

THE UNIVERSITY OF
MELBOURNE

No Spook Leaves Randomness to Chance

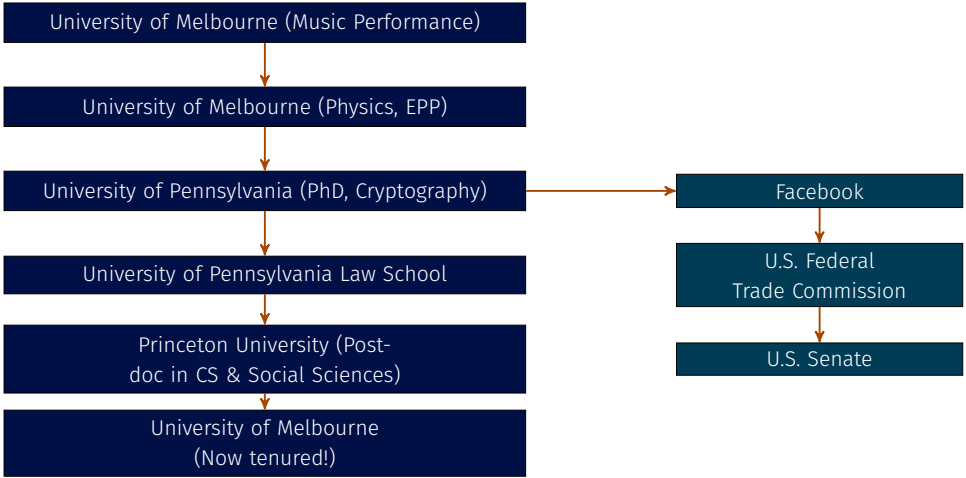
Shaanan Cohney

August 8, 2025

Acknowledgements

Stephen Checkoway, Jacob Maskiewicz, Christina Garman, Joshua Fried, Matthew Green, Nadia Heninger, Ralf-Philipp Weinmann, Eric Rescorla, Hovav Shacham, David Adrian, Antonio Sanso, Joshua Fried, Marcella Hastings, J Alex Halderman, Andrew Kwong, Shahar Paz, Daniel Genkin, Nadia Heninger, Eyal Ronen, Yuval Yarom

Me


University of Melbourne (Music Performance)
University of Melbourne (Physics, EPP)
University of Pennsylvania (PhD, Cryptography)
University of Pennsylvania Law School
Princeton University (Post-doc in CS & Social Sciences)
University of Melbourne (Now tenured!)
Facebook
U.S. Federal Trade Commission
U.S. Senate

How does an NSA-level adversary break cryptography?

- Brute-force?
- Super math?
- Clever implants and malware?
- Interception post decryption?
- AI mumbo jumbo?

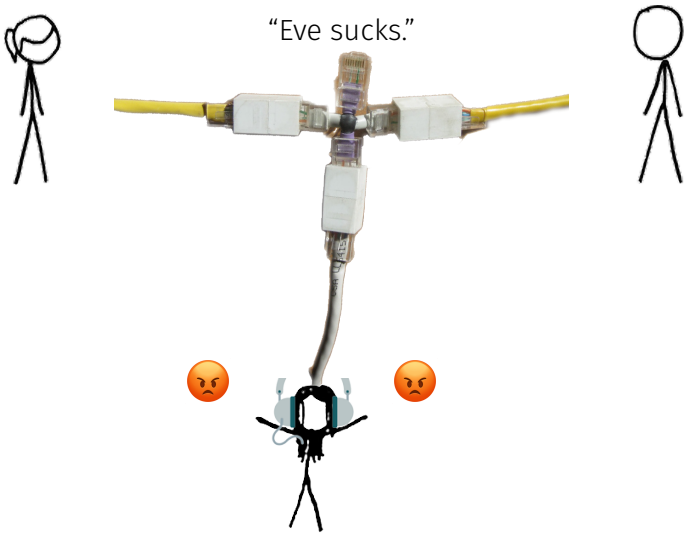
Probably a mix of these, but crypto has gotten steadily better.

This talk adds some politics to the story.

