Randomness of random in Cisco ASA

Ryad BENADJILA (CryptoExperts)*
ryad.benadjila@cryptoexperts.com
Arnaud EBALARD (ANSSI)
arnaud.ebalard@ssi.gouv.fr

SSTIC 2023 – June 7th (Rennes)

*Work performed while at ANSSI.

The beginning of the story ...

Work on development projects

- X-509 parser [x509-parser]
- Elliptic Curve Cryptography library libecc [libecc]

Tests on a >250 millions X.509 certificates set led to ...

>250 millions
X.509 Certs (TLS campaign)

The beginning of the story ...

Work on development projects

- X-509 parser [x509-parser]
- Elliptic Curve Cryptography library libecc [libecc]

Tests on a >250 millions X.509 certificates set led to ...

>250 millions
X.509 Certs (TLS campaign)

82k dup. ECDSA nonces

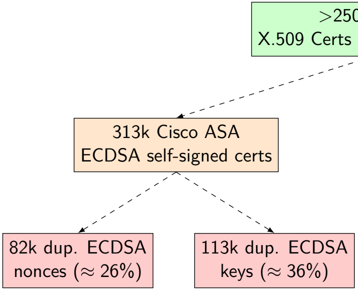
113k dup. ECDSA keys

The beginning of the story ...

Work on development projects

- X-509 parser [x509-parser]
- Elliptic Curve Cryptography library libecc [libecc]

Tests on a >250 millions X.509 certificates set led to ...

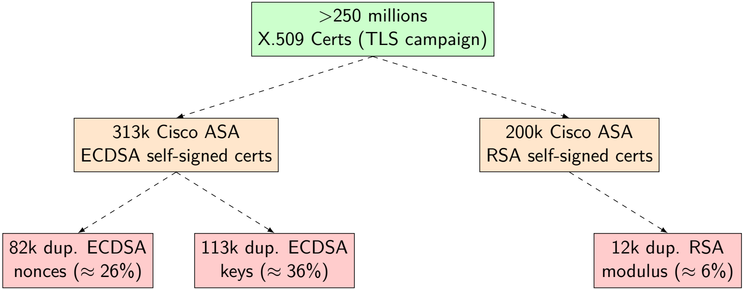


The beginning of the story ...

Work on development projects

- X-509 parser [x509-parser]
- Elliptic Curve Cryptography library libecc [libecc]

Tests on a >250 millions X.509 certificates set led to ...

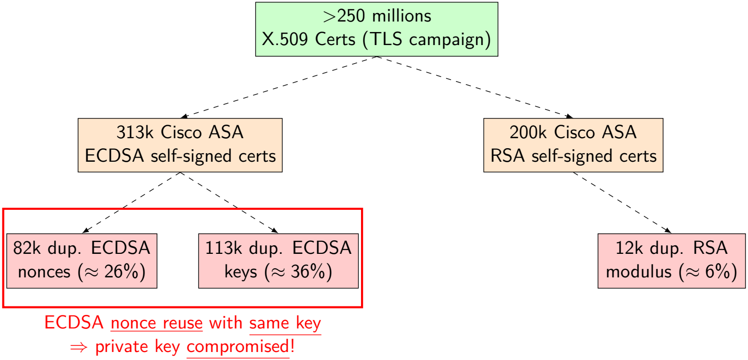


The beginning of the story ...

Work on development projects

- X-509 parser [x509-parser]
- Elliptic Curve Cryptography library libecc [libecc]

Tests on a >250 millions X.509 certificates set led to ...



ECDSA signature algorithm

...and resulting nonce and key recovery from duplicated nonce

secret / public

1: $h = H(m)$
2: $e = OS2I(h) \bmod q$
3: $k \leftarrow \mathcal{R},\ k \in]0, q[$
4: $W = (W_x, W_y) = k \times G$
5: $r = W_x\ mod\ q$
6: $s = k^{-1} \times (xr + e) mod\ q$
7: Return (r,s)

ECDSA signature algorithm

...and resulting nonce and key recovery from duplicated nonce

secret / public

1: $h = H(m)$
2: $e = OS2I(h) \bmod q$
3: $k \leftarrow \mathcal{R},\ k \in]0, q[$
4: $W = (W_x, W_y) = k \times G$
5: $r = W_x\ mod\ q$
6: $s = k^{-1} \times (xr + e) mod\ q$
7: Return (r, s)

From 6: above, we draw for two signatures (r, s_1) and (r, s_2) sharing the same duplicated nonce k for different messages:

Nonce recovery from nonce duplication

$$
\begin{aligned}
s_1 - s_2 = & k^{-1} \times (xr + e_1) - k^{-1} \times (xr + e_2)\ mod\ q \\
& k^{-1} \times (xr + e_1 - xr - e_2)\ mod\ q \\
& k^{-1} \times (e_1 - e_2)\ mod\ q
\end{aligned}
$$

$$\Longrightarrow k = (e_1 - e_2) \times (s_1 - s_2)^{-1} mod\ q$$

ECDSA signature algorithm

...and resulting nonce and key recovery from duplicated nonce

secret / public

1: $h = H(m)$
2: $e = OS2I(h) \mod q$
3: $k \leftarrow \mathcal{R}$, $k \in]0, q[$
4: $W = (W_x, W_y) = k \times G$
5: $r = W_x \ mod\ q$
6: $s = k^{-1} \times (xr + e) mod\ q$
7: Return (r, s)

From 6: above, we draw for two signatures (r, s_1) and (r, s_2) sharing the same duplicated nonce k for different messages:

Nonce recovery from nonce duplication

$$s_1 - s_2 = k^{-1} \times (xr + e_1) - k^{-1} \times (xr + e_2) \ mod\ q$$
$$k^{-1} \times (xr + e_1 - xr - e_2) \ mod\ q$$
$$k^{-1} \times (e_1 - e_2) \ mod\ q$$

