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

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

Tanja Lange
Technische Universiteit Eindhoven

---

eprint.iacr.org/2012/318,
eprint.iacr.org/2012/458

2012.02.19 Koblitz–Menezes "Another look at HMAC":

"*. . . Third, we describe a fundamental flaw in Bellare's 2006 security proof for HMAC, and show that with the flaw removed the proof gives a security guarantee that is of little value in practice.*"

2012.03.02: "*Bellare contacted us and told us that he strongly objected to our language— especially the word 'flaw'—. . .*"

Yehuda Lindell: "*This time they really outdid themselves since there is actually no error. Rather the proof of security is in the non-uniform model, which they appear to not be familiar with. . . . There is NO FLAW here whatsoever.*"

Jonathan Katz: "*Many researchers are justifiably concerned about the fact that Alfred Menezes will be giving an invited talk at Eurocrypt 2012 related to his line of papers criticizing provable security. I share this concern.*"

Bellare to Koblitz (according to 2012.10 Koblitz talk): "*It never occurred to me that a reader would not understand that when complexity is concrete, we have non-uniformity. . . . If you want . . . to gain respect among theoretical cryptographers, it would benefit from reflecting our feedback and being better informed about the basics of the field. . . . Uniform and non-uniform complexity are typically taught in a graduate course in computational complexity theory.*"

2012.03.17 Koblitz–Menezes:

"... *Third, we describe a fundamental defect from a practice-oriented standpoint in Bellare's 2006 security result for HMAC, and show that with this defect removed his proof gives a security guarantee that is of little value in practice.*"

2012.03.17 Koblitz–Menezes:
"... *Third, we describe a fundamental defect from a practice-oriented standpoint in Bellare's 2006 security result for HMAC, and show that with this defect removed his proof gives a security guarantee that is of little value in practice.*"

2012.04: Menezes gives Eurocrypt invited talk "Another look at provable security" $\Rightarrow$ $>$20 solid seconds of applause.

2012.03.17 Koblitz–Menezes: "*. . . Third, we describe a fundamental defect from a practice-oriented standpoint in Bellare's 2006 security result for HMAC, and show that with this defect removed his proof gives a security guarantee that is of little value in practice.*"

2012.04: Menezes gives Eurocrypt invited talk "Another look at provable security" $\Rightarrow$ $>$20 solid seconds of applause.

`youtube?v=l56ORg5xXkk`

Understanding the dispute

What is the best chosen-plaintext AES-128 key-recovery attack?

Attack input: a black box that contains a secret key k and computes $p \mapsto \text{AES}_k(p)$.

Attack output: k.

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

Understanding the dispute

What is the best chosen-plaintext AES-128 key-recovery attack?

Attack input: a black box that contains a secret key k and computes $p \mapsto \text{AES}_k(p)$.

Attack output: k.

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

More generally, allow attacks with <100% success probability; analyze tradeoffs between "time" and success probability.

Maybe a key-recovery attack could be turned into an AES-CBC-MAC forgery attack! Should AES-CBC-MAC users be worried about this?

Maybe a key-recovery attack could be turned into an AES-CBC-MAC forgery attack! Should AES-CBC-MAC users be worried about this?

No. Many researchers have tried and failed to find good AES key-recovery attacks.

Maybe a key-recovery attack could be turned into an AES-CBC-MAC forgery attack! Should AES-CBC-MAC users be worried about this?

No. Many researchers have tried and failed to find good AES key-recovery attacks.

Standard conjecture:
For each $p \in [0, 1]$,
each AES key-recovery attack with success probability $\geq p$ takes "time" $\geq 2^{128} p$.

See, e.g., 2005 Bellare–Rogaway.

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 256-"step" AES key-recovery 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 256-"step" AES key-recovery attack (with 100% success probability). If "time" means "steps" then the standard conjecture is wrong.

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 . . . Alternatively, the reader can think of circuits over some fixed basis of gates, like 2-input NAND gates . . . now time simply means the circuit size.*”

Side comments:

1. Definition from Crypto 1994 Bellare–Kilian–Rogaway was flawed: failed to add length.

Paper conjectured “useful” DES security bounds; any reasonable interpretation of conjecture was false, given paper’s definition.

Side comments:

1. Definition from Crypto 1994 Bellare–Kilian–Rogaway was flawed: failed to add length.

Paper conjectured "useful" DES security bounds; any reasonable interpretation of conjecture was false, given paper's definition.

2. Many more subtle issues defining RAM "time": see 1990 van Emde Boas survey.

Side comments:

1. Definition from Crypto 1994 Bellare–Kilian–Rogaway was flawed: failed to add length.

Paper conjectured “useful” DES security bounds; any reasonable interpretation of conjecture was false, given paper’s definition.

2. Many more subtle issues defining RAM “time”: see 1990 van Emde Boas survey.

3. NAND definition is easier but breaks many theorems.

Reductions

Another standard conjecture:
Each AES-CBC-MAC q-block forgery attack with success probability $\geq p + q(q-1)/2^{129}$ takes "time" $> 2^{128} p$.

