Kevin D'Souza

The bounded height discrete logarithm problem

Simon R. Blackburn

Joint work with Sam Scott

June 2014

Discrete logarithm problem (DLP)

The Discrete Logarithm Problem

Let G be a group and let $g, h \in G$.
Find an integer i (if it exists) such that $h = g^i$.

Discrete logarithm problem (DLP)

The Discrete Logarithm Problem

Let G be a group and let $g, h \in G$.
Find an integer i (if it exists) such that $h = g^i$.

Often $G = \mathbb{Z}_p$, or G is the points on an elliptic curve.

Discrete logarithm problem (DLP)

The Discrete Logarithm Problem

Let G be a group and let $g, h \in G$.
Find an integer i (if it exists) such that $h = g^i$.

Often $G = \mathbb{Z}_p$, or G is the points on an elliptic curve.

In general, this is a hard computational problem.

Discrete logarithm problem (DLP)

The Discrete Logarithm Problem

Let G be a group and let $g, h \in G$.
Find an integer i (if it exists) such that $h = g^i$.

Often $G = \mathbb{Z}_p$, or G is the points on an elliptic curve.

In general, this is a hard computational problem.

Example: Let $G = \mathbb{Z}_{11}$, $g = 2$ and $h = 9$. Solve the DLP.

Discrete logarithm problem (DLP)

The Discrete Logarithm Problem

Let G be a group and let $g, h \in G$.
Find an integer i (if it exists) such that $h = g^i$.

Often $G = \mathbb{Z}_p$, or G is the points on an elliptic curve.

In general, this is a hard computational problem.

Example: Let $G = \mathbb{Z}_{11}$, $g = 2$ and $h = 9$. Solve the DLP.

i	0	1	2	3	4	5	6	7	8	9
g^i	1	2	4	8	5	10	9	7	3	6

Discrete logarithm problem (DLP)

The Discrete Logarithm Problem

Let G be a group and let $g, h \in G$.
Find an integer i (if it exists) such that $h = g^i$.

Often $G = \mathbb{Z}_p$, or G is the points on an elliptic curve.

In general, this is a hard computational problem.

Example: Let $G = \mathbb{Z}_{11}$, $g = 2$ and $h = 9$. Solve the DLP.

i	0	1	2	3	4	5	6	7	8	9
g^i	1	2	4	8	5	10	9	7	3	6

If g has order n, naive search takes n steps.

Motivation: Diffie-Hellman key exchange

Suppose Alice and Bob want to agree on a random key K.
They decide upon a large prime p and some $g \in \mathbb{Z}_p^*$ and perform the following protocol:

Motivation: Diffie-Hellman key exchange

Suppose Alice and Bob want to agree on a random key K.
They decide upon a large prime p and some $g \in \mathbb{Z}_p^*$ and perform the following protocol:

- Alice chooses a random integer $1 \leq a < p-1$ and sends $c_1 = g^a \mod p$ to Bob.

Motivation: Diffie-Hellman key exchange

Suppose Alice and Bob want to agree on a random key K.
They decide upon a large prime p and some $g \in \mathbb{Z}_p^*$ and perform the following protocol:

- Alice chooses a random integer $1 \leq a < p-1$ and sends $c_1 = g^a \mod p$ to Bob.
- Bob chooses a random integer $1 \leq b < p-1$ and sends $c_2 = g^b \mod p$ to Alice.

Motivation: Diffie-Hellman key exchange

Suppose Alice and Bob want to agree on a random key K.
They decide upon a large prime p and some $g \in \mathbb{Z}_p^*$ and perform the following protocol:

- Alice chooses a random integer $1 \leq a < p-1$ and sends $c_1 = g^a \mod p$ to Bob.
- Bob chooses a random integer $1 \leq b < p-1$ and sends $c_2 = g^b \mod p$ to Alice.

The common key is $K = g^{ab} \mod p$.

Motivation: Diffie-Hellman key exchange

Suppose Alice and Bob want to agree on a random key K.
They decide upon a large prime p and some $g \in \mathbb{Z}_p^*$ and perform the following protocol:

- Alice chooses a random integer $1 \leq a < p-1$ and sends $c_1 = g^a \mod p$ to Bob.
- Bob chooses a random integer $1 \leq b < p-1$ and sends $c_2 = g^b \mod p$ to Alice.
- On receiving c_2 Alice computes $K = c_2^a \mod p$.

