Data Protection in Virtual Environments (DPRI VE)

Tom Rondeau
Program Manager, DARPA/MTO

DPRI VE will make fully homomorphic encryption practical by hardware acceleration

DPRI VE Proposers Day

03/02/2020

DARPA

Distribution Statement A. Approved for public release. Distribution unlimited.
Background:

Fully Homomorphic Encryption (FHE): a holy grail of cryptography

Current Art: information can be encrypted ONLY for transmit and storage

- Processing requires decryption
- Decryption makes data vulnerable
- Requires TRUSTED networks/compute resources

Data in uncontrolled/unsafe state

Fully Homomorphic Encryption (FHE): information can be processed while encrypted

- Processing on encrypted data
- Data ALWAYS encrypted
- Exploit UNTRUSTED network/compute resources

Data in controlled/safe state

FHE solves the nearly ubiquitous problem of protecting data at all times

Distribution Statement A. Approved for public release. Distribution unlimited.
Background:
What if a company can use your data, but never have access to it?

Fully Homomorphic Encryption makes it possible to analyze or manipulate encrypted data without revealing the data to anyone.

- Enables DoD to leverage untrusted resources e.g. cloud computing, uncontrolled networks
- Facilitates information sharing across authority boundaries without sacrificing methods
- Provides protection for commercial sector data such as medicine and finance

What type of data are they accessing?

From Wired:
Google is Slurping up Health Data - and it looks totally legal
Tech giants can access all of your personal medical details under existing health privacy laws. The question is how else that data might get used.
What are we trying to do?

Program Objective: Design and implement a hardware accelerator to reduce computational run time for FHE to make it comparable to similar unencrypted data operations (within 10x).

- Dominated by numerical transforms
- Post DARPA PROCEED Program

PROCEED: PROgramming Computation on Encrypted Data

Computational cost for training a 7-layer convolutional neural network (CNN)
What are we trying to do?

Program Objective: Design and implement a hardware accelerator to reduce computational run time for FHE to make it comparable to similar unencrypted data operations (within 10x).
• Today’s FHE algorithms are executed in software on CPUs
  • Information representation of CPUs limited to 64 bits
  • Information of FHE words can grow to 1000s of bits
• Noise accumulates in FHE operations
  • Quickly grows past CPU representation
  • Once beyond a threshold, data cannot be recovered
• Innovation: **Bootstrapping**
  • Enables continued computation
  • Cost of computation is huge

---

**Growth of noise during CNN**

**Growth of runtime for CNN**
How is it done today?

- Today’s FHE algorithms are executed in software on CPUs
  - Information representation of CPUs limited to 64 bits
  - Information of FHE words can grow to 1000s of bits
- Noise accumulates in FHE operations
  - Quickly grows past CPU representation
  - Once beyond a threshold, data cannot be recovered
- Innovation: **Bootstrapping**
  - Enables continued computation
  - Cost of computation is huge

Illustrative example of steps involved for CNN under FHE

- Growth of noise during CNN
- Growth of runtime for CNN
How is it done today?

- Today’s FHE algorithms are executed in software on CPUs
  - Information representation of CPUs limited to 64 bits
  - Information of FHE words can grow to 1000s of bits
- Noise accumulates in FHE operations
  - Quickly grows past CPU representation
  - Once beyond a threshold, data cannot be recovered
- Innovation: **Bootstrapping**
  - Enables continued computation
  - Cost of computation is huge

**Illustrative example of steps involved for CNN under FHE**

**Growth of noise during CNN**

**Growth of runtime for CNN**

**DPRI VE goal**

---

Distribution Statement A. Approved for public release. Distribution unlimited.
Key Insight: Large Arithmetic Word Size (LAWS) processing enables \(~10^4\) decrease in run-time by increasing cipher text SNR

Word size directly relates to the SNR of how a cipher text is stored and manipulated

![Graph showing computational run-time vs native word bit size]

*Simulated by leading HE SME under DARPA YFA

- FHE simulations show increasing word size increases SNR
  1. Better SNR reduces bootstrapping frequency
  2. Larger words reduces computational overhead at each step

