Side-Channel Analysis of SM2: A Late-Stage Featurization Case Study

Nicola Tuveri
Tampere University of Technology
Tampere, Finland
nicola.tuveri@tut.fi

Sohaib ul Hassan
Tampere University of Technology
Tampere, Finland
sohaibulhassan@tut.fi

Cesar Pereida García
Tampere University of Technology
Tampere, Finland
cesar.pereidagarcia@tut.fi

Billy Bob Brumley
Tampere University of Technology
Tampere, Finland
billy.brumley@tut.fi

ABSTRACT

SM2 is a public key cryptography suite originating from Chinese standards, including digital signatures and public key encryption. Ahead of schedule, code for this functionality was recently mainlined in OpenSSL, marked for the upcoming 1.1.1 release. We perform a security review of this implementation, uncovering various deficiencies ranging from traditional software quality issues to side-channel risks. To assess the latter, we carry out a side-channel security evaluation and discover that the implementation hits every pitfall seen for OpenSSL's ECDSA code in the past decade. We carry out remote timings, cache timings, and EM analysis, with accompanying empirical data to demonstrate secret information leakage during execution of both digital signature generation and public key decryption. Finally, we propose, implement, and empirically evaluate countermeasures.

KEYWORDS

software engineering; applied cryptography; public key cryptography; side-channel analysis; timing attacks; cache-timing attacks; power analysis; TVLA; SM2; OpenSSL

ACM Reference Format:
Nicola Tuveri, Sohaib ul Hassan, Cesar Pereida García, and Billy Bob Brumley. 2018. Side-Channel Analysis of SM2: A Late-Stage Featurization Case Study. In *2018 Annual Computer Security Applications Conference (ACSAC '18), December 3–7, 2018, San Juan, PR, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3274694.3274725

1 INTRODUCTION

SM2[1] is a suite of elliptic curve public key cryptosystems, standardized as a part of Chinese commercial cryptography mandates. Support for SM2 in OpenSSL landed in the public GitHub repository through pull request (PR) #4793,[2] created in November 2017 by external contributors. During the review process, in January 2018, the OpenSSL team assigned the PR to the Post-1.1.1 milestone, marking functionality intended to be merged *after* the upcoming 1.1.1 release of OpenSSL.

Due to this, SM2 support was excluded from the two alpha releases for OpenSSL 1.1.1. But in March 2018, just *before* the release of the first 1.1.1 beta—and the associated feature freeze—the OpenSSL development team decided to merge the PR into the 1.1.1 beta development cycle, to have a chance to work on it and possibly include SM2 support as part of the upcoming minor release rather than waiting for the next one.[3] Considering that new features can only be added with a new minor release and that the current one (OpenSSL 1.1.0) was released on August 2016, it is likely that a similar—if not longer—development cycle might be required before the SM2 functionality could be added to OpenSSL. The SM2 functionality has thus been part of the beta development cycle since the release of OpenSSL 1.1.1-pre3 (beta 1).

At the time of beta 1 release, the release timetable[4] for OpenSSL 1.1.1 envisioned four beta releases, aiming at 15th May 2018 as the first possible final release date. As such, the addition of SM2 support into the active development branch occurred at an extremely late stage to be included in the upcoming release cycle, giving a remarkably short window for public review before the final release.

The original release timeline was later[5] updated, waiting for the final publication of TLS 1.3 as RFC 8446, adding more beta releases, and eventually shifting the final release date for OpenSSL 1.1.1 to September 11, 2018.

Motivation and goal. The first contribution of our work, our initial security review revealed that the late-stage featurization process resulted in various deficiencies, ranging from code quality issues to traditional software defects, and hinted at significant side-channel analysis (SCA) risks based on previous SCA results targeting ECC within OpenSSL. The goal of this research consists in empirically verifying these SCA deficiencies, and then responsibly mitigate them, aiming at intersecting the upcoming OpenSSL 1.1.1 release to ensure these vulnerabilities do not affect released versions of the library.

[1] https://tools.ietf.org/html/draft-shen-sm2-ecdsa-02
[2] https://github.com/openssl/openssl/pull/4793

ACSAC '18, December 3–7, 2018, San Juan, PR, USA

ACM ISBN 978-1-4503-6569-7/18/12.
https://doi.org/10.1145/3274694.3274725

[3] https://github.com/openssl/openssl/pull/4793#pullrequestreview-104954310
[4] https://mta.openssl.org/pipermail/openssl-project/2018-March/000372.html
[5] https://github.com/openssl/web/pull/55

Furthermore, taking SM2 as a case study, we criticize the current status of the project. It demonstrates that implementing new functionality without reintroducing previously fixed vulnerabilities proves to be unnecessarily challenging, requiring intimate familiarity with internal details of lower level library modules (e.g. where, when, and how constant-time flags *must* be re-/enabled, which codepaths in the lower `EC` and `BIGNUM` modules require to use implementations with SCA mitigations, etc.). Hence, as a secondary goal, we also aim at reviewing the abstraction level at which current SCA countermeasures are implemented, and push for a *secure-by-default* approach—within the boundaries the project enforces for a minor release—so that future implementations will by default benefit from them.

Structure and our contributions. Section 2 reviews relevant background and previous work. We present our security analysis related to the integration of the SM2 functionality in the OpenSSL codebase in Section 3, offering an overview of the issues uncovered. In Section 4, Section 5, and Section 6, respectively, we evaluate SCA defects in the SM2 implementation related to remote timings, cache timings and EM analysis. We propose, implement and empirically evaluate appropriate mitigations in Section 7. Finally, we conclude in Section 8.

2 BACKGROUND

This section describes SM2, various SCA techniques that potentially apply to SM2 implementations, and summarizes previous work on SM2 implementation attacks.

2.1 SM2: Chinese Cryptography Standards

SM2 consists of a digital signature scheme (SM2DSA), a public key encryption scheme (SM2PKE), and a key agreement protocol. In this work, we restrict to SM2DSA and SM2PKE.

Elliptic curves and SM2. While the RFC contains cryptosystem test vectors for several different curves in simplified Weierstrass form (over both prime and binary fields), one required curve[6] consists of all the (x, y) points ($x, y \in GF(p)$) satisfying the equation

$$E : y^2 = x^3 + ax + b$$

over $GF(p)$ along with the point-at-infinity (group identity element). The domain parameters are consistent with legacy ECC, setting p a 256-bit Mersenne-like prime, $a = -3 \in GF(p)$, both $b \in GF(p)$ and generator point $G \in E$ seemingly random, and prime group order n (i.e. co-factor $h = 1$) slightly below 2^{256}.

SM2DSA digital signatures. The user's private-public keypair is (d_A, Q_A) where d_A is chosen uniformly from $[1 \,..\, n-1)$ and $Q_A = [d_A]G$ holds. Denote ZA the personalization string (hash) and m the message. Digital signatures compute as follows.

(1) Compute the digest $h = H(ZA \parallel m)$.
(2) Select a secret nonce k uniformly from $[1 \,..\, n)$.
(3) Compute $(x, y) = [k]G$.
(4) Compute $r = h + x \bmod n$.
(5) Compute $s = (1 + d_A)^{-1}(k - rd_A) \bmod n$.
(6) If any of $r = 0$, $s = 0$, or $s = k$ hold, retry.
(7) Return the SM2 digital signature (r, s).

Hash function H can be any "approved" function, including SM3[7] standardized in a parallel effort. Verification is not relevant to this work, hence we omit the description.

SM2PKE public key encryption. SM2PKE is roughly analogous to ECIES [2, Sec. 5.1]. Denote the ciphertext $C = C_1 \parallel C_2 \parallel C_3$ where, at a high level, C_1 represents the sender's ephemeral Diffie-Hellman public key (point), C_2 is the One-Time-Pad (OTP) ciphertext (with length $|C_2|$), and C_3 is the authentication tag. The recipient with private-public keypair (d_B, Q_B) recovers the plaintext from C as follows.

(1) Convert C_1 to a point on E. If C_1 is not on the curve or does not have order n, return an error.
(2) Compute $(x, y) = [d_B]C_1$, the shared ECDH point.
(3) Compute $z = KDF(x \parallel y, |C_2|)$, the OTP key; $|z| = |C_2|$.
(4) Compute $m' = z \oplus C_2$, i.e. OTP decryption.
(5) Compute $t' = H(x \parallel m' \parallel y)$, the purported tag.
(6) If $t' \neq C_3$ holds, return an error.
(7) Return the plaintext m'.

Encryption is not relevant to this work, hence we omit the description.

2.2 Remote Timing Attacks

Timing attacks exploit differences in the time required by a specific implementation to perform an operation on different inputs. In the case of hardware or software cryptosystem implementations, if there is a correlation between the timing of an operation and some secret inputs, the leaked information might be used to mount an attack to recover secret material.

In his seminal work, Kocher [49] introduces a number of simple timing attacks on modular exponentiation and modular reduction implementations, affecting implementations of public key cryptosystems with a static key such as RSA and static Diffie-Hellman or DSA implementations that precompute the ephemeral part.

Brumley and Boneh [22, 23] demonstrate that timing attacks apply also to general software systems, defying contemporary common belief, by devising a timing attack against the OpenSSL implementation of RSA decryption—exploiting time dependencies introduced by the Montgomery reduction and the multiplication routines—and ultimately retrieving the complete factorization of the key pair modulus. Moreover, they demonstrate that such attacks are practical even in a remote scenario, mounting a real-world attack through a client timing RSA decryptions during SSL handshakes with an OpenSSL server. The attack is effective when performed between two processes running on the same host, across co-located virtual machines, and in local networks. They analyze three possible defenses, favoring RSA blinding, and as a consequence several cryptographic libraries, including OpenSSL, enable RSA blinding by default as a countermeasure.[8]

Acıiçmez et al. [5] further improve the original attack, by targeting Montgomery Multiplications in the *table initialization phase*

[6] OID 1.2.156.10197.1.301

[7] https://tools.ietf.org/html/draft-oscca-cfrg-sm3-02

[8] The issue uncovered by their work was tracked in the public CVE dictionary with the id CVE-2003-0147, and, addressing it, OpenSSL issued a Security Advisory (17 March 2003), and CERT issued vulnerability note VU#997481.

of the sliding window algorithm used to perform the RSA exponentiation in OpenSSL, rather than the *exponentiation phase* itself, increasing the number of multiplications that leak timing information used to retrieve one of the secret prime factors of RSA moduli.

Chen et al. [26] build on these two attacks, improving the success rate through an error detection and correction strategy, thus reducing the number of queries required to mount a successful attack and affecting the total time of the attack and its detectability.

Brumley and Tuveri [21] present another end-to-end remote timing attack: it similarly demonstrates full key recovery in local and remote scenarios, and targets the OpenSSL Montgomery's ladder implementation for scalar multiplication on elliptic curves over binary fields. The Montgomery ladder algorithm is often recommended as a countermeasure to side-channel attacks due to a fixed sequence of curve operations, that does not depend on the values of individual bits in the secret scalar, while still being computationally fast with no large memory overhead. Nonetheless, the attack exploits exactly the regularity feature of the algorithm, as it creates a direct linear correlation between the binary logarithm (i.e. the bit length) of the secret scalar and the number of iterations (and thus curve operations) in the ladder.

The authors exploit this vulnerability by mounting an attack that collects several measures of the wall-clock execution time of a partial TLS handshake, using an `ECDHE_ECDSA` ciphersuite over a binary curve. The collected measures are heavily dominated by the EC scalar multiplication of the ECDSA signature generation, implemented using the Montgomery ladder, and thus can be directly correlated with the bit length of the secret scalar (the ephemeral nonce of the ECDSA signature generation algorithm). A second, offline, post-processing phase then uses this partial knowledge to recover the full secret key through a lattice attack.

