Generic attacks and index calculus

D. J. Bernstein
University of Illinois at Chicago

The discrete-logarithm problem

Define $p = 1000003$.
Easy to prove: p is prime.

Can we find an integer
$n \in \{1, 2, 3, \ldots, p-1\}$
such that $5^n \bmod p = 262682$?

Easy to prove: $n \mapsto 5^n \bmod p$
permutes $\{1, 2, 3, \ldots, p-1\}$.
So there *exists* an n
such that $5^n \bmod p = 262682$.

Could find n by brute force.
Is there a faster way?

attacks
ex calculus

ernstein
ty of Illinois at Chicago

The discrete-logarithm problem

Define $p = 1000003$.
Easy to prove: p is prime.

Can we find an integer
$n \in \{1, 2, 3, \ldots, p-1\}$
such that $5^n \bmod p = 262682$?

Easy to prove: $n \mapsto 5^n \bmod p$
permutes $\{1, 2, 3, \ldots, p-1\}$.
So there *exists* an n
such that $5^n \bmod p = 262682$.

Could find n by brute force.
Is there a faster way?

Typical

Imagine
in the D

User cho
publishe

Can atta
the discr
Given pu
quickly f

(Warnin
to attacl
Maybe t

is at Chicago

The discrete-logarithm problem

Define $p = 1000003$.
Easy to prove: p is prime.

Can we find an integer
$n \in \{1, 2, 3, \ldots, p - 1\}$
such that $5^n \bmod p = 262682$?

Easy to prove: $n \mapsto 5^n \bmod p$
permutes $\{1, 2, 3, \ldots, p - 1\}$.
So there *exists* an n
such that $5^n \bmod p = 262682$.

Could find n by brute force.
Is there a faster way?

Typical cryptanaly

Imagine standard
in the Diffie-Hellm

User chooses secre
publishes 5^n mod

Can attacker quick
the discrete-logarit
Given public key 5
quickly find secret

(Warning: This is
to attack the prot
Maybe there are b

ago

The discrete-logarithm problem

Define $p = 1000003$.
Easy to prove: p is prime.

Can we find an integer
$n \in \{1, 2, 3, \ldots, p - 1\}$
such that $5^n \bmod p = 262682$?

Easy to prove: $n \mapsto 5^n \bmod p$
permutes $\{1, 2, 3, \ldots, p - 1\}$.
So there *exists* an n
such that $5^n \bmod p = 262682$.

Could find n by brute force.
Is there a faster way?

Typical cryptanalytic applica

Imagine standard $p = 10000$
in the Diffie-Hellman protoc

User chooses secret key n,
publishes $5^n \bmod p = 26268$

Can attacker quickly solve
the discrete-logarithm proble
Given public key $5^n \bmod p$,
quickly find secret key n?

(Warning: This is *one* way
to attack the protocol.
Maybe there are better ways

The discrete-logarithm problem

Define $p = 1000003$.
Easy to prove: p is prime.

Can we find an integer
$n \in \{1, 2, 3, \ldots, p - 1\}$
such that $5^n \bmod p = 262682$?

Easy to prove: $n \mapsto 5^n \bmod p$
permutes $\{1, 2, 3, \ldots, p - 1\}$.
So there *exists* an n
such that $5^n \bmod p = 262682$.

Could find n by brute force.
Is there a faster way?

Typical cryptanalytic application:

Imagine standard $p = 1000003$
in the Diffie-Hellman protocol.

User chooses secret key n,
publishes $5^n \bmod p = 262682$.

Can attacker quickly solve
the discrete-logarithm problem?
Given public key $5^n \bmod p$,
quickly find secret key n?

(Warning: This is *one* way
to attack the protocol.
Maybe there are better ways.)

crete-logarithm problem

$p = 1000003$.

prove: p is prime.

find an integer
$2, 3, \ldots, p-1\}$
nt $5^n \bmod p = 262682$?

prove: $n \mapsto 5^n \bmod p$
s $\{1, 2, 3, \ldots, p-1\}$.
e *exists* an n
nt $5^n \bmod p = 262682$.

nd n by brute force.
a faster way?

Typical cryptanalytic application:

Imagine standard $p = 1000003$ in the Diffie-Hellman protocol.

User chooses secret key n, publishes $5^n \bmod p = 262682$.

Can attacker quickly solve the discrete-logarithm problem? Given public key $5^n \bmod p$, quickly find secret key n?

(Warning: This is *one* way to attack the protocol. Maybe there are better ways.)

Relation

1. Some
to ellipti
Use in e
security

2. Some
Use in e
advanta
compare

3. Trick
extra ap
See Tan
on Weil

ithm problem

03.
s prime.

teger
$- 1\}$
$p = 262682$?

$\to 5^n \bmod p$
$\ldots, p - 1\}$.
n
$p = 262682$.

rute force.
ay?

Typical cryptanalytic application:

Imagine standard $p = 1000003$ in the Diffie-Hellman protocol.

User chooses secret key n, publishes $5^n \bmod p = 262682$.

Can attacker quickly solve the discrete-logarithm problem? Given public key $5^n \bmod p$, quickly find secret key n?

(Warning: This is *one* way to attack the protocol. Maybe there are better ways.)

Relations to ECC:

1. Some DL techn
to elliptic-curve D
Use in evaluating
security of an ellip

2. Some technique
Use in evaluating
advantages of ellip
compared to multi

3. Tricky: Some t
extra applications
See Tanja Lange's
on Weil descent et

lem

82?

p

}.

82.

Typical cryptanalytic application:

Imagine standard $p = 1000003$ in the Diffie-Hellman protocol.

User chooses secret key n, publishes $5^n \bmod p = 262682$.

Can attacker quickly solve the discrete-logarithm problem? Given public key $5^n \bmod p$, quickly find secret key n?

(Warning: This is *one* way to attack the protocol. Maybe there are better ways.)

Relations to ECC:

1. Some DL techniques also
to elliptic-curve DL problem
Use in evaluating
security of an elliptic curve.

2. Some techniques don't ap
Use in evaluating
advantages of elliptic curves
compared to multiplication.

3. Tricky: Some techniques
extra applications to some c
See Tanja Lange's talk
on Weil descent etc.

Typical cryptanalytic application:

Imagine standard $p = 1000003$ in the Diffie-Hellman protocol.

User chooses secret key n, publishes $5^n \bmod p = 262682$.

Can attacker quickly solve the discrete-logarithm problem? Given public key $5^n \bmod p$, quickly find secret key n?

(Warning: This is *one* way to attack the protocol. Maybe there are better ways.)

Relations to ECC:

1. Some DL techniques also apply to elliptic-curve DL problems. Use in evaluating security of an elliptic curve.

2. Some techniques don't apply. Use in evaluating advantages of elliptic curves compared to multiplication.

3. Tricky: Some techniques have extra applications to some curves. See Tanja Lange's talk on Weil descent etc.

cryptanalytic application:

standard $p = 1000003$
iffie-Hellman protocol.

ooses secret key n,
s $5^n \bmod p = 262682$.

acker quickly solve
rete-logarithm problem?
ublic key $5^n \bmod p$,
find secret key n?

g: This is *one* way
k the protocol.
here are better ways.)

Relations to ECC:

1. Some DL techniques also apply to elliptic-curve DL problems. Use in evaluating security of an elliptic curve.

2. Some techniques don't apply. Use in evaluating advantages of elliptic curves compared to multiplication.

3. Tricky: Some techniques have extra applications to some curves. See Tanja Lange's talk on Weil descent etc.

Underst

Can com
5^1 mod
5^2 mod
5^3 mod
5^8 mod
5^9 mod
$5^{1000002}$

At some
with 5^n

Maximu
$\leq p - 1$
$\leq p - 1$
that doe

tic application:

$p = 1000003$
nan protocol.

et key n,
$p = 262682$.

kly solve
thm problem?
^{n} mod p,
key n?

one way
ocol.
etter ways.)

Relations to ECC:

1. Some DL techniques also apply to elliptic-curve DL problems. Use in evaluating security of an elliptic curve.

2. Some techniques don't apply. Use in evaluating advantages of elliptic curves compared to multiplication.

3. Tricky: Some techniques have extra applications to some curves. See Tanja Lange's talk on Weil descent etc.

<u>Understanding bru</u>

Can compute succ
$5^1 \bmod p = 5$,
$5^2 \bmod p = 25$,
$5^3 \bmod p = 125$, .
$5^8 \bmod p = 39062$
$5^9 \bmod p = 95312$
$5^{1000002} \bmod p =$

At some point we'
with $5^n \bmod p =$ 2

Maximum cost of
$\le p - 1$ mults by
$\le p - 1$ nanosecon
that does 1 mult/

ation:

003
ol.

32.

em?

s.)

Relations to ECC:

1. Some DL techniques also apply to elliptic-curve DL problems. Use in evaluating security of an elliptic curve.

2. Some techniques don't apply. Use in evaluating advantages of elliptic curves compared to multiplication.

3. Tricky: Some techniques have extra applications to some curves. See Tanja Lange's talk on Weil descent etc.

Understanding brute force

Can compute successively
$5^1 \bmod p = 5$,
$5^2 \bmod p = 25$,
$5^3 \bmod p = 125, \ldots,$
$5^8 \bmod p = 390625$,
$5^9 \bmod p = 953122, \ldots,$
$5^{1000002} \bmod p = 1$.

At some point we'll find n with $5^n \bmod p = 262682$.

Maximum cost of computati
$\le p - 1$ mults by 5 mod p;
$\le p - 1$ nanoseconds on a C
that does 1 mult/nanosecon

Relations to ECC:

1. Some DL techniques also apply to elliptic-curve DL problems. Use in evaluating security of an elliptic curve.

2. Some techniques don't apply. Use in evaluating advantages of elliptic curves compared to multiplication.

3. Tricky: Some techniques have extra applications to some curves. See Tanja Lange's talk on Weil descent etc.

Understanding brute force

Can compute successively
$5^1 \bmod p = 5$,
$5^2 \bmod p = 25$,
$5^3 \bmod p = 125, \ldots,$
$5^8 \bmod p = 390625$,
$5^9 \bmod p = 953122, \ldots,$
$5^{1000002} \bmod p = 1$.

At some point we'll find n with $5^n \bmod p = 262682$.

Maximum cost of computation:
$\le p - 1$ mults by 5 mod p;
$\le p - 1$ nanoseconds on a CPU that does 1 mult/nanosecond.

s to ECC:

e DL techniques also apply
c-curve DL problems.
valuating
of an elliptic curve.

e techniques don't apply.
valuating
ges of elliptic curves
ed to multiplication.

y: Some techniques have
plications to some curves.
ja Lange's talk
descent etc.

Understanding brute force

Can compute successively
$5^1 \bmod p = 5$,
$5^2 \bmod p = 25$,
$5^3 \bmod p = 125, \ldots,$
$5^8 \bmod p = 390625$,
$5^9 \bmod p = 953122, \ldots,$
$5^{1000002} \bmod p = 1$.

At some point we'll find n
with $5^n \bmod p = 262682$.

Maximum cost of computation:
$\leq p - 1$ mults by 5 mod p;
$\leq p - 1$ nanoseconds on a CPU
that does 1 mult/nanosecond.

This is r
for $p \approx$ 2

But user
standard
making

Attack c
$\approx 2^{50}$ m
$\approx 2^{100}$ r

(Not exa
cost of r
But this

niques also apply
L problems.

otic curve.

es don't apply.

otic curves
iplication.

echniques have
to some curves.
talk
tc.

Understanding brute force

Can compute successively
$5^1 \bmod p = 5$,
$5^2 \bmod p = 25$,
$5^3 \bmod p = 125, \ldots,$
$5^8 \bmod p = 390625$,
$5^9 \bmod p = 953122, \ldots,$
$5^{1000002} \bmod p = 1$.

At some point we'll find n
with $5^n \bmod p = 262682$.

Maximum cost of computation:
$\leq p - 1$ mults by 5 mod p;
$\leq p - 1$ nanoseconds on a CPU
that does 1 mult/nanosecond.

This is negligible v
for $p \approx 2^{20}$.

But users can
standardize a large
making the attack

Attack cost scales
$\approx 2^{50}$ mults for p
$\approx 2^{100}$ mults for p

(Not exactly linear
cost of mults grow
But this is a mino

apply
s.

pply.

have
urves.

Understanding brute force

Can compute successively
$5^1 \bmod p = 5$,
$5^2 \bmod p = 25$,
$5^3 \bmod p = 125, \ldots,$
$5^8 \bmod p = 390625$,
$5^9 \bmod p = 953122, \ldots,$
$5^{1000002} \bmod p = 1$.

At some point we'll find n
with $5^n \bmod p = 262682$.

