Kleptography, Dual EC

Applied Cryptography

Tanja Lange

(many slides from Ruben Niederhagen,
some by Moti Yung and Milou Antheunisse)

Technische Universiteit
Eindhoven
University of Technology

17 December 2010

Capstone Project

- NSA program, public since 1993, this is during the "crypto wars".
- Standards for government, also planned for commercial and private use.
- Advertised as making strong cryptography available, no risk to security of country and citizens.

Capstone Project

- NSA program, public since 1993, this is during the "crypto wars".
- Standards for government, also planned for commercial and private use.
- Advertised as making strong cryptography available, no risk to security of country and citizens.
- New designs (and acronyms):
  - Escrowed Encryption Standard (EES)
  - Law Enforcement Access Field (LEAF)
- Key escrow highly controversial: can be used to spy on citizens and adds weakness to system.

Capstone Project

- NSA program, public since 1993, this is during the "crypto wars".
- Standards for government, also planned for commercial and private use.
- Advertised as making strong cryptography available, no risk to security of country and citizens.
- New designs (and acronyms):
  - Escrowed Encryption Standard (EES)
  - Law Enforcement Access Field (LEAF)
- Key escrow highly controversial: can be used to spy on citizens and adds weakness to system.
- Most prominent example: Clipper chip.
- Matt Blaze showed how to circumvent escrow part; project stopped.

[Photo by Travis Goodspeed]

Kleptography — Introduction

What is Kleptography?

- Kleptography is the study of stealing information securely (exclusively), efficiently, and subliminally (unnoticeably).
- Stealing from your most trusted "hardware protected systems", "un-scrutinized software", systems using security by obscurity, etc.
- It employs Crypto against Crypto!
  Hiding Crypto in Crypto (as steganography hides text in text).

Kleptography — Introduction

The goal:

- To develop a robust back door within a cryptosystem satisfying:
  1. EXCLUSIVITY: Provides the desired secret information (e.g., private key of the unwary user) only to the attacker,
  2. INDISTINGUISHABILITY: Cannot be detected in black-box implementations (I/O access only as in tamper-resistant systems) except by the attacker; output (values, distribution, time) matches expectation;
  3. FORWARD SECRECY: If a reverse-engineer (i.e., not the attacker) breaches the black-box, then previously stolen information from this box or others cannot be recovered (secure against reverse-engineering).
- The successful reverse-engineer may learn that the attack is carried out (different algorithm is run), BUT will be unable to use the back door.

Kleptography — Attacking RSA

RSA Key Generation:

Let e be the public RSA exponent shared by all users (e.g., $e = 2^{16} + 1$).

1. Choose a large p randomly (e.g., p is 1024 bits long).
2. If p is composite or $\gcd(e, p - 1) \neq 1$ then goto step 1.
3. Choose a large number q randomly (e.g., q is 1024 bits long).
4. If q is composite or $\gcd(e, q - 1) \neq 1$ then goto step 3.
5. Solve for d given $d \cdot e \equiv 1 \mod \phi(n)$.
6. Output the public key $(n = pq, e)$ and private key d.

Kleptography — Attacking RSA

Kleptographic RSA Key Generation:

The attacker is using a 1024-bit RSA key with public key (Y, e).

1. Choose a large s randomly (e.g., s is 1024 bits long).
2. Compute $p = \mathrm{H}(s)$ where H is a cryptographic hash function.
3. If p is composite or $\gcd(e, p-1) \neq 1$ then goto step 1.
4. Choose a large 1024-bit string RND randomly.
5. Compute c to be an encryption of s under the attacker's RSA key, i.e., with public key (Y, e), $c = s^e \mod Y$.
6. Solve for (q, r) in $(c||\mathrm{RND}) = pq + r$ (integer division).
7. If q is composite or $\gcd(e, q-1) \neq 1$ then goto step 1.
8. Solve for d given $d \cdot e \equiv 1 \mod \phi(n)$.
9. Output the public key $(n = pq, e)$ and the private key d.

Kleptography — Attacking RSA

Observe:

c is the encryption of s with the attackers key.

$$(c||\mathrm{RND}) = pq + r \quad \Longrightarrow \quad (c||\mathrm{RND}) - r = pq = n$$

Note that r is about square root (i.e., about half the bit length) of n.
Thus the $(-r)$ operation will not ruin c by more than one bit (borrow bit).

The value c is not hidden much by the high order bits of n.

Selling argument

Could sell this as provably constructing n at random – using that an attacker could compute p if he could compute RSA decryption.
This would sell the RSA backdoor as a security feature!

Kleptography — Attacking RSA

Recovering the RSA Private Key:

- The attacker obtains the public key (n, e) of the user.
- Let u be the 1024 most significant bits of n.
- The attacker sets $c_1 = u$ and $c_2 = u + 1$.
  (c_2 accounts for a potential borrow bit having been taken from the computation $n = pq = (c||RND) - r$).
- The attacker decrypts c_1 and c_2 using his private key to get s_1 and s_2 respectively.
- Either $p_1 = \mathrm{H}(s_1)$ or $p_2 = \mathrm{H}(s_2)$ will divide n evenly.

Only the attacker can perform this operation since only the attacker knows the needed private decryption key.

Kleptography — Attacking RSA

Features of the attack:

- "Only" the key generation is tempered with.
- All messages observed outside of the black box are legitimate.
- There is no direct connection between the black box and the attacker.
- The security of the klepographic encryption is significantly lower than that of the user's key – Y has only half the length of n.
- The user's public key is used as an "subliminal" channel to exfiltrate the private key to the attacker.

Kleptography — Attacking RSA

Features of the attack:

- "Only" the key generation is tempered with.
- All messages observed outside of the black box are legitimate.
- There is no direct connection between the black box and the attacker.
- The security of the klepographic encryption is significantly lower than that of the user's key – Y has only half the length of n.
- The user's public key is used as an "subliminal" channel to exfiltrate the private key to the attacker.

What can the attacker do if there is no subliminal channel?
(e.g. Diffie-Hellman key exchange)

Kleptography — Attacking Diffie-Hellman

Diffie-Hellman key exchange:

1. Alice chooses a randomly.
2. Alice sends $A = g^a \mod p$ to Bob.
3. Bob chooses b randomly.
4. Bob sends $B = g^b \mod p$ to Alice.
5. Alice computes $k = B^a \mod p$.
6. Bob computes $k = A^b \mod p$.

Observe that $k = B^a = A^b \mod p$ since $g^{ba} = g^{ab} \mod p$.

Kleptography — Attacking Diffie-Hellman

Kleptographic Diffie-Hellman key exchange:

Klepto backdoor in Alice's device: attackers public key $y_m = g^{x_m}$.

- First exchange:
  - Alice's device picks a random a_1.
  - Alice's device computes $A_1 = g^{a_1}$ and Alice sends A_1 to Bob.
  - Alice's device stores a_1 in non-volatile memory.

Kleptography — Attacking Diffie-Hellman

Kleptographic Diffie-Hellman key exchange:

Klepto backdoor in Alice's device: attackers public key $y_m = g^{x_m}$.

- First exchange:
  - Alice's device picks a random a_1.
  - Alice's device computes $A_1 = g^{a_1}$ and Alice sends A_1 to Bob.
  - Alice's device stores a_1 in non-volatile memory.
- Second exchange:
  - Alice's device computes $a_2 = \mathrm{H}(y_m^{a_1} \mod p)$.
  - Alice's device computes $A_2 = g^{a_2}$ and Alice sends A_2 to Bob.
  - Alice's device stores a_2 in non-volatile memory.

Kleptography — Attacking Diffie-Hellman

Kleptographic Diffie-Hellman key exchange:

Klepto backdoor in Alice's device: attackers public key $y_m = g^{x_m}$.

- First exchange:
  - Alice's device picks a random a_1.
  - Alice's device computes $A_1 = g^{a_1}$ and Alice sends A_1 to Bob.
  - Alice's device stores a_1 in non-volatile memory.
- Second exchange:
  - Alice's device computes $a_2 = \mathrm{H}(y_m^{a_1} \mod p)$.
  - Alice's device computes $A_2 = g^{a_2}$ and Alice sends A_2 to Bob.
  - Alice's device stores a_2 in non-volatile memory.
- Recovering the Diffie-Hellman key:
  - The attacker obtains A_1 at first passive eavesdropping.
  - The attacker computes $a_2 = \mathrm{H}(A_1^{x_m} \mod p)$.

$$A_1^{x_m} \equiv g^{a_1 x_m} \equiv g^{x_m a_1} \equiv y_m^{a_1} \mod p$$

The attack can be chained for more key exchanges,
the attacker misses only the first encrypted communication.

Kleptography — Attacking Diffie-Hellman

Definition of a SETUP attack:

A Secretly Embedded Trapdoor with Universal Protection (SETUP) attack is an algorithmic modification C' of a cryptosystem C with the following properties:

1. Halting Correctness: C and C' are efficient algorithms.
2. Output Indistinguishability: The outputs of C and C' are computationally indistinguishable to all efficient algorithms except for the attacker.
3. Confidentiality of C: The outputs of C do not compromise the security of the cryptosystem that C implements.
4. Confidentiality of C': The outputs of C' only compromise the security of the cryptosystem that C' implements with respect to the attacker.
5. Ability to compromise C': With overwhelming probability the attacker can break/decrypt/cryptanalyze at least one private output of C' given a sufficient number of public outputs of C'.

Fancier Attack on RSA

Let the RSA keys have $2m$ bits, e.g., 3072 for 128-bit security. Attacker's key is $P_E = aP$ on a secure elliptic curve.

- Combine ECC with interesting RSA facts.
- Coppersmith's attack (and extensions): can factor efficiently if top $m/2$ bits of p are known (faster with more).
- Can build RSA modulus for any given p and top $\sim m$ bits of n.

Fancier Attack on RSA

Let the RSA keys have $2m$ bits, e.g., 3072 for 128-bit security. Attacker's key is $P_E = aP$ on a secure elliptic curve.

- Combine ECC with interesting RSA facts.
- Coppersmith's attack (and extensions): can factor efficiently if top $m/2$ bits of p are known (faster with more).
- Can build RSA modulus for any given p and top $\sim m$ bits of n.
- The device picks random b, computes bP and $k = \mathrm{hash}(bP_E)$.
- Pick random prime p. Compute authenticated encryption c of top half of p under key k. This has length $m/2$ plus the authenticator, e.g., 768+128=896 bits.
- Put the top part of n equal to an encoding of bP (256 bits, make it look uniform) followed by c, compute q and $n = pq$. Lots of space left!

Fancier Attack on RSA

Let the RSA keys have $2m$ bits, e.g., 3072 for 128-bit security. Attacker's key is $P_E = aP$ on a secure elliptic curve.

- Combine ECC with interesting RSA facts.
- Coppersmith's attack (and extensions): can factor efficiently if top $m/2$ bits of p are known (faster with more).
- Can build RSA modulus for any given p and top $\sim m$ bits of n.
- The device picks random b, computes bP and $k = \mathrm{hash}(bP_E)$.
- Pick random prime p. Compute authenticated encryption c of top half of p under key k. This has length $m/2$ plus the authenticator, e.g., 768+128=896 bits.
- Put the top part of n equal to an encoding of bP (256 bits, make it look uniform) followed by c, compute q and $n = pq$. Lots of space left!
- Attacker decodes and trial decrypts. If authenticator works it's a klepto device. Then use known bits of p to factor n.

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:

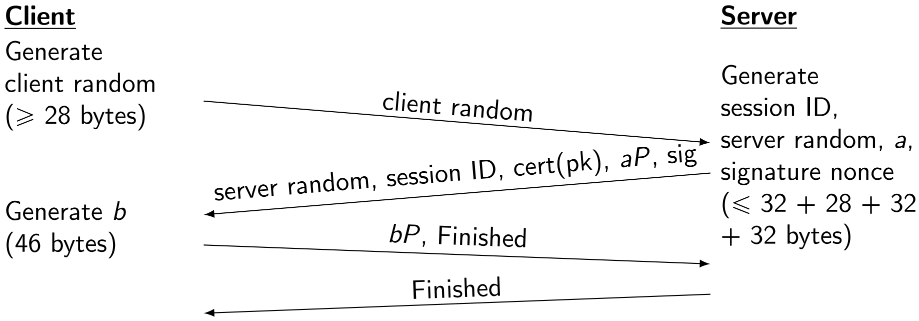
[..] we know that these encryption algorithms we are using today work typically it is the random number generators that are attacked as opposed to the encryption algorithms themselves.

SSL/TLS/HTTPS – internet security protocols

Use of randomness in internet protocols.

SSL/TLS/HTTPS – internet security protocols

Use of randomness in internet protocols.


Client
Generate client random ($\geqslant$ 28 bytes)
Generate b (46 bytes)
Server
Generate session ID, server random, a, signature nonce ($\leqslant$ 32 + 28 + 32 + 32 bytes)
client random
server random, session ID, cert(pk), aP, sig
bP, Finished
Finished

MS = PRF($x(abP)$, "master secret", client random || server random)

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_0:

$$\begin{tikzcd}
s_0 \arrow[r, "f(s_0)"] \arrow[d, "g(s_0)"] & s_1 \arrow[r, "f(s_1)"] \arrow[d, "g(s_1)"] & s_2 \arrow[r, "f(s_2)"] \arrow[d, "g(s_2)"] & s_3 \arrow[r, "f(s_3)"] \arrow[d, "g(s_3)"] & \cdots \\
r_0 & r_1 & r_2 & r_3 &
\end{tikzcd}$$

XXX's mission: Predict the "random" output bits.
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_0:

$$\begin{tikzcd}
s_0 \arrow[r, "f(s_0)"] \arrow[d, "g(s_0)"] & s_1 \arrow[r, "f(s_1)"] \arrow[d, "g(s_1)"] & s_2 \arrow[r, "f(s_2)"] \arrow[d, "g(s_2)"] & s_3 \arrow[r, "f(s_3)"] \arrow[d, "g(s_3)"] & \cdots \\
r_0 & r_1 \arrow[ur] & r_2 & r_3 &
\end{tikzcd}$$

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

Create protocols that directly output r_n for some reason.

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_0:

$$\begin{tikzcd}
s_0 \arrow[r, "f(s_0)"] \arrow[d, "g(s_0)"] & s_1 \arrow[r, "f(s_1)"] \arrow[d, "g(s_1)"] & s_2 \arrow[r, "f(s_2)"] \arrow[d, "g(s_2)"] & s_3 \arrow[r, "f(s_3)"] \arrow[d, "g(s_3)"] & \cdots \\
r_0 & r_1 \arrow[ur] & r_2 & r_3 &
\end{tikzcd}$$

XXX's mission: Predict the "random" output bits.
Create protocols that directly output r_n for some reason.
Design f, g with back door from r_n to s_{n+1}: i.e., get $f(s)$ from $g(s)$.
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_0:

$$\begin{tikzcd}
s_0 \arrow[r, "f(s_0)"] \arrow[d, "g(s_0)"] & s_1 \arrow[r, "f(s_1)"] \arrow[d, "g(s_1)"] & s_2 \arrow[r, "f(s_2)"] \arrow[d, "g(s_2)"] & s_3 \arrow[r, "f(s_3)"] \arrow[d, "g(s_3)"] & \cdots \\
r_0 & r_1 \arrow[ru] & r_2 & r_3 &
\end{tikzcd}$$

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

Create protocols that directly output r_n for some reason.
Design f, g with back door from r_n to s_{n+1}: i.e., get $f(s)$ from $g(s)$.
Standardize this design of f, g.
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 $P = dQ$ then it's easy: $sP = sdQ$.

Let's choose random Q, random d, define $P = dQ$.
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 $P = dQ$ then it's easy: $sP = sdQ$.

Let's choose random Q, random d, define $P = dQ$.
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: $y^2 = x^3 - 3x +$ 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 $P = dQ$ then it's easy: $sP = sdQ$.

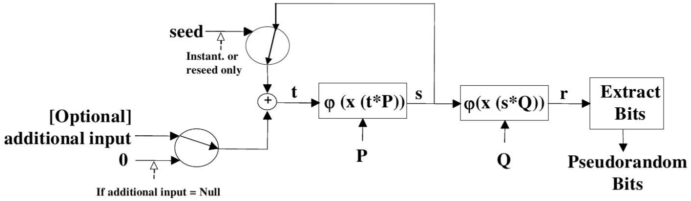
Let's choose random Q, random d, define $P = dQ$.
Standardize this P; Q; $f(s) = sP$; $g(s) = sQ$.

