Σ-protocols

- and why they should matter to every Bitcoin thinker

Adam Gibson

3rd March 2022

Advancing Bitcoin

Outline

Outline

- **Motivation** Σ-protocols in the wild

Outline

- **Motivation** Σ-protocols in the wild
- **ZK** Analogies; zero knowledge

Outline

- **Motivation** Σ-protocols in the wild
- **ZK** Analogies; zero knowledge
- **Homomorphism** the special property needed for Σ-protocols

Outline

- **Motivation** Σ-protocols in the wild
- **ZK** Analogies; zero knowledge
- **Homomorphism** the special property needed for Σ-protocols
- **The most fundamental Σ-protocol**

Outline

- **Motivation** Σ-protocols in the wild
- **ZK** Analogies; zero knowledge
- **Homomorphism** the special property needed for Σ-protocols
- **The most fundamental Σ-protocol**
- **The Fiat-Shamir transform**

Outline

- **Motivation** Σ-protocols in the wild
- **ZK** Analogies; zero knowledge
- **Homomorphism** the special property needed for Σ-protocols
- **The most fundamental Σ-protocol**
- **The Fiat-Shamir transform**
- **Combinations of Σ-protocols - AND, OR, ...**

Sigma protocols in the wild

Σ-protocols in the wild - 0

Let `M = rT` be the blinded token that C sends to S, let `(G,Y) = (G,xG)` be the commitment from above, and let H_3 be a new hash function (modelled as a random oracle for security purposes). In the protocol below, we can think of S playing the role of the 'prover' and C the 'verifier' in a traditional NIZK proof system.

- S computes `Z = xM`, as before.
- S also samples a random nonce `k ← ZZ_q` and commits to the nonce by calculating `A = kG` and `B = kM`
- S constructs a challenge `c ← H_3(G,Y,M,Z,A,B)` and computes `s = k-cx (mod q)`
- S sends `(c,s)` to the user C
- C recalculates `A' = sG + cY` and `B' = s*M + c*Z` and hashes `c' = H_3(G,Y,M,Z,A',B')`.
- C verifies that `c' =?= c`.

Note that correctness follows since

```
A' = sG + cY = (k-cx)G + cxG = kG and B' = sM + cZ = r(k-cx)T + crxT = krT = kM
```

We write DLEQ(Z/M == Y/G) to denote the proof that is created by S and validated by C.

Here are some examples of things that are based on the Σ-protocol:

Σ-protocols in the wild - 1

Here are some examples of things that are based on the Σ-protocol:

BIP340 .. edDSA .. ECDSA (kinda)

Σ-protocols in the wild - 1

Here are some examples of things that are based on the Σ-protocol:

BIP340 .. edDSA .. ECDSA (kinda)

Anonymous credentials (used in e.g. Brave, Wabisabi, Signal)

"DLEQ"s - Privacy pass, Joinmarket, ECDSA signature adaptors

"DLEQ"s - Privacy pass, Joinmarket, ECDSA signature adaptors

(Schnorr) blind sigs; Chaumian tokens (see also Brands, see Fedimint, OT etc.)

Σ-protocols in the wild - 2

"DLEQ"s - Privacy pass, Joinmarket, ECDSA signature adaptors

(Schnorr) blind sigs; Chaumian tokens (see also Brands, see Fedimint, OT etc.)

ring sigs (Monero e.g.), multisigs, threshold sigs.

Σ-protocols in the wild - 2

"DLEQ"s - Privacy pass, Joinmarket, ECDSA signature adaptors

(Schnorr) blind sigs; Chaumian tokens (see also Brands, see Fedimint, OT etc.)

ring sigs (Monero e.g.), multisigs, threshold sigs.

ZKP: Bulletproofs (extended); zkSNARKs? (not really)

"DLEQ"s - Privacy pass, Joinmarket, ECDSA signature adaptors
(Schnorr) blind sigs; Chaumian tokens (see also Brands, see Fedimint, OT etc.)
ring sigs (Monero e.g.), multisigs, threshold sigs.
ZKP: Bulletproofs (extended); zkSNARKs? (not really)
PAKE

Σ-protocols in the wild - 2

"DLEQ"s - Privacy pass, Joinmarket, ECDSA signature adaptors

(Schnorr) blind sigs; Chaumian tokens (see also Brands, see Fedimint, OT etc.)

ring sigs (Monero e.g.), multisigs, threshold sigs.

