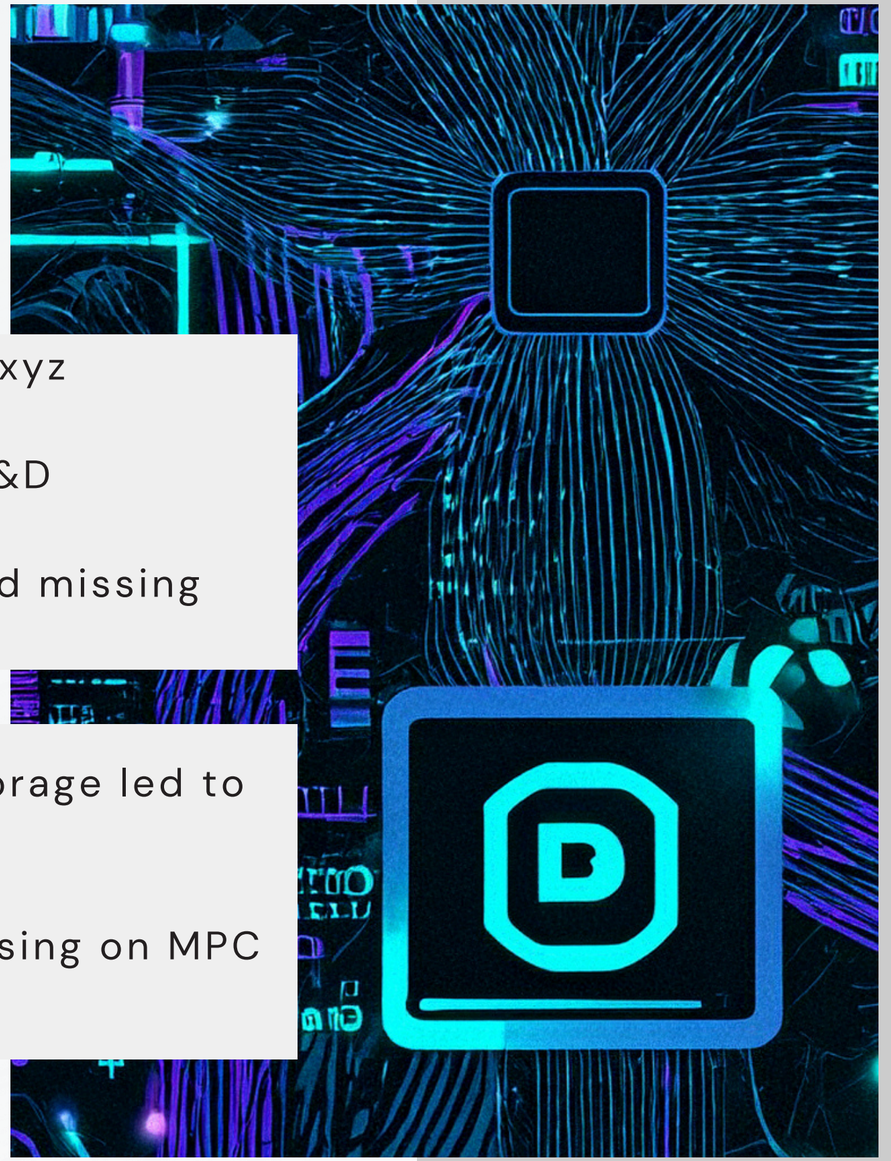
Intu Labs

MULTIPARTY COMPUTATION + CONFIDENTIAL COMPUTE

THREAT MODELING AND OPPORTUNITIES IN DECENTRALIZED ARCHITECTURES

INTRO



James Bourque, CEO + Co-Founder at INTU.xyz

2016 Blockchain Hobbyist, 2019 Full-time R&D

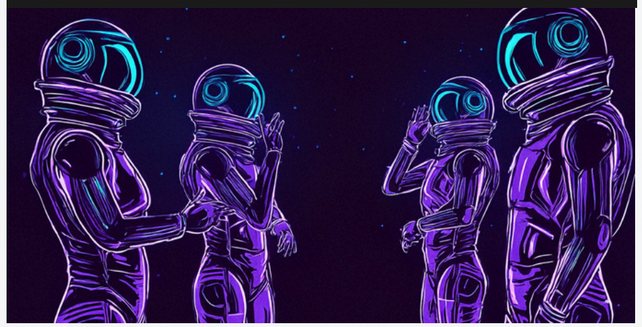
Focus on cryptography and security to build missing Web3 Infrastructure

Research into Decentralized Private Key Storage led to Private Key Generation and Management

We began with TEEs in 2019, ended up focusing on MPC in 2021

WHAT IS MPC?