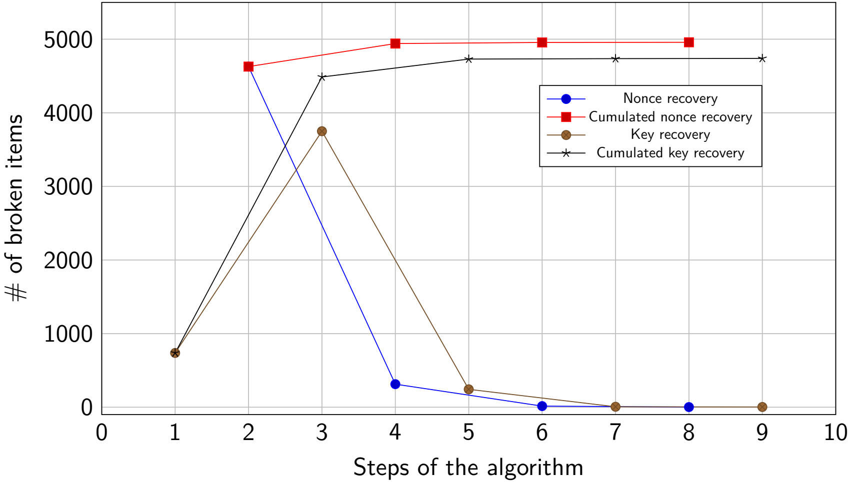
$$\Longrightarrow k = (e_1 - e_2) \times (s_1 - s_2)^{-1} mod\ q$$

key recovery from nonce

$$x = (k \times s_1 - e_1) \times {r_1}^{-1} \ mod\ q$$
$$(k \times s_2 - e_2) \times {r_2}^{-1} \ mod\ q$$

Iterative key recovery

Over 313k X.509 ASA ECDSA self-signed certificates with 216k unique keys



Some background on RNG fails ...

History

[CVE-2008-0166] 05/2008: predictible Debian OpenSSL RNG

⇒ Broken SSH/SSL RSA/DSA keys

[PS3EPICFAIL] 12/2010: Epic Fail ECDSA on the Sony PS3

⇒ Nonce reuse, compromission of the firmware signature key

[PSANDQS] 08/2012: Mining your Ps and Qs (modulus GCD)

⇒ Compromised RSA keys on many embedded devices

[NSBTCFAIL] 01/2013: Recovering BTC private keys

⇒ Nonce reuse, crypto-wallet ECDSA key compromission

[CVE-2019-1715, RWC-2019] Cisco ASA low entropy keys

Some background on RNG fails ...

History

[CVE-2008-0166] 05/2008: predictible Debian OpenSSL RNG

⇒ Broken SSH/SSL RSA/DSA keys

[PS3EPICFAIL] 12/2010: Epic Fail ECDSA on the Sony PS3

⇒ Nonce reuse, compromission of the firmware signature key

[PSANDQS] 08/2012: Mining your Ps and Qs (modulus GCD)

⇒ Compromised RSA keys on many embedded devices

[NSBTCFAIL] 01/2013: Recovering BTC private keys

⇒ Nonce reuse, crypto-wallet ECDSA key compromission

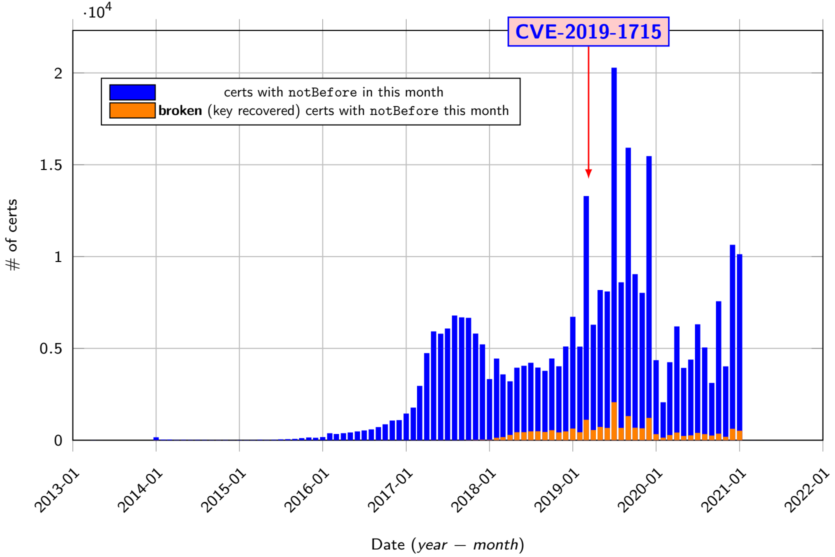
[CVE-2019-1715, RWC-2019] Cisco ASA low entropy keys 🤔

What about understanding and fixing last one for real? 😋 😇

[CVE-2023-20107] Cisco ASA low entropy keys

Distribution per month, broken / total

Over 313k certs ECDSA ASA



Cisco Adaptative Security Appliance (ASA)

- Firewall
- VPN (IPsec / TLS)
- IDS/IPS
- ...

Cisco Adaptative Security Appliance (ASA)

- Firewall
- VPN (IPsec / TLS)
- IDS/IPS
- ...

Cisco ASA 5506
40 € Livraison : à partir de 6,50 €

Hardware devices: easily available for a decent price!

Cisco Adaptative Security Appliance (ASA)

- Firewall
- VPN (IPsec / TLS)
- IDS/IPS
- ...

- Virtual appliances ASAv
- Firmware shared with HW
- Difference: no Cavium

Hardware devices: easily available for a decent price!

