1. Project Overview

Cloud FPGAs have been recently deployed in data centers. Several research papers [1,2,3] have reported Cloud FPGA side or covert channel attacks. The requirement for such attacks is the adversary knows the identification information of the Cloud FPGA instances. However, existing Cloud FPGA providers, such as AWS, secure their infrastructures through a number of countermeasures. Users do not have access to physical pins, but only a strict RTL “shell” interface that prevents access to Xilinx fUSE and Device DNA primitives, for example.

This work shows that it is still possible to fingerprint Cloud FPGAs through Physical Unclonable Functions (PUFs) based on the decay of Dynamic Random Access Memories (DRAMs) attached to the Cloud FPGA boards, and, by extension, the FPGAs themselves.

2. System Diagram

The workflow of AWS F1, shown below, requires users to upload Design Checkpoint (DCP) files generated by Xilinx Vivado design tools, and then performs Design Rule Checks (DRCs) before generating bitstream files based on them. The generated bitstream files are managed by AWS and registered as Amazon FPGA Image (AFI), users can only reconfigure their FPGAs referencing AFI numbers.

A set of FPGA boards is connected to a server over PCIe. The boards contain FPGA chips, and are placed in fixed slots in the server. Each FPGA has access to four dedicated DRAM modules.

3. DRAM PUFs on AWS F1

Each DRAM chip consists of DRAM banks, which are arrays of DRAM cells. The capacitor charge in each cell leaks over time through different leakage paths. If DRAM self-refresh and Error-Correcting Code (ECC) are disabled, bit flips (errors) will occur at a different location after a certain decay period, and the variation can be used in DRAM PUFs.

Figure 3. PUFs that exploit DRAM charge leakage on DRAMs can uniquely identify the underlying hardware

As shown in Figure 4, AFI-0 is first loaded to write all 1s to a certain area of a DRAM module. Then AFI-1 is loaded to stop memory self-refresh. Finally, after a fixed amount of time, AFI-0 is re-loaded to measure bit flips in the written addresses.

Figure 4. Three Steps to measure DRAM PUFs with two AFIs

4. Fingerprinting Metric

To quantify how similar or different DRAM PUF responses are, we use the Jaccard index [4, 5]. Let F1 and F2 denote the set of bit flips in two DRAM PUF responses. The Jaccard Index \( J(F_1, F_2) \) is defined as:

\[
J(F_1, F_2) = \frac{|F_1 \cap F_2|}{|F_1 \cup F_2|}
\]

Intra-device: from the same DRAM, \( J(F_{1I}, F_{2I}) \to 1 \)
Inter-device: from different DRAMs, \( J(F_{1I}, F_{2I}) \to 0 \)

Figure 5. Bitmap of an example DRAM PUF response

Figure 6. Distribution of Jaccard indices for each pair of DRAM PUF responses on f1.2xlarge instances

5. Evaluation

Our evaluation is the first to calculate the probability of renting the same FPGA as a function of time, and demonstrate that there is no overlap between FPGAs of different instance types.

Table 1: Number and type of FPGA instances rented, along with the number of unique sets of FPGAs found and the approximate experimental cost using spot instances

<table>
<thead>
<tr>
<th>F1 Type</th>
<th>Number of Slots</th>
<th>Number of FPGAs</th>
<th>Unique FPGAs</th>
<th>Approx. Cost ($)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2xlarge</td>
<td>1</td>
<td>60×1</td>
<td>10×1</td>
<td>3.47</td>
</tr>
<tr>
<td>4xlarge</td>
<td>2</td>
<td>60×2</td>
<td>6×2</td>
<td>8.91</td>
</tr>
<tr>
<td>16xlarge</td>
<td>8</td>
<td>60×8</td>
<td>8×8</td>
<td>83.16</td>
</tr>
</tbody>
</table>

Figure 7. Probability of renting re-allocated FPGA boards for different waiting periods. \( y \) denotes the number of pairs (out of \( n \)) for which Jaccard indices are bigger than 0.5. Although the figure only shows slot 0, the probability for all slots is identical, as FPGA ordering does not change within instances

Figure 8. Fingerprinting FPGAs on f1.16xlarge instances with 8 FPGA slots: out of 11 spot instances, only 6 different sets of FPGAs are allocated. In the remaining instances, only 2 additional sets were identified (Table 1)

6. Conclusion

The contributions of this paper are as follows:

- We introduce a novel experimental setup which uses DRAM PUFs to fingerprint AWS cloud FPGAs.
- We conduct the first fingerprinting experiments on cloud FPGAs, extracting unique and stable fingerprints of several Amazon f1.2xlarge, f1.4xlarge and f1.16xlarge instances.
- We propose a set of countermeasures against cloud FPGA fingerprinting.

Our code as well as pre-compiled AFI will be made available at https://caslab.cs.yale.edu/code/cloud-fpga-fingerprinting.

---