Time is NOT on Your Side:

Mitigating Timing Side Channels on the Web

Sebastian Schinzel

Friedrich-Alexander Universität Erlangen-Nürnberg
Lehrstuhl für Informatik 1
IT-Sicherheitsinfrastrukturen

Web: http://seecurity.org/
Twitter: @seecurity

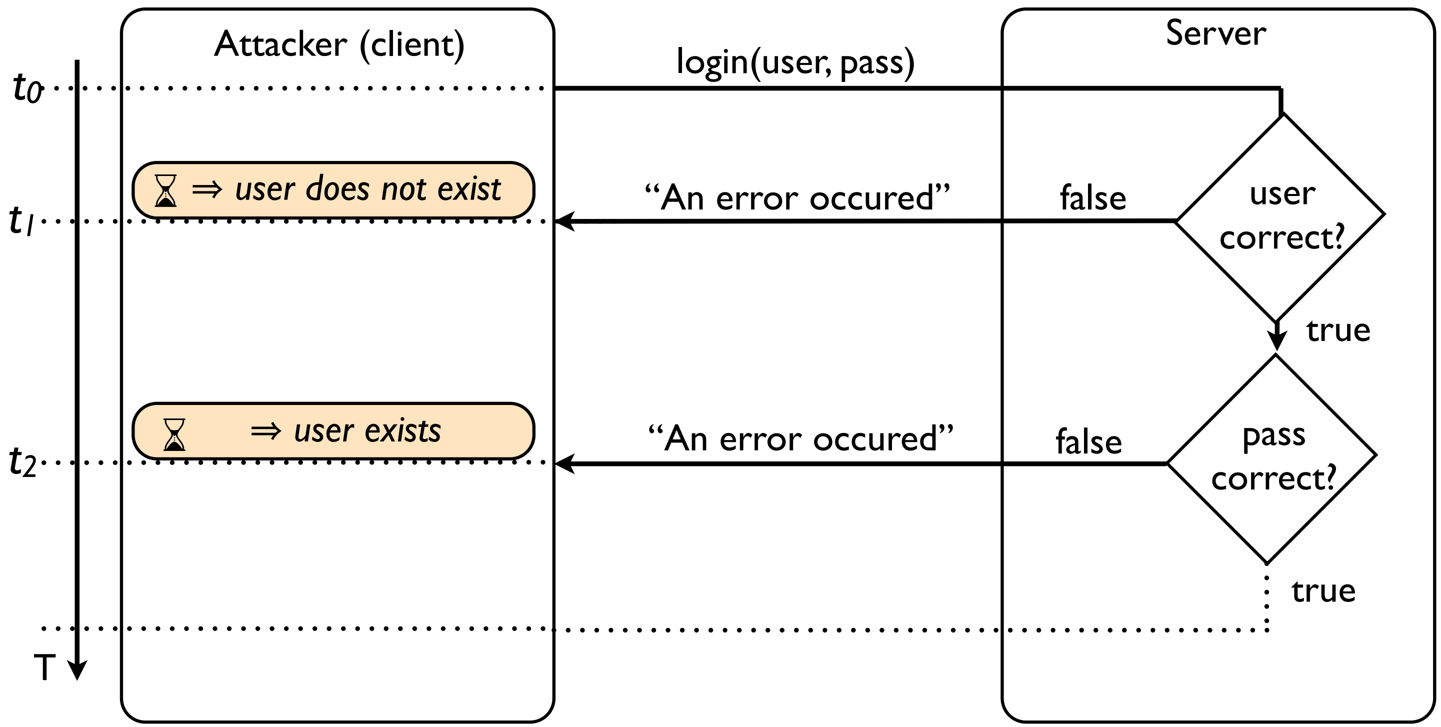
FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG
TECHNISCHE FAKULTÄT

About me

- Academia: Postdoc researcher at University of Erlangen
  - Offensive software security
  - Side channel attacks

- Industry: Penetration tester, consultant, speaker, teacher
  - Software security topics (design, implementation, test of software)
  - Focus on SAP security (ABAP)

Brief overview on timing attacks

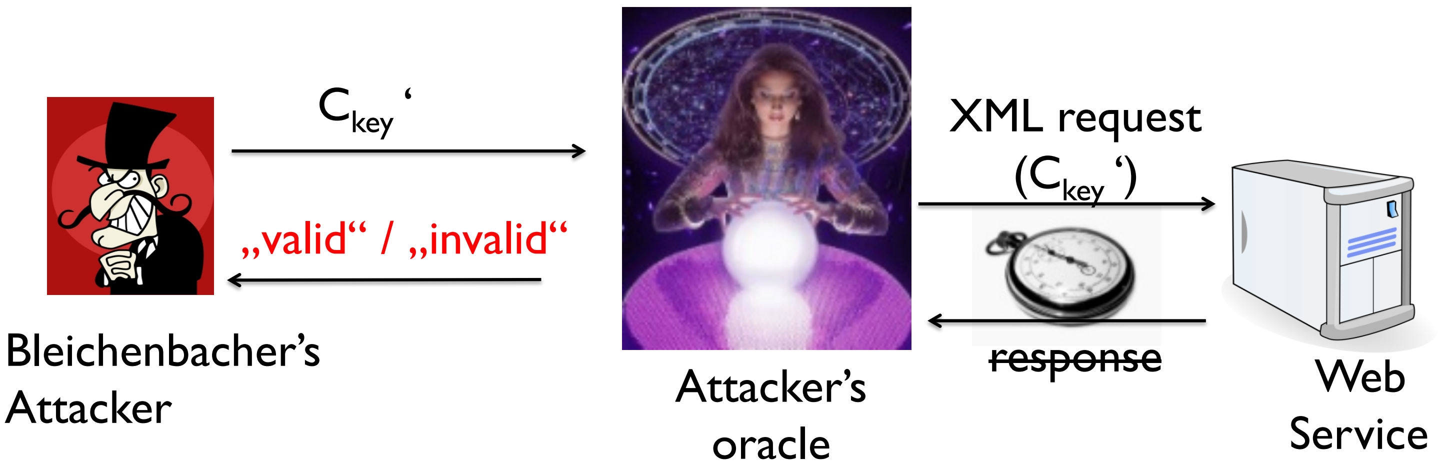


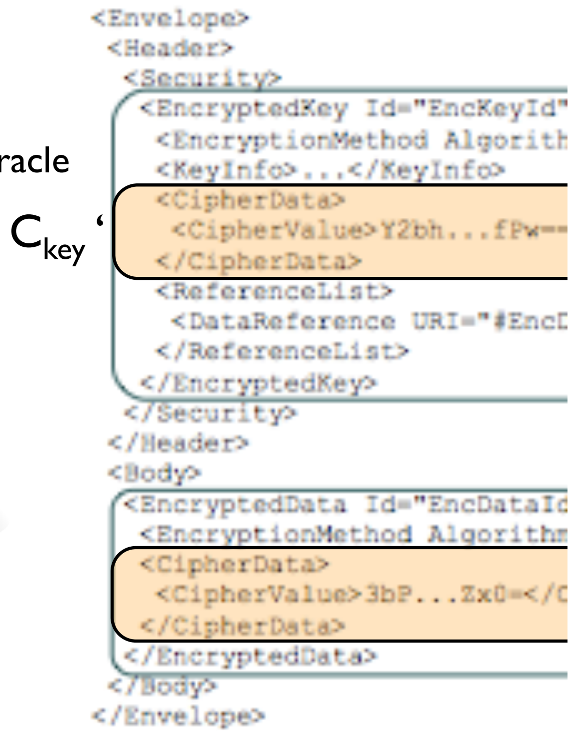
Other examples for side channels:

- sound
- visuals
- emissions
- power consumption
- motion (mobiles)
- size of encrypted network packets

Breaking XML Encryption

- Attacker eavesdrops on XML Encryption message
- Break RSA-encrypted session key with a few 100.000 requests using a Bleichbacher oracle





Tibor Jager, Sebastian Schinzel, Juraj Somorovsky
Bleichenbacher's Attack Strikes again: Breaking PKCS#1 v1.5 in XML Encryption
17th European Symposium on Research in Computer Security (ESORICS 2012)
http://www.nds.rub.de/research/publications/breaking-xml-encryption-pkcs15/

