Radboud University

Algorithms for solving the discrete log

Cryptography, Spring 2021

L. Batina, J. Daemen

June 1, 2021

Institute for Computing and Information Sciences
Radboud University

Outline

Overview

Baby-step giant-step

Pollard's ρ method

Pohlig-Hellman

Index calculus

Overview

Algorithms to compute the discrete logarithm

(Elliptic curve) Discrete log problem

Determine a given G and $A \in \langle G \rangle$ with $[a]G = A$

- We distinguish two types of methods
  - generic methods: work for any cyclic group, including EC
  - specific methods: exploit properties of the group
- Generic methods:
  - Baby-step giant-step
  - Pollard's ρ
  - Pohlig-Hellman
  - ...
- Method specific for subgroups of multiplicative modular groups $(\mathbb{Z}/p\mathbb{Z})^*$
  - index calculus
  - ...
- We explain the algorithms in blue and give an idea of those in red

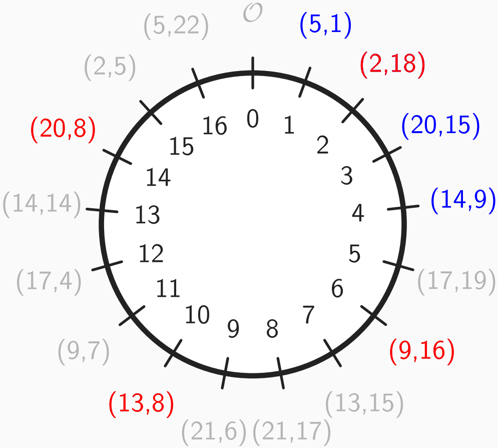
Baby-step giant-step

Baby-step giant-step, the algorithm (Daniel Shanks, 1971)

Input: A, G and table size m
Output: a that satisfies $[a]G = A$
$i \leftarrow 0$, $X \leftarrow G$, $T \leftarrow \{(X, 1)\}$
repeat
  $i \leftarrow i + 1$, $X \leftarrow X + G$, $T \leftarrow T \cup \{(X, i)\}$ {baby step}
until $i = m$
$j \leftarrow 0$, $Y \leftarrow A$
repeat
  $j \leftarrow j + 1$, $Y \leftarrow Y - [m]G$ {giant step}
until $\exists (X, i) \in T$ with $X = Y$
return $i + mj$

Baby-step giant-step, visually in $E(\mathbb{F}_{23}) : y^2 = x^3 - x - 4$

Say $G = (5,1)$ and $A = (20,8)$



$m = 4$, baby steps, giant steps $A - [3 \cdot 4]G = [2]G \Rightarrow a = 14$

Baby-step giant-step, discussion

- Generic algorithm: works for any cyclic group
- Baby steps
  - compute the values of $[i]G$ for i up to m
  - store them in table T
  - work: m point additions
  - storage: m points
- Giant steps
  - compute $A, A-[m]G, A-[2m]G$, etc.
  - until the point $A-[jm]G$, is also in table T
  - expected work: $\mathrm{ord}(G)/2m$ point additions and table checks
- The matching points satisfies $[i]G = A - [jm]G$ so $A = [i+mj]G$
- # point additions minimized by taking $m \approx \sqrt{\mathrm{ord}(G)}$
- Storage and table-check cost may favor $m \ll \sqrt{\mathrm{ord}(G)}$

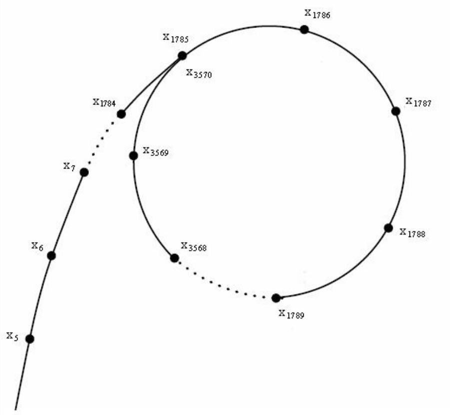
Pollard's ρ method

Pollard's ρ method (John Pollard, 1975)

