Implementing a Parallel Pollard Rho Attack on ECC

Adrian Escott

BT Laboratories

adrian.escott@bt-sys.bt.co.uk

Overview

- Background
- Parallel Pollard Rho
- Improving the Iteration Function
- Implementation
- Further Work
  - Exploiting the Inverse Point
  - Multiple Logarithms

Elliptic Curve Discrete Log Problem (ECDLP)

The set of points of an Elliptic Curve together with the point at infinity form a group.

Given an elliptic curve E, a point P on the curve of prime order n and a point $Q = dP$ (where $1 \leq d \leq n-1$), find the integer d.

Attacks on ECDLP

- Special Attacks
- Group Structure
  - Pohlig-Hellman, relies on a smooth factorization of the group order.
- Square Root Methods
  - Baby-Step Giant-Step Method
  - Parallel Pollard rho-method

Parallel Pollard Rho Method

Basic idea: Find a pair of integers (a_1, b_1) and (a_2, b_2) such that

$$a_1 P + b_1 Q \;=\; a_2 P + b_2 Q$$

Then

$$(a_1 - a_2)P \;=\; (b_2 - b_1)Q$$

Hence

$$(a_1 - a_2) \;\equiv\; (b_2 - b_1)d \pmod{n}$$

Therefore

$$d \;=\; (a_1 - a_2)(b_2 - b_1)^{-1} \bmod n$$

Pollard Rho Method

Pollard 1978

Generate (a_0, b_0) randomly.

Let $X_0 = a_0 P + b_0 Q$ and

$$X_{i+1} \;=\; \begin{cases} X_i + P & X_i \in S_1 \\ 2X_i & X_i \in S_2 \\ X_i + Q & X_i \in S_3 \end{cases}$$

for $i \geq 1$. Record a_i and b_i such that $X_i = a_i P + b_i Q$.

Repeatedly calculate $(X_i, a_i, b_i, X_{2i}, a_{2i}, b_{2i})$ for $i = 1, 2, \ldots$ until $X_i = X_{2i}$.

Use $a_i P + b_i Q = a_{2i} P + b_{2i} Q$ to get solution.

Parallel Pollard Rho Method

van Oorschot/Wiener

Select a set of D distinguished points.

Each processor generates a sequence of X_is starting from random a point until it finds a distiguished point, i.e $X_i \in D$.

The processor then submits (X_i, a_i, b_i) to the central server and starts on a new trial.

The central server checks to see if this point has been previously submitted.

If it has it uses the two pairs (a_i, b_i) to calculate the solution. Otherwise it stores (X_i, a_i, b_i) in a database.

Running Times

Let t be the time for one iteration

Single Pollard Rho

$$\text{Running time} = \left(\sqrt{\frac{\pi n}{2}}\right) t$$

Parallel Pollard Rho
Given m (equal speed) processors and let θ be the proportion of distinguished points.

$$\text{Running time} = \left(\frac{\sqrt{\frac{\pi n}{2}}}{m} + \frac{1}{\theta}\right) t$$

Also need storage for $O\left(\theta\sqrt{\pi n/2}\right)$ distinguished points.

Increasing the Number of Iterators

	Curve 1	Curve 2
Theory	2079059	3161683
Original	2749374	4266805
4 Iterators	2511313	3696255
8 Iterators	2247775	3413604
16 Iterators	2105388	3352526
32 Iterators	2082738	3156012

Different Iterating Functions

- Either
  - AB of the form add $aP + bQ$.
  - A of the form add aP.
  - and a possible double, i.e. $2X$.

  with the total number of iterators
- E.g. AB16 means 16 add iterators of the form $aP + bQ$, AD8 means 7 add iterators of the form aP and a double.

Different Iterating Functions

	Curve 1	Curve 2
AB16	2179500	3209512
ABD16	2150450	3205342
A16	2197391	3302869
AD16	2153128	3278840

Distribution of the Number of Iterations

Factor	0.5	0.75	1.0	1.25
Theory	17.8	35.7	54.4	70.7
Average	16.7	33.7	51.7	68.3

1.5	1.75	2.0	2.25	2.5
82.9	91.0	95.7	98.1	99.3
80.6	89.0	94.5	97.6	99.0

Certicom Challenges

- Why?
  - motivate research into ECDLP.
- What?
  - Three types of curve
    1. over F_p, with p large prime.
    2. over F_{2^m}.
    3. Koblitz curves over F_{2^m}.
  - Various bit sizes:- 79,89,97,109,...
- http://www.certicom.com/chal/

Challenges Broken So Far

Challenge	Group	Date	Group OPs
ECCp-79	INRIA	Dec 6 97	1.4×10^{12}
	BT	Dec 7 97	4.8×10^{11}
ECC2-79	INRIA	Dec 16 97	1.7×10^{12}
ECCp-89	INRIA/BT	Jan 12 98	3.0×10^{13}
ECC2-89	INRIA	Feb 9 98	1.8×10^{13}
ECCp-97	BT/INRIA	Mar 18 98	2.0×10^{14}
ECC2K-95	INRIA	May 21 98	2.2×10^{13}

Running a Distributed Attack

- Data passed only from client to server.
- Impossible to fake.
- Clients require little data to run.

