Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИУ «ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ»

КАФЕДРА ИУ7 «ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ЭВМ И ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ

НА ТЕМУ:

Методы асимметричного шифрования данных

Студент **ИУ7-54Б** _____ **Р.Д. Яковлев**

Руководитель _____ **А.С. Кострицкий**

2024 г.

РЕФЕРАТ

Научно-исследовательская работа 16 с., 4 рис., 2 табл., 5 ист., 1 прил.

ШИФРОВАНИЕ, АСИММЕТРИЧНОЕ ШИФРОВАНИЕ, RSA, DSA, ECDSA.

Объект исследования – асимметричное шифрование.

Цель работы – проанализировать существующие методы асимметричного шифрования данных. Поставленная цель достигается путём рассмотрения и разбора реализации существующих алгоритмов асимметричного шифрования.

В данной работе проводилось изучение принципов работы существующих алгоритмов асимметричного шифрования, а также их сравнение и анализ.

Результат данной работы показал, что каждый алгоритм имеет свои преимущества и свои недостатки и применим в зависимости от условий и требований.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Шифрование данных – ключевая технология в области информационной безопасности, обеспечивающая защиту конфиденциальной информации в процессе её передачи и хранения. В условиях быстрого развития технологий и повсеместного использования интернета, вопросы безопасности данных становятся всё более актуальными.

Важность шифрования невозможно переоценить, так как оно является основой для создания доверительных и защищённых коммуникационных каналов, а также для предотвращения утечек данных и кибератак.

Одним из вариантов шифрования данных является асимметричное шифрование данных.

Асимметричное шифрование данных, также известное как криптография с открытым ключом, представляет собой одну из самых мощных и безопасных технологий защиты информации. В отличие от симметричного шифрования, где для шифрования и расшифровки данных используется один и тот же ключ, в асимметричной криптографии применяется пара ключей: открытый и закрытый. Открытый ключ используется для шифрования данных, а закрытый — для их расшифровки. Эта схема гарантирует высокий уровень безопасности, так как даже если злоумышленник получит доступ к открытому ключу, он не сможет расшифровать информацию без закрытого ключа, который хранится в секрете у владельца. [1]

Цель работы – проанализировать существующие методы асимметричного шифрования данных.

Для достижения цели необходимо выполнить следующие задачи:

1) ознакомиться с существующими алгоритмами асимметричного шифрования данных;
2) выделить их сходства и различия;
3) сформулировать критерии для сравнения алгоритмов;
4) провести сравнительный анализ алгоритмов по сформулированным критериям.

1 Анализ предметной области

1.1 Шифрование данных

История шифрования началась с древних времён, когда ещё не было компьютеров и других современных вычислительных средств. Наши предки использовали шифрование только для замены символов. Процесс расшифровки был долгим и утомительным.

В прежние десятилетия, когда у людей не было проблем с защитой информации своих компьютеризированных систем, шифрование использовалось только государственными органами для облегчения передачи секретной информации во время их общения. В настоящее время шифрование широко применяется для передачи данных через сети, устройства Bluetooth, банкоматы и так далее [2].

1.2 Симметричное шифрование данных

Симметричное шифрование, также известное как шифрование с секретным ключом, выполняется с использованием одного и того же ключа для шифрования и расшифровки информации. Ключ также отправляется получателю вместе с секретным текстом, и с помощью этого ключа выполняется преобразование зашифрованного текста в обычный [3].

1.3 Асимметричное шифрование данных

В асимметричном шифровании, также известном как шифрование с открытым ключом, есть 2 разных ключа – открытый и закрытый. Ключи расшифровки и шифрования отличаются друг от друга. Ключ к шифру – это открытый ключ, а дешифратор – закрытый ключ. Текст, зашифрованный с помощью открытого ключа, может быть расшифрован только с помощью закрытого ключа, принадлежащего этому пользователю. Это делает асимметричный алгоритм более эффективным, чем симметричное шифрование [4].

1.4 Надёжность асимметричного шифрования

Теоретически приватный ключ от асимметричного шифра можно вычислить, зная публичный ключ и механизм, лежащий в основе алгоритма шифрования (последнее – открытая информация). Надёжными считаются шифры, для которых это нецелесообразно с практической точки зрения. Так, на взлом шифра, выполненного с помощью алгоритма RSA с ключом длиной 768 бит на компьютере с одноядерным процессором AMD Opteron с частотой 2,2 ГГц, бывшем в ходу в середине 2000-х, ушло бы 2000 лет.

