Otwartoźródłowa biblioteka JCMathLib

SYSTEMY AUTOMATYCZNEJ IDENTYFIKACJI

Olga Kryspin

Plan

- O JCMathLib – czym jest i jakie daje możliwości?
- Kryptografia krzywych eliptycznych (ECC)
- Implementacja – główne komponenty
- JavaCard a JCMathLib – funkcje kryptograficzne
- Wydajność
- Optymalizacje
- JCProfiler
- Bezpieczeństwo
- Ograniczenia
- Podsumowanie

OPENCRYPTO PROJECT

We ❤ JavaCard and we made it truly open

Vasilios Mavroudis | Petr Svenda

What's in? 🎁

JavaCard is an amazing platform. Unfortunately, it lacks certain essential features which makes development troublesome and limiting. The OpencryptoJC project releases tools that aim to aid both experienced and new developers: JCMathLib implements *Integers, Big Numbers and Elliptic Curves*, while the JCProfiler is the first *free profiler for JavaCards.*

Opencrypto is kindly supported by our awesome sponsors!

JCMathLib | Profiler

https://opencryptojc.org/

Github

- kod

https://github.com/OpenCryptoProject/JCMathLib

- dokumentacja

https://github.com/OpenCryptoProject/JCMathLib/wiki/

- JCProfiler

https://github.com/OpenCryptoProject/JCProfiler

Biblioteka JCMathLib

- biblioteka **open-source** dla platformy Java Card,
- wspiera **niskopoziomowe** operacje dodawania i mnożenia punktów na **krzywych eliptycznych**,
- projekt wspierany przez **JavaCardOS**,
- dodatkowe narzędzia do **zarządzania pamięcią współdzieloną** i **optymalizacją wydajności**. *

JCMathLib

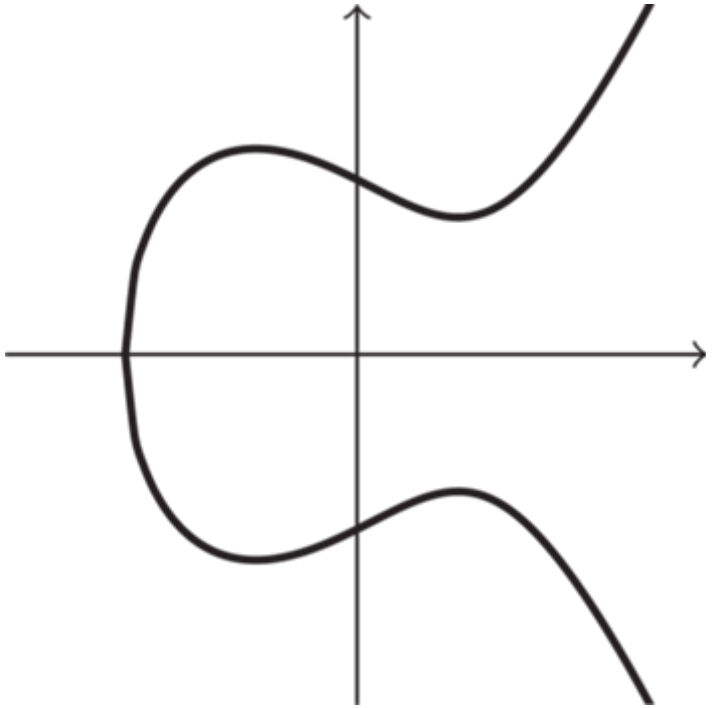
JavaCardOS

Biblioteka JCMathLib

- **JCMathLib** – biblioteka,
- **JCMathLibExamples** – klient na potrzeby testowania (po stronie PC),
- **JCMathLibTest** – klient na potrzeby testowania i pomiarów wydajności (po stronie PC).

Kryptografia krzywych eliptycznych (ECC)

- wykorzystuje matematyczne właściwości krzywych eliptycznych,
- oparta na złożoności obliczeniowej dyskretnych logarytmów na krzywych eliptycznych,
- algorytm podpisu cyfrowego krzywej eliptycznej wykorzystujący ECC to **ECDSA (*Elliptic Curve Digital Signature Algorithm*)**.



Główne komponenty (1)

- Klasa ***Integer*** – implementacja typu *int* potrafiącego przechowywać liczby 32-bitowe ze znakiem wraz z metodami dodawania, odejmowania, mnożenia, dzielenia i operacji modulo.

- Klasa ***BigNumber*** – hermetyzuje dużą wartość bez znaku o dowolnej długości i udostępnia metody wykonywania na niej operacji arytmetycznych. Zapewnia metody: dodawania, odejmowania, mnożenia, dzielenia, modulo, potęgowania i ich modularnych odpowiedników dla liczb 8-bajtowych i dłuższych.

