Open-TEE – A Virtual Trusted Execution Environment

Intel Collaborative Research Institute – Security (ICRI-SC)

<u>Brian McGillion</u>, Tanel Dettenborn, <u>N. Asokan</u>
Thomas Nyman, Atte Pelikka, Ville Kankainen

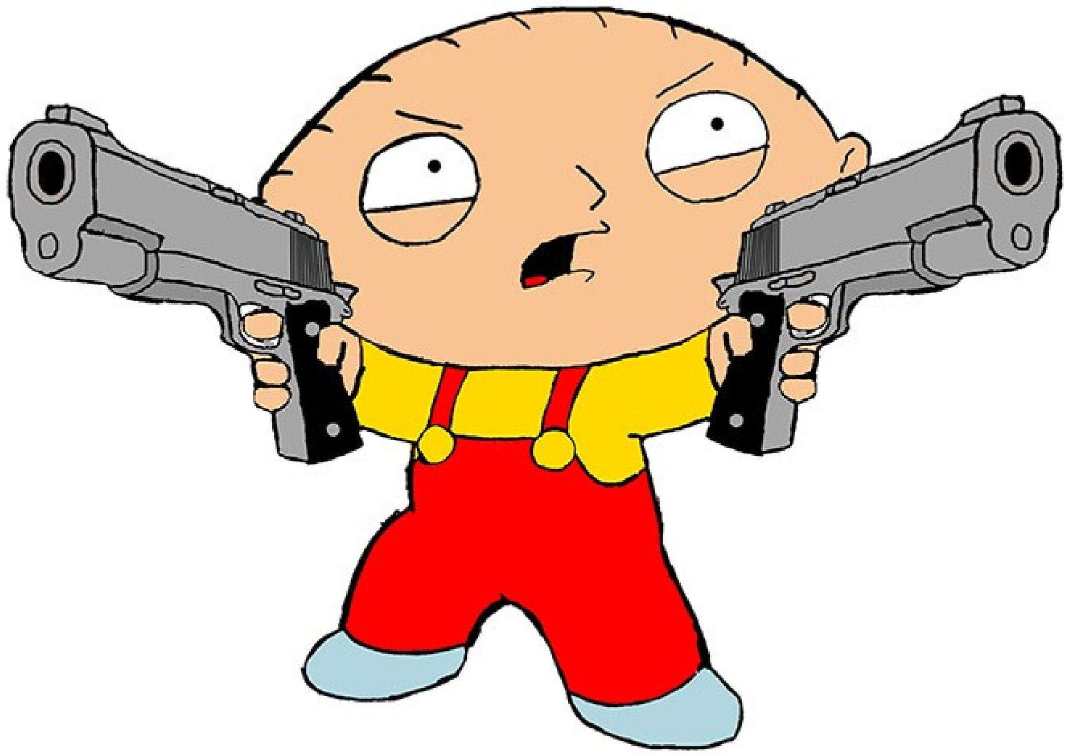
What is Trust?

- Trusted Execution Environment (TEE)
- Trusted Application (TA)
- Trusted Platform Module (TPM)
- Trusted Computing Group (TCG)

- “An entity can be trusted if it always behaves in the expected manner for the intended purpose” [1]
  - Has the system been modified?
  - Does the system securely store its secrets?
  - Is the system who it claims to be?

[1] TCG slides 2006

Who is the attacker?



- Traditional hacker
- Malware writer
- Con-artist – identity theft
- Plain thief – all about the money

Who is the attacker contd?

- Owner of the device
- Non malicious intent
- “improve the service” - modify the settings
- Share DRM content

The difference between genius and stupidity is that genius has it's limits

Surely a digital signature is enough!

- Generate a key pair
- Sign the kernel, initrd, root file system
- Deploy the kernel, initrd, rootfs, public key to the device
- Store private part on usb that never leaves the office
- We are done …
- What is our root of trust?
  - The Key Pair

Defining the trust anchor

- Immutable
- Provisioned during manufacture
- Reliable
- Extremely hard to work around

Deriving Trust

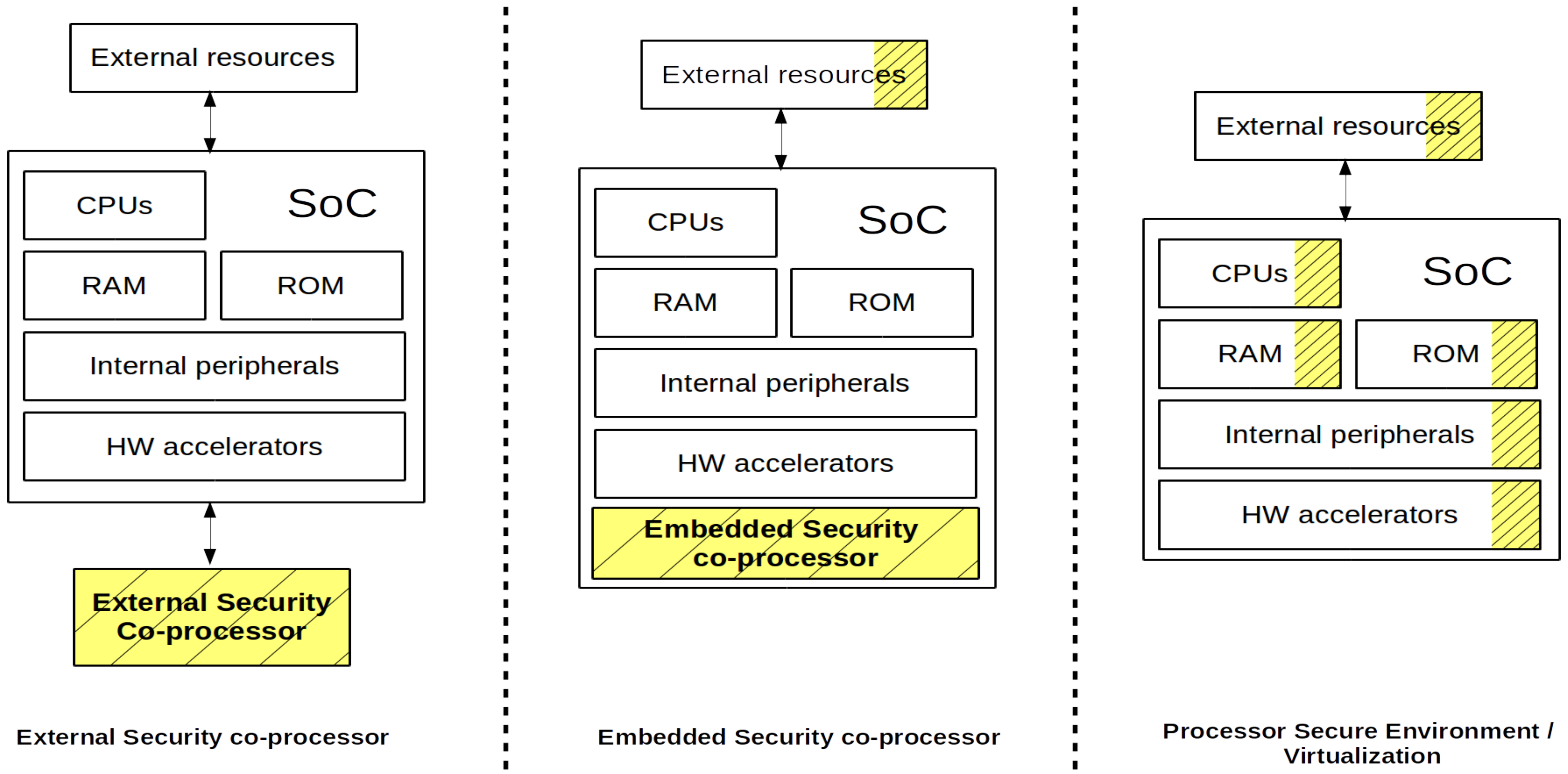
- Key hash burned into fuses at manufacture (32 bytes (sha256) vs 256 or 512 Bytes (2k or 4k RSA))
- Jump into ROM code
- Verify the Firmware / Bootloader
- Launch FW / Bootloader
- Configure System e.g. memory map, peripherals, co-processors, policy
- Verify kernel, initrd, public key
- Launch Kernel
- Verify the filesystem and applications ...

