A Project in Modern Cryptography

Homomorphic Encryption Systems

Philip M. Robinson

August 24, 2012

Presentation Overview

- Personal Background
- Introduction to Homomorphic Encryption
- Project
  - Requirements
  - Construction
  - Accomplishments
- Next Steps
- Acknowledgments

Personal Background for this Project

- BS in Computer Science from Western Washington University
- Cryptography and Security Hobbyist
  - Kryptos
  - CCDC
- Independent Study
  - Parallel Elliptic Curve Cryptography

Why I chose this Project for this Presentation

- Wide breadth of topics
- Good team
- New technology
- Difficult project

Definitions

- Encryption - process of hiding information in a recoverable manner
- Ciphertext - encrypted text
- Homomorphism $f(a \star b) = f(a) \star f(b)$
  - Partially - allows some operations
  - Somewhat - allows all operations but has limits
  - Fully - allows all operations
- Noise - distance of data from desired origin
- Depth of circuit - how much accumulated noise a circuit produces

Introduction

- What is Fully Homomorphic Encryption (FHE)

  FHE allows arbitrary operations to be performed over encrypted data without decryption or loss of security

$$Dec(Enc(f(b_i \dots b_j))) \approx Dec(f(Enc(b_i) \dots Enc(b_j)))$$

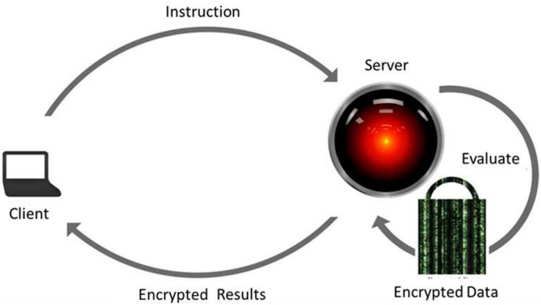
- Brief history
  - (Unpadded RSA) This problem was posed by Rivest 1978

$$\begin{aligned} c_1 &= m_1^e \text{mod } N \\ c_2 &= m_2^e \text{mod } N \\ (c_1 \cdot c_2) &= (m_1 \cdot m_2)^e \text{mod } N \end{aligned}$$

  - Craig Gentry 2009 (FHE)
  - Dijk Gentry Halevi Vaikuntanathan (DGHV) cipher 2010

Applications and Why this is Cool

- Ideally you can offload sensitive computation
- You can trust anyone - Mail Server
- You can trust everyone - Distributed Nodes
    - SETI@home
    - virtual supercomputer composed of large numbers of internet-connected nodes



Project Launch

- Recruited Adviser
- Development of Course
  - Syllabus
  - Schedule
- Recruited team members from competitions

Project Goals

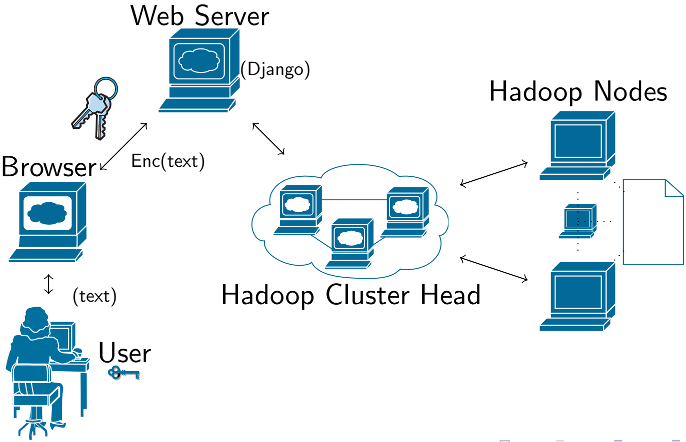
- Proof of Concept
- Product
  - Functional
  - Fully documented
- First Uniform and Accessible Implementation

Concerns Addressed

- Known
  - Steep learning curve
  - New technology
  - Time (9 weeks to completion)
- Unknown
  - Branching
  - Data bloat - to be addressed later
  - Time/Space complexity of 'basic' operations
  - Network Bandwidth