ZKP: Bulletproofs (extended); zkSNARKs? (not really)

PAKE

Core concepts:
Proving without revealing (ZKPs)

Zero knowledge, intuitively - 1

Start with a silly analogy:

Zero knowledge, intuitively - 1

Start with a silly analogy:

- Coke's secret recipe. You claim to know coke's secret recipe? ok, here's 10000 ingredients in a kitchen. I'll walk away for a day but keep you locked in the kitchen. Make me a glass of Coke.

- Claim: challenge: demonstrate - this is the Σ-protocol paradigm.

- Claim: challenge: demonstrate - this is the Σ-protocol paradigm.
- To win at this game, you have to *be ready to create a demonstration for any challenge*.

Zero knowledge, intuitively - 2

- Claim: challenge: demonstrate - this is the Σ-protocol paradigm.
- To win at this game, you have to *be ready to create a demonstration for any challenge*.
- But your demonstration shouldn't give away the secret sauce .. 🤔

Zero knowledge, intuitively - 3

Another intuition - coin flip over a telephone line.

- We assign 1BTC based on whether you succeed in 'calling' the coin flip.

Zero knowledge, intuitively - 3

Another intuition - coin flip over a telephone line.

- We assign 1BTC based on whether you succeed in 'calling' the coin flip.
- Since there's a big incentive to cheat, this wouldn't work over a telephone call, because the side who reveals their (choice or flip)*second* can always win.

Zero knowledge, intuitively - 3

Another intuition - coin flip over a telephone line.

- We assign 1BTC based on whether you succeed in 'calling' the coin flip.
- Since there's a big incentive to cheat, this wouldn't work over a telephone call, because the side who reveals their (choice or flip)*second* can always win.
- This example illustrates the idea of a **commitment** - hand fixes and hides the coin, that's a commitment.

Homomorphisms

Homomorphism

$\mathbb{G}_1 \Longrightarrow \mathbb{G}_2$

Homomorphism

$\mathbb{G}_1 \Longrightarrow \mathbb{G}_2$

Example: $f(x) = 2x$; suppose $\mathbb{G}_1 = (\mathbb{Z}, +)$.

What is $\mathbb{G}_2$?

Homomorphism

$\mathbb{G}_1 \implies \mathbb{G}_2$

Example: $f(x) = 2x$; suppose $\mathbb{G}_1 = (\mathbb{Z}, +)$.

What is $\mathbb{G}_2$?

$2 \cdot (a + b) \equiv 2 \cdot a + 2 \cdot b$.

Homomorphism

$\mathbb{G}_1 \implies \mathbb{G}_2$

Example: $f(x) = 2x$; suppose $\mathbb{G}_1 = (\mathbb{Z}, +)$.

What is $\mathbb{G}_2$?

$2 \cdot (a + b) \equiv 2 \cdot a + 2 \cdot b$. Why is this *so* important?

Cryptography: just encrypt/hide?

Homomorphism - 2

Cryptography: just encrypt/hide?

We want to do stuff under the encryption.

Homomorphism - 2

Cryptography: just encrypt/hide?

We want to do stuff under the encryption.

Guarantee correctness without knowledge.

Homomorphism - 2

Cryptography: just encrypt/hide?

We want to do stuff under the encryption.

Guarantee correctness without knowledge.

$16 + 4 = 20 \leftarrow 8 + 2 = 10.$

Homomorphism - 2

Cryptography: just encrypt/hide?

We want to do stuff under the encryption.

Guarantee correctness without knowledge.

$16 + 4 = 20 \leftarrow 8 + 2 = 10.$

Except for functions f that are *not* invertible!

Cryptography: just encrypt/hide?

We want to do stuff under the encryption.

Guarantee correctness without knowledge.

$16 + 4 = 20 \leftarrow 8 + 2 = 10.$

Except for functions f that are *not* invertible!

$a \cdot G + b \cdot G = c \cdot G \leftarrow a + b = c$

The canonical Σ-protocol

Hard to prove you know without revealing?

Schnorr ID protocol

Hard to prove you know without revealing?

To make it easier, prove **two** things instead!

Hard to prove you know without revealing?