Why do we need a TEE

- Isolated Execution
- Secure storage
- System configuration management and identification
- Hardware backed protection
- Minimized Trusted Computing Base (TCB)
  - Code size and complexity are the nemesis of security
- Defense in depth – Layers (YOUTUBE Link)

Trusted platforms not limited to mobile

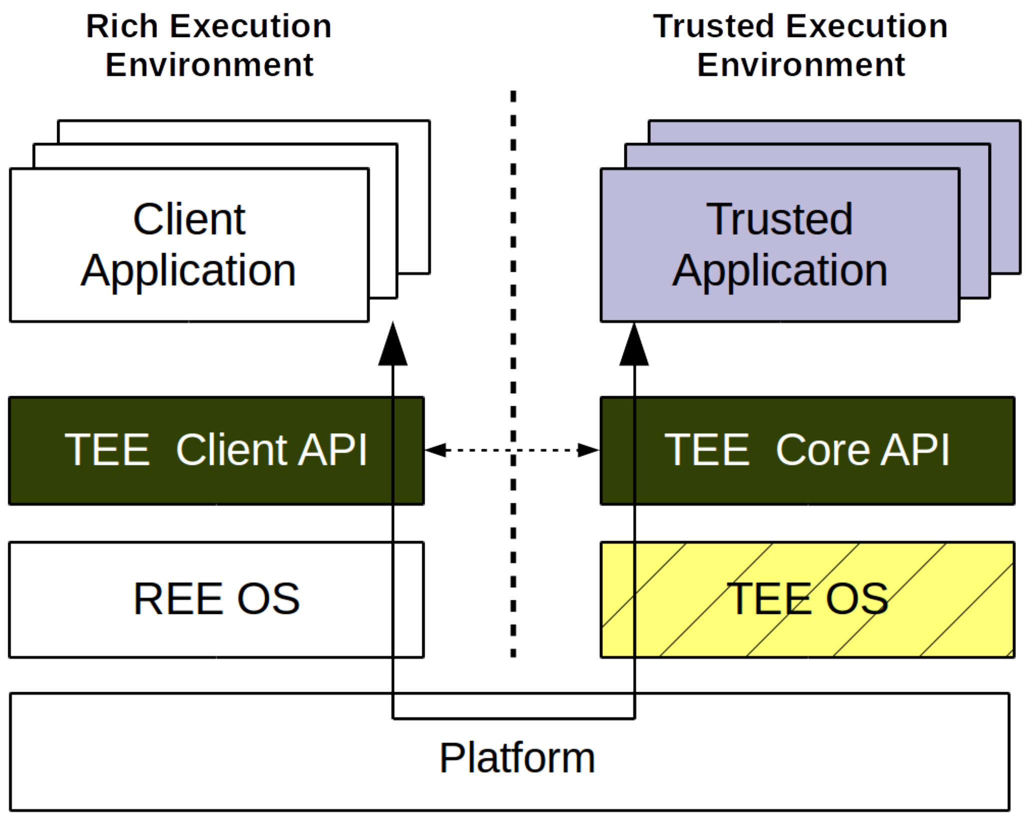
- Smart Card
- Hardware Security Module (HSM)
- Software Guard Extensions (SGX)
- Intel TXT
  - Virtual Machine Monitor (VMM, aka hypervisor, e.g. xen)
    - VT-x (Virtualized Execution)
    - VT-d (directed IO protection), isolation of peripherals
    - Enforced memory separations, in addition to paging and ring separations
    - Extended Page Tables (EPT)



Adapted from http://www.globalplatform.org/specificationsdevice.asp

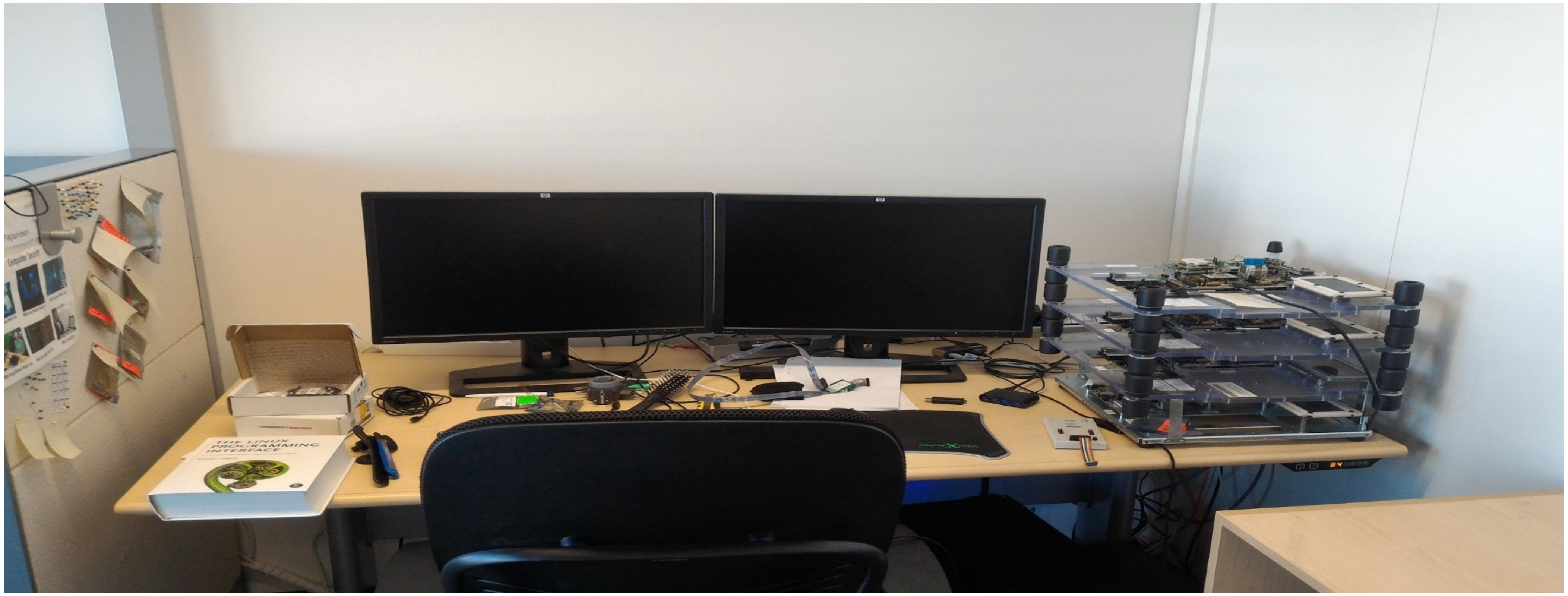
Standardizing TEE interfaces

- Incompatible – same HW
  - TEE OS differs
  - REE drivers and userspace API differ
  - Manufacturers / developers require cross-platform
- GlobalPlatform - Device Spec
  - TEE Client API
  - TEE Core API
  - Trusted UI
  - TEE debug
  - ...



