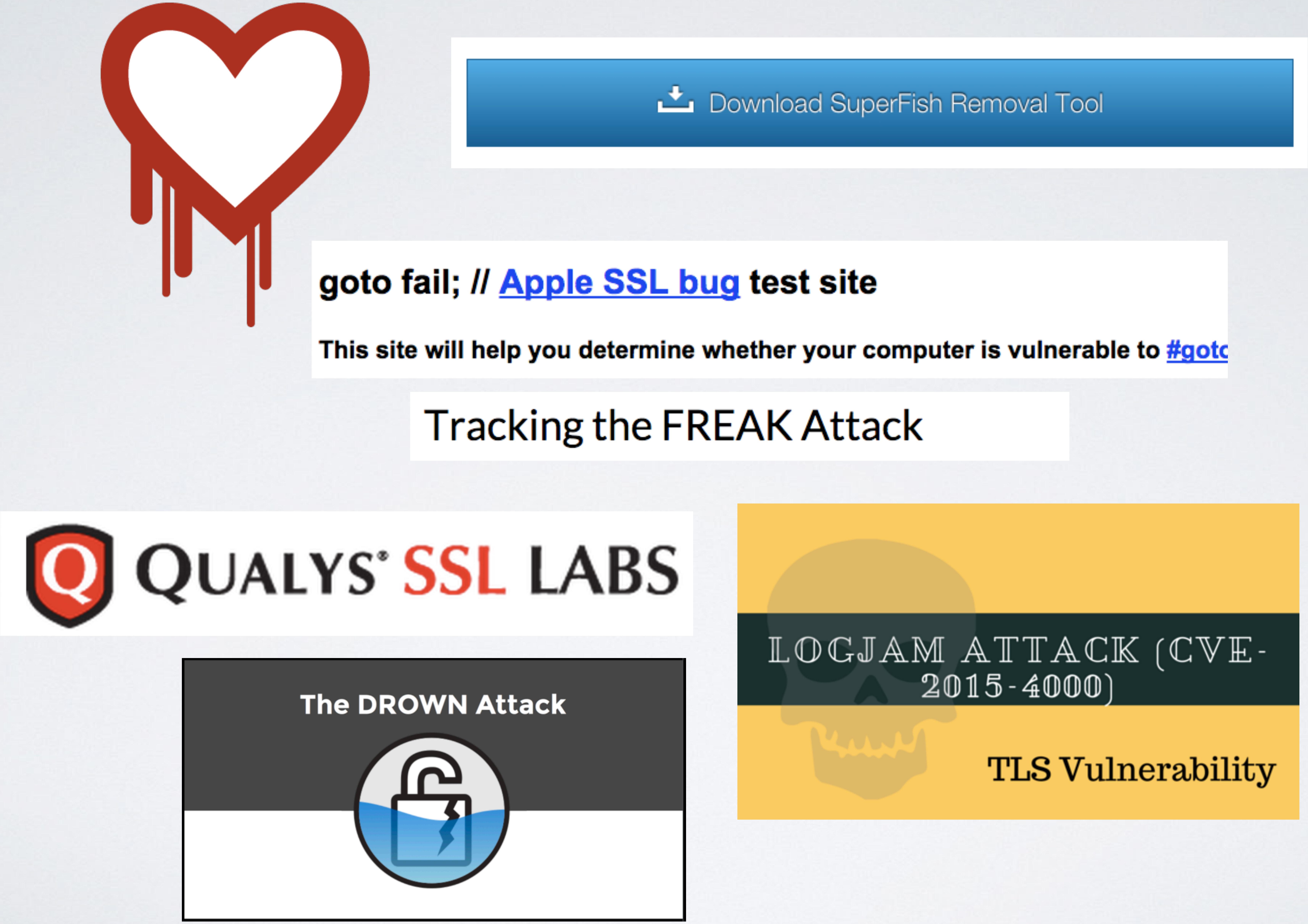
Building trust in an untrustworthy world

Matthew Green
Johns Hopkins University

Presentation for NSF SATC PI meeting (10/29/2019)


Download SuperFish Removal Tool
goto fail; // Apple SSL bug test site
This site will help you determine whether your computer is vulnerable to #gotc
Tracking the FREAK Attack
QUALYS SSL LABS
The DROWN Attack
LOGJAM ATTACK (CVE-2015-4000)
TLS Vulnerability

Tweets
47
Following
1
Followers
2,131
OpenSSL Fact
@OpenSSLFact
One terrible, frightening line of OpenSSL code each day. 365 days a year until the madness ends. Maintained by @matthew_d_green.
openssl.org
Joined September 2012
Tweets
Tweets & replies
OpenSSL Fact Retweeted
Thomas Ptacek @tqbf · 20 May 2015
In crypto:
VERSIONING > NEGOTIATION.
Nobody in the history of cryptography has ever gotten negotiation right.
4
36
43
OpenSSL Fact Retweeted
Tony "ABOLISH ICE" Arcieri @bascule · 16 Apr 2014
OpenSSL thinks 15 is a prime number: blog.hboeck.de/archives/841-D... /cc @OpenSSLFact

NEWS

In Heartbleed's wake, tech titans launch fund for crucial open-source projects

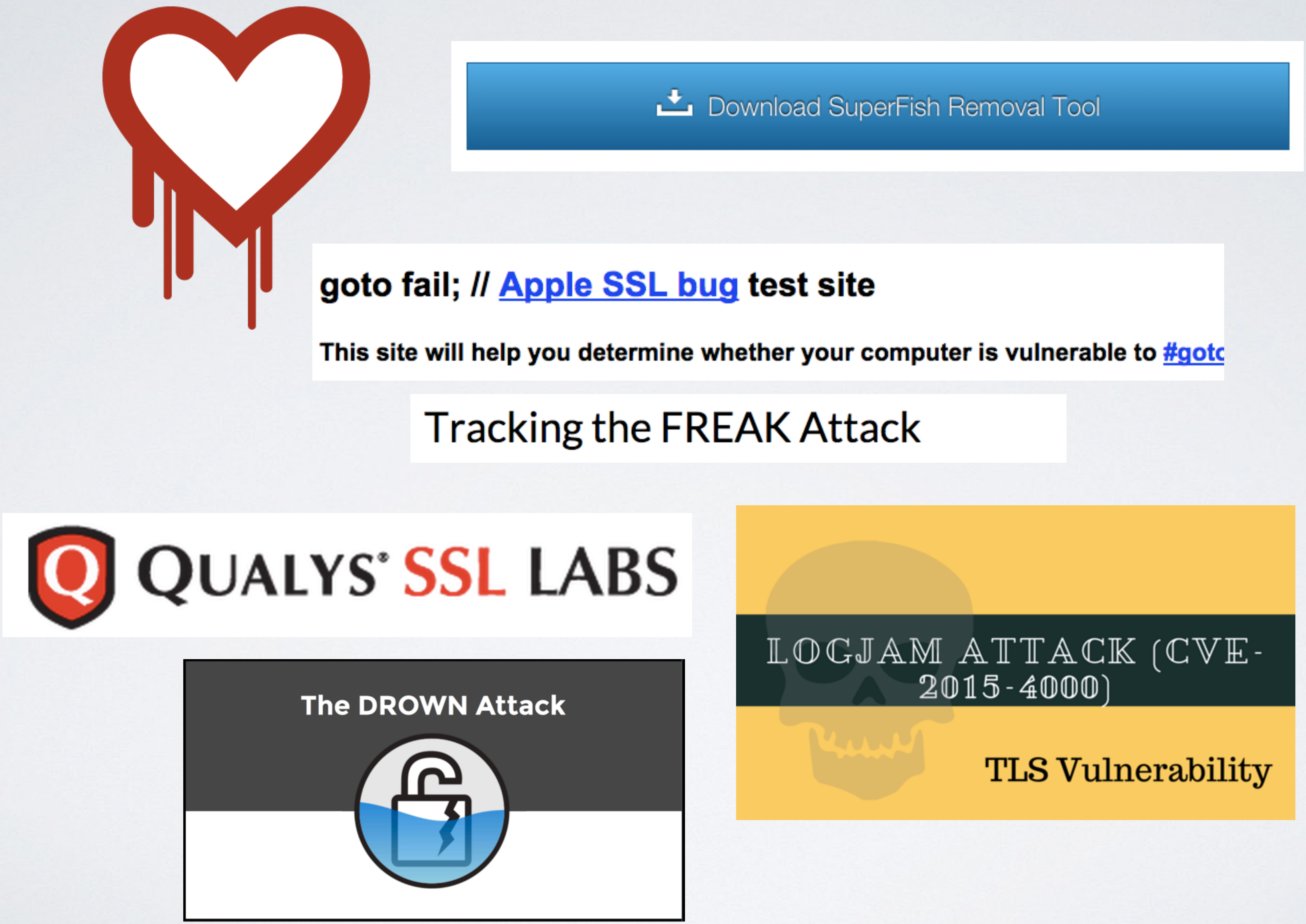
By Ian Paul
Contributor, PCWorld | APR 24, 2014 6:50 AM PDT

TODAY'S BEST TECH DEALS
Picked by PCWorld's Editors


This 15-inch Acer laptop for $150 is perfect for work and play


AMD slashes RX 590 prices as rumors of Nvidia's GTX 1660 Super heat up


The classic Logitech G402 Hyperion Fury gaming


Download SuperFish Removal Tool
goto fail; // Apple SSL bug test site
This site will help you determine whether your computer is vulnerable to #gotc
Tracking the FREAK Attack
QUALYS SSL LABS
LOGJAM ATTACK (CVE-2015-4000)
TLS Vulnerability
The DROWN Attack

Response to improving security

- For the past decade, NSA has lead an aggressive, multi-pronged effort to break widely used Internet encryption technologies
- Cryptanalytic capabilities are now coming on line
- Vast amounts of encrypted Internet data which have up till now been discarded are now exploitable
- Major new processing systems, SIGDEV efforts and tasking must be put in place to capitalize on this opportunity

PTD "We penetrate targets' defences."

GCHQ

This information is exempt from disclosure under the Freedom of Information Act 2000 and may be subject to exemption under other UK information legislation. Refer disclosure requests to GCHQ on 01242 221491 x30306 (non-sec) or email infoleg@gchq

© Crown Copyright. All rights reserved.

This talk

- **Our community has made enormous progress towards building secure cryptographic systems…**

This talk

- **Our community has made enormous progress towards building secure cryptographic systems…**
  - At the level of algorithm and protocol design…
  - At the level of implementation and deployment…

This talk

- **Our community has made enormous progress towards building secure cryptographic systems...**
  - At the level of algorithm and protocol design…
  - At the level of implementation and deployment…

Yet all of this progress is based on the assumption that <u>system designers are on our side</u>.

This talk

- **Our community has made enormous progress towards building secure cryptographic systems…**
  - At the level of algorithm and protocol design…
  - At the level of implementation and deployment…

Yet all of this progress is based on the assumption that <u>system designers are on our side</u>.

What if they aren't?

Kleptography

(n.) The study of stealing cryptographic secrets securely and subliminally.

(Young & Yung, 1996)

Kleptography: Using Cryptography Against Cryptography

Adam Young* and Moti Yung**

Abstract. The notion of a Secretly Embedded Trapdoor with Universal Protection (SETUP) has been recently introduced. In this paper we extend the study of stealing information securely and subliminally from black-box cryptosystems. The SETUP mechanisms presented here, in contrast with previous ones, leak secret key information without using an explicit subliminal channel. This extends this area of threats, which we call "kleptography".

