Secure Cryptographic Protocol Execution based on Runtime Verification

Secure Communication in the Quantum Era (SPS G5448)
February 5th, 2020

Christian Colombo
Mark Vella

Cryptographic Protocols

Design

Proofs to validate design against threat models

Implementation

Difficult to make it fully secure…
So many things can go wrong!

Levels of abstraction of security threats

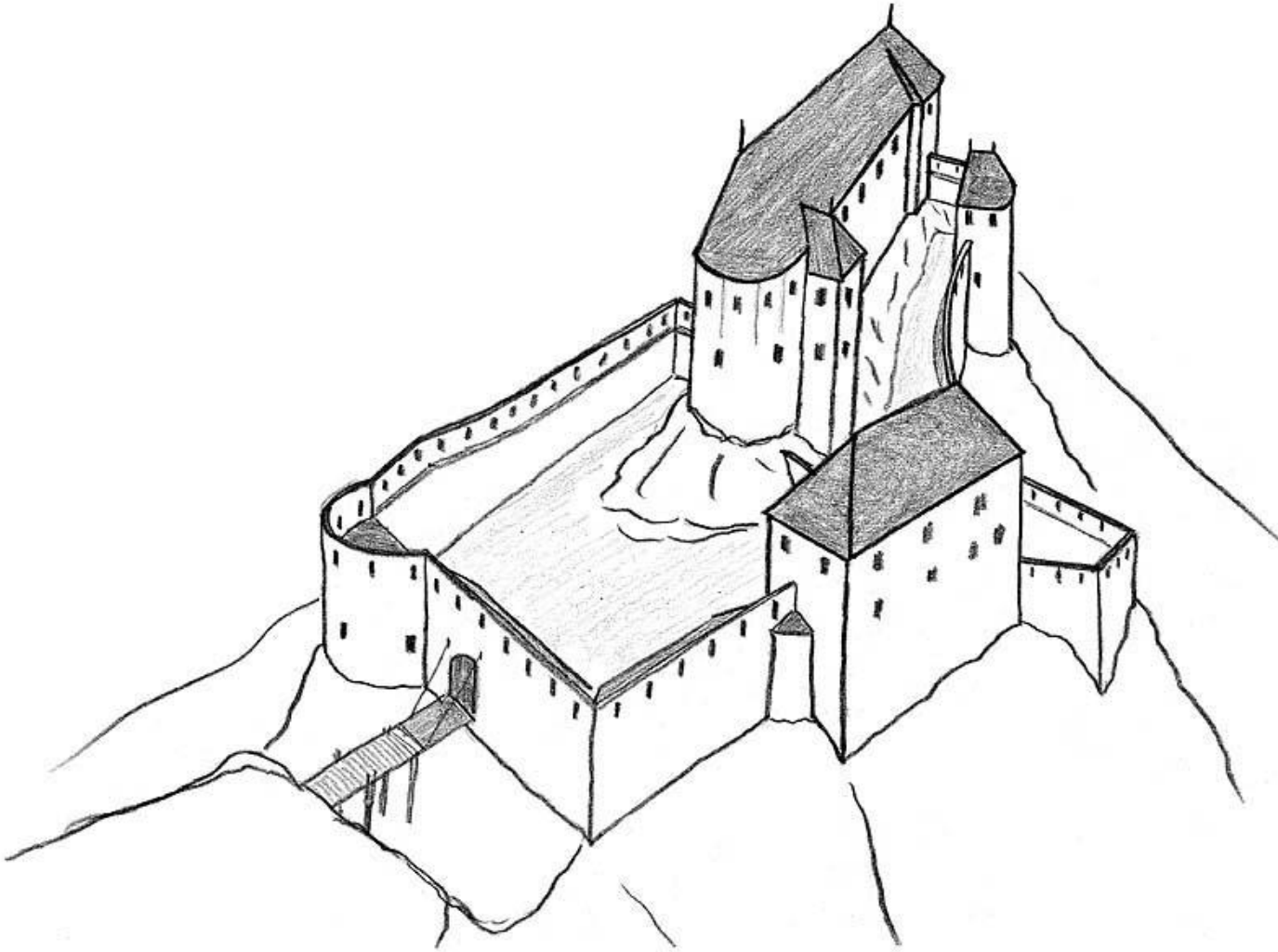
(High level) Wrong protocol implementation	The protocol implementation might deviate from the verified (theoretical) design
Medium level threats	Malware, Data leaks, etc
Low level threats	Arithmetic overflows, undefined downcasts, and invalid pointer references
Hardware	Can hardware be trusted? Side Channel attacks?