To make it easier, prove **two** things instead!

Say secret x, for public P.

Make **new** secret k for public R.

Schnorr ID protocol

Hard to prove you know without revealing?

To make it easier, prove **two** things instead!

Say secret x, for public P.

Make **new** secret k for public R.

Prover $\mathcal{P} \Longrightarrow R \Longrightarrow$ Verifier $\mathcal{V}$.

Schnorr ID protocol

Hard to prove you know without revealing?

To make it easier, prove **two** things instead!

Say secret x, for public P.

Make **new** secret k for public R.

Prover $\mathcal{P} \Longrightarrow R \Longrightarrow$ Verifier $\mathcal{V}$.

$\mathcal{P} \Longleftarrow c \Longleftarrow \mathcal{V}$

Schnorr ID protocol

Hard to prove you know without revealing?

To make it easier, prove **two** things instead!

Say secret x, for public P.

Make **new** secret k for public R.

Prover $\mathcal{P} \Longrightarrow R \Longrightarrow$ Verifier $\mathcal{V}$.

$\mathcal{P} \Longleftarrow c \Longleftarrow \mathcal{V}$

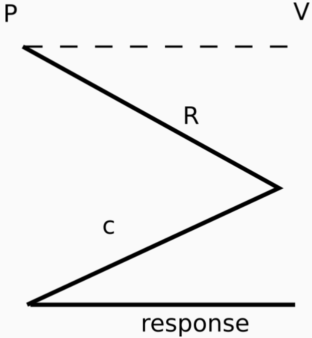
$\mathcal{P} \Longrightarrow$ "response" $\Longrightarrow \mathcal{V}$.

Schnorr ID protocol - 2

Why is it “sigma”? (  )

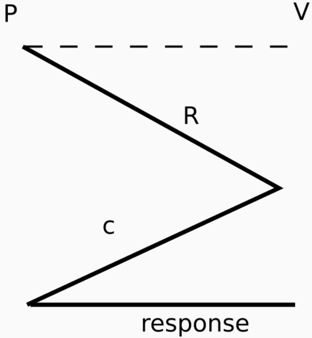
Schnorr ID protocol - 2

Why is it “sigma”? (  )



Schnorr ID protocol - 2

Why is it "sigma"? (  )



(EC)DL only? RSA, lattices - anything with homomorphism.

Schnorr ID protocol - 3

But what does this achieve?

Schnorr ID protocol - 3

But what does this achieve?

The “response” is unencrypted: $s = k + cx$

Schnorr ID protocol - 3

But what does this achieve?

The "response" is unencrypted: $s = k + cx$

But the **verification** is encrypted: $s \cdot G = R + c \cdot P$

Schnorr ID protocol - 3

But what does this achieve?

The “response” is unencrypted: $s = k + cx$

But the **verification** is encrypted: $s \cdot G = R + c \cdot P$

Verifier $\mathcal{V}$ only **tastes** the Coca Cola!

Schnorr ID protocol - 3

But what does this achieve?

The "response" is unencrypted: $s = k + cx$

But the **verification** is encrypted: $s \cdot G = R + c \cdot P$

Verifier $\mathcal{V}$ only **tastes** the Coca Cola!

k *hides* the first secret x.

Schnorr ID protocol - 3

But what does this achieve?

The "response" is unencrypted: $s = k + cx$

But the **verification** is encrypted: $s \cdot G = R + c \cdot P$

Verifier $\mathcal{V}$ only **tastes** the Coca Cola!

k *hides* the first secret x.

c *binds/fixes* the secret (P can't predict it).

Schnorr ID protocol - 3

But what does this achieve?

The "response" is unencrypted: $s = k + cx$

But the **verification** is encrypted: $s \cdot G = R + c \cdot P$

Verifier $\mathcal{V}$ only **tastes** the Coca Cola!

k *hides* the first secret x.

c *binds/fixes* the secret (P can't predict it).

Exactly because there is a homomorphism

for EC point addition, it works!

The Fiat-Shamir transform

The Schnorr ID protocol is **interactive**

The Schnorr ID protocol is **interactive**
, computationally “sound” and “HVZK”.


The Schnorr ID protocol is **interactive**
, computationally "sound" and "HVZK".
🤔 What if we fake the challenge? (like Fiat!).

