Privacy-Preserving Smart Contracts

Ranjit Kumaresan
Microsoft Research

Joint work with Iddo Bentov, Tal Moran, Vinod Vaikuntanathan, Prashant Vasudevan, Melissa Chase
Smart Contracts
[Szabo’94]

Automate Contract Management

Self-enforcing

Key unlocks car iff monthly installment is paid
Difficulties in Self-Enforcement

Information Exchange

Want: Either both get each other’s information or none get

⚠️ Counter-party risks ⚠️
Mitigating Counter-Party Risks

Escrow Services
Performs exchange iff both conditions are satisfied

Are escrows really necessary?

Impossible to enforce without trusted party

[Cleve86]
Bypass Cleve’s Impossibility?

Information has value

Money Escrow

If exchange is unsuccessful honest party is compensated

Safety deposit = value of information
Why Is This Any Better?

Possible to implement in a model where we have self-enforcing information ↔ money exchange [BK’14a]

Claim-or-Refund Transactions

Bob claims Alice’s deposit and Alice gets Bob’s information

OR

Bob does not claim Alice’s deposit and Alice gets her deposit refunded
Bitcoin Implements Information $\iff$ Money!

$\phi = \text{Knowledge of secret key}$

Blockchain: Public list of transactions

TX1: Alice -> Bob
TX2: Carol -> Dave
TX3: Bob -> Carol

Bitcoin transactions are explicitly conditional
To spend TX1: Bob produces a signature under his public key

Miners validate conditions and maintain the blockchain
Bitcoin-Inspired but Bitcoin-Independent

Privacy

- Single Point of Failure
- Data breaches
- Fee per transaction
- Depends on complexity
- Breaks consensus
- Miner dis-incentivized
- Network delays

- Need Privacy-Preserving Mechanisms

Scalability

- Need to Minimize On-Chain Complexity

Information ↔ Money

Escrow

Blockchain Implementation

Made public on-chain for verification anyway

Miners need to see Fee per transaction

Large \( \phi \) breaks consensus

- Miner dis-incentivized
- Network delays

Need to Minimize On-Chain Complexity
Information $\Leftrightarrow$ Information via Claim-or-Refund [BK’14]

Want to Implement

Main Idea: Double lock

$\phi$ and $\phi$

Parties end with up each other’s information and coins
Information $\iff$ Information via Claim-or-Refund [BK’14]

Want to Implement

What if Bob does not make a deposit

Bob does not have both information! Alice does not lose money!
Information $\iff$ Information via Claim-or-Refund [BK'14]

Want to Implement

What if Bob does not claim?

No Privacy!

are public on the blockchain
How to Get Privacy?

[Need Privacy-Preserving Computations]

Want to compute $f(x,y)$ without revealing any extra information about their inputs

_Cleve’s result:_ Not possible even for $f(x,y) = (y,x)$
Want to compute $f(x,y)$ without revealing *any* extra information about their inputs.

Secure Computation Protocol

$z = f(x,y)$

```
x
| πf(y)
|        z
|        z
| y
```

Best Possible Privacy

No “fairness”

SECURE COMPUTATION “implements” the angel via cryptography!

[Yao’86,GMW’87,BGW’88,CCD’88,RB’89]
Private Exchange via Secure Computation

[BK’14, KVV’16]

Earlier:

Privacy Problem
Conditions $\phi$ and $\phi$ check information publicly

New:

Secure Computation

Generate encryption keys
Check if information is valid, then encrypt it
Authenticate keys $\phi_K$ and $\phi_K$

Privacy!

Scalability!

Only the one-time keys go public
Encrypted information kept local
Multiple-Move Exchanges

[KMB’15, KVV’16]

\(M_i\) depends on \(M_1, \ldots, M_{i-1}\)

\(M_1, \ldots, M_i\) should satisfy conditions \(\phi_i\)

Deposits made one-by-one

\(\phi_1\)

\(\phi_2\)

\(\phi_3\)

\(\phi_4\)

Claims in Reverse order
Multiple-Move Exchanges
[KMB’15, KVV’16]

$M_i$ depends on $M_1, \ldots, M_{i-1}$
$M_1, \ldots, M_i$ should satisfy conditions $\phi_i$

Aborts after every move
must be penalized

Player in debt until
move is made

$\phi_1$

Bob gets
compensated
Multi-Party Exchanges

A *set of parties* wish to perform an *n-way exchange*

⚠️ *All might collude against one!*

If someone aborts everybody must get compensated

---

Can we implement this just using *claim-or-refund*?

[BK14,KMB15,KVV16]
Order of deposits/claims
- Roof deposits made simultaneously
- Ladder deposits made one after the other
- Ladder claims in reverse
- Roof claims at the end

High-level intuition
- At the end of ladder claims, all parties except $P_n$ have “evened out”
- If $P_n$ doesn’t make roof claims then honest parties get compensation via roof refunds
- Else $P_n$ “evens out”
**High-level Idea**

- Use “dummy” tokens to bootstrap
- After plant claims: $P_1$ has paid all parties
- After i-th ladder claim: $P_{i+1}$ paid all parties; forces $P_{i+1}$ to continue
- Reactive; force any protocol

**Diagram**

```
```

- Use “dummy” tokens to bootstrap
- After plant claims: $P_1$ has paid all parties
- After i-th ladder claim: $P_{i+1}$ paid all parties; forces $P_{i+1}$ to continue
- Reactive; force any protocol
Multiple Exchanges

A set of parties wish to perform *multiple exchanges*

On-chain transactions for every exchange

Conditions depend on information exchanged

Unbounded number of exchanges from

Can we use just

one set of transactions for all exchanges?

[KB’16]
Infrastructures for Contracts

All pairs of parties from a larger universe wish to perform exchanges

Client-Server Architecture
linear number of on-chain transactions
[KB’16,CK]

UNTRUSTED Server
Summary

• Claim-or-refund transactions are a powerful abstraction
  • Decompose complex multiparty contracts into *multiple* simple
    stateless “contracts” (i.e., CR txns) without losing functionality
  • Enables “blockchain interoperability” (not all parties need to be
    on the same cryptocurrency)
• Secure multiparty computation can provide privacy & scalability

Other Related Work: [BBSU’12, BB’14, ADMM’14a, 14b, KZZ15, KMSWP’16]
Thank You!