On the Creation of a Secure Key Enclave via the Use of Memory Isolation in Systems Management Mode

Dr James Sutherland
Dr Natalie Coull
Dr Ian Ferguson
ian.ferguson@abertay.ac.uk

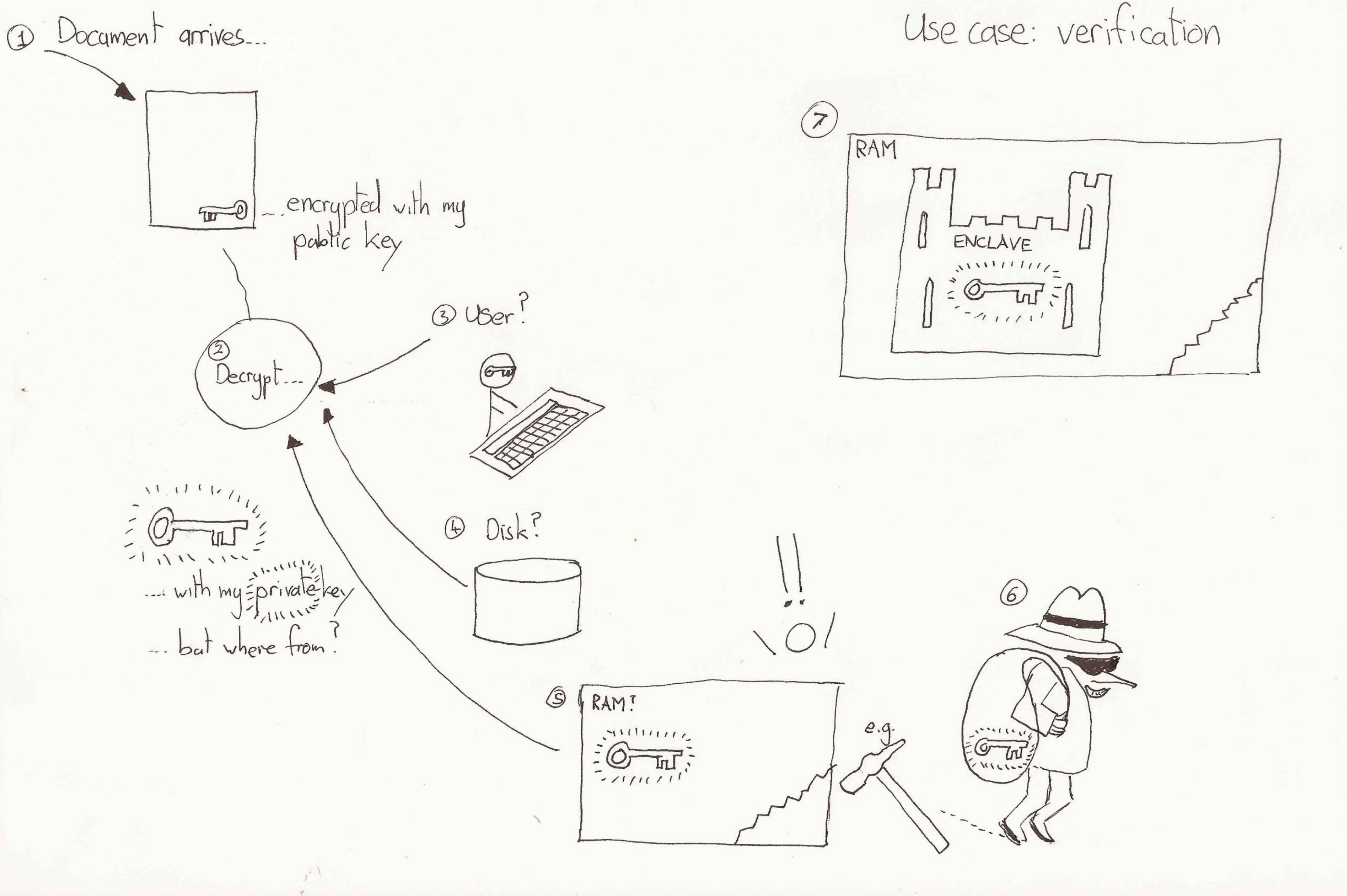
Abertay University

Intro

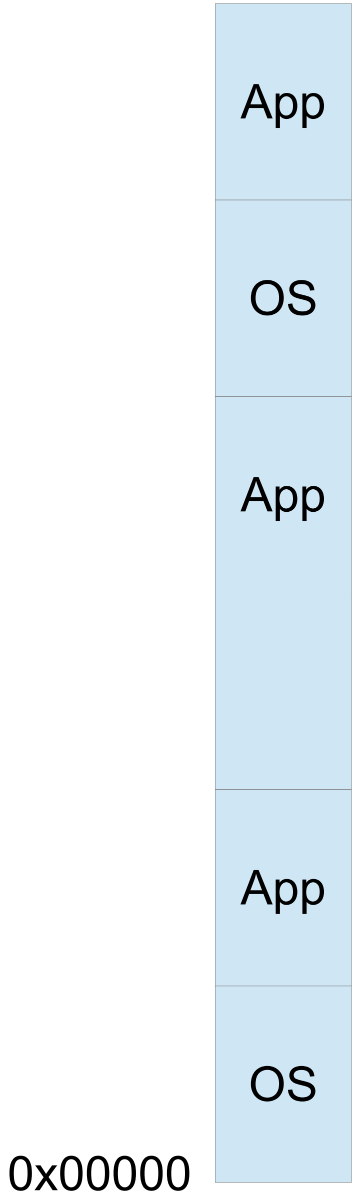
- What have we done?
  - Built a secure key-store using only commodity hardware and the existing facilities of the X86 architecture.
  - Evaluated it's functionality, security and performance.
- Talk outline
  - Problem
  - SMM