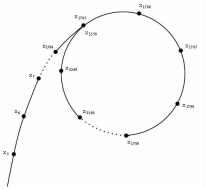
- Requires a transformation f over $\mathbb{Z}/q\mathbb{Z} \times \mathbb{Z}/q\mathbb{Z}$ with $q = \text{ord}(G)$
  - given (c_i, d_i), it computes $(c_{i+1}, d_{i+1}) = f(c_i, d_i)$
  - let $P_i = [c_i]A + [d_i]G$ then
    - f shall define a mapping f' over $\langle G \rangle$: $P_{i+1} = f'(P_i)$
    - f' shall behave like a random transformation
- Algorithm:
  - pick random couple (c_0, d_0)
  - compute the sequence (c_i, d_i) with $(c_i, d_i) = f(c_{i-1}, d_{i-1})$
  - stop if for some $i < j : P_i = P_j$
  - now $[c_i]A + [d_i]G = [c_j]A + [d_j]G$ or $[c_i - c_j]A = [d_j - d_i]G$
  - so if $(c_i, d_i) \neq (c_j, d_j)$ it follows that $a = \frac{d_j - d_i}{c_i - c_j}$
- It is unlikely this ends up in $(c_i, d_i) = (c_j, d_j)$
  - $(c_i, d_i) \in \mathbb{Z}/q\mathbb{Z} \times \mathbb{Z}/q\mathbb{Z}$ and $\#(\mathbb{Z}/q\mathbb{Z} \times \mathbb{Z}/q\mathbb{Z}) = q^2$
  - $P_i \in \langle G \rangle$ and $\text{ord}(G) = q$

Pollard's ρ: how many iterations are needed?

- Assume f' behaves like a random mapping
  - probability that P_i equals one of the previous points: $(i-1)/q$
  - probability there is a collision after n iterations $\approx n^2/2q$
  - expected value of n until the collision: $\sqrt{\frac{\pi q}{2}}$
- Experiments confirm this



Pollard's ρ: how to find colliding couples (x, y)?



- Storing all points P_i
  - requires about $\sqrt{q}$ storage and many comparisons
  - not better than baby-step giant-step
- Reducing storage with method of distinguished points
  - only store points that have some rare property
  - e.g., x-coordinate ends in ℓ trailing zeroes
  - reduces storage size by a factor $2^{-\ell}$
  - expected overshoot of $2^{\ell-1}$ additional iterations into the loop
  - taking 2^{ℓ} close to $\sqrt{q}$ solves storage problem
- There exist other methods to reduce storage

Pollard's ρ: choosing f

- Partitioning approach:
  - partition $\langle G \rangle$ in s classes $S_0, S_1, \ldots, S_{s-1}$ of similar size
  - have a different function f (and f') per class
  - choose classes so that it is easy to find the class of a point
- Classical choice: $s = 3$
  - S_0: $f(c, d) = (2c, 2d)$ so $f'(P) = [2]P$
  - S_1: $f(c, d) = (c + 1, d)$ so $f'(P) = P + A$
  - S_2: $f(c, d) = (c, d + 1)$ so $f'(P) = P + G$
- ECC-oriented choice: $s = 20$
  - per class S_k randomly choose $a_k, b_k \in \mathbb{Z}/q\mathbb{Z}$
  - S_k: $f(c, d) = (c + a_k, d + b_k)$ so $f'(P) = P + M_k$ with $M_k = [a_k]A + [b_k]G$
  - seems to work well