Maximum cost of computation:
$\leq p - 1$ mults by 5 mod p;
$\leq p - 1$ nanoseconds on a CPU
that does 1 mult/nanosecond.

This is negligible work
for $p \approx 2^{20}$.

But users can
standardize a larger p,
making the attack slower.

Attack cost scales linearly:
$\approx 2^{50}$ mults for $p \approx 2^{50}$,
$\approx 2^{100}$ mults for $p \approx 2^{100}$, e

(Not exactly linearly:
cost of mults grows with p.
But this is a minor effect.)

Understanding brute force

Can compute successively
$5^1 \bmod p = 5$,
$5^2 \bmod p = 25$,
$5^3 \bmod p = 125, \ldots,$
$5^8 \bmod p = 390625$,
$5^9 \bmod p = 953122, \ldots,$
$5^{1000002} \bmod p = 1$.

At some point we'll find n
with $5^n \bmod p = 262682$.

Maximum cost of computation:
$\le p - 1$ mults by 5 mod p;
$\le p - 1$ nanoseconds on a CPU
that does 1 mult/nanosecond.

This is negligible work
for $p \approx 2^{20}$.

But users can
standardize a larger p,
making the attack slower.

Attack cost scales linearly:
$\approx 2^{50}$ mults for $p \approx 2^{50}$,
$\approx 2^{100}$ mults for $p \approx 2^{100}$, etc.

(Not exactly linearly:
cost of mults grows with p.
But this is a minor effect.)

anding brute force

npute successively

$p = 5$,

$p = 25$,

$p = 125, \ldots,$

$p = 390625$,

$p = 953122, \ldots,$

mod $p = 1$.

point we'll find n

mod $p = 262682$.

m cost of computation:

mults by 5 mod p;

nanoseconds on a CPU

es 1 mult/nanosecond.

This is negligible work for $p \approx 2^{20}$.

But users can standardize a larger p, making the attack slower.

Attack cost scales linearly:
$\approx 2^{50}$ mults for $p \approx 2^{50}$,
$\approx 2^{100}$ mults for $p \approx 2^{100}$, etc.

(Not exactly linearly: cost of mults grows with p. But this is a minor effect.)

Comput

of finishi

Chance

1/2 char

1/10 cha

"So user

That's p

"random

choose r

compute

compute

(515040

compute

subtract

ute force

cessively

. . . ,

25,

22, . . . ,

1.

ll find n

262682.

computation:

5 mod p;

nds on a CPU

nanosecond.

This is negligible work for $p \approx 2^{20}$.

But users can standardize a larger p, making the attack slower.

Attack cost scales linearly:
$\approx 2^{50}$ mults for $p \approx 2^{50}$,
$\approx 2^{100}$ mults for $p \approx 2^{100}$, etc.

(Not exactly linearly:
cost of mults grows with p.
But this is a minor effect.)

Computation has a

of finishing earlier.

Chance scales line

1/2 chance of 1/2

1/10 chance of 1/

"So users should c

That's pointless. V

"random self-reduc

choose random r,

compute 5^r mod p

compute $5^r 5^n$ mo

$(515040 \cdot (5^n$ mod

compute discrete l

subtract r mod p -

ion:

CPU
d.

This is negligible work
for $p \approx 2^{20}$.

But users can
standardize a larger p,
making the attack slower.

Attack cost scales linearly:
$\approx 2^{50}$ mults for $p \approx 2^{50}$,
$\approx 2^{100}$ mults for $p \approx 2^{100}$, etc.

(Not exactly linearly:
cost of mults grows with p.
But this is a minor effect.)

Computation has a good ch
of finishing earlier.
Chance scales linearly:
1/2 chance of 1/2 cost;
1/10 chance of 1/10 cost; e

"So users should choose larg

That's pointless. We can ap
"random self-reduction":
choose random r, say 72637
compute $5^r \bmod p = 51504$
compute $5^r 5^n \bmod p$ as
$(515040 \cdot (5^n \bmod p)) \bmod$
compute discrete log;
subtract $r \bmod p-1$; obtai

This is negligible work
for $p \approx 2^{20}$.

But users can
standardize a larger p,
making the attack slower.

Attack cost scales linearly:
$\approx 2^{50}$ mults for $p \approx 2^{50}$,
$\approx 2^{100}$ mults for $p \approx 2^{100}$, etc.

(Not exactly linearly:
cost of mults grows with p.
But this is a minor effect.)

Computation has a good chance
of finishing earlier.
Chance scales linearly:
$1/2$ chance of $1/2$ cost;
$1/10$ chance of $1/10$ cost; etc.

"So users should choose large n."

That's pointless. We can apply
"random self-reduction":
choose random r, say 726379;
compute $5^r \bmod p = 515040$;
compute $5^r 5^n \bmod p$ as
$(515040 \cdot (5^n \bmod p)) \bmod p$;
compute discrete log;
subtract $r \bmod p-1$; obtain n.

negligible work
2^{20}.

rs can
lize a larger p,
the attack slower.

cost scales linearly:
ults for $p \approx 2^{50}$,
mults for $p \approx 2^{100}$, etc.

actly linearly:
mults grows with p.
is a minor effect.)

Computation has a good chance of finishing earlier.
Chance scales linearly:
1/2 chance of 1/2 cost;
1/10 chance of 1/10 cost; etc.

"So users should choose large n."

That's pointless. We can apply "random self-reduction":
choose random r, say 726379;
compute $5^r \bmod p = 515040$;
compute $5^r 5^n \bmod p$ as $(515040 \cdot (5^n \bmod p)) \bmod p$;
compute discrete log;
subtract $r \bmod p - 1$; obtain n.

Comput

One low
many pa
Example
2^{10} core
each 2^{30}
Maybe;
for detai

Attacker
many pa
Example
so 2^{34} c
so 2^{64} m
so 2^{89} m

work

er p,

slower.

linearly:

$\approx 2^{50}$,

$\approx 2^{100}$, etc.

rly:

vs with p.

r effect.)

Computation has a good chance of finishing earlier.

Chance scales linearly:

1/2 chance of 1/2 cost;

1/10 chance of 1/10 cost; etc.

"So users should choose large n."

That's pointless. We can apply "random self-reduction":

choose random r, say 726379;

compute $5^r \bmod p = 515040$;

compute $5^r 5^n \bmod p$ as

$(515040 \cdot (5^n \bmod p)) \bmod p$;

compute discrete log;

subtract $r \bmod p-1$; obtain n.

Computation can

One low-cost chip

many parallel sear

Example, 2^6 €: o

2^{10} cores on the c

each 2^{30} mults/se

Maybe; see SHAR

for detailed cost a

Attacker can run

many parallel chip

Example, 2^{30} €: 2

so 2^{34} cores,

so 2^{64} mults/secon

so 2^{89} mults/year.

etc.

Computation has a good chance of finishing earlier.
Chance scales linearly:
1/2 chance of 1/2 cost;
1/10 chance of 1/10 cost; etc.

“So users should choose large n.”

That’s pointless. We can apply “random self-reduction”:
choose random r, say 726379;
compute $5^r \bmod p = 515040$;
compute $5^r 5^n \bmod p$ as
$(515040 \cdot (5^n \bmod p)) \bmod p$;
compute discrete log;
subtract $r \bmod p-1$; obtain n.

Computation can be parallel

One low-cost chip can run many parallel searches.
Example, 2^6 €: one chip,
2^{10} cores on the chip,
each 2^{30} mults/second?
Maybe; see SHARCS worksh
for detailed cost analyses.

Attacker can run many parallel chips.
Example, 2^{30} €: 2^{24} chips,
so 2^{34} cores,
so 2^{64} mults/second,
so 2^{89} mults/year.

Computation has a good chance of finishing earlier.
Chance scales linearly:
1/2 chance of 1/2 cost;
1/10 chance of 1/10 cost; etc.

"So users should choose large n."

That's pointless. We can apply "random self-reduction":
choose random r, say 726379;
compute $5^r \bmod p = 515040$;
compute $5^r 5^n \bmod p$ as
$(515040 \cdot (5^n \bmod p)) \bmod p$;
compute discrete log;
subtract $r \bmod p-1$; obtain n.

Computation can be parallelized.

One low-cost chip can run many parallel searches.
Example, 2^6 €: one chip,
2^{10} cores on the chip,
each 2^{30} mults/second?
Maybe; see SHARCS workshops for detailed cost analyses.

Attacker can run many parallel chips.
Example, 2^{30} €: 2^{24} chips,
so 2^{34} cores,
so 2^{64} mults/second,
so 2^{89} mults/year.

ation has a good chance
ing earlier.
scales linearly:
nce of $1/2$ cost;
ance of $1/10$ cost; etc.

rs should choose large n."

pointless. We can apply
n self-reduction":
random r, say 726379;
e $5^r \bmod p = 515040$;
e $5^r 5^n \bmod p$ as
$\cdot (5^n \bmod p)) \bmod p$;
e discrete log;
$r \bmod p-1$; obtain n.

Computation can be parallelized.

One low-cost chip can run
many parallel searches.
Example, 2^6 €: one chip,
2^{10} cores on the chip,
each 2^{30} mults/second?
Maybe; see SHARCS workshops
for detailed cost analyses.

Attacker can run
many parallel chips.
Example, 2^{30} €: 2^{24} chips,
so 2^{34} cores,
so 2^{64} mults/second,
so 2^{89} mults/year.

<u>Multiple</u>

Computa
to many

Given 10
5^{n_2} mod
Can find
with $\leq p$

Simplest
a sorted
5^{n_1} mod
Then ch
5^1 mod

a good chance

r
arly:
2 cost;
/10 cost; etc.

choose large n."

We can apply
ction":
say 726379;
$\!\!\rho = 515040$;
d p as
p)) mod p;
og;
$- 1$; obtain n.

Computation can be parallelized.

One low-cost chip can run
many parallel searches.
Example, 2^6 €: one chip,
2^{10} cores on the chip,
each 2^{30} mults/second?
Maybe; see SHARCS workshops
for detailed cost analyses.

Attacker can run
many parallel chips.
Example, 2^{30} €: 2^{24} chips,
so 2^{34} cores,
so 2^{64} mults/second,
so 2^{89} mults/year.

<u>Multiple targets ar</u>

Computation can
to many targets at

Given 100 DL targ
5^{n_2} mod p, . . . , 5
Can find *all* of n_1
with $\le p - 1$ mult

Simplest approach
a sorted table con
5^{n_1} mod p, . . . , 5
Then check table
5^1 mod p, 5^2 mod

ance

tc.

ge n."

pply

79;

0;

p;

n n.

Computation can be parallelized.

One low-cost chip can run
many parallel searches.
Example, 2^6 €: one chip,
2^{10} cores on the chip,
each 2^{30} mults/second?
Maybe; see SHARCS workshops
for detailed cost analyses.

Attacker can run
many parallel chips.
Example, 2^{30} €: 2^{24} chips,
so 2^{34} cores,
so 2^{64} mults/second,
so 2^{89} mults/year.

Multiple targets and giant st

Computation can be applied
to many targets at once.

Given 100 DL targets 5^{n_1} m
5^{n_2} mod p, ..., $5^{n_{100}}$ mod p
Can find *all* of $n_1, n_2, \ldots, n$
with $\le p - 1$ mults mod p.

Simplest approach: First bu
a sorted table containing
5^{n_1} mod p, ..., $5^{n_{100}}$ mod p
Then check table for
5^1 mod p, 5^2 mod p, etc.

Computation can be parallelized.

One low-cost chip can run many parallel searches. Example, 2^6 €: one chip, 2^{10} cores on the chip, each 2^{30} mults/second? Maybe; see SHARCS workshops for detailed cost analyses.

Attacker can run many parallel chips. Example, 2^{30} €: 2^{24} chips, so 2^{34} cores, so 2^{64} mults/second, so 2^{89} mults/year.

Multiple targets and giant steps

Computation can be applied to many targets at once.

Given 100 DL targets $5^{n_1} \bmod p$, $5^{n_2} \bmod p$, ..., $5^{n_{100}} \bmod p$: Can find *all* of $n_1, n_2, \ldots, n_{100}$ with $\le p - 1$ mults mod p.

Simplest approach: First build a sorted table containing $5^{n_1} \bmod p$, ..., $5^{n_{100}} \bmod p$. Then check table for $5^1 \bmod p$, $5^2 \bmod p$, etc.

ation can be parallelized.

-cost chip can run
rallel searches.
e, 2^6 €: one chip,
s on the chip,
mults/second?
see SHARCS workshops
led cost analyses.

r can run
rallel chips.
e, 2^{30} €: 2^{24} chips,
ores,
nults/second,
nults/year.

Multiple targets and giant steps

Computation can be applied
to many targets at once.