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

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

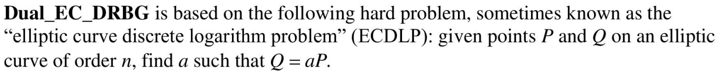
Dual EC RNG: history part I

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



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

Claim:

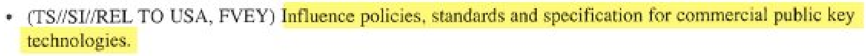

Dual_EC_DRBG is based on the following hard problem, sometimes known as the "elliptic curve discrete logarithm problem" (ECDLP): given points P and Q on an elliptic curve of order n, find a such that $Q = aP$.

Common public history part II

Various public warning signals:

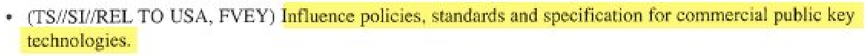
- Gjøsteen (March 2006): output sequence is biased.
- Brown (March 2006): security "proof"
  "This proof makes essential use of Q being random." If d with $dQ = P$ is known then $dR_i = S_{i+1}$, concludes that there might be distinguisher.
- Sidorenko & Schoenmakers (May 2006): output sequence is even more biased. Answer: Too late to change, already implemented.
- Included in standards ISO 18031 (2005), NIST SP 800-90 (2006), ANSI X9.82 (2007).
- 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

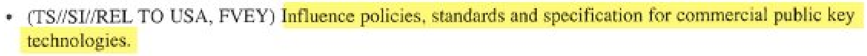


• (TS//SI//REL TO USA, FVEY) Influence policies, standards and specification for commercial public key technologies.

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



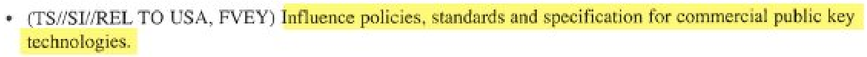
- (TS//SI//REL TO USA, FVEY) Influence policies, standards and specification for commercial public key technologies.

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.

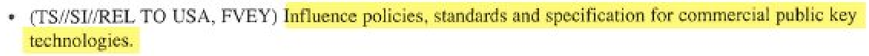
NYT:

> *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!?!

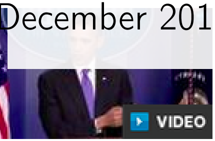
NIST's DRBG Validation List: more than 70 validations of Dual EC DRBG;
RSA's BSAFE has Dual EC DRBG enabled as default,.

NIST re-opens discussions on SP800.90; recommends against using Dual EC.
RSA suggests changing default in BSAFE.

21 April 2014 NIST removes Dual EC from the standard.

ELATED VIDEO

December 2013

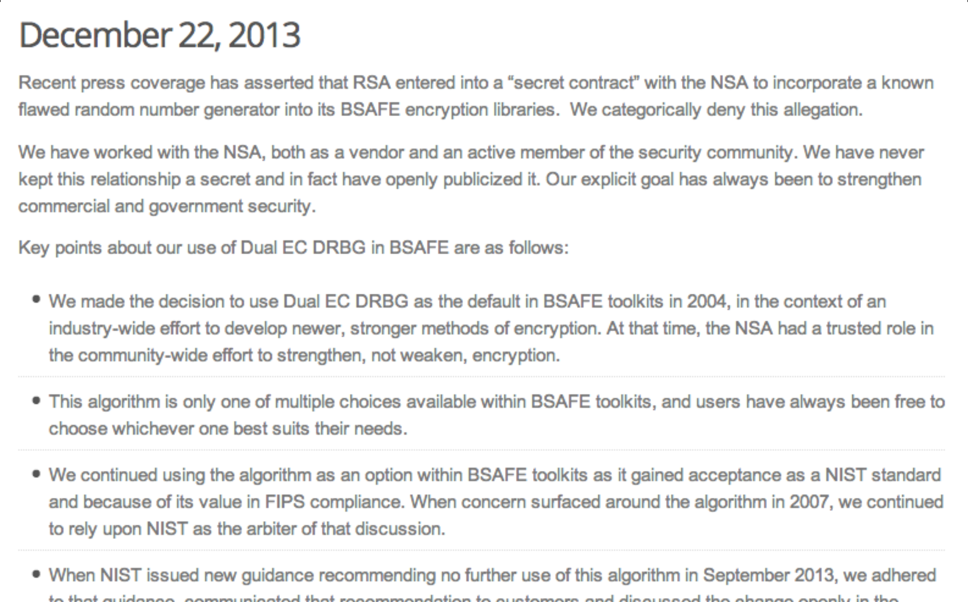


bama on surveillance: here may be another way skinning the cat"

(Reuters) - As a key part of a campaign to embed encryption software that it could crack into widely used computer products, the U.S. National Security Agency arranged a secret $10 million contract with RSA, one of the most influential firm in the computer security industry, Reuters has learned.

Documents leaked by former NSA contractor Edward Snowde show that the NSA created and promulgated a flawed formula for generating random numbers to create a "back door" in ncryption products, the New York Times reported in September. Reuters later reported hat RSA became the most important distributor of that formula by rolling it into a oftware tool called Bsafe that is used to enhance security in personal computers and nany other products.

Indisclosed until now was that RSA received $10 million in a deal that set the NSA ormula as the preferred, or default, method for number generation in the BSafe oftware, according to two sources familiar with the contract. Although that sum might eem paltry, it represented more than a third of the revenue that the relevant division at SA had taken in during the entire previous year, securities filings show.



December 22, 2013

Recent press coverage has asserted that RSA entered into a “secret contract” with the NSA to incorporate a known flawed random number generator into its BSAFE encryption libraries. We categorically deny this allegation.

We have worked with the NSA, both as a vendor and an active member of the security community. We have never kept this relationship a secret and in fact have openly publicized it. Our explicit goal has always been to strengthen commercial and government security.

Key points about our use of Dual EC DRBG in BSAFE are as follows:

- We made the decision to use Dual EC DRBG as the default in BSAFE toolkits in 2004, in the context of an industry-wide effort to develop newer, stronger methods of encryption. At that time, the NSA had a trusted role in the community-wide effort to strengthen, not weaken, encryption.
- This algorithm is only one of multiple choices available within BSAFE toolkits, and users have always been free to choose whichever one best suits their needs.
- We continued using the algorithm as an option within BSAFE toolkits as it gained acceptance as a NIST standard and because of its value in FIPS compliance. When concern surfaced around the algorithm in 2007, we continued to rely upon NIST as the arbiter of that discussion.
- When NIST issued new guidance recommending no further use of this algorithm in September 2013, we adhered to that guidance, communicated that recommendation to customers and discussed the change openly in the

Common TLS implementations:

- RSA's BSAFE
  - RSA BSAFE Share for Java (BSAFE Java)
  - RSA BSAFE Share for C and C++ (BSAFE C)
- Microsoft's SChannel
- OpenSSL

All of these offer Dual EC.

Common TLS implementations:

- RSA's BSAFE
  - RSA BSAFE Share for Java (BSAFE Java)
  - RSA BSAFE Share for C and C++ (BSAFE C)
- Microsoft's SChannel
- OpenSSL

Remember: NSA paid RSA Security $10 million to use Dual EC as the default RNG!

All of these offer Dual EC.

Dual EC

Parameters

Here: elliptic curve over finite filed with NIST prime P-256.
(NIST SP800-90A also defines curves for P-384 and P-521.)

The elliptic curve is defined over $\mathbf{F}_p$ with $p = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$.
The curve is given in short Weierstrass form

$$E : y^2 = x^3 - 3x + b, \text{ where}$$

$b =$ `0x5ac635d8aa3a93e7b3ebbd55769886bc651d06b0cc53b0f63bce3c3e27d2604b`.

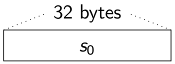
Dual EC defines two points, a base point P and a second point Q:

$P_x =$ `0x6b17d1f2e12c4247f8bce6e563a440f277037d812deb33a0f4a13945d898c296`,
$P_y =$ `0x4fe342e2fe1a7f9b8ee7eb4a7c0f9e162bce33576b315ececbb6406837bf51f5`;

$Q_x =$ `0xc97445f45cdef9f0d3e05e1e585fc297235b82b5be8ff3efca67c59852018192`,
$Q_y =$ `0xb28ef557ba31dfcbdd21ac46e2a91e3c304f44cb87058ada2cb815151e610046`.

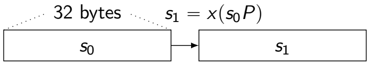
Dual EC — Basic Procedure

Points Q and P on an elliptic curve.



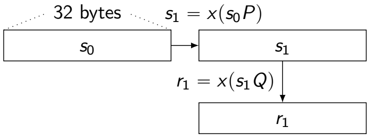
Dual EC — Basic Procedure

Points Q and P on an elliptic curve.



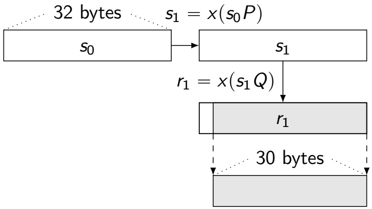
Dual EC — Basic Procedure

Points Q and P on an elliptic curve.



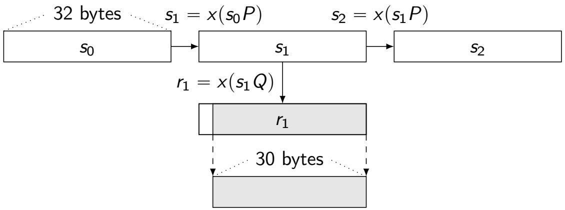
Dual EC — Basic Procedure

Points Q and P on an elliptic curve.



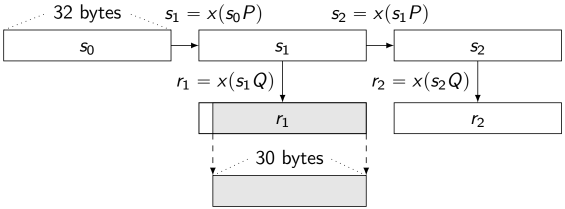
Dual EC — Basic Procedure

Points Q and P on an elliptic curve.



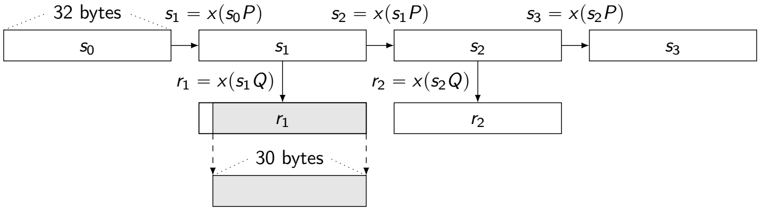
Dual EC — Basic Procedure

Points Q and P on an elliptic curve.



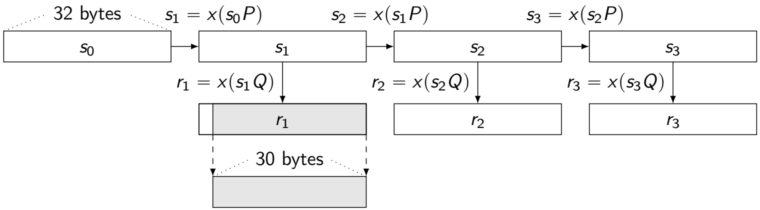
Dual EC — Basic Procedure

Points Q and P on an elliptic curve.



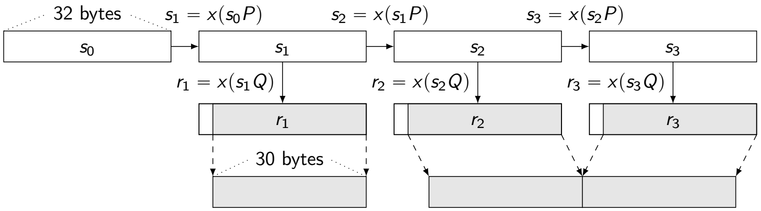
Dual EC — Basic Procedure

Points Q and P on an elliptic curve.



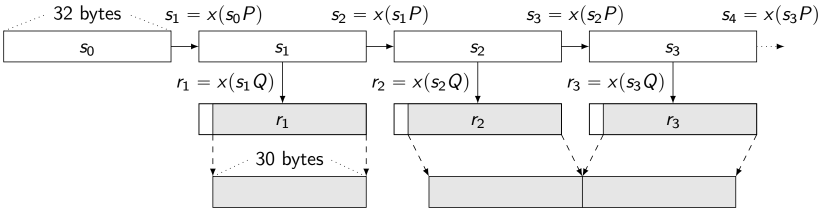
Dual EC — Basic Procedure

Points Q and P on an elliptic curve.



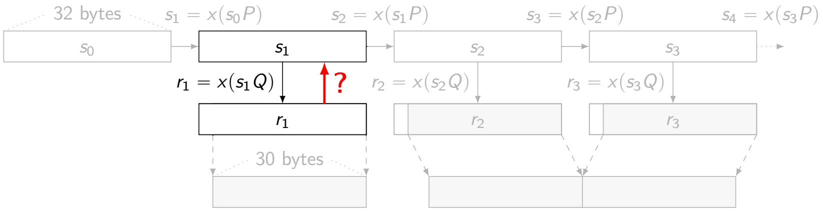
Dual EC — Basic Procedure

Points Q and P on an elliptic curve.



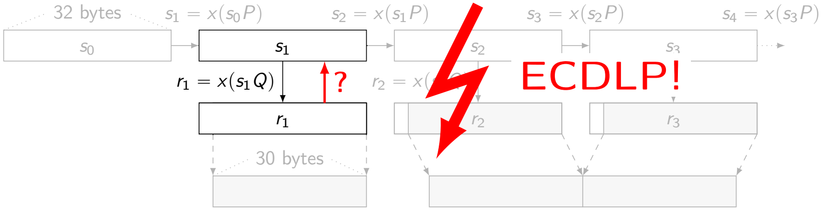
Dual EC — Basic Procedure

Points Q and P on an elliptic curve.



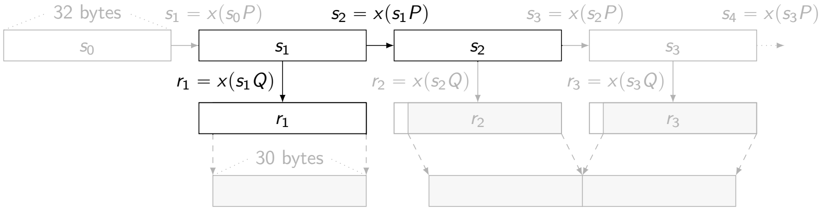
Dual EC — Basic Procedure

Points Q and P on an elliptic curve.



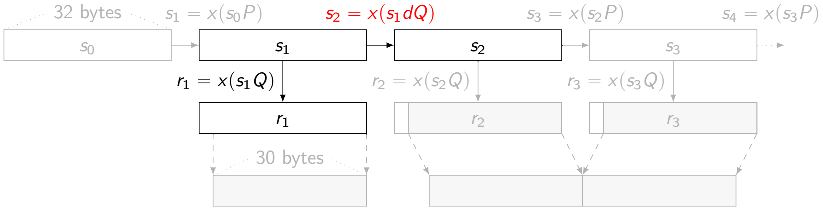
Shumow and Ferguson – the Basic Attack

Points Q and $P = dQ$ on an elliptic curve.



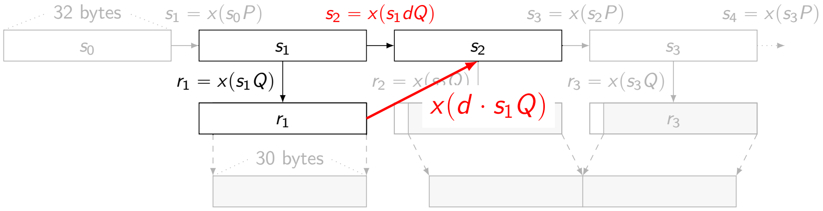
Shumow and Ferguson – the Basic Attack

Points Q and $P = dQ$ on an elliptic curve.



