Bitcoin contracts—digital economy without lawyers?

Stefan Dziembowski

University of Warsaw

UCL London weekly Bitcoin Research Seminar, 31 October 2014
This talk is based on:

Marcin Andrychowicz, Stefan Dziembowski, Daniel Malinowski, and Łukasz Mazurek

Secure Multiparty Computations on Bitcoin
IEEE Symposium on Security and Privacy (Oakland) 2014

Fair Two-Party Computations via Bitcoin Deposits
First Workshop on Bitcoin Research 2014

How to deal with malleability of Bitcoin transactions
(available on arXiv)

Independent work:

Adam Back and Iddo Bentov

Note on fair coin toss via Bitcoin
(available on arXiv)

Iddo Bentov and Ranjit Kumaresan

How to Use Bitcoin to Design Fair Protocols
Crypto 2014
Digital vs. paper currencies

Paper:

Digital:

Very useful if is also digital.
Traditional ways of paying “digitally”

PROBLEMS
1. trusted server for each transaction is needed (money doesn’t “circulate”),
2. high transaction fees,
3. no anonymity.
Can we have a true “digital analogue” of the paper money?

Yes: Bitcoin!


currency unit: Bitcoin (BTC).

$1 \text{ BTC} = 10^8 \text{ Satoshi}$

as of 30.10.2014:

- Market cap $\approx$ € 3.6 billion
- $1 \text{ BTC} \approx$ € 267
Bitcoin even made it to the British Museum!
Probably one of the most discussed cryptographic technologies ever!
PROBLEMS WITH PREVIOUS APPROACHES

1. trusted server is needed (money doesn’t “circulate”),
2. high transaction fees,
3. no anonymity.

in Bitcoin:
no trusted server, money circulates
low fees
“pseudonymity”
“No trusted server”

nobody “controls the money”, and therefore:

– The amount of money that will ever be “printer” is fixed (to around 21 mln BTC) \( \rightarrow \) **no inflation**

– The **exchange rate fluctuates**:
Bitcoin $\approx$ “real money”? 

**Bitcoin** value comes from the fact that: “people expect that other people will accept it in the future.”

*Bitcoin enthusiasts:* It’s like all the other currencies

*Bitcoin sceptics:* It’s an artificial “bubble”
Main problem with the digital money

Double spending...

Bits are easier to copy than paper!
Bitcoin idea (simplified):

The users emulate a **public trusted bulletin-board** containing a list of transactions.

A transaction is of a form:

“**User P₁** transfers a coin #16fab13fc6890 to user **P₂**”

**This prevents double spending.**

16fab13fc6890

you’ve already spent this coin!
What needs to be discussed

1. How is the **trusted bulletin-board** maintained?
2. How are the users identified?
3. Where does the money come from?
4. What is the syntax of the transactions?
The “ideal” world

A protocol that emulates the ideal world

Main difficulty: Some parties can cheat.
Classical result: Simulation is possible if the “majority is honest”. For example for 5 players we can tolerate at most 2 “cheaters”.
Problem

How to define “majority” in a situation where everybody can join the network?
The Bitcoin solution

Define the “majority” as

the majority of the computing power

Now creating multiple identities does not help!
How is this verified?

Main idea:

• use Proofs of Work
• incentivize honest users to constantly participate in the process

The honest users can use their idle CPU cycles.

Nowadays: often done on dedicated hardware.
More details

The users participating in the scheme are called the “miners”.

They maintain a chain of blocks:

- The “genesis block” created by Satoshi on 03/Jan/2009
- Approximately 10 min.

- $block_0$
  - Transactions from period 1
- $block_1$
  - Transactions from period 2
- $block_2$
  - Transactions from period 3
- $block_3$

More details
How to post on the board

Just broadcast (over the internet) your transaction to the miners.

And hope they will **add it to the next block.**

the miners are incentivized to do it.
How are the miners incentivized to participate in this game?

**Short answer:** they are paid (in Bitcoins) for this.

We will discuss it in detail later...
What needs to be discussed

1. How is the trusted bulletin-board maintained?
2. How are the users identified?
3. Where does the money come from?
4. What is the syntax of the transactions?
User identification

We use the digital signature schemes.

