A Threshold Signature Primer

Lucas Meier (commonware.xyz)

SECTION 1

The Elevator Pitch

You want to fund three whimsical bird watching / hedge mazes off the coast of Spain with some friends…

You want to fund three whimsical bird watching / hedge mazes off the coast of Spain with some friends…

ternary zany canary topiary aviaries

You want to fund three whimsical bird watching / hedge mazes off the coast of Spain with some friends...

ternary zany canary topiary aviaries

...how can you cryptographically ensure that no one can spend the funds themselves?

You want to fund three whimsical bird watching / hedge mazes off the coast of Spain with some friends...

ternary zany canary topiary aviaries

...how can you cryptographically ensure that no one can spend the funds themselves?

what if you want to just have one signature?

You want to fund three whimsical bird watching / hedge mazes off the coast of Spain with some friends...

ternary zany canary topiary aviaries

...how can you cryptographically ensure that no one can spend the funds themselves?

what if you want to just have one signature?

what if you only want majority approval?

SECTION 2

Multi-Signatures & Threshold Signatures

- **Signature:** Alice signed this message.
- **Multi-Signature:** Alice, Bob, and Charlie signed it.
- **Threshold-Signature:** 2 people among Alice, Bob, and Charlie signed it.

“Silent” vs “Loud” multisigs

- **Loud**: We can define a “signature” to consist of several signatures.
- **Silent**: The group produces the same signature as a single signer.

Why “silent”?

Why “silent”?

- Keep a stable identity over time.
- Privacy.
- Interoperability.

Applications

Applications

- Cryptocurrency wallets
- Institutional custody
- Light Client Certificates

Goals

You will be a **BIG SHOT** at the following topics:

Goals

- You will be a **BIG SHOT** at the following topics:
  - polynomial secret sharing,

Goals

You will be a **BIG SHOT** at the following topics:

- polynomial secret sharing,
- using the homomorphic properties of group commitments to great profits,

Goals

You will be a **BIG SHOT** at the following topics:

- polynomial secret sharing,
- using the homomorphic properties of group commitments to great profits,
- distributed key generation

Goals

You will be a **BIG SHOT** at the following topics:

- polynomial secret sharing,
- using the homomorphic properties of group commitments to great profits,
- distributed key generation
- threshold BLS and Schnorr signing.

Warning:

The US Government has declared the contents of this talk to be ~~outdated~~ **illegal** by the year **2031**

SECTION 3

ECC Background

From Elliptic Curves to Cryptographic Groups

- Elliptic curves are a particular model…

From Elliptic Curves to Cryptographic Groups

- Elliptic curves are a particular model…
- of a pretty simple abstraction.

A Group

- $A, B, C, ... \in \mathbb{G}$
- $0 \in \mathbb{G}$
- $A + B$
- $-A$

A Field

- $x, y, z, ... \in \mathbb{F}$
- $0, 1 \in \mathbb{F}$
- $x + y$
- $-x$
- $x \cdot y$
- $\frac{x}{y}$

Addition Rules

For $\mathbb{G}$ and $\mathbb{F}$:

- $$(a + b) + c = a + (b + c)$$
- $$a + b = b + a$$
- $$a + 0 = a$$
- $$a - a := a + (-a) = 0$$

Multiplication Rules

For $\mathbb{F}$:

- $x(yz) = (xy)z$
- $xy = yx$
- $x1 = x$
- $\frac{x}{x} = 1$ (except 0)
- $x(y + z) = xy + xz$

The Field acts on the Group

For $x, y \in \mathbb{F}$, and $A, B \in \mathbb{G}$:

- $$x \cdot A$$
- $$(x + y) \cdot A = x \cdot A + y \cdot A$$
- $$xyA = x(yA)$$

Hard Problems

We pick a point $G \in \mathbb{G}$, and then:

- Given $X := x \cdot G$, finding x is hard.
  (Discrete Logarithm)
- Given $X := x \cdot G, Y := y \cdot G$, finding $Z := (xy) \cdot G$ is hard.
  (Computational Diffie Hellman).
- Given X, Y, distinguishing $xy \cdot G$ from a random $P \in G$ is hard.
  (Decisional Diffie Hellman).