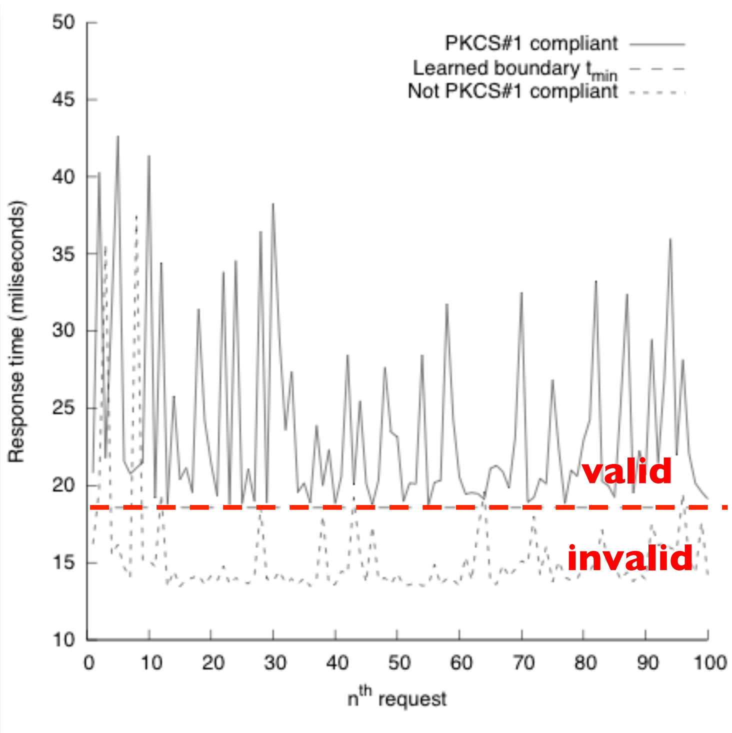
Brief overview on timing attacks

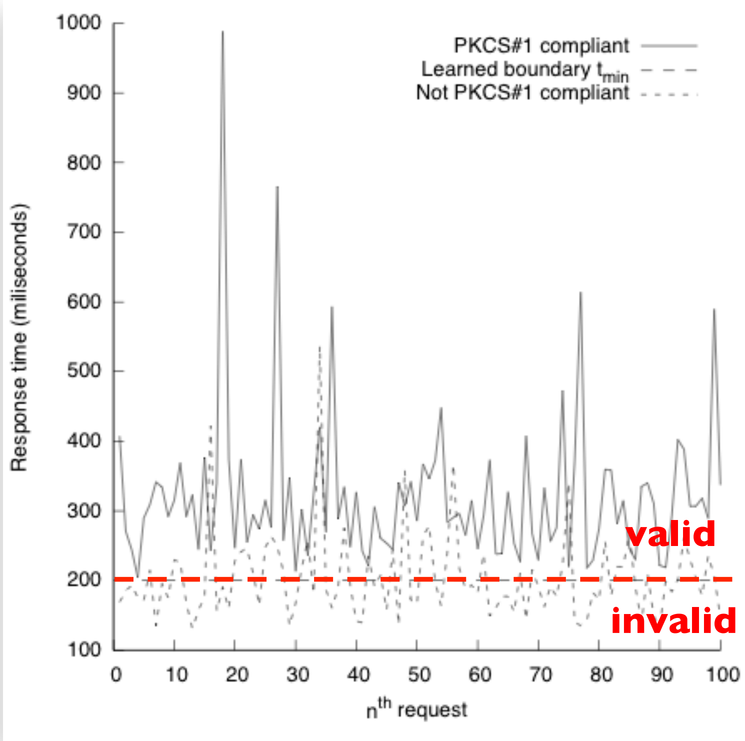
Results:
Bleichenbacher timing oracle

398,123 server requests

Localhost:
→ less than 200 minutes

Internet:
→ less than 1 week





Tibor Jager, Sebastian Schinzel, Juraj Somorovsky
Bleichenbacher's Attack Strikes again: Breaking PKCS#1 v1.5 in XML Encryption
17th European Symposium on Research in Computer Security (ESORICS 2012)
http://www.nds.rub.de/research/publications/breaking-xml-encryption-pkcs15/

Brief overview on timing attacks

A generic side channel:

```
if((secret % 2) == 0) {
    do_thing1();
} else {
    do_thing2();
}
```

t_1

t_2

?

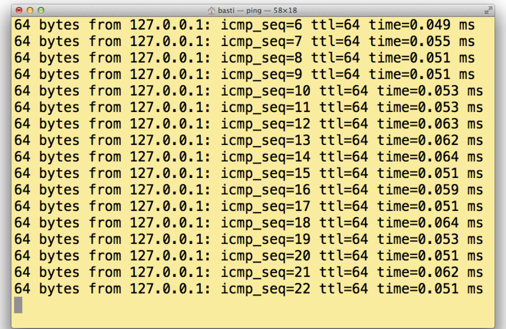
`(secret % 2) == 0`

Brief overview on timing attacks

Determinism vs. statistics:

- Buffer overflow exploit works or not → shell code is executed or not

- Statistical methods always gives ***some*** result → result is 23.42
  - "detect silent voices in a very noisy environment"
  - but what means 23.42?
  - coincidental or *statistically significant*?

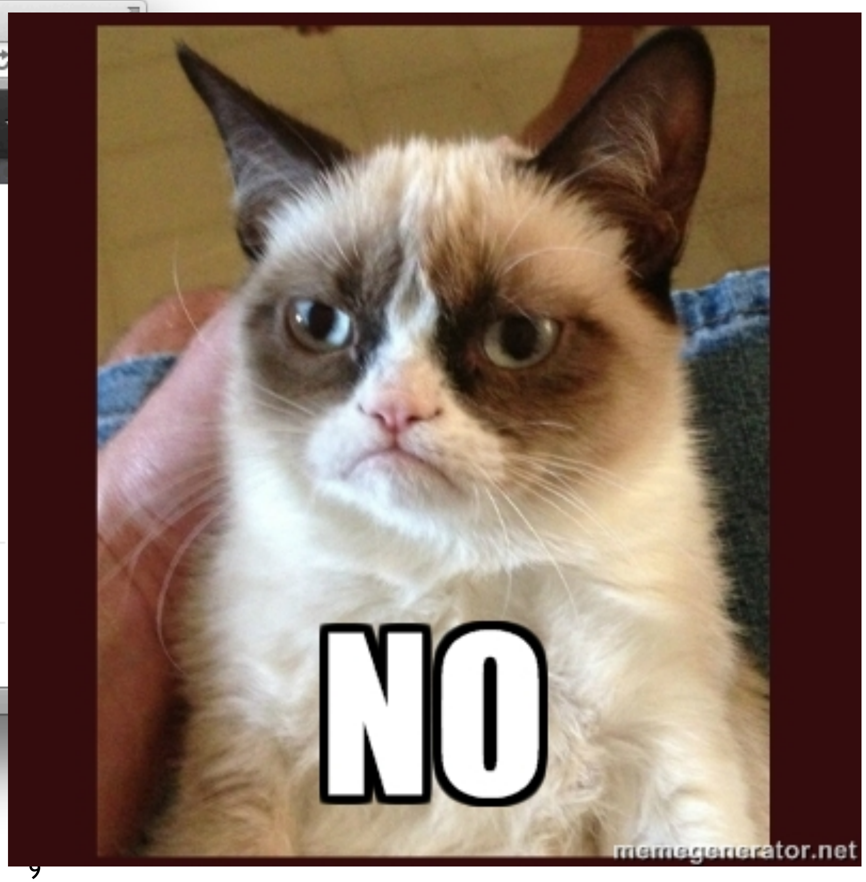


Attacker has only limited control over noise

- Choose high quality network entry point during idle times
- Crosby* results
  - successfully measured 200 nanoseconds processing time difference over a local LAN with 1000 measurements
  - successfully measured 30 microseconds processing time difference over the Internet with 1000 measurements
  - measurement hardware matters!

* Crosby, Riedi, Wallach,
Opportunities and Limits of Remote Timing Attacks
ACM Trans. Inf. Syst. Secur, 12(3), 2009
http://www.cs.rice.edu/~dwallach/pub/crosby-timing2009.pdf

Brief overview on timing attacks

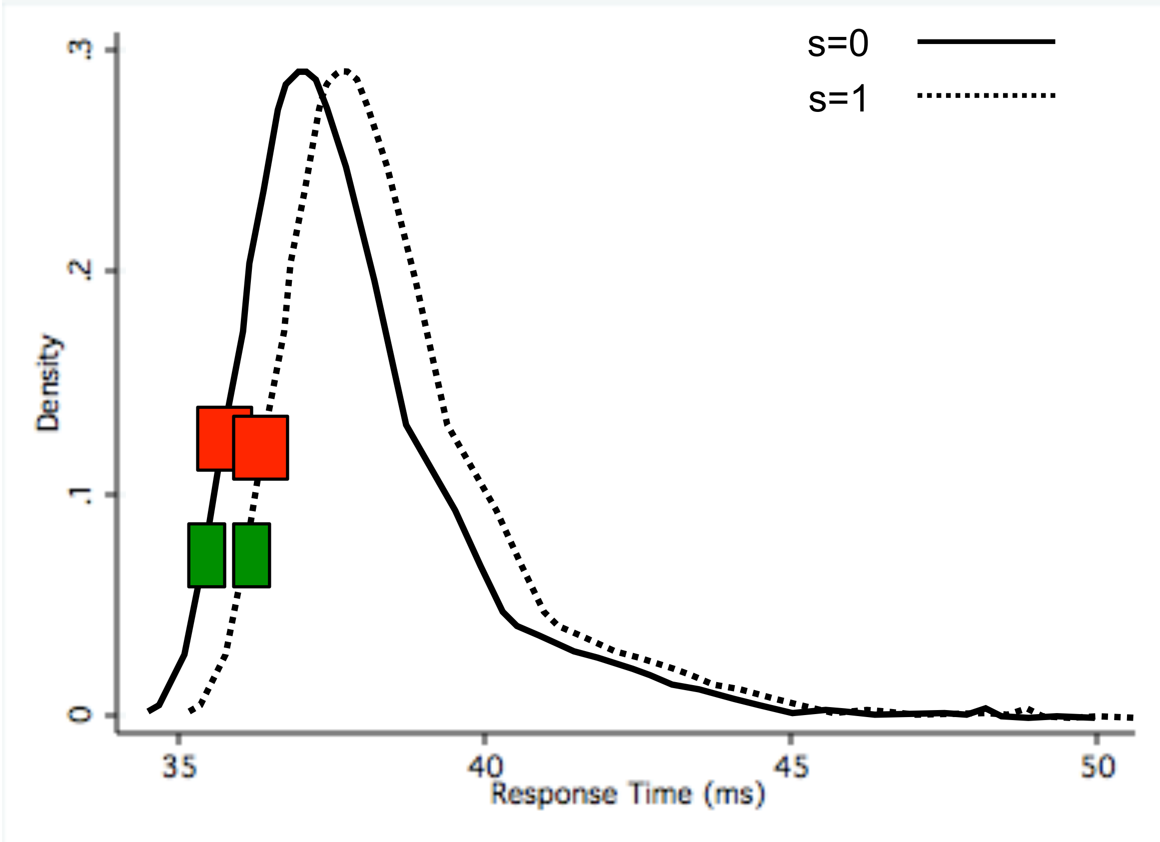


Analyzing timing data sets

- Central limit theorem says that if you measure enough times a number of independent random values, then the resulting dataset will be normally distributed
- Often true for local measurements and hardware-near measurements
- Timing measurements over networks are usually not normally distributed (see Crosby 2009)
- Standard hypothesis tests don’t work well

Analyzing timing data sets

- Crosby proposed the "Box test" as an alternative hypothesis test
- We implemented the Box test and published it at http://www1.cs.fau.de/side-channels/
- Also used for this talk



What about the accuracy of a timing attack?

