

# Elliptic Curve Cryptography for those who are afraid of maths

Martijn Grooten, Virus Bulletin

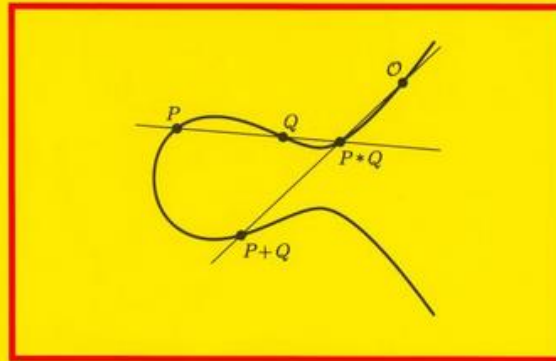
 @martijn\_grooten

BSides London, 3 June 2015

Undergraduate Texts in Mathematics

**Joseph H. Silverman**  
**John Tate**

# Rational Points on Elliptic Curves



 Springer

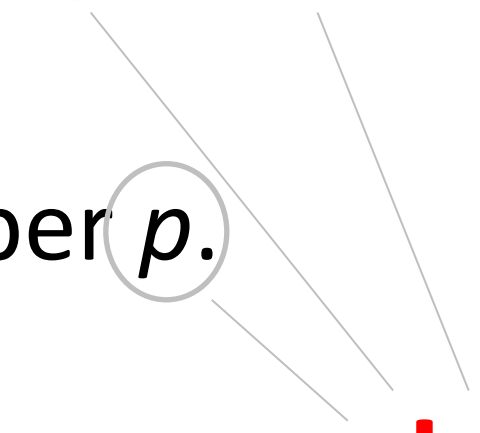
# Disclaimer:

This talk will be useless.

I am not a cryptographer.

Some things are wrong.

