On the Practical Exploitability of Dual EC DRBG in TLS Implementations

Stephen Checkoway, Matthew Fredrikson, Ruben Niederhagen, Adam Everspaugh, Matthew Green, Tanja Lange, Thomas Ristenpart,
Daniel J. Bernstein, Jake Maskiewicz, Hovav Shacham

August 2, 2014

Random numbers are important

- Cryptography needs random numbers to generate long-term secret keys for encryption and signatures.
- Many schemes expect random (or pseudorandom) numbers, e.g.
  - ephemeral keys for DH key exchange,
  - nonces for digital signatures,
  - nonces in authenticated encryption.
- Nonce reuse can reveal long-term secret keys (e.g. PlayStation disaster)
- DSA/ECDSA are so touchy that biased nonces are enough to break them.

Random numbers are important to the NSA

- Cryptography needs random numbers to generate long-term secret keys for encryption and signatures.
- Many schemes expect random (or pseudorandom) numbers, e.g.
  - ephemeral keys for DH key exchange,
  - nonces for digital signatures,
  - nonces in authenticated encryption.
- Nonce reuse can reveal long-term secret keys (e.g. PlayStation disaster)
- DSA/ECDSA are so touchy that biased nonces are enough to break them.

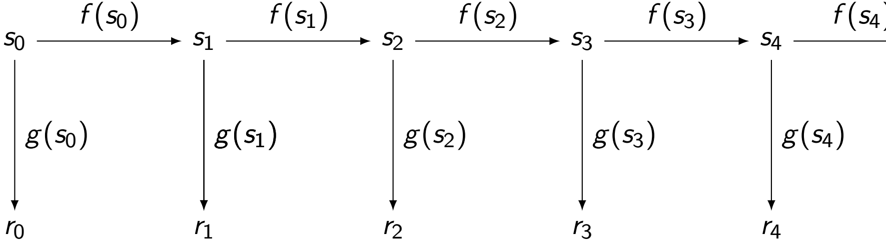
Snowden at SXSW:

> *[..] we know that these encryption algorithms we are using today work typically it is the random number generators that are attacked as opposed to the encryption algorithms themselves.*

Pseudo-random-number generators

Crypto libraries expand short seed into long stream of random bits.
Random bits are used as secret keys, DSA nonces, . . .

The usual structure, starting from short seed s_1:



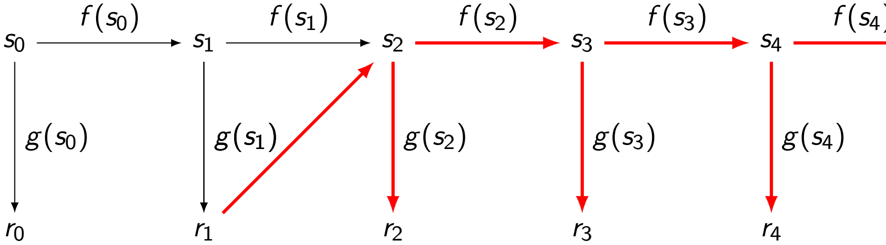
XXX's mission: Predict the "random" output bits.

1. Create protocols that directly output r_n for some reason.

Pseudo-random-number generators

Crypto libraries expand short seed into long stream of random bits.
Random bits are used as secret keys, DSA nonces, ...

The usual structure, starting from short seed s_1:



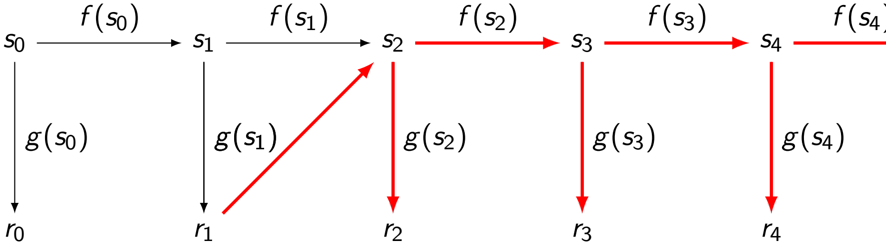
XXX's mission: Predict the "random" output bits.

1. Create protocols that directly output r_n for some reason.
2. Design f, g with back door from r_n to s_{n+1}: i.e., get $f(s)$ from $g(s)$.

Pseudo-random-number generators

Crypto libraries expand short seed into long stream of random bits.
Random bits are used as secret keys, DSA nonces, . . .

The usual structure, starting from short seed s_1:



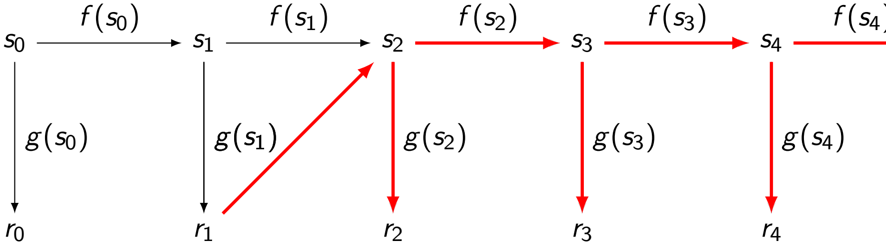
XXX's mission: Predict the "random" output bits.