The common key is $K = g^{ab} \mod p$.

Motivation: Diffie-Hellman key exchange

Suppose Alice and Bob want to agree on a random key K.
They decide upon a large prime p and some $g \in \mathbb{Z}_p^*$ and perform the following protocol:

- Alice chooses a random integer $1 \leq a < p-1$ and sends $c_1 = g^a \mod p$ to Bob.
- Bob chooses a random integer $1 \leq b < p-1$ and sends $c_2 = g^b \mod p$ to Alice.
- On receiving c_2 Alice computes $K = c_2^a \mod p$.
- On receiving c_1 Bob computes $K = c_1^b \mod p$.

The common key is $K = g^{ab} \mod p$.

Motivation: Group theory

- Given matrices A and B (over $\mathbb{F}_q$), is $\langle A, B \rangle$ cyclic?

Motivation: Group theory

- Given matrices A and B (over $\mathbb{F}_q$), is $\langle A, B \rangle$ cyclic?
- A generic algorithm for finding the order n of an element g.

Motivation: Group theory

- Given matrices A and B (over $\mathbb{F}_q$), is $\langle A, B \rangle$ cyclic?
- A generic algorithm for finding the order n of an element g.
  - Calculate $h = g^r$, where r is random.

Motivation: Group theory

- Given matrices A and B (over $\mathbb{F}_q$), is $\langle A, B \rangle$ cyclic?
- A generic algorithm for finding the order n of an element g.
  - Calculate $h = g^r$, where r is random.
  - Solve DLP to get $h = g^i$.

Motivation: Group theory

- Given matrices A and B (over $\mathbb{F}_q$), is $\langle A, B \rangle$ cyclic?
- A generic algorithm for finding the order n of an element g.
  - Calculate $h = g^r$, where r is random.
  - Solve DLP to get $h = g^i$.
  - Then $r - i$ is a multiple of n.

Motivation: Group theory

- Given matrices A and B (over $\mathbb{F}_q$), is $\langle A, B \rangle$ cyclic?
- A generic algorithm for finding the order n of an element g.
  - Calculate $h = g^r$, where r is random.
  - Solve DLP to get $h = g^i$.
  - Then $r - i$ is a multiple of n.
- Finding a transvection (element of order p) in $\mathrm{SL}(2, q)$, for classical group recognition.

Baby-step giant-step algorithm: a $\sqrt{n}$ method

Example: $G = \mathbb{Z}_{101}^*$, $g = 2$, $h = 40$. Find i so that $g^i = h$.

Baby-step giant-step algorithm: a $\sqrt{n}$ method

Example: $G = \mathbb{Z}_{101}^*$, $g = 2$, $h = 40$. Find i so that $g^i = h$.

Stage 1 (Baby steps) Compute small powers of 2:

i	0	1	2	3	4	5	6	7	8	9
g^i	1	2	4	8	16	32	64	27	54	7

Baby-step giant-step algorithm: a $\sqrt{n}$ method

Example: $G = \mathbb{Z}_{101}^*$, $g = 2$, $h = 40$. Find i so that $g^i = h$.

Stage 1 (Baby steps) Compute small powers of 2:

i	0	1	2	3	4	5	6	7	8	9
g^i	1	2	4	8	16	32	64	27	54	7

and sort:

i	0	1	2	9	3	4	7	5	8	6
g^i	1	2	4	7	8	16	27	32	54	64

Baby-step giant-step algorithm: a $\sqrt{n}$ method

Example: $G = \mathbb{Z}_{101}^*$, $g = 2$, $h = 40$. Find i so that $g^i = h$.

Stage 1 (Baby steps) Compute small powers of 2:

i	0	1	2	3	4	5	6	7	8	9
g^i	1	2	4	8	16	32	64	27	54	7

and sort:

i	0	1	2	9	3	4	7	5	8	6
g^i	1	2	4	7	8	16	27	32	54	64

Stage 2 (Giant steps) Compute $g^{-10} = 65$.

Baby-step giant-step algorithm: a $\sqrt{n}$ method

Example: $G = \mathbb{Z}_{101}^*$, $g = 2$, $h = 40$. Find i so that $g^i = h$.

Stage 1 (Baby steps) Compute small powers of 2:

i	0	1	2	3	4	5	6	7	8	9
g^i	1	2	4	8	16	32	64	27	54	7

and sort:

i	0	1	2	9	3	4	7	5	8	6
g^i	1	2	4	7	8	16	27	32	54	64

Stage 2 (Giant steps) Compute $g^{-10} = 65$.
Is h in table?