Multiparty Computation

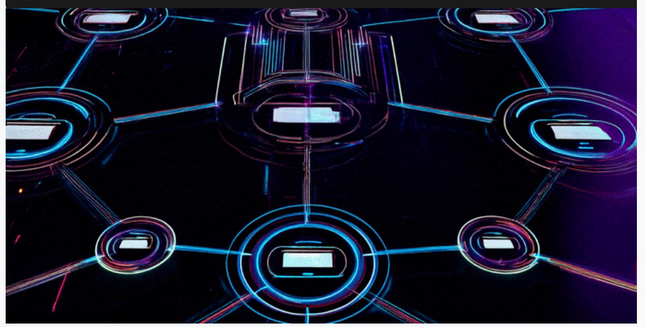


Multiple devices, typically controlled by different parties, participating collectively in computation, in an effort to more safely generate keys and signatures, minimizing opportunities for single point of failure.

Protocol is typically Leaderless, Trustless, and with a lower technical requirement.

- Focus on key generation and signatures, mostly ECDSA/EDDSA
- Rely heavily on local compute, but no single machine ever does 100% of compute
- Not for general computation - No Threshold Encryption

For Blockchains



MPC + HSM: SUBSTITUTES OR COMPLIMENTS?

In the limited context of key generation and signing for decentralized architecture, we had originally considered MPC as an alternative to HSM.

Ultimately, we prioritized preserving Web3 principles, which led us to focus on MPC.

As we approach challenges in scale or more demanding security environments, we recognize a lot of benefits to using both CC + MPC.

We will examine these complimentary opportunities through the context of threat modeling.

HSM

Confidential Compute is more mature and sophisticated than MPC.

Can achieve more in a greater variety of applications:

TEEs, TPMs, HSM

Networks, Enterprise-Grade solutions, etc

MPC

MPC mostly exists as permissioned networks, which often offer additional hardware security.

Secure MPC is typically privacy-preserving, sharing only outputs.

Typically slower than hardware-based confidential compute

DEFINING DECENTRALIZED ARCHITECTURE

*Blockchain *ideally**

P2P network with consensus mechanism, incentivized to encourage decentralization and greater distribution, minimizing single points of failure.

Think 2016 Ethereum as “Global Computer”, not Bitcoin as Digital Cash.

Ethereum has EVM, which is extremely limited in terms of compute, storage, and access management compared to cloud

Ideologically, driven by a community who felt there was an over-concentration of resources in cloud-based architecture, and the internet would benefit from more public, permissionless resources.

Emphasis on autonomy, privacy, transparency, and resilience, especially from centralized powers/organizations

