COMP39/9900 Computer Science/IT Capstone Project
School of Computer Science and Engineering, UNSW

Project Number: P19

Project Title: Accelerating Cryptographic Prototyping: A Re-evaluation of the Charm Library

Project Clients: Evan Krul

Project Specializations: Software development; Computer Science and Algorithms; Security/Cyber Security.

Number of groups: 2

Background:

This project aims to enable rapid prototyping and evaluation of cryptographic schemes by updating the crypto-charm library (<https://github.com/JHUISI/charm>).

Background

Cryptographic schemes are behind every single digital interaction.

- Websites have SSL certificates issued and digitally signed by a certificate authority.

- Traffic between a browser and server is encrypted.

- Blockchains rely on signatures and hash functions at their core.

Cryptographic protocols expand on this and describe the interaction between parties.

- Browsers and clients exchange encryption keys securely.

- Users can sign on to SSH by proving they know the private key that belongs to a public key installed on the server.

There is continual research and development into these schemes. For example, recent changes to BBS signatures \[1, 2\] are being proposed to enable privacy-preserving digital driver licenses. This proposal includes a signature scheme and a protocol to present digital identities securely. (If you want to chat more about this, please reach out to me. It is very interesting.)

When a cryptographic scheme needs to be implemented, we usually have two options.

1. Implement a _production-ready_ library in a low-level language like C or Rust

2. Implement a _prototype_ library in a high-level language like Python

Option 1 is the indisputably correct option if the intent is to distribute the scheme or protocol. Implementing crypto in a low-level language is not easy and comes with many 'foot guns' that must be addressed for the implementation to be 'secure enough' for production use.

Furthermore, a proposed cryptographic scheme is not static. They must be carefully reviewed and updated (i.e., the BBS Signature Draft \[2\] has been under revision since 2022). There is no use in spending the time and effort to implement a _production_ library for a non-production-ready scheme.

This is where Option 2 plays a key role. Being able to prototype a scheme or protocol in a high-level language rapidly means that researchers can quickly publish and evaluate working implementations of their proposals without concerning themselves with the complex inner workings of programming crypto libraries.

Requirements and Scope:

The Project

In 2011, the crypto-charm library was introduced \[3\] (<https://github.com/JHUISI/charm>) to enable Python implementations of cryptographic schemes/protocols. Charm is an abstraction on top of the same low-level libraries used to implement production-ready crypto.

Unfortunately, Charm has fallen out of date, and with no replacement in sight, Charm requires serious TLC.

This project aims to create a fork of Charm that addresses its fundamental flaws and makes it a viable library researchers can use for the next decade. This is a unique opportunity to undertake a significant technical challenge – with a wide range of skills required – while making a meaningful contribution to a heavily relied-on library.

The general goals of this project are as follows:

1. Address general critical deficiencies in Charm

2. Implement a new protocol engine

3. Implement support for recent low-level crypto libraries (more recent pairing curves, etc.)

4. Update benchmarking

5. Update the toolkit with recent schemes for comparisons

6. Create updated build profiles for deployment

7. Detailed documentation (in code and library usage)

The project's scope is adaptable to the student's progress and challenges encountered during the project.

Students are encouraged to propose further improvements as they familiarise themselves with the project.

\[1\] S. Tessaro and C. Zhu, “Revisiting BBS Signatures UPDATED.” 2023. Accessed: Apr. 24, 2024. \[Online\]. Available: <https://eprint.iacr.org/2023/275>

\[2\] T. Looker, V. Kalos, A. Whitehead, and M. Lodder, “The BBS Signature Scheme,” Internet Engineering Task Force, Internet Draft draft-irtf-cfrg-bbs-signatures-05, Dec. 2023. Accessed: May 15, 2024. \[Online\]. Available: <https://datatracker.ietf.org/doc/draft-irtf-cfrg-bbs-signatures>

\[3\] J. A. Akinyele, M. D. Green, and A. D. Rubin, “Charm: A framework for Rapidly Prototyping Cryptosystems.” 2011. Accessed: May 15, 2024. \[Online\]. Available: <https://eprint.iacr.org/2011/617>

- A fork of the Charm library that addresses, at a minimum:

  - Modern building and deployment of the library

  - Swappable support of modern low-level crypto libraries

  - A protocol engine that does not rely on sockets and is amenable to test-driven-development
- Clear documentation of new and existing functionality
  - In code is a minimum requirement.
  - Further documentation is a plus.

Required Knowledge and skills:

- One group member MUST have some understanding of cryptographic schemes and protocols
  - Integer Groups
  - Elliptic Curve Groups
  - Pairing Curve Groups
  - Basic Signature and Encryption schemes
- OR One group member MUST have taken:
  - MATH1081 (Discrete Mathematics)
  - MATH 2400 or MATH3411
  - or equivalents
- Strong knowledge of Python
- Strong knowledge of C
- Ability to read and implement academic papers describing a scheme or protocol

Expected outcomes/deliverables:

The expected outcome is a robust crypto-prototyping library for the next decade.

- Source code
- Documentation

Supervision:

Evan Krul

Additional resources: