

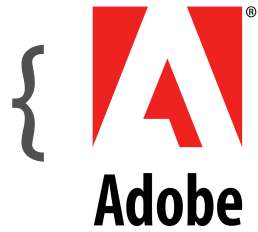
Invalid curve attack in JWE ECDH-ES

Antonio Sanso (@asanso) 

Security Engineer

Adobe Research Switzerland

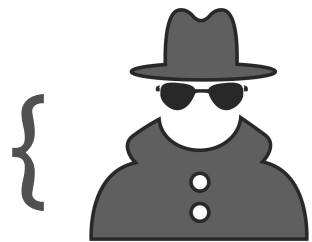
Who is this guy, BTW?



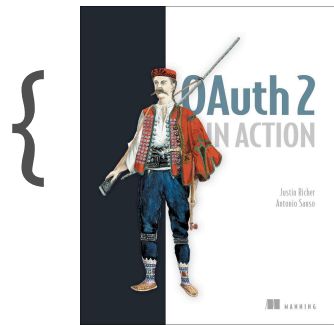
Security Engineer Adobe Research Switzerland



Internet Bug Bounty, Google Security Hall of Fame, Facebook Security Whitehat, GitHub Security Bug Bounty, Microsoft Honor Roll



Found vulnerabilities in OpenSSL , Google Chrome, Firefox, Safari

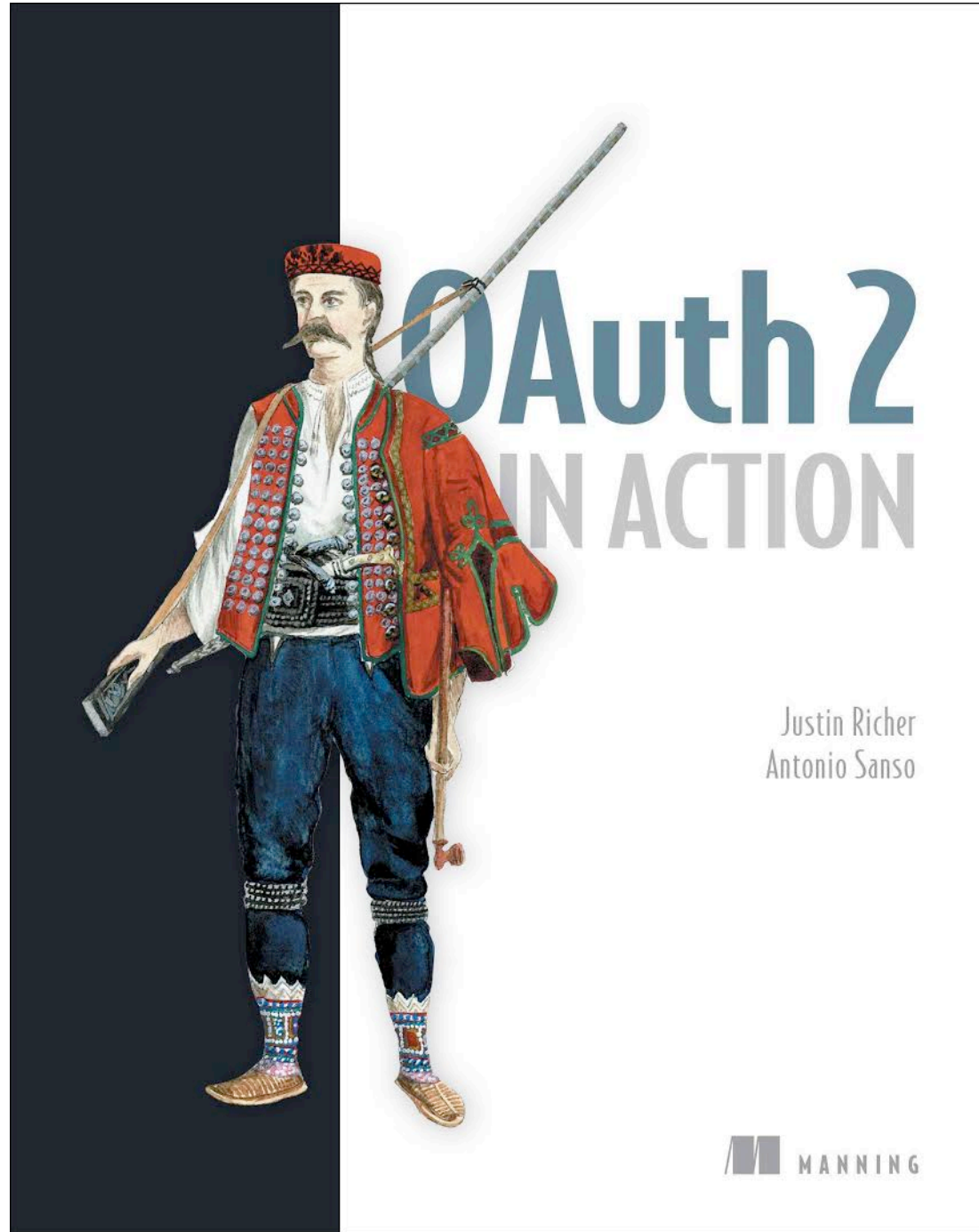


Co-Author of “OAuth 2 in Action”

Co-author of “OAuth 2 in Action”

<https://www.manning.com/books/oauth-2-in-action>

oauthsec





Agenda

{Introducing Elliptic Curve

{Introducing JOSE

{Invalid curve attack in JWE ECDH-ES

Acknowledgement

THANKS

{ Quan Nguyen from Google

{ Maintainers of the vulnerable libraries



When climbing the Elliptic learning Curve can go covfefe

**We strongly caution against
services involving the f
DashCoin, DigitalNote**

Timeline

2017-02-19: A member of
detailed discussion of the
2017-02-20: The Monero
thankfully it had not and t

On Mon, Jan 30, 2017 at 1:48 PM, Mike Hamburg <mike@shiftright.org> wrote:

> On Jan 30, 2017, at 12:41 PM, Antonio Sanso <asanso@adobe.com> wrote:

> On Nov 7, 2016, at 12:51 AM, Trevor Perrin <trevp@trevp.net> wrote:

> However, cofactor>1 can still have subtle and unexpected effects, e.g.
> see security considerations about "equivalent" public keys in RFC
> 7748, which is relevant to the cofactor multiplication "cV" in
> VXEdDSA, or including DH public keys into "AD" in Signal's (recently
> published) X3DH [3].

>
>
> may you shed some more light about this?
> What is the algorithm to find and "equivalent" public key?
[...]

> Second, two x's are equivalent if they differ by a c-torsion point. This .
> because the X25519 Diffie-Hellman key exchange algorithm is computing
> c*secret*P, which is the same as c*secret*(P+T) for points T such that c*T
> is the identity. Another way to describe these equivalent keys is that
> they're the x-coordinates of points Q such that c*Q = c*P.

I'll describe the same thing, but maybe this is simpler wording:

For X25519, just add a point of low order (i.e. order=2, 4, or 8) onto
an X25519 public key. Because X25519 private keys are multiples of
the cofactor (8), the added point won't change DH results.

I.e. for public key A, some private key b, and low-order point L:

$$b(A+L) = bA + bL = bA$$

History of Elliptic Curve (Cryptography)

{ **Diophantus** (Arithmetica ~3rd century AD)

{ **Henri Poincaré** (1901)

{ **André Weil** (1929)

{ **Hendrik Lenstra** (1984)

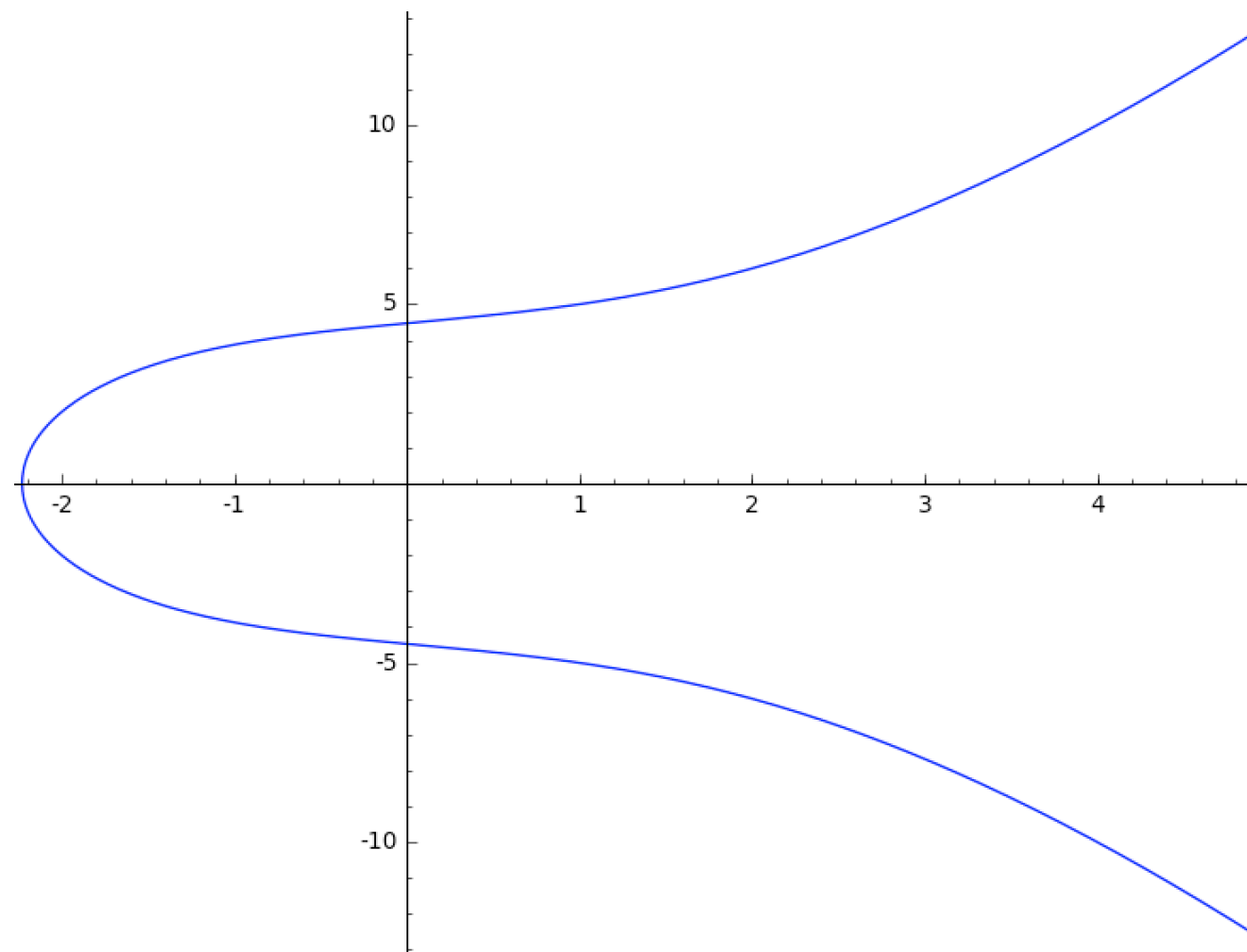
{ **Koblitz** and independently **Miller** (1985)

