Cryptographic Backdoorering

JP Aumasson
/me

@veorq  http://aumasson.jp
Agenda

Why this talk?
Backdooring 101
Sabotage tactics
A perfect backdoor
Conclusion
Why this talk?
You may not be interested in backdoors, but backdoors are interested in you
Base resources in this project are used to:

- **(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.
- **(II//FOUO)** Maintain understanding of commercial business and technology trends.

NSA’s BULLRUN program
Public/academic research mostly inexistant
Bad reputation
Surveillance, deception, etc.
“a back door for the government can easily —and quietly—become a back door for criminals and foreign intelligence services.”

Security “Front Doors” vs. “Back Doors”: A Distinction Without a Difference

By Jeffrey Vagle and Matt Blaze
Friday, October 17, 2014 at 2:06 PM

And terrorists etc.
(Like internet and encryption)
Not a great argument IMHO
“It increases the ‘attack surface’ of the system, providing new points of leverage that a nefarious attacker can exploit.”

Security “Front Doors” vs. “Back Doors”: A Distinction Without a Difference

By Jeffrey Vagle and Matt Blaze

Friday, October 17, 2014 at 2:06 PM

Crypto backdoors are dangerous even if you trust the government not to abuse them. We simply don't know how to build them reliably.
Not well understood, by the public
Especially **crypto** backdoors
Why public research?
Detect backdoors
If you have to implement a backdoor— for good or not-so-good reasons— better know how (not) to do it.
Backdooring 101
What is a backdoor?
Not a trapdoor

(Covert rather than overt)
“A feature or defect that allows surreptitious access to data”
Weakened algorithms
(A5/2, GMR, etc.)
Covert channels
(Exfiltration of keys, etc.)
Key escrow

AT&T MAKES NO WARRANTY THAT THE TSD WILL PREVENT CRYPTOANALYTIC ATTACK ON ANY ENCRYPTED TRANSMISSION BY ANY GOVERNMENT AGENCY, ITS AGENTS, OR ANY THIRD PARTIES. FURTHERMORE, AT&T MAKES NO WARRANTY THAT THE TSD WILL PREVENT ANY ATTACK ON ANY COMMUNICATION BY METHODS WHICH BYPASS ENCRYPTION.

Clipper chip phone AT&T TSD3600
May be known to exist
(Is lawful interception a backdoor?)
“An undocumented way to get access to a computer system or the data it contains”
Breakthrough silicon scanning discovers backdoor in military chip (DRAFT of 05 March 2012)

Sergei Skorobogatov  
University of Cambridge  
Cambridge, UK  
sp32@cam.ac.uk

Christopher Woods  
Quo Vadis Labs  
London, UK  
chris@quovadislabs.com
Bugs? RCE?
Only if intentional, a.k.a. **bugdoors**

(© The Gruggq)
Deniability...
What is a “good” backdoor?
Undetectable
NOBUS
(No one but us, NSA term)
Reusable
Unmodifiable
Forward-secure
Simple
To be continued...
Sabotage tactics
Constants
Choose constants that allow you to compromise the security
SHA-1 round constants
Malicious Hashing: Eve’s Variant of SHA-1

Ange Albertini\textsuperscript{1}, Jean-Philippe Aumasson\textsuperscript{2}, Maria Eichlseder\textsuperscript{3}, Florian Mendel\textsuperscript{3}, and Martin Schläffer\textsuperscript{3}

40 bits modified
Colliding binaries, images, archives
Full control on the content, NOBUS

(\textsc{BSidesLV/DEFCON/SAC} 2014)
>crypto_hash *
test0.jpg 13990732b0d16c3e112f2356bd3d0dad1....
test1.jpg 13990732b0d16c3e112f2356bd3d0dad1....

https://malicioussha1.github.io/
2 distinct files, 3 valid file formats

- `shmbrar0.mbr`
- `shmbrar0.sh`
- `shmbrar0.rar`

- `shmbrar1.mbr`
- `shmbrar1.sh`
- `shmbrar1.rar`

- Identical
- Collision
Elliptic curve coefficients
NIST curves’ coefficients:
hashes of unexplained 16-byte seeds, e.g. c49d3608 86e70493 6a6678e1 139d26b7 819f7e90
(Speculation, no public evidence of backdoor)
Notion of **rigidity**

