FASTEN: An FPGA-based Secure System for Big Data Processing

Boeui Hong, Han-Yee Kim, Minsu Kim, Lei Xu, Weidong Shi, and Taeweon Shu
What’s the problem?

• Cloud computing is becoming increasingly more important

• But it carries security risks
  • Malicious insiders
  • Side-channel attacks between co-residing VMs

• Modern FPGAs have properties that can help

• Helps establish better trust in cloud service provides
Where do we stand?

• Processor vendors offer certain security features

• FASTEN moves processing kernels into FPGAs, isolated from host

• Leverages FPGAs’ tamper resistant features
Security features in FPGAs

- FPGAs are configured by bitstreams
  - Can be encrypted
  - Only decrypted inside the FPGA’s AES engine
  - Contains HMAC key + authentication code

- Tamper resistant key-storage
  - BBRAM: Writable, but not externally readable
  - Tampering attempts nullify key/FPGA
  - PUF: Unique private key with public key

<table>
<thead>
<tr>
<th>FPGA vendors</th>
<th>Xilinx</th>
</tr>
</thead>
<tbody>
<tr>
<td>Devices</td>
<td>Spartan-7, Artix-7, Kintex-7, Virtex-7</td>
</tr>
<tr>
<td>Symmetric Encryption</td>
<td>AES-CBC 256</td>
</tr>
<tr>
<td>Asymmetric Encryption</td>
<td>RSA</td>
</tr>
<tr>
<td>Integrity/Authenticity Hash Function</td>
<td>HMAC-SHA-256</td>
</tr>
<tr>
<td>Key Storage</td>
<td>BBRAM</td>
</tr>
<tr>
<td></td>
<td>eFUSE</td>
</tr>
<tr>
<td>Side-channel Protection</td>
<td>N/A</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Altera</th>
<th>Microsemi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arria 10</td>
<td>Stratix 10</td>
</tr>
<tr>
<td>AES-CTR 256</td>
<td>AES-GCM 256</td>
</tr>
<tr>
<td>RSA, ECC</td>
<td>RSA, ECC</td>
</tr>
<tr>
<td>SHA-256/384 ECDSA</td>
<td>HMAC-SHA-256 SHA-256/384 ECDSA</td>
</tr>
<tr>
<td>BBRAM eFUSE</td>
<td>BBRAM eFUSE</td>
</tr>
<tr>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
FASTEN: Threat Model

Private, security-sensitive data

CSP – Not trusted

Provides FPGAs to CSP

Client – Trusted

Provides CAD tool + public key database

FPGA Vendor - Trusted
FASTEN: Architecture

1) FPGA vendor: private-public key pair generation

2) Client: bitstream generation and encryption, data encryption

3) CSP: FPGA configuration, data processing and output generation
FPGA vendor’s responsibility

• FPGAs must support PUFs to generate unique private key - $sk_i$

• FPGA’s crypto engine generates corresponding public key - $pk_i$

• Manage database matching public keys to FPGA IDs

• Provide CAD tools for synthesis
Client’s responsibility

- Generate **data key** - $dk$, and encrypt the data with it
- Embed $dk$ in compute kernel source code
- Generate bitstream with CAD, encrypt it using **bitstream key** - $bk$
- Encrypt $bk$ with FPGA’s public key - $pk_i$
- Send resulting $IMAGE_i$ and encrypted data to CSP
Client’s responsibility

\[ dk = \text{data key} \mid bk = \text{bit-stream key} \mid pk_i / sk_i = \text{RSA key-pair for FPGA}_i \]

\[ IMAGE_i = AES_{bk}( \text{Bitstream}( \text{program} + dk ) + \text{HMAC} ) + RSA_{pki}( bk ) \]

\[ \text{ENCRYPTED DATA} = AES_{dk}( \text{data} ) \]
CSP’s responsibility

• Deploy $\text{IMAGE}_i$ to FPGA$_i$

• Distribute encrypted data to FPGAs (Hadoop)

  • Decrypt $bk$ using $sk_i$
  • Decrypt bitstream using $bk$
  • Decrypt data while processing it, and encrypt results using $dk$