SECTION 4

Signatures

$$\mathrm{Gen}() : (\mathrm{Sk},\ \mathrm{Pk})$$

$$\mathrm{Sign}() : (\mathrm{Pk}, \mathrm{Sk}, M) \to \mathrm{Sig}$$

$$\mathrm{Verify}() : (\mathrm{Pk}, M, \mathrm{Sig})$$

$$\mathrm{Gen}() : (\mathrm{Sk},\ \mathrm{Pk})$$

$$\mathrm{Sign}() : (\mathrm{Pk}, \mathrm{Sk}, M) \to \mathrm{Sig}$$

$$\mathrm{Verify}() : (\mathrm{Pk}, M, \mathrm{Sig})$$

Security:

- you can sign $\sim$ you know sk,
- (can't change what message a signature is about)

SECTION 5

Schnorr Signatures

Algorithm

$$\underline{\text{Gen}()}$$

$$x \xleftarrow{\$} \mathbb{F}$$

$$(x, x \cdot G)$$

$$\underline{\text{Sign(X, m)}}$$

$$k \xleftarrow{\$} \mathbb{F}$$

$$K := k \cdot G$$

$$c := H(X, m)$$

$$s := k + cx$$

$$(K, s)$$

$$\underline{\text{Verify(X, m, (K, s))}}$$

$$s \cdot G \stackrel{?}{=} K + c \cdot X$$

Correctness

$$s \cdot G = (k + cx) \cdot G = k \cdot G + (cx) \cdot G = K + c \cdot X$$

SECTION 6

Pairings

From One Group to Three

$$\mathbb{G} \longrightarrow \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$$

From One Group to Three

$$\mathbb{G} \longrightarrow \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$$

but still sharing $\mathbb{F}$.

A New Operation

$$\odot : (\mathbb{G}_1, \mathbb{G}_2) \to \mathbb{G}_T$$

Rules

- $$(A + B) \odot C = A \odot C + B \odot C$$
- $$(xA) \odot B = x(A \odot B) = A \odot (xB)$$

MULTIPLICATIVE NOTATION (+ e)

= LIE

group law:	gh	← looks like field mult
inverse:	g^{-1}	← why not $1/g$???
identity:	1	← the number 1 ? really ??
exponentiation:	g^x	← confusing... is g to the power x a field thing? ugh
scalar mult:	g^x	←
addition:	gh	← looks like field mult again
scalar addition:	$g^x h^y$	← what is this ??
pairing:	$e(g, h)$	← e what ??

looks like field stuff

people forget this isn't commutative mult

ugly and inconsistent

harder to read and teach

VS

ODOT NOTATION

= OBVIOUSLY SUPERIOR

group law:	$G + H$	← like vector addition (makes sense)
inverse:	$-G$	← clear and clean
identity:	0	← 0
exponentiation:	$x\,G$	← obviously a scalar times a point
scalar mult:	$x\,G$	← consistent and clear
addition:	$G + H$	← consistent and clear
scalar addition:	$x\,G + y\,H$	← perfect
pairing:	$G \odot H$	← clean and unambiguous

matches geometry and linear algebra

consistent and easy to learn

no confusion with fields

clearly shows what's happening (points + scalars)

Multiplicative notation: looks like a lie, teaches bad habits, causes pain

$\neq$

Odot notation: honest, natural, beautiful, and mathematically sound ♡

Aside: Performance

Approximately:

- $\mathbb{F}$ operations: cheap
- $\mathbb{G}$ operations: 10x more expensive
- $\mathbb{F} \cdot \mathbb{G}$: 10x again
- $\odot$: another 10x

In summary: minimizing pairings is very impactful to performance.

Note: $\sum_i A_i \odot B_i$ is faster than n pairings.

SECTION 7

BLS Signatures

The Algorithm

$$\underline{\mathrm{Gen}()}$$

$$x \xleftarrow{\$} \mathbb{F}$$

$$(x, x \cdot G_1)$$

$$\underline{\mathrm{Sign(m)}}$$

$$xH(m)$$

$$\underline{\mathrm{Verify(X,\ m,\ \sigma)}}$$

$$G_1 \odot \sigma \stackrel{?}{=} X \odot H(m)$$

Correctness

$$G_1 \odot \sigma = G_1 \odot (xH(m)) = (xG_1) \odot H(m) = X \odot H(m)$$

Pitfall

$$H'(m) := \mathbb{F}(H(m)) \cdot G_2$$

Is this secure?