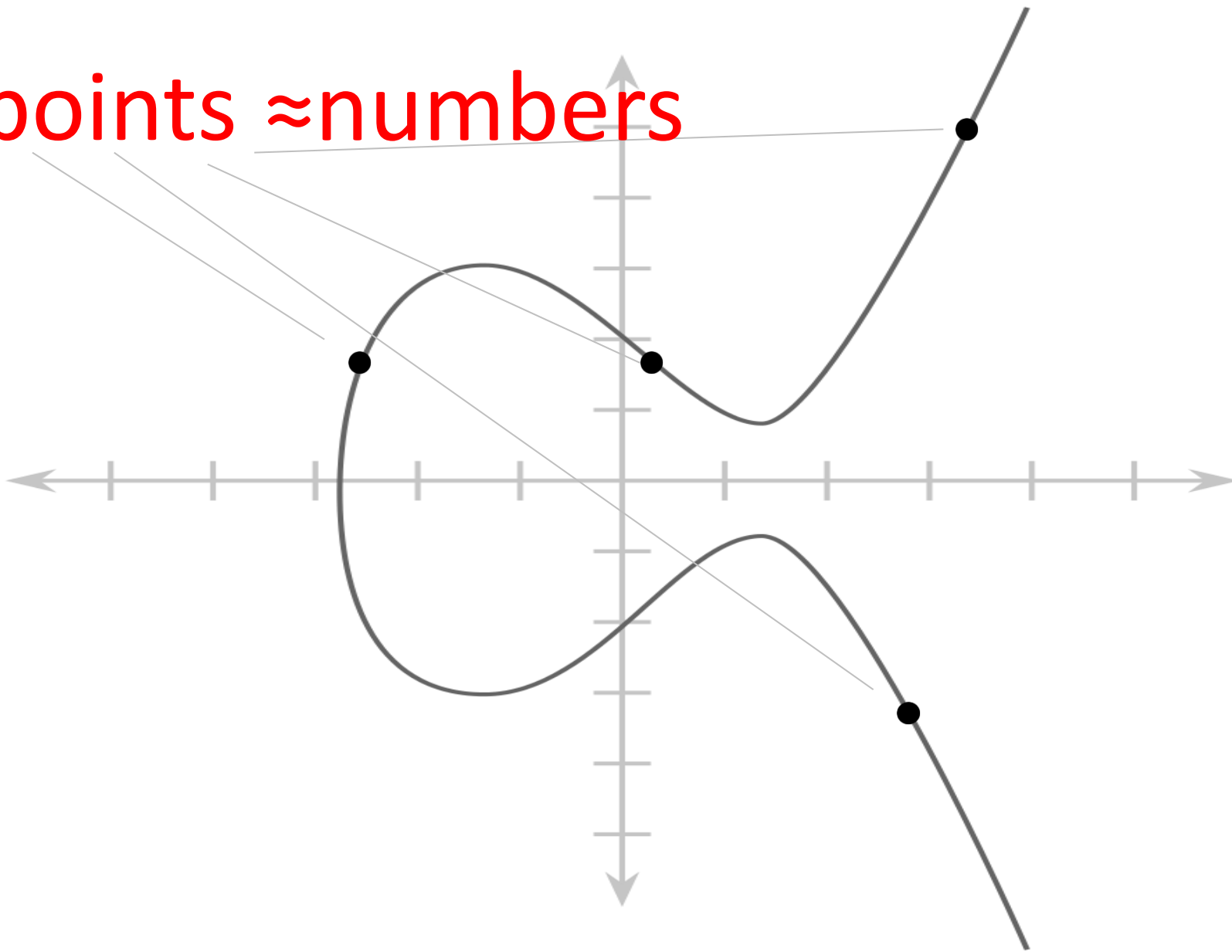
# Elliptic curves

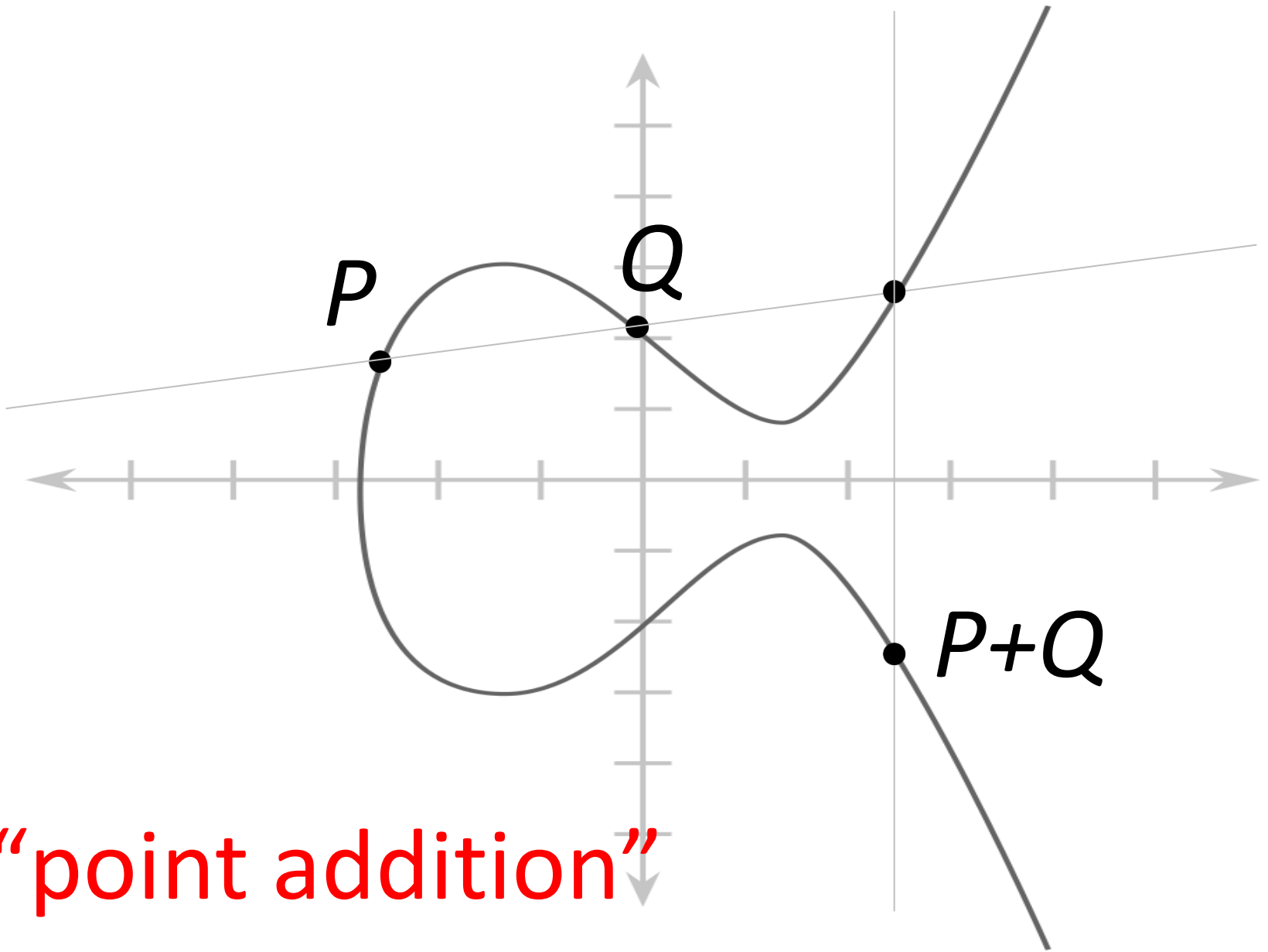
$$y^2 = x^3 + a \cdot x + b$$


...and a prime number  $p$ .