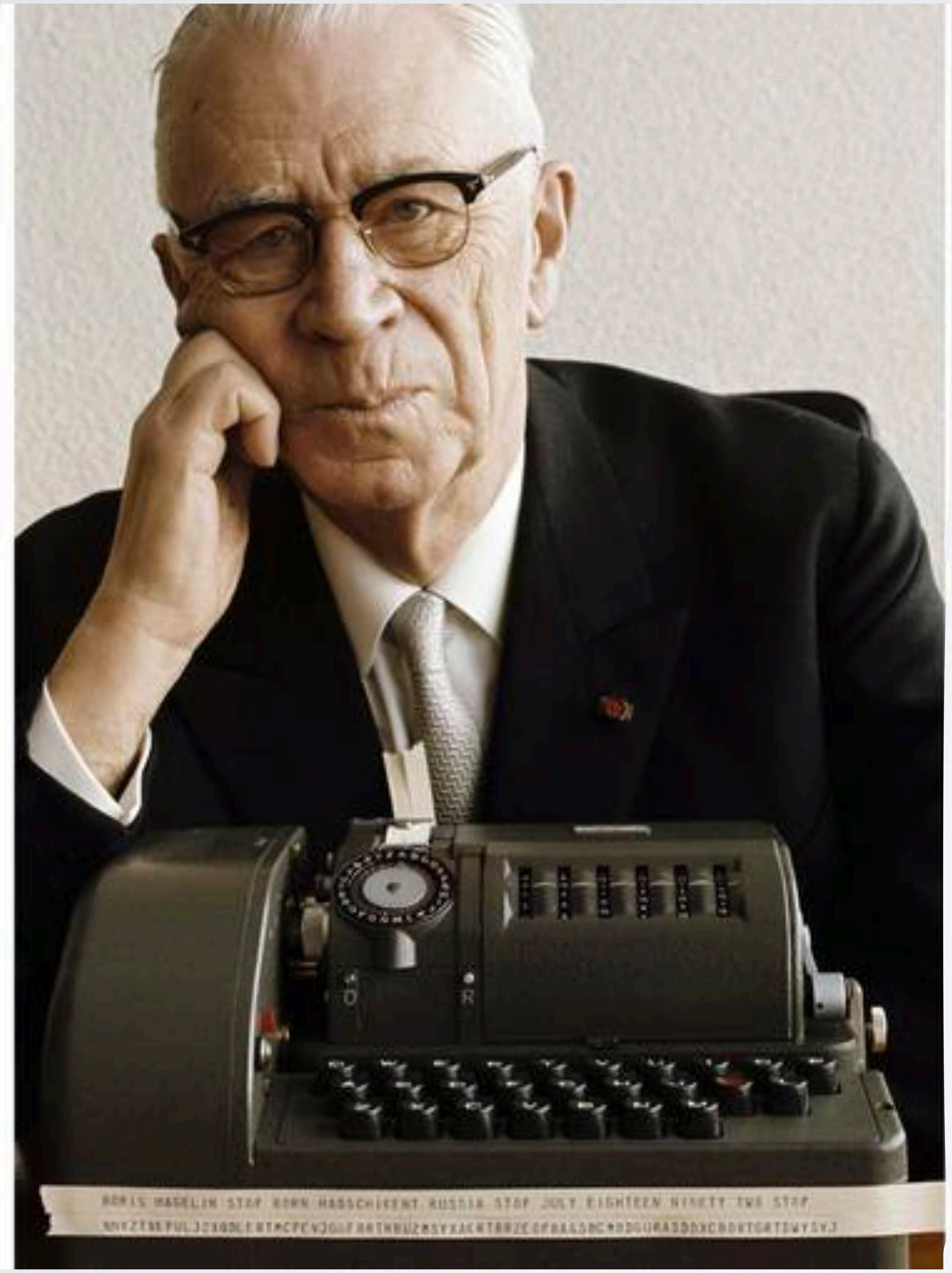
We introduce new definitions of SETUP attacks (strong, regular, and weak SETUPs) and the notion of m out of n leakage bandwidth. We show a strong attack which is based on the discrete logarithm problem.

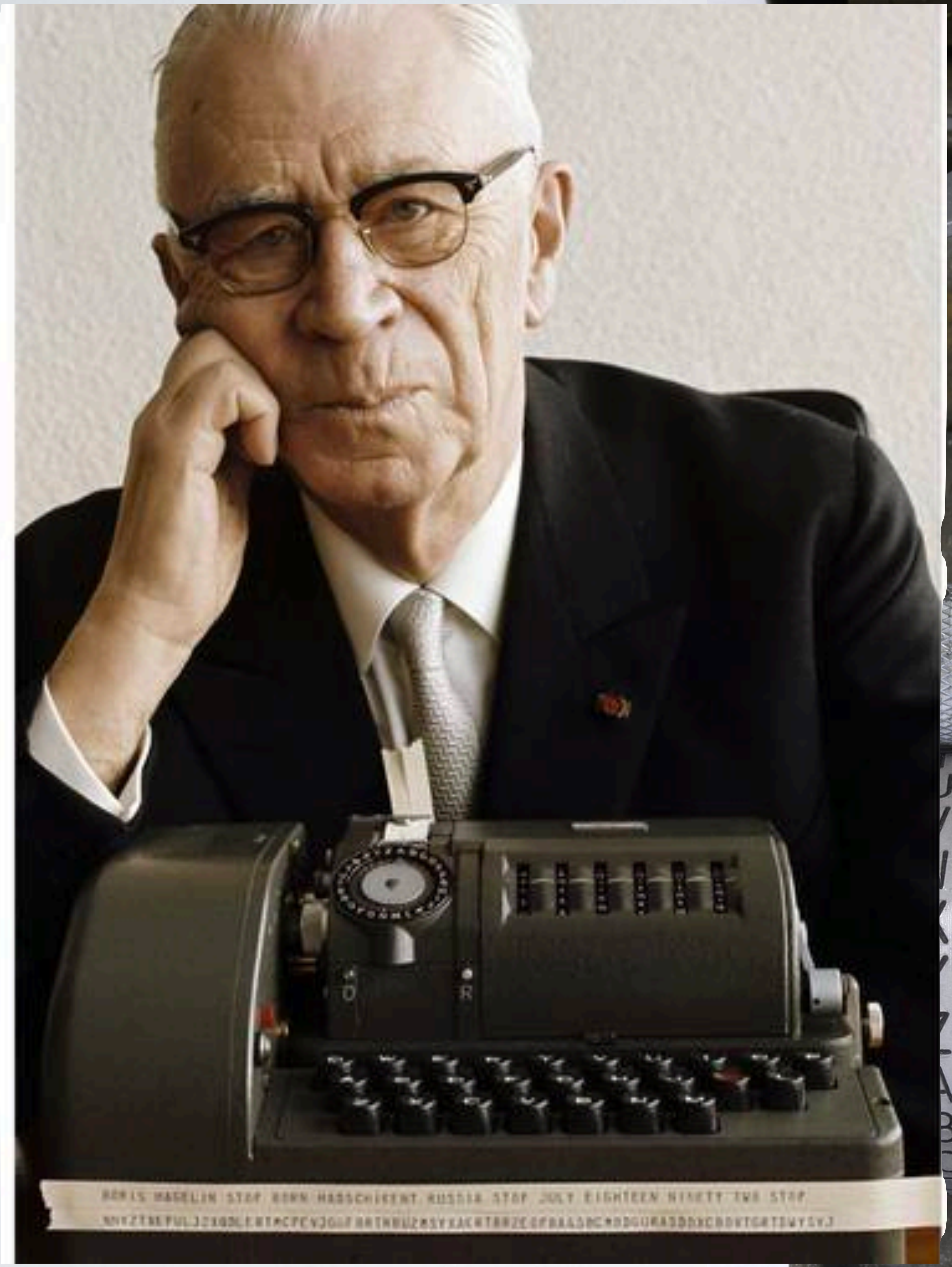
The Dark Side of "Black-Box" Cryptography or: Should We Trust Capstone?*

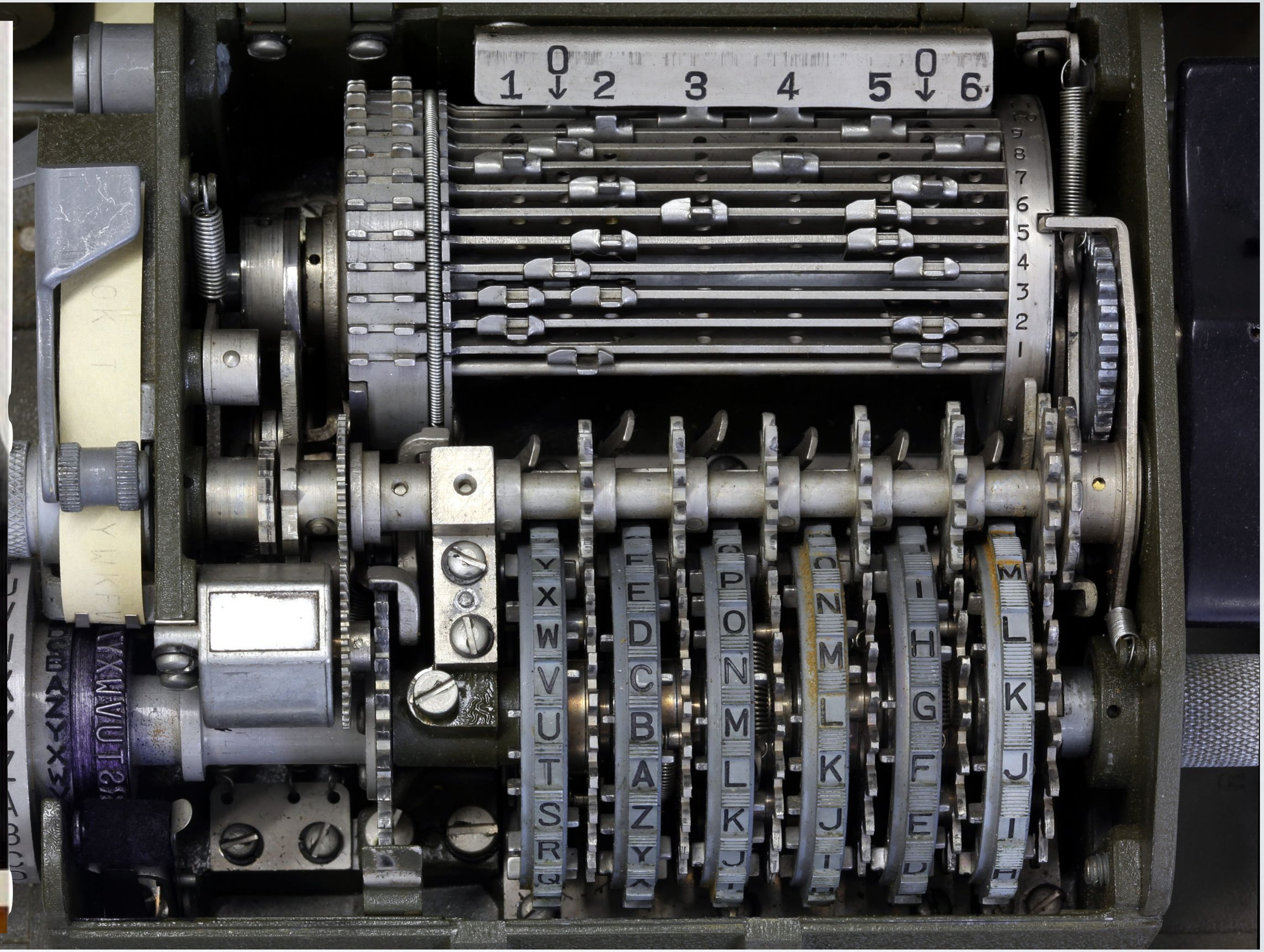
Adam Young** and Moti Yung*** Yorktown Heights, NY 10598, USA.
Email: moti@watson.ibm.com

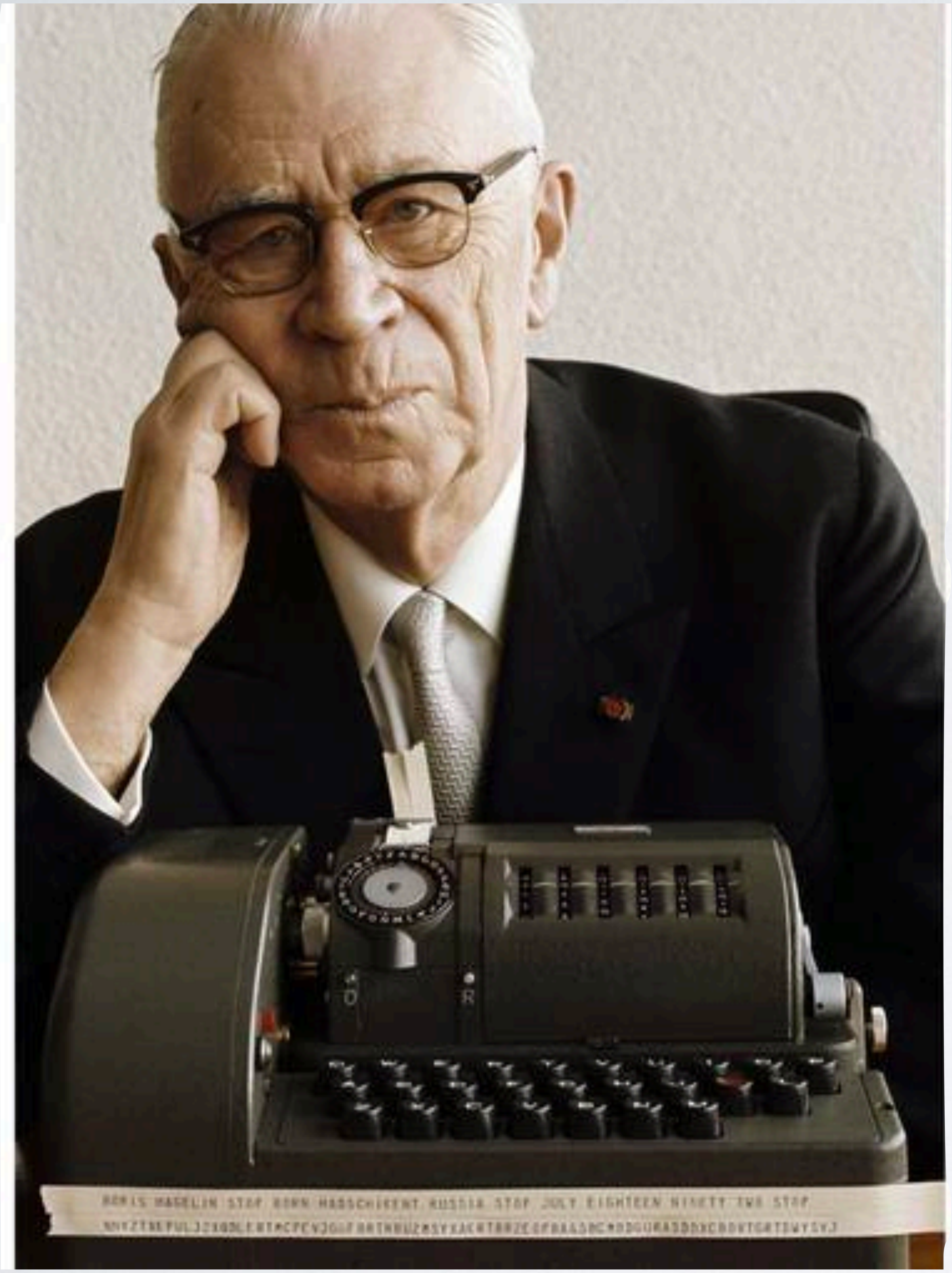
Abstract. The use of cryptographic devices as "black boxes", namely trusting their internal designs, has been suggested and in fact Capstone technology is offered as a next generation hardware-protected escrow encryption technology. Software cryptographic servers and programs are being offered as well, for use as library functions, as cryptography gets more and more prevalent in computing environments. The question we address in this paper is how the usage of cryptography as a black box exposes users to various threats and attacks that are undetectable in a black-box environment. We present the SETUP (Secretly Embedded Trapdoor with Universal Protection) mechanism, which can be embedded in a cryptographic black-box device. It enables an attacker (the manufacturer) to get the user's secret (from some stage of the output