При этом фактическая надёжность шифрования зависит в основном от длины ключа и сложности решения задачи, лежащей в основе алгоритма шифрования, для существующих технологий. Поскольку производительность вычислительных машин постоянно растёт, длину ключей необходимо время от времени увеличивать. Так, в 1977-м (год публикации алгоритма

RSA) невозможной с практической точки зрения считалась расшифровка сообщения, закодированного с помощью ключа длиной 426 бит, а сейчас для шифрования этим методом используются ключи от 1024 до 4096 бит, причём первые уже переходят в категорию ненадёжных.

Что касается эффективности поиска ключа, то она незначительно меняется с течением времени, но может скачкообразно увеличиться с появлением кардинально новых технологий (например, квантовых компьютеров). В этом случае может потребоваться поиск альтернативных подходов к шифрованию [1].

2 Существующие решения

В этом разделе рассмотрено 3 наиболее распространённых алгоритма асимметричного шифрования, а также приведён результат сравнительного анализа.

2.1 Обзор существующих алгоритм асимметричного шифрования

2.1.1 RSA

Алгоритм RSA был разработан в 1978 году Роном Ривестом, Ади Шамиром и Леонардом Адлеманом, и его название происходит от первых букв фамилий авторов. RSA используется как для шифрования, так и для дешифрования данных. Этот алгоритм применяется для обеспечения конфиденциальности и создания цифровых подписей. Его принцип работы основан на модульной арифметике больших чисел, где создаётся целое число, являющееся произведением двух больших простых чисел. Хотя размер ключа повышает надёжность алгоритма, это также является причиной медленной работы транзакций. Алгоритм RSA состоит из трёх основных шагов: генерация ключа, шифрование и дешифрование [2]. На рисунке 2.1 представлена иллюстрация работы алгоритма RSA.

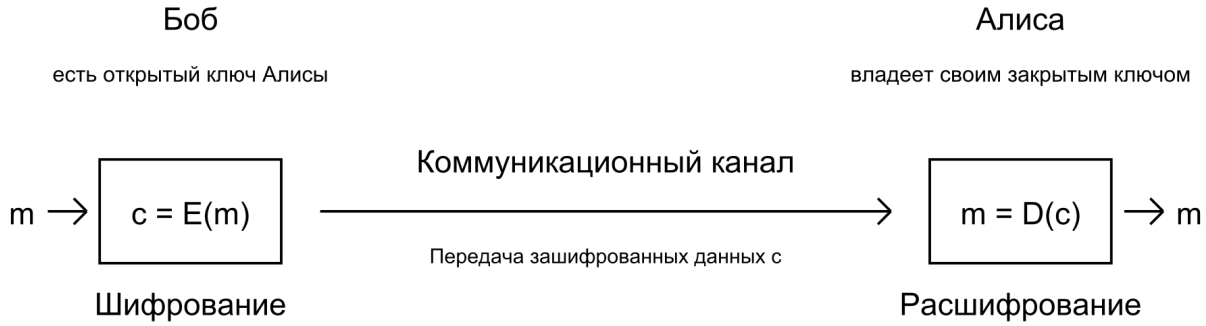

Рисунок 2.1 — Иллюстрация работы RSA

Алгоритм генерации открытого и закрытого ключей состоит из следующих шагов:

1) выбор двух больших простых чисел p и q;

2) определение n как результат $p \cdot q$;

3) выбор случайного d взаимно простого с $(p-1) \cdot (q-1)$;

4) определение e такого что $(e \cdot d) \mod ((p-1) \cdot (q-1)) = 1$.

Открытым ключом являются e и n, а закрытым – d и n.

Для шифрования данных необходимо:

1) разбить текст на блоки, каждый из которых может быть представлен в виде числа $M(i) = 0, 1, 2, ..., n-1$;

2) зашифровать блоки по формуле $C(i) = M(i)^e \mod n$.

Расшифрованное сообщение будет получаться по формуле $M(i) = C(i)^d \mod n$.

2.1.2 DSA (алгоритм цифровой подписи)

Этот алгоритм был утверждён как стандарт цифровой подписи Национальным институтом стандартов и технологий (NIST) в 1991 году. DSA, являющийся алгоритмом с открытым ключом, как и RSA, не использует закрытые ключи для шифрования и открытые — для расшифровки, в отличие от RSA. Для создания цифровой подписи, которая представляет собой 160-битное число, применяется уникальная математическая операция. После того как создаётся хэш сообщения, оно шифруется с использованием закрытого ключа отправителя, что формирует цифровую подпись. При использовании открытого ключа, который связан с закрытым, подтверждается, что сообщение поступило от правильного отправителя [2]. На рисунке 2.2 представлена иллюстрация работы алгоритма DSA.

