

Modern Cryptography Magic Workshop

Aleksei Klimenko

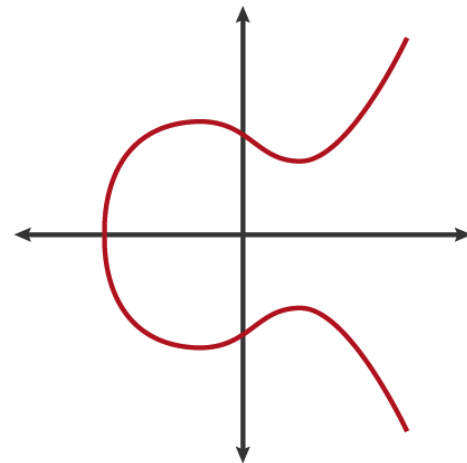
Modern Cryptography Magic Workshop

- Elliptic curves cryptography basics
- Schnorr signature algorithm
- BLS protocol
- NODR crypto-protocol (BLS-based)
- Exercise

Modern Cryptography Magic Workshop

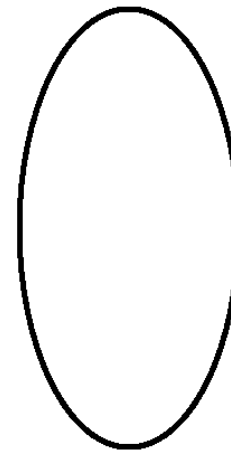
Elliptic curves cryptography basics

I DON'T UNDERSTAND WHY
PEOPLE GET CONFUSED ...
I DON'T LOOK ANYTHING
LIKE YOU!



ELLIPTIC CURVE

I THINK IT'S THE NAME!
LET'S ASK JAVA AND
JAVASCRIPT TO SEE HOW
THEY DEAL WITH IT



ELLIPSE

Modern Cryptography Magic Workshop

Elliptic curves cryptography basics

Elliptic curve equation

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

Modern Cryptography Magic Workshop

Elliptic curves cryptography basics

Elliptic curve equation

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

Point addition operation

$$P + R = Q \quad Q - R = P$$

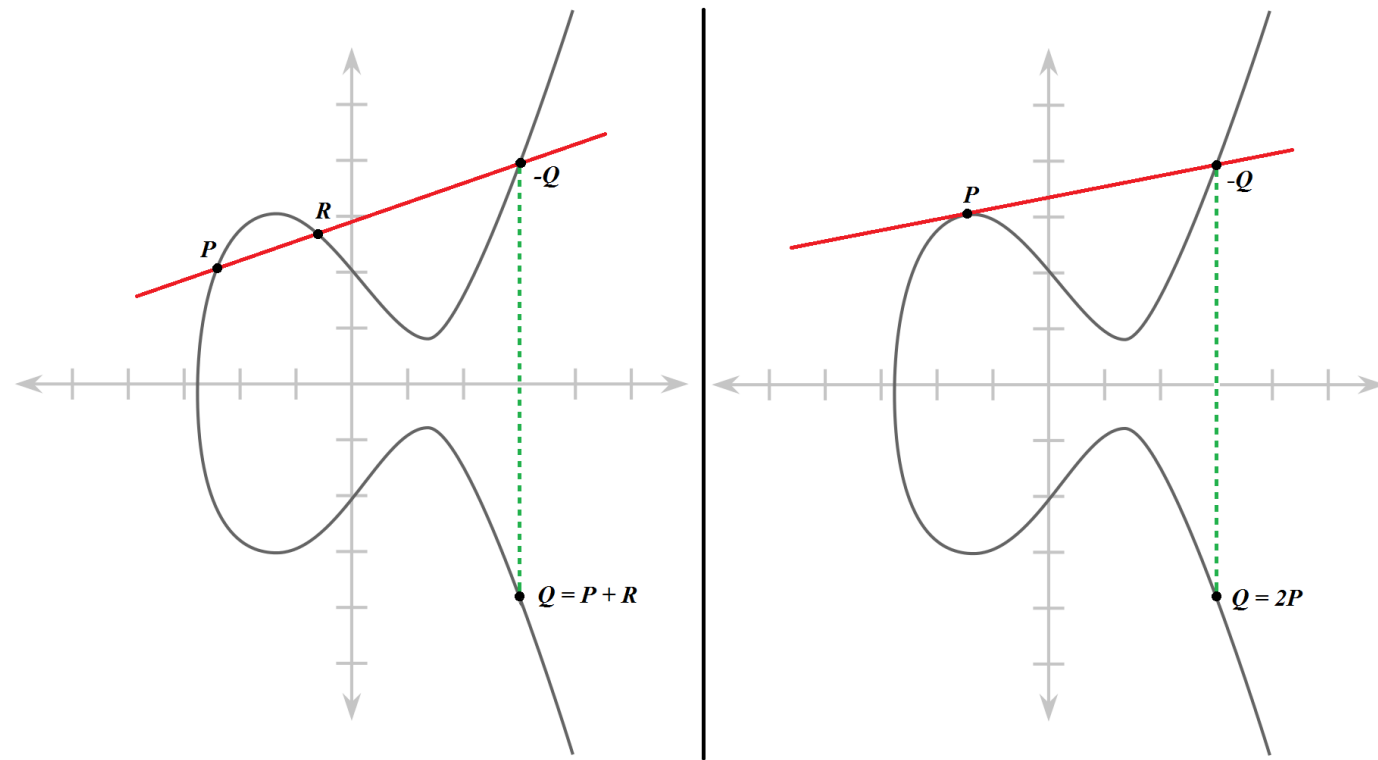
$$A + (B + C) = (A + B) + C$$

$$P + O = O + P = P$$

$$P + P = 2P$$

$$P + P + P = 3P$$

$$P + P + P + P = 4P$$



Modern Cryptography Magic Workshop

Elliptic curves cryptography basics

Addition of points $P_1=(x_1,y_1)$ and $P_2=(x_2,y_2)$ of an elliptic curve $E: y^2=x^3+ax+b$ can be easily computed using the following formulas:

$$P_1 + P_2 = P_3 = (x_3, y_3)$$

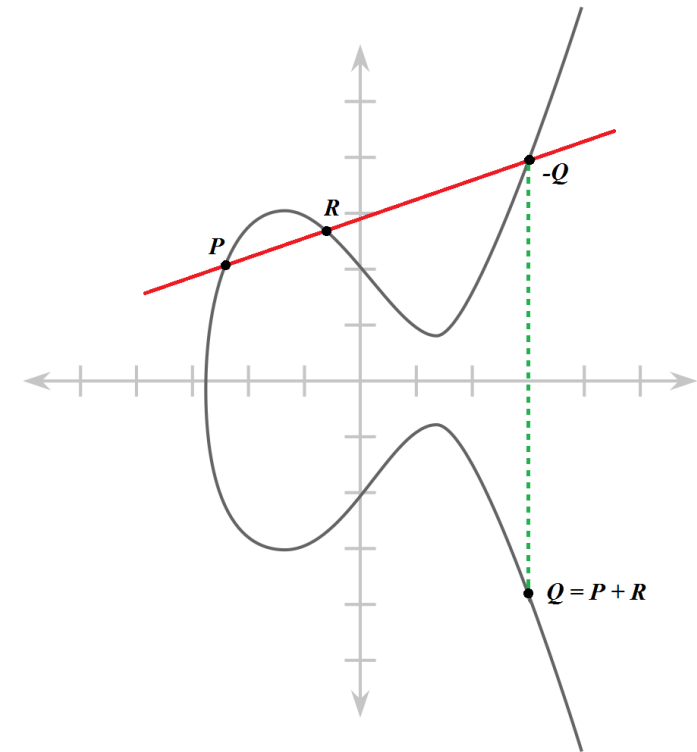
where

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

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

and

$$\lambda = \begin{cases} (y_2 - y_1) / (x_2 - x_1) & \text{If } P_1 \neq P_2 \\ (3x_1^2 + a) / (2y_1) & \text{If } P_1 = P_2 \end{cases}$$



Modern Cryptography Magic Workshop

Elliptic curves cryptography basics

Elliptic curve equation

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

Point addition operation

$$P + R = Q \quad Q - R = P$$

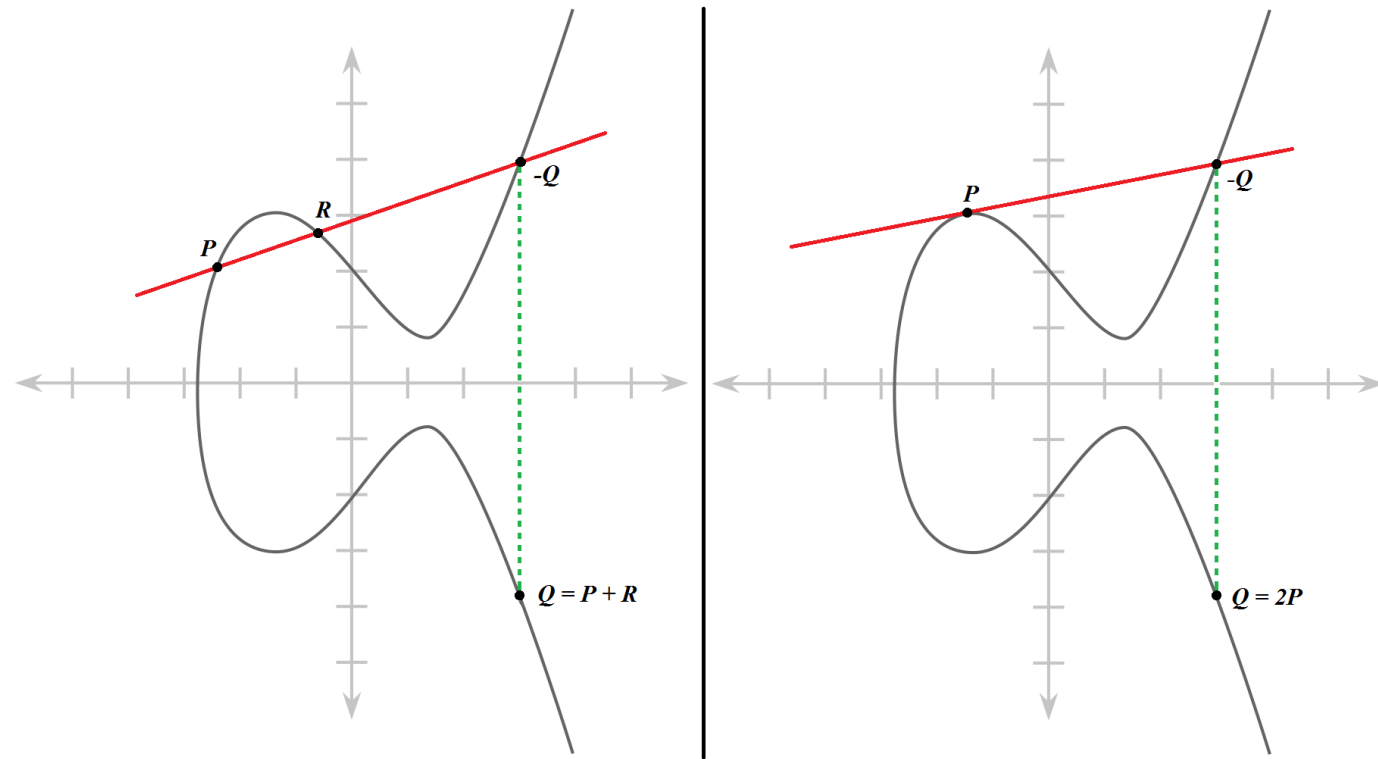
$$A + (B + C) = (A + B) + C$$

$$P + O = O + P = P$$

$$P + P = 2P$$

$$P + P + P = 3P$$

$$P + P + P + P = 4P$$



Modern Cryptography Magic Workshop

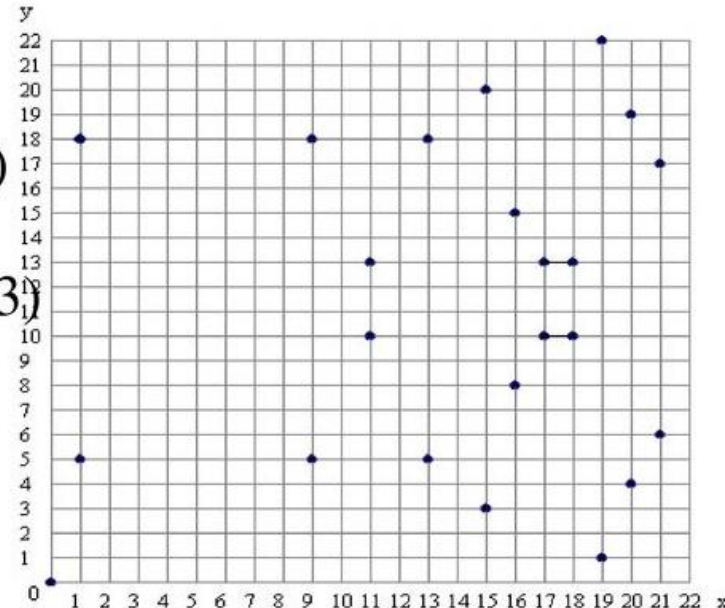
Elliptic curves cryptography basics

Set $p = 23$, $y^2 \bmod p = x^3 + 1x + 0 \bmod p$.

Choose $G = (9,5)$ (on curve: $25 \bmod 23 = 729 + 9 \bmod 23$)

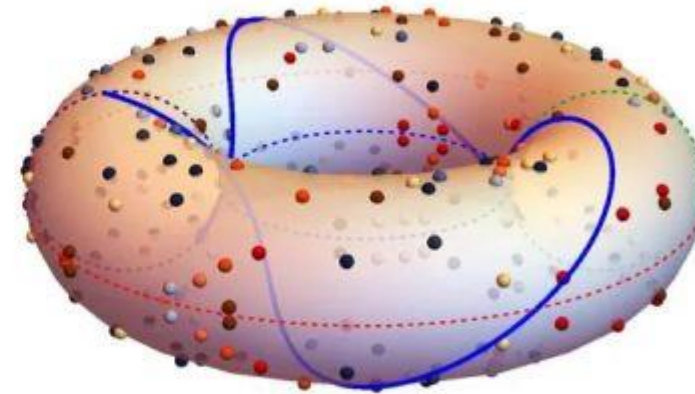
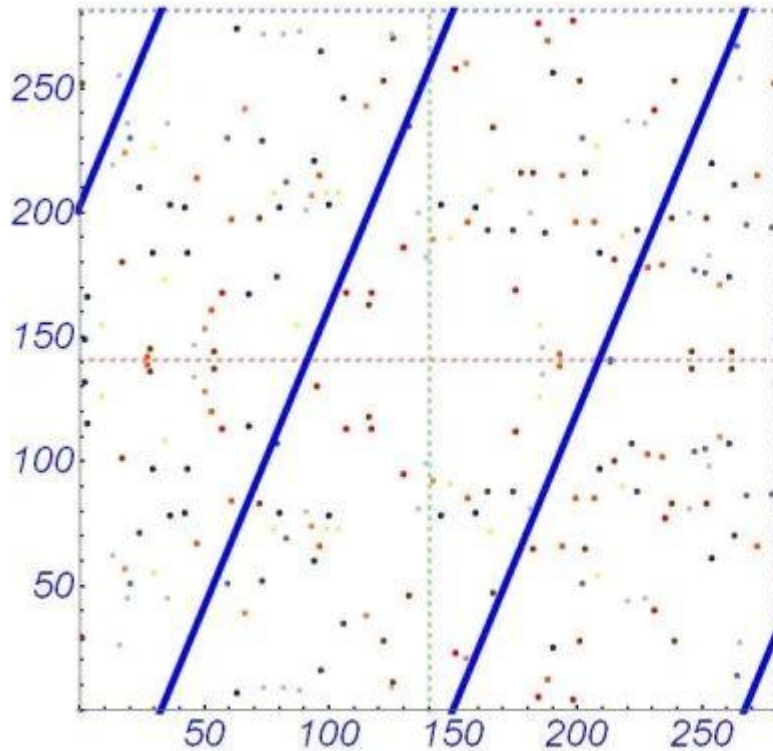
The 23 points on this curve:

$(0,0)$ $(1,5)$ $(1,18)$ $(9,5)$ $(9,18)$
 $(11,10)$ $(11,13)$ $(13,5)$ $(13,18)$
 $(15,3)$ $(15,20)$ $(16,8)$ $(16,15)$
 $(17,10)$ $(17,13)$ $(18,10)$ $(18,13)$
 $(19,1)$ $(19,22)$ $(20,4)$ $(20,19)$
 $(21,6)$ $(21,17)$



Modern Cryptography Magic Workshop

Elliptic curves cryptography basics



Modern Cryptography Magic Workshop

Elliptic curves cryptography basics

The magic here is that any point addition to itself n times is a one-way function

$$A = \alpha * G$$

Meaning that computing it forward is simple but computing it backward is hard

Modern Cryptography Magic Workshop

Elliptic curves cryptography basics

The magic here is that any point addition to itself n times is a one-way function

$$A = \alpha * G$$

Meaning that computing it forward is simple but computing it backward is hard

Imagine that α equals 477 and to compute $A = 477 * G$ we can add point G to itself 477 times

Modern Cryptography Magic Workshop

Elliptic curves cryptography basics

The magic here is that any point addition to itself n times is a one-way function

$$A = \alpha * G$$

Meaning that computing it forward is simple but computing it backward is hard

Imagine that α equals 477 and to compute $A = 477 * G$ we can add point G to itself 477 times

Or, knowing that the binary representation of 477 is

0 1 1 1 0 1 1 1 0 1

Modern Cryptography Magic Workshop

Elliptic curves cryptography basics

The magic here is that any point addition to itself n times is a one-way function

$$A = \alpha * G$$

Meaning that computing it forward is simple but computing it backward is hard

Imagine that α equals 477 and to compute $A = 477 * G$ we can add point G to itself 477 times

Or, knowing that the binary representation of 477 is

0 1 1 1 0 1 1 1 0 1

We can just compute this

$$0 * 512G + 1 * 256G + 1 * 128G + 1 * 64G + 0 * 32G + 1 * 16G + 1 * 8G + 1 * 4G + 0 * 2G + 1 * G$$

Modern Cryptography Magic Workshop

Elliptic curves cryptography basics

The magic here is that any point addition to itself n times is a one-way function

$$A = \alpha * G$$

Meaning that computing it forward is simple but computing it backward is hard

Imagine that α equals 477 and to compute $A = 477 * G$ we can add point G to itself 477 times

Or, knowing that the binary representation of 477 is

0 **1** **1** **1** 0 **1** **1** **1** 0 **1**

We can just compute this

$$0 * 512G + \mathbf{1} * 256G + \mathbf{1} * 128G + \mathbf{1} * 64G + 0 * 32G + \mathbf{1} * 16G + \mathbf{1} * 8G + \mathbf{1} * 4G + 0 * 2G + \mathbf{1} * G$$

Modern Cryptography Magic Workshop

Elliptic curves cryptography basics

D.1.2.3 Curve P-256

$$E : y^2 \equiv x^3 - 3x + b \pmod{p}$$

$p =$ 1157920892103562487626974469494075735300861434152903141955
33631308867097853951

$n =$ 1157920892103562487626974469494075735299969552241357603424
22259061068512044369

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

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

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

Modern Cryptography Magic Workshop

Elliptic curves cryptography basics

Symmetric Key Size (bits)	RSA and Diffie-Hellman Key Size (bits)	Elliptic Curve Key Size (bits)
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	521

Modern Cryptography Magic Workshop

Schnorr signature algorithm (1989)

Elliptic curve: $y^2 = x^3 + ax + b \bmod p$

Public parameters: a, b, p, G

Modern Cryptography Magic Workshop

Schnorr signature algorithm (1989)

Elliptic curve: $y^2 = x^3 + ax + b \bmod p$

Public parameters: a, b, p, G

Alice: secret key α \rightarrow public key $A = \alpha * G$

Bob: secret key β \rightarrow public key $B = \beta * G$

Modern Cryptography Magic Workshop

Schnorr signature algorithm (1989)