- one-shot attack
  - List of user names to try out
  - 50% accuracy means that 50% of the detected user names are actually correct
  - … better than nothing

Adaptive attack

- Current query depends on result of previous query
  - A single wrong conclusion during the measurements might mess up measurement efforts of days or weeks
  - See timing attack on XML Encryption (ESORICS '12)
  - 0.1% error rate might still not be sufficient

Preventing Timing Side Channels

28c3: "*Time is on my side: Exploiting Timing Side Channel Vulnerabilities on the Web*"

- explained how to do timing attacks
- presented tools to (http://www1.cs.fau.de/side-channels/)
  - perform timing measurements
  - evaluate timing data sets for significant differences
- 0day: practical timing attack to break *XML Encryption* ciphertexts
- → for details on timing attacks, watch the 28c3 talk
- In Q&A session people asked how to prevent timing attacks

http://events.ccc.de/congress/2011/Fahrplan/events/4640.en.html
http://www.youtube.com/watch?v=ykNt8pSQFZQ

A side channel:

```
if((secret % 2) == 0) {
    do_thing1();
} else {
    do_thing2();
}
```

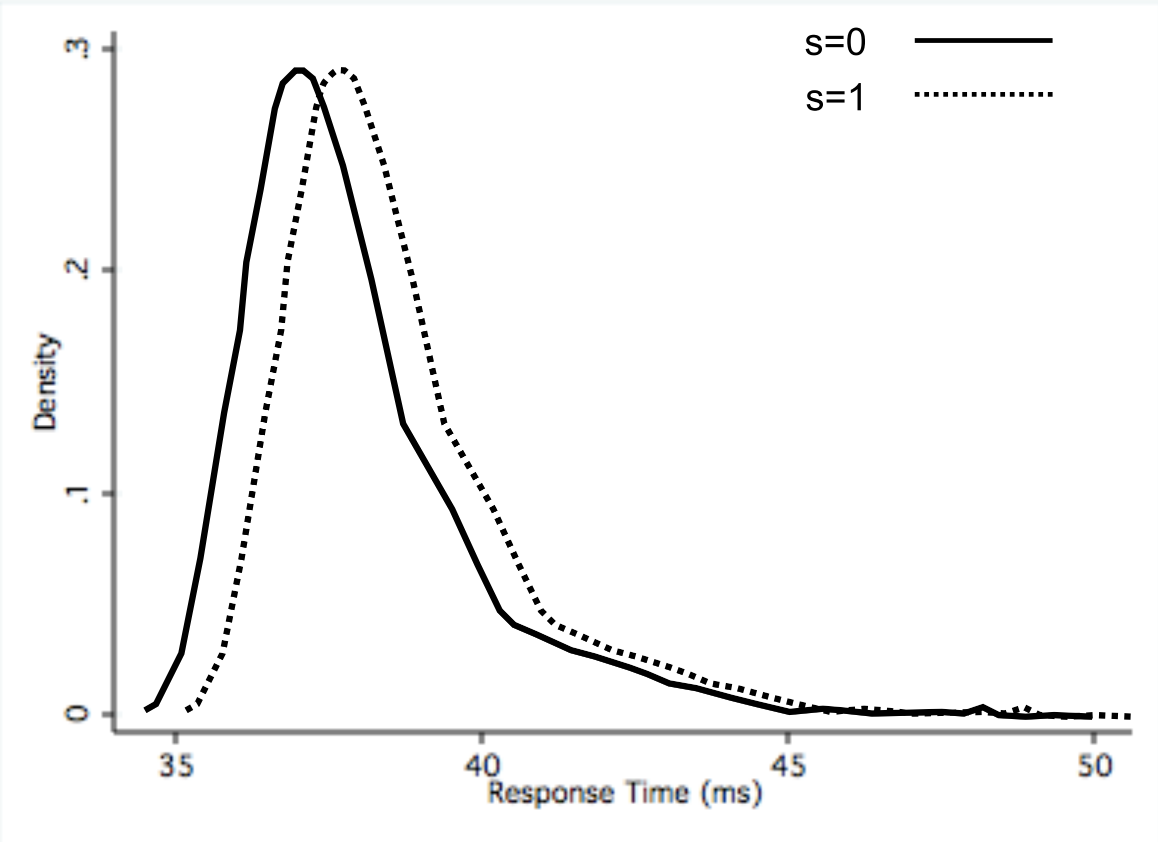
Effective prevention of the side channel:

```
do_thing1();
do_thing2();
if((secret % 2) == 0) {
    use_result1();
} else {
    use_result2();
}
```

→ Drawback: slower

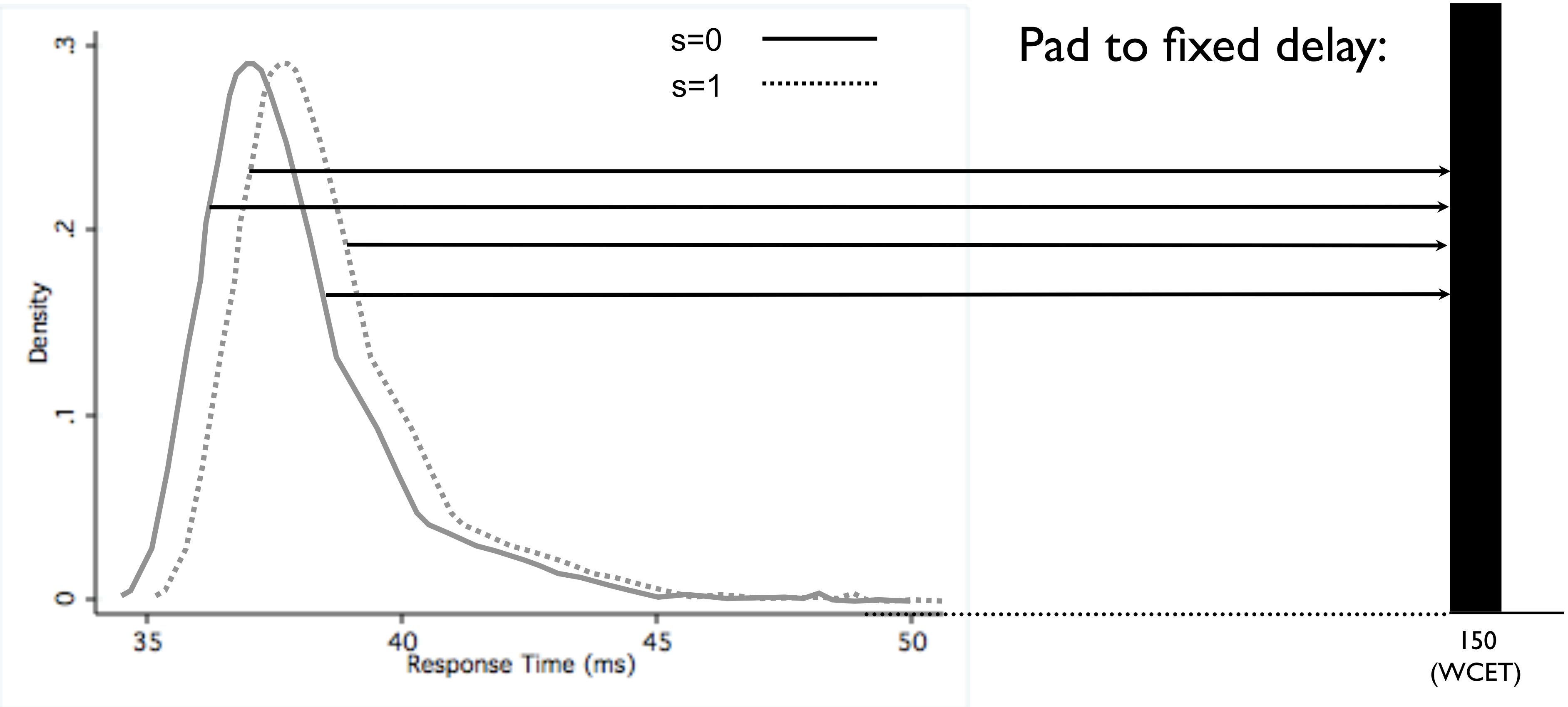
Preventing timing side channels

- Easiest is really to remove the timing side channel from the code
- But what if
  - you don’t have the code (closed-source, *“Eeeeeeeeeew!!”*)
  - you don’t have the know-how for fixing it
  - you don’t know about the vulnerability in the first place



Pad to fixed delay:

Preventing Timing Side Channels

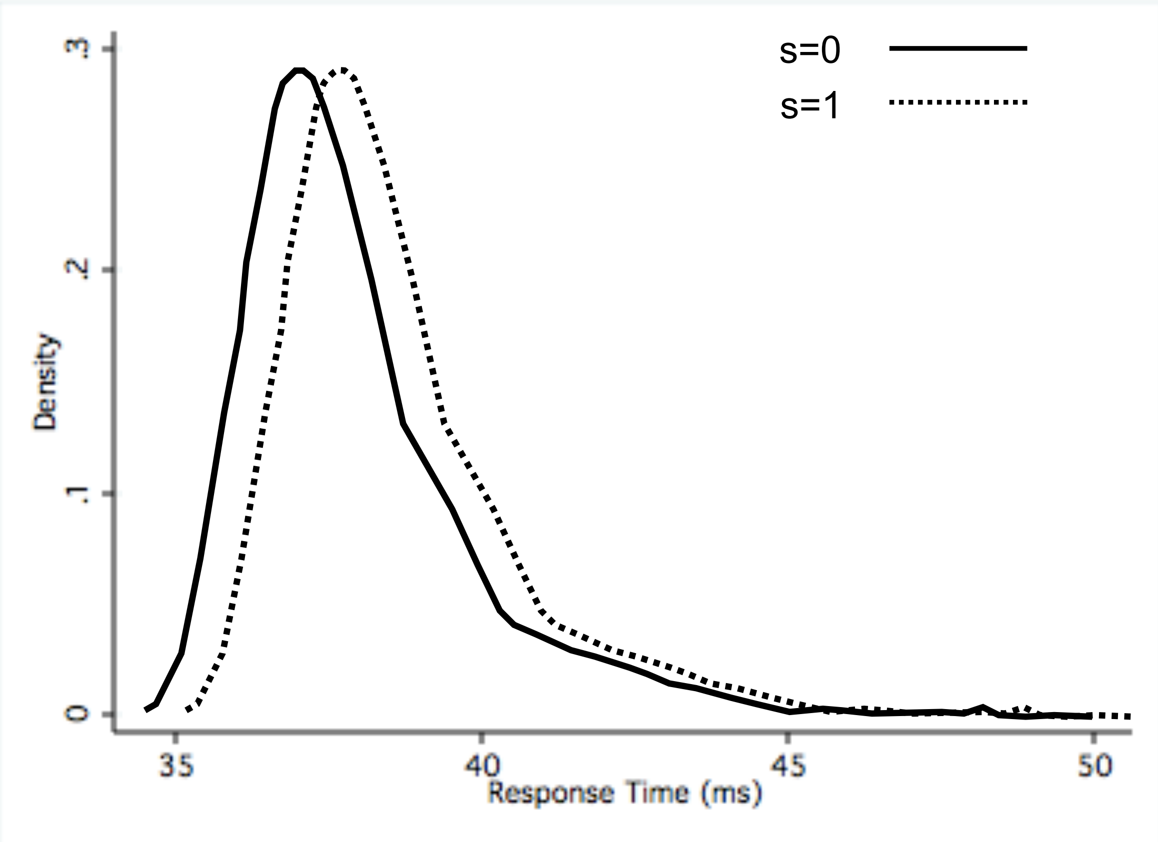


Random delay padding

- That's what everybody is asking when I'm talking about side channels
- (not only timing but also storage side channels)