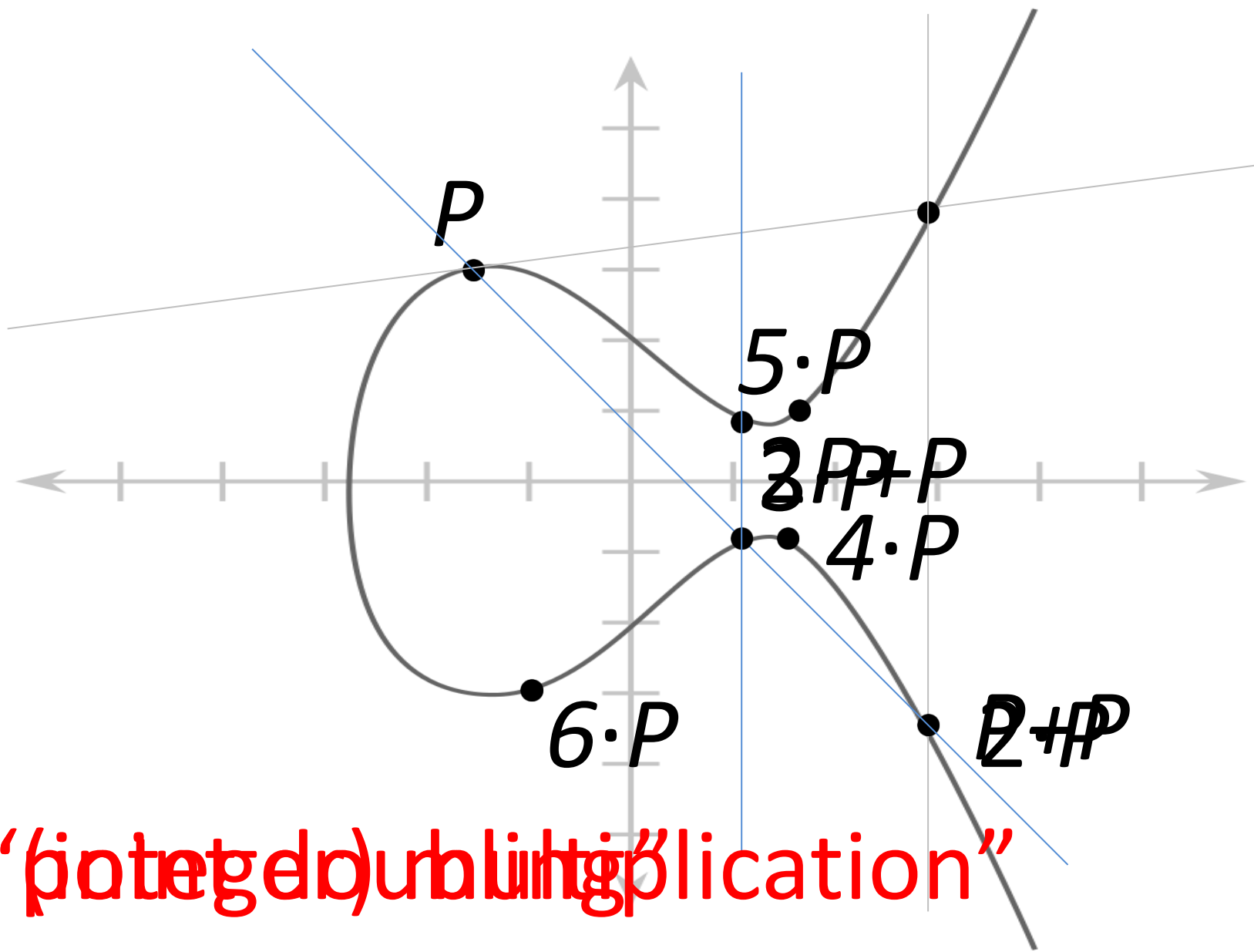
**choice!**

points  $\approx$  numbers





“point addition”



“(integer) multiplication”

So:

We can “add” points to each other.

