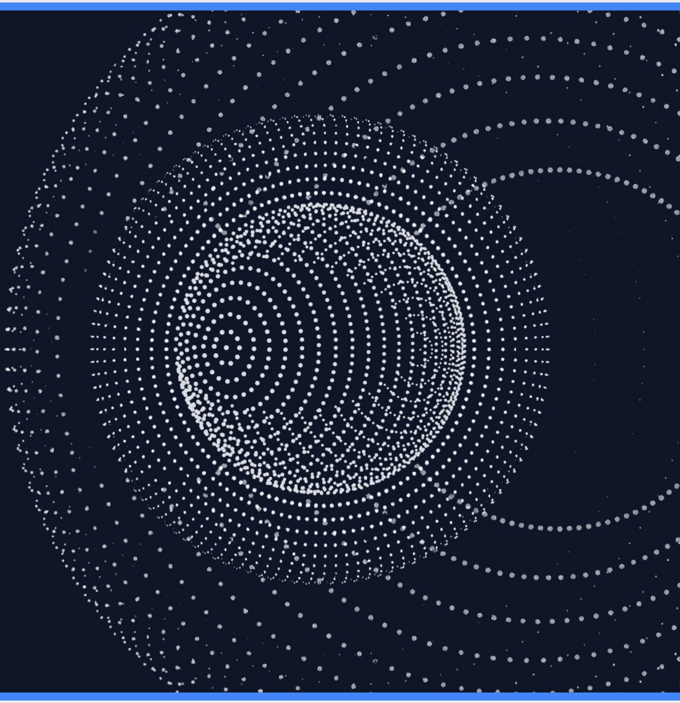
Fairgate

WISCH

An efficient data signing scheme via correlated signatures



Ramses Fernandez | Cryptographer, Fairgate Labs

bitcoin++ Istanbul, SEPT 2025

This is a joint work with

Ariel Futoransky

Gabriel Larotonda

Sergio Demian Lerner

Fairgate

Structure

The presentation has been divided into the following sections:

INTRODUCTION

WISCH

Fairgate

What is WISCH?

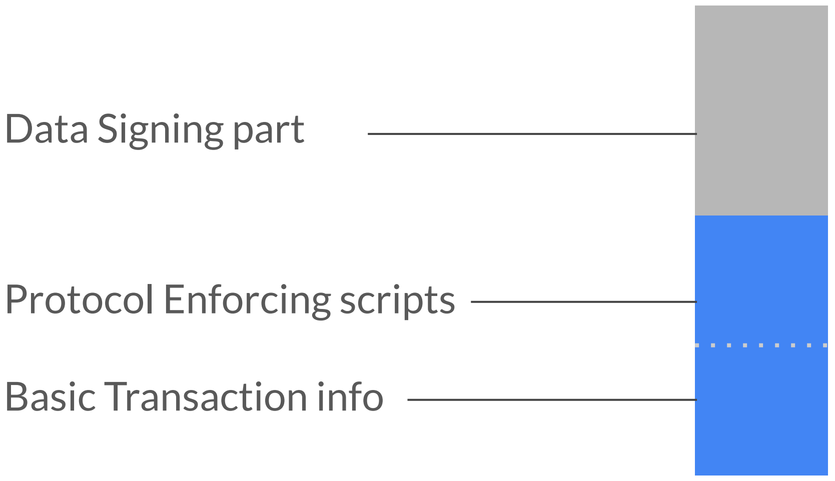
WISCH is a protocol that creates correlated sets of signatures between different signing schemes or messages.

It can be used to reduce the on-chain data signing costs by 4x compared to Winternitz and 8x compared to Lamport and GC Wire labels.