Baby-step giant-step algorithm: a $\sqrt{n}$ method

Example: $G = \mathbb{Z}_{101}^*$, $g = 2$, $h = 40$. Find i so that $g^i = h$.

Stage 1 (Baby steps) Compute small powers of 2:

i	0	1	2	3	4	5	6	7	8	9
g^i	1	2	4	8	16	32	64	27	54	7

and sort:

i	0	1	2	9	3	4	7	5	8	6
g^i	1	2	4	7	8	16	27	32	54	64

Stage 2 (Giant steps) Compute $g^{-10} = 65$.
Is h in table? What about $hg^{-10} = 75$?

Baby-step giant-step algorithm: a $\sqrt{n}$ method

Example: $G = \mathbb{Z}_{101}^*$, $g = 2$, $h = 40$. Find i so that $g^i = h$.

Stage 1 (Baby steps) Compute small powers of 2:

i	0	1	2	3	4	5	6	7	8	9
g^i	1	2	4	8	16	32	64	27	54	7

and sort:

i	0	1	2	9	3	4	7	5	8	6
g^i	1	2	4	7	8	16	27	32	54	64

Stage 2 (Giant steps) Compute $g^{-10} = 65$.
Is h in table? What about $hg^{-10} = 75$? What about $hg^{-20} = 27$?

Baby-step giant-step algorithm: a $\sqrt{n}$ method

Example: $G = \mathbb{Z}_{101}^*$, $g = 2$, $h = 40$. Find i so that $g^i = h$.

Stage 1 (Baby steps) Compute small powers of 2:

i	0	1	2	3	4	5	6	7	8	9
g^i	1	2	4	8	16	32	64	27	54	7

and sort:

i	0	1	2	9	3	4	7	5	8	6
g^i	1	2	4	7	8	16	27	32	54	64

Stage 2 (Giant steps) Compute $g^{-10} = 65$.
Is h in table? What about $hg^{-10} = 75$? What about $hg^{-20} = 27$?

So $h = g^{20+7} = g^{27}$.

Other algorithms

- Baby-step giant-step: $\approx \sqrt{n}$ steps, but $\approx \sqrt{n}$ memory.

Other algorithms

- Baby-step giant-step: $\approx \sqrt{n}$ steps, but $\approx \sqrt{n}$ memory.
- Pollard-ρ algorithm: $\approx \sqrt{n}$ steps, but constant memory.

Other algorithms

- Baby-step giant-step: $\approx \sqrt{n}$ steps, but $\approx \sqrt{n}$ memory.
- Pollard-ρ algorithm: $\approx \sqrt{n}$ steps, but constant memory.
- For g, h elts of a finite field (or g, h matrices), there are better algorithms: number field sieve, index calculus.

Other algorithms

- Baby-step giant-step: $\approx \sqrt{n}$ steps, but $\approx \sqrt{n}$ memory.
- Pollard-ρ algorithm: $\approx \sqrt{n}$ steps, but constant memory.
- For g, h elts of a finite field (or g, h matrices), there are better algorithms: number field sieve, index calculus.
- The latter algorithms are sub-exponential, but much slower than polynomial.

Other algorithms

- Baby-step giant-step: $\approx \sqrt{n}$ steps, but $\approx \sqrt{n}$ memory.
- Pollard-ρ algorithm: $\approx \sqrt{n}$ steps, but constant memory.
- For g, h elts of a finite field (or g, h matrices), there are better algorithms: number field sieve, index calculus.
- The latter algorithms are sub-exponential, but much slower than polynomial.
- The latter algorithms do not work for elliptic curves (for example).

Small exponents

The Bounded Exponent Discrete Logarithm Problem

Let G be a group and let $g, h \in G$. Let w be an integer.
Find an integer i (if it exists) such that $h = g^i$ and $0 \leq i < w$.

Small exponents

The Bounded Exponent Discrete Logarithm Problem

Let G be a group and let $g, h \in G$. Let w be an integer.
Find an integer i (if it exists) such that $h = g^i$ and $0 \leq i < w$.

Here w is much smaller than the order n of g.

Small exponents

The Bounded Exponent Discrete Logarithm Problem

Let G be a group and let $g, h \in G$. Let w be an integer.
Find an integer i (if it exists) such that $h = g^i$ and $0 \leq i < w$.

