Lecture 9 – Diffie-Hellman key exchange II, elliptic curves, conspiracies

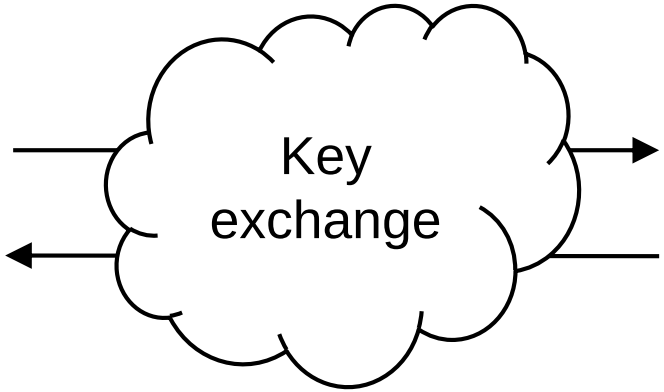
TEK4500

25.10.2023

Håkon Jacobsen

hakon.jacobsen@its.uio.no

Diffie-Hellman



Recap: groups

Definition: A **group** $(G,\circ)$ is a set G together with a binary operation $\circ$ satisfying the axioms

$\mathcal{G}1$: $(a \circ b) \circ c = a \circ (b \circ c)$ (associativity)
$\mathcal{G}2$: $\exists e \in G : e \circ a = a \circ e$ (identity)
$\mathcal{G}3$: $\exists a^{-1} \in G : a \circ a^{-1} = a^{-1} \circ a = e$ (inverse)

Definition: A group is **cyclic** if there exists $g \in G$ such that $G = \{\dots, g^{-2}, g^{-1}, g^0, g^1, g^2, g^3, \dots\} = \langle g \rangle$

Examples:

$(\boldsymbol{Z}, +) = \langle 1 \rangle = \langle -1 \rangle$

$(\boldsymbol{Z}_n, +_n) = \langle 1 \rangle$

$(\boldsymbol{Z}_p^*, \cdot)$ $(\boldsymbol{Z}_7^*, \cdot) = \langle 3 \rangle = \langle 5 \rangle = \{1, 3, 2, 6, 4, 5\}$

Not cyclic groups:

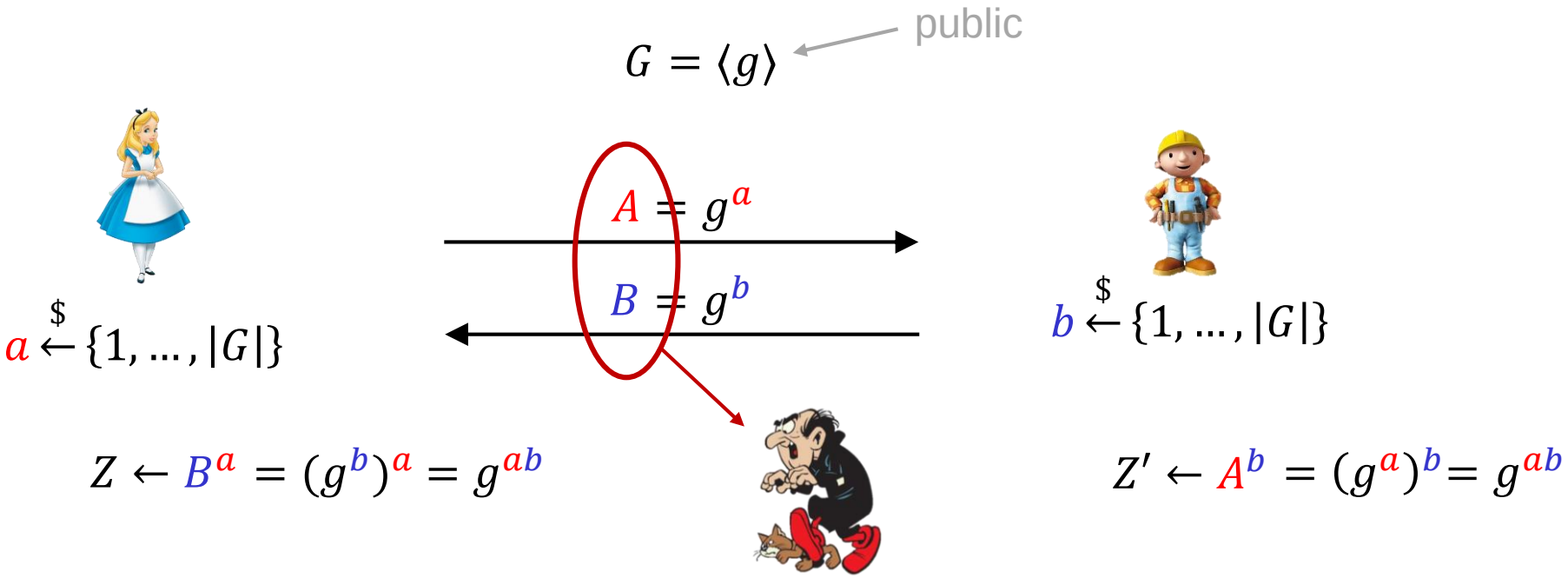
$(\boldsymbol{R}, +)$

$(\boldsymbol{R}^*, \cdot)$

$(\boldsymbol{G}, \star)$

$\star$	e	a	b	c
e	e	a	b	c
a	a	e	c	b
b	b	c	e	a
c	c	b	a	e

Diffie-Hellman



Security:

- Must be hard to compute $Z \leftarrow g^{ab}$ given g, A, B (DH assumption)
- Must be hard to find a (or b) given g, A, B (DLOG assumption)

Solving DLOG: the baby-step giant-step algorithm

Given: $X \leftarrow g^x$ $\quad n \leftarrow \left\lceil \sqrt{|G|} \right\rceil$ $\quad Y \leftarrow g^n$

Find: x

Look for i, j such that:

$$Xg^i = Y^j$$

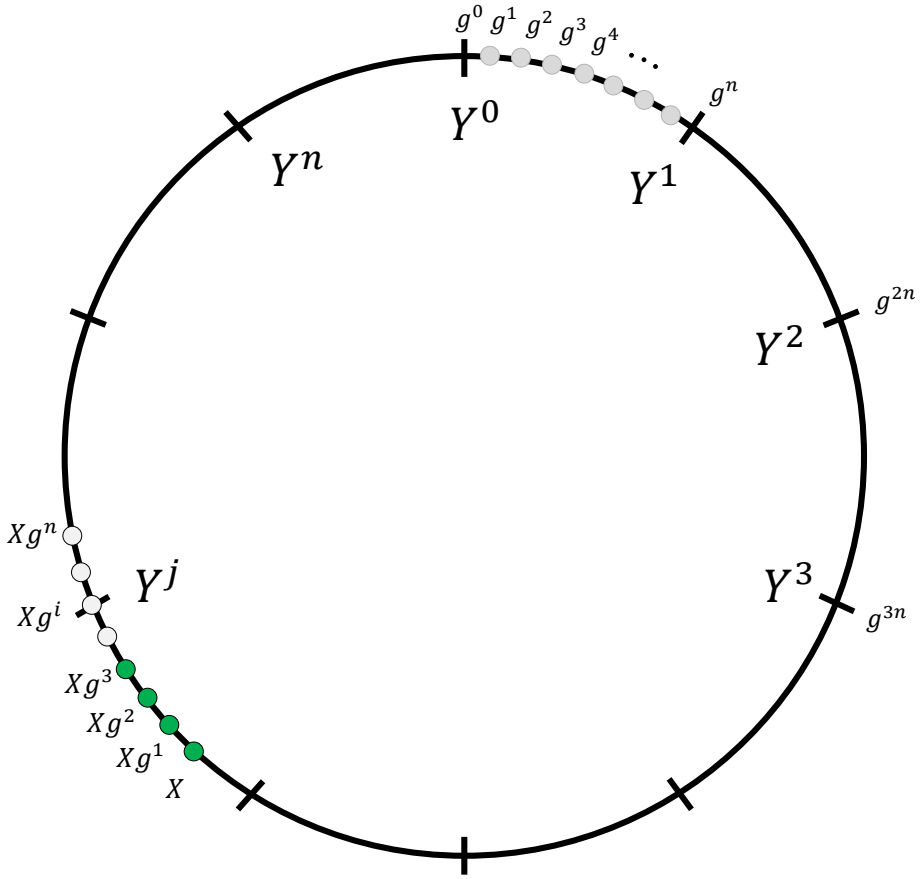
$$X = Y^j g^{-i}$$

$$X = g^{nj} g^{-i}$$

$$g^x = g^{nj} g^{-i}$$

$$g^x = g^{nj-i}$$

$$x = nj - i$$



$$\underbrace{Y^0, Y^1, \ldots, Y^n}_{\mathcal{O}\left(\sqrt{|G|}\right)}$$

$$\underbrace{Xg^0, Xg^1, \ldots, Xg^n}_{\tilde{\mathcal{O}}\left(\sqrt{|G|}\right)}$$

Time + memory: $\tilde{\mathcal{O}}\left(\sqrt{|G|}\right)$

Generic algorithms for solving DLOG

- Baby-step, giant-step: time $\mathcal{O}\left(\sqrt{|G|}\right)$ memory $\mathcal{O}\left(\sqrt{|G|}\right)$
- Pollard's rho: time $\mathcal{O}\left(\sqrt{|G|}\right)$ memory $\mathcal{O}(1)$
- Pohlig-Hellman: time $\max_p \mathcal{O}(\sqrt{p})$ memory $\mathcal{O}(1)$ $\left(|G| = p_1^{e_1} p_2^{e_2} \cdots p_t^{e_t}\right)$

- **Consequence:** $\sqrt{|G|}$ must be large enough for DLOG to be hard !
  - $|G| \approx 2^{128}$ only gives $\sqrt{2^{128}} = 2^{64}$ security
  - $|G| \approx 2^{256}$ only gives $\sqrt{2^{256}} = 2^{128}$ security
  - $|G| \approx 2^{512}$ only gives $\sqrt{2^{512}} = 2^{256}$ security
  - etc...

Non-generic algorithms for DLOG

- Unfortunately, $\left(\boldsymbol{Z}_p^*,\cdot\right)$ is *not* a generic group!

- *Much* faster specific algorithms exist for solving DLOG in $\boldsymbol{Z}_p^*$
  - Index-calculus
  - Elliptic-curve method
  - Special number-field sieve (SNFS)
  - General number-field sieve (GNFS)

  } *exceptionally* complicated algorithms, requiring very advanced mathematics!

- Current DLOG-solving record: $\left|\boldsymbol{Z}_p^*\right| \approx 2^{795}$ using GNFS (Heninger et al. '19)
  - Previous records: https://en.wikipedia.org/wiki/Discrete_logarithm_records

- $\left|\boldsymbol{Z}_p^*\right| \geq 2^{2048}$ typically required as a minimum today

Diffie-Hellman in $(Z_p^*, \cdot)$ – what is large enough?

