Scriptless Bitcoin Lotteries from Oblivious Transfer

Lloyd Fournier

lloyd.fourn@gmail.com

September 10, 2019

What's a Lottery?

Imagine a trusted party Lloyd who conducts lotteries between Alice and Bob:

- Alice and Bob send 1 BTC to Lloyd's Address
- Lloyd chooses a winner randomly and sends them 2 BTC

The goal of a lottery protocol is achieve the same result without a trusted third party.

What's Scriptless?

Bitcoin has a way of specifying the spending rules on coins with a language called "Script". A scriptless protocol doesn't use it. Instead it realises the spending rules off-chain through some cryptographic protocol. For example:

- You can replace OP_CHECKMULTISIG with a threshold mutli-signature scheme.
- You can replace HTLC by using "adaptor" signatures

Why Scriptless?

1. Scriptless protocols are more efficient and more private (no validation rules appear on the blockchain).

Why Scriptless?

1. Scriptless protocols are more efficient and more private (no validation rules appear on the blockchain).
2. To know what can and can't be done without script

Why Scriptless?

1. Scriptless protocols are more efficient and more private (no validation rules appear on the blockchain).
2. To know what can and can't be done without script

https://bitcointalk.org/index.php?topic=355174.0

> *...I think the multi-party lottery scheme still rates as the most advanced usage of script yet found in the wild*

— Mike Hearn

How to do a lottery without a trusted third party?

- Need to fairly generate randomness (no `OP_RANDOM`)
- The randomness choose the winner
- Enforce the outcome on the blockchain

All previous lotteries used hash commitment "coin tossing" to generate the randomness

Coin tossing

Idea introduced in by Manuel Blum in 1981: *Coin Flipping By Telephone: A protocol for solving impossible problems*

> *Alice and Bob want to flip a coin by telephone. (They have just divorced, live in different cities, want to decide who gets the car.)*

Coin tossing

Idea introduced in by Manuel Blum in 1981: *Coin Flipping By Telephone: A protocol for solving impossible problems*

> *Alice and Bob want to flip a coin by telephone. (They have just divorced, live in different cities, want to decide who gets the car.)*

basic idea:

1. Alice sends a "commitment" to her coin toss
2. Bob sends his coin toss
3. Alice reveals her coin toss

Hash-Commitment Coin tossing

$$
\begin{array}{lcr}
\underline{\mathit{Alice} \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \mathit{Bob}} & & \\
b_1 \xleftarrow{\$} \{0,1\} & & \\
r \xleftarrow{\$} \{0,1\}^l & & \\
t \leftarrow H(b_1, r) & \xrightarrow{\quad t \quad} & \\
 & \xleftarrow{\quad b_2 \quad} & b_2 \xleftarrow{\$} \{0,1\} \\
\Leftarrow b_1 \oplus b_2 & & \\
 & \xrightarrow{\quad b_1', r' \quad} & \\
 & & H(b_1', r') \stackrel{?}{=} t \\
 & & b_1' \oplus b_2 \Rightarrow
\end{array}
$$

Note: It's *unfair*

Lottery Protocols – First Attempt

Iddo - fair coin toss with no extortion and no need to trust a third party https://bitcointalk.org/index.php?topic=277048.0 (2013)

> *...there seemed to be an interest in implemented a gambling system that doesn't involve a 3rd-party like SatoshiDice, so we can eliminate the house edge*

Lottery Protocols – First Attempt

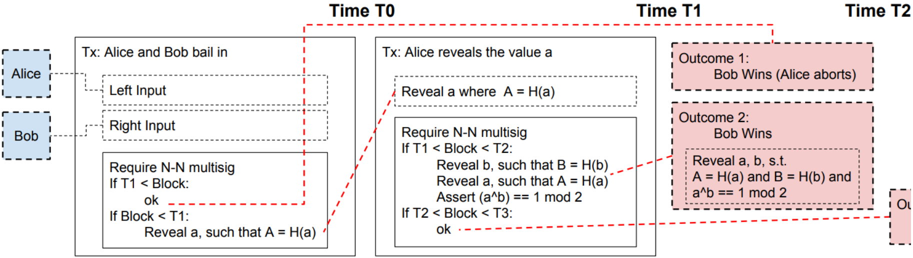
Iddo - fair coin toss with no extortion and no need to trust a third party https://bitcointalk.org/index.php?topic=277048.0 (2013)