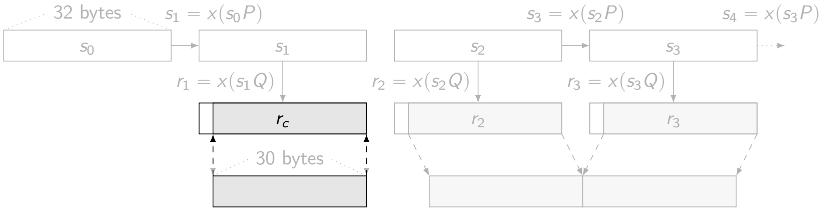
Shumow and Ferguson – the Basic Attack

Points Q and $P = dQ$ on an elliptic curve.



Shumow and Ferguson – the Basic Attack

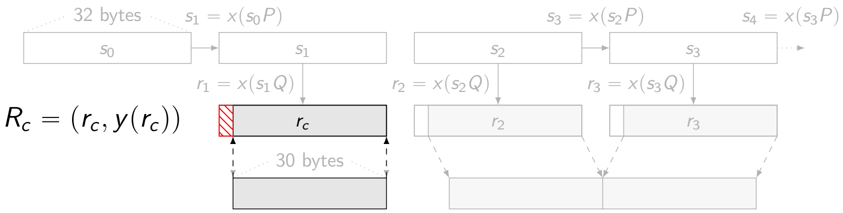
Points Q and $P = dQ$ on an elliptic curve.



Shumow and Ferguson – the Basic Attack

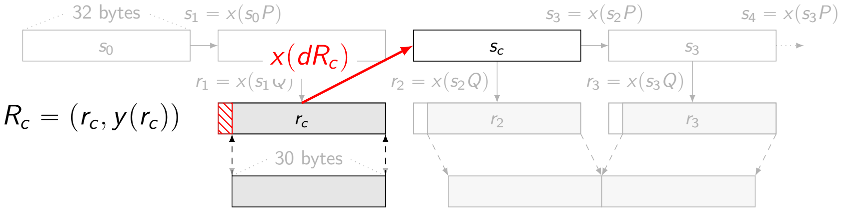
Points Q and $P = dQ$ on an elliptic curve.

$R_c = (r_c, y(r_c))$



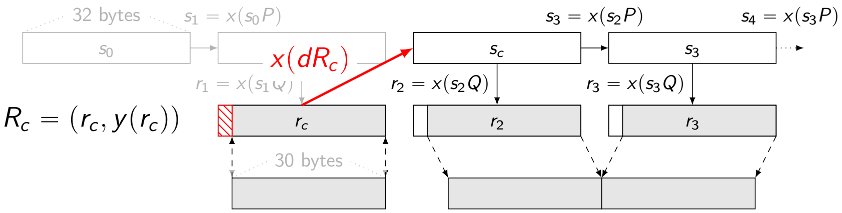
Shumow and Ferguson – the Basic Attack

Points Q and $P = dQ$ on an elliptic curve.

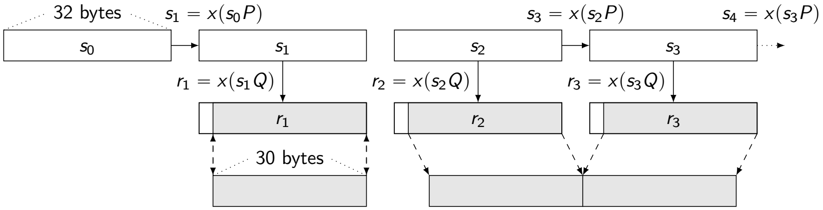


Shumow and Ferguson – the Basic Attack

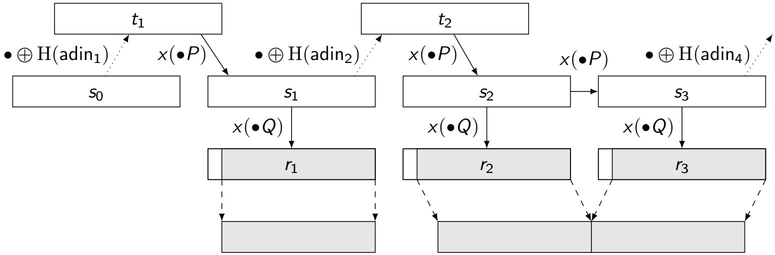
Points Q and $P = dQ$ on an elliptic curve.



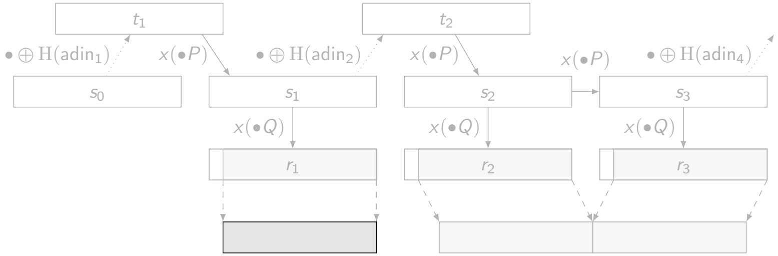
Shumow and Ferguson – the Basic Attack



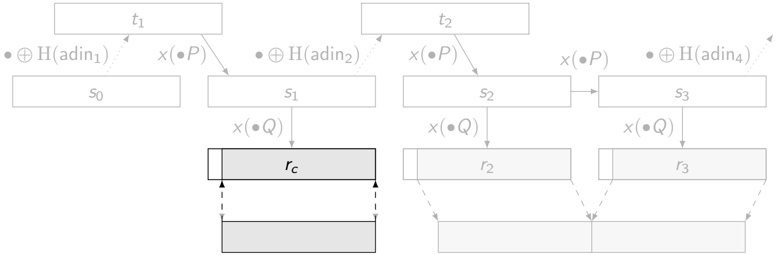
Dual EC — NIST SP800-90 in June 2006


t_1
t_2
$\bullet \oplus \mathrm{H}(\mathrm{adin}_1)$
$x(\bullet P)$
$\bullet \oplus \mathrm{H}(\mathrm{adin}_2)$
$x(\bullet P)$
$x(\bullet P)$
$\bullet \oplus \mathrm{H}(\mathrm{adin}_4)$
s_0
s_1
s_2
s_3
$x(\bullet Q)$
$x(\bullet Q)$
$x(\bullet Q)$
r_1
r_2
r_3

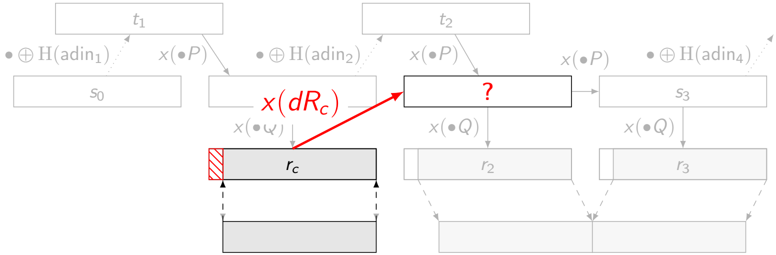
Dual EC — NIST SP800-90 in June 2006


t_1
t_2
$\bullet \oplus \mathrm{H}(\mathrm{adin}_1)$
$x(\bullet P)$
$\bullet \oplus \mathrm{H}(\mathrm{adin}_2)$
$x(\bullet P)$
$x(\bullet P)$
$\bullet \oplus \mathrm{H}(\mathrm{adin}_4)$
s_0
s_1
s_2
s_3
$x(\bullet Q)$
$x(\bullet Q)$
$x(\bullet Q)$
r_1
r_2
r_3

Dual EC — NIST SP800-90 in June 2006

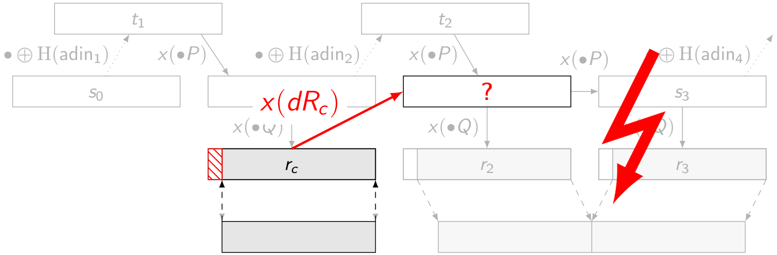

t_1
t_2
$\bullet \oplus \mathrm{H}(\mathrm{adin}_1)$
$x(\bullet P)$
$\bullet \oplus \mathrm{H}(\mathrm{adin}_2)$
$x(\bullet P)$
$x(\bullet P)$
$\bullet \oplus \mathrm{H}(\mathrm{adin}_4)$
s_0
s_1
s_2
s_3
$x(\bullet Q)$
$x(\bullet Q)$
$x(\bullet Q)$
r_c
r_2
r_3

Dual EC — NIST SP800-90 in June 2006

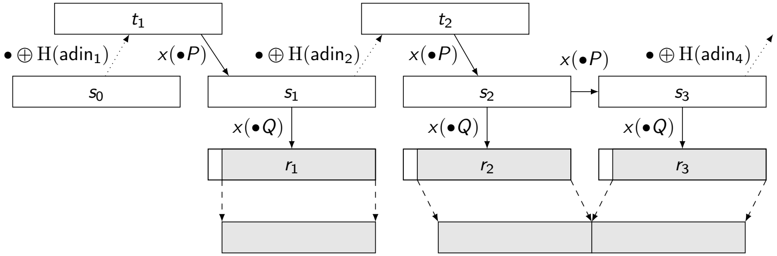


t_1
t_2
$\bullet \oplus \mathrm{H}(\mathrm{adin}_1)$
$x(\bullet P)$
$\bullet \oplus \mathrm{H}(\mathrm{adin}_2)$
$x(\bullet P)$
$x(\bullet P)$
$\bullet \oplus \mathrm{H}(\mathrm{adin}_4)$
s_0
$x(dR_c)$
?
s_3
$x(\bullet Q)$
$x(\bullet Q)$
$x(\bullet Q)$
r_c
r_2
r_3

Dual EC — NIST SP800-90 in June 2006

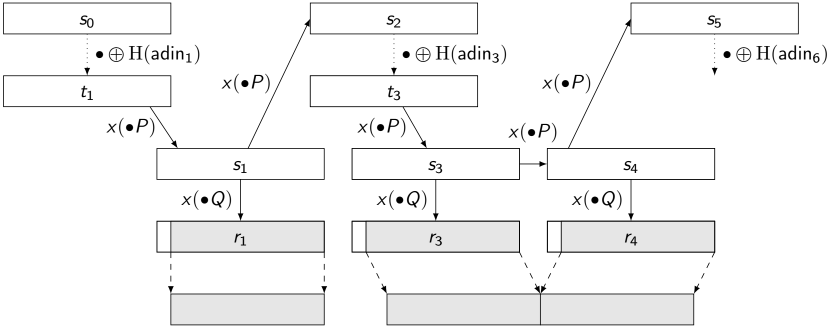

t_1
t_2
$\bullet \oplus \mathrm{H}(\mathrm{adin}_1)$
$x(\bullet P)$
$\bullet \oplus \mathrm{H}(\mathrm{adin}_2)$
$x(\bullet P)$
$x(\bullet P)$
$\oplus \mathrm{H}(\mathrm{adin}_4)$
s_0
$x(dR_c)$
?
s_3
$x(\bullet Q)$
$x(\bullet Q)$
$Q)$
r_c
r_2
r_3

Dual EC — NIST SP800-90 in June 2006

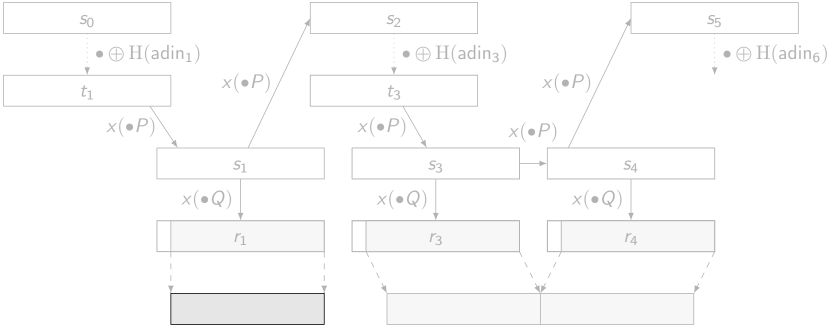


t_1
t_2
$\bullet \oplus \mathrm{H}(\mathrm{adin}_1)$
$x(\bullet P)$
$\bullet \oplus \mathrm{H}(\mathrm{adin}_2)$
$x(\bullet P)$
$x(\bullet P)$
$\bullet \oplus \mathrm{H}(\mathrm{adin}_4)$
s_0
s_1
s_2
s_3
$x(\bullet Q)$
$x(\bullet Q)$
$x(\bullet Q)$
r_1
r_2
r_3

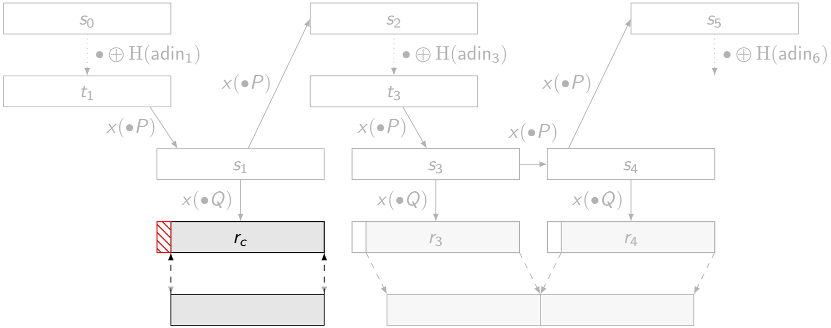
Dual EC — NIST SP800-90 in March 2007



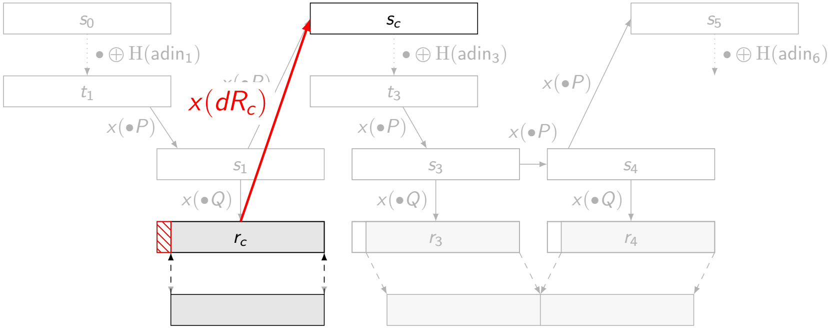
Dual EC — NIST SP800-90 in March 2007



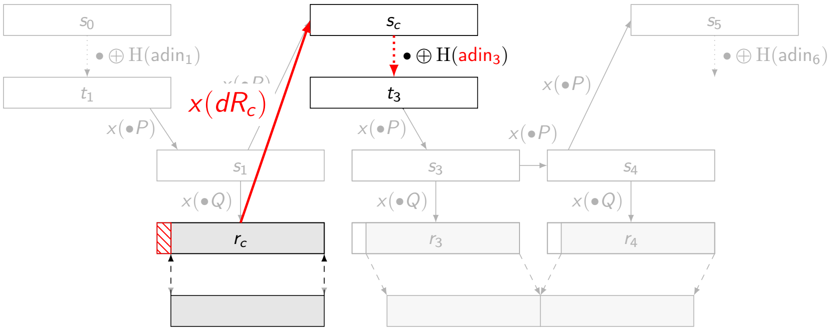
Dual EC — NIST SP800-90 in March 2007



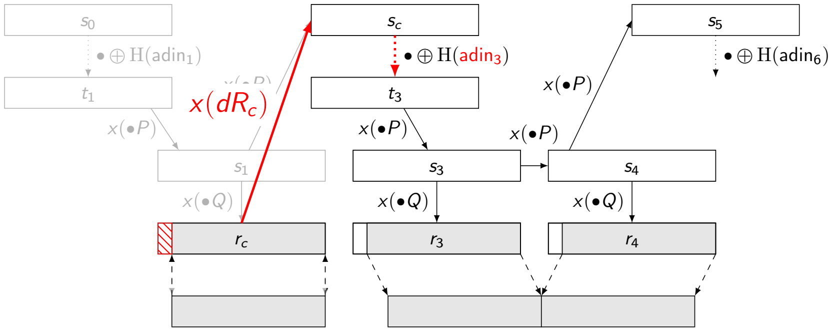
Dual EC — NIST SP800-90 in March 2007



Dual EC — NIST SP800-90 in March 2007



Dual EC — NIST SP800-90 in March 2007



Attack

Attack targets:

In the real world, the attack is more complicated. Targets of the attack:

- RSA's BSAFE
  - RSA BSAFE Share for Java (BSAFE Java)
  - RSA BSAFE Share for C and C++ (BSAFE C)
- Microsoft's SChannel
- OpenSSL

The points P and Q have been replaced with known $P = dQ$; this required some reverse engineering of BSAFE and SChannel.