$p = 171254583176141379301960419792575778264088323240375085733932929816426671397476217788024387752387285929683446135893799323484756135034769321631669738132186983438164632891441853629126025225404949830905314972329658295365245072698488256583114202993359222957097432675083225259667739503949192575768420387716327420441424710535098501236058838158571626669177751934961573726561955583057270098912760065140004093658772181713883199238963093777917625906143118496429613802248519404604217104493689272529748703958739363879096722748832953774810081504758785902705917983505634881680809238046118223875201980540029906239114543891047740921 83 \approx 2^{2048}$

$A = 2^{3231700607131100730033891392642382824881794124114023911284200975140074170663435422261968941736356934711790173790970419175460587320919502885375898618562215321217541251490177452027023579607823624888842461894775876411059286460994117232454266225221932305409190376805242355191256797158701170010580558776510388618472802579760549035697325615261670813393617995413364765591603683178967290731783845896806396719009772021941686472258710314113364293195361934716365332097170774482279885885653692086452966360772502689555059283627511211740969729980684105543595848665832916421362182310789909994486524682624169720359118525070453610905 59} \bmod p$

$3231700607131100730033891392642382824881794124114023911284200975140074170663435422261968941736356934711790173790970419175460587320919502885375898618562215321217541251490177452027023579607823624888842461894775876411059286460994117232454266225221932305409190376805242355191256797158701170010580558776510388618472802579760549035697325615261670813393617995413364765591603683178967290731783845896806396719009772021941686472258710314113364293195361934716365332097170774482279885885653692086452966360772502689555059283627511211740969729980684105543595848665832916421362182310789909994486524682624169720359118525070453610905 59 \overset{\$}{\leftarrow} \{1 \ldots p\}$

$B = 2^{6656800778101007340704764168984174080206703524413789679972241019001939572118548275696769332739359829978356380673807628090243118427154966601139766131314780517844873224468096081960318036912634861709126936255232497423832644764339784094340047703951670116142172795521980299361080233986439997465392018349331682208647898808374845365630679977612708266422235394375417910584167316831769953658998677202835904209324173779271068847731990805174776156086754909822550989805452920329083580939135314337020494290474130715255563072995414455143897212222563808337436279913337627773071183170408853057899100942935484815501309561942770687524} \bmod p$

$6656800778101007340704764168984174080206703524413789679972241019001939572118548275696769332739359829978356380673807628090243118427154966601139766131314780517844873224468096081960318036912634861709126936255232497423832644764339784094340047703951670116142172795521980299361080233986439997465392018349331682208647898808374845365630679977612708266422235394375417910584167316831769953658998677202835904209324173779271068847731990805174776156086754909822550989805452920329083580939135314337020494290474130715255563072995414455143897212222563808337436279913337627773071183170408853057899100942935484815501309561942770687524 \overset{\$}{\leftarrow} \{1 \ldots p\}$

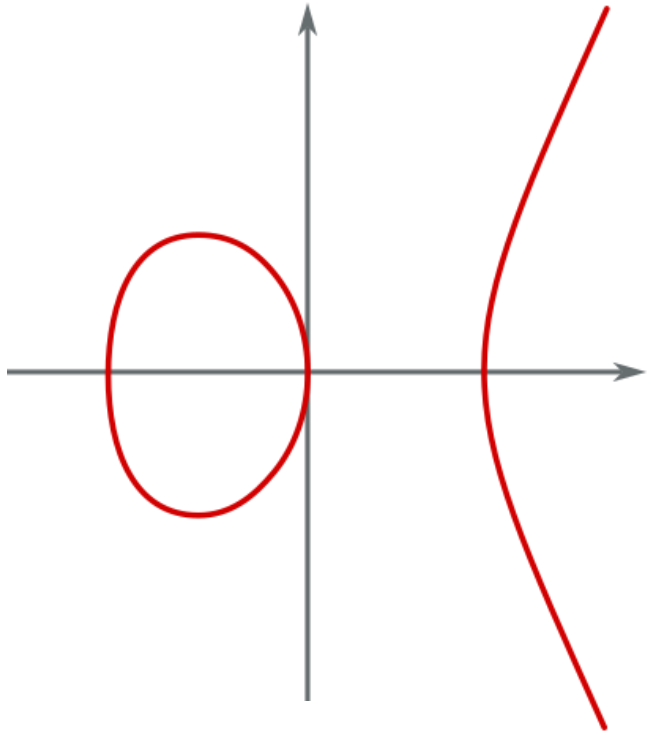
$Z \leftarrow 2^{3231700607131100730033891392642382824881794124114023911284200975140074170663435422261968941736356934711790173790970419175460587320919502885375898618562215321217541251490177452027023579607823624888842461894775876411059286460994117232454266225221932305409190376805242355191256797158701170010580558776510388618472802579760549035697325615261670813393617995413364765591603683178967290731783845896806396719009772021941686472258710314113364293195361934716365332097170774482279885885653692086452966360772502689555059283627511211740969729980684105543595848665832916421362182310789909994486524682624169720359118525070453610905 59 \times 6656800778101007340704764168984174080206703524413789679972241019001939572118548275696769332739359829978356380673807628090243118427154966601139766131314780517844873224468096081960318036912634861709126936255232497423832644764339784094340047703951670116142172795521980299361080233986439997465392018349331682208647898808374845365630679977612708266422235394375417910584167316831769953658998677202835904209324173779271068847731990805174776156086754909822550989805452920329083580939135314337020494290474130715255563072995414455143897212222563808337436279913337627773071183170408853057899100942935484815501309561942770687524} \bmod p$

Z_p^*

Group where GNFS doesn't work

Better alternatives to Z_p^*?

Elliptic curves



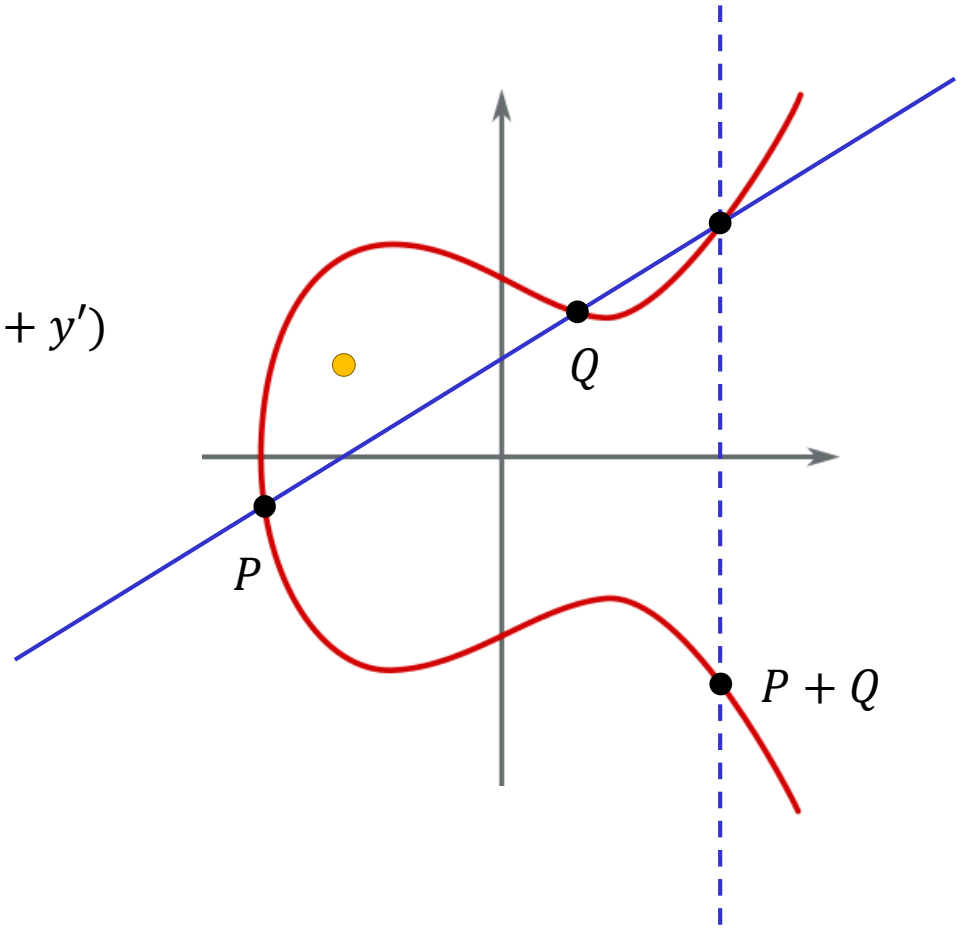
$$y^2 = x^3 - 9x - 1 \qquad x, y \in \boldsymbol{R}$$

Elliptic curves

$P = (x, y)$

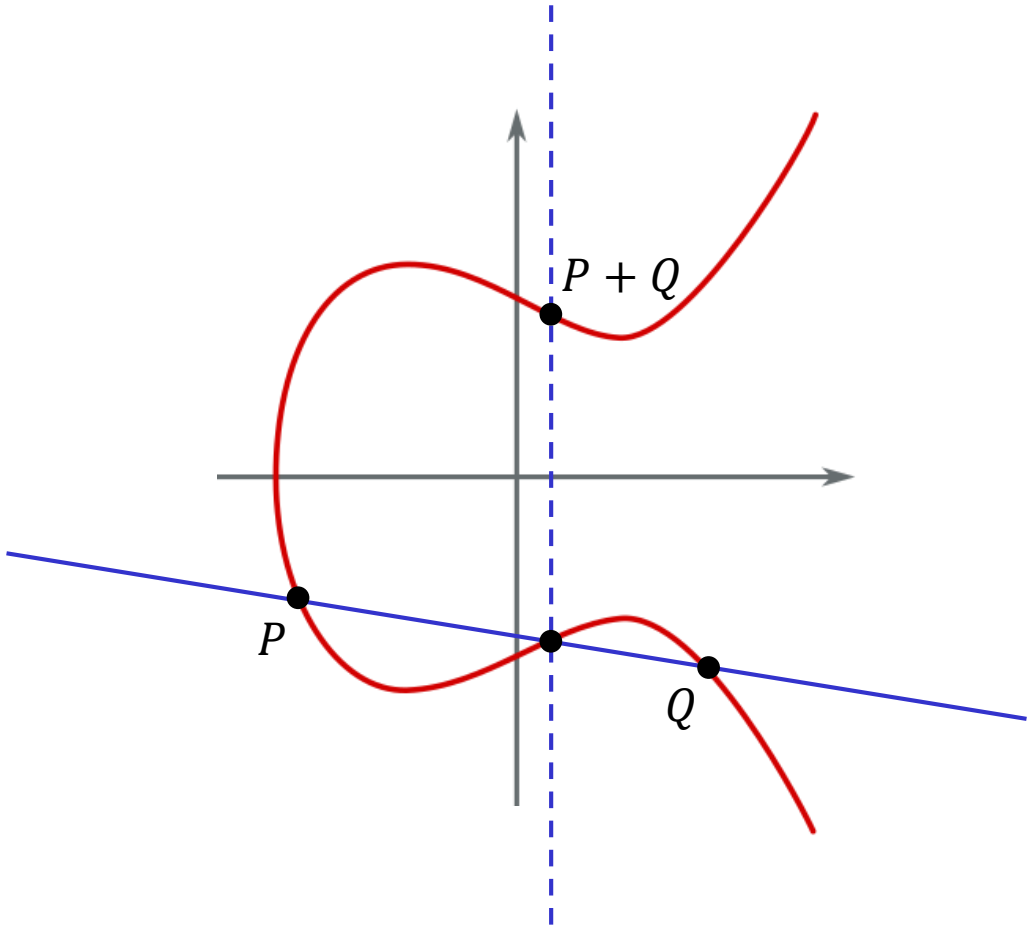
$Q = (x', y')$

$P + Q \neq (x + x', y + y')$



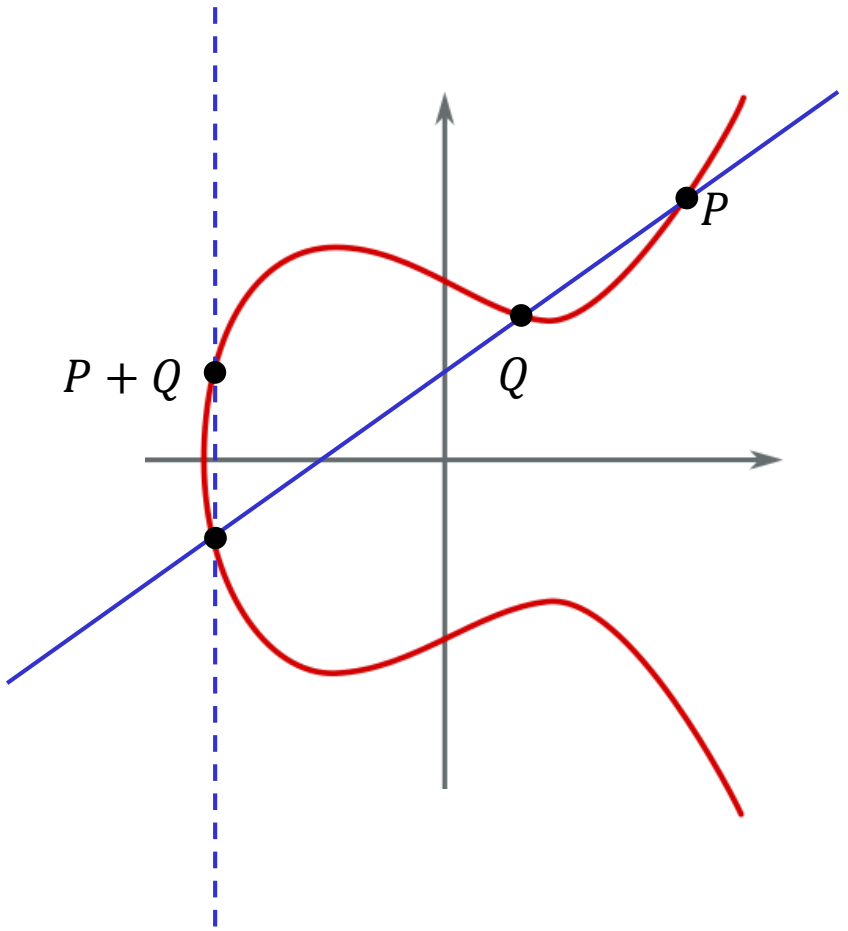
$y^2 = x^3 - 2x + 3$ $\quad x, y \in \boldsymbol{R}$