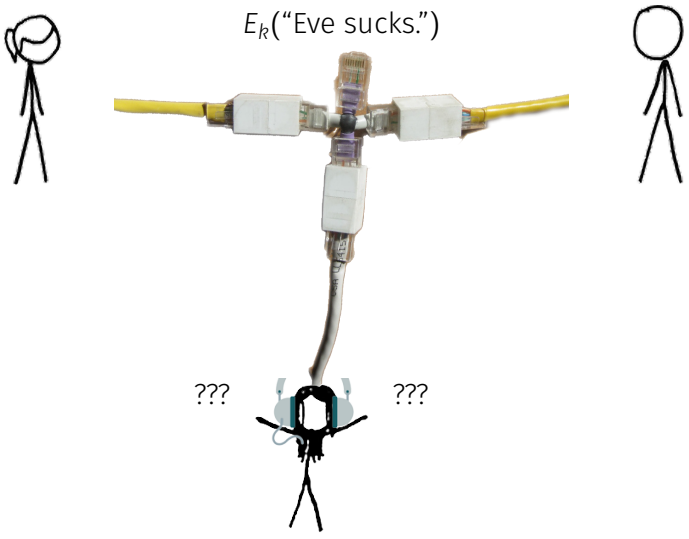
A Network



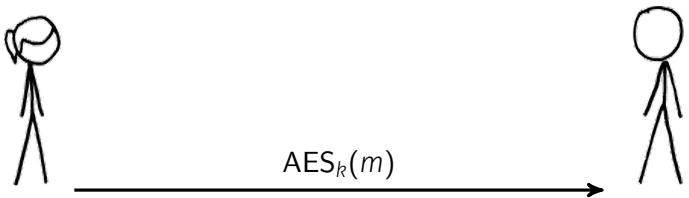
A Network



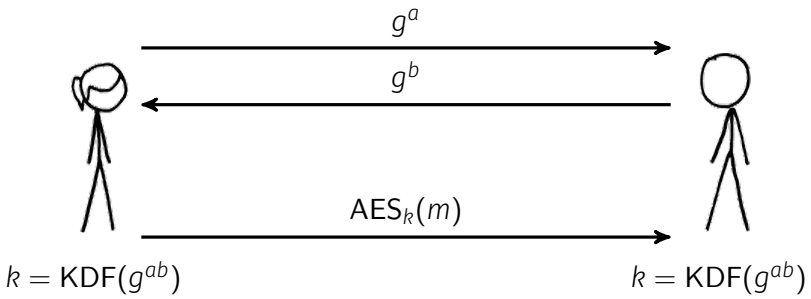
A Network with Cryptography



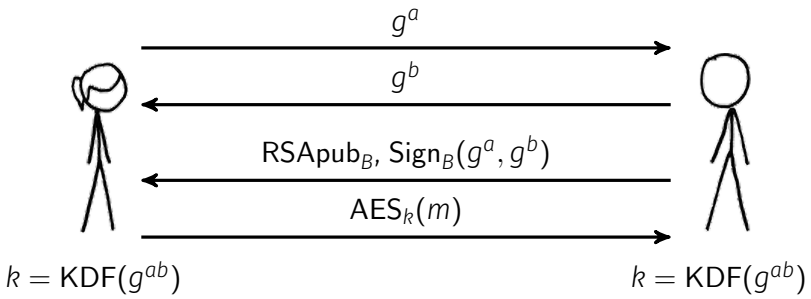
A cartoon cryptographic communication protocol



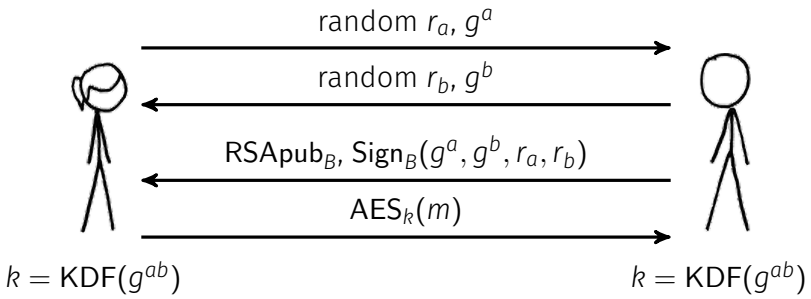
A cartoon cryptographic communication protocol



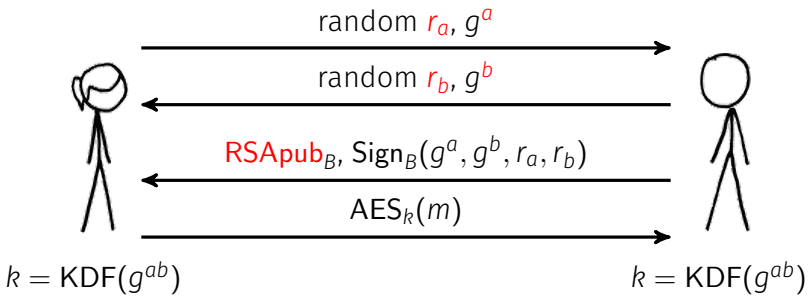
A cartoon cryptographic communication protocol



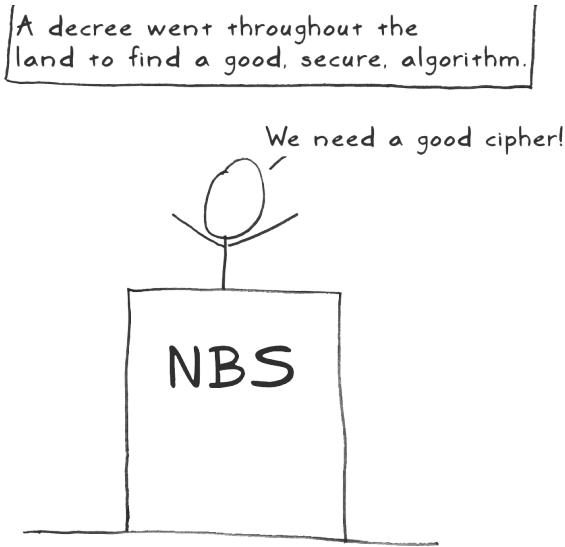
A cartoon cryptographic communication protocol



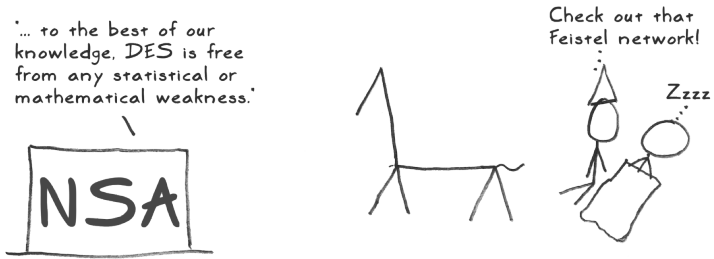
A cartoon cryptographic communication protocol



If the Eve predicts PRG output, she breaks the protocol


A decree went throughout the land to find a good, secure, algorithm.
We need a good cipher!
NBS

DES ruled in the land for over 20 years. Academics studied him intently. For the first time, there was something specific to look at. The modern field of cryptography was born.



A few decades passed...

How can we help you?

Provide Exploitation Support

- Provide VPN vulnerability analysis
- Engage Network Security Products, TAO, ESO, etc
- Convey meaningful feedback to customer
- Develop sustained exploitation threads when possible
- Suggest alternative approaches if passive exploitation is unrealistic
- DECRYPTS, DECRYPTS, DECRYPTS!!!!!!

Happy Dance!!



Ok, so what is the public crypto community missing?

Ok, so what is the public crypto community missing?

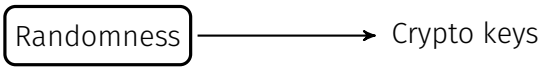
Was there a big algorithmic breakthrough?

Ok, so what is the public crypto community missing?

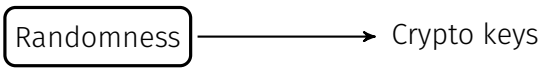
Was there a big algorithmic breakthrough?

Or is it something both more mundane and more sinister?

Cryptographic Secret Sauce



Cryptographic Secret Sauce

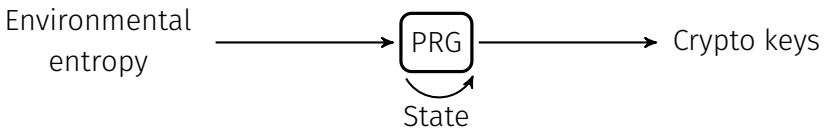


Not all cryptosystems require randomness, but it's a key ingredient to many!

