(Some Thoughts on) Cloud Security

Gabriel Scalosub

Outline

- Introduction
  - Concerns and Threats
- Approaches to Securing the Cloud
- Concrete Example: Cache Side-Channel Attacks
  - Framework and General Methods
  - Some Variants

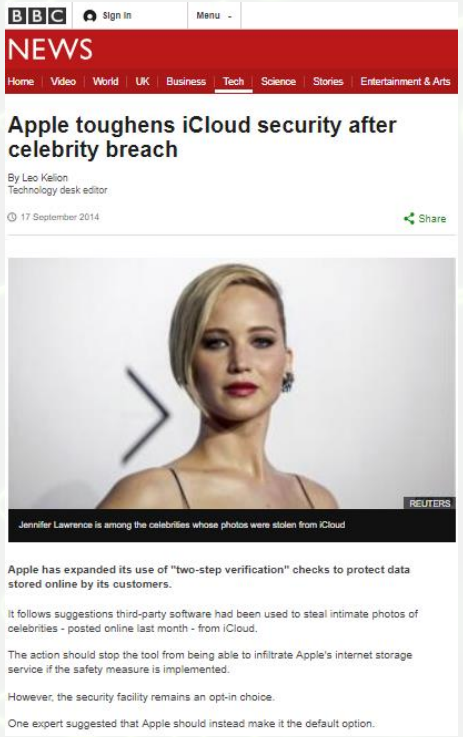
Security: (The) Major Concern in Cloud

- Multi-tenancy vs. isolation
  - It's all about the money, remember?
- Resource sharing
  - Physical resources
  - SW libraries (kernel host)
  - Networks
- VMs vs. containers:
  - Stronger isolation (VMs)
  - More agility (containers)
  - Attack surface: large (VMs) vs small(er) (containers)
  - A lot of "philosophical" debate…

Security: (The) Major Concern in Cloud

- Data breaches
- Insecure APIs
  - E.g., Web Applications
    - Cross-side scripting, SQL injection, …

      http://vulnerable-website.com/search?search_term="<script>(bad things happen here)</script>"
- Distributed denial-of-service (DDoS)
- Side channel attacks
  - Stay tuned…
- Many more



BBC Sign in Menu

NEWS

Home | Video | World | UK | Business | Tech | Science | Stories | Entertainment & Arts

Apple toughens iCloud security after celebrity breach

By Leo Kelion
Technology desk editor

17 September 2014 Share

REUTERS

Jennifer Lawrence is among the celebrities whose photos were stolen from iCloud

Apple has expanded its use of "two-step verification" checks to protect data stored online by its customers.

It follows suggestions third-party software had been used to steal intimate photos of celebrities - posted online last month - from iCloud.

The action should stop the tool from being able to infiltrate Apple's internet storage service if the safety measure is implemented.

However, the security facility remains an opt-in choice.

One expert suggested that Apple should instead make it the default option.

Source: www.bbc.com

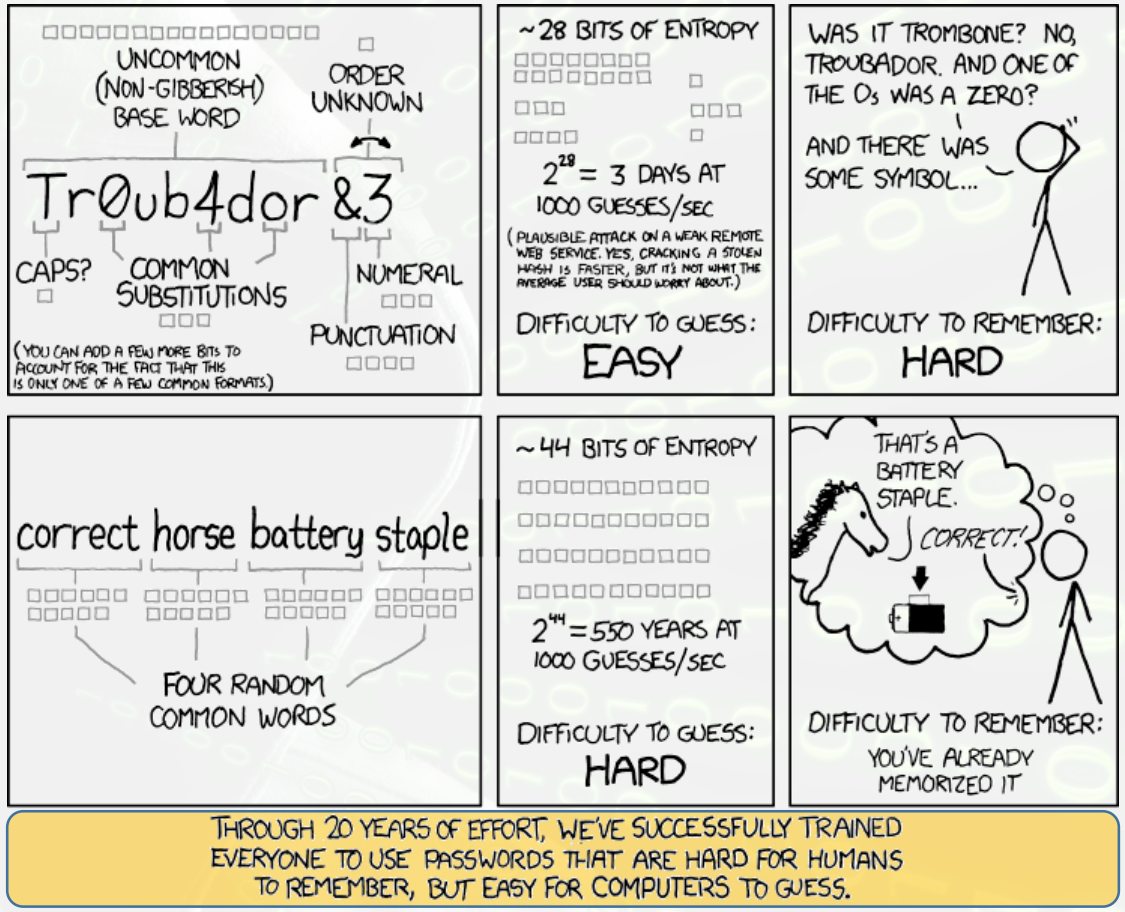
- Is that it?
- What about trust?
  - Can I trust my cloud provider with my data?

Outline

- Introduction
  - Concerns and Threats
- Approaches to Securing the Cloud
- Concrete Example: Cache Side-Channel Attacks
  - Framework and General Methods
  - Some Variants

Data-centric Security

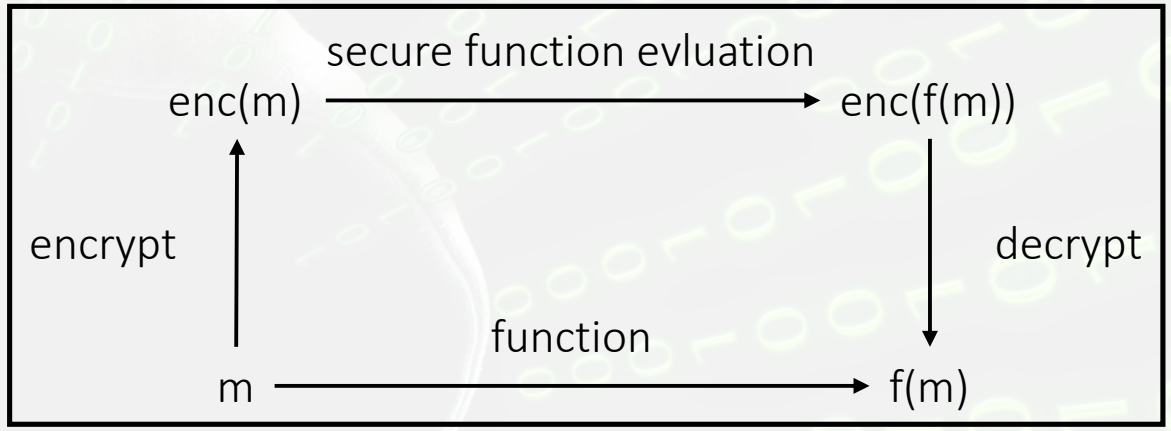
- Application-level
  - Authentication
    - 2-phase (a password is not enough)
  - Identity management & credentials
  - Data-loss
    - Backups…
  - Risk management
    - Supply chain & 3rd party
    - Know thy-own vulnerabilities
  - Usually based on cryptography
- Also system-level
  - E.g., secret sharing (information theoretic)



Source: https://www.explainxkcd.com/

Secured Cloud Data: Homomorphic Encryption

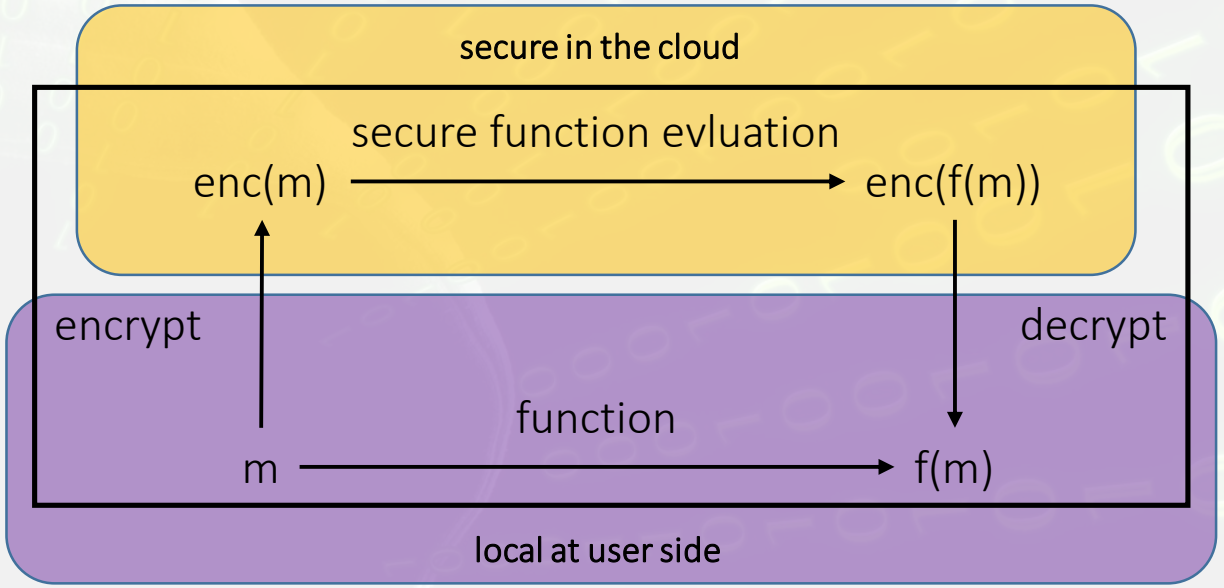
- Allows computation on ciphertext
- Simple example (not really secure…):
  - Encryption by $E(x) = 2x$
  - Decryption by $D(x) = x/2$
  - Plaintext inputs $x_1 = 3$ and $x_2 = 4$
    - Encrypted ciphertext data: $E(x_1) = 6, E(x_2) = 8$
    - Operation on ciphertext: $2 * E(x_1) + 3 * E(x_2) = 36$
    - Decrypt operation on ciphertext: $D\big(2 * E(x_1) + 3 * E(x_2)\big) = 36/2 = 18 = 2 * x_1 + 3 * x_2$



- Real example:
  - Private patients data encrypted in a DB in the cloud
  - Pharmaceutics want to compute statistics on patients data (e.g., AVG, distribution,…)
  - Draw conclusions from statistics, without access to plaintext data, only to ciphertext

Secured Cloud Data: Homomorphic Encryption

- Scheme requirement:
  - Secure!! (crypto / information theoretic)
- Partial scheme:
  - Supports just one operation (e.g., +)
- Full scheme:
  - Supports two operations (e.g., +, ·)
  - Enables computing any function
  - Current SOTA: very slow compared to plaintext operation
    - More than $\times 10^3$ slower…
- Applications:
  - Secure voting
  - Private information retrieval (PIR)
    - Query a DB without anyone knowing what your *query* was
  - Machine learning on anonymized data



Infrastructure-centric Security

- Firewalls
- Intrusion detection systems (IDS)
- Antivirus (AVs)
- (No)-deduplication
  - Duplicate shared libraries -> deduplicate (more efficient) -> no de-deuplication
  - Increase isolation
    - Less efficient
- Secure against VM escape
  - VM is able to access/attack the host-OS / hypervisor
  - Obtain privileged access
    - Hypervisor / co-located VM