Elliptic curve: $y^2 = x^3 + ax + b \bmod p$

Public parameters: a, b, p, G

Alice: secret key α \rightarrow public key $A = \alpha * G$

Bob: secret key β \rightarrow public key $B = \beta * G$

One-way function because of
discrete logarithm problem



Modern Cryptography Magic Workshop

Schnorr signature algorithm (1989)

Elliptic curve: $y^2 = x^3 + ax + b \bmod p$

Public parameters: a, b, p, G

Alice: secret key α \rightarrow public key $A = \alpha * G$

Bob: secret key β \rightarrow public key $B = \beta * G$

Signing: $s = k + \alpha * \text{hash}(m, R)$ ($R = k * G$ - random point)

Signature: (s, R)

Modern Cryptography Magic Workshop

Schnorr signature algorithm (1989)

Elliptic curve: $y^2 = x^3 + ax + b \bmod p$

Public parameters: a, b, p, G

Alice: secret key α \rightarrow public key $A = \alpha * G$

Bob: secret key β \rightarrow public key $B = \beta * G$

Signing: $s = k + \alpha * \text{hash}(m, R)$ ($R = k * G$ - random point)

Signature: (s, R)

Verify: $s * G$

Modern Cryptography Magic Workshop

Schnorr signature algorithm (1989)

Elliptic curve: $y^2 = x^3 + ax + b \bmod p$

Public parameters: a, b, p, G

Alice: secret key α \rightarrow public key $A = \alpha * G$

Bob: secret key β \rightarrow public key $B = \beta * G$

Signing: $s = k + \alpha * \text{hash}(m, R)$ ($R = k * G$ - random point)

Signature: (s, R)

Verify: $s * G = k * G + \alpha * \text{hash}(m, R) * G$

Modern Cryptography Magic Workshop

Schnorr signature algorithm (1989)

Elliptic curve: $y^2 = x^3 + ax + b \bmod p$

Public parameters: a, b, p, G

Alice: secret key α \rightarrow public key $A = \alpha * G$

Bob: secret key β \rightarrow public key $B = \beta * G$


Signing: $s = k + \alpha * \text{hash}(m, R)$ ($R = k * G$ - random point)


Signature: (s, R)

Verify: $s * G = R + A * \text{hash}(m, R)$

Modern Cryptography Magic Workshop

Schnorr signature algorithm (1989)






Method for identifying subscribers and for generating and verifying electronic signatures in a data exchange system

Abstract

In a data exchange system working with processor chip cards, a chip card transmits coded identification data l , v and, proceeding from a random, discrete logarithm r , an exponential value $x=2^r \pmod{p}$ to the subscriber who, in turn, generates and transmits a random bit sequence e to the chip card. By multiplication of a stored, private key s with the bit sequence e and by addition of the random number r , the chip card calculates a y value and transmits the y value to the subscriber who, in turn, calculates an x value from the information y , v , and e and checks whether the calculated x value coincides with the transmitted x value. For an electronic signature, a hash value e is first calculated from an x value and from the message m to be signed and a y value is subsequently calculated from the information r , s , and e . The numbers x and y then yield the electronic signature of the message m .

Images (3)



Classifications

G07F7/1008 Active credit-cards provided with means to personalise their use, e.g. with PIN-introduction/comparison system

[View 8 more classifications](#)

US4995082A
United States

[Download PDF](#) [Find Prior Art](#) [Similar](#)

Inventor: [Claus P. Schnorr](#)

Current Assignee : [PUBLIC KEY PARTNERS](#)

Worldwide applications
1989 • [EP](#) 1990 • [DE](#) [AT](#) [ES](#) [JP](#) [US](#)

Application US07/484,127 events

- 1989-02-24 • Priority to EP89103290A
- 1989-02-24 • Priority to EP89103290.6
- 1990-02-23 • Application filed by Schnorr Claus P
- 1991-02-19 • Application granted
- 1991-02-19 • Publication of US4995082A
- 1993-09-09 • Assigned to PUBLIC KEY PARTNERS
- 1996-09-25 • First worldwide family litigation filed
- 2010-02-23 • Anticipated expiration
- 2019-08-13 • Application status is Expired - Lifetime**

Info: [Patent citations \(9\)](#), [Non-patent citations \(4\)](#), [Cited by \(195\)](#), [Legal events](#), [Similar documents](#), [Priority and Related](#)


Modern Cryptography Magic Workshop

Boneh–Lynn–Shacham (BLS) signature scheme uses a bilinear pairing for verification

And signatures are elements of an elliptic curve group

Asymmetric pairings $e: G_1 \times G_2 \rightarrow G_T$

Non-supersingular curves:



The diagram illustrates the construction of asymmetric pairings on non-supersingular elliptic curves. It shows a grid of points representing the q -torsion of the extension field, $E(\mathbb{F}_{p^\alpha})[q]$. A vertical orange bar represents the group G_2 , and a horizontal orange bar represents the group G_1 . Points P and Q are marked on the horizontal bar. The horizontal bar is labeled $E(\mathbb{F}_p)$ and the vertical bar is labeled G_2 . A bracket on the right indicates the mapping to $E(\mathbb{F}_{p^\alpha})[q]$. A man is pointing at the diagram.

No known DDH algorithm in G_1 or G_2

SXDH assumption: DDH hard in G_1 and G_2

- Used for anonymous IBE, circular insecure enc., ...

[D'10] [ABBC'10, CGH'12]

Trace(G_2)

Modern Cryptography Magic Workshop

Boneh–Lynn–Shacham (BLS) signature scheme uses a bilinear pairing for verification

And signatures are elements of an elliptic curve group

Pairing function:

$$e(\alpha * P, Q) = e(P, Q)^\alpha = e(P, \alpha * Q)$$
$$e(\alpha * P, \beta * Q) = e(P, Q)^{\alpha\beta} = e(\beta * P, \alpha * Q)$$

Modern Cryptography Magic Workshop

Boneh–Lynn–Shacham (BLS) signature scheme uses a bilinear pairing for verification

And signatures are elements of an elliptic curve group

Pairing function: $e(\alpha * P, Q) = e(P, Q)^\alpha = e(P, \alpha * Q)$

$$e(\alpha * P, \beta * Q) = e(P, Q)^{\alpha\beta} = e(\beta * P, \alpha * Q)$$

$$e(3 * P, Q) = e(P + P + P, Q) = e(P, Q) * e(P, Q) * e(P, Q) = e(P, Q)^3$$

$$e(P, Q)^3 = e(P, Q) * e(P, Q) * e(P, Q) = e(P, Q + Q + Q) = e(P, 3 * Q)$$

Modern Cryptography Magic Workshop

Boneh–Lynn–Shacham (BLS) signature scheme uses a bilinear pairing for verification