What is an Elliptic Curve

An elliptic curve is the set of solutions defined by an equation of the form

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

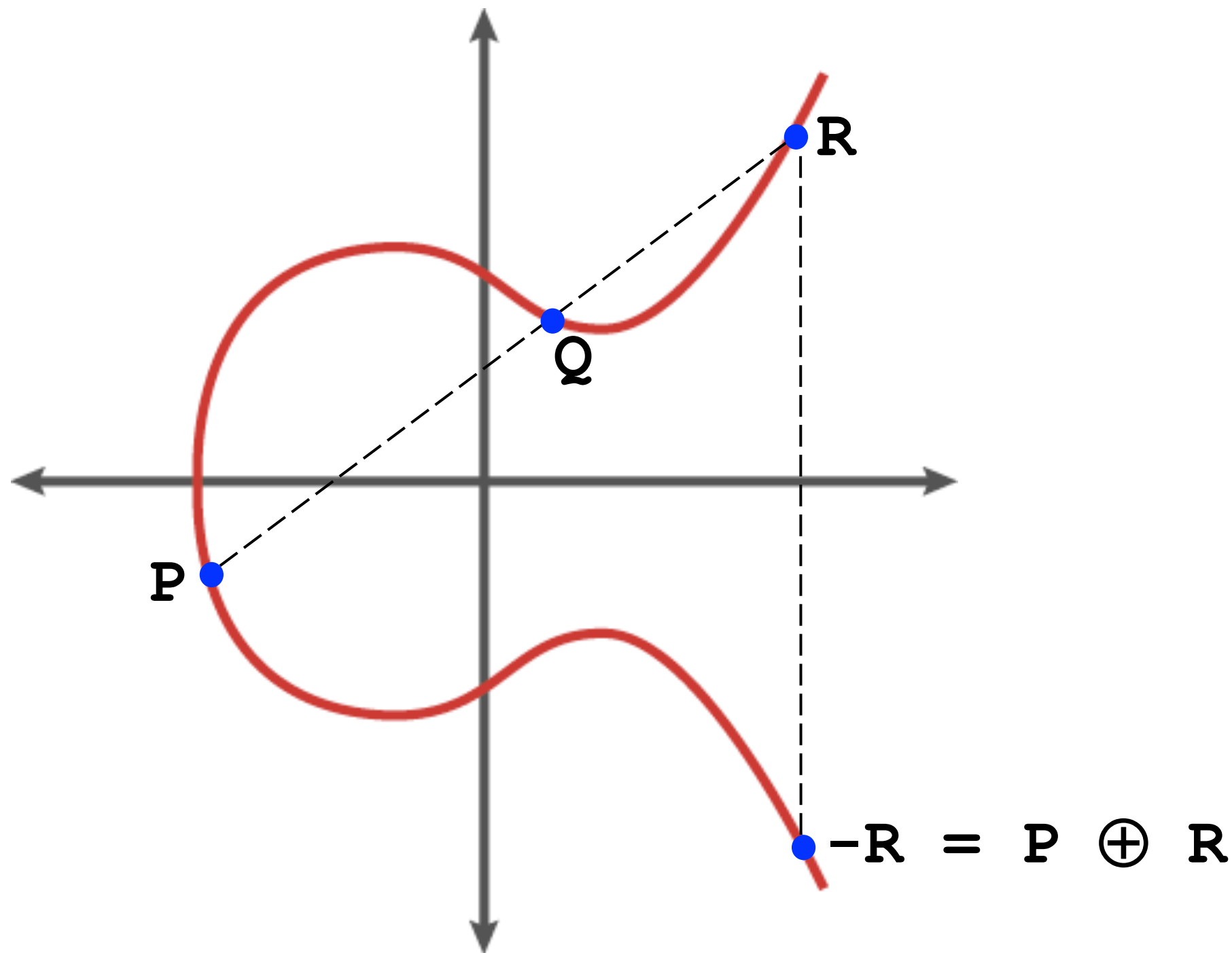
What is an Elliptic Curve



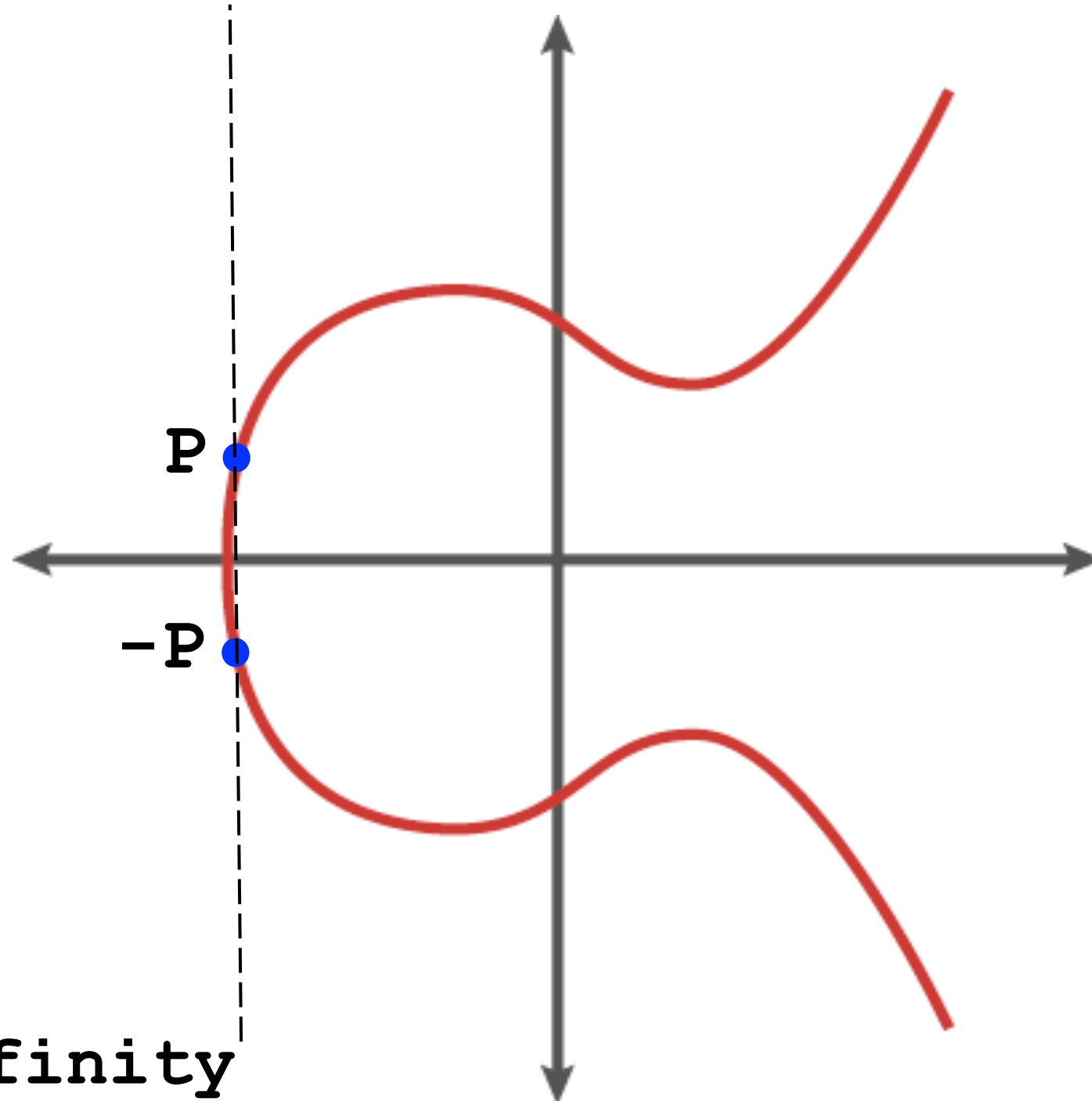
$$y^2 = x^3 + 4x + 20$$



Elliptic Curve Addition

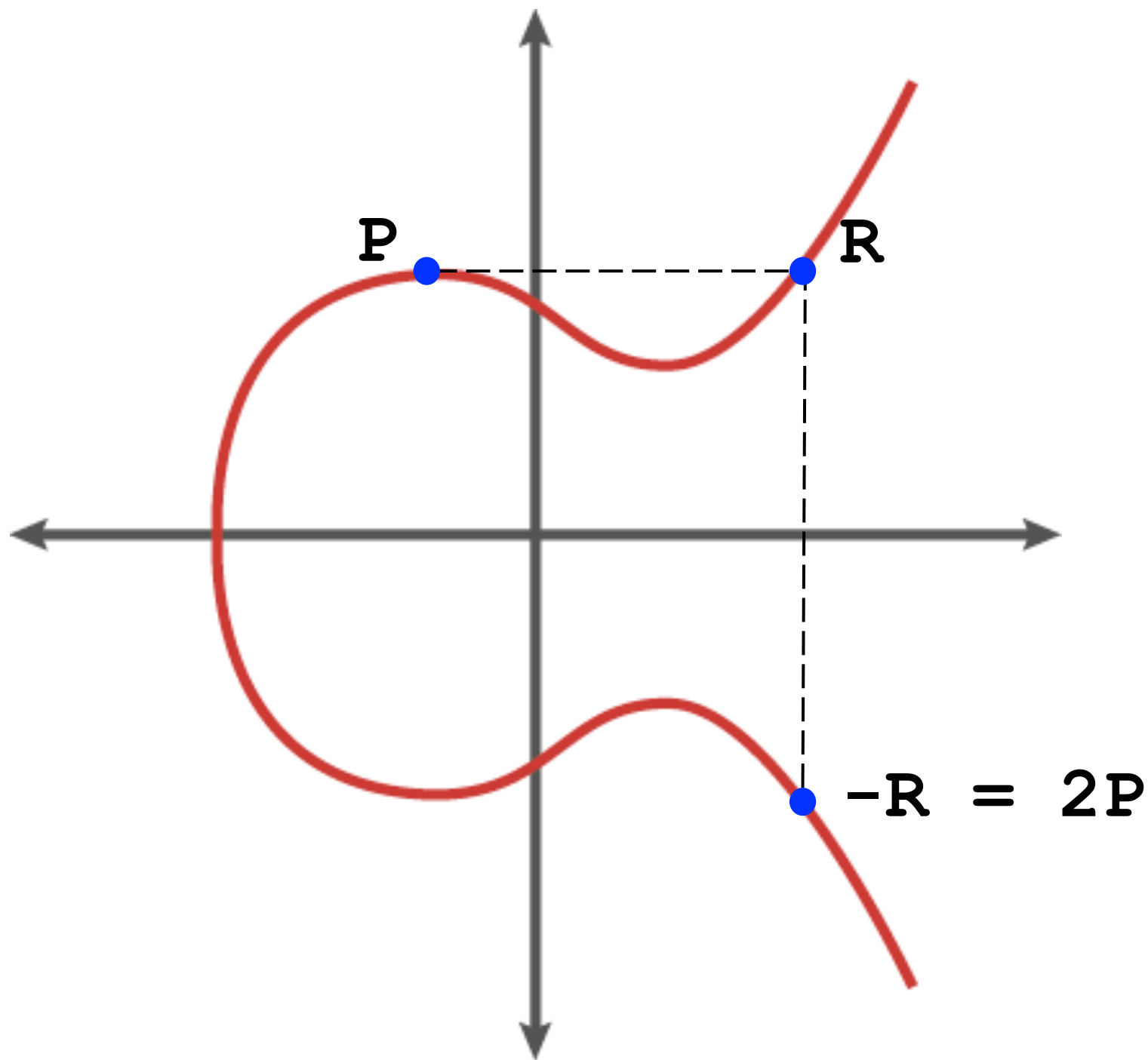


Elliptic Curve Addition

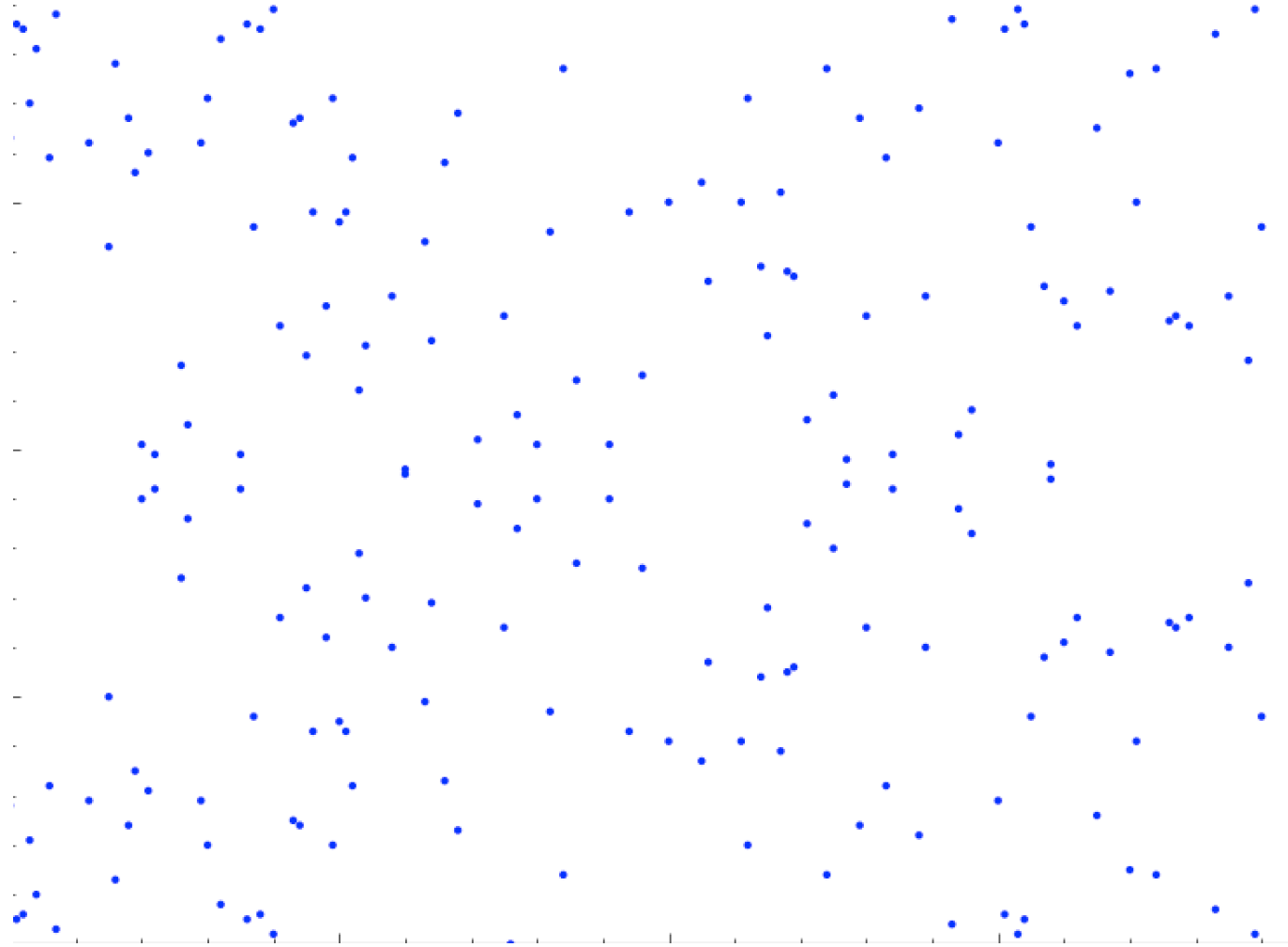




Elliptic Curve Point Multiplication



Elliptic Curve over Finite Fields



$$y^2 = x^3 + 4x + 20$$

over Finite Field of size 191

History of JOSE

- { Somehow inspired by some exploration done by **Magic Signatures**
- { Merged with Web Object Signing and Encryption WG (**WOES**)
- { First draft of **JWT** in July of 2011
- { **RFCs** in 2015

