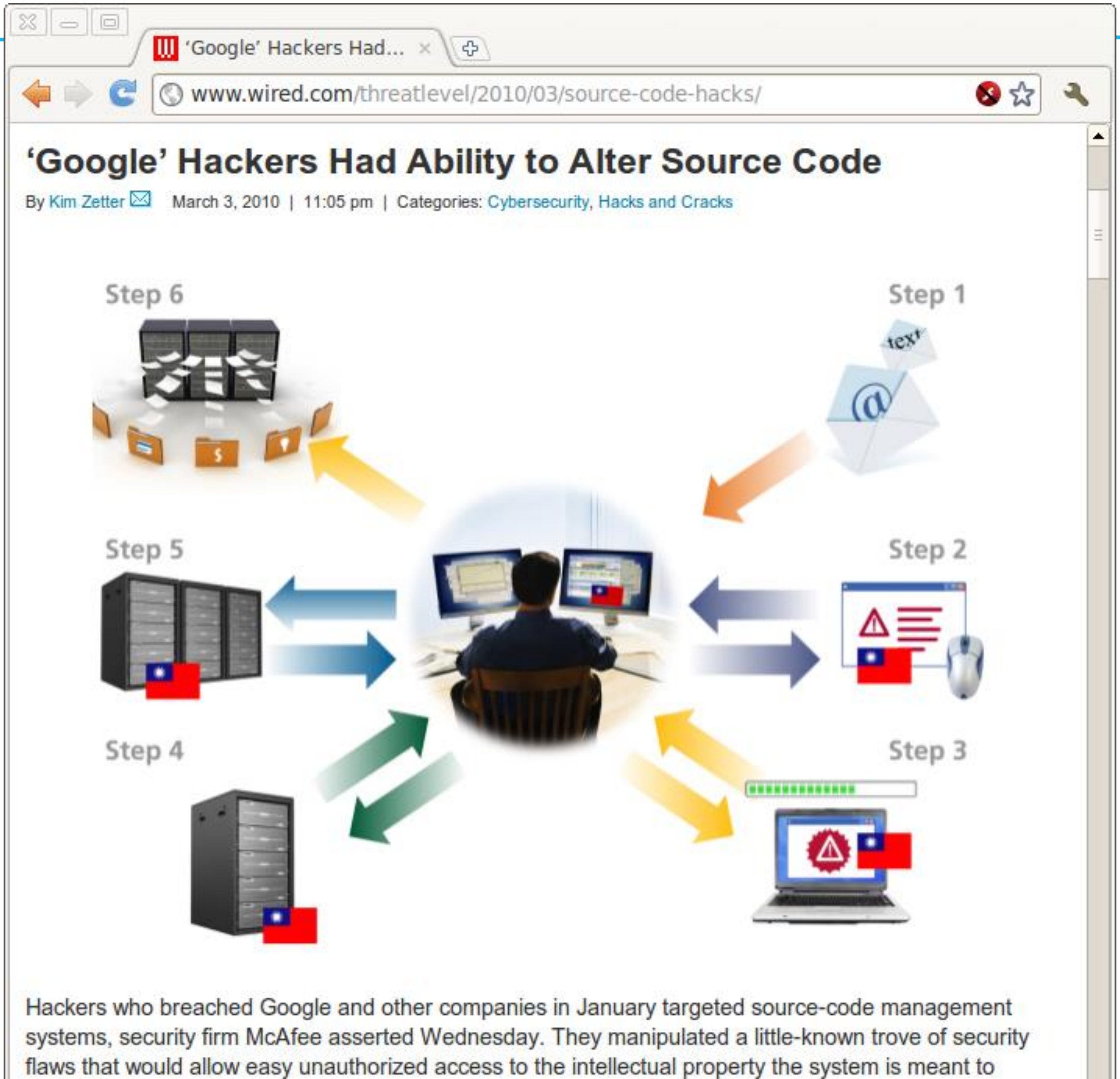
Hardware Security – Spring 2017

Marten van Dijk

together with

Kamran Haider, Chenglu Jin, Phuong Ha Nguyen

Flaws in Software Often Result in Systems Being Compromised


'Google' Hackers Had...

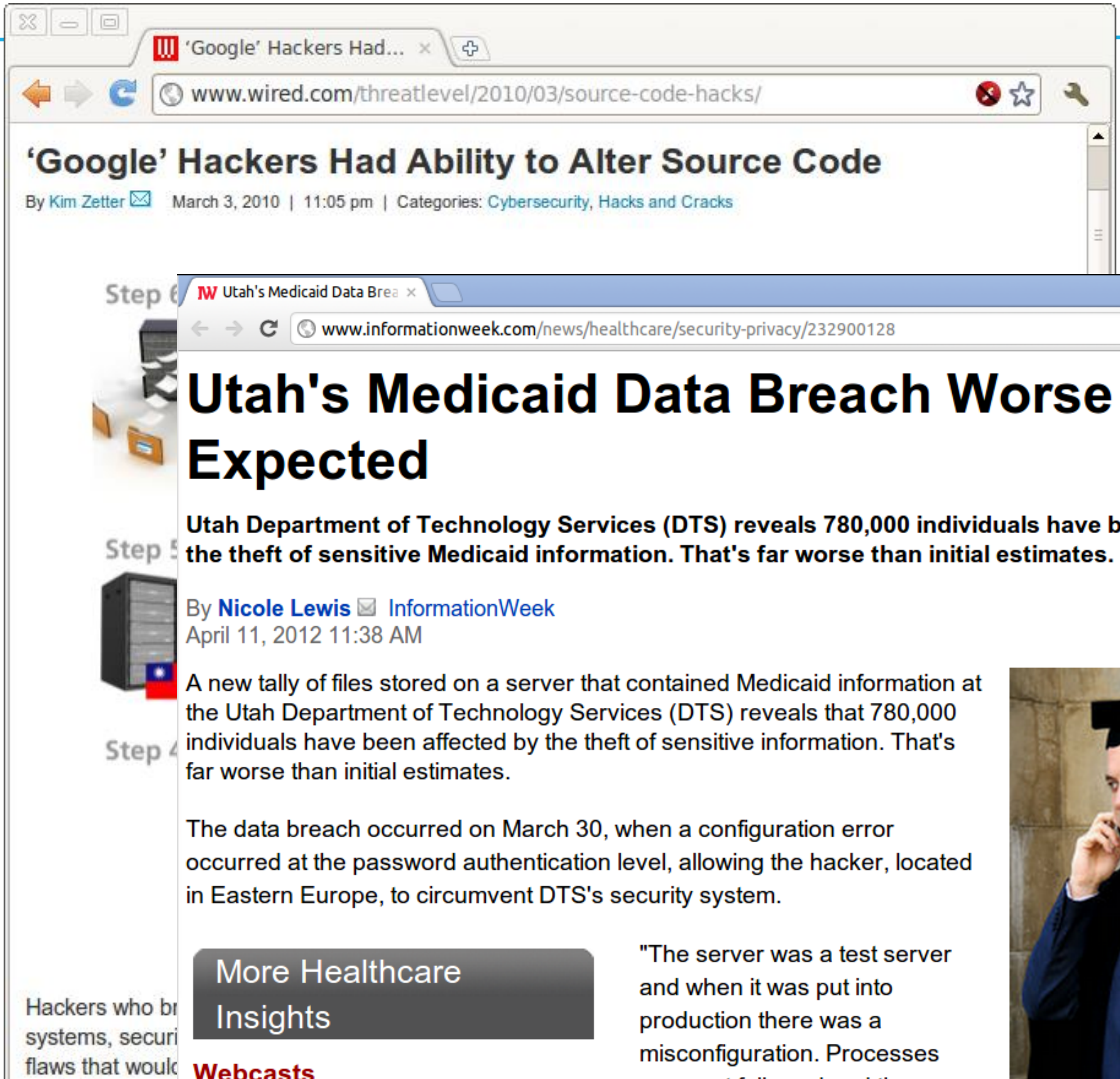
www.wired.com/threatlevel/2010/03/source-code-hacks/