Elliptic curves



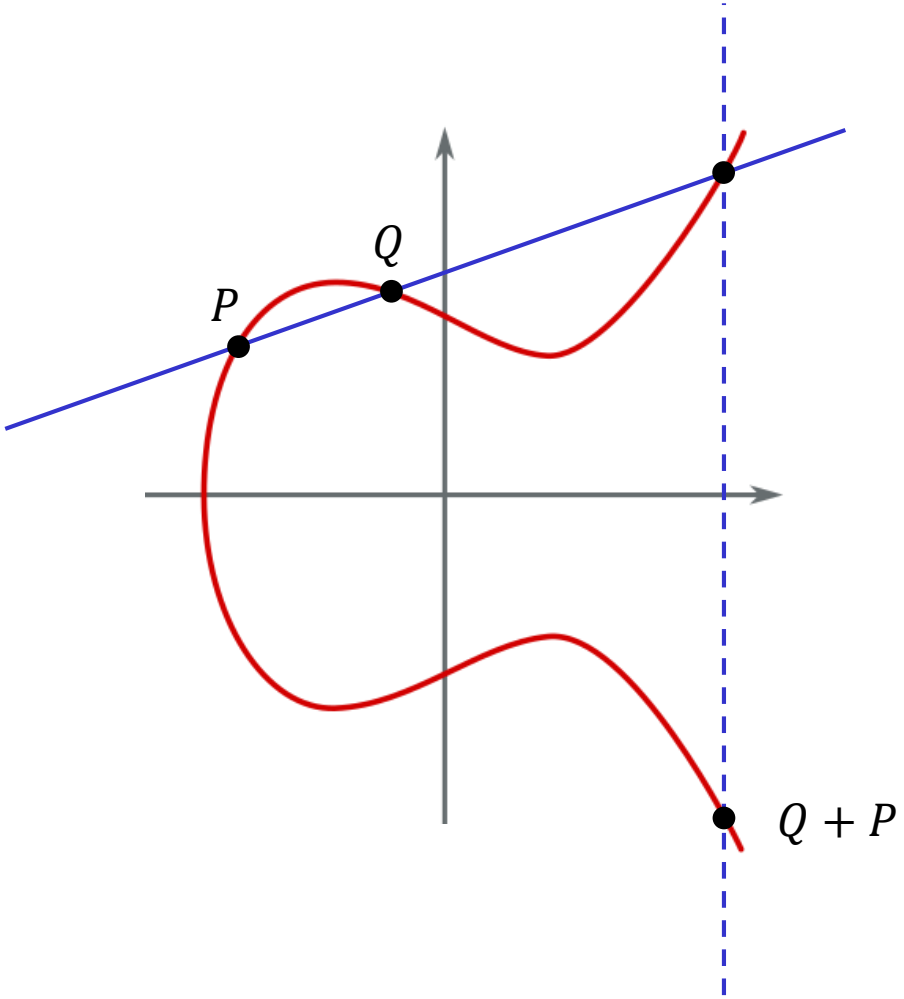
$$y^2 = x^3 - 2x + 3 \qquad x, y \in \boldsymbol{R}$$

Elliptic curves



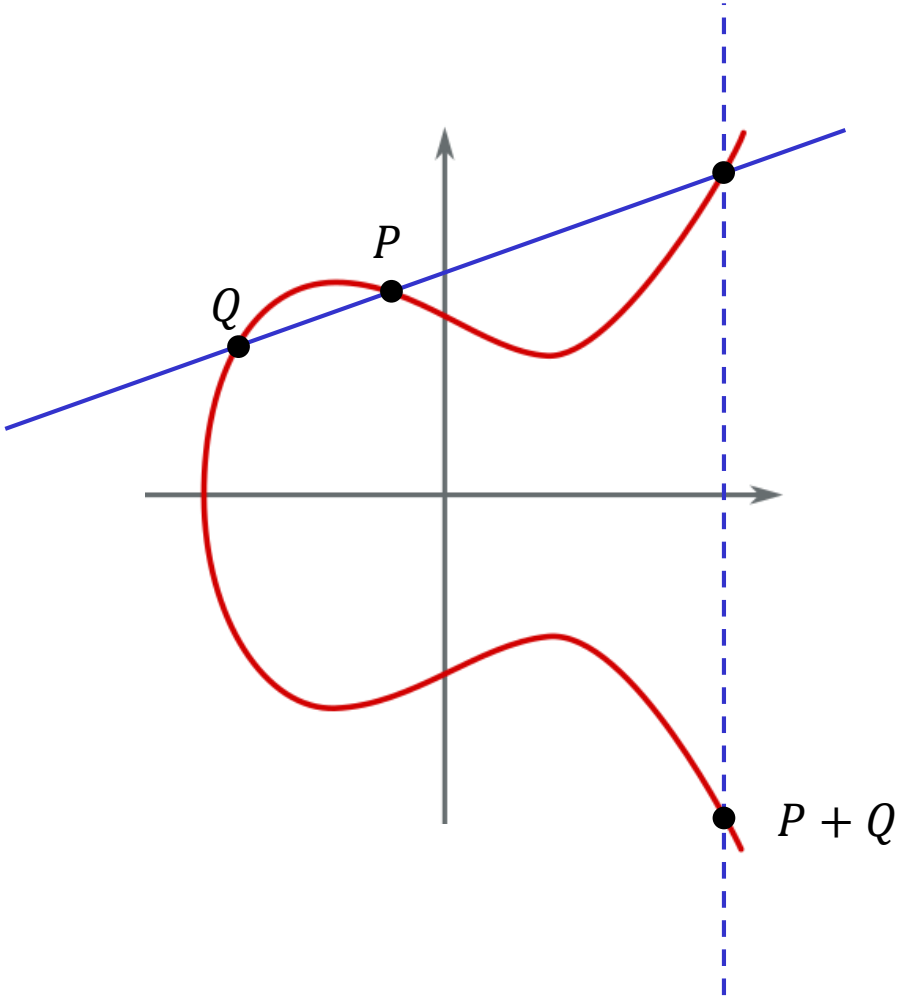
$y^2 = x^3 - 2x + 3 \qquad x, y \in \boldsymbol{R}$

Elliptic curves



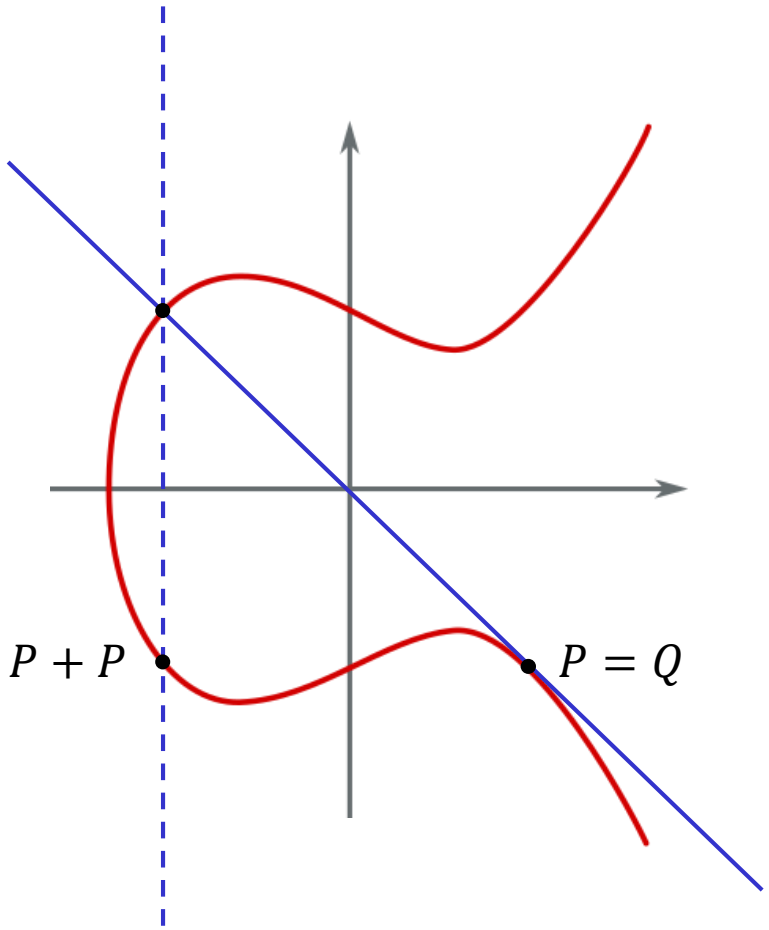
$y^2 = x^3 - 2x + 3 \qquad x, y \in \boldsymbol{R}$