> *...there seemed to be an interest in implemented a gambling system that doesn't involve a 3rd-party like SatoshiDice, so we can eliminate the house edge*

Two Ideas:

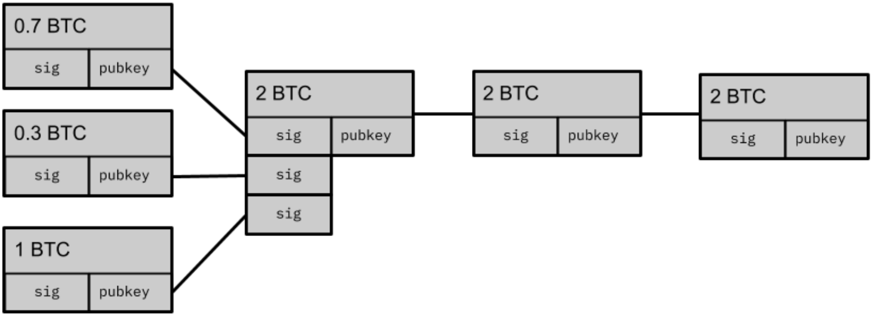
- Iddo Bentov: We can use collateral to force both parties to reveal pre-images.
- Adam Back: If a party doesn't reveal their pre-image we should just make them lose by default.

Zero-Collateral Lotteries in Bitcoin and Ethereum



Scriptless Lottery

How do we get our lottery to look like this?



Oblivious transfer

Oblivious transfer

1. Manuel Blum to the rescue (again)! He made another discovery in 1981: you can generate a random outcome with oblivious transfer.
2. University of Berkeley technical report: *Three applications of oblivious transfer: 1. Coin flipping by telephone, 2. How to exchange secrets, 3. How to send certified electronic mail.*

Oblivious transfer

1. Manuel Blum to the rescue (again)! He made another discovery in 1981: you can generate a random outcome with oblivious transfer.
2. University of Berkeley technical report: *Three applications of oblivious transfer: 1. Coin flipping by telephone, 2. How to exchange secrets, 3. How to send certified electronic mail.*

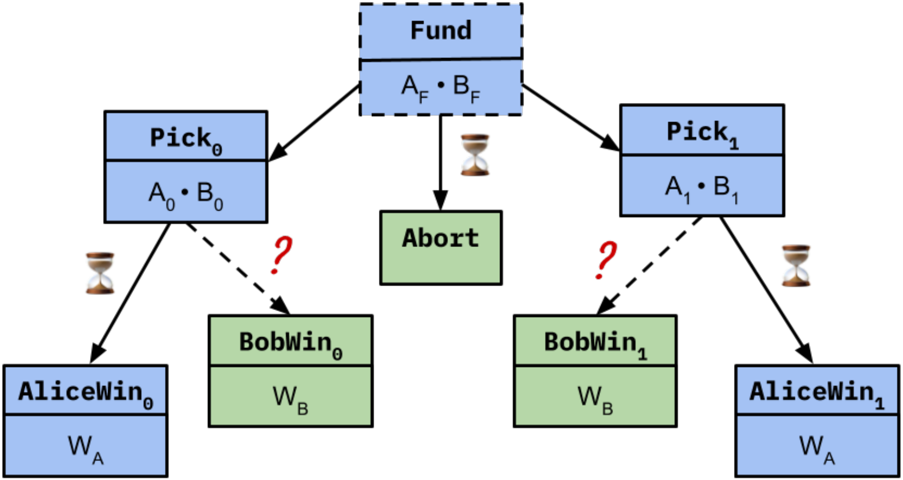
Rough Definition:

1. Alice transmits one of n message to Bob of Bob's choosing (But Alice doesn't learn which message).
2. **receiver security** Sender doesn't know which message was sent
3. **sender security** Receiver only gets one message