'Google' Hackers Had Ability to Alter Source Code

By Kim Zetter March 3, 2010 | 11:05 pm | Categories: Cybersecurity, Hacks and Cracks

Hackers who breached Google and other companies in January targeted source-code management systems, security firm McAfee asserted Wednesday. They manipulated a little-known trove of security flaws that would allow easy unauthorized access to the intellectual property the system is meant to

Flaws in Software Often Result in Systems Being Compromised



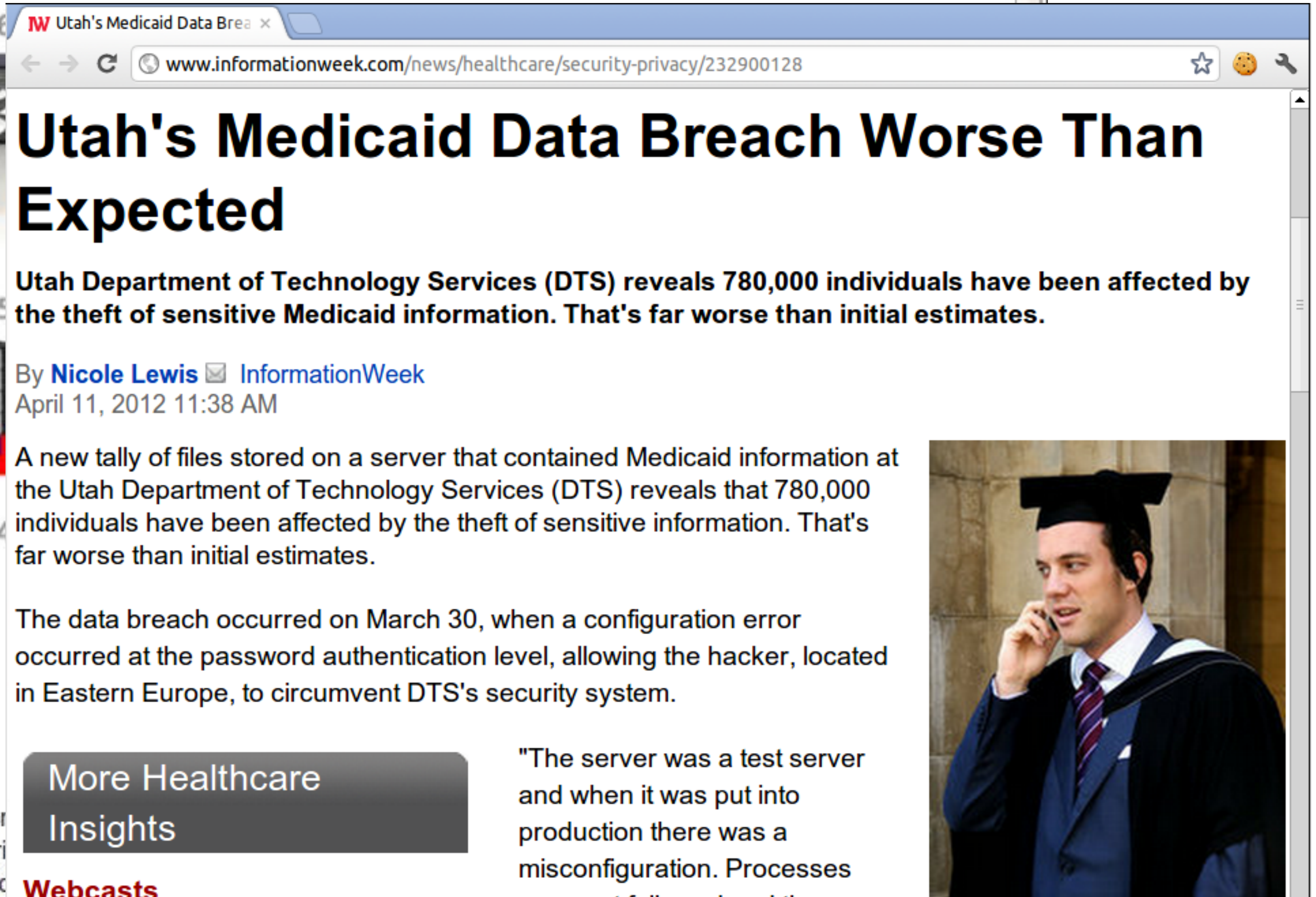
'Google' Hackers Had...

www.wired.com/threatlevel/2010/03/source-code-hacks/

'Google' Hackers Had Ability to Alter Source Code

By Kim Zetter March 3, 2010 | 11:05 pm | Categories: Cybersecurity, Hacks and Cracks

Hackers who br
systems, securi
flaws that woul



Utah's Medicaid Data Brea

www.informationweek.com/news/healthcare/security-privacy/232900128

Utah's Medicaid Data Breach Worse Than Expected

Utah Department of Technology Services (DTS) reveals 780,000 individuals have been affected by the theft of sensitive Medicaid information. That's far worse than initial estimates.

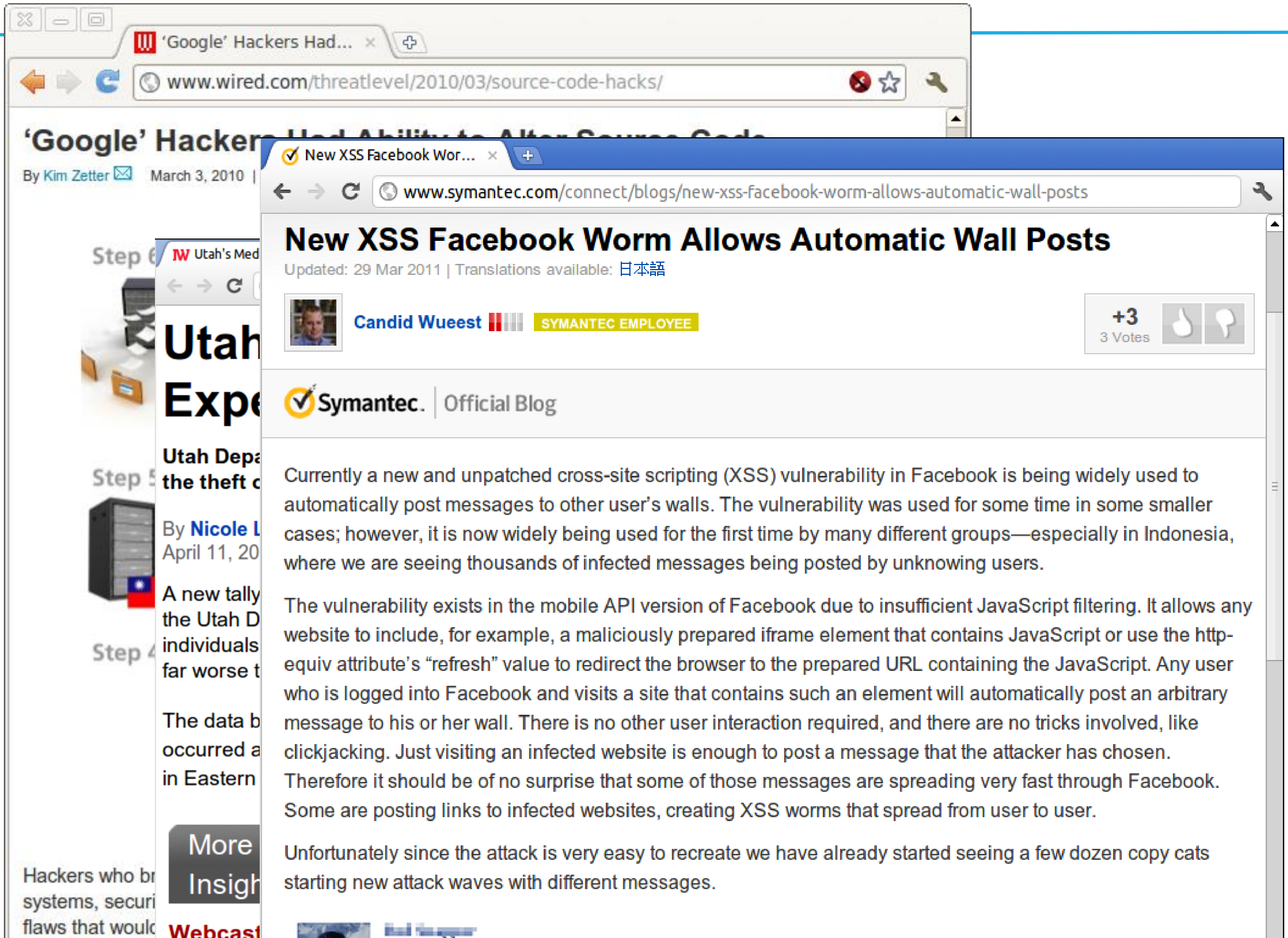
By Nicole Lewis InformationWeek
April 11, 2012 11:38 AM