Elliptic curves



$y^2 = x^3 - 2x + 3 \qquad x, y \in \boldsymbol{R}$

Elliptic curves

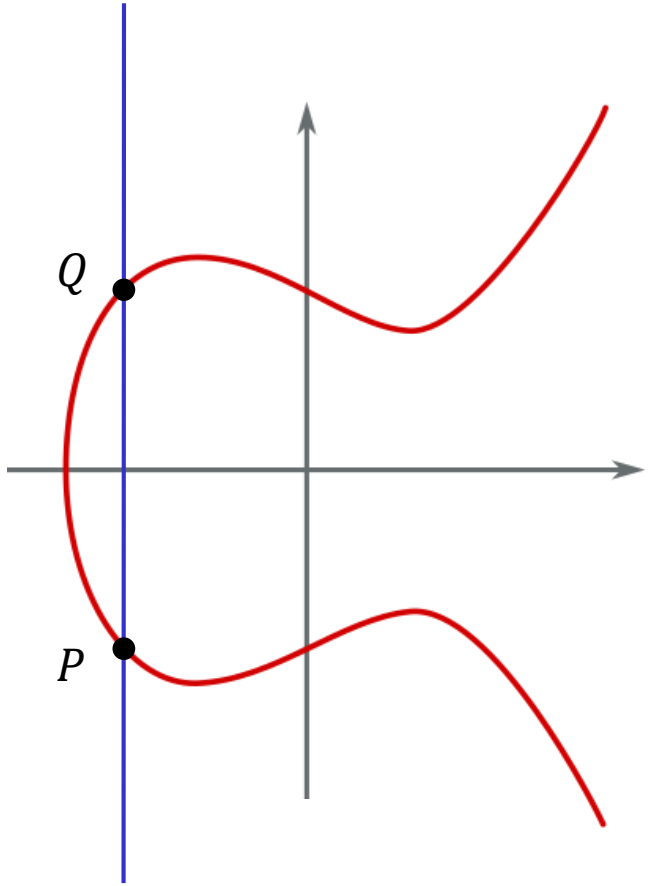


$$y^2 = x^3 - 2x + 3 \qquad x, y \in \boldsymbol{R}$$

Elliptic curves

$P + Q \overset{\text{def}}{=} \mathcal{O}$

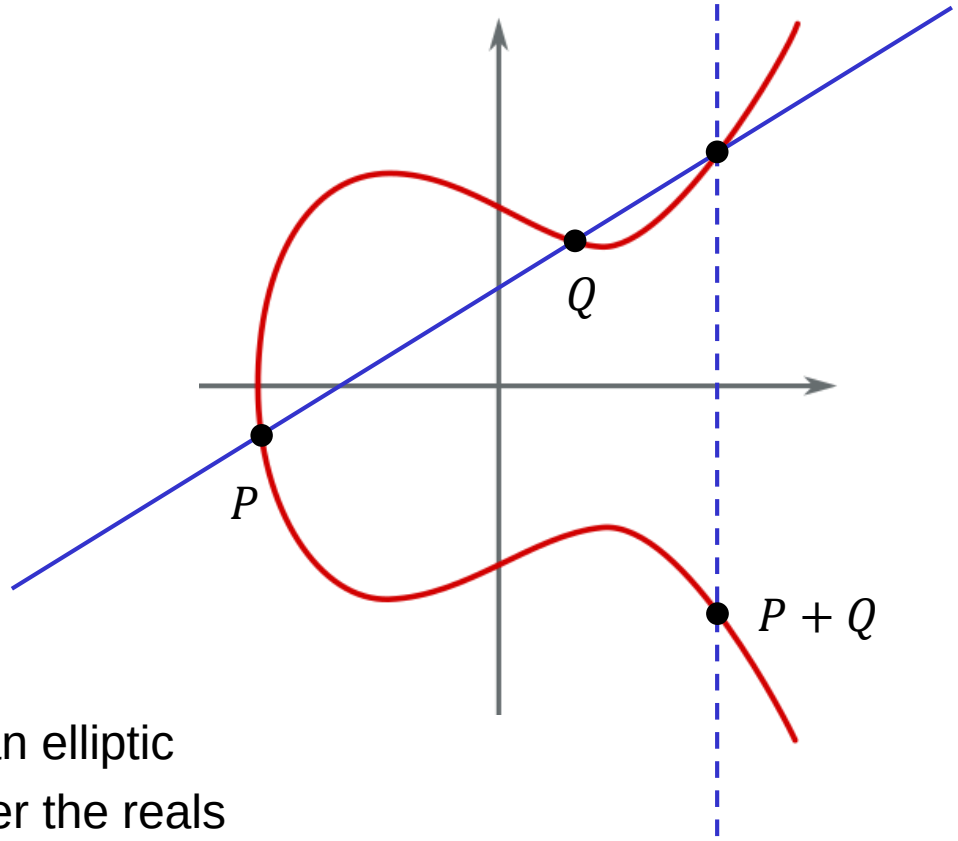
Identity element



$y^2 = x^3 - 2x + 3$ $\quad x, y \in \boldsymbol{R}$

Elliptic curves

Theorem: the points on an elliptic curve, together with $\mathcal{O}$, is an abelian group under "geometric point addition"



$$y^2 = x^3 + ax + b$$

$$a, b, x, y \in \boldsymbol{R}$$

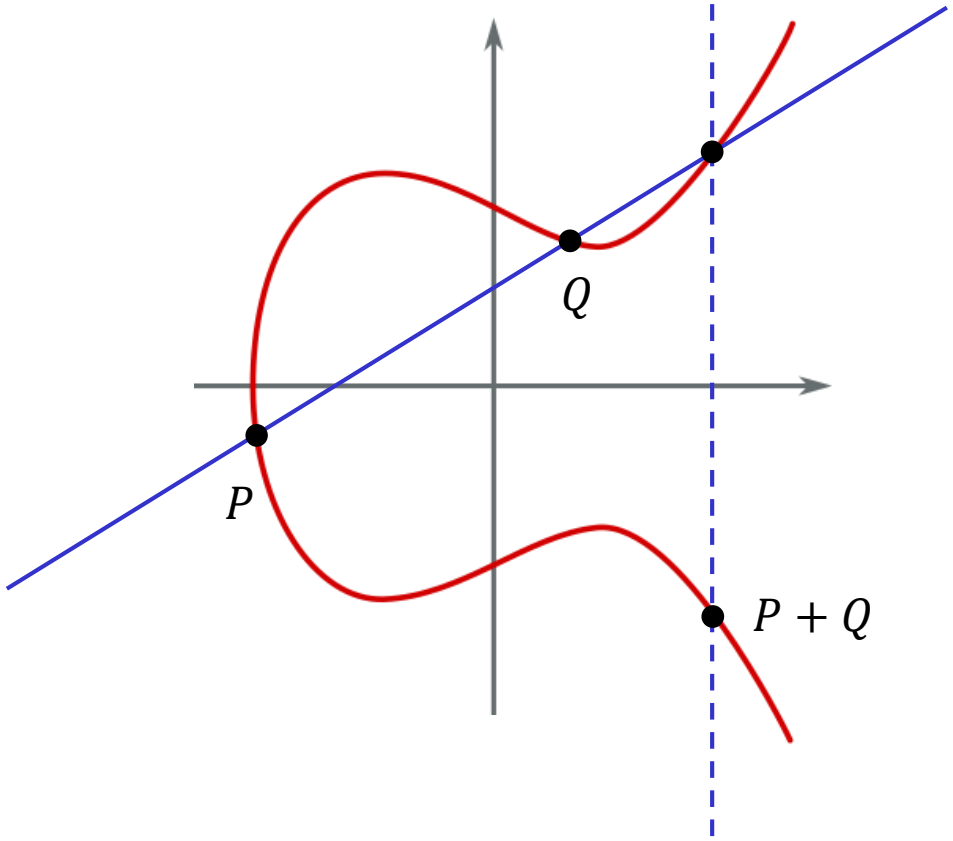
$$P + Q = (x_3, y_3)$$

$$x_3 = \frac{(x_1x_2 - 2a)x_1x_2 - 4b(x_1 + x_2) + a^2}{(x_1x_2 + a)(x_1 + x_2) + 2y_1y_2 + 2b}$$

$$y_3 = \frac{x_1x_2(x_1 + x_2) - x_3\big((x_1 + x_2)^2 - x_1x_2 + a\big) - y_1y_2 - b}{y_1 + y_2}$$

Notation: $(E(\boldsymbol{R}), +)$ – an elliptic curve group defined over the reals

Elliptic curves over Z_p



$$y^2 = x^3 + ax + b$$

$$a, b, x, y \in \boldsymbol{Z_p}$$

$$P + Q = (x_3, y_3)$$

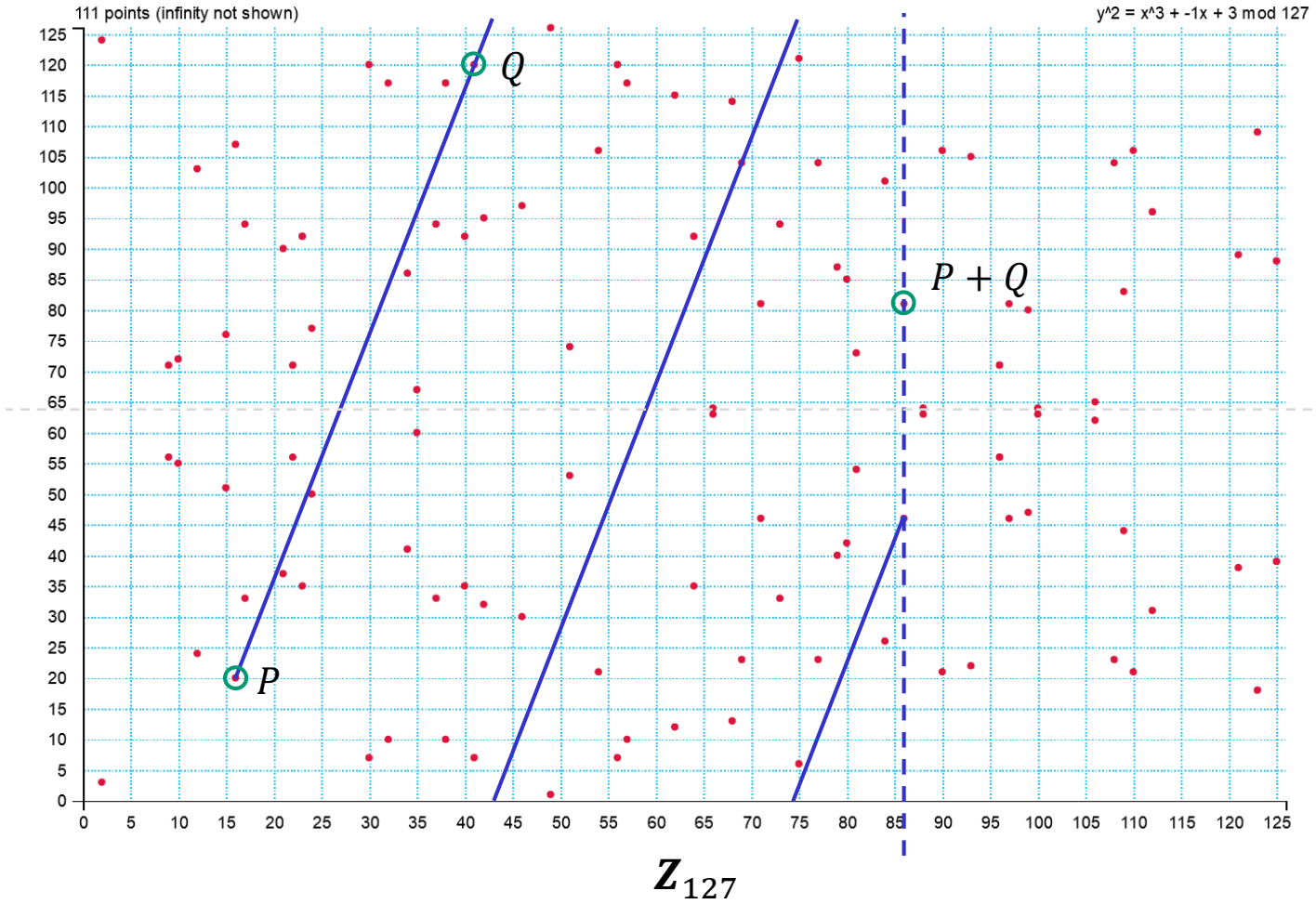
$$x_3 = \frac{(x_1x_2 - 2a)x_1x_2 - 4b(x_1 + x_2) + a^2}{(x_1x_2 + a)(x_1 + x_2) + 2y_1y_2 + 2b}$$

$$y_3 = \frac{x_1x_2(x_1 + x_2) - x_3\big((x_1 + x_2)^2 - x_1x_2 + a\big) - y_1y_2 - b}{y_1 + y_2}$$

Elliptic curves over Z_p

Theorem: the points on an elliptic curve, together with $\mathcal{O}$, is an abelian group under "geometric point addition"

Notation: $(E(\boldsymbol{F}_p), +)$ = an elliptic curve group defined over $\boldsymbol{F}_p$



$$y^2 = x^3 + ax + b$$

$$a, b, x, y \in \boldsymbol{Z_p}$$

$$P + Q = (x_3, y_3)$$

Still valid!