  - Experimental evaluation

Problem:

- Keeping crypto-keys safe whilst they are in RAM being used



Use case: verification
① Document arrives...
...encrypted with my public key
② Decrypt...
③ User?
④ Disk?
... with my private key
... but where from?
⑤ RAM?
e.g.
⑥
⑦
RAM
ENCLAVE



Paged Virtual Memory System

- Pages are 'randomly' intermingled.

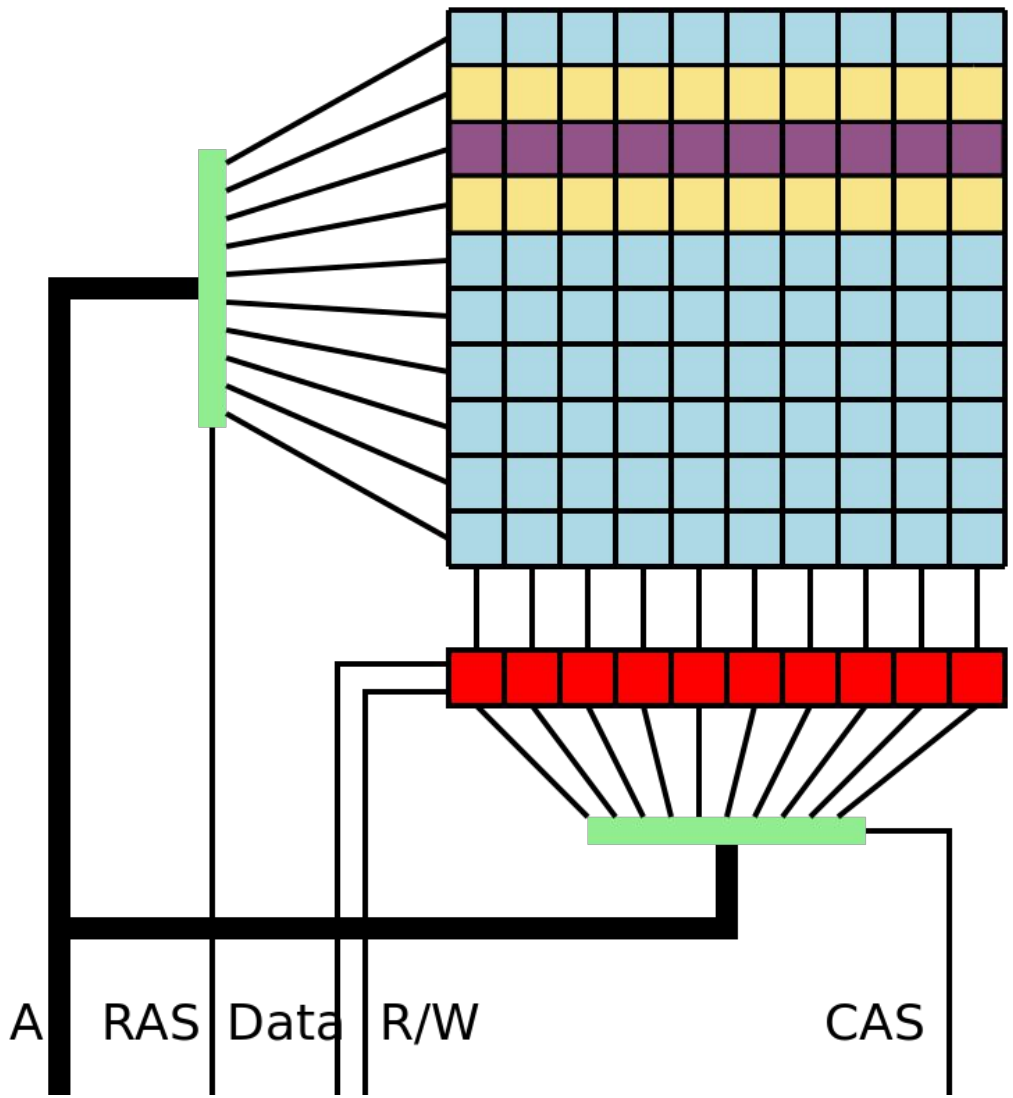
- **Should** be protected by the virtual memory system.
  - A process **should** not be able to access a page it doesn't own.....