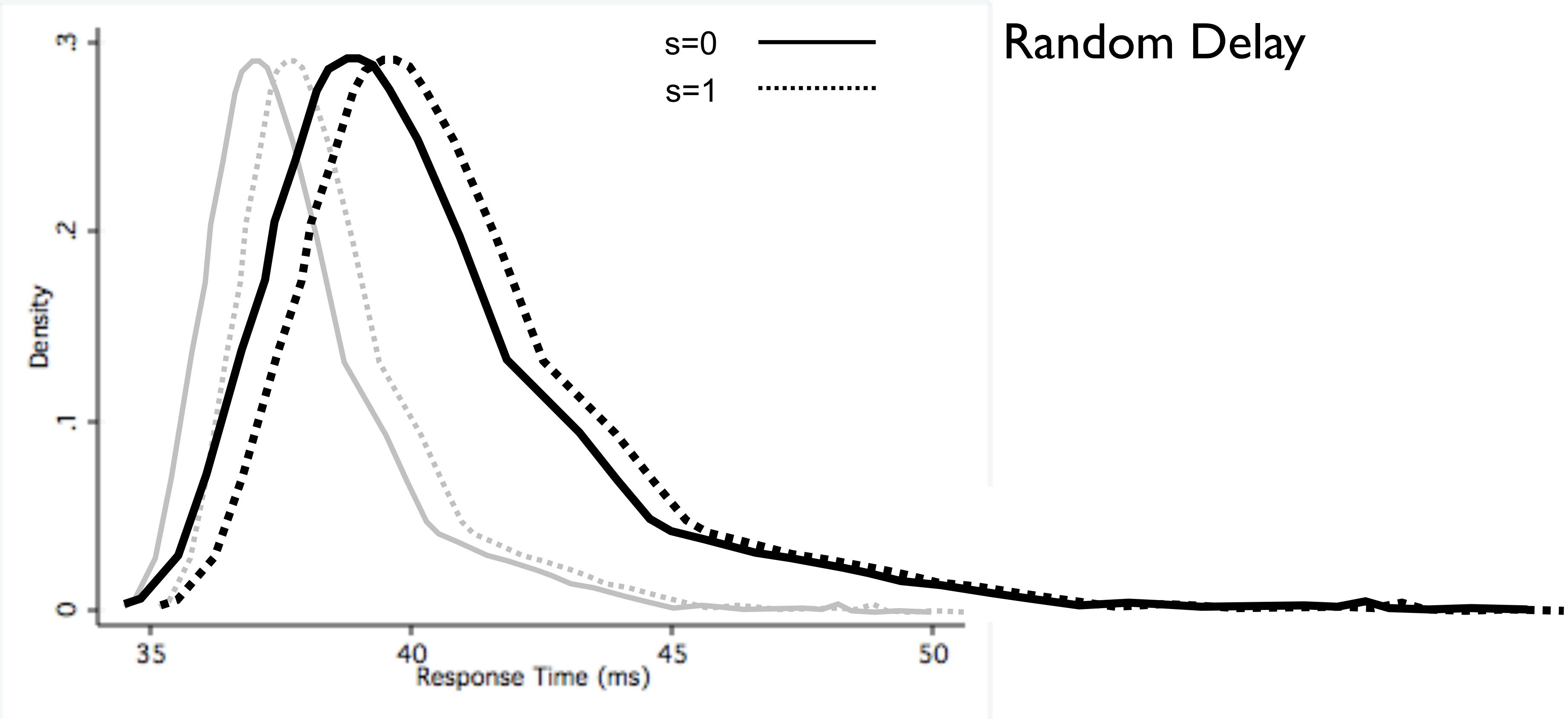
Obfuscating the timing difference using random delays

```
if((secret % 2) == 0) {
    do_thing1();
} else {
    do_thing2();
}

int r = random() % t_max;

nanosleep(r);
```



Random Delay

Random Delay

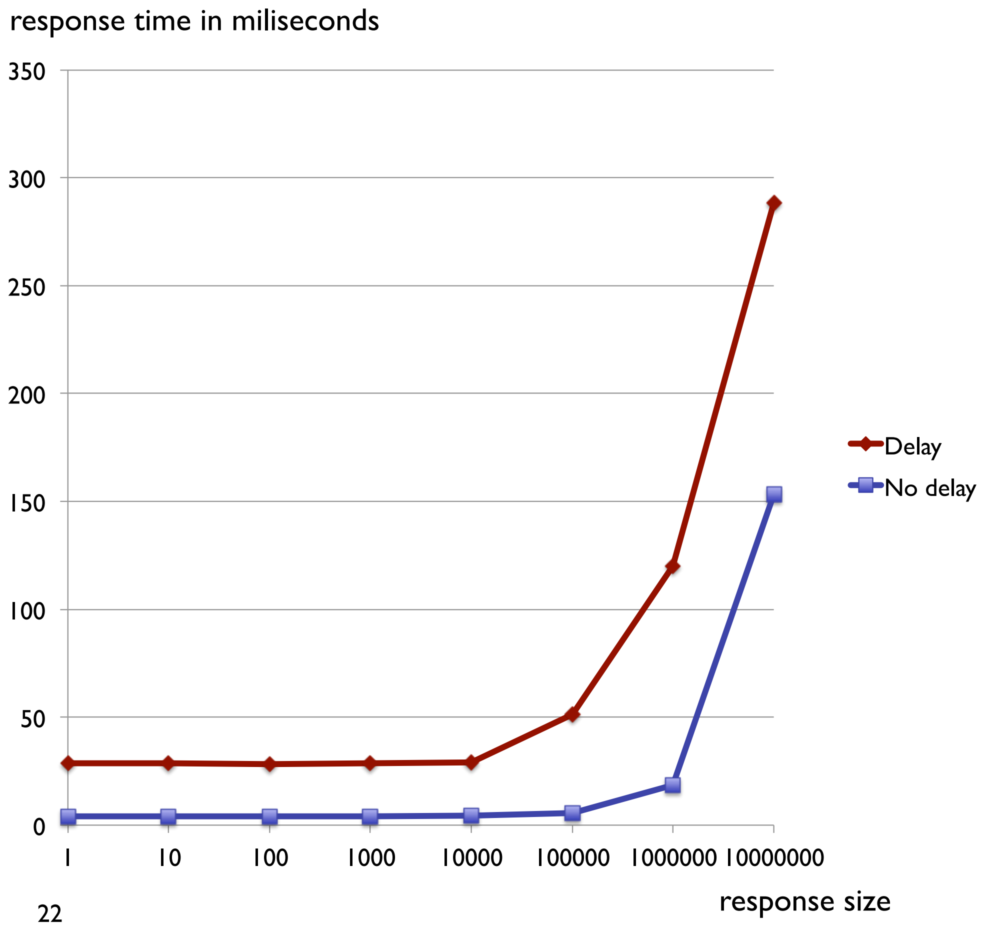


Dan Kaminsky, Black Hat 2012:

```
tc qdisc add dev eth0 root netem delay
3ms 1ms
```

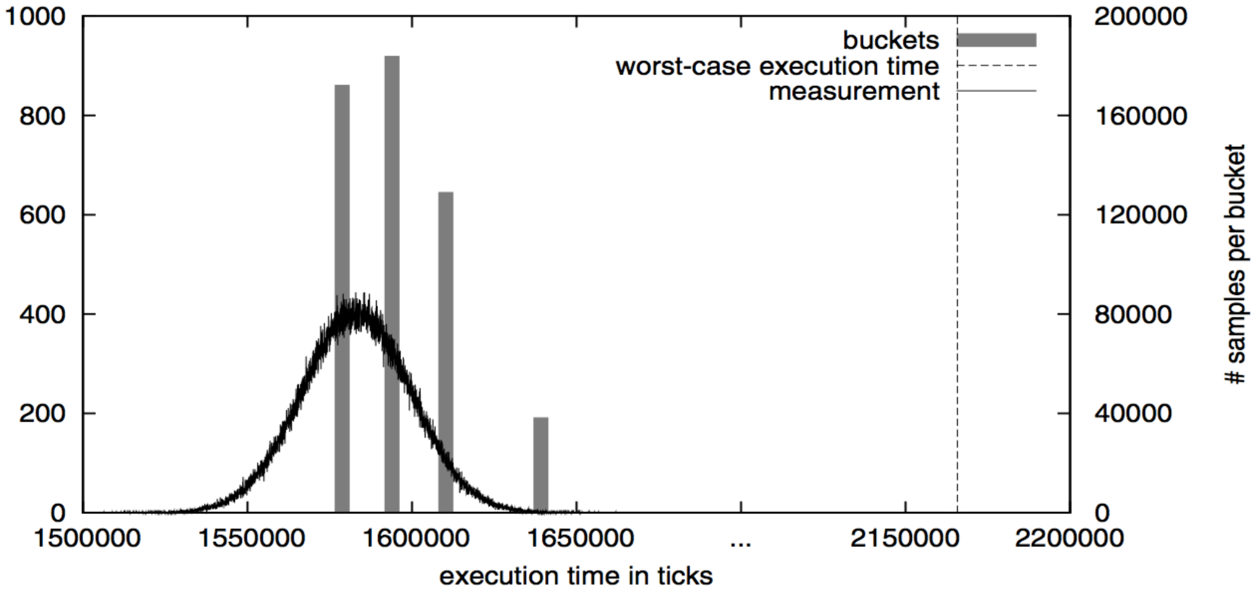
Performance reduction: factor ~7

File size	No delay	Delay
1 B	4.09 ms	
10 B	4.09 ms	
100 B	4.08 ms	
1 KB	4.09 ms	
10 KB	4.27 ms	
100 KB	5.63 ms	
1 MB	18.66 ms	
10 MB	153.43 ms	