Given 100 DL targets $5^{n_1} \bmod p$,
$5^{n_2} \bmod p$, . . . , $5^{n_{100}} \bmod p$:
Can find *all* of $n_1, n_2, \ldots, n_{100}$
with $\le p - 1$ mults mod p.

Simplest approach: First build
a sorted table containing
$5^{n_1} \bmod p$, . . . , $5^{n_{100}} \bmod p$.
Then check table for
$5^1 \bmod p$, $5^2 \bmod p$, etc.

Interesti
Solving
isn't mu
solving c

Interesti
Solving
out of 1
is much
solving c

When di
find its
Typically

be parallelized.

can run
ches.
ne chip,
hip,
cond?
CS workshops
nalyses.

s.
2^{24} chips,

nd,

Multiple targets and giant steps

Computation can be applied to many targets at once.

Given 100 DL targets 5^{n_1} mod p, 5^{n_2} mod p, . . . , $5^{n_{100}}$ mod p: Can find *all* of $n_1, n_2, \ldots, n_{100}$ with $\le p - 1$ mults mod p.

Simplest approach: First build a sorted table containing 5^{n_1} mod p, . . . , $5^{n_{100}}$ mod p. Then check table for 5^1 mod p, 5^2 mod p, etc.

Interesting conseq
Solving all 100 DL
isn't much harder
solving one DL pro

Interesting conseq
Solving *at least on*
out of 100 DL pro
is much easier tha
solving one DL pro

When did this con
find its *first* n_i?
Typically $\approx (p - 1$

ized.

hops

Multiple targets and giant steps

Computation can be applied to many targets at once.

Given 100 DL targets $5^{n_1} \bmod p$, $5^{n_2} \bmod p$, . . . , $5^{n_{100}} \bmod p$: Can find *all* of $n_1, n_2, \ldots, n_{100}$ with $\leq p - 1$ mults mod p.

Simplest approach: First build a sorted table containing $5^{n_1} \bmod p$, . . . , $5^{n_{100}} \bmod p$. Then check table for $5^1 \bmod p$, $5^2 \bmod p$, etc.

Interesting consequence #1:
Solving all 100 DL problems
isn't much harder than
solving one DL problem.

Interesting consequence #2:
Solving *at least one*
out of 100 DL problems
is much easier than
solving one DL problem.

When did this computation
find its *first* n_i?
Typically $\approx (p - 1)/100$ mu

Multiple targets and giant steps

Computation can be applied to many targets at once.

Given 100 DL targets $5^{n_1} \bmod p$, $5^{n_2} \bmod p$, ..., $5^{n_{100}} \bmod p$: Can find *all* of $n_1, n_2, \ldots, n_{100}$ with $\le p-1$ mults mod p.

Simplest approach: First build a sorted table containing $5^{n_1} \bmod p$, ..., $5^{n_{100}} \bmod p$. Then check table for $5^1 \bmod p$, $5^2 \bmod p$, etc.

Interesting consequence #1: Solving all 100 DL problems isn't much harder than solving one DL problem.

Interesting consequence #2: Solving *at least one* out of 100 DL problems is much easier than solving one DL problem.

When did this computation find its *first* n_i? Typically $\approx (p-1)/100$ mults.

targets and giant steps

ation can be applied
targets at once.

00 DL targets 5^{n_1} mod p,
l p, . . . , $5^{n_{100}}$ mod p:
l *all* of $n_1, n_2, \ldots, n_{100}$
$p - 1$ mults mod p.

t approach: First build
table containing
l p, . . . , $5^{n_{100}}$ mod p.
eck table for
p, 5^2 mod p, etc.

Interesting consequence #1:
Solving all 100 DL problems
isn't much harder than
solving one DL problem.

Interesting consequence #2:
Solving *at least one*
out of 100 DL problems
is much easier than
solving one DL problem.

When did this computation
find its *first* n_i?
Typically $\approx (p - 1)/100$ mults.

Can use
to turn
into mul

Given 5^n
Choose
Comput
$5^{r_2} 5^n$ m

Solve th
Typically
to find
$r_i + n$ m
immedia

nd giant steps

be applied
t once.

gets 5^{n_1} mod p,
$^{n_{100}}$ mod p:
$, n_2, \ldots, n_{100}$
ts mod p.

: First build
taining
$^{n_{100}}$ mod p.
for
p, etc.

Interesting consequence #1:
Solving all 100 DL problems
isn't much harder than
solving one DL problem.

Interesting consequence #2:
Solving *at least one*
out of 100 DL problems
is much easier than
solving one DL problem.

When did this computation
find its *first* n_i?
Typically $\approx (p-1)/100$ mults.

Can use random s
to turn a single ta
into multiple targe

Given 5^n mod p:
Choose random r_1
Compute $5^{r_1}5^n$ m
$5^{r_2}5^n$ mod p, etc.

Solve these 100 D
Typically $\approx (p-1$
to find *at least on*
$r_i + n$ mod $p-1$,
immediately revea

teps

od p,

p:

$_{100}$

ild

p.

Interesting consequence #1:
Solving all 100 DL problems
isn't much harder than
solving one DL problem.

Interesting consequence #2:
Solving *at least one*
out of 100 DL problems
is much easier than
solving one DL problem.

When did this computation
find its *first* n_i?
Typically $\approx (p-1)/100$ mults.

Can use random self-reducti
to turn a single target
into multiple targets.

Given $5^n \bmod p$:
Choose random $r_1, r_2, \ldots, r$
Compute $5^{r_1} 5^n \bmod p$,
$5^{r_2} 5^n \bmod p$, etc.

Solve these 100 DL problem
Typically $\approx (p-1)/100$ mu
to find *at least one*
$r_i + n \bmod p-1$,
immediately revealing n.

Interesting consequence #1:
Solving all 100 DL problems isn't much harder than solving one DL problem.

Interesting consequence #2:
Solving *at least one* out of 100 DL problems is much easier than solving one DL problem.

When did this computation find its *first* n_i?
Typically $\approx (p-1)/100$ mults.

Can use random self-reduction to turn a single target into multiple targets.

Given $5^n \bmod p$:
Choose random $r_1, r_2, \ldots, r_{100}$.
Compute $5^{r_1} 5^n \bmod p$, $5^{r_2} 5^n \bmod p$, etc.

Solve these 100 DL problems.
Typically $\approx (p-1)/100$ mults to find *at least one* $r_i + n \bmod p-1$, immediately revealing n.

ng consequence #1:
all 100 DL problems
ch harder than
one DL problem.

ng consequence #2:
at least one
00 DL problems
easier than
one DL problem.

'd this computation
first n_i?
y $\approx (p-1)/100$ mults.

Can use random self-reduction
to turn a single target
into multiple targets.

Given $5^n \bmod p$:
Choose random $r_1, r_2, \ldots, r_{100}$.
Compute $5^{r_1} 5^n \bmod p$,
$5^{r_2} 5^n \bmod p$, etc.

Solve these 100 DL problems.
Typically $\approx (p-1)/100$ mults
to find *at least one*
$r_i + n \bmod p - 1$,
immediately revealing n.

Also spe
to comp
$\approx \lg p$ m

Faster:
with r_1
Comput
$5^{r_1} 5^n$ m
$5^{2r_1} 5^n$ r
$5^{3r_1} 5^n$ r
Just 1 m

$\approx 100 +$
to find r

uence #1:
. problems
than
oblem.

uence #2:
ne
blems
n
oblem.

nputation

.)/100 mults.

Can use random self-reduction to turn a single target into multiple targets.

Given $5^n \bmod p$:
Choose random $r_1, r_2, \ldots, r_{100}$.
Compute $5^{r_1} 5^n \bmod p$, $5^{r_2} 5^n \bmod p$, etc.

Solve these 100 DL problems. Typically $\approx (p-1)/100$ mults to find *at least one* $r_i + n \bmod p-1$, immediately revealing n.

Also spent some m
to compute each 5
$\approx \lg p$ mults for ea

Faster: Choose r_i
with $r_1 \approx (p-1)/$
Compute 5^{r_1} mod
$5^{r_1} 5^n \bmod p$;
$5^{2r_1} 5^n \bmod p$;
$5^{3r_1} 5^n \bmod p$; etc
Just 1 mult for ea

$\approx 100 + \lg p + (p$
to find n given 5^n

lts.

Can use random self-reduction to turn a single target into multiple targets.

Given 5^n mod p:
Choose random $r_1, r_2, \ldots, r_{100}$.
Compute $5^{r_1} 5^n$ mod p,
$5^{r_2} 5^n$ mod p, etc.

Solve these 100 DL problems.
Typically $\approx (p-1)/100$ mults to find *at least one* $r_i + n$ mod $p - 1$, immediately revealing n.

Also spent some mults to compute each 5^{r_i} mod p: $\approx \lg p$ mults for each i.

Faster: Choose $r_i = i r_1$ with $r_1 \approx (p-1)/100$.
Compute 5^{r_1} mod p;
$5^{r_1} 5^n$ mod p;
$5^{2r_1} 5^n$ mod p;
$5^{3r_1} 5^n$ mod p; etc.
Just 1 mult for each new i.

$\approx 100 + \lg p + (p-1)/100$ to find n given 5^n mod p.

Can use random self-reduction
to turn a single target
into multiple targets.

Given $5^n \bmod p$:
Choose random $r_1, r_2, \ldots, r_{100}$.
Compute $5^{r_1} 5^n \bmod p$,
$5^{r_2} 5^n \bmod p$, etc.

Solve these 100 DL problems.
Typically $\approx (p-1)/100$ mults
to find *at least one*
$r_i + n \bmod p-1$,
immediately revealing n.

Also spent some mults
to compute each $5^{r_i} \bmod p$:
$\approx \lg p$ mults for each i.

Faster: Choose $r_i = i r_1$
with $r_1 \approx (p-1)/100$.
Compute $5^{r_1} \bmod p$;
$5^{r_1} 5^n \bmod p$;
$5^{2r_1} 5^n \bmod p$;
$5^{3r_1} 5^n \bmod p$; etc.
Just 1 mult for each new i.

$\approx 100 + \lg p + (p-1)/100$ mults
to find n given $5^n \bmod p$.

random self-reduction
a single target
tiple targets.

^{n} mod p:
random $r_1, r_2, \ldots, r_{100}$.
e $5^{r_1} 5^n$ mod p,
od p, etc.

ese 100 DL problems.
y $\approx (p-1)/100$ mults
at least one
nod $p-1$,
tely revealing n.

Also spent some mults
to compute each 5^{r_i} mod p:
$\approx \lg p$ mults for each i.

Faster: Choose $r_i = i r_1$
with $r_1 \approx (p-1)/100$.
Compute 5^{r_1} mod p;
$5^{r_1} 5^n$ mod p;
$5^{2r_1} 5^n$ mod p;
$5^{3r_1} 5^n$ mod p; etc.
Just 1 mult for each new i.

$\approx 100 + \lg p + (p-1)/100$ mults
to find n given 5^n mod p.

Faster:
Only $\approx$
to solve

"Shanks
discrete-

Example
5^n mod

Compute
Then co
$5^{1024} 5^n$
$5^{2 \cdot 1024} 5$
$5^{3 \cdot 1024} 5$
$5^{1000 \cdot 102}$

elf-reduction
rget
ets.

$, r_2, \ldots, r_{100}$.
od p,

L problems.
$)/100$ mults
e

ling n.

Also spent some mults
to compute each 5^{r_i} mod p:
$\approx \lg p$ mults for each i.

Faster: Choose $r_i = ir_1$
with $r_1 \approx (p-1)/100$.
Compute 5^{r_1} mod p;
$5^{r_1}5^n$ mod p;
$5^{2r_1}5^n$ mod p;
$5^{3r_1}5^n$ mod p; etc.
Just 1 mult for each new i.

$\approx 100 + \lg p + (p-1)/100$ mults
to find n given 5^n mod p.

Faster: Increase 1
Only $\approx 2\sqrt{p}$ mult
to solve one DL pr

"Shanks baby-step
discrete-logarithm

Example: $p = 100$
5^n mod $p = 2626$

Compute 5^{1024} mo
Then compute 100
$5^{1024}5^n$ mod $p =$
$5^{2\cdot1024}5^n$ mod $p =$
$5^{3\cdot1024}5^n$ mod $p =$
$5^{1000\cdot1024}5^n$ mod

on

r_{100}.

s.

lts

Also spent some mults
to compute each $5^{r_i} \bmod p$:
$\approx \lg p$ mults for each i.

Faster: Choose $r_i = ir_1$
with $r_1 \approx (p-1)/100$.
Compute $5^{r_1} \bmod p$;
$5^{r_1}5^n \bmod p$;
$5^{2r_1}5^n \bmod p$;
$5^{3r_1}5^n \bmod p$; etc.
Just 1 mult for each new i.