  ...but.....

- RowHammer (for example)

Motivation

- RowHammer etc.
  - Unexpected interaction between physically proximate memory components – allowed access to 'local' page
  - Privilege escalation due to sensitive system (virtual) memory pages being intermingled with low-privilege pages.
  - Virtual Machines/hypervisors
- Encryption keys stored in RAM....vulnerable



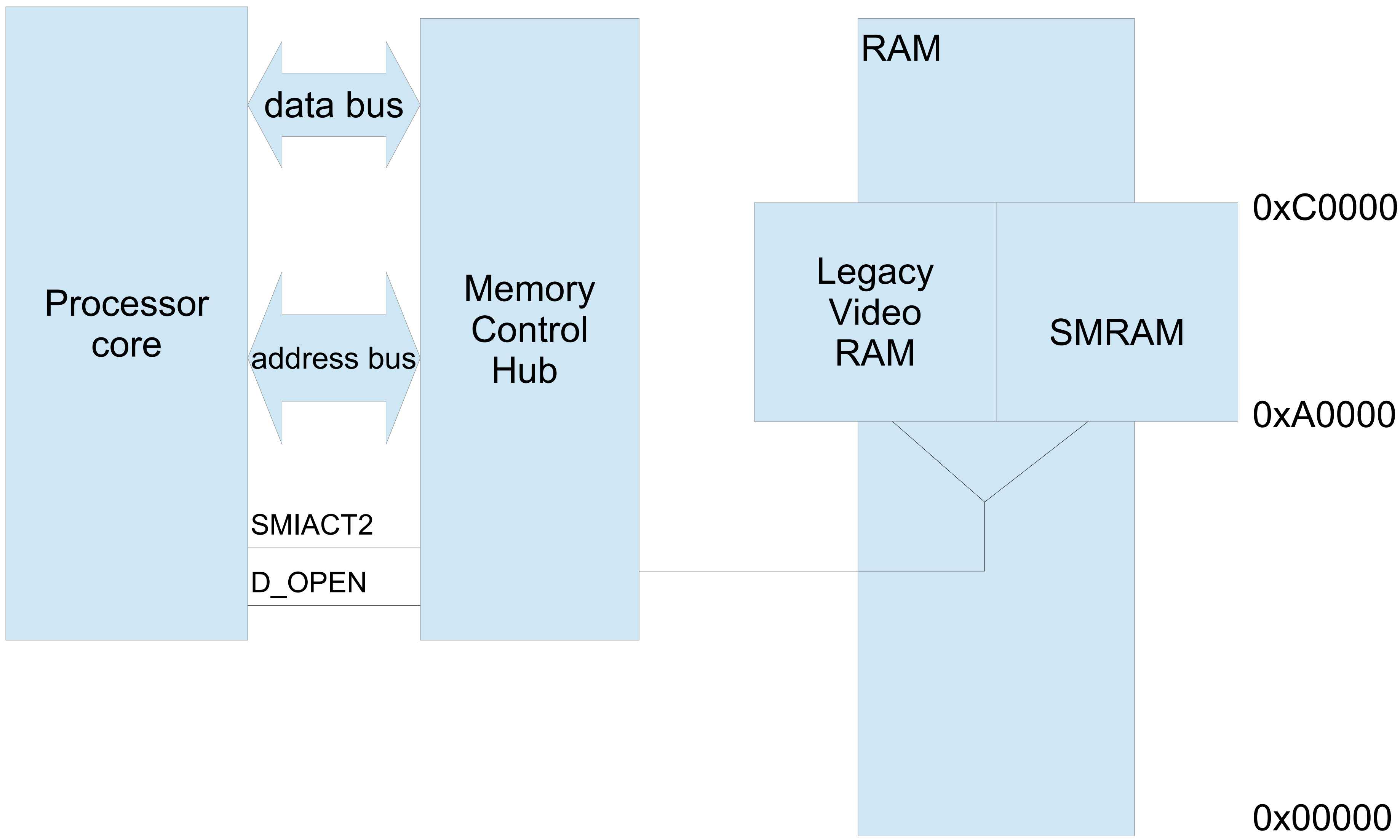
Abertay University

Existing Approaches....

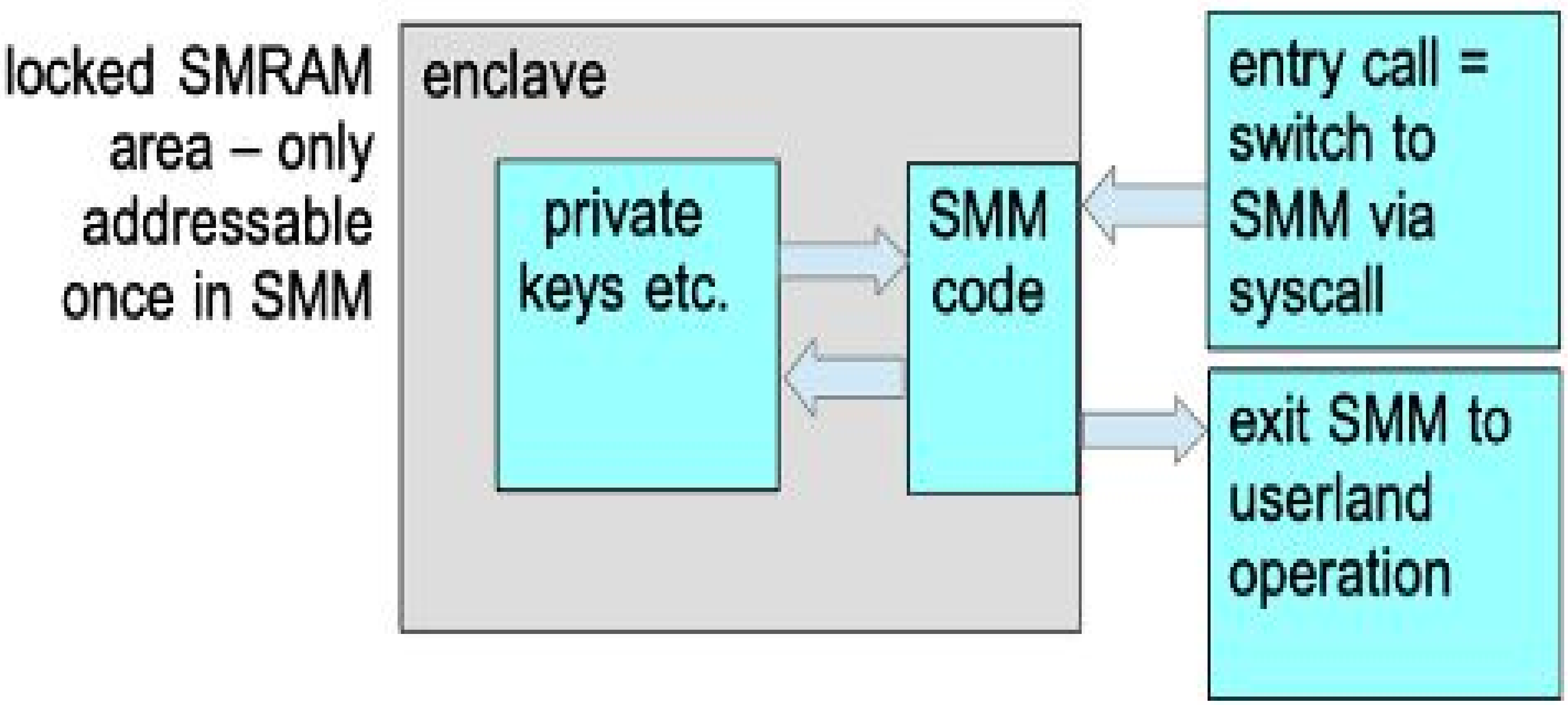
- …..to securing key enclaves.
- Protecting memory
- RAM encryption
- Address Space Layout Randomisation
- Swap encryption
- Process separation
- Process isolation
- VM isolation
- TPM
- SGX