The users are identified by their public keys.
Digital signature schemes

A **digital signature scheme** consists of algorithms $\text{Gen}$, $\text{Sign}$ and $\text{Vrfy}$, where:

**input:**
- $(sk, \text{ message } M)$
- $(pk, M, \sigma)$

**output:**
- (secret key $sk$, public key $pk$)
- signature $\sigma$
- yes/no

**Correctness:**
for every $(sk, pk) := \text{Gen}()$ and every $M$ we have
$\text{Vrfy}(pk, M, \text{Sign}(sk, M)) = \text{yes}$

**Security:**
“without knowing $sk$ it is infeasible to compute $\sigma$ such that $\text{Vrfy}(pk, M, \sigma) = \text{yes}$”
What needs to be discussed

1. How is the **trusted bulletin-board** maintained?
2. How are the users identified?
3. Where does the money come from?
4. What is the syntax of the transactions?
Where does the money come from?

A miner who finds a new block gets a “reward” in BTC:

- for the first 210,000 blocks: 50 BTC
- for the next 210,000 blocks: 25 BTC
- for the next 210,000 blocks: 12.5 BTC,

and so on...

Note: $210,000 \cdot (50 + 25 + 12.5 + \cdots) \rightarrow 21,000,000$
This is how it looks in detail

<table>
<thead>
<tr>
<th>Transaction</th>
<th>Fee</th>
<th>Size (kB)</th>
<th>From (amount)</th>
<th>To (amount)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0ac34c9949...</td>
<td>0</td>
<td>0.173</td>
<td>Generation: 25 + 0.05974785 total fees</td>
<td>1KFHE7w8BhaENAswwryaoecDb6qeT6DbYY: 25.05974785</td>
</tr>
<tr>
<td>2055f19a51...</td>
<td>0.002</td>
<td>0.259</td>
<td>1Kpv8JEcWLhUq14q8dnrwxiaZPKL4KUceR: 179.9998</td>
<td>1HCukLGfkCFdCryXT73hj2SyVAC9KzRGkC: 105 15zBXYeXbtU5xs48arouP7BHQu4A5xfZa: 74.9996</td>
</tr>
<tr>
<td>66815aff0...</td>
<td>0.001</td>
<td>0.258</td>
<td>1dice5DPtUMBpWgv8i4pG8HMjXv9qDJWN: 0.35</td>
<td>15GPjiasjMD8QJvMTs5qYsB8wtQLQGBtP: 0.00175 1HZHBNh2FbHNWicMxAh4xBPf9uxW15UPt: 0.34725</td>
</tr>
</tbody>
</table>
More details

Each block contains a transaction that transfers the reward to the miner.

**Advantages:**

1. It provides incentives to be a miner.
2. It also makes the miners interested in broadcasting new block asap.

This view was challenged in a recent paper:
Ittay Eyal, Emin Gun Sirer **Majority is not Enough: Bitcoin Mining is Vulnerable**
What needs to be discussed

1. How is the trusted bulletin-board maintained?
2. How are the users identified?
3. Where does the money come from?
4. What is the syntax of the transactions?
Transaction syntax – simplified view

\[ T_1 = \text{(User } P_1 \text{ creates 25 BTC)} \]

\[ T_2 = \text{(User } P_1 \text{ sends 25 BTC from } T_1 \text{ to } P_2 \text{ signature of } P_1 \text{ on } [T_2]) \]

\[ T_3 = \text{(User } P_2 \text{ sends 25 BTC from } T_2 \text{ to } P_3 \text{ signature of } P_2 \text{ on } [T_3]) \]

We say that \( T_3 \) redeems \( T_2 \)

in the “mining process”
How to “divide money”?  

