

Blockchains & Distributed Ledgers

Lecture 09

Aggelos Kiayias

Slide credits: AK

Secure multiparty Computation and Applications

- Sharing responsibility for signatures and cryptographic keys
 - Secret Sharing
- Security critical computations
 - Coin flipping and verifiable secret-sharing
 - Secure multiparty computation (MPC)
- Fair swaps and fair MPC

Overarching question

How to protect security critical operations?

- Key idea : share responsibility and somehow tolerate faulty participants!

Cryptographic Keys?

Cryptocurrency addresses?

Computations ?

What about computations on private data?

Multi-sig transactions

- A payment to a script (P2SH) can be used to facilitate a multi signature transaction.

Mutli-sig Transactions, II

```
scriptPubKey: OP_HASH160 <redeemscriptHash> OP_EQUAL
```

```
scriptSig: OP_0 <sig_Ai> ... <sig_An> <redeemscript>
```

```
redeemscript = OP_m <A1 pubkey> <A2 pubkey>... <An pubkey>  
<OP_n> <OP_CHECKMULTISIG>
```

Secret-Sharing

Main question:

- How to share a secret s to n shareholders so that
 - Any subset including t of them can recover the secret.
 - Any subset including less than t of them knows nothing about the secret.
- Points to ponder:
 - Can we solve this for any n and $t \leq n$?
 - What is the relation between the size of s and the size of each share.

Finite fields

Finite sets equipped with two operations behaving similarly to addition and multiplication over the real numbers (which is an infinite field).

Example :

A binary finite field $\{0, 1\}$

+	0	1
0	0	1
1	1	0

$(a+b) \bmod 2$

*	0	1
0	0	0
1	0	1

$(a*b) \bmod 2$

Finite fields exist with number of elements equal to p^k , for

- 1) Any prime number p
- 2) Any positive integer k

Secret-Sharing

- Consider random N values subject to the constraint

$$\sum_{i=1}^N x_i = x \quad (\text{over a finite field})$$

- This is called a secret-sharing.
- Observe knowledge of any $N-1$ values is not helpful in any way to infer information about x

Analysis

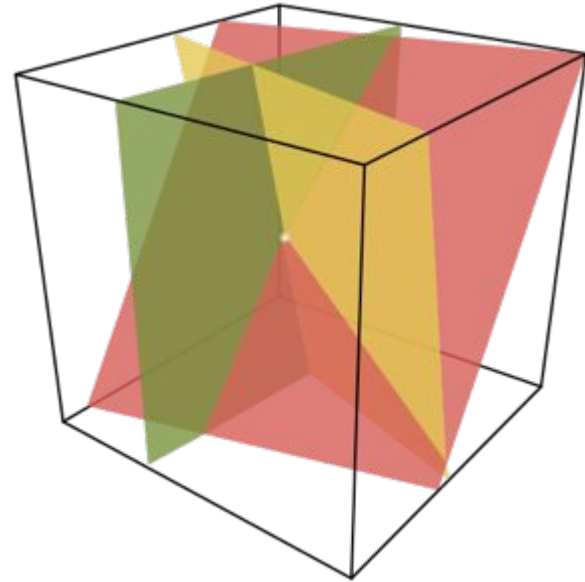
- Imagine you hold $N-1$ values.
- Example binary field.

$$x_1 + x_2 + \dots + x_N = s$$

$$s = 0 + x_2 + \dots + x_N$$

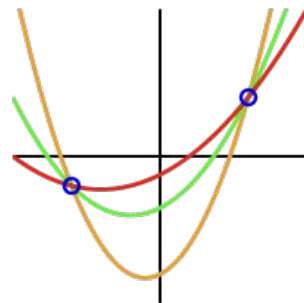
or

$$s = 1 + x_2 + \dots + x_N$$



Generalisation t-out-of-n

- Recall the notion of a polynomial.
- $p(x) = a_0 + a_1x + \dots + a_dx^d$



Any $d+1$ points of the polynomial completely determine it. Given up to d points, at least one degree of freedom remains and it remains undetermined.

We can use that to solve secret-sharing for any t, n

Secret-sharing cryptographic keys

Using polynomial secret-sharing, it is possible to split a cryptographic key to multiple shareholders.