Pollard's ρ: toy example over $E(\mathbb{F}_{1093}) : y^2 = x^3 + x + 1$

$$G = (0, 1), \mathrm{ord}(G) = 1067, A = (413, 959)$$

- $s = 3$ with $P \in S_i$ if $x \bmod 3 = i$ and $f'(P) = P + M_i$ with

$$M_0 = [4]G + [3]A, M_1 = [9]G + [17]A, M_2 = [19]G + [6]A$$

- $P_0 = [3]G + [5]A = (326, 69) \in S_2$ so $P_1 = P_0 + M_2 = (727, 589)$

$$\begin{array}{lll} P_1 = (727, 589) & P_2 = (560, 365), & P_3 = (1070, 260), \\ P_4 = (473, 903), & P_5 = (1006, 951), & P_6 = (523, 938), \ldots, \\ P_{58} = (1006, 951), & P_{59} = (523, 938), & \ldots \end{array}$$

- we see $P_5 = P_{58}$ and kept track of (c, d):

$$P_5 = [88]G + [46]A \qquad P_{58} = [685]G + [620]A$$

- So $[88]G + [46]A = [685]G + [620]A$ and hence $[-597]G = [574]A$
- Since $\mathrm{ord}(G) = 1067$: $a = (1067 - 597)574^{-1} \bmod 1067 = 499$

Pollard's ρ in the real world

Pollard's ρ is the best method for prime-order subgroups of elliptic curves

Many efforts to solve discrete log in standard curves:

- European project ECRYPT(II) 2004-2013
  - ECC2K–130 challenge: Koblitz $E(\mathbb{F}_{2^{131}}) : y^2 + xy = x^3 + 1$
  - evaluated on several platforms: FPGA, ASIC, CPU, GPU (PS3)
  - E.g.: 1 year on 3039 Intel CPU Q6850, 4 cores, 2.997 GHz
- Wenger and Wolfger in 2014:
  - ECC2K–112 challenge: Koblitz $E(\mathbb{F}_{2^{112}}) : y^2 + xy = x^3 + x^2 + 1$
  - 24 days using 18-core Virtex-6 FPGA cluster
- Kusaka et al. in 2017:
  - Some special curve (used for pairings) over a 114-bit prime field
  - 6 months with 2000 Intel CPU cores

Pohlig-Hellman

Pohlig-Hellman method, the core idea

Stephen Pohlig and Martin Hellman, 1978

Let $\mathrm{ord}(G) = p_1 p_2$ with p_1 and p_2 coprime

- We look for a that satisfies $[a]G = A$ for given G and A
- Multiply both sides by $[p_1]$: $[p_1][a]G = [p_1]A$
  - let $G_{p_2} = [p_1]G$ and $A_{p_2} = [p_1]A$
  - we have $\mathrm{ord}(G_{p_2}) = p_2$ and $A_{p_2} \in \langle G_{p_2} \rangle$
  - if a satisfies $[a]G = A$, it also satisfies $[a]G_{p_2} = A_{p_2}$
  - if a is a solution of $[a]G_{p_2} = A_{p_2}$, so is $a \bmod q$
  - so solving $[a]G_{p_2} = A_{p_2}$ gives $a_{p_2} = a \bmod p_2$
  - with Pollard's ρ this costs roughly $\sqrt{p_2}$ computations
- Multiply both sides by $[p_2]$: $[a][p_2]G = [p_2]A$
  - along similar lines this gives $a_{p_1} = a \bmod p_1$
  - costs roughly $\sqrt{p_1}$ computations
- Compute a from a_{p_1} and a_{p_2} using CRT

Pohlig-Hellman method, the implications

- If $\mathrm{ord}(G)$ is composite, Pohlig-Hellman allows to
  - solve the discrete log problem for each of the factors of $\mathrm{ord}(G)$
  - combine the results with CRT
- For each prime power $p^n \mid \mathrm{ord}(G)$, work factor is $\sqrt{p}$
  - if $n = 1$, this is straightforward
  - if $n > 1$: the sophistication of Pohlig-Hellman [out of scope]
- Bottom line: complexity is dominated by the largest prime factor(s)

Why groups $\langle G \rangle$ for discrete-log crypto have prime order

This is mostly due to the Pohlig-Hellman discrete log algorithm!

