 0:  
 d += m  
 **return** [n, e, d]  
  
**def** encode(n, e):  
 *"""This function asks for a number and encodes it using 'n' and 'e'."""* **while** 1:  
 c = int(input(**"Number to encode: "**))  
 **if not** c:  
 **return** print(pow(c, e, n))  
  
**def** decode(d, n):  
 *"""This function asks for a number and decodes it using 'd' and 'n'."""* **while** 1:  
 c = int(input(**"Number to decode: "**))  
 **if not** c:  
 **return  
 else**:  
 print(pow(c, d, n))  
  
**def** even(x):  
 *"""True if x is even."""* **return** x % 2 == 0  
  
**def** get\_e(m):  
 *"""Finds an e coprime with m."""* e = 2  
 **while** gcd(e, m) != 1:  
 e += 1  
 **return** e  
  
**def** gcd(a, b):  
 *"""Euclid's Algorithm: Takes two integers and returns gcd."""* **while** b > 0:  
 a, b = b, a % b  
 **return** a  
  
  
**def** get\_d(e, m):  
 *"""Takes encoding number, 'e' and the value for 'm' (p-1) \* (q-1).  
 Returns a decoding number."""* x = lasty = 0  
 lastx = y = 1  
 **while** m != 0:  
 q = e // m  
 e, m = m, e % m  
 x, lastx = lastx - q \* x, x  
 y, lasty = lasty - q \* y, y  
 **return** lastx  
  
  
**if** \_\_name\_\_ == **"\_\_main\_\_"**:  
 main()

Функция преобразования числа с десятичной системы исчисления в четверичную

*# converter from 10-base to 4 -base  
  
"""This fuction asks for 3 primes.  
 It sets a nconverting number, convert to\_base and convert from\_base"""***def** convert\_base(num, to\_base=10, from\_base=10):  
 *# first convert to decimal number* **if** isinstance(num, str):  
 n = int(num, from\_base)  
 **else**:  
 n = int(num)  
 *# convert decimal to 'to\_base' base* alphabet = **"0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ"  
 if** n < to\_base:  
 **return** alphabet[n]  
 **else**:  
 **return** convert\_base(n // to\_base, to\_base) + alphabet[n % to\_base]  
 *# read from file*f = open(**'d:/Diplom/files/en.txt'**,**'r'**)  
en1 = f.read()  
en = int(en1)  
f.close()  
 *# write to file*en4 = convert\_base(en,4,10)  
f = open(**'d:/Diplom/files/en4.txt'**, **'w'**)  
en41 = str(en4)  
f.write(en41)  
f.close()

Функция преобразования числа с четверичной системы исчисления в десятичную

*# converter from 4-base to 10 -base***from** functools **import** reduce  
*# read from file*  
f = open(**'d:/Diplom/files/dec4.txt'**,**'r'**)  
en1 = f.read()  
f.close()  
res=reduce(**lambda** x,y:x\*4+(**'0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ'**.find(y)), en1.strip().upper(),0)  
print(res)  
*# write to file*  
f = open(**'d:/Diplom/files/dec.txt'**, **'w'**)  
res1 = str(res)  
f.write(res1)  
f.close()

Функция преобразования числа с четверичной системы исчисления в ДНК-нуклеотиды

*#encode from 4-base to DNA*

*# read from file*f = open(**'d:/Diplom/files/en4.txt'**,**'r'**)  
en1 = f.read()  
en = int(en1)  
f.close()  
print(en)  
leng=len(en1)  
i = 1

*#conveting to DNA-Nucleotides*

**for** i **in** en1:  
 k = int(i)  
 **if** k == 0:  
 f = open(**'d:/Diplom/files/enDNA.txt'**, **'a'**)  
 f.write(**'A'**)

**elif** k == 1:  
 f = open(**'d:/Diplom/files/enDNA.txt'**, **'a'**)  
 f.write(**'T'**)  
 **elif** k == 2:  
 f = open(**'d:/Diplom/files/enDNA.txt'**, **'a'**)  
 f.write(**'C'**)  
 **elif** k == 3:  
 f = open(**'d:/Diplom/files/enDNA.txt'**, **'a'**)  
 f.write(**'G'**)

Функция преобразования числа из ДНК-нуклеотидов

в четверичную систему исчисления

*#decode from DNA to 4-base  
  
# read from file*f = open(**'d:/Diplom/files/enDNA.txt'**,**'r'**)  
en1 = f.read()  
f.close()  
print(en1)  
leng=len(en1)  
i = 1

*#conveting to Base-4*

**for** i **in** en1:  
 **if** i == **'A'**:  
 f = open(**'d:/Diplom/files/dec4.txt'**, **'a'**)  
 f.write(str(0))  
 **elif** i == **'T'**:  
 f = open(**'d:/Diplom/files/dec4.txt'**, **'a'**)  
 f.write(str(1))  
 **elif** i == **'C'**:  
 f = open(**'d:/Diplom/files/dec4.txt'**, **'a'**)  
 f.write(str(2))  
 **elif** i == **'G'**:  
 f = open(**'d:/Diplom/files/dec4.txt'**, **'a'**)  
 f.write(str(3))

ВІДОМІСТЬ АТЕСТАЦІЙНОЇ РОБОТИ

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Позначення | | | | | Найменування | Дод. від. | | | | |
|  | | | | | Текстові документи |  | | | | |
|  | | | | |  |  | | | | |
| ГЮІК. ХХХХХХ.700Ст.26Пз | | | | | Пояснювальна записка | 56 с. | | | | |
|  | | | | |  |  | | | | |
|  | | | | | Графічні матеріали |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | | Презентаційний матеріал | 12 слайдів | | | | |
|  | | | | |  |  | | | | |
|  | | | | | Інші документи |  | | | | |
|  | | | | | Рецензія | 1 с. | | | | |
|  | | | | | Відгук керівника | 1 с. | | | | |
|  | | | | | СD | 1(шт.) | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  | | | | |  |  | | | | |
|  |  |  |  |  | ГЮІК. ХХХХХХ. 700Ст.26Пз | | | | | |
|  |  |  |  |  |
| Змін. | Арк. | № докум. | Підпис | Дата |
| Розроб. | | Євгеньєв А.М. |  |  | Аналіз методів побудови криптосистем на групах | Літ. | | | Аркуш | Аркушів |
| Перевір. | | Халімов Г. З. |  |  |  |  |  | 1 | 1 |
| Н.контр. | | Шеханін К.Ю. |  |  | ХНУРЕ  Кафедра БІТ | | | | |
| Затверд. | | Халімов Г. З. |  |  |
|  | |  |  |  |