MEVAL: Multiparty EVALuator System, and Fast Secure AES Computation

Koji Chida, Koki Hamada, Dai Ikarashi, Ryo Kikuchi
@ NTT Corporation
Result

MEVAL: a system of SS-based MPC

AES computation on MEVAL

- 517 Mbps (4,041,655 AES/s) in passive security over 3x faster than [AFLNO16]
- 131 Mbps (1,025,303 AES/s) in active security with abort over 4x faster than [ABFLLNOWW17]

new technique for achieving active security with abort
Result

MEVAL: a system of SS-based MPC

AES computation on MEVAL

- **517 Mbps (4,041,655 AES/s)** in passive security over 3x faster than [AFLNO16]
- **131 Mbps (1,025,303 AES/s)** in active security with abort over 4x faster than [ABFLLNOWW17]

new technique for achieving active security with abort

Assuming 5400 AND/AES
High-throughput MPC [AFLNO16, ABFLLNOWW17]

MPC can be high-throughput when

- secret sharing
- three parties
- honest majority

\[ [a]_0 \rightarrow [a]_1 \rightarrow [a]_2 \]

High-throughput MPC [AFLNO16, ABFLLNOWW17]

MPC can be high-throughput when

- secret sharing
- three parties
- honest majority

\[ f(a) \]


Two results in different security models

**passive security**

An adversary follows the protocol

169 Mbps \((1,324,117 \text{ AES/s})\) [AFL+16]

**active security with abort**

An adversary can do arbitrary behavior

Honest parties output ⊥ if an adversary cheats

27 Mbps \((213,472 \text{ AES/s})\) [ABF+17]
MEVAL: Multiparty EVALuator
MEVAL: Multiparty EVALuator

MEVAL is an MPC system we have developed

- secret sharing
- three parties
- honest majority

User can choose either

- passive
- active w/ abort
Feature of MEVAL

- **Fulfilling functions**
  - logical/arithmetic circuit
  - high-level operations: join, sort, map, comparison, etc.
  - various SS conversion and field conversion

- **Efficiency**
  - original (optimized) protocols
  - implemented by expert programmer *Dai Ikarashi*

---

ex.) His erasure code library for OpenStack Swift storage is 10% faster than Intel's and 2 times faster than Swift-embedded "j-erasure" (USENIX FAST'13, a storage top conference)!

http://www.ntt.co.jp/news2015/1505e/150518a.html
## Efficiency of MEVAL

### Experiment on $10^7$ records, 1G LAN

<table>
<thead>
<tr>
<th>Function</th>
<th>Passive</th>
<th>Active</th>
</tr>
</thead>
<tbody>
<tr>
<td>Join (w/ duplicated keys)</td>
<td>68.5 (sec)</td>
<td>180.2 (sec)</td>
</tr>
<tr>
<td>Sort</td>
<td>46.6</td>
<td>121.6</td>
</tr>
<tr>
<td>Typecast: $GF(2^{61-1})$ to $GF(2^{127-1})$</td>
<td>5.8</td>
<td>16.9</td>
</tr>
<tr>
<td>Map (table refer)</td>
<td>49.9</td>
<td>131.9</td>
</tr>
<tr>
<td>Comparison</td>
<td>0.89</td>
<td>14.6</td>
</tr>
</tbody>
</table>

E.g.) about 200 sec in [BNTW14] (passive security)

Motivation

If speeding up AES (on MEVAL), how fast is it?
Approaches: speeding up AES

- Reconsidering the circuit of AES computation
- Dual execution of multiplication protocol
- Reusing inversion result
Reconsidering circuit of AES computation
Steps of AES

Input is $4 \times 4$ matrix of bytes

Iterate the following steps

- SubBytes: Compute inverse of each byte in $\text{GF}(2^8)$
- ShiftRows: Rotate each row
- MixColumns: Apply linear operation to each column
- AddRoundKey: Compute exclusive-or (addition in $\text{GF}(2^m)$) with a round key

All except SubBytes can be local computations

- if we use linear SS

$\rightarrow$ Focusing on SubBytes; computing inverse in $\text{GF}(2^8)$
Replicated secret sharing [ISN89, CDI05] in GF(2^8)

Efficient multiplication protocol exists [IKHC14]
each party sends a single element to another

\[ a = a_0 + a_1 + a_2 \]


Power of characteristic in replicated SS

- Input: $[a]$  
- Output: $[a^{2m}]$ for some integer $m$

If $a = a_0 + a_1 + a_2$,  
$a^{2m} = (a_0 + a_1 + a_2)^{2m} = (a_0)^{2m} + (a_1)^{2m} + (a_2)^{2m}$ in $GF(2^8)$

$[a] = a_0, a_1$  
$[a^{2m}] = (a_0)^{2m}, (a_1)^{2m}$  
$[a^{2m}] = (a_1)^{2m}, (a_2)^{2m}$  
$[a^{2m}] = (a_2)^{2m}, (a_0)^{2m}$

Local computation
Computing inverse in replicated SS and GF($2^8$)