Attack

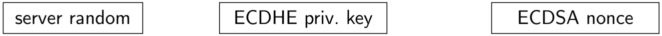
Attack targets:

In the real world, the attack is more complicated. Targets of the attack:

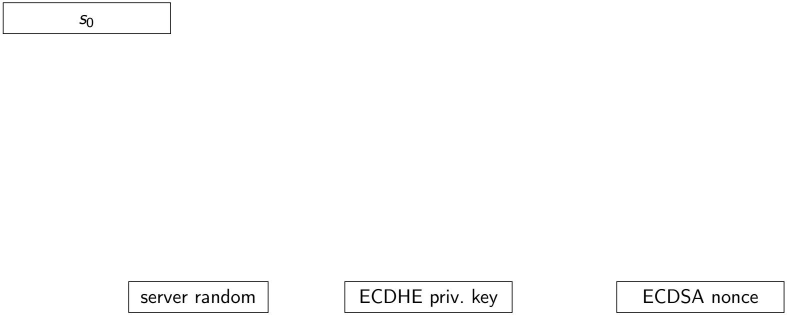
- RSA's BSAFE
  - RSA BSAFE Share for Java (BSAFE Java)
  - RSA BSAFE Share for C and C++ (BSAFE C)
- Microsoft's SChannel
- OpenSSL-fixed

The points P and Q have been replaced with known $P = dQ$; this required some reverse engineering of BSAFE and SChannel.