The proposed countermeasure, adopted by OpenSSL, is based on conditionally padding the nonce before the actual scalar multiplication, to always work on scalars of fixed length (i.e. adding once or twice the group order to the scalar yields an equivalent scalar with the topmost bit set) which in turn fixes the number of curve operations in the ladder and the associated execution time.[9]

Timing measurement noise heavily affects the success rate of the described attacks, usually resulting in the attacks being unfeasible over a wireless link and having severely limited feasibility over a WAN connection due to both decreased accuracy and the total time of the attacks (which is generally further increased to compensate the noise by collecting more samples). However, more recent results [32] address the latter scenario, studying the statistical distribution of latency over different network environments and designing specialized filters to significantly reduce the effect of *jitter* (i.e. the random noise on the latency introduced by additional hops in the route(s) of a network connection). These filters allow attackers to measure events with higher accuracy over the Internet, with potential effects on the feasibility of remote timing attacks over WAN connections.

Timing as a side-channel is not limited to the execution time of a whole cryptographic operation, and is often a gateway to retrieve information from other resources shared between an attacker and a victim, including microarchitecture components, as in the cache-timing attacks covered below or, switching to the domain of web privacy, even virtual constructs in modern web browsers [71, 72].

Alternatively, the timing side-channel can be used to build reliable oracles, often circumventing trivial implementations of countermeasures to prevent other side-channel attacks. In 1998, Bleichenbacher [16] presented a famous adaptive chosen-ciphertext attack on SSL/TLS ciphersuites based on RSA and PKCS#1 v1.5 encryption padding, based on an oracle built on top of different error messages sent by servers in case of malformed ciphertexts during the SSL/TLS handshake. As a result of the work, subsequent specifications of the TLS protocol (starting from RFC 2246 [33] TLS 1.0, in the same year) recommend "to treat incorrectly formatted messages in a manner indistinguishable from correctly formatted RSA blocks". But when implementations fail to extend this recommendation to the execution time of handling different events and conditions, the timing side-channel can be used to build an alternative oracle, effective for remote exploitation, as presented in 2014 by Meyer et al. [56]. Their work targeted, among others, the default Java Secure Socket Extension (JSSE) and OpenSSL implementations of the SSL/TLS protocol.

2.3 Cache Timing Attacks

Cache-timing attacks are a subset of microarchitecture attacks targeting specifically the cache hierarchy. Cache-timing attacks against implementations of cryptography primitives exploit two key features: (1) the timing variation introduced by the cache hierarchy; and (2) the non-constant time execution of algorithms handling confidential data used by cryptography primitives and algorithms, e.g. key generation [8, 73], digital signatures [11, 65], encryption [12] and key exchange [39]. Typically, the ultimate goal of a cache-timing attack is to recover confidential information from an algorithm execution and this is done by correlating cache timing data to either the execution time of the algorithm in use, its internal state during execution, or the output of the algorithm. Cache-timing attacks are enabled by several cache attack techniques proposed and used successfully in the past, e.g. Evict+Time [63], Prime+Probe [64], and Flush+Reload [75]. The choice of attack technique depends on the attack scenario since each technique has its own advantages and disadvantages.

Cache Architecture. Accessing data and instructions from main memory is not an instant operation since it takes time to locate and fetch the data, thus delaying the execution of the processor. To improve the efficiency of the processor, the memory hierarchy includes memory banks called caches, located between the CPU cores and the RAM. Caches are smaller and faster compared to RAM and main memory, helping to improve the performance by exploiting spacial and temporal locality during memory access.

Modern CPUs contain multiple cache levels, usually *L1* and *L2* caches are private to a specific core and the last level cache (LLC) is shared among all the cores. Typically, the LLC is said to be *inclusive*, meaning that it contains a superset of the data in the caches below it, thus it contains both instructions and data from L1 and L2. The caches are organized into fixed size *cache lines* which are grouped in *cache sets*. The number of cache lines in a cache set is

[9] To track the issue uncovered by this work the id CVE-2011-1945 was assigned and CERT issued the vulnerability note VU#536044.

the *associativity*, i.e., a cache with W lines in each set is a *W-way set-associative* cache.

When the CPU needs to fetch data from memory, it first checks in the caches; if the data is there, a *cache hit* occurs and the load delay is short. On the other hand, when the data is not found in the caches, a *cache miss* occurs and the data must be fetched from a higher level memory, causing a longer delay. A copy of the data fetched from a higher level is cached, exploiting temporal locality. In addition, data close to the accessed data will be fetched and cached too, exploiting spatial locality. If a cache miss occurs and all the cache lines are in use, one of the cache lines is evicted, freeing space for the new data. In order to determine the cache line to evict, modern CPUs use variations of the least-recently-used (LRU) replacement policy.

Flush+Reload. Proposed by Yarom and Falkner [75], this powerful technique positively identifies accesses to specific memory lines with a high resolution, high accuracy, and high signal-to-noise ratio. Moreover, the technique relies on cache sharing between the CPU cores, typically achieved through the use of shared libraries.

A round of attack consists of three phases: (1) the attacker evicts the target memory line using the `clflush` instruction; (2) the attacker waits some time for the victim to access the memory line; (3) the attacker measures the time it takes to reload the memory line. The timing reveals whether or not the memory line was accessed by the victim during the waiting period, i.e. identifies cache hits and cache misses.

In addition to cache-timing attacks on cryptography, the Flush+Reload technique has been applied in clever ways targeting the kernel [45], web server function calls [77], user input [42, 51], covert channels [55], as well as more powerful microarchitecture attacks such as the Meltdown [52] and Spectre [48] attacks.

2.4 EM Analysis

Introduced by Kocher et al. [50], power analysis exploits the correlation between sensitive data and changing power leakages on the device. These power fluctuations are a result of transistor switching between the logic levels of CMOS circuits, and the current flow on data lines, as a result of processor activity and memory accesses.

Due to the tightly packaged components on modern devices, power analysis can be difficult to perform with limited or no access to power rails and only noisy global power consumption. As an alternative, Electromagnetic (EM) emanations—a by-product caused by the current flow on data lines and power rails—originally proposed as cryptographic side-channels by Quisquater and Samyde [66], provides a spatial dimension to perform side-channel analysis in isolation from unwanted leakage.

Various techniques exploit data dependent EM leakage such as Differential Power Analysis [50], Correlation Power Analysis [17], Template Attacks [25] and Horizontal attacks [30, 34]. Identifying data dependent EM leakage can be challenging due to additional noise and other unwanted artifacts, thus in addition to simple frequency analysis, additional leakage detection statistical tools are required such as Mutual Information Analysis [29], χ^2-test [58] and Test Vector Leakage Assessment (TVLA) [41, 67].

Originally developed by Cryptography Research, Inc. for AES [41] and later adapted for public-key cryptography [47], TVLA is a preferred choice for applying black-box leakage detection testing to identify side-channel weaknesses [28, 61]. TVLA is based on Welch's T-test [74], which computes a statistical value, i.e. confidence interval (*CI*) to accept or reject the null hypothesis. More specifically, the test validates whether two sets of samples are taken from similar data by comparing the averages of the two data sets. Formally, for two sets S_1 and S_2, the T-test computes as

$$t = \frac{\mu_1 - \mu_2}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}}$$

where μ_1, σ_1, and n_1 are the mean, standard deviation, and cardinality of S_1, respectively, and similarly for S_2. The T-test will fail at some discrete sample point if the value is greater than some threshold $C\tau$. In the context of side-channel data, usually fixed vs random test samples are compared to identify points with data dependent leakage [41].

Contemporary works demonstrate the effectiveness of EM analysis on modern PCs, embedded and mobile devices on various open source libraries such as GnuPG and OpenSSL, for attacking cryptosystems like AES [54], RSA [38], ECDH [36], and ECDSA [37]. Moreover, e.g. Goller and Sigl [40] successfully demonstrate the viability of EM attacks over varying distances from mobile devices on ECC and RSA.

Longo et al. [54] performed localized EM analysis on a modern embedded device running software based OpenSSL AES, a bit-sliced optimized implementation for SIMD NEON core, and an AES hardware engine. They applied TVLA to identify EM leakages and subsequently carry out template attacks. Genkin et al. [37] were able to filter out EM emanations from a mobile device at very low frequencies using inexpensive equipment and additional signal processing steps. Their attack successfully recovered a few bits of ECDSA nonces, targeting the OpenSSL *w*NAF implementation. With roughly 100 signatures, they then successfully mounted a lattice attack for full key recovery.

2.5 SM2 Implementation Attacks: Previous Work

Due to only recently being standardized and coupled with lack of sufficient public implementations and deployments, academic results on attacking SM2 implementations are limited in number. Nevertheless, existing results suggest that implementation attacks on ECDSA generally extend—with slight modification—to SM2DSA. A brief review follows.

Liu et al. [53] were the first to construct an SM2DSA analogue of existing lattice-based ECDSA key recovery with partially known nonces. The authors model exposure of three LSBs, and with 256-bit p and n recover a private key from 100 signatures with reasonable probability and modest computation time.

Chen et al. [27] were the first to implement an SM2DSA lattice attack with real traces. They target an SM2DSA smartcard implementation and distinguish least significant byte collisions by detecting Hamming weight with PCA-based techniques. Restricting to byte values `0x00` and `0xFF`, the authors obtain 120K signatures with power traces, filter them to 48 pairs, and iteratively construct lattice problem instances to recover a private key. Interestingly, the target is not the underlying ECC itself, but data moves by the RNG

during nonce generation. In that respect, their attack is independent of the underlying ECC arithmetic.

Building on [11, 20] that focus on the LSDs of the *w*NAF for ECDSA nonces, Zhang et al. [76] extend the analysis to SM2DSA. With their own implementation of ECC including traditional *w*NAF scalar multiplication paired with SM2DSA, they demonstrate it is possible to reliably capture the sequence of ECC doubles and adds through SPA on an Atmega128. Subsequently modeling the filtered nonces with sufficient zeros in the LSDs and constructing lattice problem instances, they recover private keys with high probability. Since they target least significant zeros in the *w*NAF expansion, their attack is largely independent of the scalar representation—for example, it immediately applies to binary, sliding window, and fixed window expansions. Their work provides even further evidence that ECDSA-type leaks are similarly detrimental to SM2DSA.

While no English version is available, the abstract of [68] suggests a CPA attack to recover the SM2PKE session key exploiting potential leakage from the SM3 compression function execution. That is, the target is not the ECC but the subsequent KDF.

3 SM2 IN OPENSSL

Refer to Section 1 for the detailed timeline of the SM2 feature within OpenSSL. With the narrow review window induced by the release milestone shift, several security (and functionality) issues were mainlined into the OpenSSL codebase. We give an overview of these issue in this section. Listing 1 includes an extract of the SM2DSA signature generation implementation and Listing 2 for SM2PKE public key decryption, as of OpenSSL 1.1.1-pre5 (beta 3).

Code review. Due to the hasty review process, the code implementing SM2 in the beta releases is evidently not in line with the quality standards of analogous components of `libcrypto`,[10] lacking test coverage, including critical bugs (e.g. double frees and wrong return values), a lack of return values checking and poor error handling. These defects are particularly evident in the integration with the `EVP_PKEY` (and `EVP_DigestSign`) API, which is the main entry point for `libssl` and internal and external applications for using the cryptographic functionality included in `libcrypto`.

SCA review. Beyond these traditional software issues, we preformed an SCA evaluation of both SM2DSA and SM2PKE in OpenSSL. This integration provides a rare opportunity to see how a straightforward implementation of an EC cryptosystem mixes with the underlying EC module for arithmetic. Our review resulted in the following observations, leveraging existing SCA results (Section 2) on the OpenSSL EC module.