SMM - SMRAM

- A block of DRAM that can only be addressed by the processor (no DMA from other bus devices)...
- … when the processor is in Systems Management Mode.


Processor core
data bus
address bus
SMIACT2
D_OPEN
Memory Control Hub
RAM
Legacy Video RAM
SMRAM
0xC0000
0xA0000
0x00000

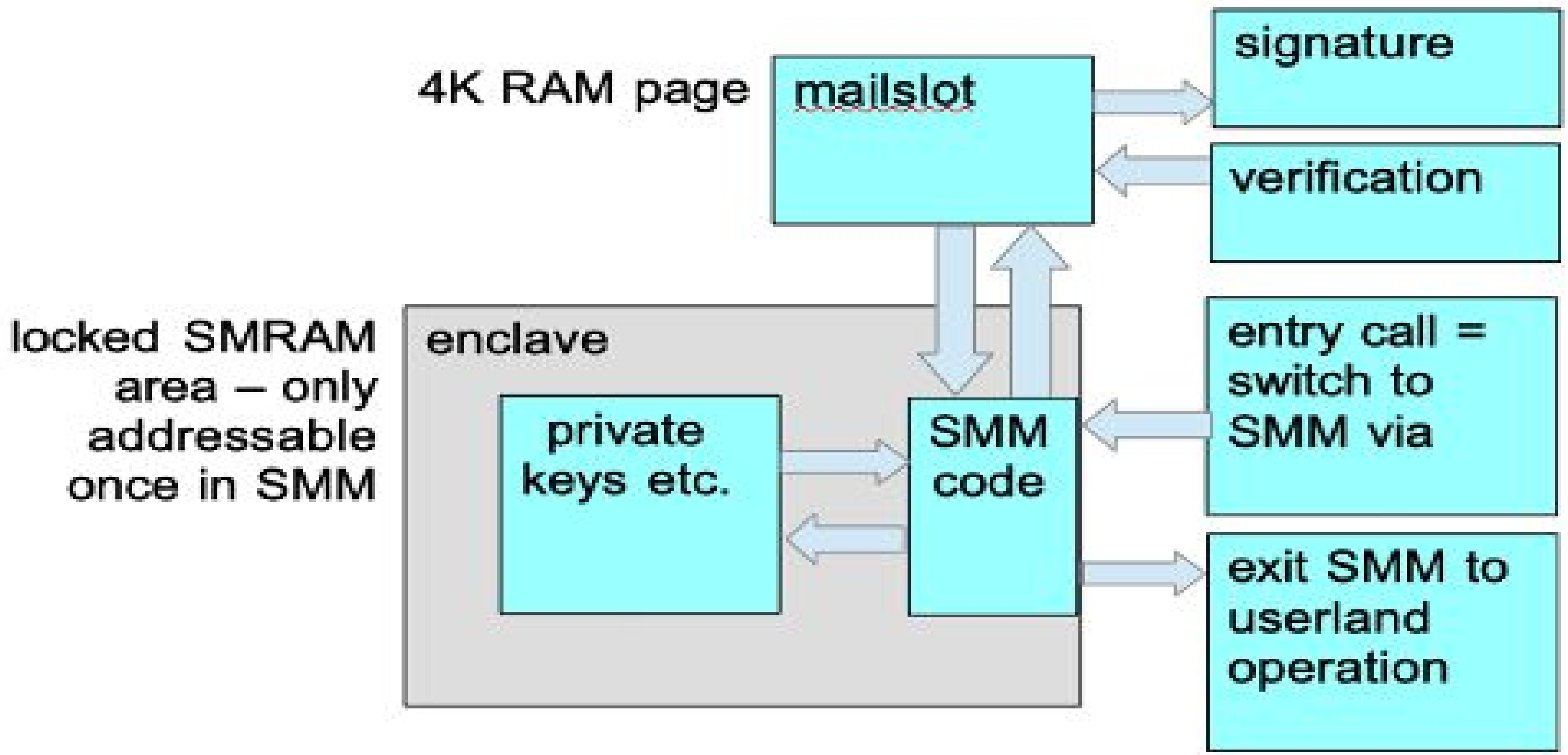
Systems Management Mode



Proposed Solution

- Overall operation
  - Key negotiation
  - Transition to SMM

Proposed Solution



Generalisable authentication

- Technique can protect keys and code for a variety of authentication/crypto purposes in the enclave

Specific example - Webserver

- To prove the SMM enclave approach works, we built a secure webserver that can prove its identify by signing responses with keys/code stored in the enclave.
  - Does it work?
  - Is it secure?
  - Is it fast enough?

