Cryptographic Protocol Simulator

Mentee: Edward Kim
Mentor: Vincent Bindshaedler

The Problem

- No efficient and easy way to simulate arbitrary cryptographic protocols
- Infeasible, undesirable, and inefficient to build full systems to evaluate complex protocols
- However, simulations are crucial to the development of new systems and improvement of old ones

Goals

- Design a easy-to-use framework to simulate any arbitrary cryptographic protocols
- Provide basic cryptographic primitives
- Easily test their performance

Progress and results

- Runtimes of most cryptographic primitives are known and verified, which means evaluation of the real system is not necessary
- Simulator framework implemented in Java
- Users define operations and their dependency relationships
- A scheduling algorithm automatically determines the fastest runtime possible
- Multi-core support, operations can run in parallel
- Generic Operation class allows custom subclassing

Example: The Millionaire Problem

Problem: Alice and Bob want to find out who is richer without revealing their bank balance

\[ \text{Alice: } K, K', a \]
\[ \text{Bob: } b \]
\[ E_k(a) \rightarrow E_k(r(a-b)) \]
\[ E_k(a-b) \rightarrow \text{Decipher: } r(a-b) \]
\[ \text{Sign of } r(a-b) \]

Node `alice = new Node("Alice", 2);`
Node `bob = new Node("Bob", 3);`

InputVar `aliceInput = new InputVar(a);`
HomomorphicEncrypt `encryptA = new HomomorphicEncrypt(aliceInput, K);`
DataVar `encryptA_out = encryptA.getOutput();`
// The result is sent to Bob

InputVar `bobInput = new InputVar(b);`
HomomorphicSubtract `AminusB = new HomomorphicSubtract(encryptA_out, bobInput, K);`
DataVar `AminusB_out = AminusB.getOutput();`
InputVar `randVal = new InputVar(r);`
HomomorphicMult `randMult = new HomomorphicMult(AminusB_out, randVal, K);`
DataVar `randMult_out = randMult.getOutput();`
// The result is sent back to Alice

HomomorphicDecrypt `decryptResult = new HomomorphicDecrypt(randMult_out, K');`
DataVar `decrypt_out = decryptResult.getOutput();`
// Sign of `decrypt_out` is sent to Bob, and he can determine who is richer.

Skills and future goals

- Designing a framework that will be used by others
- How a cryptographic protocol works, and its various components
- Working with levels of abstraction, and what primitives should be provided
- Create a more informative user interface for data visualization
- Make the framework easier to use
- Release the framework to the community, so that everyone can use it