Here w is much smaller than the order n of g.

Baby-step giant-step works: $\approx \sqrt{w}$ steps/memory.

Small exponents

The Bounded Exponent Discrete Logarithm Problem

Let G be a group and let $g, h \in G$. Let w be an integer.
Find an integer i (if it exists) such that $h = g^i$ and $0 \leq i < w$.

Here w is much smaller than the order n of g.

Baby-step giant-step works: $\approx \sqrt{w}$ steps/memory.
Pollard-ρ does not work, but Pollard-λ does: $\approx \sqrt{w}$ steps, constant memory.

Small exponents

The Bounded Exponent Discrete Logarithm Problem

Let G be a group and let $g, h \in G$. Let w be an integer.
Find an integer i (if it exists) such that $h = g^i$ and $0 \leq i < w$.

Here w is much smaller than the order n of g.

Baby-step giant-step works: $\approx \sqrt{w}$ steps/memory.
Pollard-ρ does not work, but Pollard-λ does: $\approx \sqrt{w}$ steps, constant memory.
An elliptic curve over $\mathbb{F}_q$ has order n where $|n - (q-1)| \leq 2\sqrt{q}$.

Small exponents

The Bounded Exponent Discrete Logarithm Problem

Let G be a group and let $g, h \in G$. Let w be an integer.
Find an integer i (if it exists) such that $h = g^i$ and $0 \leq i < w$.

Here w is much smaller than the order n of g.

Baby-step giant-step works: $\approx \sqrt{w}$ steps/memory.
Pollard-ρ does not work, but Pollard-λ does: $\approx \sqrt{w}$ steps, constant memory.
An elliptic curve over $\mathbb{F}_q$ has order n where $|n - (q-1)| \leq 2\sqrt{q}$. So

$$g^{q-1+2\sqrt{q}} = g^{q-1+2\sqrt{q}-n} = g^i \text{ with } 0 \leq i \leq 4\sqrt{q}.$$

The Pollard-λ algorithm

- Precomputation Decompose $G = T_1 \cup T_2 \cup T_3$. Choose exponents $u_1, u_2, u_3 \approx \sqrt{w}$.

The Pollard-λ algorithm

- Precomputation Decompose $G = T_1 \cup T_2 \cup T_3$. Choose exponents $u_1, u_2, u_3 \approx \sqrt{w}$.
- Define $f : G \to G$ by $f(x) = xg^{u_\ell}$ for $x \in T_\ell$.

The Pollard-λ algorithm

- Precomputation Decompose $G = T_1 \cup T_2 \cup T_3$. Choose exponents $u_1, u_2, u_3 \approx \sqrt{w}$.
- Define $f : G \to G$ by $f(x) = xg^{u_\ell}$ for $x \in T_\ell$.
- Pseudorandom walk Define $x_1 = g^{w/2}$, $a_1 = w/2$. When $x_i \in T_\ell$, let

$$x_{i+1} = f(x_i) \text{ and } a_{i+1} = a_i + u_\ell.$$

The Pollard-λ algorithm

- Precomputation Decompose $G = T_1 \cup T_2 \cup T_3$. Choose exponents $u_1, u_2, u_3 \approx \sqrt{w}$.
- Define $f : G \to G$ by $f(x) = xg^{u_\ell}$ for $x \in T_\ell$.
- Pseudorandom walk Define $x_1 = g^{w/2}$, $a_1 = w/2$. When $x_i \in T_\ell$, let

$$x_{i+1} = f(x_i) \text{ and } a_{i+1} = a_i + u_\ell.$$

- Dig a hole Compute and store $(y, a) = (x_{\sqrt{w}}, a_{\sqrt{w}})$.

