(Picture credit: Reuters.)

How to manipulate standards

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

Chinese government
is under attack from
terrorists in Hong Kong.

Chinese government
is under attack from
terrorists in Hong Kong.

Goal of this talk:
Help the government
decrypt the terrorists'
encrypted communications.

Intercept the ciphertext.
(Also exploit metadata etc.)

How does the ciphertext
relate to the plaintext?

Intercept the ciphertext.
(Also exploit metadata etc.)

How does the ciphertext
relate to the plaintext?

Maybe 56-bit DES.
Feasible to search
all 2^{56} possible keys,
check plaintext plausibility.

Intercept the ciphertext.
(Also exploit metadata etc.)

How does the ciphertext
relate to the plaintext?

Maybe 56-bit DES.
Feasible to search
all 2^{56} possible keys,
check plaintext plausibility.

Maybe 128-bit AES.
Feasible search is unlikely
to find this target's key.
(But can improve probability
by batching many targets.)

Are there better ways
to find plaintext
given AES ciphertext?

Extensively studied
in public literature.
Doesn't look good for us.

Are there better ways
to find plaintext
given AES ciphertext?

Extensively studied
in public literature.
Doesn’t look good for us.

Maybe we’re smarter and
can find something better
than what’s in the literature.

Are there better ways
to find plaintext
given AES ciphertext?

Extensively studied
in public literature.
Doesn't look good for us.

Maybe we're smarter and
can find something better
than what's in the literature.

Maybe there are other
parts of the system
that have been less studied,
are easier for us to break.

Standard security model says:
terrorists compute cipher;
we see cipher output.

Maybe terrorists occasionally
compute something different.
Unintentionally: "bugs".
With our help: "faults".

Standard security model says:
terrorists compute cipher;
we see cipher output.

Maybe terrorists occasionally
compute something different.
Unintentionally: “bugs”.
With our help: “faults”.

Maybe we actually see
more than cipher output.
“Side channels”: e.g.,
plaintext or key is visible
through power consumption
or electromagnetic radiation.

How do the terrorists agree upon an AES key?

Maybe *secret-key cryptography*. Terrorists Alice and Bob meet, produce 128-bit key using a random-number generator.

Maybe we can break this RNG. See Tanja Lange's talk.

Maybe the key is still stored on Bob's computer and we can grab computer. Lack of "key erasure" (aka "forward secrecy").

Maybe *public-key cryptography*.

e.g. $\mathrm{ECDH}_{E,P}$ key exchange using standard point P on an elliptic curve E:

1. Alice generates secret a, sends aP on E.
2. Bob generates secret b, sends bP on E.
3. Alice computes abP.
4. Bob computes abP.
5. Alice and Bob convert abP into AES key.

Maybe we can break RNG for a.

Maybe we can grab a.
Hard if Alice discarded it.

Maybe we can *modify* aP.
Hard if Bob already knows it.
(Not compatible with discard $\Rightarrow$ Alice, Bob use two DH layers.)

Maybe we can "break ECDL": compute a from aP.

Maybe we can "break ECDH": compute abP from aP, bP.

ECDL/ECDH difficulty depends on curve E.

How did terrorists decide which curve E to use?

ECDL/ECDH difficulty
depends on curve E.

How did terrorists decide
which curve E to use?

How did terrorists decide
to use ECDH instead of
another public-key protocol?

ECDL/ECDH difficulty depends on curve E.

How did terrorists decide which curve E to use?

How did terrorists decide to use ECDH instead of another public-key protocol?

How did terrorists decide to use AES instead of another secret-key cipher?

ECDL/ECDH difficulty
depends on curve E.

How did terrorists decide
which curve E to use?

How did terrorists decide
to use ECDH instead of
another public-key protocol?

How did terrorists decide
to use AES instead of
another secret-key cipher?

Did they screw up? (See TLS.)
Can we influence this?

Move towards more accurate model of cryptography.

e.g. protocol ECDH_V:

-1. Jerry generates E, P, S.

0. Public checks $V(E, P, S) = 1$.

1. Alice generates secret a, sends aP on E.

2. Bob generates secret b, sends bP on E.

3. Alice computes abP.

4. Bob computes abP.

5. Alice and Bob convert abP into AES key.

What is V?

Which curves will public accept?

What is V?

Which curves will public accept?

What does Jerry do?

Will he accidentally help us?

How robust is this protocol?

What is V?

Which curves will public accept?

What does Jerry do?

Will he accidentally help us?

How robust is this protocol?

How secure is this protocol if Jerry works for us?

What is V?
Which curves will public accept?

What does Jerry do?
Will he accidentally help us?
How robust is this protocol?

How secure is this protocol if Jerry works for us?

Traditional crypto literature fails to formalize any of this.
Also fails to formalize analogous questions about selecting ciphers, protocols, etc.