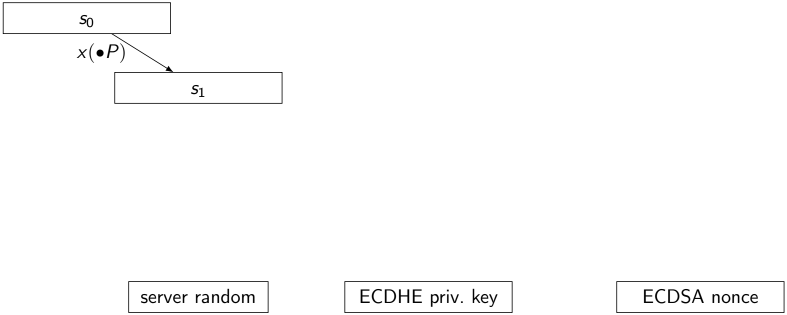
Attack — Example: BSAFE-Java



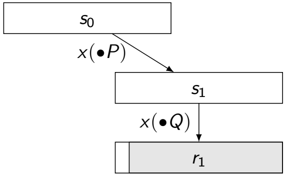
Attack — Example: BSAFE-Java

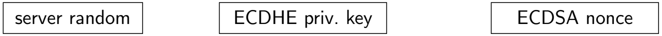


Attack — Example: BSAFE-Java

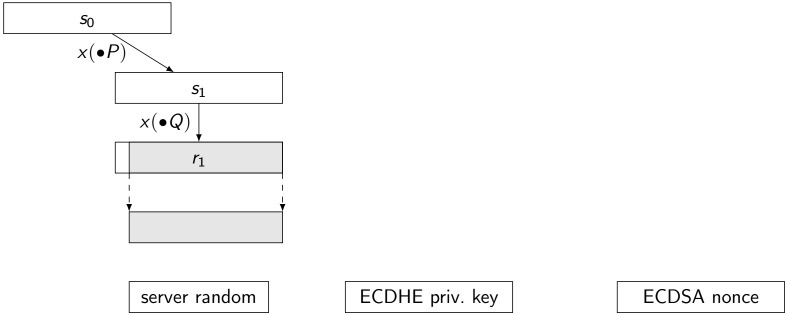


Attack — Example: BSAFE-Java

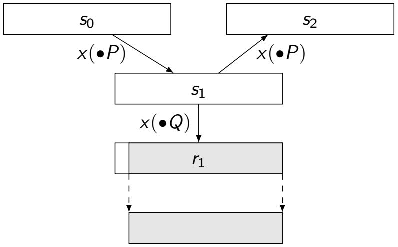




Attack — Example: BSAFE-Java



Attack — Example: BSAFE-Java

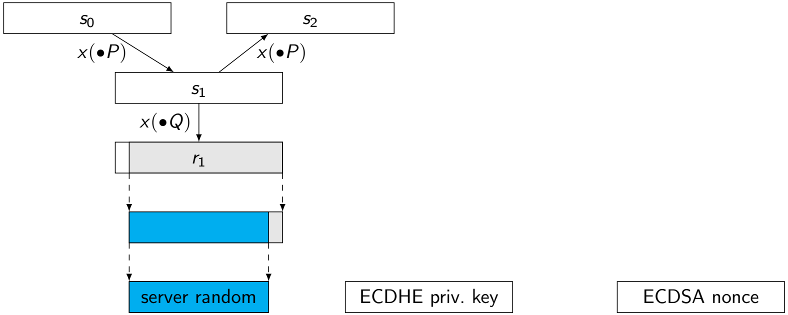


server random

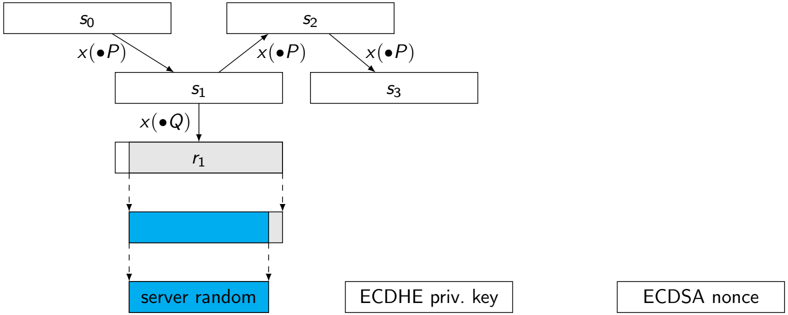
ECDHE priv. key

ECDSA nonce

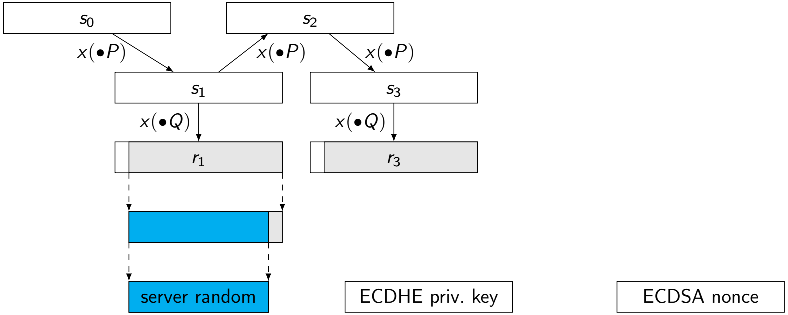
Attack — Example: BSAFE-Java



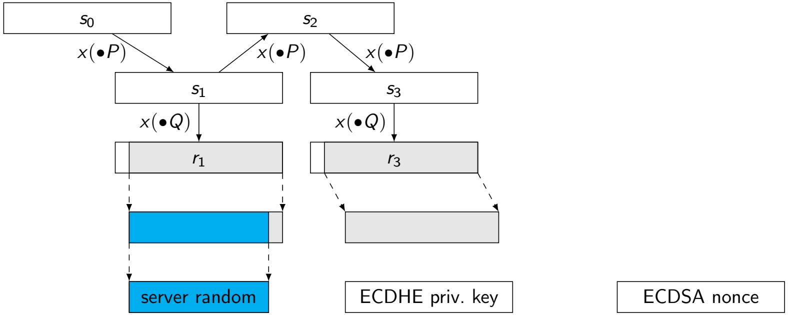
Attack — Example: BSAFE-Java



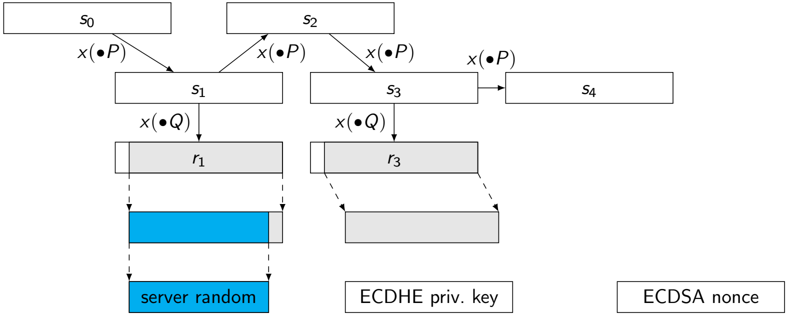
Attack — Example: BSAFE-Java



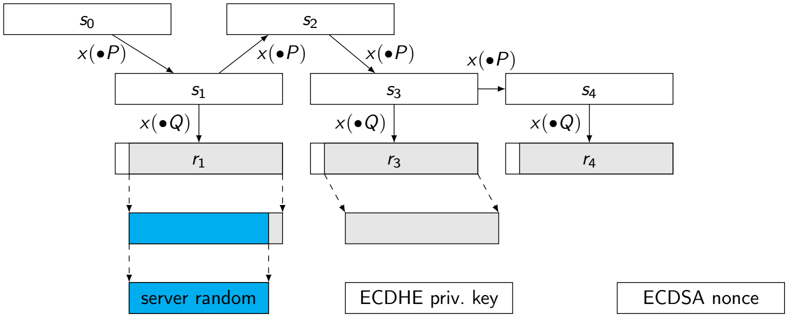
Attack — Example: BSAFE-Java



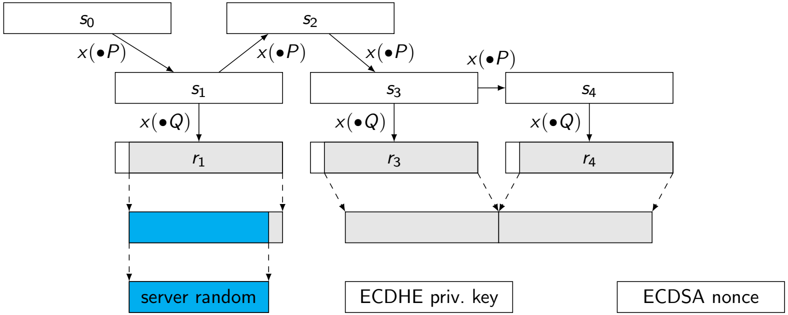
Attack — Example: BSAFE-Java



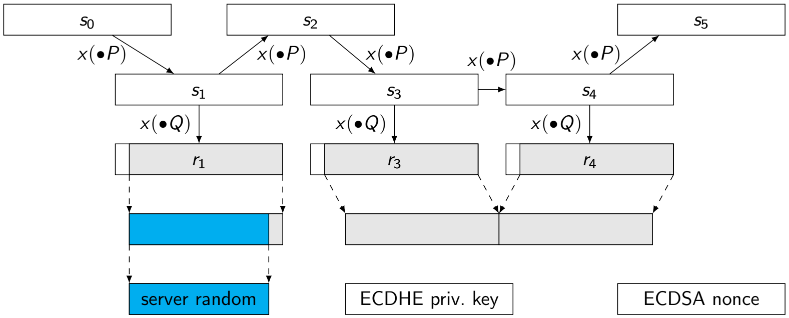
Attack — Example: BSAFE-Java



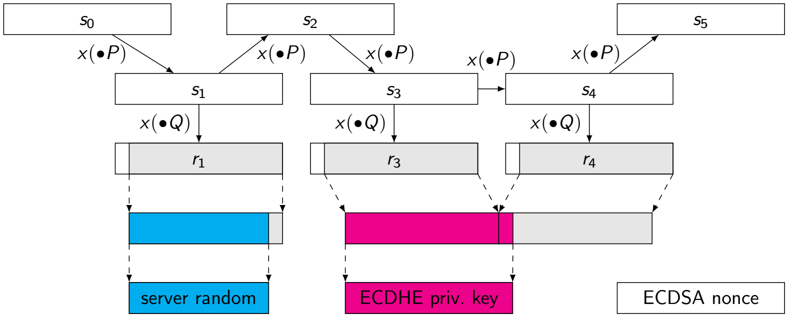
Attack — Example: BSAFE-Java



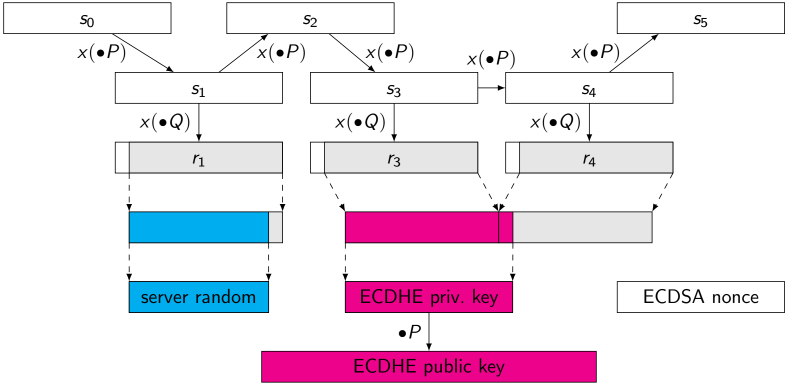
Attack — Example: BSAFE-Java



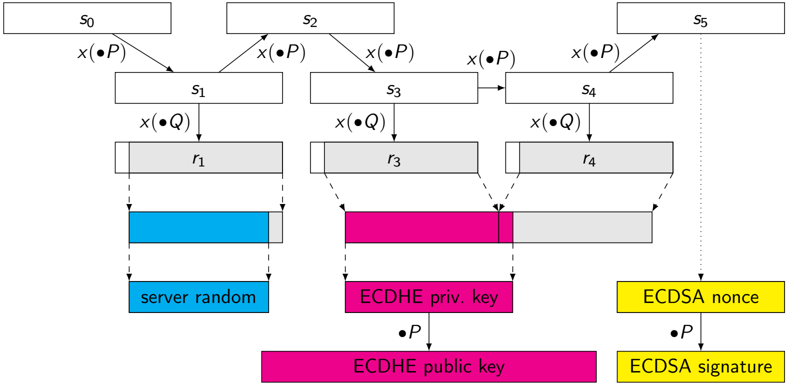
Attack — Example: BSAFE-Java



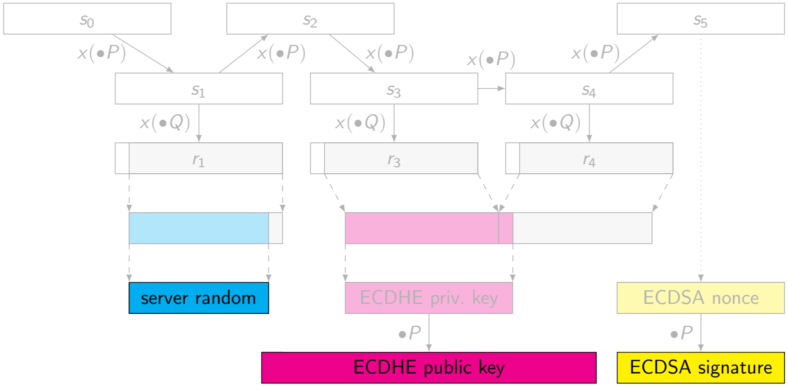
Attack — Example: BSAFE-Java



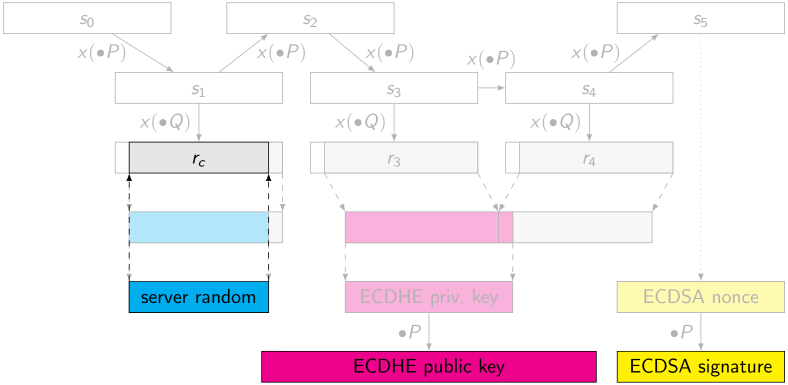
Attack — Example: BSAFE-Java



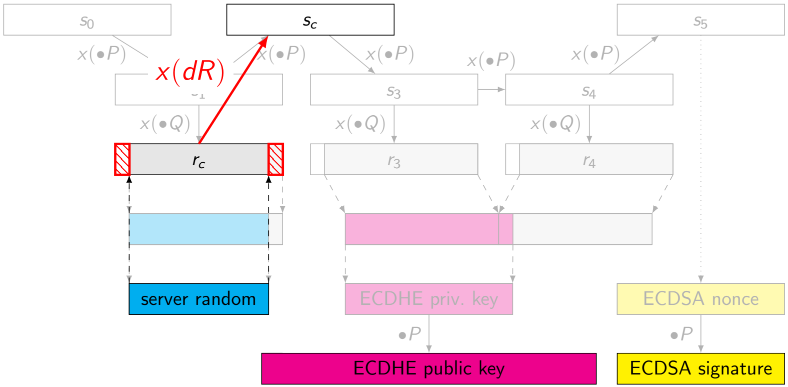
Attack — Example: BSAFE-Java



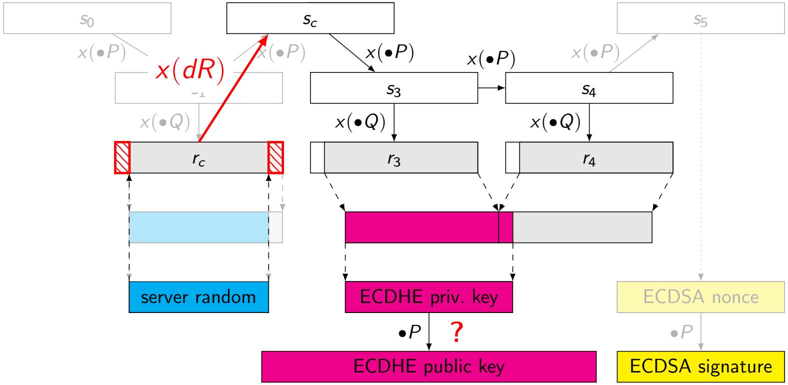
Attack — Example: BSAFE-Java



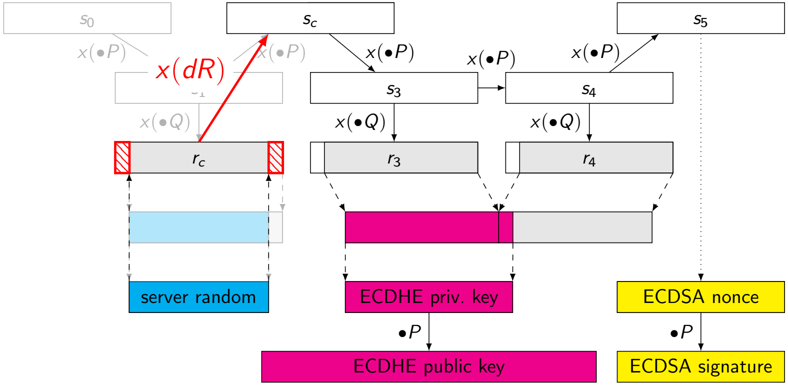
Attack — Example: BSAFE-Java



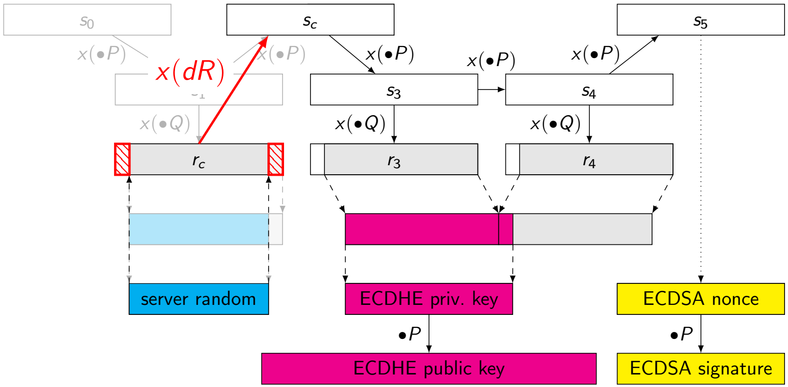
Attack — Example: BSAFE-Java



Attack — Example: BSAFE-Java

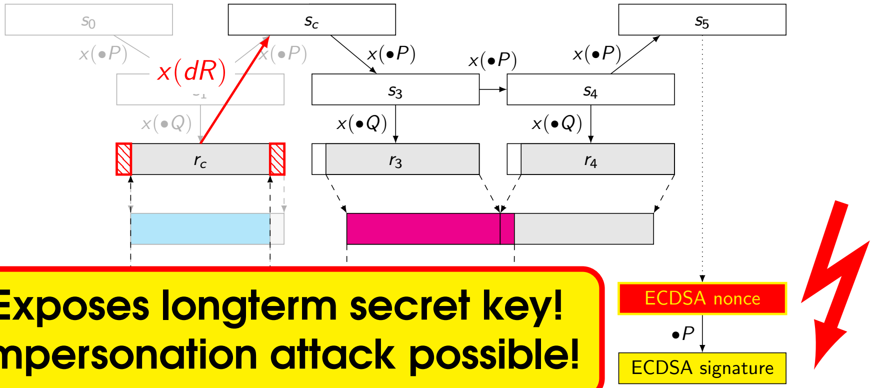


Attack — Example: BSAFE-Java



average cost: $2^{31}(C_v + 5C_f)$

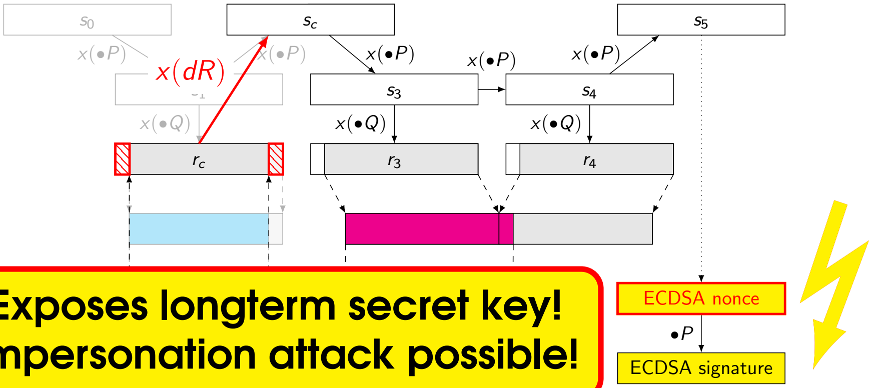
Attack — Example: BSAFE-Java



Exposes longterm secret key!
Impersonation attack possible!

average cost: $2^{31}(C_v + 5C_f)$

Attack — Example: BSAFE-Java

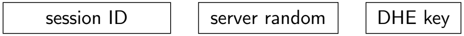


Exposes longterm secret key!
Impersonation attack possible!

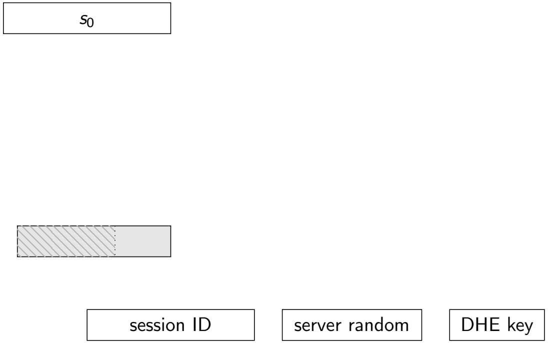
average cost: $2^{31}(C_v + 5C_f)$

Also BSAFE-Java easily identifiable by watermark.

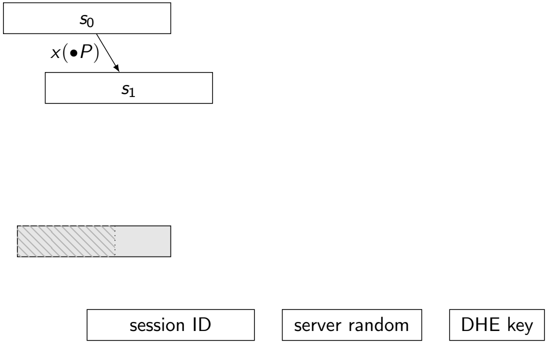
Attack — BSAFE-C



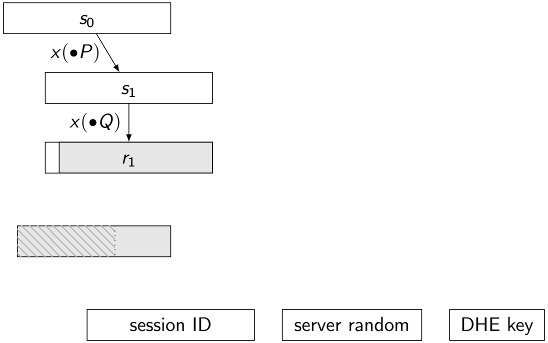
Attack — BSAFE-C



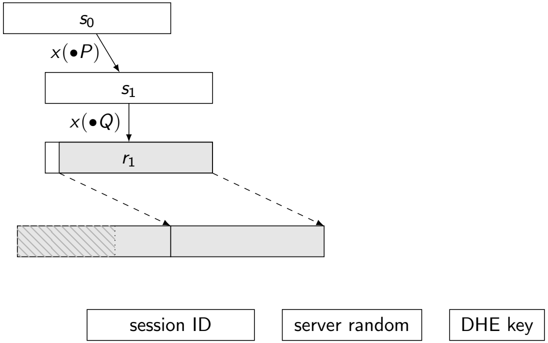
Attack — BSAFE-C



Attack — BSAFE-C

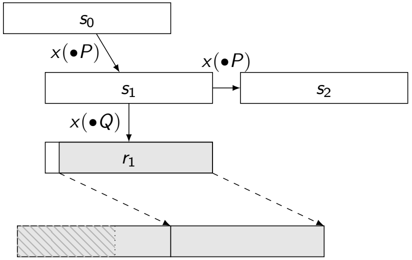


Attack — BSAFE-C



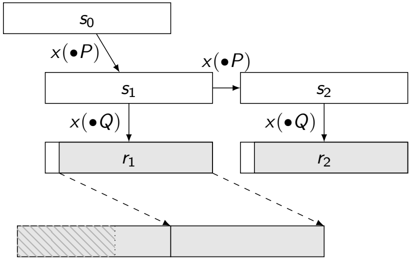
Technische Universiteit **Eindhoven** University of Technology