1. Create protocols that directly output r_n for some reason.
2. Design f, g with back door from r_n to s_{n+1}: i.e., get $f(s)$ from $g(s)$.
3. Standardize this design of f, g.

Pseudo-random-number generators

Crypto libraries expand short seed into long stream of random bits.
Random bits are used as secret keys, DSA nonces, ...

The usual structure, starting from short seed s_1:



XXX's mission: Predict the "random" output bits.

1. Create protocols that directly output r_n for some reason.
2. Design f, g with back door from r_n to s_{n+1}: i.e., get $f(s)$ from $g(s)$.
3. Standardize this design of f, g.
4. Convince users to switch to this design: e.g., publish "security proof".

Elliptic curves

If P, Q are random points on a strong elliptic curve then it's hard to predict sP given sQ.

But if we know $P = kQ$ then it's easy: $sP = ksQ$.

Let's choose random Q, random k, define $P = kQ$.
Standardize this P; Q; $f(s) = sP$; $g(s) = sQ$.

Elliptic curves

If P, Q are random points on a strong elliptic curve
then it's hard to predict sP given sQ.

But if we know $P = kQ$ then it's easy: $sP = ksQ$.

Let's choose random Q, random k, define $P = kQ$.
Standardize this P; Q; $f(s) = sP$; $g(s) = sQ$.

Wait a minute:
Curve points (x, y) don't look like random strings.
They satisfy public curve equation: $y^2 = x^3 - 3x + \text{constant}$.
This won't pass public review.

Elliptic curves

If P, Q are random points on a strong elliptic curve
then it's hard to predict sP given sQ.

But if we know $P = kQ$ then it's easy: $sP = ksQ$.

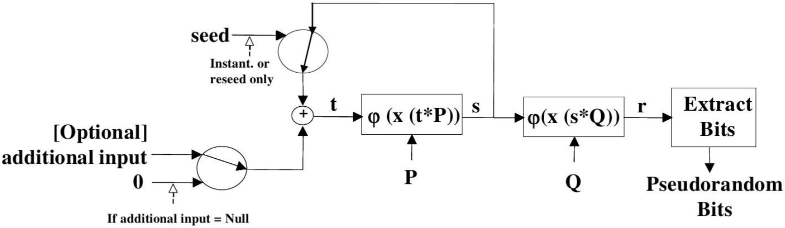
Let's choose random Q, random k, define $P = kQ$.
Standardize this P; Q; $f(s) = sP$; $g(s) = sQ$.

Wait a minute:
Curve points (x, y) don't look like random strings.
They satisfy public curve equation: $y^2 = x^3 - 3x + \text{constant}$.
This won't pass public review.

Solution: Let's throw away y and some bits of x.
Define $f(s) = x(sP)$, $g(s) = \phi(x(sQ))$ where ϕ omits 16 bits.
Not a big computation for us to recover sQ from $g(s)$.

DUAL_EC RNG: history part I

Earliest public source (?) June 2004, draft of ANSI X9.82:



Extract gives all but the top 16 bits ⇒ about 2^{15} points sQ match given string.

Claim:

Dual_EC_DRBG is based on the following hard problem, sometimes known as the "elliptic curve discrete logarithm problem" (ECDLP): given points P and Q on an elliptic curve of order n, find a such that $Q = aP$.

DUAL_EC RNG: common public history part II

Various public warning signals:

- Gjøsteen (March 2006): output sequence is biased.
  "While the practical impact of these results are modest, it is hard to see how these flaws would be acceptable in a pseudo-random bit generator based on symmetric cryptographic primitives. They should not be accepted in a generator based on number-theoretic assumptions."
- Brown (March 2006): security "proof"
  "This proof makes essential use of Q being random." If d with $dQ = P$ is known then $dR_i = S_{i+1}$, concludes that there might be distinguisher.
- Sidorenko & Schoenmakers (May 2006): output sequence is even more biased.
  Answer: Too late to change, already implemented.
- Shumow & Ferguson (August 2007): Backdoor if d is known.
- NIST SP800-90 gets appendix about choosing points verifiably at random,
  but requires use of standardized P, Q for FIPS-140 validation.

September 2013: NSA Bullrun program

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

September 2013: NSA Bullrun program

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

NYT:

> *the NSA had inserted a back door into a 2006 standard adopted by NIST [..] called the Dual EC DRBG standard.*

September 2013: NSA Bullrun program

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

NYT:

> *the NSA had inserted a back door into a 2006 standard adopted by NIST [..] called the Dual EC DRBG standard.*

. . . but surely nobody uses that!?!

September 2013: NSA Bullrun program

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

NYT:

> *the NSA had inserted a back door into a 2006 standard adopted by NIST [..] called the Dual EC DRBG standard.*

. . . but surely nobody uses that!?!