It is difficult to make implementation fully secure…

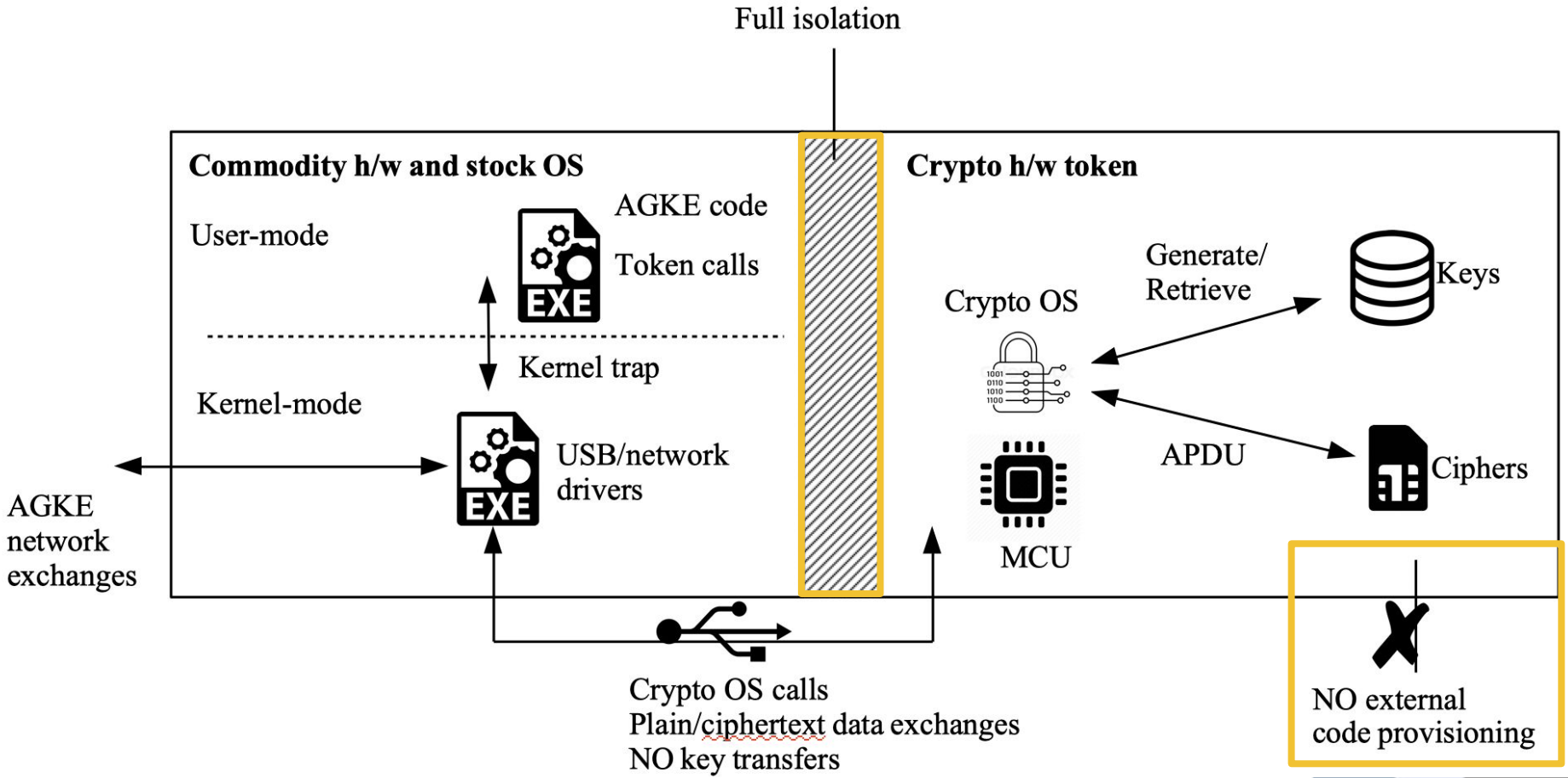
but we can raise the bar as much as possible.