Adapted from http://www.globalplatform.org/specificationsdevice.asp

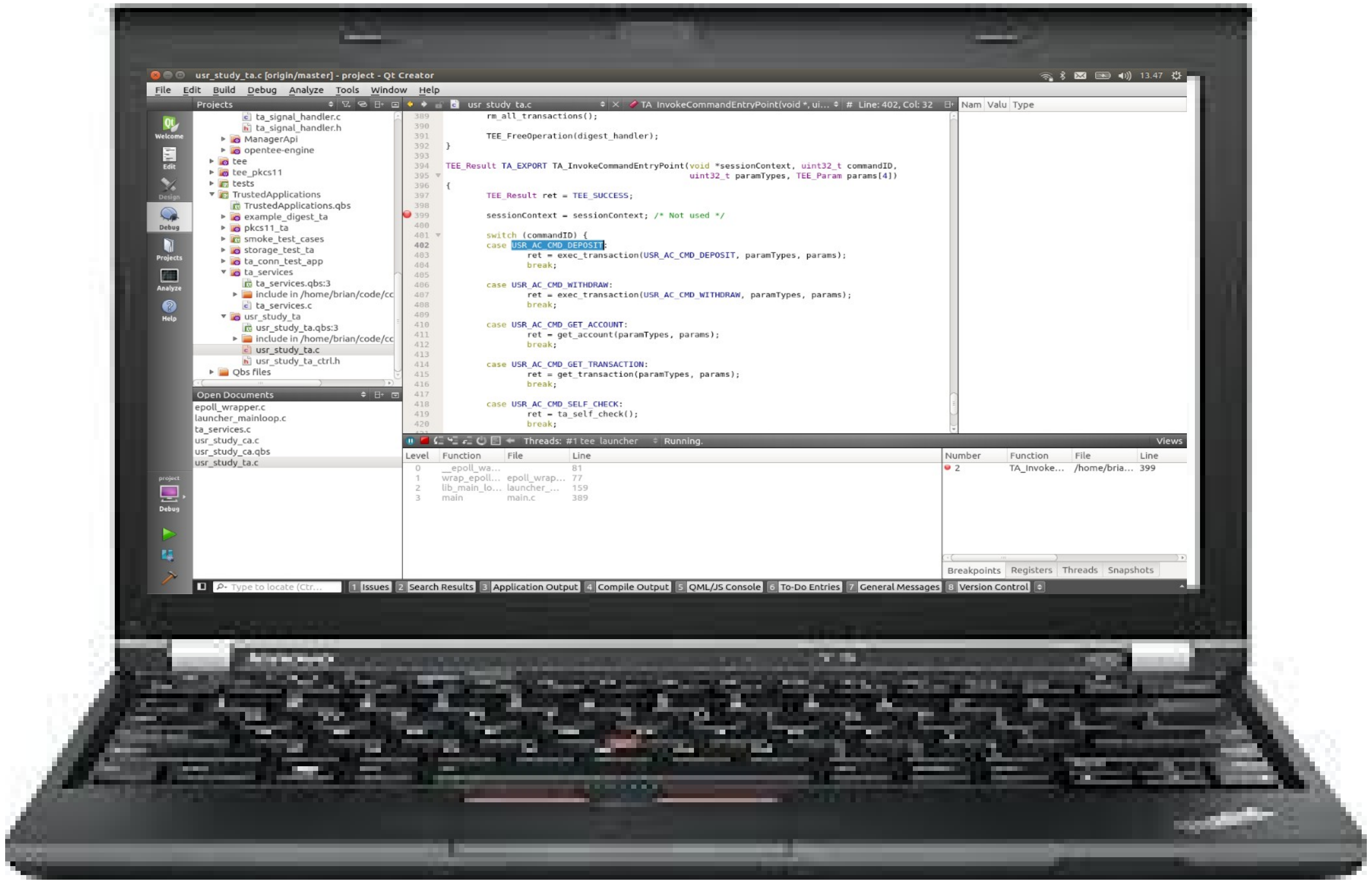
TA Development Today!



TA Development Today!

- “slow execution (flash, download, reboot, run)”
- “debugging TA is slow, you need to cross compile and push binary into target hardware”
- “TEE itself might not work without problems, because some changes have been made”
- “Main difficulty is that you need development hardware, which is problematic when working outside the office.”

Future TA Development Environment?


usr_study_ta.c [origin/master] - project - Qt Creator
File Edit Build Debug Analyze Tools Window Help
Projects
Open Documents
Threads: #1 tee_launcher
Running.
Breakpoints Registers Threads Snapshots
Issues Search Results Application Output Compile Output QML/JS Console To-Do Entries General Messages Version Control

```c
        rm_all_transactions();

        TEE_FreeOperation(digest_handler);
}

TEE_Result TA_EXPORT TA_InvokeCommandEntryPoint(void *sessionContext, uint32_t commandID,
                                                 uint32_t paramTypes, TEE_Param params[4])
{
        TEE_Result ret = TEE_SUCCESS;

        sessionContext = sessionContext; /* Not used */

        switch (commandID) {
        case USR_AC_CMD_DEPOSIT:
                ret = exec_transaction(USR_AC_CMD_DEPOSIT, paramTypes, params);
                break;

        case USR_AC_CMD_WITHDRAW:
                ret = exec_transaction(USR_AC_CMD_WITHDRAW, paramTypes, params);
                break;

        case USR_AC_CMD_GET_ACCOUNT:
                ret = get_account(paramTypes, params);
                break;

        case USR_AC_CMD_GET_TRANSACTION:
                ret = get_transaction(paramTypes, params);
                break;

        case USR_AC_CMD_SELF_CHECK:
                ret = ta_self_check();
                break;
```

Level	Function	File	Line
0	__epoll_wa...		81
1	wrap_epoll...	epoll_wrap...	77
2	lib_main_lo...	launcher_...	159
3	main	main.c	389

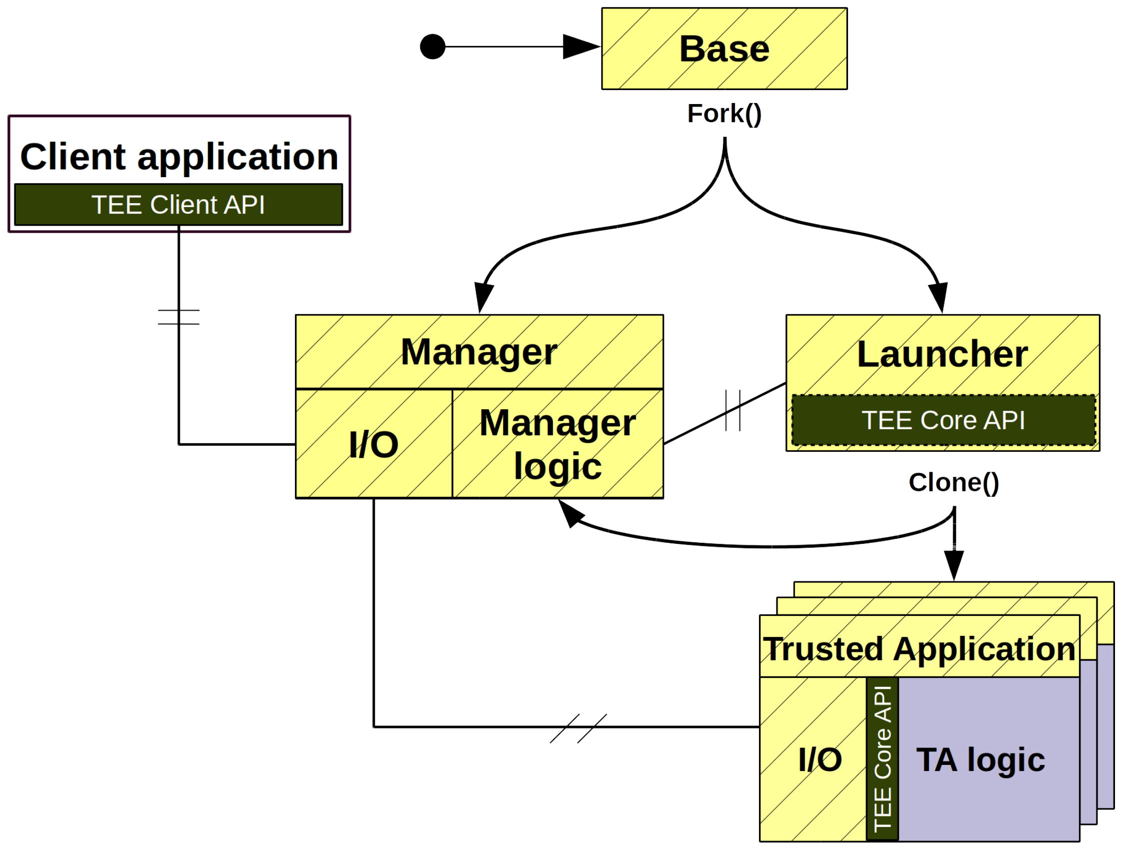
Number	Function	File	Line
● 2	TA_Invoke...	/home/bria...	399

Open-TEE

- Motivation
  - Enable developer access
  - Fast efficient prototyping environment
  - Promote research into TEE services
  - Promote Community involvement

- Requirements
  - Compliance
  - Hardware-independence
  - Reasonable performance
  - Ease-of-use !!