True Randomness Generators (TRNGs)

- Obtained from background radiation, fluctuations in electrical circuitry, quantum mechanical effects
- Most sources are low bandwidth
- Low adoption until recently

- Cryptographic security requires:
  - Good source of entropy
  - Cryptographically secure Pseudorandom Generator (PRG) to whiten output



Idea: Study pseudorandom generators to find source of NSA decrypts!

Standardized PRGs

Standards and Certification

- Federal Information Processing Standards (FIPS) 140-x
  - latest is now 140-3, we studied 140-2
- Cryptographic Module Validation Program (CMVP)
  - 140-2 certified products valid until 2026!
- Allows vendors to serve government contracts

Use of FIPS PRGs

- Incorporated into products/libraries
- Must be used for all system functions
- Sole source of randomness for cryptographic protocols

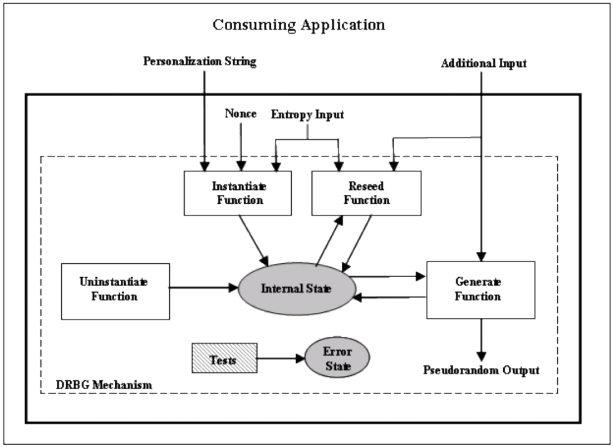
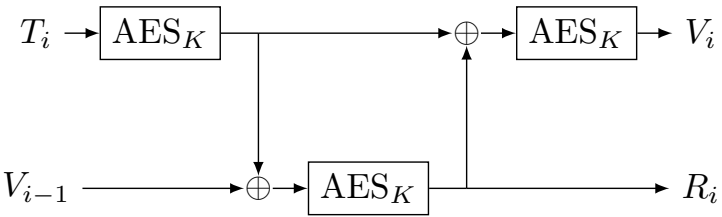


Figure 1: DRBG Functional Model

Case Study 1: ANSI X9.31

- On each iteration, mixes state V_{i-1} with timestamp T_i.
- Produces output block R_i and new state V_i.
- Uses block cipher as a mixing function.



ANSI X9.31 PRG History

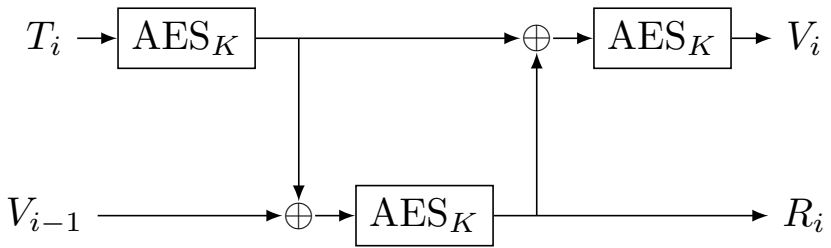
- 1985: DES-based PRG standardized in ANSI X9.17
- 1992: Adopted as a FIPS standard
- 1994: Included on list of approved PRGs in FIPS 140-1
- 1998: Variant using 3DES standardized in ANSI X9.31
- 1998: Kelsey et al.: state recovery if key known

ANSI X9.31 PRG History

- 1985: DES-based PRG standardized in ANSI X9.17
- 1992: Adopted as a FIPS standard
- 1994: Included on list of approved PRGs in FIPS 140-1
- 1998: Variant using 3DES standardized in ANSI X9.31
- 1998: Kelsey et al.: state recovery if key known
- 2004: ANSI X9.31 PRG included in FIPS 186-2
- 2005: AES-based variant published by NIST and included on FIPS 140-2 approved PRGs
- 2011: FIPS deprecates ANSI X9.31 design
- 2016: ANSI X9.31 PRG removed from FIPS 140-2

X9.31 state recovery from a known key

If key K used with block cipher is known, can recover state from output by brute forcing timestamp.



NIST ANSI X9.31 PRG standardization failure

”For AES 128-bit key, let *K be a 128 bit key.”

”This *K is reserved only for the generation of pseudo random numbers.”

NIST ANSI X9.31 PRG standardization failure

”For AES 128-bit key, let *K be a 128 bit key.”

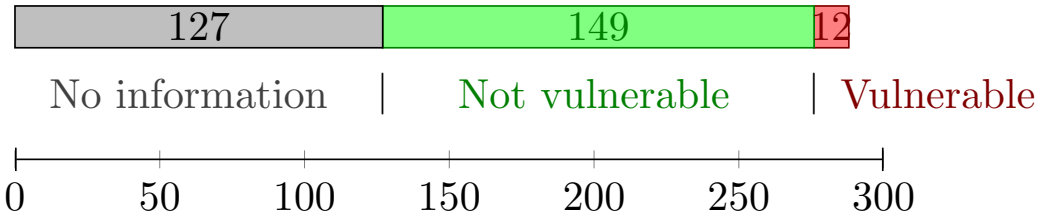
”This *K is reserved only for the generation of pseudo random numbers.”

- Standard did *not* specify key should not be hard-coded.

Using FIPS 140 to find broken implementations [CGH18]

- FIPS 140 requires vendors to document key generation and storage policies in detail.
- Searched FIPS security policies to find documented hard-coded X9.31 keys.

- FIPS 140 requires vendors to document key generation and storage policies in detail.
- Searched FIPS security policies to find documented hard-coded X9.31 keys.