The Fiat-Shamir transform

The Schnorr ID protocol is **interactive**

, computationally “sound” and “HVZK”.

🤔 What if we fake the challenge? (like Fiat!).

so — make a random challenge be a hash of R!

The Fiat-Shamir transform

The Schnorr ID protocol is **interactive**
, computationally “sound” and “HVZK”.

🤔 What if we fake the challenge? (like Fiat!).

so — make a random challenge be a hash of R!

$c = \mathbb{H}(R|P|..)$, so $s = k + \mathbb{H}(R|P|..)x$.

The Fiat-Shamir transform

The Schnorr ID protocol is **interactive**

, computationally "sound" and "HVZK".

🤔 What if we fake the challenge? (like Fiat!).

so — make a random challenge be a hash of R!

$c = \mathbb{H}(R|P|..)$, so $s = k + \mathbb{H}(R|P|..)x$.

Domain separation tags? See BIP340 $\mathbb{H}_{tag}()$

The Fiat-Shamir transform - 2

Generalize: *hash the conversation transcript up to the challenge*.

The Fiat-Shamir transform - 2

Generalize: *hash the conversation transcript up to the challenge*.

FS transform takes an **interactive identity protocol** and ...

Generalize: *hash the conversation transcript up to the challenge*.

FS transform takes an **interactive identity protocol** and ...

converts it into a signature scheme. We can attach any message we like into the transcript.

Generalize: *hash the conversation transcript up to the challenge*.

FS transform takes an **interactive identity protocol** and ...

converts it into a signature scheme. We can attach any message we like into the transcript.

Signatures are transferrable - the identity protocol is "deniable."

The Fiat-Shamir transform - 2

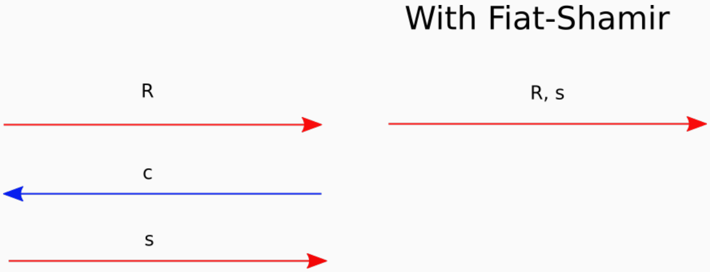
Generalize: *hash the conversation transcript up to the challenge*.

FS transform takes an **interactive identity protocol** and ...

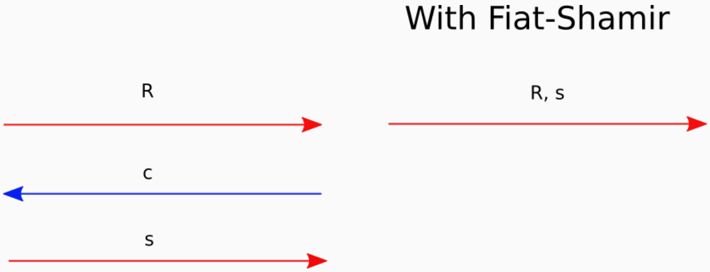
converts it into a signature scheme. We can attach any message we like into the transcript.

Signatures are transferrable - the identity protocol is "deniable." Security is based on the "Random Oracle Model".

The Fiat-Shamir transform - 3



The Fiat-Shamir transform - 3



Note that $\mathcal{V}$ must be able to recreate c as the hash (of R, etc.)

Increasing the power level: COMBINING Σ-protocols

An AND of Σ-protocols

Suppose you want to prove knowledge of x_1, x_2 for P_1, P_2

An AND of Σ-protocols

Suppose you want to prove knowledge of x_1, x_2 for P_1, P_2

Quiz: can you do this in a more compact way than just running the Σ-protocol twice?

Suppose you want to prove knowledge of x_1, x_2 for P_1, P_2

Quiz: can you do this in a more compact way than just running the Σ-protocol twice?

Answer: share the challenge.

Suppose you want to prove knowledge of x_1, x_2 for P_1, P_2

Quiz: can you do this in a more compact way than just running the Σ-protocol twice?

Answer: share the challenge.