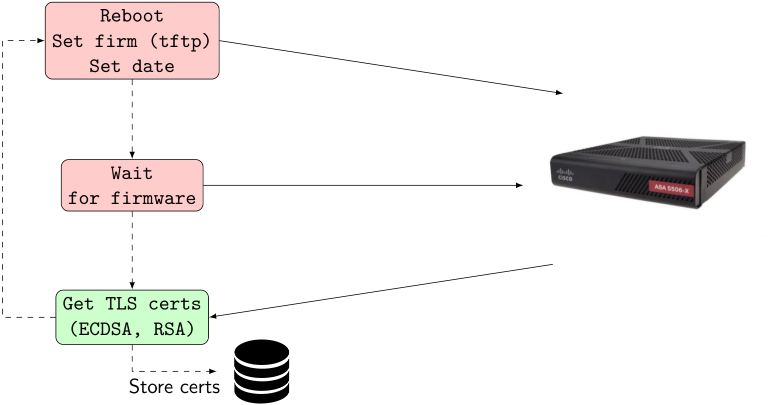
Virtual appliances ASAv images available

5506-X stats

Black box approach (through scripting)

5506-X stats

Black box approach (through scripting)



5506-X stats

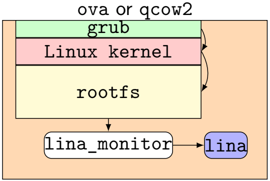
Black box approach (through scripting)

Firmware	RSA modulus	ECDSA `r` nonce	ECDSA `x` key	#generated
9.6.2-23				45
9.6.3-20				15
9.6.4-34	● (red)		● (blue)	15
9.6.4-36	● (red)		● (blue)	15
9.6.4-40	● (red)		● (blue)	15
9.6.4-41	● (red)		● (blue)	15
9.6.4-42	● (red)		● (blue)	15
9.6.4-45	● (red)		● (blue)	45
9.7.1-4				160
9.8.1				60
9.8.2	● (olive)	◗ (orange)	■ (cyan)	60
9.8.3		● (yellow)		60
9.8.4-10		● (yellow)		10
9.8.4-41		● (yellow)		30
9.9.1	● (olive)	● (yellow)	■ (cyan)	30
9.9.2-85		● (yellow)		30
9.10.1-44		◗ (crimson)		30
9.12.4				30
9.12.4-35				30
9.13.1-12				30
9.14.3-18				30
9.15.1-15				30
9.16.2-14				30
9.16.2				45

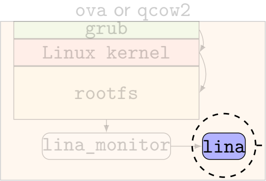
● collisions shared between firmware versions
◗ = isolated collisions
■ = collisions emerging with same certificate time

| Same color = collision values shared across versions
| Empty box = no observable collisions, inconclusive
| Versions highlighted are vulnerable and NOT concerned by CVE-2019-1715

The need for instrumentation on ASAv



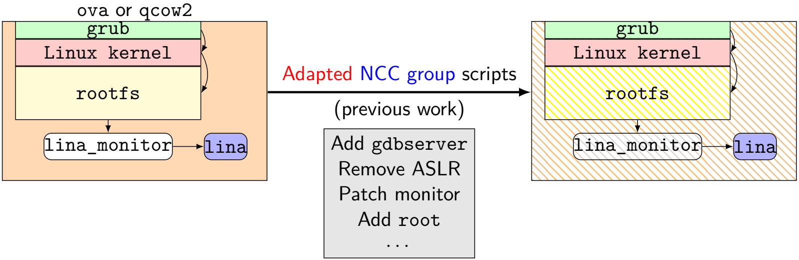
The need for instrumentation on ASAv



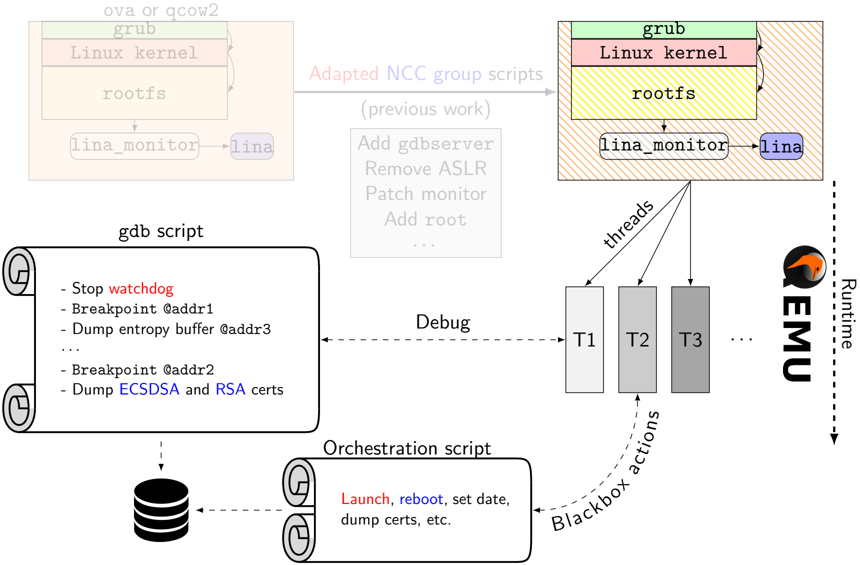
- From ≈ 100 MB in versions 9.8
  to ≈ 180 MB in recent 9.17
- More than 110k functions
- Multiple aggregated libraries (OpenSSL, etc.)
- Dedicated embedded Cavium firmware
- Redefintion of many low-level APIs (IOS era)

- Static analysis is complex
- Need for dynamic analysis and instrumentation

Instrumentation using gdb



Instrumentation using gdb

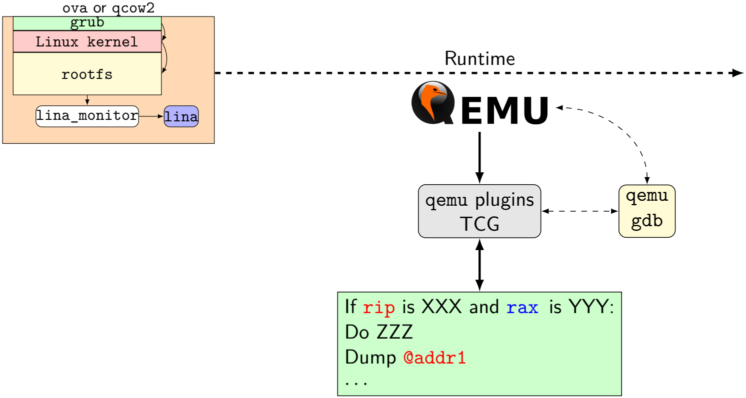


Instrumentation using qemu

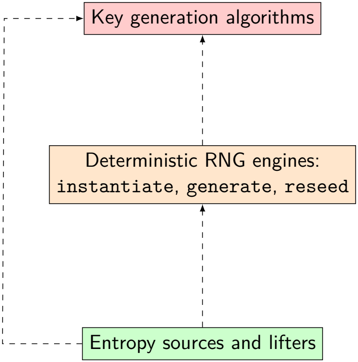
- Limitations of gdb instrumentation:
  - Multi-threading ⇒ unitialized buffers values (MD_rand)
  - No ASLR impact analysis (this is also a source of entropy)
  - Breakpoints disturb entropy based on time!

