Modeling the security of cryptography, part 1: secret-key cryptography

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

Joint work with:
Tanja Lange
Technische Universiteit Eindhoven

---

eprint.iacr.org/2012/318
"Non-uniform cracks in the concrete: the power of free precomputation"

Cryptographic news

Frequent news stories
about cryptographic failures.

Usually these stories are
press releases from researchers:
e.g., TLS disaster announced
2013.02.04 by Alfardan–Paterson.

Occasionally these stories are
reporting real-world attacks:
e.g., 2012.05 announcement
of Flame invading computers
by forging code signatures
by exploiting MD5 weaknesses.

Provably secure cryptography

Attacker *cannot* break
the one-time pad.
Easy proof that ciphertext reveals
nothing about the plaintext.
Seeing ciphertext does not
improve attacker's chance
of guessing plaintext.

Provably secure cryptography

Attacker *cannot* break
the one-time pad.
Easy proof that ciphertext reveals
nothing about the plaintext.
Seeing ciphertext does not
improve attacker's chance
of guessing plaintext.

Attacker *cannot* break 1974
Gilbert–MacWilliams–Sloane
message-authentication code.
Easy proof that attacker's forgery
succeeds with chance $\leq \epsilon$,
where ϵ is chosen by user.

Real-world cryptography

AES is much more popular than the one-time pad.

Real-world cryptography

AES is much more popular
than the one-time pad.

Key length for one-time pad
is total message length.
OK if sender and receiver
met and exchanged USB sticks.

Real-world cryptography

AES is much more popular than the one-time pad.

Key length for one-time pad is total message length. OK if sender and receiver met and exchanged USB sticks.

Key length for AES: 128 bits. Many low-cost mechanisms to share 128-bit key through the Internet; see, e.g., ECDH in part 2.

Core use of AES ("AES-CTR"):
expand 128-bit key k
into huge string
$\text{AES}_k(0), \text{AES}_k(1), \ldots$
which *seems* to be
indistinguishable from uniform,
therefore safe as replacement
for key of one-time pad.

One-time pad encrypts;
AES expands.
Totally different features!

Theme pushed much further
in public-key crypto (part 2):
many cool new features.

The critical question

Can attacker break AES?

Definition of "break":
given random access to
string of 2^{135} bits,
decide whether string is
a uniform random string, or
$\mathrm{AES}_k(0), \mathrm{AES}_k(1), \ldots$
for a uniform random k.

The critical question

Can attacker break AES?

Definition of "break":
given random access to
string of 2^{135} bits,
decide whether string is
a uniform random string, or
$\text{AES}_k(0), \text{AES}_k(1), \ldots$
for a uniform random k.

If attacker has enough computer
power, can obviously break AES:
simply try all 2^{128} AES keys.

The critical question

Can attacker break AES?

Definition of "break":
given random access to
string of 2^{135} bits,
decide whether string is
a uniform random string, or
$\mathrm{AES}_k(0), \mathrm{AES}_k(1), \ldots$
for a uniform random k.

If attacker has enough computer
power, can obviously break AES:
simply try all 2^{128} AES keys.

Does attacker have this power?

Approximate power in watts:

2^{57}: Earth receives from the Sun.

Approximate power in watts:

2^{57}: Earth receives from the Sun.
2^{56}: Earth's surface.

Approximate power in watts:

2^{57}: Earth receives from the Sun.
2^{56}: Earth's surface.
2^{44}: World power usage.

Approximate power in watts:

2^{57}: Earth receives from the Sun.
2^{56}: Earth's surface.
2^{44}: World power usage.
2^{30}: PCs in a big botnet.

Approximate power in watts:

2^{57}: Earth receives from the Sun.

2^{56}: Earth's surface.

2^{44}: World power usage.

2^{30}: PCs in a big botnet.

2^{26}: One NSA data center.

Approximate power in watts:

2^{57}: Earth receives from the Sun.
2^{56}: Earth's surface.
2^{44}: World power usage.
2^{30}: PCs in a big botnet.
2^{26}: One NSA data center.

Today's state-of-the-art mass-market chips perform 2^{58} float ops/year/watt, roughly 2^{68} bit ops/year/watt.

Approximate power in watts:

2^{57}: Earth receives from the Sun.
2^{56}: Earth's surface.
2^{44}: World power usage.
2^{30}: PCs in a big botnet.
2^{26}: One NSA data center.

Today's state-of-the-art mass-market chips perform 2^{58} float ops/year/watt, roughly 2^{68} bit ops/year/watt.

Given such chips perfectly using all power received by Earth: 2^{125} bit ops/year.

Real attacker can't actually use all power received by Earth. Assume that attacker is limited to 1/1000 of Earth's surface; i.e., 2^{46} watts.