• Hand over encrypted results to client
Security Analysis

• Conventional software-based attacks infeasible
  • E.g. memory dumping to extract key/data

• But three new points of attack:
  • Extraction of private key $sk_i$
  • Reading back configured bitstream from FPGA
  • SCA against AES engine inside FPGA
Security Analysis

- Private key extraction:
  - Secrecy is guaranteed by vendors
  - Tampering attempts lead to nullification

- Bitstream readback:
  - Could reveal data key $d_k$
  - Capability can be disabled by vendor or inside CAD tool

- SCA on AES engines:
  - Hardware capabilities to minimize this risk

<table>
<thead>
<tr>
<th>FPGA vendors</th>
<th>Devices</th>
<th>Xilinx</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Spartan-7, Artix-7, Kintex-7, Virtex-7</td>
<td>Kintex Ultrascale, Virtex Ultrascale</td>
</tr>
<tr>
<td>Symmetric Encryption</td>
<td>AES-CBC 256</td>
<td>AES-GCM 256</td>
</tr>
<tr>
<td>Asymmetric Encryption</td>
<td>RSA</td>
<td>RSA</td>
</tr>
<tr>
<td>Integrity/Authenticity Hash Function</td>
<td>HMAC-SHA-256</td>
<td>AES-GCM 256</td>
</tr>
<tr>
<td>Key Storage</td>
<td>BBRAM eFUSE</td>
<td>BBRAM eFUSE</td>
</tr>
<tr>
<td>Side-channel Protection</td>
<td>N/A</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Hadoop MapReduce implementation

• Mapper
  • Outputs $<key, value>$ pair
    • Those pairs are merged and sent to reducers according to $key$ (shuffle)

• Reducer
  • Generates output corresponding to $value$

• Software portion is only responsible for handling encrypted data
Hadoop MapReduce implementation
Evaluation

• K-means, Locally weighted linear regression (LWLR), Sobel filter

• Assumes user is not familiar with HDL
  • Direct C++ to hardware translation - no special directives used

<table>
<thead>
<tr>
<th>Resources Utilization</th>
<th>Hardware Components in FPGA Used # (Utilization %)</th>
<th>Security Modules</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>#BRAMs (18Kbit) #DSPs (4SE) #FFs #LUTs</td>
<td>w/o AES (baseline)</td>
</tr>
<tr>
<td>Application Kernels</td>
<td></td>
<td></td>
</tr>
<tr>
<td>K-means kernel</td>
<td>2 (1%) 4 (2%) 1290 (1%) 1532 (3%)</td>
<td>1.772W</td>
</tr>
<tr>
<td>LWLR kernel</td>
<td>2 (1%) 4 (2%) 1918 (2%) 2942 (6%)</td>
<td>1.845W</td>
</tr>
<tr>
<td>Sobel filter kernel</td>
<td>18 (3%) 4 (2%) 3356 (3%) 1691 (3%)</td>
<td></td>
</tr>
<tr>
<td>AXI-DMA</td>
<td>2 (~0%) 0 (0%) 1821 (1%) 1277 (2%)</td>
<td></td>
</tr>
<tr>
<td>AES</td>
<td>44 (16%) 8 (3%) 11700 (11%) 7070 (13%)</td>
<td></td>
</tr>
<tr>
<td>AXI4 memory</td>
<td>0 (0%) 0 (0%) 1163 (1%) 996 (~0%)</td>
<td></td>
</tr>
<tr>
<td>Interconnection</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Evaluation

- FASTEN beats standard Hadoop_{TE_ES} (~9%)
- FASTEN_{dup} approaches baseline at cost of hardware
- Naive software-only kernels on average 58% slower than accelerated (insecure!) version
Evaluation

• Note: This is without HDL specific directives

• Better AES engines for HDL-based design can be used
Conclusion

• Better security with relatively little overhead
  • Additional hardware

• Can make CSPs better option for critical data

• Trust gets shifted into FPGAs’ security features (& vendor)

• Still room for improvement
Unknown command. Type /help for available commands.

Unknown command. Type /help for available commands.

Unknown command. Type /help for available commands.

I can't help you.