Blockchain Ideals

HOW MPC WORKS

01 VERIFIED SECRET SHARING

02 CREATE POOL OF QUALIFYIED PARTICIPANTS

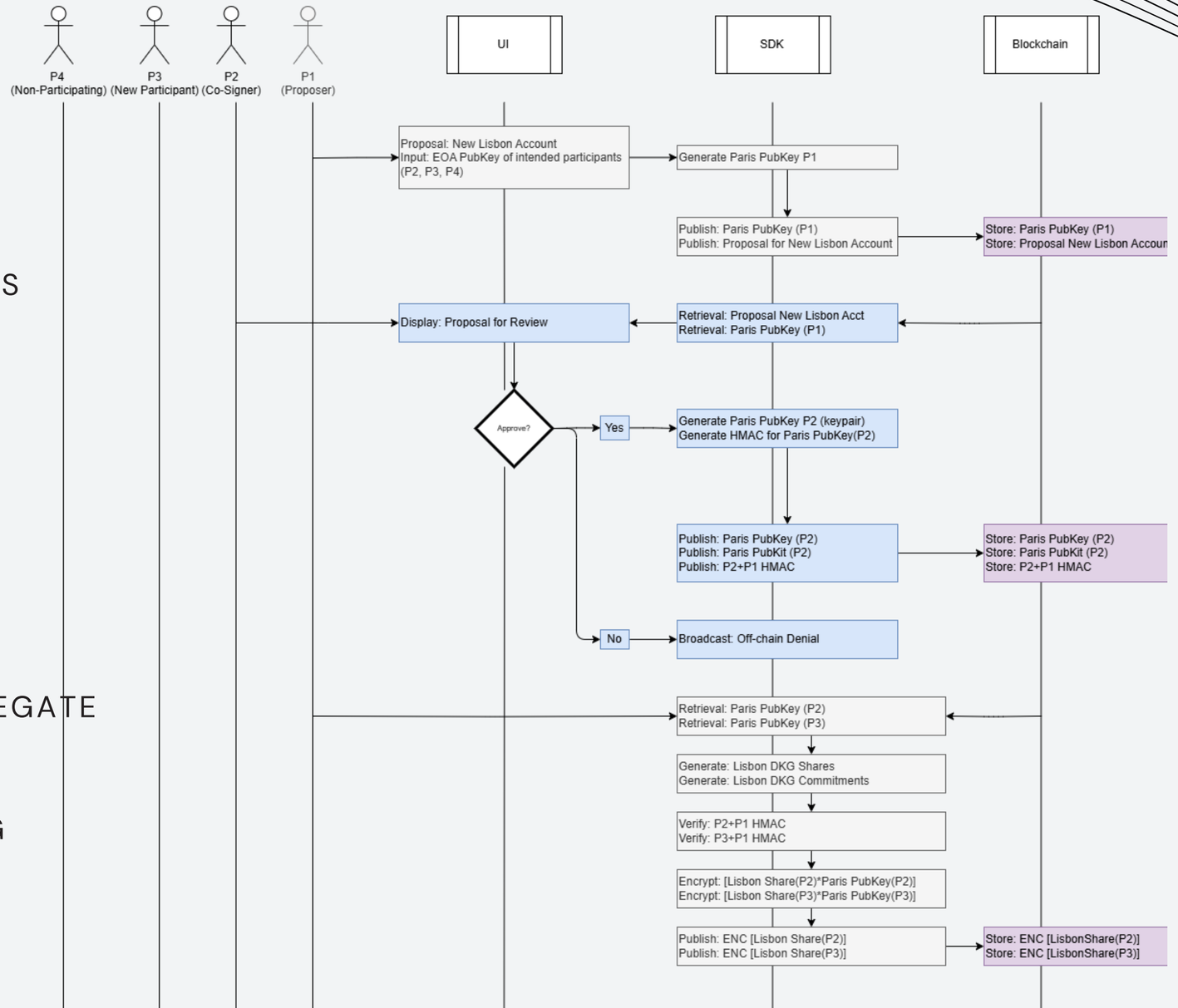
03 LOCAL COMPUTATION, SHARE OUTPUT

04 VERIFY ZK-PROOFS OTHERS SHARED

05 DISPUTE/RESOLUTION PHASE, AS NEEDED

06 QUALIFIED PARTICIPANTS COMPUTE, AGGREGATE FINAL SHARED PUBLIC ADDRESS

07 TO SIGN, SIMILAR PROTOCOL FOR FORMING THRESHOLD SIGNATURES



MPC IS NOT A SILVER BULLET

Early in development, we brought on external auditor for collaborative threat modeling.

Simplified, hybrid of STRIDE and DREAD methodologies in a more open format.

Focused on three main areas:

- General Blockchain Threats (~25)
- Decentralized Key Generation: Account Set-up and Creation (~34)
- Threshold Signatures: Using the account (~12)

GENERAL BLOCKCHAIN THREATS

- Included broad threats beyond our control, including forks, Sybil Attacks, 51% attacks
- Some threats are blockchain specific, like smart contract code, memory pools, ordering, which can be mitigated
- Many are just “internet” problems - Https/DNS, DDoS, Clickjacking, etc- base threat model

Credentials - especially at scale - need to be managed securely and efficiently.

Single point of failure with massive, one to many, often for core infrastructure.

RISK

GitHub Credentials for a popular library Web3.JS for Solana were compromised.

Updates were pushed which included malicious code to steal private keys.

EXAMPLES

Key Management Systems with hardware security are not widely used in Web3 circles. Opportunity for education on hardware security and benefits of confidential compute.

OPPORTUNITY

DKG: SET-UP AND CREATION

Distributed Key Generation using multiparty computation requires careful set-up of communication and protocol, which allows each participant to perform their local cryptographic computations.

- DKG is the most computation-intensive part of the protocol
- Most Risk, Broadest attack surface. Compromised DKG is not immediately obvious, risks digital assets

Compromised local device, i.e. Malware on PC, can compromise the DKG.

Includes bad computation, exposed secret storage, and bad randomness.