Outline

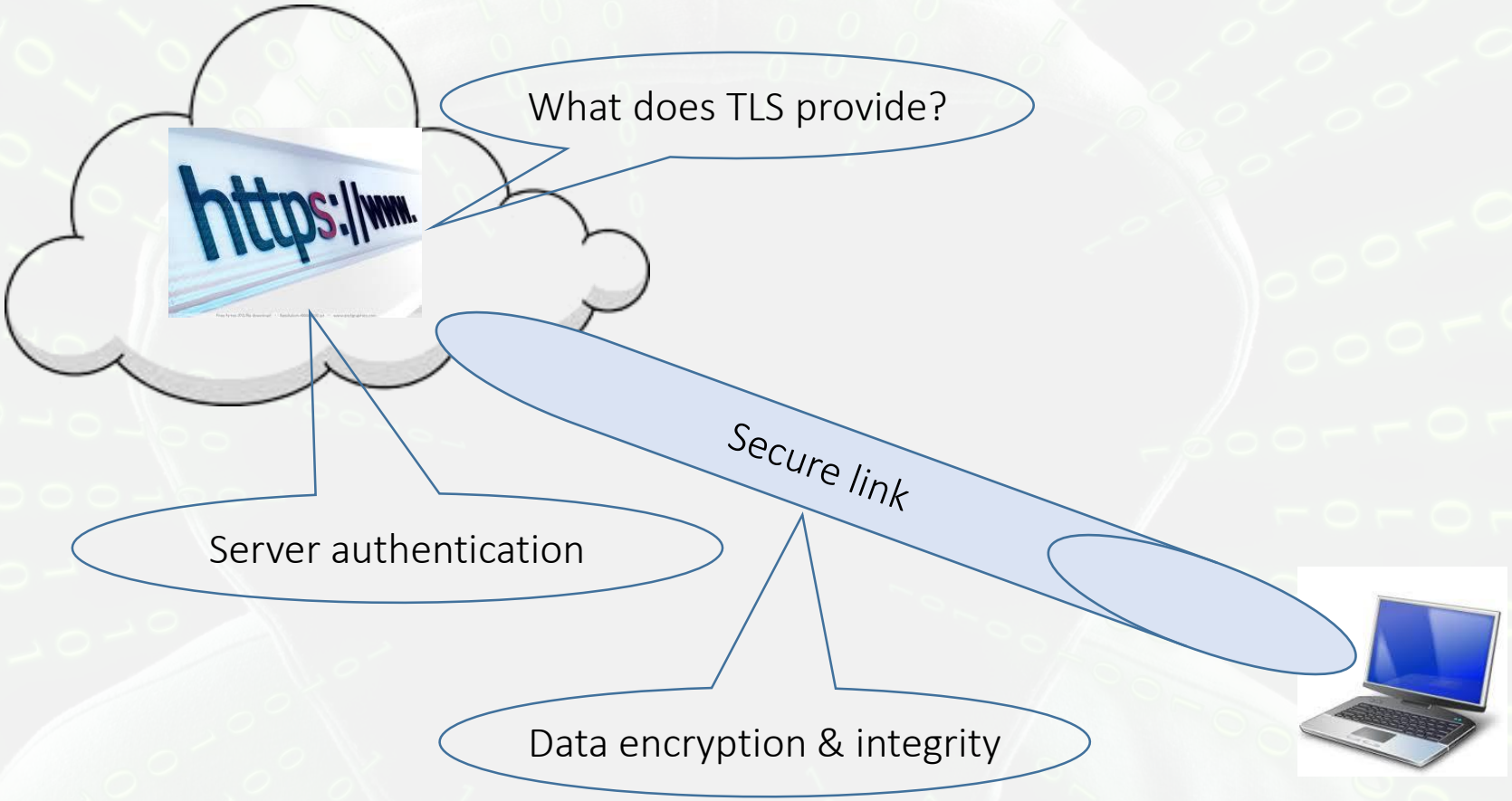
- Introduction
  - Concerns and Threats
- Approaches to Securing the Cloud
- Concrete Example: Cache Side-Channel Attacks
  - Framework and General Methods
  - Some Variants

Transport Layer Security (TLS) / HTTPS

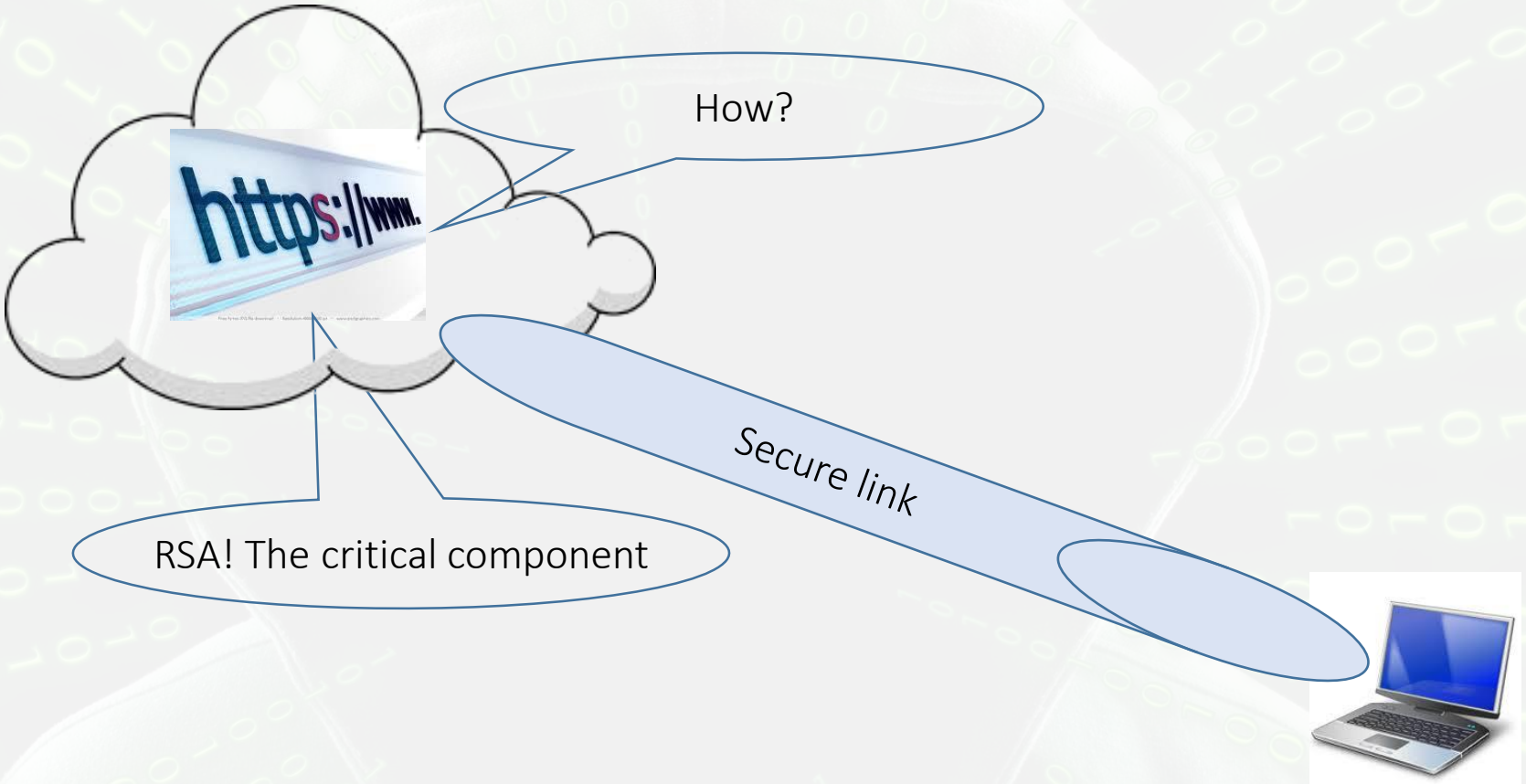
Where is it used?