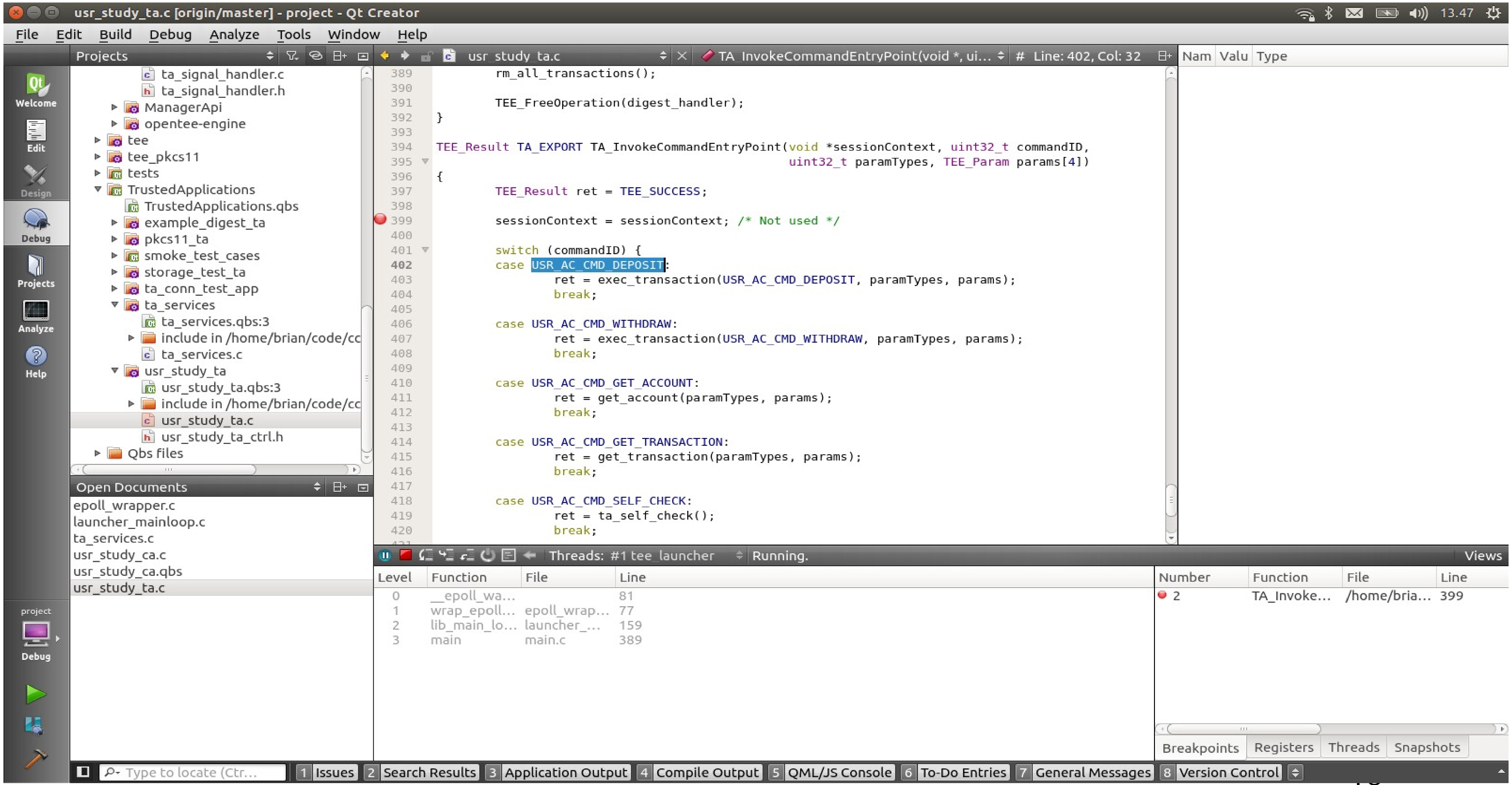
Architecture



Implementation and Tooling

- Utilize existing functionality
  - Well tested
  - Familiar to developers
  - Easy to setup/configure
- Open-TEE in use
  - Inconspicuous
  - Abstracts implementation details
  - Community involvement

- Development Process
  - Open source
  - Github, GerritHub
  - Coverity
  - Jenkins
  - Extensive testing (devs, researchers hard to please :)



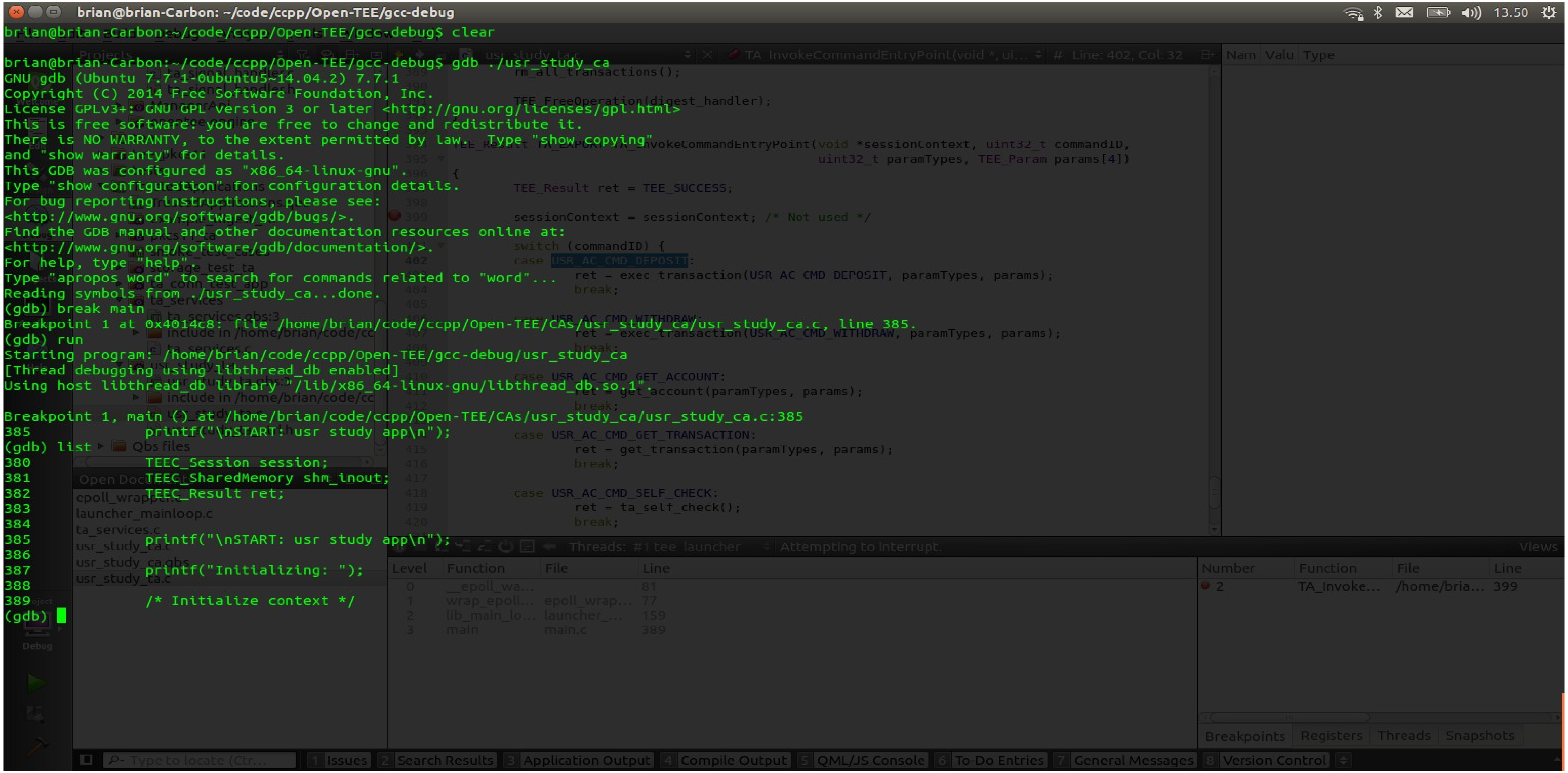
```
ta_signal_handler.c
ta_signal_handler.h
ManagerApi
opentee-engine
tee
tee_pkcs11
tests
TrustedApplications
  TrustedApplications.qbs
  example_digest_ta
  pkcs11_ta
  smoke_test_cases
  storage_test_ta
  ta_conn_test_app
  ta_services
    ta_services.qbs:3
    include in /home/brian/code/cc
    ta_services.c
  usr_study_ta
    usr_study_ta.qbs:3
    include in /home/brian/code/cc
    usr_study_ta.c
    usr_study_ta_ctrl.h
Qbs files
```

```
epoll_wrapper.c
launcher_mainloop.c
ta_services.c
usr_study_ca.c
usr_study_ca.qbs
usr_study_ta.c
```

```c
        rm_all_transactions();

        TEE_FreeOperation(digest_handler);
}

TEE_Result TA_EXPORT TA_InvokeCommandEntryPoint(void *sessionContext, uint32_t commandID,
                                                uint32_t paramTypes, TEE_Param params[4])
{
        TEE_Result ret = TEE_SUCCESS;

        sessionContext = sessionContext; /* Not used */

        switch (commandID) {
        case USR_AC_CMD_DEPOSIT:
                ret = exec_transaction(USR_AC_CMD_DEPOSIT, paramTypes, params);
                break;

        case USR_AC_CMD_WITHDRAW:
                ret = exec_transaction(USR_AC_CMD_WITHDRAW, paramTypes, params);
                break;

        case USR_AC_CMD_GET_ACCOUNT:
                ret = get_account(paramTypes, params);
                break;

        case USR_AC_CMD_GET_TRANSACTION:
                ret = get_transaction(paramTypes, params);
                break;

        case USR_AC_CMD_SELF_CHECK:
                ret = ta_self_check();
                break;
```