P-97 Server

- $2\times$PII 300 MHz running NT, 512M Ram and using a 2G partition for storing the distinguished points.
- It was someone's desktop machine and was also running a client on each processor.
- Performance of the machine unaffected as distinguished points were only arriving every 3 seconds.

P-97 Client

- Ran several paths in parallel.
- Stores result if fails to get connection, then submits the point next time.
- Ran in low priority mode.
- Windows client had GUI and local statistics.

"Soft" Side of Problem

- Getting people to run client
  - easy to install/uninstall.
  - runs in background on low priority.
  - support.
  - regularly updated statistics for individuals and teams.
  - managed to keep 2/3 machines running.

Data form P-97 attack

- Overall machines: over 1200
- Peak machines: 850
- Iterations: 200,000,000,000,000
- Peak Rate: 5,000,000,000,000 iters/day.
- Finished after roughly 45% of expected.
- Expected time at peak rate: 85 - 90 days.

Machine Data I

- Breakdown of work done by machine:
  - 55% by 222 Alphas.
  - 39% by 753 Pentiums.
  - rest by Sparcs, Macs, HPs and others
- Breakdown of work done by location:
  - 70% by BT.
  - 8% by Digital.
  - 8% By INRIA.

Machine Data II

- 600MHz 21164 Alpha
  - 440,000 iters/sec.
  - need 130 to make overall rate.
- Pentium II 300 MHz
  - 125,000 iters/sec.
  - need 460 to make overall rate.

Further Challenges

- P-109
  - Current infrastructure could deal with this challenge.
  - need 12,000 Pentium Pro 200 MHz for 1 year to do the expected number of iterations (assuming the inverse point is fully exploited and the code is 20% slower than P-97).
  - **distributed.net** claim to be running at the rate of roughly 60,000 Pentium Pro 200.

Exploiting the Inverse Point

Suppose we have

$$X = a_1 P + b_1 Q$$

and

$$X^{-1} = a_2 P + b_2 Q$$

Rearranging this gives

$$X = (n - a_2)P + (n - b_2)Q$$

The first and third equation can be used to calculate the answer.

To fully exploit this we need an iteration function that

$$\begin{aligned} X &\rightarrow D \\ X^{-1} &\rightarrow D^{-1} \end{aligned}$$

Exploiting the Inverse Point

Adding

$$X + A = (X^{-1} + A^{-1})^{-1}$$

Doubling

$$2X = (2X^{-1})^{-1}$$

Problem:

$$\begin{aligned} X &\rightarrow (X + A) \\ (X + A) &\rightarrow (X + A) + A^{-1} = X \end{aligned}$$

Hence it is possible to end up in a useless cycle.

Detecting Small Useless Cycles

Use iteration function AD16 (i.e. 15 adders of the form aP and a double).

Assumption: No small useless cycle includes a double.

Calculate the new a_i at each iteration.

If this is different to all previous ones then calculate the point, store it and continue.

If not we are in a small cycle.

When a double iteration is used, clear the list of stored points.

Exiting Small Useless Cycles

Sum the distinct point in the small cycle to exit.

This ensures that wherever the cycle was entered it is left in the same away and no collisions are lost.

Exploiting Inverse Point

Iteration	Curve 1		Curve2	
Function	Iterations	Waste	Iterations	Waste
Theory	1470117	N/A	2235647	N/A
A7D1	1604855	0.744	2446993	0.728
A15D1	1518272	0.208	2330377	0.203
A30D2	1478027	0.049	2259059	0.048
A60D4	1471242	0.012	2335387	0.012
A120D8	1466369	0.003	2291104	0.003

Iteration speed is currently 6% slower.

Multiple Logarithms

Suggested by Silverman/Stapleton 1997

Suppose you wish to calculate $d_1, d_2, \ldots$ where $Q_1 = d_1 P$, $Q_2 = d_2 P$, etc.

Choose iteration function of the form AD, so it is independent of all Q_i.

Find d_1 using the normal parallel Pollard rho method saving the distinguished points.

Then find d_2 using the parallel Pollard rho method. We now have two types of collision.

Ordinary

$$a_1 P + b_1 Q_2 = a_2 P + b_2 Q_2$$

New

$$a_1 P + b_1 Q_1 = a_2 P + b_2 Q_2$$

It is easy to solve for d_2 in both cases.

Multiple Logarithms

Let t be the expected time to solve for d_1.

The expected time to solve for d_2 is $(\sqrt{2}-1)t$.

The expected time to solve for d_3 is $(\sqrt{3}-\sqrt{2})t$.

The expected time to find $d_1 \dots d_k$ is $t\sqrt{k}$.

Hence several logarithms over the same curve are not significantly more secure, than just one.

Multiple Logarithms

Log	Curve 1		Curve2	
	Iterations	Multiple	Iterations	Mutliple
Theory	1470117	N/A	2235647	N/A
1	1479752	1.01	2252691	1.01
2	2241090	1.52	3387977	1.52
3	2799904	1.90	4177655	1.87
4	3269120	2.22	4866307	2.18
5	3654872	2.49	5491087	2.46

Conclusions

- Practice fits well with theory of parallel Pollard rho.
- P-109 achievable by internet wide effort.