Multi-output transactions:  

\[ T_2 = (\text{User } P_1 \text{ sends } 10 \text{ BTC from } T_1 \text{ to user } P_2, \text{ User } P_1 \text{ sends } 7 \text{ BTC from } T_1 \text{ to user } P_3, \text{ User } P_1 \text{ sends } 8 \text{ BTC from } T_1 \text{ to user } P_4) \]  

signature of \( P_1 \) on \([T_2]\)
Multiple inputs

User $P_1$ sends **10 BTC** from $T_1$ to user $P_4$,
User $P_2$ sends **7 BTC** from $T_2$ to user $P_4$,
User $P_3$ sends **8 BTC** from $T_3$ to user $P_4$

signature of $P_1$ on $[T_4]$,
signature of $P_2$ on $[T_4]$,
signature of $P_3$ on $[T_4]$)

all signatures need to be valid!
Time-locks

It is also possible to specify time $t$ when a transaction becomes valid.

$$T_2 = \begin{cases} \text{(User } P_1 \text{ sends 25 BTC from } T_1 \text{ to } P_2 \text{ if time } t \text{ has passed)} & \text{signature of } P_1 \text{ on } [T_2] \end{cases}$$

measured in:
- real time, or
- blocks.
Generalizations

1. All these features can be combined.
2. The total value of **in-coming transactions** can be larger that the value of the **out-going transactions**.

   (the difference is called a “fee” and goes to the **miner**)

1. The condition for redeeming a transaction can be more general (the so-called “**strange transactions**”)
Strange transactions:

- $T_2 = (\text{User } P_1 \text{ sends 1 BTC from } T_1 \text{ to } P_2 \text{ signature of } P_1 \text{ on } [T_2])$
- $T_3 = (\text{User } P_2 \text{ sends 1 BTC from } T_2 \text{ to } P_3 \text{ signature of } P_2 \text{ on } [T_3])$

- $T_2 = T_1 + 1 \text{ BTC} \quad \text{a condition } C_2 \text{ to spend } T_2 \quad \text{a “witness } W_2”$
- $T_3 = T_2 + 1 \text{ BTC} \quad \text{a condition } C_3 \text{ to spend } T_3 \quad \text{a “witness } W_3”$

A Boolean function
Redeeming condition

$T_3$ redeems $T_2$ if

$C_2$ evaluates to **true** on input $([T_3], W_3)$.  

**Note**: in the standard transactions:

$$C_2([T_3], W_3) = Vrfy(pk_2, [T_3], W_3)$$
Example

“Alice gives 1 BTC to the first person that factors 2501.”

Alice posts:

\[ T_1 \rightarrow T_2 \]

earlier transaction that can be spent by Alice

\[ T_2 = T_1 \]

\[ T_2 = \] 1 BTC

\[ T_2 = \] can be spent using \( p \) and \( q \) such that \( p,q > 1 \) and \( pq = 2501 \)

Alice’s signature

formally:

\[ C([T],[p,q]) = \text{true} \iff p,q > 1 \& pq = 2501 \]

Bob claims the money by posting:

\[ T_3 = T_2 \]

\[ T_3 = T_2 \]

\[ T_3 = \] 1 BTC

\[ T_3 = \] can be spent using Bob’s signature

\[ p = 41 \]

\[ q = 61 \]
How are the conditions written?

In **Bitcoin scripting language** (non-Turing complete stack-based)

Example:

```
OP_DUP OP_HASH160
02192cfd7508be5c2e6ce9f1b6312b7f268476d2
OP_EQUALVERIFY OP_CHECKSIG
```

A new currency with **Turing-complete** scripts:
Bitcoin contracts
Bitcoin contracts

The “strange transactions” can be used to create the “Bitcoin contracts”.

**Simple examples:**

- Pay money to anyone who knows some password.
- Assurance contracts.
- Put a “deposit” to prove you are not a spammer.
- Pay money only if some event happens (may require an oracle).

**A more advanced example:** secure multiparty computation protocols.
Multiparty Computation Protocols

protocols that allow a group of parties to emulate a trusted functionality.

the “ideal” world

a protocol that emulates the ideal world

we already saw one example (bulletin board)

other examples: auctions, lotteries, voting,...
MPCs

In general possible for any functionality, however there are some limitations:

• **lack of fairness** when there is no honest majority,

• no way to force the parties to **provide true input** (think of voting),

• and to **respect the outcome** (think of auctions).