NIST's DRBG Validation List: more than 70 validations of Dual_EC_DRBG;
RSA's BSAFE has Dual_EC_DRBG enabled as default,.

September 2013: NSA Bullrun program

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

NYT:

> *the NSA had inserted a back door into a 2006 standard adopted by NIST [..] called the Dual EC DRBG standard.*

. . . but surely nobody uses that!?!

NIST's DRBG Validation List: more than 70 validations of Dual_EC_DRBG;
RSA's BSAFE has Dual_EC_DRBG enabled as default,.

NIST re-opens discussions on SP800.90; recommmends against using Dual_EC.
RSA suggests changing default in BSAFE.

21 April 2014 NIST removes Dual EC from the standard.

How expensive is using the backdoor?

Rereading the standard:

" $x(A)$ is the x-coordinate of the point A on the curve, given in affine coordinates. An implementation may choose to represent points internally using other coordinate systems; for instance, when efficiency is a primary concern. In this case, a point shall be translated back to affine coordinates before $x()$ is applied."

How expensive is using the backdoor?

Rereading the standard:

" $x(A)$ is the x-coordinate of the point A on the curve, given in affine coordinates. An implementation may choose to represent points internally using other coordinate systems; for instance, when efficiency is a primary concern. In this case, a point shall be translated back to affine coordinates before $x()$ is applied."

Given $r_i = \varphi(x(s_i Q))$, $r_{i+1} = \varphi(x(s_{i+1} Q))$, and NSA backdoor $d = log_P(Q)$.

1. Expand r_i to candidate $Q_i = s_i Q$, [50% chance; if fail move on to next candidate]
2. compute candidate $P_{i+1} = dQ_i$ and candidate $s_{i+1} = x(P_{i+1})$
3. check, $\varphi(x(s_{i+1} Q))$ against r_{i+1}. [if fail, goto 1.; else most likely done!]

How expensive is using the backdoor?

Rereading the standard:

" $x(A)$ is the x-coordinate of the point A on the curve, given in affine coordinates. An implementation may choose to represent points internally using other coordinate systems; for instance, when efficiency is a primary concern. In this case, a point shall be translated back to affine coordinates before $x()$ is applied."

Given $r_i = \varphi(x(s_i Q))$, $r_{i+1} = \varphi(x(s_{i+1} Q))$, and NSA backdoor $d = log_P(Q)$.

1. Expand r_i to candidate $Q_i = s_i Q$, [50% chance; if fail move on to next candidate]
2. compute candidate $P_{i+1} = dQ_i$ and candidate $s_{i+1} = x(P_{i+1})$
3. check, $\varphi(x(s_{i+1} Q))$ against r_{i+1}. [if fail, goto 1.; else most likely done!]

Initial timings on i7 M620 Core

missing	16 bits	24 bits	32 bits
1 core	20s	85m	15d4h

How expensive is using the backdoor?

Rereading the standard:
" $x(A)$ is the x-coordinate of the point A on the curve, given in affine coordinates. An implementation may choose to represent points internally using other coordinate systems; for instance, when efficiency is a primary concern. In this case, a point shall be translated back to affine coordinates before $x()$ is applied."

Given $r_i = \varphi(x(s_i Q))$, $r_{i+1} = \varphi(x(s_{i+1} Q))$, and NSA backdoor $d = log_P(Q)$.

1. Expand r_i to candidate $Q_i = s_i Q$, [50% chance; if fail move on to next candidate]
2. compute candidate $P_{i+1} = dQ_i$ and candidate $s_{i+1} = x(P_{i+1})$
3. check, $\varphi(x(s_{i+1} Q))$ against r_{i+1}. [if fail, goto 1.; else most likely done!]

Initial timings on i7 M620 Core

missing	16 bits	24 bits	32 bits
1 core	20s	85m	15d4h

How expensive is using the backdoor?

Rereading the standard:
" $x(A)$ is the x-coordinate of the point A on the curve, given in affine coordinates. An implementation may choose to represent points internally using other coordinate systems; for instance, when efficiency is a primary concern. In this case, a point shall be translated back to affine coordinates before $x()$ is applied."

Given $r_i = \varphi(x(s_i Q))$, $r_{i+1} = \varphi(x(s_{i+1} Q))$, and NSA backdoor $d = log_P(Q)$.

1. Expand r_i to candidate $Q_i = s_i Q$, [50% chance; if fail move on to next candidate]
2. compute candidate $P_{i+1} = dQ_i$ and candidate $s_{i+1} = x(P_{i+1})$
3. check, $\varphi(x(s_{i+1} Q))$ against r_{i+1}. [if fail, goto 1.; else most likely done!]

From the standard:
"For performance reasons, the value of outlen should be set to the maximum value as provided in Table 4."

Initial timings on i7 M620 Core

missing	16 bits	24 bits	32 bits
1 core	20s	85m	15d4h