$$x_3 = \frac{(x_1x_2 - 2a)x_1x_2 - 4b(x_1 + x_2) + a^2}{(x_1x_2 + a)(x_1 + x_2) + 2y_1y_2 + 2b}$$

$$y_3 = \frac{x_1x_2(x_1 + x_2) - x_3\left((x_1 + x_2)^2 - x_1x_2 + a\right) - y_1y_2 - b}{y_1 + y_2}$$

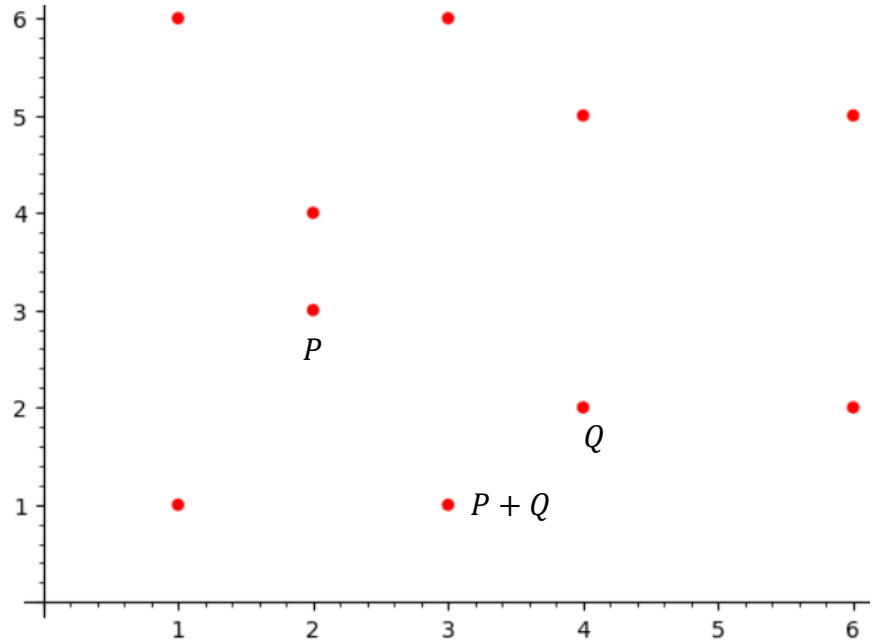
$(\boldsymbol{Z}_p, +)$ $(\boldsymbol{Z}_p^*, \cdot)$

$(\boldsymbol{F}_p, +, \cdot)$

Finite field

Elliptic curves – example

$$y^2 = x^3 + x - 1 \qquad (E(\boldsymbol{Z}_7), +)$$



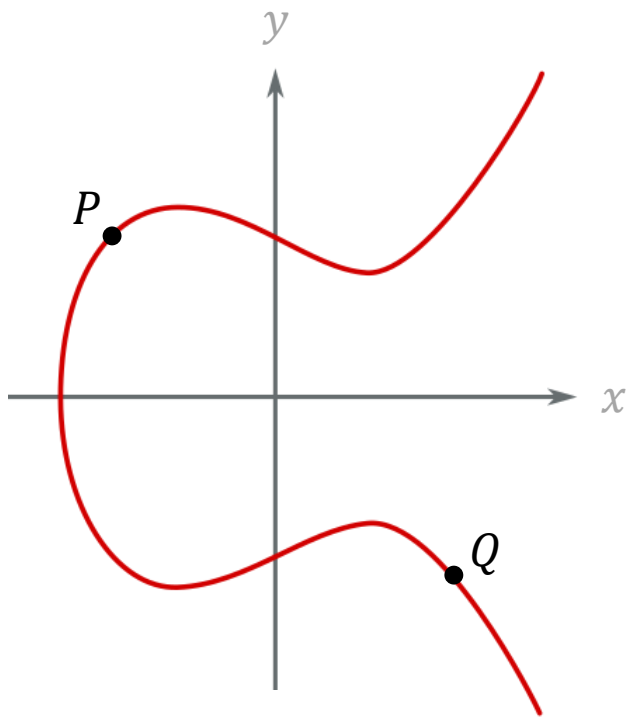
$$(E(\boldsymbol{Z_7}), +) = \{\mathcal{O}, (1,1), (1,6), (2,3), (2,4), (3,1), (3,6), (4,2), (4,5), (6,2), (6,5)\ \}$$

Elliptic curves summary

- $P = (x, y)$ point on elliptic curve
- x and y are elements in $\boldsymbol{Z}_p$
- Points P and Q can be added to get another point $P + Q$
- Special case: add P to itself n times

$$nP = P + P + \cdots + P$$

- DLOG problem: given P and nP for **secret** n, find n



$E(Z_p)$ – properties

- $(Z_p^*, \cdot)$ is *not* a generic group
  - Specialized attacks (GNFS) exploit algebraic structure ⟹ parameters must be bigger to compensate
  - $|Z_p^*| \geq 2^{2048}$ required for security today
  - Bigger parameters ⟹ slower systems

- Currently no attacks manage to exploit the algebraic structure of $(E(Z_p), +)$
  - Best-know attacks are *generic attacks*:
  - **Nechaev '94 & Shoup '97:** *Generic* algorithms for solving DLOG require time $\Omega\left(\sqrt{|G|}\right)$
  - Consequently: elliptic curve crypto can use *much* smaller parameters
  - $|E(Z_p)| = 2^{256}, 2^{384}, 2^{512}$ common in practice ⟹ much faster than Z_p^*-based crypto

Diffie-Hellman in $\left(Z_p^*, \cdot\right)$

$p =$ 171254583176141379301960419792575778264088323240375085733932929816426671397476217788024387752387285929683446135893799323484756135034769321631669738132186983438164632891441853629126025225404949830905314972329658295365245072698488256583114202993359222957097432675083225259667739503949192575768420387716327420441424710535098501236058838158571626669177751934961573726561955583057270098912760065140004093658772181713883199238963093777917625906143118496429613802248519404604217104493689272529748703958739363879096722748832953774810081504758785902705917983505634881680809238046118223875201980540029906239114543891047740921830 $\approx 2^{2048}$

$A = 2$ mod p

323170060713110073003389139264238282488179412411402391
128420097514007417066343542226196894173635693471179017
379097041917546058732091950288537589861856221532121754
125149017745202702357960782362488842461894775876411059
286460994117232454266225221932305409190376805242355191
256797158701170010580558776510388618472802579760549035
697325615261670813393617995413364765591603683178967290
731783845896806396719009772021941686472258710314113364
293195361934716365332097170774482279885885653692086452
966360772502689555059283627511211740969729980684105543
595848665832916421362182310789909994486524682624169720
359118525070453610905 59

$B = 2$ mod p

$\xleftarrow{\$} \{1 \dots p\}$

$\xleftarrow{\$} \{1 \dots p\}$

$Z \leftarrow 2$ mod p

$\times$

Diffie-Hellman in $(E(Z_p), +)$

$p = 115792089210356248762697446949407573530086143415290314195533631308867097853951 \approx 2^{256}$

$A = 89080287395774076886404465276262680764297347818898641719959470631357066012729 \cdot Q$

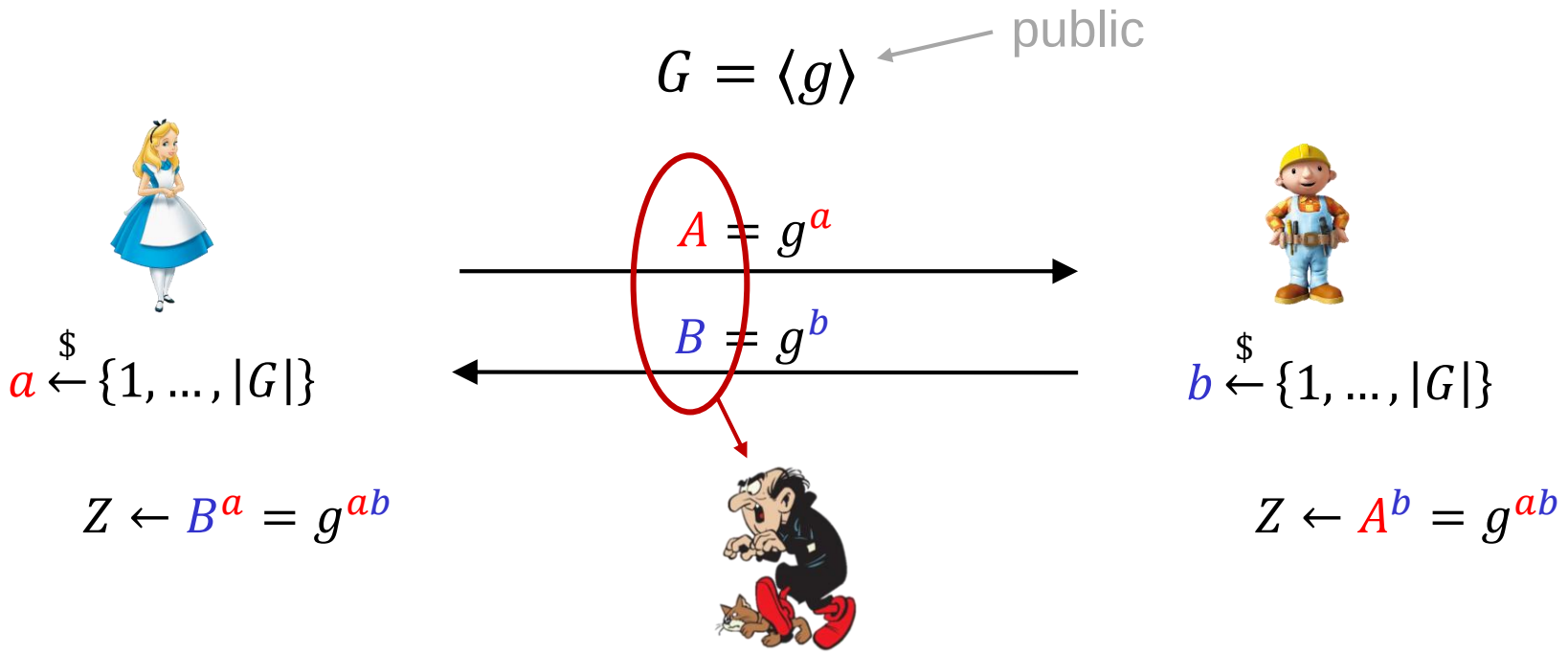
$89080287395774076886404465276262680764297347818898641719959470631357066012729 \xleftarrow{\$} \{1 \dots p\}$

$47777709944406917119610162988578419285974852857762532427086507801748071058272 \xleftarrow{\$} \{1 \dots p\}$

$B = 47777709944406917119610162988578419285974852857762532427086507801748071058272 \cdot Q$

$Z \leftarrow 89080287395774076886404465276262680764297347818898641719959470631357066012729 \times 47777709944406917119610162988578419285974852857762532427086507801748071058272 \cdot Q$

Diffie-Hellman – security



Security (given G, g, A, B):

