cTPM: A Cloud TPM for Cross-Device Trusted Applications

Chen Chen (ETH Zurich), Himanshu Raj, Stefan Saroiu, Alec Wolman (Microsoft Research)
Motivation

- People are using more than one mobile device
- Mobile devices have started to use trusted hardware

A few examples:
- Pasture, Kotla et al. [OSDI’12]
- Trusted sensors, Liu et al. [Mobisys’12]
- TLR, Santos et al. [ASPLOS’14]
Cross-device Data Sharing is Easy
TPM-protected Data Sharing is Hard
Challenge: Sharing Keys across TPMs

- TPM root key never leaves the chip
- Key migration requires PKI + secure execution mode
Main Research Question

- What is the minimal change to the TPM design to enable data sharing across TPMs?
Our Solution: cTPM

- cTPM = TPM + additional root key pre-shared with cloud

Sharing TPM-protected data is easy!
cTPM: Two Additional Benefits

1. Fast and large “remote” NVRAM storage
cTPM: Two Additional Benefits

1. Fast and large NVRAM storage
2. Trusted clock
Talk Outline

- Motivation
- Background & Design alternatives
- Design & Implementation
- Evaluation
- Conclusion
TPM Background

- Trusted Platform Module (TPM): secure co-processor
  - Crypto primitives: SHA1, RSA, ...
  - Code measurement and attestation

- Trust Computing Group (TCG) defines TPM specifications:
  - TPM 1.2: widely deployed today
  - TPM 2.0: emerging new standard to replace TPM 1.2

\[\text{cTPM is based on TPM 2.0}\]
Threat Model & Trust Assumptions

- Threat model
  - In-scope: software-based attacks: e.g., malware
  - Out-of-scope: physical hardware attacks on TPM

- Dual relationship with the cloud:
  - We trust cloud with cTPM shared root key
  - We do not trust the cloud with existent TPM root keys

- Future work: securing cloud-side of cTPM
SEM: TPM’s extensibility mechanism

- SEM: **Secure Execution Mode** (aka. Intel TXT)
  - Implements a CPU-based reboot
  - Run trusted code in an isolated environment

- Implement additional TPM functionality with SEM:
  - Step 1: Suspend OS, Enter SEM, Run additional trusted code
  - Step 2: Cleanup, Exit SEM, Resume OS

- Very challenging to use SEM
  - Performance overhead, engineering issues
  - Lack of support on mobile devices
  - No production software today uses SEM
Talk Outline

- Motivation
- Background & Design alternatives
- Design & Implementation
- Evaluation
- Conclusion
cTPM Design Challenges

- Provisioning pre-shared root keys in cTPM
- Secure key sharing across cTPMs
- Secure communication channel
- Cloud-side NV storage
- Trusted clock
Provision cTPM Root-of-Trust

- We provision each cTPM with a cloud seed:
  Unique random value pre-shared with cloud
  - iPhones, iPads share their seeds with iCloud
  - Microsoft Surfaces share their seeds with Azure

- On boot-up, cTPM deterministically generates two keys:
  - Cloud Root Key (CRK) protects all cross-device secrets
  - Cloud Communication Key (CCK) protects all communication
Secure Key Sharing

$h_1 = \text{TPM2_Load} \left( \{key\}_{CRK_1}, CRK_1 \right)$

$h_2 = \text{TPM2_Load} \left( \{key\}_{CRK_2}, CRK_2 \right)$

$\{key\}_{CRK_1} = \text{TPM2_Create}(CRK_1)$

$\{key\}_{CRK_2} = \text{TPM2_Create}(CRK_2)$
Communication Methodology

- Send/Recv = NV_Write/NV_Read
- Device-side cache masks transient connectivity loss
  - Uses secure synchronization protocol
Synchronization Protocol

- **Pull**

  - **Device-side Cache**
  - **Cloud-side NVRAM**

  
  ```
  @t_1 \{nonce, NV\_READ, NV\_Sel\}_{CCK}
  
  @t_2 \{nonce, RC\_SUCCESS, NVs\}_{CCK}
  
  Check t_2 - t_1 < Timeout, otherwise abort
  ```

- **Push**

  - **Device-side Cache**
  - **Cloud-side NVRAM**

  ```
  \{nonce, \text{ctr}, NV\_WRITE, NV\_Sel, NVs\}_{CCK}
  
  \{nonce, \text{ctr}, RC\_SUCCESS\}_{CCK}
  ```
TPM Usage Model

