Презентація до дипломної роботи на тему: «Аналіз алгоритмів і реалізацій цифрового підпису на еліптичних кривих Едвардса».

Виконав: Калінічев Гліб Олександрович, МП ІПЗ-2

Керівник: Олійник Богдана Віталіївна

Актуальність теми:

- **EdDSA (Edwards Digital Signature Algorithm)** - один з найсучасніших алгоритмів цифрового підпису.
- Вперше представлений в 2011 році у статті «High speed high security signatures» Даніелем Бернштейном і Танією Ланге.
- Потенційна заміна RSA, DSA.
- Висока швидкість підпису та верифікації.
- Малий розмір ключів та підписів.
- Стійкий до криптографічних атак.
- Сфери застосування: IoT, TLS, SSH, OS, Web і т.д.

Об’єкт дослідження:

- Варіації EdDSA: **Ed25519** та **Ed448**.
- Використовують еліптичні криві Едвардса: **Edwards25519**, **Edwards448** (Goldilocks Curve).

Крива **Edwards25519**:

- Рівняння: $-x^2 + y^2 = 1 - (121665/121666) \cdot x^2 \cdot y^2$
- Поле лишків за модулем: $p = 2^{255} - 19$

Крива **Edwards448**:

- Рівняння: $x^2 + y^2 = 1 - 39081 \cdot x^2 \cdot y^2$
- Поле лишків за модулем: $p = 2^{448} - 2^{224} - 1$

Проблематика:

- Існуючі реалізації Ed25519 та Ed448 з пакету java.security, що є частиною JDK, та криптографічної бібліотеки Bouncy Castle використовують стандартний підхід до реалізації EdDSA, описаний в специфікації RFC-8032, створеною Internet Research Task Force (IRTF).
- Існують теоретичні способи пришвидшення роботи EdDSA.
- Відносно невелика кількість pure-Java реалізацій Ed25519 та Ed448.

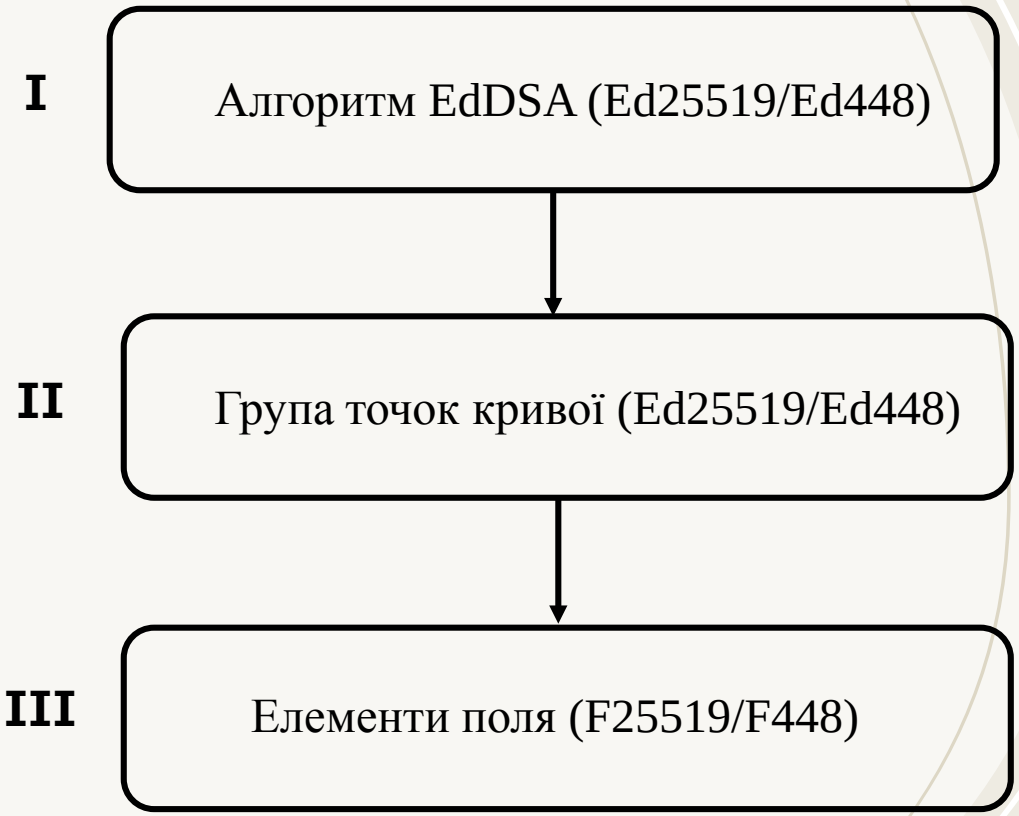
Мета роботи:

- Створення програмних реалізацій алгоритмів Ed25519 та Ed448, швидкість роботи яких перевищувала б швидкість роботи відповідних стандартних реалізацій в мові програмування Java. Визначення переваг і недоліків власних реалізацій порівняно з відповідними стандартними реалізаціями.

