Virtual Secure Platform:
A Five-Stage Pipeline Processor over TFHE

Kotaro Matsuoka
Ryotaro Banno
Naoki Matsumoto
Takashi Sato
Song Bian

Kyoto University
Cloud Computing

Program & data are in **PLAIN TEXT**
Program execution may be wiretapped
Just encrypt everything!

Makes wiretapping pointless

Proposed method: Virtual Secure Platform
How can we execute an encrypted program?

- Processor = Logic circuits = Graph of logic gates
- Above question can be reduced to “How can we evaluate logic gates over encrypted bits?”
- One way is Fully Homomorphic Encryption
  - We use TFHE

![Diagram of logic gates and execution process]
How can we execute an encrypted program?

- Processor = Logic circuits = Graph of logic gates
- Above question can be reduced to “How can we evaluate logic gates over encrypted bits?”
- One way is Fully Homomorphic Encryption
  - We use TFHE
Target

1. High-level language compatibility
   - tradeoff between security and usability
2. Speed
   - Operations in FHE are significantly slow
3. Open-source implementation
   - Providing reproducibility & possibility to verify implementation
Our Contribution

1. High-level language Compatibility
   - Implemented **RISC-V-based ISA** & **LLVM based C compiler**

2. Speed
   - Iyokan (FHE gate **evaluation engine**)  
     - Utilizing parallelism
   - Proposed and implemented **CMUX Memory**, an optimized memory structure over TFHE
   - By combining both, VSP achieves **1600x** faster than FURISC, state-of-the-art FHE-based processor emulation

3. Open-source implementation
   - Available on GitHub [https://github.com/virtualsecureplatform/kvsp](https://github.com/virtualsecureplatform/kvsp)
Our Contribution

1. High-level language Compatibility
   - Implemented **RISC-V-based ISA & LLVM based C compiler**

2. Speed
   - **Iyokan (FHE gate evaluation engine)**
     - Utilizing parallelism
   - Proposed and implemented **CMUX Memory**, an optimized memory structure over TFHE
   - By combining both, VSP achieves **1600x** faster than FURISC, state-of-the-art FHE-based processor emulation

3. Open-source implementation
   - Available on **GitHub**
VSP Components

Cloud Server

Processor Circuit

Replace by TFHE's gates

Iyokan

Processor Circuit over TFHE

Program Execution

Wiretapper
VSP Components

Cloud Server

Program & data

From Client

Replace by TFHE's gates

Iyokan

Processor Circuit over TFHE

CMUX Memory

Program Execution

Wiretapper
VSP Components

Cloud Server

Program & data

Processor Circuit

Replace by TFHE's gates

From Client

Iyokan

Processor Circuit over TFHE

CMUX Memory

Evaluate

Program Execution

Wiretapper
VSP Components

From Client

Replace by TFHE's gates

Program Execution

Program & data

Abstraction

Cloud Server

Wiretapper

Iyokan

Evaluate

Processor Circuit

Processor Circuit over TFHE

CMUX Memory
Gate-level Parallelism

Pipeline-induced Parallelism
Gate-level Parallelism

- Gates can be executed in parallel
  - But some cannot (dependency)

Pipeline-induced Parallelism
Gate-level Parallelism

- Gates can be executed in parallel
  - But some cannot (dependency)

Pipeline-induced Parallelism
Gate-level Parallelism

- Gates can be executed in parallel
  - But some cannot (dependency)

Pipeline-induced Parallelism

- Divide a circuit into parallely evaluable circuits
Gate-level Parallelism

- Gates can be executed in parallel
  - But some cannot (dependency)

Pipeline-induced Parallelism

- Divide a circuit into parallelly evaluable circuits

Parallelism Exploitation in VSP

- Relies on the physical execution environment
  - Different from physical circuit (naturally parallel)
  - Needs massively parallel architecture and scheduling (lyokan’s role)
  - lyokan can handle multicore CPU & multi GPU
**Speed Evaluation**

- Evaluation environments
  - Test program: Calculating Hamming distance
  - Case 1: AWS p3.16xlarge (64 vCPUs, 488GB RAM, and 8 V100 GPUs)
  - Case 2: Sakura Koukaryoku (16vCPUs, 128GB RAM, and 1 V100 GPU)
- To utilize parallelism, massive physical capability (= scheduling) is needed

<table>
<thead>
<tr>
<th>Case</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>w/ Pipeline</td>
<td><strong>0.8 s/cycle</strong></td>
<td>1.7 s/cycle</td>
</tr>
<tr>
<td>w/o Pipeline</td>
<td>1.3 s/cycle</td>
<td>2.4 s/cycle</td>
</tr>
</tbody>
</table>
Comparison with FURISC

- **FURISC**
  - State-of-the-art processor over FHE
  - Supports only one instruction, Subtract Branch if Negative (SBN)
  - No compiler support (VSP has C Compiler support)

- **VSP** is **1600x** faster
  - Exploitation of Parallelism by Iyokan
  - CMUX Memory (Cryptographical optimization specific to TFHE)

<table>
<thead>
<tr>
<th>Name</th>
<th>sec./cycle</th>
<th>Implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td>VSP</td>
<td>0.8</td>
<td>Public</td>
</tr>
<tr>
<td>FURISC</td>
<td>1278 (est.)</td>
<td>Private</td>
</tr>
</tbody>
</table>
Conclusion

- VSP enables offloading computation while keeping the program secret.
- Exploitation of parallelism is important in FHE circuit evaluation.
  - Both gate-level and architectural parallelism
  - Needs massive physical machine capability and scheduling

Thank you for watching!

- Our implementation: [https://github.com/virtualsecureplatform/kvsp](https://github.com/virtualsecureplatform/kvsp)
- Contact
  - GitHub Discussions: [https://github.com/virtualsecureplatform/kvsp/discussions](https://github.com/virtualsecureplatform/kvsp/discussions)
  - Email: paper@easter.kuee.kyoto-u.ac.jp