A bit of history (~1950s-1980s)






1 0 2 3 4 5 0 6



HAGELIN-CRYPTOS

CRYPTO AG

- William F. Friedman (Army SIS, NSA)
  - 1950s: visited Hagelin and son in Zug, Switzerland
  - After his death, papers donated to George C. Marshall foundation. They mention a "gentleman's agreement"
  - NSA requested papers be sequestered in 1976 (accidentally re-opened 1979-1983, then closed again)
  - **In 2015, redacted versions were released to the public**

1. In accordance with Letter Orders 273 dated 27 January 1955, as modified by L.O. 273-A dated 4 February 1955, I left Washington via MATS at 1500 hours on 18 February 1955, arrived at Orly Field, Paris, at 1430 hours on 19 February, and at Zug, Switzerland, at 1830 the same day. I spent the next few days with Mr. Boris C.W. Hagelin, Senior, and Mr. Boris Hagelin, Junior, for the purpose of learning the status of their new developments in crypto-apparatus and of making an approach and a proposal to Mr. Hagelin, Senior as was recently authorized by USCIB and concurred in by LSIB. Upon completion of that part of my mission, I left Zug at 1400 hours on 28 February and proceeded to London, arriving at 1845 that evening.

1. In accordance with Letter Orders 273 dated 27 January 1955, as modified by L.O. 273-A dated 4 February 1955, I left Washington via MATS at 1500 hours on 18 February 1955, arrived at Orly Field, Paris, at 1430 hours on 19 February, and at Zug, Switzerland, at 1830 the same day. I spent the next few days with Mr. Boris C.W. Hagelin, Senior, and Mr. Boris Hagelin, Junior, for the purpose of learning the status of their new developments in crypto-apparatus and Upon completion of that part of my mission, I left Zug at 1400 hours on 28 February and proceeded to London, arriving at 1845 that evening.

(3) Hagelin Junior was so enthusiastic about this new model that within two or three minutes immediately following our initial exchange of greetings he announced that they had decided to stop making the CX model and are switching over to a variation of the C-52 which, he said, "is simpler in mechanical effectuation and more readily adaptable to the crypto-control mechanism for the HX or electrical-rotor machine." I was, of course, rather startled by this statement and later queried Hagelin Senior about it, saying that I was astonished at the decision to switch to the C-52Y before any security evaluation at all had been made of it. Hagelin Senior said, "Oh, Bo is young and overflowing with enthusiasm. We will hold up making that model if you want us to hold up on it." I told him that I thought this might be advisable, and that in any case we would want one of these models just as soon as possible. Hagelin Senior said that it was easy to convert a

TOP SECRET

(3) Hagelin Junior was so enthusiastic about this new model that within two or three minutes immediately following our initial exchange of greetings he announced that they had decided to stop making the CX model and are switching over to a variation of the C-52 which, he said, "is simpler in mechanical effectuation and more readily adaptable to the crypto-control mechanism for the HX or electrical-rotor machine." I was, of course, rather startled by this statement and later queried Hagelin Senior about it, saying that I was astonished at the decision to switch to the C-52Y before any security evaluation at all had been made of it. Hagelin Senior said, "Oh, Bo is young and overflowing with enthusiasm. [redacted] Hagelin Senior said that it was easy to convert a

EO 3.3(h)(2)
PL 86-36/50 USC 3605

- In the 1980s, Iran arrested Crypto AG's representative Hans Buhler on suspicion that the company's machines were backdoor
- The company denied everything, paid a $1m ransom, then charged it to Buhler
- Buhler and other employees went to the press, providing a stream of accusations of government collusion that destroyed the company
- Crypto AG was saved from bankruptcy by "angel investor" Marc Rich

TOP SECRET//SI/TK//NOFORN

(U) COMPUTER NETWORK OPERATIONS
(U) SIGINT ENABLING

Source: NYT/ProPublica

(U) Project Description

(TS//SI//NF) The SIGINT Enabling Project actively engages the US and foreign IT industries to covertly influence and/or overtly leverage their commercial products' designs. These design changes make the systems in question exploitable through SIGINT collection (e.g., Endpoint, MidPoint, etc.) with foreknowledge of the modification. To the consumer and other adversaries, however, the systems' security remains intact. In this

- (TS//SI//NF) Shape the worldwide commercial cryptography marketplace to make it more tractable to advanced cryptanalytic capabilities being developed by NSA/CSS. [CCP_00090]

- (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.

TOP SECRET//SI/TK//NOFORN

(U) COMPUTER NETWORK OPERATIONS
(U) SIGINT ENABLING

Source: NYT/ProPublica

(U) Project Description

(TS//SI//NF) The SIGINT Enabling Project actively engages the US and foreign IT industries to covertly influence and/or overtly leverage their commercial products' designs. These design changes make the systems in question exploitable through SIGINT collection (e.g., Endpoint, MidPoint, etc.) with foreknowledge of the modification. To the consumer and other adversaries, however, the systems' security remains intact. In this

- (TS//SI//NF) Shape the worldwide commercial cryptography marketplace to make it more tractable to advanced cryptanalytic capabilities being developed by NSA/CSS. [CCP_00090]

- (TS//SI//REL TO USA, FVEY) Insert vulnerabilities into commercial encryption systems, IT systems, networks, and endpoint communications devices used by targets.

Insert vulnerabilities into commercial encryption systems, IT systems,

- (TS//SI//REL TO USA, FVEY) Influence policies, standards and specification for commercial public key technologies.

TOP SECRET//SI/TK//NOFORN

(U) COMPUTER NETWORK OPERATIONS
(U) SIGINT ENABLING

Source: NYT/ProPublica

(U) Project Description

(TS//SI//NF) The SIGINT Enabling Project actively engages the US and foreign IT industries to covertly influence and/or overtly leverage their commercial products' designs. These design changes make the systems in question exploitable through SIGINT collection (e.g., Endpoint, MidPoint, etc.) with foreknowledge of the modification. To the consumer and other adversaries, however, the systems' security remains intact. In this

- (TS//SI//NF) Shape the worldwide commercial cryptography marketplace to make it more tractable to advanced cryptanalytic capabilities being developed by NSA/CSS. [CCP_00090]

- (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

Influence policies, standards and specification for commercial public key technologies.

- (TS//SI//REL TO USA, FVEY) Influence policies, standards and specification for commercial public key technologies.

Source: NYT/ProPublica

TOP SECRET//SI/TK//NOFORN

(U) COMPUTER NETWORK OPERATIONS
(U) SIGINT ENABLING

(U) Project Description

(TS//SI//NF) The SIGINT Enabling Project actively engages the US and foreign IT industries to covertly influence and/or overtly leverage their commercial products' designs. These design changes make the systems in question exploitable through SIGINT collection (e.g., Endpoint, MidPoint, etc.) with foreknowledge of the modification. To the consumer and other adversaries, however, the systems' security remains intact. In this

- (TS//SI//NF) Shape the worldwide commercial cryptography marketplace to make it more tractable to advanced cryptanalytic capabilities being developed by NSA/CSS. [CCP_00090]

To the consumer and other adversaries,
however, the systems' security remains intact.

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.

How do you build a Kleptographic system?

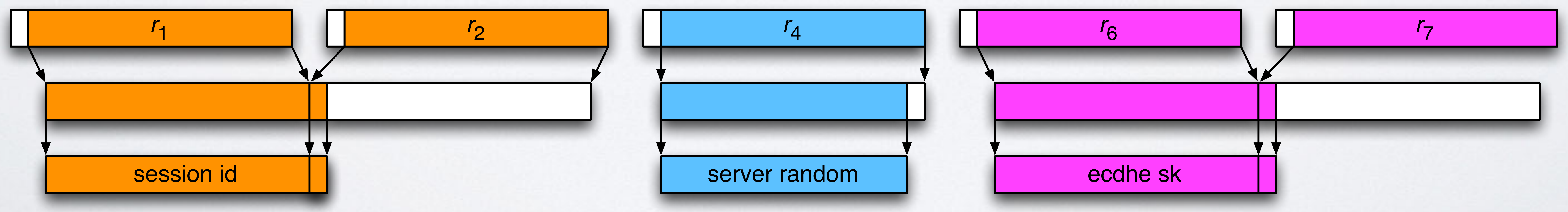
- **That is, a system many will use?**
  - Unlike Crypto AG, you can't mandate the hardware
  - The protocols are already extant (IPSec, SSL, etc.)
  - Can't really mandate the software
  - You *can* mandate cryptographic algorithms
  - You *can* **validate** cryptographic implementations

Achilles heel: randomness

- **Many protocols, one commonality:**
  - Most cryptographic protocols <u>devour</u> random bits
    - <u>Ex</u>: 108 bytes / TLS session (ECDH+ECDSA, server)
  - The quality of those bits is hugely important
  - Attacker who can <u>predict</u> (P)RNG output can break (almost) any protocol

Achilles heel: randomness

- **Moreover, a single generator may produce both <u>public</u> and <u>secret</u> values**
  - In practice an RNG must remain secure when the attacker can see some public output
  - This is something engineers take for granted, and rely on w/o conscious thought



RNG System Architecture (1)

TRNG

bits

Crypto Protocols

(SSL, TLS, IPSec, etc.)

<u>Probabilistic</u>:
system specific, hardware/entropy collection

RNG System Architecture (2)

seed

bits

Crypto Protocols

(SSL, TLS, IPSec, etc.)

TRNG

PRNG (DRBG)

Probabilistic:
system specific, hardware, entropy collection

Deterministic:
computational, fast, algorithmic

RNG System Architecture (2)

TRNG

seed

PRNG
(DRBG)

bits

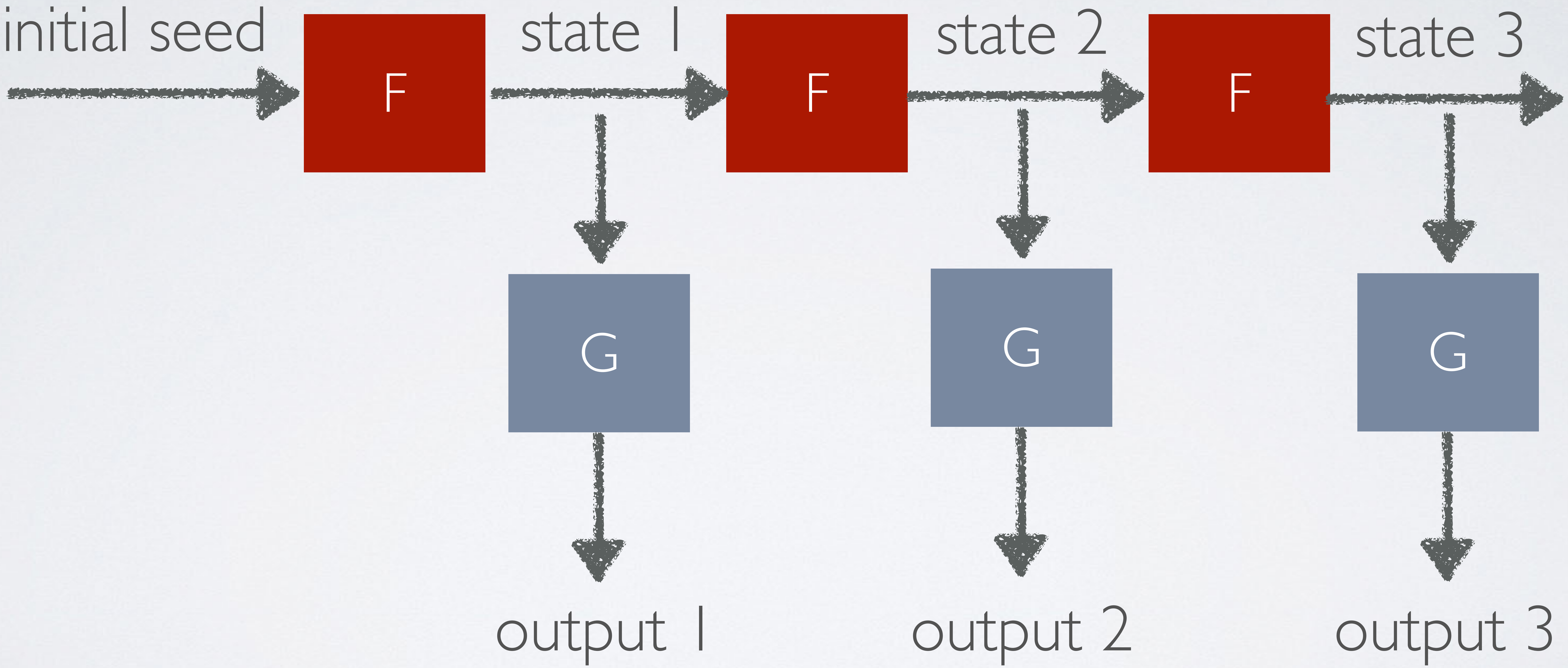
Crypto Protocols

(SSL, TLS, IPSec, etc.)