**partial remedies exist** beyond the scope of crypto
Our idea

Combine MPCs with Bitcoin!

• use “Bitcoin deposits” to enforce **fairness**

• use Bitcoin to force users to respect the “**financial consequences**” of the protocol.
Example: two-party lotteries
More complicated examples

Two party lotteries

a random party earns 2 BTC

Can be extended to the multiparty case.
Coin-flipping by telephone [Blum’81]

privacy and authenticity is not a problem

Suppose Alice and Bob are connected by a secure internet link:

The goal of Alice and Bob is to toss a coin.

*In other words:* They want to execute a protocol $\pi$ in such a way that at the end of the execution they both output the same bit $x$ distributed uniformly over \{0,1\}. 
How to define security? [1/2]

Let us just stay at an informal level...

From the point of view of Alice:

even if Bob is cheating (i.e.: he doesn’t follow the protocol):  
if the protocol terminates successfully, then x has a uniform distribution
The same holds from the point of view of Bob even if Alice is cheating (i.e.: he doesn’t follow the protocol): if the protocol terminates successfully, then $x$ has a uniform distribution.
How to solve this problem?

Idea

Remember the old game:

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

- Alice wins
- Bob wins
- Draw
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 jointly compute

$$x = A \text{xor} B$$

(in a secure way)
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 \( B := x \) xor 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
- 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”
Constructions of the bit commitments

• hash-based (in the random oracle model):
  H – hash function
  – to commit to $b \in \{0,1\}$ take random $r \in \{0,1\}^k$ and send $H(b || r)$
  – to open $b$ send $(b,r)$.

other constructions:
• based on the one-way functions,
• based on the discrete log, quadratic residuosity assumption,
So, how does it solve the coin-flipping problem?

chooses a random bit A

commits to A

sends B

opens A

chooses a random bit B

output A xor B

output A xor B
Problem

1. How to force Alice to open the commitment?

2. How to force the looser to pay to the winner?

[lack of “fairness”]
Our idea

“Embed” this protocol into Bitcoin (using the scripts).

We first present a “flawed” protocol.

Then we show how to fix it.
has a transaction \( T_A \) of value 1 BTC on the block chain

selects random \((A,R_A)\)

they prepare a transaction

\[
\text{Compute} = \]

\[
\begin{array}{c|c}
T_A & 1 \text{ BTC} \\
\hline
T_B & 1 \text{ BTC} \\
\end{array}
\]

\[
\begin{array}{c}
\text{can be claimed using } A',R'_A,B',R'_B \text{ such that } \\
H(A',R'_A) = h_A \text{ and } H(B',R'_B) = h_B \text{ and } \\
\text{• a signature of Alice if } A' \text{ xor } B' = 0 \text{ or } \\
\text{• a signature of Bob if } A' \text{ xor } B' = 1 \\
\end{array}
\]

transaction \( \text{Compute} \) is posted on the block chain

if \( A \text{ xor } B = 0 \) then

\[
\begin{array}{c}
\text{Alice can claim the money by the following transaction:} \\
\text{Get} = \\
\text{Compute} & 2 \text{ BTC} \\
\end{array}
\]

\[
\begin{array}{c}
\text{can be claimed using the signature of Alice} \\
A,R_A & A,R_A,B,R_B \\
\text{signature of Alice} \\
B,R_B \\
\end{array}
\]

otherwise \( \text{Bob} \) can claim the money with a symmetric transaction
Problem

A malicious party can refuse to open her commitment after she learned that she lost.

Our solution:

“timed commitments” – a party is forced to open her commitment before some time, or she pays a fine (more precisely: she loses her deposits).

This is done using the **time-locked transactions in Bitcoin**.
Generalizations

We show that the “Bitcoin deposits” can be used to enforce fairness in any multiparty protocol.

Moreover any multiparty function can be computed in such a way that that Bitcoin enforces the parties to respect its “financial outcome”.
Example of an application: paying for delivering secret information

I will pay **100 BTC** to anyone who provides information about secret bank accounts abroad.
Thank you!