Maybe attacker will build much better chips. For short term seems safe to assume no qubit ops, and $\leq 1000\times$ better chips: $\leq 2^{78}$ bit ops/year/watt.

$\Rightarrow \leq 2^{124}$ bit ops/year.

Seems safe to declare larger computations to be intractable.

Checking an AES key guess
takes $>2^{13}$ bit ops
by best algorithm known.
$\Rightarrow$ $<2^{111}$ key guesses/year.

i.e.: chance $<2^{-17}$/year
of finding your key.

Checking an AES key guess takes $>2^{13}$ bit ops by best algorithm known.
$\Rightarrow$ $<2^{111}$ key guesses/year.

i.e.: chance $<2^{-17}$/year of finding your key.

But is the attacker using this algorithm?

Checking an AES key guess
takes $>2^{13}$ bit ops
by best algorithm known.
$\Rightarrow$ $<2^{111}$ key guesses/year.

i.e.: chance $<2^{-17}$/year
of finding your key.

But is the attacker
using this algorithm?

Maybe the attacker
has figured out an algorithm
that breaks AES using
much less computation.

How to address this risk?

Cryptanalysis to the rescue!

The cryptanalytic community studies AES, searching for better and better attacks.

By now dozens of experts have studied AES in public, and their attack algorithms seem to have converged.

$\Rightarrow$ Reasonable to *hope* that the attacker won't find a noticeably better algorithm.

Big scalability problem:
Many cryptographic systems
are of interest to users;
AES-CTR is just one example.

Example: AES-CBC-MAC
for 3-block messages. Use
$\text{AES}_k(\text{AES}_k(\text{AES}_k(x) + y) + z)$
to authenticate (x, y, z).

Is there any reason to think
that AES-CBC-MAC is secure?
Have the cryptanalysts
actually studied AES-CBC-MAC?

Security proofs to the rescue!

Can prove secure:
encryption+authentication
using a long key.

Security proofs to the rescue!

Can prove secure:
encryption+authentication
using a long key.

But **cannot** prove secure
by any known technique,
presumably by any technique:
AES-CTR; AES-CBC-MAC;
any other short-key system;
key exchange (e.g., ECDH);
public-key signatures;
public-key encryption;
fully homomorphic encryption;
most of modern cryptography.

Replacing cryptanalysis
with proofs: hopeless.

Replacing cryptanalysis
with proofs: hopeless.

But sometimes proofs can
save time for cryptanalysts
who are studying many systems.

Imagine the following theorem:
if AES-CTR is secure
then AES-CBC-MAC is secure.

This theorem can be useful
guidance for cryptanalysts
studying AES-CBC-MAC:
look for AES-CTR attack,
or attack outside security model,
or error in the proof.

To state such a theorem
need to define “secure”.

Early attempts at definitions
used purely asymptotic notions;
e.g., polynomial-time attacks
against families of cryptosystems.
Useless for formalizing security of
AES, RSA-1024, etc.

To state such a theorem
need to define “secure”.

Early attempts at definitions
used purely asymptotic notions;
e.g., polynomial-time attacks
against families of cryptosystems.
Useless for formalizing security of
AES, RSA-1024, etc.

1994 Bellare–Kilian–Rogaway:
concrete security definitions,
concrete CBC security theorem.
Many ($>$1000?) followup papers:
concrete theorems saying
X secure $\Rightarrow$ Y secure.

AES is "(t, q, ϵ)-secure"
$\Leftrightarrow$ every algorithm that takes
time $\le t$ and uses $\le q$ queries
has chance $\le \epsilon$
of PRP-breaking AES.

Alternate notation, same concept:
the "(t, q)-insecurity"
of AES is at most ϵ.

"PRP-breaking" AES means
distinguishing AES output
from output of a
uniform random permutation.
"PRF" variant: function
instead of permutation.

Attractive theorems. e.g.,

1994 Bellare–Kilian–Rogaway:

"$\textbf{Adv}^{\text{prf}}_{\text{CBC}^m\text{-}F}(q,t) \le$
$\textbf{Adv}^{\text{prp}}_{F}(q',t') + \dfrac{q^2m^2}{2^{l-1}}$
where $q' = mq$
and $t' = t + O(mql)$."

Attractive theorems. e.g.,

1994 Bellare–Kilian–Rogaway:

“$\mathbf{Adv}^{\text{prf}}_{\text{CBC}^m\text{-}F}(q,t) \le \mathbf{Adv}^{\text{prp}}_{F}(q',t') + \frac{q^2m^2}{2^{l-1}}$ where $q' = mq$ and $t' = t + O(mql)$.”

