On the Use of the Negation Map in the Pollard Rho Method

Joppe W. Bos   Thorsten Kleinjung   Arjen K. Lenstra

Laboratory for Cryptologic Algorithms
EPFL, Station 14, CH-1015 Lausanne, Switzerland

Motivation

Study the negation map in practice when solving the elliptic curve discrete logarithm problem over prime fields.

Cryptography

- The Suite B Cryptography by the NSA allows elliptic curves over prime fields only.
- Solve ECDLPs fast $\rightarrow$ break ECC-based schemes.

Using the (parallelized) Pollard ρ method

- 79-, 89-, 97- and 109-bit (2000) prime field Certicom challenges
- the recent (2009) 112-bit prime field ECDLP

have been solved.

Textbook optimization: negation map ($\sqrt{2}$ speed-up)
(not used in any of the prime ECDLP records)

Preliminaries

The Elliptic Curve Discrete Logarithm Problem

Let p be an odd prime and $E(\mathbf{F}_p)$ an elliptic curve over $\mathbf{F}_p$. Given $\mathfrak{g} \in E(\mathbf{F}_p)$ of prime order q and $\mathfrak{h} \in \langle \mathfrak{g} \rangle$ find $m \in \mathbf{Z}$ such that $m\mathfrak{g} = \mathfrak{h}$.

Believed to be a hard problem (of order $\sqrt{q}$).
Algorithms to solve ECDLP:

Baby-step Giant-step, Pollard ρ, Pollard Kangaroo

Basic Idea

Pick random objects: $u\mathfrak{g} + v\mathfrak{h} \in \langle \mathfrak{g} \rangle$ $(u, v \in \mathbf{Z})$
Find duplicate / collision: $u\mathfrak{g} + v\mathfrak{h} = \bar{u}\mathfrak{g} + \bar{v}\mathfrak{h}$.
If $\bar{v} \not\equiv v \bmod q$, $m = \frac{u-\bar{u}}{\bar{v}-v} \bmod q$ solves the discrete logarithm problem.
Expected number of random objects: $\sqrt{\pi q/2}$

Pollard ρ, [Pollard-78]

Approximate random walk in $\langle \mathfrak{g} \rangle$.

Index function $\ell : \langle \mathfrak{g} \rangle = \mathfrak{G}_0 \cup \ldots \cup \mathfrak{G}_{t-1} \mapsto [0, t-1]$

$$\mathfrak{G}_i = \{\mathfrak{x} : \mathfrak{x} \in \langle \mathfrak{g} \rangle, \ell(\mathfrak{x}) = i\}, \qquad |\mathfrak{G}_i| \approx \frac{q}{t}$$

Precomputed partition constants: $\mathfrak{f}_0, \ldots, \mathfrak{f}_{t-1} \in \langle \mathfrak{g} \rangle$

With $\mathfrak{f}_i = u_i \mathfrak{g} + v_i \mathfrak{h}$.

r-**adding walk**	$r + s$-**mixed walk**
$t = r$	$t = r + s$
$\mathfrak{p}_{i+1} = \mathfrak{p}_i + \mathfrak{f}_{\ell(\mathfrak{p}_i)}$	$\mathfrak{p}_{i+1} = \begin{cases} \mathfrak{p}_i + \mathfrak{f}_{\ell(\mathfrak{p}_i)}, & \text{if } 0 \leq \ell(\mathfrak{p}_i) < r \\ 2\mathfrak{p}_i, & \text{if } \ell(\mathfrak{p}_i) \geq r \end{cases}$

[Teske-01]: r=20 performance close to a random walk.

The Negation Map

[Wiener,Zuccherato-98]

Equivalence relation $\sim$ on $\langle \mathfrak{g} \rangle$ by $\mathfrak{p} \sim -\mathfrak{p}$ for $\mathfrak{p} \in \langle \mathfrak{g} \rangle$.

Instead of searching $\langle \mathfrak{g} \rangle$ of size q search $\langle \mathfrak{g} \rangle / \sim$ of size about $\frac{q}{2}$ for collisions.

Advantage: Reduces the number of steps by a factor of $\sqrt{2}$.
Efficient to compute: Given $(x, y) \in \langle \mathfrak{g} \rangle \to -(x, y) = (x, -y)$

[Duursma,Gaudry,Morain-99],[Gallant,Lambert,Vanstone-00]

For Koblitz curves the Frobenius automorphism of a degree t binary extension field leads to a further $\sqrt{t}$-fold speedup.