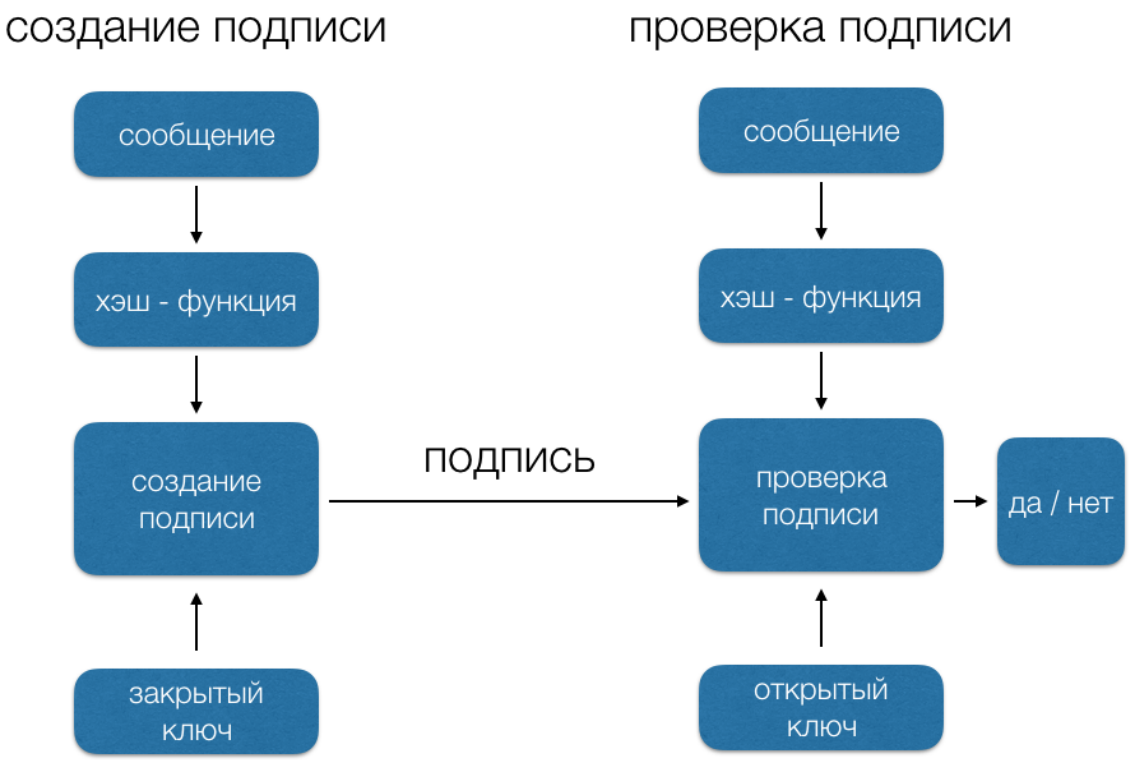

Рисунок 2.2 — Иллюстрация работы DSA

Алгоритм генерации ключа включает в себя следующие шаги:

1) выбирается простое число q, такое, что $2^{159} < q < 2^{160}$;

2) выбирается t, такое что $0 \leq t \leq 8$ и выбирается простое число p так, что $2^{511+64t} < p < 2^{512+64t}$, причём q должно делиться на $(p-1)$;

3) находится производящий элемент α для циклической группы Z_p порядка q;

4) выбирается случайное целое a, такое что $1 \leq a \leq q-1$;

5) вычисляется $y = a^{\alpha} \mod p$.

Секретным ключом является a, а открытым ключом – (p, q, α, y)

Имеется сообщение m. Подпись сообщения секретным ключом выглядит следующим образом:

1) выбирается случайное секретное число k, $0 < k < q$ (разовый секретный ключ);

2) вычисляется $r = (\alpha^k \mod p) \mod q$;

3) вычисляется $k^{-1} \mod q$;

4) вычисляется $s = k^{-1}(h(m) + \alpha \cdot r) \mod q$, где h(m) – значение хэш-функции от сообщения m.

Подписью для сообщения m является пара (r, s).