Our strategy

Isolate!



L-Università ta' Malta



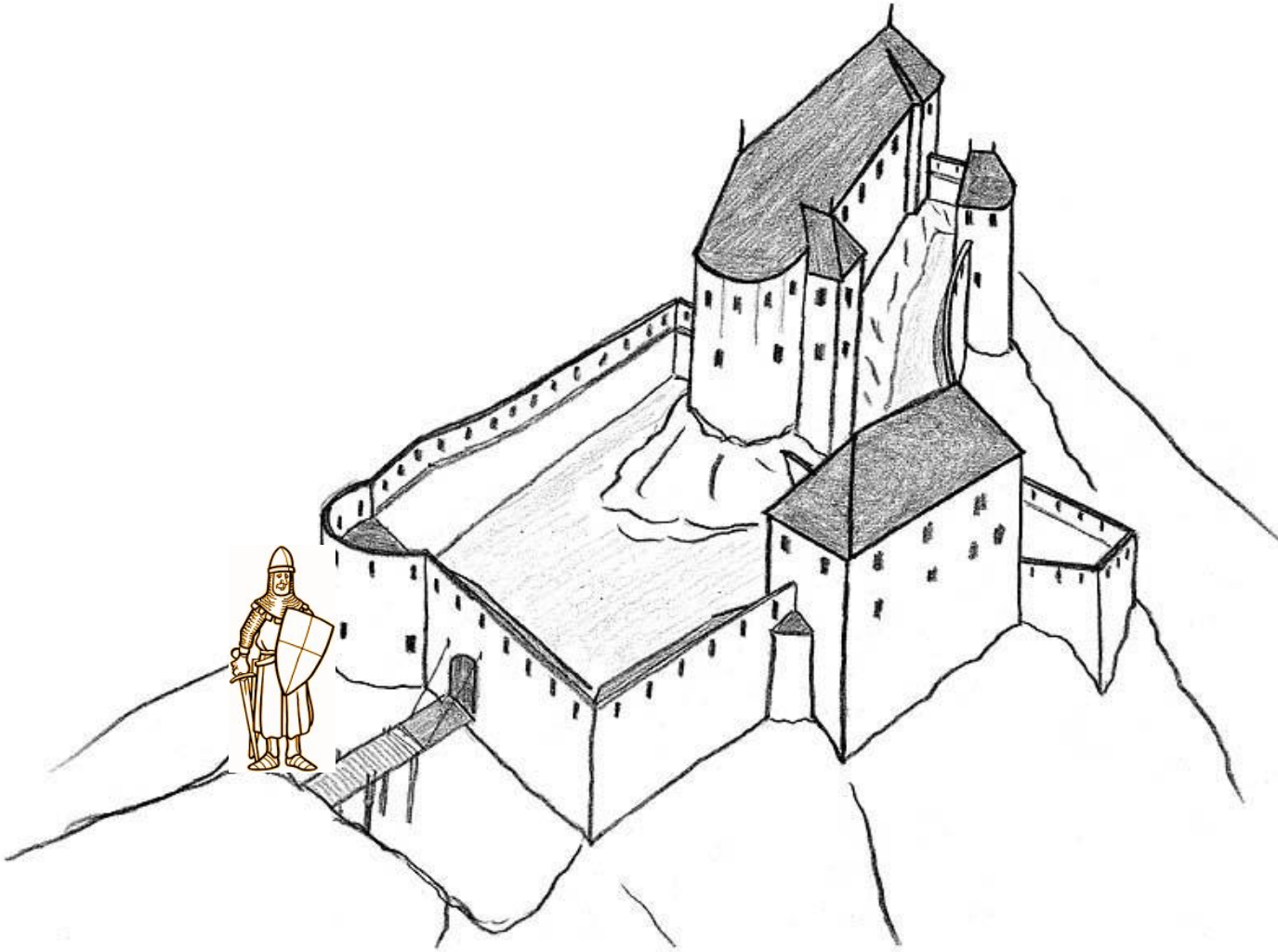