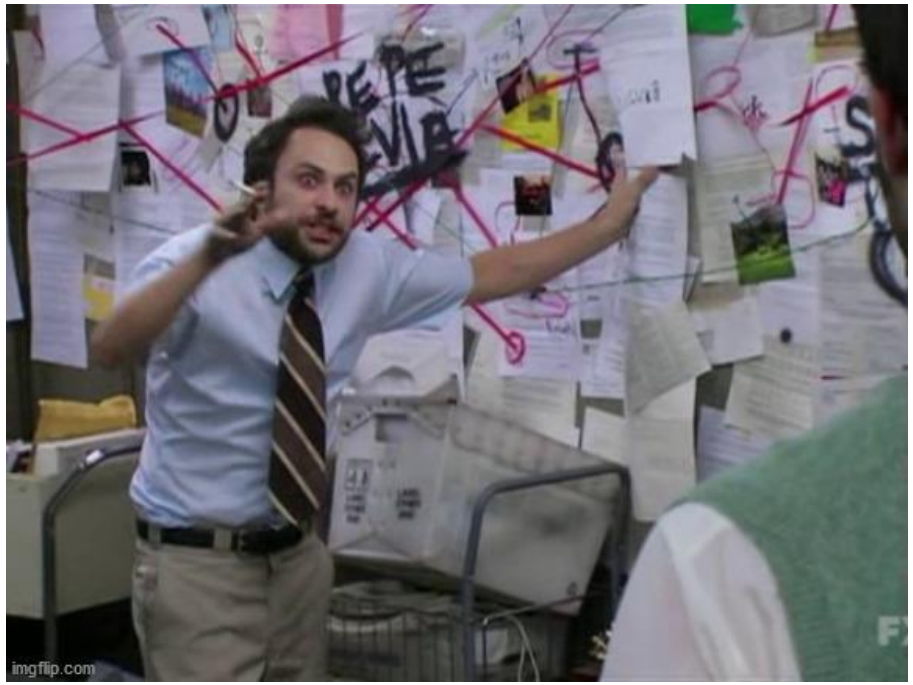
- Must be hard to compute $Z \leftarrow g^{ab}$ (DH assumption)
- Must be hard to find a (or b) (DLOG assumption)

- $G = (\boldsymbol{Z}_p, +)$ not secure
- $G = (\boldsymbol{Z}_p^*, \cdot)$ secure for $|\boldsymbol{Z}_p^*|$ **large enough** (conjectured)
- $G = (E(\boldsymbol{F}_p), +)$ secure for $|E(\boldsymbol{F}_p)|$ **large enough** (conjectured)

Cryptographic groups in practice

- $(\boldsymbol{Z}_p^*, \cdot)$ groups:
  - **TLS 1.3:** five specific groups allowed
    - size $\approx 2^{2048}, 2^{3072}, 2^{4096}, 2^{6144}, 2^{8192}$ (RFC 7919)
  - **IKEv2** (IPsec key exchange protocol): MODP groups
    - size $\approx 2^{768}, 2^{1024}, 2^{1536}, 2^{2048}, 2^{3072}, 2^{4096}, 2^{6144}, 2^{8192}$ (RFC 7296 and RFC 3526)
  - all p's are **safe primes** (i.e., of the form $p = 2q + 1$ where q is prime)

- $(E(\boldsymbol{Z}_p^*), +)$ groups
  - NIST curves: P-224, P-256, P-384, P-521
  - Curve25519 ($E: y^2 = x^3 + 486662x^2 + x$ and $p = 2^{255} - 19$) (Daniel J. Bernstein)
  - Curve448 ($E: y^2 + x^2 = 1 - 39081x^2y^2$ and $p = 2^{448} - 2^{224} - 1$) (Mike Hamburg)



After the break…

Juniper Networks

- Juniper Networks: big manufacturer of network equipment (routers, VPNs, firewalls, etc.)
  - Major customers: telcos, banks, US DoD

- **2015:**

IMPORTANT JUNIPER SECURITY ANNOUNCEMENT

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.

- Hackers obtained access to source code repository

- Only made one change:

```
--- Qx = 2c55e5e45edf713dc43475effe8813a60326a64d9ba3d2e39cb639b0f3b0ad10
+++ Qx = 9585320eeaf81044f20d55030a035b11bece81c785e6c933e4a8a131f6578107
```

PRNG standardization

NIST Special Publication 800-90

Recommendation for Random Number Generation Using Deterministic Random Bit Generators

Elaine Barker
John Kelsey

Computer Security Division
Inforamtion Technology Laboratory

COMPUTER SECURITY

June 2006

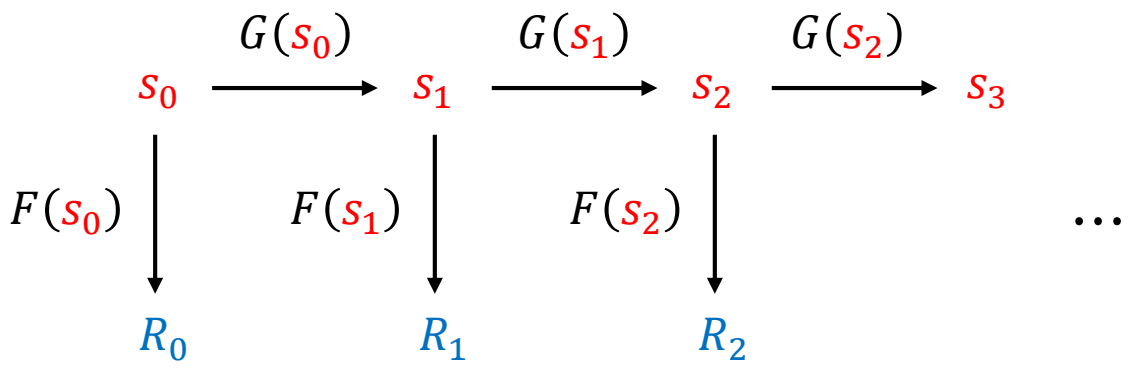
U.S. Department of Commerce
Carlos M. Gutierrez, Secretary

Technology Administration
Robert Cresanti, Under Secretary of Commerce for Technology

National Institute of Standards and Technology
William Jeffrey, Director

NIST National Institute of Standards and Technology • Technology Administration • U.S. Department of Commerce

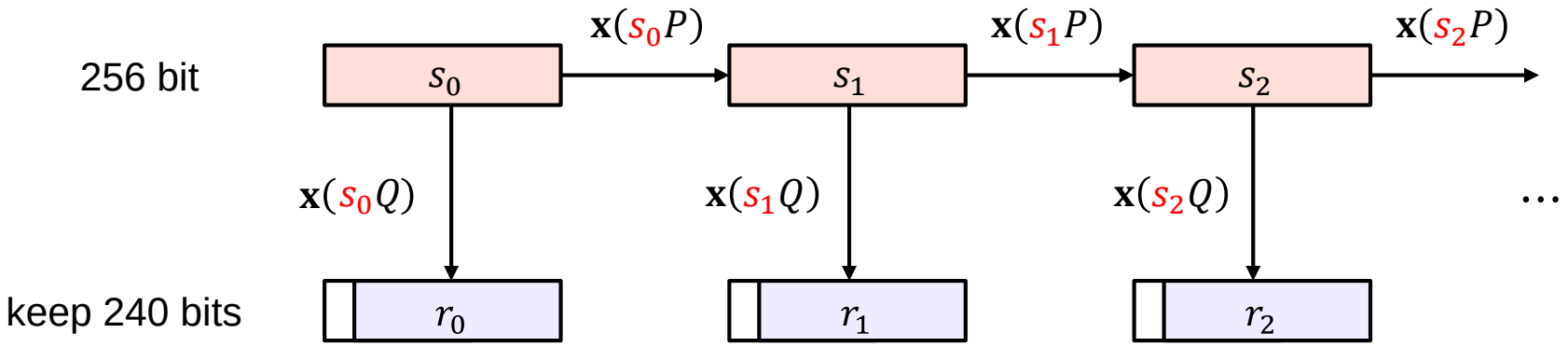
PRNG



Dual EC DRBG

P, Q: **public** points on an elliptic curve

$\mathbf{x}(\cdot)$: x-coordinate (256 bits)

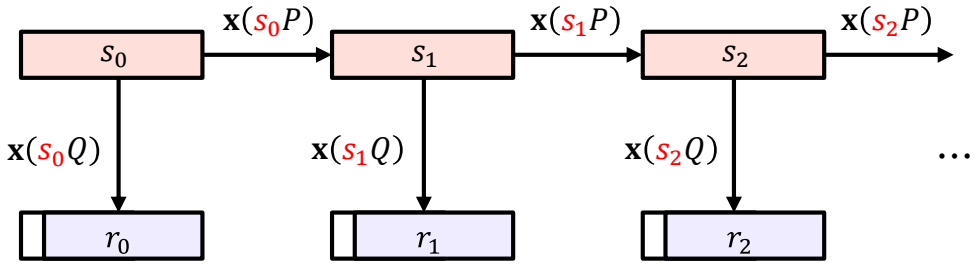


Dual EC DBRG – something's fishy

- Dual EC is *slow*
  - Orders of magnitude slower than HMAC/AES-CTR based alternatives

- **2006** – Kristian Gjøsteen: Dual EC is *not* a good PRNG
  - Can distinguish output from random with $\mathbf{Adv}^{\text{prg}}_{\text{DualEC}}(KG) \approx 0.0011$
  - Slightly improved by Schoenmakers and Sidorenko

- **2007** – Shumow and Ferguson: Dual EC can be *backdoored*
  - What if $P = kQ$ for a *secret* k only you know?

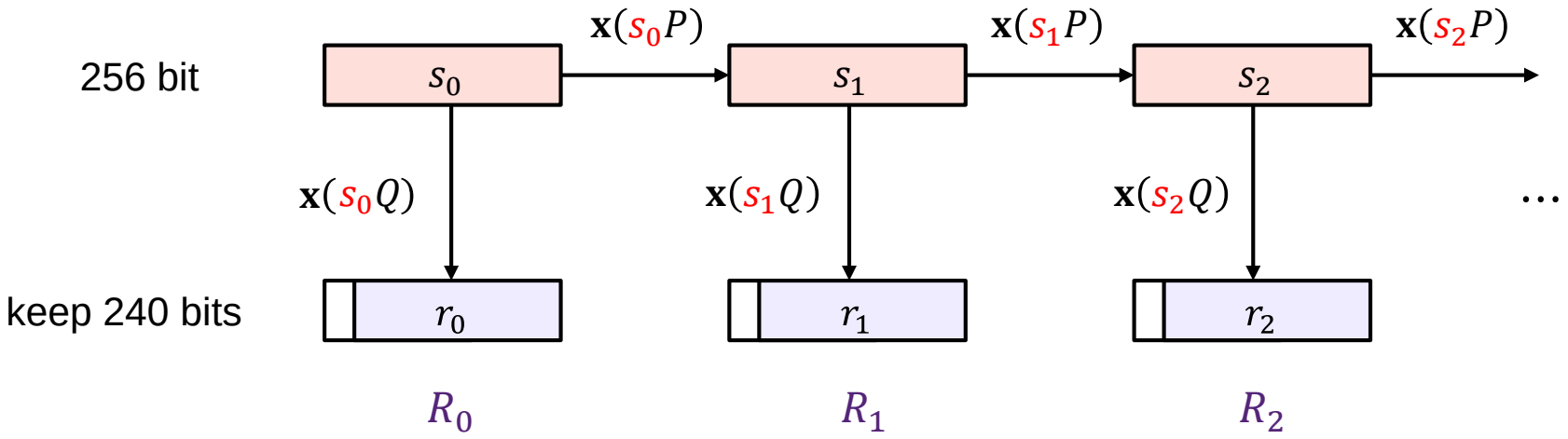


Dual EC DRBG

P, Q: **public** points on an elliptic curve

$$P = kQ$$

$\mathbf{x}(\cdot)$: x-coordinate (256 bits)

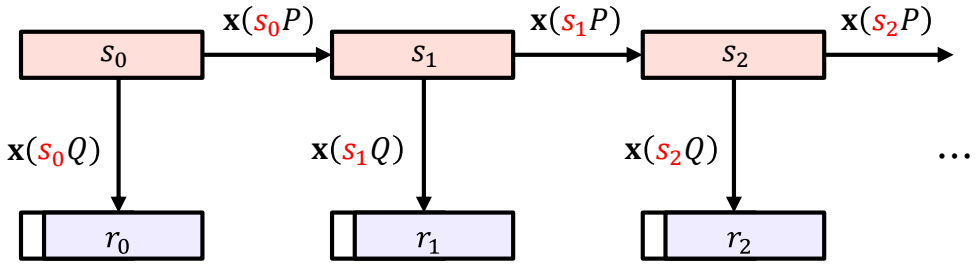


$$kR_0 = k \cdot (s_0Q) = s_0 \cdot (kQ) = s_0P \xrightarrow{\mathbf{x}(\cdot)} s_1$$

Dual EC DBRG – something's fishy

- Dual EC is *slow*
  - Orders of magnitude slower than HMAC/AES-CTR based alternatives

- **2006** – Kristian Gjøsteen: Dual EC is *not* a good PRNG
  - Can distinguish output from random with $\mathbf{Adv}^{\text{prg}}_{\text{DualEC}}(KG) \approx 0.0011$
  - Slightly improved by Schoenmakers and Sidorenko



- **2007** – Shumow and Ferguson: Dual EC can be *backdoored*
  - What if $P = kQ$ for a *secret* k only you know?
  - If you know full s_0Q, compute $d(s_0Q) = s_0(dQ) = s_0P \rightarrow s_1$
  - Because of truncation need to guess top 16 bits ($\approx 2^{16}$ additional work)

- **2007** – NIST adds appendix to standard on how to create P and Q yourself
  - Continues to recommend existing P and Q

- Most cryptographers: who cares? No one is going to use Dual EC anyway …

- **2013** – Edward Snowden leak: a project called Bullrun exists within the NSA
  - Purpose: "*Insert vulnerabilities into commercial encryption systems, IT systems, networks, and endpoint communications devices used by targets.*"
  - Turns out Juniper Networks made Dual EC their PRNG in ScreenOS from 2008

Where does Q (and P) come from?

