Identification of weak keys for Elliptic Curves Cryptography

Candidate: Enrico Talotti
Supervisor: Marino Miculan
Co-supervisor: Pietro De Poi

Università degli Studi di Udine

Master Thesis in Mathematics
December 5, 2023

Abstract

Abstract

We describe a novel type of weak cryptographic private key that can exist in any discrete logarithm-base public-key cryptosystem set in a group of prime order p where $p - 1$ has small divisors.

Keywords: Elliptic Curve Cryptography, Discrete Logarithm Problem, Weak keys, Rust

- Prabhat Kushwaha and Ayan Mahalanobis, *A probabilistic baby-step giant-step algorithm*.
- Prabhat Kushwaha Michael John Jacobson Jr., *Removable weak keys for discrete logarithm-based cryptography*.
- Enrico Talotti, *Elliptic curve*, `https://github.com/enh11/elliptic_curves`.

Elliptic Curves over Finite Fields

Let $\mathbb{K}$ be a finite field and let E be an elliptic curves over $\mathbb{K}$ given by the Weierstrass equation

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6, \text{ where } a_1, \ldots, a_6 \in \mathbb{K}$$

Theorem

Let $E(\mathbb{K})$ be the set of $\mathbb{K}$-rational points of E. We can turn $E(\mathbb{K})$ into a finite abelian group with identity the point at infinity $\mathcal{O}$ and with the chord-tangent operation denoted by $\oplus$.

We assume $E(\mathbb{K})$ to have prime order p. Let P be a generator of $E(\mathbb{K})$. The following maps is a group isomorphism:

$$\begin{aligned} \varphi : \mathbb{Z}_p &\rightarrow E(\mathbb{K}) \\ \alpha &\mapsto Q = [\alpha]P = \underbrace{P \oplus P \oplus \cdots \oplus P}_{\alpha \text{ times}}. \end{aligned}$$

The Elliptic Curve Discrete Logarithm Problem

Definition

The problem of computing the inverse of φ is called the *Elliptic Curves Discrete Logarithm Problem* (*ECDLP*) with respect P. It is the problem, given P and Q, to determine $\alpha \in \mathbb{Z}_p$ such that $Q = [\alpha]P$.

- The value $[\alpha]P$ can be computed very efficiently.
- There's no known algorithm that can solve the *ECDLP* much faster then $\mathcal{O}(\sqrt{p})$.
- The map φ is a *one-way-function*, thus we can build the Elliptic Curve Cryptosystem.
- We refer to α and $Q = [\alpha]P$ as *private-key* and *public-key* respectively.

Baby Step Giant Step

The *Baby Step Giant Step* algorithm is based on the following:

Lemma

Let p be a positive integer. Put $m := \lfloor \sqrt{p} \rfloor + 1$. Then for any α with $0 \leq \alpha < p$ there are integers $0 \leq i, j < m$, with $\alpha = i + jm$.

Suppose now $p = \text{ord}(E(\mathbb{K}))$. Then $Q = [\alpha]P$ implies

$$\boxed{Q \oplus [-jm]P = [i]P}$$

for i, j, m as in Lemma above.

Baby Step Giant Step

$$Q \oplus [-jm]P = [i]P$$

Baby Step Giant Step algorithm

Let $m = \lfloor \sqrt{p} \rfloor + 1$. Build the following two lists:

baby-step: $P, [2]P \dots, [m]P$
giant-step: $Q \oplus [-m]P, Q \oplus [-2m]P, \dots, Q \oplus [-m^2]P$

There exists a match between the two lists, that can be found in $\log m$ steps by using standard searching algorithms. Hence, the total running time for the algorithm is $\mathcal{O}(m \log m)$ steps.

The action of $\mathbb{Z}_p^*$

Assume $E(\mathbb{K})$ to be of prime order p and let P be a generator. We define the following map:

$$\begin{aligned} \rho : \mathbb{Z}_p^* &\longrightarrow \mathrm{Aut}(E(\mathbb{K})) \\ \alpha &\longmapsto \ \rho_\alpha : E(\mathbb{K}) \longrightarrow E(\mathbb{K}) \\ &\qquad\qquad\qquad\quad P \longmapsto [\alpha]P \end{aligned}$$

- This is an isomorphism between $\mathbb{Z}_p^*$ and $\mathrm{Aut}(E(\mathbb{K}))$ and we can identify $\alpha \in \mathbb{Z}_p^*$ with the automorphism ρ_α, i.e., with the point $[\alpha]P$.
- If $\alpha, \beta \in \mathbb{Z}_p^*$, then $\alpha\beta$ identifies the automorphism $\rho_{\alpha\beta}$ and thus the point $[\alpha\beta]P = [\alpha][\beta]P$.
- We can reduce the *ECDLP* to a problem in the multiplicative group $\mathbb{Z}_p^*$.