Instrumentation using qemu

- Limitations of gdb instrumentation:
  - Multi-threading ⇒ unitialized buffers values (MD_rand)
  - No ASLR impact analysis (this is also a source of entropy)
  - Breakpoints disturb entropy based on time!

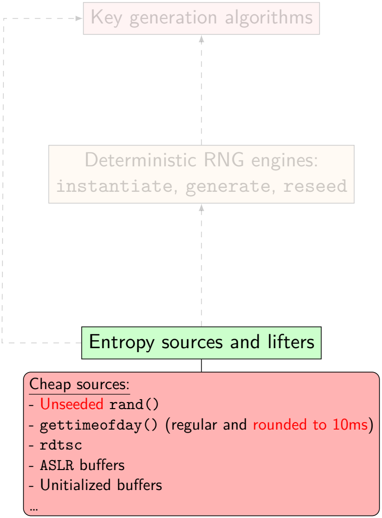


The RNG players in Cisco ASA

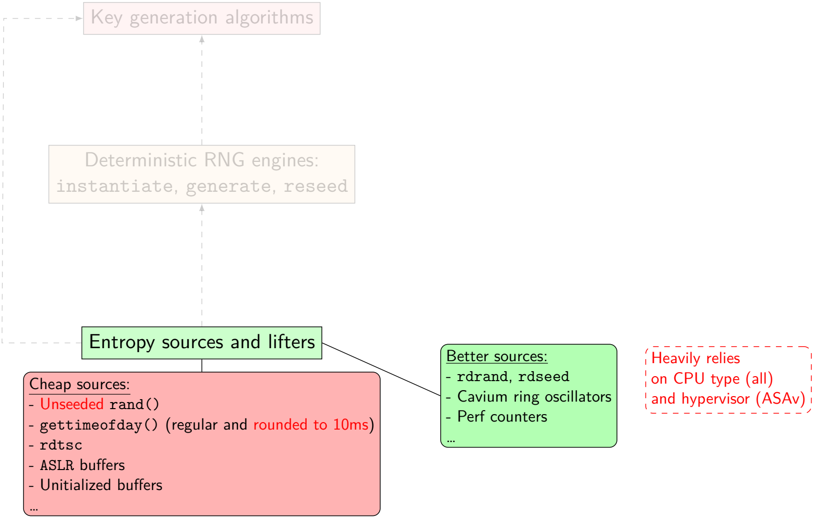


- Many primitives per layer
- Many combinations of these! (depending on ASA(v) version)
- Disclaimer: focus on important parts (not exhaustive)