Reductions

Another standard conjecture: Each AES-CBC-MAC q-block forgery attack with success probability $\geq p + q(q-1)/2^{129}$ takes "time" $>2^{128}p$.

Why should users have any confidence in this conjecture?

How many researchers have really tried to break AES-CBC-MAC? AES-CTR? AES-GCM? Other AES-based protocols? Far less attention than for key recovery.

Provable security to the rescue!

Prove: if there is
an AES-CBC-MAC attack
then there is
an AES key-recovery attack
with similar “time”
and success probability.

Provable security to the rescue!

Prove: if there is
an AES-CBC-MAC attack
then there is
an AES key-recovery attack
with similar “time”
and success probability.

Oops: This turns out to be hard.
But changing from key-recovery
attack to PRF distinguishing
attack allows a proof:
1994 Bellare–Kilian–Rogaway.

Similar pattern throughout the “provable security” literature.

Protocol designers (try to) prove that hardness of a problem P (e.g., AES PRF attacks) 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., AES PRF attacks) 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.

The big oops

These conjectures are wrong.

Example: There exists
a fast AES PRF attack
with success probability $\geq 2^{-64}$.

The big oops

These conjectures are wrong.

Example: There exists
a fast AES PRF attack
with success probability $\geq 2^{-64}$.

Good candidate for attack:
$\text{MD5}_0(7, \text{AES}_k(0), \text{AES}_k(1)) = 1$
with probability $\geq 1/2 + 2^{-64}$;
$\text{MD5}_0(7, F(0), F(1)) = 1$
with probability $\leq 1/2$.
Here $\text{MD5}_0(x) = \text{bit}_0(\text{MD5}(x))$.

The big oops

These conjectures are wrong.

Example: There exists a fast AES PRF attack with success probability $\geq 2^{-64}$.

Good candidate for attack: $\text{MD5}_0(7, \text{AES}_k(0), \text{AES}_k(1)) = 1$ with probability $\geq 1/2 + 2^{-64}$; $\text{MD5}_0(7, F(0), F(1)) = 1$ with probability $\leq 1/2$. Here $\text{MD5}_0(x) = \text{bit}_0(\text{MD5}(x))$.

If this candidate doesn't work, replace 7 with 8 or 9 or . . . .

“*We only meant the conjectures for* $p \geq 2^{-40}$*, you nitpicker.*”

“*We only meant the conjectures for* $p \geq 2^{-40}$*, you nitpicker.*”

The conjectures are still wrong!
Example: There exists an AES key-recovery attack with success probability ≈ 1 taking “time” $\approx 2^{86}$.

“*We only meant the conjectures for $p \geq 2^{-40}$, you nitpicker.*”

The conjectures are still wrong! Example: There exists an AES key-recovery attack with success probability ≈ 1 taking “time” $\approx 2^{86}$.

The attack algorithm: iterate $k \mapsto \text{AES}_k(0) \oplus 7$ 2^{43} times, look up in a size-2^{43} Hellman table; iterate $k \mapsto \text{AES}_k(0) \oplus 8$ 2^{43} times, look up in a size-2^{43} Hellman table; etc.

How about NIST P-256?

ECDL input: points P, Q,
where P is a standard generator.
ECDL output: $\log_P Q$.

How about NIST P-256?

ECDL input: points P, Q,
where P is a standard generator.
ECDL output: $\log_P Q$.

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

Cube-root 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 ECDSA security, etc.) **are false.**

Should P-256 ECDSA 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 ECDSA 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}.

What the algorithm does

What the algorithm does

1999 Escott–Sager–Selkirk–Tsapakidis, also crediting Silverman–Stapleton:

Computing (e.g.) $\log_P Q_1$, $\log_P Q_2$, $\log_P Q_3$, $\log_P Q_4$, and $\log_P Q_5$ costs only $2.49\times$ more than computing $\log_P Q$.

The basic idea:
compute $\log_P Q_1$ with rho;
compute $\log_P Q_2$ with rho, *reusing* distinguished points produced by Q_1; etc.

2001 Kuhn–Struik analysis:
cost $\Theta(n^{1/2}\ell^{1/2})$
for n discrete logarithms
in group of order ℓ
if $n \ll \ell^{1/4}$.

2001 Kuhn–Struik analysis:
cost $\Theta(n^{1/2}\ell^{1/2})$
for n discrete logarithms
in group of order ℓ
if $n \ll \ell^{1/4}$.

2004 Hitchcock–
Montague–Carter–Dawson:
View computations of
$\log_P Q_1, \ldots, \log_P Q_{n-1}$ as
precomputatation for main
computation of $\log_P Q_n$.
Analyze tradeoffs between
main-computation time and
precomputation time.

2012 Bernstein–Lange:

(1) Adapt to interval of length ℓ inside much larger group.

(2) Analyze tradeoffs between main-computation time and precomputed table size.

(3) Choose table entries more carefully to reduce main-computation time.

(4) Also choose iteration function more carefully.

(5) Reduce space required for each table entry.

(6) Break $\ell^{1/4}$ barrier.

