Implementing a Decentralized Trust Inference System

Christos Porios

BEng Personal Project 2017
Supervised by Anandha Gopalan

Introduction & Motivation

Motivation

Suppose you want to buy something from a peer-to-peer marketplace online:

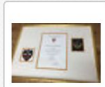
- How do you know the other party won’t try to steal your money / commit fraud by sending an item of inferior quality?
- How do you establish how much I can trust the seller for?

Most popular example: eBay, ~250B $ in transactions annually.

ebay Shop by category | Search... | All Categories | Search | Advanced

Back to search results | Listed in category: Collectibles > Historical Memorabilia > Teaching & Education > Colleges & Universities

EXTRA 10% OFF WHEN YOU SPEND $50 OR MORE See all eligible items



Have one to sell? Sell now

University of London Framed Degree Doctorate Philosophy Crest Patches M Marshak

Add to watch list

Item condition: **Used**

"*In very good condition. Some minor wear to the frame.*"

Sale ends in: 06d 10h 54m

Was: ~~US $999.99~~

You save: $150.00 (15% off)

Price: **US $849.99**

From $40 for 24 months

Buy It Now

Add to cart

Best Offer: Make Offer

Add to watch list

Add to collection

30-day returns | Experienced seller | Best offer available

Seller information

thequeenofauctions (33144)

99.9% Positive feedback

Top Rated Plus

Follow this seller

Visit store: The Queen of Aucti...

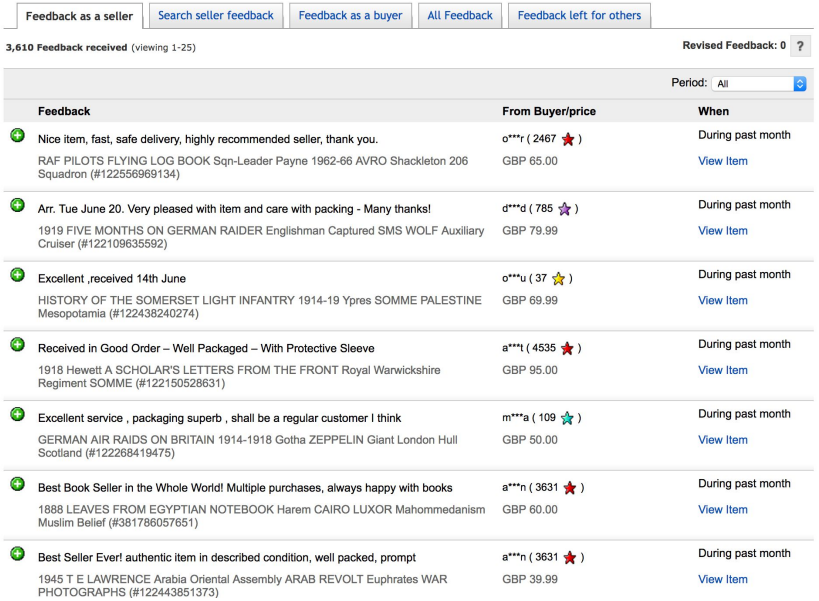
See other items

Reputation systems: eBay

On eBay, users rate each other after every transaction.

Past good behaviour indicates good intentions in the present.

Fraud still happens occasionally, but generally eBay works fairly well for most people.



Feedback as a seller | Search seller feedback | Feedback as a buyer | All Feedback | Feedback left for others

3,610 Feedback received (viewing 1-25) — **Revised Feedback: 0**

Period: All

Feedback	From Buyer/price	When
Nice item, fast, safe delivery, highly recommended seller, thank you. RAF PILOTS FLYING LOG BOOK Sqn-Leader Payne 1962-66 AVRO Shackleton 206 Squadron (#122556969134)	o***r (2467) GBP 65.00	During past month View Item
Arr. Tue June 20. Very pleased with item and care with packing - Many thanks! 1919 FIVE MONTHS ON GERMAN RAIDER Englishman Captured SMS WOLF Auxiliary Cruiser (#122109635592)	d***d (785) GBP 79.99	During past month View Item
Excellent ,received 14th June HISTORY OF THE SOMERSET LIGHT INFANTRY 1914-19 Ypres SOMME PALESTINE Mesopotamia (#122438240274)	o***u (37) GBP 69.99	During past month View Item
Received in Good Order – Well Packaged – With Protective Sleeve 1918 Hewett A SCHOLAR'S LETTERS FROM THE FRONT Royal Warwickshire Regiment SOMME (#122150528631)	a***t (4535) GBP 95.00	During past month View Item
Excellent service , packaging superb , shall be a regular customer I think GERMAN AIR RAIDS ON BRITAIN 1914-1918 Gotha ZEPPELIN Giant London Hull Scotland (#122268419475)	m***a (109) GBP 50.00	During past month View Item
Best Book Seller in the Whole World! Multiple purchases, always happy with books 1888 LEAVES FROM EGYPTIAN NOTEBOOK Harem CAIRO LUXOR Mahommedanism Muslim Belief (#381786057651)	a***n (3631) GBP 60.00	During past month View Item
Best Seller Ever! authentic item in described condition, well packed, prompt 1945 T E LAWRENCE Arabia Oriental Assembly ARAB REVOLT Euphrates WAR PHOTOGRAPHS (#122443851373)	a***n (3631) GBP 39.99	During past month View Item

Sybil attacks

“*In a **Sybil attack**, the attacker subverts the reputation system of a peer-to-peer network by creating a large number of pseudonymous identities, using them to gain a disproportionately large influence.***”** - Wikipedia

The practicality of Sybil Attacks depends on how cheap it is to create fake users accounts and reviews.

On eBay creating new accounts is intentionally made expensive:

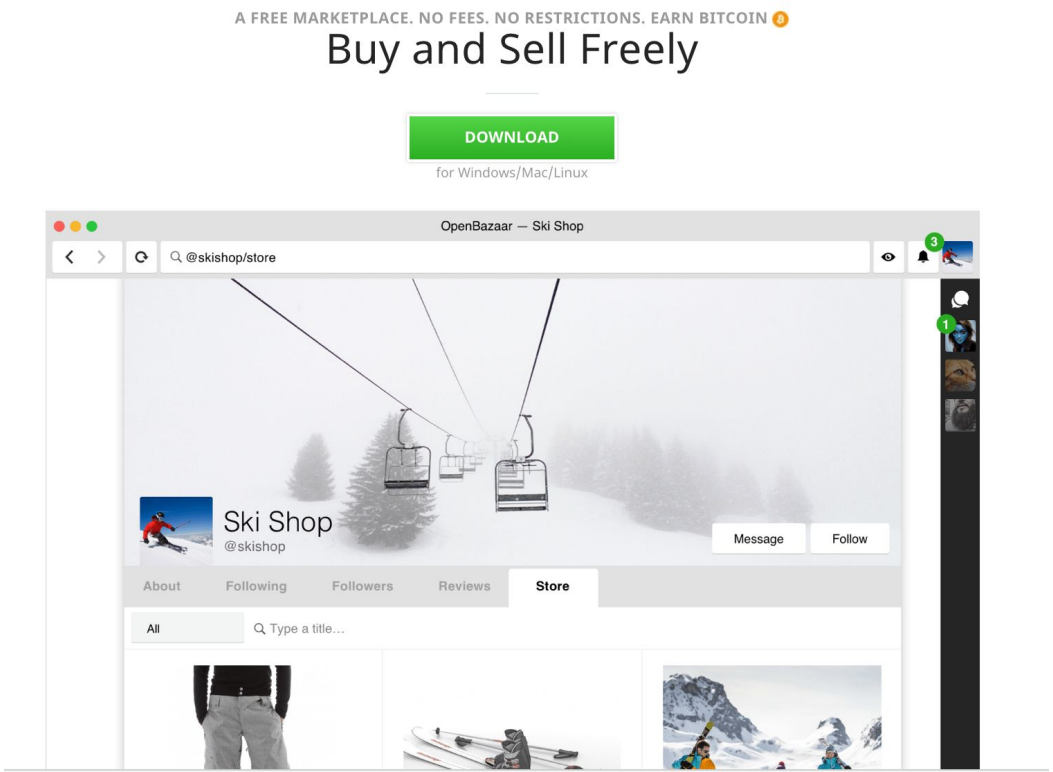
- A payment card is uniquely associated with a user.
- Captchas
- eBay transactions fees

A completely decentralized marketplace

OpenBazaar

- *No* transaction fees
- Buy and selling *anything*
- *No* central regulating authority
- *Pseudonymous* users

Operating, and actively under development by the VC-funded OB1.



How do we establish trust in OpenBazaar?

- The simple eBay-like feedback system in use is **Sybil attackable**:
  - Creating new pseudonymous users is as **cheap** as it can be.
  - **No transaction costs**.
  - **No regulating authority** that can “manually intervene”.
  - **Uses bitcoin**, so it’s difficult to catch fraudsters.
- Can we build a system which lets us trust complete strangers, who are **pseudonymous**?
- Can we make such a system **sybil resilient**, **inexpensive** and **easy to use**?

Problem Statement

To build a **general trust system** that allows its users to infer how much they can trust a **pseudonymous** stranger online, in a **decentralized** setting, while being **inexpensive**, **usable** and **Sybil attack resilient**.

Applicable to multiple applications, not just OpenBazaar!

Related Work

TrustDavis

- Participants accept limited liability for the misbehaviour of others.
- References are sold and bought.
- Sybil resilient in a decentralized setting
- Expensive to obtain references to “insure” yourself fully against fraud.

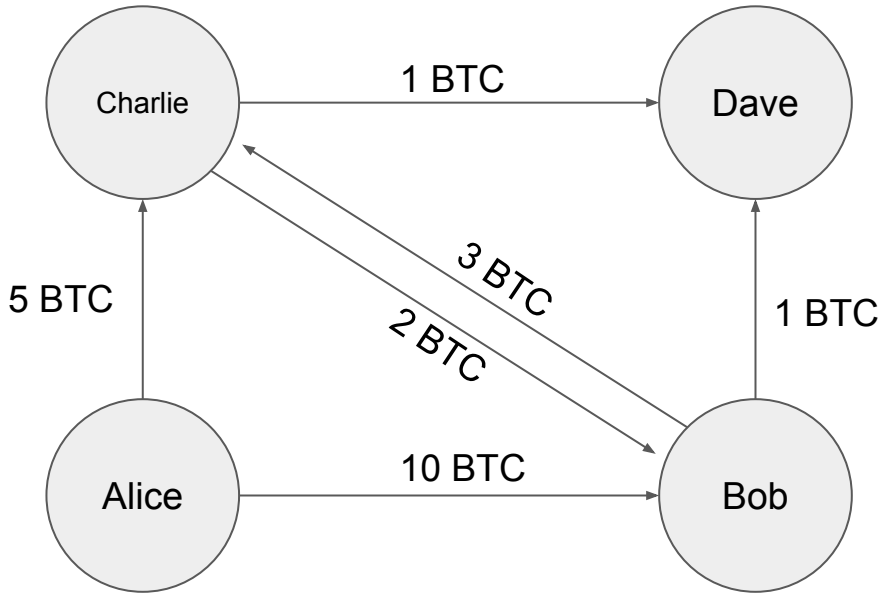
Trust Is Risk

- An inexpensive, **provably sybil-resilient** trust inference system.
- Three important properties:
  - **Monetary trust**
  - **Sybil attack resilience**
  - **Risk invariance**: When transacting with a pseudonymous entity up to a *spending allowance* which the system provides, the risk remains invariant.
- Paper by Dionysis Zindros, Orfeas Litos.
- Presented in FC'17 in April 2017.
- **The paper chosen to be implemented as the first deliverable of this project.**

Trust Is Risk

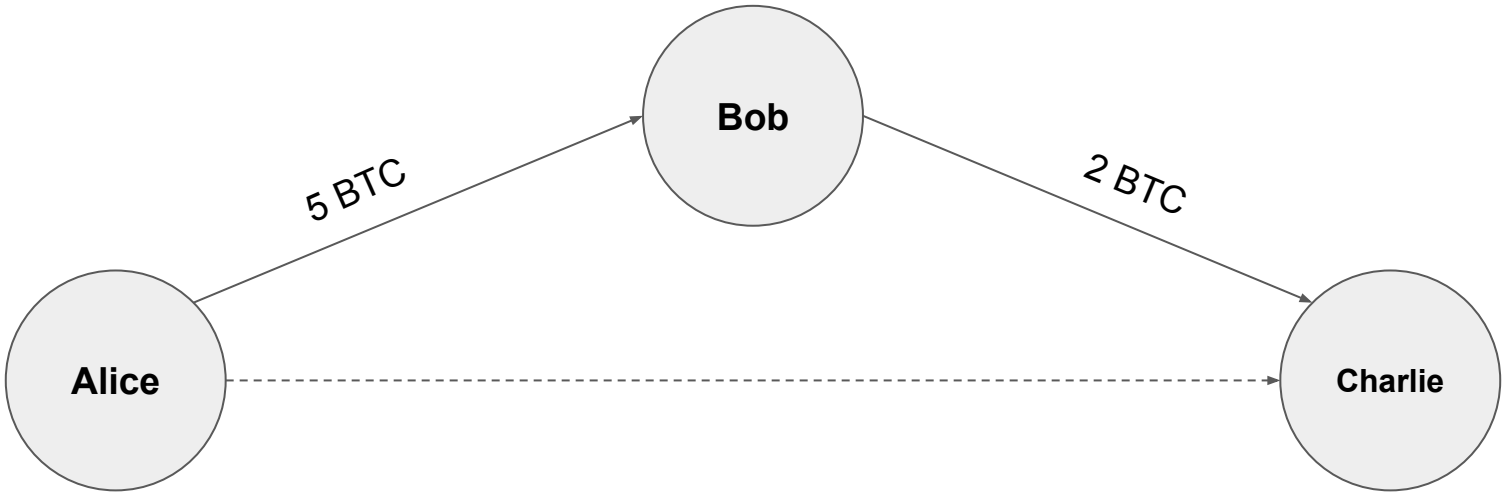
- Trust expressed in monetary terms:
  - *Alice trusts Bob for 5 BTC* instead of *Alice trusts Bob a lot*
- Direct trust is expressed by provably giving someone the capability, but not permission to spend some of your money:
  - *Alice trusts Bob for 5 BTC* means that Alice is placing 5 BTC of her own money in a “common bank account”, so that *both* Alice and Bob can spend it.
  - In Bitcoin, this is a 1-of-2 multisig output *1/{Alice, Bob}*.
- When Alice directly trusts Bob, she must be confident that Bob won’t take her money, based on any out-of-band reason:
  - Friendship or family
  - Legal contract
  - The system doesn’t care!