The Pollard-λ algorithm

- Precomputation Decompose $G = T_1 \cup T_2 \cup T_3$. Choose exponents $u_1, u_2, u_3 \approx \sqrt{w}$.
- Define $f : G \to G$ by $f(x) = xg^{u_\ell}$ for $x \in T_\ell$.
- Pseudorandom walk Define $x_1 = g^{w/2}$, $a_1 = w/2$. When $x_i \in T_\ell$, let

$$x_{i+1} = f(x_i) \text{ and } a_{i+1} = a_i + u_\ell.$$

- Dig a hole Compute and store $(y, a) = (x_{\sqrt{w}}, a_{\sqrt{w}})$.
- Kangaroo walk Define $y_1 = h$, $b_1 = 0$. When $y_i \in T_\ell$, let

$$y_{i+1} = f(y_i) \text{ and } b_{i+1} = b_i + u_\ell.$$

The Pollard-λ algorithm

- Precomputation Decompose $G = T_1 \cup T_2 \cup T_3$. Choose exponents $u_1, u_2, u_3 \approx \sqrt{w}$.
- Define $f : G \to G$ by $f(x) = xg^{u_\ell}$ for $x \in T_\ell$.
- Pseudorandom walk Define $x_1 = g^{w/2}$, $a_1 = w/2$. When $x_i \in T_\ell$, let

$$x_{i+1} = f(x_i) \text{ and } a_{i+1} = a_i + u_\ell.$$

- Dig a hole Compute and store $(y, a) = (x_{\sqrt{w}}, a_{\sqrt{w}})$.
- Kangaroo walk Define $y_1 = h$, $b_1 = 0$. When $y_i \in T_\ell$, let

$$y_{i+1} = f(y_i) \text{ and } b_{i+1} = b_i + u_\ell.$$

- Catch the kangaroo Compute (y_i, b_i) until $y_i = x$. Return $i = a - b_i$.

Small height

The height of a rational number a/b (with a and b coprime) is $\max\{|a|, |b|\}$.

Small height

The height of a rational number a/b (with a and b coprime) is $\max\{|a|, |b|\}$.

Note that $g^{a/b}$ makes sense if $\gcd(b, \mathrm{ord}(g)) = 1$.

Small height

The height of a rational number a/b (with a and b coprime) is $\max\{|a|, |b|\}$.

Note that $g^{a/b}$ makes sense if $\gcd(b, \mathrm{ord}(g)) = 1$.

Note that $g^{a/b} = h$ if and only if $g^a = h^b$.

Small height

The height of a rational number a/b (with a and b coprime) is $\max\{|a|, |b|\}$.

Note that $g^{a/b}$ makes sense if $\gcd(b, \mathrm{ord}(g)) = 1$.

Note that $g^{a/b} = h$ if and only if $g^a = h^b$.

The Bounded Height Discrete Logarithm Problem

Let G be a group and let $g, h \in G$. Let w be an integer.
Find positive integers a, b (if they exist) such that $g^a = h^b$ and $0 \leq a, b < w$.

Small height

The height of a rational number a/b (with a and b coprime) is $\max\{|a|, |b|\}$.

Note that $g^{a/b}$ makes sense if $\gcd(b, \mathrm{ord}(g)) = 1$.

Note that $g^{a/b} = h$ if and only if $g^a = h^b$.

The Bounded Height Discrete Logarithm Problem

Let G be a group and let $g, h \in G$. Let w be an integer.
Find positive integers a, b (if they exist) such that $g^a = h^b$ and $0 \leq a, b < w$.

Naive search: $\approx w^2$ time. BSGS approach: $\approx w$ time and memory.

Motivation: The EMVCo protocol

- New protocol proposal for EMV card key establishment.

Motivation: The EMVCo protocol

- New protocol proposal for EMV card key establishment.
- Currently over 1.6 billion cards, over 23 million terminals.

Motivation: The EMVCo protocol

- New protocol proposal for EMV card key establishment.
- Currently over 1.6 billion cards, over 23 million terminals.
- Based on a 256-bit EC Diffie–Hellman protocol.

Motivation: The EMVCo protocol

- New protocol proposal for EMV card key establishment.
- Currently over 1.6 billion cards, over 23 million terminals.
- Based on a 256-bit EC Diffie–Hellman protocol.
- One goal is authentication (card to terminal). Another is privacy (no tracking).

Motivation: The EMVCo protocol

- New protocol proposal for EMV card key establishment.
- Currently over 1.6 billion cards, over 23 million terminals.
- Based on a 256-bit EC Diffie–Hellman protocol.
- One goal is authentication (card to terminal). Another is privacy (no tracking).
- Card sends $g = y^r$ where r is 32-bits, y a card-dependent group element.

Motivation: The EMVCo protocol

- New protocol proposal for EMV card key establishment.
- Currently over 1.6 billion cards, over 23 million terminals.
- Based on a 256-bit EC Diffie–Hellman protocol.
- One goal is authentication (card to terminal). Another is privacy (no tracking).
- Card sends $g = y^r$ where r is 32-bits, y a card-dependent group element.
- Card tracking: given g_1, g_2, are they produced by the same card?