Pitfall

$$H'(m) := \mathbb{F}(H(m)) \cdot G_2$$

Is this secure?

$$X \odot H'(M) = X \odot (\mathbb{F}(H(m)) \cdot G_2) = (\mathbb{F}(H(m))X) \odot G_2$$

SECTION 8

BLS Multi-Signatures

SECTION 9

$$X_1 \odot H(m) \stackrel{?}{=} G_1 \odot \sigma_1$$

$$X_2 \odot H(m) \stackrel{?}{=} G_1 \odot \sigma_2$$

what if we just did...

$$X_1 \odot H(m) \stackrel{?}{=} G_1 \odot \sigma_1$$

$$X_2 \odot H(m) \stackrel{?}{=} G_1 \odot \sigma_2$$

what if we just did...

$$X_1 \odot H(m) + X_2 \odot H(m) \stackrel{?}{=} G_1 \odot \sigma_1 + G_1 \odot \sigma_2$$

$$X_1 \odot H(m) \stackrel{?}{=} G_1 \odot \sigma_1$$

$$X_2 \odot H(m) \stackrel{?}{=} G_1 \odot \sigma_2$$

what if we just did...

$$X_1 \odot H(m) + X_2 \odot H(m) \stackrel{?}{=} G_1 \odot \sigma_1 + G_1 \odot \sigma_2$$

$$(X_1 + X_2) \odot H(m) \stackrel{?}{=} G_1 \odot (\sigma_1 + \sigma_2)$$

Is this secure?

Kinda

Here's one attack:

Kinda

Here's one attack:

$$A \odot H(m), B \odot H(m)$$

Kinda

Here's one attack:

$$A \odot H(m), B \odot H(m)$$

$$C = -(A + B)\ (A + B - (A + B)) \odot H(m) = 0 \odot H(m) = 0$$

so

$\sigma = 0$ works

Why Kinda?

The attacker doesn't know x such that: $x \cdot G_1 = -(A + B)$. By requiring this knowledge (e.g. with a signature), we thwart this attack.

The Ultimate Trick to Avoid These Issues

$$r_i \xleftarrow{\$} \mathbb{F}$$

$$(r_1 X_1 + ... + r_n X_n) \odot H(m) = G_1 \odot (r_1 \sigma_1 + ...)$$

The Ultimate Trick to Avoid These Issues

$$r_i \xleftarrow{\$} \mathbb{F}$$

$$(r_1 X_1 + ... + r_n X_n) \odot H(m) = G_1 \odot (r_1 \sigma_1 + ...)$$

Or, in general:

$$\forall i, \varphi(a_i, b_i, ...) = 0$$

$$\sum_i r_i \varphi(a_i, b_i, ...) = 0$$

Batch Verification

$$\forall i, X_i \odot H(m_i) = G_1 \odot \sigma_i$$

$$\forall i, X_i \odot H(m_i) - G_1 \odot \sigma_i$$

$$\sum_i (r_i X_i \odot H(m_i)) - G_1 \odot \left(\sum_i \sigma_i \right)$$

SECTION 10

Secret Sharing

Multi-Sigs are Linear

$$x = x_1 + x_2 + ... + x_n$$

need all the keys.

What if we only want to need t of them?

The solution to 82.7% of math problems?

The solution to 82.7% of math problems?

Polynomials.

Shamir's Secret Sharing

Given t coefficients:

$$f(X) := a_0 + a_1 X + ... + a_{\{t-1\}} X^{\{t-1\}}$$

any t points:

$$f(p_0), f(p_1), ..., f\left(p_{\{t-1\}}\right)$$

determine f exactly

(Intuition: $f(p) = 0 \Leftrightarrow f = (X - p)...$)

The Sharing

$$x := f(0)$$

$$x_i := f(p_i)$$

can locally compute λ_i such that:

$$\sum_i \lambda_i x_i = f(0)$$

Some Useful Properties

$$f(x) + g(x) = (f + g)(x) = (f_0 + g_0) + (f_1 + g_1)x + ...$$

$$F := f \cdot G$$

$$F(x) = (a_0 G) + x(a_1 G) + ... = f(x) \cdot G$$

Local operations on secret shares

$$f(p_i) + g(p_i) = (f + g)(p_i)$$

$$\alpha f(p_i) = (\alpha f)(p_i)$$

Galaxy Brained Alternative

For each subset S of t signers, generate a linear sharing

$$x = x_{S1} + x_{S2} + ...$$

When a particular subset S wants to sign, use x_{Si} as your share.

What if you don't know what subset will be present?

Galaxy Brained Alternative

For each subset S of t signers, generate a linear sharing

$$x = x_{S1} + x_{S2} + ...$$

When a particular subset S wants to sign, use x_{Si} as your share.

What if you don't know what subset will be present?

Just generate a signing session in parallel for each possible subset who **might** sign.

Galaxy Brained Alternative

For each subset S of t signers, generate a linear sharing

$$x = x_{S1} + x_{S2} + ...$$

When a particular subset S wants to sign, use x_{Si} as your share.

What if you don't know what subset will be present?

Just generate a signing session in parallel for each possible subset who **might** sign.

You do need an exponential number of shares...

Galaxy Brained Alternative

For each subset S of t signers, generate a linear sharing

$$x = x_{S1} + x_{S2} + ...$$

When a particular subset S wants to sign, use x_{Si} as your share.