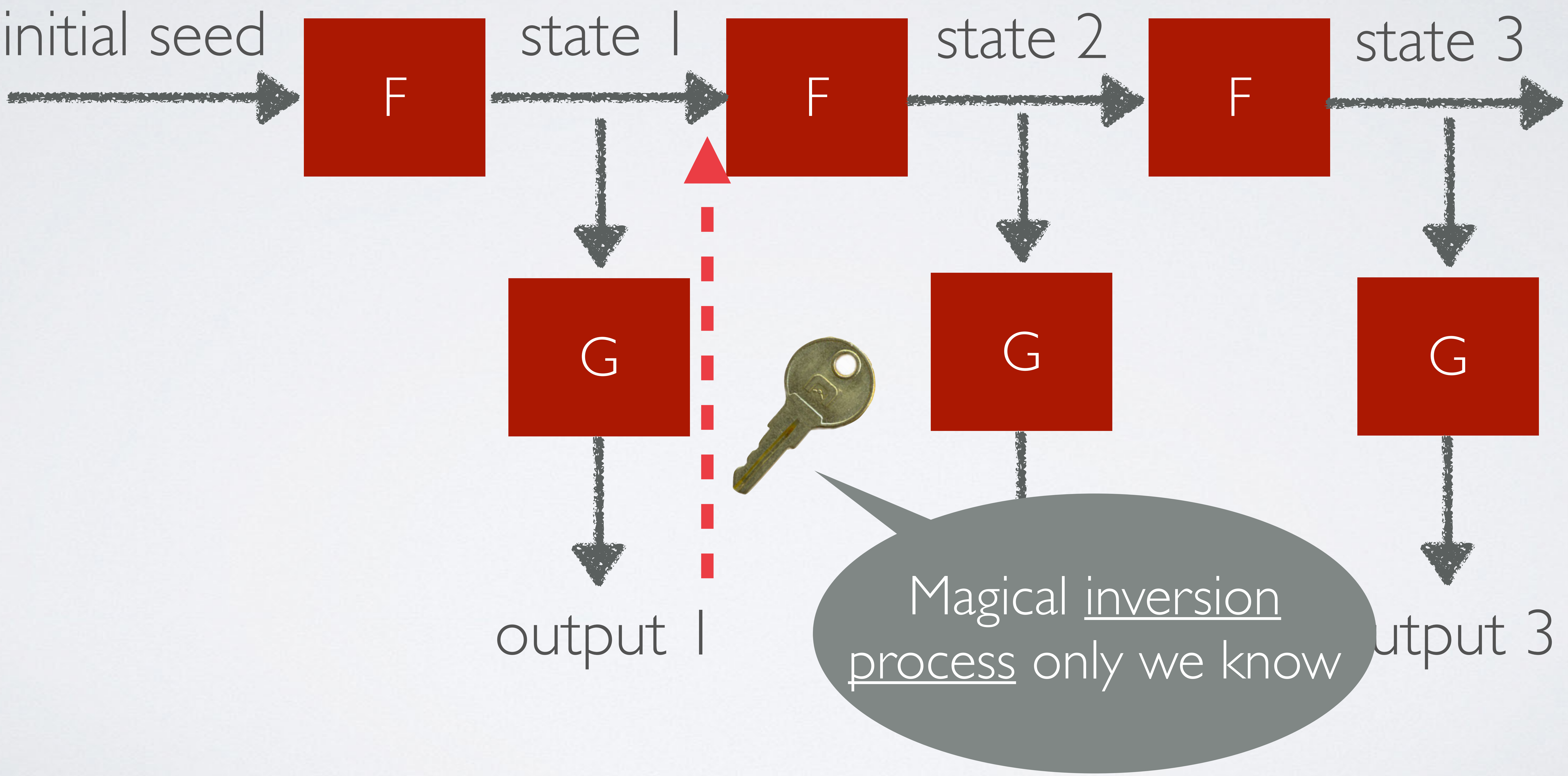
Probabilistic:
system specific, hardware, entropy collection

Deterministic:
computational, fast, algorithmic

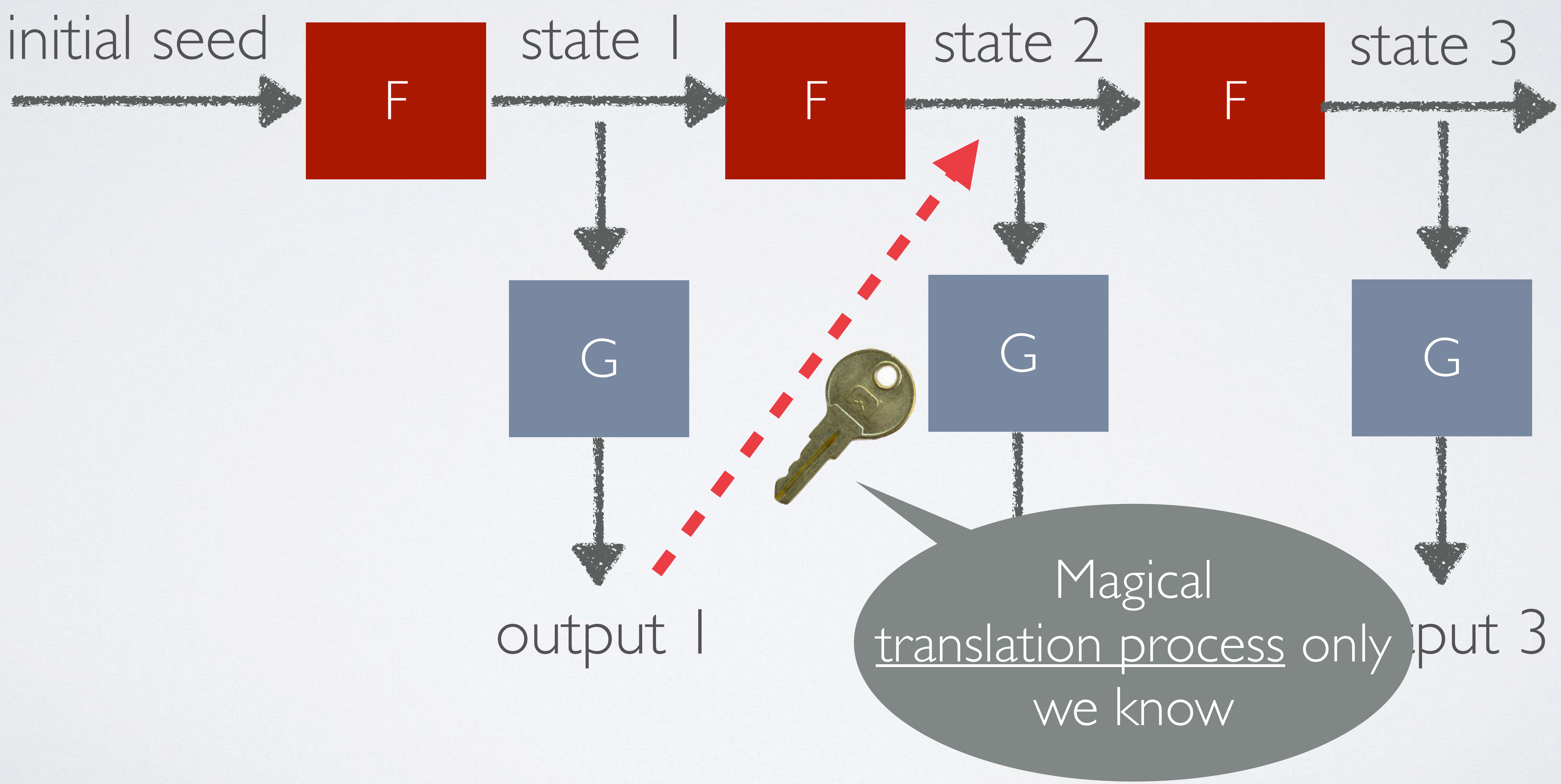
Template for a DRBG



Kleptographic proposal #1: Make G invertible



Kleptographic proposal #2: A mapping from G to F



1996: Young & Yung SETUPs

Schnorr.KG: $\langle g \rangle = \mathbb{G}$ of prime order q

$$sk = x \leftarrow \mathbb{Z}_q \text{ , } PK = g^x$$

Schnorr.Sign(1): $k \leftarrow \mathbb{Z}_q$

$$c = H(PK \| g^k \| m)$$

$$r = g^k$$

$$s = xc + k \ mod\ q$$

output: (r, s)

Young & Yung: SETUPs

SETUP.KG: MK $= g^y, msk = y$

Schnorr.KG: $\langle g \rangle = \mathbb{G}$ of prime order q

$sk = x \leftarrow \mathbb{Z}_q$, $PK = g^x$

Schnorr.Sign(1): $k \leftarrow \mathbb{Z}_q$

$c = H(PK \| g^k \| m)$

$r = g^k$

$s = xc + k \ mod\ q$

output: (r, s)

Young & Yung: SETUPs

SETUP.KG: $\mathsf{MK} = g^y, msk = y$

Schnorr.KG: $\langle g \rangle = \mathbb{G}$ of prime order q

$sk = x \leftarrow \mathbb{Z}_q$, $PK = g^x$

Schnorr.Sign(1): $k \leftarrow \mathbb{Z}_q$

$c = H(PK \| g^k \| m)$

$r = g^k$

$s = xc + k \ mod \ q$

output: (r, s)

Schnorr.Sign(2): $k' \leftarrow H(\mathsf{MK}^k) \in \mathbb{Z}_q$

Compute next signature using k'

Young & Yung: SETUPs

SETUP.KG: $\mathsf{MK} = g^y, msk = y$

Schnorr.KG: $\langle g \rangle = \mathbb{G}$ of prime order q

Given r and msk we can recover k' as:

$$k' = H(r^{msk})$$

and thus obtain the long term secret key.

Schnorr ... ut: (r, s)