$\approx 100 + \lg p + (p-1)/100$ mults
to find n given $5^n \bmod p$.

Faster: Increase 100 to $\approx \surd$
Only $\approx 2\sqrt{p}$ mults
to solve one DL problem!

"Shanks baby-step-giant-ste
discrete-logarithm algorithm

Example: $p = 1000003$,
$5^n \bmod p = 262682$.

Compute $5^{1024} \bmod p = 58$
Then compute 1000 targets
$5^{1024}5^n \bmod p = 966849$,
$5^{2\cdot1024}5^n \bmod p = 579277$,
$5^{3\cdot1024}5^n \bmod p = 579062$,
$5^{1000\cdot1024}5^n \bmod p = 3217$

Also spent some mults
to compute each $5^{r_i} \bmod p$:
$\approx \lg p$ mults for each i.

Faster: Choose $r_i = ir_1$
with $r_1 \approx (p-1)/100$.
Compute $5^{r_1} \bmod p$;
$5^{r_1}5^n \bmod p$;
$5^{2r_1}5^n \bmod p$;
$5^{3r_1}5^n \bmod p$; etc.
Just 1 mult for each new i.

$\approx 100 + \lg p + (p-1)/100$ mults
to find n given $5^n \bmod p$.

Faster: Increase 100 to $\approx \sqrt{p}$.
Only $\approx 2\sqrt{p}$ mults
to solve one DL problem!

"Shanks baby-step-giant-step
discrete-logarithm algorithm."

Example: $p = 1000003$,
$5^n \bmod p = 262682$.

Compute $5^{1024} \bmod p = 58588$.
Then compute 1000 targets:
$5^{1024}5^n \bmod p = 966849$,
$5^{2\cdot 1024}5^n \bmod p = 579277$,
$5^{3\cdot 1024}5^n \bmod p = 579062, \ldots,$
$5^{1000\cdot 1024}5^n \bmod p = 321705$.

ent some mults
ute each 5^{r_i} mod p:
nults for each i.

Choose $r_i = ir_1$
$\approx (p-1)/100$.
e 5^{r_1} mod p;
od p;
mod p;
mod p; etc.
nult for each new i.

$\lg p + (p-1)/100$ mults
n given 5^n mod p.

Faster: Increase 100 to $\approx \sqrt{p}$.
Only $\approx 2\sqrt{p}$ mults
to solve one DL problem!

"Shanks baby-step-giant-step discrete-logarithm algorithm."

Example: $p = 1000003$,
5^n mod $p = 262682$.

Compute 5^{1024} mod $p = 58588$.
Then compute 1000 targets:
$5^{1024}5^n$ mod $p = 966849$,
$5^{2\cdot1024}5^n$ mod $p = 579277$,
$5^{3\cdot1024}5^n$ mod $p = 579062$, $\ldots$,
$5^{1000\cdot1024}5^n$ mod $p = 321705$.

Build a
2573 =
3371 =
3593 =
4960 =
5218 =
999675

Look up
5^3 mod

5^{755} mo
966603
in the ta
so 755 =
deduce

nults

5^{r_i} mod p:

ach i.

$= ir_1$

/100.

p;

.

ch new i.

$-1)/100$ mults

mod p.

Faster: Increase 100 to $\approx \sqrt{p}$.
Only $\approx 2\sqrt{p}$ mults
to solve one DL problem!

"Shanks baby-step-giant-step
discrete-logarithm algorithm."

Example: $p = 1000003$,
$5^n \bmod p = 262682$.

Compute $5^{1024} \bmod p = 58588$.
Then compute 1000 targets:
$5^{1024}5^n \bmod p = 966849$,
$5^{2\cdot 1024}5^n \bmod p = 579277$,
$5^{3\cdot 1024}5^n \bmod p = 579062$, $\ldots$,
$5^{1000\cdot 1024}5^n \bmod p = 321705$.

Build a sorted tab
$2573 = 5^{430\cdot 1024}5^{n}$
$3371 = 5^{192\cdot 1024}5^{n}$
$3593 = 5^{626\cdot 1024}5^{n}$
$4960 = 5^{663\cdot 1024}5^{n}$
$5218 = 5^{376\cdot 1024}5^{n}$
$999675 = 5^{344\cdot 102}$

Look up 5^1 mod p
5^3 mod p, etc. in t

$5^{755} \bmod p = 966$
$966603 = 5^{332\cdot 102}$
in the table of targ
so $755 = 332 \cdot 102$
deduce $n = 66078$

mults

Faster: Increase 100 to $\approx \sqrt{p}$.
Only $\approx 2\sqrt{p}$ mults
to solve one DL problem!

"Shanks baby-step-giant-step
discrete-logarithm algorithm."

Example: $p = 1000003$,
$5^n \bmod p = 262682$.

Compute $5^{1024} \bmod p = 58588$.
Then compute 1000 targets:
$5^{1024}5^n \bmod p = 966849$,
$5^{2\cdot1024}5^n \bmod p = 579277$,
$5^{3\cdot1024}5^n \bmod p = 579062$, $\ldots$,
$5^{1000\cdot1024}5^n \bmod p = 321705$.

Build a sorted table of targe
$2573 = 5^{430\cdot1024}5^n \bmod p$,
$3371 = 5^{192\cdot1024}5^n \bmod p$,
$3593 = 5^{626\cdot1024}5^n \bmod p$,
$4960 = 5^{663\cdot1024}5^n \bmod p$,
$5218 = 5^{376\cdot1024}5^n \bmod p$, .
$999675 = 5^{344\cdot1024}5^n \bmod p$

Look up $5^1 \bmod p$, $5^2 \bmod$
$5^3 \bmod p$, etc. in this table.

$5^{755} \bmod p = 966603$; find
$966603 = 5^{332\cdot1024}5^n \bmod p$
in the table of targets;
so $755 = 332\cdot1024 + n$ mod
deduce $n = 660789$.

Faster: Increase 100 to $\approx \sqrt{p}$.
Only $\approx 2\sqrt{p}$ mults
to solve one DL problem!

"Shanks baby-step-giant-step discrete-logarithm algorithm."

Example: $p = 1000003$,
$5^n \bmod p = 262682$.

Compute $5^{1024} \bmod p = 58588$.
Then compute 1000 targets:
$5^{1024} 5^n \bmod p = 966849$,
$5^{2\cdot 1024} 5^n \bmod p = 579277$,
$5^{3\cdot 1024} 5^n \bmod p = 579062$, ...,
$5^{1000\cdot 1024} 5^n \bmod p = 321705$.

Build a sorted table of targets:
$2573 = 5^{430\cdot 1024} 5^n \bmod p$,
$3371 = 5^{192\cdot 1024} 5^n \bmod p$,
$3593 = 5^{626\cdot 1024} 5^n \bmod p$,
$4960 = 5^{663\cdot 1024} 5^n \bmod p$,
$5218 = 5^{376\cdot 1024} 5^n \bmod p$, ...,
$999675 = 5^{344\cdot 1024} 5^n \bmod p$.

Look up $5^1 \bmod p$, $5^2 \bmod p$, $5^3 \bmod p$, etc. in this table.

$5^{755} \bmod p = 966603$; find
$966603 = 5^{332\cdot 1024} 5^n \bmod p$
in the table of targets;
so $755 = 332 \cdot 1024 + n \bmod p - 1$;
deduce $n = 660789$.

Increase 100 to $\approx \sqrt{p}$.

$2\sqrt{p}$ mults

one DL problem!

baby-step-giant-step

-logarithm algorithm."

e: $p = 1000003$,

$p = 262682$.

e $5^{1024} \bmod p = 58588$.

mpute 1000 targets:

mod $p = 966849$,

^{n} mod $p = 579277$,

^{n} mod $p = 579062$, . . . ,

$^{24}5^{n} \bmod p = 321705$.

Build a sorted table of targets:

$2573 = 5^{430\cdot1024}5^{n} \bmod p$,

$3371 = 5^{192\cdot1024}5^{n} \bmod p$,

$3593 = 5^{626\cdot1024}5^{n} \bmod p$,

$4960 = 5^{663\cdot1024}5^{n} \bmod p$,

$5218 = 5^{376\cdot1024}5^{n} \bmod p$, . . . ,

$999675 = 5^{344\cdot1024}5^{n} \bmod p$.

Look up $5^1 \bmod p$, $5^2 \bmod p$, $5^3 \bmod p$, etc. in this table.

$5^{755} \bmod p = 966603$; find $966603 = 5^{332\cdot1024}5^{n} \bmod p$ in the table of targets; so $755 = 332\cdot1024 + n \bmod p-1$; deduce $n = 660789$.

Eliminat

Improved

$x_{i+1} = 5$

$x_{i+1} = 3$

$x_{i+1} = 5$

Then x_i

where $(a$

$(a_{i+1}, b_i$

$(a_{i+1}, b_i$

$(a_{i+1}, b_i$

Search f

$x_1 = x_2$

$x_4 = x_8$

Deduce

00 to $\approx \sqrt{p}$.
s
roblem!

-giant-step
algorithm."

00003,
32.

od $p = 58588$.
00 targets:
966849,
= 579277,
= 579062, ...,
$p = 321705$.

Build a sorted table of targets:
$2573 = 5^{430\cdot 1024} 5^n \bmod p$,
$3371 = 5^{192\cdot 1024} 5^n \bmod p$,
$3593 = 5^{626\cdot 1024} 5^n \bmod p$,
$4960 = 5^{663\cdot 1024} 5^n \bmod p$,
$5218 = 5^{376\cdot 1024} 5^n \bmod p$, ...,
$999675 = 5^{344\cdot 1024} 5^n \bmod p$.

Look up $5^1 \bmod p$, $5^2 \bmod p$,
$5^3 \bmod p$, etc. in this table.

$5^{755} \bmod p = 966603$; find
$966603 = 5^{332\cdot 1024} 5^n \bmod p$
in the table of targets;
so $755 = 332 \cdot 1024 + n \bmod p - 1$;
deduce $n = 660789$.

Eliminating storag

Improved method:
$x_{i+1} = 5x_i \bmod p$
$x_{i+1} = x_i^2 \bmod p$
$x_{i+1} = 5^n x_i \bmod$

Then $x_i = 5^{a_i n + b_i}$
where $(a_0, b_0) = ($
$(a_{i+1}, b_{i+1}) = (a_i$
$(a_{i+1}, b_{i+1}) = (2a$
$(a_{i+1}, b_{i+1}) = (a_i$

Search for a collisi
$x_1 = x_2$? $x_2 = x_4$
$x_4 = x_8$? $x_5 = x_1$
Deduce linear equa

$\sqrt{p}$.

p

."

588.

;

. . . ,

5.

Build a sorted table of targets:

$2573 = 5^{430 \cdot 1024} 5^n \bmod p$,

$3371 = 5^{192 \cdot 1024} 5^n \bmod p$,

$3593 = 5^{626 \cdot 1024} 5^n \bmod p$,

$4960 = 5^{663 \cdot 1024} 5^n \bmod p$,

$5218 = 5^{376 \cdot 1024} 5^n \bmod p$, . . . ,

$999675 = 5^{344 \cdot 1024} 5^n \bmod p$.

Look up $5^1 \bmod p$, $5^2 \bmod p$, $5^3 \bmod p$, etc. in this table.

$5^{755} \bmod p = 966603$; find $966603 = 5^{332 \cdot 1024} 5^n \bmod p$ in the table of targets; so $755 = 332 \cdot 1024 + n \bmod p - 1$; deduce $n = 660789$.

Eliminating storage

Improved method: Define $x_{\text{}}$

$x_{i+1} = 5x_i \bmod p$ if $x_i \in 3\mathbf{Z}$

$x_{i+1} = x_i^2 \bmod p$ if $x_i \in 2 +$

$x_{i+1} = 5^n x_i \bmod p$ otherwis

Then $x_i = 5^{a_i n + b_i} \bmod p$ where $(a_0, b_0) = (0, 0)$ and $(a_{i+1}, b_{i+1}) = (a_i, b_i + 1)$, o $(a_{i+1}, b_{i+1}) = (2a_i, 2b_i)$, or $(a_{i+1}, b_{i+1}) = (a_i + 1, b_i)$.

Search for a collision in x_i: $x_1 = x_2$? $x_2 = x_4$? $x_3 = x$ $x_4 = x_8$? $x_5 = x_{10}$? etc. Deduce linear equation for n

Build a sorted table of targets:

$2573 = 5^{430\cdot 1024} 5^n \bmod p$,
$3371 = 5^{192\cdot 1024} 5^n \bmod p$,
$3593 = 5^{626\cdot 1024} 5^n \bmod p$,
$4960 = 5^{663\cdot 1024} 5^n \bmod p$,
$5218 = 5^{376\cdot 1024} 5^n \bmod p, \ldots,$
$999675 = 5^{344\cdot 1024} 5^n \bmod p$.