What if you don't know what subset will be present?

Just generate a signing session in parallel for each possible subset who **might** sign.

You do need an exponential number of shares...

but that didn't stop Monero from deploying this for their wallet.

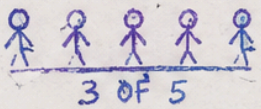
SPAMTON G. SPAMTON ON SECRET SHARING

SHAMIR'S SECRET SHARING?

WHAT A SCAM!!!

It's too simple!!! It doesn't serve my [$!@#]ing AMBITION!!!

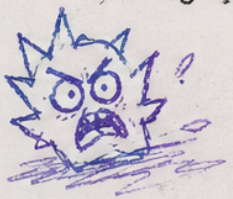
- FIXED THRESHOLD?? So rigid!!! what if i need 3 today but 5 tomorrow!?



NOT FLEXIBLE! :(

- CAN'T PICK WHO CAN COMBINE??? what if i don't trust someone today!?

- NO SUBSET MAGIC!!! what if i want ANY group to be able to sign!?



SO LIMITING...

MORE SHARES!
MORE OPTIONS!
MORE [[KROMER]]!!!

INSTEAD, DO THIS:

LINEAR SECRET SHARES FOR EVERY POSSIBLE SUBSET!!!

ANY GROUP!
ANY SIZE!
ANY TIME!
PERFECT!!!

FOR ALL $S \subseteq \{1...n\}$, $|S| \geq t$
PRECOMPUTE LINEAR SHARES!

THERE ARE $2^n - 1$ SUBSETS!
WE COVER THEM ALL!!!

STORE YOUR EXPONENTIALLY SIZED SHARE SET WITH ME!
$$$$$$$/MONTH
PREMIUM KEY STORAGE
(WORTH EVERY [KROMER])

- REDUNDANCY? THAT'S CALLED FLEXIBILITY!!!
- STORAGE? THAT'S CALLED INVESTING!!! :)

SECTION 11

BLS Threshold Signatures

$$x \cdot H(m) = f(0) \cdot H(m) = \lambda_i x_i \cdot H(m) = \lambda_i \cdot (x_i \cdot H(m)) = \lambda_i \cdot \sigma_i$$

SECTION 12

Distributed Key Generation

Warmup: Linear Sharing

$$x = x_1 + x_2 + ... + x_n$$

$$X = x \cdot G$$

Protocol

How can each person get their linear share x_i?

Protocol

How can each person get their linear share x_i?

$$x_i \xleftarrow{\$} \mathbb{F}$$

Now, how can we learn $\sum_i x_i \cdot G$?

Protocol

How can each person get their linear share x_i?

$$x_i \xleftarrow{\$} \mathbb{F}$$

Now, how can we learn $\sum_i x_i \cdot G$?

$$X_i := x_i \cdot G \rightrightarrows$$

$$X := \sum_i X_i$$

Trusted Key Generation

$$f \xleftarrow{\$} \mathbb{F}[X]_{\leq t}$$

$$f(p_i) \longrightarrow P_i$$

$$F := f \cdot G$$

$$F$$

$F(0)$ is the public key, and $F(p_i)$ lets us verify partial signatures.

Honest Distributed Key Generation

$\underline{\text{Dealer}()_i}$

$$f_i \xleftarrow{\$} \mathbb{F}[X]_{\leq t}$$

$$f_i(p_j) \longrightarrow P_j$$

$$F_i := f_i \cdot G$$

$$F_i \rightrightarrows$$

$\underline{\text{Player}()_j}$

$$D_i \longrightarrow f_i(p_j)$$

$$f(p_j) := \sum_i f_{i(p_j)}$$

$$\rightrightarrows F_i$$

$$F := \sum_i F_i$$

$$\left(F, f(p_j)\right)$$

Key Insight

$$\sum_i f_i(p_j) = \left(\sum_i f_i\right)(p_j)$$

Problem Points

- What if I send $s_{ij} \neq f_i(p_j)$?
- What if I send different F_i to different players?
- What if I don't send anything at all?

SECTION 13

Two Round DKG

The Idea

- $f_i(p_j) \stackrel{?}{=} F_i(p_j)$ lets us check a dealer's share.
- We can complain if it's wrong.
- We can exclude bad dealers.
- Dealers reveal shares to protect against bad players.