We can “multiply” points by an integer.

Nice:  $P + Q = Q + P$

$$3 \cdot P + P = 2 \cdot P + 2 \cdot P = 4 \cdot P$$

$$5 \cdot (7 \cdot P) = 7 \cdot (5 \cdot P)$$

etc.

The points on a curve form an *Abelian Group* (very exciting!).



# Multiplication is very fast

To go from a point  $P$  to  $100 \cdot P$ :

$$P \rightarrow 2 \cdot P$$

$$12 \cdot P \rightarrow 24 \cdot P$$

$$2 \cdot P \rightarrow 3 \cdot P$$

$$24 \cdot P \rightarrow 25 \cdot P$$

$$3 \cdot P \rightarrow 6 \cdot P$$

$$25 \cdot P \rightarrow 50 \cdot P$$

$$6 \cdot P \rightarrow 12 \cdot P$$

$$50 \cdot P \rightarrow 100 \cdot P$$

Only eight steps!

“Division” is very slow

Given points  $P$  and  $Q$ , where  $Q=n\cdot P$ , the best way to find the number  $n$  is to try  $P, 2\cdot P, 3\cdot P$ , etc. That is very slow.

*The Discrete Logarithm Problem for elliptic curves.*

# ECDH (Elliptic Curve Diffie Hellman)

The challenge: Alice and Bob want to agree on a secret key over a public channel.

For example: Alice is a web server, Bob a browser and they want to exchange a key to encrypt a TLS session.

# ECDH (Elliptic Curve Diffie Hellman)

Alice and Bob have agreed on an elliptic curve and a “base point”  $P$  on the curve.

Alice chooses secret large random number  $a$ .

Bob chooses secret large random number  $b$ .

# ECDH (Elliptic Curve Diffie Hellman)

Alice computes  $a \cdot P$  ( $a$  times the point  $P$ ) and shares the answer with Bob.

Bob computes  $b \cdot P$  and shares this too.

Alice computes  $a \cdot (b \cdot P)$  ( $a$  times the point Bob gave her).

Bob computes  $b \cdot (a \cdot P)$ .

Secret key:  $a \cdot (b \cdot P) = b \cdot (a \cdot P)$ .

# Wireshark (client to server)

Session ID Length: 0

Cipher Suites Length: 22

▽ Cipher Suites (11 suites)

Cipher Suite: TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_GCM\_SHA256 (0xc02b)

Cipher Suite: TLS\_ECDHE\_RSA\_WITH\_AES\_128\_GCM\_SHA256 (0xc02f)

Cipher Suite: TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_CBC\_SHA (0xc00a)

Cipher Suite: TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CBC\_SHA (0xc009)

Cipher Suite: TLS\_ECDHE\_RSA\_WITH\_AES\_128\_CBC\_SHA (0xc013)

Cipher Suite: TLS\_ECDHE\_RSA\_WITH\_AES\_256\_CBC\_SHA (0xc014)

Cipher Suite: TLS\_DHE\_RSA\_WITH\_AES\_128\_CBC\_SHA (0x0033)

Cipher Suite: TLS\_DHE\_RSA\_WITH\_AES\_256\_CBC\_SHA (0x0039)

Cipher Suite: TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA (0x002f)

Cipher Suite: TLS\_RSA\_WITH\_AES\_256\_CBC\_SHA (0x0035)

Cipher Suite: TLS\_RSA\_WITH\_3DES\_EDE\_CBC\_SHA (0x000a)

Compression Methods Length: 1

▷ Compression Methods (1 method)

“11 cipher suites you didn’t know I supported”

# Wireshark (client to server)

```

  ▾ extension: elliptic_curves
    Type: elliptic_curves (0x000a)
    Length: 8
    Elliptic Curves Length: 6
    ▾ Elliptic curves (3 curves)
      Elliptic curve: secp256r1 (0x0017)
      Elliptic curve: secp384r1 (0x0018)
      Elliptic curve: secp521r1 (0x0019)
    ▸ Extension: ec_point_formats
    ▸ Extension: SessionTicket TLS
    ▸ Extension: next_protocol_negotiation
    ▸ Extension: ...