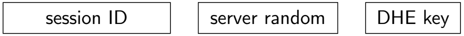
Attack — BSAFE-C



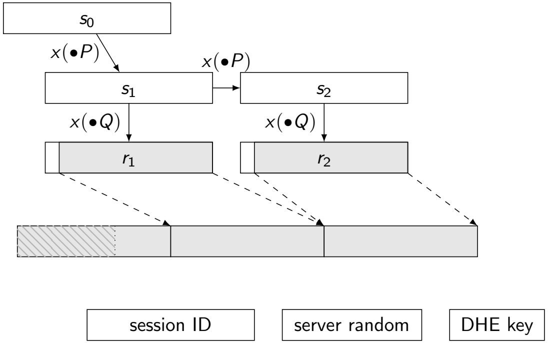
session ID | server random | DHE key

Attack — BSAFE-C

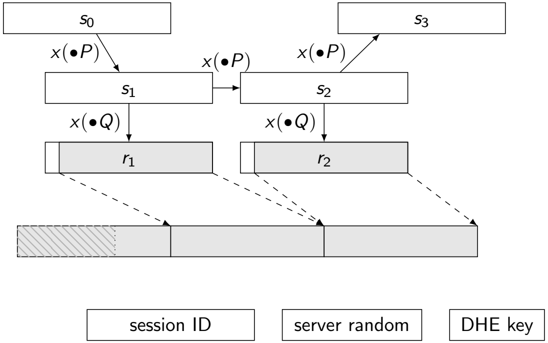




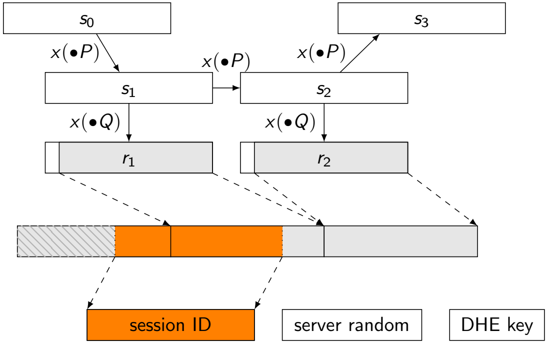
Attack — BSAFE-C



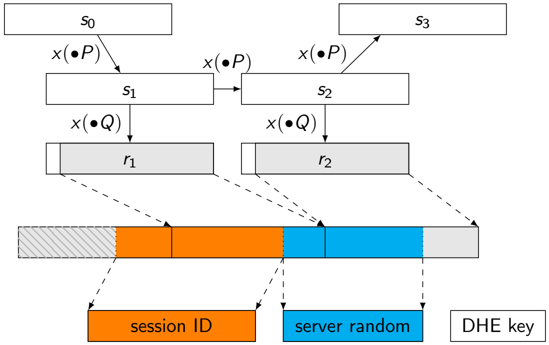
Attack — BSAFE-C



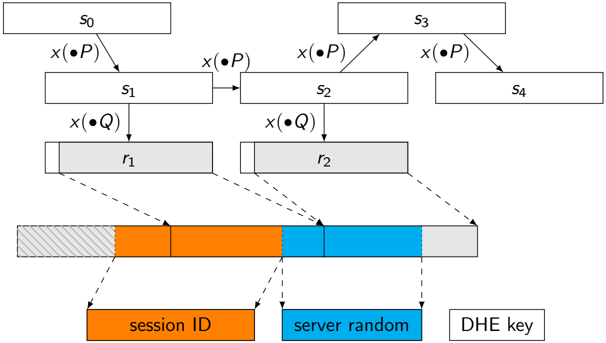
Attack — BSAFE-C



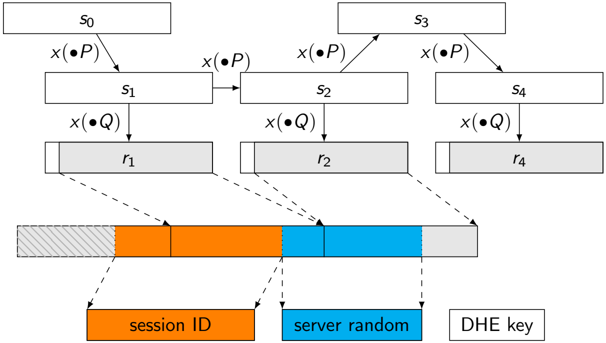
Attack — BSAFE-C



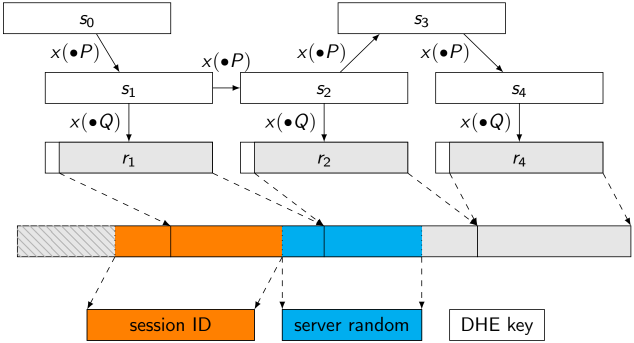
Attack — BSAFE-C



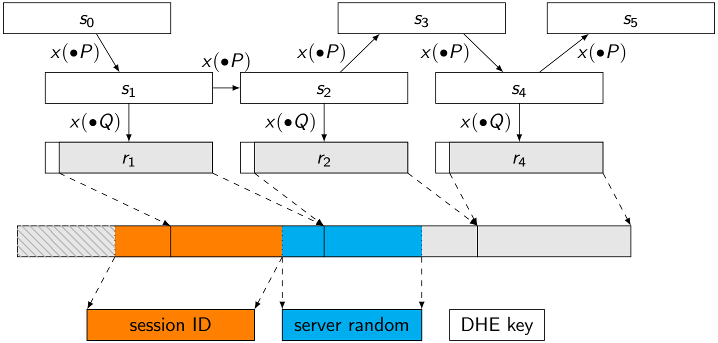
Attack — BSAFE-C



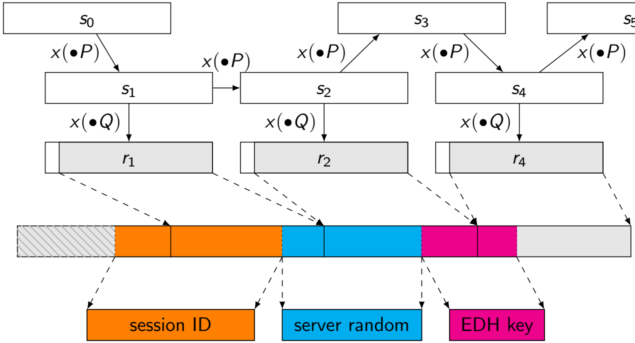
Attack — BSAFE-C



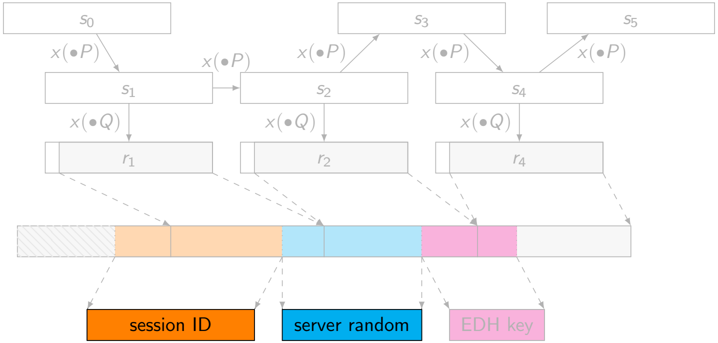
Attack — BSAFE-C



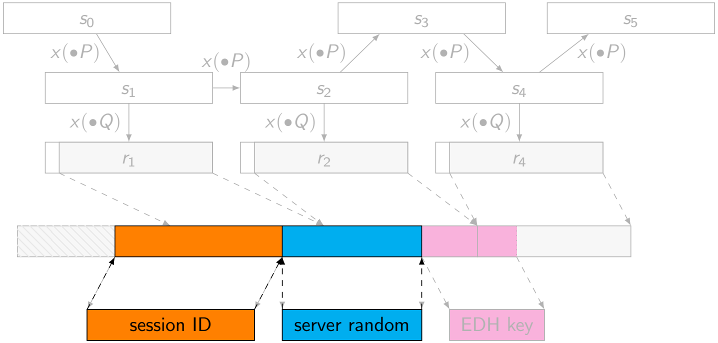
Attack — BSAFE-C



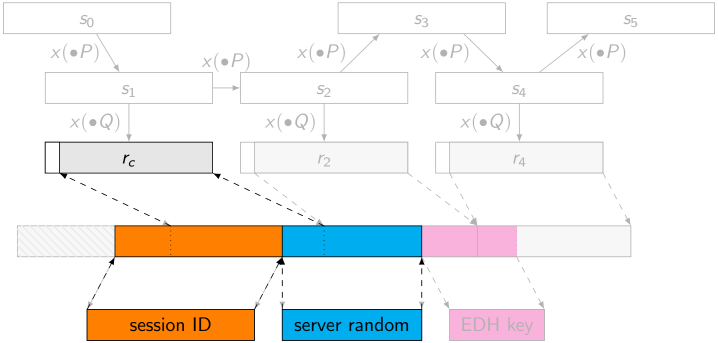
Attack — BSAFE-C



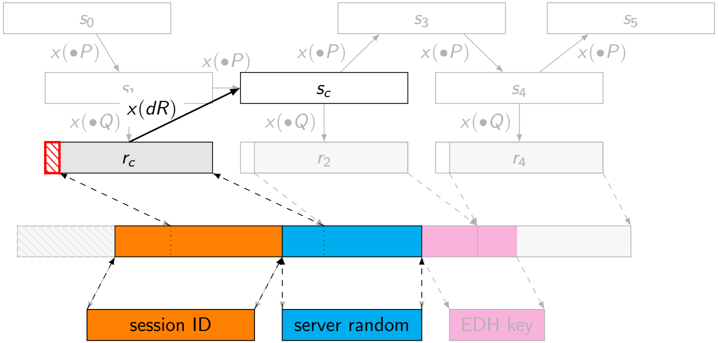
Attack — BSAFE-C



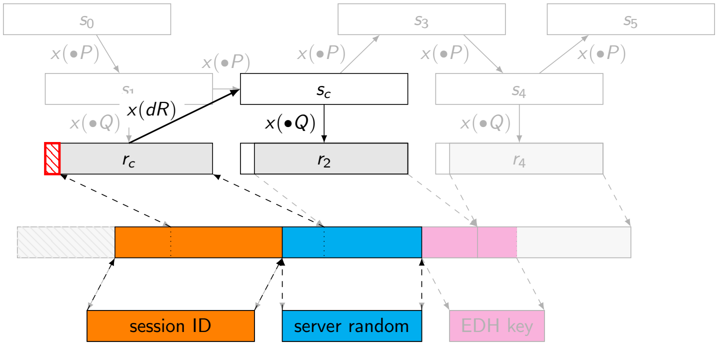
Attack — BSAFE-C



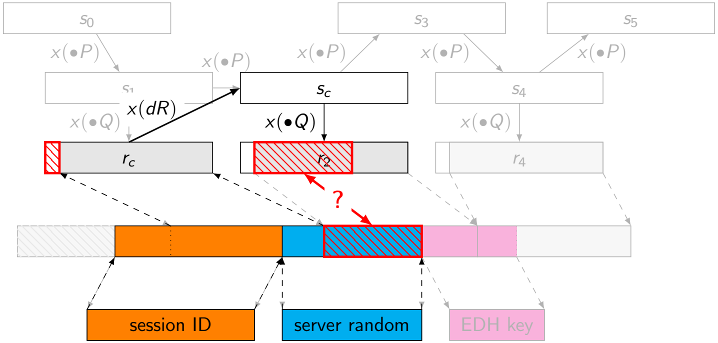
Attack — BSAFE-C



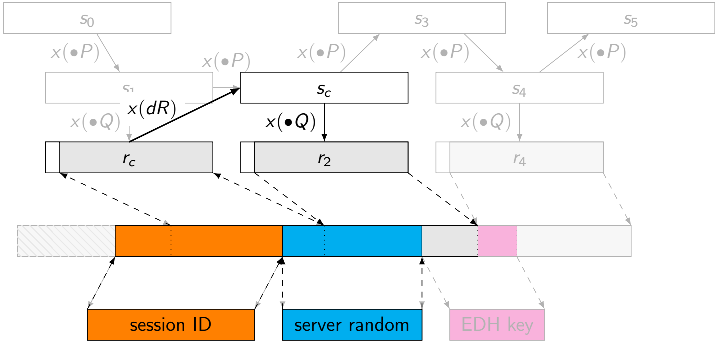
Attack — BSAFE-C



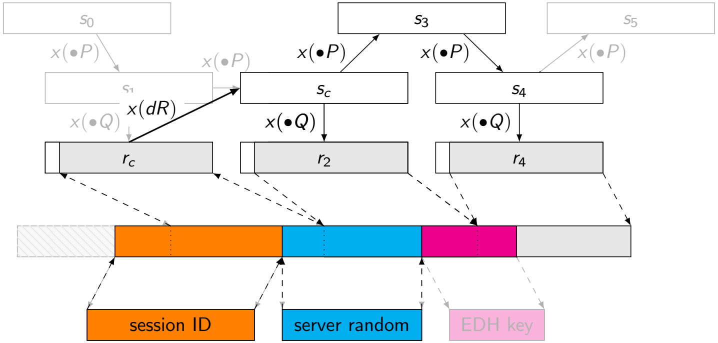
Attack — BSAFE-C



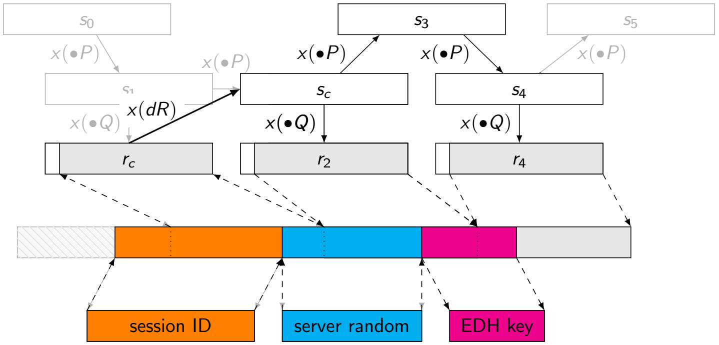
Attack — BSAFE-C



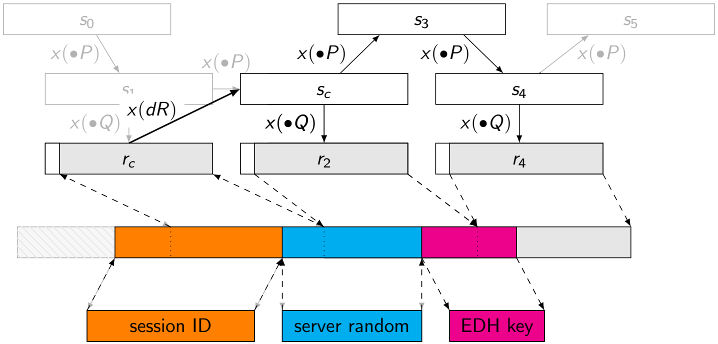
Attack — BSAFE-C



Attack — BSAFE-C

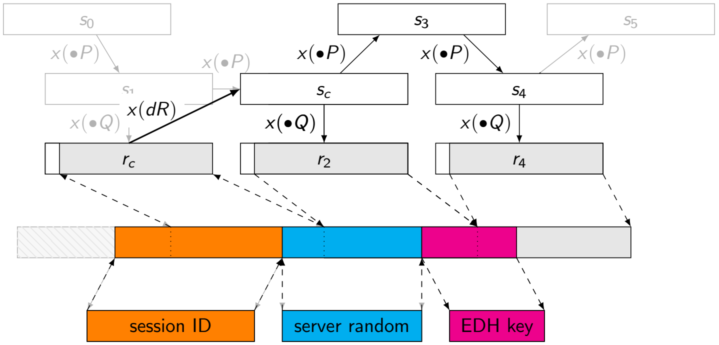


Attack — BSAFE-C



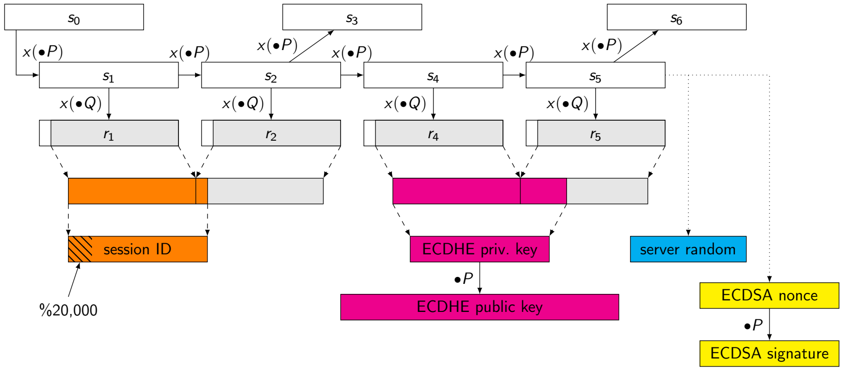
average cost: $2^{15}(C_v + C_f)$

Attack — BSAFE-C

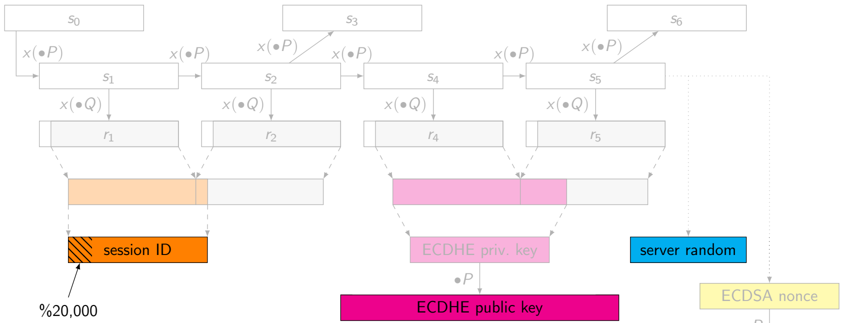


average cost: $30 \cdot 2^{15}(C_v + C_f)$

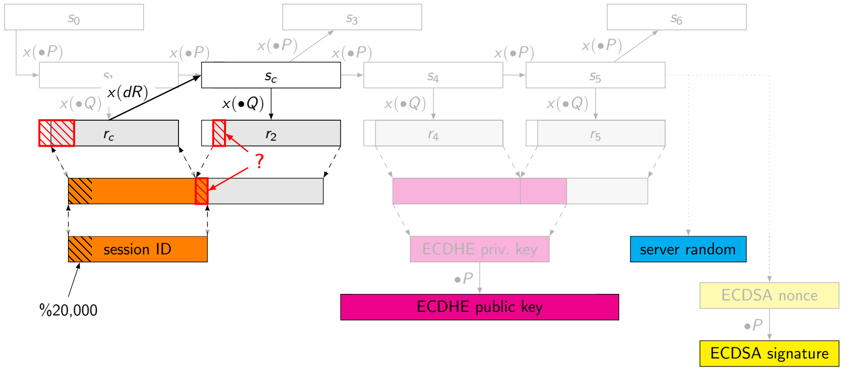
Attack — SChannel



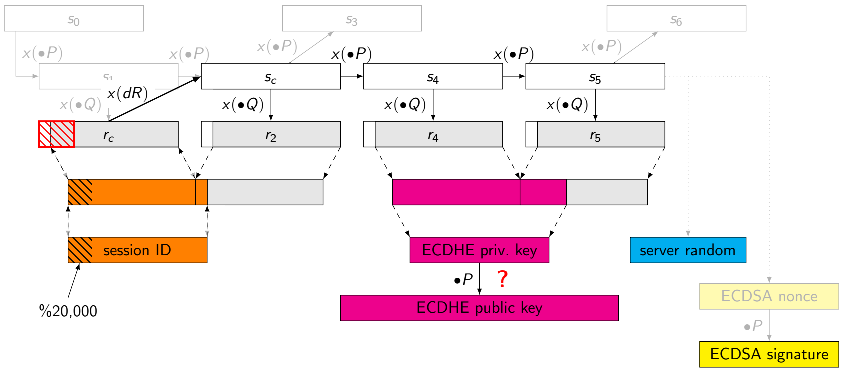