$k_1, k_2 \Longrightarrow R_1, R_2 \Longrightarrow \mathcal{V}$

An AND of Σ-protocols

Suppose you want to prove knowledge of x_1, x_2 for P_1, P_2

Quiz: can you do this in a more compact way than just running the Σ-protocol twice?

Answer: share the challenge.

$k_1, k_2 \Longrightarrow R_1, R_2 \Longrightarrow \mathcal{V}$

$\mathcal{P} \Longleftarrow c \Longleftarrow$

An AND of Σ-protocols

Suppose you want to prove knowledge of x_1, x_2 for P_1, P_2

Quiz: can you do this in a more compact way than just running the Σ-protocol twice?

Answer: share the challenge.

$k_1, k_2 \Longrightarrow R_1, R_2 \Longrightarrow \mathcal{V}$

$\mathcal{P} \Longleftarrow c \Longleftarrow$

$s_1 = k_1 + cx_1, \; s_2 = k_2 + cx_2$

An AND of Σ-protocols

Suppose you want to prove knowledge of x_1, x_2 for P_1, P_2

Quiz: can you do this in a more compact way than just running the Σ-protocol twice?

Answer: share the challenge.

$k_1, k_2 \Longrightarrow R_1, R_2 \Longrightarrow \mathcal{V}$

$\mathcal{P} \Longleftarrow c \Longleftarrow$

$s_1 = k_1 + cx_1,\ s_2 = k_2 + cx_2$

$\mathcal{V}$: $s_1 \cdot G \stackrel{?}{=} R_1 + c \cdot P_1 \ \wedge\ s_2 \cdot G \stackrel{?}{=} R_2 + c \cdot P_2$.

An AND of Σ-protocols - 2

Quiz: what should be in the $\mathbb{H}$ in this case?

An AND of Σ-protocols - 2

Quiz: what should be in the $\mathbb{H}$ in this case?

Answer: $R_1, R_2, P_1, P_2, \ldots$.

An OR of Σ-protocols

I know 1 of x_1, x_2 for P_1, P_2.

I know 1 of x_1, x_2 for P_1, P_2.

CDS 94 (but AOS 2002 is better):

I know 1 of x_1, x_2 for P_1, P_2.

CDS 94 (but AOS 2002 is better):

$\mathcal{P}$: Choose s_1, c_1 and k_2. Calculate:

I know 1 of x_1, x_2 for P_1, P_2.

CDS 94 (but AOS 2002 is better):

$\mathcal{P}$: Choose s_1, c_1 and k_2. Calculate:

$R_1 = s_1 \cdot G - c_1 \cdot P_1, R_2 = k_2 \cdot G$

I know 1 of x_1, x_2 for P_1, P_2.

CDS 94 (but AOS 2002 is better):

$\mathcal{P}$: Choose s_1, c_1 and k_2. Calculate:

$R_1 = s_1 \cdot G - c_1 \cdot P_1, R_2 = k_2 \cdot G$

Send R_1, R_2 to $\mathcal{V}$.

An OR of Σ-protocols

I know 1 of x_1, x_2 for P_1, P_2.

CDS 94 (but AOS 2002 is better):

$\mathcal{P}$: Choose s_1, c_1 and k_2. Calculate:

$R_1 = s_1 \cdot G - c_1 \cdot P_1, R_2 = k_2 \cdot G$

Send R_1, R_2 to $\mathcal{V}$.

$\mathcal{V}$ sends **single** challenge c.

An OR of Σ-protocols

I know 1 of x_1, x_2 for P_1, P_2.

CDS 94 (but AOS 2002 is better):

$\mathcal{P}$: Choose s_1, c_1 and k_2. Calculate:

$R_1 = s_1 \cdot G - c_1 \cdot P_1, R_2 = k_2 \cdot G$

Send R_1, R_2 to $\mathcal{V}$.

$\mathcal{V}$ sends **single** challenge c.

$\mathcal{P}$: $c_2 = c \oplus c_1,\ s_2 = k_2 + c_2 x_2$, send $(s_1, s_2), (c_1, c_2)$

An OR of Σ-protocols

I know 1 of x_1, x_2 for P_1, P_2.

CDS 94 (but AOS 2002 is better):

$\mathcal{P}$: Choose s_1, c_1 and k_2. Calculate:

$R_1 = s_1 \cdot G - c_1 \cdot P_1, R_2 = k_2 \cdot G$

Send R_1, R_2 to $\mathcal{V}$.