A new tally of files stored on a server that contained Medicaid information at the Utah Department of Technology Services (DTS) reveals that 780,000 individuals have been affected by the theft of sensitive information. That's far worse than initial estimates.

The data breach occurred on March 30, when a configuration error occurred at the password authentication level, allowing the hacker, located in Eastern Europe, to circumvent DTS's security system.

"The server was a test server and when it was put into production there was a misconfiguration. Processes

Flaws in Software Often Result in Systems Being Compromised



'Google' Hackers Had...

www.wired.com/threatlevel/2010/03/source-code-hacks/

'Google' Hacker

By Kim Zetter March 3, 2010 |

Step 6

Step 5

Step 4

Hackers who br
systems, securi
flaws that woul

Utah's Med

Utah
Expe

Utah Depa
the theft c

By Nicole L
April 11, 20

A new tally
the Utah D
individuals
far worse t

The data b
occurred a
in Eastern

More
Insigh

Webcast

New XSS Facebook Wor...

www.symantec.com/connect/blogs/new-xss-facebook-worm-allows-automatic-wall-posts

New XSS Facebook Worm Allows Automatic Wall Posts

Updated: 29 Mar 2011 | Translations available: 日本語

Candid Wueest SYMANTEC EMPLOYEE

+3
3 Votes

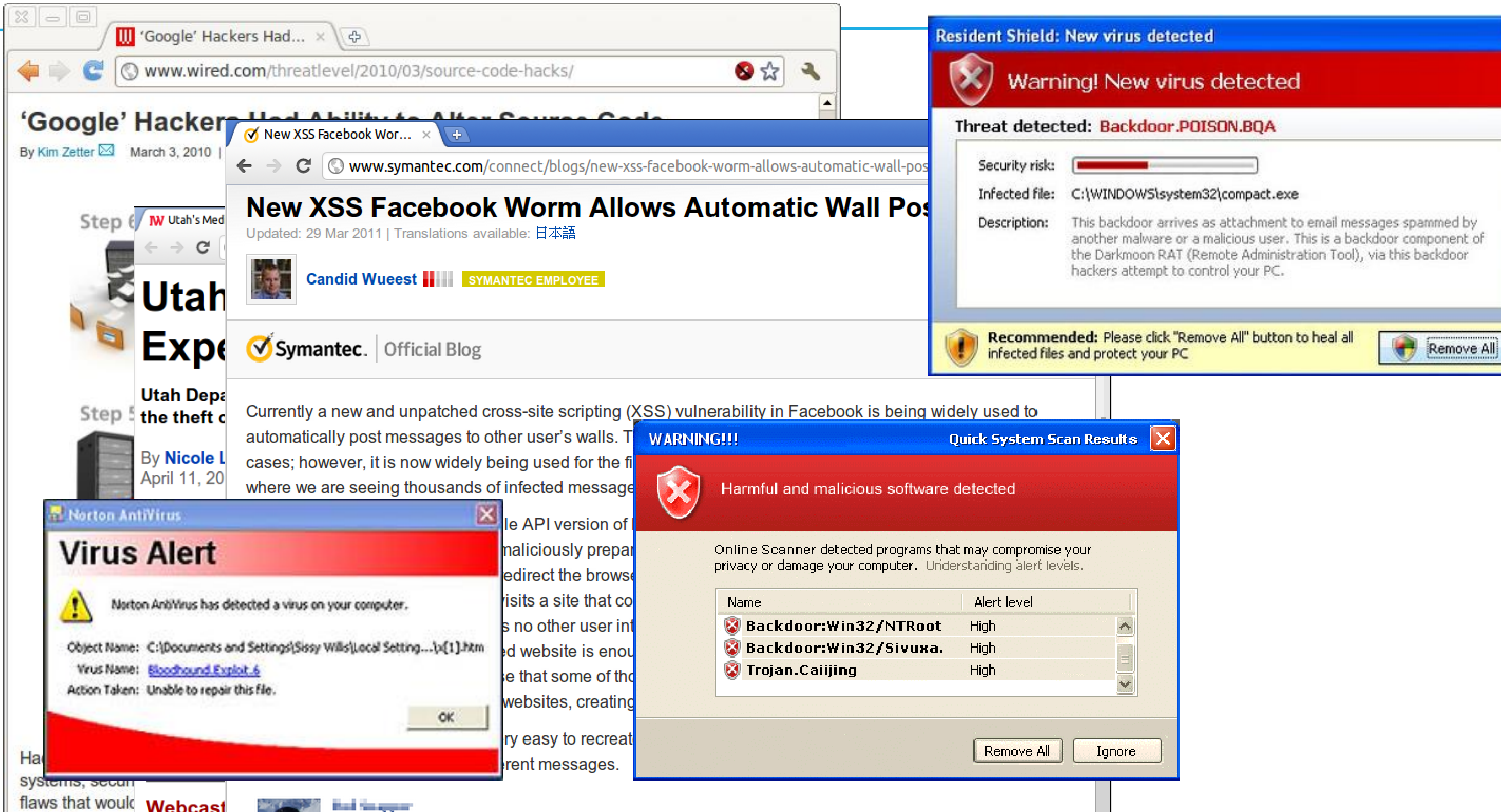
Symantec. | Official Blog

Currently a new and unpatched cross-site scripting (XSS) vulnerability in Facebook is being widely used to automatically post messages to other user's walls. The vulnerability was used for some time in some smaller cases; however, it is now widely being used for the first time by many different groups—especially in Indonesia, where we are seeing thousands of infected messages being posted by unknowing users.