Optimize word size for FHE, design foundational compute building blocks, implement in a realizable chip
Technical Approach: Design and implement a large word size hardware accelerator to reduce the computational runtime of FHE algorithms

Today:

- Software and Algorithms from DARPA PROCEED provides basis for today’s FHE capabilities

DPRIE:

- Low-level FHE Programming Model
  - Optimize data representation and compute for FHE

- Local Memory
  - Optimize cache/data access/latency/power for FHE

- Memory management unit

- Add, Mul, Mod, Shift, Transforms

- Optimize compute circuits for native FHE word size

Multiple potential algorithms

Challenge 3
Flexible data structures and programming models to enable parameterization of FHE algorithms

Challenge 2
LAWS memory management that provides ≥10x run-time speed up with tolerable circuit area

Challenge 1
Design an FHE hardware accelerator based on a LAWS architecture that can be formally verified within minutes to hours

A new hardware accelerator co-designed and optimized across the full stack to handle different data representations, data structures, and memory management is needed to reduce the computational burden of FHE operations

Distribution Statement A. Approved for public release. Distribution unlimited.
Challenge 1: Design an FHE hardware accelerator based on a LAWS architecture that can be formally verified within minutes to hours

What's needed: Methods to prove mathematically correct circuits to process huge amounts of information

- LAWS requires word sizes greater than 1 kb
- Tools time out and run out of memory beyond 12 bit multiplier

Time and memory usage to verify a multiplier circuit

• FHE LAWS circuits will be large
  • Optimized for latency, power, and size
  • Leads to complex and irregular structures
  • Cannot assume linear growth in verification complexity
• Cryptographic operations must be verifiably correct
  • Previous verification for multipliers quickly timeout
  • Satisfiability (SAT) solvers are exponential in time: \( O(c^n) \)
  • Gröbner basis solvers (like binary decision diagrams - BDD) are quadratic in time: \( O(n^2) \)

\( n \) is the number of bits per word and \( c \) is a constant

Distribution Statement A. Approved for public release. Distribution unlimited.
Approach: New hierarchical verification concept for large bit size operators

- New formal verification method
  - Proof of mathematical correctness
  - Symbolic computational algebra that proves lemmas about circuits
- Shown only with multipliers
- Scale approach to other circuits
  - Different challenges for adds, modulo, etc.
  - Circuit optimizations not using regular structures
  - Represent integers & float/fixed point numbers

New approach

- Initially faster and grows slower

1 year

1 month

1 day

SOA data from published Gröbner multiplication method
Symbolic method from DARPA seedling with UT Austin (PI: Warren Hunt)

New verification methods now available to implement FHE circuits in silicon in reasonable time
Challenge 2: LAWS memory management that provide $\geq 10x$ run-time speed up with tolerable circuit area

Typical 64-bit CPU architecture currently used for FHE

- Memory access should match word size
  - Current chips fetch 128 – 512 bits at a time
  - Excess latency due to multiple fetches
- Cache size management
  - Scaling to LAWS makes chips too large
  - Excess latency to move data
  - Execution models to manage simultaneous data movement and processing

What’s needed: Matching data movement and execution together for optimal performance
Approach: Structure I/O and cache sizes to FHE algorithm needs

Size of registers and cache I/O for a 4096-bit word circuit (simulation)

<table>
<thead>
<tr>
<th>parameter</th>
<th>unit</th>
<th>Word size (bits)</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exec time</td>
<td>ns</td>
<td>288</td>
<td>1.4</td>
</tr>
<tr>
<td>Energy/bit</td>
<td>nJ</td>
<td>9.7</td>
<td>1.5</td>
</tr>
</tbody>
</table>

Matching data sizes:
- Reduces # of data accesses
- Reduces cache misses

Source – Shekhar Borkar, Qualcomm

Flexible data movement to get improved performance out of cache structure

- Low-latency, energy efficient memory access
- Better memory fetch models
- Maintain performance while reducing chip area

Matching data movement and execution together provides pathway for ≥ 10x improvement in FHE run-time

Source – Univ. Michigan’s Transmuter project under SDH
Challenge 3: Flexible data structures and programming models to enable parameterization of FHE algorithms