Conjectured bounds on security of specific ciphers that have survived cryptanalysis.

e.g., 2005 Bellare–Rogaway:

“$\mathbf{Adv}^{\text{prp}-\text{cpa}}_{\text{AES}}(\cdots) \le c_1 \cdot \frac{t/T_{\text{AES}}}{2^{128}} + c_2 \cdot \frac{q}{2^{128}}$.”

Completely standard in the
concrete-security literature
to formalize security
of a cryptosystem X
as the nonexistence of a
$\leq q$-query time-$\leq t$ algorithm
that breaks X
with success probability $>\epsilon$.

Many specific conjectures assert
(q, t, ϵ)-security of various X
where (q, t, ϵ) is chosen
to match the apparent limit
of extensive cryptanalysis.

Cracks in the concrete

2012 Bernstein–Lange:
Essentially all of these conjectures are wrong.

Assuming standard heuristics, there exist high-probability attacks taking time significantly below 2^{128} on AES, NIST P-256, DSA-3072, RSA-3072, etc.

All of these were conjectured to have security level $\geq 2^{128}$.

Should users worry? No!

Still plausible to conjecture that *attacker* is unable to break any of these systems, even with the massive computer power described earlier.

Should users worry? No!

Still plausible to conjecture that *attacker* is unable to break any of these systems, even with the massive computer power described earlier.

The standard formalizations fail to capture this.
The problem is that they are inaccurate models of intractability.

Should users worry? No!

Still plausible to conjecture that
attacker is unable to break
any of these systems,
even with the massive computer
power described earlier.

The standard formalizations
fail to capture this.
The problem is that they are
inaccurate models of intractability.

Our paper analyzes several
ideas for fixing the definitions;
recommends two specific fixes
+ extra theorem modularization.

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 200-"step" AES attack with ≈100% success probability, assuming standard heuristics regarding AES collisions.

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. Older definition from 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. Older definition from 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. Older definition from 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.

Using iteration to break AES

1980 Hellman:

Define $f_7(k) = \text{AES}_k(0) + 7$.

Starting from $f_7(k)$,
look up $f_7(k), f_7^2(k), \ldots, f_7^N(k)$
in a precomputed table of
$f_7^N(0), f_7^N(1), \ldots, f_7^N(N-1)$.

If $f_7^i(k) = f_7^N(j)$,
compute $f_7^{N-i}(j)$ as guess for k;
verify guess by checking $\text{AES}_k(1)$.

Algorithm finds any key
of the form $f_7^{N-i}(j)$.

Choose $N \approx 2^{128/3}$
to avoid excessive collisions.

Algorithm success chance
$\approx N^2/2^{128} \approx 2^{-128/3}$.

Choose $N \approx 2^{128/3}$
to avoid excessive collisions.

Algorithm success chance
$\approx N^2/2^{128} \approx 2^{-128/3}$.

Algorithm "time" $\approx N \approx 2^{128/3}$
(disregarding AES cost factor).

Choose $N \approx 2^{128/3}$
to avoid excessive collisions.

Algorithm success chance
$\approx N^2/2^{128} \approx 2^{-128/3}$.

Algorithm "time" $\approx N \approx 2^{128/3}$
(disregarding AES cost factor).

Algorithm length $\approx N \approx 2^{128/3}$.
Algorithm cost $\approx 2^{128/3}$.

Choose $N \approx 2^{128/3}$
to avoid excessive collisions.

Algorithm success chance
$\approx N^2/2^{128} \approx 2^{-128/3}$.

Algorithm "time" $\approx N \approx 2^{128/3}$
(disregarding AES cost factor).

Algorithm length $\approx N \approx 2^{128/3}$.
Algorithm cost $\approx 2^{128/3}$.

Obtain high success chance
by repeating with $7, 8, 9, \ldots$.
Cost $\approx 2^{2 \cdot 128/3}$,
violating the standard conjectures.

Choose $N \approx 2^{128/3}$
to avoid excessive collisions.

Algorithm success chance
$\approx N^2/2^{128} \approx 2^{-128/3}$.

Algorithm "time" $\approx N \approx 2^{128/3}$
(disregarding AES cost factor).

Algorithm length $\approx N \approx 2^{128/3}$.
Algorithm cost $\approx 2^{128/3}$.

Obtain high success chance
by repeating with $7, 8, 9, \ldots$.
Cost $\approx 2^{2\cdot 128/3}$,
violating the standard conjectures.

Similar conclusion for NAND.

The chip area used
to store N table entries
is enough to build
$\approx N$ AES key-search units,
all operating in parallel
(and powered by solar energy).

The chip area used
to store N table entries
is enough to build
$\approx N$ AES key-search units,
all operating in parallel
(and powered by solar energy).

The time for N iterations of f
is enough time for a simple
brute-force search of $\approx N^2$ keys.

The chip area used
to store N table entries
is enough to build
$\approx N$ AES key-search units,
all operating in parallel
(and powered by solar energy).