Additional points to consider:

- How should the value of t be determined (in comparison, say, to n) ?
- In order to engage in the cryptographic operation is it necessary to reconstruct the original key?
- How to accomodate an evolving set of shareholders?

Security critical computations

How to obtain the output of a security critical computation.

1. Deterministic with public inputs?
 - a. it can be repeated multiple times, and consensus can be reached about its output. This is what blockchain systems with smart contracts do.
2. What if it is probabilistic with public inputs?
 - a. Coin flipping protocol.
3. What if it uses private data?
 - a. Secure multiparty computation

Application: coin-flipping

- Recall the classic two-party coin flipping protocol: Alice commits to Bob a random coin, Bob sends her a random coin and they return the XOR of them (with Alice opening her coin).
- Consider: how do you deal with aborts?
- Can the situation be improved in an N party coin flip? what about when $>N/2$ parties are honest?

A first step towards coin flipping

- Players commit their coin publicly and also include a secret-sharing of the opening of the commitment so that any subset of players with cardinality $> N/2$ can reconstruct the opening. Shares should be encrypted with the respective public-keys of the parties.
- Thus if some parties abort the protocol, assuming that a subset of $> N/2$ parties continue they can recover the share and terminate.
- At the same time any number of parties up to $N/2$ are unable to gain any advantage against the honest parties (they will be one party short).

Example

- Consider 5 parties. We choose polynomials of degree 2.
 - Any three parties, holding 3 points, are able to interpolate such polynomials.
 - Any two parties, have no information about the shared secret.

A critical issue that remains

- What happens when some parties deliver incorrect shares?
 - Such parties can still selectively abort, causing the remaining parties to be unable to reconstruct the secret.
 - Possible solution: have all commitments open at the end irrespectively of aborts.
 - Still this does not help: deviating players will be caught but they still have the option to selectively abort the execution if they wish (and other parties will only know this when it is too late).
 - One possible approach: issue monetary penalties to those that abort.

Publicly verifiable secret-sharing

- The dealer creates shares that are distributed in encrypted form.
- The shares provided by the dealer can be **publicly verified** as correct.
- Verifiability should not leak information about the secret.

Employing PVSS

- Using PVSS enables parties to detect improper share distribution at the onset.
- Protocol can still be aborted, but any abort would be independent of the coin!

PVSS Design Challenges

- Assuming $\sum_{i=1}^N x_i = x$ $\psi_i = \mathcal{E}_i(x_i)$
 $\psi = \mathbf{Com}(x)$
- Verify that the value committed in ψ_i satisfies the equation with respect to the values encrypted in ψ

The cryptographic tool that solves the above problem is called a **zero-knowledge proof**.

Secure Multiparty Computation

- (Secure) Multiparty Computation (MPC)
 - Parameterized by function $f(\cdot)$
 - A set of n parties contribute inputs x_1, x_2, \dots, x_n
 - At the end of the protocol they compute $f(x_1, x_2, \dots, x_n)$

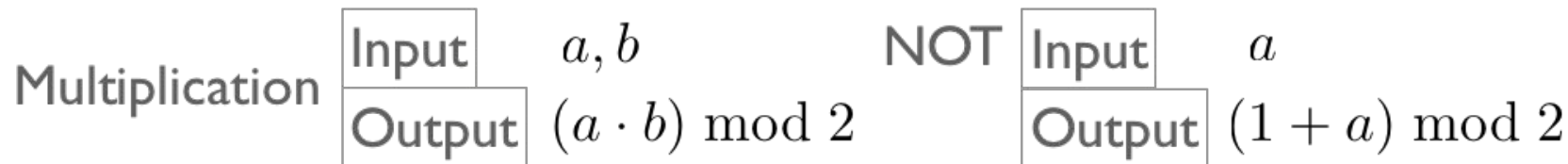
MPC Construction Idea

- Consider three roles:
 - Input-providers, Processors, Output-receivers
- Input providers secret-share their input to processors
 - Secret-sharing:

Additive Secret Sharing

MPC Construction Idea, II

Represent function f as Boolean circuit, e.g., XOR, AND, NOT and arithmetize it!