And signatures are elements of an elliptic curve group

Pairing function: $e(\alpha * P, Q) = e(P, Q)^\alpha = e(P, \alpha * Q)$

$$e(\alpha * P, \beta * Q) = e(P, Q)^{\alpha\beta} = e(\beta * P, \alpha * Q)$$

$$e(3 * P, Q) = e(P + P + P, Q) = e(P, Q) * e(P, Q) * e(P, Q) = e(P, Q)^3$$

$$e(P, Q)^3 = e(P, Q) * e(P, Q) * e(P, Q) = e(P, Q + Q + Q) = e(P, 3 * Q)$$

That's why it is a bilinear mapping – we have two different ways to get same result $e(P, Q)^\alpha$

One way is $e(\alpha * P, Q)$

Another way is $e(P, \alpha * Q)$

Modern Cryptography Magic Workshop

Boneh–Lynn–Shacham (BLS) signature scheme uses a bilinear pairing for verification

And signatures are elements of an elliptic curve group

Pairing function: $e(\alpha * P, Q) = e(P, Q)^\alpha = e(P, \alpha * Q)$

$$e(\alpha * P, \beta * Q) = e(P, Q)^{\alpha\beta} = e(\beta * P, \alpha * Q)$$

Key generation: private key is a random pk \rightarrow public key $P = pk * G$

Signing: $S = pk * H(m)$ (just pk times hash of a message)

Signature: S (just a single point on a curve, no additional randomness!)

Verify: $e(P, H(m)) = e(G, S)$

Modern Cryptography Magic Workshop

Boneh–Lynn–Shacham (BLS) signature scheme uses a bilinear pairing for verification

And signatures are elements of an elliptic curve group

Pairing function: $e(\alpha * P, Q) = e(P, Q)^\alpha = e(P, \alpha * Q)$

$$e(\alpha * P, \beta * Q) = e(P, Q)^{\alpha\beta} = e(\beta * P, \alpha * Q)$$

Key generation: private key is a random pk \rightarrow public key $P = pk * G$

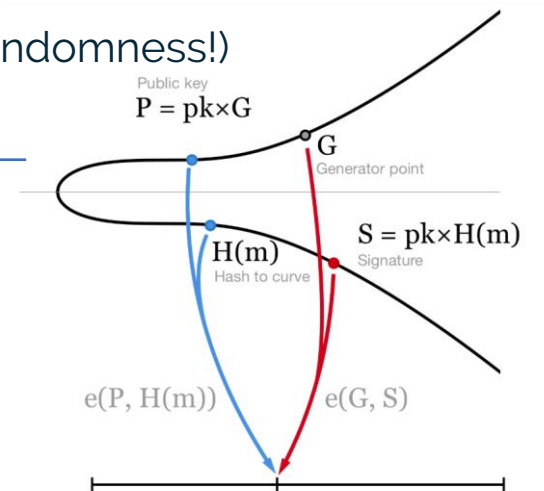
Signing: $S = pk * H(m)$ (just pk times hash of a message)

Signature: S (just a single point on a curve, no additional randomness!)

Verify: $e(P, H(m)) = e(G, S)$

$$\begin{aligned} &\Downarrow & \Downarrow \\ &e(G, H(m))^{pk} \end{aligned}$$

We just need to check that the public key and the message hash are mapped to the same number as the curve generator point and the signature



Modern Cryptography Magic Workshop

Boneh–Lynn–Shacham (BLS) signature scheme uses a bilinear pairing for verification

And signatures are elements of an elliptic curve group

Pairing function: $e(\alpha * P, Q) = e(P, Q)^\alpha = e(P, \alpha * Q)$

$$e(\alpha * P, \beta * Q) = e(P, Q)^{\alpha\beta} = e(\beta * P, \alpha * Q)$$

Key generation: private key is a random pk \rightarrow public key $P = pk * G$

Signing: $S = pk * H(m)$ (just pk times hash of a message)

Signature: S (just a single point on a curve, no additional randomness!)

Verify: $e(P, H(m)) = e(G, S)$

More info:

Modern Cryptography Magic Workshop

Boneh–Lynn–Shacham (BLS) signature scheme uses a bilinear pairing for verification

And signatures are elements of an elliptic curve group

Key aggregation and n-of-n multisignature

If we are using multisignature addresses, we are signing *the same transaction* with different keys. In this case we can do key aggregation like in Schnorr, where we combine all signatures and all keys to a single pair of a key and a signature. Let's take a common 3-of-3 multisig scheme (it can be done similarly for any number of signers).

A simple way to combine them is to add all the signatures and all the keys together. The result will be a signature $S=S1+S2+S3$ and a key $P=P1+P2+P3$. It's easy to see that the same verification equation still works:

$$e(G, S) = e(P, H(m))$$

$$e(G, S) = e(G, S1+S2+S3) = e(G, (pk1+pk2+pk3) \times H(m)) = e((pk1+pk2+pk3) \times G, H(m)) = e(P1+P2+P3, H(m)) = e(P, H(m))$$

Modern Cryptography Magic Workshop

NODR creates an open marketplace where millions of people around the world can lease their idle bandwidth to thousands of video streaming services and receive instant payment in cryptocurrency



Ever wondered?

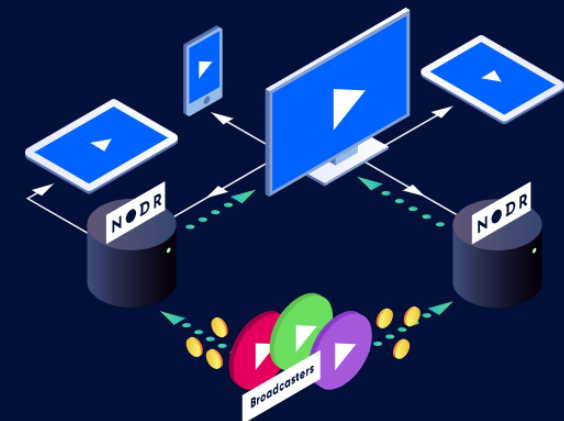
Only **10%** of the Internet bandwidth that you buy every month for your home or office is used by your device. The rest stays idle while you rest, travel, or spend time with your friends and family.

There are thousands of premium online video services that are dreaming to utilize this bandwidth.

Did you know?

Every time you press “play” on a video streaming platform, a group of servers called a CDN delivers the video to your device. The broadcaster pays the CDN for every megabyte of video that your device receives.

Roughly \$8 billion has been spent on CDNs in 2017. This number will increase to \$30 billion in just five years!