(1) For SM2DSA, in Listing 1 there is no scalar padding before calling `EC_POINT_mul`, suggesting an SM2DSA analogue of CVE-2011-1945 for remote timing attacks; see Section 4 for our empirical evaluation.
(2) For SM2DSA, since there is no custom `EC_METHOD` for the SM2 curve, `EC_POINT_mul` is a wrapper to `ec_wNAF_mul`, suggesting an SM2DSA analogue for cache timing attacks

[10] The OpenSSL binaries can be roughly split in three blocks: `libcrypto`, providing the cryptographic and abstraction layer; `libssl`, providing the networking layer; `apps`, consisting in a CLI toolkit using the two libraries to perform various tasks.

```
84  k = BN_CTX_get(ctx);
85  rk = BN_CTX_get(ctx);
86  x1 = BN_CTX_get(ctx);
87  tmp = BN_CTX_get(ctx);
88
89  if (tmp == NULL)
90      goto done;
91
92  /* These values are returned and so should not be allocated out of the
    ↪  context */
93  r = BN_new();
94  s = BN_new();
95
96  if (r == NULL || s == NULL)
97      goto done;
98
99  for (;;) {
100     BN_priv_rand_range(k, order);
101
102     if (EC_POINT_mul(group, kG, k, NULL, NULL, ctx) == 0)
103         goto done;
104
105     if (EC_POINT_get_affine_coordinates_GFp(group, kG, x1, NULL, ctx) == 0)
106         goto done;
107
108     if (BN_mod_add(r, e, x1, order, ctx) == 0)
109         goto done;
110
111     /* try again if r == 0 or r+k == n */
112     if (BN_is_zero(r))
113         continue;
114
115     BN_add(rk, r, k);
116
117     if (BN_cmp(rk, order) == 0)
118         continue;
119
120     BN_add(s, dA, BN_value_one());
121     BN_mod_inverse(s, s, order, ctx);
122
123     BN_mod_mul(tmp, dA, r, order, ctx);
124     BN_sub(tmp, k, tmp);
125
126     BN_mod_mul(s, s, tmp, order, ctx);
127
128     sig = ECDSA_SIG_new();
129
130     if (sig == NULL)
131         goto done;
132
133      /* takes ownership of r and s */
134     ECDSA_SIG_set0(sig, r, s);
135     break;
136 }
```

Listing 1: Source code from `crypto/sm2/sm2_sign.c` in OpenSSL 1.1.1-pre5 for SM2DSA signature generation.

```
270 C1 = EC_POINT_new(group);
271 if (C1 == NULL)
272     goto done;
273
274 if (EC_POINT_set_affine_coordinates_GFp
275     (group, C1, sm2_ctext->C1x, sm2_ctext->C1y, ctx) == 0)
276     goto done;
277
278 if (EC_POINT_mul(group, C1, NULL, C1, EC_KEY_get0_private_key(key), ctx) ==
279     0)
280     goto done;
281
282 if (EC_POINT_get_affine_coordinates_GFp(group, C1, x2, y2, ctx) == 0)
283     goto done;
```

Listing 2: Source code from `crypto/sm2/sm2_crypt.c` in OpenSSL 1.1.1-pre5 for SM2PKE decryption.

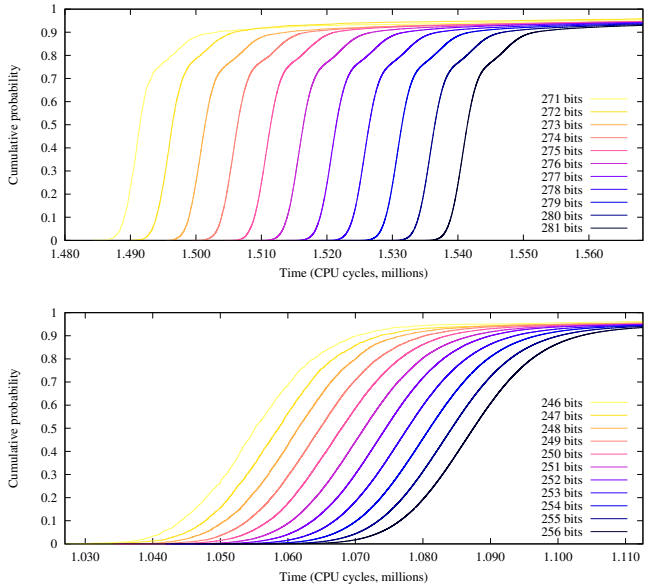

Figure 1: SM2DSA latency dependency on the nonce length on amd64 architecture in OpenSSL 1.1.1-pre3. Top: K-283 binary curve. Bottom: Recommended SM2 prime curve.

targeting scalar multiplication; see Section 5.1 for our empirical evaluation.

(3) The SM2DSA implementation uses `BN_mod_inverse` without setting `BN_FLG_CONSTTIME`, suggesting an SM2DSA analogue for cache timing attacks targeting inversion via BEEA; see Section 5.2 for our empirical evaluation.

(4) For SM2PKE, in Listing 2 there are no SCA considerations, suggesting (at least) DPA-style attacks on `EC_POINT_mul` during decryption; see Section 6 for our empirical evaluation.

The remainder of this paper is dedicated to evaluating these SCA leaks, proposing and implementing mitigations (Section 7), and empirical SCA evaluation of the mitigations (Section 7.3).

4 SM2DSA: REMOTE TIMINGS

We note the lack of scalar padding before calling `EC_POINT_mul`, suggesting an SM2DSA analogue of CVE-2011-1945. To evaluate the impact of this vulnerability, we correlate nonce lengths and the execution time of signature generations, adopting a process similar to the one presented by Brumley and Tuveri [21].

We wrote an OpenSSL client application which repeatedly generates SM2DSA signatures for a given plaintext, under the same private key. For each generated signature, the program measures the execution time of the operation (in CPU cycles) and retrieves the associated nonce by monitoring the PRNG. We repeated the experiment using both the recommended SM2 prime curve and the standardized K-283 binary (Koblitz) curve [3], as the library executes two different code paths for `EC_POINT_mul` over prime and binary curves. We then analyzed the captured data to correlate the timings with the binary logarithm (bit-length) of the nonces.

We ran these experiments on a 4-cores/4-threads Intel Core i5-6500 CPU (Skylake) running at 3.2GHz, with Enhanced Intel

```
lfence
rdtsc
lfence
mov %rax, %r10
shr $31, %rdx
mov (%rsi, %rdx), %rax
lfence
rdtsc
shl $32, %rax
or %r10, %rax
mov %rax, (%rdi)
shr $31, %rdx
clflush (%rsi, %rdx)
```

```
1:
clflush (%rdi)
clflush (%rsi)
clflush (%rdx)
clflush (%rcx)
jmp 1b
```

Listing 3: Flush+Reload (left) and performance degradation (right) implemented for our cache-timing attacks.

SpeedStep Technology and Intel Turbo Boost Technology disabled. Figure 1 shows cumulative distribution functions (CDF) for different nonce bit-lengths for the two curves, collating 4 million samples for each curve. Both plots show a strong correlation between the bit-length of the nonce and the execution time of the signature generation, which in turn is distinctly dominated by the execution time of the underlying `EC_POINT_mul` operation. For no correlation, these curves should essentially be on top of each other, i.e. indistinguishable; see Section 7.3.

Generic binary curves. The top plot of Figure 1 shows that, using a generic binary curve as the underlying elliptic curve for SM2DSA, the timing correlation appears easily exploitable to mount a remote timing attack similar to [21]. For generic binary curves, OpenSSL implements the `EC_POINT_mul` operation through a Montgomery ladder algorithm, which due to its extreme regularity in the sequence of EC additions and doublings, results in an overall execution time directly proportional to the binary logarithm of the secret `EC_POINT_mul` scalar (i.e. the SM2DSA nonce). As a result, each nonce bit-length exhibits a clearly distinct CDF, and suggests simple thresholding on the execution time to filter signatures associated with a specific nonce length with high probability.

Recommended SM2 curve. When using the recommended SM2 prime curve, OpenSSL 1.1.1-pre3 implements the `EC_POINT_mul` operation using the generic prime curve codepath, using a *w*NAF algorithm (see Section 5.1). The bottom plot of Figure 1 shows that, similarly to the previous case, there is a strong correlation between the execution time of SM2DSA and the associated nonce length. We note that in this case, mounting a practical attack poses more challenges due to a less distinct separation between the different CDFs, likely compensated by collecting more samples.

5 SM2DSA: CACHE TIMINGS

As mentioned in Section 2.5, several previous works show SM2DSA vulnerable to ECDSA-type SCA attacks. For that reason, we explore and analyze the cryptosystem applying existing cache-timing attack techniques to code paths known for leaking information, and exploited successfully in the past for ECDSA [11, 65].

For our analysis, we use the Flush+Reload technique [75] paired with a performance degradation attack [9, 65]. Listing 3 shows code snippets used to implement both techniques. This combination of techniques allows us to accurately probe relevant memory addresses with enough granularity to confirm bit leakage on both scalar multiplication and modular inversion operations.

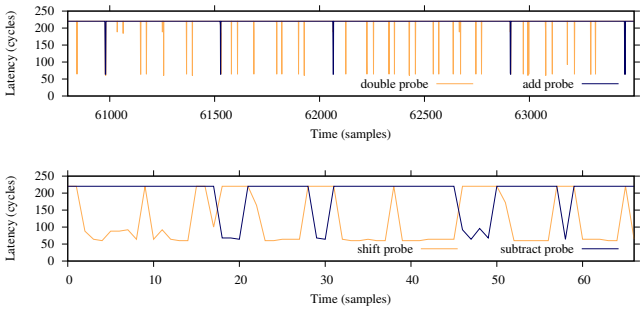

Figure 2: Partial raw cache-timing traces during SM2DSA. Top: Scalar multiplication. Bottom: Binary GCD modular inversion. Both traces reveal partial information on the secret scalar and the long-term private key, respectively.

5.1 Scalar Multiplication

SM2DSA in OpenSSL performs scalar multiplication operations by calling the `EC_POINT_mul` function in `SM2_sig_gen` @ `crypto/sm2/sm2_sign.c`, which is only a wrapper to the underlying `ec_wNAF_mul` function. The `ec_wNAF_mul` function is a generic code path performing scalar multiplication, i.e. $[k]G$ in SM2DSA, by executing a series of double and add operations based on the wNAF representation of k. This code path is vulnerable to cache-timing attacks due to its non constant-time execution, targeted previously using cache-timing techniques [9, 20, 70, 75]. Generally, the strategy is to trace the sequence of double and add operations, which leaks LSDs of k, leading to private key recovery.

Unlike previous attacks, during our analysis we do not probe memory lines directly used in functions `EC_POINT_add` and `EC_POINT_dbl`, but instead we focus in low level functions `BN_rshift1` and `BN_lshift`. The `BN_rshift1` function is one of several functions called during `EC_POINT_add` execution and, unlike the rest of the functions in the routine, `BN_rshift1` is a representative of the add operation. Similarly, `BN_lshift` is a representative of the double operation, allowing to identify add and double operations respectively during scalar multiplication. Therefore, these low level functions allow accurately detecting when add and double operations execute. By tracing the sequence of `BN_rshift1` and `BN_lshift` operations, we are able to determine with high accuracy the sequence of double and add operations, leaking LSDs of k. Top trace in Figure 2 shows a post-filtered cache-timing trace of a scalar multiplication with a random nonce k during SM2DSA. The probes detect the sequence of curve operations from left to right as follows: 1 double, 1 add, 4 doubles, 1 add, 4 doubles, 1 add, 7 doubles, 1 add, 4 doubles, and 1 add; thus revealing partial information on k.

