

SIDE CHANNEL ANALYSIS

and the Gap Between Research and Practice

23 February 2021

Billy Brumley

billy.brumley AT tuni DOT fi

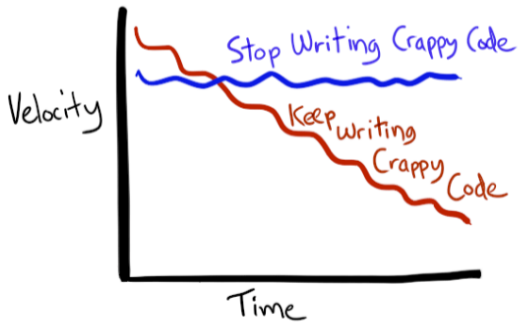
Network and Information Security Group (NISEC)

Tampere University, Tampere, FINLAND

Buzzword: Technical Debt

Technical debt is a concept in software development that reflects the implied cost of additional rework **caused by choosing an easy (limited) solution now** instead of using a better approach that would take longer. Technical debt can be compared to monetary debt. If technical debt is not repaid, it can accumulate 'interest', making it **harder to implement changes later on**.

-Wikipedia



-Agile Pearls

Elliptic curve cryptography (ECC) and ECDSA

Legacy curves over prime fields

$$E(\mathbb{F}_p) : y^2 = x^3 + ax + b$$

ECDSA

$\#\langle G \rangle = n$ prime, private key $0 < d < n$, public key $[d]G$, nonce $0 < k < n$:

$$r = ([k]G)_x \pmod n$$

$$s = (h(m) + dr)k^{-1} \pmod n$$

Lattices

The problem

Assume j equations of the form

$$m_i - s_i k_i + dr_i \equiv 0 \pmod{n}$$

so j equations and $j + 1$ unknowns. What if we know part of each of j of the unknowns?

Howgrave-Graham and Smart (1999)

“Lattice Attacks on Digital Signature Schemes”

The solution

Here we've filtered signatures that suggest a “small” k_i . Use lattice methods to produce a “small” solution. There's a good chance it might be the right one.

Lattice attack: intuition

```
1 1?????...  ---- AVERAGE SOLUTION
2 1?????...  n 111111...
3 1?????...
4 1?????...
5 1?????...
6 1?????...
7 1?????...
8 1?????...  1/2
9 1?????...
10 1?????...
11 1?????...
12 1?????...
13 1?????...
14 1?????...
15 1?????...
16 1?????...  ----
17 01?????...
18 01?????...
19 01?????...
20 01?????...  1/4
21 01?????...
22 01?????...
23 01?????...
24 01?????...  ----
25 001????...
26 001????...  1/8
27 001????...
28 001????...  ----
29 0001???...  1/16
30 0001???...  ----
31 00001?...  1/32
```



```
1 00000?...  ---- OUR SOLUTION
2 00000?...  n 111111...
3 00000?...
4 00000?...
5 00000?...
6 00000?...
7 00000?...
8 00000?...
9 00000?...
10 00000?...
11 00000?...
12 00000?...
13 00000?...
14 00000?...
15 00000?...
16 00000?...
17 00000?...
18 00000?...
19 00000?...
20 00000?...
21 00000?...
22 00000?...
23 00000?...
24 00000?...
25 00000?...
26 00000?...
27 00000?...
28 00000?...
29 00000?...
30 00000?...
31 00000?...
```