- **Input:** $[a]$
- **Output:** $[a^{-1}] = [a^{254}]$

1. $[a^2] := [a]^2$

2. $[a^3] \leftarrow Mult([a^2], [a])$

3. $[a^{12}] := [a^3]^4$

4. $[a^{15}] \leftarrow Mult([a^{12}], [a^3])$

5. $[a^{14}] \leftarrow Mult([a^{12}], [a^2])$

6. $[a^{240}] := [a^{15}]^{16}$

7. $[a^{254}] \leftarrow Mult([a^{240}], [a^{14}])$

**Communication:** 4 Mult

**Round:** 3 Mult
### Comm. comp. of single AES-round

<table>
<thead>
<tr>
<th></th>
<th>Amount of communication</th>
<th># of rounds</th>
<th># of procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inversion circuit in GF(2^8) [BP10, Tal16]</td>
<td>512 bit =32 bit x 16 SubBytes</td>
<td>6</td>
<td>115 gates</td>
</tr>
<tr>
<td>Ours</td>
<td>512 bit =4 Mult x 16 SubBytes x 8 bit</td>
<td>3</td>
<td>7 steps</td>
</tr>
</tbody>
</table>

- **Same communication, a half round**
- **Simple implementation**
  - Ours requires 7 steps while existing method requires 115 gates

Dual execution of multiplication protocol
[AFL+16] uses 10G network in ring topology

-> Each party can send data with **20 Gbps in total**

10G is full-duplex: parties interact with each other with 10 Gbps simultaneously
Multiplication protocol is asymmetric
Each party sends single element to another

Execute both directions in parallel

\[ \text{Mult (clockwise)} \]

\[ \text{Mult (anticlockwise)} \]
Reusing inversion result
Reusing inversion result

A part of input is the same in different blocks in AES-CTR

Optimization makes AES-CTR 1.25x faster

message \oplus \text{AES.Enc} \rightarrow \text{c0a984dd...00000000}

message \oplus \text{AES.Enc} \rightarrow \text{c0a984dd...00000001}
Experiment (passive security)
Environment

- **MEVAL2**
  - Intel core i7 6900K (3.2GHz x 8core), total 25.6 GHz
  - Intel X550 10G-Ether, **ring topology**, ping 0.112 ms, actual measured throughput 9,094 Mbps

- **[AFL+16]**
  - Intel Xeon E5-2650 v3 (2.3GHz x 10 cores) x 2 socket, total 46 GHz
  - (No article name) 10 Gbps Network, **ring topology**, ping 0.13 ms, no actual measured throughput
Result (passive security)

<table>
<thead>
<tr>
<th></th>
<th>Environment</th>
<th>Throughput</th>
</tr>
</thead>
<tbody>
<tr>
<td>[AFL+16]</td>
<td>10G NW, ring structure 2.3 GHz x 20 core x 2 socket, total 46 GHz</td>
<td>169 Mbps (1,324,117 AES/s)</td>
</tr>
<tr>
<td>MEVAL</td>
<td>10G NW, ring structure 3.2 GHz × 8 core, total 25.6 GHz</td>
<td>517 Mbps (4,041,655 AES/s)</td>
</tr>
</tbody>
</table>

Experiment: AES-CTR, 128 bit key, 240MB input
MEVAL uses PCs in spite of [AFL+16] uses servers

over 3x faster!
Experiment (active security with abort)
Two methods for active security with abort on MEVAL2

- [IKHC14]: adding homomorphic MAC

Observation

- AES-CTR is SIMD operation

- Shuffling does NOT affect each output
Compute original and shuffled inputs and compare the outputs

plaintext (128bit) → AES → ciphertext (128bit)

AES
AES
AES
AES

Oblivious shuffle

AES
AES
AES
AES

Open $\pi$

un-shuffle

??
Why is it secure? 

Pr[A wins] ≈ 1/N, where N is the number of records

plaintext (128bit)  AES  ciphertext (128bit)

AES  AES

AES  AES

AES  AES

which row corresponds to?

Open $\pi$

un-shuffle

Oblivious shuffle
Guess throughput as

$$1,152,751,967 \text{ [AND/sec]} / 5400 \text{ [AND/AES]} \times 128 \text{ [bit/AES]} = 27,324,491 \text{ [bps]}$$
Summary

- **MEVAL**
  - SS-based MPC, three-party, honest majority
  - User can choose passive or active w/ abort
  - efficient, providing various functions

- **AES computation on MEVAL**
  - 517 Mbps *(4,041,655 AES/s)* in passive security
  - 131 Mbps *(1,025,303 AES/s)* in active security with abort
  - New method for achieving active security with abort
Comparison in gate/sec (= bit/sec)

<table>
<thead>
<tr>
<th>Measure</th>
<th>Passive</th>
<th>Active</th>
</tr>
</thead>
<tbody>
<tr>
<td>[AFL+16]</td>
<td>7,150,231,800</td>
<td>-</td>
</tr>
<tr>
<td>2x [AFL+16]</td>
<td>14,300,463,600</td>
<td>-</td>
</tr>
<tr>
<td>[ABF+17]</td>
<td>-</td>
<td>1,152,751,967</td>
</tr>
<tr>
<td>2x [ABF+17]</td>
<td>-</td>
<td>2,305,503,934</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Measure</th>
<th>Passive</th>
<th>Active</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ours</td>
<td>2,069,327,200</td>
<td>524,955,200</td>
</tr>
<tr>
<td>mult (GF(2^8)) /sec</td>
<td>16,554,617,600</td>
<td>4,199,641,600</td>
</tr>
</tbody>
</table>
Theoretical value of throughput (passive)

If we ignore local computation,

9094 (Mbps) * 2 / (512 * 0.8 * 10) * 128 = 568.37 (Mbps)

<table>
<thead>
<tr>
<th>Passive</th>
<th>Actual/theory</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEVAL</td>
<td>517 Mbps</td>
</tr>
<tr>
<td></td>
<td>0.91</td>
</tr>
</tbody>
</table>