L-Università ta' Malta

NATO OTAN

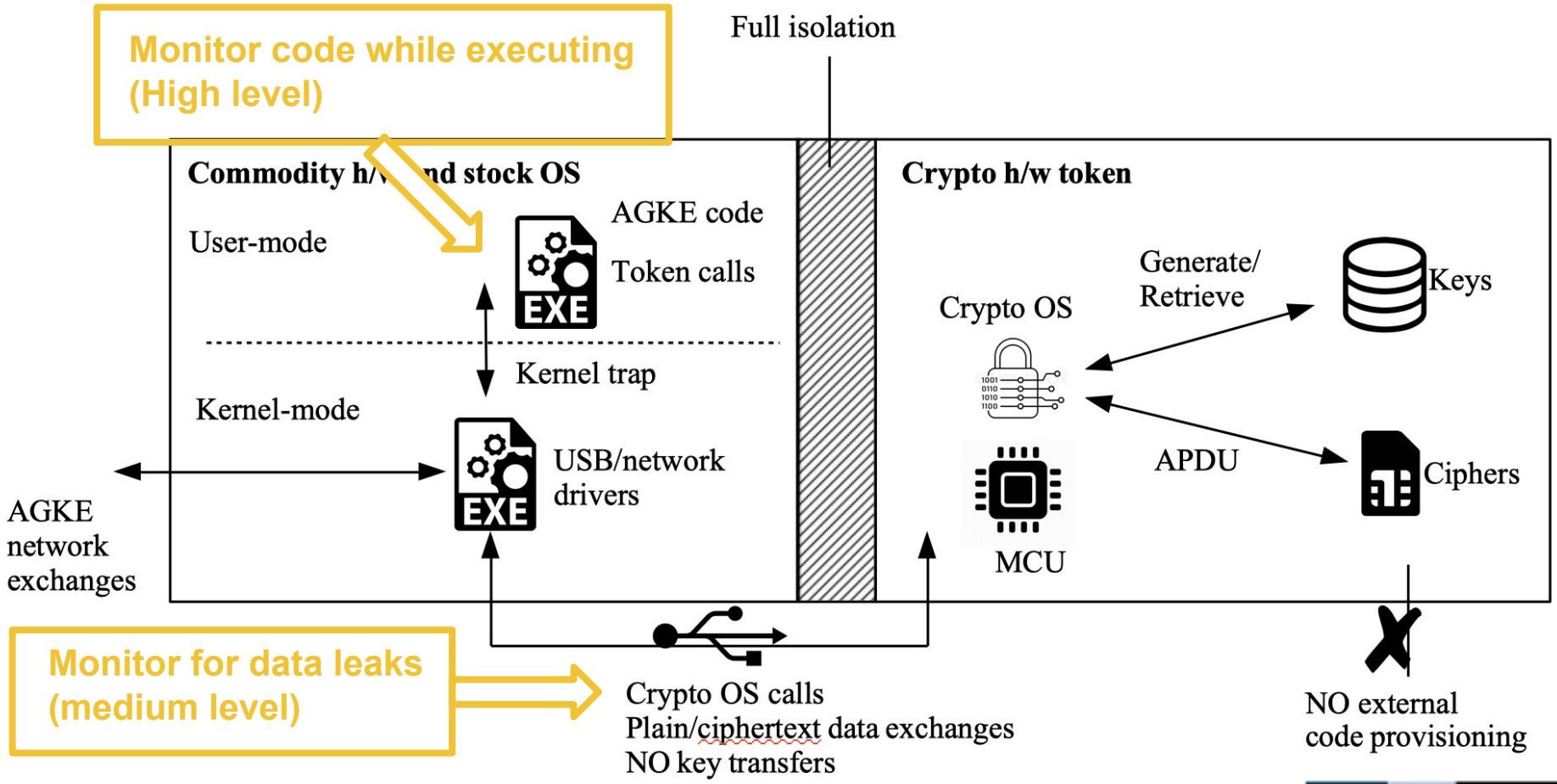
Our strategy

Isolate!

Monitor!