Lattice attack: intuition

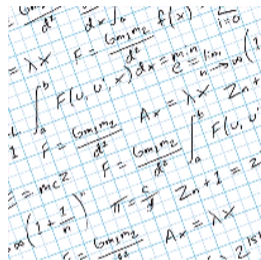
```
1 1?????...  ---- AVERAGE SOLUTION
2 1?????...  n 111111...
3 1?????...
4 1?????...
5 1?????...
6 1?????...
7 1?????...
8 1?????...  1/2
9 1?????...
10 1?????...
11 1?????...
12 1?????...
13 1?????...
14 1?????...
15 1?????...
16 1?????...  ----
17 01?????...
18 01?????...
19 01?????...
20 01?????...  1/4
21 01?????...
22 01?????...
23 01?????...
24 01?????...  ----
25 001????...
26 001????...  1/8
27 001????...
28 001????...  ----
29 0001???...  1/16
30 0001???...  ----
31 00001?...  1/32
```



```
1 00000?...  ---- OUR SOLUTION
2 00000?...  n 111111...
3 00000?...
4 00000?...
5 00000?...
6 00000?...
7 00000?...
8 00000?...
9 00000?...
10 00000?...
11 00000?...
12 00000?...
13 00000?...
14 00000?...
15 00000?...
16 00000?...
17 00000?...
18 00000?...
19 00000?...
20 00000?...
21 00000?...
22 00000?...
23 00000?...
24 00000?...
25 00000?...
26 00000?...
27 00000?...
28 00000?...
29 00000?...
30 00000?...
31 00000?...
```



Lattice attacks for applied cryptographers



OpenSSL: 1998–now

*OpenSSL is a software library for applications that secure communications over computer networks against eavesdropping or need to identify the party at the other end. It is widely used by Internet servers, including the majority of HTTPS websites. OpenSSL contains an **open-source implementation of the SSL and TLS protocols**. The core library, written in the C programming language, **implements basic cryptographic functions** and provides various utility functions.*

–Wikipedia

Let's Encrypt is the most popular authority with 58.2% of all issued certificates.

–Censys (2019)

RSA: remote timing attack, decryption

OpenSSL CVE-2003-0147 (USENIX 2003)

Remote Timing Attacks are Practical

David Brumley
Stanford University
dbrumley@cs.stanford.edu

Dan Boneh
Stanford University
dabo@cs.stanford.edu

Abstract

Timing attacks are usually used to attack weak computing devices such as smartcards. We show that timing attacks apply to general software systems. Specifically, we devise a timing attack against OpenSSL. Our experiments show that we can extract private keys from an OpenSSL-based web server running on a machine in the local network. Our results demonstrate that timing at-

The attacking machine and the server were in different buildings with three routers and multiple switches between them. With this setup we were able to extract the SSL private key from common SSL applications such as a web server (Apache+mod_SSL) and a SSL-tunnel.

Interprocess. We successfully mounted the attack between two processes running on the same machine. A hosting center that hosts two domains on the same machine might give management access to

RSA: L1 dcache attack, decryption

OpenSSL CVE-2005-0109 (BSDCan 2004)

CACHE MISSING FOR FUN AND PROFIT

COLIN PERCIVAL

ABSTRACT. We describe the construction of a channel between processes via the state of a shared memory cache, and its use in the cryptanalysis of RSA. Unlike earlier side-channel attacks involving memory caches, our attack has the remarkable property of only requiring that a single private key operation be observed.

Countermeasure: “constant time” exponentiation flag

```
    if (bits == 0)
@@ -364,6 +379,11 @@ int BN_mod_exp_mont(BIGNUM *rr, const BIGNUM *a, const BIGNUM *p,
    BIGNUM *val[TABLE_SIZE];
    BN_MONT_CTX *mont=NULL;

+    if (BN_get_flags(p, BN_FLG_EXP_CONSTTIME) != 0)
+        {
+            return BN_mod_exp_mont_consttime(rr, a, p, m, ctx, in_mont);
+        }
+
```

RSA: multiple vulnerabilities

OpenSSL 2007 (IMACC 2007)

New Branch Prediction Vulnerabilities in OpenSSL and Necessary Software Countermeasures