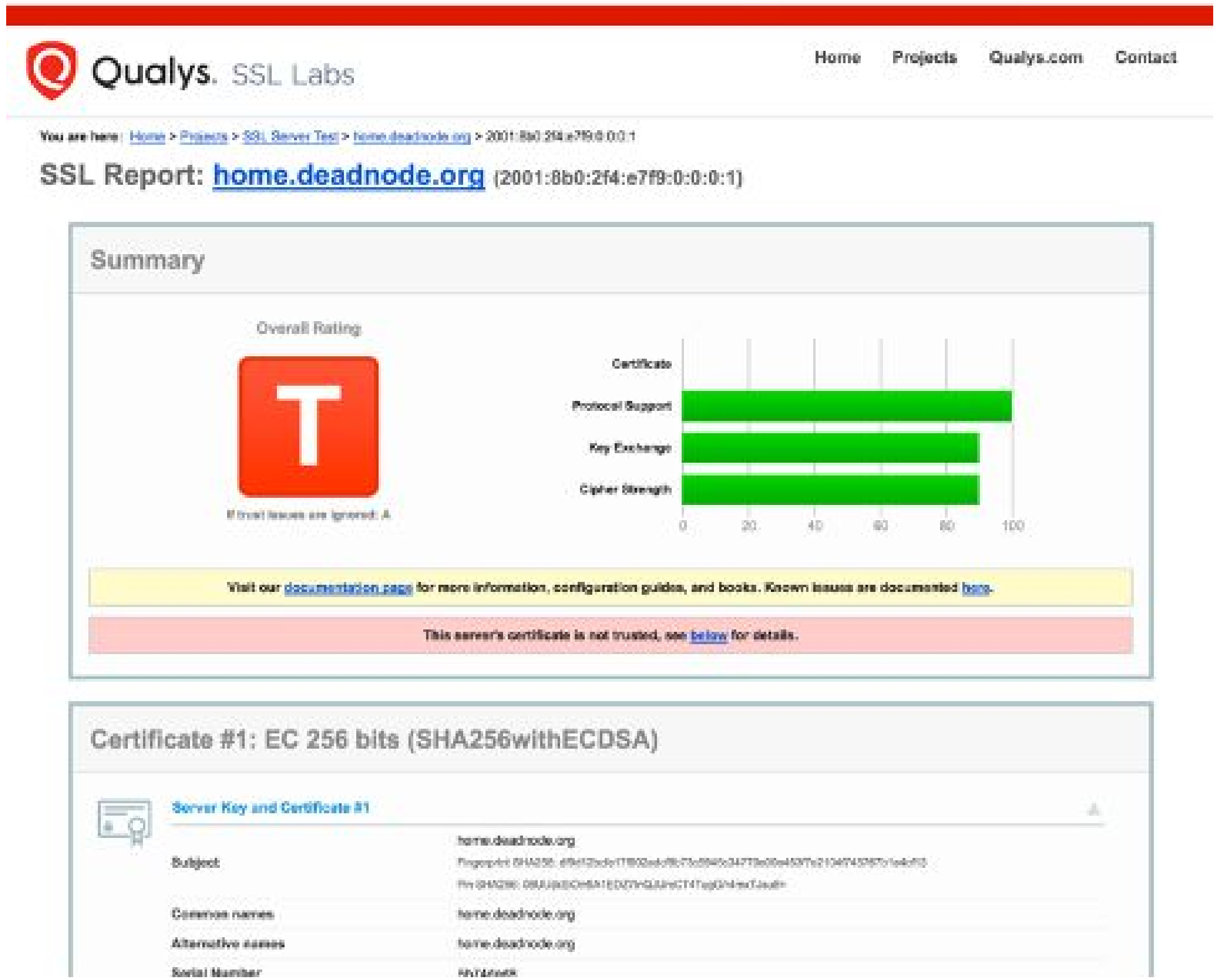
Evaluation – Four Experiments

Num	Experiment	Purpose
1	Use with range of browsers	Verifying basic webserver functionality
2	Qualys - SSL Labs	Verifying webserver SSL protocol compliance
3	Micro-benchmarking	Measuring the 'real-time' overhead imposed by entering and exiting SMM
4a	Comparison of webserver performance with crypto operation performed with 3 different levels of protection	Measuring the rate that pages could be served with crypto-keys handled in-process, i.e., with no protection
4b		Measuring the rate that pages could be served with crypto-keys handled in a separate process, i.e., with process-separation protection
4c		Measuring the rate that pages could be served with crypto-keys handled in SMM

Evaluation Process - Functionality

- Tested with a range of browsers/web-clients
  - No problems

Evaluation – Security



Abertay University

Evaluation – Performance

- Is using SMM practical?
- Does it slow down the system too much to be useful?
  - Micro-benchmarking
    - Real time measurements of the transitions to-from SMM
  - Webserving comparision