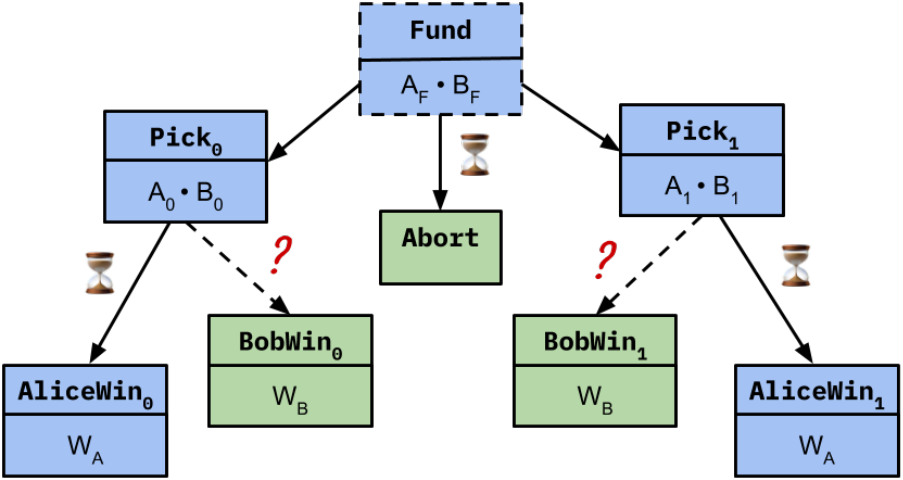
Alice		*Bob*
$m_0, m_1 \xleftarrow{\$} \{0,1\}^k$		$b_2 \xleftarrow{\$} \{0,1\}$
	$\rightleftarrows \mathsf{OT}((b_2),(m_0,m_1)) \leftrightarrows$	learns m_{b_2}
$b_1 \xleftarrow{\$} \{0,1\}$	$\xrightarrow{\quad b_1 \quad}$	$\Rightarrow b_1 \oplus b_2$
	$\xleftarrow{\quad m' \quad}$	

$$b_2' := \begin{cases} 0 & m' = m_0 \\ 1 & m' = m_1 \\ \perp & \text{otherwise} \end{cases}$$

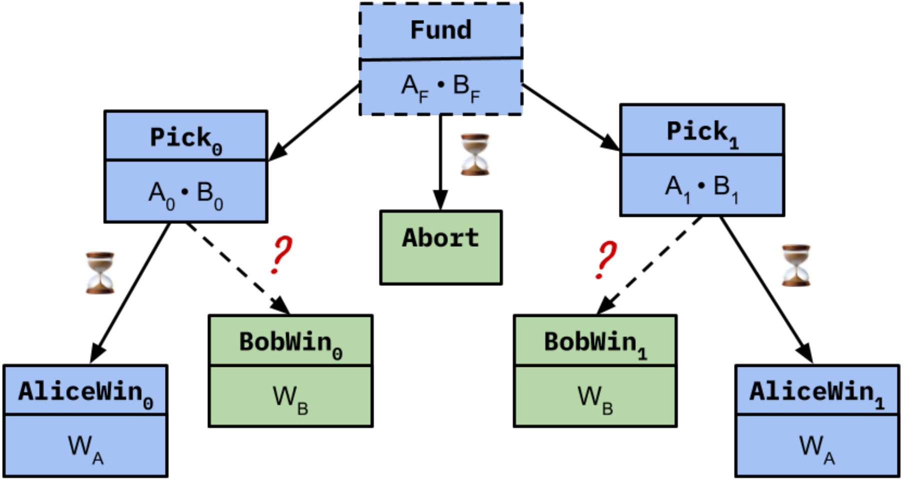
$\Leftarrow b_1 \oplus b_2'$


Fund
$A_F \cdot B_F$
Pick$_0$
$A_0 \cdot B_0$
Pick$_1$
$A_1 \cdot B_1$
Abort
?
?
AliceWin$_0$
W_A
BobWin$_0$
W_B
BobWin$_1$
W_B
AliceWin$_1$
W_A