Onur Aciicmez¹, Shay Gueron^{2,3}, and Jean-Pierre Seifert^{1,4}

¹ Samsung Information Systems America, San Jose, 95134, USA

² Department of Mathematics, University of Haifa, Haifa, 31905, Israel

³ Intel Corporation, IDC, Israel

⁴ Institute for Computer Science, University of Innsbruck, 6020 Innsbruck, Austria

onur.aciicmez@gmail.com, shay@math.haifa.ac.il,

jeanpierreseifert@yahoo.com

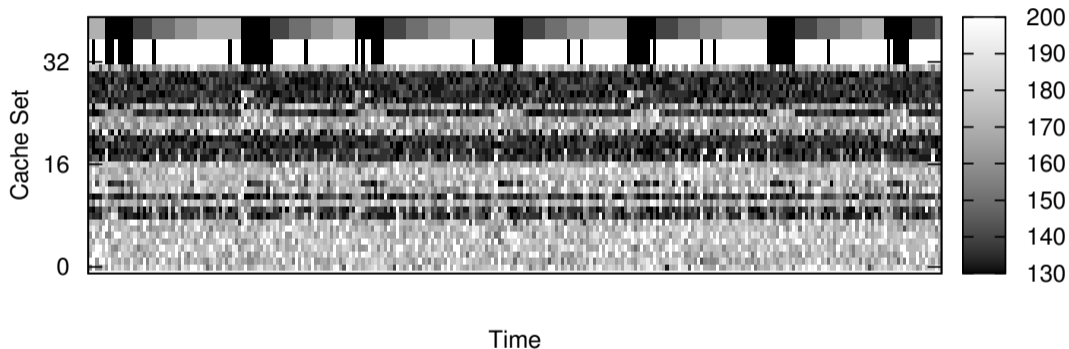
Abstract. Software based side-channel attacks allow an unprivileged spy process to extract secret information from a victim (cryptosystem) process by exploiting some indirect leakage of “side-channel” information. It has been realized that some components of modern computer microarchitectures leak certain side-channel information and can create unforeseen security risks. An example of such

Countermeasure: “constant time” flag, “no branch” inversion

```
@@ -210,6 +210,11 @@ BIGNUM *BN_mod_inverse(BIGNUM *in,  
    BIGNUM *ret=NULL;  
    int sign;  
  
+     if (BN_get_flags(n, BN_FLG_CONSTTIME) != 0)  
+         {  
+             return BN_mod_inverse_no_branch(in, a, n, ctx);  
+         }  
+
```

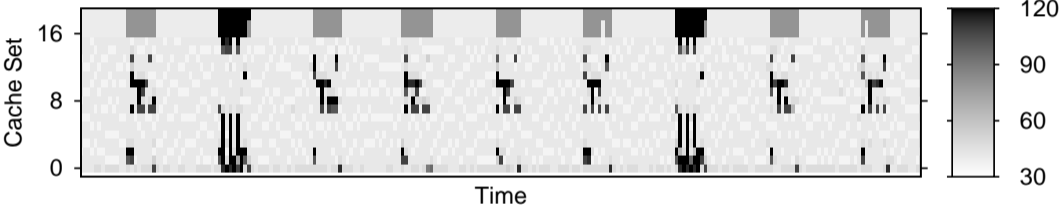
ECDSA: L1 dcache attack, point multiplication

OpenSSL 2009 (Asiacrypt 2009)