Negation Map, Side-Effects

Well-known disadvantage: as presented no solution to large ECDLPs

Negation Map, Side-Effects

Well-known disadvantage: fruitless cycles

$$\mathfrak{p} \xrightarrow{(i,-)} -(\mathfrak{p} + \mathfrak{f}_i) \xrightarrow{(i,-)} \mathfrak{p}.$$

At any step in the walk the probability to enter a fruitless 2-cycle is $\frac{1}{2r}$ [Duursma,Gaudry,Morain-99] (Proposition 31)

Negation Map, Side-Effects

Well-known disadvantage: fruitless cycles

$$\mathfrak{p} \xrightarrow{(i,-)} -(\mathfrak{p} + \mathfrak{f}_i) \xrightarrow{(i,-)} \mathfrak{p}.$$

At any step in the walk the probability to enter a fruitless 2-cycle is $\frac{1}{2r}$ [Duursma,Gaudry,Morain-99] (Proposition 31)

2-cycle reduction technique: [Wiener,Zuccherato-98]

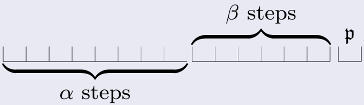
$$f(\mathfrak{p}) = \begin{cases} E(\mathfrak{p}) & \text{if } j = \ell(\sim(\mathfrak{p} + \mathfrak{f}_j)) \text{ for } 0 \leq j < r \\ \sim(\mathfrak{p} + \mathfrak{f}_i) & \text{with } i \geq \ell(\mathfrak{p}) \text{ minimal s.t. } \ell(\sim(\mathfrak{p} + \mathfrak{f}_i)) \neq i \bmod r. \end{cases}$$

once every r^r steps: $E : \langle \mathfrak{g} \rangle \to \langle \mathfrak{g} \rangle$ may restart the walk

Cost increase $c = \sum_{i=0}^{r} \frac{1}{r^i}$ with $1 + \frac{1}{r} \leq c \leq 1 + \frac{1}{r-1}$.

Dealing With Fruitless Cycles In General [Gallant,Lambert,Vanstone-00]

Cycle detection



Compare $\mathfrak{p}$ to all β points. Detect cycles of length $\leq \beta$.

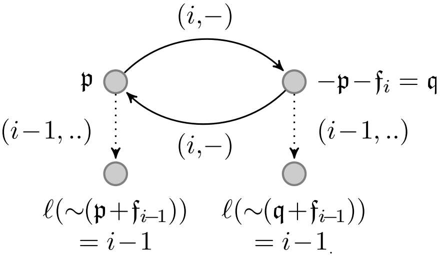
Cycle Escaping

Add

- $\mathfrak{f}_{\ell(\mathfrak{p})+c}$ for a fixed $c \in \mathbf{Z}$
- a precomputed value $\mathfrak{f}'$
- $\mathfrak{f}''_{\ell(\mathfrak{p})}$ from a distinct list of r precomputed values $\mathfrak{f}''_0, \mathfrak{f}''_1, \ldots, \mathfrak{f}''_{r-1}$

to a representative element of this cycle.

2-cycles When Using The 2-cycle Reduction Technique



Lemma

The probability to enter a fruitless 2-cycle when looking ahead to reduce 2-cycles while using an r-adding walk is

$$\frac{1}{2r}\left(\sum_{i=1}^{r-1}\frac{1}{r^i}\right)^2 = \frac{(r^{r-1}-1)^2}{2r^{2r-1}(r-1)^2} = \frac{1}{2r^3} + O\left(\frac{1}{r^4}\right).$$

4-cycle Reduction

$$\mathfrak{p} \xrightarrow{(i,+)} \mathfrak{p} + \mathfrak{f}_i \xrightarrow{(j,-)} -\mathfrak{p} - \mathfrak{f}_i - \mathfrak{f}_j \xrightarrow{(i,+)} -\mathfrak{p} - \mathfrak{f}_j \xrightarrow{(j,-)} \mathfrak{p}.$$

Fruitless 4-cycle starts with probability $\frac{r-1}{4r^3}$.