The Protocol

$$\underline{\text{Dealer}()_i}$$

$$f_i \xleftarrow{\$} \mathbb{F}[X]_{\leq t}$$

$$F_i := f_i \cdot G$$

$$F_i \rightrightarrows$$

$$s_{ij} := f_i(p_j) \longrightarrow P_j$$

$$\sigma_j \longleftarrow ? \; P_j$$

$$\sigma_j \text{ or } s_{ij} \rightrightarrows$$

$$\underline{\text{Player}()_j}$$

$$\rightrightarrows F_i, D_i \longrightarrow s_{ij}$$

$$f(p_j) := \sum_i f_{i(p_j)}$$

$$\text{if } F_i(p_j) \stackrel{?}{=} s_{ij}$$

$$D_i \longleftarrow \text{sign}(D_i, F_i, \ldots)$$

$$\text{good} := \{D_i \mid \sigma_i \text{ good} \vee s_{ij} \text{ good}\}$$

$$F := \sum_{i \in \text{ good}} F_i, s_j := \sum_{i \in \text{ good}} s_{ij}$$

$$(F, s_j)$$

Questions to ponder

- How many dealers need to be good to get an acceptable result?

Questions to ponder

- How many dealers need to be good to get an acceptable result?
- What should a dealer do if they have $\geq t$ reveals?

Questions to ponder

- How many dealers need to be good to get an acceptable result?
- What should a dealer do if they have $\geq t$ reveals?
- What if too many dealers reveal the share for a given player?

Questions to ponder

- How many dealers need to be good to get an acceptable result?
- What should a dealer do if they have $\geq t$ reveals?
- What if too many dealers reveal the share for a given player?
- What needs to be signed by the player?

Questions to ponder

- How many dealers need to be good to get an acceptable result?
- What should a dealer do if they have $\geq t$ reveals?
- What if too many dealers reveal the share for a given player?
- What needs to be signed by the player?
- Is it necessary to broadcast F_i?

Questions to ponder

- How many dealers need to be good to get an acceptable result?
- What should a dealer do if they have $\geq t$ reveals?
- What if too many dealers reveal the share for a given player?
- What needs to be signed by the player?
- Is it necessary to broadcast F_i?
- What happens if players disagree about the dealer's final message?

SECTION 14

Practical Challenges

Synchrony

- How long should I wait to receive an acknowledgement?

Synchrony

- How long should I wait to receive an acknowledgement?
- How long should I wait for a dealer to reveal?

Observability

- How can external observers be convinced the protocol ran correctly?

Faulty Assumptions

- The protocol ties its security to assumptions about faults.

SECTION 15

One Round DKG

Idea

- private share $\Rightarrow$ public ciphertext
- complaints $\Rightarrow$ ZK Proof

Initial Protocol

$$f_i \xleftarrow{\$} \mathbb{F}[X]_{\leq t}$$

$$F_i := f_i \cdot G$$

$$c_{ij} := \mathrm{Enc}\big(\mathrm{sk}_i, \mathrm{pk}_j, f_i(p_j)\big)$$

$$\pi_{ij} := \mathrm{Prove}\big(\mathrm{sk}_i, s_{ij} \mid c_{ij} = \mathrm{Enc}\big(\mathrm{sk}_i, \mathrm{pk}_j, s_{ij}\big) \wedge s_{ij} \cdot G = S_{ij}\big)$$

Dealer posts c_{ij}, π_{ij}, S_{ij}, and everyone immediately knows whether the dealer is honest, and has their share.

Improving on this

What operations does Enc do?

Improving on this

What operations does Enc do?

- $$P := \mathrm{sk}_i \cdot \mathrm{pk}_j$$

Improving on this

What operations does Enc do?

- $P := \mathrm{sk}_i \cdot \mathrm{pk}_j$
- $K := H(P)$

Improving on this

What operations does Enc do?

- $P := \mathrm{sk}_i \cdot \mathrm{pk}_j$
- $K := H(P)$
- $\mathrm{AES}\big(K, s_{ij}\big)$?

Improving on this

What operations does Enc do?

- $P := \mathrm{sk}_i \cdot \mathrm{pk}_j$
- $K := H(P)$
- $\mathrm{AES}\big(K, s_{ij}\big)$?
- instead, just do $K \in \mathbb{F}$, and do $K + s_{ij}$

Draft Two

$$\mathrm{Exch} : (\mathrm{Sk}, \mathrm{Pk}) \to \mathbb{F}$$

$$f_i \xleftarrow{\$} \mathbb{F}[X]_{\leq t}$$

$$F_i := f_i \cdot G$$

$$m_{ij} := \mathrm{Exch}\big(\mathrm{sk}_i, \mathrm{pk}_j\big)$$

$$M_{ij} := m_{ij} \cdot G$$

$$c_{ij} := m_{ij} + f_i\big(p_j\big)$$

$$\pi_{ij} := \mathrm{Prove}\big(\mathrm{sk}_i, \mid m_{ij} = \mathrm{Exch}\big(\mathrm{sk}_i, \mathrm{pk}_j\big) \wedge m_{ij} \cdot G = M_{ij}\big)$$

then broadcast M_{ij}, π_{ij}, c_{ij}

Players check $c_{ij} \cdot G \stackrel{?}{=} M_{ij} + F_i\big(p_j\big)$

One Trick

In some proof systems (Bulletproofs), something like:

$$\{x_1, ... \mid \text{stuff}(x_1, ...) \wedge \text{x\_i dot.c G} = \text{X\_i}\}$$

is partially free.