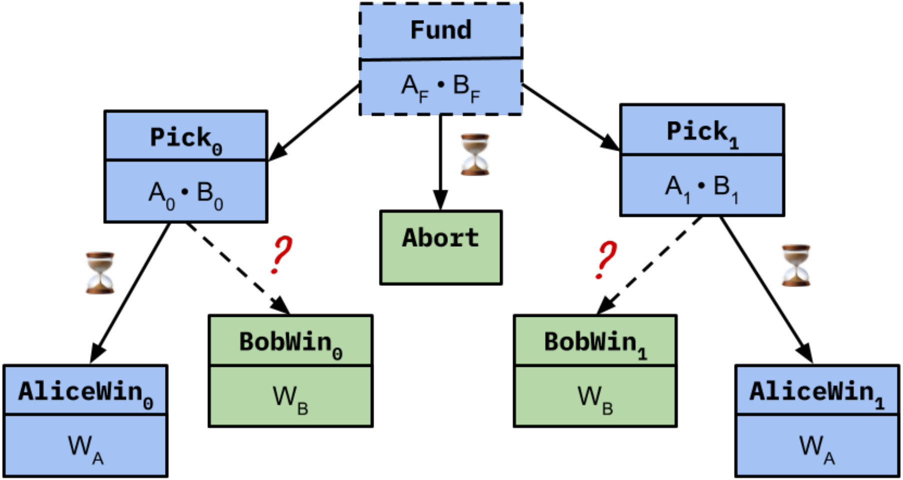
1. Choose keys securely



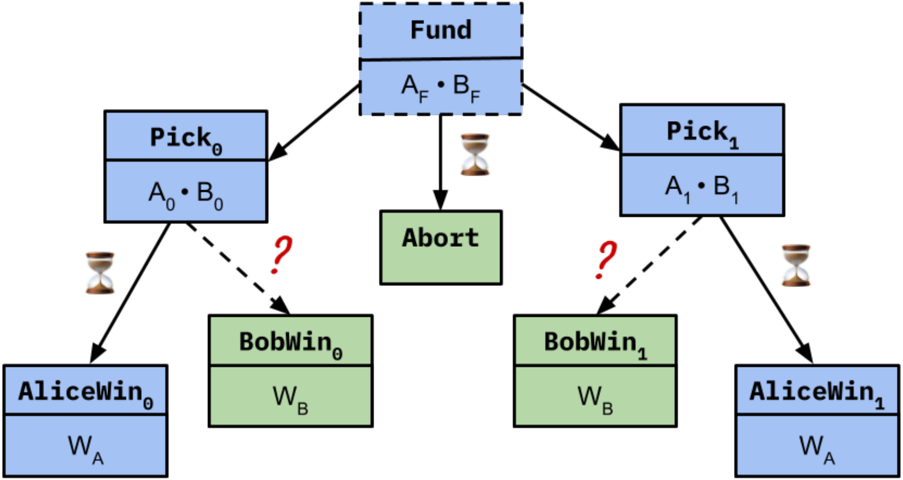
1. Choose keys securely
2. Sign the transaction scaffold



1. Choose keys securely
2. Sign the transaction scaffold
3. Alice **obliviously signs** `BobWin`



1. Choose keys securely
2. Sign the transaction scaffold
3. Alice **obliviously signs** `BobWin`
4. Sign and broadcast `Fund`



How to realise Oblivious Signing?

We want Bob to have a signature on $\texttt{BobWin}_0$ OR $\texttt{BobWin}_1$ but without Alice knowing which one.

How to realise Oblivious Signing?

We want Bob to have a signature on $\texttt{BobWin}_0$ OR $\texttt{BobWin}_1$ but without Alice knowing which one.

Approach: Use adaptor signatures where Bob only knows the completion for one of them.

Adaptor Signatures (Poelstra 2017)

Adaptor signatures enable the signer to create a simple access structure to a particular signature on a particular message.