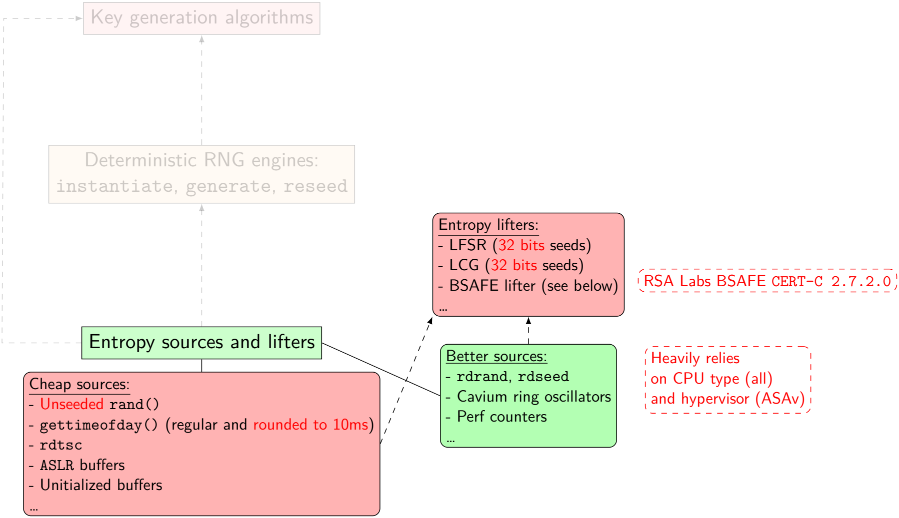
Entropy sources and lifters



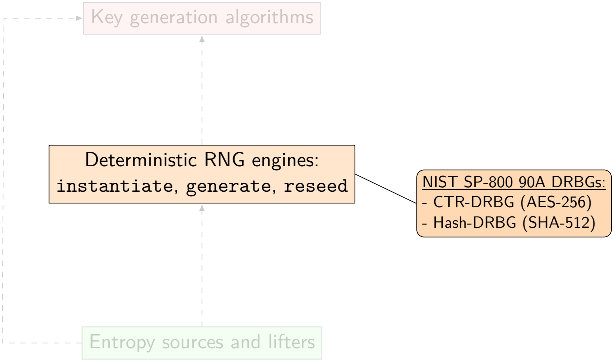
Entropy sources and lifters



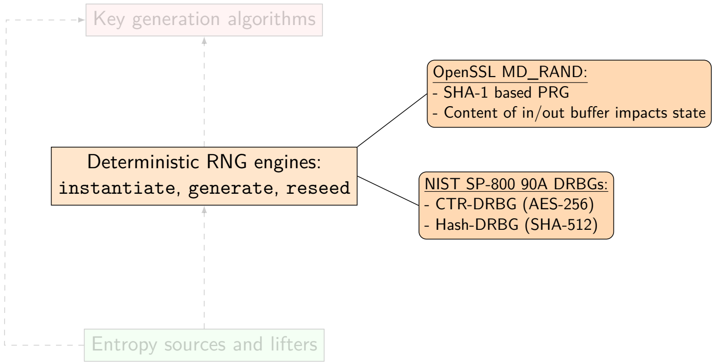
Entropy sources and lifters



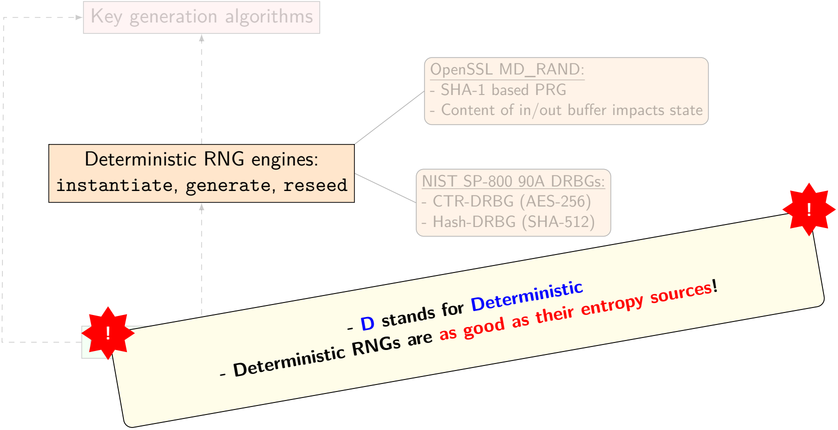
Deterministic generators



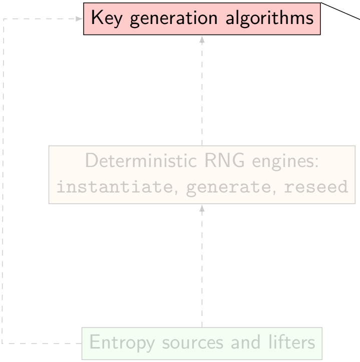
Deterministic generators



Deterministic generators

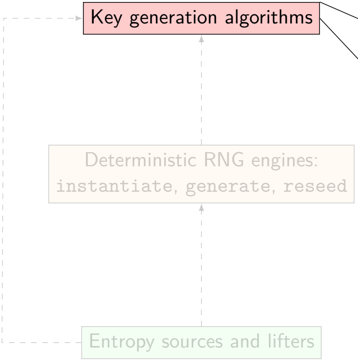


Key generation details



RSA key generation:
- FIPS 186-4 seeded generation of P and Q
- Seed of 28 bytes
- Explains why GCD attacks fail!

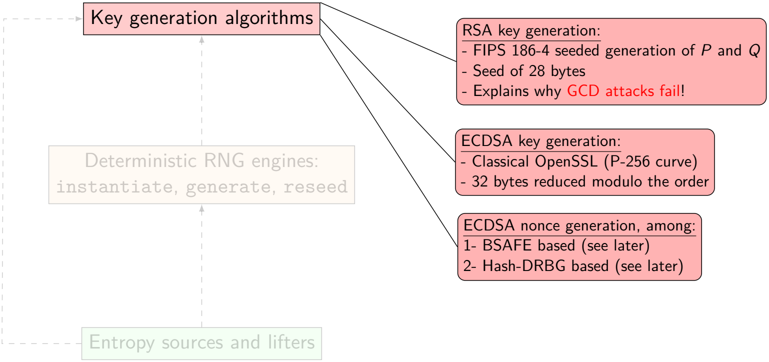
Key generation details



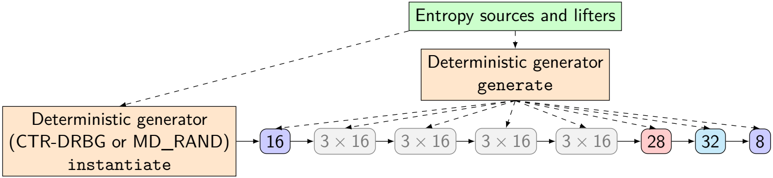
RSA key generation:
- FIPS 186-4 seeded generation of P and Q
- Seed of 28 bytes
- Explains why GCD attacks fail!

ECDSA key generation:
- Classical OpenSSL (P-256 curve)
- 32 bytes reduced modulo the order