Other timing delay padding strategies:

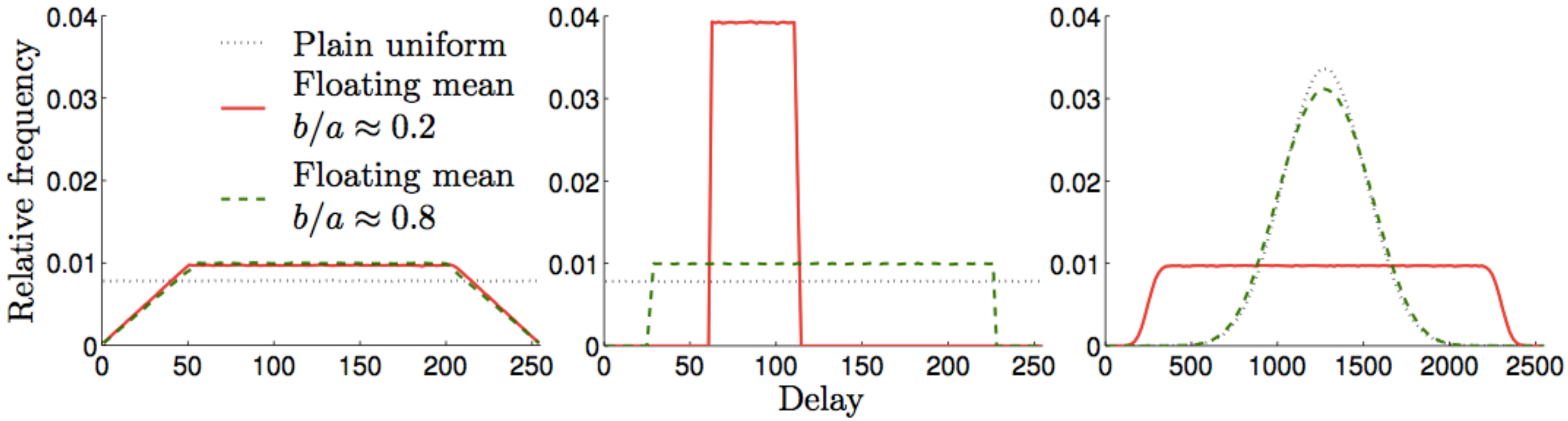
- reducing the precision of timing delays using "bucketing"



Boris Köpf and Markus Dürmuth.
A Provably Secure and Efficient Countermeasure against Timing Attacks.
CSF, pp. 324-335, IEEE Computer Society, 2009.

Other timing delay padding strategies:

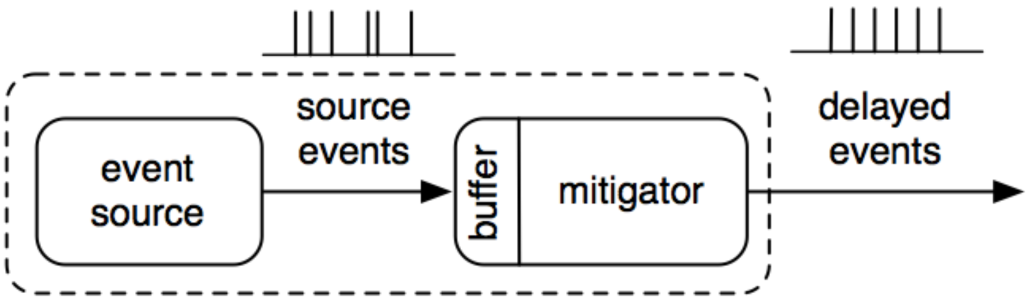
- use non-uniform random distributions



Jean-Sébastien Coron and Ilya Kizhvatov
An Efficient Method for Random Delay Generation in Embedded Software
CHES 2009

Other timing delay padding strategies:

- create a stream of "events" with constant timings



Aslan Askarov and Danfeng Zhang and Andrew C. Myers
Predictive black-box mitigation of timing channels
ACM Conference on Computer and Communications Security 2010

"Adding random padding
to hide the length of compressed/encrypted data
is like setting your [Toyota] Prius on fire
because it doesn't pollute enough." (tweet by Matthew Green)

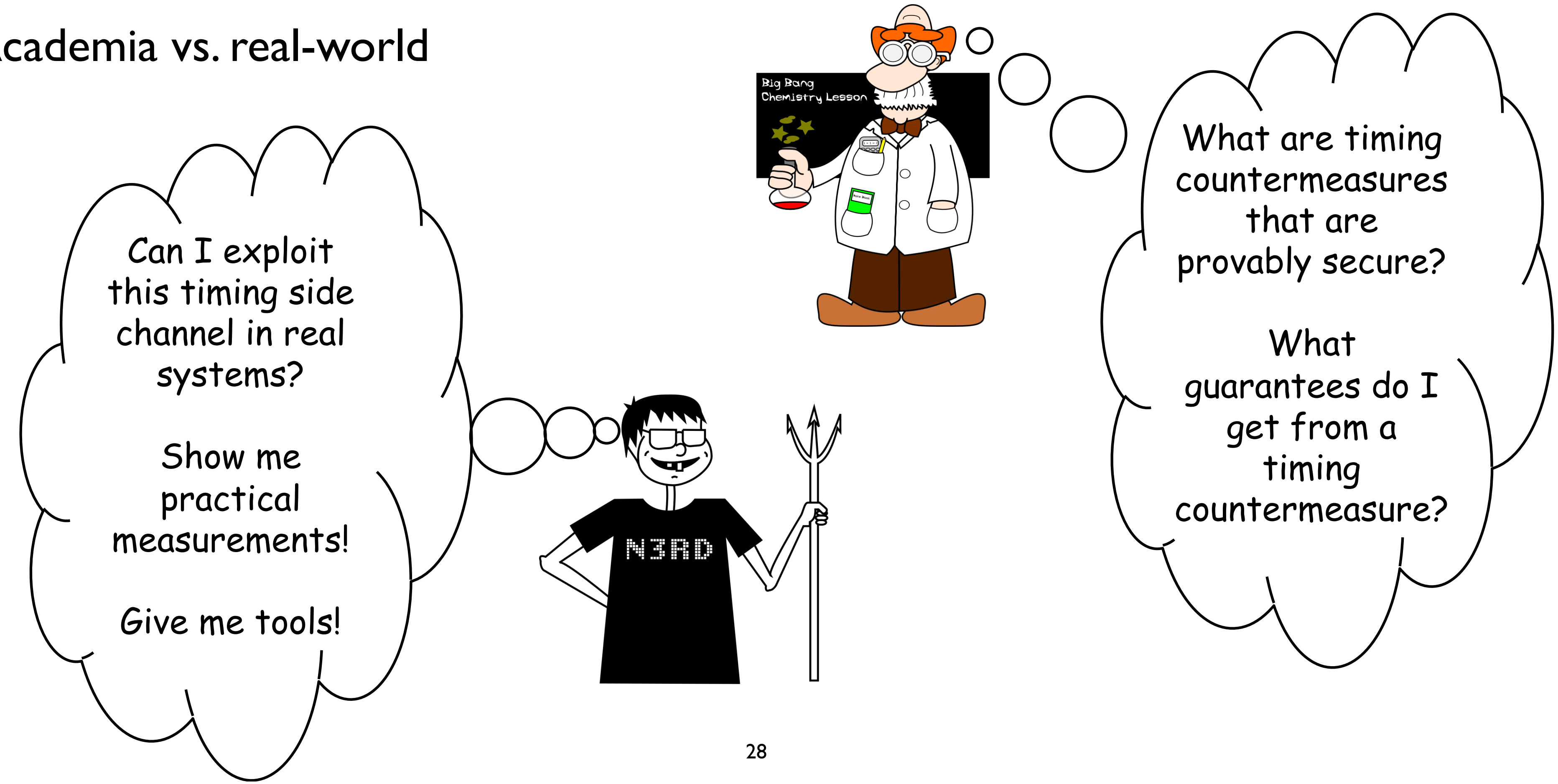
→ if possible, go fix your protocol / your code / (your hardware)

https://twitter.com/seecurity/status/259026201736253440

My research questions for this talk:

- Does random delay padding effectively prevent timing side channels?
- What maximum size of random delay padding works, and how well does it work?
- Given a timing side channel with a random delay padding protection: what can an attacker still do?

Academia vs. real-world



“Butter bei die Fische”

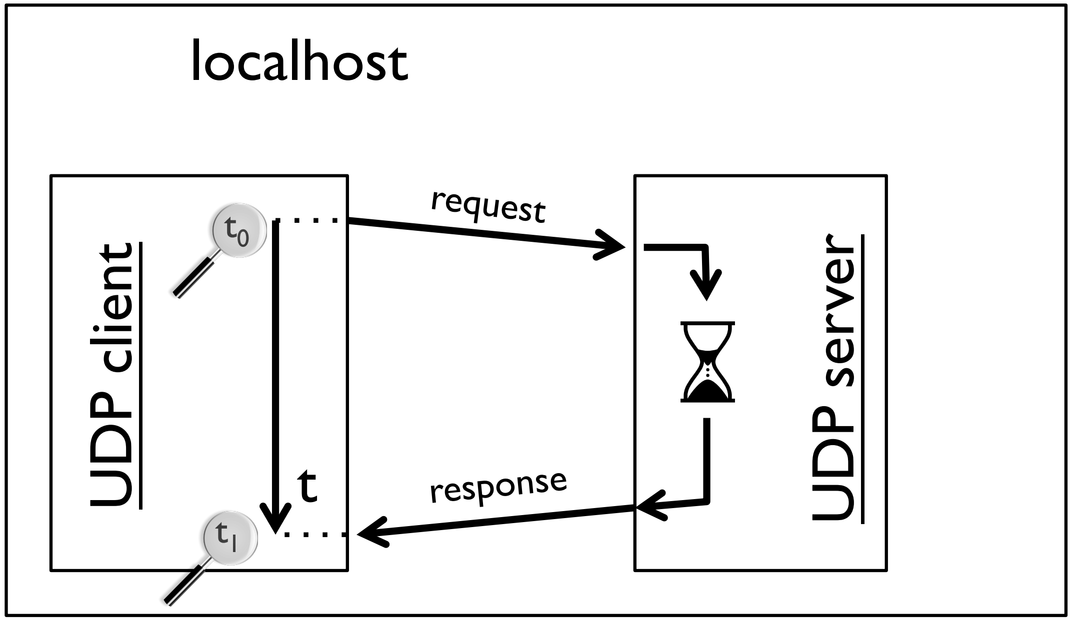
Attacker scenario

- Ristenpart et al. *
  - mapped the internal cloud infrastructure of the Amazon EC2 service
  - instantiate new VMs until one is placed co-resident with the target
  - *"just a few dollars invested in launching VMs can produce a* ***40% chance*** *of placing a malicious VM on* ***the same physical server*** *as a target customer"*
- We want to show the efficiency of countermeasures, not attacks
- → For this talk, we need a very strong (but still practical) attacker **(local)**

* Thomas Ristenpart and Eran Tromer and Hovav Shacham and Stefan Savage.
Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds
ACM Conference on Computer and Communications Security, pp. 199-212, ACM, 2009.