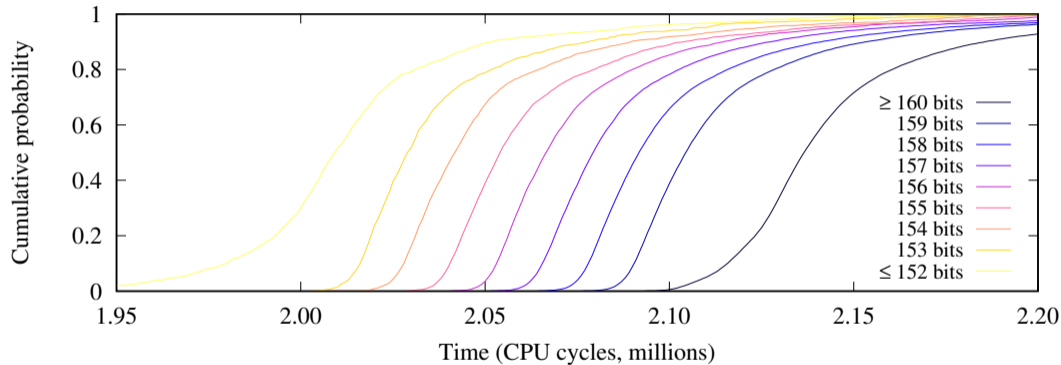
DSA: L1 icache attack, exponentiation

OpenSSL 2010 (CHES 2010)



ECDSA remote timing attack, point multiplication

OpenSSL CVE-2011-1945 (ESORICS 2011)



ECDH remote bug attack

OpenSSL CVE-2011-4354 (CT-RSA 2012)

```
bbrumley@cb35: ~/svnrepos/ecc_bug_attack
bbrumley@cb35:~/svnrepos/ecc_bug_attack$ root/bin/stunnel stunnel.conf 2>&1 >/dev/null | grep --color=never 'accepted|failed'
2017.04.13 11:16:09 LOG5[5110:4151196480]: Service foobar accepted connection from 127.0.0.1:53890
2017.04.13 11:16:09 LOG3[5110:4151196480]: SSL_accept: 1408F119: error:1408F119:SSL routines:SSL3_GET_RECORD:decryption failed or bad record mac
2017.04.13 11:16:11 LOG5[5110:4151196480]: Service foobar accepted connection from 127.0.0.1:53892
2017.04.13 11:16:11 LOG5[5110:4151196480]: Service foobar accepted connection from 127.0.0.1:53896
2017.04.13 11:16:11 LOG3[5110:4151196480]: SSL_accept: 1408F119: error:1408F119:SSL routines:SSL3_GET_RECORD:decryption failed or bad record mac
2017.04.13 11:16:12 LOG5[5110:4151196480]: Service foobar accepted connection from 127.0.0.1:53898
2017.04.13 11:16:12 LOG5[5110:4151196480]: Service foobar accepted connection from 127.0.0.1:53902
2017.04.13 11:16:13 LOG5[5110:4151196480]: Service foobar accepted connection from 127.0.0.1:53906
2017.04.13 11:16:13 LOG5[5110:4151196480]: Service foobar accepted connection from 127.0.0.1:53910
2017.04.13 11:16:14 LOG5[5110:4151196480]: Service foobar accepted connection from 127.0.0.1:53914
2017.04.13 11:16:14 LOG5[5110:4151196480]: Service foobar accepted connection from 127.0.0.1:53918
2017.04.13 11:16:15 LOG5[5110:4151196480]: Service foobar accepted connection from 127.0.0.1:53922
2017.04.13 11:16:15 LOG5[5110:4151196480]: Service foobar accepted connection from 127.0.0.1:53926
2017.04.13 11:16:15 LOG3[5110:4151196480]: SSL_accept: 1408F119: error:1408F119:SSL routines:SSL3_GET_RECORD:decryption failed or bad record mac
bbrumley@cb35:~/svnrepos/ecc_bug_attack$ python ptfind_tree.py ECDH-ECDSA
HANDSHAKE: FAIL points/ptfind_0007.out 7 press a key to continue
HANDSHAKE: PASS points/ptfind_0077.out 7:0:0:0:7 press a key to continue
HANDSHAKE: FAIL points/ptfind_0069.out 7:0:0:0:-7 press a key to continue
HANDSHAKE: PASS points/ptfind_0697.out 7:0:0:0:-7:0:0:0:7 press a key to continue
HANDSHAKE: PASS points/ptfind_0689.out 7:0:0:0:-7:0:0:0:-7 press a key to continue
HANDSHAKE: PASS points/ptfind_0695.out 7:0:0:0:-7:0:0:0:5 press a key to continue
HANDSHAKE: PASS points/ptfind_068b.out 7:0:0:0:-7:0:0:0:-5 press a key to continue
HANDSHAKE: PASS points/ptfind_0693.out 7:0:0:0:-7:0:0:0:3 press a key to continue
HANDSHAKE: PASS points/ptfind_068d.out 7:0:0:0:-7:0:0:0:-3 press a key to continue
HANDSHAKE: PASS points/ptfind_0691.out 7:0:0:0:-7:0:0:0:1 press a key to continue
HANDSHAKE: FAIL points/ptfind_068f.out 7:0:0:0:-7:0:0:0:-1 press a key to continue
PARTIAL KEY GUESS: 7:0:0:0:-7:0:0:0:-1:0:0:0
bbrumley@cb35:~/svnrepos/ecc_bug_attack$ ./keyprint < p256.pem | ./naf
7:0:0:0:-7:0:0:0:-1:0:0:0:-3:0:0:0:0:-3:0:0:0:0:-7:0:0:0:0:-7:0:0:0:1:0:0:0:-5:0:0:0:1:0:0:0:7:0:0:0:0:0:-1:0:0:0:0:0:-3:0:0:0:-5:0:0:0:-5:0:0:0:-5:0:0:0:0:-3:0:0:0:0:3:0:0:0:0:-5:0:0:0:3:0:0:0:0:-1:0:0:0:0:0:-5:0:0:0:1:0:0:0:-5:0:0:0:0:7:0:0:0:0:0:-5:0:0:0:3:0:0:0:0:0:-7:0:0:0:1:0:0:0:0:0:-5:0:0:0:0:0:1:0:0:0:0:-5:0:0:0:7:0:0:0:5:0:0:0:-1:0:0:0:-1:0:0:0:0:0:-5:0:0:0:7:0:0:0:5:0:0:0:-3:0:0:0:0:5:0:0:0:0:-3:0:0:0:7:0:0:0:1:0:0:0:0:3:0:0:0:-1:0:0:0:0:5:0:0
bbrumley@cb35:~/svnrepos/ecc_bug_attack$
```