4-cycle Reduction

$$\mathfrak{p} \xrightarrow{(i,+)} \mathfrak{p}+\mathfrak{f}_i \xrightarrow{(j,-)} -\mathfrak{p}-\mathfrak{f}_i-\mathfrak{f}_j \xrightarrow{(i,+)} -\mathfrak{p}-\mathfrak{f}_j \xrightarrow{(j,-)} \mathfrak{p}.$$

Fruitless 4-cycle starts with probability $\frac{r-1}{4r^3}$.
Extend the 2-cycle reduction method to reduce 4-cycles:

$$g(\mathfrak{p}) = \begin{cases} E(\mathfrak{p}) & \text{if } j \in \{\ell(\mathfrak{q}), \ell(\sim(\mathfrak{q}+\mathfrak{f}_{\ell(\mathfrak{q})}))\} \text{ or } \ell(\mathfrak{q}) = \ell(\sim(\mathfrak{q}+\mathfrak{f}_{\ell(\mathfrak{q})})) \\ & \text{where } \mathfrak{q} = \sim(\mathfrak{p}+\mathfrak{f}_j), \text{ for } 0 \le j < r, \\ \mathfrak{q} = \sim(\mathfrak{p}+\mathfrak{f}_i) & \text{with } i \ge \ell(\mathfrak{p}) \text{ minimal s.t.} \\ & i \bmod r \ne \ell(\mathfrak{q}) \ne \ell(\sim(\mathfrak{q}+\mathfrak{f}_{\ell(\mathfrak{q})})) \ne i \bmod r. \end{cases}$$

Disadvantage: more expensive iteration function: $\geq \frac{r+4}{r}$

Advantage: positive effect of $\sqrt{\frac{r-1}{r}}$ since

$$\text{image}(g) \subset \langle \mathfrak{g} \rangle \text{ with } |\text{image}(g)| \approx \frac{r-1}{r} |\langle \mathfrak{g} \rangle|.$$

Example: 4-cycle With 4-cycle reduction

$$
\begin{tikzcd}[row sep=large, column sep=huge]
\ell(\sim(\tilde{\mathfrak{p}} + \mathfrak{f}_k)) \in \{i,k\} & \ell(\sim(\tilde{\mathfrak{q}} + \mathfrak{f}_n) \in \{j,n\} \\
\tilde{\mathfrak{p}} = \sim(\mathfrak{p} + \mathfrak{f}_i) \arrow[u, dotted, "{(k,..)}"] & \sim(-\mathfrak{p} - \mathfrak{f}_{j+1} + \mathfrak{f}_j) = \tilde{\mathfrak{q}} \arrow[u, dotted, "{(n,..)}"'] \\
\mathfrak{p} \arrow[u, dotted, "{(i,..)}"] \arrow[d, "{(i+1,+)}"'] & -\mathfrak{p} - \mathfrak{f}_{j+1} \arrow[u, dotted, "{(j,..)}"'] \arrow[l, "{(j+1,-)}"'] \\
\mathfrak{p} + \mathfrak{f}_{i+1} \arrow[r, "{(j+1,-)}"'] \arrow[d, dotted, "{(j,..)}"'] & -\mathfrak{p} - \mathfrak{f}_{i+1} - \mathfrak{f}_{j+1} \arrow[u, "{(i+1,+)}"'] \arrow[d, dotted, "{(i,..)}"] \\
\bar{\mathfrak{p}} = \sim(\mathfrak{p} + \mathfrak{f}_{i+1} + \mathfrak{f}_j) \arrow[d, dotted, "{(l,..)}"'] & \sim(-\mathfrak{p} - \mathfrak{f}_{i+1} - \mathfrak{f}_{j+1} + \mathfrak{f}_i) = \bar{\mathfrak{q}} \arrow[d, dotted, "{(m,..)}"] \\
\ell(\sim(\bar{\mathfrak{p}} + \mathfrak{f}_l)) \in \{j,l\} & \ell(\sim(\bar{\mathfrak{q}} + \mathfrak{f}_m)) \in \{i,m\}
\end{tikzcd}
$$

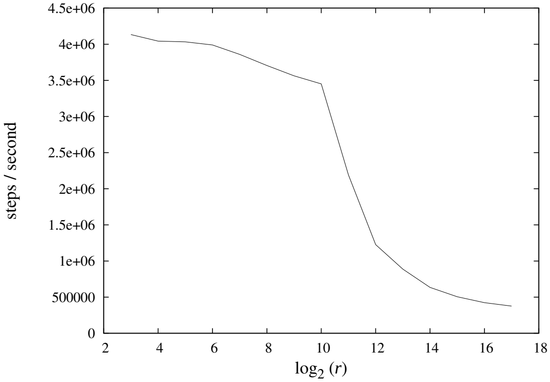
$$\frac{r-1}{4r^3} \text{ reduced to } \geq \frac{4(r-2)^4(r-1)}{r^{11}}$$

Large r-adding Walks

- Probability to enter cycle depends on the number of partitions r
- Why not simply increase r?