Here and there…

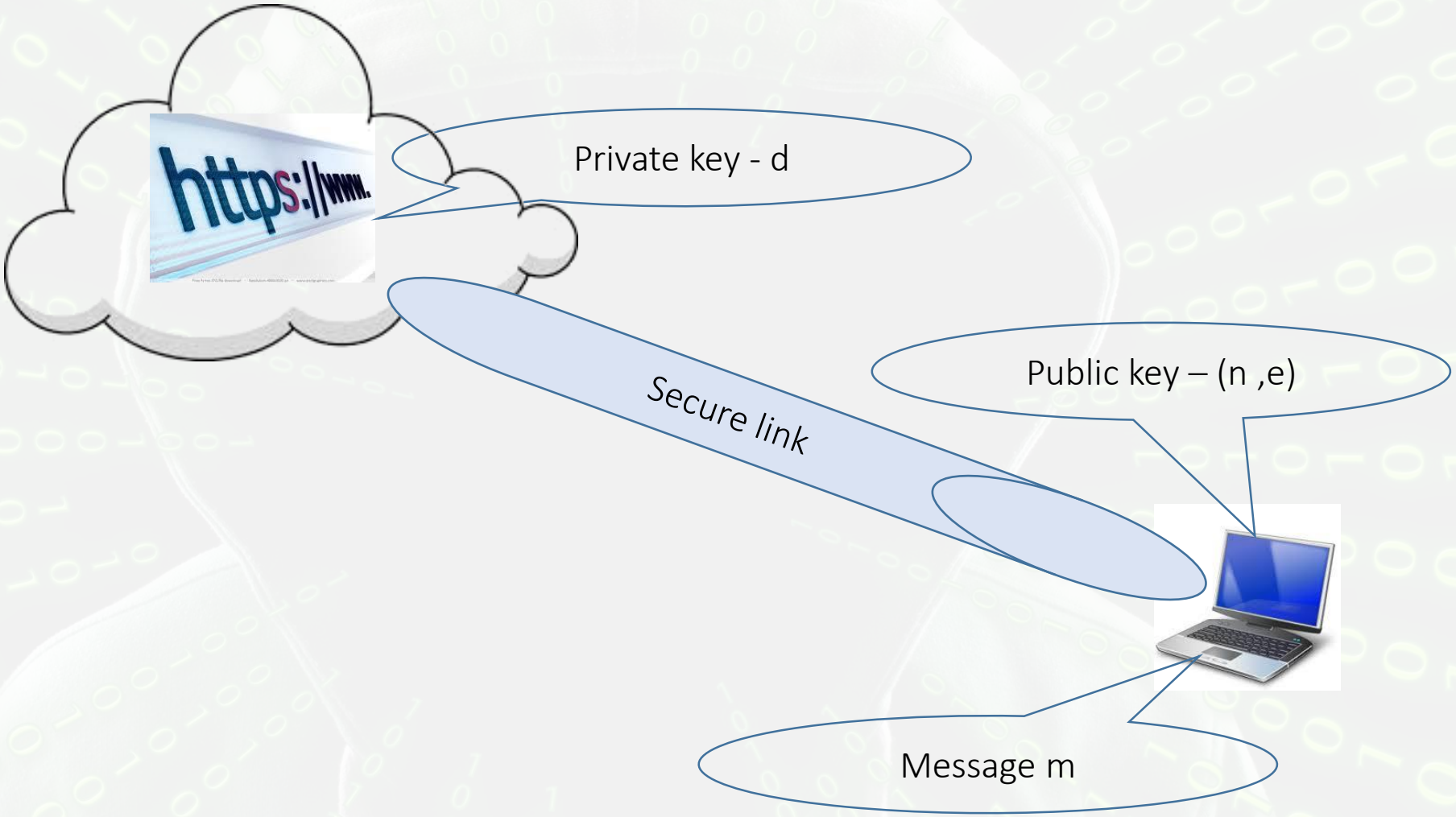
Transport Layer Security (TLS) / HTTPS



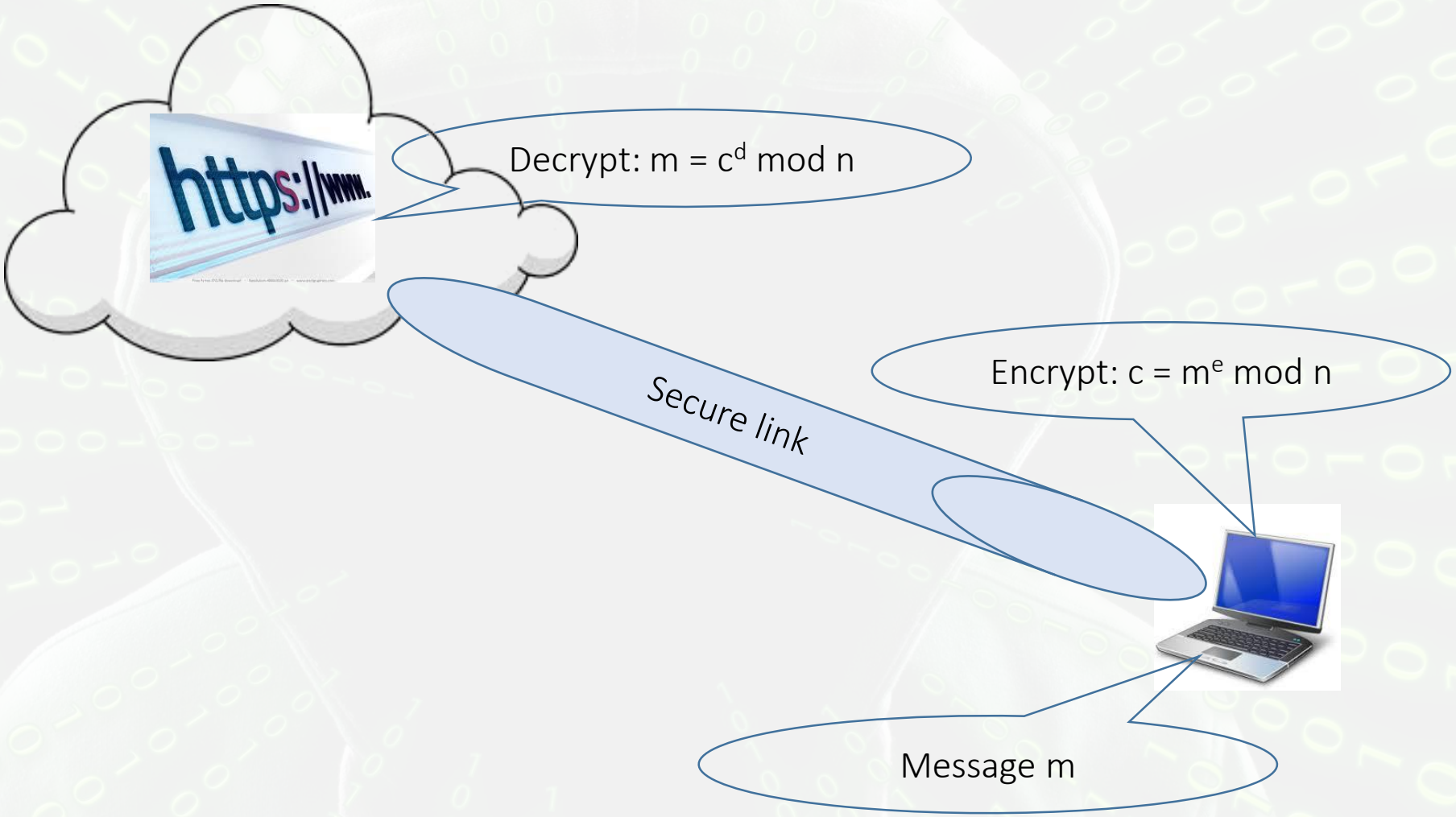
Transport Layer Security (TLS) / HTTPS



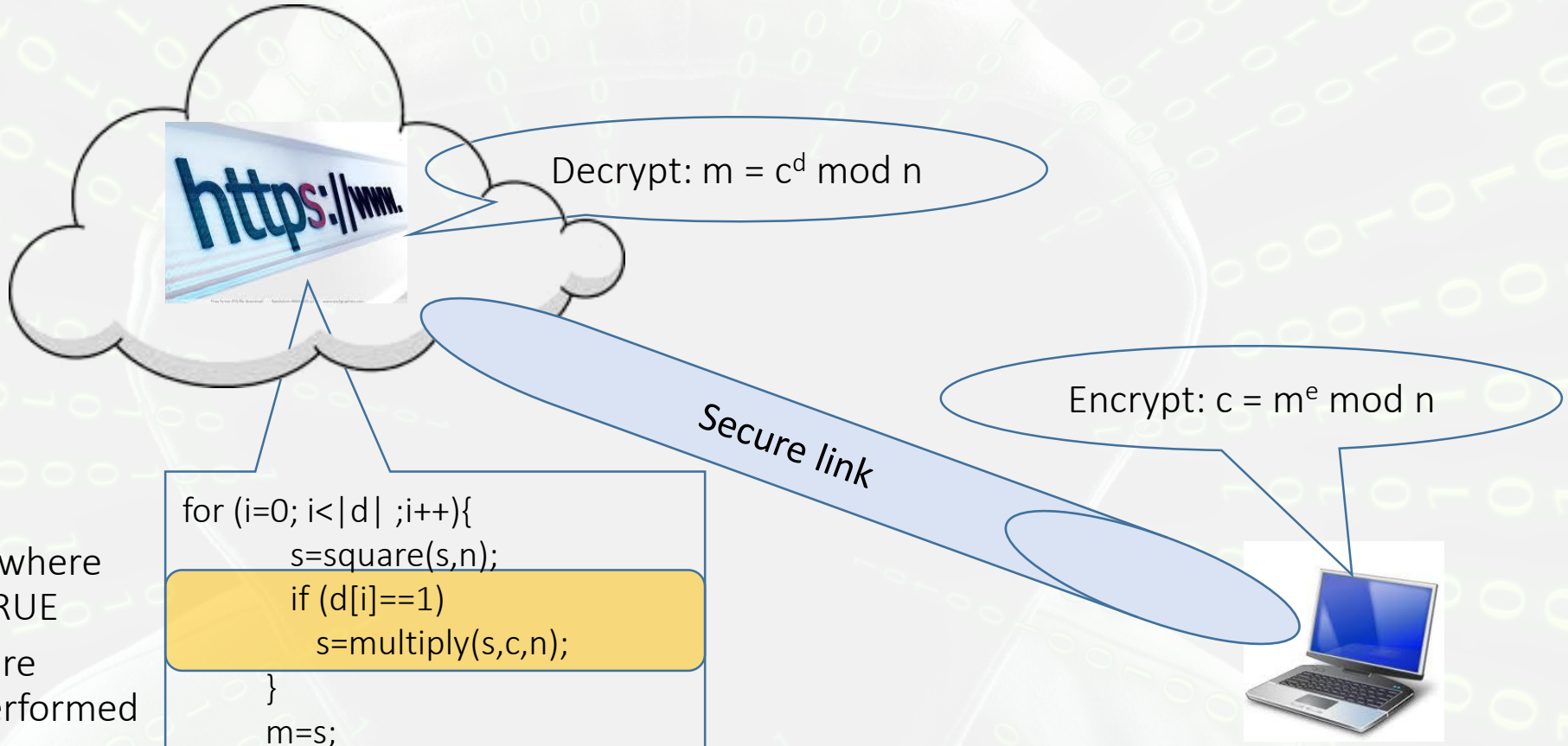
TLS / HTTPS (Specifically, RSA)



TLS / HTTPS (Specifically, RSA)



TLS / HTTPS (Specifically, RSA)

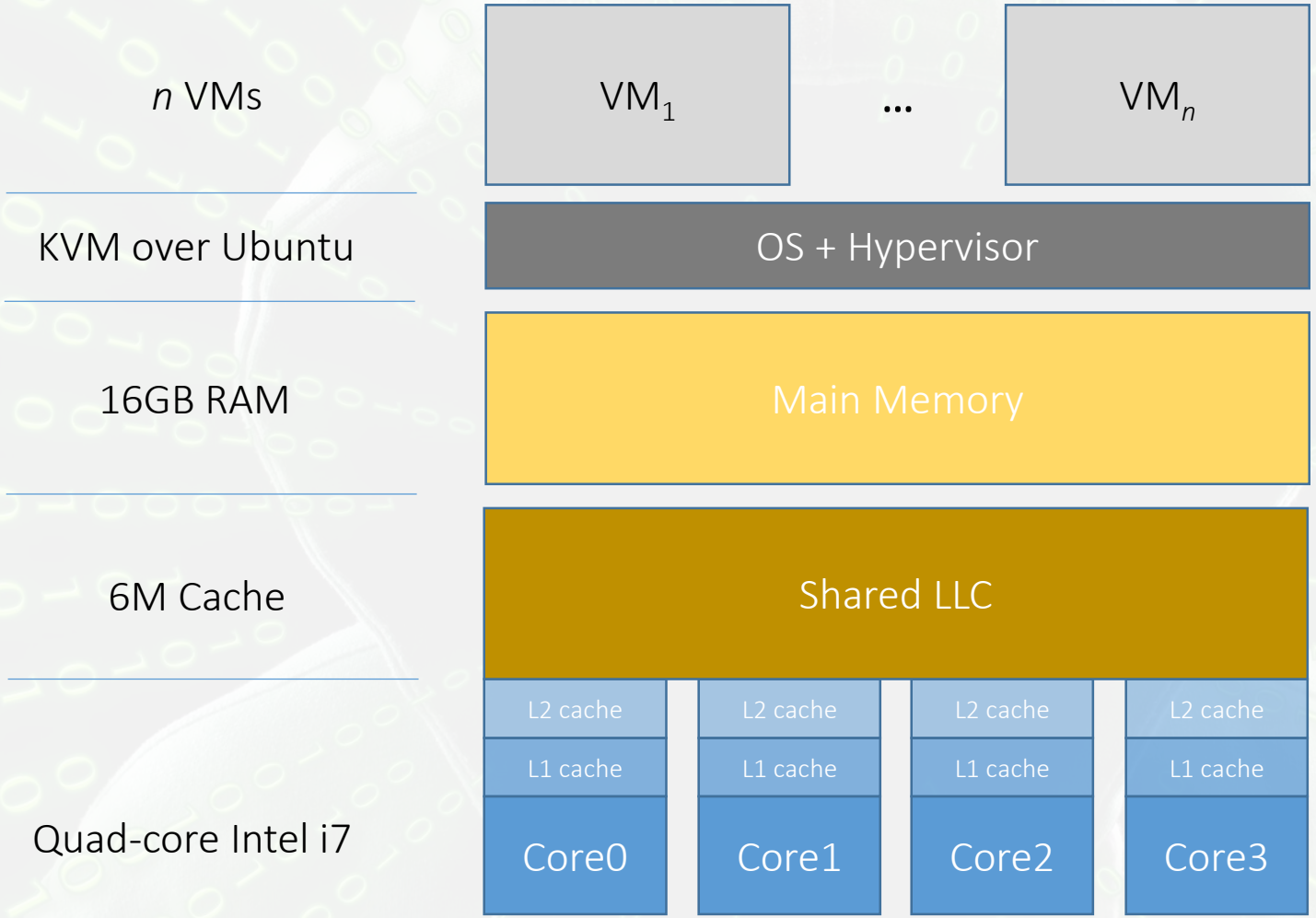


- Goal:
  - Identify d-bits where statement is TRUE
  - I.e., d-bits where multiply() is performed

```
for (i=0; i<|d| ;i++){
        s=square(s,n);
        if (d[i]==1)
          s=multiply(s,c,n);
    }
    m=s;
```

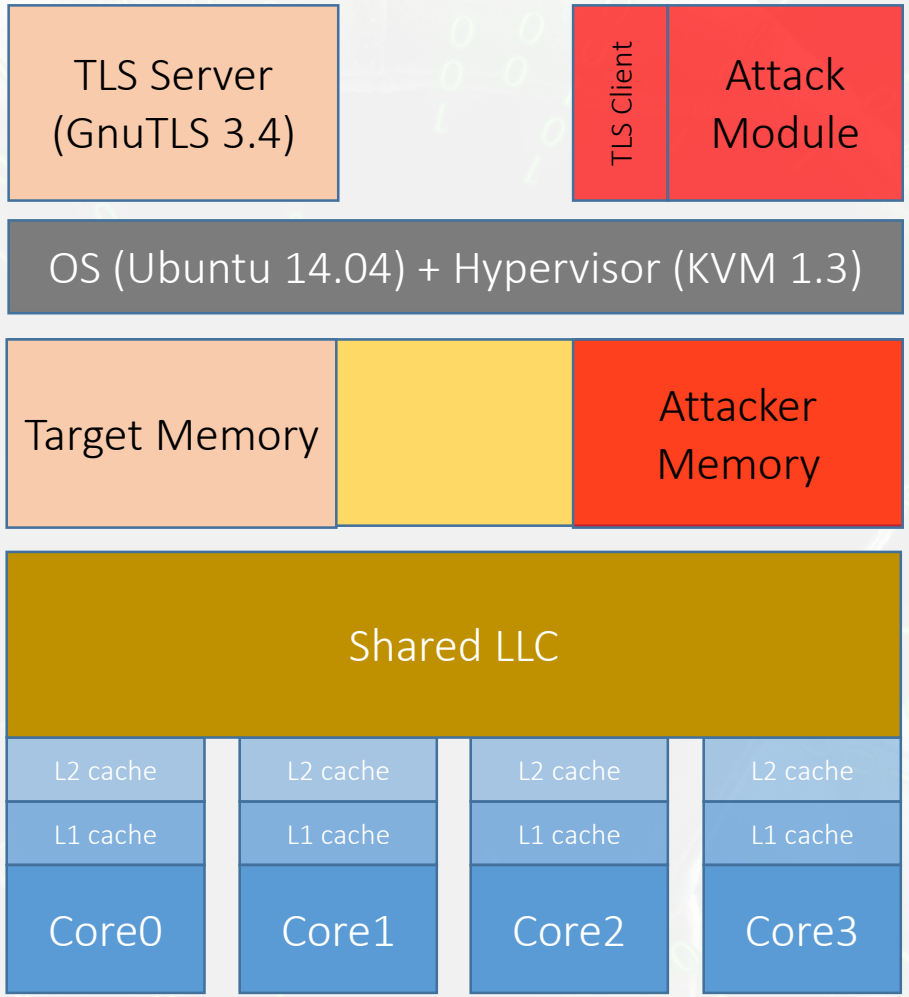
Typical implementation of Decrypt operation