Intuition: ZK Proof = Checks + Commitment Scheme

$x_i \cdot G$ commits to x_i.

One Trick

In some proof systems (Bulletproofs), something like:

$$\{x_1, \ldots \mid \text{stuff}(x_1, \ldots) \wedge \text{x\_i dot.c G} = \text{X\_i}\}$$

is partially free.

Intuition: ZK Proof = Checks + Commitment Scheme

$x_i \cdot G$ commits to x_i.

Hence:

$$\text{Prove}\big(\text{sk}_i \mid m_{ij} = \text{Exch}\big(\text{sk}_i, \text{pk}_j\big)\big)$$

Another Trick

In many schemes (again Bulletproofs), proof size << statement size, e.g. $O(n \lg n)$

If we have n proofs, maybe...

Another Trick

In many schemes (again Bulletproofs), proof size << statement size, e.g. $O(n \lg n)$

If we have n proofs, maybe…

prove them all together, rather than one at a time.

Another Trick

In many schemes (again Bulletproofs), proof size << statement size, e.g. $O(n \lg n)$

If we have n proofs, maybe...

prove them all together, rather than one at a time.

We can also verify n proofs much faster than one proof n times.

A Final Trick

We have $\mathbb{G}$ with scalars $\mathbb{F}$.

Our computation is in $\mathbb{F}$.

We want $\mathrm{Exch}\big(\mathrm{sk}_i, \mathrm{pk}_j\big)$. What group to use?

A Final Trick

We have $\mathbb{G}$ with scalars $\mathbb{F}$.

Our computation is in $\mathbb{F}$.

We want $\mathrm{Exch}\left(\mathrm{sk}_i, \mathrm{pk}_j\right)$. What group to use?

- We want a group $\mathbb{H}$ where operations are arithmetic over $\mathbb{F}$.