Applications:

(7) Disprove the standard 2^{128} P-256 security conjectures.

(8) Accelerate trapdoor DL etc.

(9) Accelerate BGN etc.; this needs (1).

Bonus:

(10) Disprove the standard 2^{128} AES, DSA-3072, RSA-3072 security conjectures.

Credit to earlier Lee–Cheon–Hong paper for (2), (6), (8).

Standard walk function:
choose uniform random
$c_1, \ldots, c_r \in \{1, 2, \ldots, \ell - 1\}$;
walk from R to $R + c_{H(R)} P$.

Standard walk function:
choose uniform random
$c_1, \ldots, c_r \in \{1, 2, \ldots, \ell - 1\}$;
walk from R to $R + c_{H(R)}P$.

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

Standard walk function:
choose uniform random
$c_1, \ldots, c_r \in \{1, 2, \ldots, \ell - 1\}$;
walk from R to $R + c_{H(R)}P$.

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.

Standard walk function:
choose uniform random
$c_1, \ldots, c_r \in \{1, 2, \ldots, \ell - 1\}$;
walk from R to $R + c_{H(R)} P$.

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.)

DSA-3072

Assume that DLP subgroup
is extended to 384 bits
to counter previous attack

DSA-3072

Assume that DLP subgroup
is extended to 384 bits
to counter previous attack
(and assume field $\mathbf{F}_p$ to avoid
Antoine coming after you).

DSA-3072

Assume that DLP subgroup
is extended to 384 bits
to counter previous attack
(and assume field $\mathbf{F}_p$ to avoid
Antoine coming after you).

The following sketch
is not the state of the art —
but good enough to break
the 2^{128} assumption.

Let $g \in \mathbf{F}_p^*$ have order q, $h = g^k$.
Goal: Find k.

Precomputation:
Take $y = 2^{110}$,
compute $\log_g x^{(p-1)/q}$
for every prime number $x \le y$.

Precomputation:
Take $y = 2^{110}$,
compute $\log_g x^{(p-1)/q}$
for every prime number $x \le y$.

Main computation:
Try to write h as
quotient h_1/h_2 in $\mathbf{F}_p^*$
with $h_2 \in \{1, 2, 3, \ldots, 2^{1535}\}$,
$h_1 \in \{-2^{1535}, \ldots, 0, 1, \ldots, 2^{1535}\}$,
and $\gcd\{h_1, h_2\} = 1$;

Precomputation:
Take $y = 2^{110}$,
compute $\log_g x^{(p-1)/q}$
for every prime number $x \le y$.

Main computation:
Try to write h as
quotient h_1/h_2 in $\mathbf{F}_p^*$
with $h_2 \in \{1, 2, 3, \ldots, 2^{1535}\}$,
$h_1 \in \{-2^{1535}, \ldots, 0, 1, \ldots, 2^{1535}\}$,
and $\gcd\{h_1, h_2\} = 1$;

and then try to factor h_1, h_2
into primes $\le y$.

Precomputation:
Take $y = 2^{110}$,
compute $\log_g x^{(p-1)/q}$
for every prime number $x \le y$.

Main computation:
Try to write h as
quotient h_1/h_2 in $\mathbf{F}_p^*$
with $h_2 \in \{1, 2, 3, \ldots, 2^{1535}\}$,
$h_1 \in \{-2^{1535}, \ldots, 0, 1, \ldots, 2^{1535}\}$,
and $\gcd\{h_1, h_2\} = 1$;

and then try to factor h_1, h_2
into primes $\le y$.

If this fails, try again
with hg, hg^2, etc.

Analysis

About $y/\log y \approx 2^{103.75}$ primes $\leq y$
for a total of $2^{109.33}$ bytes
to store all small DLs.

Can write h as h_1/h_2 with
probability $\approx (6/\pi^2) 2^{3071}/p$.

h_i is y-smooth with probability
very close to $u^{-u} \approx 2^{-53.06}$
where $u = 1535/110$.

Overall the attack requires
between $2^{107.85}$ and $2^{108.85}$
iterations; batch smoothness
detection is fast.

Possible responses

Possible responses

(1) Accept 2^{85} etc. as security;
live with it. Protect the proofs!

Possible responses

(1) Accept 2^{85} etc. as security;
live with it. Protect the proofs!

(2) Switch to NAND metric; or
(3) switch to AT metric.
Breaks most theorems;
still bogus results in NAND.

Possible responses

(1) Accept 2^{85} etc. as security; live with it. Protect the proofs!

(2) Switch to NAND metric; or
(3) switch to AT metric.
Breaks most theorems;
still bogus results in NAND.

(4) Add effectivity. Include cost for finding the algorithm.

Possible responses

(1) Accept 2^{85} etc. as security;
live with it. Protect the proofs!

(2) Switch to NAND metric; or
(3) switch to AT metric.
Breaks most theorems;
still bogus results in NAND.

(4) Add effectivity. Include
cost for finding the algorithm.

(5) Add uniformity.
Clearly stops attacks
but breaks most theorems.
Abandons goal of defining concrete security of AES etc.