"Compiled into binary" "statically stored in the code" "Hard Coded" "generated external to the module" "Stored in flash" "Static key, Stored in the firmware" "Entered in factory" "loaded at factory" "Static" "Embedded in FLASH" "Injected During Manufacture" "Hard-coded in the module"

Validating Findings

- We couldn’t obtain most devices

Validating Findings

- We couldn't obtain most devices
- One manufacturer had lost access to the source code

Validating Findings

- We couldn't obtain most devices
- One manufacturer had lost access to the source code
- We *did* obtain a copy of FortiOSv4 and a compatible device

```c
int initialize_X931()
{
  char rng_state[16];
  char timestamp_buffer[16];
  int aes_key[4];
  int result = key_set;
  aes_key[0] = 0x6D66B1F3;
  aes_key[1] = 0x42726013;
  aes_key[2] = 0xAB1C06ED;
  aes_key[3] = 0x0262D4B8;
  if ( !key_set ) result = set_aeskey(aes_key);
  if ( !state_set )
  {
    /* removed for clarity */
    save_state(rng_state);
    fill_timestamp(timestamp_buffer);
    result = x931(&timestamp_buffer, output_buffer,
        rng_state, 16);
  }
  return result;
```

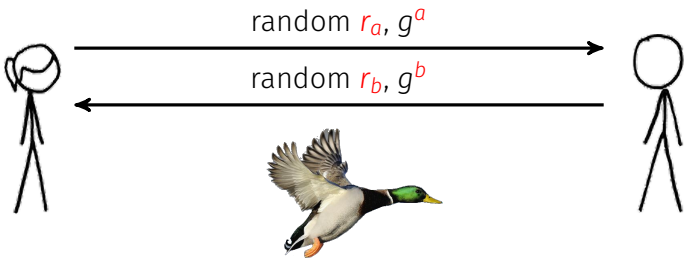
B.1.9 ANSI931_AES128VST.req

```
#  CAVS 4.3
#  "ANSI X9.31" information for "DEMO PRODUCT"
#  Generated on Tue Jan 11 12:11:44 2005

[X9.31]
[AES 128-Key]

COUNT = 0
Key = f3b1666d13607242ed061cabb8d46202
DT = e6b3be782a23fa62d71d4afbb0e922f9
V = 80000000000000000000000000000000
```

Passive PRG state recovery in the IPsec protocol



- Need raw PRG outputs for state recovery attack.
- Idea: Use the random nonces.
- After state recovered, then recover secret exponents.

- FortiOS v4 hard-coded NIST test vector key

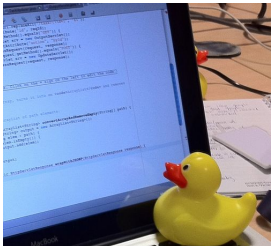
Passive decryption for Fortigate IPsec VPNs

- FortiOS v4 hard-coded NIST test vector key
- 2^{25} work brute-forcing timestamps for state recovery

Passive decryption for Fortigate IPsec VPNs

- FortiOS v4 hard-coded NIST test vector key
- 2^{25} work brute-forcing timestamps for state recovery
- Performed internet-wide scans and successfully recovered private keys against hosts in the wild.

- FortiOS v4 hard-coded NIST test vector key
- 2^{25} work brute-forcing timestamps for state recovery
- Performed internet-wide scans and successfully recovered private keys against hosts in the wild.
- ANSI X9.31 PRG no longer included in FortiOS v5; FortiOS v4 patched since November 2016



Case Study: CTR_DRBG

CTR_DRBG

- Standardized in SP 800-90, FIPS 140-2

CTR_DRBG

- Standardized in SP 800-90, FIPS 140-2
- Successor design to X9.31

CTR_DRBG

- Standardized in SP 800-90, FIPS 140-2
- Successor design to X9.31
- Changes:

CTR_DRBG

- Standardized in SP 800-90, FIPS 140-2
- Successor design to X9.31

- Changes:
  - Encrypts incrementing counter

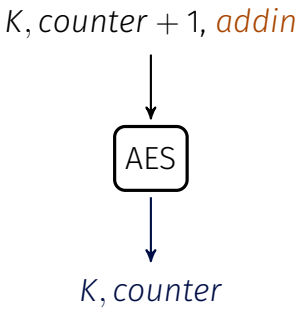
CTR_DRBG

- Standardized in SP 800-90, FIPS 140-2
- Successor design to X9.31

- Changes:
  - Encrypts incrementing counter
  - Adds support for *additional entropy*

Three Stage Process:

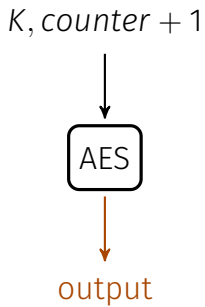
1. Update State & Add Entropy



CTR_DRBG: Generate Function

Three Stage Process:

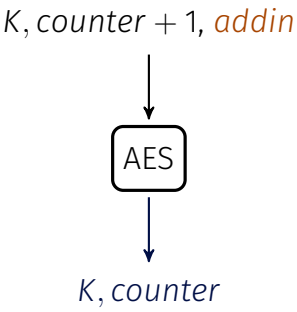
1. Update State & Add Entropy
2. Produce Output



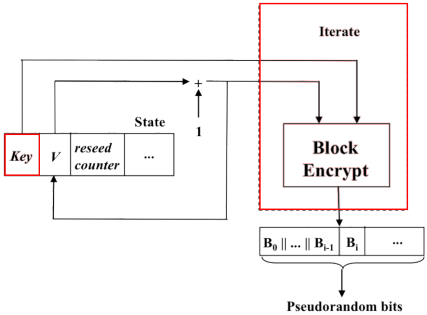
CTR_DRBG: Generate Function

Three Stage Process:

1. Update State & Add Entropy
2. Produce Output
3. Update State & Add Entropy



Key Rotation Flaw



Problem 1: Key is not rotated until after encryptions are done.

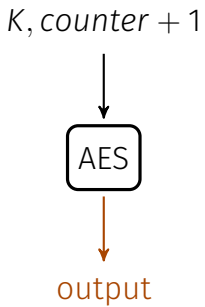
Problem 2: Additional entropy is optional and implementer chosen.

Problem 1: Key Not Rotated Often Enough

Attacker may be able to compromise K using a side channel attack

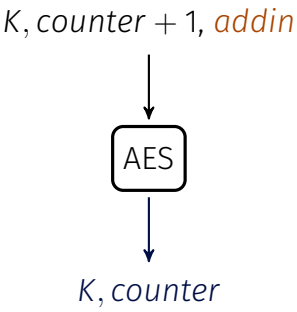
An attacker then decrypts PRG output to learn state:

$$D_K(\text{output}) = \text{counter} + 1$$



Problem 2: Lack of Entropy

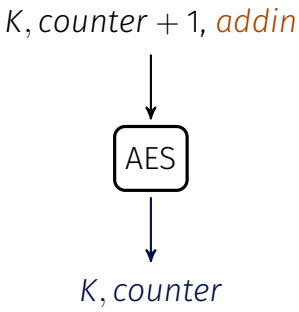
Once attacker has $(K, counter)$, guesses *addin* and calculates updated state



Problem 2: Lack of Entropy

Once attacker has $(K, counter)$, guesses *addin* and calculates updated state

If *addin* is used at all!



