Mitigating DoS Through Trusted Computing Components

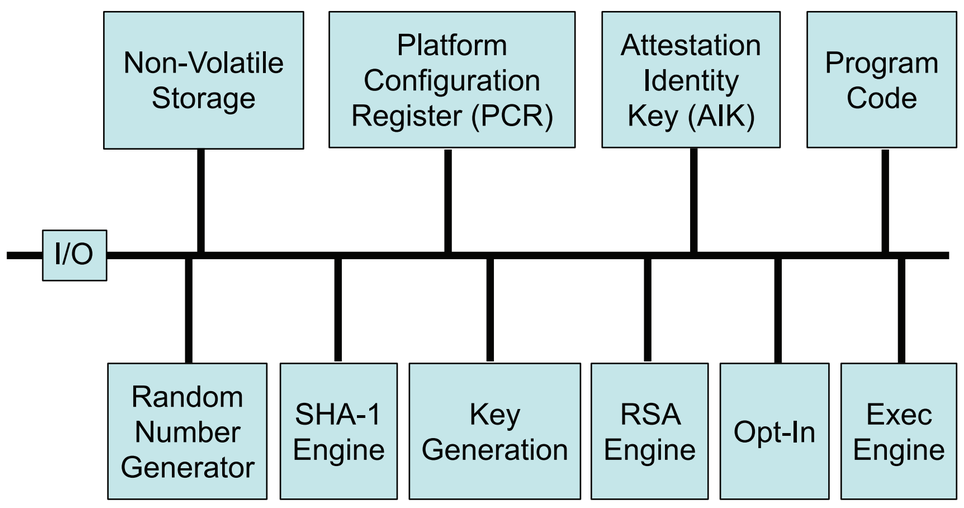
William Enck, Patrick Traynor, Trent Jaeger, Patrick McDaniel, Hong Li*, and Alan Ross*

The Problem

- The Internet provides high throughput and availability of digital content.
- Automation is indigenous to computers.
- Unfortunately, these properties facilitate Denial of Service (DoS) attacks.
- Increasingly, critical resources are subject to DoS attacks.
- At the root of DoS attacks is a cost imbalance between the request and the service.

Trusted Platform Module

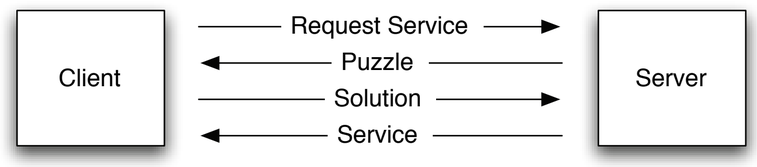
- The Trusted Platform Module (TPM) is the core hardware component defined by the Trusted Computing Group (TCG) architecture.
- The TPM was designed to keep track of state, as well as allow for attestation of that state.
- A TPM device can be viewed as a glorified smartcard, rather than a cryptographic coprocessor.



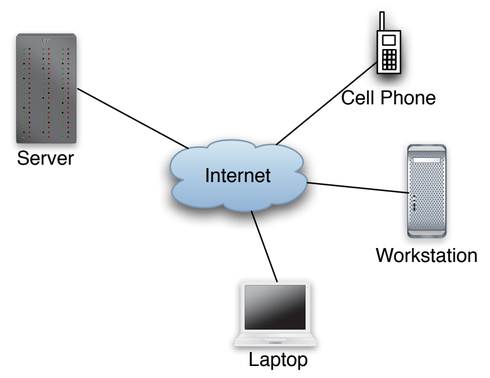
- In current instantiations, the TPM resides on a slow system bus, incurring high latency for each access.

Client Puzzles

- Client puzzles attempt to equal the client/server cost imbalance by requiring the client to "pay" for resources.
- The core idea is to slow down the rate at which a client can occupy server resources.
- Traditional client puzzles leverage the hardness of cryptographic operations, causing the client to spend CPU cycles



- The cryptographic puzzles range from reversing hash functions to square root extraction.
- Computationally bound client puzzles work well when all clients contain relatively equal resources.
- How do you tune puzzle difficulty in a heterogeneous environment?