```

---

```

0) 00 18 4d 67 16 04 0c ee e6 d2 a1 1b 08 00 45 00

```

“These are my three favourite curves.”

# Wireshark (server to client)

```
version: TLS 1.2 (0x0303)
  Random
  Session ID Length: 0
  Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)
  Compression Method: null (0)
  Extensions Length: 21
  Extension: server_name
    Type: server_name (0x0000)
    Length: 0
-----
060 87 49 52 69 f9 26 6c 1c eb ef 06 5c 74 00 c0 2f .IRi.&l. ... \t.
```

“OK, let’s go for  
TLS\_ECDHE\_RSA\_WITH\_AES\_128\_GCM\_SHA256.”



# Wireshark (server to client)

```
0ca0 3c 3f 4c 10 8d ef bb 75 27 d2 ae 83 a7 a8 ce 5b <?L....u '.....[
0cb0 a7 16 03 03 01 4d 0c 00 01 49 03 00 17 41 04 28 .....M.. .I...A.(
0cc0 8f 17 33 62 43 e7 da 2f 78 a7 7d 85 b7 94 72 10 ..3bC../ x.}...r.
0cd0 ef 47 2d ed 64 a6 67 4f be b5 be f9 95 09 f6 31 G. d a0 1
0ce0 26 f5 30 59 fe 28 43 07 2 ae 83 a7 a8 ce 5
0cf0 1f 38 ab a8 a8 7b d3 ae 9 03 00 17 41 04 2
0d00 01 01 00 8c 5c 97 f4 ee 7 7d 85 b7 94 72 1
0d10 cb 45 f8 fa 65 f8 a2 f9 5 be f9 95 09 f6 3
0d20 e0 1b 68 dc 94 d4 c9 ef 1 41 76 9a 8c 22 a
0d30 1e d8 0b f3 66 5c 1d 84 50 99 ce 08 5a cc ce e4 .....1b.. P...Z...
0d40 98 f9 8f 71 5e 6d bd 93 e1 f5 f9 97 a0 4b 8f 0e .C..?.." .....K..
0d50 04 2d a5 7b 56 cd e4 cf a1 86 35 18 3a 6d af 41 0..4R..G ..5.:m.A
0d60 b5 b7 19 87 0d ed 95 0d a5 f0 4d 49 ec b0 2f 80 ..=..... ..MI../.
0d70 7f d8 45 69 5e 52 bc 3c 85 b2 7f fd 90 84 8a 0f !...(.n. ....
0d80 cd 18 d8 6c 97 57 f6 2c d2 41 5d b8 33 16 5a 7a .w...E. .A].3.Zz
0d90 ef af fc da 31 38 aa fc c7 f3 e8 08 0c 20 c2 c7 ..b(s... ..
0da0 f7 43 f3 dd 3f a9 f8 22 e0 00 00 00 .....
0db0 30 c9 a9 34 52 a9 1c 47 a1 86 35 18 3a 6d af 41
0dc0 9a f3 3d a5 a5 ab fc 01 a5 f0 4d 49 ec b0 2f 80
0dd0 21 db f3 1c 28 89 6e f9 85 b2 7f fd 90 84 8a 0f
0de0 19 57 d0 94 ac e4 45 cd d2 41 5d b8 33 16 5a 7a
0df0 c0 9e 62 28 73 1c 09 e0 c7 f3 e8 08 0c 20 c2 c7
0e00 83 c6 fb 16 03 03 00 04 0e 00 00 00
```

“And curve NIST P-256. And this is my point.”

# Wireshark (client to server)

EC Diffie-Hellman Client Params

Pubkey Length: 65

pubkey: 04b6d623a732967c66508cc3d7760bd160269f7e2a34cc8a...