Level	Function	File	Line
0	__epoll_wa...		81
1	wrap_epoll...	epoll_wrap...	77
2	lib_main_lo...	launcher_...	159
3	main	main.c	389

Number	Function	File	Line
● 2	TA_Invoke...	/home/bria...	399



```
brian@brian-Carbon:~/code/ccpp/Open-TEE/gcc-debug$ clear

brian@brian-Carbon:~/code/ccpp/Open-TEE/gcc-debug$ gdb ./usr_study_ca
GNU gdb (Ubuntu 7.7.1-0ubuntu5~14.04.2) 7.7.1
Copyright (C) 2014 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./usr_study_ca...done.
(gdb) break main
Breakpoint 1 at 0x4014c8: file /home/brian/code/ccpp/Open-TEE/CAs/usr_study_ca/usr_study_ca.c, line 385.
(gdb) run
Starting program: /home/brian/code/ccpp/Open-TEE/gcc-debug/usr_study_ca
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".

Breakpoint 1, main () at /home/brian/code/ccpp/Open-TEE/CAs/usr_study_ca/usr_study_ca.c:385
385             printf("\nSTART: usr study app\n");
(gdb) list
380             TEEC_Session session;
381             TEEC_SharedMemory shm_inout;
382             TEEC_Result ret;
383
384
385             printf("\nSTART: usr study app\n");
386
387             printf("Initializing: ");
388
389             /* Initialize context */
(gdb) 
```

Install and configure Open-TEE

- http://open-tee.github.io/documentation/#quick-setup-guide
- Setup QtCreator
- http://open-tee.github.io/tutorial/qtcreator

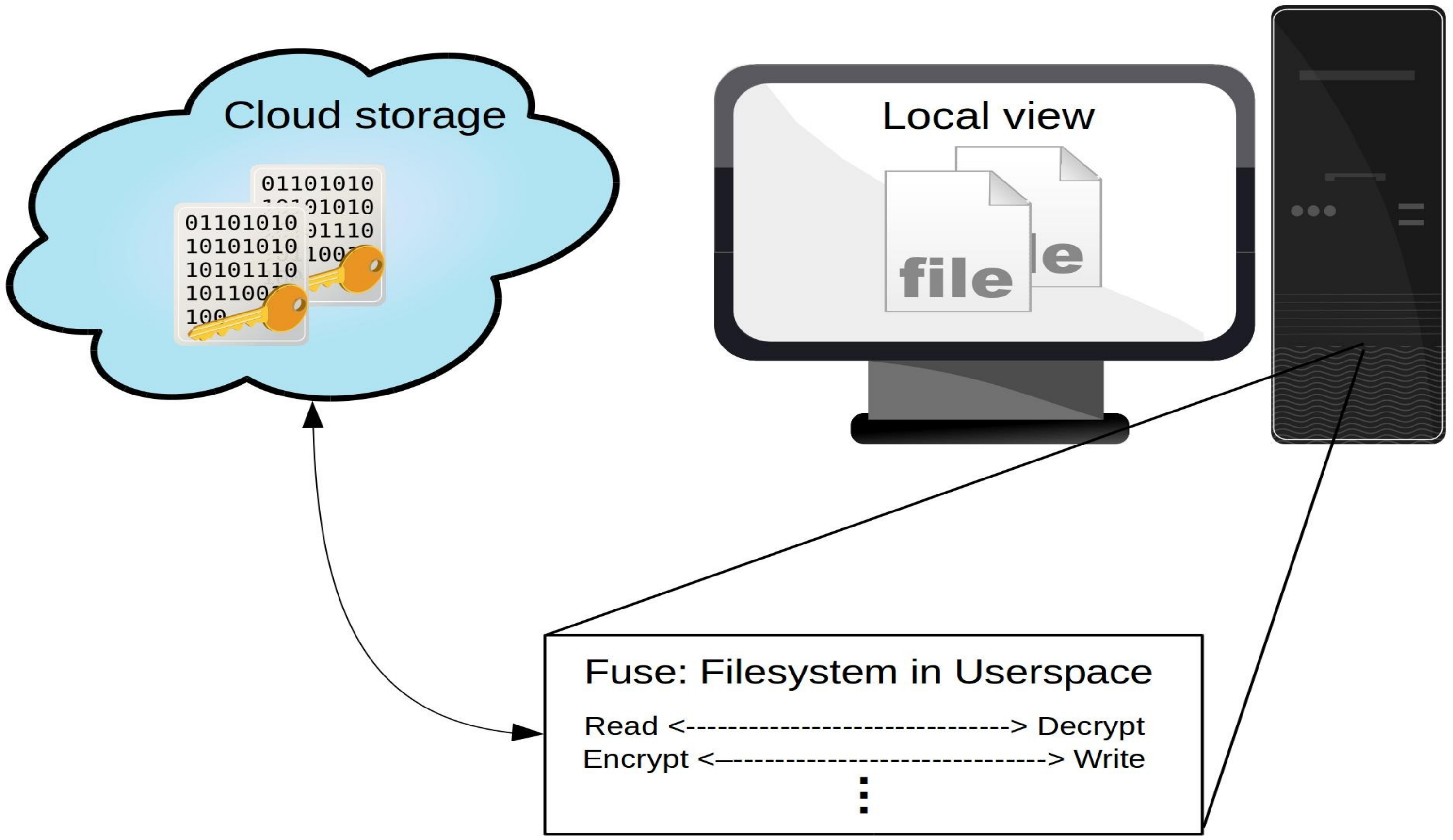
Omnishare

Omnishare is a scheme to allow client-side encryption with high entropy keys for public cloud services such as dropbox and google drive. In addition it defines an intuitive key distribution mechanism enabling data access from multiple devices.