Trust Is Risk: Trust Graph

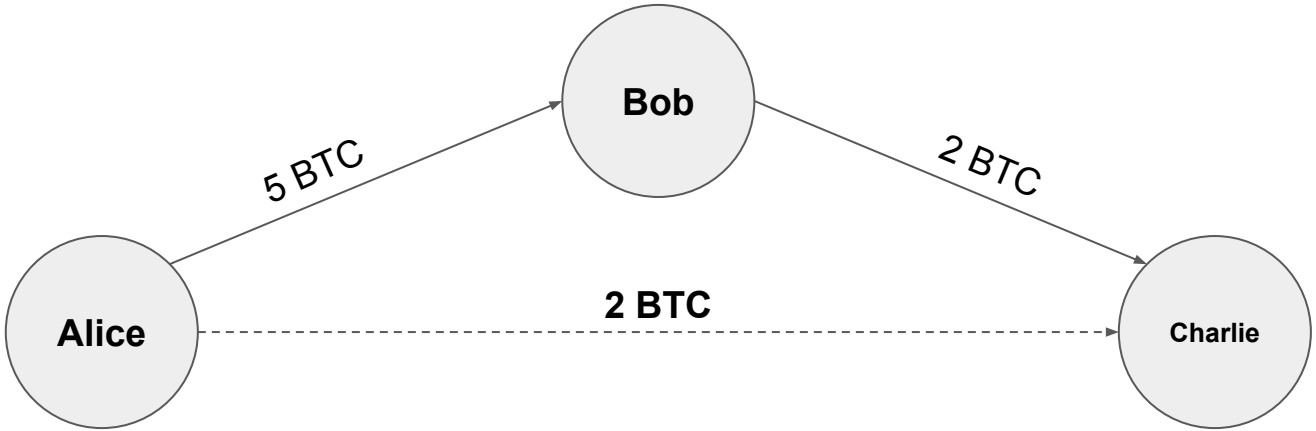


Trust Is Risk: Trust Transitivity

Alice trusts Bob* and *Bob trusts Charlie* => *Alice trusts Charlie



Trust Is Risk: Trust Transitivity



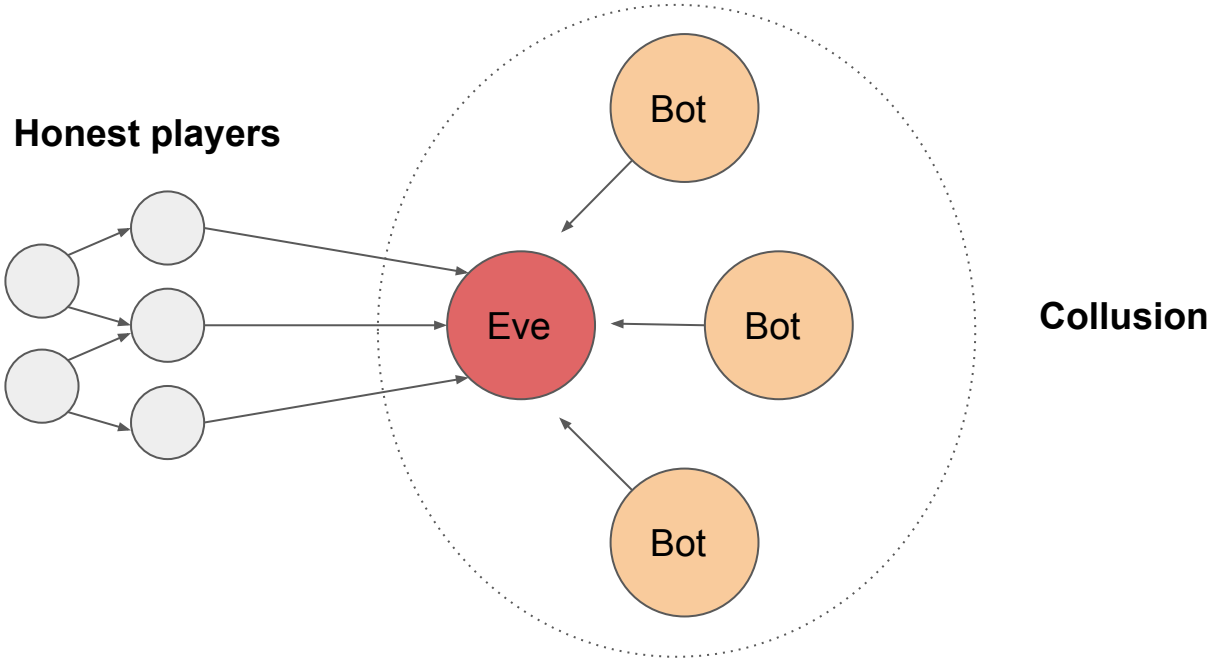
IndirectTrust(Alice, Bob) = MaxFlow(Alice, Bob)

Trust Is Risk: Risk Invariance Theorem

- **Assumption**: Alice wants to engage in a transaction with Charlie during which she will risk X BTC. Before the transaction she indirectly trusts Charlie for Y BTC with Y > X.
- **Then:** She can redistribute her outgoing direct trusts to reduce her indirect trust towards Charlie to Y − X.
- **Then when she engages in the transaction the money being risked remains invariant.**
- Thus, indirect trust towards a vendor is a **spending allowance**.
- Formal proof in the paper.

Trust Is Risk: Sybil Resilience

- Sybil attacks offer **no benefit** to the adversary Eve:

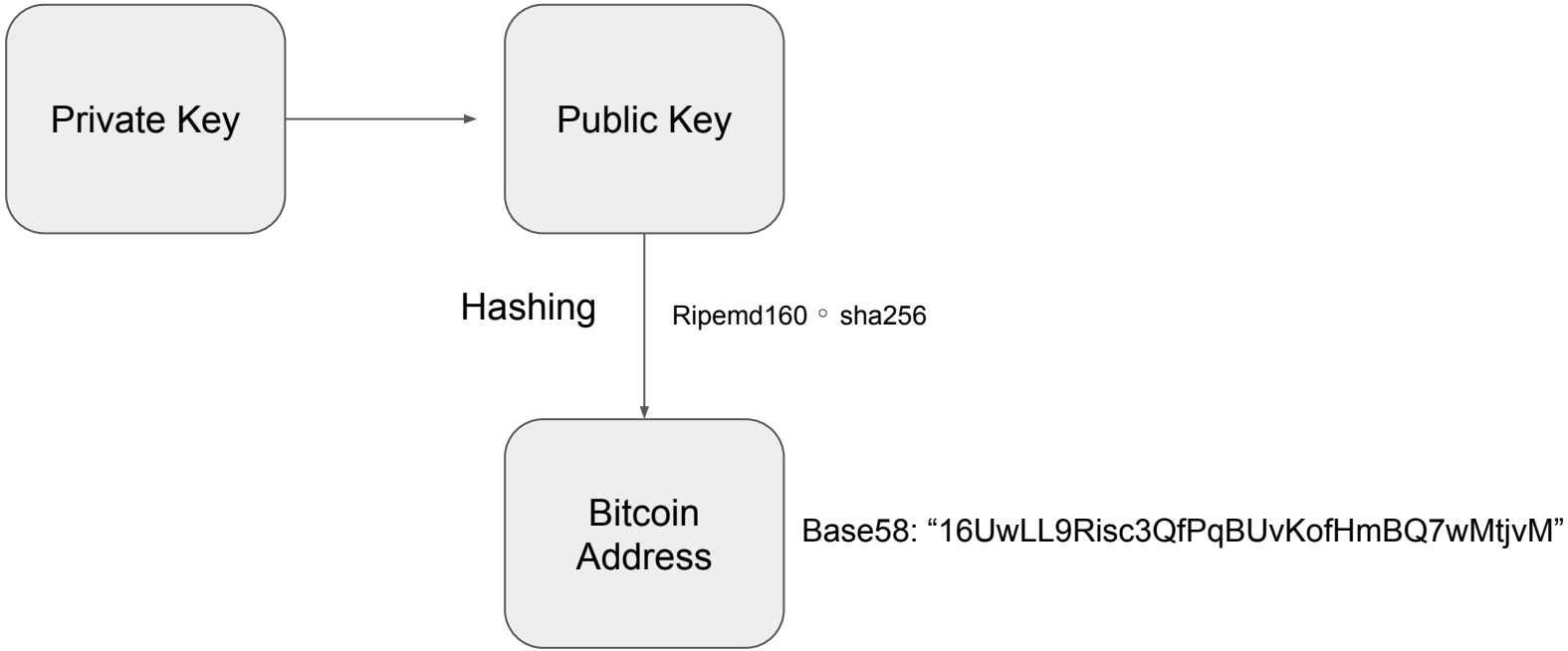


Bitcoin

Bitcoin

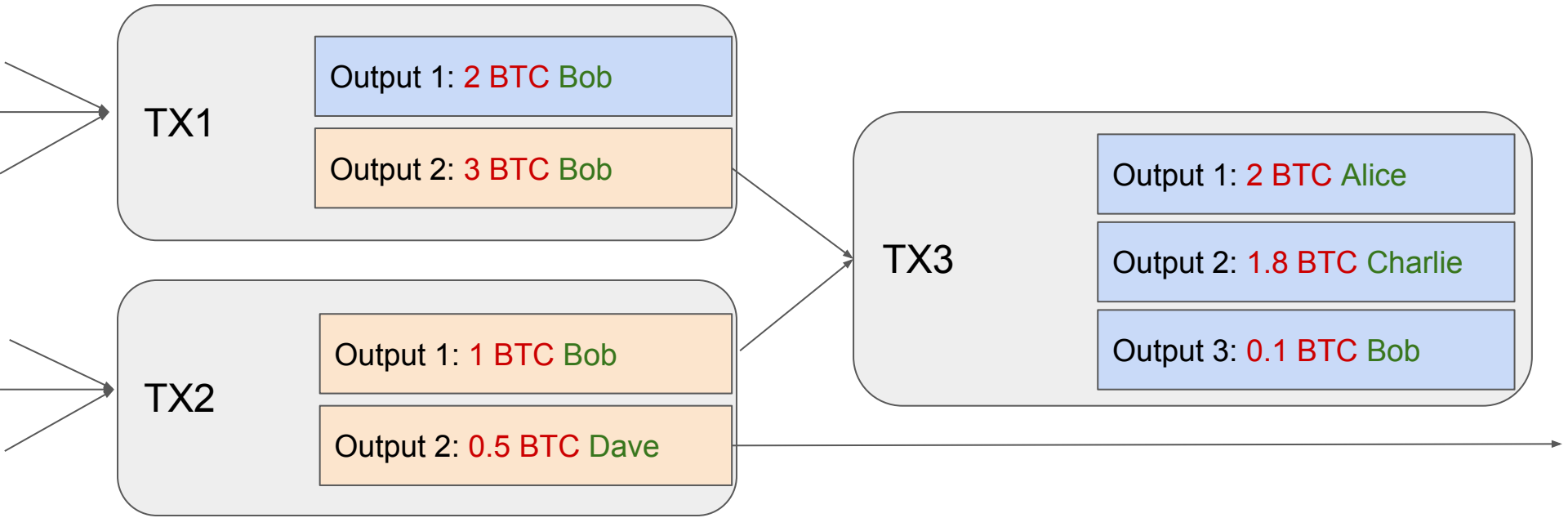
- **The most popular decentralized currency**.
- **We built the implementation of Trust Is Risk on top of Bitcoin**, to benefit from its proof-of-work consensus establishing mechanism.
- **Sparked a “crypto anarchy” culture**: Decentralization enthusiasts reject the idea that a central trusted authority is required for regulating many of the systems they interact with in their every-day lives

Bitcoin Addresses



Bitcoin Transactions

- The transfer of Bitcoin value from one entity to another.