Look up $5^1 \bmod p$, $5^2 \bmod p$, $5^3 \bmod p$, etc. in this table.

$5^{755} \bmod p = 966603$; find $966603 = 5^{332\cdot 1024} 5^n \bmod p$ in the table of targets; so $755 = 332 \cdot 1024 + n \bmod p - 1$; deduce $n = 660789$.

Eliminating storage

Improved method: Define $x_0 = 1$;
$x_{i+1} = 5x_i \bmod p$ if $x_i \in 3\mathbf{Z}$;
$x_{i+1} = x_i^2 \bmod p$ if $x_i \in 2 + 3\mathbf{Z}$;
$x_{i+1} = 5^n x_i \bmod p$ otherwise.

Then $x_i = 5^{a_i n + b_i} \bmod p$ where $(a_0, b_0) = (0, 0)$ and
$(a_{i+1}, b_{i+1}) = (a_i, b_i + 1)$, or
$(a_{i+1}, b_{i+1}) = (2a_i, 2b_i)$, or
$(a_{i+1}, b_{i+1}) = (a_i + 1, b_i)$.

Search for a collision in x_i:
$x_1 = x_2$? $x_2 = x_4$? $x_3 = x_6$?
$x_4 = x_8$? $x_5 = x_{10}$? etc.
Deduce linear equation for n.

sorted table of targets:

$5^{430\cdot 1024}5^n \bmod p$,

$5^{192\cdot 1024}5^n \bmod p$,

$5^{626\cdot 1024}5^n \bmod p$,

$5^{663\cdot 1024}5^n \bmod p$,

$5^{376\cdot 1024}5^n \bmod p$, . . . ,

$= 5^{344\cdot 1024}5^n \bmod p$.

$5^1 \bmod p$, $5^2 \bmod p$,

p, etc. in this table.

d $p = 966603$; find

$= 5^{332\cdot 1024}5^n \bmod p$

ble of targets;

$= 332\cdot 1024 + n \bmod p-1$;

$n = 660789$.

Eliminating storage

Improved method: Define $x_0 = 1$;

$x_{i+1} = 5x_i \bmod p$ if $x_i \in 3\mathbf{Z}$;

$x_{i+1} = x_i^2 \bmod p$ if $x_i \in 2 + 3\mathbf{Z}$;

$x_{i+1} = 5^n x_i \bmod p$ otherwise.

Then $x_i = 5^{a_i n + b_i} \bmod p$

where $(a_0, b_0) = (0, 0)$ and

$(a_{i+1}, b_{i+1}) = (a_i, b_i + 1)$, or

$(a_{i+1}, b_{i+1}) = (2a_i, 2b_i)$, or

$(a_{i+1}, b_{i+1}) = (a_i + 1, b_i)$.

Search for a collision in x_i:

$x_1 = x_2$? $x_2 = x_4$? $x_3 = x_6$?

$x_4 = x_8$? $x_5 = x_{10}$? etc.

Deduce linear equation for n.

The x_i's

typically

Example

Modulo

$x_1 = 5^n$

$x_2 = 5^{2}$

$x_3 = 5^{2}$

$x_4 = 5^{2}$

$x_5 = 5^{2}$

$x_6 = 5^{2}$

$x_7 = 5^{4}$

$x_8 = 5^{4}$

etc.

le of targets:

^{n} mod p,

^{n} mod p,

^{n} mod p,

^{n} mod p,

^{n} mod p, . . . ,

$^4 5^n$ mod p.

, 5^2 mod p,

this table.

603; find

$^4 5^n$ mod p

gets;

$4+n$ mod $p-1$;

39.

Eliminating storage

Improved method: Define $x_0 = 1$;
$x_{i+1} = 5x_i \bmod p$ if $x_i \in 3\mathbf{Z}$;
$x_{i+1} = x_i^2 \bmod p$ if $x_i \in 2 + 3\mathbf{Z}$;
$x_{i+1} = 5^n x_i \bmod p$ otherwise.

Then $x_i = 5^{a_i n + b_i} \bmod p$
where $(a_0, b_0) = (0, 0)$ and
$(a_{i+1}, b_{i+1}) = (a_i, b_i + 1)$, or
$(a_{i+1}, b_{i+1}) = (2a_i, 2b_i)$, or
$(a_{i+1}, b_{i+1}) = (a_i + 1, b_i)$.

Search for a collision in x_i:
$x_1 = x_2$? $x_2 = x_4$? $x_3 = x_6$?
$x_4 = x_8$? $x_5 = x_{10}$? etc.
Deduce linear equation for n.

The x_i's enter a c

typically within $\approx$

Example: 100000

Modulo 1000003:

$x_1 = 5^n = 26268$

$x_2 = 5^{2n} = 2626$

$x_3 = 5^{2n+1} = 5{\cdot}6$

$x_4 = 5^{2n+2} = 5{\cdot}1$

$x_5 = 5^{2n+3} = 5{\cdot}6$

$x_6 = 5^{2n+4} = 5{\cdot}2$

$x_7 = 5^{4n+8} = 324$

$x_8 = 5^{4n+9} = 5{\cdot}7$

etc.

ets:

$\ldots,$

$p.$

$p,$

p

$l\ p-1;$

Eliminating storage

Improved method: Define $x_0 = 1$;
$x_{i+1} = 5x_i \bmod p$ if $x_i \in 3\mathbf{Z}$;
$x_{i+1} = x_i^2 \bmod p$ if $x_i \in 2 + 3\mathbf{Z}$;
$x_{i+1} = 5^n x_i \bmod p$ otherwise.

Then $x_i = 5^{a_i n + b_i} \bmod p$
where $(a_0, b_0) = (0, 0)$ and
$(a_{i+1}, b_{i+1}) = (a_i, b_i + 1)$, or
$(a_{i+1}, b_{i+1}) = (2a_i, 2b_i)$, or
$(a_{i+1}, b_{i+1}) = (a_i + 1, b_i)$.

Search for a collision in x_i:
$x_1 = x_2$? $x_2 = x_4$? $x_3 = x_6$?
$x_4 = x_8$? $x_5 = x_{10}$? etc.
Deduce linear equation for n.

The x_i's enter a cycle,
typically within $\approx \sqrt{p}$ steps.

Example: 1000003, 262682.

Modulo 1000003:
$x_1 = 5^n = 262682.$
$x_2 = 5^{2n} = 262682^2 = 6261$
$x_3 = 5^{2n+1} = 5 \cdot 626121 = 1$
$x_4 = 5^{2n+2} = 5 \cdot 130596 = 6$
$x_5 = 5^{2n+3} = 5 \cdot 652980 = 2$
$x_6 = 5^{2n+4} = 5 \cdot 264891 = 3$
$x_7 = 5^{4n+8} = 324452^2 = 78$
$x_8 = 5^{4n+9} = 5 \cdot 784500 = 9$
etc.

Eliminating storage

Improved method: Define $x_0 = 1$;
$x_{i+1} = 5x_i \bmod p$ if $x_i \in 3\mathbf{Z}$;
$x_{i+1} = x_i^2 \bmod p$ if $x_i \in 2 + 3\mathbf{Z}$;
$x_{i+1} = 5^n x_i \bmod p$ otherwise.

Then $x_i = 5^{a_i n + b_i} \bmod p$
where $(a_0, b_0) = (0, 0)$ and
$(a_{i+1}, b_{i+1}) = (a_i, b_i + 1)$, or
$(a_{i+1}, b_{i+1}) = (2a_i, 2b_i)$, or
$(a_{i+1}, b_{i+1}) = (a_i + 1, b_i)$.

Search for a collision in x_i:
$x_1 = x_2$? $x_2 = x_4$? $x_3 = x_6$?
$x_4 = x_8$? $x_5 = x_{10}$? etc.
Deduce linear equation for n.

The x_i's enter a cycle,
typically within $\approx \sqrt{p}$ steps.

Example: 1000003, 262682.

Modulo 1000003:
$x_1 = 5^n = 262682$.
$x_2 = 5^{2n} = 262682^2 = 626121$.
$x_3 = 5^{2n+1} = 5 \cdot 626121 = 130596$.
$x_4 = 5^{2n+2} = 5 \cdot 130596 = 652980$.
$x_5 = 5^{2n+3} = 5 \cdot 652980 = 264891$.
$x_6 = 5^{2n+4} = 5 \cdot 264891 = 324452$.
$x_7 = 5^{4n+8} = 324452^2 = 784500$.
$x_8 = 5^{4n+9} = 5 \cdot 784500 = 922491$.
etc.

ing storage

d method: Define $x_0 = 1$;

$5x_i$ mod p if $x_i \in 3\mathbf{Z}$;

x_i^2 mod p if $x_i \in 2 + 3\mathbf{Z}$;

$5^n x_i$ mod p otherwise.

$= 5^{a_i n + b_i}$ mod p

$a_0, b_0) = (0, 0)$ and

$_{+1}) = (a_i, b_i + 1)$, or

$_{+1}) = (2a_i, 2b_i)$, or

$_{+1}) = (a_i + 1, b_i)$.

or a collision in x_i:

? $x_2 = x_4$? $x_3 = x_6$?

? $x_5 = x_{10}$? etc.

linear equation for n.

The x_i's enter a cycle,
typically within $\approx \sqrt{p}$ steps.

Example: 1000003, 262682.

Modulo 1000003:

$x_1 = 5^n = 262682.$

$x_2 = 5^{2n} = 262682^2 = 626121.$

$x_3 = 5^{2n+1} = 5 \cdot 626121 = 130596.$

$x_4 = 5^{2n+2} = 5 \cdot 130596 = 652980.$

$x_5 = 5^{2n+3} = 5 \cdot 652980 = 264891.$

$x_6 = 5^{2n+4} = 5 \cdot 264891 = 324452.$

$x_7 = 5^{4n+8} = 324452^2 = 784500.$

$x_8 = 5^{4n+9} = 5 \cdot 784500 = 922491.$

etc.

$x_{1785} =$

$x_{3570} =$

(Cycle le

Conclud

249847n

388795n

so $n \equiv 1$

Only 6 p

Try each

Find tha

for $n =$

for $n =$

e

Define $x_0 = 1$;

if $x_i \in 3\mathbf{Z}$;

if $x_i \in 2 + 3\mathbf{Z}$;

p otherwise.

^{i} mod p

$0, 0)$ and

$, b_i + 1)$, or

$_i, 2b_i)$, or

$+ 1, b_i)$.

on in x_i:

$?$ $x_3 = x_6$?

$_0$? etc.

ation for n.

The x_i's enter a cycle,
typically within $\approx \sqrt{p}$ steps.

Example: 1000003, 262682.

Modulo 1000003:

$x_1 = 5^n = 262682.$

$x_2 = 5^{2n} = 262682^2 = 626121.$

$x_3 = 5^{2n+1} = 5{\cdot}626121 = 130596.$

$x_4 = 5^{2n+2} = 5{\cdot}130596 = 652980.$

$x_5 = 5^{2n+3} = 5{\cdot}652980 = 264891.$

$x_6 = 5^{2n+4} = 5{\cdot}264891 = 324452.$

$x_7 = 5^{4n+8} = 324452^2 = 784500.$

$x_8 = 5^{4n+9} = 5{\cdot}784500 = 922491.$

etc.

$x_{1785} = 5^{249847n+}$

$x_{3570} = 5^{388795n+}$

(Cycle length is 35

Conclude that

$249847n + 759123$

$388795n + 632781$

so $n \equiv 160788 \quad ($

Only 6 possible n'

Try each of them.

Find that 5^n mod

for $n = 160788 +$

for $n = 660789$.

$_0 = 1$;

Z;

- 3**Z**;

se.

or

$_6$?

n.

The x_i's enter a cycle,
typically within $\approx \sqrt{p}$ steps.

Example: 1000003, 262682.

Modulo 1000003:

$x_1 = 5^n = 262682$.

$x_2 = 5^{2n} = 262682^2 = 626121$.

$x_3 = 5^{2n+1} = 5 \cdot 626121 = 130596$.

$x_4 = 5^{2n+2} = 5 \cdot 130596 = 652980$.

$x_5 = 5^{2n+3} = 5 \cdot 652980 = 264891$.

$x_6 = 5^{2n+4} = 5 \cdot 264891 = 324452$.

$x_7 = 5^{4n+8} = 324452^2 = 784500$.

$x_8 = 5^{4n+9} = 5 \cdot 784500 = 922491$.

etc.

$x_{1785} = 5^{249847n+759123} = 5$

$x_{3570} = 5^{388795n+632781} = 5$

(Cycle length is 357.)