For the purposes of this workshop we will look at enabling the key protection, encryption and decryption using a GP compliant TEE.

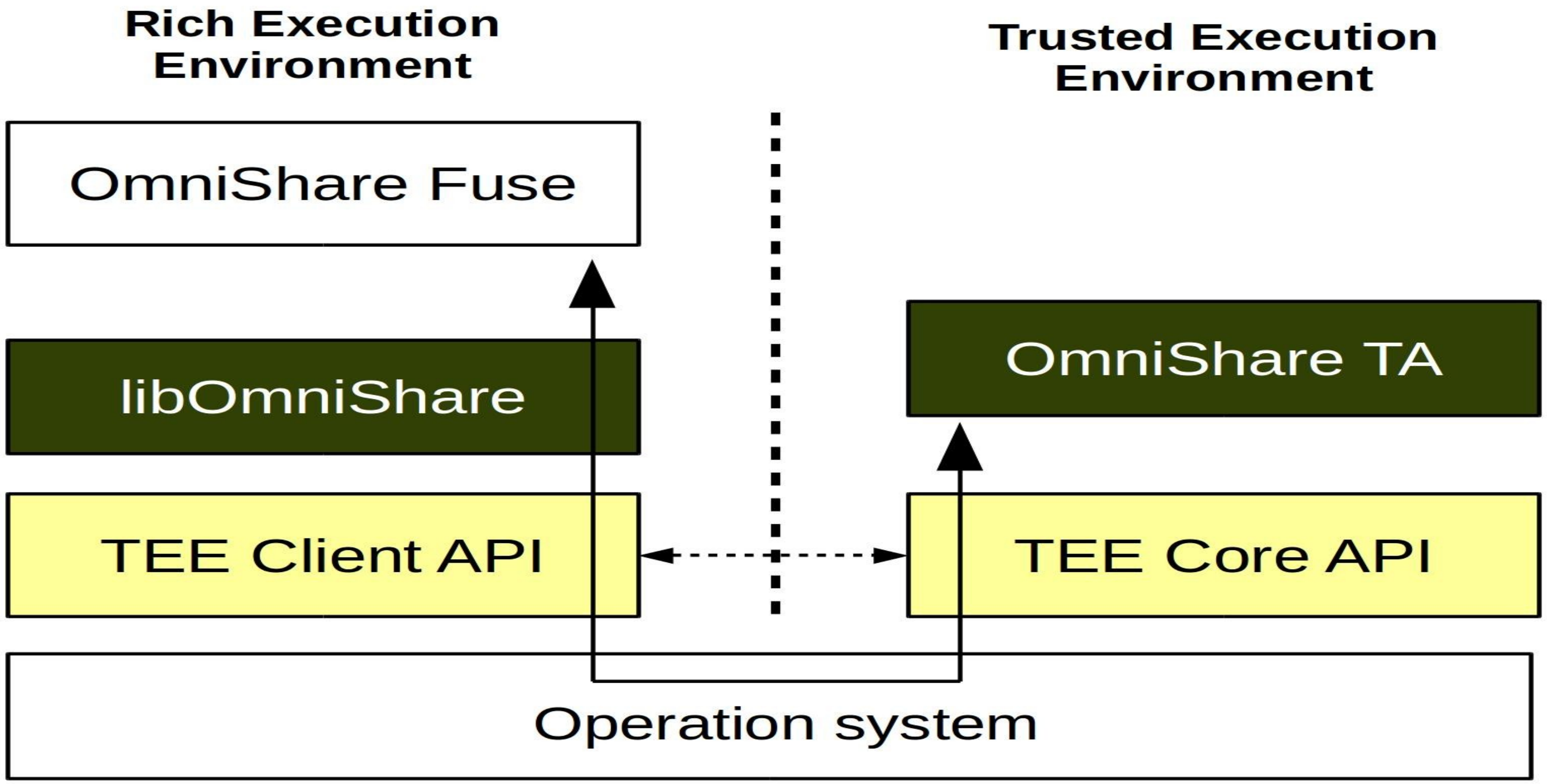
https://git.ssg.aalto.fi/close/OmniShare/blob/master/README.md

File System in User Space (FUSE)



Cloud storage
01101010
10101010
10101110
1011001
100
Local view
file
Fuse: Filesystem in Userspace
Read <------------------------------> Decrypt
Encrypt <------------------------------> Write
⋮

Omnishare Stack



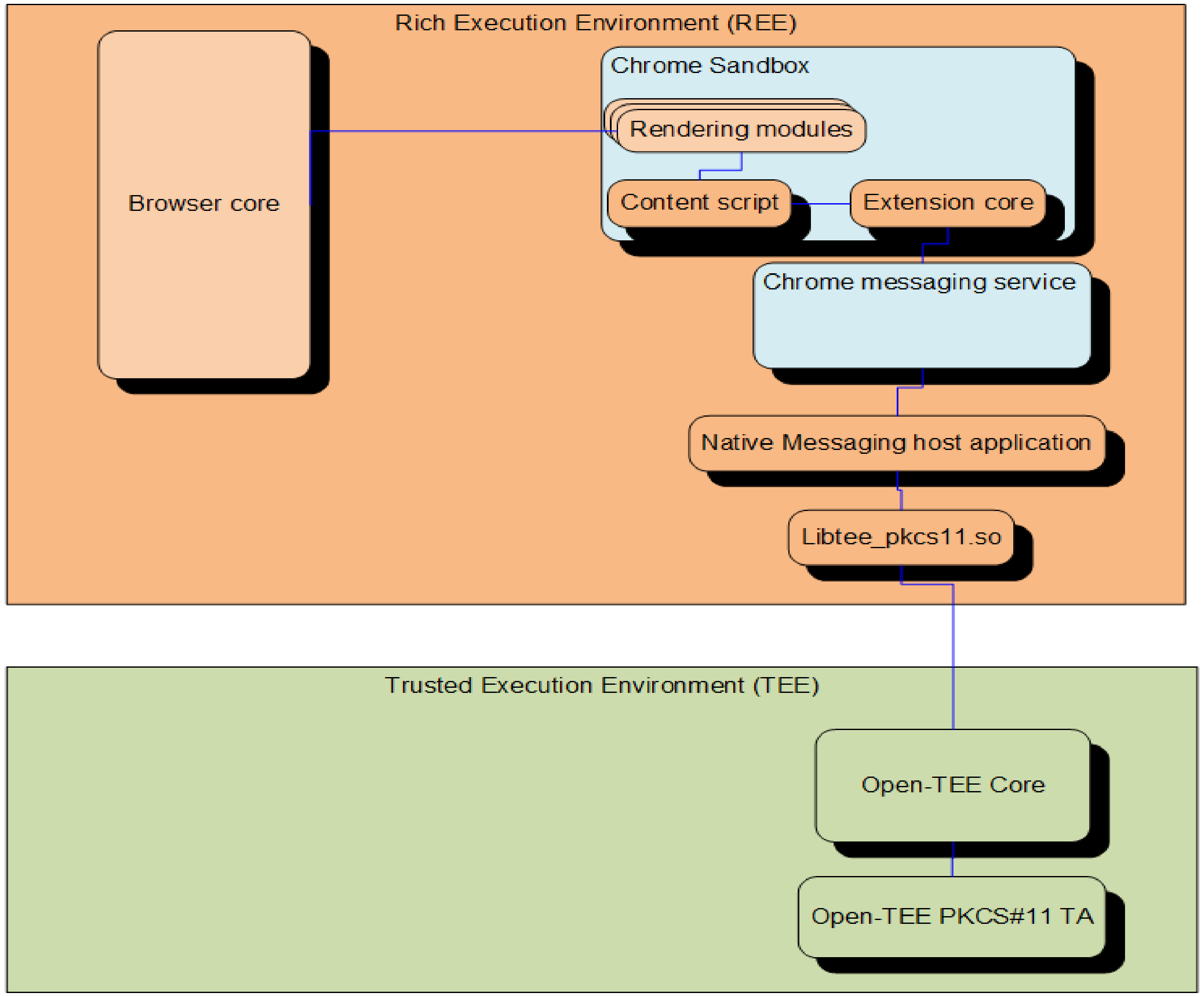
Demo Omnishare

- cd Open-TEE
- opentee start
- CAs/omnishare_fuse/omnishare start
- CAs/omnishare_fuse/omnishare copy
- less /tmp/cloud_store/test_file.c
  - "/tmp/cloud_store/test_file.c" may be a binary file. See it anyway?
- less /tmp/local_view/test_file.c
- Cloud store e.g. Dropbox