Имеется открытый ключ (p, q, α, y), сообщение m, подпись сообщения (r, s). Алгоритм проверки подписи:

1) проверить, что $0 < r < q$ и $0 < s < q$. Если это не так, отвергнуть подпись;

2) вычислить $w = s^{-1} \mod q$ и $h(m)$;

3) вычислить $u_1 = wh(m) \mod q$ и $u_2 = rw \mod q$;

4) вычислить $v = (\alpha^{u_1} \cdot y^{u_2} \mod p) \mod q$;

5) подпись верна, только если $v = r$.

2.1.3 ECDSA

Алгоритм, основанный на сложности вычисления дискретного логарифма в группе точек эллиптической кривой. Эллиптическая кривая в ECDSA – это линия на плоскости, задаваемая уравнением $y^2 = x^3 + a \cdot x + b$, где a и b – такие числа, что $4 \cdot a^3 + 27 \cdot b^2 \neq 0$. Например, $Bitcoin$ и $Ethereum$ используют кривую $y^2 = x^3 + 7$ (Рисунок 2.3).

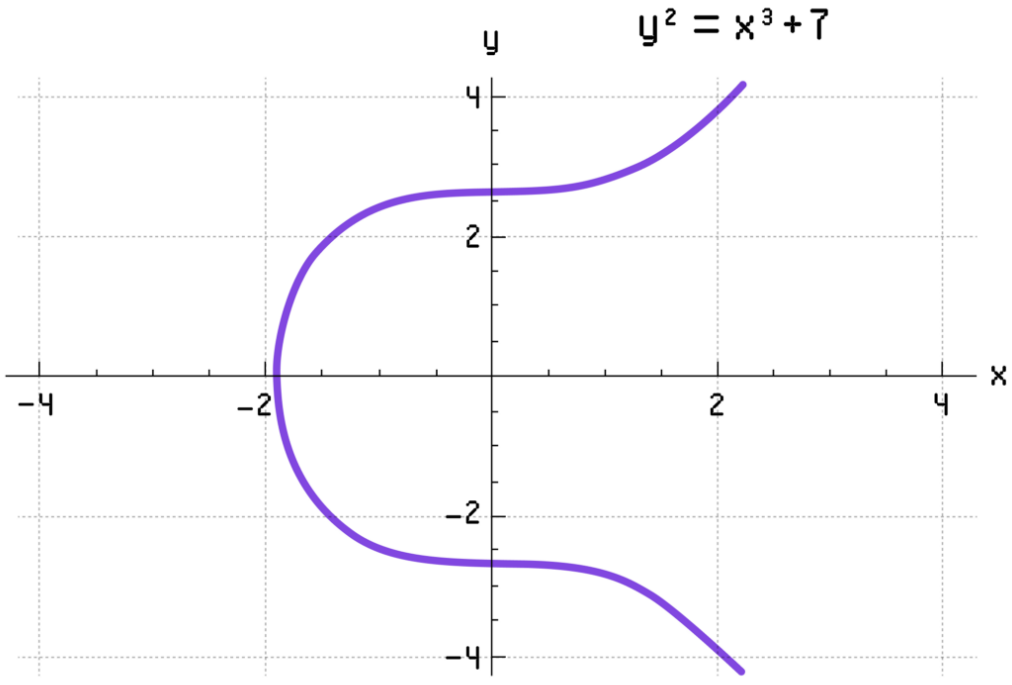

Рисунок 2.3 — Эллиптическая кривая

Главная особенность эллиптической кривой заключается в том, что её точки можно по

особому правилу умножать на целые положительные числа. В результате точка перемещается определенным образом.

Если G – точка на кривой, а k – число, то для новой точки $k \cdot G$ есть три варианта расположения:

1) $k \cdot G = G$, то есть умножение на k ничего не делает и точка $k \cdot G$ совпадает с G. Например, умножение на единицу всегда оставляет точку на месте: $1 \cdot G = G$ для любой G. Тут работает аналогия с обычными числами – умножение на 1 не меняет исходного числа. Так же и с точкой – её положение на кривой не меняется;

2) $k \cdot G$ не совпадает с G, но при этом все равно лежит на кривой. То есть в результате умножения на q точка как-то скользит вдоль кривой. На рисунке 2.4 приведена схема перемещений конкретной точки P по кривой $y^2 = x^3 + 7$ при умножении на числа от 1 до 7.