    - How fast can we serve pages with different levels of key-isolation?

Abertay University

Evaluation – Micro-benchmarking

Operation	Purpose
NOP SMI	Round trip to/from SMM
open-close	System call requiring access to kernel memory
`getpid()`	Trivial system call to reflect minimal kernel transition cost
signing	Execute a cryptographic operation - specifically generate a signed certificate

TABLE IV. TEST PLATFORMS FOR BENCHMARKING

Model	X200	T60	Qemu-VM
CPU	Core 2 Duo P8400	Core 2 Duo T5600	Core 2 Duo T5600
Clockspeed	2.26 GHz	1.83GHz	1.83GHz
RAM	4 GiB	3 GiB	1 GiB
BIOS	Libreboot	Lenovo original	SeaBIOS

Micro-benchmarking results

TABLE V. EXECUTION TIME FOR SYSTEM CALLS AND SMI INVOCATIONS

Operation	**X200**	**T60**		**T60 Qemu-KVM**	
Units	µs	µs	TSC	µs	TSC
NOP SMI	448	Not available		1310	2.4m
`getpid`	0.4	1.1	620	21	12k
open/close	3	7.1	3900	26	26k
signing	Not available	878	1.606m	905	1.65m

Abertay University

TABLE VI. EXECUTION TIME (TSC TICKS) ON BARE METAL

Operation	Minimum	1st Quartile	Median	3rd Quartile	Maximum
getpid	1133	1155	1155	1155	5211503
open-close	6347	6479	6512	6545	3776872
signing	1534995	1542285.25	1544378	1547757.75	2924856