Modern Cryptography Magic Workshop

NODR crypto-protocol (BLS-based)

Tracker:	secret key is a random τ	\rightarrow	public key $T = \tau * G$
Alice (viewer):	secret key is a random α	\rightarrow	public key $A = \alpha * G$
Bob (distributing node):	secret key is a random β	\rightarrow	public key $B = \beta * G$

Original video is split into segments:

Segment #1	hash of a file f_1	\rightarrow	hash to the point $F_1 = f_1 * G$
Segment #2	hash of a file f_2	\rightarrow	hash to the point $F_2 = f_2 * G$
.....			
Segment #n	hash of a file f_n	\rightarrow	hash to the point $F_n = f_n * G$

Modern Cryptography Magic Workshop

NODR crypto-protocol (BLS-based)

Segment key generation

Tracker assign each viewer with a sequence of video segments

by generating a set of **segment keys** which is in fact just

a BLS signature of a point that link viewers public key and a hash of a file

Alice public key bind to segment #1 $(A + F_1) \rightarrow$ segment key $K^A_1 = \tau * (A + F_1)$

Alice public key bind to segment #3 $(A + F_3) \rightarrow$ segment key $K^A_3 = \tau * (A + F_3)$

Alice public key bind to segment #8 $(A + F_8) \rightarrow$ segment key $K^A_8 = \tau * (A + F_8)$

Alice public key bind to segment #9 $(A + F_9) \rightarrow$ segment key $K^A_9 = \tau * (A + F_9)$

Modern Cryptography Magic Workshop

NODR crypto-protocol (BLS-based)

Segment key generation

Tracker assign each viewer with a sequence of video segments

by generating a set of **segment keys** which is in fact just


a BLS signature of a point that link viewers public key and a hash of a file

Alice public key bind to segment #1 $(A + F_1) \rightarrow$ segment key $K^A_1 = \tau * (A + F_1)$

Alice public key bind to segment #3 $(A + F_3) \rightarrow$ segment key $K^A_3 = \tau * (A + F_3)$

Alice public key bind to segment #8 $(A + F_8) \rightarrow$ segment key $K^A_8 = \tau * (A + F_8)$

Alice public key bind to segment #9 $(A + F_9) \rightarrow$ segment key $K^A_9 = \tau * (A + F_9)$



*Only Tracker can produce such segment keys
because of secret key τ
Therefore, these keys can be considered as
authorization from Tracker to Alice*

Modern Cryptography Magic Workshop

NODR crypto-protocol (BLS-based)

Segment key generation

Tracker assign each viewer with a sequence of video segments

by generating a set of **segment keys** which is in fact just

a BLS signature of a point that link viewers public key and a hash of a file

Alice public key bind to segment #1 $(A + F_1) \rightarrow$ segment key $K^A_1 = \tau * (A + F_1)$

Alice public key bind to segment #3 $(A + F_3) \rightarrow$ segment key $K^A_3 = \tau * (A + F_3)$

Alice public key bind to segment #8 $(A + F_8) \rightarrow$ segment key $K^A_8 = \tau * (A + F_8)$

Alice public key bind to segment #9 $(A + F_9) \rightarrow$ segment key $K^A_9 = \tau * (A + F_9)$

Each actor in the system **must** access to the files only by this **segment keys**

It mean that if distributing node Bob store some files, he was also received

authorization from Tracker to download this files earlier by segment keys $K^B_i = \tau * (B + Fi)$

Modern Cryptography Magic Workshop

NODR crypto-protocol (BLS-based)

Signal messaging in p2p-network

Bob send **have** messages to Alice that contains information about all files that Bob store, for example:

$$\begin{aligned} S_2 &= \beta * (A + F_2) & S_3 &= \beta * (A + F_3) & S_4 &= \beta * (A + F_4) \\ S_5 &= \beta * (A + F_5) & S_7 &= \beta * (A + F_7) & S_8 &= \beta * (A + F_8) \end{aligned}$$

Alice have segment keys received from Tracker:

$$\begin{aligned} K^A_1 &= \tau * (A + F_1) & K^A_3 &= \tau * (A + F_3) & K^A_8 &= \tau * (A + F_8) \\ K^A_9 &= \tau * (A + F_9) \end{aligned}$$

Searching for this match on Alice's side is done by two pairing for each segment key:

$$\begin{aligned} \text{if } e(B, K^A_3) &= e(T, S_3) \rightarrow e(\beta * G, \tau * (A + F_3)) = e(\tau * G, \beta * (A + F_3)) & \text{than Bob have file for } K^A_3 \\ \text{if } e(B, K^A_8) &= e(T, S_8) \rightarrow e(\beta * G, \tau * (A + F_8)) = e(\tau * G, \beta * (A + F_8)) & \text{than Bob have file for } K^A_8 \end{aligned}$$

Modern Cryptography Magic Workshop

NODR crypto-protocol (BLS-based)

Signal messaging in p2p-network

Bob send **have** messages to Alice that contains information about all files that Bob store, for example:

$$\begin{aligned} S_2 &= \beta * (A + F_2) & S_3 &= \beta * (A + F_3) & S_4 &= \beta * (A + F_4) \\ S_5 &= \beta * (A + F_5) & S_7 &= \beta * (A + F_7) & S_8 &= \beta * (A + F_8) \end{aligned}$$

Alice have segment keys received from Tracker:

$$\begin{aligned} K^A_1 &= \tau * (A + F_1) & K^A_3 &= \tau * (A + F_3) & K^A_8 &= \tau * (A + F_8) \\ K^A_9 &= \tau * (A + F_9) \end{aligned}$$

Searching for this match on Alice's side is done by two pairing for each segment key:

$$\begin{aligned} \text{if } e(B, K^A_3) &= e(T, S_3) \rightarrow e(\beta * G, \tau * (A + F_3)) = e(\tau * G, \beta * (A + F_3)) \text{ then Bob have file for } K^A_3 \\ \text{if } e(B, K^A_8) &= e(T, S_8) \rightarrow e(\beta * G, \tau * (A + F_8)) = e(\tau * G, \beta * (A + F_8)) \text{ then Bob have file for } K^A_8 \end{aligned}$$

*BLS allows to check does two signatures from two actors created for the same message or not **without knowing that message!***

Modern Cryptography Magic Workshop

NODR crypto-protocol (BLS-based)

Downloading files in p2p-network

When Alice receive file F_3 from Bob she checks correctness of received file by:

(1) calculate hash of a file content

$$f_3$$

(2) calculate hash of a file to the point of a file

$$F_3 = f_3 * G$$

(3) add her own public key with point of a file

$$(A + F_3)$$

(4) check that this point match with segment key

$$e(G, K_3^A) = e(T, (A + F_3))$$

(5) sign and send delivery confirm to Bob

$$S'_3 = \alpha * (B + F_3)$$

After receiving and checking delivery confirm for file F_3 (checking Alice's signature S'_3) Bob starts transferring the next file F_8 (which checks same way at the end of the transfer)

Modern Cryptography Magic Workshop

NODR crypto-protocol (BLS-based)