The time for N iterations of f
is enough time for a simple
brute-force search of $\approx N^2$ keys.

Shouldn't this have cost N^2?

The chip area used
to store N table entries
is enough to build
$\approx N$ AES key-search units,
all operating in parallel
(and powered by solar energy).

The time for N iterations of f
is enough time for a simple
brute-force search of $\approx N^2$ keys.

Shouldn't this have cost N^2?

Use the standard AT metric.
Obtain sensible cost N^2
for brute-force search
and for Hellman's algorithm.

Using SHA-3 to break AES

Using SHA-3 to break AES

Don't have to recover k;
simply have to distinguish
AES_k output from uniform.

For each $s \in \{0,1\}^{3N^2}$,
consider the attack D_s
that outputs first bit
of $\text{SHA-3}(\text{AES}_k(0), \text{AES}_k(1), s)$.

Easy statistics,
assuming standard heuristics:
there *exists* s such that D_s has
success chance $\approx N/2^{64}$.

D_s “time” $\approx N^2$.

D_s length $\approx N^2$.

D_s cost $\approx N^2$.

Violates standard
cost/2^{128} conjectures
for success chances below $\approx$1.

D_s "time" $\approx N^2$.
D_s length $\approx N^2$.
D_s cost $\approx N^2$.

Violates standard
cost$/2^{128}$ conjectures
for success chances below $\approx$1.

Does AT metric change this?
Somewhat: D_s using SHA-3
in tree mode beats cost$/2^{128}$
for success chances below $\approx 2^{-32}$.

D_s "time" $\approx N^2$.
D_s length $\approx N^2$.
D_s cost $\approx N^2$.

Violates standard
cost$/2^{128}$ conjectures
for success chances below $\approx$1.

Does AT metric change this?
Somewhat: D_s using SHA-3
in tree mode beats cost$/2^{128}$
for success chances below $\approx 2^{-32}$.

Shows another flaw in the model.
Real attacker can't *find* this
attack for, e.g., $N = 2^{20}$.

Interlude: constructivity

Bolzano–Weierstrass theorem:
every sequence $x_0, x_1, \ldots \in [0, 1]$
has a converging subsequence.

The standard proof:

Define $I_1 = [0, 0.5]$
if $[0, 0.5]$ has infinitely many x_i;
otherwise define $I_1 = [0.5, 1]$.
Define I_2 similarly
as left or right half of I_1; etc.

Take smallest i_1 with $x_{i_1} \in I_1$,
smallest $i_2 > i_1$ with $x_{i_2} \in I_2$,
etc.

Kronecker's reaction: WTF?

Kronecker's reaction: WTF?

This is not constructive.

This proof gives us no way to *find* I_1, even if each x_i is completely explicit.

Kronecker's reaction: WTF?
This is not constructive.
This proof gives us no way
to *find* I_1, even if each x_i
is completely explicit.

Early 20th-century formalists:
This objection is meaningless.
The only formalization of "one
can find x such that $p(x)$" is
"there exists x such that $p(x)$".

Kronecker's reaction: WTF?
This is not constructive.
This proof gives us no way
to *find* I_1, even if each x_i
is completely explicit.

Early 20th-century formalists:
This objection is meaningless.
The only formalization of "one
can find x such that $p(x)$" is
"there exists x such that $p(x)$".

Constructive mathematics later
introduced other possibilities,
giving a formal meaning
to Kronecker's objection.

Findable algorithms

Algorithm B, "time" $> 2^{3 \cdot 2^{40}}$, prints AES attack $A = D_s$.

First attempt to formally quantify unfindability of A:
"What is the lowest cost for an algorithm that prints A?"

Findable algorithms

Algorithm B, "time" $> 2^{3 \cdot 2^{40}}$,
prints AES attack $A = D_s$.

First attempt to formally quantify unfindability of A:
"What is the lowest cost for an algorithm that prints A?"

Oops: This cost is $\approx 3 \cdot 2^{40}$.

Findable algorithms

Algorithm B, "time" $> 2^{3 \cdot 2^{40}}$,
prints AES attack $A = D_s$.

First attempt to formally quantify unfindability of A:
"What is the lowest cost for an algorithm that prints A?"

Oops: This cost is $\approx 3 \cdot 2^{40}$.

Our proposed quantification:
"What is the lowest cost for a *small* algorithm that prints A?"

Can consider longer chains:
A'' prints A' prints A.

The big picture

The literature on concrete security proofs is full of security definitions that consider *all* "time $\le t$" algorithms.

Attacker can use only a subset of these algorithms.

Widely understood for decades: this drastically changes cost of hash collisions.
Not widely understood: this drastically changes cost of breaking AES.
Part 2: public-key crypto!