Idea: Find PRGs with leaky AES implementations and magnify leakage over many encryptions

Idea: Find PRGs with leaky AES implementations and magnify leakage over many encryptions

Systematically reviewed CMVP certified candidates.

Idea: Find PRGs with leaky AES implementations and magnify leakage over many encryptions

Systematically reviewed CMVP certified candidates.

FortiOS was again the best candidate for analysis. This time version 5.

Targeted Implementations

Reverse engineered FortiOSv5 code, and varied parameters to simulate:

- OpenSSL FIPS Module
- NetBSD
- FortiOS

State Recovery

- Used FLUSH+RELOAD attack on T-Table AES

State Recovery

- Used FLUSH+RELOAD attack on T-Table AES
- Recovered long-term client ECDSA private keys

State Recovery

- Used FLUSH+RELOAD attack on T-Table AES
- Recovered long-term client ECDSA private keys
- But... this attack probably wasn't useful for the NSA?

Case Study: Dual EC DRBG

Dual Elliptic Curve PRG

- 2004 – Draft Proposal

Dual Elliptic Curve PRG

- 2004 – Draft Proposal
- 2006 (Early) – Distinguishing attack (Schoenmakers and Sidorenko)

Dual Elliptic Curve PRG

- 2004 – Draft Proposal
- 2006 (Early) – Distinguishing attack (Schoenmakers and Sidorenko)
- 2006 (Late) – Published in NIST SP 800-90A

Dual Elliptic Curve PRG

- 2004 – Draft Proposal
- 2006 (Early) – Distinguishing attack (Schoenmakers and Sidorenko)
- 2006 (Late) – Published in NIST SP 800-90A
- 2007 – Shumow and Ferguson demonstrate backdoor usage

Dual Elliptic Curve PRG

- 2004 – Draft Proposal
- 2006 (Early) – Distinguishing attack (Schoenmakers and Sidorenko)
- 2006 (Late) – Published in NIST SP 800-90A
- 2007 – Shumow and Ferguson demonstrate backdoor usage
- 2013 – NSA and other US officials identified as key designers of the standard, Snowden reveals project 'Bullrun'

Dual Elliptic Curve PRG

- 2004 – Draft Proposal
- 2006 (Early) – Distinguishing attack (Schoenmakers and Sidorenko)
- 2006 (Late) – Published in NIST SP 800-90A
- 2007 – Shumow and Ferguson demonstrate backdoor usage
- 2013 – NSA and other US officials identified as key designers of the standard, Snowden reveals project 'Bullrun'
- 2014 – NIST removes Dual EC from standard

Dual Elliptic Curve PRG

- 2004 – Draft Proposal
- 2006 (Early) – Distinguishing attack (Schoenmakers and Sidorenko)
- 2006 (Late) – Published in NIST SP 800-90A
- 2007 – Shumow and Ferguson demonstrate backdoor usage
- 2013 – NSA and other US officials identified as key designers of the standard, Snowden reveals project 'Bullrun'
- 2014 – NIST removes Dual EC from standard
- 2015 – Juniper Announces unauthorized code modifications

Dual Elliptic Curve PRG

- Requires specification of a curve and two points on it

Dual Elliptic Curve PRG

- Requires specification of a curve and two points on it
- NIST standard sets the curve as P-256, and defines values for P,Q

Dual Elliptic Curve PRG

- Requires specification of a curve and two points on it
- NIST standard sets the curve as P-256, and defines values for P,Q
- Juniper used the same curve, but different points

September 2013: NSA Bullrun in NY Times

- (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.



the grugq
@thegrugq

Follow

Woah! Juniper discovers a backdoor to decrypt VPN traffic (and remote admin) has been inserted into their OS source

Important Announcement about ScreenOS®

IMPORTANT JUNIPER SECURITY ANNOUNCEMENT CUSTOMER UPDATE: DECEMBER 20, 2015 Administrative Access (CVE-2015-7755) only affects ScreenOS 6.3.0r17 through

forums.juniper.net

Diff of VPN code change [CMGF+16]

P-256 Weierstraß *b*

```
5AC635D8AA3A93E7B3EBBD55 D06B0CC53B0F63BCE3C3E27D2604B
6B17D1F2E12C4247F8BCE6E563A440F277037D812DEB33 98C296
FFFFFFFF00000000FFFFFFFFFFFFFFFFBCE6FAADA7179E84F3B9CAC2FC632551
bad: 9585320EEAF81044F20D55030A035B11BECE81C785E6C933E4A8A131F6578107
good:2c55e5e45edf713dc43475effe8813a60326a64d9ba3d2e39cb639b0f3b0ad10
nist:c97445f45cdef9f0d3e05e1e585fc297235b82b5be8ff3efca67c59852018192
```

P-256 *P* x coord

P-256 field order

Reverse engineering shows changed values are **x-coordinates** for the Dual EC point *Q*

Juniper cascaded Dual EC with ANSI X9.31

- ScreenOS only certified for ANSI X9.31, not Dual EC

Juniper cascaded Dual EC with ANSI X9.31

- ScreenOS only certified for ANSI X9.31, not Dual EC
- Juniper used non-default points for Dual EC

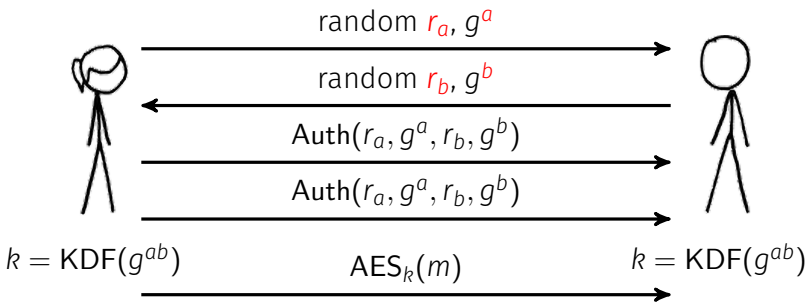
Juniper cascaded Dual EC with ANSI X9.31