JOSE's family

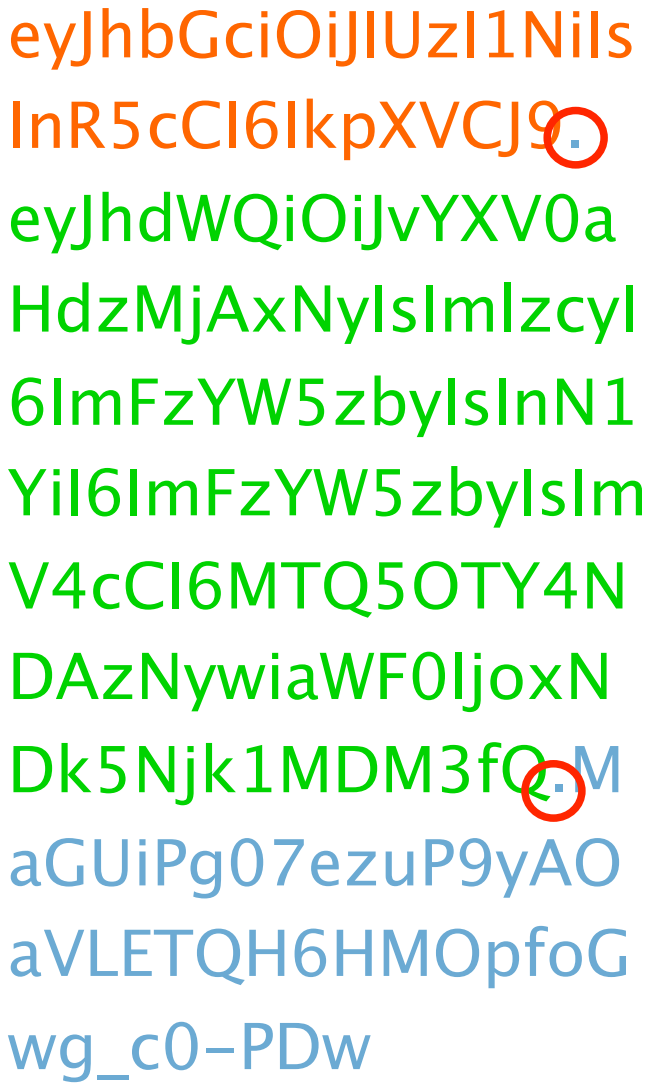
{JWT (JSON Web Token RFC 7519)

{JWA (JSON Web Algorithms RFC 7518)

{JWK (JSON Web Key RFC 7517)

{JWS (JSON Web Signature RFC 7515)

{JWE (JSON Web Encryption RFC 7516)



Header

Claims

Signature

HMAC

JWE (JSON Web Encryption RFC 7516)

- { **The JWE Protected Header**
- { **The JWE Encrypted Key**
- { **The JWE Initialization Vector**
- { **The JWE Ciphertext**
- { **The JWE Authentication Tag**

JWE (JSON Web Encryption RFC 7516)

Encoded

PASTE A TOKEN HERE

```
eyJhbGciOiJSU0EtT0FFUCIsImVuYyI6IkeYNTZHQ0  
0ifQ
```

Decoded

EDIT THE PAYLOAD AND SECRET (ONLY HS256 SUPPORTED)

HEADER: ALGORITHM & TOKEN TYPE

```
{  
  "alg": "RSA-OAEP",  
  "enc": "A256GCM"  
}
```

Encoded

PASTE A TOKEN HERE

```
eyJhbGciOiJFQ0RILUVTK0ExMjhLVyIsImVuYyI6Ik  
ExMjhdQkMtSFMyNTYiLCJlcGsiOi0nsia3R5Ijo  
iRUMiLCJ4Ijo  
iWE9YR1E5XzZ  
RQ3ZCZzN1OH  
ZDSS1VZEJ2SU  
NBRWN0TkJyZ  
nFkN3RHN29R  
NCIsInkiOiJ  
oUW9XTm90bk  
56S2x3aUNuZ  
UprTElxRG5U  
Tnc3SXNkQk  
M1M1ZVcVZq  
VkpjIiwiY3J  
2Ijo  
iUC0yNTYifX0
```

Decoded

EDIT THE PAYLOAD AND SECRET (ONLY HS256 SUPPORTED)

HEADER: ALGORITHM & TOKEN TYPE

```
{  
  "alg": "ECDH-ES+A128KW",  
  "enc": "A128CBC-HS256",  
  "epk": {  
    "kty": "EC",  
    "x": "X0XGQ9_6QCvBg3u8vCI-UdBvICAecNNBrfqd7tG7oQ4",  
    "y": "hQoWNotnNzKlwiCneJkLIqDnTNw7IsdBC53VUqVjVJc",  
    "crv": "P-256"  
  }  
}
```

Public key cryptography

New direction in cryptography

New Directions in Cryptography

Invited Paper

Whitfield Diffie and Martin E. Hellman

Abstract Two kinds of contemporary developments in cryptography are examined. Widening applications of teleprocessing have given rise to a need for new types of cryptographic systems, which minimize the need for secure key distribution channels and supply the equivalent of a written signature. This paper suggests ways to solve these currently open problems. It also discusses how the theories of communication and computation are beginning to provide the tools to solve cryptographic problems of long standing.

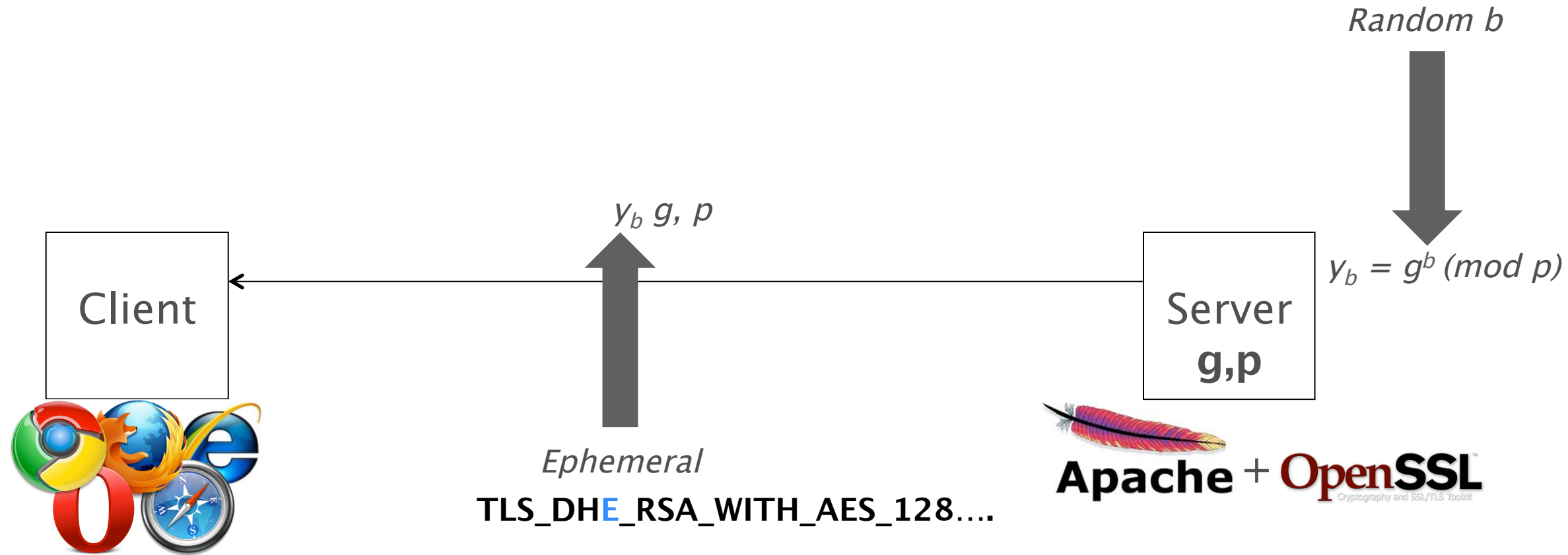
1 INTRODUCTION

We stand today on the brink of a revolution in cryptography. The development of cheap digital hardware has freed it from the design limitations of mechanical computing and brought the cost of high grade cryptographic devices down to where they can be used in such commercial applications as secure

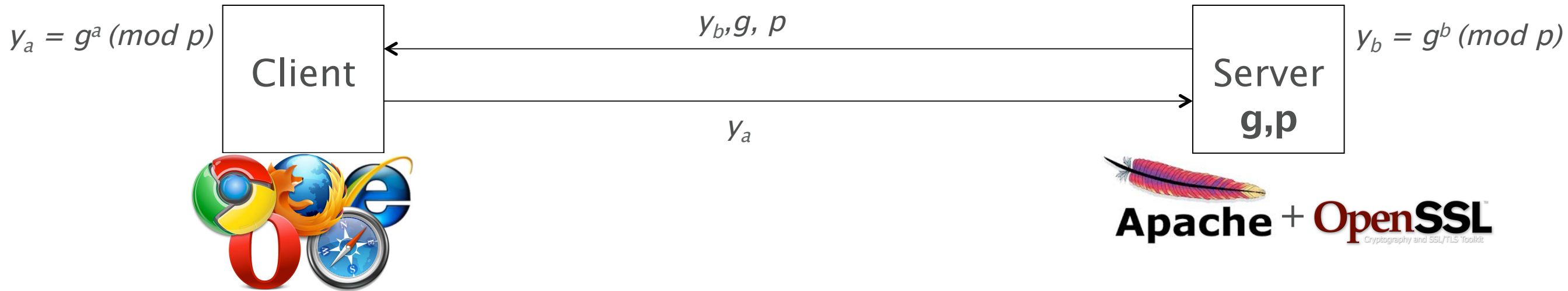
communications over an insecure channel order to use cryptography to insure privacy, however, it currently necessary for the communicating parties to share a key which is known to no one else. This is done by sending the key in advance over some secure channel such a private courier or registered mail. A private conversation between two people with no prior acquaintance is a common occurrence in business, however, and it is unrealistic to expect initial business contacts to be postponed long enough for keys to be transmitted by some physical means. The cost and delay imposed by this key distribution problem is a major barrier to the transfer of business communications to large teleprocessing networks.