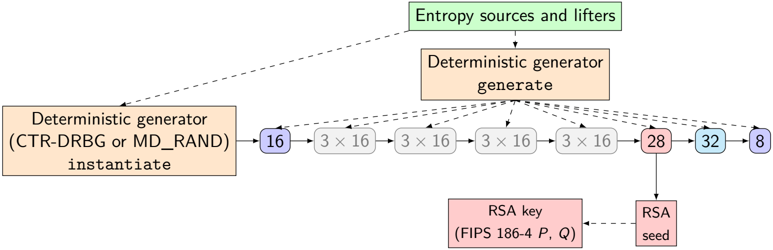
Key generation details



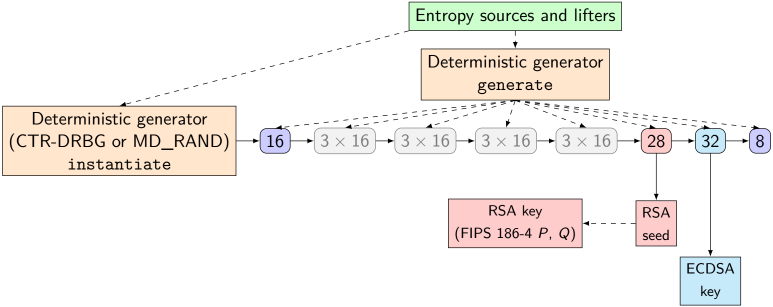
Calls to the DRBG and random generate



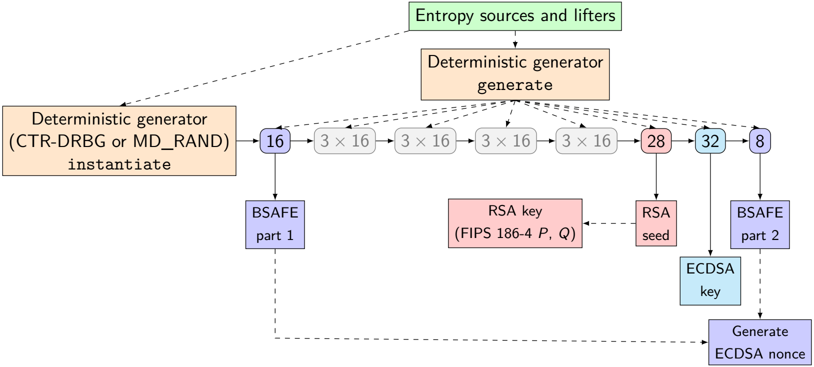
Calls to the DRBG and random generate



Calls to the DRBG and random generate

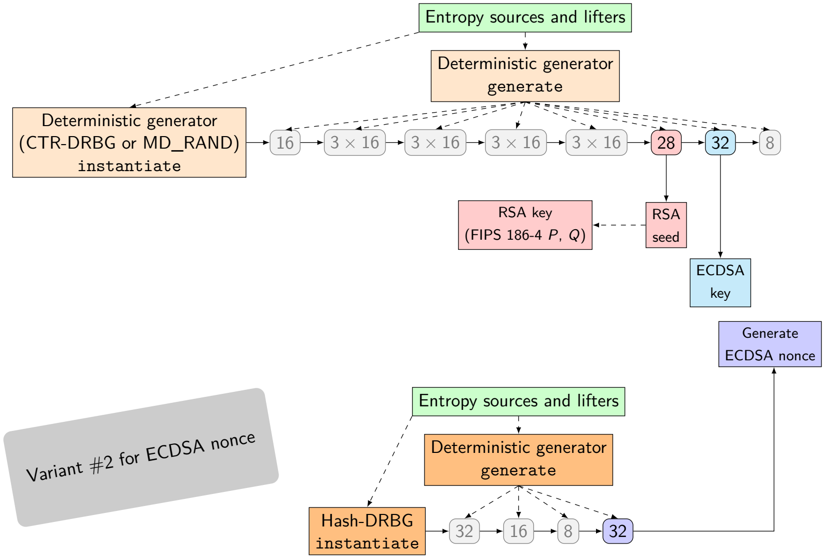


Calls to the DRBG and random generate

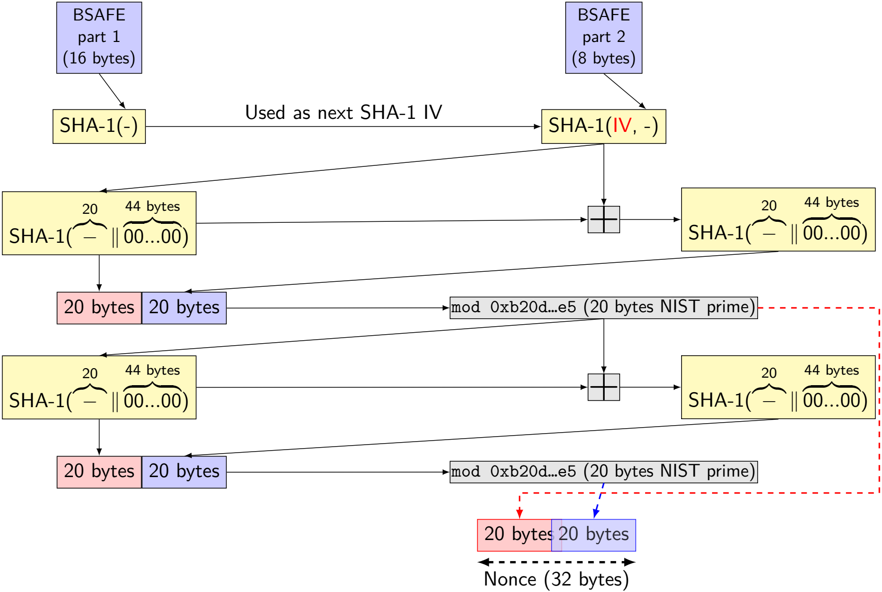


Variant #1 for ECDSA nonce

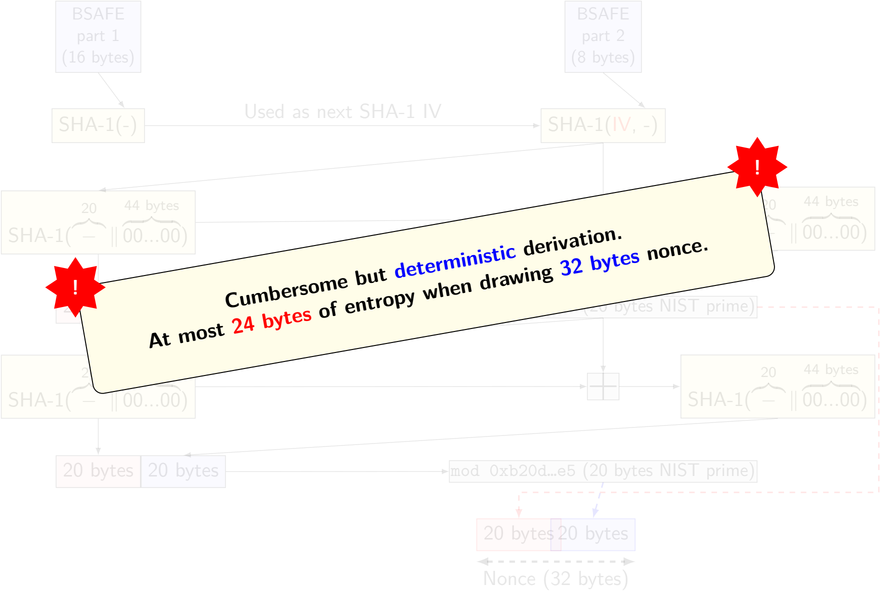
Calls to the DRBG and random generate



BSAFE lifter for ECDSA nonce



BSAFE lifter for ECDSA nonce



Cumbersome but deterministic derivation.
At most 24 bytes of entropy when drawing 32 bytes nonce.

