Design and Implementation of the Ascend Secure Processor

ABSTRACT

As embedded and mobile devices become ubiquitous, users have become more computationally limited and computation outsourcing is becoming more common. From financial information to medical records, sensitive user data is being sent to and computed upon by the cloud. Data privacy has therefore become a huge concern, as sensitive user data is being revealed to and can be attacked by malicious cloud applications, hypervisors/operating systems, or by insiders. One example is health diagnoses given a user's private medical records. In this case, a mobile device is constantly monitoring its user's symptoms and wants to compute the likelihood that the user has some condition given these symptoms.

EXISTING SYSTEM

Today's secure processors (e.g., Intel+TXT, XOM, Aegis and Intel SGX leak private information when running badly written or malicious programs. Consider the following scenario. A user sends the server an encryption of its symptoms and condition of interest (ciphertext M) and expects the server to run the MedComp() program described above. Which event happens depends on a private bit of the user's data, which can be selected by the server by changing I and the bitmask 0x1. This attack allows the server to learn the value of the bit of interest. The server can then repeat this experiment with different bit masks and values for I, in order to leak multiple bits in the user's data. Of course, even if the server does run a non-malicious program, program characteristics or bugs can leak private information in the same way as curious(). This attack is difficult to prevent. Encrypting data that leaves the secure processor won't help because the attack succeeds if the server sees either

- Memory is accessed sequentially.
- The program terminates immediately vs. makes multiple memory accesses.

Furthermore, adding on-chip cache to the secure processor (to add noise to the memory access patter and/or growing the TCB to include (say) the main memory will not help because the server can just rewrite the curious() program to ensure that visible resources (e.g., disk) are involved.

Disadvantages

- Leak private information when running badly written or malicious programs.
- If the server does run a non-malicious program, program characteristics or bugs can leak private information in the same way as curious().

PROPOSED SYSTEM

This paper has described the Ascend execution model, for running untrusted programs operating safely on sensitive user data, as well as detailed implementation results for an Ascend prototype chip in silicon. This work proves the viability of a single-chip secure processor which can protect the privacy of software intellectual property or user data, as it interacts with an external memory device. The evaluation results are encouraging. The hardware mechanisms needed to support Ascend, when integrated into the 25 core test chip, are roughly the size of a single processor core. Further, average program slowdown considering these mechanisms is estimated to be roughly the cost of running a program in an interpreted language. The Ascend execution model in its current form is somewhat constrained. Ascend does not support multiple tenants sharing the same chip, since on-chip resource sharing can leak private information. Other modules cannot write to Ascend main memory using DMA, and Ascend cannot be used in a multisocket shared memory architecture.

Advantages

- It prevents information leakage over a processor's digital I/O pins.
- The server would then run the program on the user's private data on an Ascend processor, which decrypts user input, runs the program and returns encrypted results to the user.

SOFTWARE REQUIREMENTS

Front-end	:	JSP
Back-End	:	MySQL
Server	:	Tomcat Server
OS	:	WINDOWS 7/above

HARDWARE REQUIREMENTS

PROCESSOR	:	CORE i3
RAM	:	512MB-2GB
HARD DISK	:	40GB