- ScreenOS only certified for ANSI X9.31, not Dual EC
- Juniper used non-default points for Dual EC

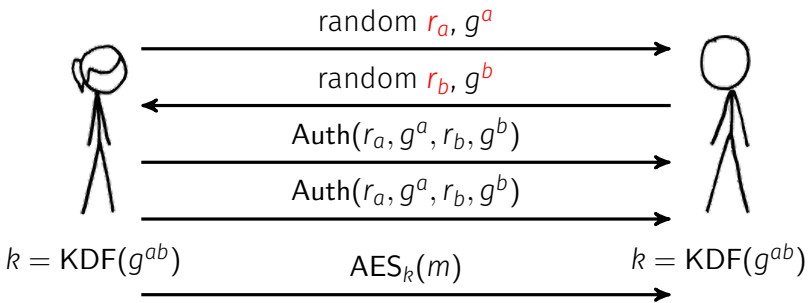
The following product families do utilize Dual_EC_DRBG, but do not use the pre-defined points cited by NIST:

1. ScreenOS*

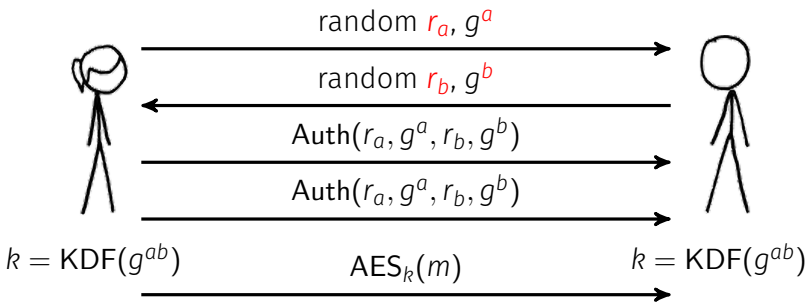
* ScreenOS does make use of the Dual_EC_DRBG standard, but is designed to not use Dual_EC_DRBG as its primary random number generator. ScreenOS uses it in a way that should not be vulnerable to the possible issue that has been brought to light. Instead of using the NIST recommended curve points it uses self-generated basis points and then takes the output as an input to FIPS/ANSI X.9.31 PRNG, which is the random number generator used in ScreenOS cryptographic operations.



- Use random nonces to carry out state recovery attack.



- Use random nonces to carry out state recovery attack.
- ScreenOS used 32-byte nonce $\implies$ efficient attack.



- Use random nonces to carry out state recovery attack.
- ScreenOS used 32-byte nonce $\implies$ efficient attack.
- After state recovered, then recover secret exponents.

ScreenOS Version History

ScreenOS 6.1.0r7

- ANSI X9.31
- Seeded by interrupts
- Reseed every 10k calls
- 20-byte IKE nonces

ScreenOS Version History

ScreenOS 6.1.0r7

- ANSI X9.31
- Seeded by interrupts
- Reseed every 10k calls
- 20-byte IKE nonces

ScreenOS 6.2.0r0 (2008)

- Dual EC → ANSI X9.31
- Reseed bug exposes raw Dual EC
- Reseed every call
- Nonces generated before keys
- 32-byte IKE nonces

ScreenOS Version History

ScreenOS 6.1.0r7

- ANSI X9.31
- Seeded by interrupts
- Reseed every 10k calls
- 20-byte IKE nonces

ScreenOS 6.2.0r0 (2008)

- Dual EC → ANSI X9.31
- Reseed bug exposes raw Dual EC
- Reseed every call
- Nonces generated before keys
- 32-byte IKE nonces

- Attacker changed constant in 6.2.0r15 (2012).
- But passive decryption enabled in earlier release.
- Juniper's "fix" was to reinstate original Q value. After our work they removed Dual EC completely.

- Demonstrated full passive VPN decryption against ScreenOS 6.2

- Demonstrated full passive VPN decryption against ScreenOS 6.2
- Only possible because of implementation flaws *and* standards failure

- Demonstrated full passive VPN decryption against ScreenOS 6.2
- Only possible because of implementation flaws *and* standards failure
- Developed scanning techniques to find $\approx$ 3000 devices… so was this actually useful for the NSA?



Technology | Cybersecurity

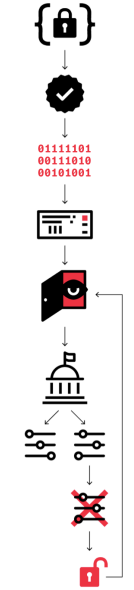
Juniper Breach Mystery Starts to Clear With New Details on Hackers and U.S. Role

Tech company installed a flawed NSA algorithm that became ‘a perfect example of the danger of government backdoors.’

By Jordan Robertson

2 September 2021 at 11:00 pm AEST

The Pentagon and an Alleged NSA Backdoor



The NSA introduces an algorithm called **Dual Elliptic Curve**...

...which it urges the **National Institute of Standards and Technology** to approve.

The Pentagon—NSA's parent agency—insists that **Juniper Networks** use the algorithm as a condition for some contracts. Juniper adds it to **NetScreen**'s ScreenOS software.

This introduces an alleged backdoor that could be used to spy on select...

...NetScreen **customers**, which include telecommunications providers and government agencies.

Juniper includes two tweaks that it says neutralize the vulnerability.

But one tweak would later be found to have failed, exposing customers to the alleged NSA backdoor.

Sources: People familiar with the matter, company and

APT 5 Hijacks Alleged NSA Backdoor in 2012



APT 5—Suspected Chinese state-sponsored hackers—break into...

...**Juniper Networks** and alter NetScreen's ScreenOS software...

...by replacing the **"Q value,"** a large number in the algorithm used to help create encryption keys.

```
2c55e5e45edf713d
c43475effe8813a6
0326a64d9ba3d2e3
9cb639b0f3b0ad10
```

```
9585320EEAF81044
F20D55030A035B11
BECE81C785E6C933
E4A8A131F6578107
```

This hijacks the alleged backdoor in the NSA algorithm, enabling the hackers to...

...potentially decode the encrypted communications of NetScreen **customers**.