ASAv v9.10.1.44

Overview of instantiated mechanisms

Used mechanisms

- CTR-DRBG used for RSA seed, ECDSA key
- ECDSA nonce using BSAFE with seeds from CTR-DRBG

CTR-DRBG Instantiate

- DRBG Personalization string:
  - Fixed `"CiscoSSL DRBG60"`
  - time from boot rounded to 10ms
- Entropy/nonce:
  - 40/20 bytes from MD_RAND …
  - … seeded by LFSR …
  - … seeded by 32 bits RDTSC.

CTR-DRBG Generate calls

- Addin: counter + time from boot rounded to 10ms

ASAv v9.10.1.44

Key aspects of a tricky keygenning

Estimated complexity

- 2^{32} possible LFSR seeds
- $\approx 2^{13}$ possible tuples for the 15 rounded time values

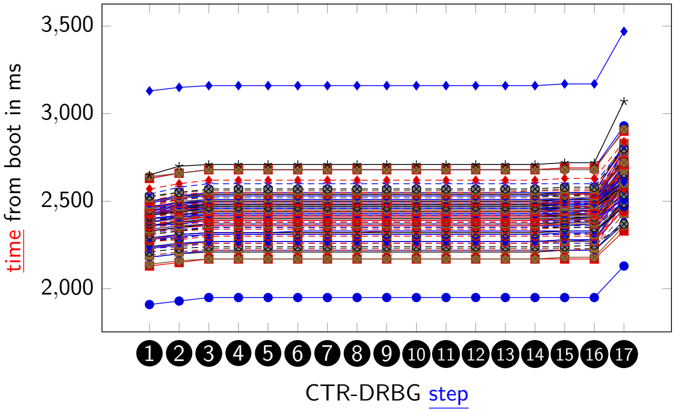
$\Rightarrow$ Exhaustive search for $\approx 2^{45}$ (w/ heavy DRBG calls)

Meet in the middle solution

- Patch the binary with a known fixed seed, do some stats on the timings as independent variables (valid approach)
- Take the most probable paths to reduce complexity, generate enough target certs and validate approach

ASAv v9.10.1.44

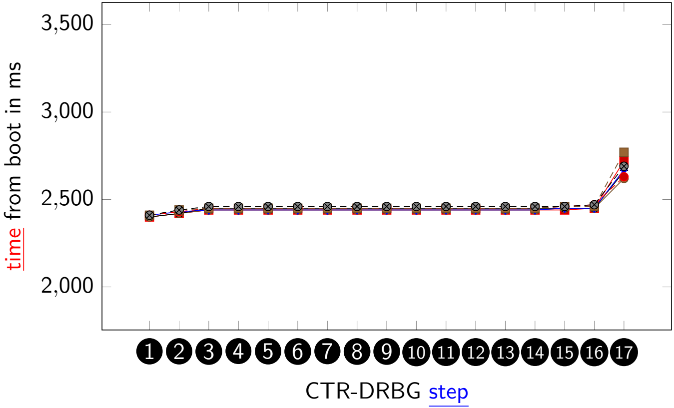
Timing statistics using patched binary (fixed seed)



- Pros: complexity reduced to $\approx 2^{13}$ for stats gathering

ASAv v9.10.1.44

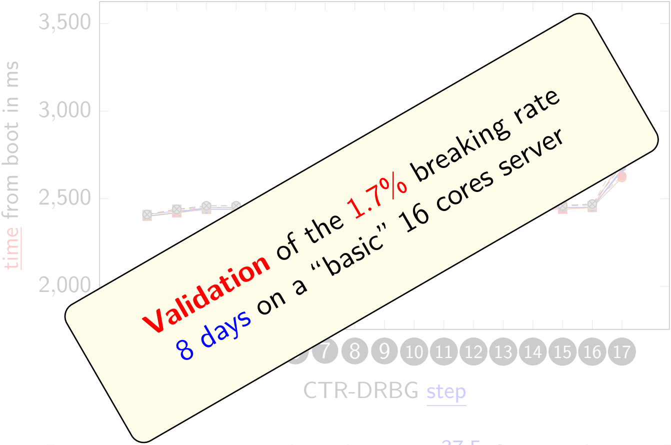
Timing statistics using patched binary (fixed seed) + envelope reduction



- Pros: complexity reduced to $\approx 2^{37.5}$ for validation PoC on **unpatched** binary by reusing these envelope stats
- Cons: only 1.7% of possible certs remains accessible

ASAv v9.10.1.44

Timing statistics using patched binary (fixed seed)



- Pros: complexity reduced to $\approx 2^{37.5}$ for validation PoC on **unpatched** binary by reusing these envelope stats
- Cons: only 1.7% of possible certs remains accessible

ASAv firmware analysis: overview of results

Firmware	RSA modulus	ECDSA nonce	ECDSA key	Comment	Keygen time complexity
ASAv9.6.4-36	◗	●	◗ ▲	HASH-DRBG seeded by LFSR seeded by 32 bits rdtsc, used for nonce. CTR-DRBG is seeded by MD_RAND, itself seeded by HASH-DRBG itself seeded by a LFSR, itself seeded by rdtsc rounded to 32 bits	2^{32} (nonce)
ASAv9.8.1		●	▲	CTR-DRBG "saved" by addin with true gettimeofday(), HASH-DRBG seeded by a LFSR itself seeded by rdtsc rounded to 32 bits	2^{32} (nonce)
ASAv9.8.2	●	●	●	MD_RAND seeded by rand(), ASLR in input buffers for MD_RAND (nonce), BSAFE seeded by MD_RAND	$\approx 2^{33}$
ASAv9.8.3	●	●	●	CTR-DRBG seeded by rand() BSAFE seeded by CTR_DRBG	$\approx 2^{16}$
ASAv9.9.1	●	●	●	MD_RAND seeded by rand(), ASLR in input buffers for MD_RAND (nonce), BSAFE seeded by MD_RAND	$\approx 2^{33}$
ASAv9.10.1-44	● (red)	● (red)	● (red)	CTR-DRBG seeded by MD_RAND seeded by LFSR seeded by 32 bits rdtsc. Bad gettimeofday is also used.	Full: $\approx 2^{45}$ PoC: $\approx 2^{37.5}$