“a feature of a curve-generation process, limiting the number of curves that can be generated by the process”

http://safecurves.cr.yp.to/rigid.html
| Curve25519 | fully rigid | Prime chosen "as close as possible to a power of 2" for efficiency reasons ("save time in field operations"). Prime chosen "slightly below 32k bits, for some k" for efficiency reasons ("no serious concerns regarding wasted space"). k=8 chosen for "a comfortable security level". 2^255-19 chosen above 2^255+95, 2^255-31, 2^254+79, 2^253+51, 2^253+39 because 19 is smaller than 31, 39, 51, 79, 95. Montgomery curve shape y^2=x^3+Ax^2+x chosen for efficiency ("to allow extremely fast x-coordinate point operations"). (A-2)/4 selected as a small integer for efficiency ("to speed up the multiplication by (A-2)/4"). Curve and twist orders required to be \{4*prime,8*prime\} for security ("protect against various attacks ... here 4, 8 are minimal"). Primes required to be above 2^252 for security ("theoretical possibility of a user's secret key matching the prime"), ruling out A=358990 and A=464586. A=486662 chosen as smallest positive integer meeting these requirements. |
| BN(2,254) | fully rigid | p chosen sparse, close to 2^256, within BN family; using u=-(2^62 + 2^55 + 1). p congruent 3 modulo 4 to have z^2+1 irreducible; b=2 to have twist be y^2=x^3+(1-2i). |
| brainpoolP25611 | somewhat rigid | Several unexplained decisions: Why SHA-1 instead of, e.g., RIPEMD-160 or SHA-256? Why use 160 bits of hash input independently of the curve size? Why pi and e instead of, e.g., sqrt(2) and sqrt(3)? Why handle separate key sizes by more digits of pi and e instead of hash derivation? Why counter mode instead of, e.g., OFB? Why use overlapping counters for A and B (producing the repeated 26DC5C5CE94A4B44F330B5D9)? Why not derive separate seeds for A and B? |
| ANSSI FRP256v1 | trivially manipulatable | No explanation provided. |
| NIST P-256 | manipulatable | Coefficients generated by hashing the unexplained seed c49d3608 86e70493 6a6678e1 139d26b7 819f7e90. |
| secp256k1 | somewhat rigid | GLV curve with 256 bits and prime order group; prime and coefficients not fully explained but might be minimal |
| E-382 | fully rigid | |
| M-383 | fully rigid | |
| Curve383187 | fully rigid | p is largest prime smaller than 2^383; B=1; A > 2 is as small as possible. |
| brainpoolP384t1 | somewhat rigid | See brainpoolP25611. |
| NIST P-384 | manipulatable | Coefficients generated by hashing the unexplained seed a335926a a319a27a 1d00896a 6773a482 7acdac73. |
Limitation: there may be an exponential number of fully-rigid generation methods
Math structure elements
If $n$ s.t. $nQ = P$ is known, the RNG is broken
Key generation
Make session keys predictable
3G/4G AKA

Session keys = hash( master key, rand )

Delegate tactical intercepts with low-entropy rand values

Precompute and share session keys

(A possibility, not allegations)
Hide weak parameters
RSA
Hide small public exponent with some tricks to avoid detection and recover using Boneh-Durfee-Frankel result

Simple Backdoors for RSA Key Generation

Claude Crépeau¹ and Alain Slakmon²

(CT-RSA 2003)
Key gen as a covert channel for itself
RSA

Hide bits of prime factors in $n$
Recover using Coppersmith’s method
Similar to “Pretty-Awful-Privacy” (Young-Yung)

Simple Backdoors for RSA Key Generation

Claude Crépeau and Alain Slakmon

(CT-RSA 2003)
Lesson: don’t outsource keygen
Implementations
Slightly deviate from the specs
Omit some verifications etc.
Small subgroup attacks
Omit (EC)DH pubkey validation

A Key Recovery Attack on Discrete Log-based Schemes Using a Prime Order Subgroup*

Chae Hoon Lim¹ and Pil Joong Lee²

(CRYPTO 1997)
Small subgroup attacks
Omit (EC)DH pubkey validation

Validation of Elliptic Curve Public Keys

Adrian Antipa¹, Daniel Brown¹, Alfred Menezes²,
René Struik¹, and Scott Vanstone²

(PKC 2003)
“domain parameter shifting attacks”
Omit ECC domain parameters validation