Główne komponenty (2)

- Klasa ***ECCurve*** – przechowuje parametry danej krzywej eliptycznej i zapewnia wygodny sposób obsługi wielu krzywych w jednym aplecie. Zapewnia wywołania generowania par kluczy na określonej krzywej i jest zależnością klasy *ECPoint*.

- Klasa ***ECPoint*** – implementuje punkt krzywej eliptycznej i zapewnia metody dodawania punktów, inwersji, podwajania i mnożenia przez skalar.

JavaCard API a JCMathLib

Package	Functionality
javacardx.crypto.Cipher	Symmetric Encryption Schemes
javacard.crypto.Signature	Asymmetric Signing Schemes
javacard.security.MessageDigest	Cryptographic Hash Functions
javacard.security.RandomData	Random Number Generators
javacard.security.KeyAgreement	Key Agreement Algorithms
javacard.security.KeyBuilder	Key Generation
javacard.security.KeyPair	Key Pair Storage

Wydajność

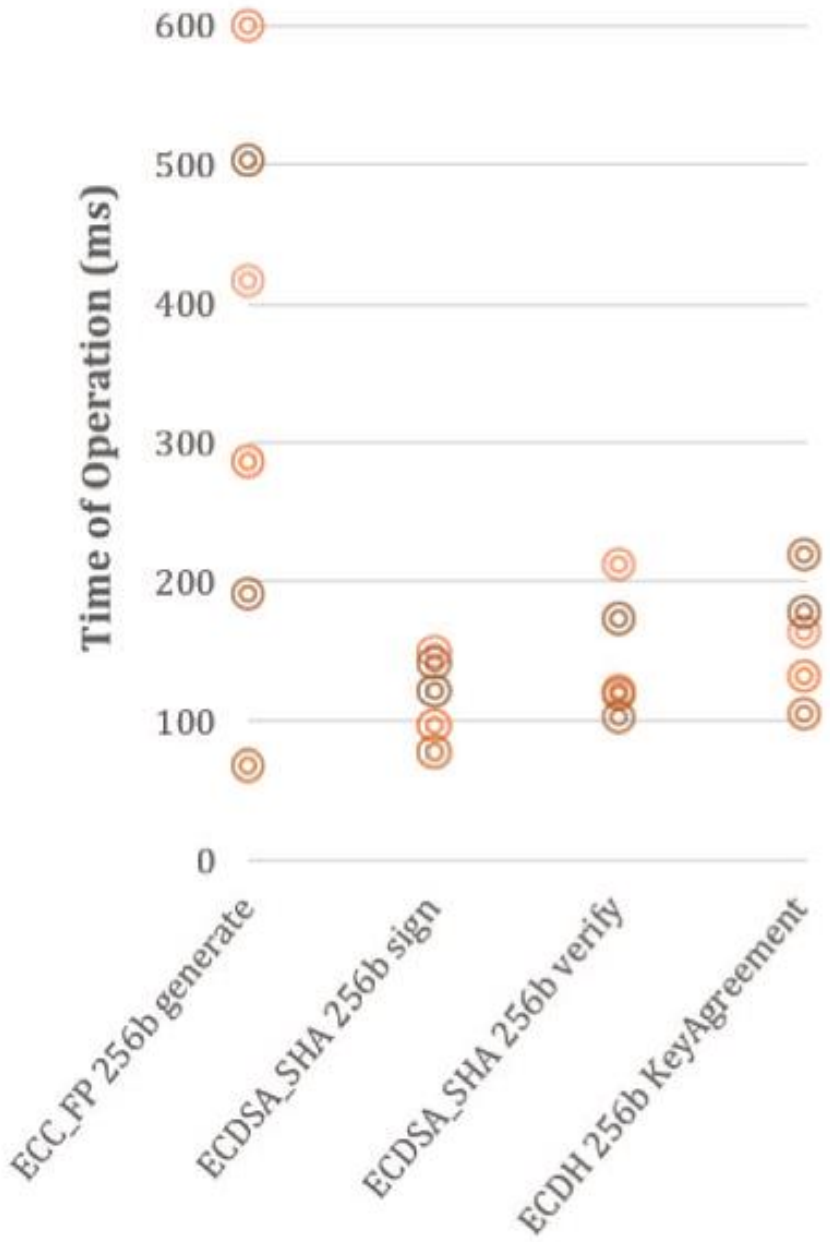
	NXP J2E081	NXP J2D081	G&D S@C 6.0
BigNumber operations			
add(256b)	7 ms	10 ms	10 ms
subtract(256b)	14 ms	22 ms	11 ms
multiplication(256b)	112 ms	113 ms	117 ms
mod(256b)	30 ms	31 ms	23 ms
mod_add(256b, 256b)	71 ms	72 ms	56 ms
mod_mult(256b, 256b)	872 ms	855 ms	921 ms
mod_exp(2, 256b)	766 ms	697 ms	667 ms
ECPoint operations			
randomize(256b)	296 ms	245 ms	503 ms
add(256b)	2995 ms	2892 ms	2747 ms
inversion(256b)	112 ms	109 ms	94 ms
multiplication(256b)	4157 ms	3981 ms	3854 ms
RAM overhead			
JCMathLib(256b)	1144 B	1152 B	923 B
new ECPoint(256b)	0* B	0* B	0* B
new BigNumber(256b)	32 B	32 B	32 B