Section III proposes two approaches to transmitting keying information over public (i.e., insecure) channel without compromising the security of the system. In *public key cryptosystem* enciphering and deciphering are governed by distinct keys, E and D , such that computing D from E is computationally infeasible.

DH Key Exchange



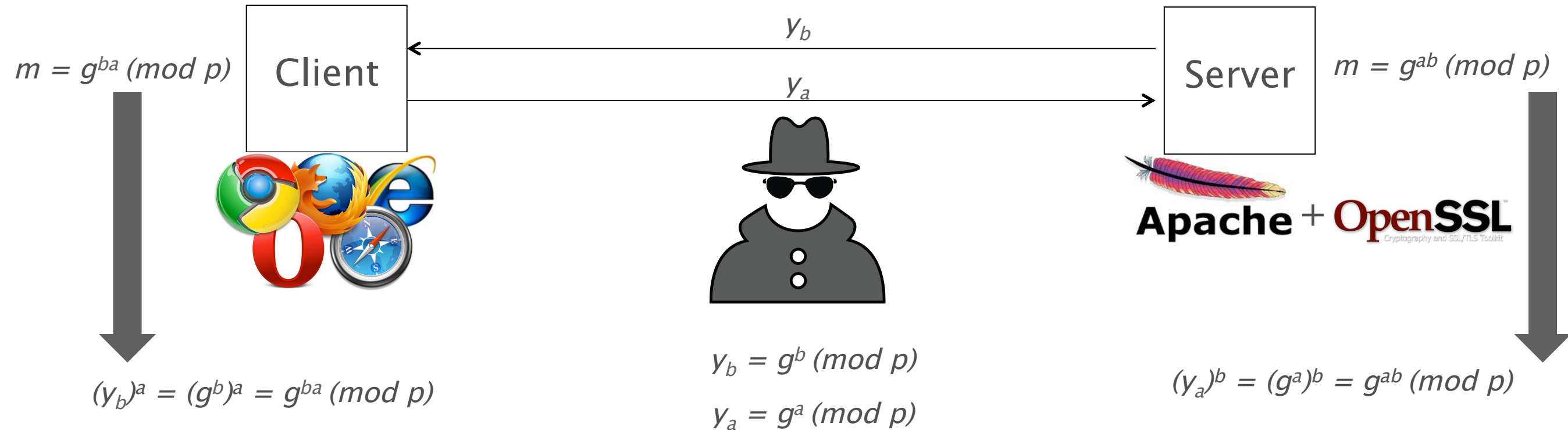
DH Key Exchange





DH Key Exchange

$m = \text{premaster key}$

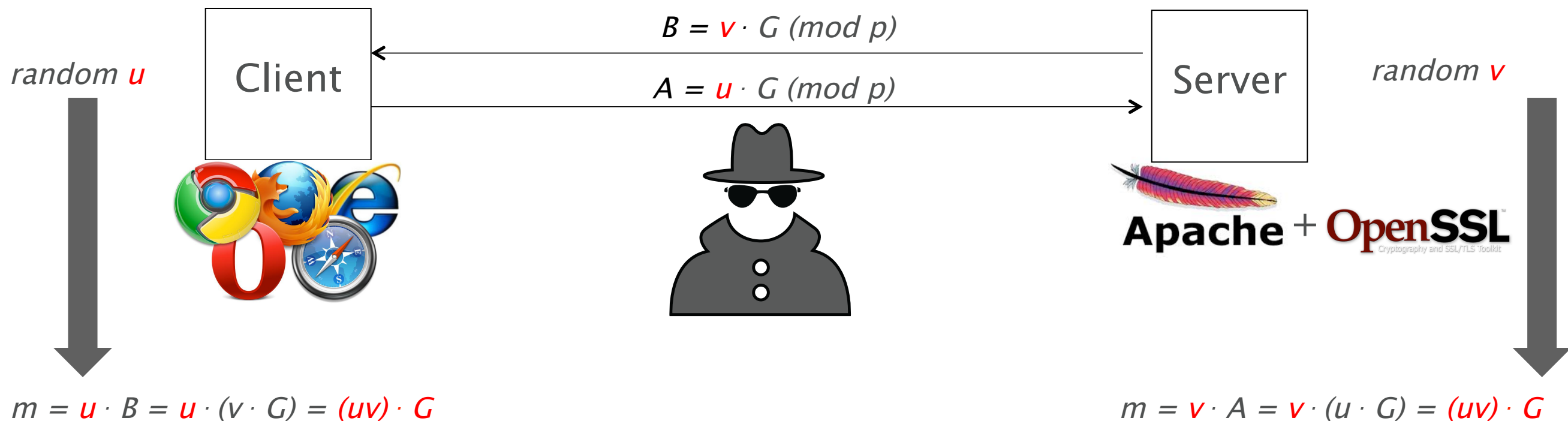




ECDH Key Exchange

TLS_ECDHE_RSA_WITH_AES_128

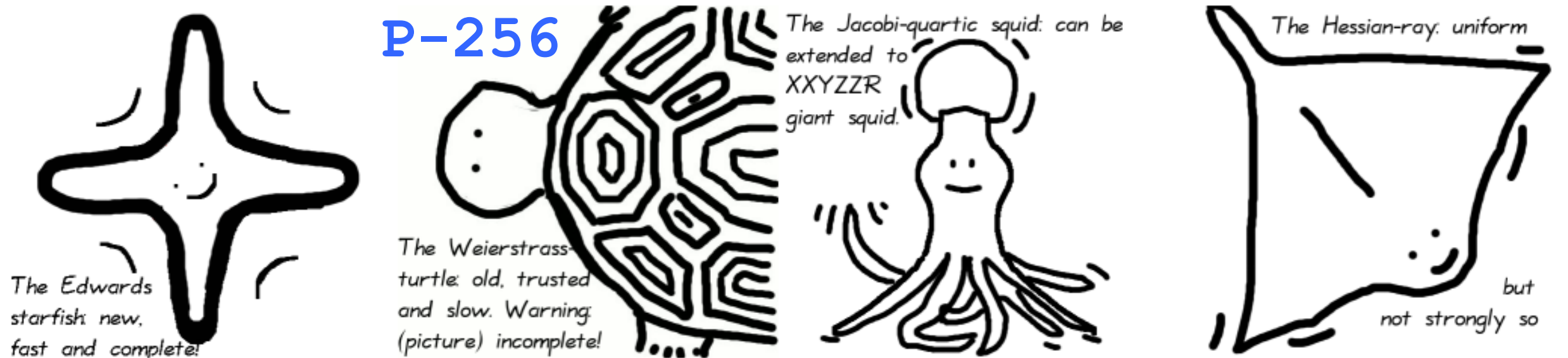
Fix prime p and point G on a curve



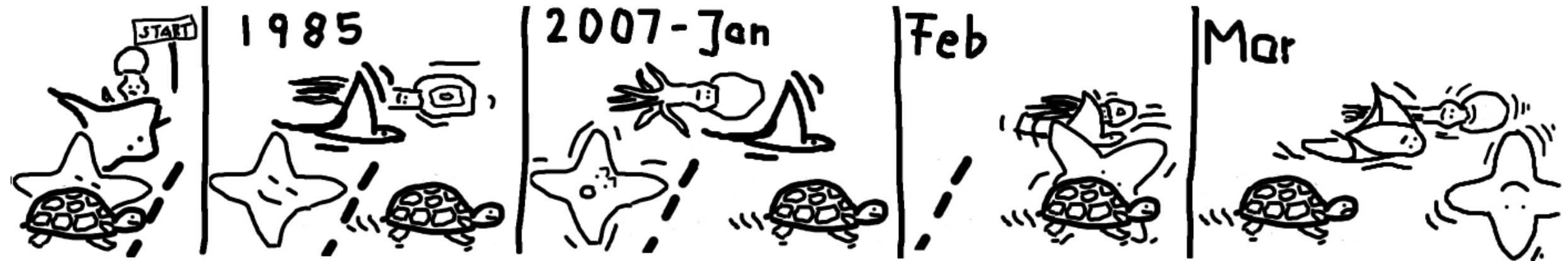
$m = \text{premaster key}$

Which Curve?