Simple UDP-Ping-Pong protocol

- Measurement setup
  - measurement on localhost
  - idle Ubuntu machine, no GUI
  - switched off Intel Speedstepping, C-States, all unnecessary services
  - unplugged network cable
  - …



Our Timing Measurement Dataset

- 20 different datasets
- Measured 1 mio. times per delay
- Minimum timing difference was 10 nanoseconds
- Maximum timing difference was 5 milliseconds
- Manually removed obvious outliers (50-100 per dataset)

Timing difference
10 ns
20 ns
40 ns
80 ns
160 ns
320 ns
640 ns
1,280 us
2,560 us
5,120 us
10,240 us
20,480 us
40,960 us
81,920 us
163,840 us
327,680 us
655,360 us
1,310720 ms
2,621,440 ms
5,242,880 ms

Our Timing Measurement Dataset

Results with no time delay padding

- Delays <100 nanoseconds hardly distinguishable
  - further research e.g. with other hardware, other hypothesis tests, more measurements
- Delays > 5 microseconds distinguishable with high confidence (p=0.00) with just ~20 measurements

Timing difference	# required measurements
10 ns	?
20 ns	>300.000, p=?
40 ns	>300.000, p=?
80 ns	31, p=0.14
160 ns	16, p=0.02
320 ns	16, p=0.02
640 ns	16, p=0.02
1,280 us	16, p=0.02
2,560 us	16, p=0.01
5,120 us	16, p=0.00
10,240 us	16, p=0.00
20,480 us	16, p=0.00
40,960 us	16, p=0.00
81,920 us	16, p=0.00
163,840 us	16, p=0.00
327,680 us	16, p=0.00
655,360 us	16, p=0.00
1,310720 ms	16, p=0.00
2,621,440 ms	16, p=0.00
5,242,880 ms	16, p=0.00

Random Delay Padding

Timing difference
10 ns
20 ns
40 ns
80 ns
160 ns
320 ns
640 ns
1,280 us
2,560 us
5,120 us
10,240 us
20,480 us
40,960 us
81,920 us
163,840 us
327,680 us
655,360 us
1,310720 ms
2,621,440 ms
5,242,880 ms

Creating the random delay padding datasets:

- for each of the datasets, add a random uniform delay per entry (with nanosecond accuracy)
- random delays were: 1 microsecond, 10 microseconds, …, 100 milliseconds (6 different delays)
- → 120 different datasets

Random Delay Padding

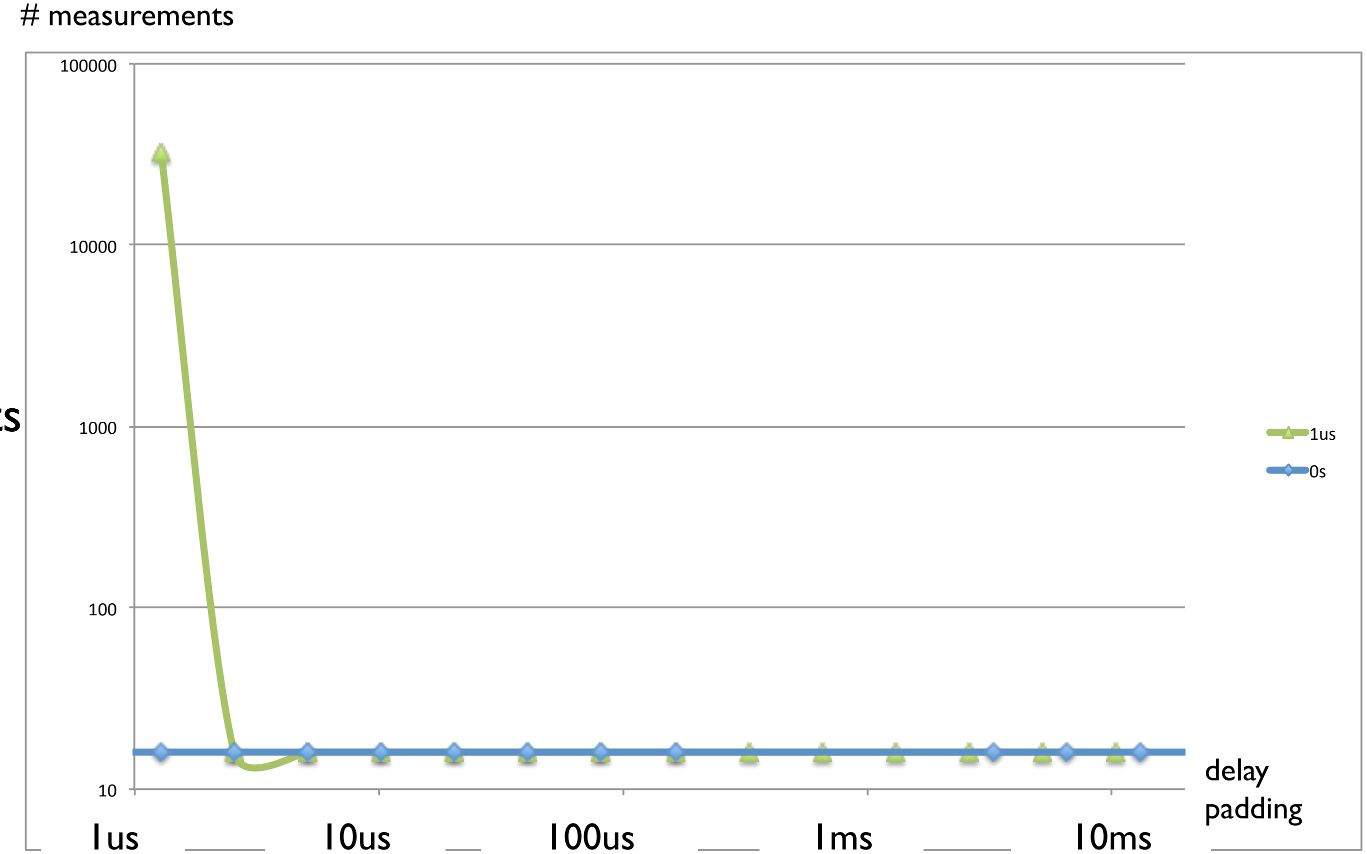
I microsecond random delay padding

Delay	# measurements	Random Delay 1us
10 ns	?	X
20 ns	>300.000, p=?	X
40 ns	>300.000, p=?	X
80 ns	31, p=0.14	X
160 ns	16, p=0.02	X
320 ns	16, p=0.02	X
640 ns	16, p=0.02	X
1,280 us	16, p=0.02	32768, p=0.00
2,560 us	16, p=0.01	16, p=0.01
5,120 us	16, p=0.00	16, p=0.00
10,240 us	16, p=0.00	16, p=0.00
20,480 us	16, p=0.00	16, p=0.00
40,960 us	16, p=0.00	16, p=0.00
81,920 us	16, p=0.00	16, p=0.00
163,840 us	16, p=0.00	16, p=0.00
327,680 us	16, p=0.00	16, p=0.00
655,360 us	16, p=0.00	16, p=0.00
1,310720 ms	16, p=0.00	16, p=0.00
2,621,440 ms	16, p=0.00	16, p=0.00
5,242,880 ms	16, p=0.00	16, p=0.00

Random Delay Padding

1 microsecond random delay padding

- Timing delay of **1 microsecond** distinguishable with **~32.000** measurements
- Timing delay of **2** microseconds distinguishable with **~16** measurements