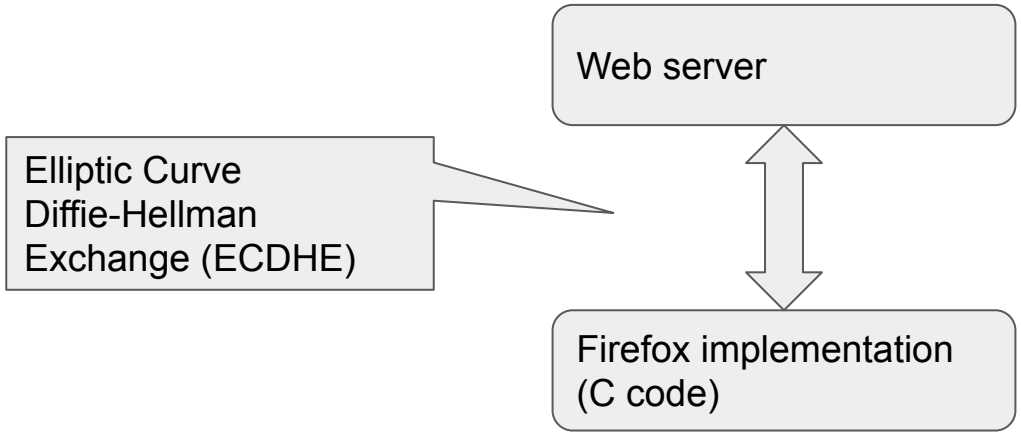
L-Università ta' Malta



L-Università ta' Malta

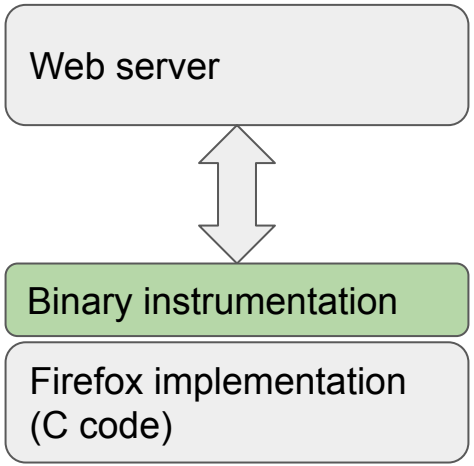
NATO OTAN

Preliminary case study



Preliminary implementation

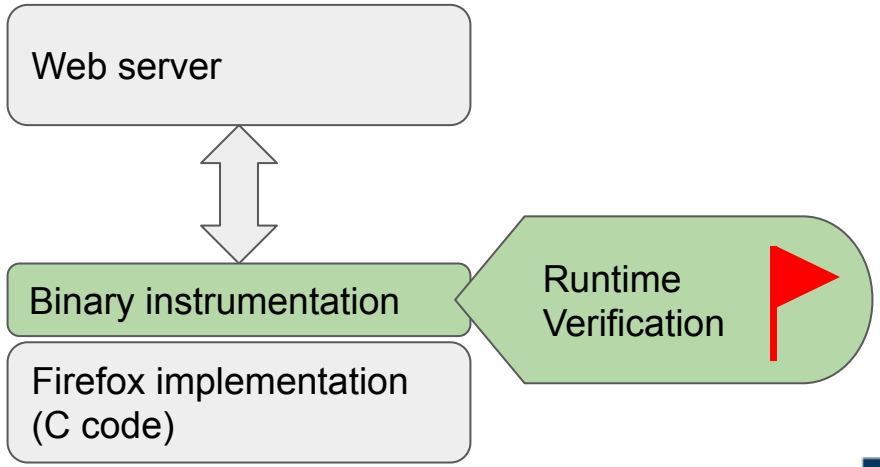
Setup using Binary-level instrumentation



Preliminary implementation

Setup using Binary-level instrumentation

