Making sure crypto stays insecure

Daniel J. Bernstein

University of Illinois at Chicago &
Technische Universiteit Eindhoven

Terrorist in Hong Kong
prepares to throw deadly weapon
at Chinese government workers.

Image credit: Reuters.

Drug-dealing cartel “Starbucks” invades city in Morocco; begins selling addictive liquid.

Image credit: Wikipedia.

Pedophile convinces helpless child to remove most of her clothing; sexually abuses child in public.

Image credit: Child pornographer.

theguardian

News | Sport | Comment | Culture | Business | Money | Life & Offers | Jobs

News › US news › US national security

Series: Glenn Greenwald on security and liberty

NSA collecting phone records of millions of Verizon customers daily

Exclusive: Top secret court order requiring Verizon to hand over all call data shows scale of domestic surveillance under Obama

• Read the Verizon court order in full here

Criminal organization
calling itself “The Guardian”
sells classified government secrets.

Image credit: The Guardian.

We have to watch and listen to everything that people are doing so that we can catch terrorists, drug dealers, organized criminals, pedophiles, murderers, etc.

We have to watch and listen to everything that people are doing so that we can catch terrorists, drug dealers, organized criminals, pedophiles, murderers, etc.

We try to systematically monitor and record all Internet traffic.
But what if it's encrypted?

We have to watch and listen to
everything that people are doing
so that we can catch terrorists,
drug dealers, organized criminals,
pedophiles, murderers, etc.

We try to systematically monitor
and record all Internet traffic.
But what if it's encrypted?

This talk gives some examples
of how we've manipulated
the world's crypto ecosystem
so that we can understand
almost all of this traffic.

- (TS//SI//REL TO USA, FVEY) Insert vulnerabilities into commercial encryption systems, IT systems, networks, and endpoint communications devices used by targets.
- (TS//SI//REL TO USA, FVEY) Collect target network data and metadata via cooperative network carriers and/or increased control over core networks.
- (TS//SI//REL TO USA, FVEY) Leverage commercial capabilities to remotely deliver or receive information to and from target endpoints.
- (TS//SI//REL TO USA, FVEY) Exploit foreign trusted computing platforms and technologies.
- (TS//SI//REL TO USA, FVEY) Influence policies, standards and specification for commercial public key technologies.
- (TS//SI//REL TO USA, FVEY) Make specific and aggressive investments to facilitate the development of a robust exploitation capability against Next-Generation Wireless (NGW) communications.

(TS//SI//ECI SOL) Fact that NSA/CSS works with and has contractual relationships with specific named U.S. commercial entities (A/B/C) to conduct SIGINT enabling programs and operations.
(TS//SI// ECI SOL) Fact that NSA/CSS works with specific named U.S. commercial entities (A/B/C) and operational details (devices/products) to make them exploitable for SIGINT.
(TS//SI// ECI SOL) Fact that NSA/CSS works with specific foreign partners (X/Y/Z) and foreign commercial industry entities (M/N/O) and operational details (devices/products) to make them exploitable for SIGINT.
(TS//SI//ECI SOL) Facts related to NSA personnel (under cover), operational meetings, specific operations, specific technology, specific locations and covert communications related to SIGINT enabling with specific commercial entities (A/B/C).
(TS//SI//ECI SOL) Facts related to NSA/CSS working with U.S. commercial entities on the acquisition of communications (content and metadata) provided by the U.S. service provider to worldwide customers; communications transiting the U.S.; or access to international communications (cable, satellite, etc.) mediums provided by the U.S. entity.
(TS//SI// ECI SOL) Facts that identify a U.S. or foreign commercial platform conducting SIGINT operations, or human asset cooperating with NSA/CSS.

Other useful strategies,
not covered in this talk:

Manipulate *software* ecosystem
so that software stays insecure.
Break into computers; access
hundreds of millions of disks,
screens, microphones, cameras.

Other useful strategies,
not covered in this talk:

Manipulate *software* ecosystem
so that software stays insecure.
Break into computers; access
hundreds of millions of disks,
screens, microphones, cameras.

Add back doors to *hardware*.
e.g. 2012 U.S. government report
says that Chinese-manufactured
routers provide "Chinese
intelligence services access to
telecommunication networks".

Some important clarifications

1. "We" doesn't include me.

I want secure crypto.

Some important clarifications

1. "We" doesn't include me.

I want secure crypto.