3) Точки $k \cdot G$ не существует. На практике это означает, что в формулах для подсчёта её координат встретилось деление на 0. В этом случае используется специальная терминология: говорят, что $k \cdot G$ – это бесконечно удалённая точка и пишут $k \cdot G = 0$.

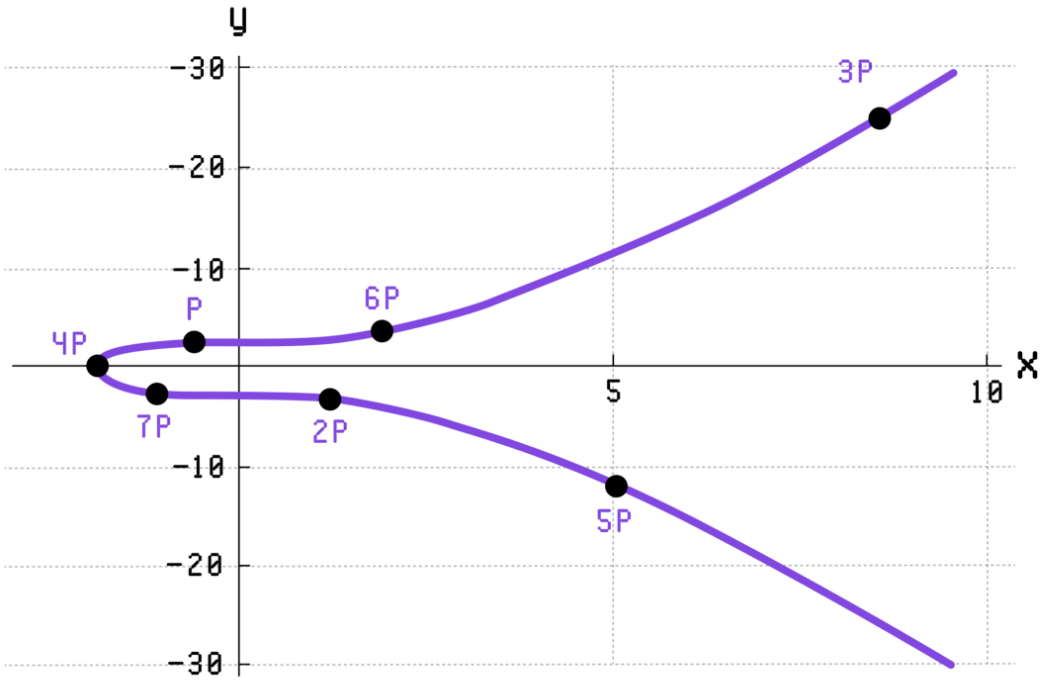

Рисунок 2.4 — Движение точки по эллиптической кривой

Алгоритм ECDSA предполагает, что в системе заранее выбрано пять параметров, одинаковых для всех пользователей. Два из них относятся к модульной арифметике в ECDSA. Остальные три:

1) сама эллиптическая кривая. Чтобы определить кривую, достаточно просто выбрать два числа a и b из её уравнения $y^2 = x^3 + a \cdot x + b$, $4 \cdot a^3 + 27 \cdot b^2 \neq 0$;

2) простое число n;

3) такая точка G на кривой, что n является порядком G. Точка G называется базовой точкой.

Секретный ключ sk выбирается случайным образом от 1 до n-1. В качестве публичного ключа pk берётся точка $pk = sk \cdot G$.

Здесь в силу вступает ещё одно ключевое свойство операции умножения точки на число. Пусть даны точки G и $d \cdot G$, где d неизвестно. Тогда оказывается, что найти d — это неразрешимая с вычислительной точки зрения задача. Этот факт называется проблемой дискретного логарифмирования для эллиптических кривых. Отсюда следует, что, зная базовую точку G и публичный ключ pk, невозможно найти соответствующий секретный ключ sk. На выходе алгоритм выдаёт пару sk, pk, где sk – число, а pk – точка, то есть пара чисел.

Алгоритм генерации подписи Sig принимает на вход приватный ключ sk и число m — хеш подписываемого сообщения. А на выходе в качестве подписи выдаётся не одно число, а два: $(r, s) = Sig(sk, m)$. Это связано с тем, что внутри самого алгоритма присутствует выбор дополнительного случайного числа k, влияющего на вид подписи. Этот случайный параметр нужен для того, чтобы обеспечить секретность ключу sk. Если подпись полностью определятся лишь входными параметрами sk и m, то секретный ключ sk может быть вычислен на основе всего лишь двух оставленных с его помощью подписей. Наличие случайного параметра приводит к тому, что на одних и тех же входных значениях алгоритм Sig может выдавать разный результат. Поэтому, чтобы подпись можно было проверить, необходимо поделиться некоторой информацией о параметре k. Первая компонента подписи r как раз содержит в себе эту информацию. Число r полностью определяется числом k, при его подсчёте sk и m не используются. В то же время в вычислениях числа s участвуют уже все три параметра k, sk и m.