Through which monitors can gain visibility

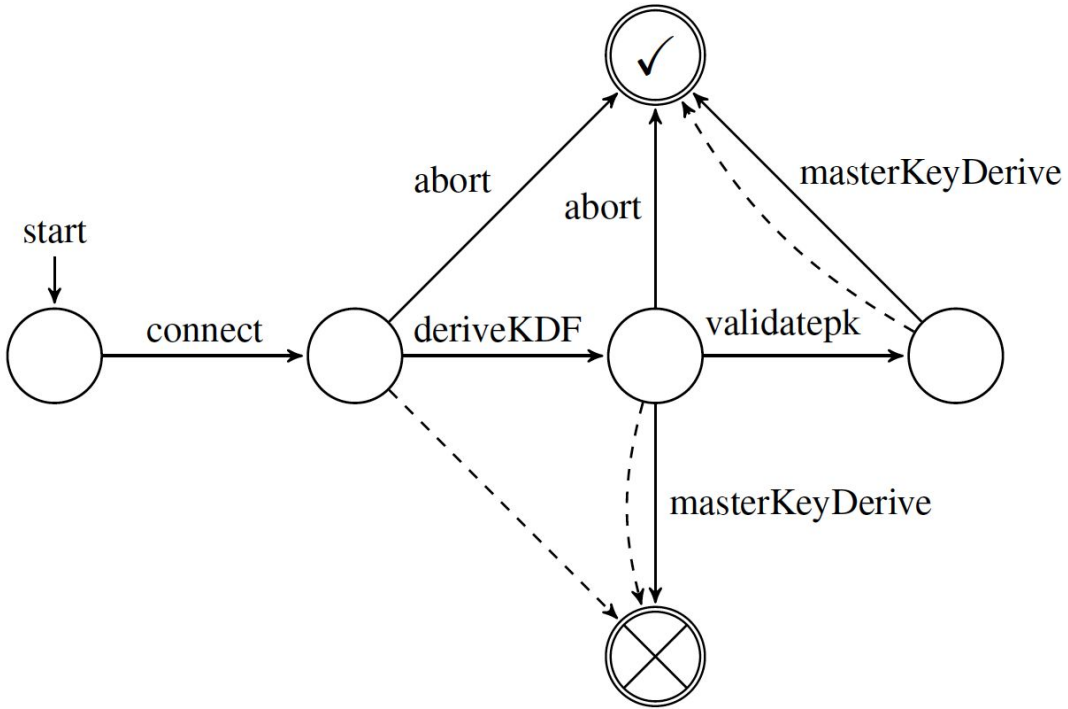


Properties verified (High level) on ECDHE

Digital certificate verification is done (in order to authenticate public keys sent by peers)

Properties verified (High level) on ECDHE

Validation of remote peer's **public key** on each exchange is done (unless the session is aborted)