A Concrete System

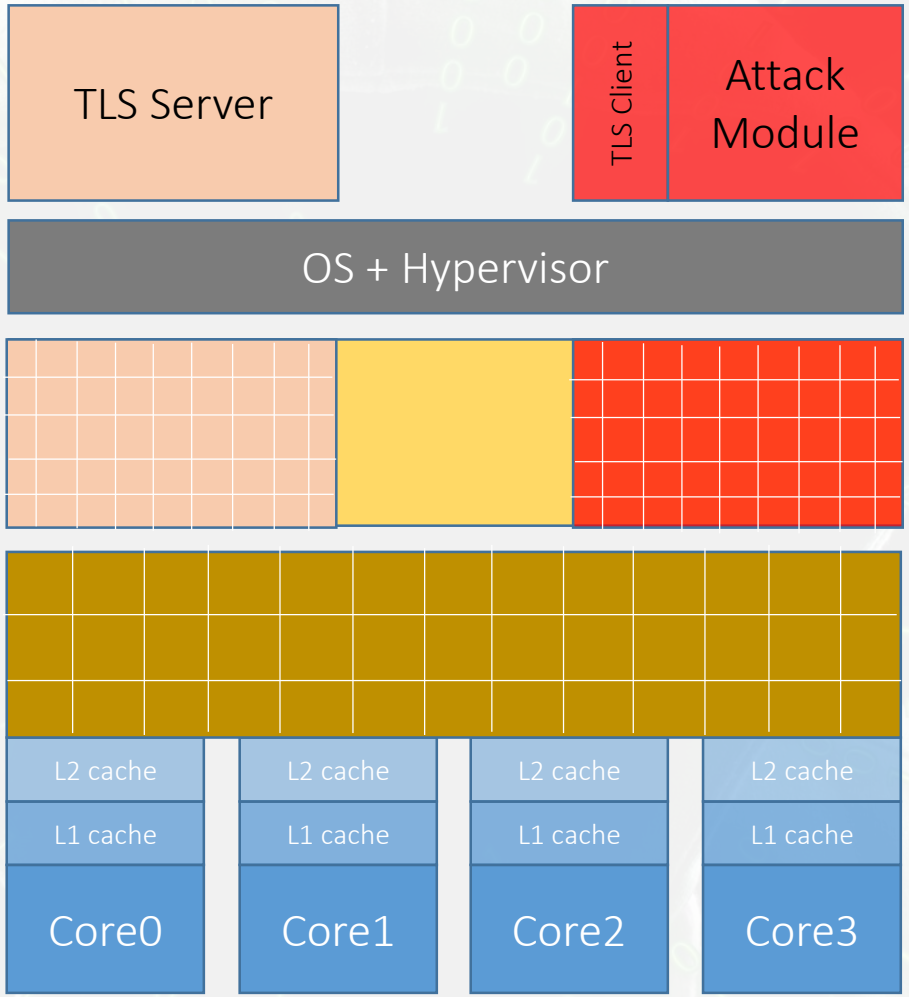


A Concrete System

Secret key:
|d|=4096 bit



A Concrete System

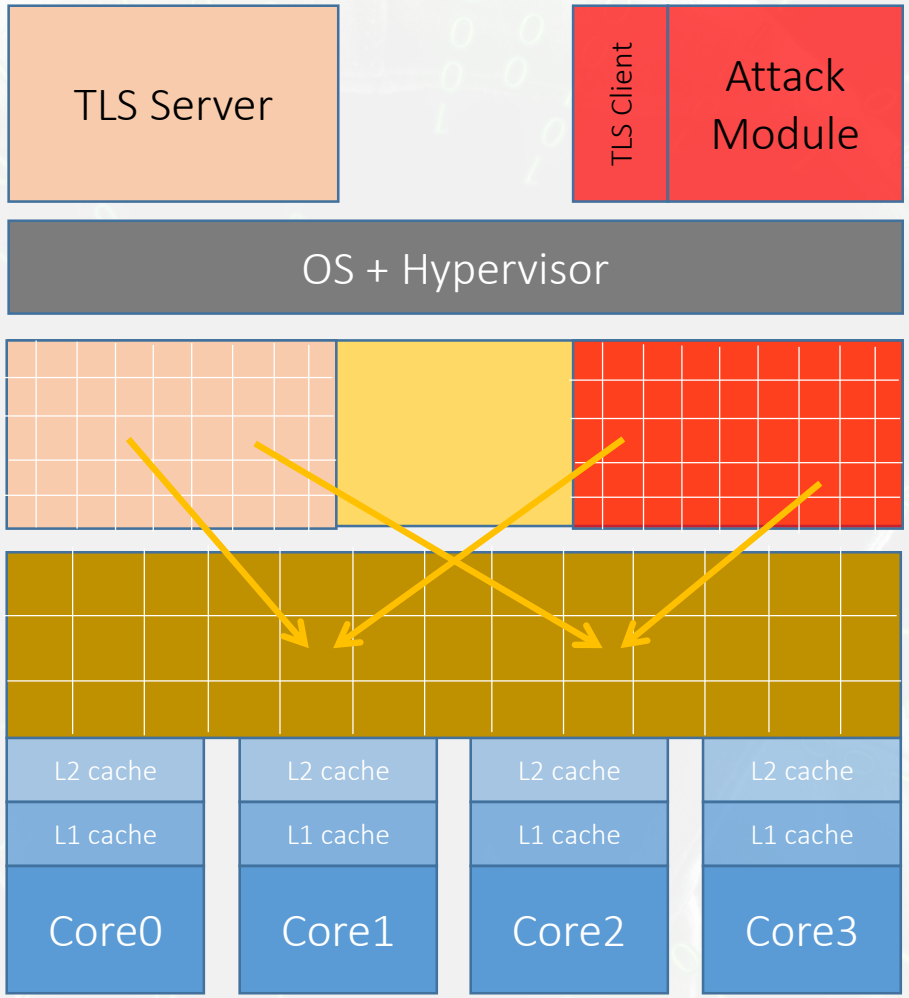


6M Cache = 8192 sets
12-way associative

- 12 lines / set

A Concrete System: Memory vs. Cache

Many-to-one



A Concrete System: Memory vs. Cache

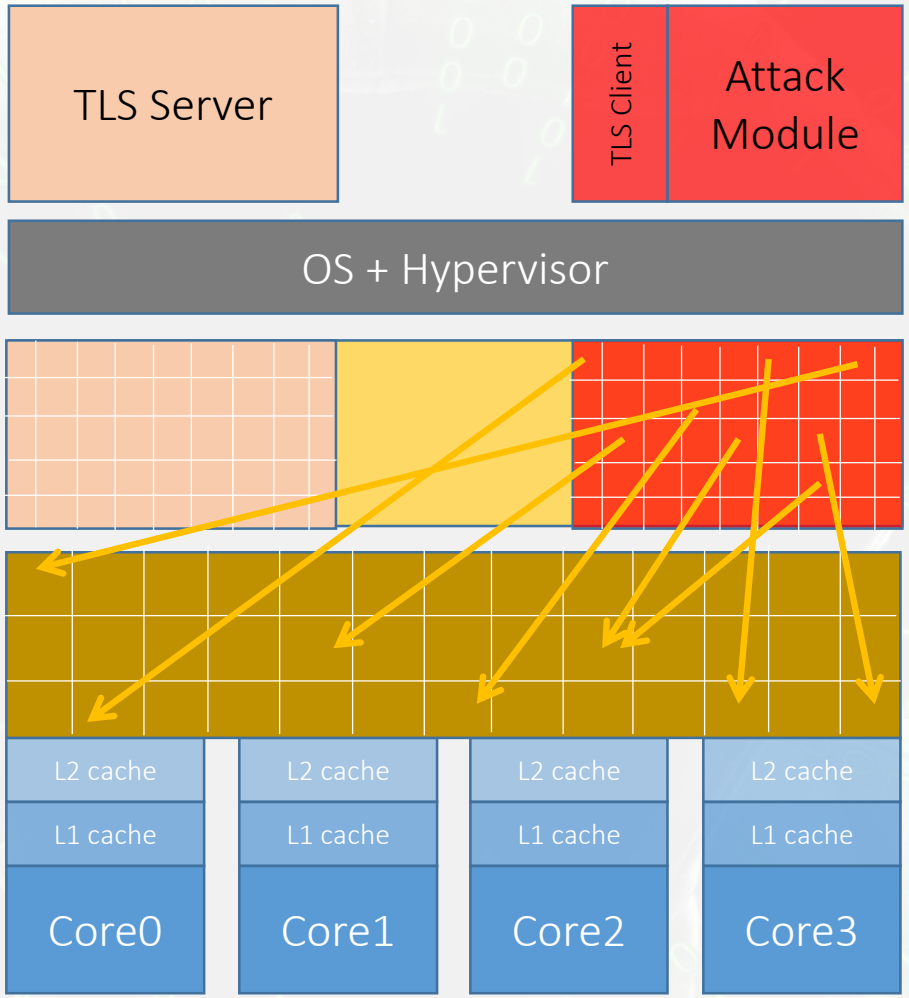
Attacker knows
its own mapping

Builds collection of
12 lines / set

Enables the attacker to
“Flush” entire set

Prime&Probe:

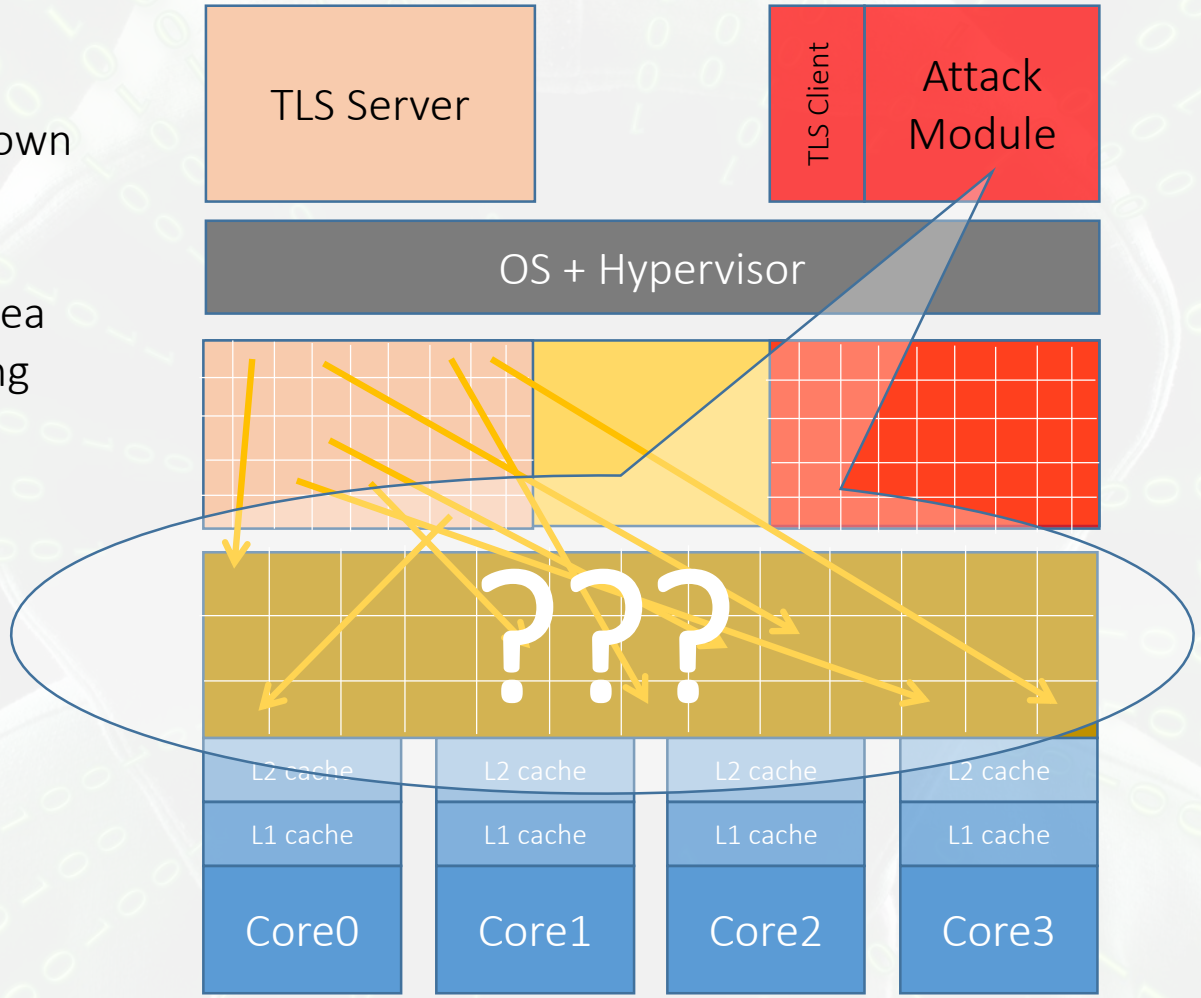
- Prime: “Flush”
- Wait (for target activity)
- Probe: check for miss