ECDSA Flush+Reload attack, point multiplication

OpenSSL 2014 (CHES 2014)

“Ooh Aah... Just a Little Bit” : A Small Amount of Side Channel Can Go a Long Way

Naomi Benger¹, Joop van de Pol², Nigel P. Smart², and Yuval Yarom¹

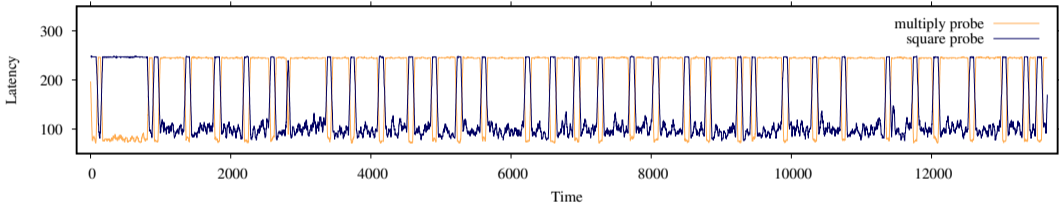
¹ School of Computer Science, The University of Adelaide, Australia
mail.for.minnie@gmail.com, yval@cs.adelaide.edu.au

² Dept. Computer Science, University of Bristol, United Kingdom
joop.vandepol@bristol.ac.uk, nigel@cs.bris.ac.uk

Abstract. We apply the FLUSH+RELOAD side-channel attack based on cache hits/misses to extract a small amount of data from OpenSSL ECDSA signature requests. We then apply a “standard” lattice technique to extract the private key, but unlike previous attacks we are able to make use of the side-channel information from almost all of the observed executions. This means we obtain private key recovery by observing a relatively small number of executions, and by ex-