Optymalizacje

Pomiar podstawowych operacji dostępnych za pośrednictwem publicznego interfejsu API JavaCard na sześciu różnych kartach elektronicznych.

Każdy punkt odpowiada pojedynczej operacji wykonywanej na określonej karcie inteligentnej.

Operacje JCMathLib składają się z takich operacji.



JCProfiler – performance profiler

- Kod źródłowy apletu jest rozszerzany o dodatkowe wiersze kodu, zwane „pułapkami wydajnościowymi” (ang. *performance trap*).
- Aplikacja testująca po stronie klienta wielokrotnie wykonuje aplet dla wszystkich możliwych identyfikatorów pułapek kontrolujących.
- Pomiary czasu są gromadzone i przetwarzane w celu oszacowania różnicy czasu między dwiema kolejnymi pułapkami.
- Planowane położenie pułapki wydajności jest definiowane przez programistę.
- Po sesji profilowania zmierzone czasy (w milisekundach) są automatycznie wstawiane do źródła apletu jako komentarze obok odpowiednich linii.

Bezpieczeństwo

- implementacje sprzętowe są bardziej odporne na ataki fizyczne niż programowe,
- zastrzeżone interfejsy API często uniemożliwiają programistom udostępnienie kodu celem przeprowadzenia audytu bezpieczeństwa,
- JCMathLib umożliwia programistom zastępowanie zastrzeżonych wywołań API, przeprowadzenie analiz dynamicznych oraz publikowanie kodu źródłowego do przeglądu.

Ograniczenia

- JCMathLib jest oparta na wersji **JavaCard 3.0.x** ,
- klasa *ECPublicKey* i silnik *KeyAgreement* obsługują parametry krzywych eliptycznych wyrażone wyłącznie **w postaci Weierstrassa***,
- silnik RSA karty musi akceptować publiczny wykładnik równy 2 w obliczeniach,
- licencja **MIT**.

* $y^2 = x^3 + Ax + B, A, B \in K$

Podsumowanie

- Dostarcza zestaw transformacji operacji, które łączą operacje kryptograficzne wysokiego poziomu w celu rekonstrukcji operacji niskopoziomowych.
- JCMathLib jest pierwszą biblioteką open source dla kart JavaCard, która realizuje podstawowe operacje na liczbach całkowitych, dużych liczbach i prymitywnych krzywych eliptycznych bez użycia jakichkolwiek zastrzeżonych API.
- Wydajność biblioteki jest szeroko optymalizowana i oceniana na różnych kryptograficznych kartach inteligentnych.
- JCMathLib wymaga tylko 1 KB pamięci RAM w wersji zoptymalizowanej pod kątem wydajności.

Dziękuję za uwagę!

Materiały źródłowe

- Projekt w serwisie GitHub https://github.com/OpenCryptoProject/JCMathLib
- Dokumentacja biblioteki https://github.com/OpenCryptoProject/JCMathLib/wiki/
- JCProfiler – kod oraz dokumentacja https://github.com/OpenCryptoProject/JCProfiler
- JCProfiler tutorial https://www.youtube.com/watch?v=4eYKty6G4fg
- V. Mavroudis and P. Svenda, „JCMathLib: Wrapper Cryptographic Library for Transparent and Certifiable JavaCard Applets," 2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), Genoa, Italy, 2020, pp. 89-96, doi: 10.1109/EuroSPW51379.2020.00022.
- V. Mavroudis, and P. Svenda. (2018). Towards Low-level Cryptographic Primitives for JavaCards
- T.Kazana, „Krzywe eliptyczne w kryptografii", sierpień 2018, strona internetowa [dostęp 18.03.2021] http://www.deltami.edu.pl/temat/matematyka/kryptologia/2018/07/22/Krzywe_eliptyczne_w_kryptografii/