Portraits of the competitors



The race between all competitors

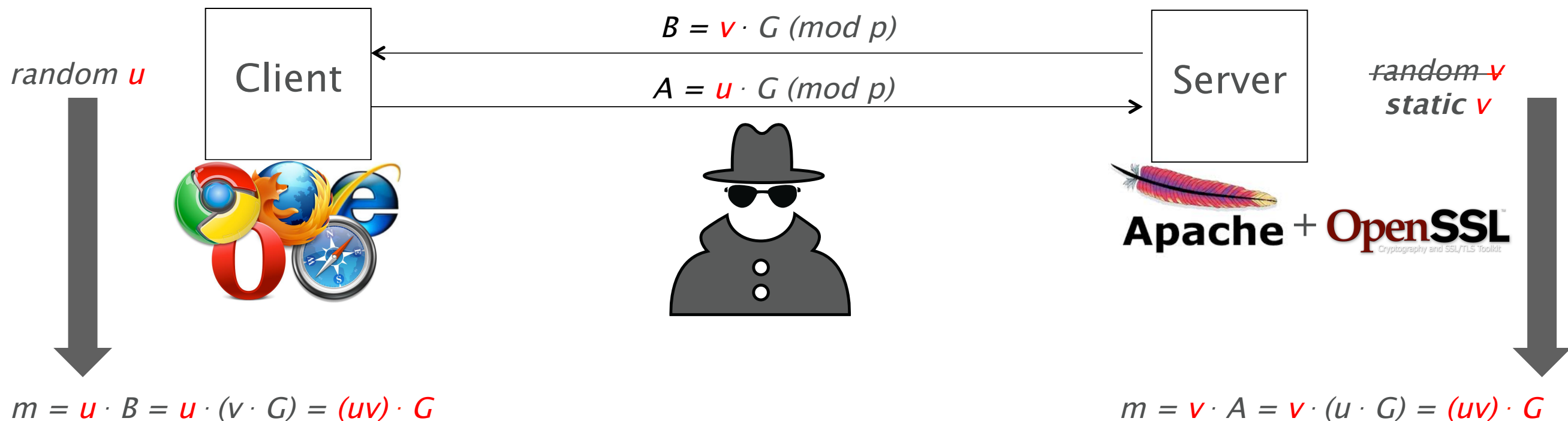




Invalid Curve Attack Requirement #1

(Static ephemeral keys) TLS_ECDHE_RSA_WITH_AES_128

Fix prime p and point G on a curve



$m = \text{premaster key}$

Invalid curve attack in JWE ECDH-ES



Thomas H. Ptáček

@tqbf

Following



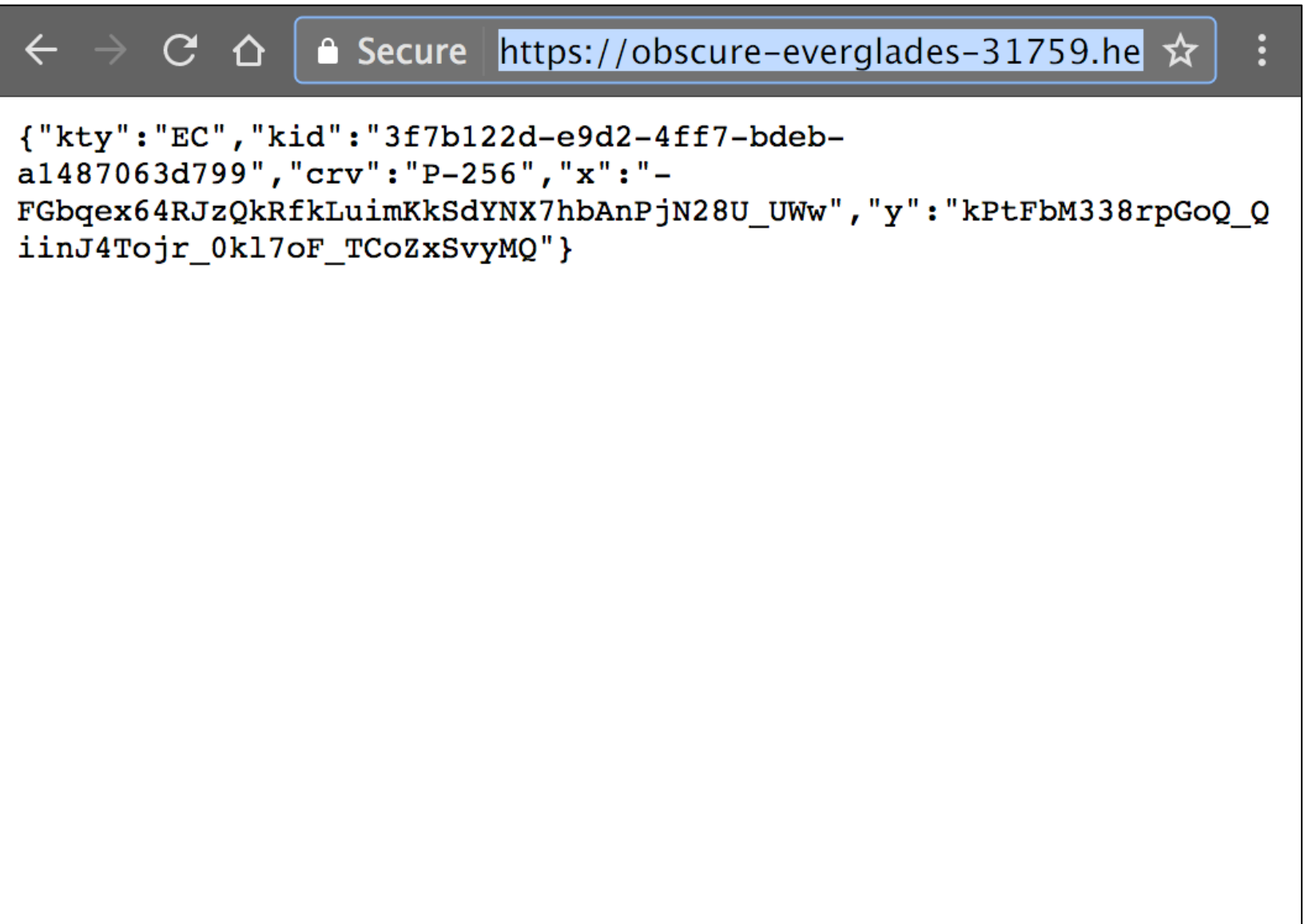
P-Curve ECDH Ephemeral-Static is literally standards speak for “invalid curve attack”. JWE named this construction “KICK ME”.

11:04 PM - 13 Mar 2017 from Near West Side, Chicago

JWE ECDH-ES

Key Agreement with Elliptic Curve Diffie-Hellman Ephemeral Static (ECDH-ES)

*Server
Key Static*



A screenshot of a web browser window. The address bar shows a secure connection to `https://obscure-everglades-31759.he`. The main content area displays a JSON object representing a JSON Web Key (JWK). The JSON object is: `{"kty": "EC", "kid": "3f7b122d-e9d2-4ff7-bdeb-a1487063d799", "crv": "P-256", "x": "-FGbgex64RJzQkRfkLuimKkSdYNX7hbAnPjN28U_UWw", "y": "kPtFbM338rpGoQ_QiinJ4Tojr_0kl7oF_TCoZxSvyMQ"}`

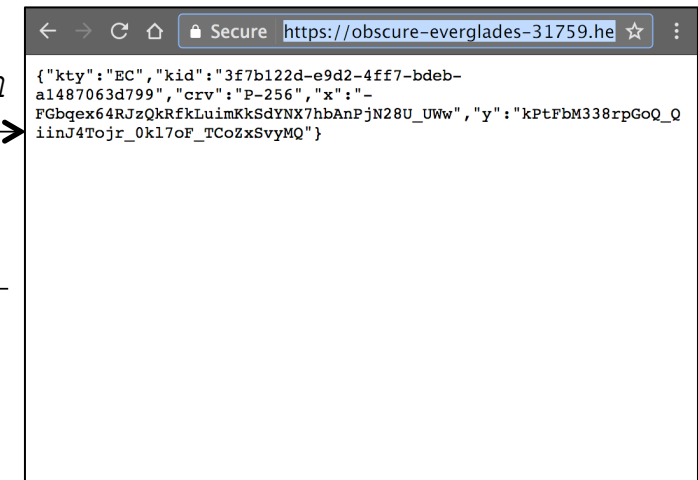
```
{ "kty": "EC", "kid": "3f7b122d-e9d2-4ff7-bdeb-a1487063d799", "crv": "P-256", "x": "-FGbgex64RJzQkRfkLuimKkSdYNX7hbAnPjN28U_UWw", "y": "kPtFbM338rpGoQ_QiinJ4Tojr_0kl7oF_TCoZxSvyMQ" }
```

JWE ECDH-ES

static **v**

GET **/ecdh-es-public.json** HTTP 1.1
Host: *obscure-everglades-31759.herokuapp.com*