2. Their actions violate fundamental human rights.

Some important clarifications

1. "We" doesn't include me.
I want secure crypto.

2. Their actions violate
fundamental human rights.

3. I don't know how much
of today's crypto ecosystem
was deliberately manipulated.

Some important clarifications

1. "We" doesn't include me.
I want secure crypto.

2. Their actions violate
fundamental human rights.

3. I don't know how much
of today's crypto ecosystem
was deliberately manipulated.

This talk is actually
a thought experiment:
how *could* an attacker manipulate
the ecosystem for insecurity?

Timing attacks

2005 Osvik–Shamir–Tromer:
65ms to steal Linux AES key
used for hard-disk encryption.
Attack process on same CPU
but without privileges.

Almost all AES implementations
use fast lookup tables.
Kernel's secret AES key
influences table-load addresses,
influencing CPU cache state,
influencing measurable timings
of the attack process.
65ms: compute key from timings.

2011 Brumley–Tuveri:
minutes to steal another
machine's OpenSSL ECDSA key.
Secret branch conditions
influence timings.

Most cryptographic software
has many more small-scale
variations in timing:
e.g., `memcmp` for IPsec MACs.

Many more timing attacks: e.g.
2014 van de Pol–Smart–Yarom
extracted Bitcoin secret keys
from 25 OpenSSL signatures.

Manufacture public denials that such attacks exist.

Maybe terrorists Alice and Bob won't try to stop the attacks.

2001 NIST "Report on the development of the Advanced Encryption Standard (AES)": "A general defense against timing attacks is to ensure that each encryption and decryption operation runs in the same amount of time. . . . Table lookup: not vulnerable to timing attacks."

2008 RFC 5246 “The Transport Layer Security (TLS) Protocol, Version 1.2”: “This leaves a small timing channel, since MAC performance depends to some extent on the size of the data fragment, but it is not believed to be large enough to be exploitable, due to the large block size of existing MACs and the small size of the timing signal.”

2008 RFC 5246 "The Transport Layer Security (TLS) Protocol, Version 1.2": "This leaves a small timing channel, since MAC performance depends to some extent on the size of the data fragment, but it is not believed to be large enough to be exploitable, due to the large block size of existing MACs and the small size of the timing signal."

2013 AlFardan–Paterson "Lucky Thirteen: breaking the TLS and DTLS record protocols": exploit these timings; steal plaintext.

Some instructions have no data flow from their inputs to CPU timings: e.g., logic instructions, constant-distance shifts, multiply (on most CPUs), add, subtract.

What if Alice and Bob use crypto software built solely from these instructions? Yikes: we won't see anything from timings!

Some instructions have no data flow from their inputs to CPU timings: e.g., logic instructions, constant-distance shifts, multiply (on most CPUs), add, subtract.

What if Alice and Bob use crypto software built solely from these instructions? Yikes: we won't see anything from timings!

Try to scare implementors away from constant-time software.
e.g. "It will be too slow."
"It's too hard to write."

Fund variable-time software,
maybe with “countermeasures”
that make the timings difficult
for researchers to analyze
but that are still breakable
with our computer resources.

Fund variable-time software,
maybe with "countermeasures"
that make the timings difficult
for researchers to analyze
but that are still breakable
with our computer resources.

Continue expressing skepticism
that constant time is needed.
e.g. 2012 Mowery–Keelveedhi–
Shacham "Are AES x86 cache
timing attacks still feasible?",
unfortunately shredded by 2014
Irazoqui–Inci–Eisenbarth–Sunar
"Wait a minute! A fast,
cross-VM attack on AES".

What if terrorists Alice and Bob use a different cipher for which constant-time implementations are simple and fast? Yikes!

Don't standardize that cipher. e.g. choose Rijndael as AES, not higher-security Serpent. Watch out for any subsequent standardization efforts.

Discourage use of the cipher. Pretend that standardization is a guarantee of security while anything non-standard has questionable security.

Padding oracles

1998 Bleichenbacher:
Decrypt SSL RSA ciphertext
by observing server responses
to $\approx 10^6$ variants of ciphertext.

SSL first inverts RSA,
then checks for "PKCS padding"
(which many forgeries have).
Subsequent processing applies
more serious integrity checks.

Server responses reveal
pattern of PKCS forgeries;
pattern reveals plaintext.