Sources: People familiar with the matter, company records, researchers



Summer 2018



Summer 2018

In a July 2020 response to Wyden and other members of Congress, Juniper provided few new details of the case but blamed the intrusions on a “sophisticated nation-state hacking unit.” NSA told Wyden’s staff in 2018 that there was a “lessons learned” report, but the agency “now asserts that it cannot locate this document,” according to a Wyden aide. Reuters previously reported NSA’s claim that the document had been lost.

Big Picture: Standards and certification are targets

Adversarial actions that:

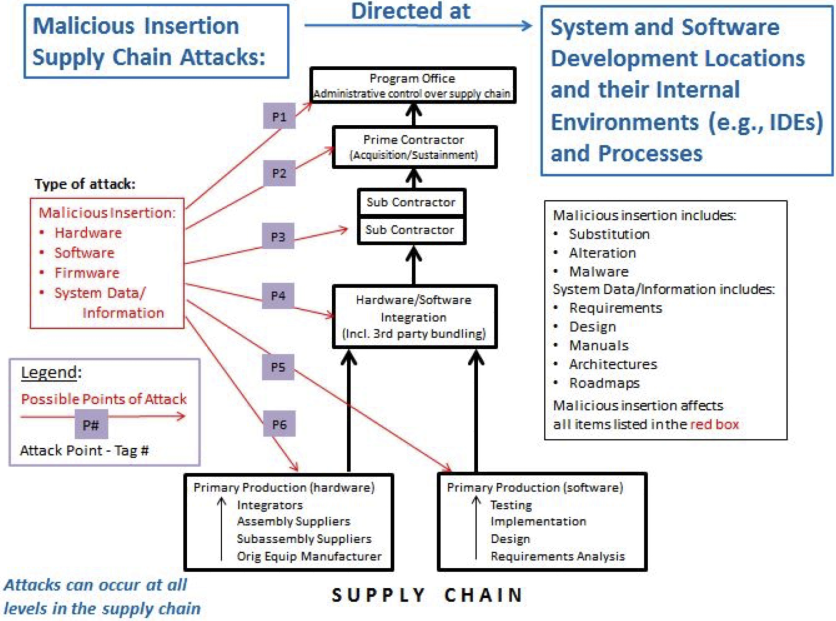
1. Are surreptitious

Adversarial actions that:

1. Are surreptitious
2. Leverage broader procurement and deployment ecosystem to carry out the attack

Adversarial actions that:

1. Are surreptitious
2. Leverage broader procurement and deployment ecosystem to carry out the attack
3. Are outside the immediate control of the administrator of the resource


Malicious Insertion Supply Chain Attacks:
Directed at
System and Software Development Locations and their Internal Environments (e.g., IDEs) and Processes
Program Office
Administrative control over supply chain
Prime Contractor
(Acquisition/Sustainment)
Sub Contractor
Sub Contractor
Hardware/Software Integration
(Incl. 3rd party bundling)
Type of attack:
Malicious Insertion:
• Hardware
• Software
• Firmware
• System Data/ Information
P1
P2
P3
P4
P5
P6
Legend:
Possible Points of Attack
P#
Attack Point - Tag #
Malicious insertion includes:
• Substitution
• Alteration
• Malware
System Data/Information includes:
• Requirements
• Design
• Manuals
• Architectures
• Roadmaps
Malicious insertion affects all items listed in the red box
Primary Production (hardware)
Integrators
Assembly Suppliers
Subassembly Suppliers
Orig Equip Manufacturer
Primary Production (software)
Testing
Implementation
Design
Requirements Analysis
Attacks can occur at all levels in the supply chain
SUPPLY CHAIN

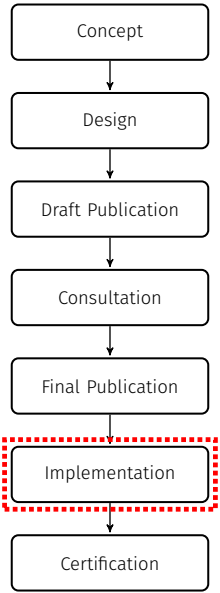
Supply Chain Attacks





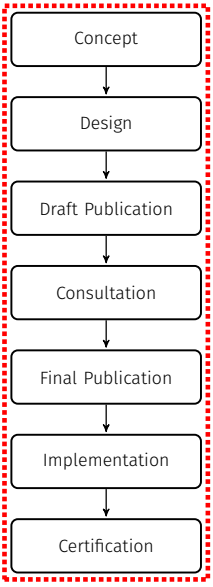
(TS//SI//NF) Not all SIGINT tradecraft involves accessing signals and networks from thousands of miles away… In fact, sometimes it is very hands-on (literally!). Here's how it works: shipments of computer network devices (servers, routers, etc.) being delivered to our targets throughout the world are ***intercepted***. Next, they are ***redirected to a secret location*** where Tailored Access Operations/Access Operations (AO – S326) employees, with the support of the Remote Operations Center (S321), enable the ***installation of beacon implants*** directly into our targets' electronic devices. These devices are then re-packaged and ***placed back into transit*** to the original destination. All of this happens with the support of Intelligence Community partners and the technical wizards in TAO.

Traditional Supply Chain Attack





Pre-Supply Chain Operations (PSYCHOs)



Pre-Supply Chain Operations (PSYCHOs)

Adversarial actions that:

1. Are surreptitious

Adversarial actions that:

1. Are surreptitious
2. Corrupt the design, standardization or certification stages of a system life-cycle

- **Regulatory Rollover.** Create fragility as a side effect of requirements

- **Regulatory Rollover.** Create fragility as a side effect of requirements
- **Direct Design Flaw.** Design system with subtle flaw, that if noticed, is downplayed. Includes intentional omission of details.

- **Regulatory Rollover.** Create fragility as a side effect of requirements
- **Direct Design Flaw.** Design system with subtle flaw, that if noticed, is downplayed. Includes intentional omission of details.
- **Parameter Manipulation.** Standardize fixed, malicious, parameters that would otherwise be implementer chosen

- **Regulatory Rollover.** Create fragility as a side effect of requirements
- **Direct Design Flaw.** Design system with subtle flaw, that if noticed, is downplayed. Includes intentional omission of details.
- **Parameter Manipulation.** Standardize fixed, malicious, parameters that would otherwise be implementer chosen
- **Intentional Complexity.** Make system complex to encourage bugs

Taxonomy

- **Regulatory Rollover.** Create fragility as a side effect of requirements
- **Direct Design Flaw.** Design system with subtle flaw, that if noticed, is downplayed. Includes intentional omission of details.
- **Parameter Manipulation.** Standardize fixed, malicious, parameters that would otherwise be implementer chosen
- **Intentional Complexity.** Make system complex to encourage bugs
- **Threat Model Exclusion.** Exclude classes of vulnerabilities from a standard's threat model

Does this actually apply?