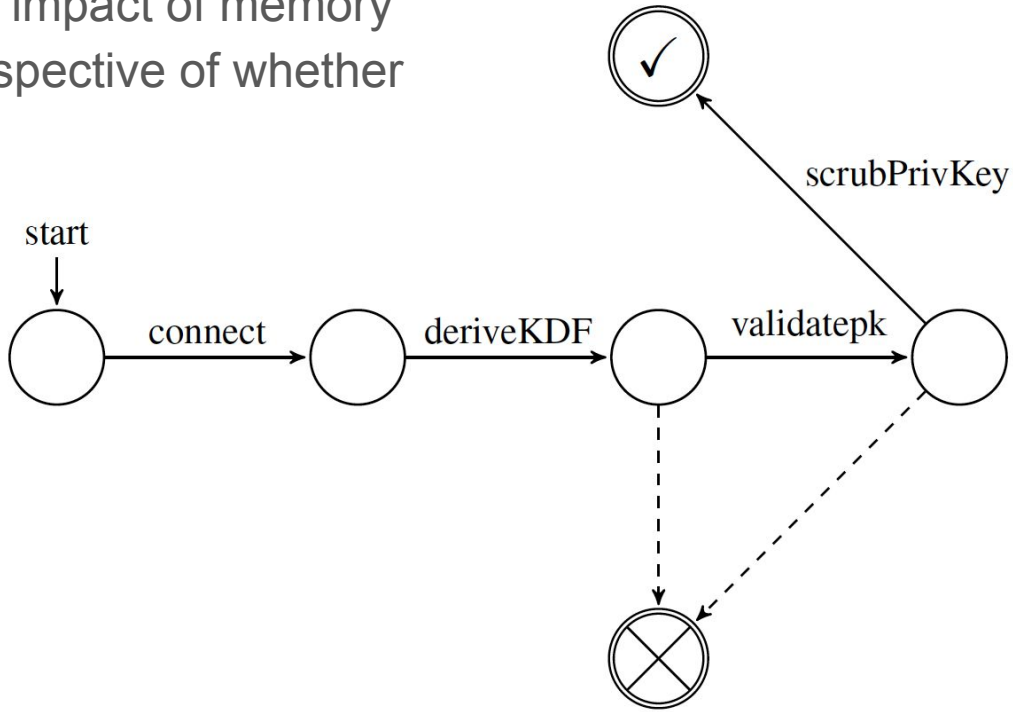


L-Università ta' Malta

NATO OTAN

Properties verified (High level) on ECDHE

Once master secret is established, private keys should be **scrubbed from memory** (to limit the impact of memory leak attacks such as Heartbleed, irrespective of whether the session is aborted)