5.2 Modular Inversion

Modular inversion is a common operation during digital signatures and in OpenSSL, SM2DSA uses the `BN_mod_inverse` function for this purpose. This function executes one of several GCD algorithm variants. Unfortunately, most of these variants are based on the Euclidean algorithm which executes in a non constant-time fashion. The Euclidean algorithm and variants are highly dependent on their inputs and previous research exploits some of these variants [4, 8, 65].

During SM2DSA execution, none of the input values has the flag `BN_FLG_CONSTTIME` set when entering to the `BN_mod_inverse` function, therefore the function takes the default insecure path, calculating the modular inverse of $d_A + 1$ through the Binary Extended Euclidean Algorithm (BEEA). More importantly, this operation executes every time a signature is generated with the exact same input values, therefore an attacker has several opportunities to trace the BEEA execution on the private key.

Similar to the scalar multiplication case, we identify the low level operations leaking bits from the input values. In the BEEA case, this means functions `BN_rshift1` and `BN_sub`. By placing probes in memory lines in these routines, we are able to trace the sequence of shift and subtraction operations performed during modular inversion, leading to partial bit recovery of $d_A + 1$. Using algebraic methods [7], it is possible to recover a variable amount of private key LSDs from these sequences. Bottom trace in Figure 2 shows the end of a post-filtered cache-timing trace capturing the execution of the BEEA during SM2DSA. The trace matches the sequence 2 shifts, 1 subtract, 1 shift, 1 subtract, 2 shifts, 1 subtract, 1 shift, 1 subtract, 1 shift; obtained from a perfect trace computed by executing BEEA on the inputs taken from the SM2DSA signature test, demonstrating private key leakage.

6 SM2PKE: EM ANALYSIS

As discussed in Section 2.1, SM2PKE decryption computes the shared secret using the receiver's private key d_B and the sender's ECDHE public key C_1. The point multiplication $[d_B]C_1$ can leak intermediate values which can be exploited using both vertical attacks [31, 35, 62] and horizontal attacks [10, 34] for key recovery.

To evaluate the side-channel leakage for SM2PKE, we applied Test Vector Leakage Analysis (TVLA) using Welch's T-test [41, 67]. We took an approach similar to [28, 61] for ECC, with a reduced set of test vectors. We divided the test vectors into three different sets $\{S_i\}$ for $i = 1, 2, 3$. The sets S_1, S_2 and S_3 contained traces for fixed key d_B and fixed cipher text C_1, fixed d_B and varying C_1, fixed C_1 and varying d_B respectively. We performed the tests in pairs, such that $\{(S_1, S_2), (S_1, S_3)\}$ would fail the T-test if the resulting confidence threshold satisfies $|C\tau| > 4.5$. We selected the value $C\tau$ (a function of number of samples) based upon empirical evidence from Jaffe et al. [47].

Experimental setup. We performed the experiments on an AM335x Sitara SoC[11] featuring a 32-bit ARM Cortex-A8 embedded on a BeagleBone Black[12] development board. We used the standard BeagleBone Debian distribution ("Wheezy" 7.8) while keeping all the default configurations intact. For capturing the EM traces, we used a Langer LF-U5 near-field probe (500kHz to 50MHz) and 30dB Langer PA-303 low noise amplifier. We positioned the probe head directly on the SoC, seeking to strengthen the acquisition quality. We procured the traces using a PicoScope 5244B digital oscilloscope at a sampling rate of 125 MSamples/sec with a 12-bit ADC resolution. Figure 3 shows our setup for the EM analysis.

[11] http://www.ti.com/processors/sitara/arm-cortex-a8/am335x/overview.html
[12] https://beagleboard.org/black

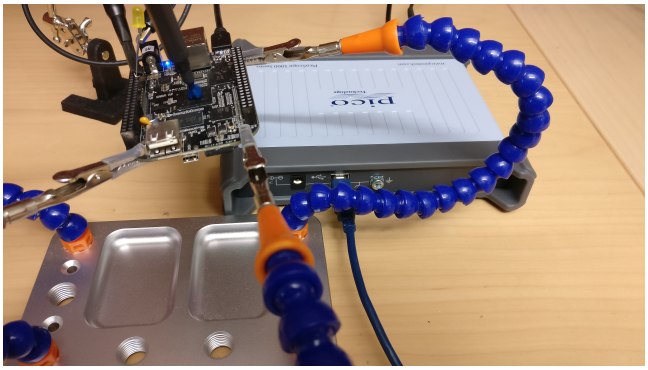

Figure 3: Capturing EM traces from the BeagleBone Black using a Langer probe positioned on the SoC and procured using the Picoscope USB oscilloscope.

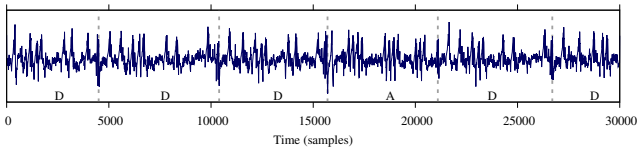

Figure 4: The filtered EM trace clearly reveals the sequence of ECC double and add operations during SM2PKE decryption.

EM acquisition. For the purpose of this analysis, we captured 1500 EM traces for each set (S_1, S_2, S_3) while performing the decryption operation. We fixed the clock frequency at 1GHz to avoid any bias in the captured traces. To acquire traces, we initially utilized the GPIO pin of the board to trigger the oscilloscope. However, this trigger proved unreliable as it encountered random delays. To improve this, we applied correlation based matching to locate the beginning of the trace. As most of the EM signal energy was concentrated at much lower frequencies, we also applied a Low Pass filter with a cut-off frequency at 15MHz.

Due to noise in the traces, we performed additional processing steps. For the envelope detection, we applied a Digital Hilbert Transform, followed by a Low Pass Filter to smooth out any high frequency noise. From the sets, we dropped traces containing noise due to preemptive interrupts and other unwanted signal features. In the end, we retained a total of 1000 traces per set. Since the T-test required averaging multiple traces, we aligned the traces at each point of interest (i.e. ECC operations). Figure 4 shows part of an actual processed EM trace, depicting a sequence of ECC double and add operations.

T-test. To validate the results, we divided each set into subsets $\{S_{ia}\}$ and $\{S_{ib}\}$ and performed an independent T-test between sets $\{(S_{1a}, S_{ka})\}$ and $\{(S_{1b}, S_{kb})\}$ for $k = 2, 3$. We performed a further test by combining an equal number of randomly selected traces from both $\{S_1\}$ and $\{S_k\}$ such that the two resulting subsets were disjoint. A correct T-test for the random sets $R_1 = \{(S_1 \cup S_k)\}$ and $R_k = \{(S_1 \cup S_k) - R1\}$ should result in confidence threshold $|C\tau| < 4.5$ for all the points in the traces.

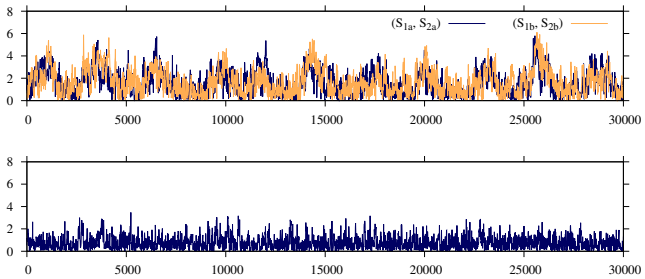

Figure 5: TVLA during SM2PKE decryption. Top: T-test results between sets S_1 and S_2 versus sample index; for fixed vs random k the test fails since many peaks exceed the 4.5 threshold for both sets. Bottom: T-test results between random sets R_1 and R_2 versus sample index; it shows no peaks exceeding 4.5 since the means are similar due to balanced and random selection of fixed and random k in both sets.

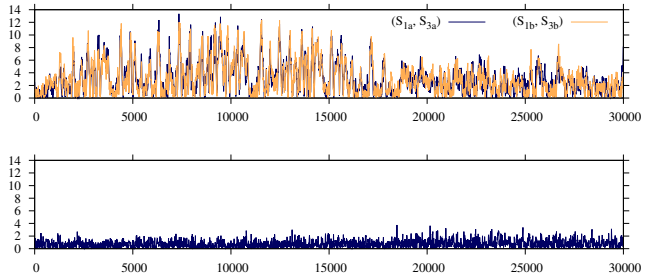

Figure 6: TVLA during SM2PKE decryption. Top: T-test results between sets S_1 and S_3 versus sample index; for fixed vs random C the test fails since many peaks exceed the 4.5 threshold for both sets. Bottom: T-test results between random sets R_1 and R_3 versus sample index; shows no peaks exceeding 4.5 since the means are similar due to equal and random selection of fixed and random C in both sets.

The experiments showed multiple points where the T-test failed for both $\{(S_1, S_2), (S_1, S_3)\}$. Figure 5 shows two T-test results for $\{(S_{1a}, S_{2a})\}$ and $\{(S_{1b}, S_{2b})\}$. It is clear from the figure that the T-test values have a significant number of peaks satisfying $|C\tau| > 4.5$ for both tests, roughly at the same points. This demonstrates there is a strong leak at these points, since we performed both tests on different sets of traces. From the random sets $\{R_1, R_2\}$ the confidence threshold remains $|C\tau| < 4.5$ which further validates our hypothesis. Similarly, Figure 6 shows the failed T-test for $\{(S_{1a}, S_{3a})\}$ and $\{(S_{1b}, S_{3b})\}$.

7 SCA MITIGATIONS

The attacker effort required to achieve full key recovery using the previously described leaks is very low. Taking the cache-timing leak in Section 5.1 as an example, SM2DSA lattice attacks discussed in Section 2.5 analogous to ECDSA would require roughly a mere 500 signatures with traces, immediately discarding roughly 75% of those that statistically will not reveal enough information about the LSDs (i.e. three or more bits are needed in practice for lattice attacks). In

this section, we describe our results on mitigating the discovered leaks. We claim no novelty for the mitigations themselves, only their application and implementation within the OpenSSL library; they are standard techniques known since at least the 90s.

We stress that the focus of our mitigation effort is not on SM2 nor any individual cryptosystem, but rather on the EC module itself, to provide transparent secure-by-default behavior to cryptosystems at the architecture level. That is, conceptually it should be completely reasonable to drop in a cryptosystem implementation like it was done with SM2DSA or SM2PKE and have it resist SCA, with absolutely no esoteric knowledge of OpenSSL internals that control SCA features such as constant-time flags.

7.1 Scalar Multiplication: SCA Mitigations

Ladder. While it is indeed feasible to reduce leakage in OpenSSL's *w*NAF scalar multiplication code path [18], tediously straightlining conditions and making table lookups regular adds significant code complexity, increases the probability of defects, and generally results in low maintainability code. Even then, there is no guarantee that all leakage issues are addressed: the code path was not initially intended to resist SCA, and retrofitting mitigations becomes awkward.

We instead implemented an early exit from `ec_wNAF_mul` that—irrespective of the constant time flag—diverts to a new single scalar multiplication function for all instances of $[k]G$ (fixed point, e.g. ECC key generation, SM2DSA signing, ECDSA signing, first half of ECDH) or $[k]P$ (variable point, e.g. SM2PKE decryption, last half of ECDH), and falls back to the existing (insecure) *w*NAF code in all other cases (e.g. $[a]G + [b]P$ in various digital signature scheme verifications). For cryptosystem use cases internal to the OpenSSL library, this provides secure-by-default scalar multiplication code path traversal.

