Efficient Side-Channel Attacks on Scalar Blinding on Elliptic Curves with Special Structure

Werner Schindler and Andreas Wiemers
Bundesamt für Sicherheit in der Informationstechnik (BSI)
Bonn, Germany

Gaithersburg, June 12, 2015

Introduction

- Elliptic curves over finite prime fields.
- Coron's first countermeasure: Use blinding factors to protect the long term key.
- Originally 20 bits; used in practice: typically 32-64 bits.
- General primes: If at all practical, attacks on $>$ 64 bit blinding require large workload, see [3].
- Our contribution: Special prime fields need much larger blinding factors!

Notation

- P = base point of an elliptic curve over GF(p).
- $y = \text{ord}(P)$
- d = long-term key; $0 < d < y$
- $r_j \in \{0, \ldots, 2^R - 1\}$, $r_j = j^{\text{th}}$ blinding factor
- R = blinding length

- $v_j = d + r_j y$ blinded j^{th} scalar

Applications of Static Scalar Multiplications

- static Diffie Hellman
- ECIES
- signature-less authentication process for TLS 1.3 (proposal of H. Krawczyk)
- deterministic signatures

Attack Scenario

- The attacker guesses v_j for $j = 1, \ldots, N$ on the basis of a side-channel attack (e.g. a single-trace template attack)
- $\widetilde{v}_j$ (guessed blinded scalar)
- $\epsilon_b := \mathrm{Prob}(\widetilde{v}_{j,i} \neq v_{j,i})$ (probability of a wrong bit guess)
- The papers [2, 3] consider attacks on general elliptic curves (and on RSA)

Special Curves

- Examples: NIST P-384, ED448, M-511, Curve41417, Curve25519.
- $y = 2^k \pm y_0$ with $y_0 = 2^t + \cdots + 1$ and $t \approx k/2$ (valid for elliptic curves over GF(p) when $p \approx 2^{k+b}$ with cofactor 2^b, $b \geq 0$)
- 'gap' $g := k - t - 1$ (if $y = 2^k + y_0$: no. of zeroes between the two most significant '1's in the binary representation of y)

Observation and basic attack idea (for $y = 2^k + y_0$):

- $v_j = d + r_j y = r_j 2^k + (d + r_j y_0)$
- If $R \leq g - 7$, for instance, a carry of $(d + r_j y_0)$ from bit $k-1$ to k is rather unlikely.
- $\Longrightarrow \widetilde{v}_{j;k}, \ldots, \widetilde{v}_{j;k+R-1}$ are initial guesses for $r_{j;0}, .., r_{j;R-1}$
- Note: Our attacks work even for $R \leq g - 2$

The Wide Window Attack

- $v_j = d + r_j y = r_j 2^k + (d + r_j y_0)$
- $\widetilde{v}_j$: Consider the pair $[\lfloor \widetilde{v}_j / 2^k \rfloor, \widetilde{v}_j (\bmod 2^k)]$.
- It is just $[r_j, d + r_j y_0] \oplus \delta_j$
- δ_j has low Hamming weight (error vector).
- Reduce each component $\bmod 2^w$ with $0 < w < k$.

Algorithm 1: Find $d \pmod{2^w}$.

- Try to correct the guessing errors of δ_j in each component of the pair $[\lfloor \widetilde{v}_j / 2^k \rfloor \pmod{2^w}, \widetilde{v}_j \pmod{2^w}]$.
- If the correction is successful, the corrected pair is just $[r_j \pmod{2^w}, d + r_j y_0 \pmod{2^w}]$.
- Compute a candidate $d \pmod{2^w}$ as $d + r_j y_0 - r_j y_0 \pmod{2^w}$.
- The set of all candidates for $d \pmod{2^w}$ is a small subset of $\{0, \ldots, 2^w - 1\}$. Hope: The correct one shows up at least 2 times.
- Problem: For the next step, we need to know <u>all</u> bits of r_j.

Algorithm 2: Find $r_j \pmod{2^w}$, if $d \pmod{2^w}$ is known.

- Try to correct the guessing errors in the first component of the pair $[\lfloor \widetilde{v}_j/2^k \rfloor \pmod{2^w}, \widetilde{v}_j \pmod{2^w}]$.
- If the correction is successful, we recover $r_j \pmod{2^w}$.
- Compute $\widetilde{v}_j \pmod{2^w} \oplus (d + r_j y_0 \pmod{2^w}) = \Delta$. Does Δ have <u>low</u> Hamming weight?
- Keep all candidates for $r_j \pmod{2^w}$, where Δ has low Hamming weight.

Example: $R = 120$. Run through all the bits of d from 0 to 240

- w=24: Algorithm 1, Algorithm 2.
- w=48: Algorithm 1, Algorithm 2, (adapted to the most significant 24 bits).
- w=72: Algorithm 1, Algorithm 2, (adapted to the most significant 24 bits).
- w=96: Algorithm 1, Algorithm 2, (adapted to the most significant 24 bits).
- w=120: Algorithm 1, (adapted to the most significant 24 bits).
- w=144: Algorithm 1, (adapted to the most significant 24 bits).
- w=168: Algorithm 1, (adapted to the most significant 24 bits).
- . . .