$$c = H(PK \| g^k \| m)$$

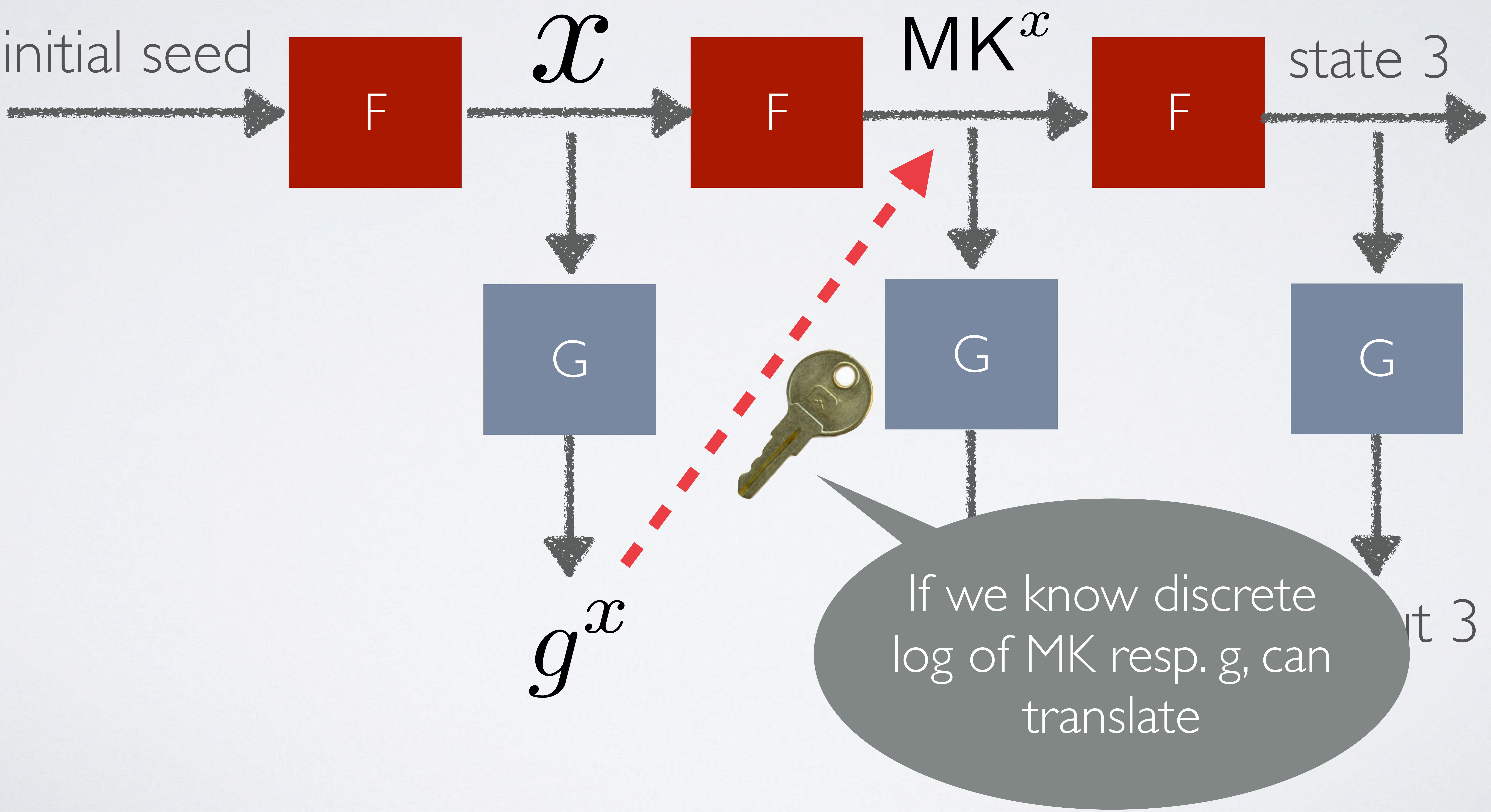
$$r = g^k$$

$$s = xc + k \mod q$$

Schnorr.Sign(2): $k' \leftarrow H(\mathsf{MK}^k) \in \mathbb{Z}_q$

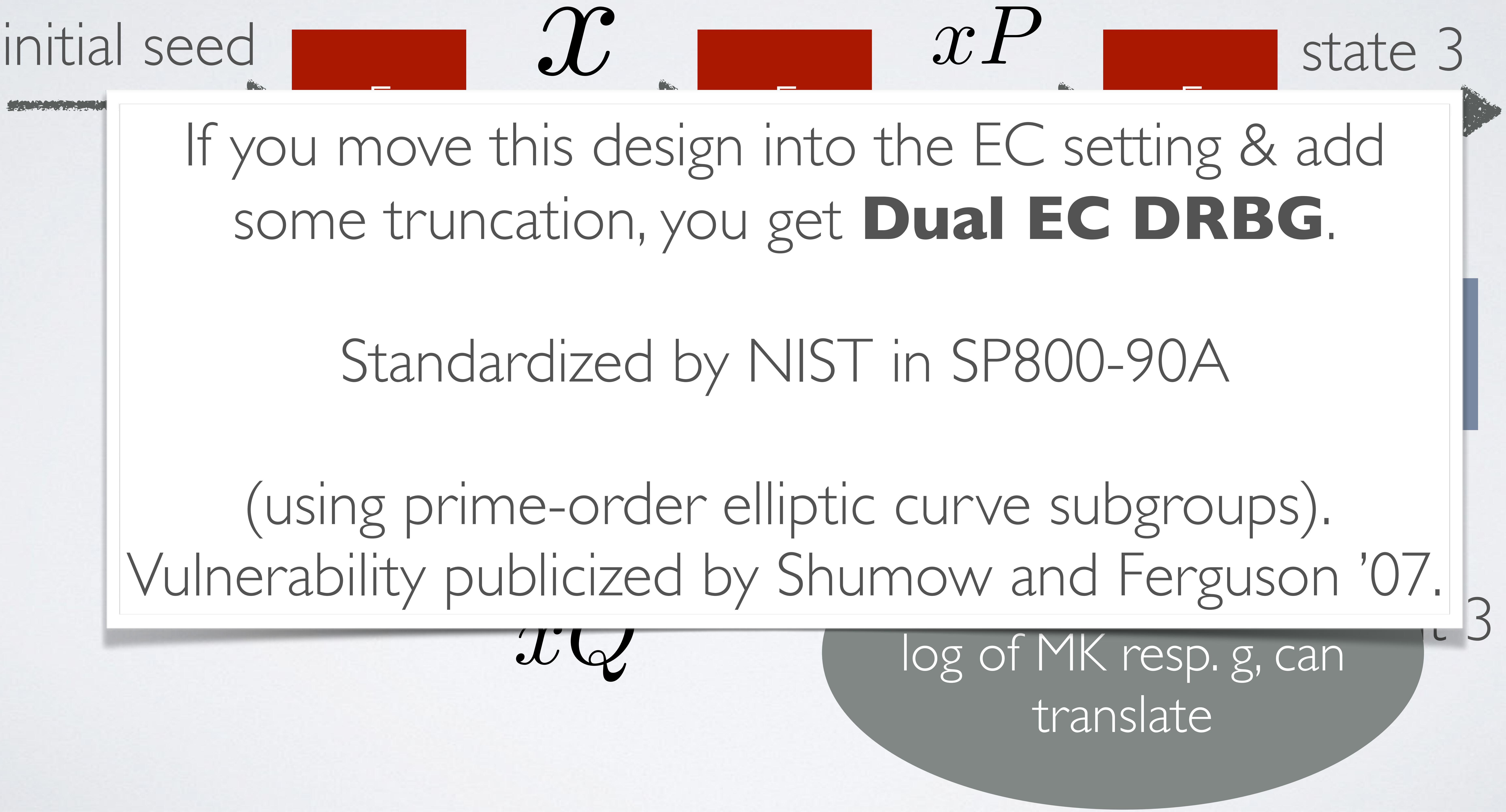
Compute next signature using k'

SETUP



initial seed
F
x
F
MK^x
F
state 3
G
G
G
g^x
If we know discrete log of MK resp. g, can translate

Dual EC DRBG



If you move this design into the EC setting & add some truncation, you get **Dual EC DRBG**.

Standardized by NIST in SP800-90A

(using prime-order elliptic curve subgroups).
Vulnerability publicized by Shumow and Ferguson ’07.

N.S.A. Able to Foil Basic Safeguards of Privacy on Web

By NICOLE PERLROTH, JEFF LARSON and SCOTT SHANE
Published: September 5, 2013 | 1466 Comments

Cryptographers have long suspected that the agency planted vulnerabilities in a standard adopted in 2006 by the National Institute of Standards and Technology and later by the International Organization for Standardization, which has 163 countries as members.

Classified N.S.A. memos appear to confirm that the fatal weakness, discovered by two Microsoft cryptographers in 2007, was engineered by the agency. The N.S.A. wrote the standard and aggressively pushed it on the international group, privately calling the effort "a challenge in finesse."

"Eventually, N.S.A. became the sole editor," the memo says.

Fast forward to 2015

Next several slides, joint work with:

Stephen Checkoway, Jacob Maskiewicz, Christina Garman, Joshua Fried, Shaanan Cohney, Nadia Heninger, Ralf-Philipp Weinmann, Eric Rescorla, Hovav Shacham

Juniper is committed to maintaining the integrity and security of our products and wanted to make customers aware of critical patched releases we are issuing today to address vulnerabilities in devices running ScreenOS® software.

During a recent internal code review, Juniper discovered unauthorized code in ScreenOS that could allow a knowledgeable attacker to gain administrative access to NetScreen® devices and to decrypt VPN connections. Once we identified these vulnerabilities, we launched an investigation into the matter, and worked to develop and issue patched releases for the latest versions of ScreenOS.

At this time, we have not received any reports of these vulnerabilities being exploited; however, we strongly recommend that customers update their systems and apply the patched releases with the highest priority.

CVE-2015-**7756**

VPN Decryption (CVE-2015-7756) may allow a knowledgeable attacker who can monitor VPN traffic to decrypt that traffic. It is independent of the first issue.

This issue affects ScreenOS 6.2.0r15 through 6.2.0r18 and 6.3.0r12 through 6.3.0r20. **No other Juniper products or versions of ScreenOS are affected by this issue.**

There is no way to detect that this vulnerability was exploited.

This issue has been assigned CVE-2015-7756.

VPN Decryption (CVE-2015-7756) may allow a knowledgeable attacker who can monitor VPN traffic to decrypt that traffic.

Vulnerable -> Patched

ScreenOS 6.3.0r20
(vulnerable)

```
2551....9585320EEAF81044F20D5503
0A035B11BECE81C785E6C933E4A8A131
F6578107....interrupt disabled a
2551....2c55e5e45edf713dc43475ef
fe8813a60326a64d9ba3d2e39cb639b0
f3b0ad10....interrupt disabled a
```

ScreenOS 6.3.0r21
(patched)

Sources: Adam Caudill, Peter Bowen, HD Moore, Ralf Phillip Weinmann

Dual EC DRBG

P-256 Weierstraß *b*

P-256 *P* x coord

P-256 field order

```
      5AC635D8AA3A93E7B3EBBD55769886BC651D06B ... C3E27D2604B
      6B17D1F2E12C4247F8BCE6E563A440F277037D812DEB33A0F4A13 ...
      FFFFFFFF00000000FFFFFFFFFFFFFFFFBCE6FAADA7179E84F3B9CAC2FC632551
bad:  9585320EEAF81044F20D55030A035B11BECE81C785E6C933E4A8A131F6578107
good: 2c55e5e45edf713dc43475effe8813a60326a64d9ba3d2e39cb639b0f3b0ad10
nist: c97445f45cdef9f0d3e05e1e585fc297235b82b5be8ff3efca67c5985201819 2
```

NIST SP 800-90A January 2012

NIST Special Publication 800-90A

Recommendation for Random Number Generation Using Deterministic Random Bit Generators

FIPS 140-2 SECURITY POLICY

Juniper Networks, Inc.

SSG 5 and SSG 20

HW P/N SSG-5 and SSG-20, FW Version ScreenOS 6.2.0

Document # 530-023728-01

FIPS Approved Algorithms

The following FIPS approved algorithms are supporte

- DSA , ECDSA Sign Verify
- SHA-1, SHA-256
- Triple-DES (CBC)

Juniper Networks SSG 5 and SSG 20 Security Policy

- AES (CBC)
- HMAC-SHA-1, HMAC-SHA-256
- RSA Sign/Verify (PKCS #1)
- ANSI X9.31 DRNG

Juniper doesn’t appear to use Dual EC…

Dual EC in ScreenOS

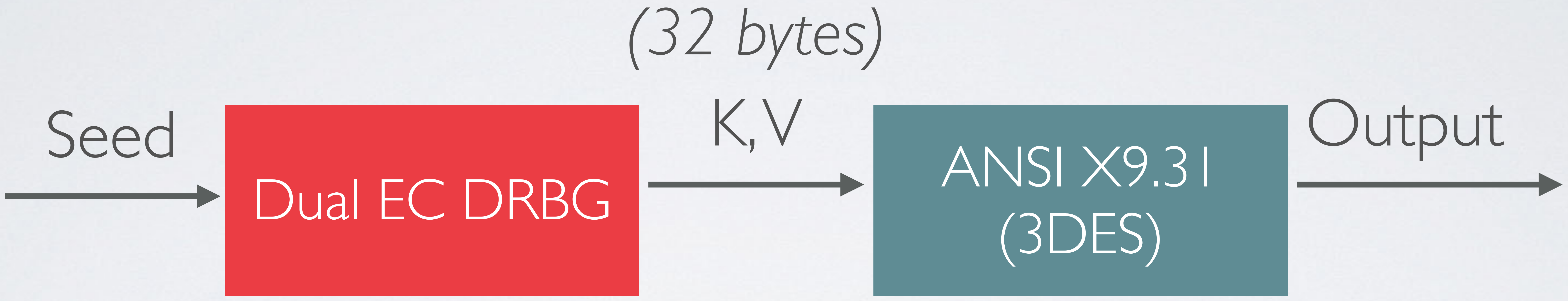
The following product families do utilize Dual_EC_DRBG, but do not use the pre-defined points cited by NIST:

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.

"ScreenOS does make use of the Dual_EC_DRBG standard, but is designed not to use Dual_EC_DRBG as its primary random number generator. ScreenOS uses it it in a way that shouldn't be vulnerable to the possible issue that has been brought to light." (2013)

RNG Cascade



This approach *should* neutralize any backdoor

```c
void prng_reseed(void) {
  blocks_generated_since_reseed = 0;
  if (dualec_generate(prng_temporary, 32) != 32)
    error_handler("FIPS ERROR: PRNG failure, unable to reseed\n", 11);
  memcpy(prng_seed, prng_temporary, 8);
  prng_output_index = 8;
  memcpy(prng_key, &prng_temporary[prng_output_index], 24);
  prng_output_index = 32;
}
void prng_generate(void) {
  int time[2];

  time[0] = 0;
  time[1] = get_cycles();
  prng_output_index = 0;
  ++blocks_generated_since_reseed;
  if (!one_stage_rng())
    prng_reseed();
  for (; prng_output_index <= 0x1F; prng_output_index += 8) {
    // FIPS checks removed for clarity
    x9_31_generate_block(time, prng_seed, prng_key, prng_block);
    // FIPS checks removed for clarity
    memcpy(&prng_temporary[prng_output_index], prng_block, 8);
  }
}
```

Calls Dual EC to fill a buffer

Credit: William Pinckaers

```c
void prng_reseed(void) {
  blocks_generated_since_reseed = 0;
  if (dualec_generate(prng_temporary, 32) != 32)
    error_handler("FIPS ERROR: PRNG failure, unable to reseed\n", 11);
  memcpy(prng_seed, prng_temporary, 8);
  prng_output_index = 8;
  memcpy(prng_key, &prng_temporary[prng_output_index], 24);
  prng_output_index = 32;
}
void prng_generate(void) {
  int time[2];

  time[0] = 0;
  time[1] = get_cycles();
  prng_output_index = 0;
  ++blocks_generated_since_reseed;
  if (!one_stage_rng())
    prng_reseed();
  for (; prng_output_index <= 0x1F; prng_output_index += 8) {
    // FIPS checks removed for clarity
    x9_31_generate_block(time, prng_seed, prng_key, prng_block);
    // FIPS checks removed for clarity
    memcpy(&prng_temporary[prng_output_index], prng_block, 8);
  }
}
```