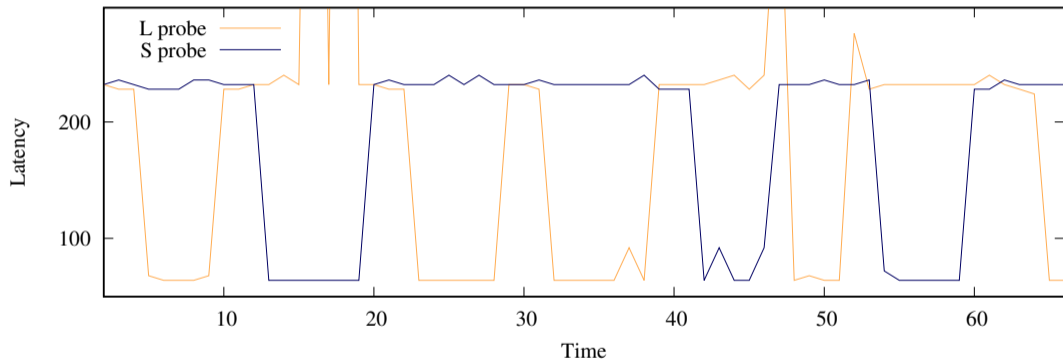
DSA Flush+Reload attack, exponentiation

OpenSSL CVE-2016-2178 (CCS 2016)



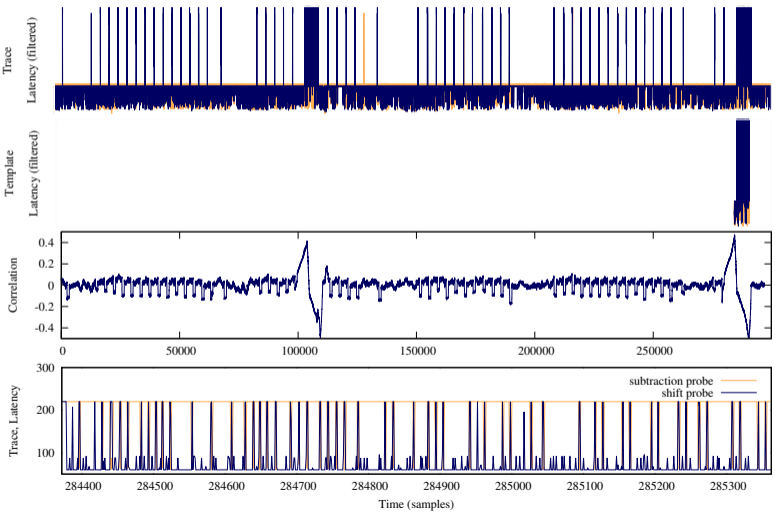
ECDSA Flush+Reload attack, inversion

OpenSSL CVE-2016-7056 (USENIX 2017)



























RSA keygen Flush+Reload attack: multiple vulnerabilities

OpenSSL CVE-2018-0737 (TCHES 2019)



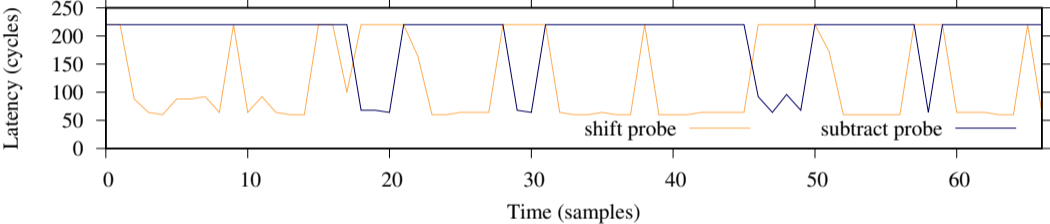
TriggerFlow: Continuous Integration for execution paths