The vulnerability exists in the mobile API version of Facebook due to insufficient JavaScript filtering. It allows any website to include, for example, a maliciously prepared iframe element that contains JavaScript or use the http-equiv attribute's "refresh" value to redirect the browser to the prepared URL containing the JavaScript. Any user who is logged into Facebook and visits a site that contains such an element will automatically post an arbitrary message to his or her wall. There is no other user interaction required, and there are no tricks involved, like clickjacking. Just visiting an infected website is enough to post a message that the attacker has chosen. Therefore it should be of no surprise that some of those messages are spreading very fast through Facebook. Some are posting links to infected websites, creating XSS worms that spread from user to user.

Unfortunately since the attack is very easy to recreate we have already started seeing a few dozen copy cats starting new attack waves with different messages.

Flaws in Software Often Result in Systems Being Compromised



'Google' Hackers Had...

www.wired.com/threatlevel/2010/03/source-code-hacks/

'Google' Hacker

By Kim Zetter March 3, 2010 |

Step 6

Step 5

Utah's Med

Utah

Expe

Utah Depa
the theft c

By Nicole L
April 11, 20

Ha
systems, secur
flaws that woul

Webcast

New XSS Facebook Wor...

www.symantec.com/connect/blogs/new-xss-facebook-worm-allows-automatic-wall-pos

New XSS Facebook Worm Allows Automatic Wall Pos

Updated: 29 Mar 2011 | Translations available: 日本語

Candid Wueest SYMANTEC EMPLOYEE

Symantec. | Official Blog

Currently a new and unpatched cross-site scripting (XSS) vulnerability in Facebook is being widely used to automatically post messages to other user's walls. T
cases; however, it is now widely being used for the fi
where we are seeing thousands of infected message
le API version of
naliciously prepar
edirect the brows
isits a site that co
s no other user int
d website is enou
e that some of th
websites, creating
ry easy to recreat
rent messages.

Resident Shield: New virus detected

Warning! New virus detected

Threat detected: Backdoor.POISON.BQA

Security risk:

Infected file: C:\WINDOWS\system32\compact.exe

Description: This backdoor arrives as attachment to email messages spammed by another malware or a malicious user. This is a backdoor component of the Darkmoon RAT (Remote Administration Tool), via this backdoor hackers attempt to control your PC.

Recommended: Please click "Remove All" button to heal all infected files and protect your PC

Remove All

Norton AntiVirus

Virus Alert

Norton AntiVirus has detected a virus on your computer.

Object Name: C:\Documents and Settings\Sissy Willis\Local Setting...\x[1].htm
Virus Name: Bloodhound.Exploit.6
Action Taken: Unable to repair this file.

OK

WARNING!!! Quick System Scan Results

Harmful and malicious software detected

Online Scanner detected programs that may compromise your privacy or damage your computer. Understanding alert levels.

Name	Alert level
Backdoor:Win32/NTRoot	High
Backdoor:Win32/Sivuxa.	High
Trojan.Caiijing	High

Remove All Ignore

What can be done?

SW:

- Application code
- OS
- Hypervisor

SW has hundred thousands lines of (legacy) code: What does it do? Can it be compromised?

New future: Proof carrying code

- Goes beyond safe languages
- Not formal verification: the code is designed together with proofs of its (security) properties
- Only captures security properties we know about, i.e., adversarial capabilities we know about

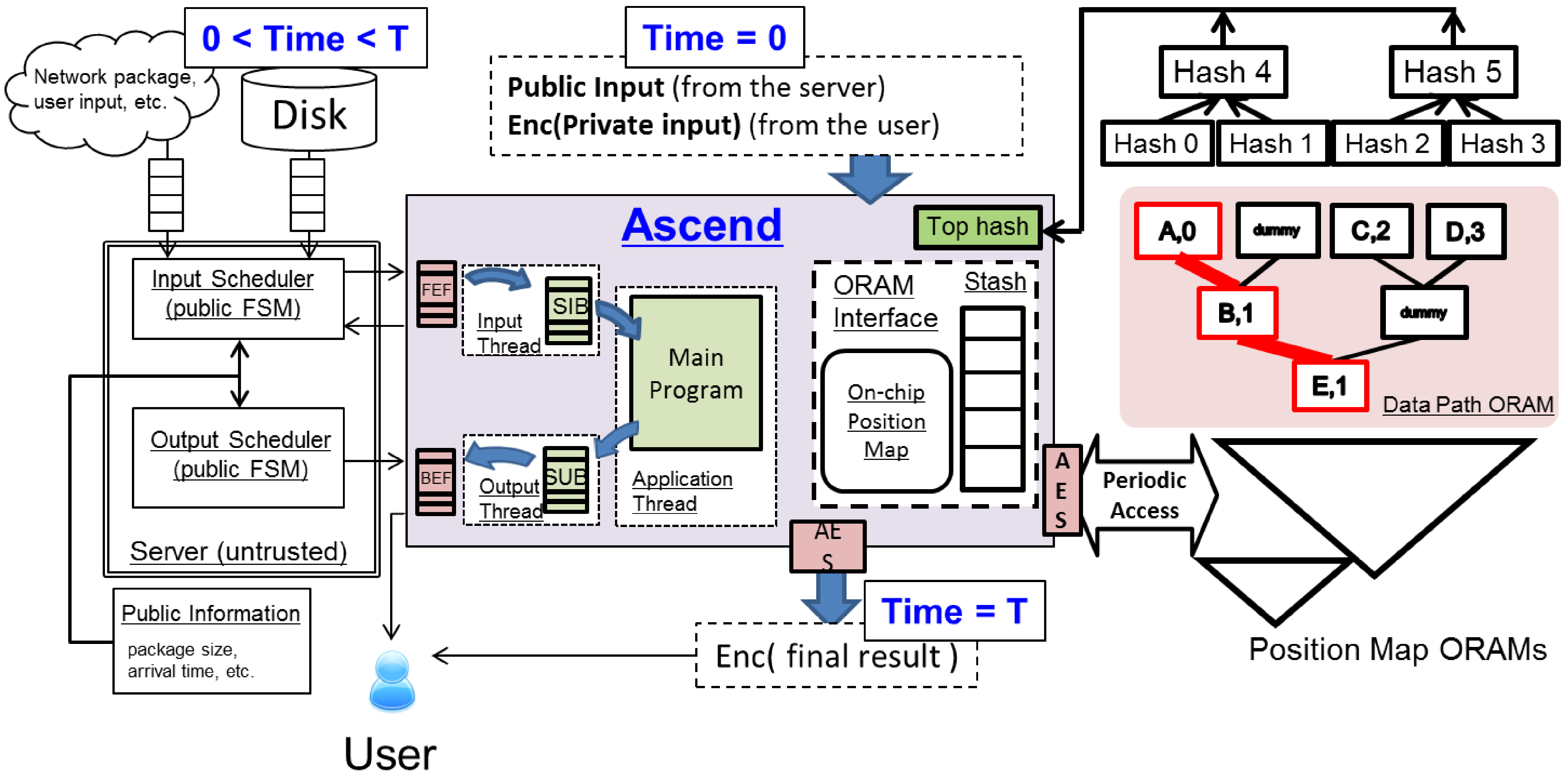
Today

- Cannot rely on third party OS and Hypervisor
- Application SW is the responsibility of the application designer
- Trust in computing can only be enforced through strong cryptographic primitives based on computational hard assumptions and HW isolation
- In theory there exists very strong crypto which allows secure computation: Prohibitively large performance overhead
- HW isolation is potentially an easy and cheap solution !! Together with small HW modules offering basic crypto functionality, trustworthy and secure computation may be possible !!