Рівні абстракції EdDSA:

Що було зроблено?

- Оптимізації на кожному рівні абстракції.



Оптимізації III рівня (операції в полі лишків):

- Представлення 255-бітного числа у вигляді 10 слотів по 25 або 26 біт кожний. (limbs формат)

- Швидкі операції множення та піднесення до квадрату елементів поля F25519.

- Швидка операція знаходження мультиплікативного оберненого елемента до елемента поля (modular inverse: маючи в полі елемент a знайти a^{-1}): використовується спеціально збудований для поля F25519 ланцюг множень та піднесень до квадрату для отримання $a^{2^{255-21}}$. (використання Малої теореми Ферма).

- Швидка операція знаходження кореня квадратного в полі F22519: використовується ланцюг множень та піднесень до квадрату для отримання $a^{2^{252-2}}$.

Оптимізації II рівня (операції в групі точок кривої):

- Використання «швидких» формул додавання та подвоєння точок кривих Ed25519 та Ed448. Дані формули представлені в статтях Hisil, H., Wong, K.K.-H., Carter, G., Dawson, E.: Twisted Edwards Curves Revisited та Bernstein, D.J., Lange, T.: Faster Addition and Doubling on Elliptic Curves.

- Використання алгоритму скалярного множення точки з використанням wNAF представлення скаляру.

- Використання кешу попередньо обчислених точок для публічних ключів.

Оптимізації І рівня (алгоритми Ed25519/Ed448):

- Паралелізація незалежних обчислень в операціях підпису та верифікації.
- Використання кешу публічних ключів для уникнення виконання надлишкових операцій скалярного множення точок.

Реалізації, що порівнювались:

- StandardEd25519 - власна примітивна реалізація;
- Ed25519Affine - власна примітивна реалізація з використанням «швидкого» аналогу BigInteger;
- Ed25519ExtHom – власна не паралельна реалізація з усіма оптимізаціями;
- ParallelEd25519 – власна паралельна реалізація з усіма оптимізаціями;
- StandardEd448;
- Ed448Affine;
- Ed448ExtHom;
- Ed448Parallel;
- BouncyCastleEd25519 – реалізація ed25519 з бібліотеки Bouncy Castle;
- BouncyCastleEd448;
- JDKEd25519 – реалізація ed25519 з пакету java.security;
- JDKEd448

Методика порівняння:

- Тестові дані - згенерований текст, розміром 5000 слів.
- Кожне слово окремо підписується, а потім відповідний підпис верифікується.
- Вимірюється загальний час підпису і верифікації для кожної реалізації.
- Обраховується середній час підпису і верифікації для кожної реалізації.
- Всі реалізації використовують однакові приватні ключі для підпису.

Результати порівняння реалізацій Ed25519:

Назва реалізації	**Середній час підпису, мс.**	**Середній час верифікації, мс.**
BouncyCastleEd25519	0,2569	0,2487
JDKEd25519	0,9563	0,9425
ParallelEd25519	0,2983	0,1982
Ed25519ExtHom	0,2590	0,2864
Ed25519Affine	0,9721	0,9685
StandardEd25519	2,1035	2,0956

Результати порівняння реалізацій Ed448:

Назва реалізації	**Середній час підпису, мс.**	**Середній час верифікації, мс.**
BouncyCastleEd448	0,4562	0,4682
JDKEd448	2,3866	2,3521
ParallelEd448	0,9851	1,0035
Ed448Hom	0,8969	1,5524
Ed448Affine	2,4782	2,9658
StandardEd448	4,5962	5,0148

Висновок:

- Досліджено способи оптимізації алгоритмів Ed25519 та Ed448 на трьох рівнях абстракції.

- Створено низку власних реалізацій Ed25519 та Ed448: StandardEd25519, StandardEd448, Ed25519Affine, Ed448Affine, Ed25519ExtHom, Ed448ExtHom, ParallelEd25519, ParallelEd448.

- Проведено порівняльний аналіз власних реалізацій Ed25519 та Ed448 з відповідними реалізаціями в стандартному пакеті java.security та бібліотеці Bouncy Castle. За результатами аналізу встановлено, що серед реалізацій Ed25519 найшвидше верифікацію виконує ParallelEd25519, а підпис – BouncyCastleEd25519 та Ed25519ExtHom. Серед реалізацій Ed448 найшвидшою виявилась BouncyCastleEd448. Всі власні реалізації, що використовують розглянуті оптимізації, працюють значно швидше за стандартні реалізації з пакету java.security.

Дякую за увагу!