DIMVA 2018

Status	Pipeline	Commit	Stages
	#1495 by  latest	 patched/mas...  f3b5c690  [master:c8147d37ccaaf28c...	  00:07:42  1 hour ago
	#1494 by 	 patched/mas...  81d96fbd  [master:fe16ae5f95fa86ddb...	  00:07:52  1 hour ago
	#1493 by 	 patched/mas...  9fb8e7df  [master:0b76ce99aaa5678b...	  00:07:46  1 hour ago

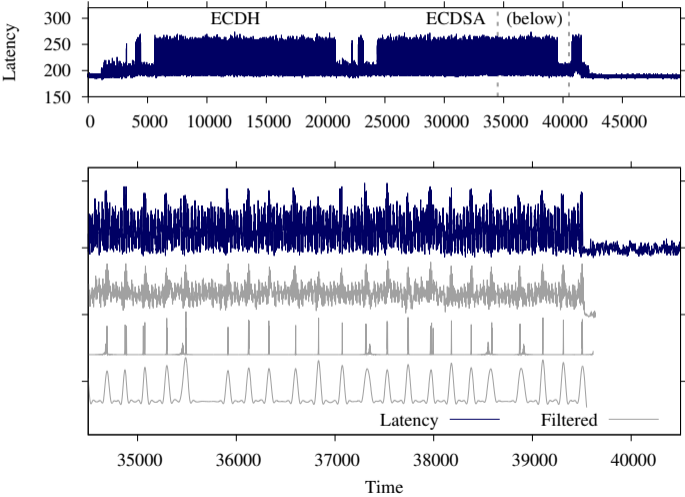
SM2: multiple vulnerabilities

OpenSSL 2018 (ACSAC 2018)



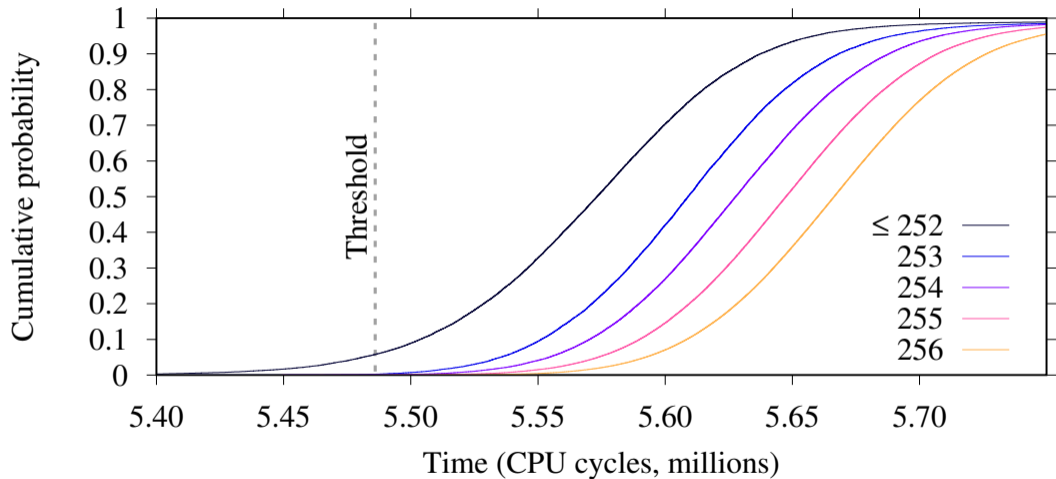
ECDSA: port contention attack, scalar multiplication

OpenSSL CVE-2018-5407 (Oakland 2019)