Attack — SChannel



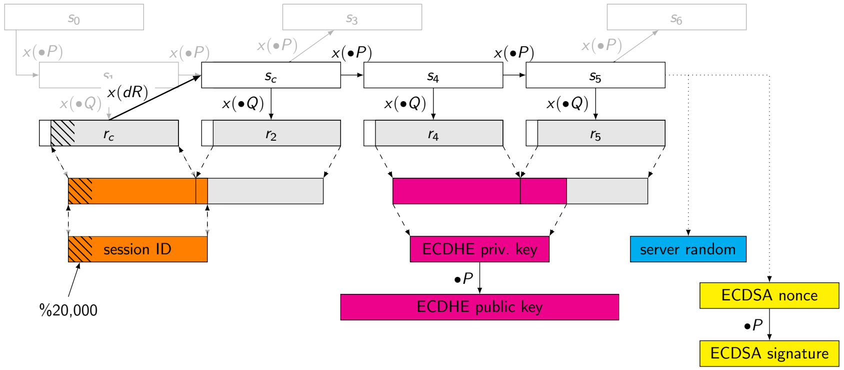
Attack — SChannel



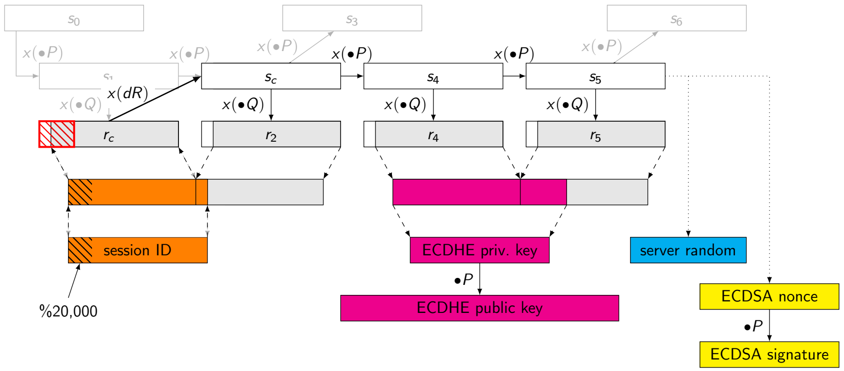
Attack — SChannel



Attack — SChannel

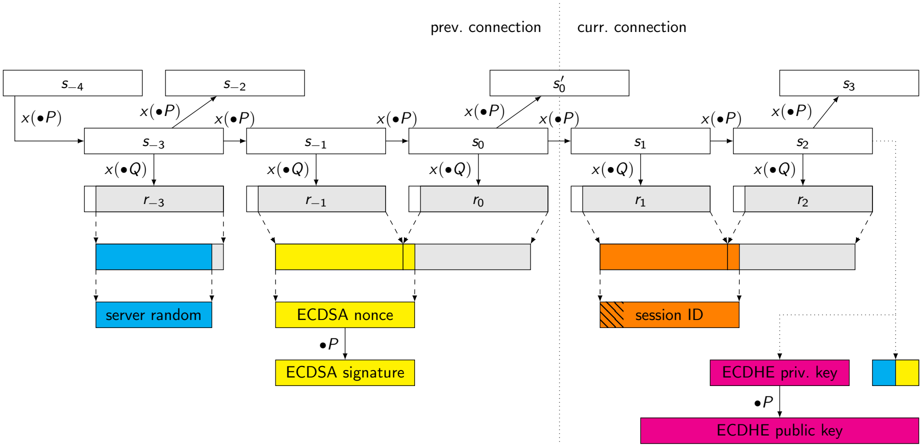


Attack — SChannel

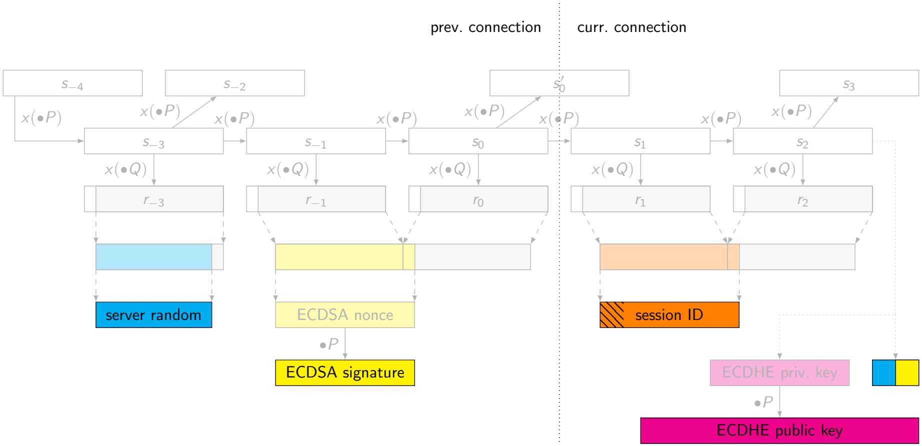


average cost: $2^{33}(C_v + C_f) + 2^{17}(5C_f)$

Attack — SChannel (cont.)

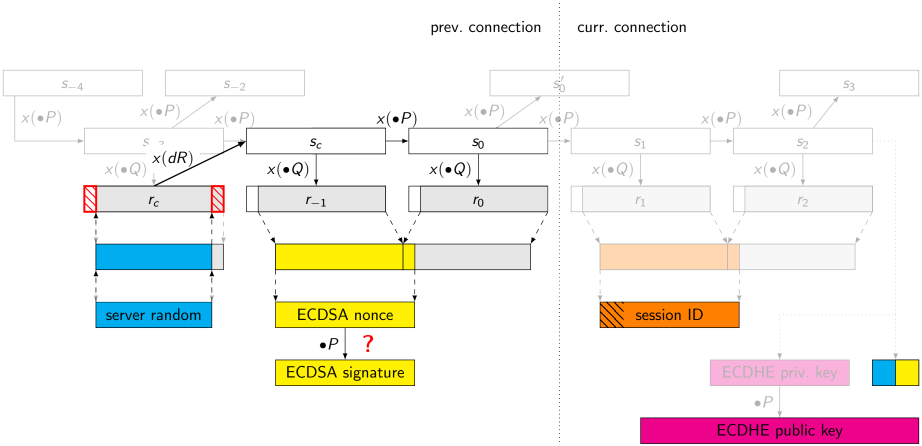


Attack — SChannel (cont.)

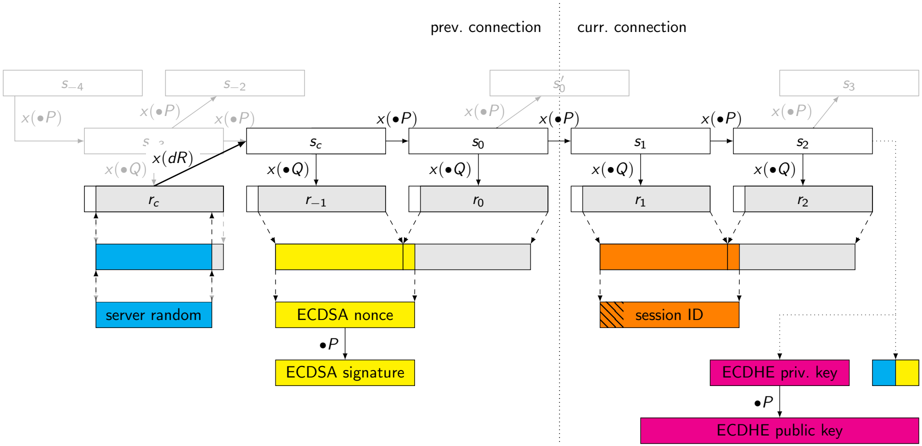


Technische Universiteit **Eindhoven** University of Technology

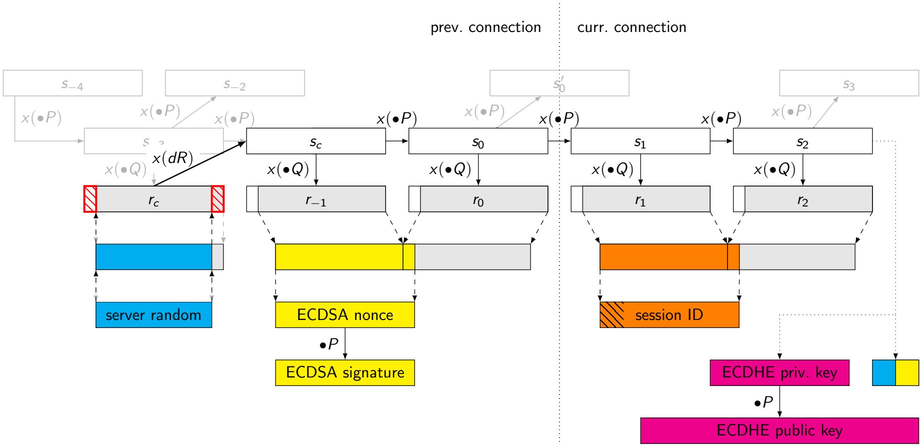
Attack — SChannel (cont.)



Attack — SChannel (cont.)

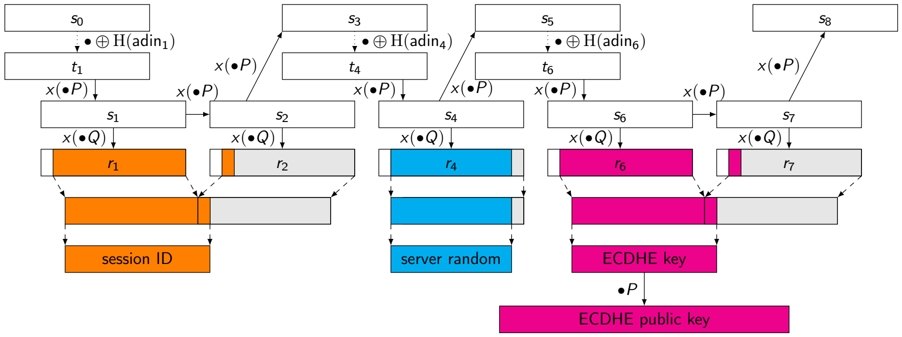


Attack — SChannel (cont.)

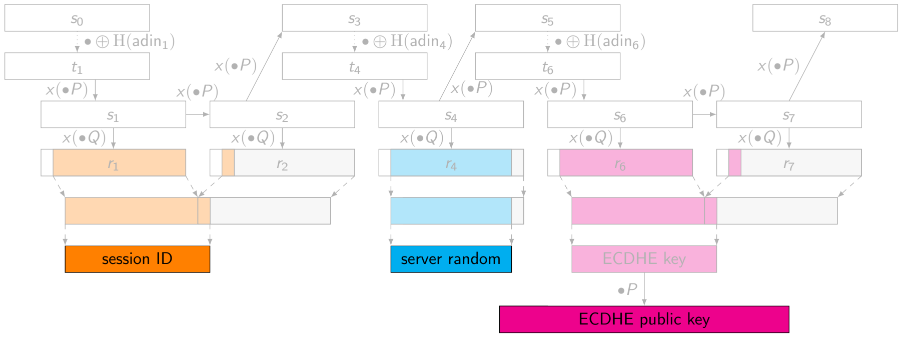


$$\text{average cost: } 2^{31}(C_v + 4C_f)$$

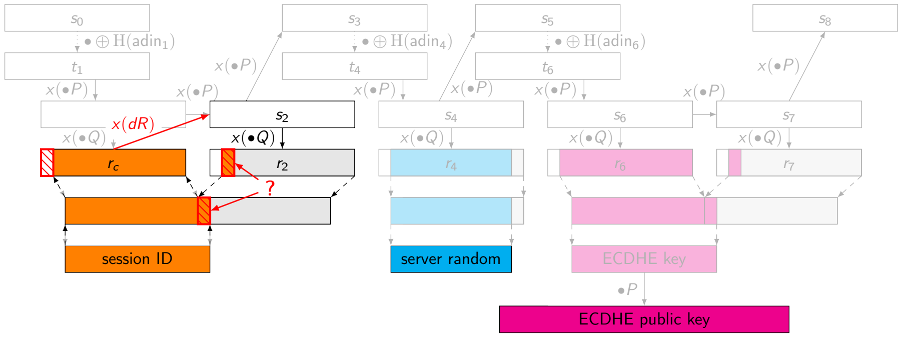
Attack — OpenSSL-fixed



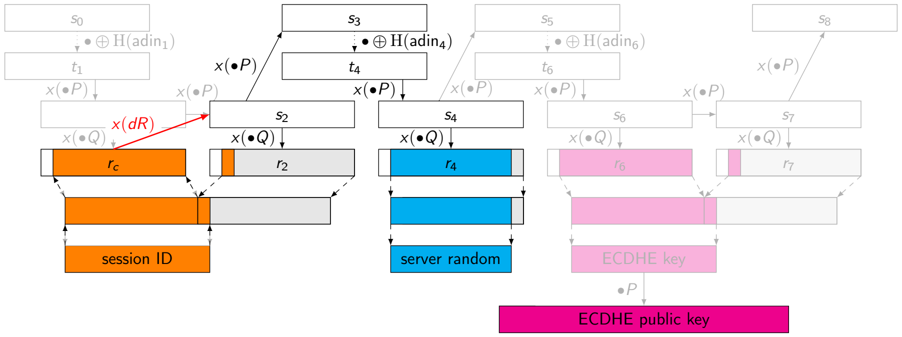
Attack — OpenSSL-fixed



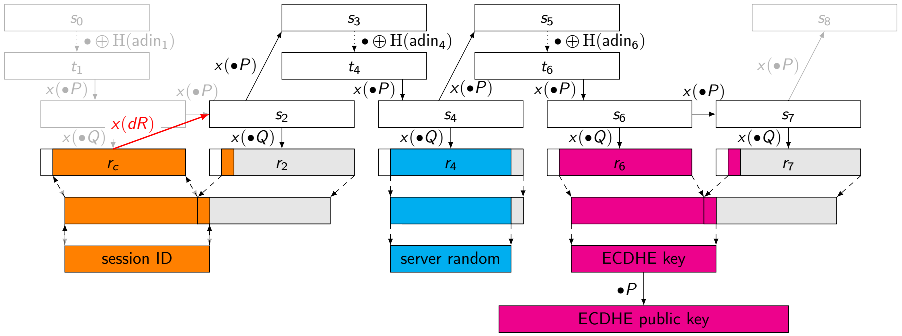
Attack — OpenSSL-fixed



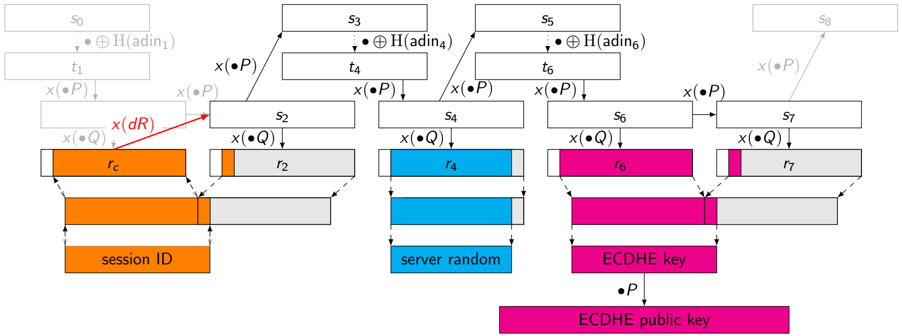
Attack — OpenSSL-fixed



Attack — OpenSSL-fixed



Attack — OpenSSL-fixed



average cost: $2^{15}(C_v + C_f) + 2^{20+k+l}(2C_f) + 2^{13}(5C_f)$

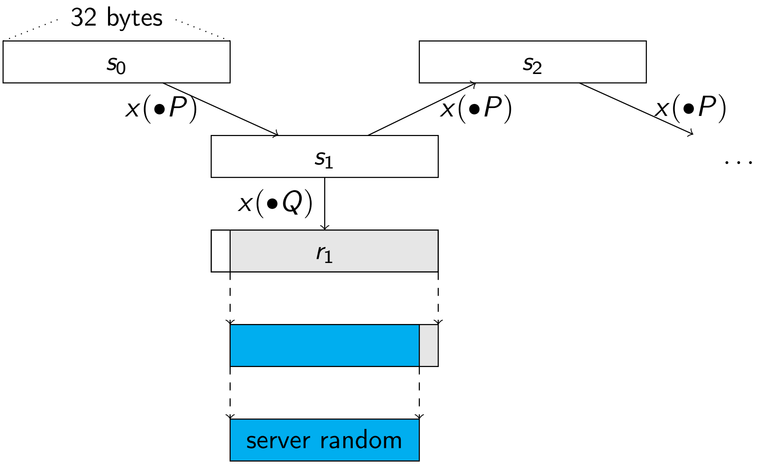
Attack — Attack Timings

	Intel Xeon CPU		16 $\times$ AMD CPU
Attack	Avg. Time (min)	# for 1s	Tot. Time (min)
BSAFE-C v1.1	0.26	16	0.04
BSAFE-Java v1.1	641	38,500	63.96
SChannel I	619	37,100	62.97
SChannel II	1,760	106,000	182.64
OpenSSL-fixed I	0.04	3	0.02
OpenSSL-fixed II	707	44,200	83.32
OpenSSL-fixed III	$2^k \cdot 707$	$2^k \cdot 44{,}200$	$2^k \cdot 83.32$