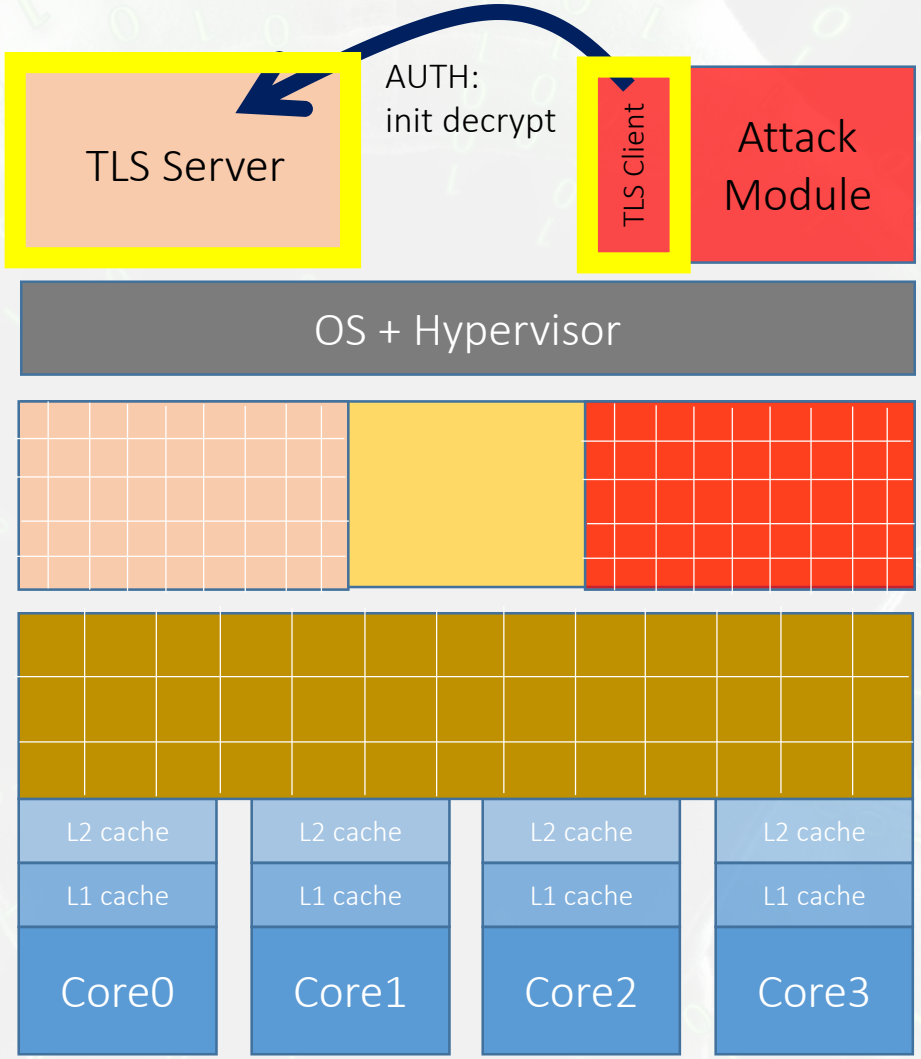
A Concrete System: Memory vs. Cache

Target also has it own mapping

Attacker has no idea of target's mapping



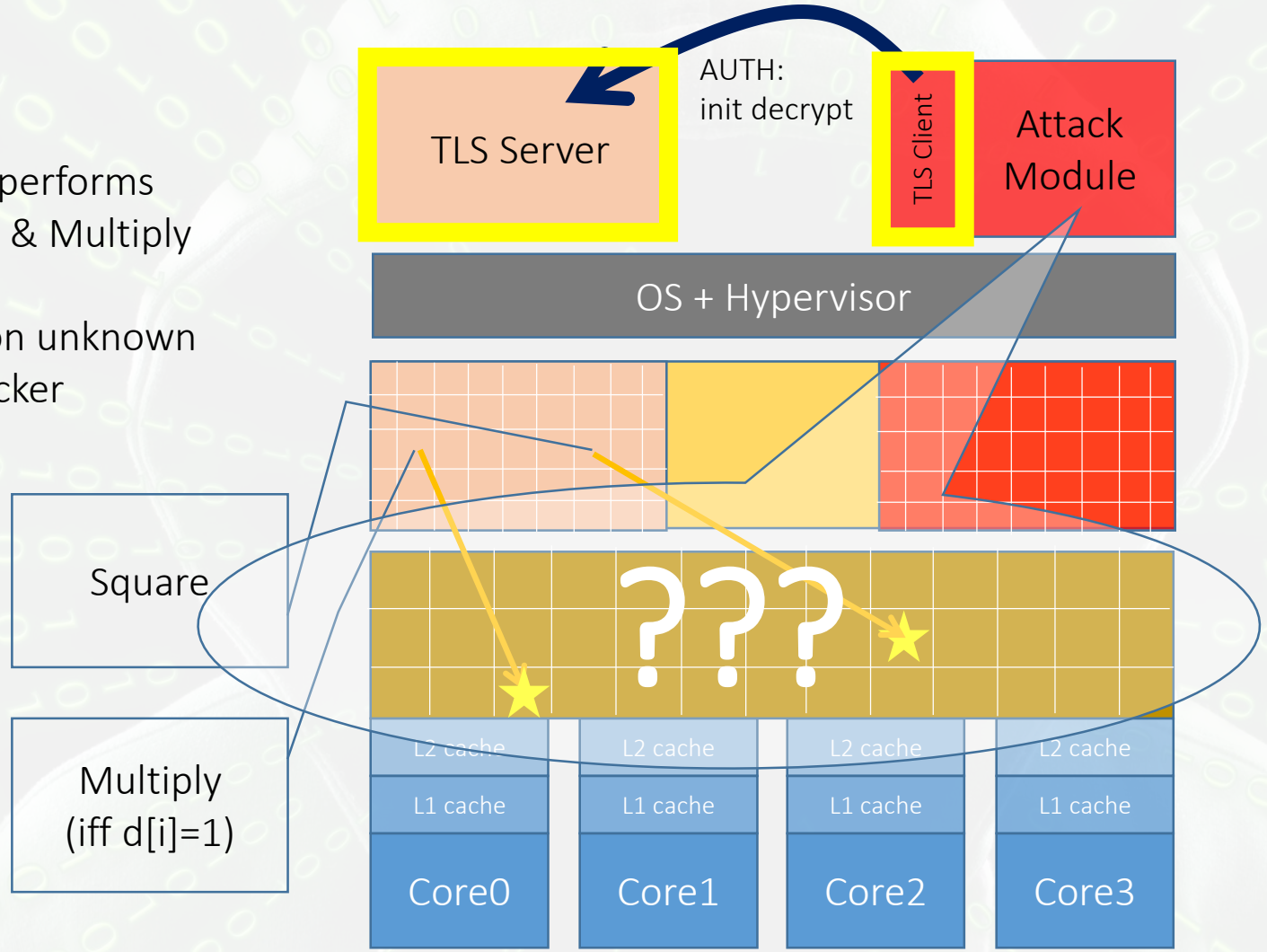
Attack Initialization



Attack: Under the Hood

Target performs
Square & Multiply

Location unknown
to attacker



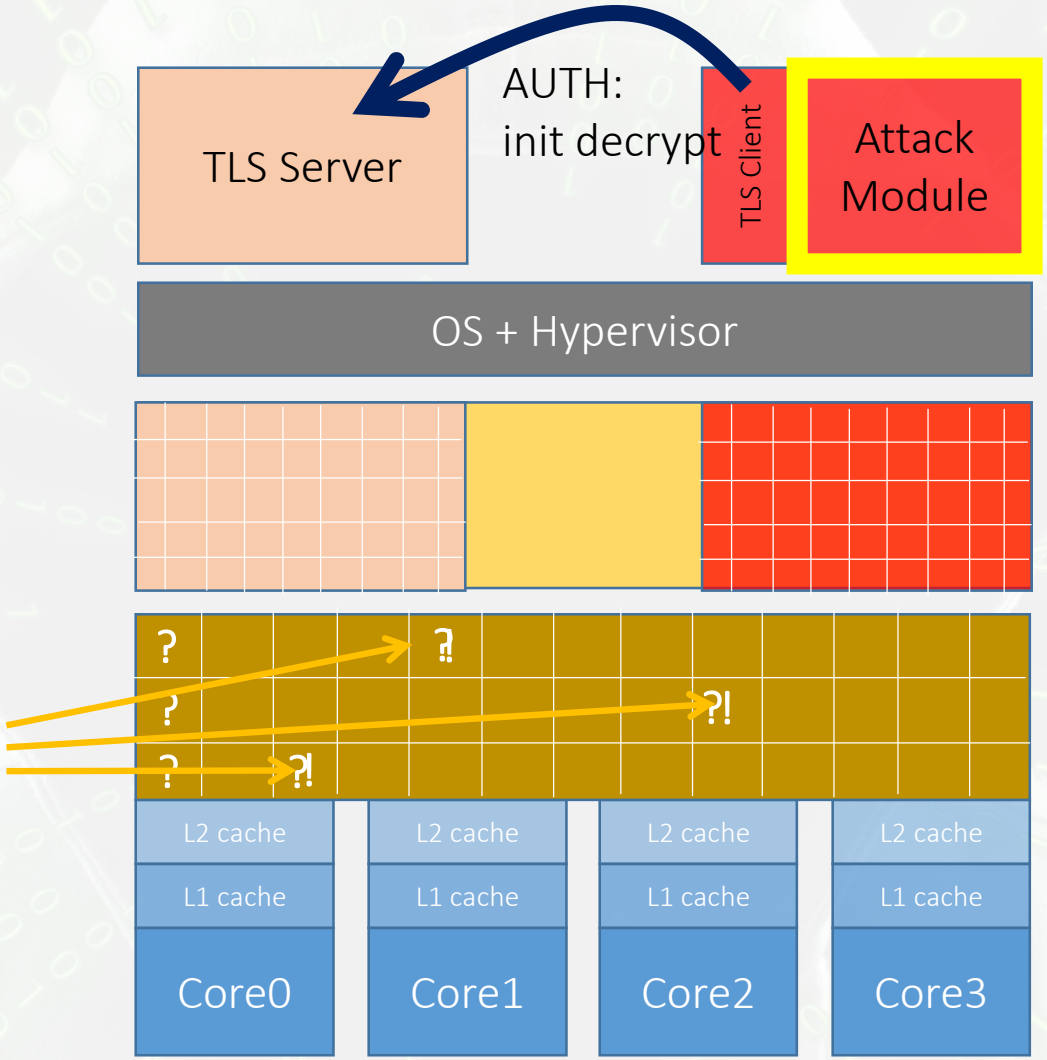
Attack: Prime & Probe

Attack Module:

Scans cache sets for activity

Prime & Probe
Cache hit/miss

Identify active sets (by score)

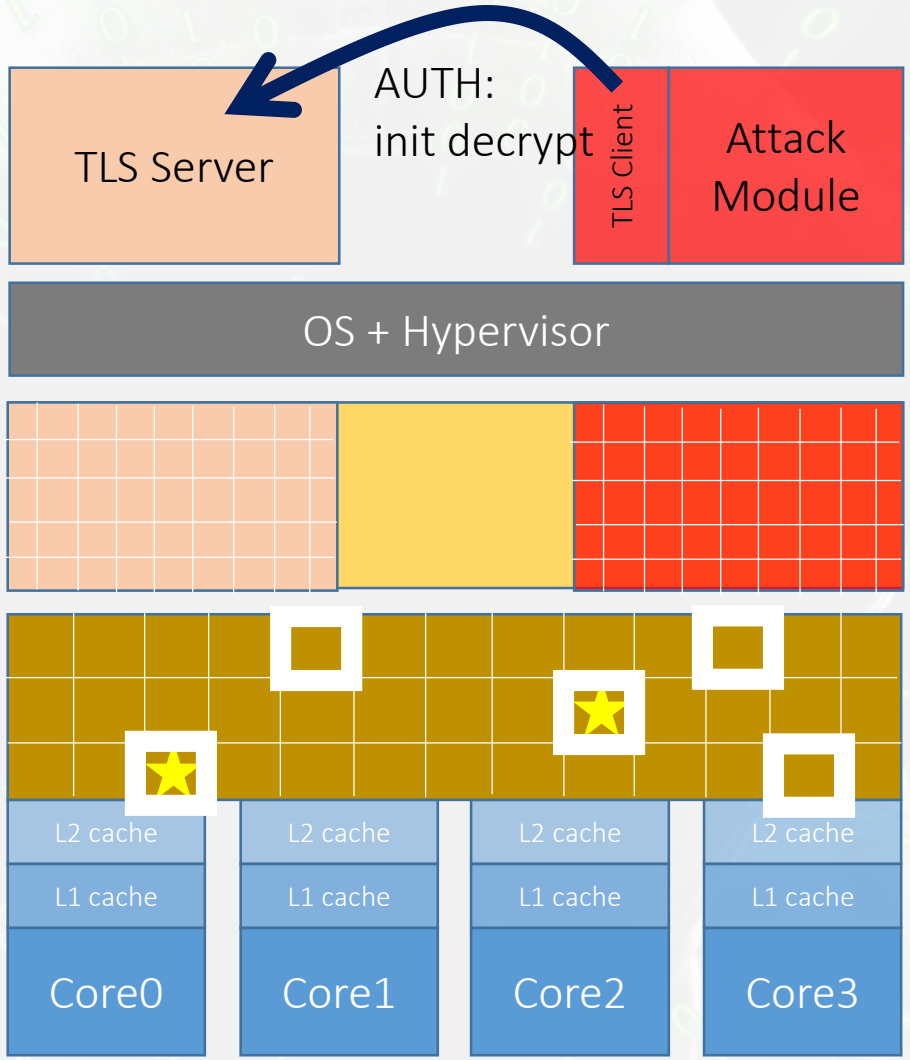


Attack: Prime & Probe

List of active sets

Attacker doesn't know which are "correct"

Test each active set for signature traces

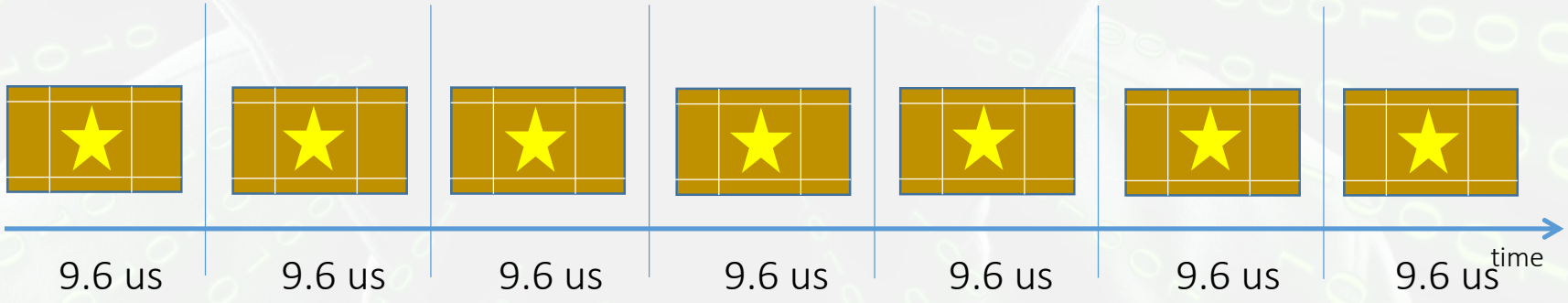


Attack: Extract Samples

Suspected “Multiply” set

Used if-and-only-if secret key bit d[i]=1

Target takes 9.6us to process a single d-bit

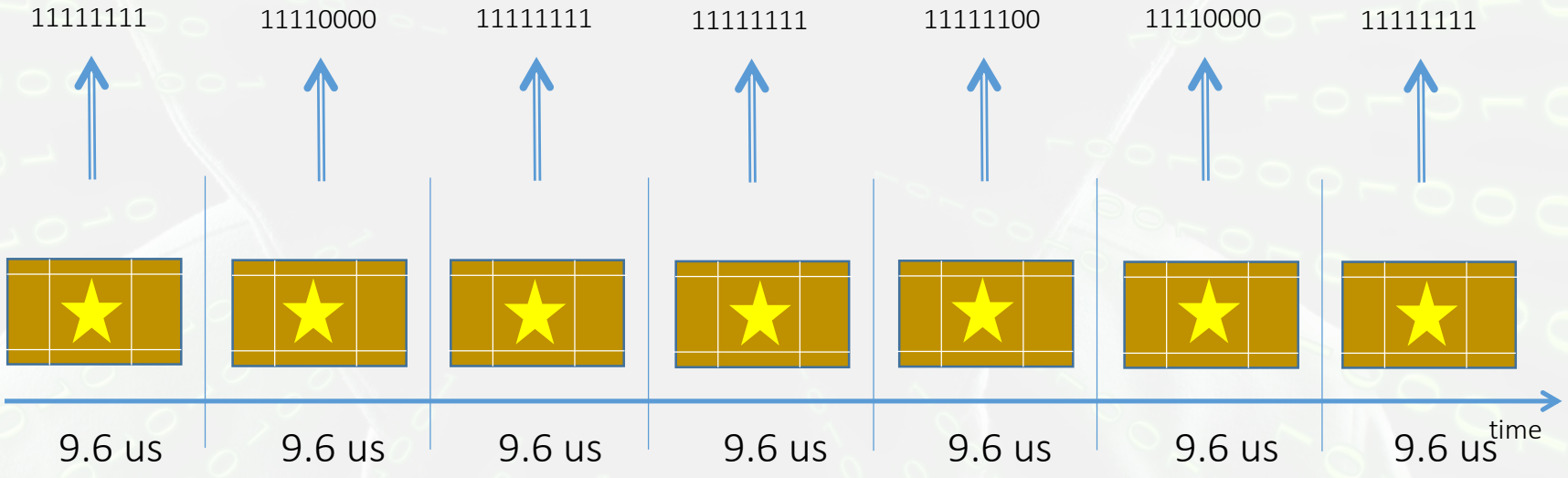


Attack: Extract Samples

Sample: Prime&Probe

Extract samples from each d-bit

8 samples per d-bit
(one sample every 1.2us)