RISK

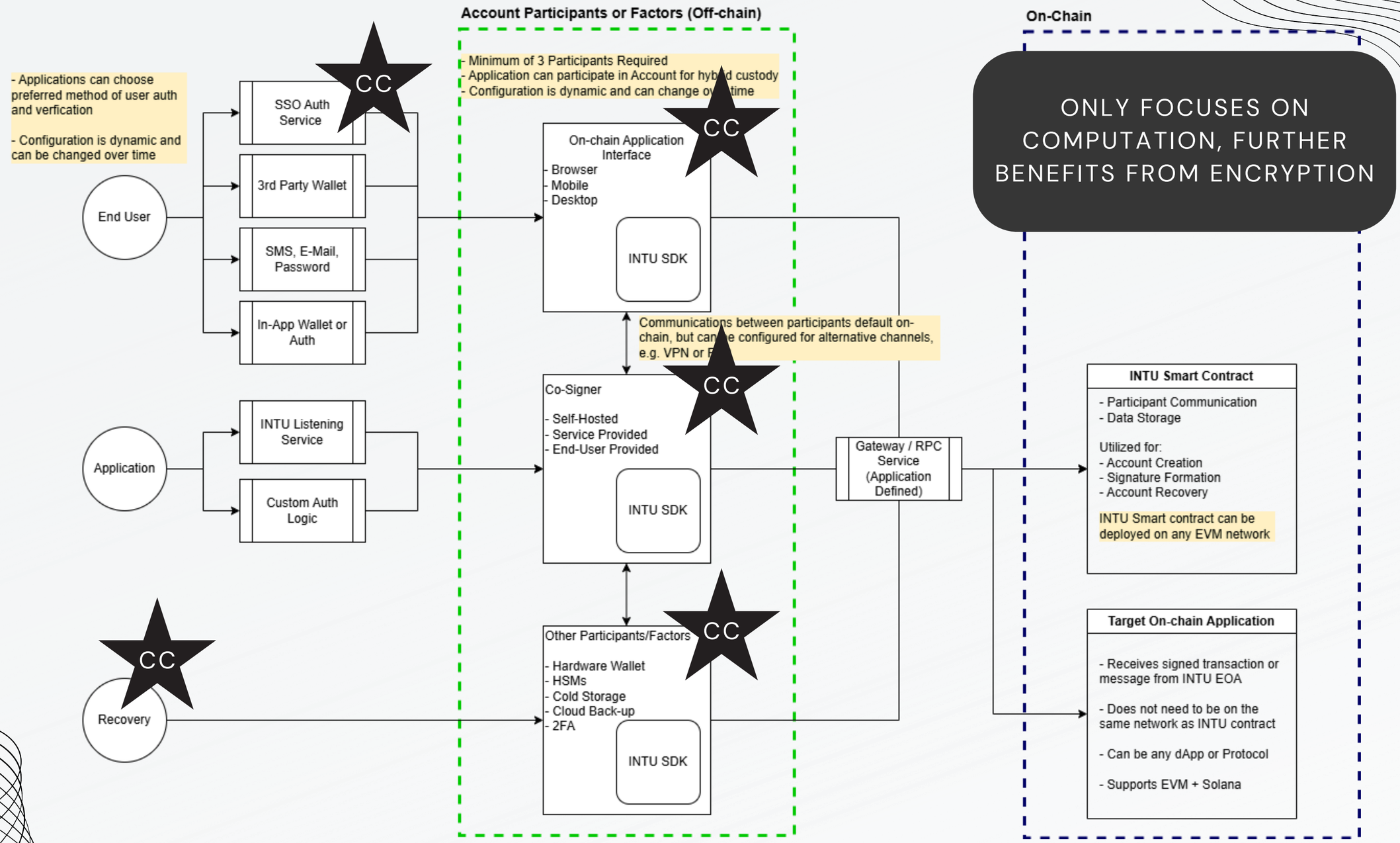
A compromised machine for P1 Proposer injects bad randomness, resulting in poor encryption for secret storage and sharing. At low threshold, compromise entire protocol.

EXAMPLES

By performing all computation and secret storage on TEEs, and using TEE's randomness, ensure safe DKG, as well as future resharing of key material.

OPPORTUNITY

CC THROUGHOUT INTEGRATION



TSS: SIGNING USING MPC

With no private key, all signatures from the MPC Public address must follow a Threshold Signature Scheme (TSS). A defined threshold of participants can contribute partial signatures, which when aggregated, form a valid signature for the key pair, with no special verification.