Simulation

- 10 simulations for each $N \in \{250, 500, 1000\}$. We counted an attack to be successful if, in each of the 10 steps, algorithm 1 outputs the correct $d (\bmod 2^w)$.

curve	R	ϵ_b	N	success rate
Curve25519	120	0.10	250	2/10
Curve25519	120	0.10	500	7/10
Curve25519	120	0.10	1,000	9/10

The Narrow Window Attack

- The Narrow Window Attack considers much smaller windows than the Wide Window Attack.
- Within these windows the information is exploited in an optimal way (maximum likelihood estimates).
- In our simulation experiments we used $w' = 8$ (Phase 1) and $w' = 10$ (Phase 3).

Narrow Window Attack (Generic description)

- **Phase 1** Guess iteratively ($\rightarrow$ windows) the R lowest bits of the long-term key d and the blinding factors $r_1, \ldots, r_N$. (Within Phase 1 trace j may be removed if the intermediate guess for $r_j \pmod{2^w}$ is assumed to be false.)
- **Phase 2** Identify the correct guesses of the blinding factors. Remove the other guesses.
- **Phase 3** Guess the remaining bits of d from the guesses $\widetilde{r}_{j_1}, \widetilde{r}_{j_2}, \ldots, \widetilde{r}_{j_u}$, which have survived Phase 2.

Narrow Window Attack (I)

curve	R	ϵ_b	N	success rate
Curve25519	64	0.12	400	9/10
Curve25519	120	0.10	700	19/20
Curve25519	120	0.12	5,000	19/20
Curve25519	120	0.13	15,000	23/30
Curve25519	120	0.14	60,000	18/30
Curve25519	120	0.15	400,000	5/10
Curve25519	125	0.10	1000	10/10
Curve25519	125	0.12	6,000	16/20
Curve25519	125	0.13	17,000	8/10
Curve25519	125	0.14	60,000	14/30

Tabelle: $g = 127$

Narrow Window Attack (II)

curve	R	ϵ_b	N	success rate
M-511	250	0.07	500	10/10
M-511	250	0.10	30,000	9/10
M-511	253	0.10	40,000	8/10
ED448	220	0.10	30,000	10/10
ED448	220	0.11	120,000	9/10
ED448	220	0.12	700,000	9/10
Curve41417	200	0.07	400	10/10
Curve41417	200	0.10	7,000	8/10
NIST P-384	190	0.10	4,000	10/10
NIST P-384	190	0.12	70,000	9/10

Tabelle: $g = 255$ (M-511), $g = 222$ (ED448), $g = 206$ (Curve41417), $g = 194$ (NIST P-384)

Attack Efficiency and Countermeasures

- For the above parameter sets the Narrow Window Attack essentially costs from $O(2^{29})$ to $O(2^{34})$ operations (each consisting of several inexpensive basic operations).

- Countermeasure:
- The length of the blinding factors R must at least exceed the gap $g \approx k/2$.
  - Example: Curve25519: $R > 127$ (minimum size)
  - Example: ED448: $R > 222$ (minimum size)

- Note: For Curve25519 D. Bernstein proposes 512-bit nonces ($\rightarrow R > 256$) in the context of signatures [1].

Conclusion

- Both the Wide Window Attack and the Narrow Window Attack are very efficient.
- To prevent both attacks elliptic curves over GF(p) for special primes p require blinding factors of length $R > g \approx k/2$.
- This feature at least reduces their efficiency gain over general curves.

Contact

Bundesamt für Sicherheit in der Informationstechnik (BSI),
Bonn, Germany

Werner Schindler and
Andreas Wiemers

P.O. Box 200363, 53133 Bonn,
Germany
Tel.: +49 (0)228-9582-5652/-5627
Fax: +49 (0)228-10-9582-5652/-5627

{Werner.Schindler,Andreas.Wiemers}
@bsi.bund.de

https://www.bsi.bund.de
https://www.bsi-fuer-buerger.de

[1] D.J. Bernstein and T. Lange: SafeCurves: Choosing Safe Curves for Elliptic-Curve Cryptography. http://safecurves.cr.yp.to, accessed at April 21, 2015.

[2] W. Schindler and K. Itoh: Exponent Blinding Does not Always Lift (Partial) SPA Resistance to Higher-Level Security. In: J. Lopez, G. Tsudik (eds.): Applied Cryptography and Network Security — ACNS 2011, Springer, LNCS 6715, Berlin 2011, 73 – 90.

[3] W. Schindler and A. Wiemers: Power Attacks in the Presence of Exponent Blinding. J Cryptogr Eng 4 (2014), 213-236.
online http://dx.doi.org/10.1007/s13389-014-0081-y