Design cryptographic systems
so that forgeries are sent through
as much processing as possible.

e.g. Design SSL to decrypt
and check padding before
checking a serious MAC.
Broken by padding-oracle attacks
such as BEAST and POODLE.

e.g. Design "encrypt-only"
IPsec options. Broken by 2006
Paterson–Yau for Linux and 2007
Degabriele–Paterson for RFCs.

Randomness

1995 Goldberg–Wagner: Netscape SSL keys had <50 bits of entropy.

2008 Bello: Debian/Ubuntu OpenSSL keys for years had <20 bits of entropy.

2012 Lenstra–Hughes–Augier–Bos–Kleinjung–Wachter and 2012 Heninger–Durumeric–Wustrow–Halderman broke the RSA public keys for 0.5% of all SSL servers. The primes had so little randomness that they collided.

Make randomness-generation code extremely difficult to audit.

Have each application maintain its own RNG "for speed".

Maintain separate RNG *code* for each application. "For simplicity" build this RNG in ad-hoc ways from the inputs conveniently available to that application.

Pay people to use backdoored RNGs such as Dual EC.
Claim "provable security".

What if the terrorists
merge all available inputs
into a central entropy pool?

This pool can survive many
bad/failing/malicious inputs
if there is one good input.
Merging process is auditable.
Yikes!

What if the terrorists
merge all available inputs
into a central entropy pool?

This pool can survive many
bad/failing/malicious inputs
if there is one good input.
Merging process is auditable.
Yikes!

Claim performance problems in
writing to a central pool,
reading from a central pool.
Modify pool to make it unusable
(`random`) or scary (`urandom`).

What if the terrorists realize that RNG speed isn't an issue?

Make it an issue! Design crypto to use randomness as often as possible. This also complicates tests, encouraging bugs.

e.g. DSA and ECDSA use a new random number k to sign m; could have replaced k with $H(s, m)$. 1992 Rivest: "the poor user is given enough rope with which to hang himself". 2010 Bushing–Marcan–Segher–Sven "PS3 epic fail": PS3 forgeries.

Pure crypto failures

2008 Stevens–Sotirov–Appelbaum–Lenstra–Molnar–Osvik–de Weger exploited MD5 $\Rightarrow$ rogue CA for TLS.

Pure crypto failures

2008 Stevens–Sotirov–Appelbaum–Lenstra–Molnar–Osvik–de Weger exploited MD5 $\Rightarrow$ rogue CA for TLS.

2012 Flame: new MD5 attack.

Pure crypto failures

2008 Stevens–Sotirov–Appelbaum–Lenstra–Molnar–Osvik–de Weger exploited MD5 $\Rightarrow$ rogue CA for TLS.

2012 Flame: new MD5 attack.

Fact: By 1996, a few years after the introduction of MD5, Preneel and Dobbertin were calling for MD5 to be scrapped.

Pure crypto failures

2008 Stevens–Sotirov–Appelbaum–Lenstra–Molnar–Osvik–de Weger exploited MD5 $\Rightarrow$ rogue CA for TLS.

2012 Flame: new MD5 attack.

Fact: By 1996, a few years after the introduction of MD5, Preneel and Dobbertin were calling for MD5 to be scrapped.

We managed to keep MD5. How? Speed; standards; compatibility.

2014: DNSSEC uses RSA-1024 to “secure” IP addresses.

e.g. `dnssec-deployment.org` address is signed by RSA-1024.

2014: DNSSEC uses RSA-1024
to "secure" IP addresses.
e.g. `dnssec-deployment.org`
address is signed by RSA-1024.

Fact: Analyses in 2003 concluded
that RSA-1024 was breakable;
e.g., 2003 Shamir–Tromer
estimated 1 year, $\approx 10^7$ USD.

2014: DNSSEC uses RSA-1024
to "secure" IP addresses.
e.g. `dnssec-deployment.org`
address is signed by RSA-1024.

Fact: Analyses in 2003 concluded
that RSA-1024 was breakable;
e.g., 2003 Shamir–Tromer
estimated 1 year, $\approx 10^7$ USD.

DNSSEC's main excuse
for sticking to RSA-1024: speed.
"Tradeoff between the risk of key
compromise and performance."

How to convince terrorists
that secure crypto is too slow?

Many techniques: obsolete data,
incompetent benchmarks, fraud.

How to convince terrorists
that secure crypto is too slow?

Many techniques: obsolete data,
incompetent benchmarks, fraud.

