Non-uniform cracks in the concrete: the power of free precomputation

D. J. Bernstein
University of Illinois at Chicago &
Technische Universiteit Eindhoven

Tanja Lange
Technische Universiteit Eindhoven

Full 53-page paper,
including progress towards
formalizing collision resistance:
eprint.iacr.org/2012/318

Concrete security: an example

What is the best NIST P-256 discrete-log attack algorithm?

ECDL input: P-256 points P, Q, where P is a standard generator.

ECDL output: $\log_P Q$.

Standard definition of "best": minimize "time".

Concrete security: an example

What is the best NIST P-256 discrete-log attack algorithm?

ECDL input: P-256 points P, Q, where P is a standard generator.

ECDL output: $\log_P Q$.

Standard definition of "best": minimize "time".

More generally, allow attacks with $<$100% success probability; analyze tradeoffs between "time" and success probability.
This talk focuses on high prob.

P-256 discrete-log attack $\Rightarrow$
total TLS-ECDHE-P-256 break!
Should TLS users worry?

P-256 discrete-log attack $\Rightarrow$
total TLS-ECDHE-P-256 break!
Should TLS users worry?

No. Many researchers have
tried and failed to find good
P-256 discrete-log attacks.

P-256 discrete-log attack $\Rightarrow$
total TLS-ECDHE-P-256 break!
Should TLS users worry?

No. Many researchers have tried and failed to find good P-256 discrete-log attacks.

Standard conjecture:
For each $p \in [0, 1]$,
each P-256 ECDL algorithm with success probability $\geq p$ takes "time" $\geq 2^{128} p^{1/2}$.

Similar conjectures for AES-128, RSA-3072, etc.: see, e.g., 2005 Bellare–Rogaway.

Concrete reductions

Another conjecture:
Each TLS-ECDHE-P-256 attack with success probability $\geq p$ takes "time" $\geq 2^{128} p^{1/2}$.

Concrete reductions

Another conjecture:
Each TLS-ECDHE-P-256 attack with success probability $\geq p$ takes "time" $\geq 2^{128} p^{1/2}$.

Why should users have any confidence in this conjecture?

How many researchers have really tried to break ECDHE-P-256? ECDSA-P-256? ECIES-P-256? ECMQV-P-256? Other P-256-based protocols? Far less attention than for ECDL.

Provable security to the rescue!

Prove: if there is
a TLS-ECDHE-P-256 attack
then there is
a P-256 discrete-log attack
with similar “time”
and success probability.

Provable security to the rescue!

Prove: if there is
a TLS-ECDHE-P-256 attack
then there is
a P-256 discrete-log attack
with similar "time"
and success probability.

Oops: This turns out to be hard.
But changing DL to DDH
+ adding more assumptions
allows a proof: Crypto 2012
Jager–Kohlar–Schäge–Schwenk
"On the security of TLS-DHE
in the standard model".

Similar pattern throughout the "provable security" literature.

Protocol designers (try to) prove that hardness of a problem P (e.g., P-256 DDH) implies security of various protocols Q.

After extensive cryptanalysis of P, maybe gain confidence in hardness of P, and hence in security of Q.

Similar pattern throughout the "provable security" literature.

Protocol designers (try to) prove that hardness of a problem P (e.g., P-256 DDH) implies security of various protocols Q.

After extensive cryptanalysis of P, maybe gain confidence in hardness of P, and hence in security of Q.

Why not directly cryptanalyze Q? Cryptanalysis is hard work: have to focus on *a few* problems P. Proofs scale to *many* protocols Q.

Interlude regarding "time"

How much "time" does the following algorithm take?

```
def pidigit(n0,n1,n2):
  if n0 == 0:
    if n1 == 0:
      if n2 == 0: return 3
      return            1
    if n2 == 0: return   4
    return               1
  if n1 == 0:
    if n2 == 0: return   5
    return               9
  if n2 == 0: return     2
  return                 6
```

Students in algorithm courses
learn to count executed “steps”.
Skipped branches take 0 “steps”.