Client



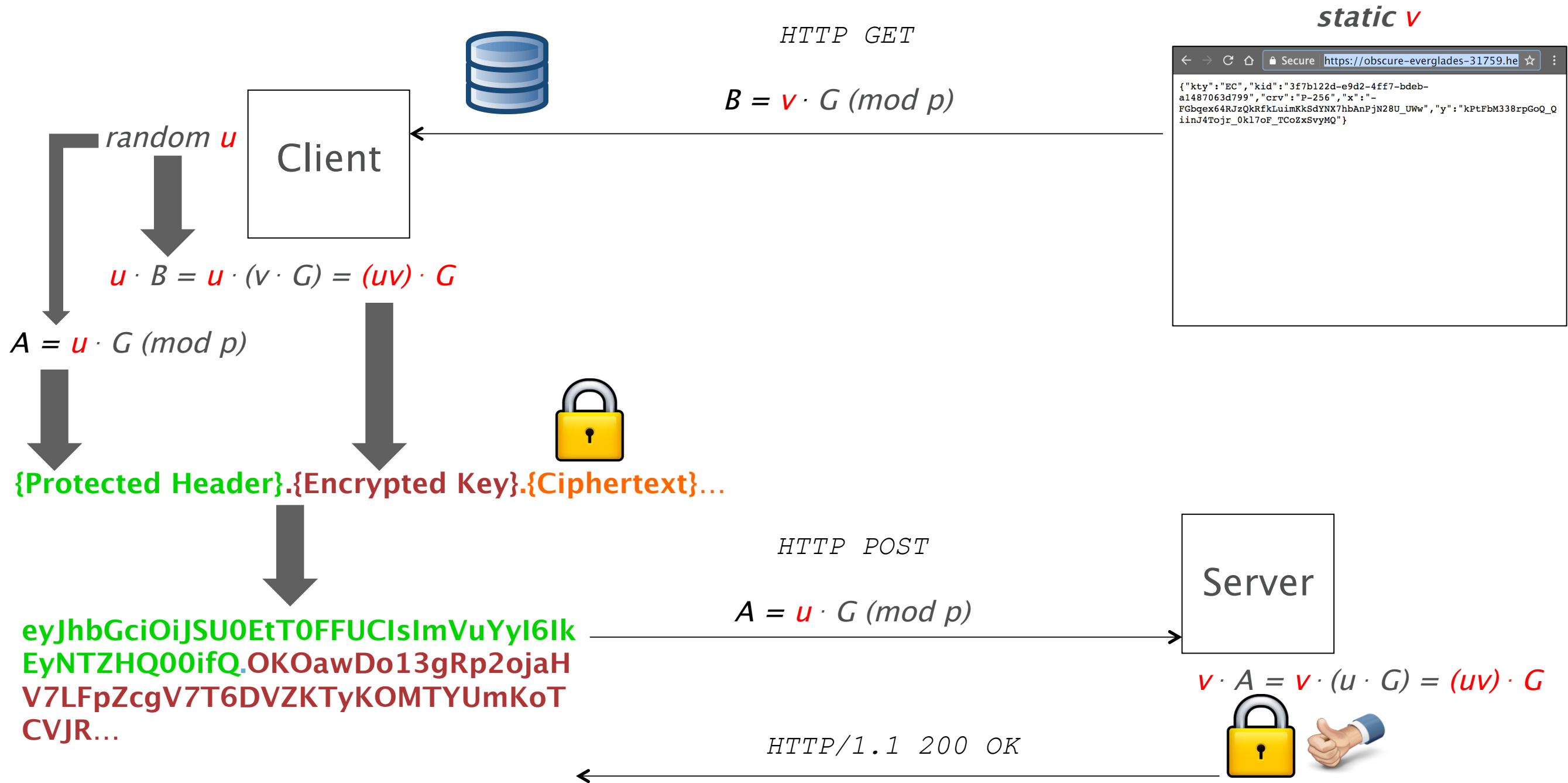
{"kty": "EC", "kid": "3f7b122d-e9d2-4ff7-bdeb-a1487063d799", "crv": "P-256", "x": "ufQkXr3L841ATLhZ4rQ4e--udQtLWawOxmjVjg88Y8Q", "y": "mEfxmKOAIqlEo9oAWpI8KUk82G7xh_2BOfTg1U5GPss"}

$$B = \mathbf{v} \cdot G \pmod{p}$$





JWE ECDH-ES



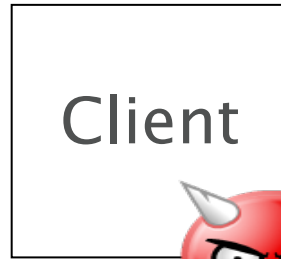


JWE ECDH-ES

static **v**

HTTP GET

$$B = \mathbf{v} \cdot G \pmod{p}$$



COVFEFE...

HTTP POST

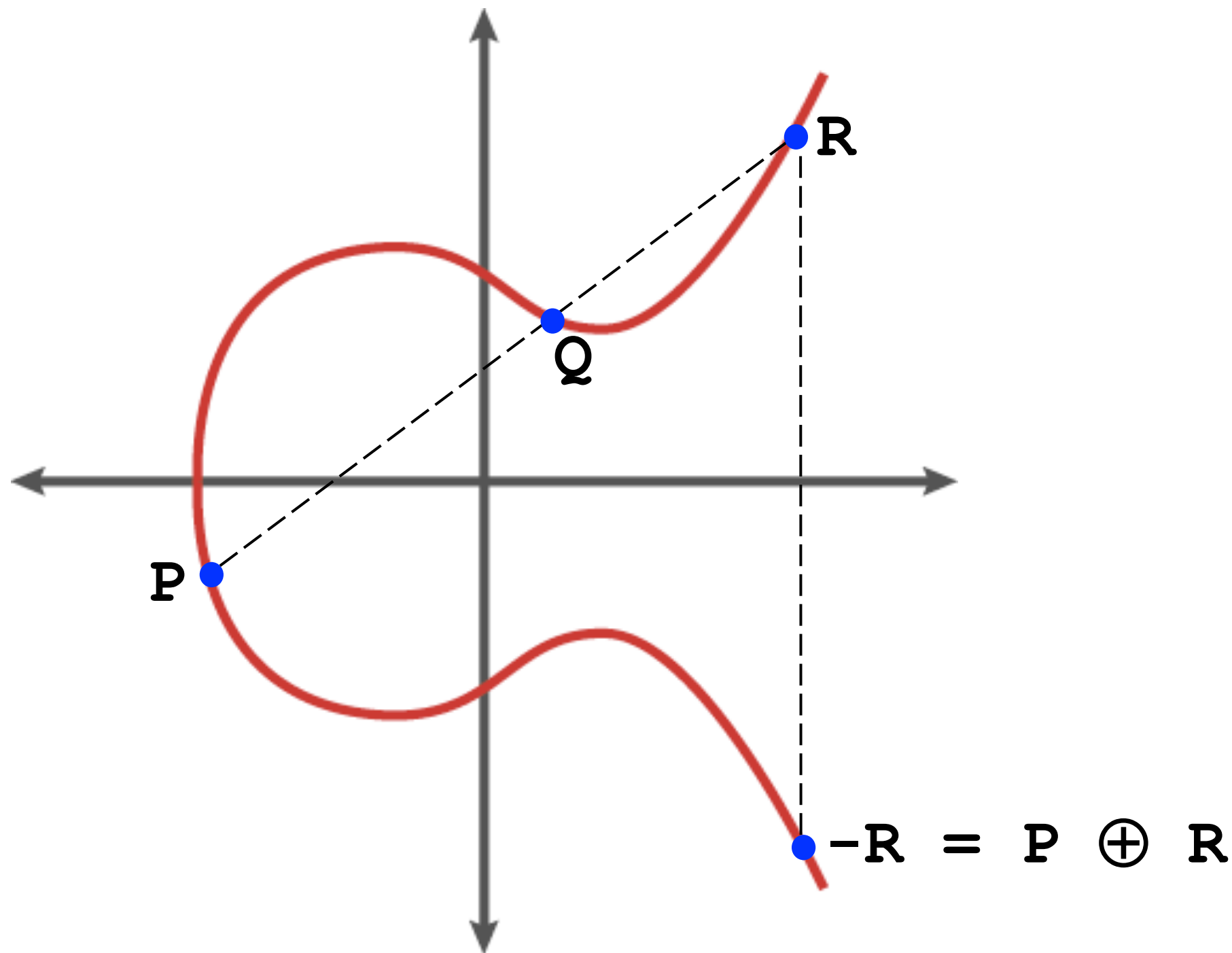


HTTP/1.1 400 BAD REQUEST





Addition on Weierstrass curve





Addition on Weierstrass curve

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

$$\text{for } x_1 \neq x_2 \quad (x_1, y_1) + (x_2, y_2) =$$

$$\lambda = (y_2 - y_1) / (x_2 - x_1)$$

$$x_3 = \lambda^2 - x_1 - x_2$$

$$y_3 = \lambda(x_1 - x_3) - y_1$$

$$\text{for } y_1 \neq 0 \quad (x_1, y_1) + (x_1, y_1) =$$

$$\lambda = (3x_1^2 + a) / (2y_1)$$

$$x_3 = \lambda^2 - x_1 - x_2$$

$$y_3 = \lambda(x_1 - x_3) - y_1$$

$$(x_1, y_1) + (x_1, -y_1) = \infty$$

$$(x_1, y_1) + \infty = (x_1, y_1)$$

No b here!!