Random Delay Padding

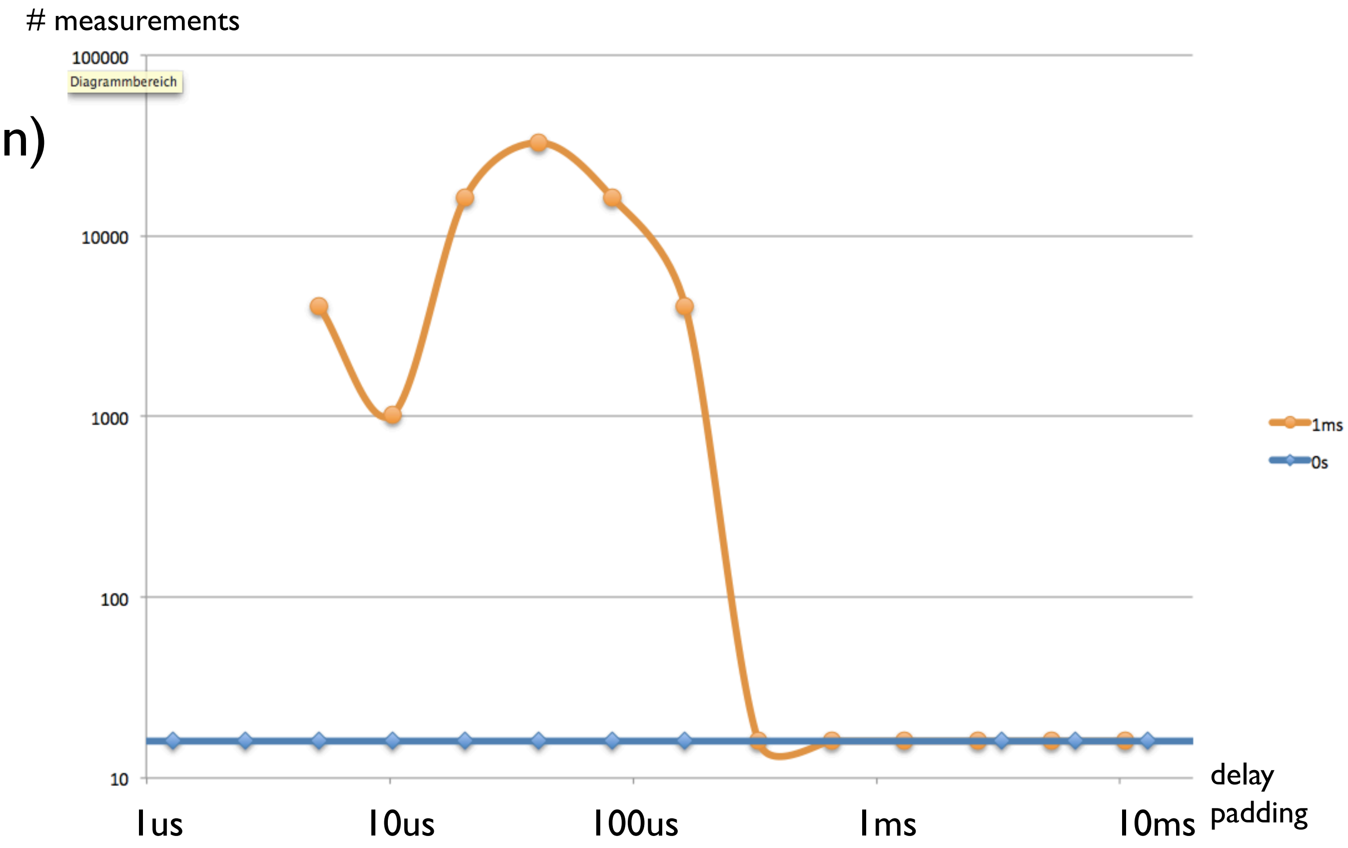
1 millisecond random delay padding (Dan's mitigation)

Delay	# measurements	Random Delay 1ms
10 ns	?	X
20 ns	>300.000, p=?	X
40 ns	>300.000, p=?	X
80 ns	31, p=0.14	X
160 ns	16, p=0.02	X
320 ns	16, p=0.02	X
640 ns	16, p=0.02	X
1,280 us	16, p=0.02	X
2,560 us	16, p=0.01	X
5,120 us	16, p=0.00	4096, p=0.00
10,240 us	16, p=0.00	1024, p=0.05
20,480 us	16, p=0.00	16384, p=0.03
40,960 us	16, p=0.00	32768, p=0.00
81,920 us	16, p=0.00	16384, p=0.03
163,840 us	16, p=0.00	4096, p=0.04
327,680 us	16, p=0.00	16, p=0.00
655,360 us	16, p=0.00	16, p=0.00
1,310720 ms	16, p=0.00	16, p=0.00
2,621,440 ms	16, p=0.00	16, p=0.00
5,242,880 ms	16, p=0.00	16, p=0.00

Random Delay Padding

I millisecond random padding (Dan's mitigation)

- Timing delay of **5 microseconds** distinguishable with **~4.000** measurements
- Timing delay of **300** microseconds distinguishable with **~16** measurements



Random Delay Padding

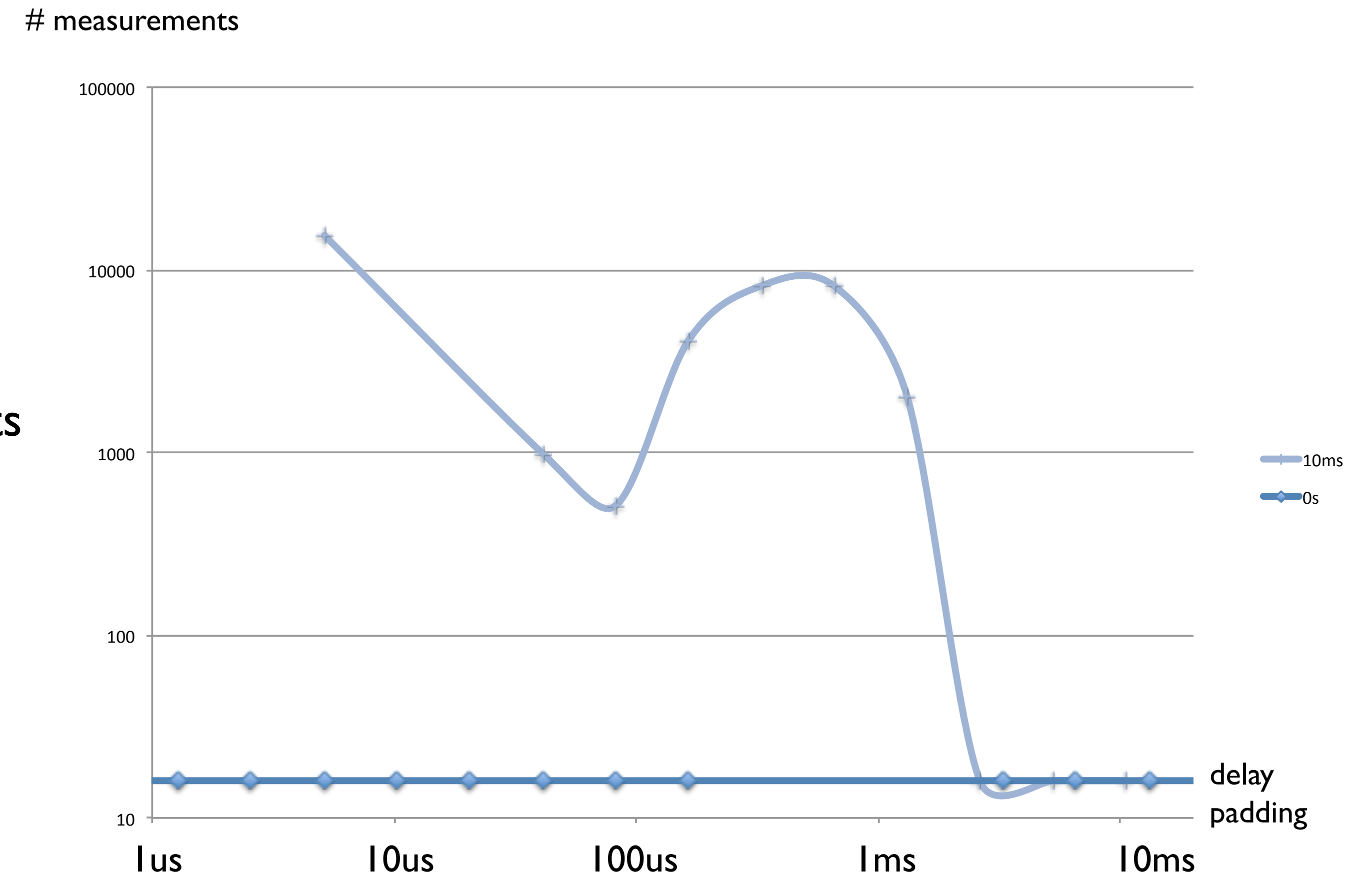
10 millisecond random delay padding

Delay	# measurements	Random Delay 10ms
10 ns	?	X
20 ns	>300.000, p=?	X
40 ns	>300.000, p=?	X
80 ns	31, p=0.14	X
160 ns	16, p=0.02	X
320 ns	16, p=0.02	X
640 ns	16, p=0.02	X
1,280 us	16, p=0.02	X
2,560 us	16, p=0.01	X
5,120 us	16, p=0.00	15625, p=0.0
10,240 us	16, p=0.00	X
20,480 us	16, p=0.00	X
40,960 us	16, p=0.00	X
81,920 us	16, p=0.00	992, p=0.00
163,840 us	16, p=0.00	4096, p=0.02
327,680 us	16, p=0.00	8192, p=0.04
655,360 us	16, p=0.00	8192, p=0.00
1,310720 ms	16, p=0.00	2048, p=0.05
2,621,440 ms	16, p=0.00	16, p=0.00
5,242,880 ms	16, p=0.00	16, p=0.00

Random Delay Padding

10 milliseconds random padding

- Timing delay of **5 microseconds** distinguishable with ~**15.000** measurements
- Timing delay of **2** milliseconds distinguishable with **~16** measurements