Motivation: The EMVCo protocol

- New protocol proposal for EMV card key establishment.
- Currently over 1.6 billion cards, over 23 million terminals.
- Based on a 256-bit EC Diffie–Hellman protocol.
- One goal is authentication (card to terminal). Another is privacy (no tracking).
- Card sends $g = y^r$ where r is 32-bits, y a card-dependent group element.
- Card tracking: given g_1, g_2, are they produced by the same card?
- Does $g_1^{r_2} = g_2^{r_1}$?

The algorithm

Inspired by the Gaudry-Schost algorithm (for the multidimensional DLP).

The algorithm

Inspired by the Gaudry-Schost algorithm (for the multidimensional DLP).

Let $S = \{(i,j) : 0 \leq i,j \leq 2w\}$.

Define $\omega : S \to G$ by

$$\omega(i,j) = g^i h^j.$$

The algorithm

Inspired by the Gaudry-Schost algorithm (for the multidimensional DLP).

Let $S = \{(i,j) : 0 \leq i,j \leq 2w\}$.

Define $\omega : S \to G$ by

$$\omega(i,j) = g^i h^j.$$

If we find $(i,j),(i',j') \in S$ with $\omega(i,j) = \omega(i',j')$ then

$$g^{i-i'} = h^{j'-j}.$$

The basic idea

Decompose $G = D \cup T_1 \cup T_2 \cup T_3$ where $|D| \approx |G|(\log w)/w$.

The basic idea

Decompose $G = D \cup T_1 \cup T_2 \cup T_3$ where $|D| \approx |G|(\log w)/w$.

Choose $(u_1, v_1), (u_2, v_2), (u_3, v_3)$ where $|u_\ell|, |v_\ell| \approx w/100 \log w$.

The basic idea

Decompose $G = D \cup T_1 \cup T_2 \cup T_3$ where $|D| \approx |G|(\log w)/w$.

Choose $(u_1, v_1), (u_2, v_2), (u_3, v_3)$ where $|u_\ell|, |v_\ell| \approx w/100 \log w$.

Define $f : S \to S$ by

$$f(i, j) = (i + u_\ell, j + v_\ell)$$

when $\omega(i, j) \in T_\ell$.

The basic idea

Decompose $G = D \cup T_1 \cup T_2 \cup T_3$ where $|D| \approx |G|(\log w)/w$.

Choose $(u_1, v_1), (u_2, v_2), (u_3, v_3)$ where $|u_\ell|, |v_\ell| \approx w/100 \log w$.

Define $f : S \to S$ by

$$f(i,j) = (i + u_\ell, j + v_\ell)$$

when $\omega(i,j) \in T_\ell$.

Run pseudorandom walks from random points in S until $\omega(i,j) \in D$. Store $(\omega(i,j), (i,j))$.
Stop when we have stored $(x, (i,j))$ and $(x, (i',j'))$ with $(i,j) \neq (i',j')$.

Complexity

Theorem

If we model our pseudorandom walks as truely random on S, the algorithm terminates in $O(w)$ time, using little memory.

Complexity

Theorem

If we model our pseudorandom walks as truely random on S, the algorithm terminates in $O(w)$ time, using little memory.

The heuristic is not realistic.

Complexity

Theorem

If we model our pseudorandom walks as truely random on S, the algorithm terminates in $O(w)$ time, using little memory.

The heuristic is not realistic.

Open problem

Find a good algorithm with a reasonable heuristic. Or find a good heuristic for this algorithm.

Complexity

Theorem

If we model our pseudorandom walks as truely random on S, the algorithm terminates in $O(w)$ time, using little memory.

The heuristic is not realistic.

Open problem

Find a good algorithm with a reasonable heuristic. Or find a good heuristic for this algorithm.

The Gaudry–Schost algorithm has the same issues.

Some Links

This talk will appear soon on my home page:

`www.ma.rhul.ac.uk/sblackburn`

Simon R. Blackburn and Sam Scott, 'The discrete logarithm problem for exponents of bounded height', *LMS J. Comp. Math.*, to appear.

Steven D. Galbraith, *Mathematics of Public Key Cryptography*, CUP 2012.

Douglas R. Stinson, *Cryptography: Theory and Practice*, Chapman & Hall / CRC Press, 2005.