"Runs" ANSI generator in place

Credit: William Pinckaers

```c
void prng_reseed(void) {
  blocks_generated_since_reseed = 0;
  if (dualec_generate(prng_temporary, 32) != 32)
    error_handler("FIPS ERROR: PRNG failure, unable to reseed\n", 11);
  memcpy(prng_seed, prng_temporary, 8);
  prng_output_index = 8;
  memcpy(prng_key, &prng_temporary[prng_output_index], 24);
  prng_output_index = 32;
}
void prng_generate(void) {
  int time[2];

  time[0] = 0;
  time[1] = get_cycles();
  prng_output_index = 0;
  ++blocks_generated_since_reseed;
  if (!one_stage_rng())
    prng_reseed();
  for (; prng_output_index <= 0x1F; prng_output_index += 8) {
    // FIPS checks removed for clarity
    x9_31_generate_block(time, prng_seed, prng_key, prng_block);
    // FIPS checks removed for clarity
    memcpy(&prng_temporary[prng_output_index], prng_block, 8);
  }
}
```

Generates Dual EC output
Sets prng_output_index = 32

Credit: William Pinckaers

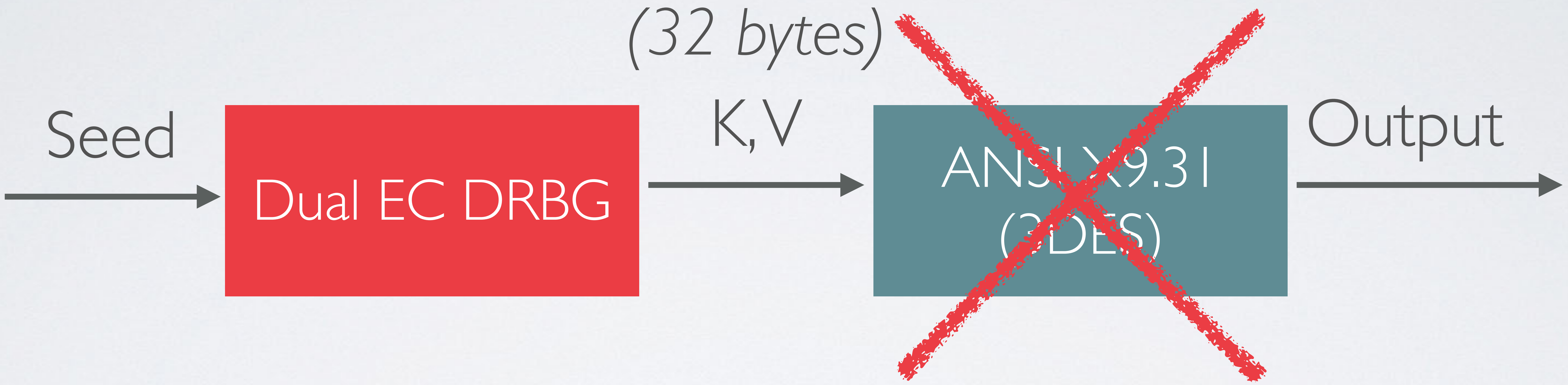
```c
void prng_reseed(void) {
  blocks_generated_since_reseed = 0;
  if (dualec_generate(prng_temporary, 32) != 32)
    error_handler("FIPS ERROR: PRNG failure, unable to reseed\n", 11);
  memcpy(prng_seed, prng_temporary, 8);
  prng_output_index = 8;
  memcpy(prng_key, &prng_temporary[prng_output_index], 24);
  prng_output_index = 32;
}
void prng_generate(void) {
  int time[2];

  time[0] = 0;
  time[1] = get_cycles();
  prng_output_index = 0;
  ++blocks_generated_since_reseed;
  if (!one_stage_rng())
    prng_reseed();
  for (; prng_output_index <= 0x1F; prng_output_index += 8) {
    // FIPS checks removed for clarity
    x9_31_generate_block(time, prng_seed, prng_key, prng_block);
    // FIPS checks removed for clarity
    memcpy(&prng_temporary[prng_output_index], prng_block, 8);
  }
}
```

ANSI generator is never run.
Dual EC output emitted.

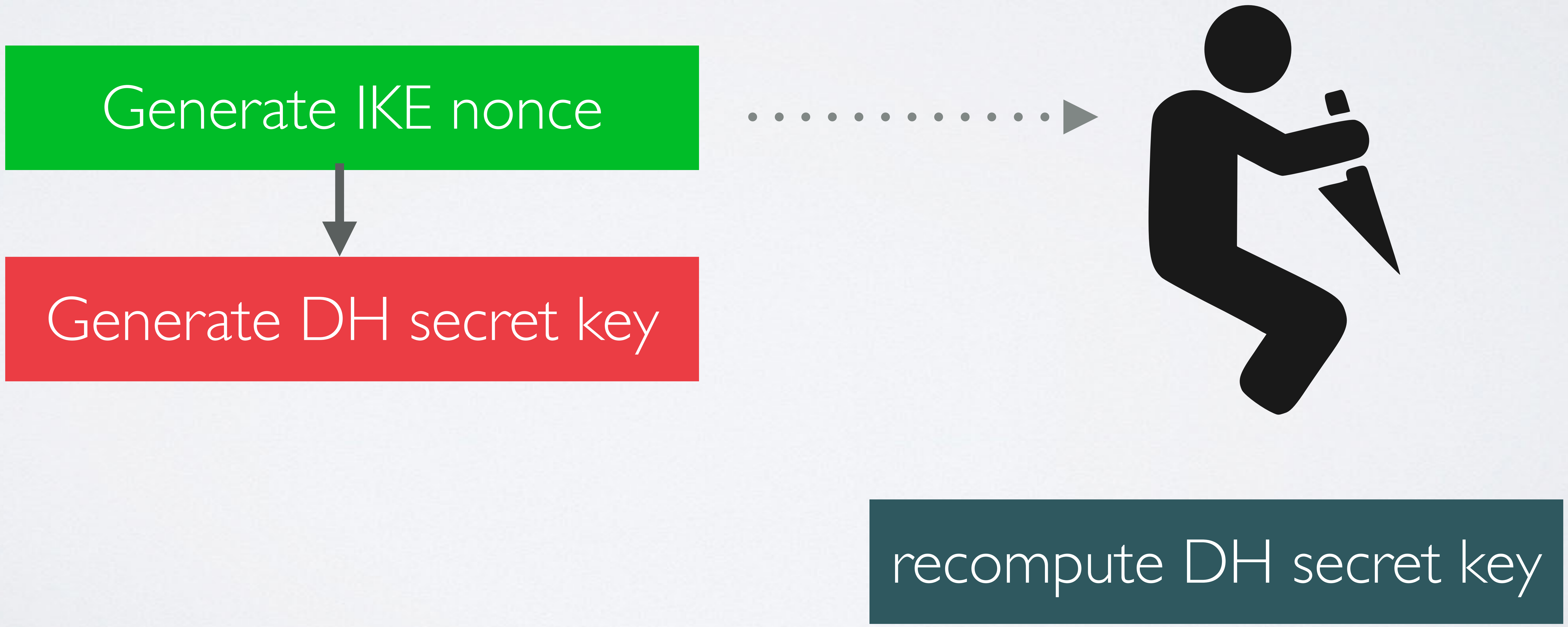
Credit: William Pinckaers

Revised Cascade



Exploiting IKE (Ideal)

- Like many protocols, outputs *nonces*
  - In ScreenOS 6.1 (pre-Dual EC): 20 bytes
    In ScreenOS 6.2 (with Dual EC): 32 bytes
    (>= 28 bytes is sufficient to recover Dual EC state)



Exploiting IKE (Ideal)

- Li

This is (apparently) <u>not</u> what Juniper does

Generate DH secret key

recompute DH secret key

Exploiting IKE (ScreenOS 6.1)

- All versions of ScreenOS appear to generate key first Nonce second
  - Even with Dual EC, hinders the attack