(any function can be implemented using these gates)

MPC Construction Idea, III

XOR GATE

- Suppose m parties hold shares of two inputs to an XOR gate.

$$[a], [b] = \langle a_1, \dots, a_m \rangle, \langle b_1, \dots, b_m \rangle$$

- How do they calculate shares of the output of the XOR gate?

$$[a] + [b] \bmod 2$$

MPC Construction Idea, IV

NOT GATE

- Suppose m parties hold shares of two inputs to a NOT gate.

$$[a] = \langle a_1, \dots, a_m \rangle$$

- How do they calculate shares of the output of the NOT gate?

$$[\bar{a}] = \langle 1 + a_1 \bmod 2, a_2, \dots, a_m \rangle$$

MPC Construction Idea, V

- Suppose m parties hold shares of two inputs to an AND gate.

AND GATE

$$[a], [b] = \langle a_1, \dots, a_m \rangle, \langle b_1, \dots, b_m \rangle$$

- How do they calculate shares of the output of the AND gate?

$$[a] \cdot [b] = \langle a_1 b_1 \bmod 2, \dots, a_m b_m \bmod 2 \rangle$$

but we want: $s_1 + \dots + s_m = \left(\sum_{i=1}^m a_i \right) \left(\sum_{i=1}^m b_i \right)$

MPC Construction Idea, VI

- A Beaver triple is an initial secret-sharing of random values

$$x \cdot y = z$$

$$[x] = \langle x_1, \dots, x_m \rangle, [y] = \langle y_1, \dots, y_m \rangle, [z] = \langle z_1, \dots, z_m \rangle$$

AND GATE :

publish $d_i = a_i - x_i$ reconstruct d, e

$$e_i = b_i - y_i$$

define $s_i = de + dy_i + ex_i + z_i$ share calculation

$$\sum s_i = de + d \sum y_i + e \sum x_i + xy \quad (\text{assuming } m \text{ is odd})$$

$$\begin{aligned} &= de + dy + ex + xy = (a - x)(b - y) + (a - x)y + (b - y)x + xy \\ &= ab \end{aligned}$$

Constructing Beaver Triples

The above construction idea requires the setup of all servers with a sufficient number of Beaver triples (how many?)

- Constructing Beaver triples can be done via a special purpose cryptographic protocols.

MPC systems to check

- SCALE MAMBA
 - <https://homes.esat.kuleuven.be/~nsmart/SCALE/>
- LIBSCAPI
 - <https://github.com/cryptobiu/libscapi>

MPC strengths and weaknesses

- Possible to compute any function f privately on parties' inputs.
- **But**, unless honest majority is present, there is no way to provide:
 - fairness: either all parties learn the output or none.
 - guaranteed output delivery.

Workarounds for fairness

- Optimistic fairness (by involving a third party). The protocol is basically not fair, but a third party is guaranteed to be able to engage and amend the execution in case of an abort/deviation.
- Gradual/timed release. Such protocols engage in many rounds where parties gradually come closer to computing the output (“gradual closeness” can be measured in terms of probability of guessing the output or # of computational steps remaining to compute the output).

Example of gradual release.

Protocol proceeds in rounds.

- At each round $l=1,\dots,n$ the two parties can compute the output in $2^{\{n-l\}}$ steps.
- If a party aborts the interaction the other party will be 2 times more steps “behind” in the calculation of the output.