Digital Signature Schemes with Domain Parameters

Serge Vaudenay

(ACISP 2004)
TLS MitM
Incomplete cert verification
“Misuse”
Repeated stream cipher nonces
NOBUS unlikely...
Software
Bugdoors in the crypto
Deniability may be plausible
goto fail;
goto fail;
Those 2 are probably unintentional
RC4 bugdoor (Wagner/Biondi)

```c
#define TOBYTE(x) (x) & 255
#define SWAP(x,y) do { x^=y; y^=x; x^=y; } while (0)

static unsigned char A[256];
static int i=0, j=0;

unsigned char encrypt_one_byte(unsigned char c) {
    int k;
    i = TOBYTE( i+1 );
    j = TOBYTE( j + A[i] );
    SWAP( A[i], A[j] );
    k = TOBYTE( A[i] + A[j] );
    return c ^ A[k];
}
```
Hardware
IC trojans
Malicious modification of a chip
At design (HDL) or fab (netlist)
Detection difficult
Stealthy Dopant-Level Hardware Trojans *

Georg T. Becker¹, Francesco Regazzoni², Christof Paar¹,³, and Wayne P. Burleson¹

(CHES 2013)
Reversing Stealthy Dopant-Level Circuits

Takeshi Sugawara\textsuperscript{1}, Daisuke Suzuki\textsuperscript{1}, Ryoichi Fujii\textsuperscript{1}, Shigeaki Tawa\textsuperscript{1}
Ryohei Hori\textsuperscript{2}, Mitsuru Shiozaki\textsuperscript{2}, and Takeshi Fujino\textsuperscript{2}

(CHES 2014)
Bug Attacks

Eli Biham¹, Yaniv Carmeli¹, and Adi Shamir²

CPU multiplier $X \times Y = Z$ correct except for one “magic” pair $(X, Y)$

Exploitable to break RSA, ECC, etc.

$2^{128}$ pairs for 64-bit MUL, detection unlikely
A perfect backdoor

http://phili89.wordpress.com/2010/05/24/the-perfect-crime-project-38/
Covert channel with a malicious RNG

Public-key encryption (NOBUS)

Indistinguishability from random strings (for undetectability)
Compute $X = \text{Enc}(\text{pk}, \text{data to exfiltrate})$

$X$ should look like a random string

Use $X$ as (say) IVs for AES-CTR
Pubkey encryption scheme with ciphertexts indistinguishable from random strings?
Elligator: Elliptic-curve points indistinguishable from uniform random strings

Daniel J. Bernstein\textsuperscript{1,4}  
djb@cr.yp.to

Mike Hamburg\textsuperscript{2}  
mhamburg@cryptography.com

Anna Krasnova\textsuperscript{3}  
an@mechanical-mind.org

Tanja Lange\textsuperscript{4}  
tanja@hyperelliptic.org

\[(X, Y)\]

101010011001
Elligator curves

<table>
<thead>
<tr>
<th>Curve</th>
<th>Elligator?</th>
<th>Elligator 2?</th>
</tr>
</thead>
<tbody>
<tr>
<td>E-382</td>
<td>True ✓</td>
<td>Yes</td>
</tr>
<tr>
<td>M-383</td>
<td>True ✓</td>
<td>Yes</td>
</tr>
<tr>
<td>Curve383187</td>
<td>True ✓</td>
<td>Yes</td>
</tr>
<tr>
<td>brainpoolP384t1</td>
<td>False</td>
<td>No</td>
</tr>
<tr>
<td>NIST P-384</td>
<td>False</td>
<td>No</td>
</tr>
<tr>
<td>Curve41417</td>
<td>True ✓</td>
<td>Yes</td>
</tr>
<tr>
<td>Ed448-Goldilocks</td>
<td>True ✓</td>
<td>Yes</td>
</tr>
<tr>
<td>M-511</td>
<td>True ✓</td>
<td>Yes</td>
</tr>
<tr>
<td>E-521</td>
<td>True ✓</td>
<td>Yes</td>
</tr>
</tbody>
</table>

http://safecurves.cr.yp.to/ind.html
RNG circuit must be hidden
(For example in FPGA/PLD, difficult to RE)
Communications and computations appear identical to those of a clean system.
Full reverse-engineering: Backdoor detected but unexploitable, and previous covert coms remain safe
What can be exfiltrated? **RNG state**

Can give past and future session keys, depending on the RNG construction
Many other techniques...
Conclusion
All this is quite basic
And that’s only for crypto
Should we worry about backdoors?

or

First fix bugs and usability issues?
Draw your own conclusions
“a competition to write or modify crypto code that appears to be secure, but actually does something evil.”

Send you submission(s) before Dec 2, 2014

https://underhandedcrypto.com/
Merci!

“Secrets… are the very root of cool.”

William Gibson, *Spook Country*