- Greatest risk in terms of value - enables the transfer of digital assets, and transfer of account ownership
- Substantial computational risk, as key material is used directly in a variety of signature schemes and processes
- Includes manipulation of conditions: time, deadlines, protocol ordering, etc

Manipulated protocols and signing schemes enable impersonation of participants, front running, misordering of deadlines

RISK

If time, ordering, or deadlines are manipulated, participants could submit partial signatures for incomplete proposals, resulting in lost assets or accounts.

EXAMPLES

Running verified protocols and libraries within a confidential compute environment ensures protocol conditions are followed correctly.

OPPORTUNITY

SO...
DOES CC INCREASE OR REDUCE ATTACK SURFACE?

PRO

Lower risk of compromised local device, including:

- Malware
- Side Channel Attacks
- Human Error

Protections against hypervisor attacks

Better verifiable randomness for both encryption and key generation

Secure execution of critical libraries

Greater support for encryption at scale

CON

Centralized attestation “Proof of Intel”

Supply Chain Hacks, e.g. MSI hack

Limitations for provisioning at scale

TEEs make protocol device-specific, sharing blobs reliably is limiting

Limited support for cryptographic schemes

COLLABORATION: CC + MPC

Confidential Compute dramatically minimizes computational and storage risk of using local devices, especially “in the wild”

Confidential Compute enables greater controls for MPC, including protocol verification, improved randomness, and protections against manipulated conditions

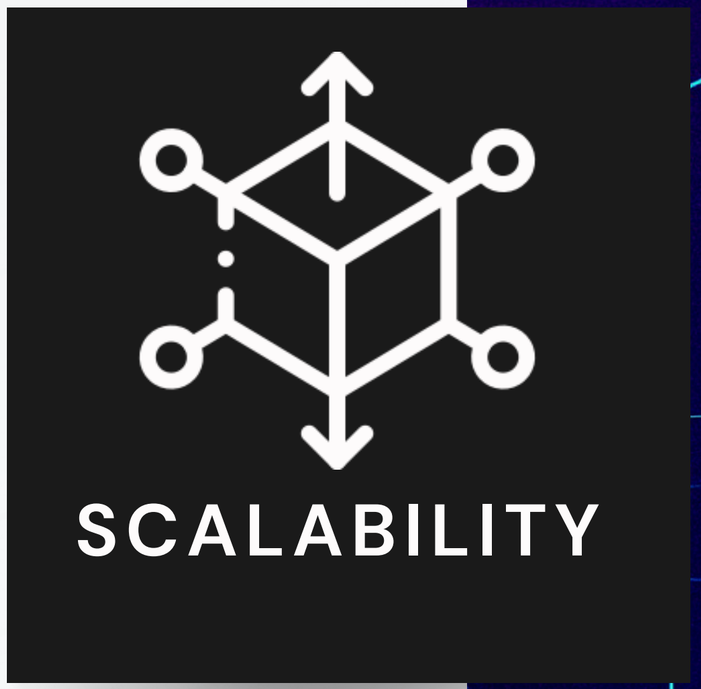
CC is more accessible than ever...TPMs, Virtual TEEs, CC Services and networks

COLLABORATION: CC + MPC

Credential Management at scale is a major risk and represents a one-to-many point of failure for decentralized architectures

Provisioning at scale for participation in DKGs and TEEs is only possible with KSMs + CC.
With MPC, 1 action = 2+ signatures...

Enables greater quantity of available participants for signing, increasing speed and efficiency of keys without compromising on security