Basic math for known FHE algorithms:

\[ c = \text{mod}_q(\text{mod}_q(p(e; h)) + \text{mod}_q(ps) + m) \]

Computation speed and security are complex combinations of the following:

- Over-optimized designs can lead to static chips
- Prevents exploration of applications and parameters
- Would limit future applicability and use by DoD
- Lack of programming model limits execution of:
  - Different data types (float vs. integer)
  - Parameter range to support classified use cases

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Purpose</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plaintext Modulus</td>
<td>Create ambiguity in crypto space</td>
<td>2</td>
<td>1024</td>
</tr>
<tr>
<td>Ciphertext Modulus</td>
<td>Smaller is better but increases frequency of bootstrapping</td>
<td>(2^{15})</td>
<td>(2^{500})</td>
</tr>
<tr>
<td>RingSize</td>
<td>Parallelism in crypto computational space</td>
<td>512</td>
<td>16384</td>
</tr>
</tbody>
</table>

What’s needed: Support future growth and algorithm exploration for many possible applications


Distribution Statement A. Approved for public release. Distribution unlimited.
Approach: Build low-level primitives and libraries for LAWS hardware

Vectorize operations based on FHE math
Map vector data structures to memory models
Historical success in optimizing for other processors
Porting to a new processor is a big effort
VOLK and Spiral shown significant benefits
Need to scale to LAWS word sizes and memory limits

Vectorize across ring size

\[
\begin{bmatrix}
    c_0 \\
    c_1 \\
     \vdots \\
    c_{n-1}
\end{bmatrix} = \text{mod}_q \left( p \cdot \text{mod}_q \left( \begin{bmatrix}
    e_0 \\
    e_1 \\
     \vdots \\
    e_{n-1}
\end{bmatrix}, \begin{bmatrix}
    h_0 \\
    h_1 \\
     \vdots \\
    h_{n-1}
\end{bmatrix} \right) + \text{mod}_q \left( p \cdot \begin{bmatrix}
    s_0 \\
    s_1 \\
     \vdots \\
    s_{n-1}
\end{bmatrix}, \begin{bmatrix}
m_0 \\
m_1 \\
     \vdots \\
m_{n-1}
\end{bmatrix} \right)
\]

Parallelize math kernels
Map algorithm transforms via Spiral-like approach

- Vectorize operations based on FHE math
- Map vector data structures to memory models
- Historical success in optimizing for other processors
  - Porting to a new processor is a big effort
  - VOLK and Spiral shown significant benefits
  - Need to scale to LAWS word sizes and memory limits

Enabling exploration of FHE parameters for many possible applications
## DPRIVE Program Metrics

<table>
<thead>
<tr>
<th>Challenge</th>
<th>Phase 1: Building Blocks in emulation</th>
<th>Phase 2: Full Design in emulation</th>
<th>Phase 3: Prototype and Software Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Ability to build and emulate all blocks (add, sub, mul, mod, shifts, transforms)</td>
<td>Binary yes/no</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Verification coverage of logic circuits</td>
<td>≥ 90%</td>
<td>100%</td>
</tr>
<tr>
<td>1</td>
<td>Time to perform logic circuit verification</td>
<td>≤ 1 day</td>
<td>≤ 1 day</td>
</tr>
<tr>
<td>2</td>
<td>Chip dimensions</td>
<td>≤ 150 mm² (estimate)</td>
<td>≤ 150 mm² (RTL)</td>
</tr>
<tr>
<td>3</td>
<td>FHE parameter range: Plaintext Modulus</td>
<td>2 – 1024</td>
<td>2 – 1024</td>
</tr>
<tr>
<td>3</td>
<td>FHE parameter range: Ciphertext Modulus</td>
<td>$2^{15} – 2^{500}$</td>
<td>$2^{15} – 2^{500}$</td>
</tr>
<tr>
<td>3</td>
<td>FHE parameter range: RingSize</td>
<td>512 – 16384</td>
<td>512 – 16384</td>
</tr>
</tbody>
</table>