Conclude that
$249847n + 759123 \equiv$
$388795n + 632781 \pmod{p}$
so $n \equiv 160788 \pmod{(p -}$

Only 6 possible n's.
Try each of them.
Find that $5^n \bmod p = 26268$
for $n = 160788 + 3(p-1)/$
for $n = 660789$.

The x_i's enter a cycle, typically within $\approx \sqrt{p}$ steps.

Example: 1000003, 262682.

Modulo 1000003:
$x_1 = 5^n = 262682.$
$x_2 = 5^{2n} = 262682^2 = 626121.$
$x_3 = 5^{2n+1} = 5 \cdot 626121 = 130596.$
$x_4 = 5^{2n+2} = 5 \cdot 130596 = 652980.$
$x_5 = 5^{2n+3} = 5 \cdot 652980 = 264891.$
$x_6 = 5^{2n+4} = 5 \cdot 264891 = 324452.$
$x_7 = 5^{4n+8} = 324452^2 = 784500.$
$x_8 = 5^{4n+9} = 5 \cdot 784500 = 922491.$
etc.

$x_{1785} = 5^{249847n+759123} = 555013.$
$x_{3570} = 5^{388795n+632781} = 555013.$

(Cycle length is 357.)

Conclude that
$249847n + 759123 \equiv$
$388795n + 632781 \pmod{p-1}$,
so $n \equiv 160788 \pmod{(p-1)/6}$.

Only 6 possible n's.
Try each of them.
Find that $5^n \bmod p = 262682$
for $n = 160788 + 3(p-1)/6$, i.e.,
for $n = 660789$.

s enter a cycle,

within $\approx \sqrt{p}$ steps.

e: 1000003, 262682.

1000003:

$= 262682.$

$^{n} = 262682^2 = 626121.$

$^{n+1} = 5 \cdot 626121 = 130596.$

$^{n+2} = 5 \cdot 130596 = 652980.$

$^{n+3} = 5 \cdot 652980 = 264891.$

$^{n+4} = 5 \cdot 264891 = 324452.$

$^{n+8} = 324452^2 = 784500.$

$^{n+9} = 5 \cdot 784500 = 922491.$

$x_{1785} = 5^{249847n+759123} = 555013.$
$x_{3570} = 5^{388795n+632781} = 555013.$

(Cycle length is 357.)

Conclude that
$249847n + 759123 \equiv$
$388795n + 632781 \pmod{p-1}$,
so $n \equiv 160788 \pmod{(p-1)/6}$.

Only 6 possible n's.
Try each of them.
Find that $5^n \bmod p = 262682$
for $n = 160788 + 3(p-1)/6$, i.e.,
for $n = 660789$.

This is “

Optimize

Another

“Pollard

Can para

“van Oo

DL using

Bottom

distribut

have cha

of findin

With 2^{90}

have cha

Negligib

ycle,

$\sqrt{p}$ steps.

3, 262682.

2.

$2^2 = 626121$.

$26121 = 130596$.

$30596 = 652980$.

$52980 = 264891$.

$64891 = 324452$.

$452^2 = 784500$.

$84500 = 922491$.

$$x_{1785} = 5^{249847n+759123} = 555013.$$
$$x_{3570} = 5^{388795n+632781} = 555013.$$

(Cycle length is 357.)

Conclude that
$249847n + 759123 \equiv$
$388795n + 632781 \pmod{p-1}$,
so $n \equiv 160788 \pmod{(p-1)/6}$.

Only 6 possible n's.
Try each of them.
Find that $5^n \bmod p = 262682$
for $n = 160788 + 3(p-1)/6$, i.e.,
for $n = 660789$.

This is "Pollard's
Optimized: $\approx \sqrt{p}$
Another method, s
"Pollard's kangaro

Can parallelize bot
"van Oorschot/Wi
DL using distingui

Bottom line: With
distributed across
have chance $\approx c^2/$
of finding n from

With 2^{90} mults (a
have chance $\approx 2^{18}$
Negligible if, e.g.,

121.

30596.

52980.

64891.

24452.

34500.

22491.

$x_{1785} = 5^{249847n+759123} = 555013.$

$x_{3570} = 5^{388795n+632781} = 555013.$

(Cycle length is 357.)

Conclude that
$249847n + 759123 \equiv$
$388795n + 632781 \pmod{p-1}$,
so $n \equiv 160788 \pmod{(p-1)/6}$.

Only 6 possible n's.
Try each of them.
Find that $5^n \bmod p = 262682$
for $n = 160788 + 3(p-1)/6$, i.e.,
for $n = 660789$.

This is "Pollard's rho metho
Optimized: $\approx \sqrt{p}$ mults.
Another method, similar spe
"Pollard's kangaroo method

Can parallelize both method
"van Oorschot/Wiener paral
DL using distinguished point

Bottom line: With c mults,
distributed across many core
have chance $\approx c^2/p$
of finding n from $5^n \bmod p$.

With 2^{90} mults (a few years
have chance $\approx 2^{180}/p$.
Negligible if, e.g., $p \approx 2^{256}$.

$x_{1785} = 5^{249847n+759123} = 555013.$
$x_{3570} = 5^{388795n+632781} = 555013.$

(Cycle length is 357.)

Conclude that
$249847n + 759123 \equiv$
$388795n + 632781 \pmod{p-1}$,
so $n \equiv 160788 \pmod{(p-1)/6}$.

Only 6 possible n's.
Try each of them.
Find that $5^n \bmod p = 262682$
for $n = 160788 + 3(p-1)/6$, i.e.,
for $n = 660789$.

This is "Pollard's rho method."
Optimized: $\approx \sqrt{p}$ mults.
Another method, similar speed:
"Pollard's kangaroo method."

Can parallelize both methods.
"van Oorschot/Wiener parallel
DL using distinguished points."

Bottom line: With c mults,
distributed across many cores,
have chance $\approx c^2/p$
of finding n from $5^n \bmod p$.

With 2^{90} mults (a few years?),
have chance $\approx 2^{180}/p$.
Negligible if, e.g., $p \approx 2^{256}$.

$5^{249847n+759123} = 555013.$
$5^{388795n+632781} = 555013.$

ength is 357.)

e that
$n + 759123 \equiv$
$n + 632781 \pmod{p-1}$,
$160788 \pmod{(p-1)/6}$.

possible n's.
n of them.
t $5^n \bmod p = 262682$
$160788 + 3(p-1)/6$, i.e.,
660789.

This is "Pollard's rho method."
Optimized: $\approx \sqrt{p}$ mults.
Another method, similar speed:
"Pollard's kangaroo method."

Can parallelize both methods.
"van Oorschot/Wiener parallel DL using distinguished points."

Bottom line: With c mults, distributed across many cores, have chance $\approx c^2/p$ of finding n from $5^n \bmod p$.

With 2^{90} mults (a few years?), have chance $\approx 2^{180}/p$.
Negligible if, e.g., $p \approx 2^{256}$.

Factors

Assume

Given x,

5^a has o
x^a is a p
Comput

5^b has o
$x/5^\ell$ is a
Comput

Then x

$^{759123} = 555013.$
$^{632781} = 555013.$

$57.)$

$3 \equiv$
$1 \pmod{p-1},$
$\bmod (p-1)/6).$

s.

$p = 262682$
$3(p-1)/6$, i.e.,

This is "Pollard's rho method."
Optimized: $\approx \sqrt{p}$ mults.
Another method, similar speed:
"Pollard's kangaroo method."

Can parallelize both methods.
"van Oorschot/Wiener parallel
DL using distinguished points."

Bottom line: With c mults,
distributed across many cores,
have chance $\approx c^2/p$
of finding n from 5^n mod p.

With 2^{90} mults (a few years?),
have chance $\approx 2^{180}/p$.
Negligible if, e.g., $p \approx 2^{256}$.

Factors of the gro

Assume 5 has orde

Given x, a power o

5^a has order b, and
x^a is a power of 5
Compute $\ell = \log_{5}$

5^b has order a, an
$x/5^\ell$ is a power of
Compute $m = \log$

Then $x = 5^{\ell+mb}$.

555013.

555013.

$-1)$,

$1)/6)$.

32

6, i.e.,

This is "Pollard's rho method."
Optimized: $\approx \sqrt{p}$ mults.
Another method, similar speed:
"Pollard's kangaroo method."

Can parallelize both methods.
"van Oorschot/Wiener parallel
DL using distinguished points."

Bottom line: With c mults,
distributed across many cores,
have chance $\approx c^2/p$
of finding n from $5^n \bmod p$.

With 2^{90} mults (a few years?),
have chance $\approx 2^{180}/p$.
Negligible if, e.g., $p \approx 2^{256}$.

Factors of the group order

Assume 5 has order ab.

Given x, a power of 5:

5^a has order b, and
x^a is a power of 5^a.
Compute $\ell = \log_{5^a} x^a$.

5^b has order a, and
$x/5^\ell$ is a power of 5^b.
Compute $m = \log_{5^b}(x/5^\ell)$.

Then $x = 5^{\ell+mb}$.

This is "Pollard's rho method."
Optimized: $\approx \sqrt{p}$ mults.
Another method, similar speed:
"Pollard's kangaroo method."

Can parallelize both methods.
"van Oorschot/Wiener parallel DL using distinguished points."

Bottom line: With c mults, distributed across many cores, have chance $\approx c^2/p$ of finding n from 5^n mod p.

With 2^{90} mults (a few years?), have chance $\approx 2^{180}/p$.
Negligible if, e.g., $p \approx 2^{256}$.

Factors of the group order

Assume 5 has order ab.

Given x, a power of 5:

5^a has order b, and x^a is a power of 5^a.
Compute $\ell = \log_{5^a} x^a$.

5^b has order a, and $x/5^\ell$ is a power of 5^b.
Compute $m = \log_{5^b}(x/5^\ell)$.

Then $x = 5^{\ell+mb}$.

"Pollard's rho method."
ed: $\approx \sqrt{p}$ mults.
method, similar speed:
's kangaroo method."

allelize both methods.
rschot/Wiener parallel
g distinguished points."

line: With c mults,
ed across many cores,
ance $\approx c^2/p$
g n from 5^n mod p.

^{0} mults (a few years?),
ance $\approx 2^{180}/p$.
le if, e.g., $p \approx 2^{256}$.

Factors of the group order

Assume 5 has order ab.

Given x, a power of 5:

5^a has order b, and
x^a is a power of 5^a.
Compute $\ell = \log_{5^a} x^a$.

5^b has order a, and
$x/5^\ell$ is a power of 5^b.
Compute $m = \log_{5^b}(x/5^\ell)$.

Then $x = 5^{\ell+mb}$.

This "Po
converts
an order
and a fe

e.g. $p =$
$p - 1 =$
Compute
Compute
Compute
Then x

Use rho:
Better if
apply Po

rho method."
mults.
similar speed:
o method."

th methods.
iener parallel
shed points."

c mults,
many cores,
$/p$
$5^n \bmod p$.

few years?),
$^{30}/p$.
$p \approx 2^{256}$.

Factors of the group order

Assume 5 has order ab.

Given x, a power of 5:

5^a has order b, and
x^a is a power of 5^a.
Compute $\ell = \log_{5^a} x^a$.

5^b has order a, and
$x/5^\ell$ is a power of 5^b.
Compute $m = \log_{5^b}(x/5^\ell)$.

Then $x = 5^{\ell + mb}$.

This "Pohlig-Hell
converts an order-
an order-a DL, an
and a few exponen

e.g. $p = 1000003,$
$p - 1 = 6b$ where
Compute $\log_{5^6}(x^6$
Compute $x/5^{16078}$
Compute $\log_{5^b} 10$
Then $x = 5^{160788-}$

Use rho: $\approx \sqrt{a} +$
Better if ab factor
apply Pohlig-Hellm

od.”

eed:

.”

s.

llel

ts.”

es,

.

?),

Factors of the group order

Assume 5 has order ab.

Given x, a power of 5:

5^a has order b, and
x^a is a power of 5^a.
Compute $\ell = \log_{5^a} x^a$.

5^b has order a, and
$x/5^\ell$ is a power of 5^b.
Compute $m = \log_{5^b}(x/5^\ell)$.

Then $x = 5^{\ell+mb}$.

This “Pohlig-Hellman metho
converts an order-ab DL int
an order-a DL, an order-b D
and a few exponentiations.

e.g. $p = 1000003$, $x = 262$
$p - 1 = 6b$ where $b = 1666$
Compute $\log_{5^6}(x^6) = 1607$
Compute $x/5^{160788} = 1000$
Compute $\log_{5^b} 1000002 = 3$
Then $x = 5^{160788+3b} = 5^{660}$

Use rho: $\approx \sqrt{a} + \sqrt{b}$ mults.
Better if ab factors further:
apply Pohlig-Hellman recurs

Factors of the group order

Assume 5 has order ab.