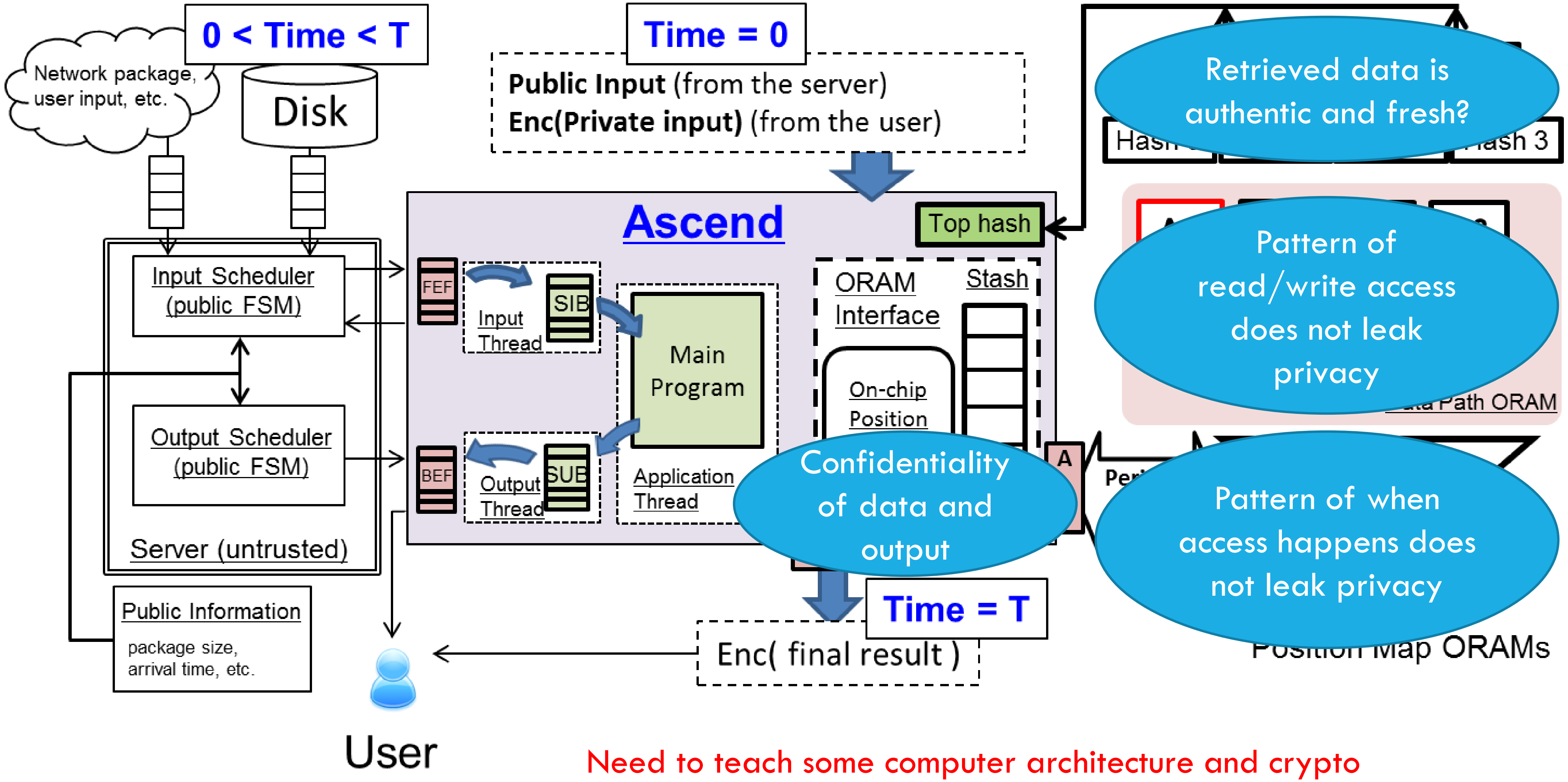
What is the course about??

- Where does computation ultimately take place?
- In HW:
  - A (multi-core) processor, e.g., Intel SGX, a micro controller (MCU), OpenSPARC on FPGA
  - Specialized HW, e.g., an encryption module, sensing devices, etc
- Focus is on general computation in processor technology
- Can we trust such computation?
  - Does execution of code use the correct code and the most fresh and authentic data?
  - Does the HW leak information, e.g., through side channels or hardware Trojans? Does the HW allow backdoors?
  - Can output be *verified* to be generated by correct code and input data on proper HW managed by a trustworthy service?
  - Is the output encrypted, is data encrypted? How is *confidentiality* managed? We need secret keys, who is in charge?
- What adversary do we protect against?
  - Each solution deals with its own adversarial model
  - Once outside the specified model, the HW/system can be attacked
  - Want to reason from an economics perspective whether the adversarial model is strong enough

An example: Ascend (Lec7b)



An example: Ascend (Lec7b)


0 < Time < T
Network package, user input, etc.
Disk
Time = 0
Public Input (from the server)
Enc(Private input) (from the user)
Ascend
Top hash
Input Scheduler (public FSM)
Output Scheduler (public FSM)
Server (untrusted)
FEF
Input Thread
SIB
Main Program
BEF
Output Thread
SUB
Application Thread
ORAM Interface
Stash
On-chip Position
Public Information
package size, arrival time, etc.
User
Time = T
Enc(final result)
Retrieved data is authentic and fresh?
Pattern of read/write access does not leak privacy
Confidentiality of data and output
Pattern of when access happens does not leak privacy
Hash 3
Path ORAM
A
Position Map ORAMs

Need to teach some computer architecture and crypto

Ascend (Lec7b) vs Sanctum (Lec7a)

Adversarial Model:
- Adversary with full physical access to the data bus
- HW TCB = CPU chip (with caches, mem. interface), Package
- SW TCB = Application processes, Trusted OS
- Not trusted: External storage, in particular, DRAM.

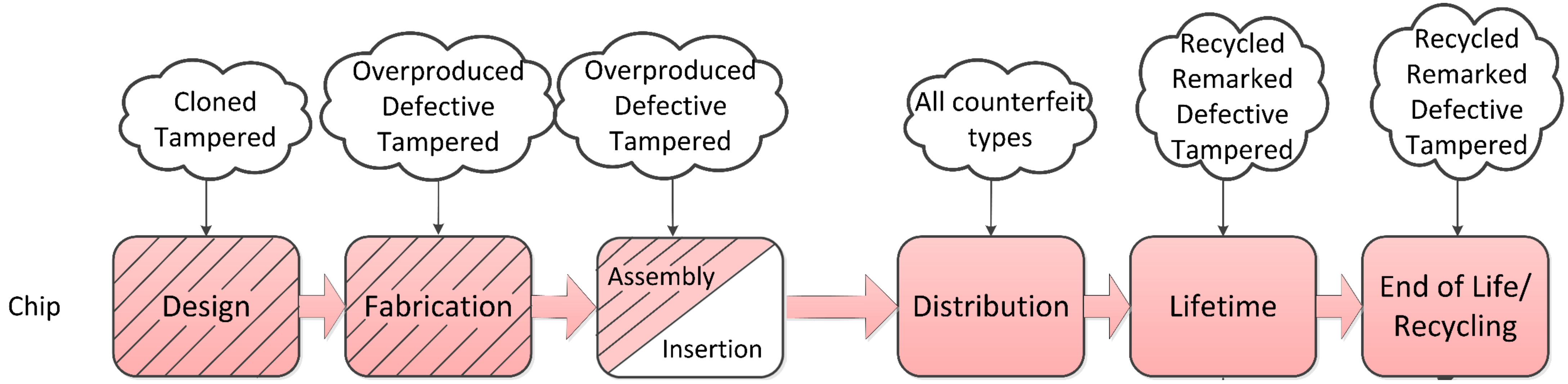
Leakage LLC-DRAM boundary:
- Data blocks → AES
- Timing channel of reads/writes → Periodic access
- Address access pattern of reads/writes → Path ORAM