Project Requirements

- Public key (Fully Homomorphic Encryption) FHE
- Website for computing word counts WLS
- Data shipping and setup will not be part of website
- Poster and paper



Steps for Fully Homomorphic System

1. Build a Somewhat Homomorphic System
   - Bit-wise encryption rather than block-wise
   - Well defined limits wrt. noise (n)
2. Bootstrapping the Somewhat Homomorphic System
   - *Size* of Decryption Circuit
   - Circular Security
   - $Dec_0(Enc_1(Enc_0(b_i)), Enc_1(p)) = Enc_1(b_i)$
3. Squashing the Recrypt Circuit
   - Asymmetric Keys
   - $Rec\Big(Enc(pk, b_i) + \text{noise}, Enc(pk, sk)\Big) = Enc(pk, b_i) + 0$
4. Distribute computing for space and time restrictions

How Somewhat Homomorphic System Works I

λ : Security Parameter is analogous to the bit-width of AES

b_i : denotes the i^{th} 'bit' of data

c_i : denotes the cipher-text of b_i

n_i : denotes a λ-bit random number corresponding to c_i
behaves as a 'noise' characteristic

q_i : denotes a λ^5-bit random number corresponding to c_i
works as displacement of data

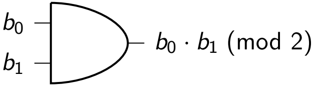
p : denotes a λ^2-bit random *odd* number (private key)

$$Enc(b_i) = c_i = p \cdot q_i + 2 \cdot n_i + b_i$$

$$Dec(c_i) = b_i = [p \cdot q_i + 2 \cdot n_i + b_i]_p \,(mod\ 2)$$

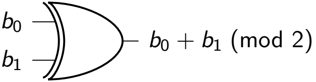
How Somewhat Homomorphic System Works II

AND



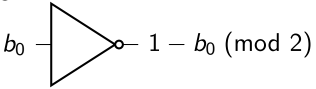
$$\begin{aligned} c_i \cdot c_j &= (p \cdot q_i + 2 \cdot n_i + b_i) \\ &\quad \cdot (p \cdot q_j + 2 \cdot n_j + b_j) \\ &= p \cdot \hat{q} + 2 \cdot \hat{n} + (b_i \cdot b_j) \end{aligned}$$

XOR



$$\begin{aligned} c_i + c_j &= (p \cdot q_i + 2 \cdot n_i + b_i) \\ &\quad + (p \cdot q_j + 2 \cdot n_j + b_j) \\ &= p \cdot \hat{q} + 2 \cdot \hat{n} + (b_i + b_j) \end{aligned}$$

NOT



$$\begin{aligned} 1 - c_i &= (p \cdot q_i + 2 \cdot n_i + b_i) - 1 \\ &= p \cdot (-q_i) + 2 \cdot (-n_i) + (1 - b_i) \end{aligned}$$

How Somewhat Homomorphic System Works III

A look at 'AND' wrt. noise (n)

$$
\begin{aligned}
c_i \cdot c_j =& (p \cdot q_i + 2 \cdot n_i + b_i) \cdot (p \cdot q_j + 2 \cdot n_j + b_j) \\
=& p \cdot (p \cdot q_i \cdot q_j + 2 \cdot q_i \cdot n_j + 2 \cdot q_j \cdot n_i + q_i \cdot b_j + q_j \cdot b_i) \\
& + 2 \cdot (2 \cdot n_i \cdot n_j + n_i \cdot b_j + n_j \cdot b_i) \\
& + (b_i \cdot b_j) \\
=& p \cdot \hat{q} + 2 \cdot \hat{n} + (b_i \cdot b_j)
\end{aligned}
$$

- As long as $\hat{n} < \frac{p}{2}$, then we can still correctly decrypt

$$
\left.
\begin{aligned}
\underset{\text{sizeOf}(\hat{n})}{\text{AND}} &\approx 2 \cdot \text{Max}(\text{sizeOf}(n_i), \text{sizeOf}(n_j)) + 3 \\
\underset{\text{sizeOf}(\hat{n})}{\text{XOR}} &\approx \text{Max}(\text{sizeOf}(n_i), \text{sizeOf}(n_j)) + 1
\end{aligned}
\right\} < \left(\text{sizeOf}\left(\frac{p}{2}\right) = \lambda^2 - 1\right)
$$

Asymmetric Keys

Asymmetric keys allow us to encrypt information using *public keys*, and decrypt using a *private key*.