Given x, a power of 5:

5^a has order b, and
x^a is a power of 5^a.
Compute $\ell = \log_{5^a} x^a$.

5^b has order a, and
$x/5^\ell$ is a power of 5^b.
Compute $m = \log_{5^b}(x/5^\ell)$.

Then $x = 5^{\ell+mb}$.

This "Pohlig-Hellman method" converts an order-ab DL into an order-a DL, an order-b DL, and a few exponentiations.

e.g. $p = 1000003$, $x = 262682$:
$p - 1 = 6b$ where $b = 166667$.
Compute $\log_{5^6}(x^6) = 160788$.
Compute $x/5^{160788} = 1000002$.
Compute $\log_{5^b} 1000002 = 3$.
Then $x = 5^{160788+3b} = 5^{660789}$.

Use rho: $\approx \sqrt{a} + \sqrt{b}$ mults.
Better if ab factors further: apply Pohlig-Hellman recursively.

of the group order

5 has order ab.

a power of 5:

rder b, and
ower of 5^a.
e $\ell = \log_{5^a} x^a$.

rder a, and
a power of 5^b.
e $m = \log_{5^b}(x/5^\ell)$.

$= 5^{\ell+mb}$.

This "Pohlig-Hellman method" converts an order-ab DL into an order-a DL, an order-b DL, and a few exponentiations.

e.g. $p = 1000003$, $x = 262682$:
$p - 1 = 6b$ where $b = 166667$.
Compute $\log_{5^6}(x^6) = 160788$.
Compute $x/5^{160788} = 1000002$.
Compute $\log_{5^b} 1000002 = 3$.
Then $x = 5^{160788+3b} = 5^{660789}$.

Use rho: $\approx \sqrt{a} + \sqrt{b}$ mults.
Better if ab factors further:
apply Pohlig-Hellman recursively.

All of th
apply to

An ellipt
has $\approx q$
so can c
$\approx \sqrt{q}$ el
Need qu

If larges
of numb
is much
then Po
compute
Need lar
or chang

up order

er ab.

of 5:

d
a.
$_a\ x^a$.

d
5^b.
$_{5^b}(x/5^\ell)$.

This "Pohlig-Hellman method" converts an order-ab DL into an order-a DL, an order-b DL, and a few exponentiations.

e.g. $p = 1000003$, $x = 262682$:
$p - 1 = 6b$ where $b = 166667$.
Compute $\log_{5^6}(x^6) = 160788$.
Compute $x/5^{160788} = 1000002$.
Compute $\log_{5^b} 1000002 = 3$.
Then $x = 5^{160788+3b} = 5^{660789}$.

Use rho: $\approx \sqrt{a} + \sqrt{b}$ mults.
Better if ab factors further: apply Pohlig-Hellman recursively.

All of the techniqu
apply to elliptic cu

An elliptic curve o
has $\approx q + 1$ points
so can compute E
$\approx \sqrt{q}$ elliptic-curv
Need quite large q

If largest prime di
of number of poin
is much smaller th
then Pohlig-Hellm
computes ECDL m
Need larger q;
or change choice c

This "Pohlig-Hellman method" converts an order-ab DL into an order-a DL, an order-b DL, and a few exponentiations.

e.g. $p = 1000003$, $x = 262682$: $p - 1 = 6b$ where $b = 166667$. Compute $\log_{5^6}(x^6) = 160788$. Compute $x/5^{160788} = 1000002$. Compute $\log_{5^b} 1000002 = 3$. Then $x = 5^{160788+3b} = 5^{660789}$.

Use rho: $\approx \sqrt{a} + \sqrt{b}$ mults. Better if ab factors further: apply Pohlig-Hellman recursively.

All of the techniques so far apply to elliptic curves.

An elliptic curve over $\mathbf{F}_q$ has $\approx q + 1$ points so can compute ECDL using $\approx \sqrt{q}$ elliptic-curve adds. Need quite large q.

If largest prime divisor of number of points is much smaller than q then Pohlig-Hellman method computes ECDL more quick Need larger q; or change choice of curve.

This "Pohlig-Hellman method" converts an order-ab DL into an order-a DL, an order-b DL, and a few exponentiations.

e.g. $p = 1000003$, $x = 262682$:
$p - 1 = 6b$ where $b = 166667$.
Compute $\log_{5^6}(x^6) = 160788$.
Compute $x/5^{160788} = 1000002$.
Compute $\log_{5^b} 1000002 = 3$.
Then $x = 5^{160788+3b} = 5^{660789}$.

Use rho: $\approx \sqrt{a} + \sqrt{b}$ mults.
Better if ab factors further: apply Pohlig-Hellman recursively.

All of the techniques so far apply to elliptic curves.

An elliptic curve over $\mathbf{F}_q$ has $\approx q + 1$ points so can compute ECDL using $\approx \sqrt{q}$ elliptic-curve adds.
Need quite large q.

If largest prime divisor of number of points is much smaller than q then Pohlig-Hellman method computes ECDL more quickly.
Need larger q;
or change choice of curve.

ohlig-Hellman method"
an order-ab DL into
-a DL, an order-b DL,
w exponentiations.

$= 1000003$, $x = 262682$:
$6b$ where $b = 166667$.
e $\log_{5^6}(x^6) = 160788$.
e $x/5^{160788} = 1000002$.
e $\log_{5^b} 1000002 = 3$.
$= 5^{160788+3b} = 5^{660789}$.

$\approx \sqrt{a} + \sqrt{b}$ mults.
f ab factors further:
ohlig-Hellman recursively.

All of the techniques so far
apply to elliptic curves.

An elliptic curve over $\mathbf{F}_q$
has $\approx q + 1$ points
so can compute ECDL using
$\approx \sqrt{q}$ elliptic-curve adds.
Need quite large q.

If largest prime divisor
of number of points
is much smaller than q
then Pohlig-Hellman method
computes ECDL more quickly.
Need larger q;
or change choice of curve.

Index ca

Have ge
group el
Deduced
from rar

Index ca
discrete-
in a diffe

Example
Can com
$-3/(p -$
so $-3^1 \equiv$
so $\log_5($
$6 \log_5 2$

man method"
ab DL into
order-b DL,
ntiations.

$x = 262682$:
$b = 166667$.
$) = 160788$.
$^{88} = 1000002$.
$00002 = 3$.
$^{+3b} = 5^{660789}$.

$\sqrt{b}$ mults.
s further:
nan recursively.

All of the techniques so far apply to elliptic curves.

An elliptic curve over $\mathbf{F}_q$ has $\approx q+1$ points so can compute ECDL using $\approx \sqrt{q}$ elliptic-curve adds. Need quite large q.

If largest prime divisor of number of points is much smaller than q then Pohlig-Hellman method computes ECDL more quickly. Need larger q; or change choice of curve.

Index calculus

Have generated m
group elements 5^{a}
Deduced equation
from random colli

Index calculus obt
discrete-logarithm
in a different way.

Example for $p = 1$
Can completely fa
$-3/(p-3)$ as -3
so $-3^1 \equiv 2^6 5^6$ (
so $\log_5(-1) + \log$
$6\log_5 2 + 6\log_5 5$

od”
o
L,

682:
57.
38.
002.
.
789.

ively.

All of the techniques so far apply to elliptic curves.

An elliptic curve over $\mathbf{F}_q$ has $\approx q+1$ points so can compute ECDL using $\approx \sqrt{q}$ elliptic-curve adds. Need quite large q.

If largest prime divisor of number of points is much smaller than q then Pohlig-Hellman method computes ECDL more quickly. Need larger q; or change choice of curve.

Index calculus

Have generated many group elements 5^{an+b} mod p Deduced equations for n from random collisions.

Index calculus obtains discrete-logarithm equations in a different way.

Example for $p = 1000003$:
Can completely factor
$-3/(p-3)$ as $-3^1/2^6 5^6$ in
so $-3^1 \equiv 2^6 5^6 \pmod{p}$
so $\log_5(-1) + \log_5 3 \equiv$
$6\log_5 2 + 6\log_5 5 \quad (\text{mod } p$

All of the techniques so far apply to elliptic curves.

An elliptic curve over $\mathbf{F}_q$ has $\approx q+1$ points so can compute ECDL using $\approx \sqrt{q}$ elliptic-curve adds. Need quite large q.

If largest prime divisor of number of points is much smaller than q then Pohlig-Hellman method computes ECDL more quickly. Need larger q; or change choice of curve.

Index calculus

Have generated many group elements $5^{an+b} \bmod p$. Deduced equations for n from random collisions.

Index calculus obtains discrete-logarithm equations in a different way.

Example for $p = 1000003$: Can completely factor $-3/(p-3)$ as $-3^1/2^6 5^6$ in $\mathbf{Q}$ so $-3^1 \equiv 2^6 5^6 \pmod{p}$ so $\log_5(-1) + \log_5 3 \equiv 6\log_5 2 + 6\log_5 5 \pmod{p-1}$.

e techniques so far
elliptic curves.

tic curve over $\mathbf{F}_q$
$+ 1$ points
ompute ECDL using
liptic-curve adds.
ite large q.

t prime divisor
er of points
smaller than q
hlig-Hellman method
es ECDL more quickly.
ger q;
ge choice of curve.

Index calculus

Have generated many
group elements $5^{an+b} \bmod p$.
Deduced equations for n
from random collisions.

Index calculus obtains
discrete-logarithm equations
in a different way.

Example for $p = 1000003$:
Can completely factor
$-3/(p-3)$ as $-3^1/2^6 5^6$ in $\mathbf{Q}$
so $-3^1 \equiv 2^6 5^6 \pmod{p}$
so $\log_5(-1) + \log_5 3 \equiv$
$6 \log_5 2 + 6 \log_5 5 \pmod{p-1}$.

Can com
as $2^1 31^1$
so $\log_5 2$
$\log_5 3 +$
$\log_5 19 -$

Try to c
$1/(p+1$
Find fac
as produ
$2, 3, 5, 7$
for each
-5100,
-403, $-$
62, 957,

ues so far
urves.

ver $\mathbf{F}_q$
s
CDL using
e adds.
.

visor
ts
an q
an method
nore quickly.

of curve.

<u>Index calculus</u>

Have generated many group elements $5^{an+b} \bmod p$. Deduced equations for n from random collisions.

Index calculus obtains discrete-logarithm equations in a different way.

Example for $p = 1000003$: Can completely factor $-3/(p-3)$ as $-3^1/2^6 5^6$ in $\mathbf{Q}$ so $-3^1 \equiv 2^6 5^6 \pmod{p}$ so $\log_5(-1) + \log_5 3 \equiv$ $6\log_5 2 + 6\log_5 5 \pmod{p-1}$.

Can completely fa
as $2^1 31^1/3^1 5^1 11^2$
so $\log_5 2 + \log_5 31$
$\log_5 3 + \log_5 5 + 2$
$\log_5 19 + \log_5 29$

Try to completely
$1/(p+1)$, $2/(p+$
Find factorization
as product of pow
$2, 3, 5, 7, 11, 13, 17$
for each of the fol
-5100, -4675, $-$
-403, -368, -14
62, 957, 2912, 385

g

d

ly.

Index calculus

Have generated many group elements $5^{an+b} \bmod p$. Deduced equations for n from random collisions.

Index calculus obtains discrete-logarithm equations in a different way.

Example for $p = 1000003$: Can completely factor $-3/(p-3)$ as $-3^1/2^6 5^6$ in $\mathbf{Q}$ so $-3^1 \equiv 2^6 5^6 \pmod{p}$ so $\log_5(-1) + \log_5 3 \equiv$ $6\log_5 2 + 6\log_5 5 \pmod{p-1}$.

Can completely factor 62/(*p*
as $2^1 31^1/3^1 5^1 11^2 19^1 29^1$
so $\log_5 2 + \log_5 31 \equiv$
$\log_5 3 + \log_5 5 + 2\log_5 11 +$
$\log_5 19 + \log_5 29 \quad (\bmod\ p -$

Try to completely factor $1/(p+1)$, $2/(p+2)$, etc.
Find factorization of $a/(p +$
as product of powers of -1,
$2, 3, 5, 7, 11, 13, 17, 19, 23, 2$
for each of the following a's
-5100, -4675, -3128, -403, -368, -147, -3, 62, 957, 2912, 3857, 6877.

Index calculus

Have generated many group elements $5^{an+b} \bmod p$. Deduced equations for n from random collisions.

Index calculus obtains discrete-logarithm equations in a different way.

Example for $p = 1000003$:
Can completely factor $-3/(p-3)$ as $-3^1/2^6 5^6$ in $\mathbf{Q}$
so $-3^1 \equiv 2^6 5^6 \pmod p$
so $\log_5(-1) + \log_5 3 \equiv 6\log_5 2 + 6\log_5 5 \pmod{p-1}$.

