ACA Project

Functional Programming and Cryptography

Prannay Khosla[1]

[1]Department of Computer Science
IIT Kanpur

Semester 2, 2016-17

Outline

Functional Programming

- Functional Programming: What and Why
- Haskell

Cryptography

- Cryptosystems

Outline

Functional vs Imperative Languages

- Specific instruction
- Abstraction constraints/freedom
- Implementation vs Overview
- Mathematical extensions

Examples

- Haskell
- Scheme (LISP)
- Erlang

Outline

Functional Programming with Haskell

Strong Recursion

Strongly typed

Type systems and Monads

Strong packages

GHC and it's benefits

Advantages over other languages

Faster

Large Developer base

Stack development

Applicative, not academic

Implementation allow concurrency and parallelism

Outline

Secret Key Cryptography

AES/DES

Symmetric key cryptographic systems which encrypt system in blocks. Widely used and theoretically broken, but strong enough for practical uses.

XOR based

Weak cryptosystems where you do a bit wise XOR of data using a defined key. This is again symmetric.

Public Key Cryptography

- RSA based cryptosystems
- SHA1 : broken thoeretically
- SHA2, SHA3 : unbroken

Elliptic Curves

- Similar to what RSA is based on
- Operation over Elliptic Curves over Finite Fields ($\mathbb{F}_{p^k}$).
- Very hard to break since involves breaking discrete logarithm problem

Summary

What you will learn

- Basics of functional programming in the form of Haskell
- Introduction to type systems and Monads and high level programming language constructs.
- Implementing basic cryptosystems, but not more than that. You will learn how to optimize them using GHC and Haskell.
- Linux, basic Command Line, basic of systems,
- Future
  - Haskell based Crypto libraries and optimization for same.
  - Programming Languages and Verification
  - Monadic approaches to programming (it is the classes of the 21st Century).