For this new functionality, we chose the traditional powering ladder due to Montgomery [57], heralded for its favorable SCA properties. In modern implementations, straightlining the key-dependent ladder branches happens in one of two ways [60, Sec. 2]: "either by loading from (or storing to) addresses that depend on the secret scalar, or by using arithmetic operations to perform a conditional register-to-register move. The latter approach is very common on large processors with cache, where the former approach leaks through cache-timing information."

We see both in practice: For example, TomsFastMath[13] does not branch but reads and stores using (secret) pointer offsets, while Mbed TLS[14] parses all the data and performs a manual conditional swap with arithmetic, even documenting their function `mbedtls_mpi_safe_cond_swap` with the comment: "Here it is not OK to simply swap the pointers, which would lead to different memory access patterns when X and Y are used afterwards." This is in contrast to e.g. [44, Sec. 8.5]: "we implement the conditional swap operation after each ladder step by swapping pointer variables instead of data. We expect slightly better performance and also a reduced side-channel leakage." While that is perhaps a valid strategy on architectures lacking cache memory, we feel it is generally dubious advice since typical engineers are usually unaware of SCA subtleties.

Regardless, the "standard way" according to Bernstein [14, Sec. 3] uses arithmetic to implement conditional swaps on the data, not the pointers; the work also reviews a slight optimization, which we also implement. The two contiguous swaps conditional on bits k_i and k_{i-1} reduce to a single swap by XOR-merging the condition bits, i.e. only swap if the bit values differ. This optimization halves the number of conditional swaps.

Scalar padding. The above conditional swaps ensure favorable SCA behavior for ladder iterations. But [21] exploits the number of said iterations, fixed in an ECDSA-only fashion in 2011 by padding nonces. We remove this padding, and instead push it to the underlying EC module to ensure a constant number of ladder iterations. To accomplish this in an SCA-friendly way, we construct two values $k' = k+n$ and $k'' = k'+n$, subsequently using the above conditional swap to set k to either k' or k'', whichever has bit-length precisely one more than n. We apply this padding directly preceding ladder execution.

Coordinate blinding. Originally proposed by Coron [31, Sec. 5.3] for standard projective coordinates as a DPA countermeasure, coordinate blinding transforms the input point to a random representative of the equivalence class. For generic curves over $GF(p)$, OpenSSL's formulae are a fairly verbatim implementation of Jacobian projective coordinates [1, A.9.6] where the relation

$$(X, Y, Z) \equiv (\lambda^2 X, \lambda^3 Y, \lambda Z)$$

holds for all $\lambda \neq 0$ in $GF(p)$. Our implemented mitigation generates λ randomly, applying the map a single time directly preceding the ladder execution. This is, for example, the approach taken by Mbed TLS (function `ecp_randomize_jac`).

7.2 Modular Inversion: SCA Mitigations

Directly due to the work by Gueron and Krasnov [43, Sec. 6], OpenSSL integrated a contribution from Intel that included (1) high-speed, constant-time P-256 ECC on AVX2 architectures; (2) constant-time modular inversion modulo ECDSA group orders. It did the latter by internally exposing a function pointer within the `EC_METHOD` structure. If set, ECDSA signing code path calls said pointer (for which the custom P-256 method has a dedicated function), otherwise a series of default fallbacks including (1) FLT inversion with Montgomery modular exponentiation; (2) normal EEA-based inversion. We refactored the structure to expose this default behavior within the wrapper that checks the function pointer, the end goal being to expose it to the EC module as a whole and not limit to ECDSA, in turn allowing SM2DSA access to a strictly secure-by-default functionality. We explored two different options for inversion default behavior that resist SCA, summarized below.

Blinding. The classical way to compute modular inversions is through the EEA utilizing divisions, or binary variants utilizing shifts and subtracts. However, as previously described their control flow can leak critical algorithm state. Nevertheless, to prevent direct input deduction from this state one option is to choose blinding value b uniformly at random from $[1\,..\,n)$ then compute $k^{-1} = b(bk)^{-1}$ at

[13] https://github.com/libtom/tomsfastmath/
[14] https://github.com/ARMmbed/mbedtls/

the additional cost of two multiplications. This is, for example, the approach taken by Mbed TLS for ECDSA nonces.

Exponentiation. Although initially motivated by binary fields with normal basis representation where squaring is a simple bit rotation, the algorithm by Itoh and Tsujii [46] is one of the earliest examples of favorable implementation aspects of using FLT for finite field inversion. SCA benefits followed thereafter, e.g. Curve25519 where Bernstein [13, Sec. 5] weighs blinded EEA methods versus FLT: "An extended-Euclid inversion, randomized to protect against timing attacks, might be faster, but the maximum potential speedup is very small, while the cost in code complexity is large."

Performance and security. Regarding security, it is clear that either method is a leap forward for OpenSSL with respect to secure-by-default. We feel that blinding has an intrinsic advantage over FLT-based methods, since the former resists bug attacks [15, 19] that exploit predictable execution flows. Regarding performance, we benchmarked both approaches to measure the potential differences alluded to by Bernstein, and found the results consistent. On an Intel Core i5-6500 CPU (Skylake) running at 3.2GHz, after all of our described and implemented countermeasures, one SM2DSA execution takes on average 1760913 cycles with FLT, and 1750984 cycles with blinded BEEA—a difference of a fraction of a percent.

In the end, the OpenSSL team declined our blinding contribution. They plan to increase the usage of the Montgomery arithmetic context within the EC module, so in that sense their decision is rational from a software architecture perspective. The team instead integrated our FLT refactoring, sufficient to thwart the attack in Section 5, and furthermore provide secure-by-default behavior to future callers conforming to the convention set by this API.

7.3 SCA Mitigations: Evaluation

Remote timings: evaluation. Using the same approach adopted in Section 4, Figure 7 shows the cumulative effect of three countermeasures: adopting the Montgomery ladder instead of the wNAF algorithm for regular scalar multiplication, scalar padding, and computation of modular inversion via exponentiation through FLT.

Both plots clearly show that the latencies measured for signature generation using nonces of different bit-lengths are indistinguishable, effectively preventing the attack, and a comparison with Figure 1 immediately shows the extent of the leakage reduction.

Cache-timings: evaluation. After introducing the mitigations, when SM2DSA performs a scalar multiplication it first calls the `EC_POINT_mul` function, a wrapper to `ec_wNAF_mul`. There the code takes an early exit, jumping to the powering ladder regular algorithm to perform a fixed point scalar multiplication $[k]G$. From the cache perspective, the ladder implementation consists of an always-double-and-add algorithm, largely unrelated to the wNAF representation of the nonce k. To support our claim, we follow the same approach as in Section 5, placing probes in the same underlying functions `BN_rshift1` and `BN_lshift`—called by `EC_POINT_add` and `EC_POINT_dbl`—to trace the sequence of operations during scalar multiplication. Top trace in Figure 8 shows an example trace, which indeed tracks the sequence of double and add operations successfully, but due to the regular nature of the powering ladder

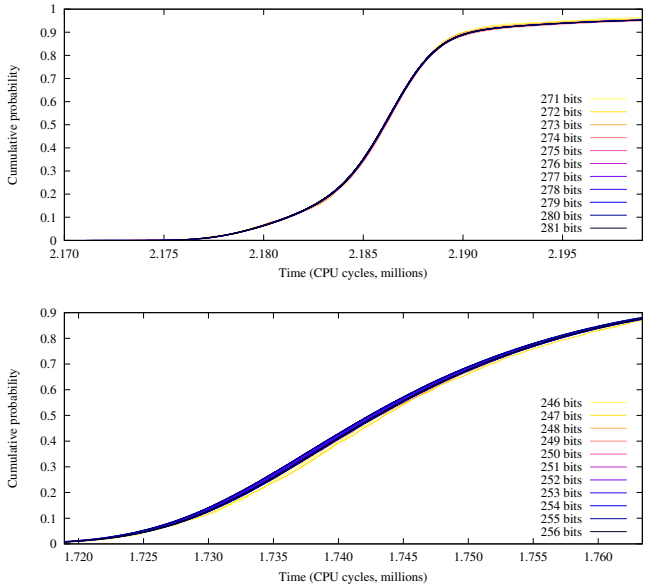

Figure 7: SM2DSA latency dependency on the nonce length on amd64 architecture, using (1) a Montgomery ladder algorithm for scalar multiplication instead of wNAF; (2) scalar padding; (3) modular inversion through exponentiation (FLT). Top: K-283 binary curve. Bottom: Recommended SM2 curve.

algorithm, no meaningful information can be retrieved from this sequence.

During modular inversion, the high level function `SM2_sig_gen` in SM2DSA no longer calls `BN_mod_inverse` but instead it calls directly `EC_GROUP_do_inverse_ord` on the private key $d_A + 1$. This function computes modular inversion by performing an exponentiation using FLT, therefore the underlying algorithm and its implementation are completely different compared to the Euclidean algorithm (and variants) used previously. Recall that during modular inversion using FLT, the exponent value is public; said value does not require SCA protection. Bottom trace in Figure 8 shows an example trace during modular inversion, probing the square and multiply operations based on the public exponent.

EM leakage: evaluation. To validate the efficacy of the applied mitigations, we repeated the T-test experiments (Section 6). Figure 10 shows the results of the new T-test for both fixed vs random key (S_1, S_2) and fixed vs random point (S_1, S_3). Figure 9 shows the EM traces, reflecting a regular sequence of ECC double and add operations due to ladder point multiplication. One interesting observation is the increase in the number of peaks for the add operation compared to Figure 4. This is due to the fact that `ec_wNAF_mul` uses mixed coordinates (projective and affine), a code path with less field operations compared to the fully projective coordinate path taken by the ladder.

It is clear from Figure 10 that the T-test shows a significant improvement due to the combined ladder application and projective coordinate randomization. The T-test easily passed for fixed vs random point (S_1, S_3) with $|C\tau| < 4$. In case of fixed vs random key

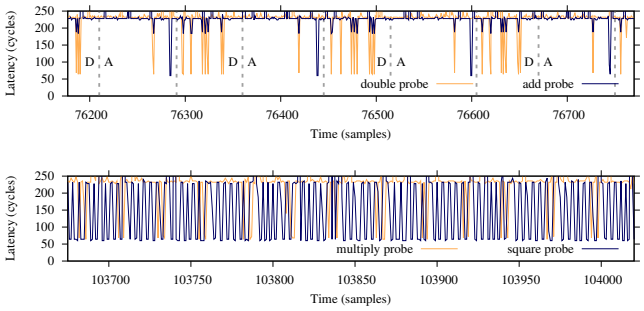

Figure 8: Partial raw cache-timing traces during SM2DSA. Top: Ladder scalar multiplication composed of regular double and add operations. Bottom: FLT modular inversion via exponentiation composed of regular squaring windows followed by a single multiply.

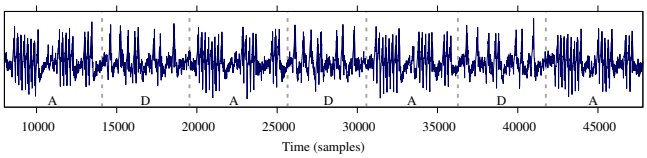

Figure 9: The filtered EM trace after applying the ladder countermeasure. As expected, it clearly reveals the sequence of ECC double and add operations during SM2PKE decryption, yet this sequence is regular and not useful for SCA-enabled attackers.