Bitcoin Output Scripts

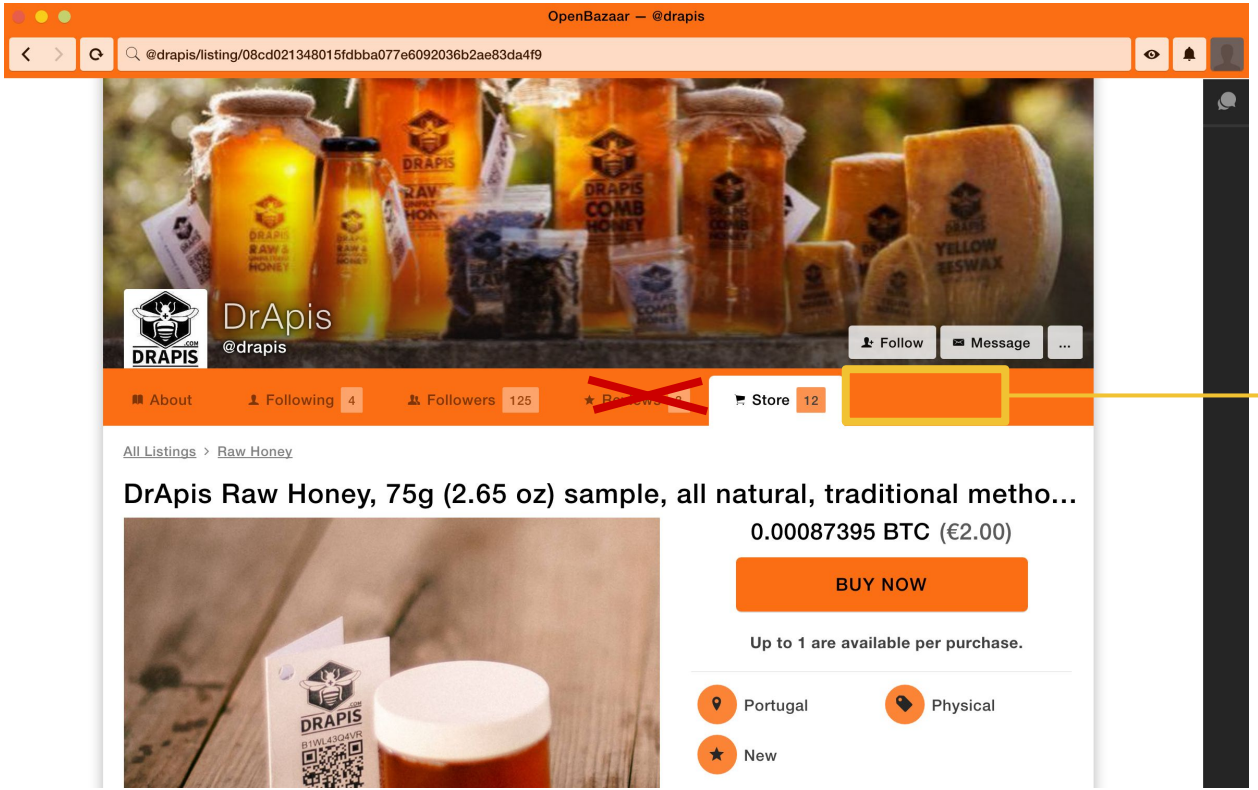
- **Defines who can spend an output**
- Most common type of script: Pay To Public Key Hash (P2PKH)
  - Requires the spender to prove they own the private key corresponding to a bitcoin address.
  - Can be thought of as simply *sending money to the owner of an address*.
- More complicated scripts exist.
- We will use 1-of-2 multi-signature outputs:
  - The sender makes an amount spendable by two people at the same time
  - In 1/{Alice, Bob}, **both Alice and Bob** can spend the money.

TrustIsRisk.js

TrustIsRisk.js

- The implementation of the Trust Is Risk paper.
- **The first-ever implementation of a generic, decentralized, sybil-resilient trust inference library.**
- Written in EcmaScript 7 (JavaScript) on top of the Bitcoin Blockchain.
- Essentially an extended Bitcoin Full Node.
- Developed with **OpenBazaar** in mind, but it can be used in any application!
- **Funded by OB1**, the company building OpenBazaar.
- Developed in collaboration with the paper authors and OpenBazaar developers.

TrustIsRisk.js: End Goal



Display a “spending allowance” here:

The indirect trust from the user towards the vendor

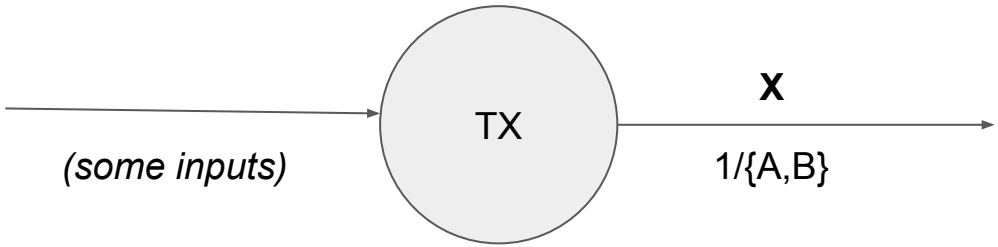
TrustIsRisk.js: Functionality

- **Encodes direct trust changes as bitcoin transactions**. Provides methods that modify the trust graph by producing bitcoin transactions that:
  - **Increase direct trust** from an entity the user controls to some other entity.
  - **Decrease direct trust** from an entity the user controls to some other entity.
  - **Decrease direct incoming trust** from some other entity towards an entity the user controls (stealing).
- **Maintains the current trust graph** by monitoring all bitcoin transactions on the network.
- Provides methods that let the user **inspect the trust graph** by:
  - **Getting the direct trust** between two entities.
  - **Getting the indirect trust** between two entities.
  - **Getting the full adjacency matrix** of the graph.

TrustIsRisk.js: Trust Direction Problem

Important principle: Direct trust is expressed by giving others the capability to spend your money, but not your permission.

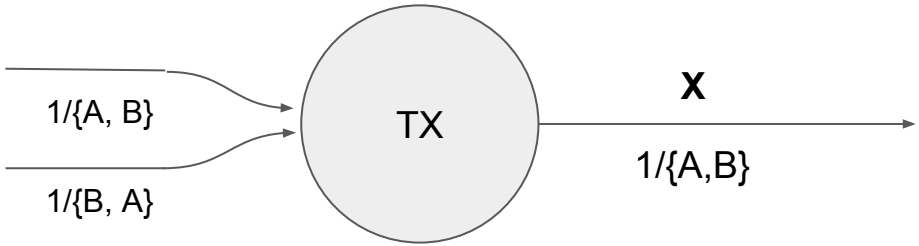
Therefore, **direct trusts must be based on (unspent) 1-of-2 multisig outputs**.



Problem: Is this direct trust from A to B, or direct trust from B to A?

TrustIsRisk.js: Trust Direction Problem

More complicated cases can arise: How do we parse this?



We need to define a strict format for trust-modifying transactions.

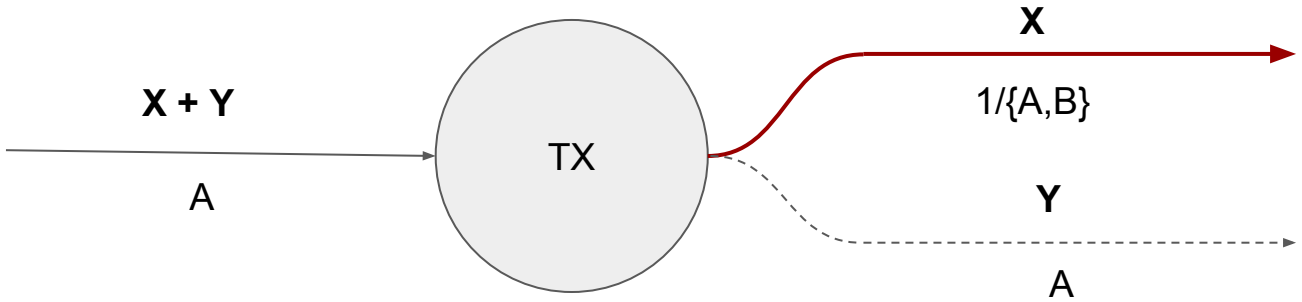
We want to preserve two properties:

1. Direct trust is always based on **unspent multisig outputs** of the format 1/{origin, dest}.
2. At most one trust relationship can be based on a single output.

TrustIsRisk.js: Trust Increasing Transactions

A trust increasing transaction increases trust from entity A to entity B by an amount of X.

- Exactly one input:
  - **P2PKH**: Spendable by A, with a value of at least X.
- One or two outputs:
  - **Trust output**: 1/{A, B} with a value of X.
  - **Change output** (optional): P2PKH spendable by A.