Large r-adding Walks

- Probability to enter cycle depends on the number of partitions r
- Why not simply increase r?



- Practical performance penalty (cache-misses)
- Fruitless cycles still occur

Recurring Cycles

Using

- r-adding walk with a medium sized r **and**
- { 2, 4 }-reduction technique **and**
- cycle escaping techniques

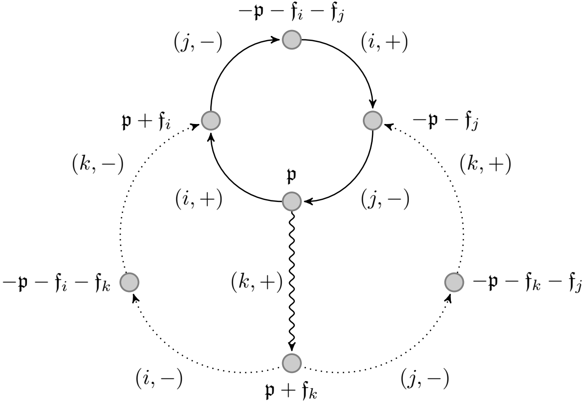
it is still very unlikely to solve any large ECDLP.

Recurring Cycles

Using

- r-adding walk with a medium sized r **and**
- { 2, 4 }-reduction technique **and**
- cycle escaping techniques

it is still very unlikely to solve any large ECDLP.



Dealing With Recurring Cycles

Reduce the number of fruitless (recurring) cycles by using a mixed-walk

- a cycle with at least one doubling is most likely not fruitless
- doublings are more expensive than additions

Use doublings to escape cycles, eliminates recurring cycles.

$$\bar{f}(\mathfrak{p}) = \begin{cases} \sim(\mathfrak{p} + \mathfrak{f}_{\ell(\mathfrak{p})}) & \text{if } \ell(\mathfrak{p}) \neq \ell(\sim(\mathfrak{p} + \mathfrak{f}_{\ell(\mathfrak{p})})), \\ \sim(2\mathfrak{p}) & \text{otherwise,} \end{cases}$$

$$\bar{g}(\mathfrak{p}) = \begin{cases} \mathfrak{q} = \sim(\mathfrak{p} + \mathfrak{f}_{\ell(\mathfrak{p})}) & \text{if } \ell(\mathfrak{q}) \neq \ell(\mathfrak{p}) \neq \ell(\sim(\mathfrak{q} + \mathfrak{f}_{\ell(\mathfrak{q})})) \neq \ell(\mathfrak{q}), \\ \sim(2\mathfrak{p}) & \text{otherwise.} \end{cases}$$

Experiments @ AMD Phenom 9500

	$r = 16$	$r = 32$	$r = 64$	$r = 128$	$r = 256$	$r = 512$
Without negation map						
	7.29: 0.98	7.28: 0.99	**7.27**: 1.00	7.19: 0.99	6.97: 0.96	6.78: 0.94
With negation map						
just g	0.00: 0.00	0.00: 0.00	0.00: 0.00	0.00: 0.00	0.04: 0.01	3.59: 0.70
just $\bar{\text{e}}$	3.34: 0.64	4.89: 0.95	5.85: 1.14	6.10: 1.19	6.28: 1.23	6.18: 1.21
f, e	0.00: 0.00	0.00: 0.00	1.52: 0.30	5.93: 1.16	6.47: 1.27	6.36: 1.25
f, $\bar{\text{e}}$	3.71: 0.72	6.36: 1.24	6.50: 1.27	6.57: **1.29**	6.47: 1.27	6.30: 1.25
g, e	0.00: 0.00	0.01: 0.00	4.89: 0.96	6.22: 1.22	6.23: 1.22	6.05: 1.19
g, $\bar{\text{e}}$	0.76: 0.15	5.91: 1.17	6.02: 1.18	6.25: 1.23	6.13: 1.20	6.00: 1.18

Conclusions

Using the negation map optimization technique for solving prime ECDLPs is useful in practice when

- { 2, 4 }-cycle reduction techniques are used
- recurring cycles are avoided; e.g. escaping by doubling
- medium sized r-adding walk ($r = 128$) are used

Using all this we managed to get a speedup of at most:

$$\mathbf{1.29} < \sqrt{2}\ (\approx 1.41)$$

More details and experiments in the article.

Future Work

Better cycle reduction or escaping techniques?
Faster implementations?
Can we do better than 1.29 speedup?