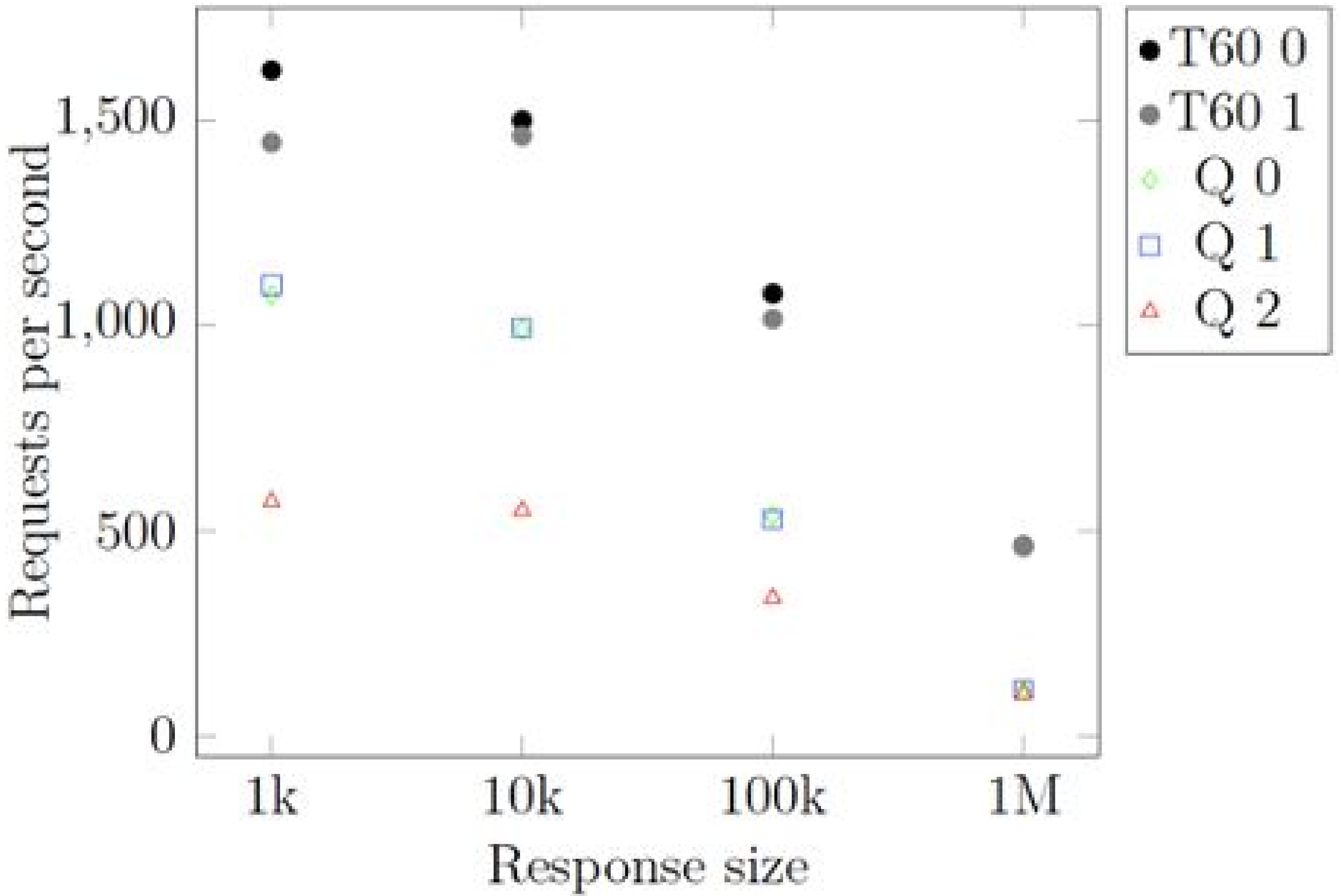
TABLE VII. EXECUTION TIME (TSC TICKS) UNDER KVM

Operation	Minimum	1st Quartile	Median	3rd Quartile	Maximum
NOP SMI	2235276	2326436.75	2921712.5	3618389	26339800
getpid	20229	20295	20317	20361	33031357
open-close	44902	45397	45496	45595	29565196
signing	1536480	1543069	1546578	1596921	12533972

Webserving

- Testing speed of page serving with 3 level of key protection:
  - Q0 - None
  - Q1 - Process separation (None SMM)
  - Q2 - Full SMM isolation
  -
- https requests generated via `curl`
- Page size varied

Performance in each configuration



Abertay University

Conclusions

- The SMM technique offers greater key protection than process separation with minimal impact on processing speed.

Abertay University

Future Work

- Intrusion counter-measures
- Operation batching
- Other applications/protocols