Can completely factor $62/(p+62)$ as $2^1 31^1/3^1 5^1 11^2 19^1 29^1$
so $\log_5 2 + \log_5 31 \equiv \log_5 3 + \log_5 5 + 2\log_5 11 + \log_5 19 + \log_5 29 \pmod{p-1}$.

Try to completely factor $1/(p+1)$, $2/(p+2)$, etc.
Find factorization of $a/(p+a)$ as product of powers of -1, $2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31$ for each of the following a's:
-5100, -4675, -3128,
-403, -368, -147, -3,
62, 957, 2912, 3857, 6877.

lculus

nerated many
ements 5^{an+b} mod p.
l equations for n
ndom collisions.

lculus obtains
-logarithm equations
erent way.

e for $p = 1000003$:
npletely factor
- 3) as $-3^1/2^6 5^6$ in $\mathbf{Q}$
$\equiv 2^6 5^6 \pmod{p}$
$-1) + \log_5 3 \equiv$
$+ 6 \log_5 5 \pmod{p-1}$.

Can completely factor $62/(p+62)$
as $2^1 31^1 / 3^1 5^1 11^2 19^1 29^1$
so $\log_5 2 + \log_5 31 \equiv$
$\log_5 3 + \log_5 5 + 2 \log_5 11 +$
$\log_5 19 + \log_5 29 \pmod{p-1}$.

Try to completely factor
$1/(p+1)$, $2/(p+2)$, etc.
Find factorization of $a/(p+a)$
as product of powers of -1,
$2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31$
for each of the following a's:
-5100, -4675, -3128,
-403, -368, -147, -3,
62, 957, 2912, 3857, 6877.

Each co
produce

Now hav
for $\log_5$ 2
Free equ
$\log_5(-1$

By linea
$\log_5 2$, lo

(If this h
could ha

By simil
discrete

any
$n+b$ mod p.
s for n
sions.

ains
equations

000003:
ctor
$^1/2^6 5^6$ in $\mathbf{Q}$
mod p)
$_5\, 3 \equiv$
(mod $p-1$).

Can completely factor $62/(p+62)$ as $2^1 31^1 / 3^1 5^1 11^2 19^1 29^1$ so $\log_5 2 + \log_5 31 \equiv \log_5 3 + \log_5 5 + 2\log_5 11 + \log_5 19 + \log_5 29 \pmod{p-1}$.

Try to completely factor $1/(p+1)$, $2/(p+2)$, etc. Find factorization of $a/(p+a)$ as product of powers of -1, $2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31$ for each of the following a's: -5100, -4675, -3128, -403, -368, -147, -3, 62, 957, 2912, 3857, 6877.

Each complete fac
produces a log equ

Now have 12 linea
for $\log_5 2, \log_5 3, .$
Free equations: lo
$\log_5(-1) = (p-1$

By linear algebra c
$\log_5 2, \log_5 3, \ldots,$ l

(If this hadn't bee
could have searche

By similar techniq
discrete log of any

p.

Q

$- 1)$.

Can completely factor $62/(p+62)$ as $2^1 31^1 / 3^1 5^1 11^2 19^1 29^1$ so $\log_5 2 + \log_5 31 \equiv$ $\log_5 3 + \log_5 5 + 2\log_5 11 +$ $\log_5 19 + \log_5 29 \pmod{p-1}$.

Try to completely factor $1/(p+1)$, $2/(p+2)$, etc. Find factorization of $a/(p+a)$ as product of powers of -1, $2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31$ for each of the following a's: -5100, -4675, -3128, -403, -368, -147, -3, 62, 957, 2912, 3857, 6877.

Each complete factorization produces a log equation.

Now have 12 linear equation for $\log_5 2, \log_5 3, \ldots, \log_5 31$ Free equations: $\log_5 5 = 1$, $\log_5(-1) = (p-1)/2$.

By linear algebra compute $\log_5 2, \log_5 3, \ldots, \log_5 31$.

(If this hadn't been enough, could have searched more a'

By similar technique obtain discrete log of any target.

Can completely factor $62/(p+62)$ as $2^1 31^1/3^1 5^1 11^2 19^1 29^1$ so $\log_5 2 + \log_5 31 \equiv \log_5 3 + \log_5 5 + 2\log_5 11 + \log_5 19 + \log_5 29 \pmod{p-1}$.

Try to completely factor $1/(p+1)$, $2/(p+2)$, etc. Find factorization of $a/(p+a)$ as product of powers of -1, $2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31$ for each of the following a's: -5100, -4675, -3128, -403, -368, -147, -3, 62, 957, 2912, 3857, 6877.

Each complete factorization produces a log equation.

Now have 12 linear equations for $\log_5 2, \log_5 3, \ldots, \log_5 31$. Free equations: $\log_5 5 = 1$, $\log_5(-1) = (p-1)/2$.

By linear algebra compute $\log_5 2, \log_5 3, \ldots, \log_5 31$.

(If this hadn't been enough, could have searched more a's.)

By similar technique obtain discrete log of any target.

mpletely factor $62/(p+62)$
$/3^1 5^1 11^2 19^1 29^1$
$2 + \log_5 31 \equiv$
$\log_5 5 + 2\log_5 11 +$
$+ \log_5 29 \pmod{p-1}$.

ompletely factor
$1)$, $2/(p+2)$, etc.
torization of $a/(p+a)$
uct of powers of -1,
$, 11, 13, 17, 19, 23, 29, 31$
of the following a's:
-4675, -3128,
-368, -147, -3,
2912, 3857, 6877.

Each complete factorization produces a log equation.

Now have 12 linear equations for $\log_5 2, \log_5 3, \ldots, \log_5 31$. Free equations: $\log_5 5 = 1$, $\log_5(-1) = (p-1)/2$.

By linear algebra compute $\log_5 2, \log_5 3, \ldots, \log_5 31$.

(If this hadn't been enough, could have searched more a's.)

By similar technique obtain discrete log of any target.

For $p \to$
scales su
cost p^ϵ v

Compare

Specifica
$a \in \{1, 2$
$\lg y \in O$
finds y c
into prin
and com

(Assumi
Have ext

ctor $62/(p+62)$
$19^1 29^1$
$\equiv$
$2\log_5 11 +$
$\pmod{p-1}$.

factor
2), etc.
of $a/(p+a)$
ers of -1,
$, 19, 23, 29, 31$
lowing a's:
3128,
7, -3,
57, 6877.

Each complete factorization produces a log equation.

Now have 12 linear equations for $\log_5 2, \log_5 3, \ldots, \log_5 31$.
Free equations: $\log_5 5 = 1$, $\log_5(-1) = (p-1)/2$.

By linear algebra compute $\log_5 2, \log_5 3, \ldots, \log_5 31$.

(If this hadn't been enough, could have searched more a's.)

By similar technique obtain discrete log of any target.

For $p \to \infty$, index
scales surprisingly
cost p^ϵ where $\epsilon \to$

Compare to rho:

Specifically: search
$a \in \{1, 2, \ldots, y^2\}$
$\lg y \in O(\sqrt{\lg p \lg \lg}$
finds y complete f
into primes $\le y$,
and computes disc

(Assuming standar
Have extensive evi

$+ 62)$

$1)$.

$a)$

$9, 31$

:

Each complete factorization produces a log equation.

Now have 12 linear equations for $\log_5 2, \log_5 3, \ldots, \log_5 31$. Free equations: $\log_5 5 = 1$, $\log_5(-1) = (p-1)/2$.

By linear algebra compute $\log_5 2, \log_5 3, \ldots, \log_5 31$.

(If this hadn't been enough, could have searched more a's.)

By similar technique obtain discrete log of any target.

For $p \to \infty$, index calculus scales surprisingly well: cost p^ϵ where $\epsilon \to 0$.

Compare to rho: $\approx p^{1/2}$.

Specifically: searching $a \in \{1, 2, \ldots, y^2\}$, with $\lg y \in O(\sqrt{\lg p \lg \lg p})$, finds y complete factorizatio into primes $\leq y$, and computes discrete logs.

(Assuming standard conjectu Have extensive evidence.)

Each complete factorization produces a log equation.

Now have 12 linear equations for $\log_5 2, \log_5 3, \ldots, \log_5 31$. Free equations: $\log_5 5 = 1$, $\log_5(-1) = (p-1)/2$.

By linear algebra compute $\log_5 2, \log_5 3, \ldots, \log_5 31$.

(If this hadn't been enough, could have searched more a's.)

By similar technique obtain discrete log of any target.

For $p \to \infty$, index calculus scales surprisingly well: cost p^ϵ where $\epsilon \to 0$.

Compare to rho: $\approx p^{1/2}$.

Specifically: searching $a \in \{1, 2, \ldots, y^2\}$, with $\lg y \in O(\sqrt{\lg p \lg \lg p})$, finds y complete factorizations into primes $\leq y$, and computes discrete logs.

(Assuming standard conjectures. Have extensive evidence.)

mplete factorization
s a log equation.

ve 12 linear equations
$2, \log_5 3, \ldots, \log_5 31$.
uations: $\log_5 5 = 1$,
$) = (p-1)/2$.

r algebra compute
$\mathrm{og}_5 3, \ldots, \log_5 31$.

hadn't been enough,
ve searched more a's.)

ar technique obtain
log of any target.

For $p \to \infty$, index calculus
scales surprisingly well:
cost p^ϵ where $\epsilon \to 0$.

Compare to rho: $\approx p^{1/2}$.

Specifically: searching
$a \in \{1, 2, \ldots, y^2\}$, with
$\lg y \in O(\sqrt{\lg p \lg \lg p})$,
finds y complete factorizations
into primes $\leq y$,
and computes discrete logs.

(Assuming standard conjectures.
Have extensive evidence.)

Latest in
use the
and the

To comp
$\lg \mathrm{cost} \in$
$O((\lg q)$

For secu
$q \approx 2^{256}$
$q \approx 2^{204}$

We don'
index-ca
. . . exce

ctorization
uation.

r equations
$\ldots, \log_5 31$.
$g_5 5 = 1$,
$.)/2$.

compute
$og_5 31$.

n enough,
ed more a's.)

ue obtain
target.

For $p \to \infty$, index calculus
scales surprisingly well:
cost p^ϵ where $\epsilon \to 0$.

Compare to rho: $\approx p^{1/2}$.

Specifically: searching
$a \in \{1, 2, \ldots, y^2\}$, with
$\lg y \in O(\sqrt{\lg p \lg \lg p})$,
finds y complete factorizations
into primes $\le y$,
and computes discrete logs.

(Assuming standard conjectures.
Have extensive evidence.)

Latest index-calcu
use the "number-f
and the "function-

To compute discre
$\lg \text{cost} \in$
$O((\lg q)^{1/3}(\lg \lg q)$

For security:
$q \approx 2^{256}$ to stop r
$q \approx 2^{2048}$ to stop

We don't know an
index-calculus met
. . . except for som

ns

.

’s.)

For $p \to \infty$, index calculus scales surprisingly well: cost p^ϵ where $\epsilon \to 0$.

Compare to rho: $\approx p^{1/2}$.

Specifically: searching $a \in \{1, 2, \ldots, y^2\}$, with $\lg y \in O(\sqrt{\lg p \lg \lg p})$, finds y complete factorizations into primes $\leq y$, and computes discrete logs.

(Assuming standard conjectures. Have extensive evidence.)

Latest index-calculus variant use the “number-field sieve” and the “function-field sieve

To compute discrete logs in $\lg \text{cost} \in$ $O((\lg q)^{1/3}(\lg \lg q)^{2/3})$.

For security:
$q \approx 2^{256}$ to stop rho;
$q \approx 2^{2048}$ to stop NFS.

We don’t know any index-calculus methods for E
. . . except for some curves.

For $p \to \infty$, index calculus scales surprisingly well: cost p^ϵ where $\epsilon \to 0$.

Compare to rho: $\approx p^{1/2}$.

Specifically: searching $a \in \{1, 2, \ldots, y^2\}$, with $\lg y \in O(\sqrt{\lg p \lg \lg p})$, finds y complete factorizations into primes $\leq y$, and computes discrete logs.

(Assuming standard conjectures. Have extensive evidence.)

Latest index-calculus variants use the "number-field sieve" and the "function-field sieve."

To compute discrete logs in $\mathbf{F}_q$: $\lg \text{cost} \in O((\lg q)^{1/3} (\lg \lg q)^{2/3})$.

For security:
$q \approx 2^{256}$ to stop rho;
$q \approx 2^{2048}$ to stop NFS.

We don't know any index-calculus methods for ECDL!
. . . except for some curves.