SIMON & SPECK (1)

- NSA designed lightweight ciphers

SIMON & SPECK (1)

- NSA designed lightweight ciphers
- Attempted standardized at ISO

SIMON & SPECK (1)

- NSA designed lightweight ciphers
- Attempted standardized at ISO
- Included in Linux/Android

SIMON & SPECK (2)

- Little explanation of parameter choices

SIMON & SPECK (2)

- Little explanation of parameter choices
- Procedural mischief at ISO

SIMON & SPECK (2)

- Little explanation of parameter choices
- Procedural mischief at ISO
- Ashur et al. raised concern!

SIMON & SPECK (2)

- Little explanation of parameter choices
- Procedural mischief at ISO
- Ashur et al. raised concern!
- Failed standardization. Removed from Linux.

EXPORT **Grade Cryptography**

- Legislation & Regulation to prevent export of military technology
- Applied to limit key lengths in products shipped outside of the USA
- Support for EXPORT-grade ciphersuites in TLS led to multiple protocol breaks

Diffie-Hellman Attacks (1)

- RFC5114 is a standard with DH groups of questionable provenance

Diffie-Hellman Attacks (1)

- RFC5114 is a standard with DH groups of questionable provenance
- Valenta et al. show that under certain conditions using these groups can lead to MiTM attacks

Diffie-Hellman Attacks (2)

- Adrian et al. explore precomputation attacks against TLS key exchange using standardized/fixed DH groups

Diffie-Hellman Attacks (2)

- Adrian et al. explore precomputation attacks against TLS key exchange using standardized/fixed DH groups
- Combining with a EXPORT downgrade they are able to compromise 7% of the Alexa 1M

TETRA TEA1/TEA3

- European standard for secure radio communications, widely used

TETRA TEA1/TEA3

- European standard for secure radio communications, widely used
- TEA1/TEA3: encryption algorithms for commercial/outside European use

TETRA TEA1/TEA3

- European standard for secure radio communications, widely used
- TEA1/TEA3: encryption algorithms for commercial/outside European use
- TEA1/TEA3 had *critical design flaws* (incl. key strength reduction from 80-bits to breakable on a 1990s laptop) (Midnight Blue, BH '23)

TETRA TEA1/TEA3

- European standard for secure radio communications, widely used
- TEA1/TEA3: encryption algorithms for commercial/outside European use
- TEA1/TEA3 had *critical design flaws* (incl. key strength reduction from 80-bits to breakable on a 1990s laptop) (Midnight Blue, BH '23)
- The standardization process lacks transparency — violate's Kerchoff's principle

Classifying Failures

Affected	Suspected PSYCHO	Years Active	Flaw Type
X9.31	✗	1985-2016	D
Dual_EC	✓	2004-2016	D,P
CTR_DRBG	✗	2006-	D,T
RFC 5114	✓	2008-	P
Simon & Speck	✓	—	D
Logjam	?	1990s-2010s	R,P,C
TLS Attacks	?	1990s-	R,C
TETRA TEA1/TEA3	✓	1994-	D

D: design flaw

P: poor parameter choices

C: intentionally introduced complexity

R: regulatory impositions

T: inadequate threat model

Where standards can be bent

Body	Who sits and votes	Decision principle	Crypto risk signals
ETSI	Full and Associate members	Consensus else seventy one per cent weighted vote	NDAs; some annexes restricted
IETF	Individuals	Rough consensus and running code	Open by default, but weak parameters can linger
ISO IEC JTC 1	National bodies	Two thirds yes and less than one quarter no	Closed meetings and paywalled outputs
3GPP	Firms via partner SDOs; org. partners steer	Consensus else seventy one per cent vote in meetings	Some cipher specs gated by Restricted Usage Undertaking

- Standardization is not security

- Standardization is not security
- Certification is not a pentest

So how is the NSA actually breaking crypto?

Sexy (maybe)

- Hypothetical $L_N[1/4]$ algorithm for DH mod p
- Trapdoored "random" primes (hidden SNFS)
- Side-channel attacks

Boring (probable)

- ~~Massive precomputation on reused DH groups (esp. 1024-bit)~~
- ~~RNG/DRBG abuse~~
- Legacy attacks
- HSM/firmware implants/0-days
- Regulatory attacks
- More intentional protocol flaws?
- Lots of ASICs
- Supply-chain insiders (CryptoAG)

$L_N(\alpha, c) = \exp((c + o(1))(\ln N)^{\alpha}(\ln \ln N)^{1-\alpha})$.

Lessons

- Standards and standards processes are a target for adversaries
- Opaque vendor (hardware) implementations likewise.
- Being an a**hole in a meeting could be an intentional strategy to introduce flaws!

Lessons

- Standards and standards processes are a target for adversaries
- Opaque vendor (hardware) implementations likewise.
- Being an a**hole in a meeting could be an intentional strategy to introduce flaws!
- Actually breaking crypto oft. needs a design flaw **and** significant implementation weaknesses.

Questions?

How to generate random numbers

Q: Can I use RDRAND or equivalent?

How to generate random numbers

Q: Can I use RDRAND or equivalent?

A: Probably not a good idea. Hard to find bugs and backdoors.

How to generate random numbers

Q: Can I use RDRAND or equivalent?

A: Probably not a good idea. Hard to find bugs and backdoors.

Use it as one of many entropy sources to a software PRG.

How to generate random numbers

Q: Are there any PRGs that are safe? What if I need to be FIPS compliant?

How to generate random numbers

Q: Are there any PRGs that are safe? What if I need to be FIPS compliant?

A: Not everything is broken!

How to generate random numbers

Q: Are there any PRGs that are safe? What if I need to be FIPS compliant?

A: Not everything is broken!

Hash_DRBG is fine if implemented correctly and securely.