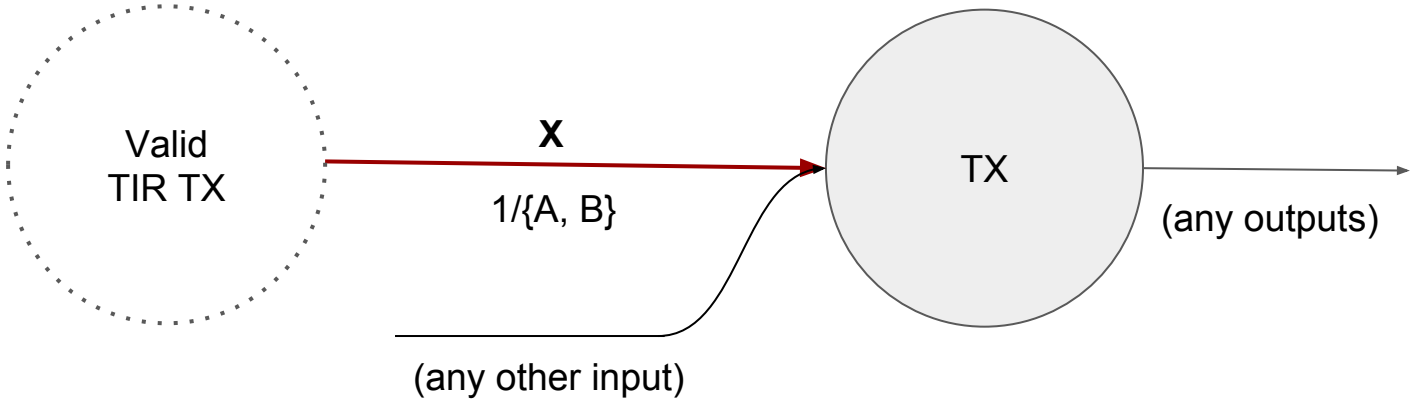
TrustIsRisk.js: Trust Decreasing Transaction

A trust decreasing transaction spends one or more trust outputs. Two kinds:

- **Proper trust decreasing transactions**: Decrease trust in a “controlled” way, by some amount X. Produced by the implementation.
- **Improper trust decreasing transaction**:
  - Serve to invalidate the trust output being spent.
  - Decrease trust by the value of the trust output being spent.
  - Necessary to ensure property (1) holds: “Direct trust is always based on unspent multisig outputs of the format 1/{origin, dest}.”.

TrustIsRisk.js: Improper Trust Decreasing Transaction

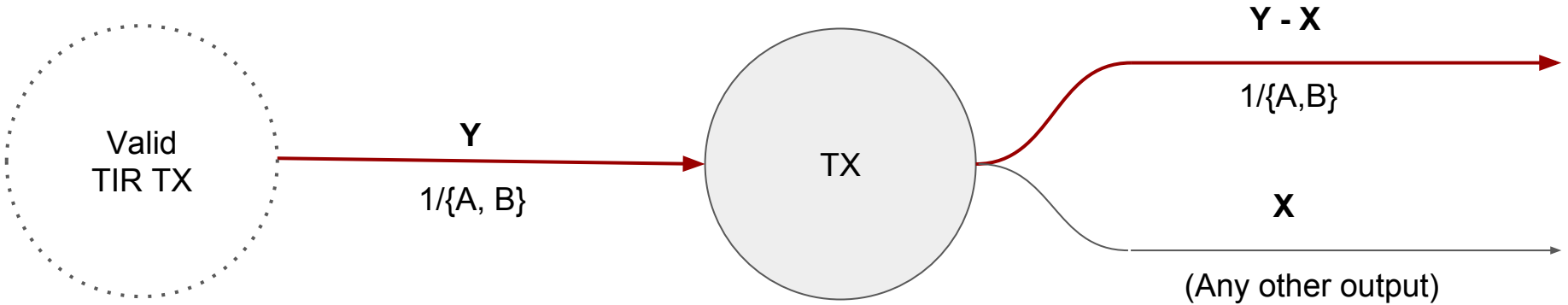
Spends a trust output of value X and reduces trust by X. No other restrictions.



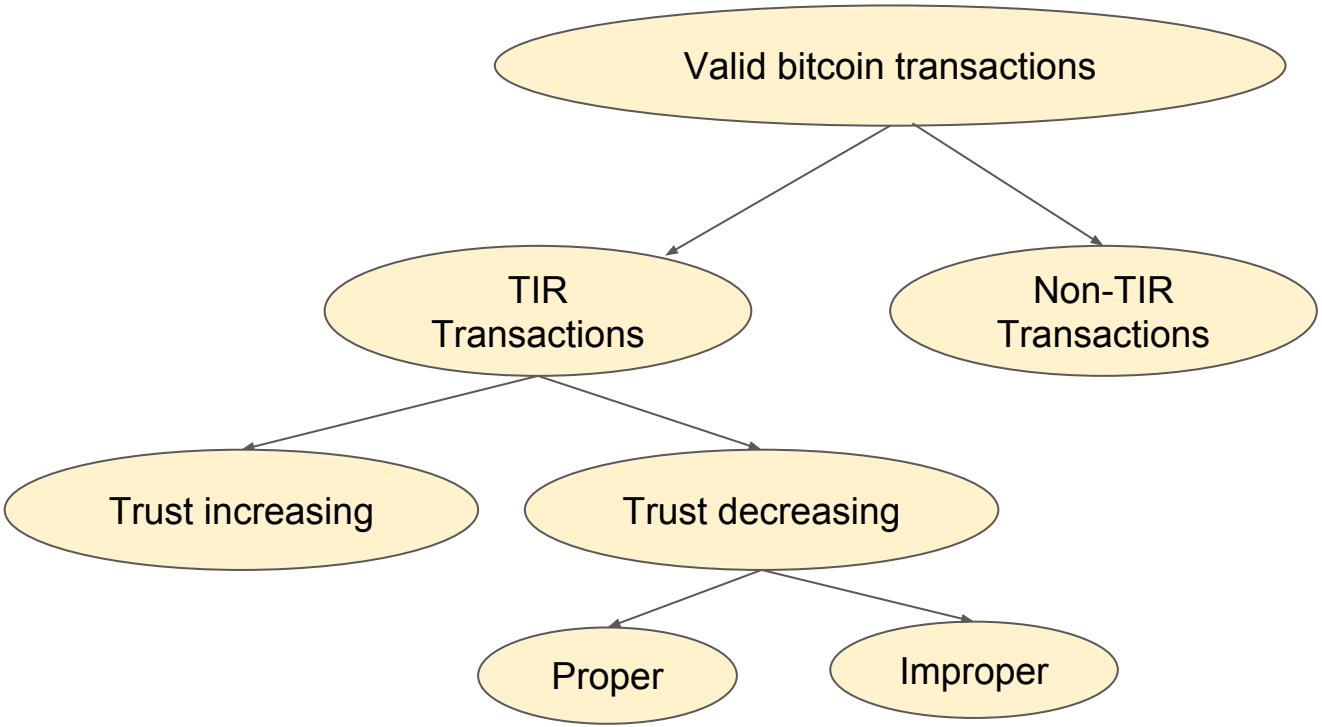
TrustIsRisk.js: Proper Trust Decreasing Transactions

Decreases trust from A to B by X. Format:

- Exactly one input:
  - **Trust output**: 1/{A, B} with value Y.
- One or two outputs:
  - **Trust output**: 1/{A, B} with value Y - X.
  - **Any other outputs** that are not multisigs.