Generate DH secret key

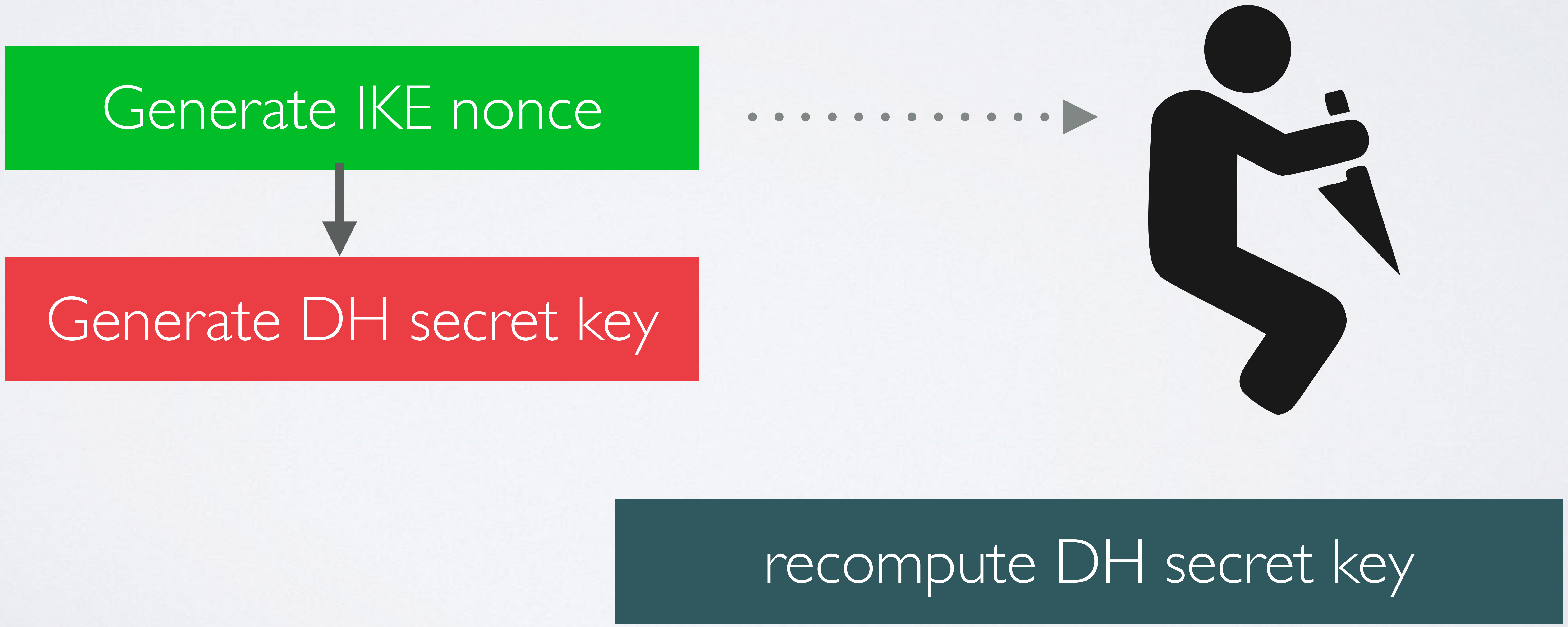
↓

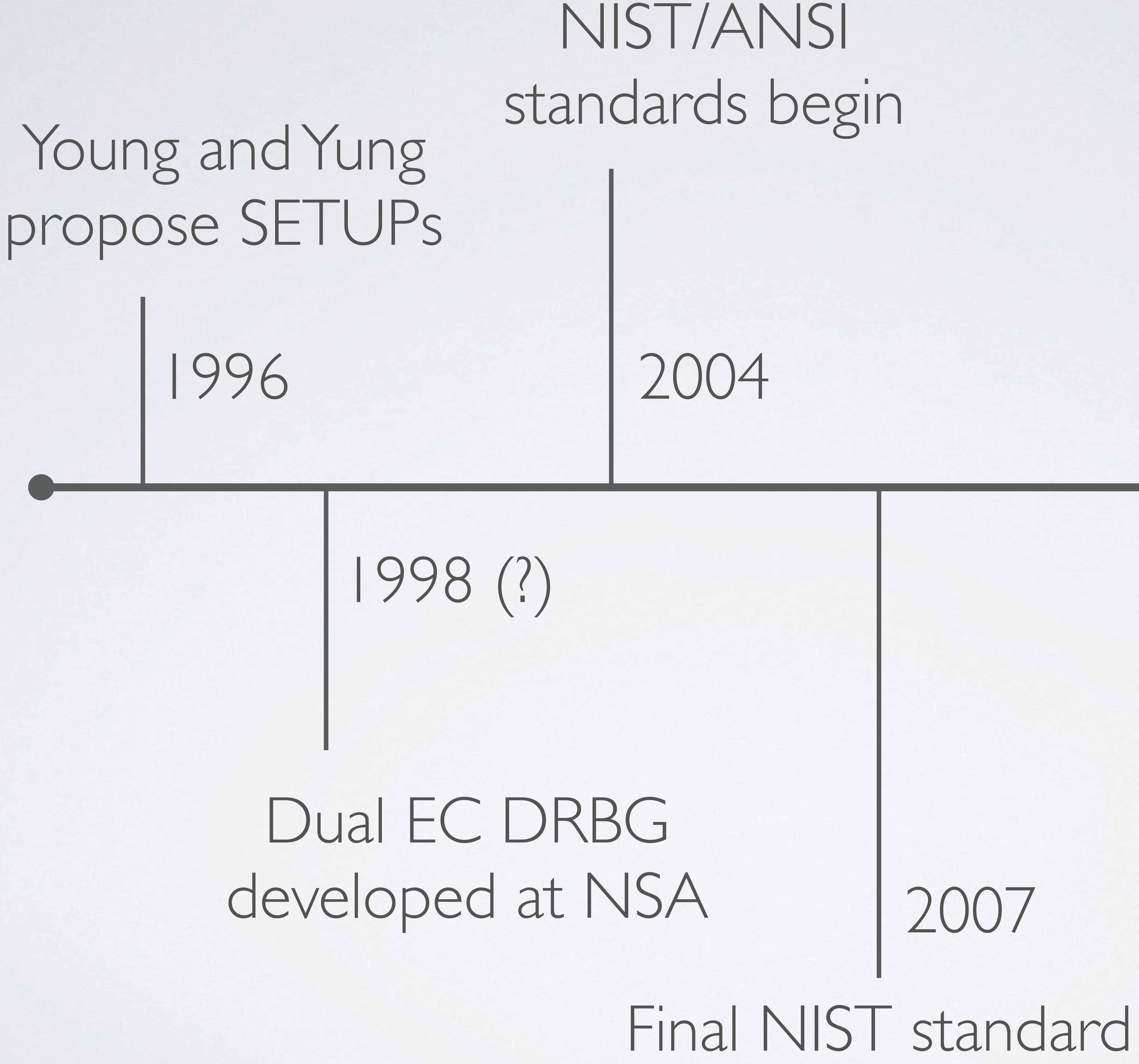
Generate IKE nonce

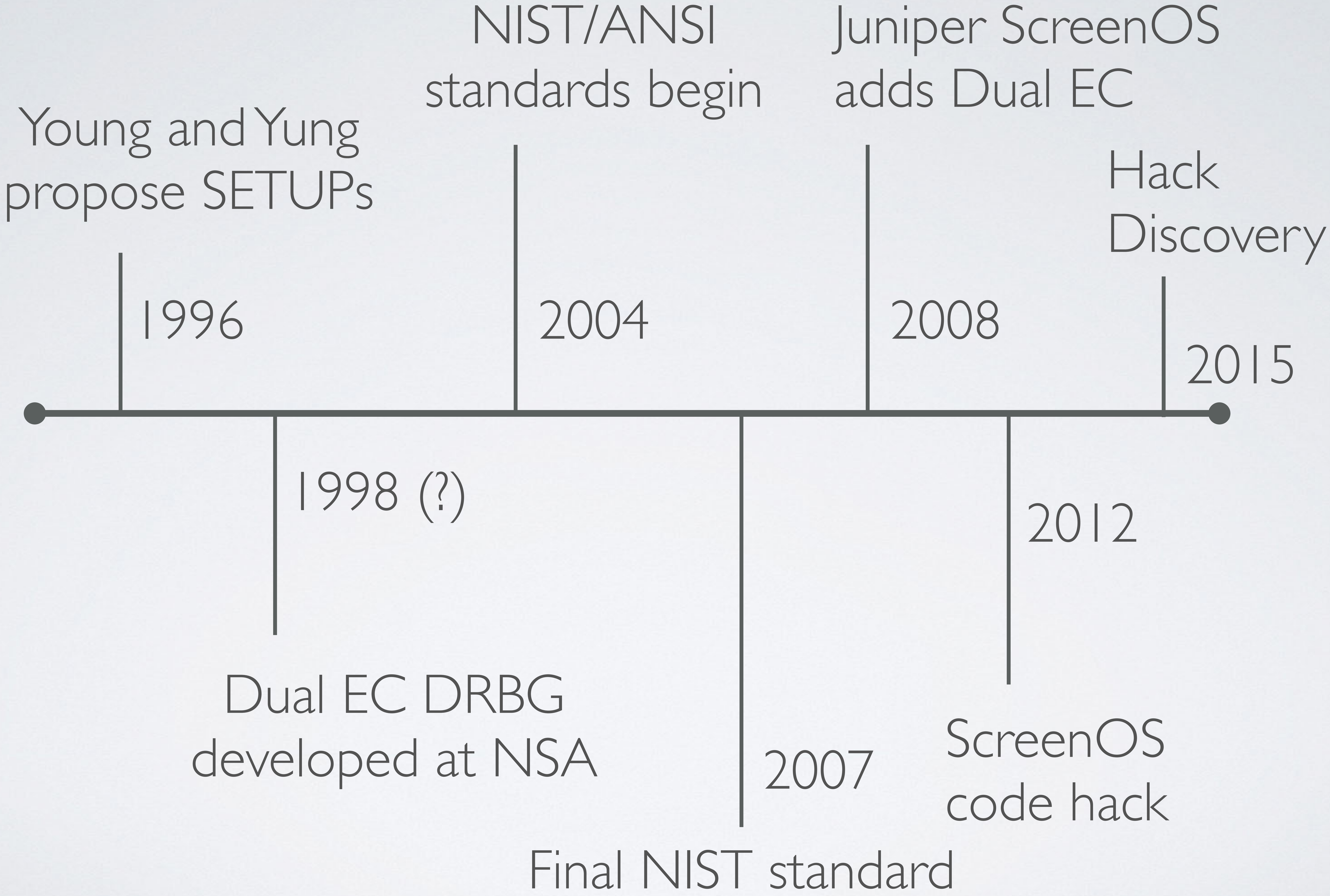
(must wait for next handshake)

Exploiting IKE (ScreenOS 6.2)

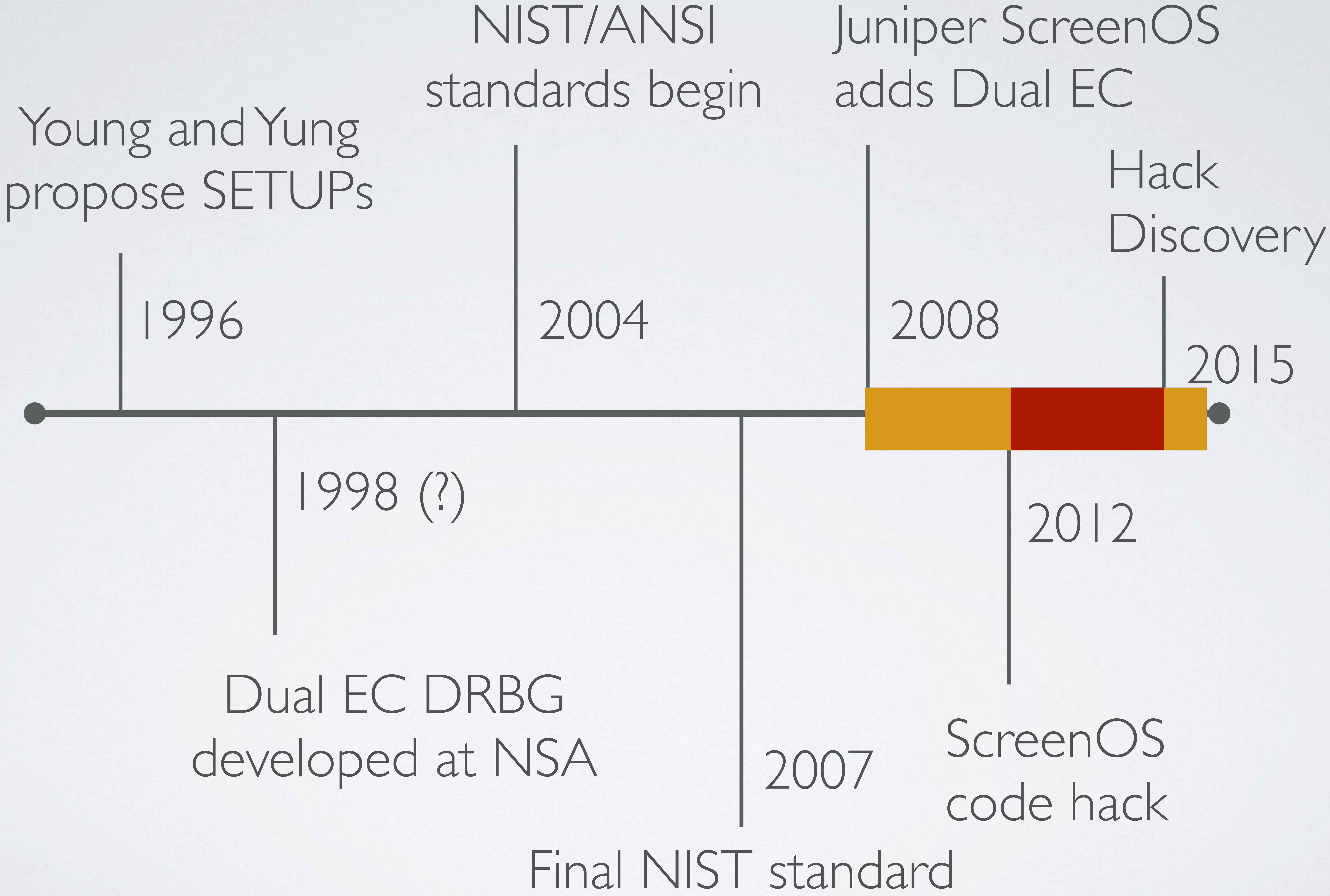
- ScreenOS 6.2 (the version that adds Dual EC)
  - Adds a *nonce pre-generation queue*
  - Effectively ensures that nonces are always generated first




NIST/ANSI
standards begin
Young and Yung
propose SETUPs
1996
2004
1998 (?)
Dual EC DRBG
developed at NSA
2007
Final NIST standard



Young and Yung propose SETUPs
1996
Dual EC DRBG developed at NSA
1998 (?)
NIST/ANSI standards begin
2004
Final NIST standard
2007
Juniper ScreenOS adds Dual EC
2008
ScreenOS code hack
2012
Hack Discovery
2015


Young and Yung
propose SETUPs
1996
Dual EC DRBG
developed at NSA
1998 (?)
NIST/ANSI
standards begin
2004
Final NIST standard
2007
Juniper ScreenOS
adds Dual EC
2008
ScreenOS
code hack
2012
Hack
Discovery
2015

Who was the target?

U.S. Department of Justice

Federal Bureau of Investigation

Washington, D.C. 20535

August 15, 2019

MUCKROCK NEWS
DEPT MR 77296
411A HIGHLAND AVENUE
SOMERVILLE, MA 02144

FOIPA Request No.: 1442628-000
Subject: Office of Personnel Management Security Breach
(2015 – 2015)

Dear Mr. Green:

This responds to your Freedom of Information/Privacy Acts (FOIPA) request. Please see the paragraphs below for relevant information specific to your request as well as the enclosed FBI FOIPA Addendum for standard responses applicable to all requests.

The FBI has completed its search for records responsive to your request. The material you requested is located in an investigative file which is exempt from disclosure pursuant to 5 U.S.C. § 552(b)(7)(A). 5 U.S.C. § 552(b)(7)(A) exempts from disclosure:

The records responsive to your request are law enforcement records; there is a pending or prospective law enforcement proceeding relevant to these responsive records, and release of the information could reasonably be expected to interfere with enforcement proceedings. Therefore, your request is being administratively closed. For a further explanation of this exemption, see the enclosed Explanation of Exemptions.

Who was the target?

OPM/OTM/NM/FIS Juniper Netscreen Support Consolidation - Bill of Materials FY13								
Support Coverage	**Product**	**Serial#**	**Start Date**	**End Date**	**State**	**Zip Code**	**Unit Cost**	**Extended Cost**
SVC-ND-ISG2000	NS-ISG-2000-SK1	0079062005000001	03-Feb-2014	02-Feb-2015	DC	20415		
SVC-ND-ISG2000	NS-ISG-2000	0079072009000003	03-Feb-2014	02-Feb-2015	DC	20415		
SVC-ND-ISG2000	NS-ISG-2000	0079082007000018	03-Feb-2014	02-Feb-2015	DC	20415		
SVC-ND-ISG2000	NS-ISG-2000	0079082007000393	03-Feb-2014	02-Feb-2015	DC	20415		
SVC-ND-ISG1000	NS-ISG-1000	0133032010000094	26-Jun-2013	02-Feb-2015	DC	20415		
SVC-EXT-WAR-ISG1000	NS-ISG-1000	0133032010000094	26-Jun-2012	25-Jun-2013	DC	20415		
SVC-ND-ISG1000	NS-ISG-1000	0133042010000041	26-Jun-2013	02-Feb-2015	DC	20415		
SVC-EXT-WAR-ISG1000	NS-ISG-1000	0133042010000041	26-Jun-2012	25-Jun-2013	DC	20415		
SVC-ND-ISG1000	NS-ISG-1000-SK1	0133042010000048	26-Jun-2013	02-Feb-2015	DC	20415		
SVC-EXT-WAR-ISG1000	NS-ISG-1000-SK1	0133042010000048	26-Jun-2012	25-Jun-2013	DC	20415		
SVC-ND-SSG140	SSG-140-SH	0185012008000001	26-Jun-2013	02-Feb-2015	DC	20415		
SVC-EXT-WAR-SSG140	SSG-140-SH	0185012008000001	26-Jun-2012	25-Jun-2013	DC	20415		
SVC-ND-SSG140	SSG-140-SH	0185012008000129	26-Jun-2013	02-Feb-2015	DC	20415		
SVC-EXT-WAR-SSG140	SSG-140-SH	0185012008000129	26-Jun-2012	25-Jun-2013	DC	20415		
SVC-ND-SSG140	SSG-140-SH	0185022008001383	26-Jun-2013	02-Feb-2015	DC	20415		
SVC-EXT-WAR-SSG140	SSG-140-SH	0185022008001383	26-Jun-2012	25-Jun-2013	DC	20415		
SVC-ND-SSG140	SSG-140-SH	0185032008000185	26-Jun-2013	02-Feb-2015	DC	20415		
SVC-EXT-WAR-SSG140	SSG-140-SH	0185032008000185	26-Jun-2012	25-Jun-2013	DC	20415		

Ok, how might this have been exploited?