Leakage Input/Output:
- Private user input → PKI
- Streaming in server data → Fixed I/O scheduler / AES
- Termination channel → T leaks bits

Authenticity/Freshness of DRAM → Merkle Tree

Adversarial Model:
- Adversary can only launch remote SW attacks
- HW TCB = CPU chip (with caches, mem. interface), Package
    - Access to DRAM by peripherals is controlled by a trusted MCU (as in Intel SGX)
    - Sanctum forbids access by untrusted OS to reserved DRAM for Secure Enclaves
- SW TCB = App. Mod. (in Secure Enclave @ lowest priv. level), Sec. Monitor (@ highest priv. level)
    - Allows multiple modules (Sanctum prevents cache timing channel attacks using locality preserving cache-coloring)
    - Allows untrusted OS

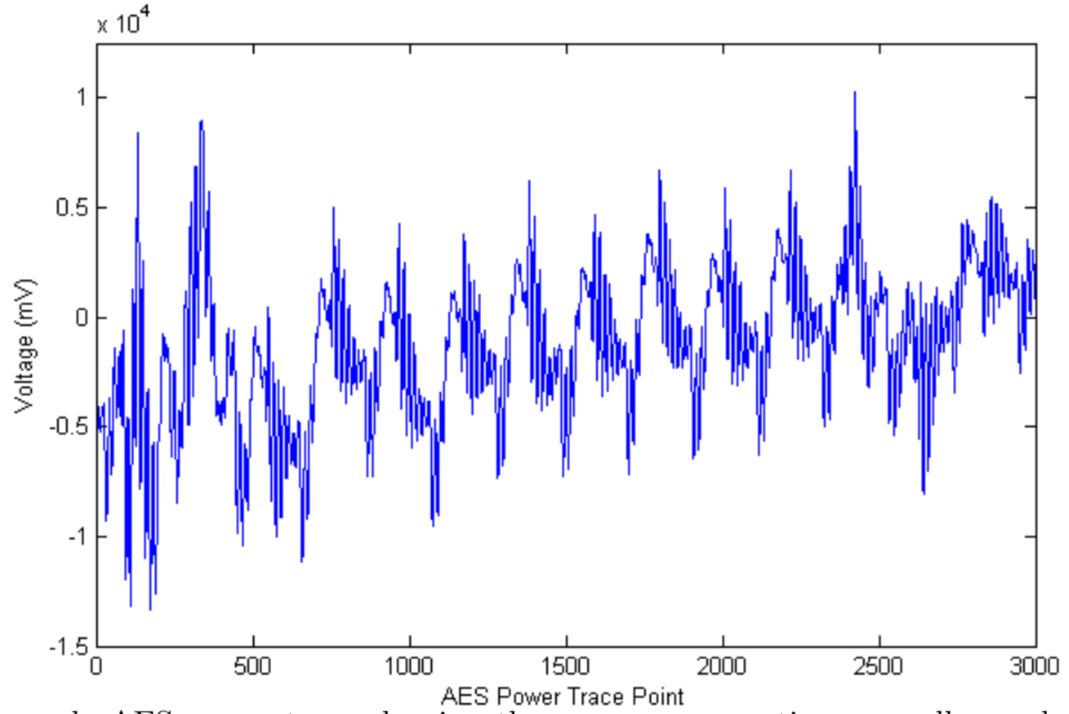
We need a Root of Trust in HW



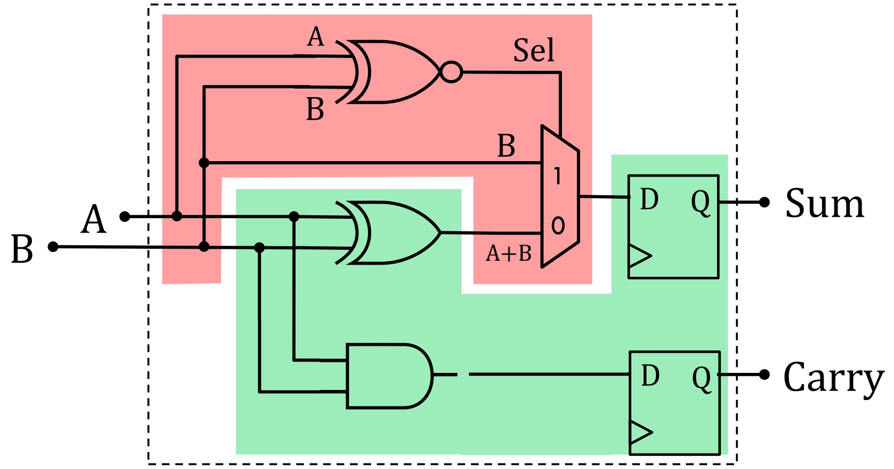
- Nowadays, untrusted IC supply chain introduces a variety of security threats.
- Many countermeasures have been proposed. In general, they are specific for one security vulnerability in the supply chain.
- More discussion in Lec9a

We need a Root of Trust in HW

- What about hardware Trojans … Lec9b
- Can we really trust the package: What other side channels are possible? E.g., the power side channel, see Lec8a and Lec8b



A sample AES power trace showing the power consumption over all rounds of an AES-128 encryption.