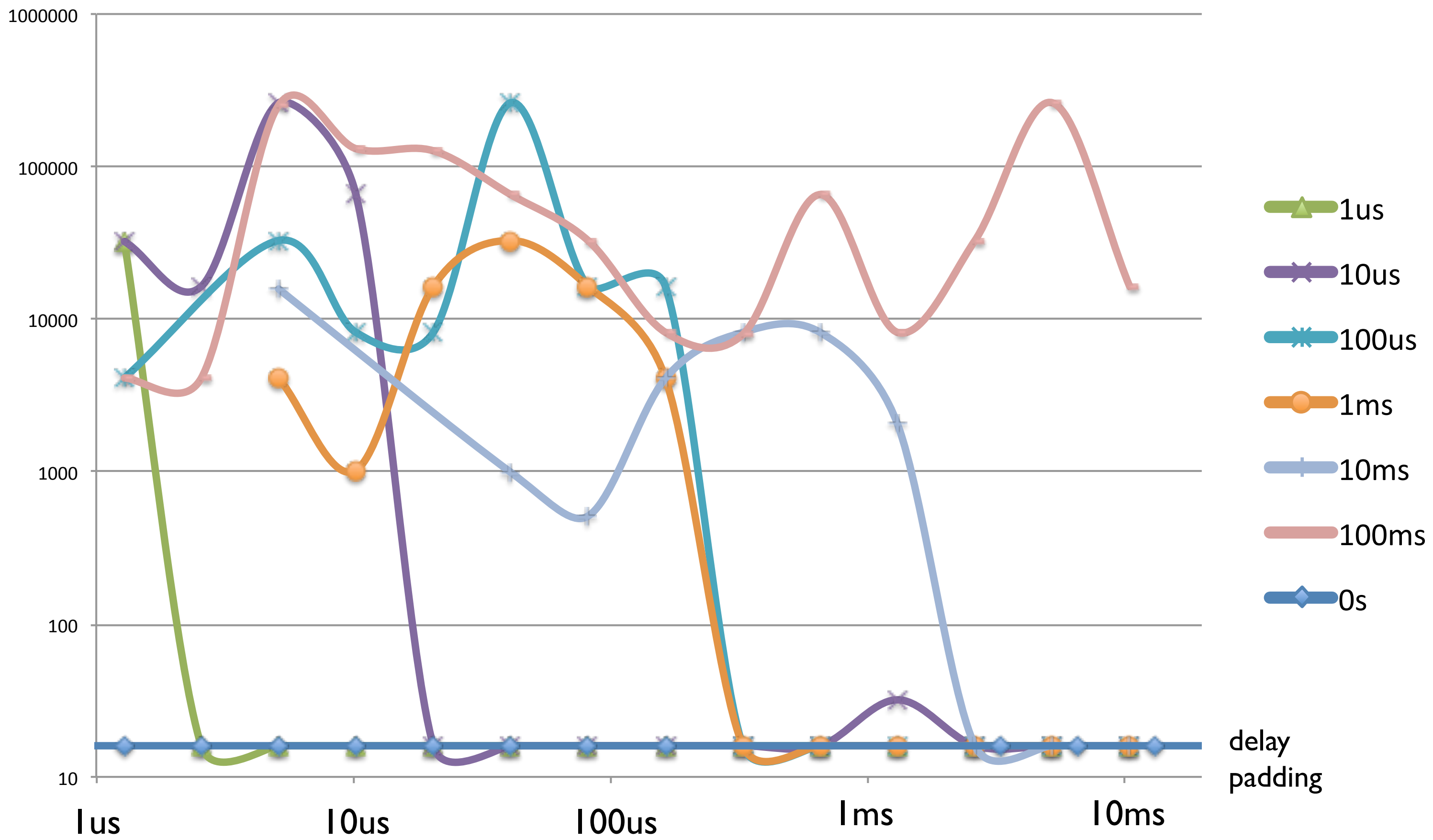


Overview of random delay paddings.

<u>Important</u>:

- where does function start (distinguishable)?

- where does function drop to base line (trivial)?



Summary random delay padding:

Delay type	None	Delay to WCET	Random Delay
Impact on Performance	Best	Worst	$t_{max}/2$
Impact on Security	Worst	Best	Requires more probes to cancel out noise

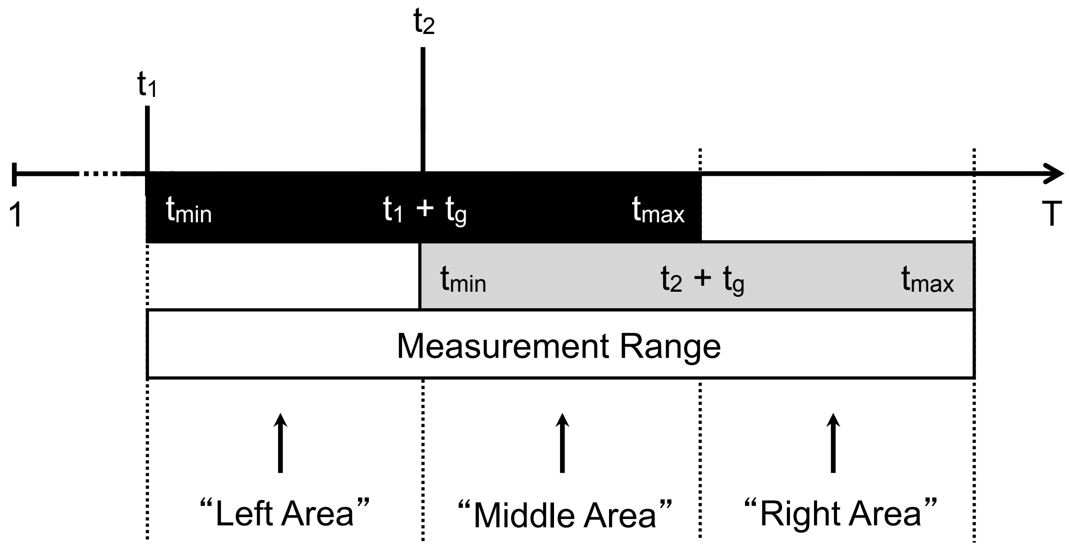
Deterministic and Unpredictable delay padding

Sebastian Schinzel, *An Efficient Mitigation Method for Timing Side Channels on the Web*,
2nd International Workshop on Constructive Side-Channel Analysis and Secure Design (COSADE 2011)
http://sebastian-schinzel.de/_download/cosade-2011-extended-abstract.pdf

Sebastian Schinzel, *Unintentional and Hidden Information Leaks in Networked Software Applications*
PhD Thesis 2012, Universität Erlangen-Nürnberg - Lehrstuhl für Informatik 1
http://www.opus.ub.uni-erlangen.de/opus/frontdoor.php?source_opus=3271

"Deterministic and Unpredictable Delay (DUD)"

- Delay $\boldsymbol{t_g}$ is deterministic for any given user input ***u***
- Attacker cannot guess delay without knowing secret configuration value ***k***
- Protects user-adjustable portion of all values from leaking

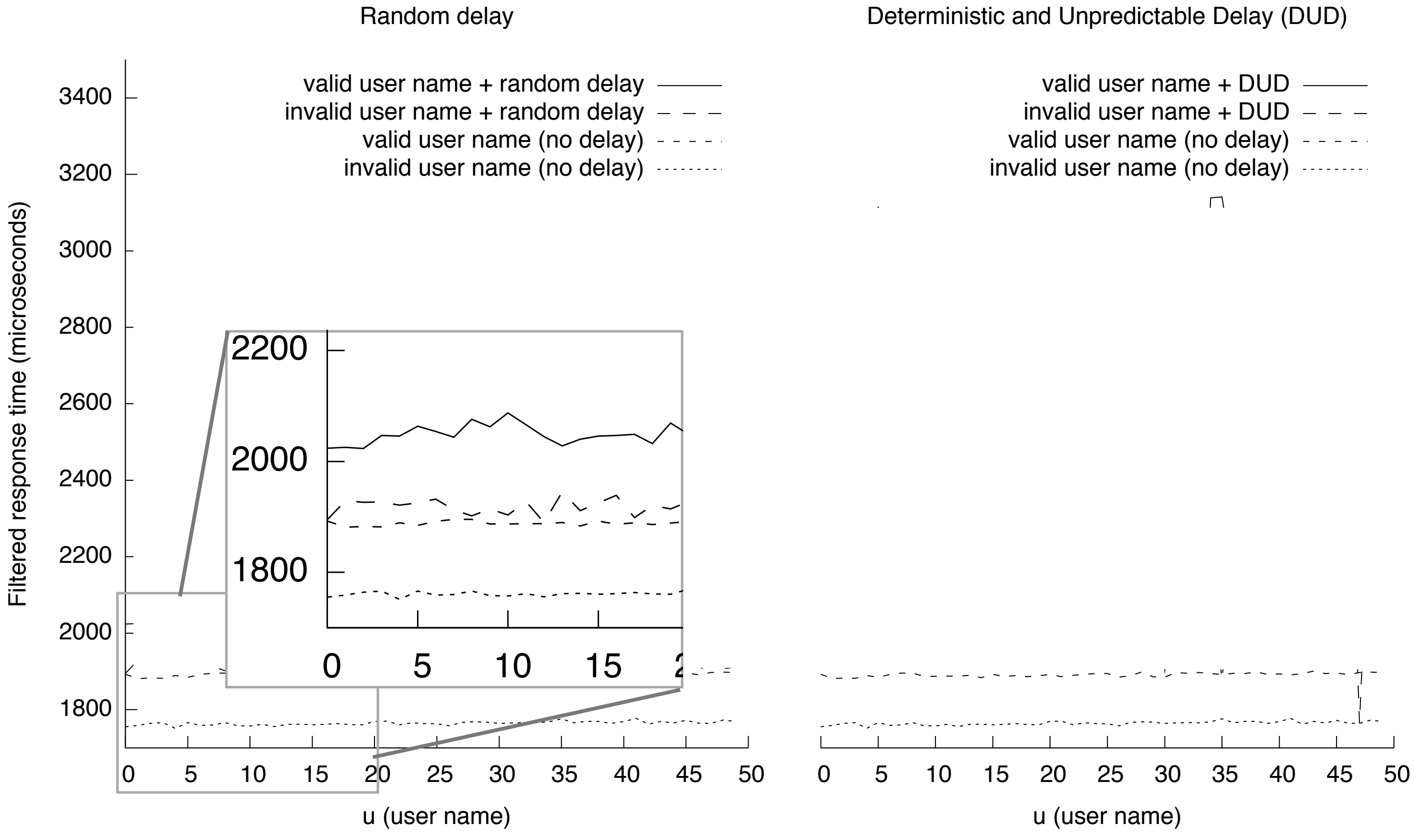


<u>Pseudo implementation:</u>

function $g(u)$ {
  $k := \langle$secret key unknown to the attacker$\rangle$
  $t_g := h(u, k) \bmod t_{max}$
  sleep_nano_seconds(t_g)
}

Comparison of filtered measurements:

- Timing difference: 250 μs
- Maximum delay: t_{max} = 1250 μs
- same performance impact for both delay strategies
- DUD produces much more noise (independently of the amount of measurements)



Delay type	**None**	**Delay to WCET**	**Random Delay**	**Deterministic and Unpredictable Delay**
Impact on Performance	Best	Worst	$t_{max}/2$	$t_{max}/2$
Impact on Security	Worst	Best	Requires more probes to cancel out noise	Offers best protection for fraction of values (adjustable via t_{max})

Summary

- Local attacker are relevant in practice
- Attacker can distinguish ~160 nanoseconds with few measurements and low error rate
- Random delays are neither an effective, nor an efficient mitigation for timing side channels
- Others mitigation techniques work better, depending on the usage scenario
- Deterministic and unpredictable delay is one example

Find information on measurement setups, the datasets, the code, and the scripts here (shortly after the talk):

http://seecurity.org/29c3/

Thanks for your attendance!

Questions? Discussion.

Web: http://seecurity.org/

Twitter: @seecurity