**Overall**

- Execution of a 1024-point logistic regression model: ≤ 10 ms (100x) ≤ 1 ms (10x) ≤ 0.1 ms (1x)
- Execution of 7-layer CNN **inference** w/ CIFAR-10 data set per image: ≤ 10 hours (10x)
- Execution of 7-layer CNN **training** w/ CIFAR-10 data set over 10 epochs: ≤ 250 ms (100x) ≤ 25 ms (10x)

*Metrics in parenthesis indicate expected penalty vs. plaintext operations.*

**Distribution Statement A. Approved for public release. Distribution unlimited.**
## Program Schedule

<table>
<thead>
<tr>
<th>FY20</th>
<th>FY21</th>
<th>FY22</th>
<th>FY23</th>
<th>FY24</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Building Blocks</strong></td>
<td><strong>Full Design</strong></td>
<td><strong>Prototype and Software Port</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Phase 1 (15mo.)</td>
<td>Phase 2 (15mo.)</td>
<td>Phase 3 (12mo.)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FHE algorithm development</td>
<td>Full accelerator design ready for tapeout</td>
<td>Accelerator fabrication</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Design and emulate FHE building blocks</td>
<td>Fully emulate integrated accelerator design</td>
<td>Accelerator, CPU, and I/O integration</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Phase 2 selection decisions</strong></td>
<td>Formal verification of accelerator design</td>
<td>Full software integration</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Phase 3 selection decisions</strong></td>
<td></td>
<td>Accelerator chip demonstration</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Building block CDR
- Emulation of logistic regression problem
- FHE algorithm development
- Design and emulate FHE building blocks
- Building block CDR
- Emulation of CNN inference problem
- Full accelerator design ready for tapeout
- Fully emulate integrated accelerator design
- Formal verification of accelerator design
- **Accelerator design CDR**
- Emulation of CNN inference problem
- **Accelerator chip demonstration**
- **Accelerator execution of CNN training problem**
- **Accelerator chip demonstration**
- **Accelerator execution of CNN training problem**
- Final FHE accelerator design
- FHE accelerator prototype

Distribution Statement A. Approved for public release. Distribution unlimited.
Transition opportunities

Align to OSD priority

OSD 5G security priorities and funding
- Encrypted operations in complex networks
- Unique chip enhancements for 5G security
  - Accelerators for encryption
- Military operations in the 5G environment
  - Operating within existing infrastructure

Influence standards

V&V with IC

Provide DPRIVE accelerator to NSA for analysis and adoption to classified problems running classified algorithms

DPRIVE fits the Electronics Resurgence Initiative by impacting both DoD and commercial needs
## Algorithms and Software Packages for FHE

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Strengths</th>
<th>Weakness</th>
<th>Sample Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>BGV (Brakerski-Gentry-Vaikuntanathan)</td>
<td>Efficient bootstrapping</td>
<td>Inefficient floating point</td>
<td>CNN Training</td>
</tr>
<tr>
<td>BFV (Brakerski/Fan-Vercauteren)</td>
<td>Better integer computations</td>
<td>Inefficient floating point &amp; inefficient bootstrapping</td>
<td>CNN Training</td>
</tr>
<tr>
<td>CKKS (Cheon-Kim-Kim-Song)</td>
<td>Floating point</td>
<td>Precision issues</td>
<td>CNN Evaluation</td>
</tr>
<tr>
<td>FHEW (fastest homomorphic encryption in the west)</td>
<td>Logical evaluations; fast bootstrapping</td>
<td>Not standardized (underexplored)</td>
<td>Dijkstra's algorithm</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Library</th>
<th>Comment</th>
<th>Algorithms Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>HEBlib</td>
<td>Developed by IBM; original library by Gentry and others at IBM</td>
<td>BGV, CKKS</td>
</tr>
<tr>
<td>Palisade</td>
<td>Funded originally by the DARPA PROCEED and SafeWare programs</td>
<td>BGV, BFV, CKKS, FHEW</td>
</tr>
<tr>
<td>SEAL</td>
<td>Developed by Microsoft</td>
<td>BFV, CKKS</td>
</tr>
<tr>
<td>FHEW</td>
<td>Univ. California San Diego to showcase the new FHEW algorithm</td>
<td>FHEW</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Testing Framework</th>
<th>Comment</th>
<th>Supported Libraries</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHEEP (SHEEP is a Homomorphic Encryption Evaluation Platform)</td>
<td>A platform for practitioners to evaluate the state-of-the-art of (fully) homomorphic encryption technology</td>
<td>HEBlib, SEAL, Palisade, and TFHE</td>
</tr>
<tr>
<td>E3 (Encrypt Everything Everywhere)</td>
<td>New York University built to enable using different libraries</td>
<td>TFHE, FHEW, HEBlib and SEAL</td>
</tr>
</tbody>
</table>
Primer on Lattice cryptography: core of HE - hard math problem