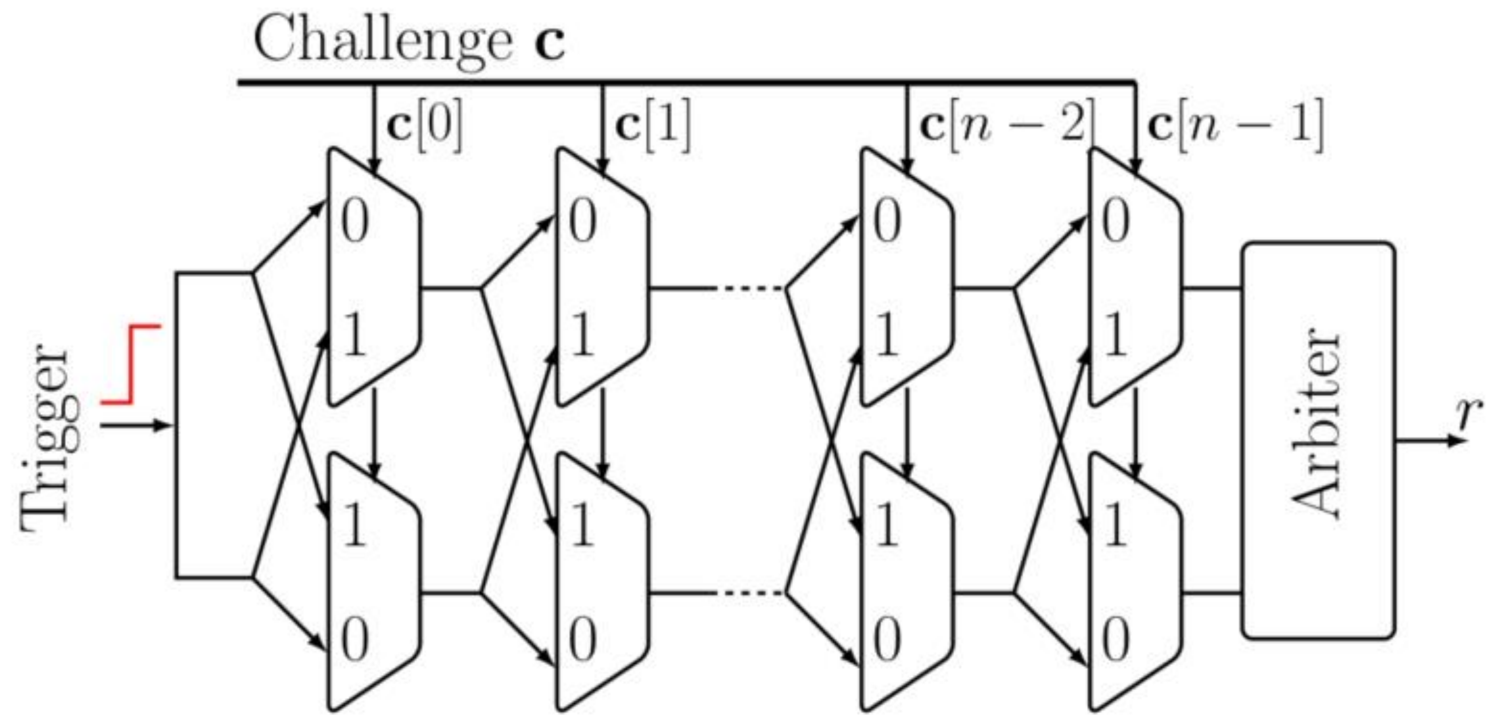


Trigger: *A=B*

Payload: *Sum=B when A=B*

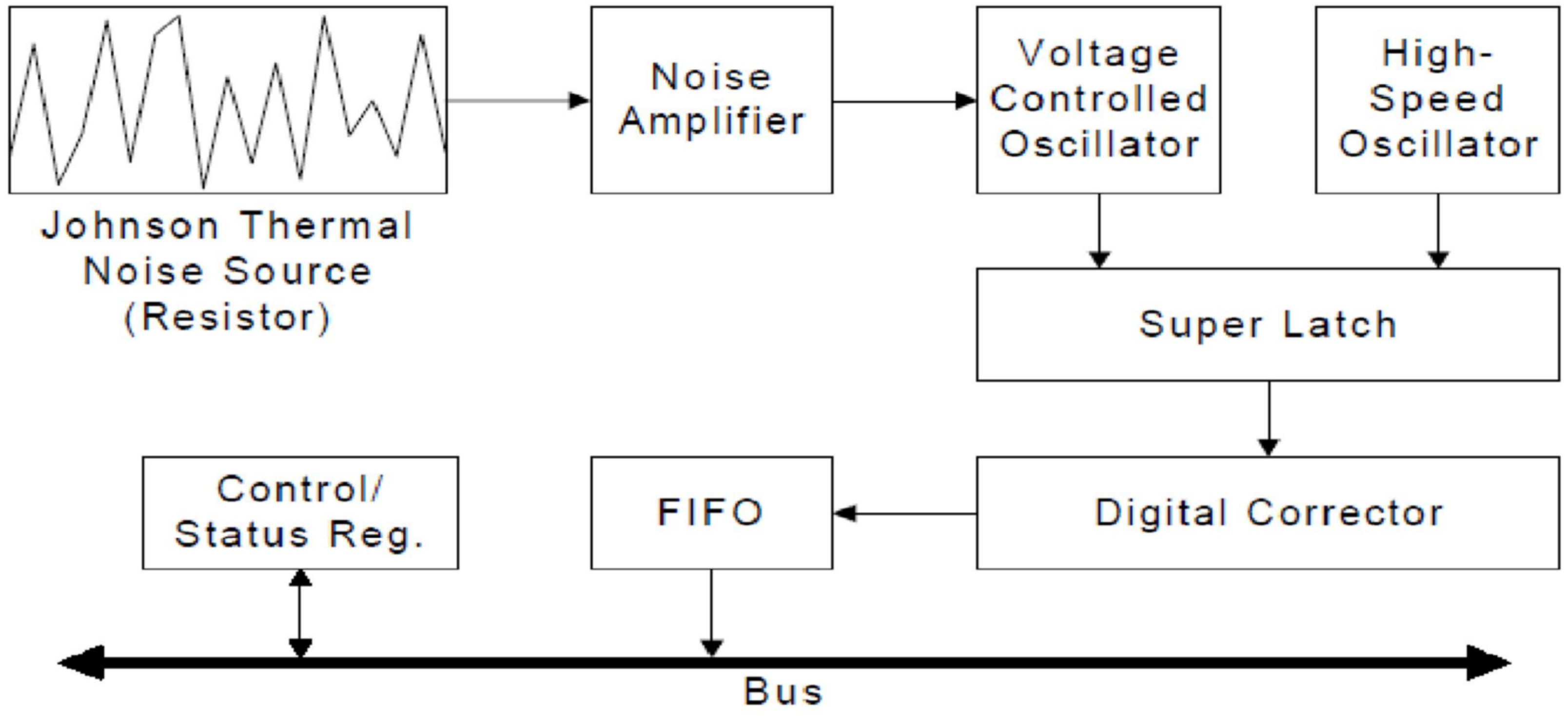
Key storage in HW
Identification and Authentication of HW

- We can use Phyiscal Unclonable Functions …. Lec11a, Lec11b



A n-stage classic Arbiter PUF with challenge $\mathbf{c} \in \{0,1\}^n$.

Computation often needs true random numbers (Lec12a)



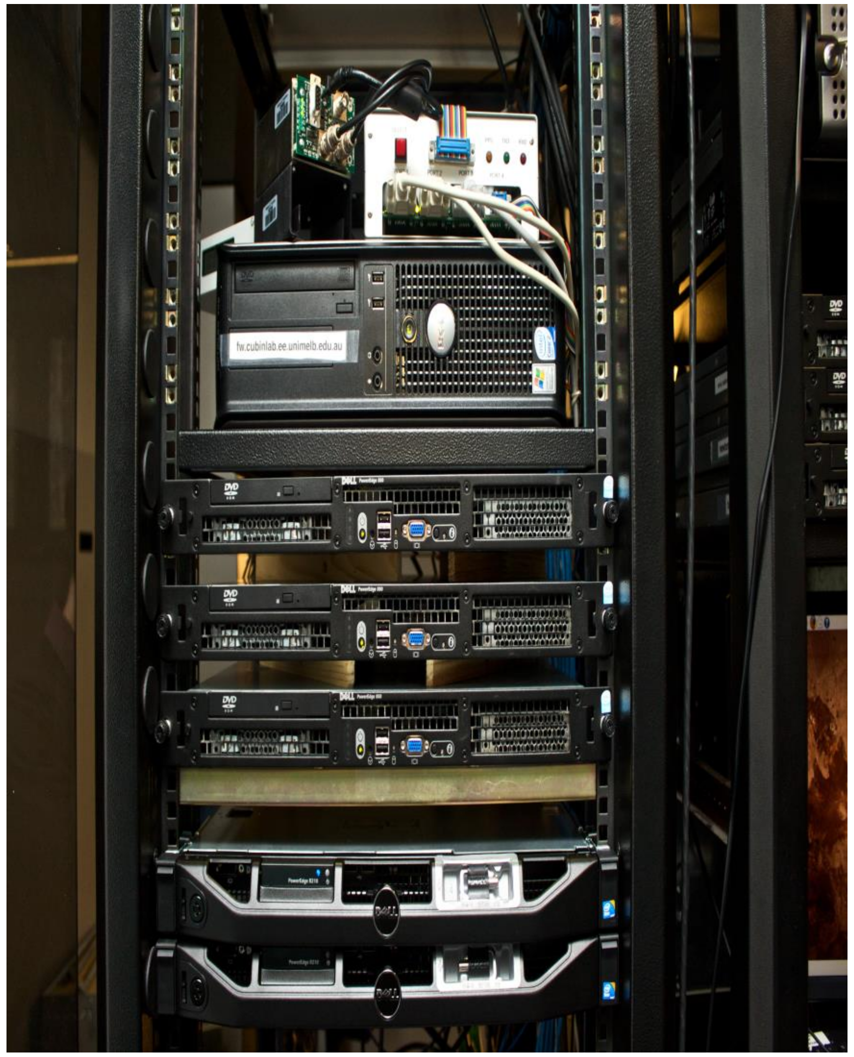
Block Diagram of Intel RNG

How are keys managed?