PKCS#11

- Secondary device login (using phone to unlock laptop, desktop)
- Remote authentication (ssh, VPN, website with client certificate)
- Mobile payments / Banking / Transport cards
- Cryptography (Disk encryption, Android keymaster)
- Personal Data security in public (Files, health … Dropbox, Box, …)
- Storing biometric data (iris, finger print …)
- Application specific secure storage (personal devices, data centers)
- Locks (home, car, office)
- E-Mail Encryption and Signing

Chrome Extension



Demo Gmail Encryption

- Cherry pick - https://review.gerrithub.io/#/c/261626/
- Rebuild and restart Open-TEE
- Open-TEE/chrome/INSTALL

Hands On

- Download GP Core API and Client API
  - http://www.globalplatform.org/specificationsdevice.asp
- User study
  - cd Open-TEE
  - opentee start
  - gcc-debug/usr_study_ca
  - gdb attach `pgrep tee_launcher`
  - break TA_InvokeCommandEntryPoint
  - set follow-fork-mode child
  - C
  - C

Hands On contd

- Hash example
  - checkout the oxford branch
    - cd Open-TEE/TAs/
    - git checkout -b oxford -t origin/oxford
    - edit example_digest_ta/example_digest_ta.c
  - Fix the TODOs compile and run

Looking for a Group Project / M.Sc thesis ?

- Harden Open-TEE
  - Deployable on systems as a true TEE
  - Same system can potentially be deployed outside mobile
  - On Making Trusted Execution Environments available
- SGX
  - move the trusted portions of the system to an SGX enclave.
    - Open-SGX
      - https://github.com/sslab-gatech/opensgx
      - https://taesoo.gtisc.gatech.edu/pubs/2016/opensgx/opensgx.pdf
- Hypervisor
  - move the manager to an isolated VM or embed the functionality in a VMM

Thank You

http://open-tee.github.io/

intel®

Security

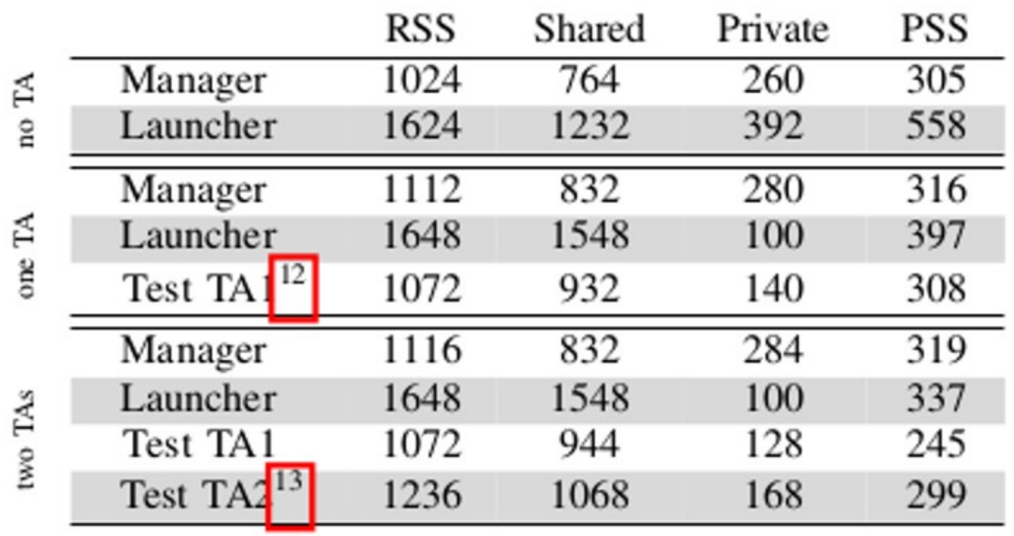
Backup Slides

Evaluation

- “Compliance”
  - 100% function coverage, ~80% algorithm coverage
- Hardware independence
  - Servers, desktop, laptop, tablet, phone
  - Linux and Android support
  - x86 and ARM
  - TAs developed in Open-TEE have been compiled and run in production TEE e.g. Trustonic's

Footprint and Performance

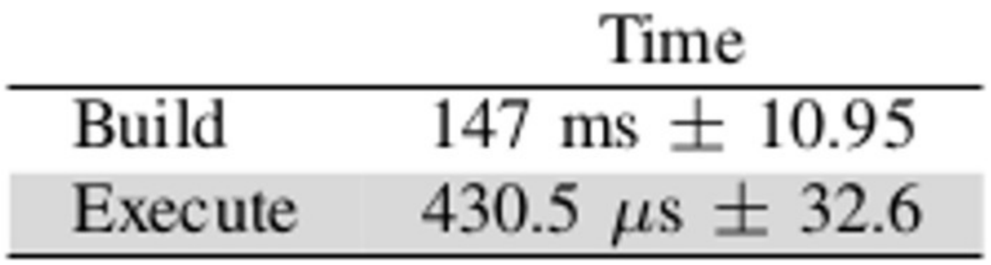
- System Impact (KB)



		RSS	Shared	Private	PSS
no TA	Manager	1024	764	260	305
	Launcher	1624	1232	392	558
one TA	Manager	1112	832	280	316
	Launcher	1648	1548	100	397
	Test TA1[12]	1072	932	140	308
two TAs	Manager	1116	832	284	319
	Launcher	1648	1548	100	337
	Test TA1	1072	944	128	245
	Test TA2[13]	1236	1068	168	299

12 – ta_conn_test_app ~100 lines of c
13 – example_digest_ta ~140 lines of c

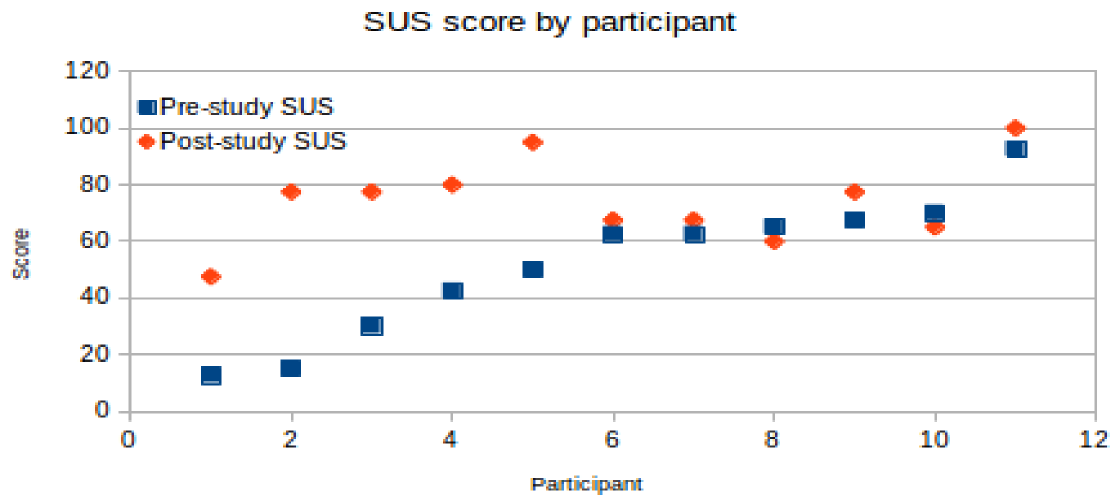
- Developer Impact



	Time
Build	147 ms $\pm$ 10.95
Execute	430.5 μs $\pm$ 32.6

Ease of Use

- Standard System Usability Scale (SUS)
  - 14 testers
    - 3 → 33 years experience
    - 11 with TEE experience
  - Pre-Study
    - Understand their current environment
  - Post-Study
    - Install, build, deploy, debug