TLSv1.2 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec

Content Type: Change Cipher Spec (20)

Version: TLS 1.2 (0x0303)

Length: 1

Change Cipher Spec Message

00	00 18 4d 67 16 04 0c ee	e6 d2 a1 1b 08 00 45 00	..Mg.... ..E.
10	00 b2 31 08 40 00 40 06	c3 05 c0 a8 00 07 5e 17	..1.@.@. ....^.
20	27 72 a3 79 01 bb f8 ce	68 32 af cf a5 10 80 18	'r.y.... h2.....
30	05 ad 2d d4 00 00 01 01	08 0a 14 a8 56 91 e4 b5	..-..... ....V...
40	f6 df 16 03 03 00 46 10	00 00 42 41 04 b6 d6 23	.....F. ..BA...#
50	a7 32 96 7c 66 50 8c c3	d7 76 0b d1 60 26 9f 7e	.2. fP.. .v..`&~
60	2a 34 cc 8a 17 9c 55 2c	94 37 94 64 d0 b2 0b dc	*4...U, .7.d...
70	a0 8d ce 40 6b d9 a0 af	42 ae 15 b8 86 0a 1d a8	...@k... B.....
80	a2 d7 f7 28 3c 98 8b d2	4d 55 64 51 30 14 03 03	...(<... MUdQq...
90	00 01 01 16 03 03 00 28	00 00 00 00 00 00 00 00	.....( .....
a0	17 85 68 35 24 d3 9b 15	0a 9b 2c e1 bf 1c d3 2d	..h5\$... .....
b0	ed f1 10 99 97 b8 3f 32	22 1b dc 69 13 a1 ae 25	.....?2 "...i...%

“Cheers – here’s mine!”

What could possibly go wrong?

What if there is a 'loop'?

If  $1001 \cdot P = P$ , then there are only 1000 possible values for  $n \cdot P$ , **no matter how large  $n$  is!**

Loops can be avoided. Other (known and unknown!) weaknesses remain possible.

# Are we using 'weak' curves?

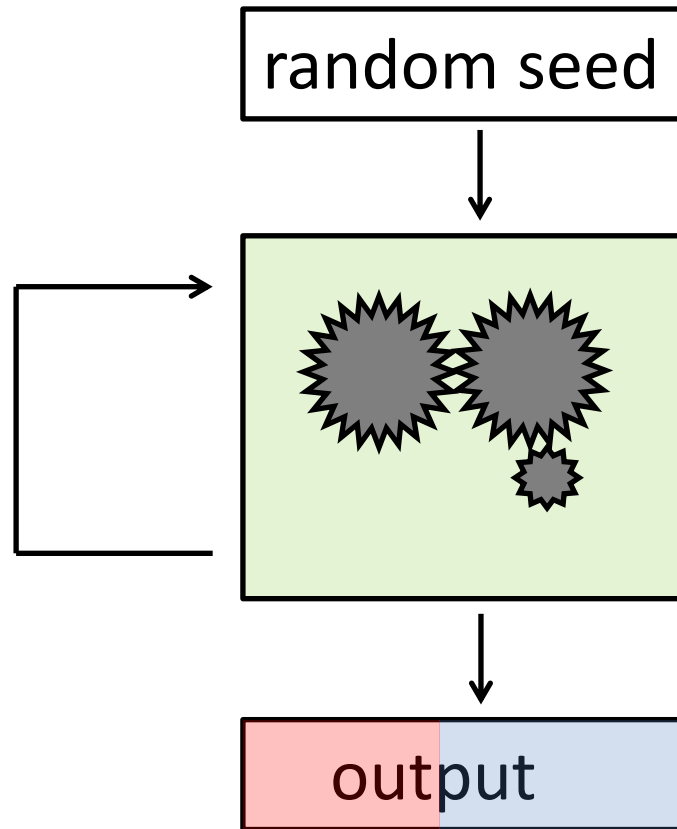
NIST P-256:

$$y^2 = x^3 - 3x +$$

4105836372515214212932612978004  
7268409114441015993725554835256  
314039467401291