L-Università ta' Malta

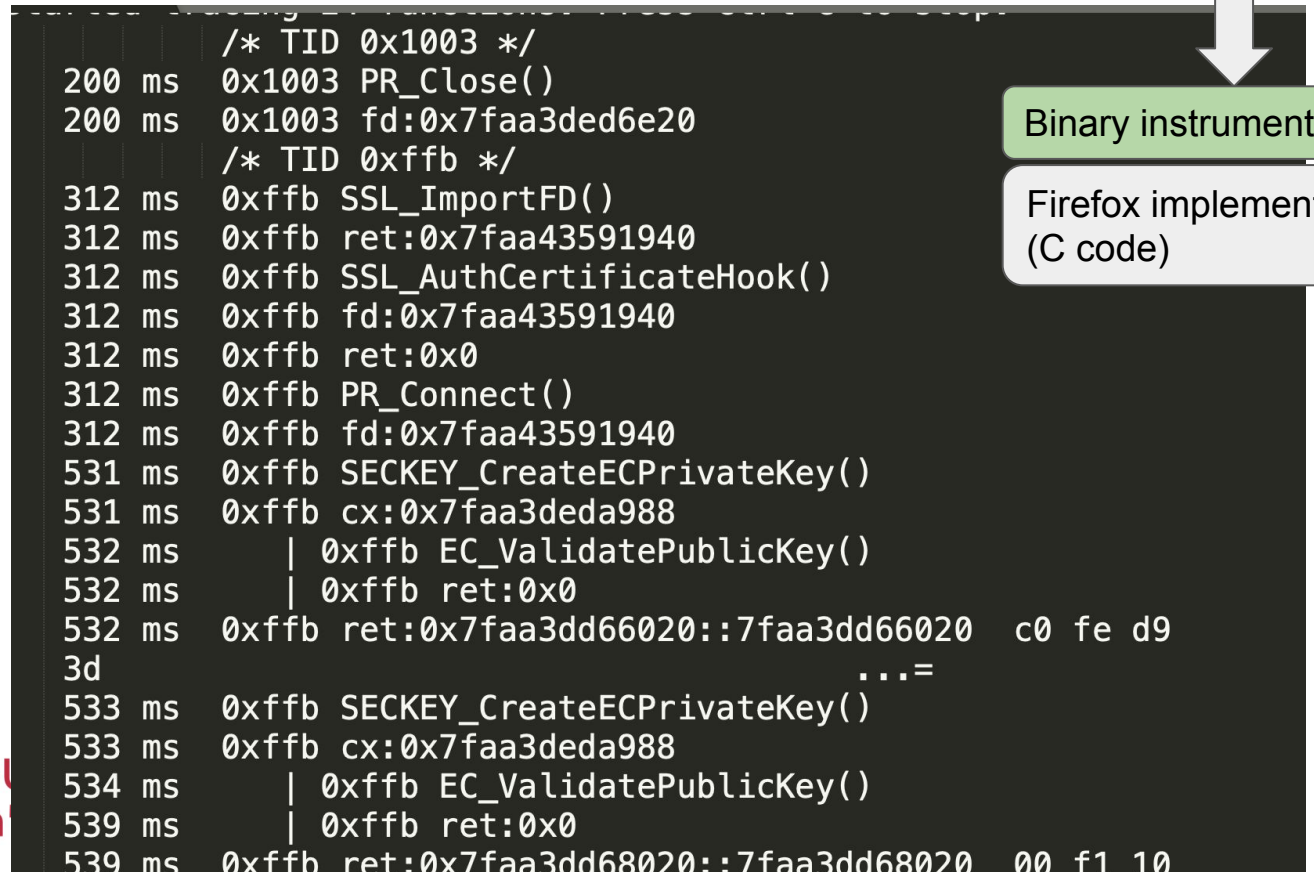
Feasibility study of approach

Is the approach possible for a realistic code base?

Is the approach feasible in terms of overheads?

Used the Firefox case study on top 100 Alexa sites

Feasibility study



Web server

Binary instrumentation

Runtime Verification

Firefox implementation (C code)

Overheads measurement

Configuration	Pages	Page load time (ms)	
		mean	*std. dev.*
No RV	1,000	6,918.37	24,870.86
With RV	1,000	7,282.35	27,328.9
Mean overhead	0.05		
Wilcoxon signed-rank test	p=0.281		

Overheads measurement

Configuration	Pages	Page lo	
		mean	*d. dev.*
No RV	1,000	6,918.37	24,870.86
With RV	1,000	7,282.35	27,328.9
Mean overhead	0.05		
Wilcoxon signed-rank test	p=0.281		

0.05 ms per page

L-Università ta' Malta

NATO OTAN

Lessons learnt

Good start with promising results - approach seems feasible

Beware:

> **Program comprehension is required**, both for setting up function hooks as well as to enable individual TLS session monitoring
>
> Real-world code tends to be written in a manner to **favor efficient execution rather than monitorability** (eg, was difficult to keep track of particular sessions on the server)

Secure Communication in the Quantum Era

NATO Science for Peace and Security Programme, Project no. G5448

Partners:

- Slovakia - Slovak University of Technology
- Malta - University of Malta
- Spain - Universidad Rey Juan Carlos
- US - Florida Atlantic University

http://re-search.info/

L-Università ta' Malta

OTAN