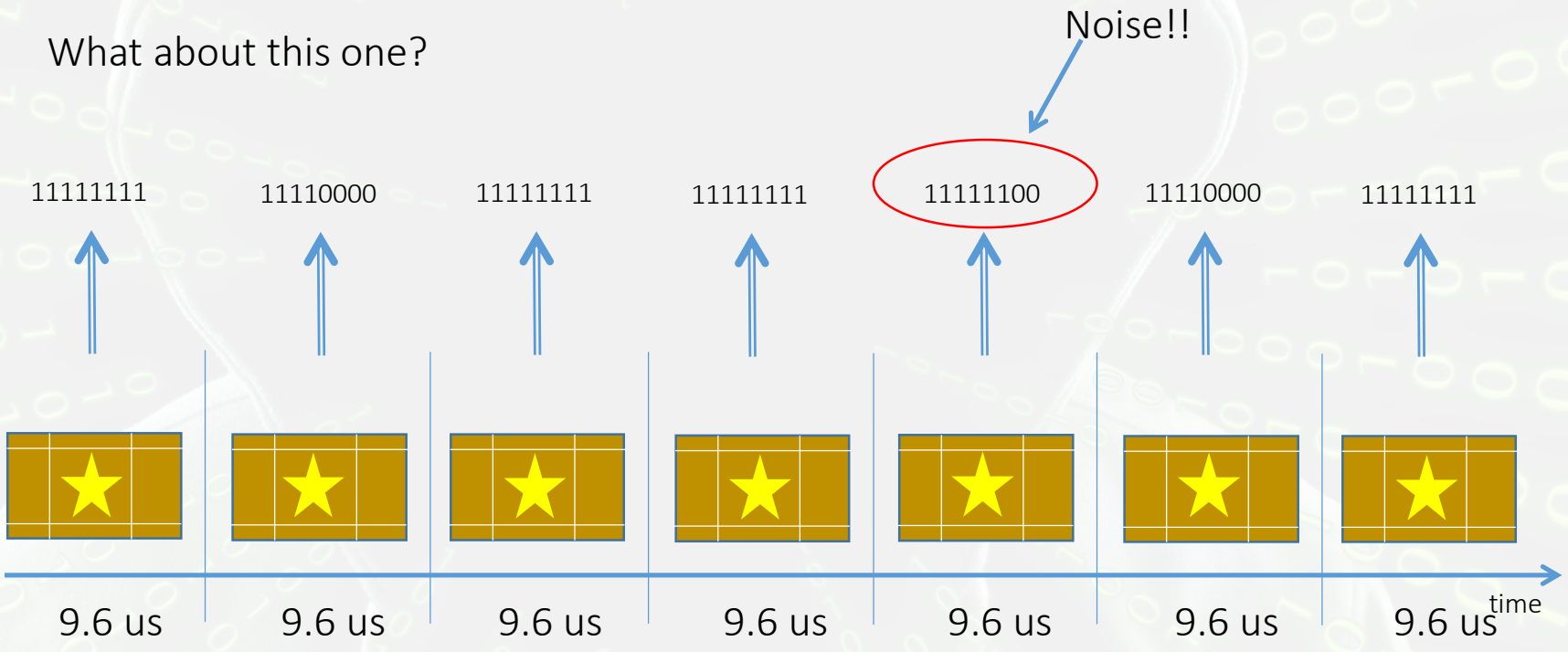


Attack: Extract Samples

Good vs. bad patterns

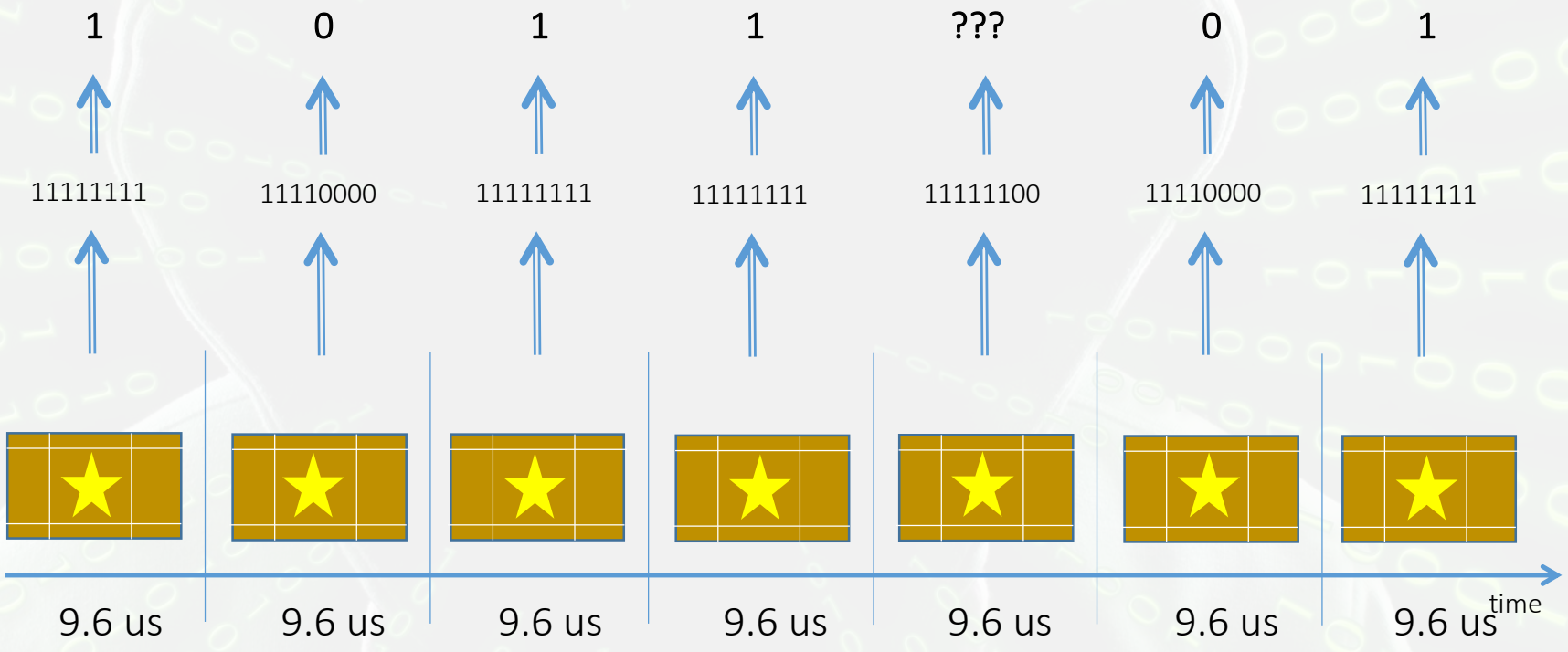
Can you tell which are which?

What about this one?



Attack: From Samples to Bits

Translate samples to candidate d-bits



Attack: From Bits to Complete Keys

“Good” candidate sequences:

- Long, few missing bits

1 1 1 0 1 0 0 1 0 0 0 1 0 1 0 0 1 0 0 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 1 0 0 1 1 1 1 1 0 0 0 1 0 1 0 0 0 1 1 1 0 1 0 1 0 0 1 0 1 1 0 1 1 0 0
1 0 0 0 1 1 1 0 1 0 0 1 0 0 0 1 0 1 0 0 1 0 0 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 1 0 0 1 1 0 1 1 1 0 0 0 1 0 1 0 0 0 1 1 1 0 1 0 1 0 0 1 0 1 1 0
0 0 0 0 1 1 1 0 0 0 1 1 1 0 1 0 0 1 0 0 0 1 0 1 0 0 1 0 1 1 0 1 0 1 0 0 1 0 1 0 1 0 0 1 1 1 1 1 0 0 0 1 0 1 0 0 0 1 1 1 0 1 0 1 0
0 0 0 0 1 1 1 0 0 0 1 1 1 0 1 0 0 1 0 0 0 1 0 1 0 0 1 0 1 1 0 1 0 1 0 0 1 0 1 0 1 0 0 1 1 1 1 1 0 0 0 1 0 1 0 0 0 1 1 1 0 1 0 1 0
0 0 1 1 1 0 1 0 0 1 0 0 0 1 0 1 0 1 1 0 0 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 1 0 0 1 1 1 1 1 0 0 0 1 0 1 0 0 0 1 1 1 0 1 0 1 0 0 1 0 1 1 0 1 1
0 0 1 0 0 0 1 1 1 0 1 0 0 0 1 0 0 1 0 1 0 0 1 0 0 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 1 0 0 1 1 1 1 1 0 0 0 1 0 1 0 0 0 1 1 1 0 1 0 1 0 0 1 0 0 1 0 1
1 0 0 0 1 1 1 0 1 0 0 1 0 0 0 1 0 1 0 0 1 0 0 0 0 1 0 1 0 1 0 0 1 0 1 0 1 0 1 0 0 1 1 1 1 1 0 0 0 1 0 1 0 0 0 1 1 1 0 1 0 1 0 0 1 0 1 1 0

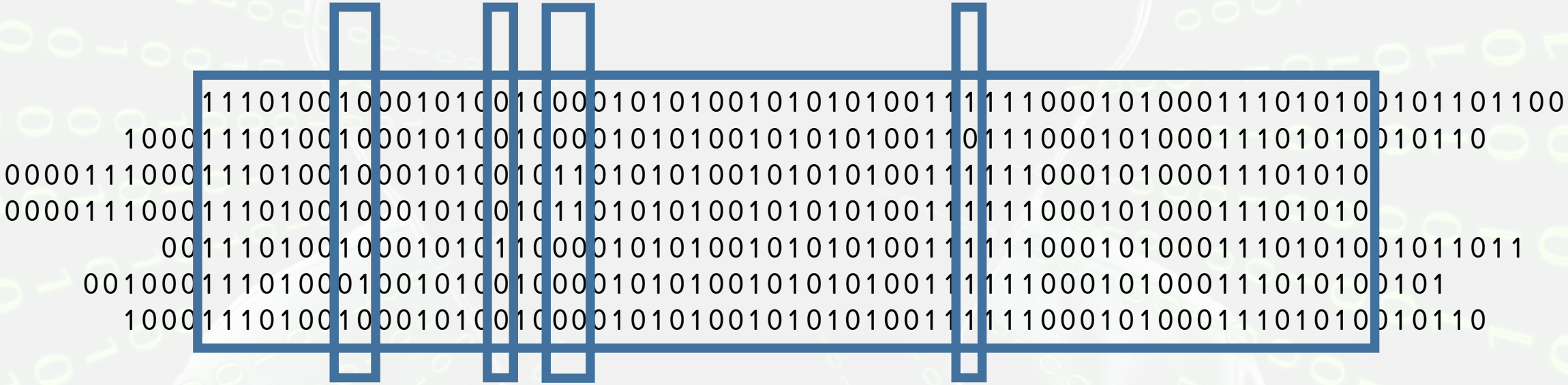
Attack: From Bits to Complete Keys

Align

```
        11101001000101001000010101001010101001111110001010001110101010101101100
    1000111010010001010010000101010010101010011011100010100011101010010110
00001110001110100100010100101101010100101010100111111000101000111010 10
00001110001110100100010100101101010100101010100111111000101000111010 10
      0011101001000101011000010101001010101010011111100010100011101010101011011
  0010001110100010010100100001010100101010100111111000101000111010101 0101
    1000111010010001010010000101010010101010011111100010100011101010010110
```

Attack: From Bits to Complete Keys

Majority



Attack: From Bits to Complete Keys

Candidate key extracted

Validate?

- encrypt-decrypt…

```
     1110100100010100100001010100101010100111111000101000111010101001011011 00
   1000111010010001010010000101010010101010011111100010100011101010010110
0000111000111010010001010010000101010010101010011111100010100011101010
0000111000111010010001010010000101010010101010011111100010100011101010
       0011101001000101001000010101001010101001111110001010001110101010101011011
  0010001110100100010100100001010100101010100111111000101000111010101 0101
    1000111010010001010010000101010010101010011111100010100011101010010110
```

Attack: From Bits to Complete Keys

After a few iterations…

"Correct" set is tested

Target Key is:

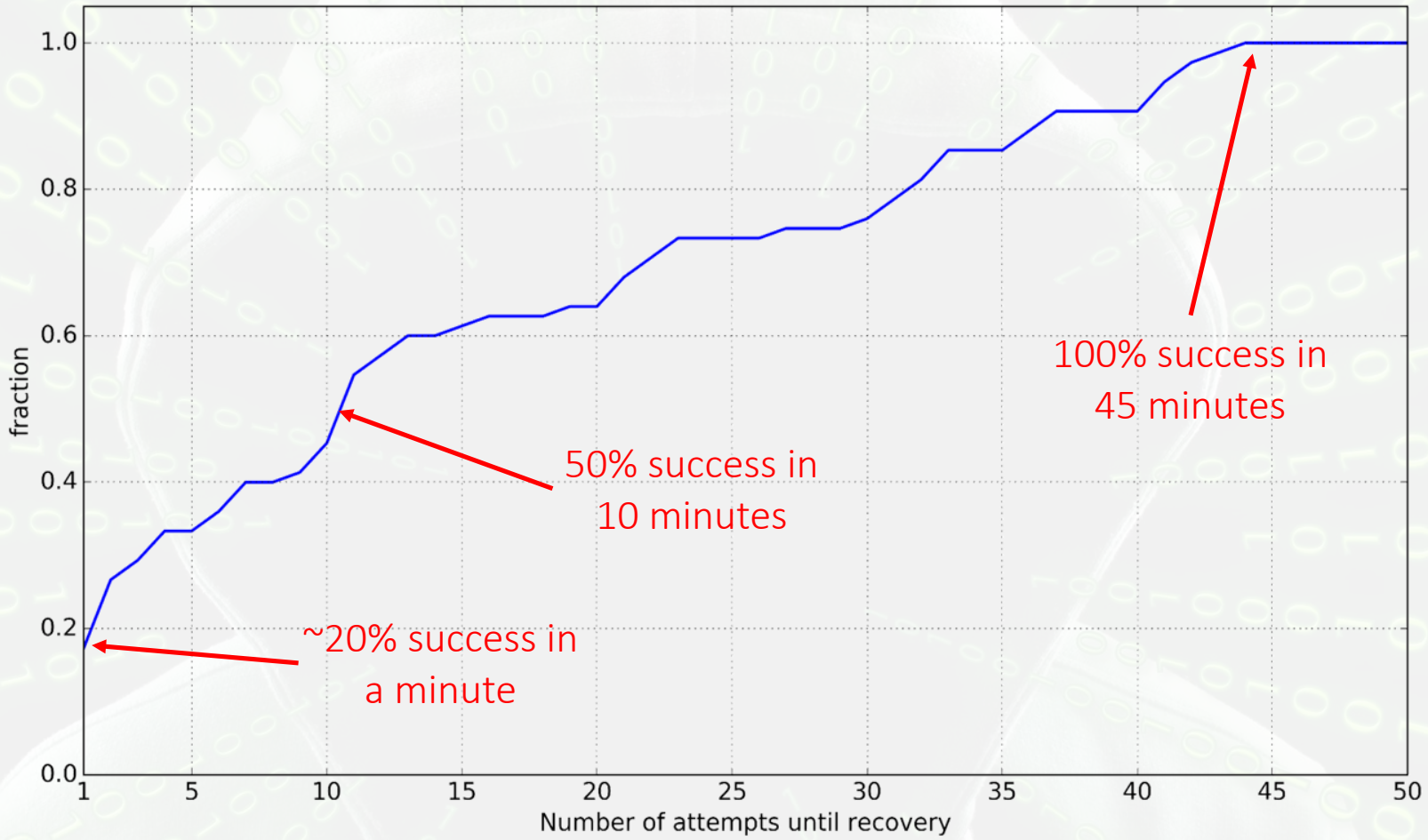
```
0xdb3e5914a0b076f8288edf676
115b5c881dbebc87d6d5ce79a55

. . . . . .

6f1bb6acd3b4d4454bbbbd9752c
9f6ccf209054b3f803d7c63df11
```

True private key obtained!!

Attack: Success Rate



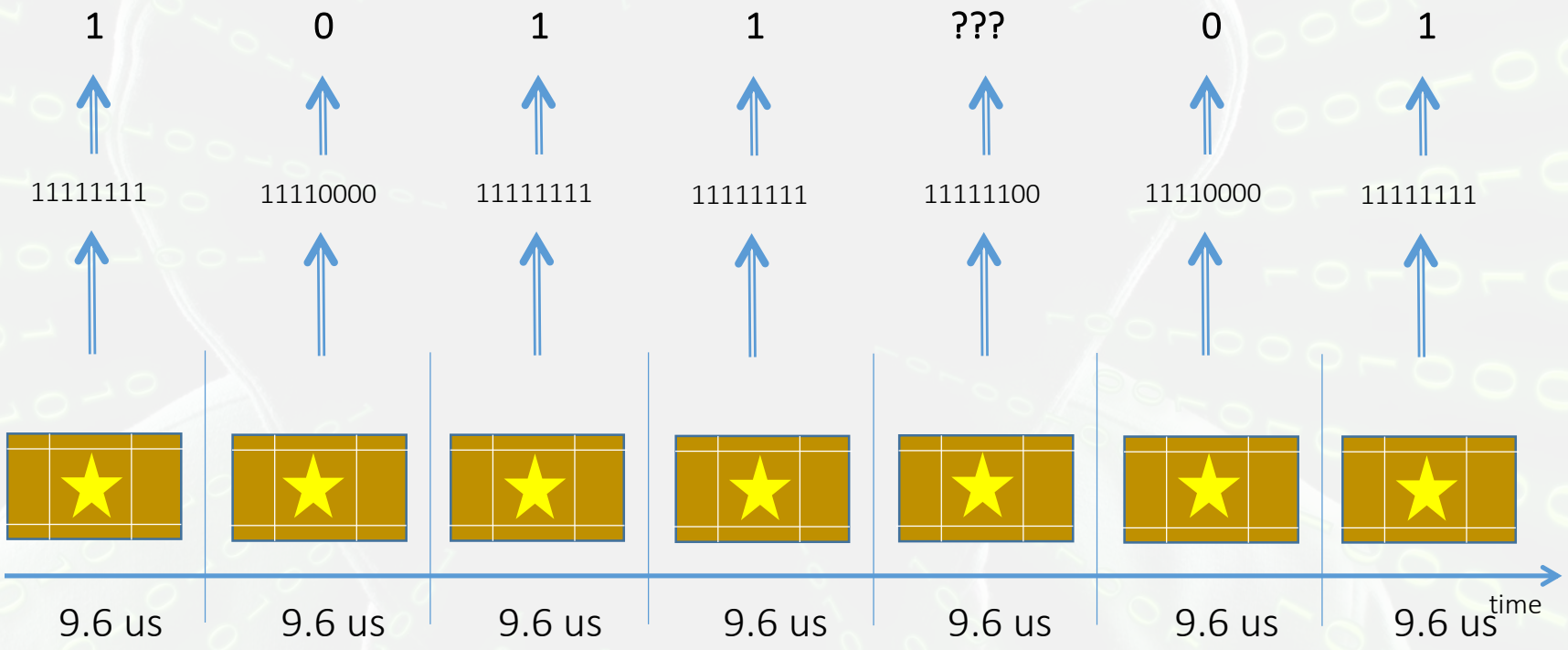
75 tests

50 attempts per test,
200 samples per attempt

~1 minute per attempt

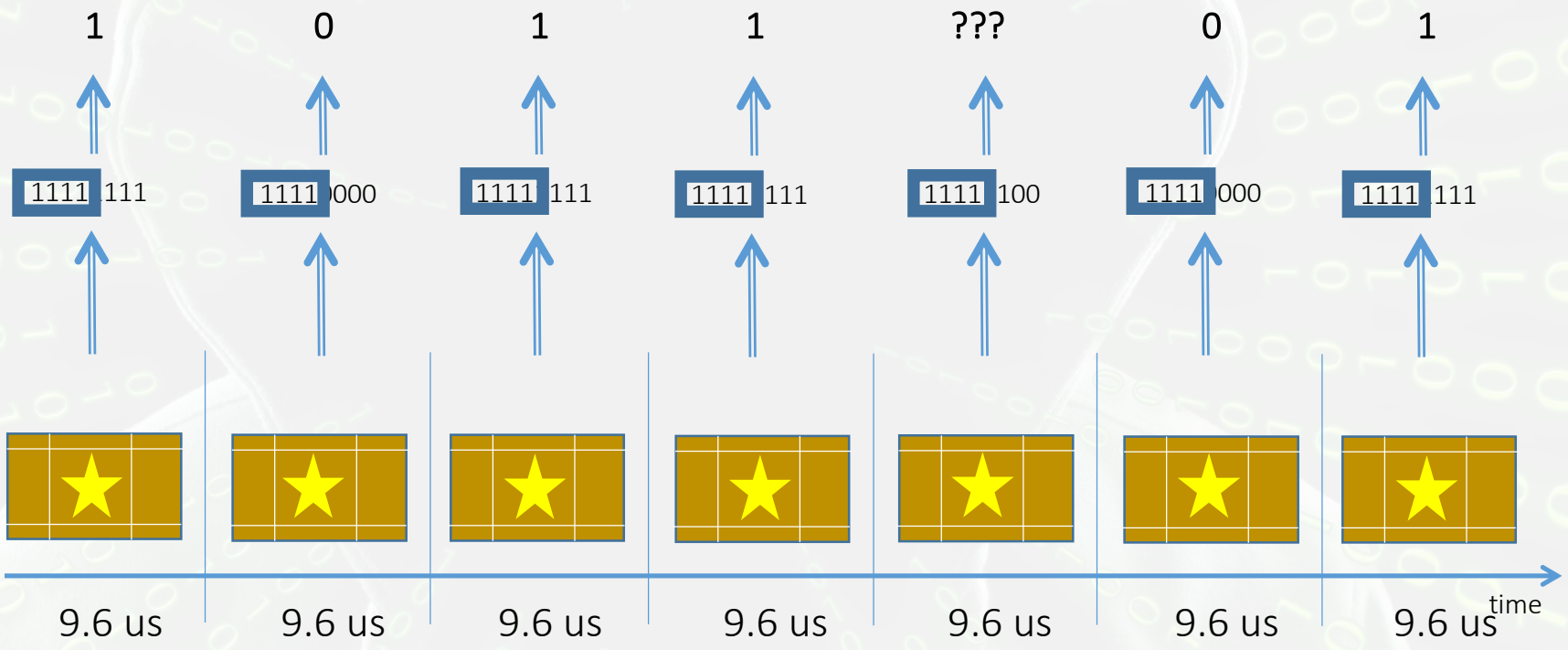
Monitoring: How/What?

Recall samples gathered during the Prime&Probe attack



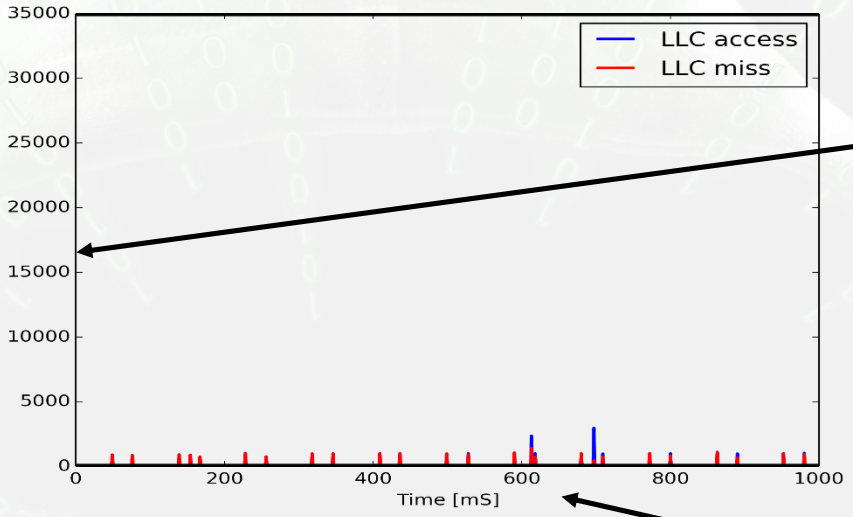
Monitoring: How/What?