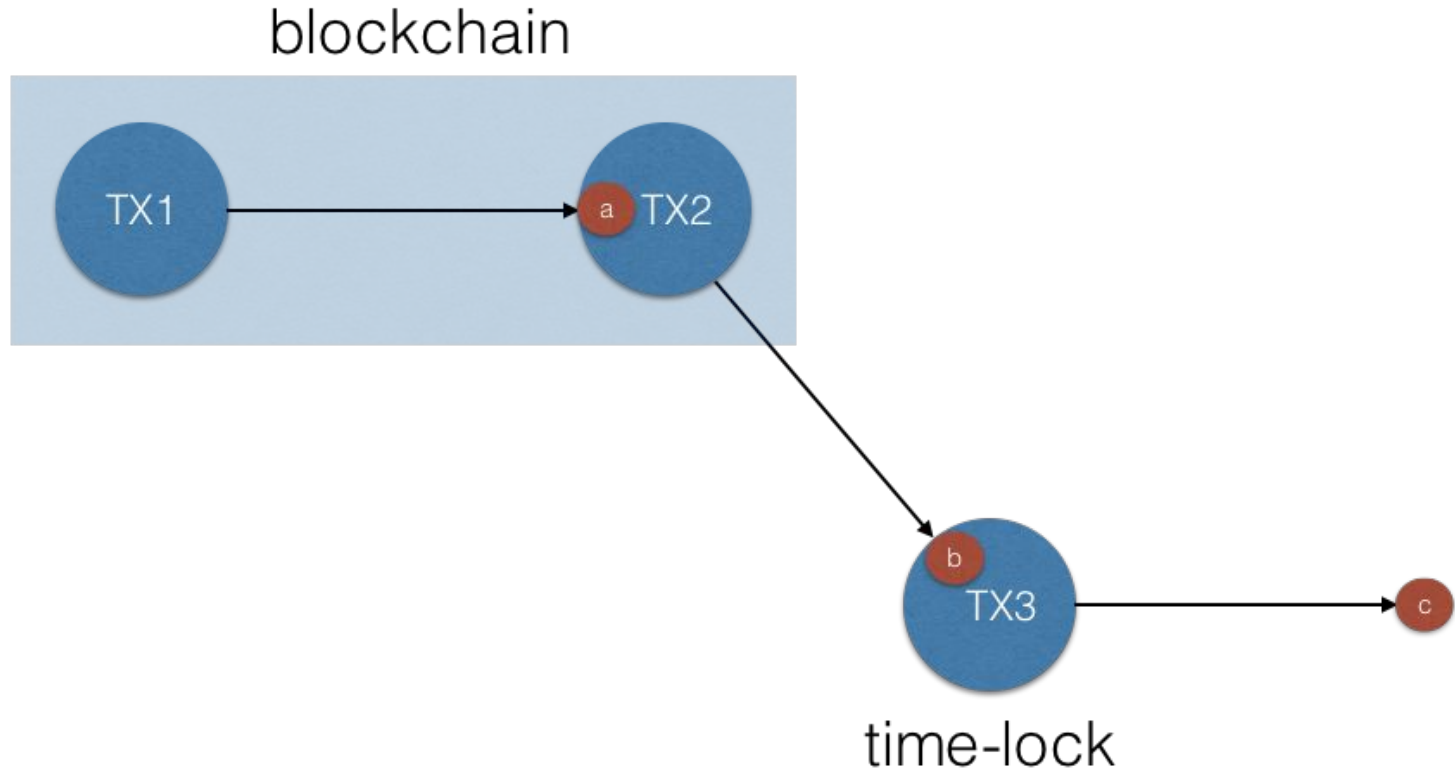
Using a blockchain

- Along the lines of optimistic fairness, but substituting the trusted third party with the blockchain.
- How is that possible? (blockchain cannot keep secrets)
Rationale: penalize parties that deviate from the protocol.

Basic tool : time-lock transactions

- Time-lock transactions
 - part of transaction data : specifies the earliest time that a transaction can be included in a block.
- Key observation: if a conflicting transaction has already being included in the ledger then the time-lock transaction will be rejected.