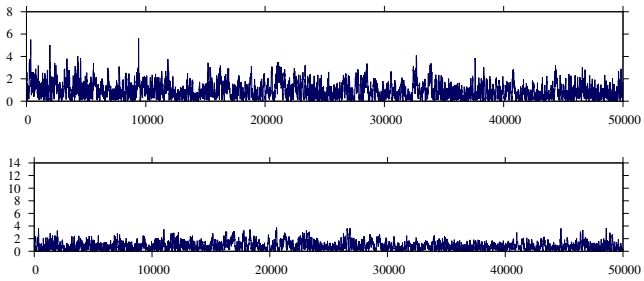

Figure 10: Top: T-test results between sets S_1 and S_2 versus sample index; for fixed vs random k the test marginally fails with leaks at the few points where the threshold is around 6. Bottom: T-test results between sets S_1 and S_3 versus sample index; for fixed vs random C the test passes since no peaks exceed the 4.5 threshold.

(S_1, S_2) we still observe a marginal number of peaks with magnitude roughly 6. In theory, it is still possible to exploit this; e.g. a key value that leading to special intermediate points on the curve such as zero-value [6] or same value points [59]. However, the leakage is so minimal, our analysis suggests mounting such attacks would be extremely difficult and feature significant data complexity. Moreover, the scalar randomization countermeasure [31] to thwart this leak introduces performance overhead, in this case unacceptable to OpenSSL when weighed vs risk.

8 CONCLUSION

Subsequent to an accelerated OpenSSL milestone to support SM2 cryptosystems, our work began with a security review of SM2DSA and SM2PKE implementations within OpenSSL pre-releases. Part of our review uncovered several side-channel deficiencies in the merged code, which we then verified with empirical remote timing, cache-timing, and EM traces. To mitigate these discovered vulnerabilities, we proposed and implemented several mitigations, now mainlined into the OpenSSL codebase. These mitigations target the underlying EC module, providing secure-by-default behavior not only for SM2 but future cryptosystems in the ECC family. Notably, the mitigations also bring security to the generic curve scalar multiplication code path in OpenSSL, a longstanding vulnerability since 2009. Finally, we performed an empirical SCA evaluation of these mitigations to demonstrate their efficacy.

We met our goal to intersect the recent OpenSSL 1.1.1 release and ensure these vulnerabilities do not affect release versions. However, given a more relaxed schedule, we outline future work to improve this secure-by-default approach: (1) the antiquated ECC point addition and doubling formulae should be renovated to more recent exception and/or branch-free versions; (2) support for ladder step function pointers, for more efficient ladder operations w.r.t. finite field operations; (3) at the standardization level, SM2DSA private key formats that, similar to RSA private keys with CRT parameters, store the value $(d_A + 1)^{-1}$ alongside the private key d_A for accelerated performance and a reduced SCA attack surface.

From the software engineering perspective, lessons learned from our work are twofold: (1) software projects, OpenSSL included, should maintain a stronger separation between release, beta, and development branches to inhibit "feeping creaturism" [69, Ch. 6] that can adversely shift milestones; (2) milestones for security-critical features should be set consistent with the complexity of the review process, to prevent premature merging. Luckily, in this case our responsible disclosure with the OpenSSL security team coupled with our mitigation development efforts yielded a favorable outcome. We strongly encourage adhering to the above two points to assist averting future security vulnerabilities.

ACKNOWLEDGMENTS

Supported in part by Academy of Finland grant 303814.

The third author was supported in part by a Nokia Foundation Scholarship and by the Pekka Ahonen Fund through the Industrial Research Fund of Tampere University of Technology.

This article is based in part upon work from COST Action IC1403 CRYPTACUS, supported by COST (European Cooperation in Science and Technology).

REFERENCES

[1] 1999. *Standard Specifications for Public Key Cryptography.* IEEE P1363/D13. Institute of Electrical and Electronics Engineers.

[2] 2009. *Elliptic Curve Cryptography.* SEC 1. Standards for Efficient Cryptography Group. http://www.secg.org/sec1-v2.pdf

[3] 2013. *Digital Signature Standard (DSS).* FIPS PUB 186-4. National Institute of Standards and Technology. https://doi.org/10.6028/NIST.FIPS.186-4

[4] Onur Acıiçmez, Shay Gueron, and Jean-Pierre Seifert. 2007. New Branch Prediction Vulnerabilities in OpenSSL and Necessary Software Countermeasures. In *Cryptography and Coding, 11th IMA International Conference, Cirencester, UK, December 18-20, 2007, Proceedings (Lecture Notes in Computer Science),*

Steven D. Galbraith (Ed.), Vol. 4887. Springer, 185–203. https://doi.org/10.1007/978-3-540-77272-9_12

[5] Onur Acıiçmez, Werner Schindler, and Çetin Kaya Koç. 2005. Improving Brumley and Boneh timing attack on unprotected SSL implementations. In *Proceedings of the 12th ACM Conference on Computer and Communications Security, CCS 2005, Alexandria, VA, USA, November 7-11, 2005*, Vijay Atluri, Catherine A. Meadows, and Ari Juels (Eds.). ACM, 139–146. https://doi.org/10.1145/1102120.1102140

[6] Toru Akishita and Tsuyoshi Takagi. 2005. Zero-Value Register Attack on Elliptic Curve Cryptosystem. *IEICE Transactions* 88-A, 1 (2005), 132–139. https://doi.org/10.1093/ietfec/e88-a.1.132

[7] Alejandro Cabrera Aldaya, Alejandro J. Cabrera Sarmiento, and Santiago Sánchez-Solano. 2017. SPA vulnerabilities of the binary extended Euclidean algorithm. *J. Cryptographic Engineering* 7, 4 (2017), 273–285. https://doi.org/10.1007/s13389-016-0135-4

[8] Alejandro Cabrera Aldaya, Cesar Pereida García, Luis Manuel Alvarez Tapia, and Billy Bob Brumley. 2018. Cache-Timing Attacks on RSA Key Generation. *IACR Cryptology ePrint Archive* 2018, 367 (2018). https://eprint.iacr.org/2018/367

[9] Thomas Allan, Billy Bob Brumley, Katrina E. Falkner, Joop van de Pol, and Yuval Yarom. 2016. Amplifying side channels through performance degradation. In *Proceedings of the 32nd Annual Conference on Computer Security Applications, ACSAC 2016, Los Angeles, CA, USA, December 5-9, 2016*, Stephen Schwab, William K. Robertson, and Davide Balzarotti (Eds.). ACM, 422–435. https://doi.org/10.1145/2991079.2991084

[10] Aurélie Bauer, Éliane Jaulmes, Emmanuel Prouff, Jean-René Reinhard, and Justine Wild. 2015. Horizontal collision correlation attack on elliptic curves – Extended Version. *Cryptography and Communications* 7, 1 (2015), 91–119. https://doi.org/10.1007/s12095-014-0111-8

[11] Naomi Benger, Joop van de Pol, Nigel P. Smart, and Yuval Yarom. 2014. "Ooh Aah... Just a Little Bit" : A Small Amount of Side Channel Can Go a Long Way. In *Cryptographic Hardware and Embedded Systems - CHES 2014 - 16th International Workshop, Busan, South Korea, September 23-26, 2014. Proceedings (Lecture Notes in Computer Science)*, Lejla Batina and Matthew Robshaw (Eds.), Vol. 8731. Springer, 75–92. https://doi.org/10.1007/978-3-662-44709-3_5

[12] Daniel J. Bernstein. 2005. Cache-timing attacks on AES. http://cr.yp.to/papers.html#cachetiming

[13] Daniel J. Bernstein. 2006. Curve25519: New Diffie-Hellman Speed Records. In *Public Key Cryptography - PKC 2006, 9th International Conference on Theory and Practice of Public-Key Cryptography, New York, NY, USA, April 24-26, 2006, Proceedings (Lecture Notes in Computer Science)*, Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin (Eds.), Vol. 3958. Springer, 207–228. https://doi.org/10.1007/11745853_14

[14] Daniel J. Bernstein. 2009. Batch Binary Edwards. In *Advances in Cryptology - CRYPTO 2009, 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings (Lecture Notes in Computer Science)*, Shai Halevi (Ed.), Vol. 5677. Springer, 317–336. https://doi.org/10.1007/978-3-642-03356-8_19

[15] Eli Biham, Yaniv Carmeli, and Adi Shamir. 2016. Bug Attacks. *J. Cryptology* 29, 4 (2016), 775–805. https://doi.org/10.1007/s00145-015-9209-1

[16] Daniel Bleichenbacher. 1998. Chosen Ciphertext Attacks Against Protocols Based on the RSA Encryption Standard PKCS #1. In *Advances in Cryptology - CRYPTO '98, 18th Annual International Cryptology Conference, Santa Barbara, California, USA, August 23-27, 1998, Proceedings (Lecture Notes in Computer Science)*, Hugo Krawczyk (Ed.), Vol. 1462. Springer, 1–12. https://doi.org/10.1007/BFb0055716

[17] Eric Brier, Christophe Clavier, and Francis Olivier. 2004. Correlation Power Analysis with a Leakage Model. In *Cryptographic Hardware and Embedded Systems - CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings (Lecture Notes in Computer Science)*, Marc Joye and Jean-Jacques Quisquater (Eds.), Vol. 3156. Springer, 16–29. https://doi.org/10.1007/978-3-540-28632-5_2

[18] Billy Bob Brumley. 2015. Faster Software for Fast Endomorphisms. In *Constructive Side-Channel Analysis and Secure Design - 6th International Workshop, COSADE 2015, Berlin, Germany, April 13-14, 2015. Revised Selected Papers (Lecture Notes in Computer Science)*, Stefan Mangard and Axel Y. Poschmann (Eds.), Vol. 9064. Springer, 127–140. https://doi.org/10.1007/978-3-319-21476-4_9

[19] Billy Bob Brumley, Manuel Barbosa, Dan Page, and Frederik Vercauteren. 2012. Practical Realisation and Elimination of an ECC-Related Software Bug Attack. In *Topics in Cryptology - CT-RSA 2012 - The Cryptographers' Track at the RSA Conference 2012, San Francisco, CA, USA, February 27 - March 2, 2012. Proceedings (Lecture Notes in Computer Science)*, Orr Dunkelman (Ed.), Vol. 7178. Springer, 171–186. https://doi.org/10.1007/978-3-642-27954-6_11

[20] Billy Bob Brumley and Risto M. Hakala. 2009. Cache-Timing Template Attacks. In *Advances in Cryptology - ASIACRYPT 2009, 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009. Proceedings (Lecture Notes in Computer Science)*, Mitsuru Matsui (Ed.), Vol. 5912. Springer, 667–684. https://doi.org/10.1007/978-3-642-10366-7_39

[21] Billy Bob Brumley and Nicola Tuveri. 2011. Remote Timing Attacks Are Still Practical. In *Computer Security - ESORICS 2011 - 16th European Symposium on Research in Computer Security, Leuven, Belgium, September 12-14, 2011. Proceedings (Lecture Notes in Computer Science)*, Vijay Atluri and Claudia Díaz (Eds.), Vol. 6879. Springer, 355–371. https://doi.org/10.1007/978-3-642-23822-2_20

[22] David Brumley and Dan Boneh. 2003. Remote Timing Attacks Are Practical. In *Proceedings of the 12th USENIX Security Symposium, Washington, D.C., USA, August 4-8, 2003*. USENIX Association. https://www.usenix.org/conference/12th-usenix-security-symposium/remote-timing-attacks-are-practical

[23] David Brumley and Dan Boneh. 2005. Remote timing attacks are practical. *Computer Networks* 48, 5 (2005), 701–716. https://doi.org/10.1016/j.comnet.2005.01.010

[24] Certicom Research. 2010. *Standards for Efficient Cryptography 2 (SEC 2): Recommended Elliptic Curve Domain Parameters (Version 2.0).* Technical Report. Certicom Corp. http://www.secg.org/sec2-v2.pdf