- **If the attacker is not a U.S. agency, this would require some means to gain network perspective**
  - (Passive access to network traffic)
  - This is relatively hard to do for non-US agencies
  - Idea: look for incidents of network traffic re-routing in the 2012-2015 period

One final note

- **In 2013, researchers noted the first durable BGP "MITM" interception events**
  - These are BGP events in which traffic is misdirected via one path, and reaches its destination via a different return path
  - These events were not detected before 2013, and have never been explained by any concrete software flaws
  - These deserve some more scrutiny

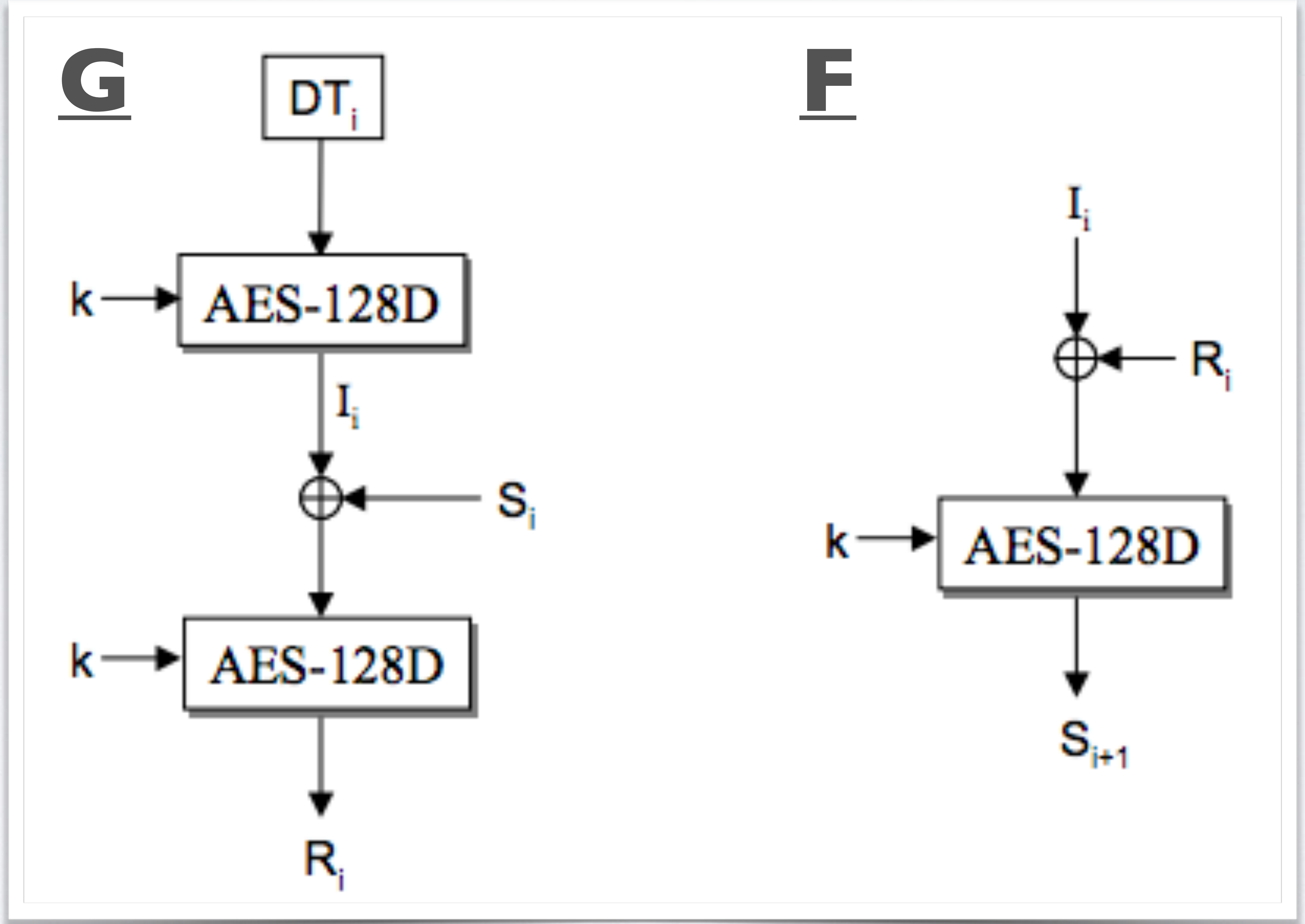
Question:

Let's assume the Dual EC DRBG flaws were deliberate, not an accident.

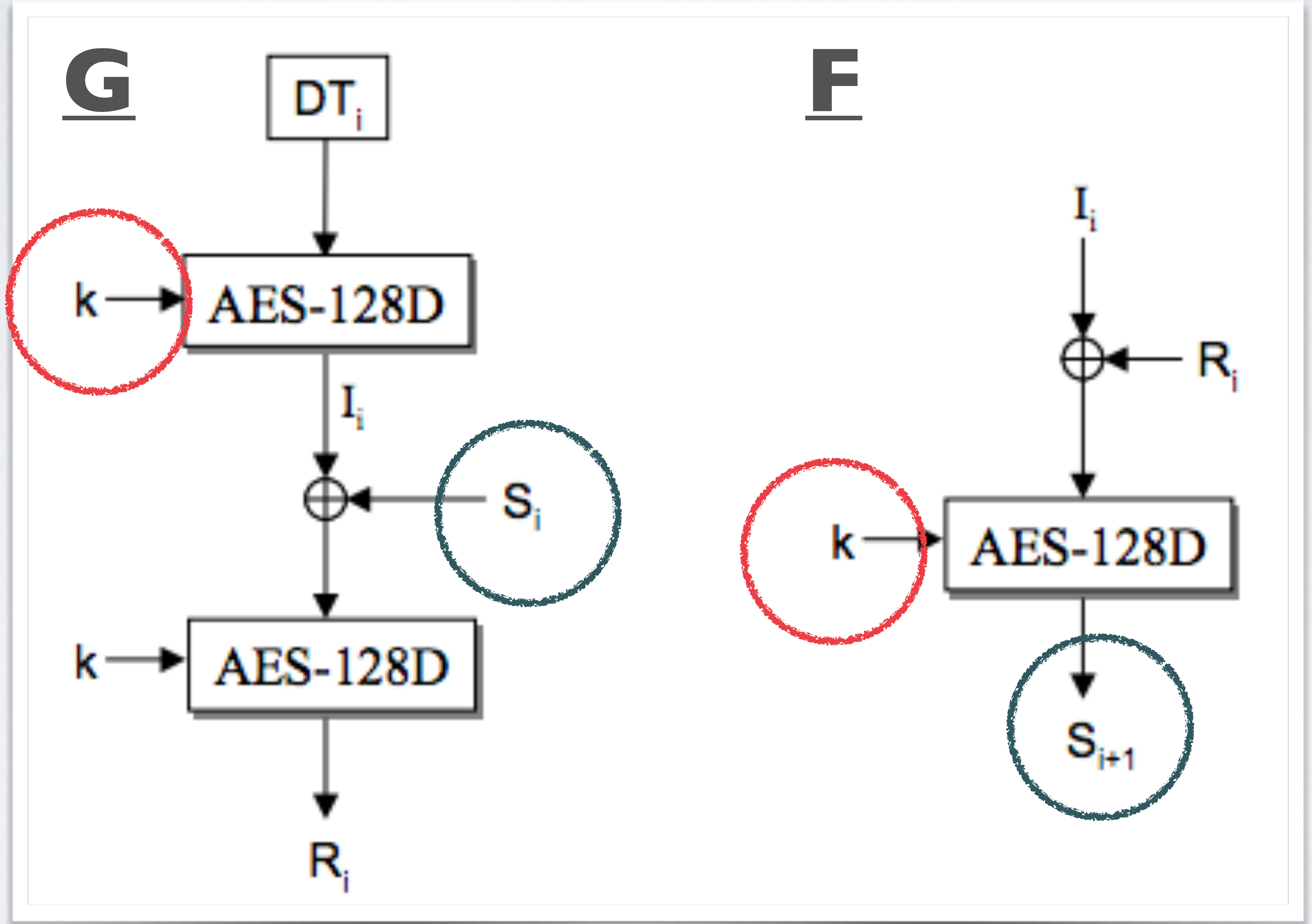
Let's assume that there is exists policy to promote vulnerabilities in VPN devices.

How would you implement kleptographic systems before SETUPs?

ANSI X9.31



ANSI X9.31



Attacking ANSI X9.31

- **Most common DRBG (PRG) in FIPS devices**
  - Well-known vulnerability: Kelsey, Schneier, Wagner and Hall
    - Attacker who knows the key **k** (but not the seed **S**) can recover internal generator state from 16-32 out bytes
    - Key is never updated
    - Standard says <u>nothing about this</u>
    - <u>To an attacker without knowledge of **k**, output is indistinguishable from random</u>

Security Target for the Fortinet FortiGate™-200B and 620B Unified Threat Management Solution and FortiOS 4.0 CC Compliant Firmware: EAL4+

The storage area for private cryptographic keys, plaintext cryptographic keys and all other critical cryptographic security parameters is a flash RAM device. Zeroization of these storage areas occurs when the Security Administrator executes a factory reset or enables FIPS-CC mode. At these times, all non-hard-coded keys and critical security parameters are zeroized by lifting the voltage from the bits comprising the key, which has the same effect as overwriting the storage area with zeroes. The hard-coded keys are ANSI X9.31 RNG AES Key, Firmware update key, configuration integrity key, configuration backup key.

Prepared by:

Electronic Warfare Associates-Canada, Ltd.
55 Metcalfe St., Suite 1600
Ottawa, Ontario
K1P 6L5

Exploiting FortiOS

- **Reverse-engineered FortiOS implementation**
  - With Shaanan Cohney and Nadia Heninger
  - Not a trivial attack to implement: requires guessing a microsecond-level timestamp value updated at each block
  - By adjusting granularity of this timer, can make the attack cost 2^40 or 2^50 AES operations (and up)
  - Many optimizations. Full recovery of Diffie-Hellman private keys from a protocol transcript in about 15 seconds

Summing up

- **Catastrophic RNG vulnerabilities in 2 major VPN device manufacturers during the same time period**
  - Hanlon's razor: "Never attribute to **malice** that which is adequately explained by **stupidity**"
  - Heinlein's Razor: "Never attribute to **malice** that which can be adequately explained by **stupidity**, *but don't rule out malice*."

So how do we fix this?

We can't do everything

- **Basic problem: if adversary has unlimited control over device implementation, we lose**
  - E.g., perfectly correct implementation +
    exfiltrate 256 bit PRG state through some timing channel
  - But in the main, the adversary is constrained by two factors:
    1. Complexity of the modifications (code does get reviewed!)
    2. Effect on the protocol transcript (can be monitored)
  - <u>This explains why corrupted RNG designs are so popular</u>

1. Build more resilient protocols

- **Example: PSK in IKEv1**
  - PSK is fed into the KDF
  - If PSK is high entropy, devices not exploitable!

1. Build more resilient protocols

- **Example: PSK in IKEv1**
  - PSK is fed into the KDF
  - If PSK is high entropy, devices not exploitable
- **Example: PSK in IKEv2**
  - PSK is not fed into the KDF
  - Devices may be exploitable!

2. Replace FIPS validation

- **FIPS validation <u>does not work</u>**
  - Each of the devices I've discussed went through high-level FIPS (CMVP) validation, some at high EAL levels!
  - All the preceding vulnerabilities should have been caught
  - FIPS validation == alg tests + compliance
  - Worse, the FortiOS hard-coded key <u>was a testing key</u>
    - Why is there a testing key in the device?
    - Because FIPS mandates runtime tests!

2a. Whole-protocol evaluation

- **Validate devices by speaking their language**
  - Rather than testing individual algorithms, run live tests with the device
  - Protocol should complete correctly with a testing endpoint
  - Now:
    - 1. Have device prove the correctness of its protocols using efficient 2PC (many challenges!)
    - 2. Fuzz firmware to identify hard-coded parameters

2b. Full formal verification

- **Formally verify the entire DRBG stack**
  - Joint work with Andrew Appel, Katherine Ye, Lennart Beringer: developed a formal proof of security in Coq/FCF
    - **Step 1:** Machine-prove that the NIST HMAC-DRBGs are actually PRGs
    - **Step 2:** Machine-prove that mbed-TLS (C) stack implements the specification (including HMAC and SHA — already done)
    - **Step 3**: Link the proofs together

3. Develop systems that can survive malice

- **Assume some trusted component, use this to ensure the rest of the implementation is safe**
  - For example, an offline or online "watchdog" that can evaluate the protocol or monitor it for malice
  - Russel *et al.*
  - This is particularly important as more of our manufacturing infrastructure is in jurisdictions not under out control

This should drive our research

- **Mostly it doesn't. But some notable exceptions:**
  - Algorithm Substitution Attacks (Bellare, Paterson, Rogaway)
  - Kleptography (Young, Yung)
  - Formal Treatments of RNGs (Dodis *et al.)*
  - CPA-security via trusted components (Russel et al.)
  - *Formal Verification Approaches (INRIA, MSR, Princeton)*


I WANT TO
BELIEVE