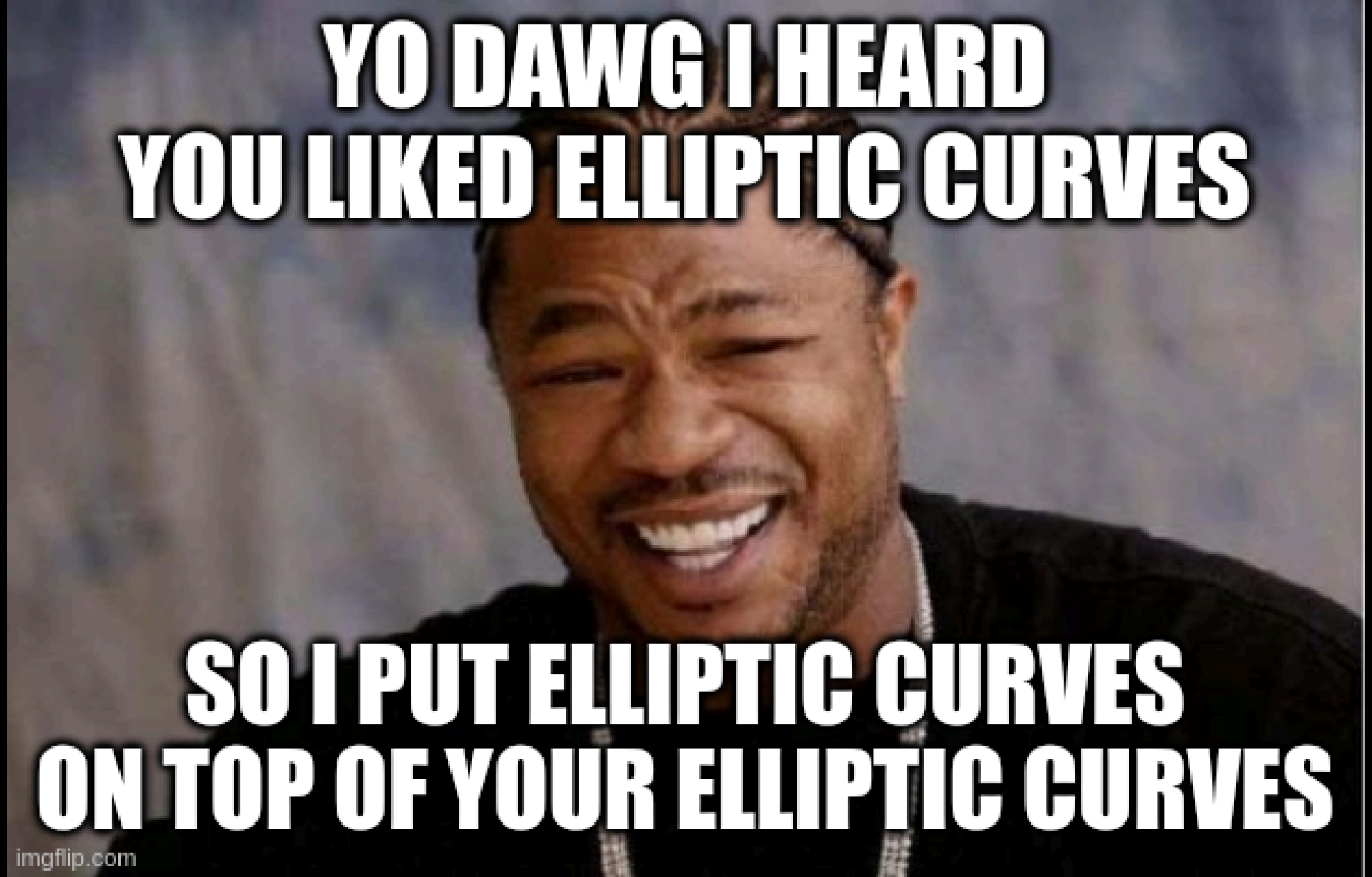
A Final Trick

We have $\mathbb{G}$ with scalars $\mathbb{F}$.

Our computation is in $\mathbb{F}$.

We want $\text{Exch}\left(\text{sk}_i, \text{pk}_j\right)$. What group to use?

- We want a group $\mathbb{H}$ where operations are arithmetic over $\mathbb{F}$.
- An Elliptic Curve with coordinates in $\mathbb{F}$ can be found.


YO DAWG I HEARD
YOU LIKED ELLIPTIC CURVES
SO I PUT ELLIPTIC CURVES
ON TOP OF YOUR ELLIPTIC CURVES
imgflip.com

Examples of this trick

- Bandersnatch

Examples of this trick

- Bandersnatch
- Balthizard

Examples of this trick

- Bandersnatch
- Balthizard
- Baby Jub Jub

Examples of this trick

- Bandersnatch
- Balthizard
- Baby Jub Jub
- Werewerewire

Examples of this trick

- Bandersnatch
- Balthizard
- Baby Jub Jub
- Werewerewire
- Organikk

Examples of this trick

- Bandersnatch
- Balthizard
- Baby Jub Jub
- Werewerewire
- Organikk
- Pasta curves

Should we always use the one-round DKG?

Two Example Environments

Institutional Custody

- Offline signing, communication over group chat,
- air-gapped secret-share storage,
- completely private operation,
- DKG happens once.

Validator Certificate Key

- DKG happens often,
- completely online and connected,
- convenient public blockchain to post information.

Some cons of the one-round DKG

- Latency might be a lot worse (minutes of computation for 100 people).
- Bugs in ZK circuit.
- Implementing broadcast might take two rounds anyways.

Reshare

What if we want to keep the public key X, but change the shares?

Reshare

What if we want to keep the public key X, but change the shares?

$$f \xleftarrow{\$} \mathbb{F}[X]_{\leq t}$$

$$f(0) := x$$

Reshare

What if we want to keep the public key X, but change the shares?

$$f \xleftarrow{\$} \mathbb{F}[X]_{\leq t}$$

$$f(0) := x$$

$$f_i \xleftarrow{\$} \mathbb{F}[X], f_{i(0)} := x_i$$

$$? \sum_{i \in \text{ good}} f_i ?$$

Reshare

What if we want to keep the public key X, but change the shares?

$$f \xleftarrow{\$} \mathbb{F}[X]_{\leq t}$$

$$f(0) := x$$

$$f_i \xleftarrow{\$} \mathbb{F}[X], f_{i(0)} := x_i$$

$$? \sum_{i \in \text{ good}} f_i ?$$

$$\left(\sum_{i \in \text{ good}} \lambda_i f_i \right)(0) = \sum_{i \in \text{ good}} \lambda_i x_i = x$$

SECTION 16

Schnorr Threshold Signatures

Recap

$$
\begin{aligned}
k &\xleftarrow{\$} \mathbb{F} \\
K &:= k \cdot G \\
c &:= H(X, K, m) \\
s &:= \mathrm{k} + c\ \mathrm{x}
\end{aligned}
$$

What tools can we use to do this in a threshold setting?