Example:
"*PRESERVE contributes to the security and privacy of future vehicle-to-vehicle and vehicle-to-infrastructure communication systems by addressing critical issues like performance, scalability, and deployability of V2X security systems.*"
preserve-project.eu

"[In] most driving situations . . . the packet rates do not exceed 750 packets per second. Only the maximum highway scenario . . . goes well beyond this value (2,265 packets per second). . . .

Processing 1,000 packets per second and processing each in 1 ms can hardly be met by current hardware. As discussed in [32], a Pentium D 3.4 GHz processor needs about 5 times as long for a verification . . . a dedicated cryptographic co-processor is likely to be necessary."

Compare to “NEON crypto”
on 1GHz Cortex-A8 core:
5.48 cycles/byte (1.4 Gbps),
2.30 cycles/byte (3.4 Gbps)
for Salsa20, Poly1305.
498349 cycles (2000/second),
624846 cycles (1600/second)
for Curve25519 DH, verify.

Compare to "NEON crypto"
on 1GHz Cortex-A8 core:
5.48 cycles/byte (1.4 Gbps),
2.30 cycles/byte (3.4 Gbps)
for Salsa20, Poly1305.
498349 cycles (2000/second),
624846 cycles (1600/second)
for Curve25519 DH, verify.

1GHz Cortex-A8 was high-end
smartphone core in 2010: e.g.,
Samsung Exynos 3110 (Galaxy S);
TI OMAP3630 (Motorola Droid
X); Apple A4 (iPad 1/iPhone 4).

Compare to "NEON crypto"
on 1GHz Cortex-A8 core:
5.48 cycles/byte (1.4 Gbps),
2.30 cycles/byte (3.4 Gbps)
for Salsa20, Poly1305.
498349 cycles (2000/second),
624846 cycles (1600/second)
for Curve25519 DH, verify.

1GHz Cortex-A8 was high-end
smartphone core in 2010: e.g.,
Samsung Exynos 3110 (Galaxy S);
TI OMAP3630 (Motorola Droid
X); Apple A4 (iPad 1/iPhone 4).

2013: Allwinner A13, $5 in bulk.

What if the terrorists
hear about fast secure crypto?
Yikes!

Similar to constant-time story.
Don't standardize good crypto.
Discourage use of good crypto.

What if the terrorists
hear about fast secure crypto?
Yikes!

Similar to constant-time story.
Don't standardize good crypto.
Discourage use of good crypto.

If the good crypto persists,
try to bury it behind
a huge menu of bad options.
Advertise "cryptographic agility";
actually cryptographic fragility.
Pretend that this "agility"
justifies using breakable crypto.

Precomputed signatures

Try to build cryptographic fragility into many layers of the system.

e.g. Complicate the protocols.

Precomputed signatures

Try to build cryptographic fragility into many layers of the system.

e.g. Complicate the protocols.

Split cryptographic security
into "the easy problem"
of protecting integrity
and "the hard problem"
of protecting confidentiality.

Precomputed signatures

Try to build cryptographic fragility into many layers of the system.

e.g. Complicate the protocols.

Split cryptographic security
into "the easy problem"
of protecting integrity
and "the hard problem"
of protecting confidentiality.

e.g. argue against encrypted SNI since DNS is unencrypted,
and argue against encrypted DNS since SNI is unencrypted.

Solve “the easy problem”
by precomputing signatures.
Insist that the protocol
allow precomputation “for speed”.
e.g. DNSSEC.

Solve “the easy problem”
by precomputing signatures.
Insist that the protocol
allow precomputation “for speed”.
e.g. DNSSEC.

The protocol has trouble handling
dynamically generated answers,
and unpredictable questions; also,
trouble guaranteeing freshness.
Deployment hits many snags.

Solve “the easy problem”
by precomputing signatures.
Insist that the protocol
allow precomputation “for speed”.
e.g. DNSSEC.

The protocol has trouble handling
dynamically generated answers,
and unpredictable questions; also,
trouble guaranteeing freshness.
Deployment hits many snags.

Argue that it’s too early
to look at “the hard problem”
when most data is still unsigned.

More strategies

Divert "crypto" funding
and human resources
into activities that don't
threaten mass surveillance.

Set up centralized systems
encrypting data to companies
that collaborate with us.

More distraction: build systems
breakable by active attacks.

Declare crypto success
without encrypting the Internet.