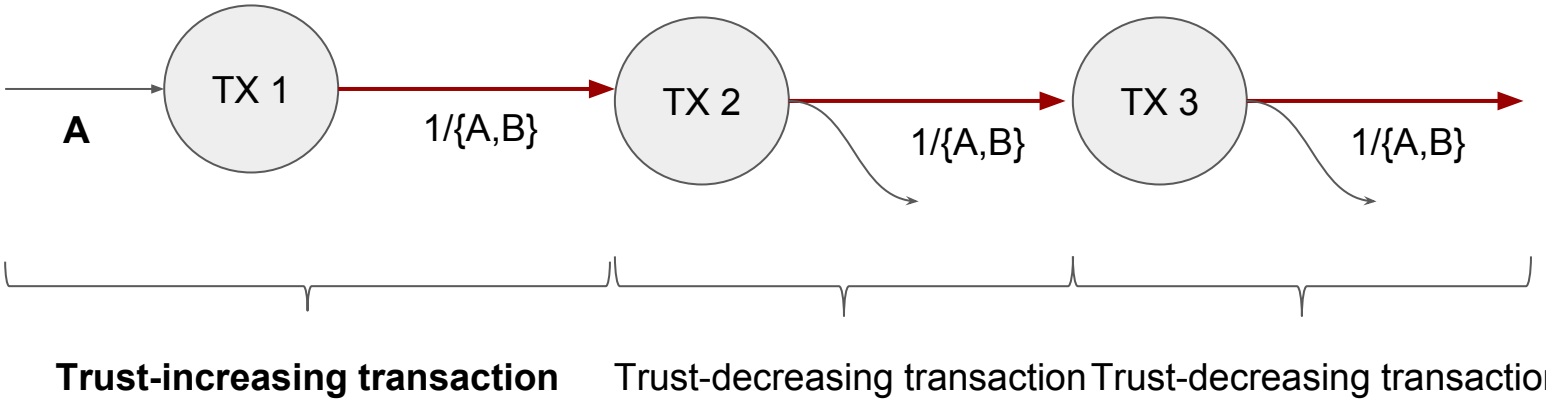
TrustIsRisk.js: Transaction Types



TrustIsRisk.js: Trust Chains

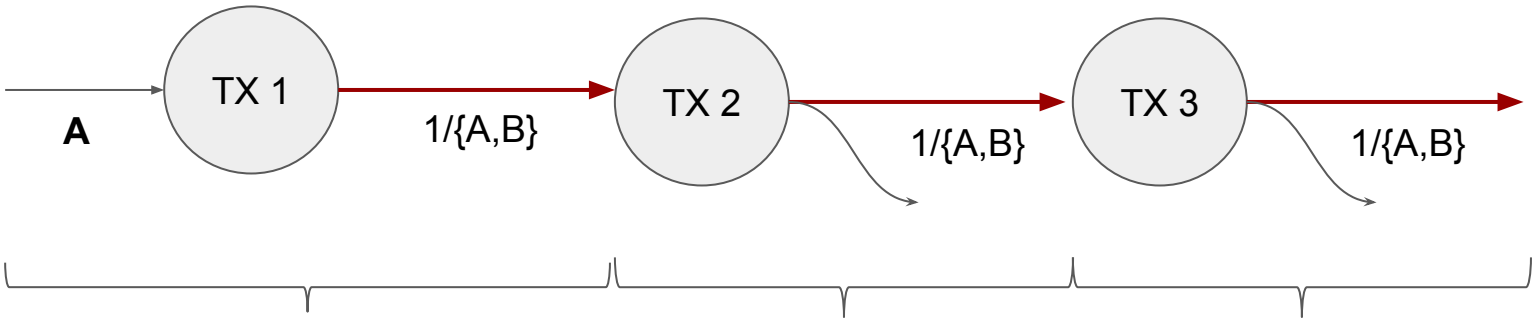
Trust chain: A series of TIR transactions that “carry” trust forward.

- Associated with a sender and a recipient.
- Always starts with a trust increasing transaction.
- Continues with trust decreasing transactions.
- An improper trust decreasing transaction always terminates the chain.

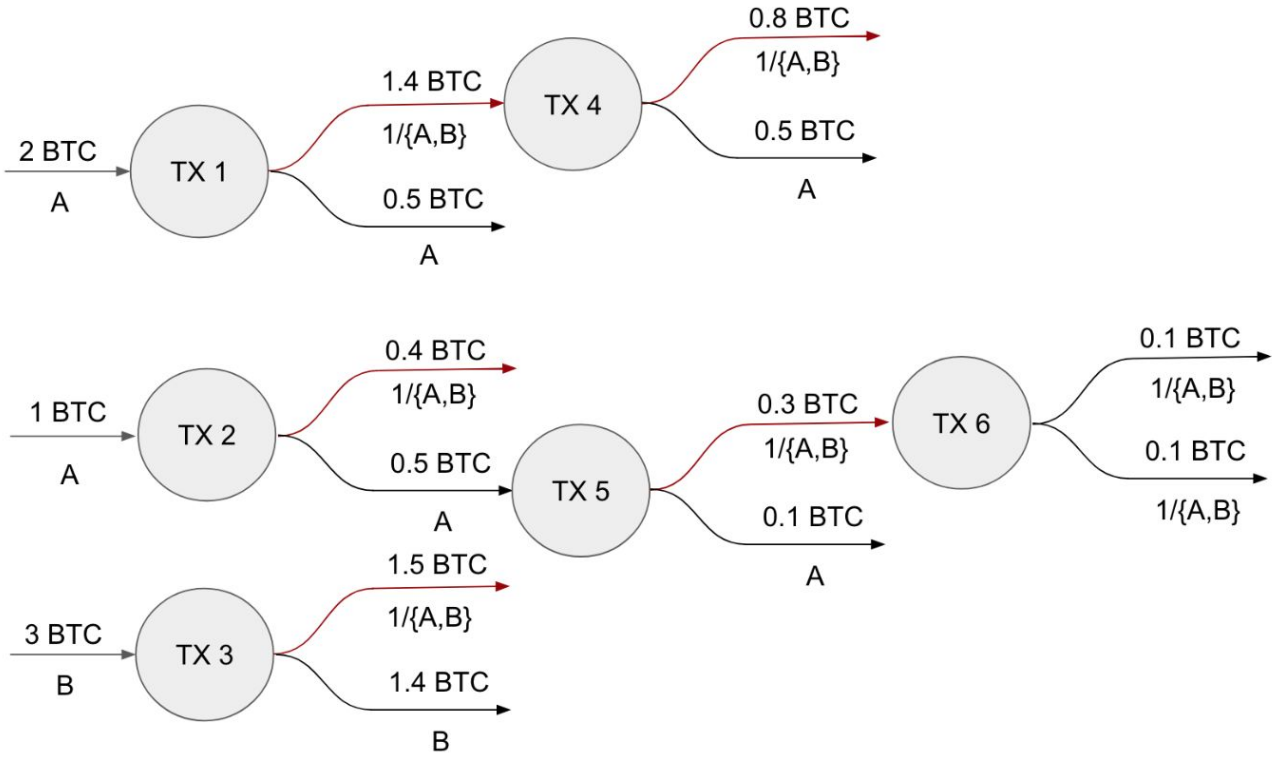


TrustIsRisk.js: Trust Chains

- We can prove the two properties by structural induction on trust chains.
- **The direct trust from A to B is the sum of the unspent trust outputs.**



TrustIsRisk.js: Example



TrustIsRisk.js: Technologies and Testing

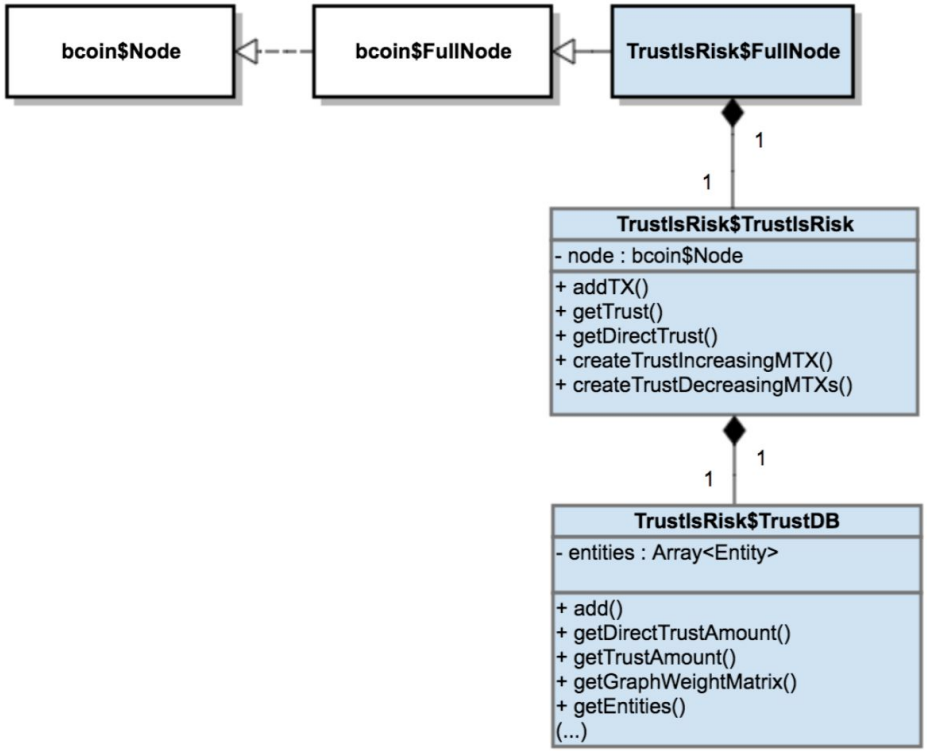
- ECMAScript 7
- Typed with Flow
- Bcoin for the Bitcoin Full Node
- Lots of unit and end to end tests

```
FullNode
  ✓ should call trust.addTX() on every transaction (1925ms)
  with the nobodyLikesFrank.json example
    ✓ computes trusts correctly
    ✓ after decreasing some trusts computes trusts correctly (755ms)

TrustIsRisk
  .getDirectTrust()
    ✓ returns zero for two arbitary parties that do not trust each other
    ✓ returns Infinity for one's direct trust to themselves
  .addTX()
    with a non-TIR transaction
      ✓ does not change trust
    with a trust increasing transaction
      ✓ correctly increases trust
      ✓ which has more than one input does not change trust
      ✓ which has a change output correctly increases trust
      ✓ which has two change outputs does not change trust
      ✓ which has a second output that is not a change output does not change trust
      ✓ which has been processed before throws
    with a trust decreasing transaction
      ✓ correctly decreases trust
      ✓ which has a second input decreases trust to zero
      ✓ which has more than one trust outputs decreases trust to zero
    .getTrustIncreasingMTX()
      ✓ creates valid trust-increasing transactions
    .getTrustDecreasingMTX()
      ✓ creates correct trust decreasing transactions
      ✓ creates correct trust stealing transactions
      ✓ throws when trying to decrease self-trust
      ✓ throws when there is not enough trust
    .getTrust()
      ✓ returns zero for two arbitary parties that do not trust each other
      ✓ returns Infinity for one's trust to themselves
      after applying the Nobody Likes Frank graph example
        ✓ correctly computes trusts
        ✓ correctly computes trusts when bob trusts frank


24 passing (13s)
```