1. **Key generation**
   - Create a lattice from secret basis functions

2. **Encryption**
   - Map plaintext onto lattice \( \rightarrow \) cipher text
   - Representative bits of plaintext data

3. **Decryption**
   - Find nearest basis to map back to plaintext

This is the **shortest vector problem**
- Known to be HP-hard
- Makes it quantum computer resistant
- Real lattice crypto schemes have 10's of thousands of dimensions

Distribution Statement A. Approved for public release. Distribution unlimited.
HE limited by noise generated in today’s computer architectures

1. Embedded noise accumulates through processing
2. Accumulated noise corrupts data
3. Bootstrapping strips noise to allow for continued processing

Proved by Gentry ‘09

Distribution Statement A. Approved for public release. Distribution unlimited.
Homomorphic encryption (HE) protects data even when operating on untrusted hardware.

Homomorphism is a mapping between two algebraic structures that preserves the operations of the structures.

\[ \begin{array}{c}
\text{The quick brown fox jumped over the lazy dog} \\
\Rightarrow \begin{array}{c}
\begin{array}{c}
\quad \\
\Rightarrow \\
\quad \\
\end{array}
\end{array}
\end{array} = 42 \]

\[ \begin{array}{c}
\text{Enc(42)}
\end{array} \]

**Set of Boolean functions**

- Increasing number of operations under cipher → more complex algorithms
- Decreasing computation burden → more complex protocols

**Partial HE (PHE)**
- Run 1 algorithm stage
- Easy to compute
- Hard to ensure privacy

Applying to neural networks
- Each layer is multiplies, adds, and an activation (e.g., nonlinear) function

**Somewhat HE (SWHE)**
- Run a few algorithm stages
- Harder to compute
- Private but not complete

Good for small, fixed challenges
- One or a few layers
- Some operations in the clear

**Fully HE (FHE)**
- Run any algorithm
- Very hardtop compute
- Proven privacy

Every stage requires bootstrapping for normalization
- Full networks
- Only results
Matrix of alternatives for protected computing methods

<table>
<thead>
<tr>
<th></th>
<th>Provably Secure</th>
<th>Processing Overhead</th>
<th>Communication Overhead</th>
<th>Protocol complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current encryption</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>Trusted enclaves</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Multi-Party Computation (MPC)</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>PHE</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>S/W FHE</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>DPRIVE</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

- Current encryption (e.g., AES) has to decrypt data to process, which leads to difficult key sharing and complex protocols; requires fully trusted hardware
- Trusted enclaves can be easily built in and used in processors. These have never theoretically been shown to be secure and most have been compromised (e.g., a recent talk at 36C3 shows breaking Arm's latest TrustZone-M trusted enclave)
- MPC requires a significant amount of communication between platforms and complex protocols that all have to be coordinated and work together
- PHE schemes are used only because of the computational complexity of FHE but lead to kludgy and cumbersome protocols that are unlikely to be usable
- Only FHE satisfies all of the needs for security and enables real-world capabilities like protecting communication’s metadata and protecting sensitive operational information while still enabling data processing
- DPRIVE will enabling realizing the full benefits of FHE for real problems
How DPRIVE operates as an FHE accelerator

Standard Accelerator Operation

Storage

CPU

Plaintext in

Plaintext out

Accelerator

(math computations)

e.g., GPU, FPGA

DPRIVE FHE Operation

Storage

CPU

Ciphertext in

Ciphertext out

Public Key

DPRIVE Accelerator

The DPRIVE accelerator allows data to stay encrypted and operates on the cipher text with the *public* key. Computations and data processing without sharing the *private* key or ever exposing the plaintext.