The action of $\mathbb{Z}_p^*$

Let P be a generator of the prime order group $E(\mathbb{K})$ and let $Q = [\alpha]P$. We want to find such an α.

- Let z be a primitive element of $\mathbb{Z}_p^*$, then $\alpha = z^k$ for some $0 \leq k < p-1$ and $Q = [z^k]P$.
- Let $m := \lfloor \sqrt{p-1} \rfloor + 1$. By the lemma above we have $k = i + mj$, for some $0 \leq i, j < m$.
- It follows that $Q = [z^k]P = [z^{i+jm}]P = [z^i][z^{jm}]P$, which leads to

$$[z^{-jm}]Q = [z^i]P.$$

- Hence, if we find such an i and j, we can compute $\alpha = z^{i+jm}$ and we have the solution of the *ECDLP*.

The implicit algorithm

Implicit Baby Step Giant Step

Let $m = \lfloor \sqrt{p-1} \rfloor + 1$. Build the following two lists:

$$\begin{aligned} \textit{baby-step}: &\quad [z]P,\ [z^2], \ldots,\ [z^m]P \\ \textit{giant-step}: &\quad [z^{-m}]Q,\ [z^{-2m}]Q,\ \ldots,\ [z^{-m^2}]Q. \end{aligned}$$

There exists a match between the two lists, that can be found in $\mathcal{O}(m \log m)$ steps.

This idea can be improved if a divisor d of $p-1$ is known.

- Let $z_d = z^{\frac{p-1}{d}}$ be a generator for the order d subgroup of $\mathbb{Z}_p^*$. Put $m := \lfloor \sqrt{d} \rfloor + 1$ and run the implicit baby step giant step by using z_d instead of z.
- If α happens to lie in the d order subgroup of $\mathbb{Z}_p^*$, then the algorithm finds α in $\mathcal{O}(\sqrt{d} \log \sqrt{d})$ steps.

Analysis of weak keys

Testing whether a key is weak

- Set a bound B for the order of subgroups of $\mathbb{Z}_p^*$.
- Generate the list $R(p, B)$ of integers $d_1 < d_2 < \cdots < d_t \leq B$ dividing $p - 1$ such that $d_i \nmid d_j$ for all $1 \leq i < j \leq t$.
- Run the implicit baby step giant step algorithm

Number of weak keys within the bound B and computational costs

- Set a bound B for the order of subgroups of $\mathbb{Z}_p^*$.
- $\log_2$ of the number of weak keys with order bounded by B; $n_B = \log_2 \sum_{\substack{d \mid p-1 \\ d \leq B}} \phi(d)$;
- $\log_2$ of the worst-case number of elliptic curve scalar multiplications required to test a key within the bound B; $c_B = \log_2 \sum_{d \in R(p,B)} 2\lceil \sqrt{d} \rceil$.

Numerical results

Table: Weak keys analysis of some standardized curves

Curve	$b(p)$	$n_{2^{32}}$	$c_{2^{32}}$	$n_{2^{64}}$	$c_{2^{64}}$	$n_{2^{128}}$	$c_{2^{128}}$	$n_{2^{160}}$	$c_{2^{160}}$
secp224k1	224	2.6	2.6	2.6	2.6	2.6	2.6	2.6	2.6
brainpoolP224r1	224	10.0	6.0	10.0	6.0	10.0	6.0	10.0	6.0
brainpoolP256r1	256	4.2	3.3	4.2	3.3	4.2	3.3	4.2	3.3
ECCp-359	359	5.2	3.6	5.2	3.6	5.2	3.6	5.2	3.6
sect193r2	193	2.0	2.0	2.0	2.0	110.2	56.1	110.2	56.1
Curve25519	253	7.04	4.8	7.04	4.8	114.3	58.2	144.7	73.4
ECCp-353	353	6.3	4.3	6.3	4.3	108.9	55.5	158.3	80.2
c2pnb163v3	162	8.8	5.4	8.8	5.4	8.8	5.4	160.9	82.3
secp256k1	256	24.1	13.1	64.7	34.2	129.4	67.0	147.9	75.0
secp256r1	256	36.0	21.5	69.3	38.8	133.2	70.8	165.3	86.9
SM2	256	32.5	18.13	59.7	30.8	59.7	30.8	59.7	30.8
P-521	521	31.4	16.7	50.0	26.0	128.8	66.3	130.5	66.2

The End