SIDE-CHANNEL ANALYSIS OF SM2
A Late-Stage Featurization Case Study

ACSAC 34 — 2018 Annual Computer Security Applications Conference

Nicola Tuveri    Sohaib ul Hassan
Cesar Pereida García    Billy Bob Brumley

{nicola.tuveri,sohaibulhassan,cesar.pereidagarcia,billy.brumley}@tut.fi
Laboratory of Pervasive Computing
Tampere University of Technology, Finland
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).
OUTLINE

1. About the title

2. Side-channel Analysis

3. Goals

4. Mitigations

5. Conclusions
SM2 is a suite of elliptic curve public key cryptosystems, standardized as a part of Chinese commercial cryptography mandates.

SM2DSA signature generation

The user’s private-public keypair is \((d_A, Q_A)\); \(ZA\) is the personalization string (hash) and \(m\) the message.

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

SM2PKE decryption

SM2PKE is roughly analogous to ECIES. Ciphertext \(C = C_1 || C_2 || C_3\); \(C_1\) is ECDHE public key (point), \(C_2\) is OTP ciphertext, \(C_3\) is the authentication tag. \((d_B, Q_B)\) is recipient keypair

1. Convert \(C_1\) to a point on \(E\). (return error if invalid point)
2. Compute \((x, y) = [d_B]C_1\), the shared ECDH point.
3. Compute \(z = KDF(x \| y, |C_2|)\), the OTP key; \(|z| = |C_2|\).
4. Compute \(m' = z \oplus C_2\), i.e. OTP decryption.
5. Compute \(t' = H(x \| m' \| y)\), the purported tag.
6. If \(t' \neq C_3\) holds, return an error.
7. Return the plaintext \(m'\).
A TIMELINE

→ **Nov’17 #4793** is opened to add SM2 support to OpenSSL
→ **Jan’18 #4793** is assigned to the Post-1.1.1 milestone
→ **Mar’18, 18th #4793** is merged into 1.1.1-pre3 (beta 1)
→ **Mar’18, end** SM2 code catches our attention
→ **Apr’18, 10th** Preliminary security assessment sent to OpenSSL Security
→ **May’18, 15th** tentative final release date for 1.1.1, *as planned at the time of 1.1.1-pre3*
→ **Sep’18, 11th** actual final release date for 1.1.1, *delayed for RFC 8446 (TLS 1.3) publication*
SM2DSA: REMOTE TIMINGS

(a) K-283 binary curve

(b) Recommended SM2 prime curve

Figure: Latency dependency on the nonce length in SM2DSA in OpenSSL 1.1.1-pre3.

Partial knowledge on the nonce leads to recovering the long-term key!
SM2DSA: CACHE TIMINGS

(a) Scalar multiplication: partial info on the secret nonce

(b) Binary GCD modular inversion: partial info on long-term private key

**Figure:** Partial raw cache-timing traces during SM2DSA in OpenSSL 1.1.1-pre3.

→ **Partial knowledge on the nonce leads to recovering the long-term key!**

→ **Modular inversion directly leaks knowledge on the long-term key, and is the same for each generated signature!**
**SM2PKE: EM ANALYSIS**

(a) EM capture setup.

(b) A filtered EM trace, revealing the sequence of ECC Double and Add depending on the long-term secret key.

**Figure:** EM analysis on SM2PKE decryption in OpenSSL 1.1.1-pre3.

→ The attacker can choose the input EC point;
→ **The sequence of operations directly leaks knowledge on the long-term key!**
→ We use statistical tools to assess the leakage across several traces.
GOALS

Mitigation of SCA deficiencies

After empirically verifying the SCA deficiencies, mitigate them, intersecting OpenSSL 1.1.1 release.

Push for secure-by-default approach

Review the abstraction level at which SCA countermeasures are implemented, to avoid reintroducing previously fixed vulnerabilities when adding new functionality.
MITIGATIONS: REMOTE TIMING ON SM2DSA

→ Scalar padding for scalar multiplication
→ Montgomery ladder instead of wNAF for scalar multiplication
→ Modular inversion through FLT exponentiation

**Figure:** Latency dependency on the nonce length in SM2DSA after mitigations.
MITIGATIONS: CACHE TIMING ON SM2DSA

→ Scalar padding for scalar multiplication
→ Montgomery ladder instead of wNAF for scalar multiplication
→ Modular inversion through FLT exponentiation

(a) Scalar multiplication: regular sequence of Double and Add ops
(b) FLT modular inversion: exponent is public (regular patterns anyway)

Figure: Partial raw cache-timing traces during SM2DSA after mitigations.
MITIGATIONS: EM ANALYSIS ON SM2PKE DECRYPTION

→ Scalar padding + Montgomery ladder
→ EC point coordinate blinding

**Figure:** A filtered EM trace after countermeasures: regular sequence of Double and Add operations.
### SECURE-BY-DEFAULT: PULL REQUESTS

1. **#6009+#6070** (and **#6535, #6608, #6648**):
   - Use **Montgomery ladder** if wNAF is called for single scalar multiplication;
   - Move **scalar padding** from ECDSA to EC_POINT_mul;
   - Unify binary and prime curves code path for scalar multiplication (*+blinding for binary field modular inversion*)

2. **#6062**: improve SM2 code quality

3. **#6116+#6521**: dedicated function for inversion modulo EC Group order with SCA countermeasures (**FLT modular inversion**)

4. **#6501**: GFp **coordinate blinding**

5. **#6690**: optimized GF2m ladder implementation + GF2m **coordinate blinding**

6. **#6772+#7000**: optimized GFp ladder implementation
CONCLUSIONS

→ SCA mitigations at a lower abstraction level
→ Also bring security to generic curve scalar multiplication (longstanding issue, since 2009)

(a) OpenSSL 1.1.1-pre3, wNAF for scalar multiplication

(b) After mitigations (ladder, padding, FLT modular inversion)

Figure: Latency dependency on the nonce length for ECDSA signature generation over the secp256k1 prime curve.
CONCLUSIONS

→ SCA mitigations at a lower abstraction level
  → Also bring security to generic curve scalar multiplication (longstanding issue, since 2009)
  → New EC cryptosystem implementations benefit from a **secure-by-default** approach
→ Met the goal of intersecting 1.1.1 release: **no CVE can be good!!!**
→ Ladder implementation with flexible structure for optimized (e.g., performance, exception/branch free) implementations
CONCLUSIONS: SOFTWARE ENGINEERING PERSPECTIVES

Lessons learned:

1. SW projects, OpenSSL included, should maintain a stronger separation between release, beta and dev branches to inhibit “feeping creaturism”

2. Milestones for security critical features should be set consistent with the complexity of the review process
QUESTIONS?

Thank you!!
**SM2DSA: REMOTE TIMINGS (COMPARISON)**

(a) K-283 binary curve

(b) Recommended SM2 prime curve

*Figure:* Latency dependency on the nonce length in SM2DSA in OpenSSL 1.1.1-pre3.

(a) K-283 binary curve

(b) Recommended SM2 prime curve

*Figure:* Latency dependency on the nonce length in SM2DSA after mitigations.
**SM2DSA: CACHE TIMINGS (COMPARISON)**

(a) Scalar multiplication

(b) Binary GCD modular inversion

**Figure:** Partial raw cache-timing traces during SM2DSA in OpenSSL 1.1.1-pre3.

(a) Scalar multiplication: regular sequence of Double and Add ops

(b) FLT modular inversion: exponent is public *(regular patterns anyway)*

**Figure:** Partial raw cache-timing traces during SM2DSA after mitigations.