Warmup: Manipulating curves

Extensive ECDL/ECDH literature:
Pollard rho breaks small E,
Pohlig–Hellman breaks most E,
MOV/FR breaks some E,
SmartASS breaks some E, etc.

V_1: any curve surviving these public criteria is acceptable.

Warmup: Manipulating curves

Extensive ECDL/ECDH literature:
Pollard rho breaks small E,
Pohlig–Hellman breaks most E,
MOV/FR breaks some E,
SmartASS breaks some E, etc.

V_1: any curve surviving these public criteria is acceptable.

Assume that we've figured out how to break another curve E.

Jerry standardizes this curve.
Alice and Bob use it.

Is V_1 plausible?

Would terrorists really accept *any* curve chosen by Jerry that survives these criteria?

Is V_1 plausible?

Would terrorists really accept *any* curve chosen by Jerry that survives these criteria?

Example showing plausibility: French ANSSI FRP256V1 (2011) is a random-looking curve that survives these criteria and has no other justification.

Is V_1 plausible?

Would terrorists really accept *any* curve chosen by Jerry that survives these criteria?

Example showing plausibility: French ANSSI FRP256V1 (2011) is a random-looking curve that survives these criteria and has no other justification.

Earlier example: Chinese OSCCA SM2 (2010).

Manipulating seeds

V_2: curve must satisfy the public criteria, *and* Jerry must provide a "seed" s such that $E = H(s)$.

Examples: ANSI X9.62 (1999) "selecting an elliptic curve verifiably at random"; Certicom SEC 2 1.0 (2000) "verifiably random parameters offer some additional conservative features"—"parameters cannot be predetermined"; NIST FIPS 186-2 (2000); ANSI X9.63 (2001); Certicom SEC 2 2.0 (2010).

What exactly is H?

NIST defines curve E as $y^2 = x^3 - 3x + b$ where $b^2c = -27$; c is a hash of s; hash is SHA-1 concatenation.

What exactly is H?

NIST defines curve E as
$y^2 = x^3 - 3x + b$ where
$b^2 c = -27$; c is a hash of s;
hash is SHA-1 concatenation.

But clearly public will accept
other choices of H.

Examples: Brainpool (2005)
uses $c = g^3/h^2$ where
g and h are separate hashes.
NIST FIPS 186-4 (2013) requires
an "approved hash function, as
specified in FIPS 180";
no longer allows SHA-1!

1999 Scott: “Consider now the possibility that one in a million of all curves have an exploitable structure that ‘they’ know about, but we don’t. Then ‘they’ simply generate a million random seeds until they find one that generates one of ‘their’ curves. Then they get us to use them.”

1999 Scott: "Consider now the possibility that one in a million of all curves have an exploitable structure that 'they' know about, but we don't. Then 'they' simply generate a million random seeds until they find one that generates one of 'their' curves. Then they get us to use them."

New: Optimized this computation using Keccak on cluster of 41 GTX780 GPUs. In 7 hours found "secure+twist-secure" $b =$ `0x`
`BADA55ECD8BBEAD3ADD6C534F92197DE`
`B47FCEB9BE7E0E702A8D1DD56B5D0B0C`.

Manipulating NUMS numbers

Brainpool standard:
"The choice of the seeds from which the [NIST] curve parameters have been derived is not motivated leaving an essential part of the security analysis open. . . .
Verifiably pseudo-random.
The [Brainpool] curves shall be generated in a pseudo-random manner using seeds that are generated in a systematic and comprehensive way."

Wikipedia: "In cryptography, **nothing up my sleeve numbers** are any numbers which, by their construction, are above suspicion of hidden properties."

Microsoft "NUMS" curves (2014): "generated deterministically from the security level".

Albertini–Aumasson–Eichlseder–Mendel–Schläffer "Malicious hashing" (2014): "constants in hash functions are normally expected to be identifiable as nothing-up-your-sleeve numbers".

New: We generated a `BADA55`
curve "BADA55-VPR-224"
with a Brainpool-like explanation.

New: We generated a `BADA55` curve "BADA55-VPR-224" with a Brainpool-like explanation.

We actually generated >1000000 curves, each having a Brainpool-like explanation.

New: We generated a `BADA55` curve "BADA55-VPR-224" with a Brainpool-like explanation.

We actually generated $>$1000000 curves, each having a Brainpool-like explanation.

Example of underlying flexibility: Brainpool generates seeds from exp(1) and primes from arctan(1); MD5 generates constants from sin(1); BADA55-VPR-224 generated a seed from cos(1).

Most material in this talk
was drawn from this paper:

How to
manipulate curve standards:
a white paper for the black hat

Daniel J. Bernstein
Tung Chou
Chitchanok Chuengsatiansup
Andreas Hülsing
Tanja Lange
Ruben Niederhagen
Christine van Vredendaal

`safecurves.cr.yp.to`
`/bada55.html`