This algorithm uses 4 “steps”.

Students in algorithm courses
learn to count executed “steps”.
Skipped branches take 0 “steps”.

This algorithm uses 4 “steps”.

Generalization: There exists an
algorithm that, given $n < 2^k$,
prints the nth digit of π
using $k + 1$ “steps”.

Students in algorithm courses learn to count executed “steps”. Skipped branches take 0 “steps”.

This algorithm uses 4 “steps”.

Generalization: There exists an algorithm that, given $n < 2^k$, prints the nth digit of π using $k + 1$ “steps”.

Variant: There exists a 258-“step” P-256 discrete-log attack (with 100% success probability).

Students in algorithm courses learn to count executed "steps". Skipped branches take 0 "steps".

This algorithm uses 4 "steps".

Generalization: There exists an algorithm that, given $n < 2^k$, prints the nth digit of π using $k+1$ "steps".

Variant: There exists a 258-"step" P-256 discrete-log attack (with 100% success probability). If "time" means "steps" then the standard conjectures are wrong.

1994 Bellare–Kilian–Rogaway:

"*We say that*
A is a (t, q)*-adversary if*
A runs in at most t *steps and*
makes at most q *queries to* $\mathcal{O}$."

1994 Bellare–Kilian–Rogaway:
"*We say that*
A is a (t, q)*-adversary if*
A runs in at most t *steps and*
makes at most q *queries to* $\mathcal{O}$."

Oops: table-lookup attack
has very small t.

Paper conjectured "useful" DES
security bounds. Any reasonable
interpretation of conjecture was
false, given paper's definition.
Theorems in paper were vacuous.

2000 Bellare–Kilian–Rogaway: "*We fix some particular Random Access Machine (RAM) as a model of computation. . . . A's running time [means] A's actual execution time plus the length of A's description . . . This convention eliminates pathologies caused [by] arbitrarily large lookup tables . . .*"

2000 Bellare–Kilian–Rogaway:
“*We fix some particular Random Access Machine (RAM) as a model of computation. . . . A's running time [means] A's actual execution time plus the length of A's description . . . This convention eliminates pathologies caused [by] arbitrarily large lookup tables . . .*”

Main point of our paper:
There are more pathologies!

Illustrative example: ECDL.

The rho method

Simplified, non-parallel rho:

Make a pseudo-random walk $R_0, R_1, R_2, \ldots$ in the group $\langle P \rangle$, where current point determines the next point: $R_{i+1} = f(R_i)$.

Birthday paradox:
Randomly choosing from ℓ elements picks one element twice after about $\sqrt{\pi\ell/2}$ draws.
P-256: $\ell \approx 2^{256}$ so $\approx 2^{128}$ draws.

The walk now enters a cycle.
Cycle-finding algorithm (e.g., Floyd) quickly detects this.

Goal: Compute $\log_P Q$.

Assume that for each i
we know $x_i, y_i \in \mathbf{Z}/\ell\mathbf{Z}$
so that $R_i = y_i P + x_i Q$.

Then $R_i = R_j$ means that
$y_i P + x_i Q = y_j P + x_j Q$
so $(y_i - y_j)P = (x_j - x_i)Q$.
If $x_i \neq x_j$ the DLP is solved:
$\log_P Q = (y_j - y_i)/(x_i - x_j)$.

Goal: Compute $\log_P Q$.

Assume that for each i
we know $x_i, y_i \in \mathbf{Z}/\ell\mathbf{Z}$
so that $R_i = y_i P + x_i Q$.

Then $R_i = R_j$ means that
$y_i P + x_i Q = y_j P + x_j Q$
so $(y_i - y_j)P = (x_j - x_i)Q$.
If $x_i \neq x_j$ the DLP is solved:
$\log_P Q = (y_j - y_i)/(x_i - x_j)$.

e.g. "base-(P, Q) r-adding walk":
precompute $S_1, S_2, \ldots, S_r$
as random combinations $aP + bQ$;
define $f(R) = R + S_{H(R)}$
where H hashes to $\{1, 2, \ldots, r\}$.