	Schnorr	Adaptor
args	sk, m	$sk, m, Y = g^y$
$s =$	$r + H(g^r \| \| pk \| \| m)sk$	$r + H(g^r Y \| \| pk \| \| m)sk$
valid	(s, g^r)	$(s + y, g^r Y)$

Adaptor Signatures (Poelstra 2017)

Adaptor signatures enable the signer to create a simple access structure to a particular signature on a particular message.

	Schnorr	Adaptor
args	sk, m	$sk, m, Y = g^y$
$s =$	$r + H(g^r \| \| pk \| \| m)sk$	$r + H(g^r Y \| \| pk \| \| m)sk$
valid	(s, g^r)	$(s + y, g^r Y)$

Important feature: The signer learns y if they see the completed signature.

Oblivious Signing

Approach: Alice to give Bob adaptor signatures where Bob only knows the completion for one of them.

Oblivious Signing

Approach: Alice to give Bob adaptor signatures where Bob only knows the completion for one of them.

- A Pedersen commitment for x is in the form $T = g^x h^c$.
- It's impossible to decommit to more than one value without knowing the $\mathrm{DLOG}_g(h)$

Oblivious Signing

Approach: Alice to give Bob adaptor signatures where Bob only knows the completion for one of them.

- A Pedersen commitment for x is in the form $T = g^x h^c$.
- It's impossible to decommit to more than one value without knowing the $\text{DLOG}_g(h)$
- Fix $c \in \{0, 1\}$, committer can only know discrete log of T OR Th^{-1}
    - $c = 0 \longrightarrow T = g^x$ (knows DLOG(T))
    - $c = 1 \longrightarrow T = g^x h$ (knows DLOG(Th^{-1}))

Oblivious Signing

$Alice(sk)$	(m_0, m_1, h)	$Bob(pk, c \in \{0,1\})$
		$y \leftarrow_{\$} \mathbb{Z}_q$
	$\xleftarrow{T = g^y h^c}$	
$(s_0, R_0) = \sigma(sk, m_0, T)$ $(s_1, R_1) = \sigma(sk, m_1, Th^{-1})$		
	$\xrightarrow{(s_0, R_0), (s_1, R_1)}$	
		verify both $(s_c + y, R_c \cdot g^y) \Rightarrow$

Secure?

1. **security for sender (Alice)**: If Bob can complete both adaptor signatures then he can solve $DLOG(h)$ (from the completion of both adaptor signatures we learn $DLOG$ of T, Th^{-1}).

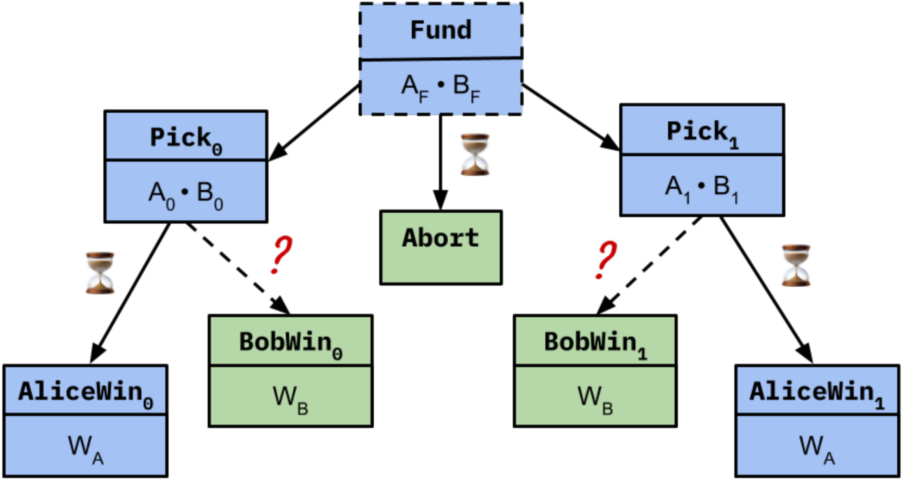
Secure?

1. **security for sender (Alice)**: If Bob can complete both adaptor signatures then he can solve $DLOG(h)$ (from the completion of both adaptor signatures we learn $DLOG$ of T, Th^{-1}).
2. **security for receiver (Bob)**: It's information theoretically impossible for Alice to learn which choice Bob made (until he reveals it).