$\mathcal{V}$ sends **single** challenge c.

$\mathcal{P}$: $c_2 = c \oplus c_1,\ s_2 = k_2 + c_2 x_2$, send $(s_1, s_2), (c_1, c_2)$

$\mathcal{V}$: $s_n \cdot G \stackrel{?}{=} R_n + c_n \cdot P_n \ \wedge \ c \stackrel{?}{=} c_1 \oplus c_2$

An OR of Σ-protocols - 2

Nice trick! $\oplus$ perfectly hides *which* “signature equation” $s_n = k_n + cx_n$ is real and which are faked.

An OR of Σ-protocols - 2

Nice trick! $\oplus$ perfectly hides *which* “signature equation” $s_n = k_n + cx_n$ is real and which are faked. Wagner?

An OR of Σ-protocols - 2

Nice trick! $\oplus$ perfectly hides *which* "signature equation" $s_n = k_n + cx_n$ is real and which are faked. Wagner?

AOS style is different: form a causal loop over the whole set of 4 by each challenge hashing the *previous* index. More efficient.

Special case AND - DLEQs

Equality of the discrete log of two points w.r.t. two bases G, H.

Equality of the discrete log of two points w.r.t. two bases G, H.

$P = x \cdot G \ \wedge \ Q = x \cdot H$.

Special case AND - DLEQs

Equality of the discrete log of two points w.r.t. two bases G, H.

$P = x \cdot G \ \wedge \ Q = x \cdot H$.

$\mathcal{P} : k \Longrightarrow R_1 = k \cdot G, R_2 = k \cdot H \Longrightarrow$

Special case AND - DLEQs

Equality of the discrete log of two points w.r.t. two bases G, H.

$P = x \cdot G \ \wedge \ Q = x \cdot H$.

$\mathcal{P} : k \Longrightarrow R_1 = k \cdot G, R_2 = k \cdot H \Longrightarrow$

$\Longleftarrow c \Longleftarrow \mathcal{V}$

Special case AND - DLEQs

Equality of the discrete log of two points w.r.t. two bases G, H.

$P = x \cdot G \;\wedge\; Q = x \cdot H$.

$\mathcal{P} : k \Longrightarrow R_1 = k \cdot G, R_2 = k \cdot H \Longrightarrow$

$\Longleftarrow c \Longleftarrow \mathcal{V}$

$\mathcal{P}$ sends *one* response: $s = k + cx$

Special case AND - DLEQs

Equality of the discrete log of two points w.r.t. two bases G, H.

$P = x \cdot G \;\wedge\; Q = x \cdot H$.

$\mathcal{P} : k \Longrightarrow R_1 = k \cdot G, R_2 = k \cdot H \Longrightarrow$

$\Longleftarrow c \Longleftarrow \mathcal{V}$

$\mathcal{P}$ sends *one* response: $s = k + cx$

$\mathcal{V}$ checks: $s \cdot G \stackrel{?}{=} R_1 + c \cdot P \;\wedge\; s \cdot H \stackrel{?}{=} R_2 + c \cdot Q$.

Many keys in linear relationships

Give $P_1 = x_1 \cdot G_1, P_2 = x_2 \cdot G_2$, prove in ZK that $3x_1 + 10x_2 = 15$

Many keys in linear relationships

Give $P_1 = x_1 \cdot G_1, P_2 = x_2 \cdot G_2$, prove in ZK that
$3x_1 + 10x_2 = 15$
Choose two commitments $R_1 = k_1 \cdot G_1, R_2 = k_2 \cdot G_2$,
where $3k_1 + 10k_2 = 0$

Many keys in linear relationships

Give $P_1 = x_1 \cdot G_1, P_2 = x_2 \cdot G_2$, prove in ZK that $3x_1 + 10x_2 = 15$

Choose two commitments $R_1 = k_1 \cdot G_1, R_2 = k_2 \cdot G_2$, where $3k_1 + 10k_2 = 0$