[25] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. 2002. Template Attacks. In *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers (Lecture Notes in Computer Science)*, Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar (Eds.), Vol. 2523. Springer, 13–28. https://doi.org/10.1007/3-540-36400-5_3

[26] Cai-Sen Chen, Tao Wang, and Jun-Jian Tian. 2013. Improving timing attack on RSA-CRT via error detection and correction strategy. *Information Sciences* 232 (2013), 464–474. https://doi.org/10.1016/j.ins.2012.01.027

[27] Jiazhe Chen, Mingjie Liu, Hexin Li, and Hongsong Shi. 2015. Mind Your Nonces Moving: Template-Based Partially-Sharing Nonces Attack on SM2 Digital Signature Algorithm. In *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security, ASIA CCS '15, Singapore, April 14-17, 2015*, Feng Bao, Steven Miller, Jianying Zhou, and Gail-Joon Ahn (Eds.). ACM, 609–614. https://doi.org/10.1145/2714576.2714587

[28] Łukasz Chmielewski, Pedro Massolino, Jo Vliegen, Lejla Batina, and Nele Mentens. 2017. Completing the Complete ECC Formulae with Countermeasures. *Journal of Low Power Electronics and Applications* 7, 1 (2017), 3. https://doi.org/10.3390/jlpea7010003

[29] Tom Chothia and Apratim Guha. 2011. A Statistical Test for Information Leaks Using Continuous Mutual Information. In *Proceedings of the 24th IEEE Computer Security Foundations Symposium, CSF 2011, Cernay-la-Ville, France, 27-29 June, 2011*. IEEE Computer Society, 177–190. https://doi.org/10.1109/CSF.2011.19

[30] Christophe Clavier, Benoit Feix, Georges Gagnerot, Mylène Roussellet, and Vincent Verneuil. 2010. Horizontal Correlation Analysis on Exponentiation. In *Information and Communications Security - 12th International Conference, ICICS 2010, Barcelona, Spain, December 15-17, 2010. Proceedings (Lecture Notes in Computer Science)*, Miguel Soriano, Sihan Qing, and Javier López (Eds.), Vol. 6476. Springer, 46–61. https://doi.org/10.1007/978-3-642-17650-0_5

[31] Jean-Sébastien Coron. 1999. Resistance against Differential Power Analysis for Elliptic Curve Cryptosystems. In *Cryptographic Hardware and Embedded Systems, First International Workshop, CHES'99, Worcester, MA, USA, August 12-13, 1999, Proceedings (Lecture Notes in Computer Science)*, Çetin Kaya Koç and Christof Paar (Eds.), Vol. 1717. Springer, 292–302. https://doi.org/10.1007/3-540-48059-5_25

[32] Scott A. Crosby, Dan S. Wallach, and Rudolf H. Riedi. 2009. Opportunities and Limits of Remote Timing Attacks. *ACM Transactions on Information and System Security (TISSEC)* 12, 3 (2009), 17:1–17:29. https://doi.org/10.1145/1455526.1455530

[33] T. Dierks and C. Allen. 1999. *The TLS Protocol Version 1.0.* RFC 2246. RFC Editor. https://doi.org/10.17487/RFC2246

[34] Margaux Dugardin, Louiza Papachristodoulou, Zakaria Najm, Lejla Batina, Jean-Luc Danger, and Sylvain Guilley. 2016. Dismantling Real-World ECC with Horizontal and Vertical Template Attacks. In *Constructive Side-Channel Analysis and Secure Design - 7th International Workshop, COSADE 2016, Graz, Austria, April 14-15, 2016, Revised Selected Papers (Lecture Notes in Computer Science)*, François-Xavier Standaert and Elisabeth Oswald (Eds.), Vol. 9689. Springer, 88–108. https://doi.org/10.1007/978-3-319-43283-0_6

[35] Pierre-Alain Fouque and Frédéric Valette. 2003. The Doubling Attack - *Why Upwards Is Better than Downwards*. In *Cryptographic Hardware and Embedded Systems - CHES 2003, 5th International Workshop, Cologne, Germany, September 8-10, 2003, Proceedings (Lecture Notes in Computer Science)*, Colin D. Walter, Çetin Kaya Koç, and Christof Paar (Eds.), Vol. 2779. Springer, 269–280. https://doi.org/10.1007/978-3-540-45238-6_22

[36] Daniel Genkin, Lev Pachmanov, Itamar Pipman, and Eran Tromer. 2016. ECDH Key-Extraction via Low-Bandwidth Electromagnetic Attacks on PCs. In *Topics in Cryptology - CT-RSA 2016 - The Cryptographers' Track at the RSA Conference 2016, San Francisco, CA, USA, February 29 - March 4, 2016, Proceedings (Lecture Notes in Computer Science)*, Kazue Sako (Ed.), Vol. 9610. Springer, 219–235. https://doi.org/10.1007/978-3-319-29485-8_13

[37] Daniel Genkin, Lev Pachmanov, Itamar Pipman, Eran Tromer, and Yuval Yarom. 2016. ECDSA Key Extraction from Mobile Devices via Nonintrusive Physical Side Channels. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi (Eds.). ACM, 1626–1638. https://doi.org/10.1145/2976749.2978353

[38] Daniel Genkin, Adi Shamir, and Eran Tromer. 2014. RSA Key Extraction via Low-Bandwidth Acoustic Cryptanalysis. In *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I (Lecture Notes in Computer Science)*, Juan A. Garay and Rosario Gennaro (Eds.), Vol. 8616. Springer, 444–461. https://doi.org/10.1007/978-3-662-44371-2_25

[39] Daniel Genkin, Luke Valenta, and Yuval Yarom. 2017. May the Fourth Be With You: A Microarchitectural Side Channel Attack on Several Real-World Applications of Curve25519. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu (Eds.). ACM, 845–858. https://doi.org/10.1145/3133956.3134029

[40] Gabriel Goller and Georg Sigl. 2015. Side Channel Attacks on Smartphones and Embedded Devices Using Standard Radio Equipment. In *Constructive Side-Channel Analysis and Secure Design - 6th International Workshop, COSADE 2015, Berlin, Germany, April 13-14, 2015. Revised Selected Papers (Lecture Notes in Computer Science)*, Stefan Mangard and Axel Y. Poschmann (Eds.), Vol. 9064. Springer, 255–270. https://doi.org/10.1007/978-3-319-21476-4_17

[41] Gilbert Goodwill, Benjamin Jun, Josh Jaffe, and Pankaj Rohatgi. 2011. A testing methodology for side-channel resistance validation. In *Non-Invasive Attack Testing Workshop, NIAT 2011, Nara, Japan, September 26-27, 2011. Proceedings*. NIST. https://csrc.nist.gov/csrc/media/events/non-invasive-attack-testing-workshop/documents/08_goodwill.pdf

[42] Daniel Gruss, Raphael Spreitzer, and Stefan Mangard. 2015. Cache Template Attacks: Automating Attacks on Inclusive Last-Level Caches. In *24th USENIX Security Symposium, USENIX Security 15, Washington, D.C., USA, August 12-14, 2015*, Jaeyeon Jung and Thorsten Holz (Eds.). USENIX Association, 897–912. https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/gruss

[43] Shay Gueron and Vlad Krasnov. 2015. Fast prime field elliptic-curve cryptography with 256-bit primes. *J. Cryptographic Engineering* 5, 2 (2015), 141–151. https://doi.org/10.1007/s13389-014-0090-x

[44] Björn Haase and Benoît Labrique. 2017. Making Password Authenticated Key Exchange Suitable for Resource-Constrained Industrial Control Devices. In *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings (Lecture Notes in Computer Science)*, Wieland Fischer and Naofumi Homma (Eds.), Vol. 10529. Springer, 346–364. https://doi.org/10.1007/978-3-319-66787-4_17

[45] Ralf Hund, Carsten Willems, and Thorsten Holz. 2013. Practical Timing Side Channel Attacks Against Kernel Space ASLR. In *20th Annual Network and Distributed System Security Symposium, NDSS 2013, San Diego, California, USA, February 24-27, 2013*. The Internet Society. https://www.ndss-symposium.org/ndss2013/practical-timing-side-channel-attacks-against-kernel-space-aslr

[46] Toshiya Itoh and Shigeo Tsujii. 1988. A fast algorithm for computing multiplicative inverses in GF(2^m) using normal bases. *Inform. and Comput.* 78, 3 (1988), 171–177. https://doi.org/10.1016/0890-5401(88)90024-7

[47] Josh Jaffe, Pankaj Rohatgi, and Marc Witteman. 2011. Efficient side-channel testing for public key algorithms: RSA case study. In *Non-Invasive Attack Testing Workshop, NIAT 2011, Nara, Japan, September 26-27, 2011. Proceedings*. NIST. https://csrc.nist.gov/CSRC/media/Events/Non-Invasive-Attack-Testing-Workshop/documents/09_Jaffe.pdf

[48] Paul Kocher, Jann Horn, Anders Fogh, Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, Moritz Lipp, Stefan Mangard, Thomas Prescher, Michael Schwarz, and Yuval Yarom. 2019. Spectre Attacks: Exploiting Speculative Execution. In *2019 IEEE Symposium on Security and Privacy, SP 2019, Proceedings, 20-29 May 2019, San Francisco, California, USA*. IEEE, 19–37. https://doi.org/10.1109/SP.2019.00002

[49] Paul C. Kocher. 1996. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings (Lecture Notes in Computer Science)*, Neal Koblitz (Ed.), Vol. 1109. Springer, 104–113. https://doi.org/10.1007/3-540-68697-5_9

[50] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. 1999. Differential Power Analysis. In *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings (Lecture Notes in Computer Science)*, Michael J. Wiener (Ed.), Vol. 1666. Springer, 388–397. https://doi.org/10.1007/3-540-48405-1_25

[51] Moritz Lipp, Daniel Gruss, Raphael Spreitzer, Clémentine Maurice, and Stefan Mangard. 2016. ARMageddon: Cache Attacks on Mobile Devices. In *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016*, Thorsten Holz and Stefan Savage (Eds.). USENIX Association, 549–564. https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/lipp

[52] Moritz Lipp, Michael Schwarz, Daniel Gruss, Thomas Prescher, Werner Haas, Anders Fogh, Jann Horn, Stefan Mangard, Paul Kocher, Daniel Genkin, Yuval Yarom, and Mike Hamburg. 2018. Meltdown: Reading Kernel Memory from User Space. In *27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018*, William Enck and Adrienne Porter Felt (Eds.). USENIX Association, 973–990. https://www.usenix.org/conference/usenixsecurity18/presentation/lipp

[53] Mingjie Liu, Jiazhe Chen, and Hexin Li. 2013. Partially Known Nonces and Fault Injection Attacks on SM2 Signature Algorithm. In *Information Security and Cryptology - 9th International Conference, Inscrypt 2013, Guangzhou, China, November 27-30, 2013, Revised Selected Papers (Lecture Notes in Computer Science)*, Dongdai Lin, Shouhuai Xu, and Moti Yung (Eds.), Vol. 8567. Springer, 343–358. https://doi.org/10.1007/978-3-319-12087-4_22

[54] Jake Longo, Elke De Mulder, Dan Page, and Michael Tunstall. 2015. SoC It to EM: ElectroMagnetic Side-Channel Attacks on a Complex System-on-Chip. In *Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings (Lecture Notes in Computer Science)*, Tim Güneysu and Helena Handschuh (Eds.), Vol. 9293. Springer, 620–640. https://doi.org/10.1007/978-3-662-48324-4_31

