HOW TO COMPUTE SECURELY USING BITCOIN SCRIPTS

Stefan Dziembowski

International Workshop on P2P Financial Systems
Deutsche Bundesbank, Frankfurt, Germany, 29.01.2015
Goal of this talk

A short introduction to a new research area:

Secure Multiparty Computations on Cryptocurrencies


But what are the secure multiparty computations?
Cryptography in the past

Cryptography ≈ securing communication
main applications: military and diplomacy
Traditional goal of cryptography: emulate “secure channels”

reality:

ideal situation:
Traditional encryption

message $m$ → encryption $Enc$ → ciphertext $c$ → decryption $Dec$ → $m$

adversary

shouldn’t learn $m$
Modern cryptography (from 1970s)

cryptography = much more than “secure communication”!

- public-key encryption
- key agreement
- mental poker
- digital signatures
- e-cash
- zero-knowledge
- coin-tossing by telephone
- electronic auctions
- e-voting
- multiparty-computations

1970s now
One prominent example: **Multiparty Computation (MPC) protocols**

Protocols where the **users of the protocol** don’t trust each other, but nevertheless they want to achieve a common goal
Example 1: coin tossing

output: Y

where \( Y = \begin{cases} \text{Y} & \text{with probability } \frac{1}{2} \\ \text{Y} & \text{with probability } \frac{1}{2} \end{cases} \)
Example 2: marriage proposal

input: \[ A = \begin{cases} 1 & \text{if Alice loves Bob} \\ 0 & \text{otherwise} \end{cases} \]

\[ B = \begin{cases} 1 & \text{if Bob loves Alice} \\ 0 & \text{otherwise} \end{cases} \]

output: \[ Y = A \land B \]
Example 3: set operations

input: $\mathcal{A} =$ a set of Alice’s friends $\mathcal{B} =$ a set of Bob’s friends

output: $Y = \mathcal{A} \cap \mathcal{B}$ (or $Y = \mathcal{A} \cup \mathcal{B}$)
Possible applications

• cloud computing

• online auctions

• e-voting

But is it possible to construct such protocols?
With a “trusted third party” – it’s easy

But can we do it **without** a trusted third party?

In other words: can we “simulate” the **ideal world** in the **real world**?
Note the difference

**encryption**: the adversary is **external**

---

**secure computation**: the adversary is “**internal**”

---

**Alice’s point of view:**

---

**Bob’s point of view:**
So, can we construct such protocols?

**Answer:** Yes! (under some assumptions and with certain limitations)

Every can be “simulated” in a secure way.
The limitations

• **lack of fairness** when there is no honest majority (we will explain it in a moment),

• no way to force the parties to **provide true input**,

• and to **respect the outcome**.

partial remedies exist

beyond the scope of crypto
Our idea

Deal with these problems using Bitcoin
Example: Two party lotteries

- a random party earns 1 BTC
- the other one loses 1 BTC
Looks similar to the “coin-tossing problem”.

Output: \( Y \)

where \( Y = \) \begin{cases} 
\text{output} & \text{with probability } 1/2 \\
\text{output} & \text{with probability } 1/2 
\end{cases}
How to solve the coin-tossing problem?

Idea

Remember the old game:

rock-paper-scissors?
<table>
<thead>
<tr>
<th></th>
<th>Alice</th>
<th>Bob</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rock</td>
<td><strong>draw</strong></td>
<td><strong>Bob wins</strong></td>
</tr>
<tr>
<td>Paper</td>
<td><strong>Alice wins</strong></td>
<td><strong>Alice wins</strong></td>
</tr>
<tr>
<td>Scissors</td>
<td><strong>Bob wins</strong></td>
<td><strong>draw</strong></td>
</tr>
<tr>
<td>Scissors</td>
<td><strong>Alice wins</strong></td>
<td><strong>Bob wins</strong></td>
</tr>
</tbody>
</table>

**Legend:**
- **Draw**: Neither player wins.
- **Win**: The player in the cell wins.
Let’s simplify this game

In other words: Alice wins iff $A \text{xor} B = 0$. 
Another way to look at it

Bob has an input A
Alice has an input B

they should jointly compute
\[ x = A \text{ xor } B \]
(in a secure way)
What to do?

**Problem:**

A and B should be sent at the same time
(e.g. if A is sent before B then a malicious Bob can set $B := x \oplus A$, where $x$ is chosen by him).
How to guarantee this?

Seems hard:

the internet is not synchronous...

A solution:

bit commitments
Commitment schemes – an intuition

Alice sends a locked box to Bob

[a bit B]

Alice “commits herself to B”

[binding] from now Alice cannot change B,
[hiding] but Bob doesn’t know B

Alice can later send the key to Bob

Alice “opens the commitment”
Hash-based commitments

hash-based (in the random oracle model):

- \( H \) – hash function (eg. SHA256)
  - to commit to \( B \in \{0,1\} \) take random \( R \in \{0,1\}^k \) and send \( H(B,R) \)
  - to open \( B \) send \( (B,R) \).
How does it solve the coin-flipping problem?

- **chooses a random bit** $A$
- **chooses a random bit** $B$
- **commits to** $A$
- **sends** $B$
- **opens** $A$
- **output** $A \ xor B$

eto
Problem 1

How to force Alice to open the commitment?

This is precisely the lack of fairness problem.

It’s inherent to most of the interesting MPC protocols...
Problem 2

commits to A

sends B

opens A

You lost

So what?

This is the problem of forcing the parties to respect the output.

Even more inherent (it is present also in the “ideal world” solution)
Fact: Bitcoin allows for more advanced transactions than the simple money transfers (this is used for the “Bitcoin contracts”).

“simple” transactions: “1 BTC from transaction $T_1$ is transferred from Alice to Bob”

- $T_1$ 1 BTC can be spent by Bob signature of Alice

“strange” transactions:

- written in a “Bitcoin scripting language”
- can involve timing constraints.
Force the parties to open their commitments using the “deposits”

**Transaction commit**
- has value **1 BTC**
- can be redeemed by **Alice**
- claiming the transaction requires revealing **b**
How to implement it?

• We use the **Bitcoin scripting language**.

• Luckily the **hash functions** are there as a primitive.

• So we can use the **hash-based commitments**.
This solves the problem of the lack of fairness!

Alice commits with a Bitcoin-based commitment to A.

B sends A.

A opens A.

If Alice does not open her commitment within 1 hour then Bob can get her 1 BTC.

Otherwise she gets her 1 BTC back.
What about the problem of respecting the outcome?

This can also be solved. Main idea:

- A commits with a Bitcoin-based commitment to A
- B commits with a Bitcoin-based commitment to B
- A transaction that takes the opening of the committed values and "decides" who won, with a probability of 1/2
Our results

We show that any functionality can be "simulated" in this way.

The simulation can enforce the financial consequences.
An example: selling secret information

“set-sum with rewards for each record”

output: $A \cup B$

plus a money transfer between Alice and Bob depending on the number of new records that the parties learned.
Thank you!