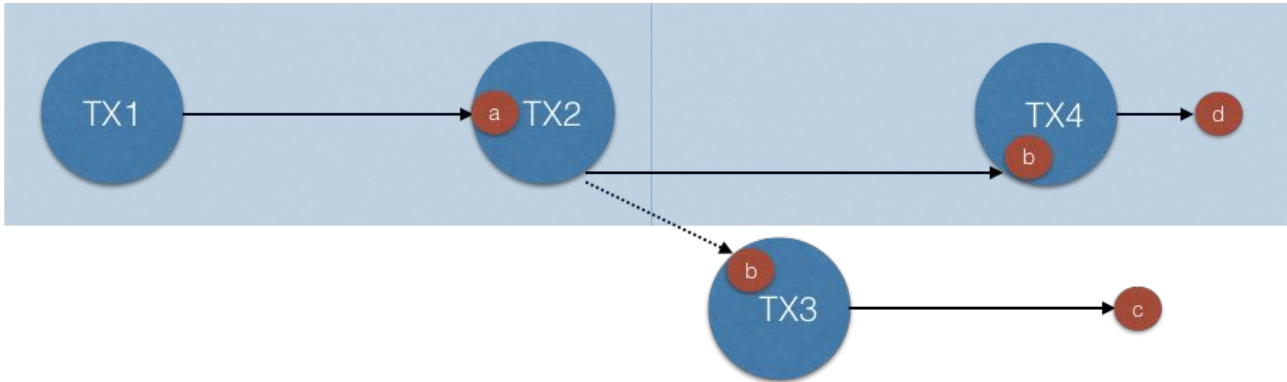
Time-lock example



Time-lock example, II



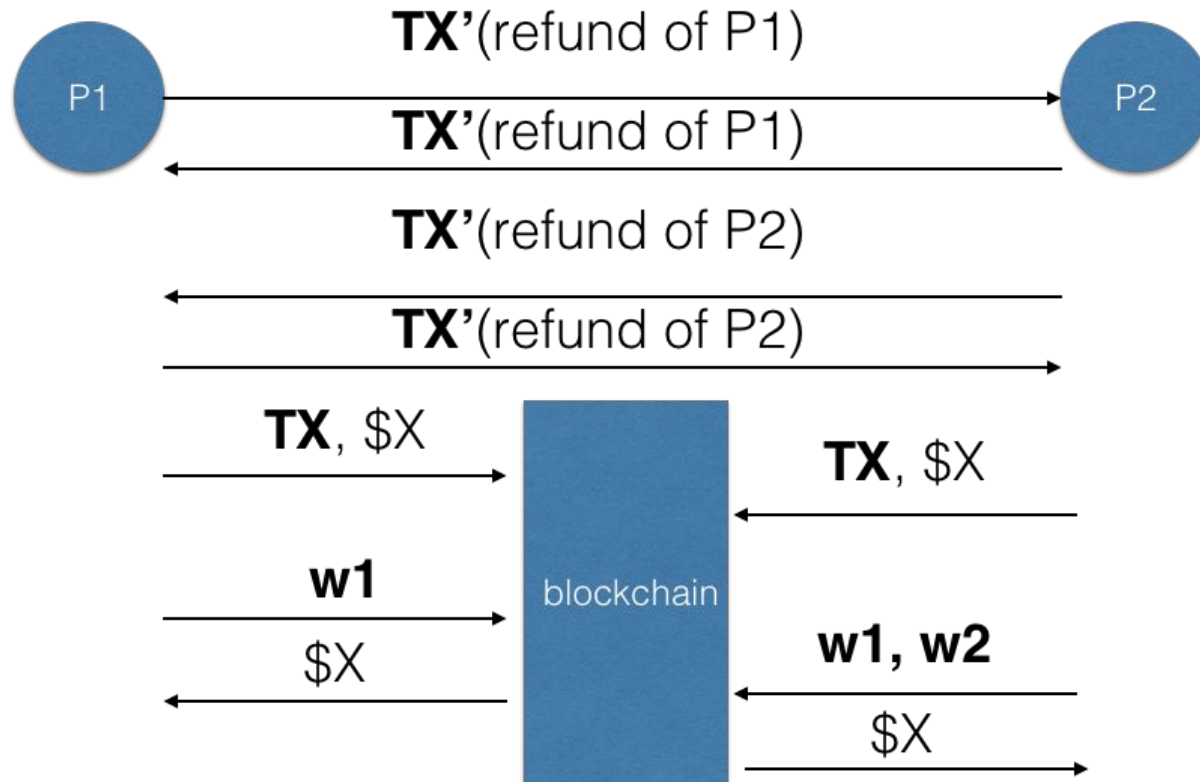
OR



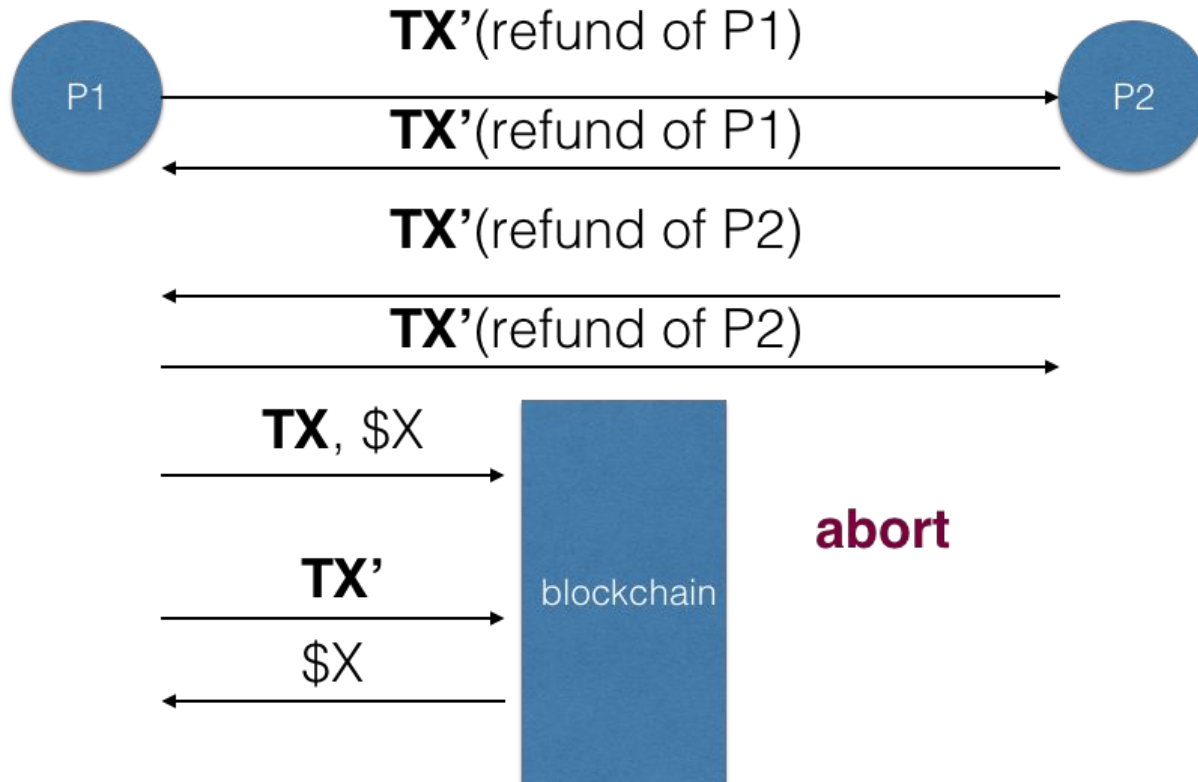
Fair swap of values using time-locks

- P1 holds w_1 , $h_2=H(w_2)$. P2 holds w_2 , $h_1=H(w_1)$. They want to exchange w_1, w_2 .
- P1 creates a P2SH transaction **TX** that can be redeemed for \$X provided that: (P1 and P2 sign, as 2-out-of-2 multisignature) or ($H(\dots)=h_1$ and $H(\dots)=h_2$ and P2 signs). P1 also creates a P2PKH **TX'** that spends **TX output with a time-lock in the near future**. P1 sends **TX'** to P2 to sign it. (P2 does not see **TX**, only the transaction ID is needed to refer to it).
- P2 acts in the same way by creating a **TX** that can be redeemed via a 2-out-of-2 multisig or ($H(w_1)=h_1$ and P1 signs it) and a corresponding **TX'**
- P1 by publishing **TX** enables P2 to redeem \$X revealing w_1 , w_2 . P2 by publishing its own **TX** enables P1 to redeem \$X revealing w_1 .
- If either party does not reveal their w value then the other can claim \$X after the time-lock is activated by signing & publishing **TX'**.