[55] Clémentine Maurice, Christoph Neumann, Olivier Heen, and Aurélien Francillon. 2015. C5: Cross-Cores Cache Covert Channel. In *Detection of Intrusions and Malware, and Vulnerability Assessment - 12th International Conference, DIMVA 2015, Milan, Italy, July 9-10, 2015, Proceedings (Lecture Notes in Computer Science)*, Magnus Almgren, Vincenzo Gulisano, and Federico Maggi (Eds.), Vol. 9148. Springer, 46–64. https://doi.org/10.1007/978-3-319-20550-2_3

[56] Christopher Meyer, Juraj Somorovsky, Eugen Weiss, Jörg Schwenk, Sebastian Schinzel, and Erik Tews. 2014. Revisiting SSL/TLS Implementations: New Bleichenbacher Side Channels and Attacks. In *Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, August 20-22, 2014*, Kevin Fu and Jaeyeon Jung (Eds.). USENIX Association, 733–748. https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/meyer

[57] Peter L. Montgomery. 1987. Speeding the Pollard and elliptic curve methods of factorization. *Math. Comp.* 48, 177 (1987), 243–264. https://doi.org/10.2307/2007888

[58] Amir Moradi, Bastian Richter, Tobias Schneider, and François-Xavier Standaert. 2018. Leakage Detection with the χ^2-Test. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2018, 1 (2018), 209–237. https://doi.org/10.13154/tches.v2018.i1.209-237

[59] Cédric Murdica, Sylvain Guilley, Jean-Luc Danger, Philippe Hoogvorst, and David Naccache. 2012. Same Values Power Analysis Using Special Points on Elliptic Curves. In *Constructive Side-Channel Analysis and Secure Design - Third International Workshop, COSADE 2012, Darmstadt, Germany, May 3-4, 2012. Proceedings (Lecture Notes in Computer Science)*, Werner Schindler and Sorin A. Huss (Eds.), Vol. 7275. Springer, 183–198. https://doi.org/10.1007/978-3-642-29912-4_14

[60] Erick Nascimento, Lukasz Chmielewski, David Oswald, and Peter Schwabe. 2016. Attacking Embedded ECC Implementations Through cmov Side Channels. In *Selected Areas in Cryptography - SAC 2016 - 23rd International Conference, St. John's, NL, Canada, August 10-12, 2016, Revised Selected Papers (Lecture Notes in Computer Science)*, Roberto Avanzi and Howard M. Heys (Eds.), Vol. 10532. Springer, 99–119. https://doi.org/10.1007/978-3-319-69453-5_6

[61] Erick Nascimento, Julio López, and Ricardo Dahab. 2015. Efficient and Secure Elliptic Curve Cryptography for 8-bit AVR Microcontrollers. In *Security, Privacy, and Applied Cryptography Engineering - 5th International Conference, SPACE 2015, Jaipur, India, October 3-7, 2015, Proceedings (Lecture Notes in Computer Science)*, Rajat Subhra Chakraborty, Peter Schwabe, and Jon A. Solworth (Eds.), Vol. 9354. Springer, 289–309. https://doi.org/10.1007/978-3-319-24126-5_17

[62] Katsuyuki Okeya and Kouichi Sakurai. 2002. A Second-Order DPA Attack Breaks a Window-Method Based Countermeasure against Side Channel Attacks. In *Information Security, 5th International Conference, ISC 2002 Sao Paulo, Brazil, September 30 - October 2, 2002, Proceedings (Lecture Notes in Computer Science)*, Agnes Hui Chan and Virgil D. Gligor (Eds.), Vol. 2433. Springer, 389–401. https://doi.org/10.1007/3-540-45811-5_30

[63] Dag Arne Osvik, Adi Shamir, and Eran Tromer. 2006. Cache Attacks and Countermeasures: The Case of AES. In *Topics in Cryptology - CT-RSA 2006, The Cryptographers' Track at the RSA Conference 2006, San Jose, CA, USA, February 13-17, 2006, Proceedings (Lecture Notes in Computer Science)*, David Pointcheval (Ed.), Vol. 3860. Springer, 1–20. https://doi.org/10.1007/11605805_1

[64] Colin Percival. 2005. Cache Missing for Fun and Profit. In *BSDCan 2005, Ottawa, Canada, May 13-14, 2005, Proceedings*. http://www.daemonology.net/papers/cachemissing.pdf

[65] Cesar Pereida García and Billy Bob Brumley. 2017. Constant-Time Callees with Variable-Time Callers. In *26th USENIX Security Symposium, USENIX Security 2017, Vancouver, BC, Canada, August 16-18, 2017*, Engin Kirda and Thomas Ristenpart (Eds.). USENIX Association, 83–98. https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/garcia

[66] Jean-Jacques Quisquater and David Samyde. 2001. ElectroMagnetic Analysis (EMA): Measures and Counter-Measures for Smart Cards. In *Smart Card Programming and Security, International Conference on Research in Smart Cards, E-smart 2001, Cannes, France, September 19-21, 2001, Proceedings (Lecture Notes in Computer Science)*, Isabelle Attali and Thomas P. Jensen (Eds.), Vol. 2140. Springer, 200–210. https://doi.org/10.1007/3-540-45418-7_17

[67] Tobias Schneider and Amir Moradi. 2016. Leakage assessment methodology – Extended version. *Journal of Cryptographic Engineering* 6, 2 (2016), 85–99. https://doi.org/10.1007/s13389-016-0120-y

[68] Ru-Hui Shi, Zeng-Ju Li, Lei Du, Qian Peng, and Jiu-Ba Xu. 2015. Side Channel Analysis on SM2 Decryption Algorithm. *Journal of Cryptologic Research* 2, 5 (2015), 467–476. https://doi.org/10.13868/j.cnki.jcr.000093

[69] Sam Tregar. 2002. *Writing Perl Modules for CPAN*. Apress. https://doi.org/10.1007/978-1-4302-1152-5

[70] Joop van de Pol, Nigel P. Smart, and Yuval Yarom. 2015. Just a Little Bit More. In *Topics in Cryptology - CT-RSA 2015, The Cryptographer's Track at the RSA Conference 2015, San Francisco, CA, USA, April 20-24, 2015. Proceedings (Lecture Notes in Computer Science)*, Kaisa Nyberg (Ed.), Vol. 9048. Springer, 3–21. https://doi.org/10.1007/978-3-319-16715-2_1

[71] Tom van Goethem, Wouter Joosen, and Nick Nikiforakis. 2015. The Clock is Still Ticking: Timing Attacks in the Modern Web. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-16, 2015*, Indrajit Ray, Ninghui Li, and Christopher Kruegel (Eds.). ACM, 1382–1393. https://doi.org/10.1145/2810103.2813632

[72] Pepe Vila and Boris Köpf. 2017. Loophole: Timing Attacks on Shared Event Loops in Chrome. In *26th USENIX Security Symposium, USENIX Security 2017, Vancouver, BC, Canada, August 16-18, 2017*, Engin Kirda and Thomas Ristenpart (Eds.). USENIX Association, 849–864. https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/vila

[73] Samuel Weiser, Raphael Spreitzer, and Lukas Bodner. 2018. Single Trace Attack Against RSA Key Generation in Intel SGX SSL. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security, AsiaCCS 2018, Incheon, Republic of Korea, June 04-08, 2018*, Jong Kim, Gail-Joon Ahn, Seungjoo Kim, Yongdae Kim, Javier López, and Taesoo Kim (Eds.). ACM, 575–586. https://doi.org/10.1145/3196494.3196524

[74] Bernard L. Welch. 1947. The generalization of 'Student's' problem when several different population variances are involved. *Biometrika* 34 (1947), 28–35. http://www.jstor.org/stable/2332510

[75] Yuval Yarom and Katrina Falkner. 2014. FLUSH+RELOAD: A High Resolution, Low Noise, L3 Cache Side-Channel Attack. In *Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, August 20-22, 2014*. USENIX Association, 719–732. https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/yarom

[76] Kaiyu Zhang, Sen Xu, Dawu Gu, Haihua Gu, Junrong Liu, Zheng Guo, Ruitong Liu, Liang Liu, and Xiaobo Hu. 2017. Practical Partial-Nonce-Exposure Attack on ECC Algorithm. In *13th International Conference on Computational Intelligence and Security, CIS 2017, Hong Kong, China, December 15-18, 2017*. IEEE, 248–252. https://doi.org/10.1109/CIS.2017.00061

[77] Yinqian Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. 2014. Cross-Tenant Side-Channel Attacks in PaaS Clouds. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014*, Gail-Joon Ahn, Moti Yung, and Ninghui Li (Eds.). ACM, 990–1003. https://doi.org/10.1145/2660267.2660356

A REMOTE TIMINGS SCA EVALUATION: ECDSA

As stated in the Introduction, as a secondary goal, we aimed at reviewing the abstraction level at which SCA countermeasures are implemented. Specifically, pushing for a *secure-by-default* approach, we proposed to move each one of the SCA countermeasures discussed in this work at the lowest possible abstraction level.

As a result, the changes we proposed affected also other existing cryptosystems, increasing their resistance to SCA. In particular, in this section we evaluate the impact of our patchset on the ECDSA cryptosystem, specifically when using *generic* prime curves (as opposed to curves for which an alternative optimized implementation is specifically provided).

Figure 11 shows an empirical evaluation —similar to the one presented in Section 4 and Section 7—on the impact of the proposed mitigations on ECDSA over the secp256k1 [24] GLV prime curve used in the BitCoin protocol.

Both plots show the latency dependency on the nonce length: the top plot related to the OpenSSL implementation as of version 1.1.1-pre3, while the bottom plot shows the results after applying the proposed patchset. Specifically, the original implementation already applied the *nonce padding* and the *FLT modular inversion*

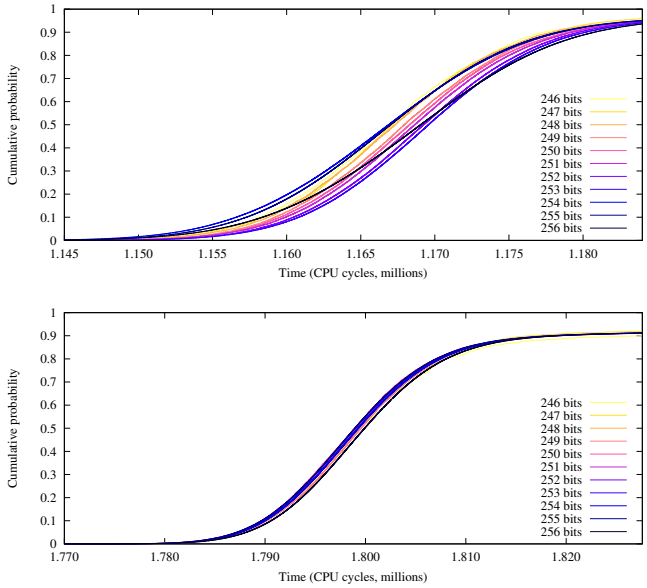

Figure 11: Latency dependency on the nonce length for ECDSA signature generation over the secp256k1 prime curve on amd64 architecture. Top: OpenSSL 1.1.1-pre3, in which a *w*NAF algorithm is used to implement EC scalar multiplication (see Section 5.1). Bottom: After applying our patchset (see Section 7), most notably switching to a Montgomery ladder algorithm for scalar multiplication instead of *w*NAF.

countermeasures, so the main change between the two implementations is due to adopting a Montgomery ladder algorithm for EC scalar multiplication instead of the *w*NAF algorithm adopted in the original implementation (see Section 7).

Comparing the two plots, our mitigations introduce an improvement centered around the median values. This is due in part to the fact that timings in the top plot depend on the weight of scalars, while the timings in the bottom plot are independent of the weight. This leads to lower deviations for the majority of data points centered around the median.