Creating crypto-report

To create on Bob's side a report that proves Alice download file F_3 from Bob, and both was authorized by Tracker we need next signatures from all three actors:

$$\text{from Tracker} \quad K^A_3 = \tau * (A + F_3)$$

$$\text{from Tracker} \quad K^B_3 = \tau * (B + F_3)$$

$$\text{from Alice} \quad S'_3 = \alpha * (B + F_3)$$

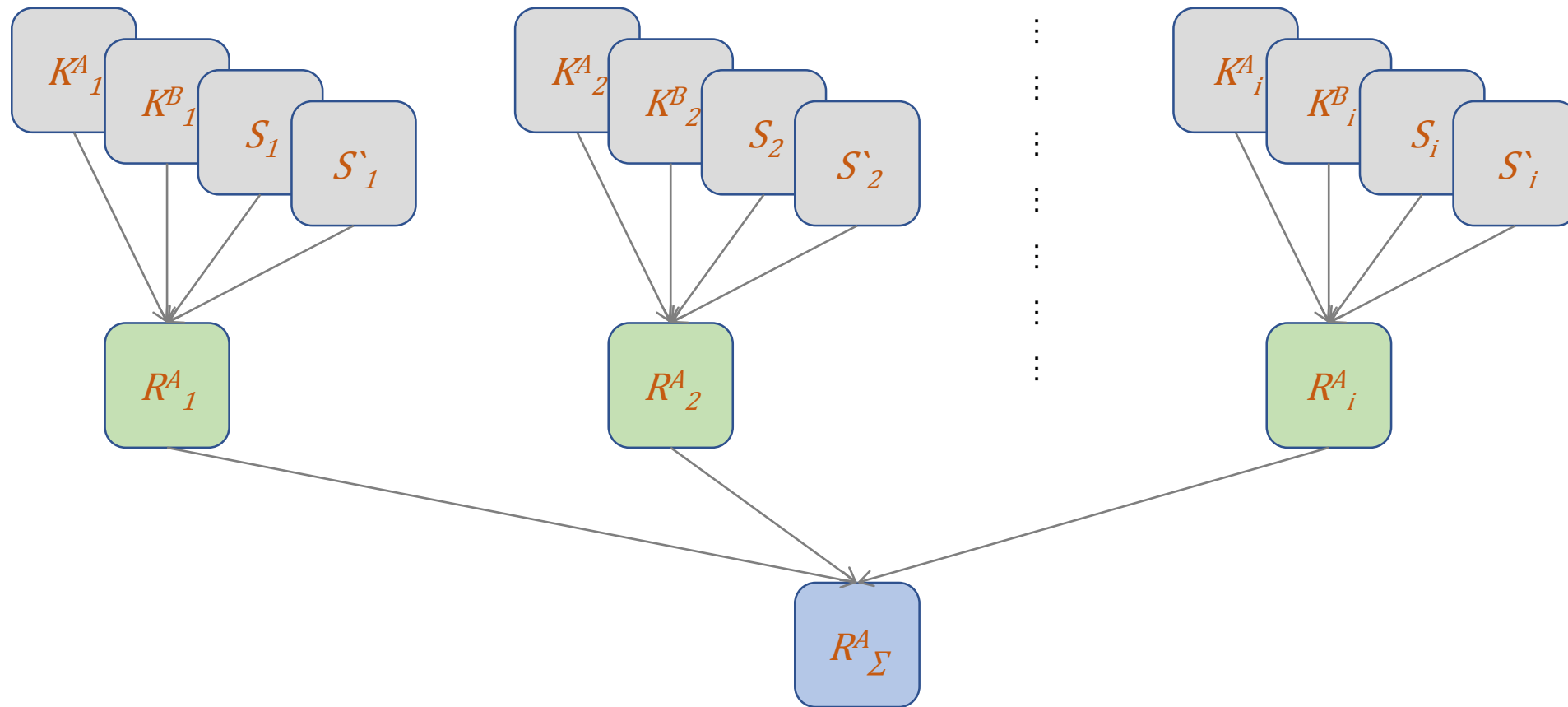
$$\text{from Bob} \quad S_3 = \beta * (A + F_3)$$

Once Bob have all four needed parts, he can produce aggregated signature by just combine all four signatures in one signature:

$$R^A_3 = K^A_3 + K^B_3 + S'_3 + S_3$$

Modern Cryptography Magic Workshop

NODR crypto-protocol (BLS-based)



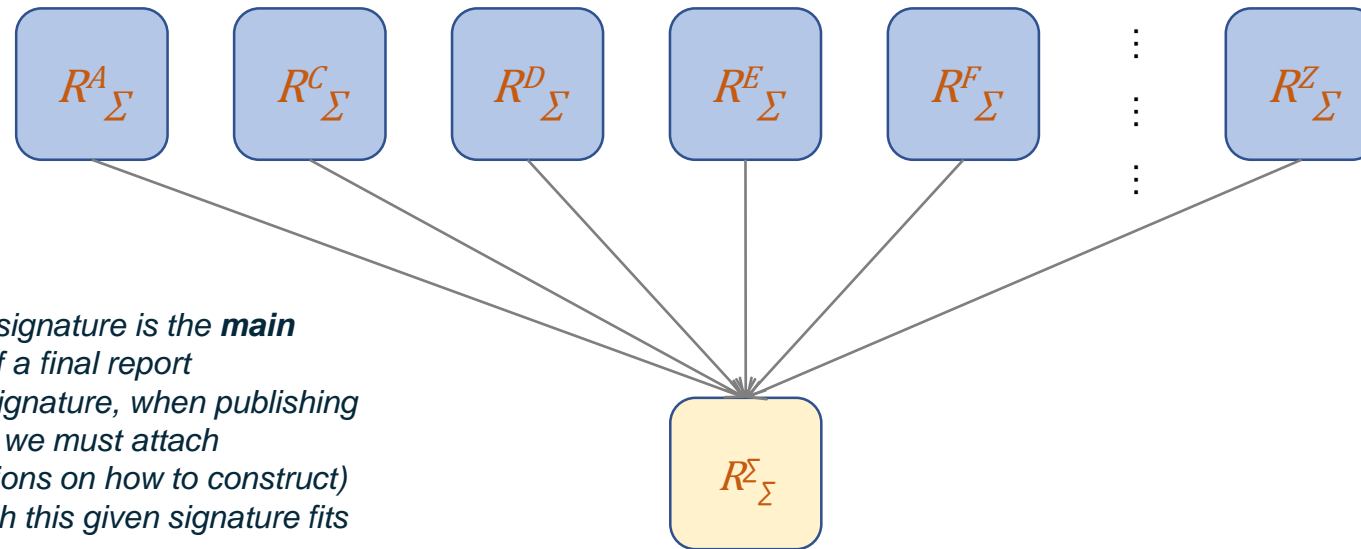
Modern Cryptography Magic Workshop

NODR crypto-protocol (BLS-based)

Creating crypto-report (**N-viewers extension**)

But what if Bob distribute files not just to Alice but to several other viewers as well?