From: John Kelsey [mailto:john.kelsey@nist.gov]
Sent: Wednesday, October 27, 2004 11:17 AM
To: Don Johnson
Subject: Minding our Ps and Qs in Dual_EC

Do you know where Q comes from in Dual_EC_DRBG?

Thanks,
-Joh

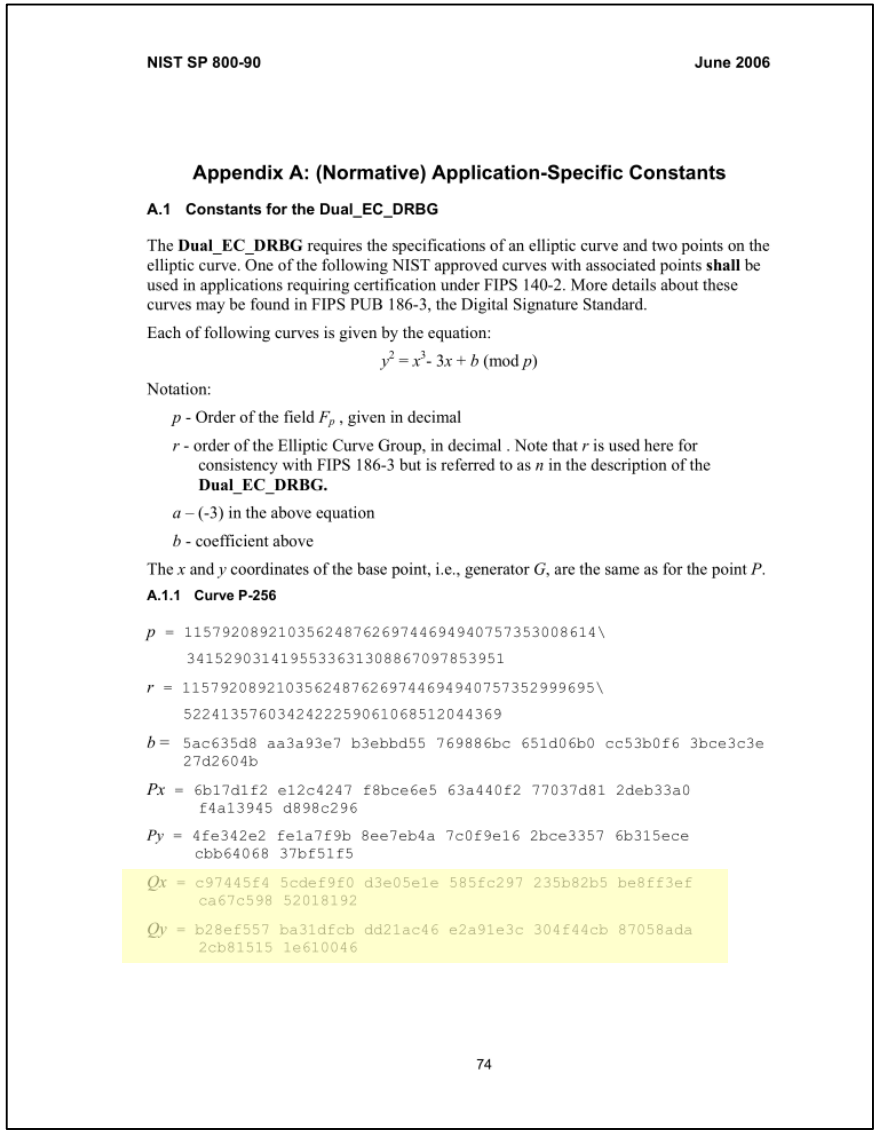
Subject: RE: Minding our Ps and Qs in Dual_EC
From: "Don Johnson" <DJohnson@cygnacom.com>
Date: Wed, October 27, 2004 11:42 am
To: "John Kelsey" <john.kelsey@nist.gov>
John,

P=G.

Q is (in essence) the public key for some random private key.

It could also be generated like a(nother) canonical G, but NSA kyboshed this idea, and I was not allowed to publicly discuss it, just in case you may think of going there.

Don B. Johnson



NIST SP 800-90 June 2006

Appendix A: (Normative) Application-Specific Constants

A.1 Constants for the Dual_EC_DRBG

The **Dual_EC_DRBG** requires the specifications of an elliptic curve and two points on the elliptic curve. One of the following NIST approved curves with associated points **shall** be used in applications requiring certification under FIPS 140-2. More details about these curves may be found in FIPS PUB 186-3, the Digital Signature Standard.

Each of following curves is given by the equation:

$$y^2 = x^3 - 3x + b \pmod{p}$$

Notation:

p - Order of the field F_p, given in decimal

r - order of the Elliptic Curve Group, in decimal . Note that r is used here for consistency with FIPS 186-3 but is referred to as n in the description of the **Dual_EC_DRBG.**

a – (-3) in the above equation

b - coefficient above

The x and y coordinates of the base point, i.e., generator G, are the same as for the point P.

A.1.1 Curve P-256

p = 115792089210356248762697446949407573530086143415290314195533631308867097853951

r = 115792089210356248762697446949407573529996955224135760342422259061068512044369

b = 5ac635d8 aa3a93e7 b3ebbd55 769886bc 651d06b0 cc53b0f6 3bce3c3e 27d2604b

Px = 6b17d1f2 e12c4247 f8bce6e5 63a440f2 77037d81 2deb33a0 f4a13945 d898c296

Py = 4fe342e2 fe1a7f9b 8ee7eb4a 7c0f9e16 2bce3357 6b315ece cbb64068 37bf51f5

Qx = c97445f4 5cdef9f0 d3e05e1e 585fc297 235b82b5 be8ff3ef ca67c598 52018192

Qy = b28ef557 ba31dfcb dd21ac46 e2a91e3c 304f44cb 87058ada 2cb81515 1e610046

74

Juniper PRNG

ScreenOS does make use of the Dual_EC_DRBG standard, but is designed to not use Dual_EC_DBRG 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.

Juniper Knowledge Base Article KB28205

Juniper PRNG

ScreenOS v.6.1

```
1.   char seed[8]                          // dual_ec output
2.   char block[8];                  // X9.31 temporary output
3.
4.   unsigned int calls_since_reseed;
5.
6.   void prng_reseed(void) {
7.
8.       // same as v6.2
9.
10.   }
11.
12.  void prng_generate(char *output) {
13.     unsigned int index = 0;
14.     calls_since_reseed++;
15.
16.     if (calls_since_reseed > 10 000)
17.       prng_reseed();
18.
19.     for (; index < 20; index += 8) {
20.       ...
21.       x9_31_generate_block(seed, block);
22.       ...
23.       memcpy(&output[index], block, 8);
24.     }
25.  }
```

caller-supplied buffer

private variable

only 20 bytes

ScreenOS v.6.2

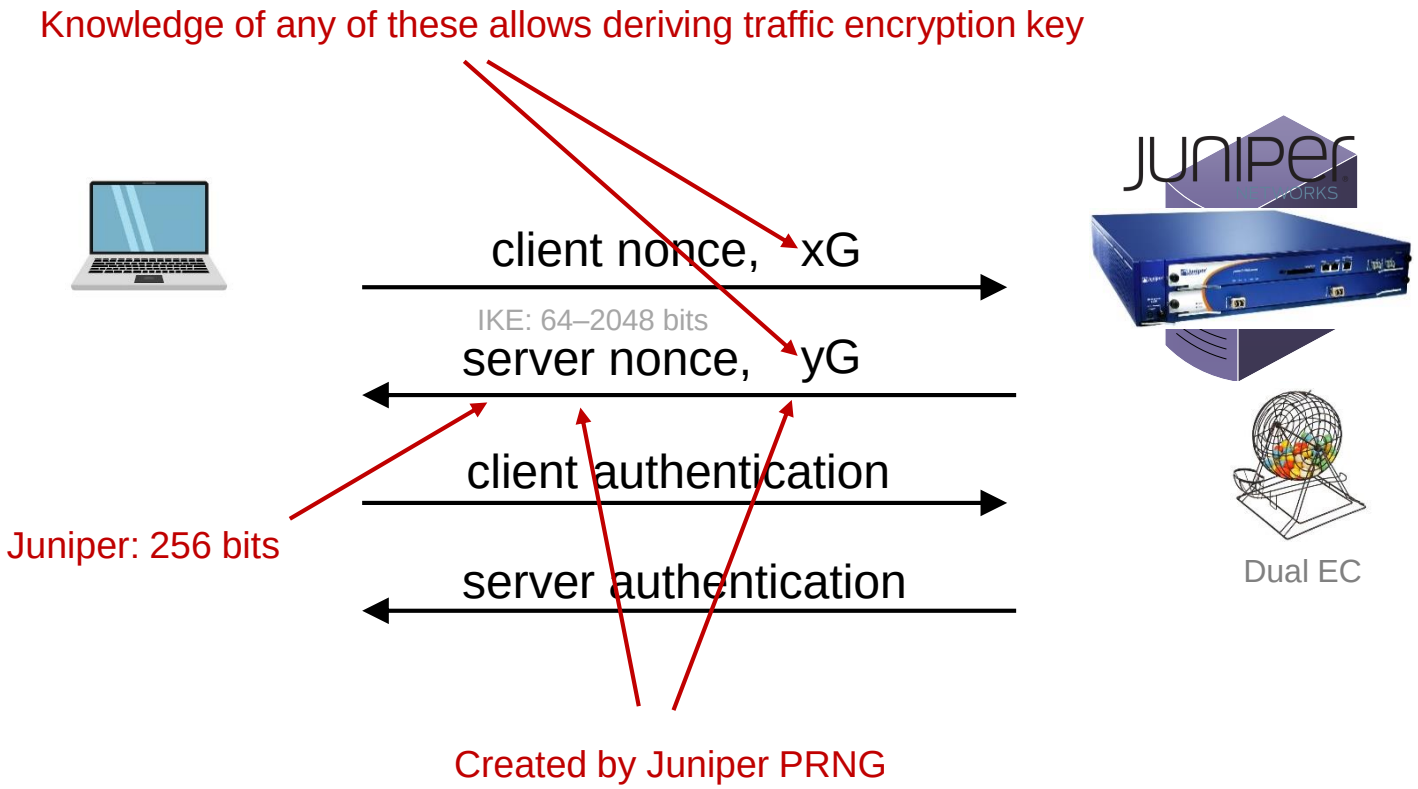
```
1.   char seed[32];                          // dual_ec output
2.   char block[8];                    // X9.31 temporary output
3.   char output[32];              // prng_generate final output
4.   unsigned int index;
5.
6.   void prng_reseed(void) {
7.      dual_ec_generate(output, 32);
8.      memcpy(seed, output, 32);
9.      index = 32;
10.  }
11.
12.  void prng_generate(void) {
13.     index = 0;
14.
15.     prng_reseed();
16.
17.     for (; index < 32; index += 8) {
18.       ...
19.       x9_31_generate_block(seed, block);
20.       ...
21.       memcpy(&output[index], block, 8);
22.     }
23.  }
```

global variable

global shared buffer

Source: *Where did I leave my keys? Lessons from the Juniper Dual EC incident* https://dl.acm.org/doi/10.1145/3266291

IKEv2 / IPsec


Knowledge of any of these allows deriving traffic encryption key
client nonce, xG
IKE: 64–2048 bits
server nonce, yG
client authentication
server authentication
Juniper: 256 bits
Dual EC
Created by Juniper PRNG

Juniper Networks backdoor

- Juniper Networks: big manufacturer of network equipment (routers, VPNs, firewalls, etc.)
  - Major customers: telcos, banks, US DoD

- **2015:**

IMPORTANT JUNIPER SECURITY ANNOUNCEMENT

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.

