Σ-protocols
- and why they should matter to every Bitcoin thinker

Adam Gibson
3rd March 2022
Advancing Bitcoin
Motivation \( \Sigma \)-protocols in the wild
Outline

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

- **Motivation** $\Sigma$-protocols in the wild
- **ZK** Analogies; zero knowledge
- **Homomorphism** the special property needed for $\Sigma$-protocols
- **The most fundamental $\Sigma$-protocol**
M. Motivation $\Sigma$-protocols in the wild

- ZK Analogies; zero knowledge

- Homomorphism the special property needed for $\Sigma$-protocols

- The most fundamental $\Sigma$-protocol

- The Fiat-Shamir transform
Outline

- Motivation \( \Sigma \)-protocols in the wild
- ZK Analogies; zero knowledge
- Homomorphism the special property needed for \( \Sigma \)-protocols
- The most fundamental \( \Sigma \)-protocol
- The Fiat-Shamir transform
- Combinations of \( \Sigma \)-protocols - AND, OR, ...
Sigma protocols in the wild
Let \( M = rT \) be the blinded token that \( S \) sends to \( C \), let \((G,Y) = (g, g^x)\) be the commitment from above, and let \( H_3 \) be a new hash function (modeled 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 \( k \leftarrow \mathbb{Z}_q \) 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 \( B' = sM + cZ \) and hashes \( c' = H_3(G,Y,M,Z,A',B') \).
- \( C \) verifies that \( c' = c \).

Note that correctness follows since

\[
A' = sG + cY = (k-cx)G + cxG = kG \quad \text{and} \quad B' = sM + cZ = r(k-cx)T + crxT = krT = kM
\]

We write \( \text{DLEQ}(Z/M \equiv Y/G) \) to denote the proof that is created by \( S \) and validated by \( C \).
Here are some examples of things that are based on the $\Sigma$-protocol:
Here are some examples of things that are based on the Σ-protocol:
BIP340 .. edDSA .. ECDSA (kinda)
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.)
"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.
"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
"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)
Start with a silly analogy:
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 $\Sigma$-protocol paradigm.
● Claim: challenge: demonstrate - this is the \( \Sigma \)-protocol paradigm.

● To win at this game, you have to be ready to be ready to create a demonstration for any challenge.
Claim: challenge: demonstrate - this is the $\Sigma$-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 ..
Another intuition - coin flip over a telephone line.
- We assign 1BTC based on whether you succeed in ’calling’ the coin flip.
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.
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

$G_1 \rightarrow G_2$
Homomorphism

\[ G_1 \rightarrow G_2 \]

Example: \( f(x) = 2x \); suppose \( G_1 = (\mathbb{Z}, +) \).

What is \( G_2 \)?
\( \mathcal{G}_1 \longrightarrow \mathcal{G}_2 \)

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

What is \( \mathcal{G}_2 \)?

\[ 2 \cdot (a + b) \equiv 2 \cdot a + 2 \cdot b. \]
Homomorphism

$G_1 \rightarrow G_2$

Example: $f(x) = 2x$; suppose $G_1 = (\mathbb{Z}, +)$.

What is $G_2$?

$2 \cdot (a + b) \equiv 2 \cdot a + 2 \cdot b$. Why is this so important?
Cryptography: just encrypt/hide?
Cryptography: just encrypt/hide?
We want to do stuff under the encryption.
Cryptography: just encrypt/hide?
We want to do stuff under the encryption.
Guarantee correctness without knowledge.
Cryptography: just encrypt/hide? We want to do stuff under the encryption. Guarantee correctness without knowledge. \[16 + 4 = 20 \leftarrow 8 + 2 = 10.\]
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 $\Sigma$-protocol
Schnorr ID 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!
Schnorr ID protocol

Hard to prove you know without revealing? To make it easier, prove \textbf{two} things instead!
Say secret $x$, for public $P$.
Make \textbf{new} secret $k$ for public $R$. 
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 $P \rightarrow R \rightarrow$ Verifier $V$. 
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 $P \Rightarrow R \Rightarrow$ Verifier $V$.
$P \leftarrow c \leftarrow V$
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 $P \implies R \implies$ Verifier $V$.
$P \leftarrow c \leftarrow V$
$P \implies \text{“response”} \implies V.$
Why is it “sigma”? ( )
Why is it “sigma”? (P V R C response)
Why is it “sigma”? (EC)DL only? RSA, lattices - anything with homomorphism.
But what does this achieve?
But what does this achieve?
The “response” is unencrypted: $s = k + cx$
But what does this achieve?
The “response” is unencrypted: \( s = k + cx \)
But the **verification** is encrypted: \( s \cdot G = R + c \cdot P \)
But what does this achieve?
The “response” is unencrypted: \( s = k + cx \)
But the \textbf{verification} is encrypted: \( s \cdot G = R + c \cdot P \)
Verifier \( V \) only \textbf{tastes} the Coca Cola!
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 \( x \).
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).
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 Fiat-Shamir transform