Once again, the solution is signature aggregation – **N-viewers** aggregation report



*Note that this final signature is the **main** but just **one part** of a final report
In addition to this signature, when publishing final report to DLT, we must attach
(or give an instructions on how to construct)
a “**message**” which this given signature fits*

Modern Cryptography Magic Workshop

NODR crypto-protocol (BLS-based)

Once Bob creates **One-viewers** aggregation signature and decide to get his token reward there need to be an instruction for validators on how to construct a message that fits given signature

First lets remember how pairing and BLS signature verification works

Pairing function:

$$e(\alpha * P, Q) = e(P, Q)^\alpha = e(P, \alpha * Q)$$

$$e(3 * P, Q) = e(P + P + P, Q) = e(P, Q) * e(P, Q) * e(P, Q) = e(P, Q)^3$$

$$e(P, Q)^3 = e(P, Q) * e(P, Q) * e(P, Q) = e(P, Q + Q + Q) = e(P, 3 * Q)$$

Verification:

$$e(G, S) = e(P, H(m))$$

"Some" message

Verification in our case:

$$e(G, R_\Sigma^A) = e(P, H(m))$$

Aggregated signature that Bob produced

"Some" public key

Modern Cryptography Magic Workshop

NODR crypto-protocol (BLS-based)

Once Bob creates **One-viewers** aggregation signature and decide to get his token reward there need to be an instruction for validators on how to construct a message that fits given signature

Now lets construct verification equation for our case:

$$e(G, R^A_\Sigma) = e(G, \tau * \Sigma(A + F_i) + \tau * \Sigma(B + F_i) + \alpha * \Sigma(B + F_i) + \beta * \Sigma(A + F_i))$$

$$e(G, R^A_\Sigma) = e(G, (\tau + \beta) * \Sigma(A + F_i) + (\tau + \alpha) * \Sigma(B + F_i))$$

$$e(G, R^A_\Sigma) = e(G, (\tau + \beta) * \Sigma(A + F_i)) * e(G, (\tau + \alpha) * \Sigma(B + F_i))$$

$$e(G, R^A_\Sigma) = e((\tau + \beta) * G, \Sigma(A + F_i)) * e((\tau + \alpha) * G, \Sigma(B + F_i))$$

$$e(G, R^A_\Sigma) = e((T + B), \Sigma(A + F_i)) * e((T + A), \Sigma(B + F_i))$$

Modern Cryptography Magic Workshop

NODR crypto-protocol (BLS-based)

Once Bob creates **One-viewers** aggregation signature and decide to get his token reward there need to be an instruction for validators on how to construct a message that fits given signature

Finally this is our verification equation:

$$e(G, R_{\Sigma}^A) = e((T+B), \Sigma(A+F_i)) * e((T+A), \Sigma(B+F_i))$$

On the one hand verifier must pair point G and Bob's aggregated signature R_{Σ}^A

On the another hand verifier must calculate two pairings:

- (1) sum of a public key's $(T+B)$ with summary point $\Sigma(A+F_i)$
- (2) sum of a public key's $(T+A)$ with summary point $\Sigma(B+F_i)$

then multiply them, and if this two results are equal, then verifier convinced that job of transfer files F_i from Bob to Alice has been successfully completed and therefore must be paid



Perm Winter School 2020

Modern Cryptography Magic Workshop

Exercise

Modern Cryptography Magic Workshop

Exercise

How to prevent double reward?

Modern Cryptography Magic Workshop

Exercise

How to prevent double reward?

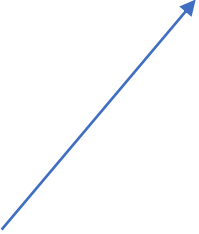
The idea is to save "some information" to the DLT as cheap as it possible (in terms of fees and throughput)
So that later validators can check that "information" and decide
Whether distributing node reports a job that was ordered by Tracker or not and pay for such job only once

Modern Cryptography Magic Workshop

Exercise

How to prevent double reward?

The idea is to save "some information" to the DLT as cheap as it possible (in terms of fees and throughput)
So that later validators can check that "information" and decide
Whether distributing node reports a job that was ordered by Tracker or not and pay for such job only once



*Sounds like unspent
transaction in Bitcoin*

Modern Cryptography Magic Workshop

Exercise

How to prevent double reward?

Imagine that Alice willing to download files with segment keys once received from Tracker:

$$K^A_1 = \tau * (A + F_1)$$

$$K^A_2 = \tau * (A + F_2)$$

... ..

$$K^A_n = \tau * (A + F_n)$$

The BLS magic allows to combine all this signatures in one signature:

$$K^A_\Sigma = \tau * \sum (A + F_i)$$

So this is just a one random looking curve point published to DLT that hides all Alice's segment keys (lets call it **grain**):

$$K^A_\Sigma = K^A_1 + K^A_2 + K^A_3 + K^A_4 + K^A_5 + K^A_6 + \dots + K^A_{n-1} + K^A_n$$

Modern Cryptography Magic Workshop

Exercise

How to prevent double reward?

Imagine that Alice willing to download files with segment keys once received from Tracker:

$$K^A_1 = \tau * (A + F_1)$$

$$K^A_2 = \tau * (A + F_2)$$

... ..

$$K^A_n = \tau * (A + F_n)$$

The BLS magic allows to combine all this signatures in one signature:

$$K^A_\Sigma = \tau * \sum (A + F_i)$$

So this is just a one random looking curve point published to DLT that hides all Alice's segment keys (lets call it **grain**):

$$K^A_\Sigma = K^A_1 + K^A_2 + K^A_3 + K^A_4 + K^A_5 + K^A_6 + \dots + K^A_{n-1} + K^A_n$$

You can think of this as an accumulator – one element that “contains” many points that we can check whether particular point is in the set or not

Modern Cryptography Magic Workshop

Exercise

How to prevent double reward?

Now lets imagine case when one node (say Bob) serve Alice *j* part of files and another node (say Bill) serve Alice *k* part of files, we can represent such case with the following decomposition:

$$K_{\Sigma}^A = \sum K_j^A + \sum K_k^A$$

And now, once all initial segment keys are available to Alice (from Tracker) and to Bob (from get messages from Alice) and to validator (from Bob's claiming transaction) each of this actors can check:

- whether particular **segment key** is a part of a particular **grain**?
- does this particular **grain** was spent?
- and if yes, does it was spent **fully** or **partially**?

Modern Cryptography Magic Workshop

Exercise

$$K^A_{\Sigma} = \sum K^A_j + \sum K^A_k + \sum K^A_l$$