Parallel rho

1994 van Oorschot–Wiener:

Declare some subset of $\langle P \rangle$ to be the set of *distinguished points*: e.g., all $R \in \langle P \rangle$ where last 20 bits of representation of R are 0.

Perform, in parallel, walks for different starting points $Q + yP$ but same update function f.

Terminate each walk once it hits a distinguished point. Report point to central server. Server receives, stores, and sorts all distinguished points.

State of the art

Can break DLP in group of order ℓ in $\sqrt{\pi\ell/2}$ group operations.

Use negation map to gain factor $\sqrt{2}$ for elliptic curves.

Solving DLP on NIST P-256 takes $\approx 2^{128}$ group operations.

This is the best algorithm that *cryptanalysts have published*.

State of the art

Can break DLP in group of order ℓ in $\sqrt{\pi\ell/2}$ group operations.

Use negation map to gain factor $\sqrt{2}$ for elliptic curves.

Solving DLP on NIST P-256 takes $\approx 2^{128}$ group operations.

This is the best algorithm that *cryptanalysts have published*.

But is it the best algorithm that *exists*?

This paper's ECDL algorithms

Assuming plausible heuristics, overwhelmingly verified by computer experiment:

There exists a P-256 ECDL algorithm that takes "time" $\approx 2^{85}$ and has success probability ≈ 1.

"Time" includes algorithm length.

Inescapable conclusion: **The standard conjectures** (regarding P-256 ECDL hardness, P-256 ECDHE security, etc.) **are false.**

Should P-256 ECDHE users be worried about this P-256 ECDL algorithm A?

No!

We have a program B that prints out A, but B takes "time" $\approx 2^{170}$.

We conjecture that nobody will ever print out A.

Should P-256 ECDHE users be worried about this P-256 ECDL algorithm A?

No!

We have a program B that prints out A, but B takes “time” $\approx 2^{170}$.

We conjecture that nobody will ever print out A.

But A *exists*, and the standard conjecture doesn’t see the 2^{170}.

Cryptanalysts *do* see the 2^{170}.

Common parlance: We have a 2^{170} "precomputation" (independent of Q) followed by a 2^{85} "main computation".

For cryptanalysts: This costs 2^{170}, much worse than 2^{128}.

For the standard security definitions and conjectures: The main computation costs 2^{85}, much better than 2^{128}.

Almost standard walk function: redefine steps S_i to depend on P only; i.e., $S_i = c_i P$ with c_i chosen uniformly at random.

Almost standard walk function:
redefine steps S_i to depend
on P only; i.e., $S_i = c_i P$ with
c_i chosen uniformly at random.

Precomputation:
Start some walks at yP
for random choices of y.
Build table of distinct
distinguished points D
along with $\log_P D$.

Almost standard walk function: redefine steps S_i to depend on P only; i.e., $S_i = c_i P$ with c_i chosen uniformly at random.

Precomputation:
Start some walks at yP for random choices of y.
Build table of distinct distinguished points D along with $\log_P D$.

Main computation:
Starting from Q, walk to distinguished point $Q + yP$.
Check for $Q + yP$ in table.

Almost standard walk function: redefine steps S_i to depend on P only; i.e., $S_i = c_i P$ with c_i chosen uniformly at random.

Precomputation:
Start some walks at yP for random choices of y.
Build table of distinct distinguished points D along with $\log_P D$.

Main computation:
Starting from Q, walk to distinguished point $Q + yP$.
Check for $Q + yP$ in table.
(If this fails, rerandomize Q.)

What you find in the full paper:

P-256 isn't the only problem!
There *exist* algorithms breaking
AES-128, RSA-3072, DSA-3072
at cost below 2^{128};
e.g., time 2^{85} to break AES.
(Assuming standard heuristics.)

$\Rightarrow$ Very large separation
between standard definition
and actual security.

Also: Analysis of various ideas
for fixing the definitions.

`eprint.iacr.org/2012/318`