Recap

$$
\begin{aligned}
k &\xleftarrow{\$} \mathbb{F} \\
K &:= k \cdot G \\
c &:= H(X, K, m) \\
s &:= \mathrm{k} + c\ \mathrm{x}
\end{aligned}
$$

What tools can we use to do this in a threshold setting?

How to do $\mathrm{k} \xleftarrow{\$} \mathbb{F}$ jointly?

Recap

$$
\begin{aligned}
k &\xleftarrow{\$} \mathbb{F} \\
K &:= k \cdot G \\
c &:= H(X, K, m) \\
s &:= \mathrm{k} + c\ \mathrm{x}
\end{aligned}
$$

What tools can we use to do this in a threshold setting?

How to do $\mathrm{k} \xleftarrow{\$} \mathbb{F}$ jointly?

How to do $\mathrm{k} + c\ \mathrm{x}$ jointly?

“Lazy” Threshold Schnorr

$$(X, x_i) := \mathrm{DKG}()$$

———

$$(K, k_i) := \mathrm{DKG}()$$

$$c := H(X(0), K(0), m)$$

$$s_i := k_i + cx_i$$

$$s_i \rightrightarrows$$

$$s_i \stackrel{?}{=} K(p_i) + cX(p_i)$$

$$s := \sum_i s_i$$

$$(K, s)$$

DKG Component Tweaks

DKG Component Tweaks

- Should we use the two-round or one-round DKG?

DKG Component Tweaks

- Should we use the two-round or one-round DKG?
- What if we use a linear secret sharing DKG?

Inlined Two-Round DKG

Attempt:

- Do a DKG for k, and a reshare for x,
- You receive x_{ij} and k_{ij}, compute s_{ij}, send that.
- Use the complaints to compute:

$$K := \sum_{j \in \text{ good}} \lambda_j K_j(0)$$

$$s := \sum_{i,j \in \text{ good}} \lambda_i \lambda_j s_{ij}$$

Does this work?

SECTION 17

Threshold ECDSA

The Annoying Algorithm

$$k \xleftarrow{\$} \mathbb{F}$$

$$R := \left(\frac{1}{k}\right) G$$

$$s := k(m + h(R)x)$$

$$(R, s)$$

tricky part:

$$s = \mathrm{k}\ m + h(R)\ \mathrm{k}\ \mathrm{x}$$

Multiplication is hard, so do it in advance

If a trusted third party gives us $a(p_j), b(p_j), c(p_j) s.t. a(0)b(0) = c(0)$, can we use that?

Multiplication is hard, so do it in advance

If a trusted third party gives us $a(p_j), b(p_j), c(p_j) s.t. a(0)b(0) = c(0)$, can we use that?

$$(k + a)x - a(x + b) + c$$

$$kx + ax - ax - ab + c$$

$$kx$$

if we use a and b just once, then $k + a$ hides k, and $x + b$ hides b.

Idea: compute $k + a$, $x + b$ secret shared, then "open" the values. With $k + a$ and $x + b$ public, we can compute kx secret shared.

Inversion?

We also need to compute $\left(\frac{1}{k}\right)G$.

If we generate a mask $d \xleftarrow{\$} \mathbb{F}$, then we can safely reveal kd, and dG

Inversion?

We also need to compute $\left(\frac{1}{k}\right)G$.

If we generate a mask $d \xleftarrow{\$} \mathbb{F}$, then we can safely reveal kd, and dG

then, make kd public, letting us compute

$$\left(\frac{1}{k}d\right)(dG) = \left(\frac{1}{k}\right)G$$

Methods to compute Beaver Triples (the hard part)

- Paillier Encryption and fancy ZK Proofs

  (bad)

- Oblivious Transfer

  (also bad, unfortunately)

ECDSA

- ✗ SLOW
- ✗ COMPLICATED
- ✗ NOT LINEAR
- ✗ WEIRD NONCE (uh oh bad nonce)
- ✗ HARD TO COMBINE

SCHNORR

- ✓ FAST
- ✓ SIMPLE
- ✓ LINEAR
- ✓ SAFE NONCES
- ✓ EASY TO COMBINE

HERE'S A KROMER KID, $
NOW GET YOURSELF A
REAL SIGNATURE SCHEME!!!

SECTION 18

Some Questions to Ponder

- Do we need the message before starting threshold signing?
- Can we amortize signing multiple things at once?
- Can we pipeline these protocols?
- Do post-quantum schemes work this nicely?

Thank You!