Invalid curve attack in JWE ECDH-ES

```
SageMath Version 6.10, Release Date: 2015-12-18
Type "notebook()" for the browser-based notebook interface.
Type "help()" for help.
```

```
sage: ##### setup some NIST p-256 values
sage: #
sage: p256 = 115792089210356248762697446949407573530086143415290314195533631308867097853951
sage: a256 = p256 - 3
sage: b256 = 41058363725152142129326129780047268409114441015993725554835256314039467401291
sage:
sage: ## base point values
sage: gx = 48439561293906451759052585252797914202762949526041747995844080717082404635286
sage: gy = 36134250956749795798585127919587881956611106672985015071877198253568414405109
sage:
sage: ## order of the curve
sage: qq = 115792089210356248762697446949407573529996955224135760342422259061068512044369
sage:
sage: FF = GF(p256)
sage: E = EllipticCurve([FF(a256), FF(b256)])
sage:
sage: E
Elliptic Curve defined by  $y^2 = x^3 + 115792089210356248762697446949407573530086143415290314195533631308867097853948x + 41058363725152142129326129780047268409114441015993725554835256314039467401291$  over Finite Field of size 115792089210356248762697446949407573530086143415290314195533631308867097853951
```

```
sage: b256 = 5
sage: E = EllipticCurve([FF(a256), FF(b256)])
sage: E.order().factor()
2^2 * 3 * 7 * 13 * 2447 * 43333351749528183857746664058033075385207938864390055103100636196733549
sage: P = E.lift_x(2)*Integer(E.order()/2447)
sage: P.order()
2447
```

JWE ECDH-ES



static **v**

HTTP GET

$$B = v \cdot G \pmod{p}$$

Client



Use a low order (e.g. 2447) point M on an invalid curve



{Protected Header}.{Encrypted Key}.{Ciphertext}...

$$A = 1 \cdot M$$

$$A = 2 \cdot M$$

$$A = 3 \cdot M$$

...

HTTP POST

Server

HTTP/1.1 400 BAD REQUEST

$$v \cdot A = v \cdot (u \cdot M) = (uv) \cdot G$$



JWE ECDH-ES

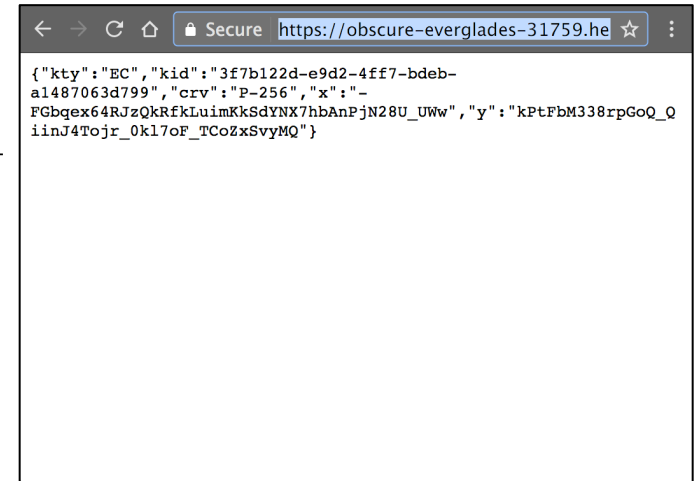


static **v**

HTTP GET

$$B = v \cdot G \pmod{p}$$

Client



Use a low order (e.g. 2447) point M on an invalid curve

$$A = 1 \cdot M$$

$$A = 2 \cdot M$$

$$A = 3 \cdot M$$

...

...

{Protected Header} .**{Encrypted Key}**.**{Ciphertext}**...

$$A = x \cdot M$$

HTTP POST

Server

$$v \equiv x \pmod{2447}$$

$$v \cdot A = v \cdot (x \cdot M) = (xv) \cdot M$$

HTTP/1.1 200 OK

Invalid curve attack in JWE ECDH-ES

Finally we can repeat the same steps for different **invalid curves** and **combine** the result using the Chinese Remainder Theorem **(CRT) for the win!!**

It's demo time

Rapid

A fatal exception 0E has occurred at 0028:C0011E36 in UXD UMM(01) + 00010E36. The current application will be terminated.

- * Press any key to terminate the current application.
- * Press CTRL+ALT+DEL again to restart your computer. You will lose any unsaved information in all applications.

Press any key to continue _

Affected libraries

{ **node-jose** (from Cisco Systems)

{ **jose2go**

{ **jose4j, Nimbus JOSE+JWT, Apache CFX ***

{ **go-jose** (this is the original library found vulnerable by Quan Nguyen from Google)

* Affected was the default Java SUN JCA provider that comes with Java prior to version 1.8.0_51.

Fixes

- { **For libraries:** validate the received public key (being sure they are on the curve)
- { **For the specification:** Elliptic Curve are kind theory is kind of tough, DO HELP practitioners to do the right thing!!

Small subgroup attacks on EC



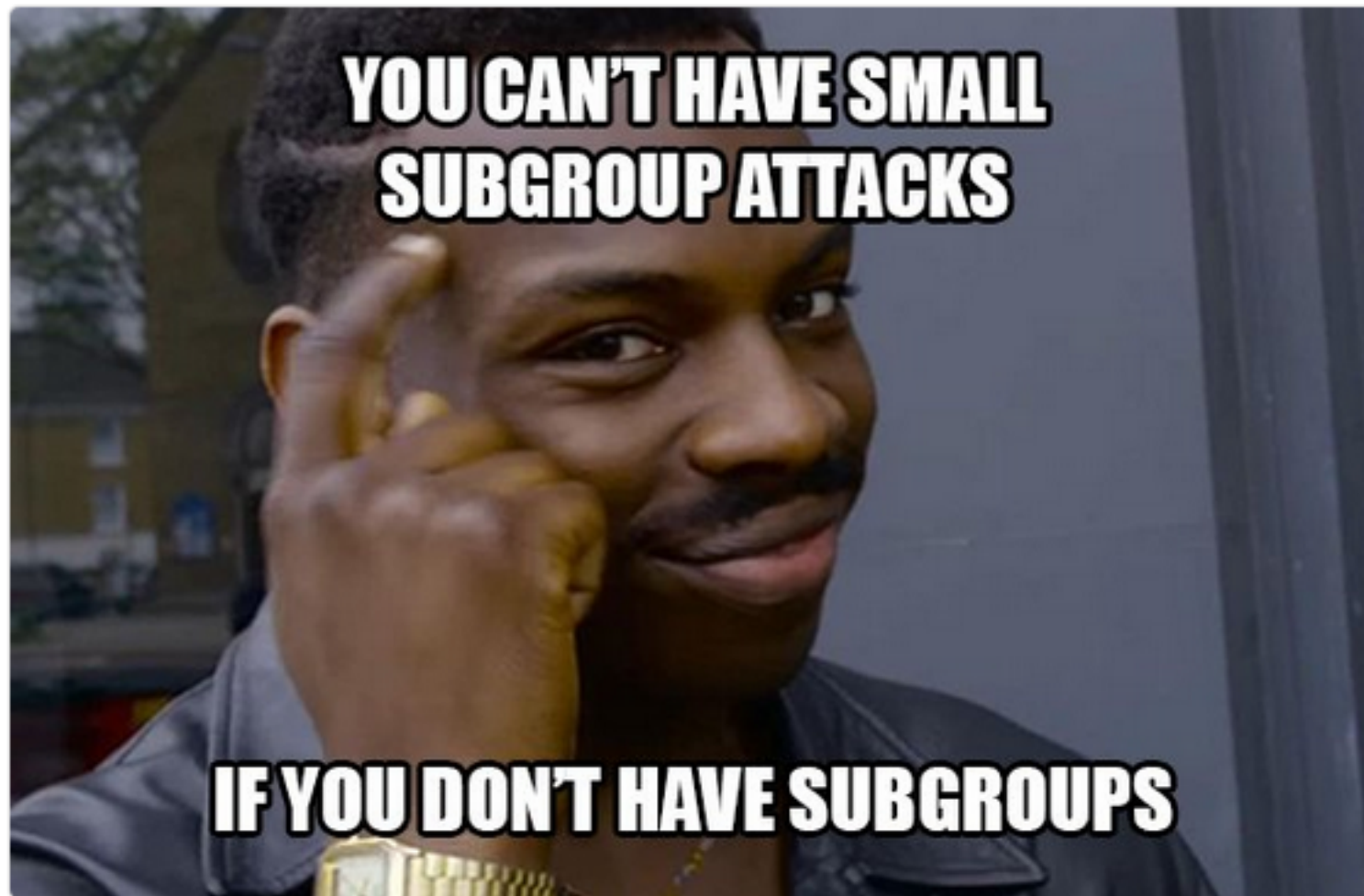
Tony Arcieri

@bascule

Following



Replying to @asanso @isislovecruft



6:08 PM - 21 Jun 2017

<https://www.rfc-editor.org/rfc/rfc8037.txt>

References

- { <http://intothesynergy.blogspot.ch/>
- { <https://www.manning.com/books/oauth-2-in-action>
- { <http://blogs.adobe.com/security/2017/03/critical-vulnerability-uncovered-in-json-encryption.html>
- { <https://getmonero.org/2017/05/17/disclosure-of-a-major-bug-in-cryptonote-based-currencies.html>
- { @asanso 

Questions?