The Schnorr ID protocol is \textit{interactive}
The Schnorr ID protocol is **interactive**, computationally “sound” and “HVZK”.
The Fiat-Shamir transform

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 \( R \! \)

\[
c = \mathbb{H}(R|P|..), \text{ so } s = k + \mathbb{H}(R|P|..)x.
\]
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 $\mathbb{H}_{tag}()$
Generalize: hash the conversation transcript up to the challenge.
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.”
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

With Fiat-Shamir

R

C

S

R, s
Note that $\mathcal{V}$ must be able to recreate $c$ as the hash (of $R$, etc.)
Increasing the power level: COMBINING $\Sigma$-protocols
An AND of $\Sigma$-protocols

Suppose you want to prove knowledge of $x_1, x_2$ for $P_1, P_2$
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 $\Sigma$-protocol twice?
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.
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 $\Sigma$-protocol twice?

Answer: share the challenge.

$k_1, k_2 \implies R_1, R_2 \implies \forall$
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 $\Sigma$-protocol twice?

Answer: share the challenge.

$k_1, k_2 \implies R_1, R_2 \implies V$

$\mathcal{P} \leftarrow c \leftarrow$
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 $\Sigma$-protocol twice?

Answer: share the challenge.

$k_1, k_2 \implies R_1, R_2 \implies V$

$P \leftarrow c \leftarrow$

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

\[
\begin{align*}
k_1, k_2 \rightarrow & R_1, R_2 \rightarrow V \\
\mathcal{P} & \leftarrow c \leftarrow \\
s_1 = k_1 + cx_1, & s_2 = k_2 + cx_2 \\
V: s_1 \cdot G \equiv & R_1 + c \cdot P_1 \land s_2 \cdot G \equiv R_2 + c \cdot P_2.
\end{align*}
\]
An AND of \( \Sigma \)-protocols - 2

Quiz: what should be in the \( \mathbb{H} \) in this case?
An AND of $\sum$-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 $\Sigma$-protocols

I know 1 of $x_1, x_2$ for $P_1, P_2$. 
An OR of $\Sigma$-protocols

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:
An OR of $\Sigma$-protocols

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

CDS 94 (but AOS 2002 is better):

$P$: 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:

\[ R_1 = s_1 \cdot G - c_1 \cdot P_1, \quad R_2 = k_2 \cdot G \]

Send $R_1$, $R_2$ to $\mathcal{V}$.
An OR of $\Sigma$-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 $\Sigma$-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:

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

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

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

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

$\mathcal{V}$: $s_n \cdot G = R_n + c_n \cdot P_n \land \quad c = c_1 \oplus c_2$
An OR of $\Sigma$-protocols - 2

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

\[ P = x \cdot G \land Q = x \cdot H. \]
Equality of the discrete log of two points w.r.t. two bases $G, H$.

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

$\mathcal{P} : k \implies R_1 = k \cdot G, R_2 = k \cdot H \implies$
Special case AND - DLEQs

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

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

$\mathcal{P} : k \implies R_1 = k \cdot G, R_2 = k \cdot H \implies \leftarrow c \leftarrow \mathcal{V}$
Equality of the discrete log of two points w.r.t. two bases $G, H$.

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

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

$\leftarrow c \leftarrow \mathcal{V}$

$\mathcal{P}$ sends one response: $s = k + cx$
Special case AND - DLEQs

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

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

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

$\iff c \iff \mathcal{V}$

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

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

Given $P_1 = x_1 \cdot G_1, P_2 = x_2 \cdot G_2$, 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: $c = \mathbb{H}(P_1, P_2, R_1, R_2, G_1, G_2)$
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: $(c, s_1 = k_1 + cx_1, s_2 = k_2 + cx_2)$
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: $(c, s_1 = k_1 + cx_1, s_2 = k_2 + cx_2)$

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

$3s_1 + 10s_2 \equiv 15c \land c \equiv \mathbb{H}(P_1, P_2, R_1, R_2, G_1, G_2)$.

Can generalize to a whole set of linear simultaneous equations
Conclusion
Sketch an outline of a proof of knowledge of the opening of a Pedersen commitment.
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?
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 $\Sigma$-protocols - video lecture
- standardisation of $\Sigma$-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