**WHAT???**

# Random number generators

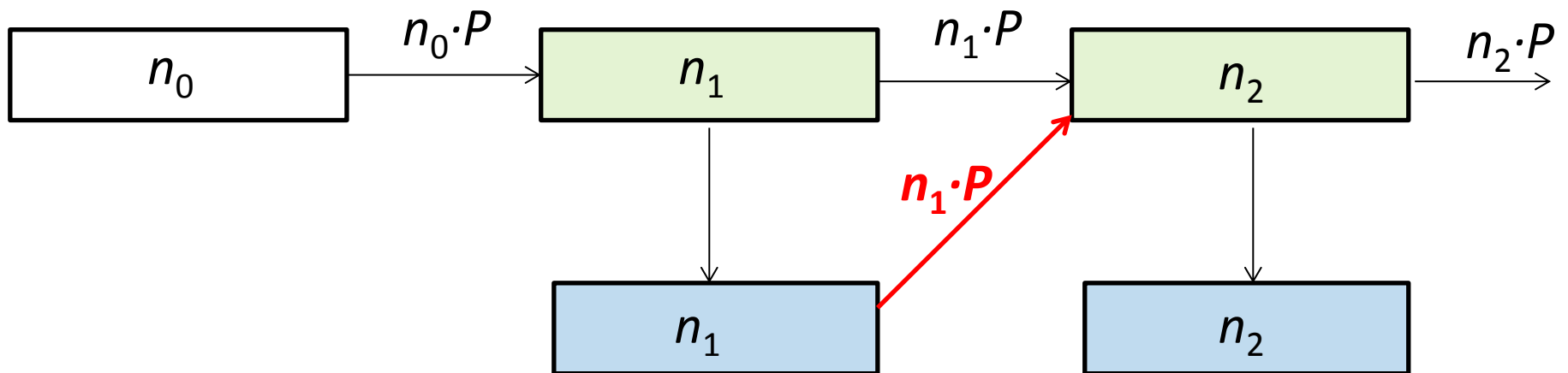


# Random number generators

Discrete Logarithm Problem:

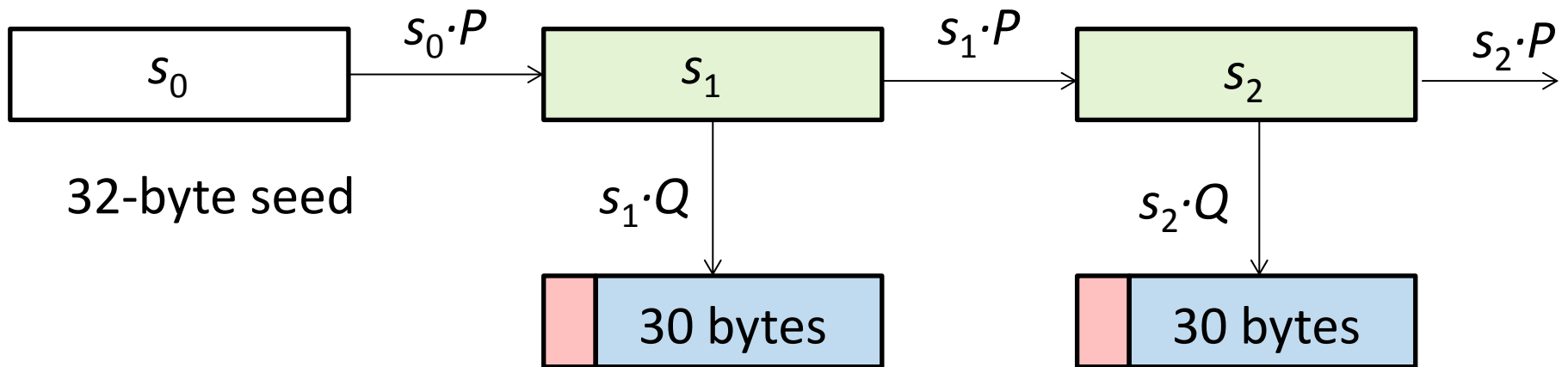
$$n \rightarrow n \cdot P$$

gives “random” points/numbers.