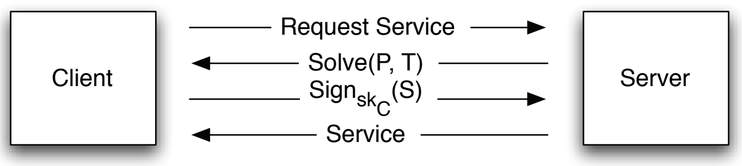


- While the available computational power of processors rapidly increases, the latency to access memory and system devices has remained approximately the same.
- Trusted computing components will soon exist in many devices ranging from workstations to laptops to cell phones and PDAs.

TPM-based Client Puzzles

A Novice Approach

- The strait forward application of the TPM to client puzzles is to use remote attestation and require the client to prove a particular solver was used.
- This solver takes both a simple puzzle and a minimum time as parameters.
- The solver ensures that at least the minimum time has elapsed.



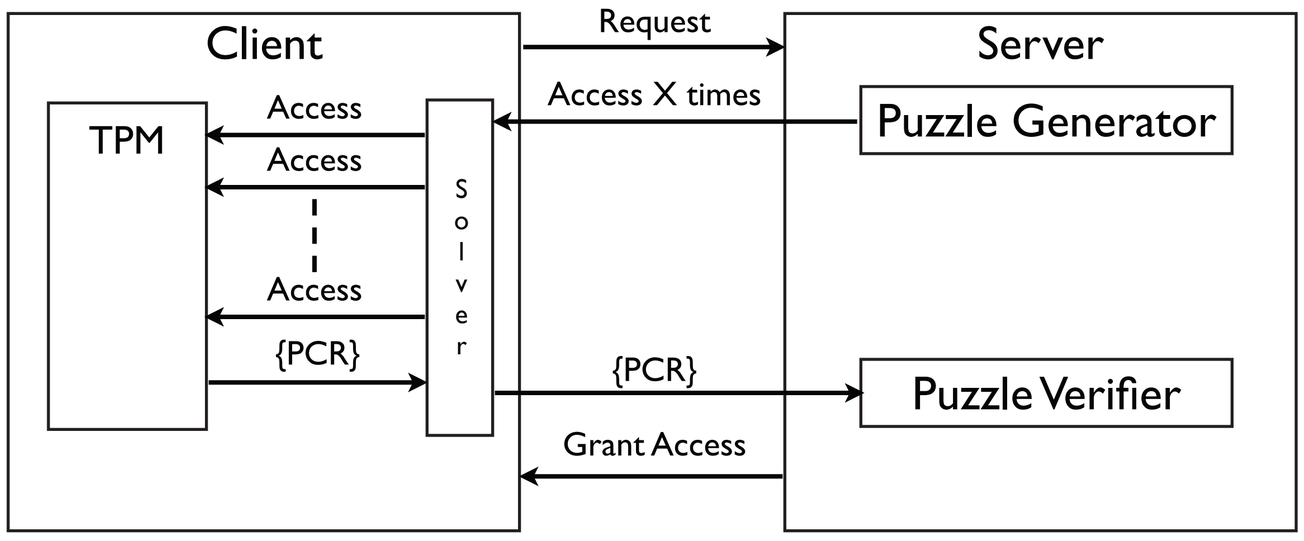
Exploiting the TPM Properties

- The novice approach is not feasible, because it requires the server to know and verify the operating state of each client in the heterogeneous environment.
- The TPM holds state in Platform Configuration Registers (PCR)
- The state contained in the PCRs is updated through the Extend operation:

  Extend(PCR[i], value): PCR[i] = SHA1(PCR[i] · value)

- In remote attestation, the TPM proves PCR values came from hardware (i.e., they were not simulated).
- Accessing the TPM incurs high latency.
- **IDEA: Record accesses to the TPM with PCR extensions.**

High-level Architecture



Challenges

- The TPM cannot execute arbitrary code, therefore the backdoor used in cryptographic client puzzles is not available.
- Therefore, the server must simulate each client's extend operations.
- An adversary may attempt to simulate the extend operations as well, but cannot create a valid signature.
- By creating an innovative lower level protocol exploiting the orders of magnitude difference between real and simulated access, we overcome these challenges.

* Denotes authors affiliated with Intel Corporation