Защита

**Слайд 1. Тема** моей выпускной квалификационной работы «Исследование методов дискретного логарифмирования».

**Слайд 2. Актуальность** исследования обуславливается тем, что задача дискретного логарифмирования является одной из основных задач, на которых базируется криптография с открытым ключом. Классическими криптографическими схемами на её основе являются схема выработки общего ключа Диффи-Хеллмана, схема электронной подписи Эль-Гамаля, криптосистема Мэсси-Омуры для передачи сообщений.

**Слайд 3. Цель данной работы** – исследование и разработка алгоритмов дискретного логарифмирования с экспоненциальной и субэкспоненциальной сложностью.

Для достижения цели был поставлен ряд **задач**, из которых можно выделить следующие:

- реализовать вспомогательные математические функции для проверки алгоритмов дискретного логарифмирования,

- исследовать и реализовать базовые и модифицированные алгоритмы дискретного логарифмирования,

- провести эксперименты и сравнительный анализ на реализованных базовых и модифицированных методах дискретного логарифмирования.

**Слайд 4. Используемыми технологиями** являются C# на .NET8 в Windows Forms.

**Слайд 5.** Для исследования рабочим **определением** послужило определение дискретного логарифмирования, которое звучит следующим образом: «Дискретное логарифмирование является задачей обращения функции в некоторой конечной мультипликативной группе ». Пусть в некоторой конечной мультипликативной абелевой группе задано уравнение:

Решение задачи дискретного логарифмирования состоит в нахождении некоторого целого неотрицательного числа , удовлетворяющего уравнению.

**Слайд 6.** В данной работе исследованы алгоритмы экспоненциальной сложности, которые включают в себя алгоритмы Шенкса, Полига-Хеллмана, ро-метод Полларда, а также алгоритмы субэкспоненциальной сложности, а именно алгоритмы Адлемана, COS и решето числового поля.

**Слайд 7. Алгоритм Шенкса.** Модификация алгоритма состоит в распараллеливании 2 и 3 шага алгоритма. На 2 шаге алгоритма параллельно вычисляются два ряда чисел. На 3 шаге был сделан параллельный поиск результата с начала и с конца ряда.

(Показал лучше во всех тестах)

**Слайд 8. Алгоритм Полига-Хеллмана**. Была реализована модификация алгоритма, состоящая в том, что на 1 шаге алгоритма число было разложено на простые множители и данные простые множители были возведены в свои степени, чтобы на 2 шаге была составлена таблица из единичных значений без степеней.

(Лучше в скорости при 16 и 8, лучше при 32 и 8, лучше при 32 и 16)

**Слайд 9. Ро-метод Полларда.** Была реализована модификация алгоритма, состоящая в том, что на 4 шаге алгоритма увеличилась степень вычисляемого . При вычислении степень полинома увеличилась до 3.

(Лучше в скорости при 64 и 128 битах)

**Слайд 10. Алгоритм Адлемана.** Была реализована модификация алгоритма, состоящая в том, что на 1 шаге алгоритма был изменён показатель степени при вычислении числа , тем самым повысив факторную базу.

(из-за повышения факторной базы увеличилась размерность СЛАУ, поэтому проигрыш в затраченной скорости и памяти)

**Слайд 11. Алгоритм COS.** Была реализована модификация алгоритма, состоящая в том, что на 2 шаге был увеличен наименьший вычет, добавив значение , чтобы увеличить разложение чисел при формировании СЛАУ.

(из-за повышения факторной базы увеличилась размерность СЛАУ, поэтому проигрыш в затраченной скорости и памяти)

**Слайд 12. Решето числового поля.** Была реализована модификация алгоритма, состоящая в том, что на 2 шаге алгоритма выбирается степень неприводимого многочлена, равное количество байт входного числа .

(Лучше затраченная память при 64 битах.)

**Слайд 13. В результате эксперимента можно сделать вывод о том, что** модифицированные экспоненциальные алгоритмы при определённых параметрах имеют лучшие показатели в скорости выполнения и затраченной памяти, в то время как модифицированные субэкспоненциальные алгоритмы не показали прирост эффективности.

**Слайд 14.** Спасибо за внимание. Я готов ответить на ваши вопросы.