- Hackers had obtained access to source code repository

- Only one change:

```
--- Qx = 2c55e5e45edf713dc43475effe8813a60326a64d9ba3d2e39cb639b0f3b0ad10
+++ Qx = 9585320eeaf81044f20d55030a035b11bece81c785e6c933e4a8a131f6578107
```

Juniper created Q (yay! No NSA)

Who dis?

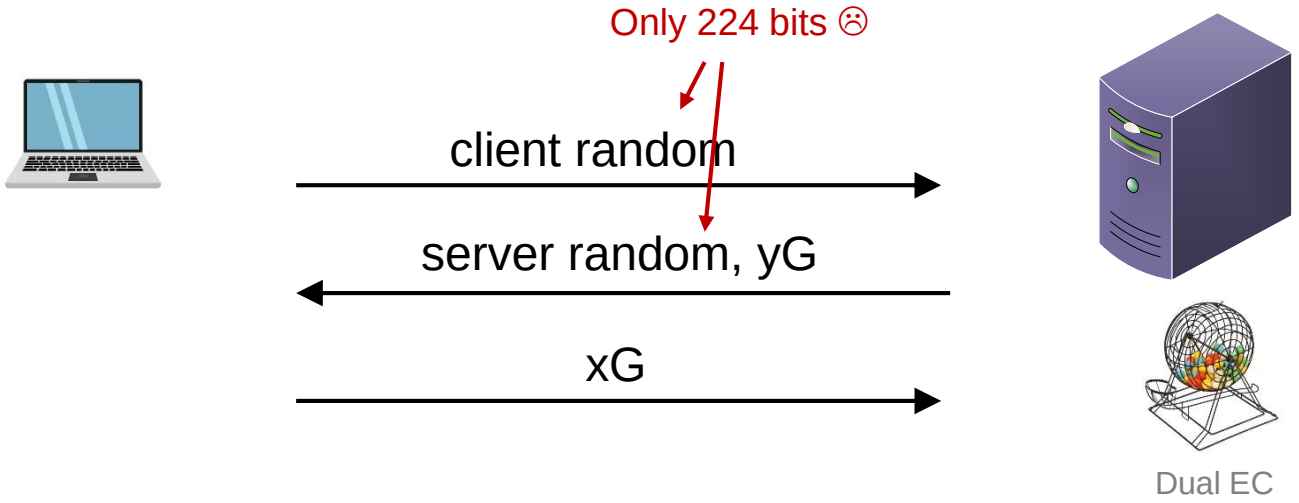
Juniper Networks backdoor

- **2008** – Juniper starts using Dual EC in ScreenOS

- **2012** – Someone hacks into Juniper's code repositories
  - Changes Q point in Dual EC

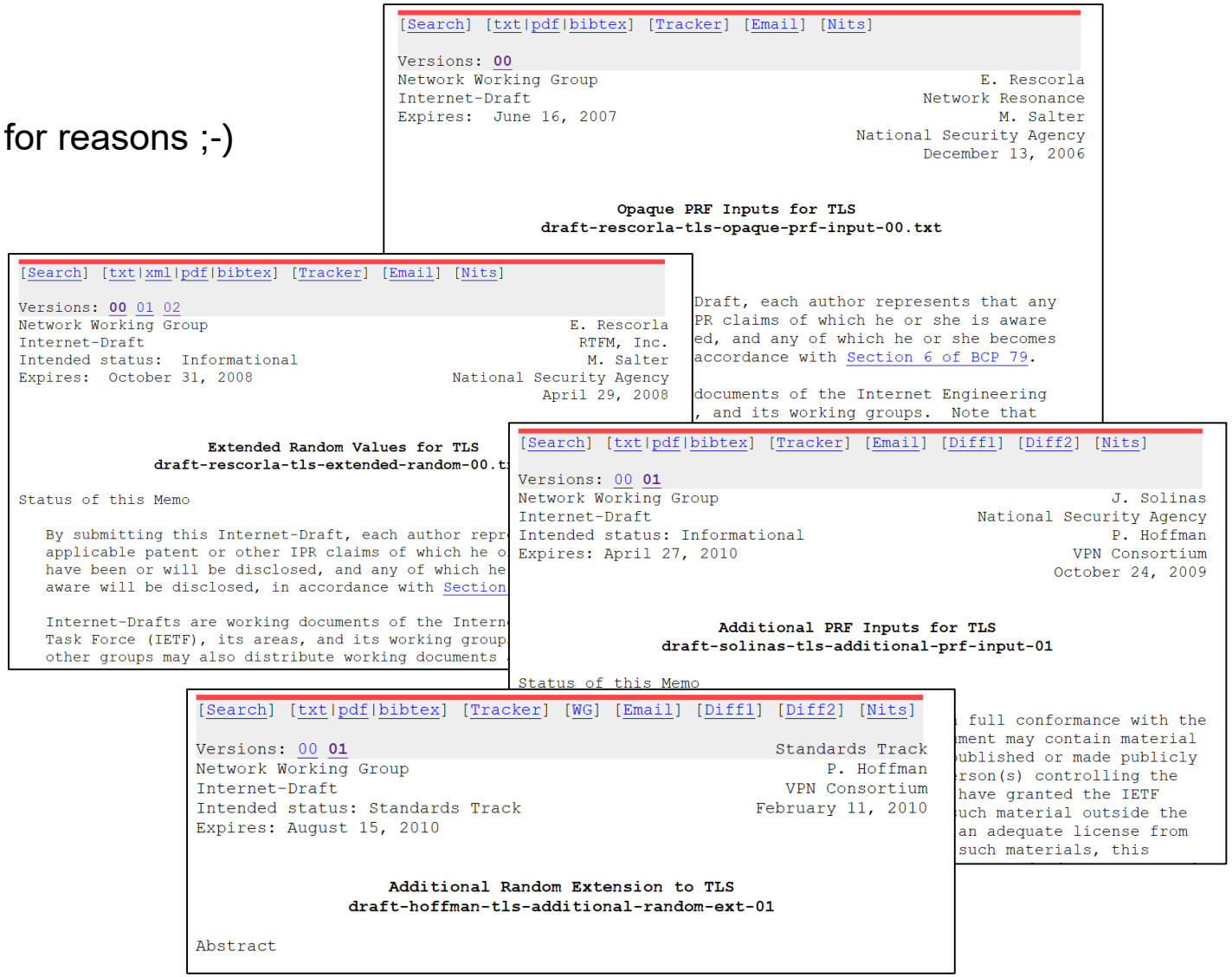
- **2015** – Juniper discovers intrusion
  - Changes Q back to its original value

TLS



Extended Random

- NSA: please make the TLS nonces bigger…for reasons ;-)
- Implementing Extended Random makes exploiting Dual EC 10,000 times easier
- No real cryptographic justification exists for making them longer



[Search] [txt|pdf|bibtex] [Tracker] [Email] [Nits]

Versions: 00
Network Working Group — E. Rescorla
Internet-Draft — Network Resonance
Expires: June 16, 2007 — M. Salter
National Security Agency
December 13, 2006

Opaque PRF Inputs for TLS
draft-rescorla-tls-opaque-prf-input-00.txt

[Search] [txt|xml|pdf|bibtex] [Tracker] [Email] [Nits]

Versions: 00 01 02
Network Working Group — E. Rescorla
Internet-Draft — RTFM, Inc.
Intended status: Informational — M. Salter
Expires: October 31, 2008 — National Security Agency
April 29, 2008

Extended Random Values for TLS
draft-rescorla-tls-extended-random-00.t

Status of this Memo

[Search] [txt|pdf|bibtex] [Tracker] [Email] [Diff1] [Diff2] [Nits]

Versions: 00 01
Network Working Group — J. Solinas
Internet-Draft — National Security Agency
Intended status: Informational — P. Hoffman
Expires: April 27, 2010 — VPN Consortium
October 24, 2009

Additional PRF Inputs for TLS
draft-solinas-tls-additional-prf-input-01

Status of this Memo

[Search] [txt|pdf|bibtex] [Tracker] [WG] [Email] [Diff1] [Diff2] [Nits]

Versions: 00 01
Standards Track
Network Working Group — P. Hoffman
Internet-Draft — VPN Consortium
Intended status: Standards Track — February 11, 2010
Expires: August 15, 2010

Additional Random Extension to TLS
draft-hoffman-tls-additional-random-ext-01

Abstract

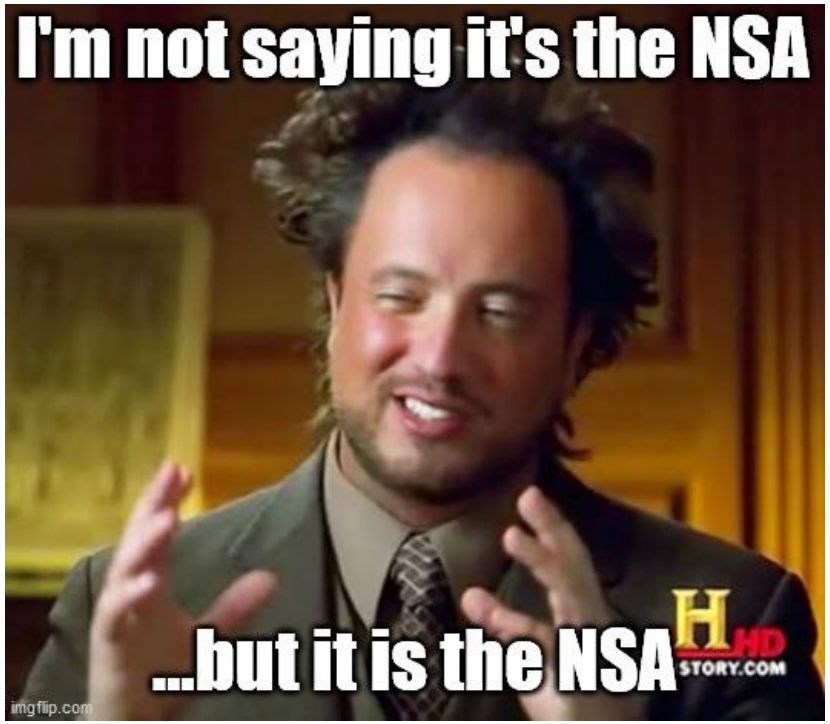
RSA Security – BSAFE

- Turns out Juniper weren't the only one using Dual EC

- RSA Security
  - Big computer and network security company
  - Creator of BSAFE cryptographic library

- **2004** – accepted $10 million from the NSA in order to make Dual EC the *default* in BSAFE

- **2014** – adapted the TLS Extended Random extension


I'm not saying it's the NSA
...but it is the NSA
HISTORY.COM
imgflip.com

Next week

- More Diffie-Hellman
- Implementation issues