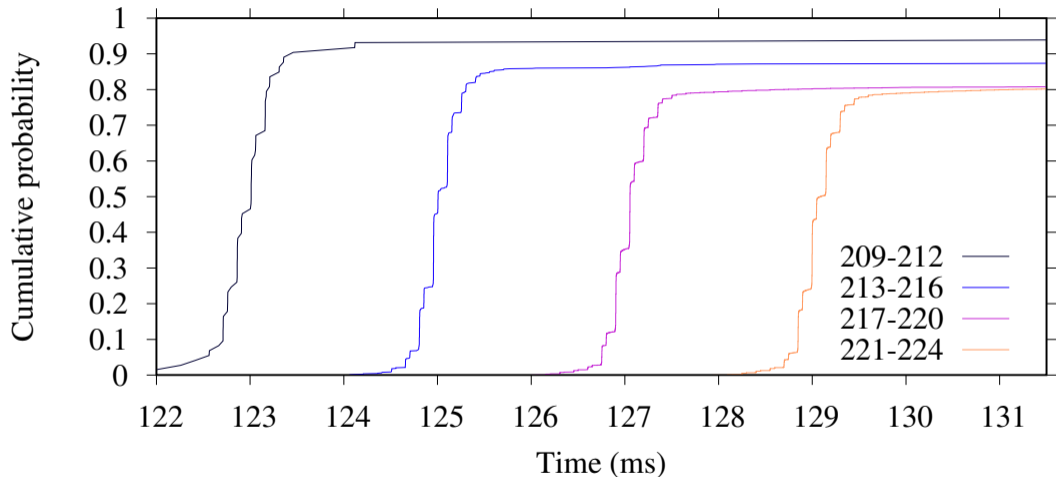
ECDSA: remote timing attack, scalar multiplication

OpenSSL CVE-2019-1547 (USENIX 2020)



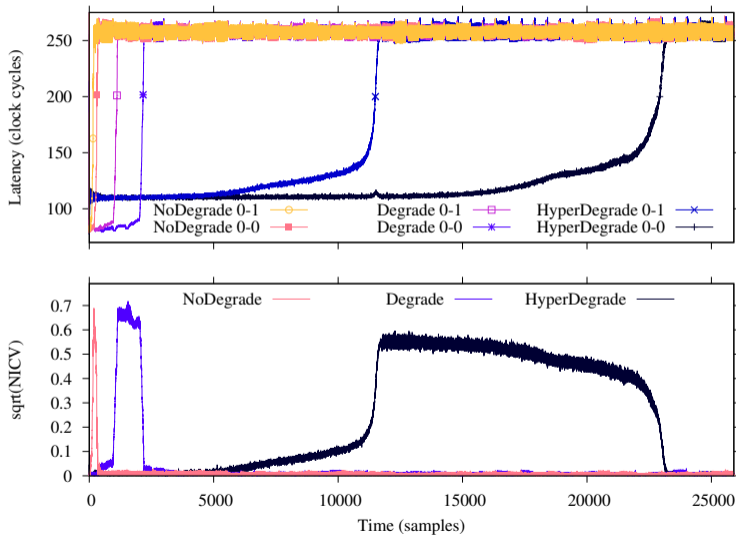
DSA: remote timing attack, exponentiation

NSS CVE-2020-12399 (CCS 2020)



DH: Flush+Reload attack, padding

OpenSSL 2021 (arXiv:2101.01077 2021)



Lessons Learned

- ▶ Stop gap countermeasures bite you in the end.
- ▶ Breaking things isn't enough. Fix things.
- ▶ Deployed libraries are not your research playground.
- ▶ Stop looking at crypto in a vacuum.
- ▶ OpenSSL is very different pre-HeartBleed vs post. (contributions, policies)

Contributing to OpenSSL

- ▶ PRs welcome: we will help you.
- ▶ Disclosure: show them your data.
- ▶ New OpenSSL security policy: the good, bad, ugly
- ▶ OpenSSL 3.0 architecture changes

Future Work

- ▶ More TriggerFlow unit tests
- ▶ Many ubiquitous crypto libs have similar design patterns—what about them?
- ▶ Re-design OpenSSL EC module
- ▶ More OpenSSL engines