Алгоритм проверки подписи Ver принимает на вход публичный ключ pk, хеш сообщения m и подпись (r, s). На основе подписи и хеша этот алгоритм находит два числа u_1 и u_2. Далее рассматриваются две точки на эллиптической кривой: $u_1 \cdot G$ и $u_2 \cdot pk$ (pk — точка, значит, $u_2 \cdot pk$ — тоже точка). Если подпись s была действительно посчитана с помощью ключа sk, соответствующего ключу pk, и сообщение действительно не менялось с момента создания подписи, то точки $u_1 \cdot G$ и $u_2 \cdot pk$ будут находится относительно друг друга в некотором специальном положении. Алгоритм Ver смотрит, так это или нет. Если расположение точек относительно друг друга и правда такое, какое должно быть, то подпись считается корректной, и Ver возвращает 1 (TRUE). В противном случае возвращается 0 (FALSE). [5]

2.2 Критерии сравнения алгоритмов

В таблице 2.1 приведены критерии, которые будут использоваться для сравнительного анализа существующих методов асимметричного шифрования.

Таблица 2.1 — Критерии сравнения алгоритмов асимметричного шифрования

Критерий	Описание
Безопасность	Сложность криптоанализа: Оценка того, насколько устойчив алгоритм к попыткам взлома, включая анализ стойкости против атак, таких как атаки с использованием вычислительных мощностей, и атаки на основе математических принципов. Размер ключа: Чем больше длина ключа, тем сложнее взломать алгоритм, но это также может повлиять на производительность.
Производительность	Скорость шифрования/дешифрования: Важный фактор для практического применения, особенно в системах с ограниченными ресурсами (например, мобильных устройствах или IoT). Эффективность вычислений: Требования к вычислительным мощностям (процессору и памяти) для выполнения операций шифрования и дешифрования.

2.3 Сравнительный анализ алгоритмов

В таблице 2.2 представлены результаты сравнительного анализа. Цифра в ячейке обозначает преимущество перед другими алгоритмами: 3 – алгоритм наиболее эффективен по данному критерию, 1 – алгоритм наименее эффективен по данному критерию (среди представленных).

Таблица 2.2 — Результаты сравнительного анализа

Критерий	RSA	DSA	ECDSA
Безопасность (Сложность криптоанализа)	1	2	3
Безопасность (Размер ключа)	2	1	3
Производительность (Скорость шифрования/дешифрования)	2	3	1
Производительность (Эффективность вычислений)	1	3	2

ЗАКЛЮЧЕНИЕ

В ходе работы были выполнены следующие задачи:

1) ознакомление с существующими алгоритмами асимметричного шифрования данных;
2) выделены сходства и различия этих алгоритмов;
3) сформулированы критерии для сравнения алгоритмов;
4) проведён сравнительный анализ алгоритмов по сформулированным критериям.

Цель работы была достигнута.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Ассиметричное шифрование. Энциклопедия «Касперского» [Электронный ресурс]. Режим доступа: `https://encyclopedia.kaspersky.ru/glossary/asymmetric-encryption/`(Дата обращения: 10.12.2024)

2. Зацепа Ангелина Ивановна ШИФРОВАНИЕ ДАННЫХ // StudNet. 2022. №1. [Электронный ресурс] Режим доступа: https://cyberleninka.ru/article/n/shifrovanie-dannyh (дата обращения: 10.12.2024).

3. Бабенко, Л. К. Современные алгоритмы блочного шифрования и методы их анализа / Л.К. Бабенко, Е.А. Ищукова. – М.: Гелиос АРВ, 2006. – 376 с

4. Баричев, С. Г. Основы современной криптографии / С.Г. Баричев, В.В. Гончаров, Р.Е. Серов. - Москва: СИНТЕГ, 2011. – 176 с

5. Johnson, D., Menezes, A. & Vanstone, S. The Elliptic Curve Digital Signature Algorithm (ECDSA) – IJIS 1, 2001 – 36-63

Приложение А

Презентация к научно-исследовательской работе состоит из 3 слайдов.