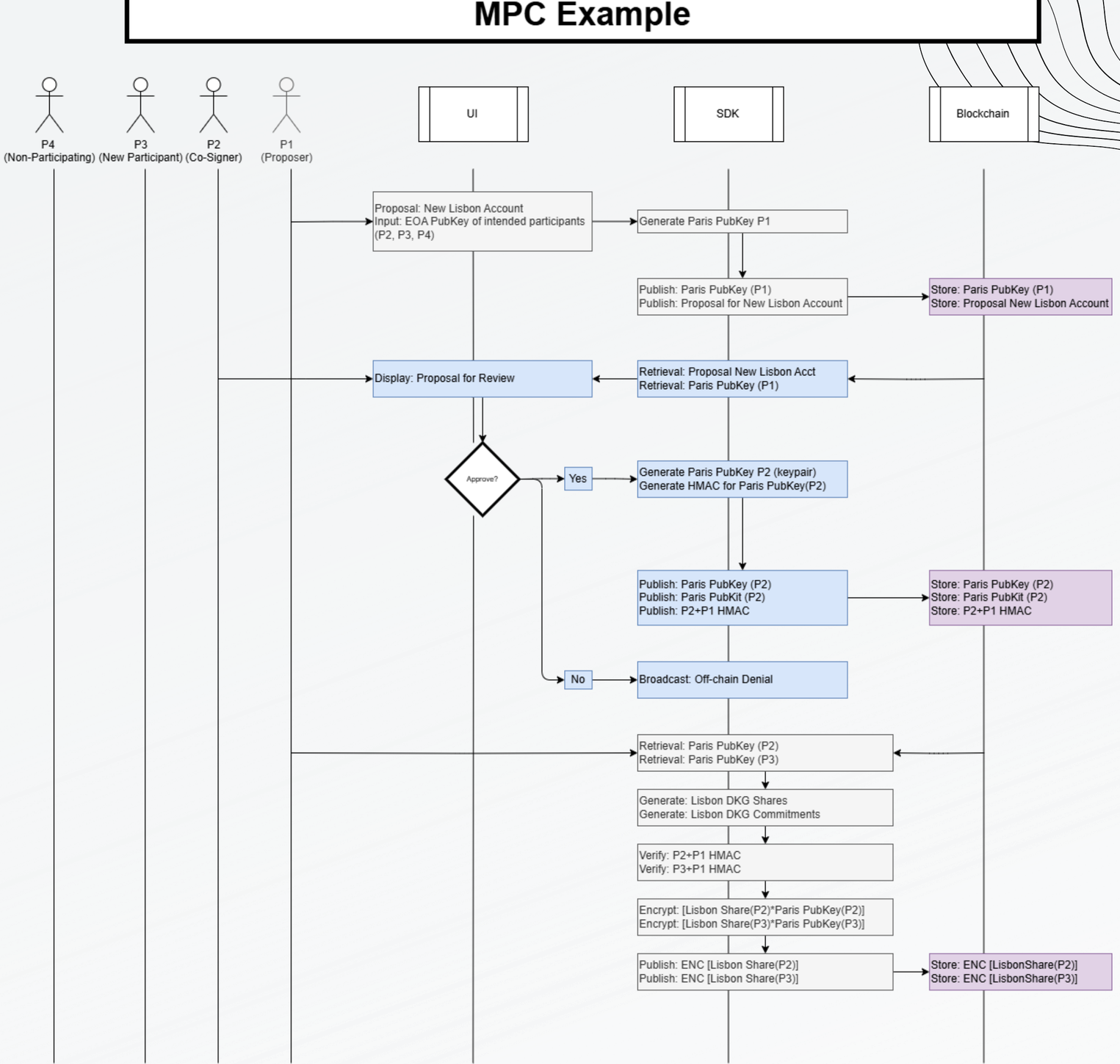


THANKS!

James Bourque, CEO
James@intu.xyz

Intu Labs

MPC


MPC Example
P4
(Non-Participating)
P3
(New Participant)
P2
(Co-Signer)
P1
(Proposer)
UI
SDK
Blockchain
Proposal: New Lisbon Account
Input: EOA PubKey of intended participants
(P2, P3, P4)
Generate Paris PubKey P1
Publish: Paris PubKey (P1)
Publish: Proposal for New Lisbon Account
Store: Paris PubKey (P1)
Store: Proposal New Lisbon Account
Display: Proposal for Review
Retrieval: Proposal New Lisbon Acct
Retrieval: Paris PubKey (P1)
Approve?
Yes
Generate Paris PubKey P2 (keypair)
Generate HMAC for Paris PubKey(P2)
Publish: Paris PubKey (P2)
Publish: Paris PubKit (P2)
Publish: P2+P1 HMAC
Store: Paris PubKey (P2)
Store: Paris PubKit (P2)
Store: P2+P1 HMAC
No
Broadcast: Off-chain Denial
Retrieval: Paris PubKey (P2)
Retrieval: Paris PubKey (P3)
Generate: Lisbon DKG Shares
Generate: Lisbon DKG Commitments
Verify: P2+P1 HMAC
Verify: P3+P1 HMAC
Encrypt: [Lisbon Share(P2)*Paris PubKey(P2)]
Encrypt: [Lisbon Share(P3)*Paris PubKey(P3)]
Publish: ENC [Lisbon Share(P2)]
Publish: ENC [Lisbon Share(P3)]
Store: ENC [LisbonShare(P2)]
Store: ENC [LisbonShare(P3)]