Target activity in correct set causes many misses



Monitoring: Idle Attacker

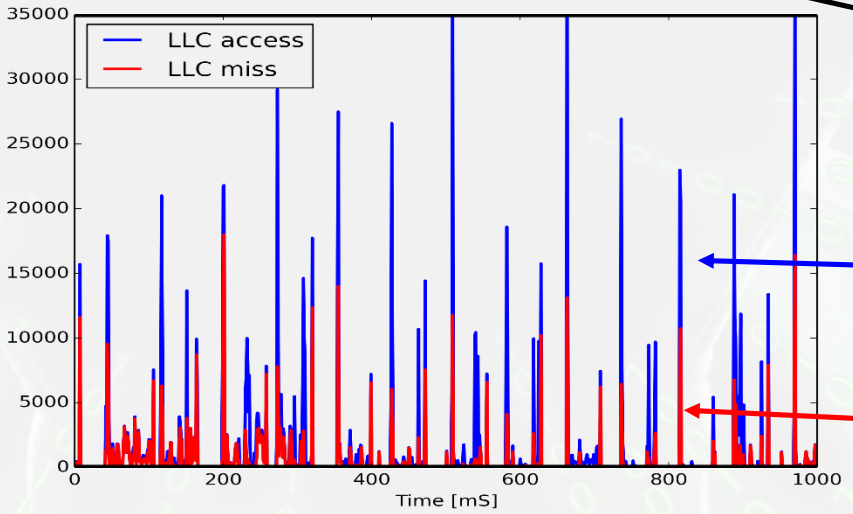
Attacker Process



Y-axis:
LLC access/miss in time slot

X-axis:
time in ms

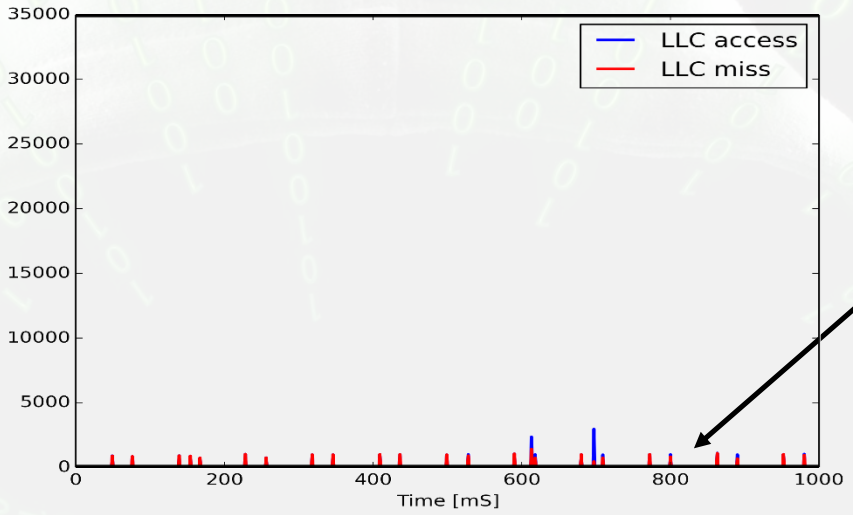
Server Process



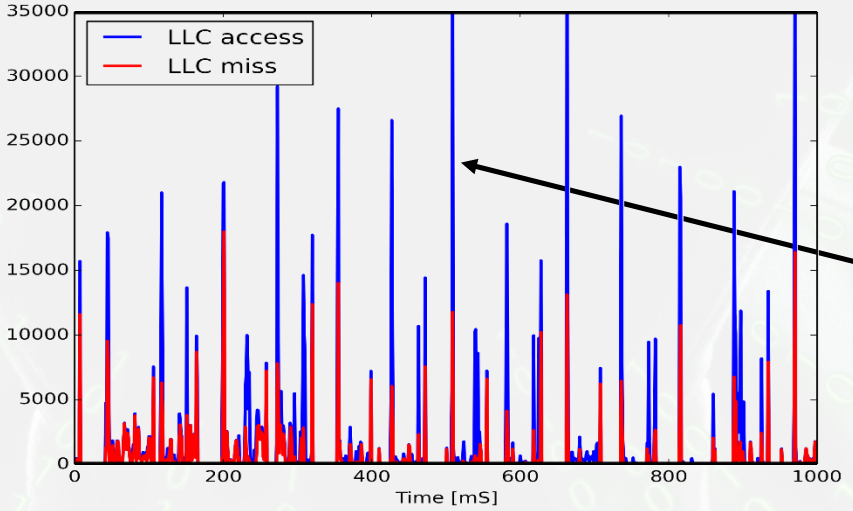
Blue line:
cache access

Red line:
cache miss

Monitoring: Idle Attacker

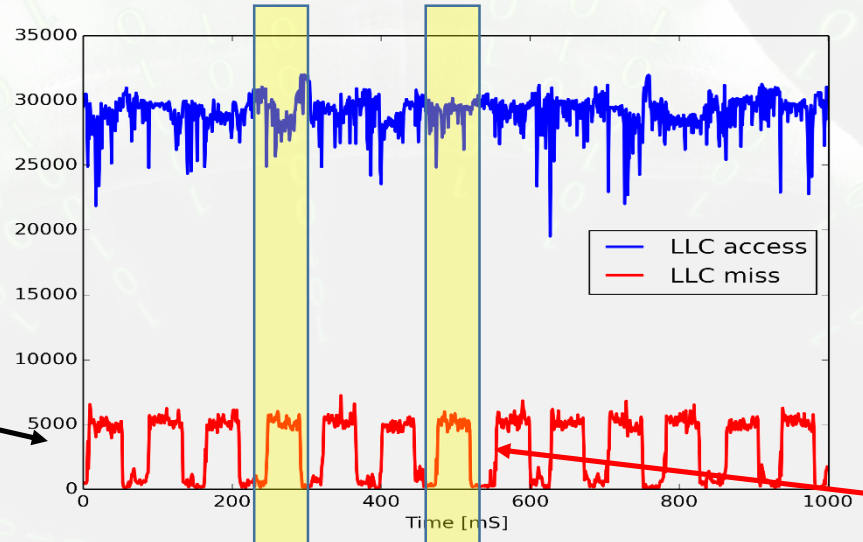


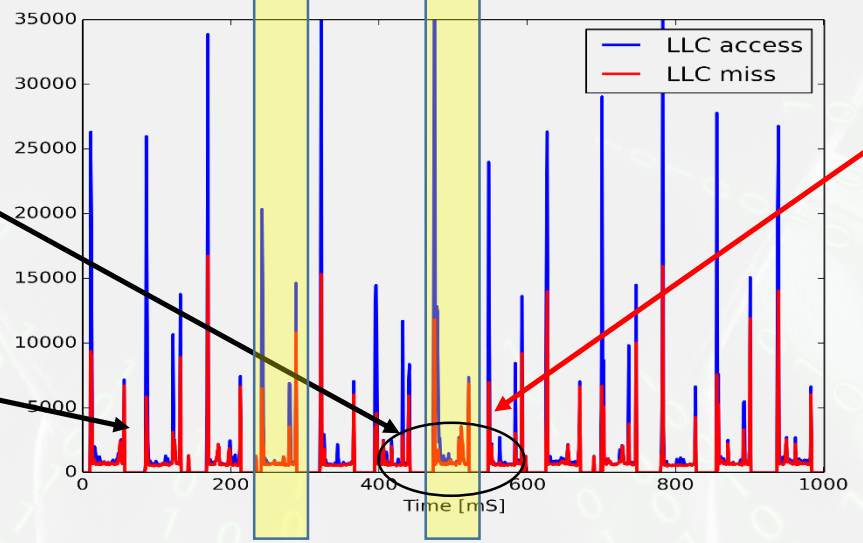
Attacker inactive – almost no cache activity



Server activity – bursts of LLC accesses & misses

Monitoring: Active Attacker





Nice…

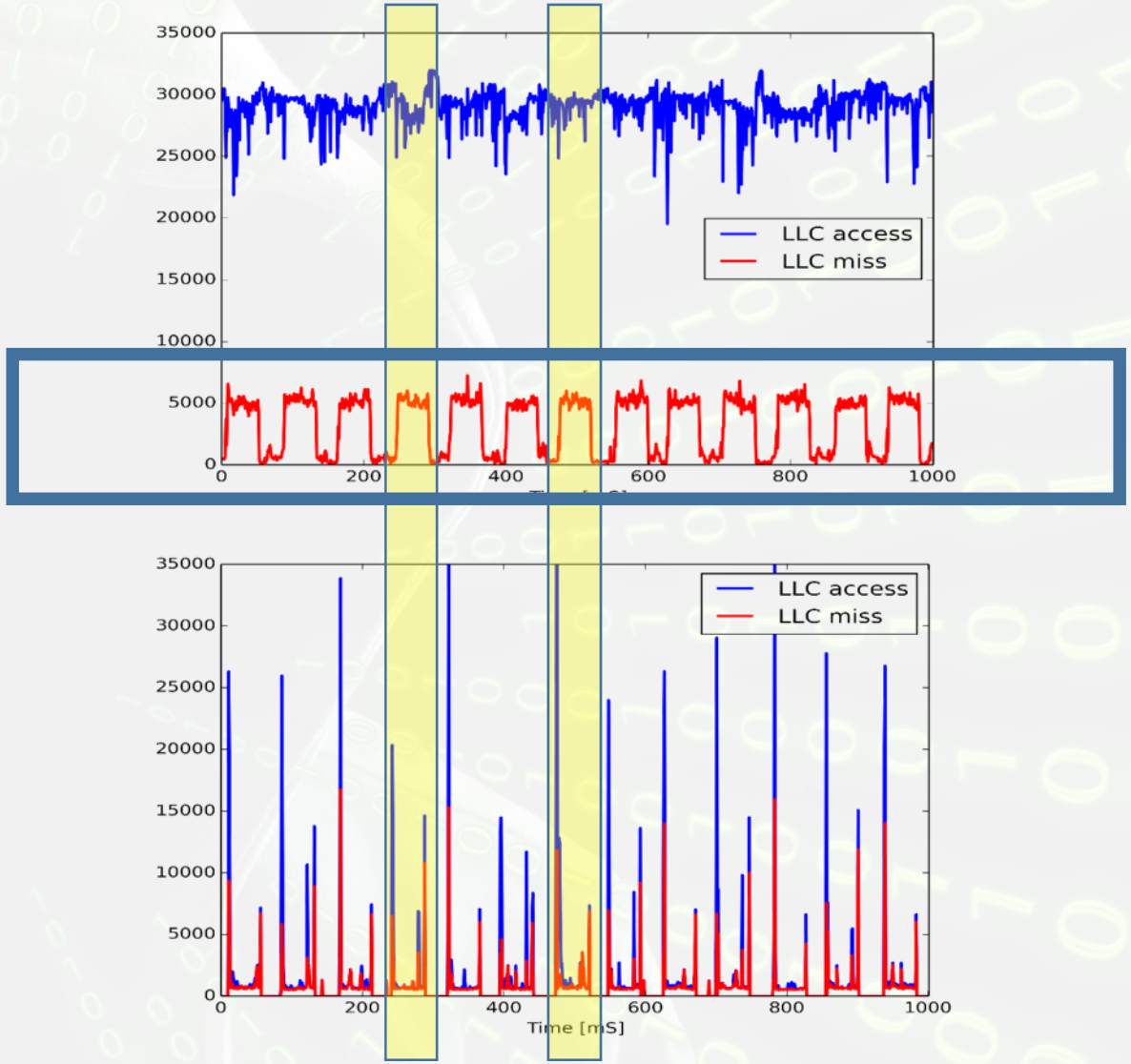
RSA decryption

Not so nice…

server and attacker activity correlated!

Monitoring: Active Attacker

- Can this be identified by focusing on target alone?
  - If so, monitoring could be provided as-a-service…

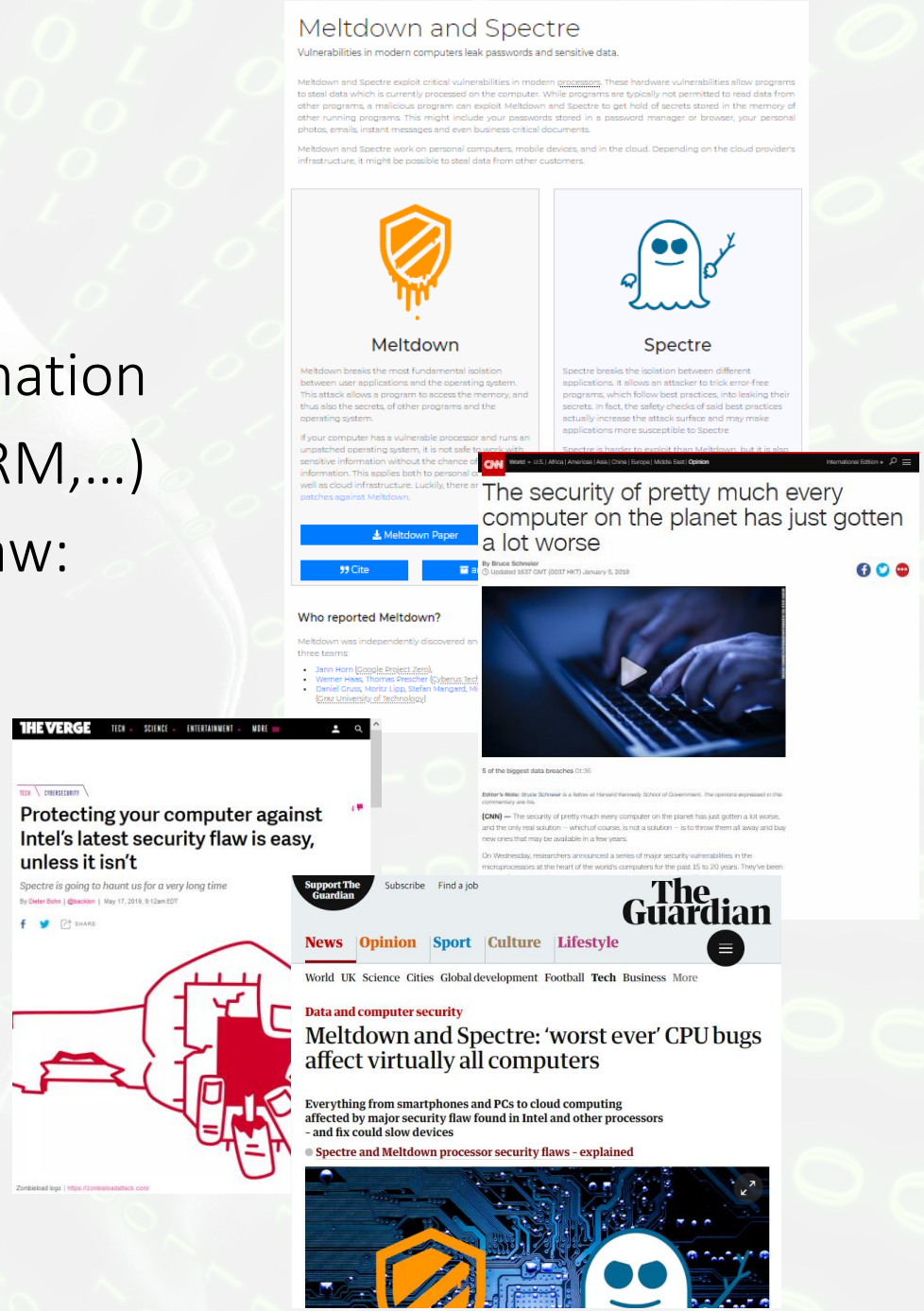


Monitoring: Target-side Only (During Attack)

Sample taken every 100µs, MA window of 50 samples. Detection in 30ms < single RSA decryption ~70ms.

And Now for Real…

- Cache side-channel attack used to extract information
- New attacks on many chip architectures (x86, ARM,…)
- Heavily based on mechanisms like the one we saw:
  - Prime&Probe
    - Separate cores, only LLC, very noisy
  - Flush&Reload
    - Attacker/target share memory, e.g., run on same core
    - Significantly easier than Prime&Probe
- Exploit various mechanisms
  - Shared memory / LLC
  - Out of order execution
    - Specifically, speculative execution
- Buzzwords: Meltdown, Spectre, …
  - And more



(Partial) Bibliography

- Ristenpart et al., “Hey you get off my cloud: Exploring information leakage in third-party compute clouds”, in CCS, pp. 199-212, 2009
- Yarom and Falkner, “FLUSH+RELOAD: A High Resolution, Low Noise, L3 Cache Side-Channel Attack”. USENIX Security, pp. 719-732, 2014
- Liu et al, “Last-Level Cache Side-Channel Attacks are Practical”, IEEE S&P, pp. 605-622, 2015
- Martins et al., “A Survey on Fully Homomorphic Encryption: An Engineering Perspective”, ACM Comp. Surv. 50(6): 83:1-83:33, 2017
- Kotcher et al., “Spectre Attacks: Exploiting Speculative Execution”, ArXiv e-prints, 2018
- Lipp et al., “Meltdown”, ArXiv e-prints, 2018
- Schwarz et al., “ZombieLoad: Cross-Privilege-Boundary Data Sampling”, ArXiv e-prints, 2019
- Joint work with Niv Gilboa, Ben Amos, Arbel Levy (and others)