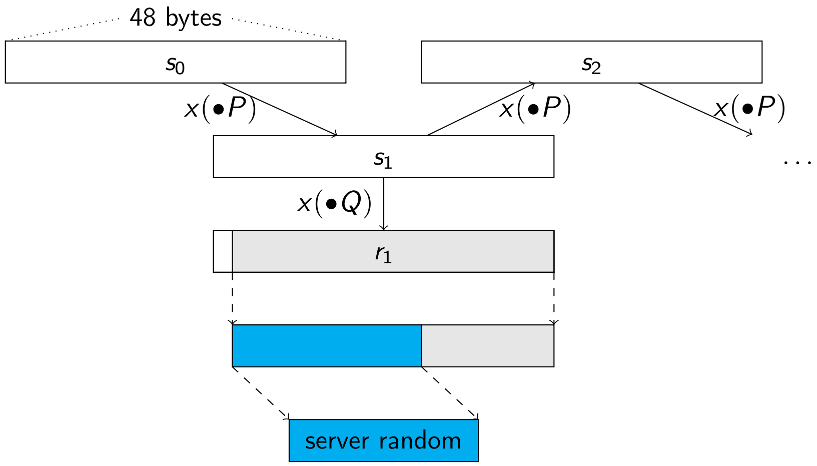
Attack — Attack Timings

Attack	Intel Xeon CPU Avg. Time (min)	Intel Xeon CPU # for 1s	16 $\times$ AMD CPU Tot. Time (min)
BSAFE-C v1.1	0.26	16	0.04
BSAFE-Java v1.1	641	38,500	63.96
SChannel I	619	37,100	62.97
SChannel II	1,760	106,000	182.64
OpenSSL-fixed I	0.04	3	0.02
OpenSSL-fixed II	707	44,200	83.32
OpenSSL-fixed III	$2^k \cdot 707$	$2^k \cdot 44{,}200$	$2^k \cdot 83.32$

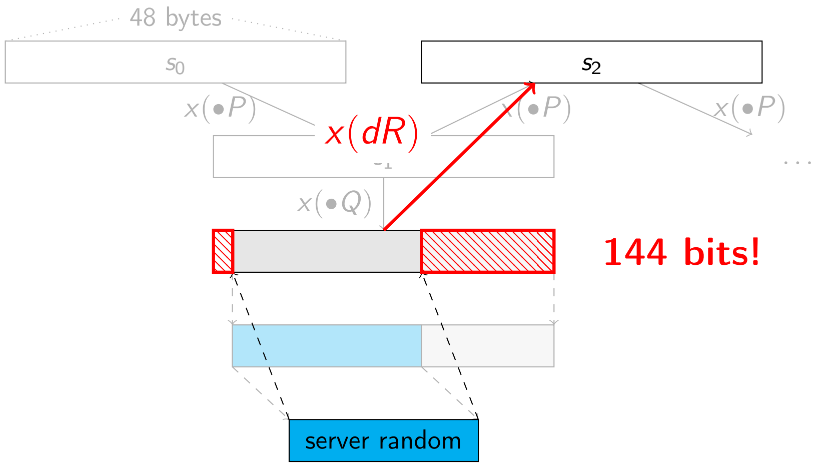
Dual EC with P-384 and P-521



Dual EC with P-384 and P-521



Dual EC with P-384 and P-521

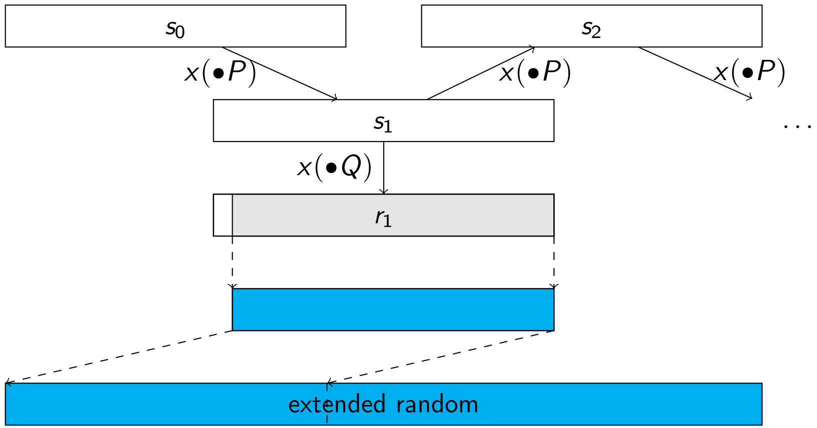


Draft Proposal – Extended Random

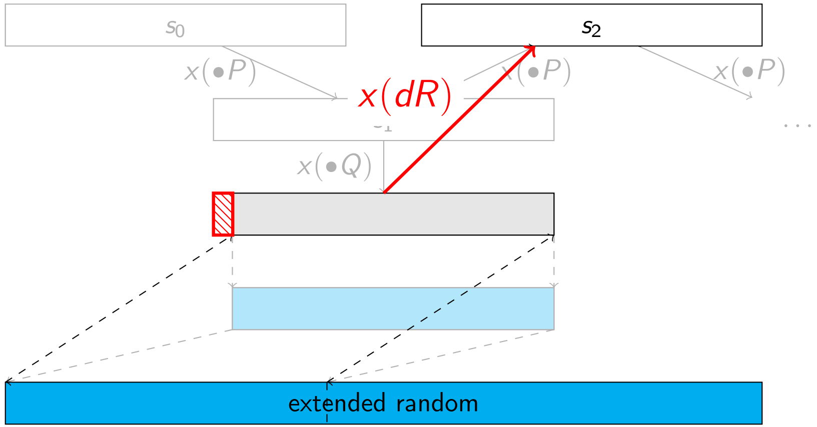
Draft for a proposed TLS extension named "Extended Random":

- allows client to request up to 2^{16} random bytes,
- has a weak motivation:
  `The rationale for this as stated by DoD is that the public randomness for each side should be at least twice as long as the security level for cryptographic parity, which makes the 224 bits of randomness provided by the current TLS random values insufficient.`
- was co-authored by an employee of NSA.

Draft Proposal – Extended Random



Draft Proposal – Extended Random



How did we get here ...

Official editors of SP800-90 are Elaine Barker and John Kelsey.

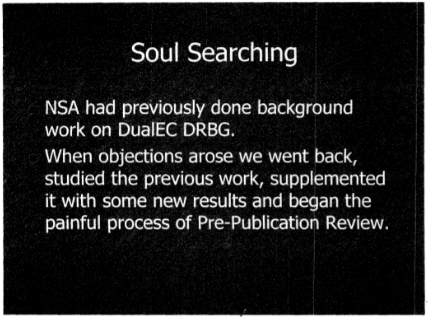
No editors stated for ANSI X9.82 nor for ISO 18031.

Interesting Dec 2013 slide deck by John Kelsey 800 – 90 and Dual EC DRBG.

- Standardization effort by NIST and NSA, with some participation from CSE.
- Most of work on standards done by US federal employees (NIST and NSA, with some help from CSE).
- The standard Dual EC parameters P and Q come ultimately from designers of Dual EC DRBG at NSA.

NIST FOIA

Two FOIA requests by Andrew Crocker and Nate Cardozo of EFF and Matthew Stoller and Rep. Alan Grayson. Files hosted by Matt Green at `https://github.com/matthewdgreen/nistfoia`.
Interesting documents, e.g.



Soul Searching

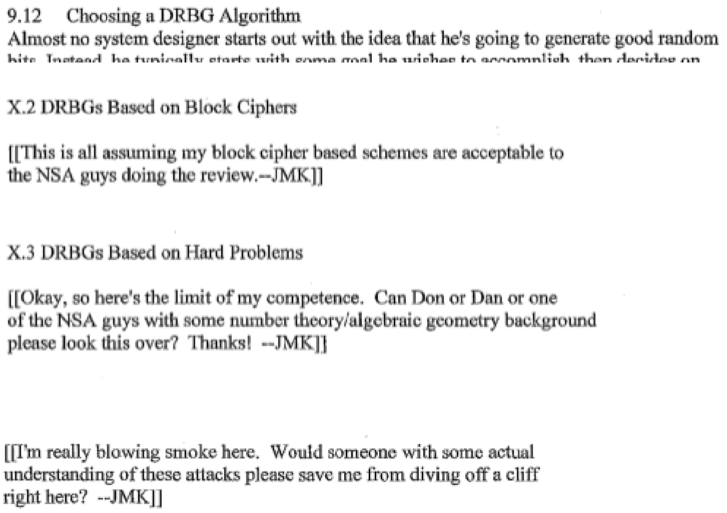
NSA had previously done background work on DualEC DRBG.

When objections arose we went back, studied the previous work, supplemented it with some new results and began the painful process of Pre-Publication Review.

This is most likely a reaction to the research on biases.

Draft Proposal – Extended Random

From 011 – 9.12 Choosing a DRBG Algorithm.pdf



9.12 Choosing a DRBG Algorithm

Almost no system designer starts out with the idea that he's going to generate good random bits. Instead, he typically starts with some goal he wishes to accomplish, then decides on

X.2 DRBGs Based on Block Ciphers

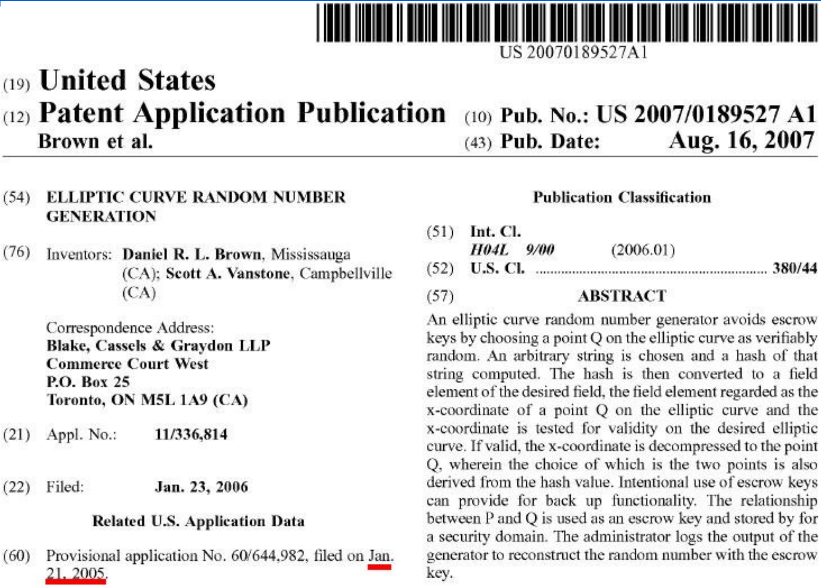
[[This is all assuming my block cipher based schemes are acceptable to the NSA guys doing the review.--JMK]]

X.3 DRBGs Based on Hard Problems

[[Okay, so here's the limit of my competence. Can Don or Dan or one of the NSA guys with some number theory/algebraic geometry background please look this over? Thanks! --JMK]]

[[I'm really blowing smoke here. Would someone with some actual understanding of these attacks please save me from diving off a cliff right here? --JMK]]

Draft Proposal – Extended Random



US 20070189527A1

(19) **United States**
(12) **Patent Application Publication**
Brown et al.

(10) **Pub. No.: US 2007/0189527 A1**
(43) **Pub. Date: Aug. 16, 2007**

(54) **ELLIPTIC CURVE RANDOM NUMBER GENERATION**

(76) Inventors: **Daniel R. L. Brown**, Mississauga (CA); **Scott A. Vanstone**, Campbellville (CA)

Correspondence Address:
Blake, Cassels & Graydon LLP
Commerce Court West
P.O. Box 25
Toronto, ON M5L 1A9 (CA)

(21) Appl. No.: **11/336,814**

(22) Filed: **Jan. 23, 2006**

Related U.S. Application Data

(60) Provisional application No. 60/644,982, filed on Jan. 21, 2005.

Publication Classification

(51) **Int. Cl.**
H04L 9/00 (2006.01)
(52) **U.S. Cl.** **380/44**

(57) **ABSTRACT**

An elliptic curve random number generator avoids escrow keys by choosing a point Q on the elliptic curve as verifiably random. An arbitrary string is chosen and a hash of that string computed. The hash is then converted to a field element of the desired field, the field element regarded as the x-coordinate of a point Q on the elliptic curve and the x-coordinate is tested for validity on the desired elliptic curve. If valid, the x-coordinate is decompressed to the point Q, wherein the choice of which is the two points is also derived from the hash value. Intentional use of escrow keys can provide for back up functionality. The relationship between P and Q is used as an escrow key and stored by for a security domain. The administrator logs the output of the generator to reconstruct the random number with the escrow key.

Hat tip @nymble.

Technische Universiteit Eindhoven University of Technology

Certicom patents

The Canadian company Certicom (now part of Blackberry) has patents in multiple countries on

- Dual EC exploitation: the use of Dual EC for key escrow (i.e., for a deliberate back door)
- Dual EC escrow avoidance: modifying Dual EC to avoid key escrow.

The patent filing history also shows that

- Certicom knew the Dual EC back door by 2005;
- NSA was informed of the Dual EC back door by 2005, even if they did not know it earlier;
- the patent application, including examples of Dual EC exploitation, was publicly available in July 2006, just a month after SP800-90 was standardized.

http://projectbullrun.org/dual-ec/patent.html

Summary

Dual EC — a standardized back door:

- (co-)authored by NSA,

Summary

Dual EC — a standardized back door:

- (co-)authored by NSA,
- may contain a back door (can neither be proven nor disproven),

Summary

Dual EC — a standardized back door:

- (co-)authored by NSA,
- may contain a back door (can neither be proven nor disproven),
- allows the back-door owner to compute all future random outputs,

Summary

Dual EC — a standardized back door:

- (co-)authored by NSA,
- may contain a back door (can neither be proven nor disproven),
- allows the back-door owner to compute all future random outputs,
- makes flaw in DSS a back door that allows impersonation,

Summary

Dual EC — a standardized back door:

- (co-)authored by NSA,
- may contain a back door (can neither be proven nor disproven),
- allows the back-door owner to compute all future random outputs,
- makes flaw in DSS a back door that allows impersonation,
- proven to be practical in various TLS libraries,

Summary

Dual EC — a standardized back door:

- (co-)authored by NSA,
- may contain a back door (can neither be proven nor disproven),
- allows the back-door owner to compute all future random outputs,
- makes flaw in DSS a back door that allows impersonation,
- proven to be practical in various TLS libraries,
- was default RNG in RSA's BSAFE library,

Summary

Dual EC — a standardized back door:

- (co-)authored by NSA,
- may contain a back door (can neither be proven nor disproven),
- allows the back-door owner to compute all future random outputs,
- makes flaw in DSS a back door that allows impersonation,
- proven to be practical in various TLS libraries,
- was default RNG in RSA's BSAFE library,
- back door becomes even stronger with proposed Extended Random,

Summary

Dual EC — a standardized back door:

- (co-)authored by NSA,
- may contain a back door (can neither be proven nor disproven),
- allows the back-door owner to compute all future random outputs,
- makes flaw in DSS a back door that allows impersonation,
- proven to be practical in various TLS libraries,
- was default RNG in RSA's BSAFE library,
- back door becomes even stronger with proposed Extended Random,
- it is not only standardized but even patented.

Summary

Dual EC — a standardized back door:

- (co-)authored by NSA,
- may contain a back door (can neither be proven nor disproven),
- allows the back-door owner to compute all future random outputs,
- makes flaw in DSS a back door that allows impersonation,
- proven to be practical in various TLS libraries,
- was default RNG in RSA's BSAFE library,
- back door becomes even stronger with proposed Extended Random,
- it is not only standardized but even patented.

How to fix it?

Summary

Dual EC — a standardized back door:

- (co-)authored by NSA,
- may contain a back door (can neither be proven nor disproven),
- allows the back-door owner to compute all future random outputs,
- makes flaw in DSS a back door that allows impersonation,
- proven to be practical in various TLS libraries,
- was default RNG in RSA's BSAFE library,
- back door becomes even stronger with proposed Extended Random,
- it is not only standardized but even patented.

Don't use Dual EC!