- We will not discuss this during this class … we simply assume this is done in a trusted way (public keys are signed)
- Together with an appropriate HW root of trust assumption, HW can provide isolation of computation

Scalability

- What is the scale of computation?
- Application programs run on top of an OS, hypervisor.
- Computation may be distributed over several nodes in a data center infrastructure.
- Small computations can be isolated, large ones need interaction across compute nodes.



Complex Infrastructures

- Smart power grid (Lec12b)
- Smart city (Lec13a)
- Data center
- Automotive systems
- …

Outline

- Lec2a + Lab1: A deeper understanding of how code is executed on a processor shows how e.g. a buffer overflow attack can be implemented
- Lec2b to Lec4b + Lab2: Secure processor architectures focusing on Intel SGX (with background in computer architecture and crypto, the lab implements a cache controller)
- Quiz
- Lec6a gives more background in crypto; Lec6b explains memory integrity checking in Intel SGX and Aegis (one of the first academic proposals); Lab3 implements memory integrity checking with caching
- Lec7a goes beyond Intel SGX taking its features and fixing security flaws with minimal HW modifications
- Lec7b explains Ascend with its (strong) adversarial model which requires an "Oblivious RAM" module

Outline

- Lec8a + Lec8b: Leakage of sensitive information over side channels. Lab4 is about the cache side channel
- Lec9a + Lec9b: Specific topics, in particular, supply chain management and hardware Trojans
- Quiz
- Lec11a + Lec11b: Physical Unclonable Functions for key storage and identification/authentication of HW. Lab5 implements a modeling attack
- Lec12a: True random number generator
- Lec12b + Lec13a: Power grid and smart city. Lab6 asks to write an essay on the security of automotive systems
- Lec13b to Lec14b: Various topics superficially covered (HaaS, voting machines, secure JTAG, reverse engineering, bitcoin HW, medical devices)
- Final

Grading

- Current proposal:
  - 60%: 2 Quizes, Final each count for 20%
  - 42%: 6 Labs each count for 7%

- This is a pilot course
  - Good news: This allows you to play a role in improving its material – where do we need additional explanations? Given your own background, you may need to study extra for certain lectures. If you want to capture the additional needed understanding in slides (to be inserted in existing lecture slides), you can suggest a proposal (and we will negotiate extra % bonus grade if completed successfully ☺).
  - Bad news: At some moments I may push too far, for some lectures I may have had less time to prepare … as the course design is still ongoing.
  - Let’s make it a success together!!!

Lots of Reading

- Yes, lots of required as well as suggested reading …
- Lec2a:
  - Aleph One, “Smashing the stack for fun and profit,” http://phrack.org/issues/49/14.html#article
  - Y. Younan, W. Joosen, and F. Piessens, “Runtime countermeasures for code injection attacks against C and C++ programs,” ACM Computing Surveys 44(3):1-28, June 2012
- Lec2b:
  - Ch 2 and App B in D. A. Patterson, “Computer architecture: A quantitative approach,” 5th edition
- Lec3a:
  - J. Rutkowska, “Intel x86 considered harmful,” 2015
  - “ORWL – The first open source, physically secure computer,” https://www.crowdsupply.com/design-shift/orwl
  - Chapter 5 in J. H. Saltzer and M. F. Kaashoek, “Principles of computer system design: An introduction”
- Lec3b:
  - V. Coston and S. Devadas, “Intel SGX explained,” https://eprint.iacr.org/2016/086.pdf

Lots of Reading

- Lec4b, Lec6a:
  - Sections 10.1 -- 10.4, 12.1 – 12.7 in J. Katz and Y. Lindell, "Introduction to modern cryptography"
  - Sections 4.1 – 4.7, 5.1 -- 5.5 in J. Katz and Y. Lindell, "Introduction to modern cryptography"
- Lec6b:
  - R. Elbaz, D. Champagne, C. Gebotys, R. B. Lee, N. Potlapally, and L. Torres, "Hardware mechanisms for memory authentication: A survey of existing techniques and engines," Transactions on Computational Science IV, LNCS 5340, 2009
  - G. E. Suh, D. Clarke, B. Gassend, M. van Dijk, and S. Devadas, "Efficient memory integrity verification and encryption for secure processors," MICRO 2003
- Lec7a:
  - V. Costan, I. Lebedev, and S. Devadas, "Sanctum: Minimal hardware extensions for strong software isolation," Usenix Security 2016
- Lec7b:
  - S. H. Kamran and M. van Dijk, "Flat ORAM: A simplified write-only oblivious RAM construction for secure processor architectures," https://arxiv.org/pdf/1611.01571.pdf

Lots of Reading

- Lec9a, Lec9b:
  - C. Jin and M. van Dijk, "Secure and efficient initialization and authentication protocols for SHIELD," https://eprint.iacr.org/2015/210
  - S. H. Haider, C. Jin, and M. van Dijk, "Advancing the state-of-the-art in hardware Trojan design," https://arxiv.org/abs/1605.08413
- Lec11b:
  - Notes to be written (lead: Phuong Ha Ngyuen)
- Lec13a to Lec14b:
  - E. Ronen and A. Shamir, "Extended functionality attacks on IoT devices: The case of smart lights," IEEE S&P 2016
  - J. Hennessey, C. Hill, I. Denhardt, V. Venugopal, G. Silvis, O. Krieger, and P. Desnoyers, "Hardware as a service – enabling dynamic, user-level bare metal provisioning of pools of data center resources," HPEC 2014
  - S. Davtyan, A. Kiayias, L. Michel, A. Russel, and A. A. Shvartsman, "Integrity of electronic voting systems: Fallacious use of cryptography," SAC 2012
  - M. B. Taylor, "Bitcoin and the age of bespoke silicon," CASES 2013
  - D. Kotz, K. Fu, and A. Rubin, "Security for mobile and cloud frontiers in healthcare," Communications of the ACM, 2015

How to study??

- Slides and required reading is Quiz and Final material
  - Lectures are about the core of the reading material: you need to be well-prepared – this requires independence and responsibility
  - Since slides will often be finalized the night before … you cannot rely on looking at these before lecture
  - Try to limit to 2 hours per reading assignment – don't go overboard trying to understand every single detail: I myself do not know every single detail by heart; We test conceptual understanding (possibly with details if we provide extra info from which such detail can be extracted).
- Labs are all about implementing and simulating HW modules and attacks
  - May require 7-10 hours each week
  - Collaborate !! Set up a joint weekly study group – share ideas – help one another!
  - Ask for help: Phuong Ha Nguyen will be our TA and will hold OH (he is traveling till Jan 31st)