Legend:
- ● Fully broken with a PoC **keygen**
- ● (red) Broken with a PoC **keygen** with higher time complexity
- ◗ **Fragile** entropy sources, harder to exploit (but seems feasible)
- ▲ Broken as a side effect of nonce breaking

Versions highlighted are vulnerable and NOT concerned by previous CVE-2019-1715

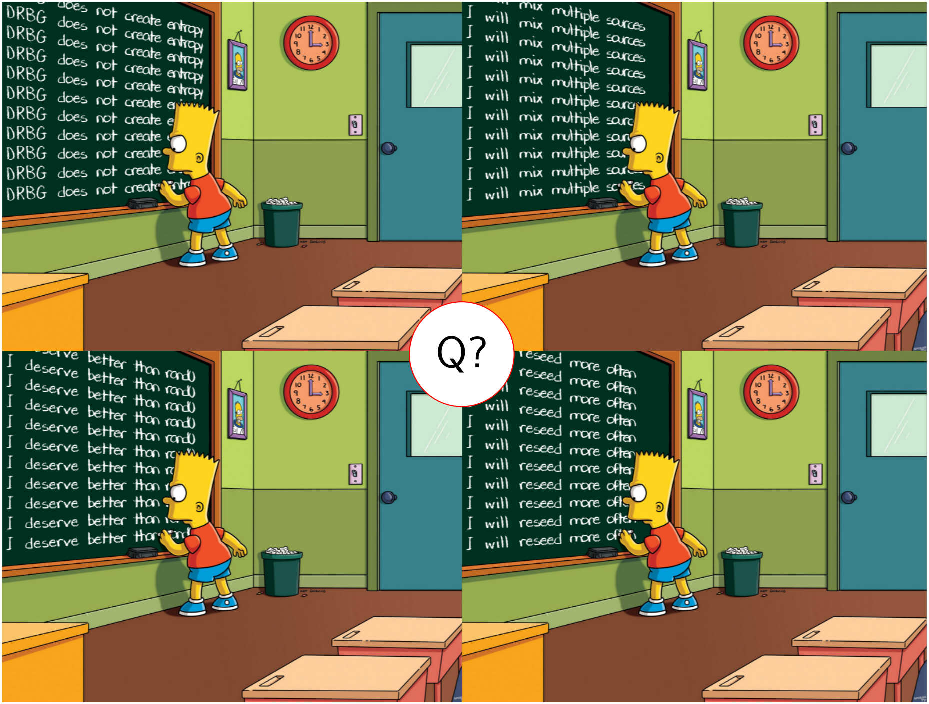
Conclusion

What we ~~learned~~ already knew.

- Fail instead of fallback to a bad entropy source
- Consider worst code path, remove if unacceptable/unsure
- Mix multiple sources instead of using a single one
- DRBG specific
  - DRBG security depends on `instantiate()` source
  - Poor addins for `DRBG generate()` calls is risky
  - Reseeding often is a requirement [DRBG-ANALYSIS]

Final thoughts

- Good looking keys, etc $\not\Rightarrow$ good random
- Good DRBG/PRNG $\not\Rightarrow$ good random
- Full 50 pages article in SSTIC proceedings


DRBG does not create entropy
I will mix multiple sources
I deserve better than rand()
I will reseed more often
Q?

- Ryad Benadjila, Arnaud Ebalard, Jean-Pierre Flori **“libecc: an ecc-based signature mechanisms library”**. Available at https://github.com/libecc/libecc.
- Arnaud Ebalard **“x509-parser: a RTE-free X.509 parser”**. Available at https://github.com/ANSSI-FR/x509-parser. More details at https://www.sstic.org/2019/presentation/journey-to-a-rte-free-x509-parser/
- Nils Schneider **“Recovering Bitcoin private keys using weak signatures from the blockchain”**, Blog entry, 28 January 2013, http://www.nilsschneider.net/2013/01/28/recovering-bitcoin-private-keys.html, broken link use https://archive.org.
- Nadia Heninger and Zakir Durumeric and Eric Wustrow and J. Alex Halderman **“Mining Your Ps and Qs: Detection of Widespread Weak Keys in Network Devices”**, https://www.usenix.org/system/files/conference/usenixsecurity12/sec12-final228.pdf.

- Luciano Bello, "**DSA-1571-1 openssl – predictable random number generator**" available at https://www.debian.org/security/2008/dsa-1571.
- failoverflow, https://web.archive.org/web/20150627235425/https://events.ccc.de/congress/2010/Fahrplan/attachments/1780_27c3_console_hacking_2010.pdf, 29 December 2010
- Cisco Adaptive Security Appliance Software and Firepower Threat Defense Software Low-Entropy Keys Vulnerability, https://tools.cisco.com/security/center/content/CiscoSecurityAdvisory/cisco-sa-20190501-asa-ftd-entropy, May 2019
- Joanne Woodage and Dan Shumow **"An Analysis of the NIST SP 800-90A Standard"**, https://eprint.iacr.org/2018/349.pdf, 2018.

- Greg Zaverucha and Dan Shumow **“Are Certificate Thumbprints Unique?”**, https://eprint.iacr.org/2019/130.pdf, 2019
- Cisco Adaptive Security Appliance Software and Firepower Threat Defense Software Low-Entropy Keys Vulnerability, https://sec.cloudapps.cisco.com/security/center/content/CiscoSecurityAdvisory/cisco-sa-asa5500x-entropy-6v9bHVYP, March 2023