- TPM alone can’t establish connection with cloud

![Diagram showing TPM and Caller App with cmd and results connections, and a block other commands note]
Secure Asynchronous Communication

- Design asynchronous commands: better availability and responsiveness
- Establish secure channel out of untrusted entities

Untrusted Secure Asynchronous Communication

Untrusted

Trusted

Phase 1

Phase 2

Phase 3

Local Device

cTPM

Caller App

Cloud

blob = TPM2_Sync_Begin(RD, NV42)

blob' = TPM2_Sync_Poc(blob)

RET_CODE = TPM2_Sync_End(blob')
Cloud-backed NV Storage

- Local cached NV entries have TTLs
  - Once TTL expires, NV entry in the cache becomes invalid
  - Apps can re-sync with cloud to refresh TTL

- cTPM timeout can abort pending cloud commands
  - Cloud can adjust this timeout, default value is 5 minutes

- Trusted clock: “special” NV entry updated by cloud
  - Clock accuracy controlled by separate clock timeout
  - Default timeout value is 1 second
Implementation Details

- 3 new commands (TPM 2.0 has 108 commands)
  - TPM2.Sync_Begin()
  - TPM2.Sync_End()
  - TPM2.Sync_Proc()

- cTPM is based on TPM2.0
  - 1,304 lines of code (TPM 2.0 is 23,163 LoC)

- cTPM versions of:
  - Pasture, Kotla et al. [OSDI’12]
  - TrInc, Levin et al. [NSDI’09]
Talk Outline

- Motivation
- Background & Design alternatives
- Design & Implementation
- Evaluation
- Conclusion
Evalution Setup

Wide-area Network Emulator (NEWT)
Used 3G and Wi-Fi typical RTTs
(reported by J. Huang et al. [Mobisys’12])
Evaluation Questions

- Are the cTPM protocols secure?
- What is the performance of crypto operations?
Protocol Verification

- Verified correctness of sync. protocols with ProVerif

- Attacker model:
  - Attacker has unrestricted access to OS, applications & network

- We did not verify the correctness of implementation
Latency of RSA-2048 key creation

- Creating RSA-2048 key in the cloud is 12X faster than on local device
Conclusions

- cTPM: Shared cloud seed in TPM 2.0
  - Small design change to support cross-device scenarios

- Additional cTPM benefits:
  - High performance NV storage
  - Trusted clock

- Full implementation of cTPM in TPM 2.0
  - Implemented Pasture and TrInc on top of cTPM
Thank you

- Chen Chen (ETH Zurich): chen.chen@inf.ethz.ch
Backup #1: Related Work


Backup #2: cTPM Owner Change

- When the mobile device is stolen
  - Ask the cloud to stop providing service to the device

- When the mobile device is sold to a new owner
  - Re-keying:
    - Hardware method
      - The merchant use a special device for re-keying
    - Software method
      - Implement a re-key protocol in cTPM
Backup #3: Why not a design alternative

- A shared key with the cloud in current domains
  - Problem: no domain to use
    - Privacy domain is for endorsement key
    - Owner domain is cleared when the owner takes ownership
    - Platform domain is used by the manufacturer for testing
Dual relationship with the cloud

Securing the cloud is an active area of research:
- Emerging technology (Intel SGX) can help protect secrets in the cloud
Semantics of Pasture with cTPM are complex

- **Unmodified Pasture** with multiple devices:
  - Hard to keep Pasture protocols in lockstep across multiple devices

- **cTPM Pasture** with multiple devices:
  - Decisions made on one device are automatically synced to all other devices
Backup #6: TrInc with cTPM

- TrInc requires a counter in NVRAM to solve equivocation problem for distributed system
  - Each counter in NVRAM only supports ~10k writes

- TrInc with cTPM
  - Provides high performance counter in “remote” NVRAM
  - Offers unlimited number of writes
Backup #7: TPM 2.0

- Offer Algorithm Agility
- Offer three control domains
- TPM 2.0 specification == reference implementation

Crypto Primitives: SHA1, SHA256, RSA, ECC, RC4, ... ...
Latency of cloud-side NVRAM accesses

- Cloud-side NV write is **3.5X faster** than local-side