Index calculus

Index calculus by example [for info only]

Example: $3^a \equiv 37 \pmod{1217}$, so $A = 37$ and $g = 3$

It uses a *factor base*, in this case $\{2, 3, 5, 7, 11, 13, 17, 19\}$

First step: find the discrete log of the elements of the factor base

$\pmod{1217}$	$\pmod{1216}$	$\pmod{1216}$
$3^1 \equiv 3$	$1 \equiv L(3)$	$L(2) \equiv 216$
$3^{24} \equiv -2^2 \cdot 7 \cdot 13$	$24 \equiv 608 + 2L(2) + L(7) + L(13)$	$L(3) \equiv 1$
$3^{25} \equiv 5^3$	$25 \equiv 3L(5)$	$L(5) \equiv 819$
$3^{30} \equiv -2 \cdot 5^2$	$30 \equiv 608 + L(2) + 2L(5)$	$L(7) \equiv 113$
$3^{34} \equiv -3 \cdot 7 \cdot 19$	$34 \equiv 608 + L(3) + L(7) + L(19)$	$L(11) \equiv 1059$
$3^{54} \equiv -5 \cdot 11$	$54 \equiv 608 + L(5) + L(11)$	$L(13) \equiv 87$
$3^{71} \equiv -17$	$71 \equiv 608 + L(17)$	$L(17) \equiv 679$
$3^{87} \equiv 13$	$87 \equiv L(13)$	$L(19) \equiv 528$

Find powers of g that have only factors in the base, take log and solve

Independent of A: valid for all key pairs with same domain parameters!

Index calculus by example (cont'd) [for info only]

Our equation: $3^a \equiv 37 \pmod{1217}$, so $A = 37$ and $g = 3$

For the factor base we have:

$$\begin{array}{cccc} L(2) \equiv 216 & L(3) \equiv 1 & L(5) \equiv 819 & L(7) \equiv 113 \\ L(11) \equiv 1059 & L(13) \equiv 87 & L(17) \equiv 679 & L(19) \equiv 528 \end{array}$$

Now find j such that $g^j \cdot A = 3^j \cdot 37$ factors over elements of the base

We find

$$3^{16} \cdot 37 \equiv 2^3 \cdot 7 \cdot 11 \pmod{1216}$$

We now have

$$\begin{aligned} L(37) &\equiv 3L(2) + L(7) + L(11) - 16 \pmod{1216} \\ &\equiv 3 \cdot 216 + 113 + 1059 - 16 \pmod{1216} \\ &\equiv 588 \pmod{1216} \end{aligned}$$

Index calculus, discussion

- Works for $\langle g \rangle$ a subgroup of multiplicative groups $(\mathbb{Z}/p\mathbb{Z})^*$
- Index calculus is much faster than generic attacks discussed and scales better with increasing p
- Forces us to take $p \geq 2^{3072}$ for 128 bits of security
- Works even better for subgroups of the multiplicative groups in prime power fields $\mathbb{F}_{p^n}$
- Many variants and sophistications, also for factoring
- Logs of factor base can be pre-computed for given domain parameters
- Index calculus does not work on elliptic-curve groups!

Conclusions

Conclusions

- Index calculus
  - forces us to choose large p for subgroups of $(\mathbb{Z}/p\mathbb{Z})^*$
  - very unlikely that it can be extended to ECC
- Pohlig-Hellman
  - forces us to take prime-order groups $\langle G \rangle$ for discrete-log crypto as it reduces security strength to that of the largest prime-order subgroup of $\langle G \rangle$
- Baby-step giant-step
  - has expected complexity $\mathrm{ord}(G)/m$ point additions
  - ... with m the $\#$ points the attacker can store
- Pollard's ρ
  - has expected complexity $\sqrt{\mathrm{ord}(G)}$ point additions
  - ... and uses little memory
- Latter two force us to $\langle G \rangle$ with $\mathrm{ord}(G) \geq 2^{256}$ for 128-bit security