F-S: $c = \mathbb{H}(P_1, P_2, R_1, R_2, G_1, G_2)$

Give $P_1 = x_1 \cdot G_1, P_2 = x_2 \cdot G_2$, prove in ZK that
$3x_1 + 10x_2 = 15$
Choose two commitments $R_1 = k_1 \cdot G_1, R_2 = k_2 \cdot G_2$,
where $3k_1 + 10k_2 = 0$
F-S: $c = \mathbb{H}(P_1, P_2, R_1, R_2, G_1, G_2)$
Send proof: $(c, s_1 = k_1 + cx_1,\ s_2 = k_2 + cx_2)$

Many keys in linear relationships

Give $P_1 = x_1 \cdot G_1, P_2 = x_2 \cdot G_2$, prove in ZK that $3x_1 + 10x_2 = 15$

Choose two commitments $R_1 = k_1 \cdot G_1, R_2 = k_2 \cdot G_2$, where $3k_1 + 10k_2 = 0$

F-S: $c = \mathbb{H}(P_1, P_2, R_1, R_2, G_1, G_2)$

Send proof: $(c, s_1 = k_1 + cx_1,\ s_2 = k_2 + cx_2)$

$\mathcal{V}$: $R_1 := s_1 \cdot G_1 - c \cdot P_1, R_2 := s_2 \cdot G_2 - c \cdot P_2$

Many keys in linear relationships

Give $P_1 = x_1 \cdot G_1, P_2 = x_2 \cdot G_2$, prove in ZK that $3x_1 + 10x_2 = 15$

Choose two commitments $R_1 = k_1 \cdot G_1, R_2 = k_2 \cdot G_2$, where $3k_1 + 10k_2 = 0$

F-S: $c = \mathbb{H}(P_1, P_2, R_1, R_2, G_1, G_2)$

Send proof: $(c, s_1 = k_1 + cx_1,\ s_2 = k_2 + cx_2)$

$\mathcal{V}$: $R_1 := s_1 \cdot G_1 - c \cdot P_1, R_2 := s_2 \cdot G_2 - c \cdot P_2$

$3s_1 + 10s_2 \stackrel{?}{=} 15c \wedge c \stackrel{?}{=} \mathbb{H}(P_1, P_2, R_1, R_2, G_1, G_2)$.

Can generalize to a whole set of linear simultaneous equations

Conclusion

Homework!

Sketch an outline of a proof of knowledge of the opening of a Pedersen commitment.

Homework!

Sketch an outline of a proof of knowledge of the opening of a Pedersen commitment.

$C(a) = r \cdot G + a \cdot H$

Sketch an outline of a proof of knowledge of the opening of a Pedersen commitment.
$C(a) = r \cdot G + a \cdot H$

- CLUE: what is the homomorphism?

Homework!

Sketch an outline of a proof of knowledge of the opening of a Pedersen commitment.

$C(a) = r \cdot G + a \cdot H$

- CLUE: what is the homomorphism?
- (See: "Okamoto's protocol for representations".)

References - 1

- Boneh and Shoup, see Chapter 19
- Dan Boneh on Σ-protocols - video lecture
- standardisation of Σ-protocols (survey)
- concept of DLEQ
- explainer on DLEQs
- Nadav Kohen on blind sigs

References - 2

- How PrivacyPass uses DLEQs
- My Bulletproofs writeup
- Camenisch & Stadler '97 Proof systems for discrete logs
- Ring signatures - my blog
- Matt Green on PAKE
- Brands' book on credentials (see also: uprove)

References - 3

- BIP 340 Bitcoin Schnorr signatures
- Chaum's original blind signatures paper
- Fedimint - federated Chaumian mints
- Open Transactions - earlier Chaumian mints
- Security proofs for Schnorr - my blog
- Lloyd Fournier on adaptors as 'otVES'
- Kohen on ECDSA adaptors via DLEQ

References - 4

- Gabizon on zkSNARKs
- Nadav Kohen on payment points (PTLCs)
- Wabisabi paper (anonymous credentials)
- WabiSabi and its precursors
- Algebraic MACs and Key-Verified Anonymous Credentials Chase, Meiklejohn, Zaveruccha 2013
- Anonymous Credentials in Signal - Chase, Perrin, Zaveruccha 2019

Thank you

Contact info:

@waxwing@x0f.org (mastodon)

https://github.com/AdamISZ

blog: https://reyify.com/blog (email there)

gpg: 4668 9728 A9F6 4B39 1FA8 71B7 B3AE 09F1 E9A3 197A