Secure?

1. **security for sender (Alice)**: If Bob can complete both adaptor signatures then he can solve $DLOG(h)$ (from the completion of both adaptor signatures we learn $DLOG$ of T, Th^{-1}).
2. **security for receiver (Bob)**: It's information theoretically impossible for Alice to learn which choice Bob made (until he reveals it).
3. Original protocol due to: Tso R., Okamoto T., Okamoto E. (2008) 1-out-of-n Oblivious Signatures. In: Chen L., Mu Y., Susilo W. (eds) Information Security Practice and Experience. ISPEC 2008. Lecture Notes in Computer Science, vol 4991. Springer, Berlin, Heidelberg

Not so fast! Our transactions outputs are locked by joint public keys!

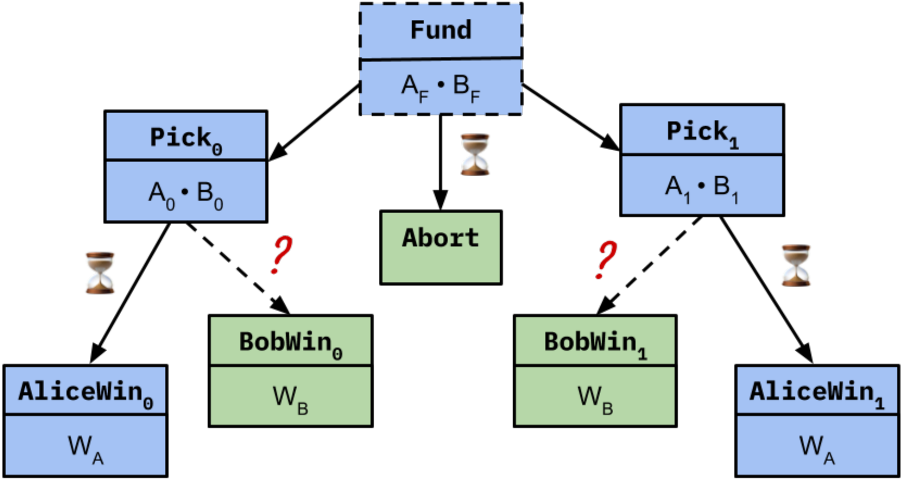


Two-party oblivious signing

One party knows which of two messages they are both signing but the other doesn't. Make a few changes:

1. They first choose joint nonces $R_{\texttt{BobWin}_0}, R_{\texttt{BobWin}_1}$ (we did this already).
2. Along with T Bob sends half adaptor signatures for m_0 ($\texttt{BobWin}_0$) and m_1 ($\texttt{BobWin}_1$) under B_0 and B_1
3. Alice calculates the full adaptor signatures and sends them back to Bob

Ok Done!



Fund
A_F • B_F
Pick_0
A_0 • B_0
Pick_1
A_1 • B_1
Abort
?
?
BobWin_0
W_B
BobWin_1
W_B
AliceWin_0
W_A
AliceWin_1
W_A

Summary

Summary:

- Lotteries were previously an example of something that could only be done with non-trivial script.

Summary

Summary:

- Lotteries were previously an example of something that could only be done with non-trivial script.
- They can be (and are actually more efficient on-chain)

Summary

Summary:

- Lotteries were previously an example of something that could only be done with non-trivial script.
- They can be (and are actually more efficient on-chain)
- Oblivious signing with Schnorr signatures is secure and efficient (at least as it is used here).

Summary

Summary:

- Lotteries were previously an example of something that could only be done with non-trivial script.
- They can be (and are actually more efficient on-chain)
- Oblivious signing with Schnorr signatures is secure and efficient (at least as it is used here).

Unsubstantiated claims:

- You can do lotteries with different odds by doing $1/N$ oblivious signing rather than $1/2$.
- You can cooperatively complete the lottery in two on-chain transactions.
- You can execute cooperative protocol in a payment channel (and even do it "multi-hop" I think).

The End