- The public keys are a set of encrypted zeros with a few special properties
- The private key is the same
- Bootstrappable

$$Dec_0(Enc_1(Enc_0(b_i)), Enc_1(p)) = Enc_1(b_i)$$

- Recrypt

$$Rec\Big(Enc(pk, b_i) + \text{noise}, Enc(pk, sk)\Big) = Enc(pk, b_i) + 0$$

Recrypt

$$Dec(c_i) = b_i = [p \cdot q_i + 2 \cdot n_i + b_i]_p \, (mod\ 2)$$

- Modulo and division circuits both use *division algorithm* which is too deep a circuit
- *Squashing* Circuit

$$Dec(c_i) = \text{LSB}\,(c_i) \oplus \text{LSB}\left(\left\lfloor \frac{c_i}{\hat{p}} \right\rceil\right)$$

- Subset Sum Problem
  - $\hat{p}$ ends up being a vector of rationals w/ subset $= \frac{1}{p}$
  - $\hat{sk}$ is now a encrypted 0,1 vector corresponding to hamming weight of $\hat{p}$

Voluminous Data Challenges

- Data Bloat (decided to explore Hadoop)
  - Time consequences for logic gates
  - "Sage-cat" experiment (23 min)
    - Fast multiplication algorithm
    - Considered PyCuda
  - We spill over to Swap
- PyPy
- Memory Management in Python

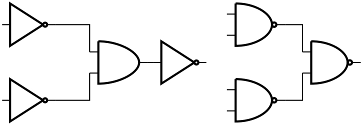
	bit-size	volume	$\lambda = 8$	$\lambda = 16$	$\lambda = 64$
bidgit	1	$\approx \lambda^7$	$\approx$ 256 KB	32 MB	512 GB
"cadadr"	6 Bytes		$\approx$ 12 MB	1.5 GB	24 TB
Dec. of Ind.	8.5 KB		$\approx$ 17 GB	2.1 TB	34 PB

Hadoop

- Hadoop
  "The Apache Hadoop project develops open-source software for reliable, scalable, distributed computing."
  - Map
    - Use of logic gates to implement strcmp
  - Reduce
    - Fixed width binary adder
- Hadoop FS data layout
  - Encrypted and split document
  - Fixed character overlap between adjacent nodes
- Circuit for word count w/ out branching

Optimization Implementation

- Construction of fundamental gates to reduce noise



- Multiplexers use binary reduction
- Complex circuits used in final product are fixed width
- True zero is used instead of encrypted zero for any possible locations
  - Results in trouble w/ operations for negative numbers
  - Allowed for *Jitter Multiplication*
- Tail recursion and loops exclusively
- For testing, we had to identify which parameters could be modified

Accomplishments

- Highly readable code and comments
  - 2600 lines Python
  - Comments have citations to papers and page numbers
- Public key system nearly complete
  - Recrypt funny and difficult to debug
- Web site sends encrypted word to Hadoop head
- Hadoop cluster setup
  - Map and reduce functions written
  - Setup scripts written for key and data distribution
- Secure systems analysis
- Small library of circuits for use in system
- Reduced 'Sage-cat' expirement to 15 min

Next Steps

- Debug and clean up remaining features
- Publish article online

- Modular reduction vs. recrypt (2010)
- Use of generators (Jean-Sebastien Coron)
- Provide code for known system attacks
- Currently binary model
  - more compact n-ary systems
  - ideals in vector spaces

- More complex user operation support
- Huffman codes

Acknowledgments

I want to express my appreciation to Dr. David Bover, for overseeing this course; and my team members, for participating in this project.

Team members include Jeremy Caci, Ali Hajy, Jonah Jolley, Clark Rinker.