- Wilcoxon signed-rank test showed that the difference in SUS scores is statistically significant (z = −2.50, p < .05, r = −0.53)
- 9/11 (82%) devs rated Open-TEE higher than current environment

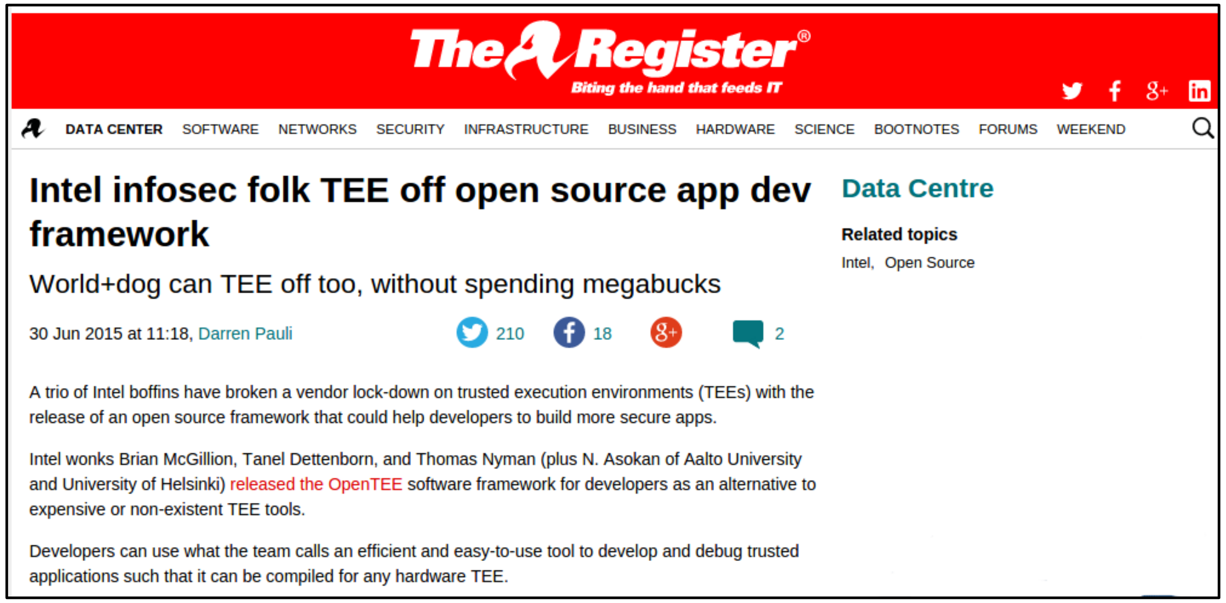
Post-Study free-form comments

- “Debugging is easy”
- “debugging is fast”
- “[Open-TEE] complements nicely my previous SDE - first preliminary testing with Open-TEE & gdb & OT_LOG(..), and only after that ARM cross compiler & FVP emulation”

Recap

- Compliant to GP specifications
- Hardware independent
- Easy to use
  - Minimal system requirements
  - Minimal developer impact
- Promote research

The Aftermath


The Register
Biting the hand that feeds IT
DATA CENTER SOFTWARE NETWORKS SECURITY INFRASTRUCTURE BUSINESS HARDWARE SCIENCE BOOTNOTES FORUMS WEEKEND

Intel infosec folk TEE off open source app dev framework

World+dog can TEE off too, without spending megabucks

30 Jun 2015 at 11:18, Darren Pauli 210 18 2

A trio of Intel boffins have broken a vendor lock-down on trusted execution environments (TEEs) with the release of an open source framework that could help developers to build more secure apps.

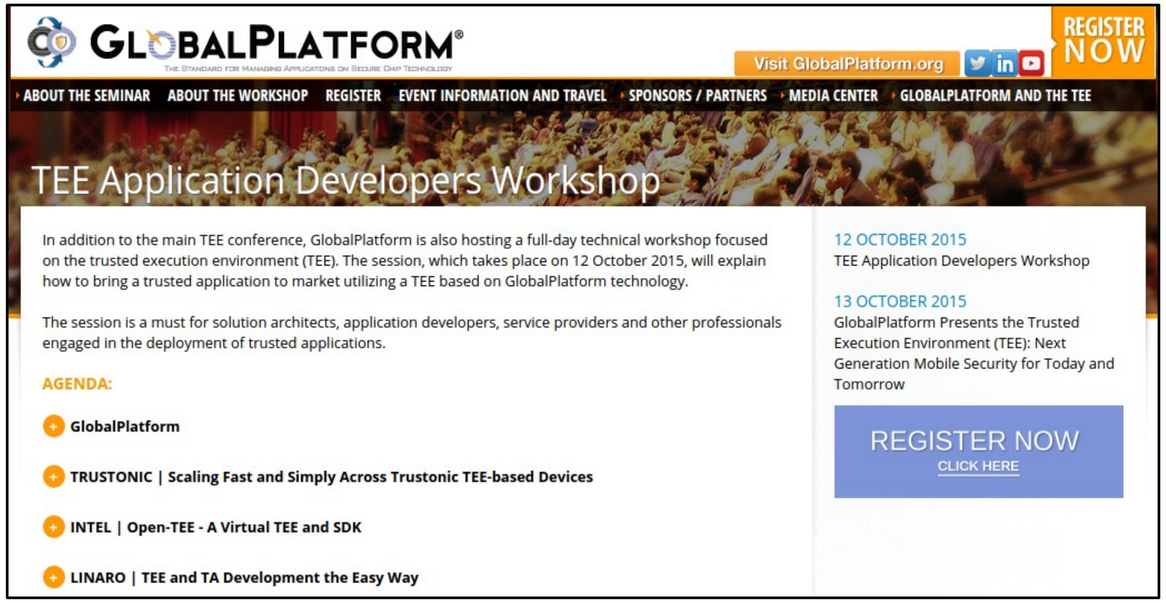
Intel wonks Brian McGillion, Tanel Dettenborn, and Thomas Nyman (plus N. Asokan of Aalto University and University of Helsinki) released the OpenTEE software framework for developers as an alternative to expensive or non-existent TEE tools.

Developers can use what the team calls an efficient and easy-to-use tool to develop and debug trusted applications such that it can be compiled for any hardware TEE.

Data Centre

Related topics

Intel, Open Source


GLOBALPLATFORM
Visit GlobalPlatform.org
REGISTER NOW
ABOUT THE SEMINAR ABOUT THE WORKSHOP REGISTER EVENT INFORMATION AND TRAVEL SPONSORS / PARTNERS MEDIA CENTER GLOBALPLATFORM AND THE TEE

TEE Application Developers Workshop

In addition to the main TEE conference, GlobalPlatform is also hosting a full-day technical workshop focused on the trusted execution environment (TEE). The session, which takes place on 12 October 2015, will explain how to bring a trusted application to market utilizing a TEE based on GlobalPlatform technology.

The session is a must for solution architects, application developers, service providers and other professionals engaged in the deployment of trusted applications.

AGENDA:

- GlobalPlatform
- TRUSTONIC | Scaling Fast and Simply Across Trustonic TEE-based Devices
- INTEL | Open-TEE - A Virtual TEE and SDK
- LINARO | TEE and TA Development the Easy Way

12 OCTOBER 2015
TEE Application Developers Workshop

13 OCTOBER 2015
GlobalPlatform Presents the Trusted Execution Environment (TEE): Next Generation Mobile Security for Today and Tomorrow

REGISTER NOW
CLICK HERE

http://www.globalplatform.org/TEEevent/about_the_workshop.asp

http://www.theregister.co.uk/2015/06/30/opentee_an_open_virtual_trusted_execution_environment/