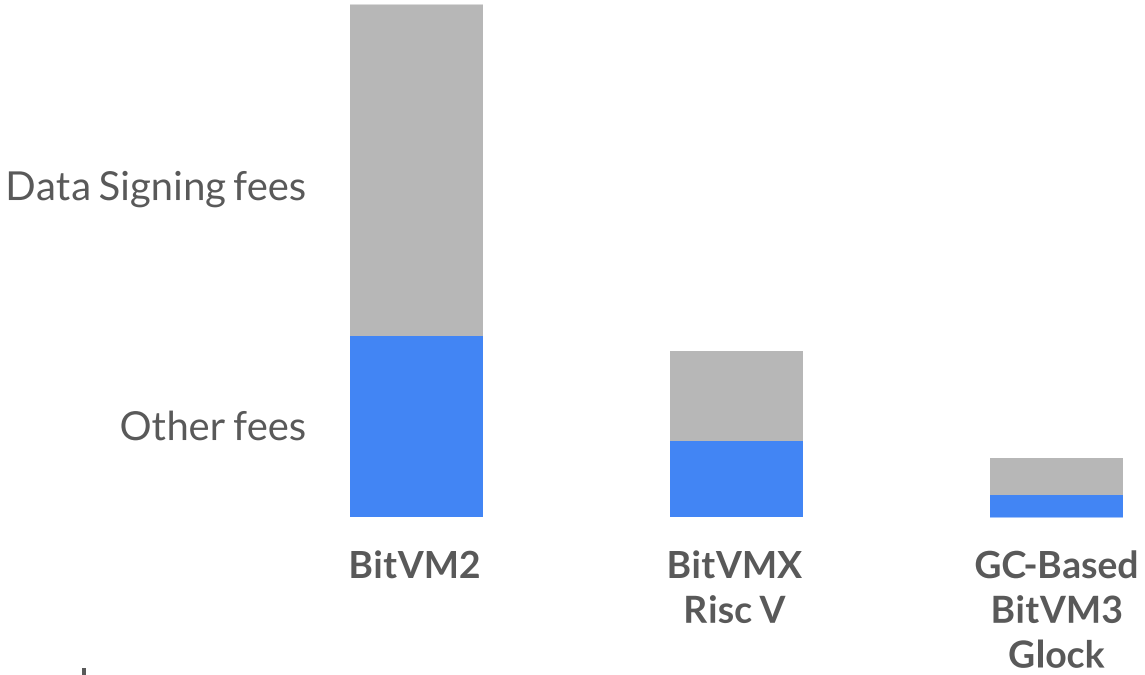
Disputable Computing and Protocol Fees

The on-chain cost of disputable computing protocols on bitcoin is significant

Protocol Fees



Protocol Fees



* not to scale

Measuring Efficiency

Blow-out factor:

Weight units per byte. WU/Byte

Lamport Signatures	≈ 400 WU/Byte
Winternitz Signatures	≈ 200 WU/Byte
Wisch	≈ 50 WU/Byte

Fairgate

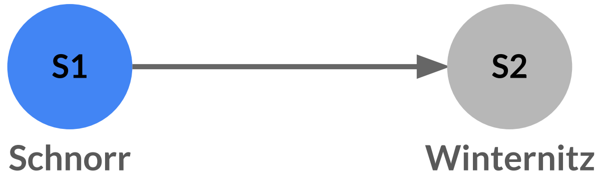
PROTOCOL DESCRIPTION

WISCH

Fairgate

General Idea

Connect signature schemes: for example **WI**nternitz with **SCH**norr



A setup phase allows off-chain processing to recover signatures from S2 by using signatures from S1 as decryption keys

The on-chain cost is the cost of signature verification on bitcoin for the source signature: schnorr verification

Overview

Bob will like to let Alice sign n-bits of data for a Bitcoin Protocol.

Wisch will generate connected pairs of signatures $S1_i$, $S2_i$ for i in $\{1..2^n\}$

Each signature in S1 is tied to TX1, a bitcoin single transaction with multiple nonces and a timelock alternate path.

S2 can be a set of Lamport, Winternitz, or additional Schnorr.

It can also be a set of binary or arithmetic labels for a garbled circuit input wires.

Setup

1. Truncated Musig2

   Only Alice gets $S1_i$ from an aggregated Alice-Bob key. Bob only gets the associated nonces.

2. Alice generates $S2_i$
3. Alice generates $CS2_i=E_{S1i}(S2_i)$
4. Alice creates ZKP, a proof that CS2 is correct
5. Bob Verifies the ZKP, given TX1, CS2 and public key information about S2
6. They proceed to the creation of transaction or circuits as required by the protocol.

Fairgate

Signing

1. Alice chose i as her data element, and uses the associated $S1_i$ to sign and broadcast transaction TX1
2. Bob sees TX1 signed with $S1_i$ published on bitcoin
3. Bob decrypts $CS2_i$ using $S1_i$ as key, recovering $S2_i$

   $S2_i = Dec_{S1i}(CS2_i)$

4. Bob can now use $S2_i$ on other transactions or circuits

PROTOCOL DETAILS

Truncated Musig2

A Musig2 protocol variant with unilateral aggregation, only Alice will obtain the aggregated signatures

Bob will get the aggregated nonce

Sequence Diagram

Encryption Table

For example, with S1 Schnorr, S2 Garbled wires, and n=2

$W = \{ W_{0,0} W_{0,1} W_{1,0} W_{1,1} \}$

$S1 = \{ S1_0 \; S1_1 \; S1_2 \; S1_3 \}$

$S2 = \{ (W_{0,0} \; W_{1,0}) (W_{0,0} \; W_{1,1})(W_{0,1} \; W_{1,0}) (W_{0,1} \; W_{1,1}) \}$

CS2 =

$E_{S10}(W_{0,0} | W_{1,0})$

$E_{S11}(W_{0,0} | W_{1,1})$

$E_{S12}(W_{0,1} | W_{1,0})$

$E_{S13}(W_{0,1} | W_{1,1})$

Fairgate

Zero Knowledge Proof

Standard hash functions for Lamport / Winternitz for S2 and W

ZK friendly hashing for S1

ZK friendly encryption for S2 and CS2

Given encryption table CS2, Public Keys for W, Seed, and aggregated signatures (AS, AR)

Verifies CS2 is correctly formed, Pubkeys are correct, and aggregated schnorr signature for S1 = AS * X_i, AR * X_i and coefficients X_i = Hash(seed, i)

Multi-signature verification script

A single script can verify multiple signatures

Public key is reused

Amortized cost per signature is ≈80 WU

Decryption

Bob sees Alice's signature $S1_i$ on TX1 Bitcoin

Bob uses the signature nonce to efficiently locate the associated ciphertext on the encryption table. $CS2_i$

CONCLUSIONS

Security

Let’s add the security assumptions here

Performance

Signature cost is independent of N

Blow-out is in $O(1/2^n)$

Communication is in $O(2^n)$

The sweet spot for n is in the 10-20 range.

Using n=12 and signing 144 bytes we get:

#Musig2 executions = 393216

Transaction Size = **7769 WU**

Blow-out factor = **53.95**

Thank You!

Fairgate

www.fairgate.io

https://github.com/FairgateLabs

BitVMX

https://bitvmx.org