# Random number generators

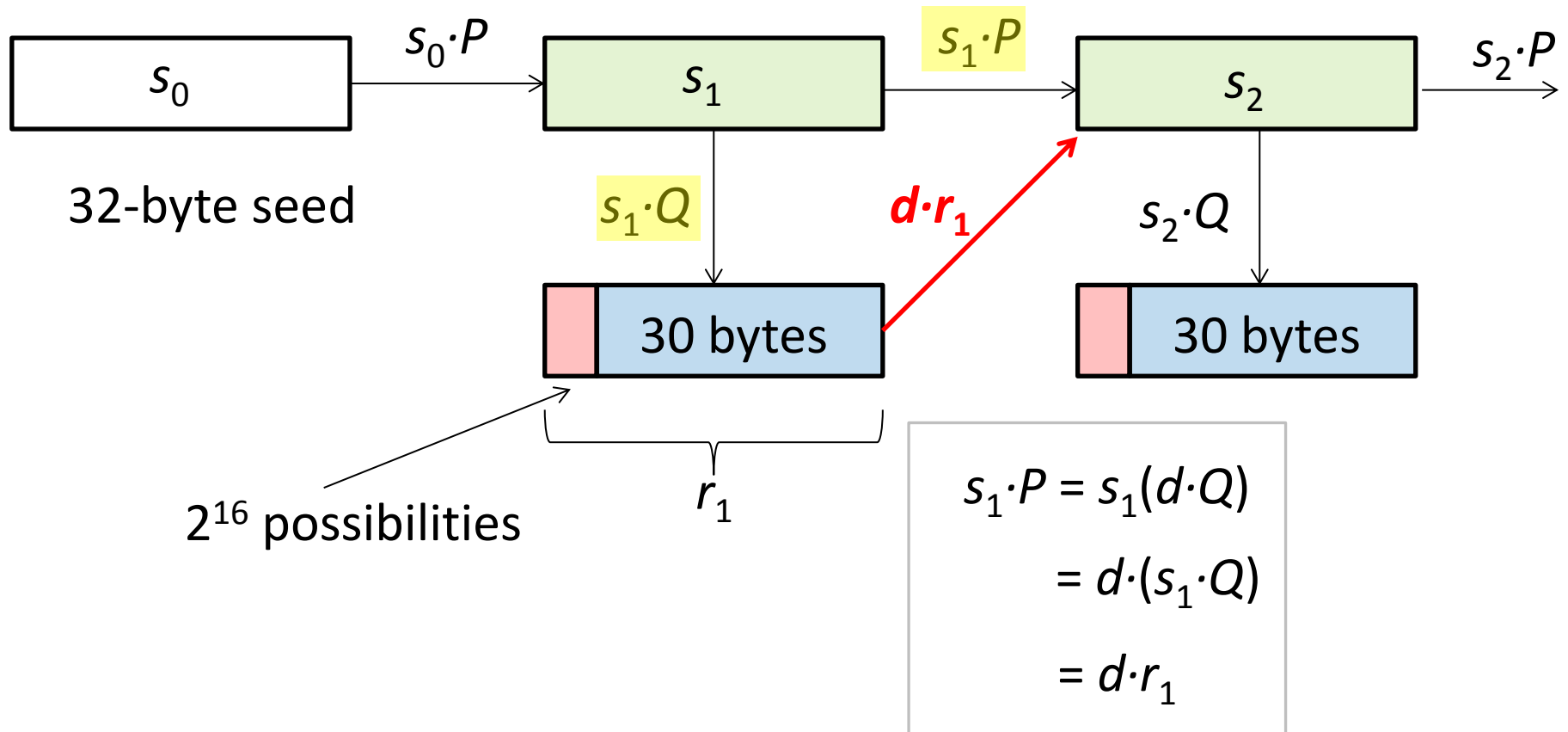
Given: elliptic curve with two points  $P$  and  $Q$ .



*Note: ideas from this slide and the next are borrowed from Bernstein, Heninger and Lange (NCSC '14).*

# Random number generators

Fact:  $P=d \cdot Q$  for some (large) number  $d$ .





# So who, if anyone, knows *d*?

NIST SP 800-90A

January 2012

## **Acknowledgements**

The National Institute of Standards and Technology (NIST) gratefully acknowledges and appreciates contributions by Mike Boyle and Mary Baish from NSA for assistance in the development of this Recommendation. NIST also thanks the many contributions by the public and private sectors.

“Dual\_EC\_DRBG”

# Conclusion

Elliptic curve cryptography is a good idea because we can do with much smaller keys.

256-bit ECC  $\approx$  3072-bit RSA.

Elliptic curve crypto uses complicated maths. **That is its biggest weakness.**

# Thank you!

 @martijn\_grooten

[martijn.grooten@virusbtn.com](mailto:martijn.grooten@virusbtn.com)

[www.virusbtn.com](http://www.virusbtn.com)

**PS VB2015, Prague 30 Sep-2 Oct**