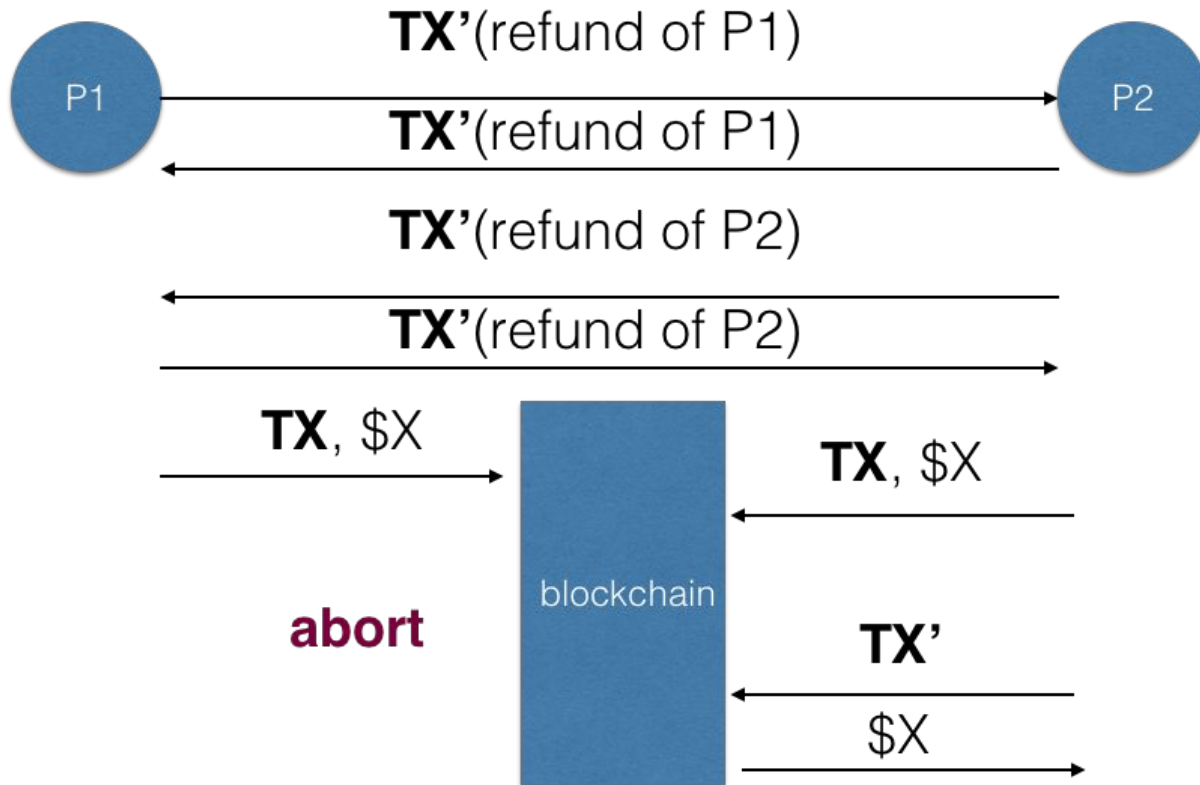
Fair swap of values using time-locks, II



Fair swap of values using time-locks, II



Fair swap of values using time-locks, III



Fair swap of values using time-locks, IV

- If P1's **TX** can be redeemed by " $H(w2)=h2$ and P2 signs it" then P2 can rush to obtain payment of \$X by revealing **w2** while hold back from publishing its own **TX** transaction. This enables P1 to obtain the output but they lost \$X.
(note that there is no way to ensure that the respective **TX** transactions will appear concurrently in the blockchain).
- Similarly, if a multisig is not used for the refunds, then a player may rush to obtain its refund after submitting its value thus invalidating the **TX** payment of the other player.

Fair Computation

- Have the two parties use MPC to compute a secret sharing of the output of the computation, i.e.,
 $w1 + w2 = \text{MPC_output}$
- Subsequently have parties do a fair swap of values, to obtain the MPC_output.
 - If a party aborts, then the other will be compensated.

N-party ladder construction, I

- Uses N -out-of- N multisig for refunds.
- P_N can redeem $\$X$ from each player if it reveals $\mathbf{w1}, \mathbf{w2}, \dots, \mathbf{wN}$. (i.e., the $N-1$ parties prepare these “roof” **TX** transactions).
- For $i=1, \dots, N-1$, player P_{N-i} can redeem from player P_{N-i+1} an amount equal to $\$X(N-i)$ if it reveals $\mathbf{w1}, \mathbf{w2}, \dots, \mathbf{wN-i}$. (The $N-1$ parties also prepare these “ladder” **TX** transactions).
- Redeeming follows the sequence P_1, P_2, \dots, P_N .

N-party ladder construction, II

- P1 will redeem $\$X$ from P2 for publishing **w_1** .
- P2 will redeem $\$2X$ from P3 for publishing **w_1, w_2** .
- ...
- P_{N-1} will redeem $\$(N-1)X$ from P_N for publishing **w_1, w_2, \dots, w_{N-1}** .
- P_N will redeem $\$X$ from each of P_1, \dots, P_{N-1} for publishing **w_1, w_2, \dots, w_N** .