TrustIsRisk.js: Design



TrustIsRisk.js: API example

```
var node = new TrustIsRisk.FullNode({
     network: testnet
     // ...other options...
});

await node.open(); await node.connect(); node.startSync();

node.trust.getDirectTrust(alice.address, bob.address);
node.trust.getTrust(alice.address, bob.address);

var mtxs = node.trust.getTrustDecreasingMTXs(
     alice.privKey,
     bob.pubKey,
     1000 );
mtxs.each((mtx) => node.sendTX(mtx.toTX()));
```

TrustIsRisk.js Demo

Trust In Friends

Trust In Friends: Overview and Intuition

- My own trust inference system.
- Theoretical concept; No implementation.
- Can be implemented in Ethereum.
- Basic principles:
  - Users insure other users against the misbehaviour of their friends. (TrustDavis)
  - This happens in a transitive way. (Trust Is Risk)
  - Indirect trust = Maximum Flow. (Trust Is Risk)
  - Entities further away in the trust graph are trusted less than entities that are closer.
- Properties:
  - Sybil resilience
  - **Fraud is only rational in the presence of incorrect direct trust relationships.**

Trust In Friends: Proof of Fraud

We assume that fraud can be proved without disputes by a machine.

If a party X commits fraud to deceive party A for an amount L, then A obtains a virtual *Proof of Fraud* token:

$$\mathrm{PoF}_{A \to X}^{L}$$

L: The token value
A: The beneficiary entity
X: The deceiver entity.

PoF tokens can be denominated into multiple tokens of smaller value.

Trust In Friends: The λ parameter

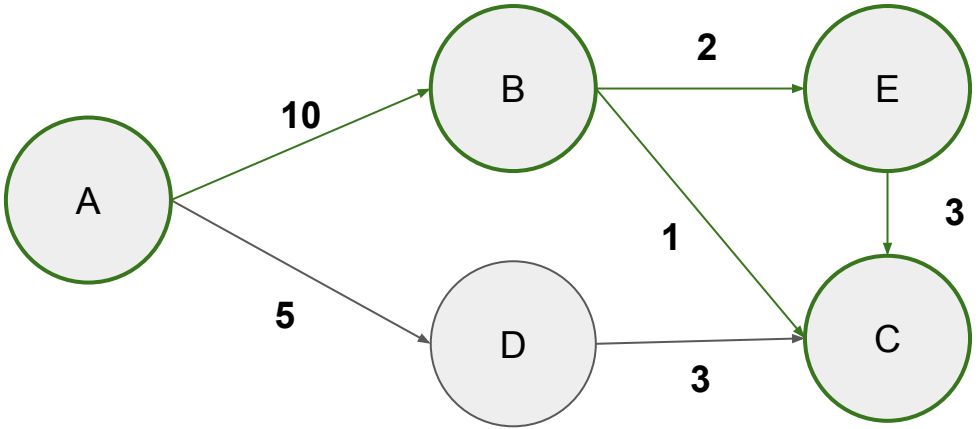
The system has a global parameter λ where $0 < \lambda \leq 1$ which has been agreed on beforehand. Two purposes:

- **Trust Attenuation**: Small λ => More distant entities are trusted less.
- **Participation incentive**: High λ => More incentive for parties to participate in the network.

Trust In Friends: Maximum Neighbour Flow

MNF(A, B, C) is the maximum flow from A to C, but by only considering paths that pass directly through B.

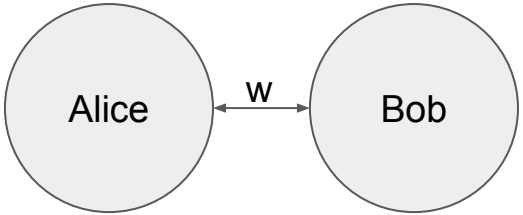
$$\text{MaxNeighbourFlow}(A,B,C) = \begin{cases} max(w(A,B), MaxFlow_{G\setminus(B,A)}(B,C)), & \text{if}(A,B) \in E \\ \text{undefined}, & \text{otherwise} \end{cases}$$



MaxFlow(A, C) = 7
MNF(A, B, C) = 4
MNF(A, D, C) = 3

Trust In Friends: Direct Trust

A direct trust relationship is a mutual promise between two entities:



I promise to always buy a PoF token from **Bob** for $L \cdot \lambda^d$:

- With any deceiver X that is d hops away in the trust graph.
- With **Bob** as the beneficiary.
- Of value L up to MNF(**Bob**, **Alice**, X)

\- *Alice*

I promise to always buy a PoF token from **Alice** for $L \cdot \lambda^d$:

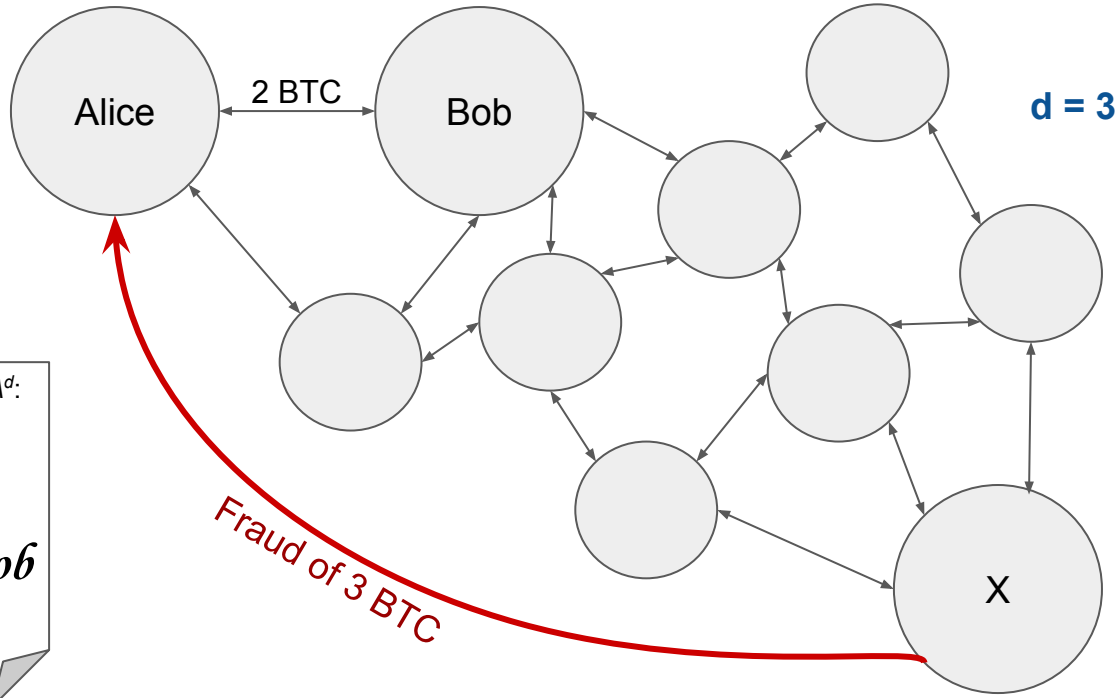
- With any deceiver X that is d hops away in the trust graph.
- With **Alice** as the beneficiary.
- Of value L up to MNF(**Alice**, **Bob**, X)

\- *Bob*

Trust In Friends: Direct Trust

A direct trust relationship is a mutual promise between two entities:

Suppose λ = 0.9 and MaxFlow(Bob, X) = 10 BTC



I promise to always buy a PoF token from **Alice** for $L \cdot \lambda^d$:

- With any deceiver X that is d hops away in the trust graph.
- With **Alice** as the beneficiary.
- Of value L up to MNF(**Alice**, **Bob**, X)

\- *Bob*

Trust In Friends: Direct Trust

A direct trust relationship is a mutual promise between two entities:

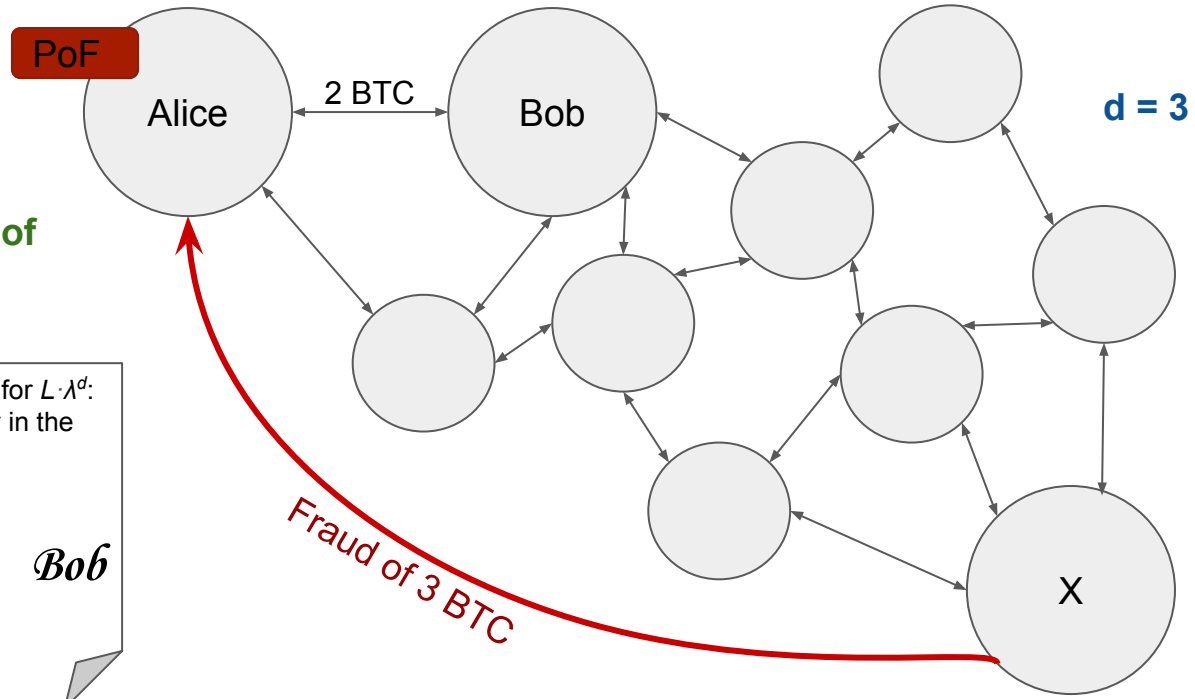
Suppose λ = 0.9 and MaxFlow(Bob, X) = 10 BTC

Bob now has to buy the PoF of value 2 from Alice for $2 \cdot 0.9^3 = 1.46$ BTC

I promise to always buy a PoF token from **Alice** for $L \cdot \lambda^d$:

- With any deceiver X that is d hops away in the trust graph.
- With **Alice** as the beneficiary.
- Of value L up to MNF(**Alice**, **Bob**, X)

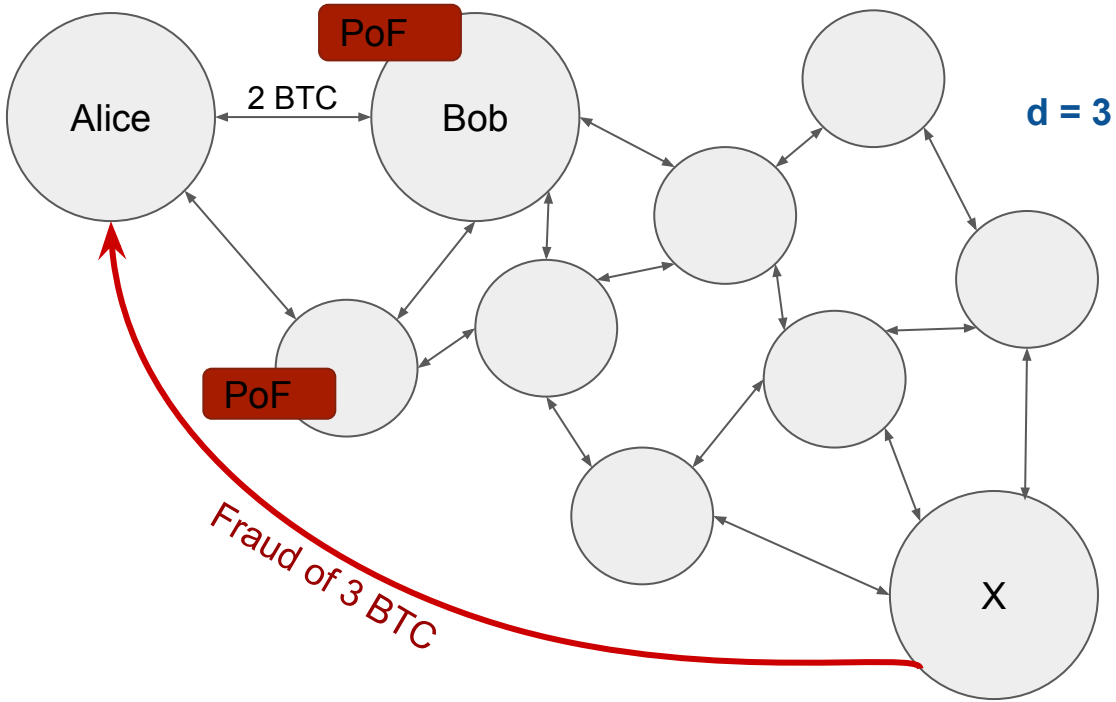
\- *Bob*



Trust In Friends: Direct Trust

A direct trust relationship is a mutual promise between two entities:

Bob will now will use the promise of his neighbours to sell the PoF to them.



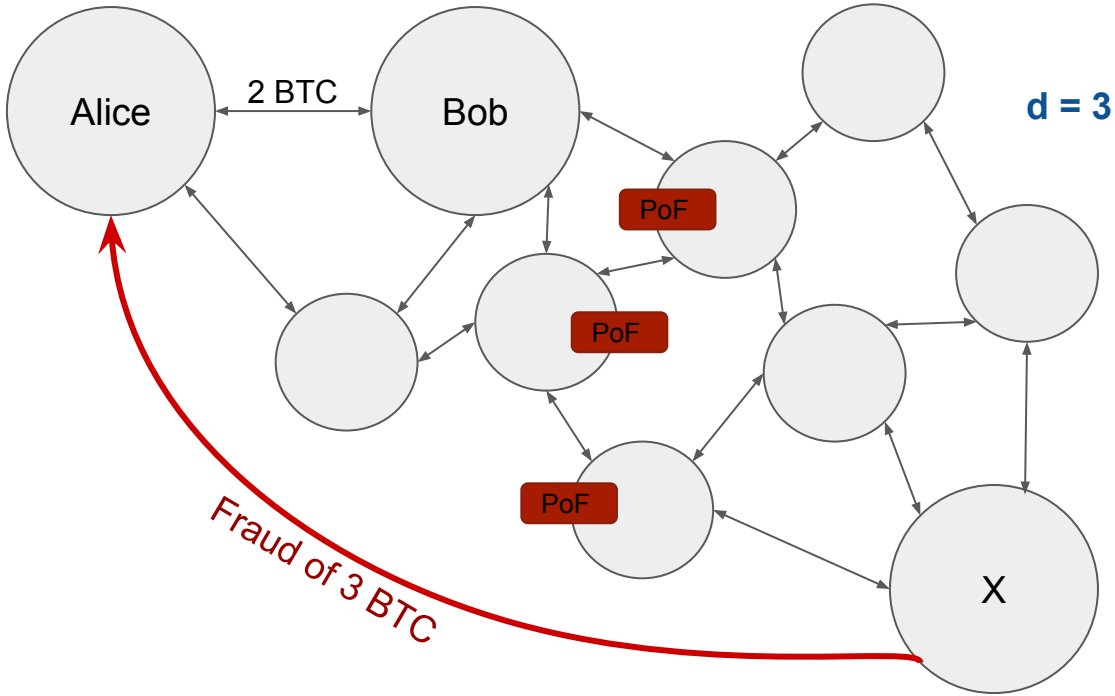
Trust In Friends: Direct Trust

A direct trust relationship is a mutual promise between two entities:

At each step, the PoFs move closer to X.

Alice recovered most of her loss.

Intermediaries like Bob make a small profit.



Trust In Friends: Indirect Trust

The indirect trust from A to B, T(A, B) is the MaxFlow(A, B) in the trust graph.

Like in Trust Is Risk, indirect trust is a “*spending allowance*”.

Makes the system sybil-resilient: same intuition as in Trust Is Risk.

Trust In Friends: Wrong direct trust relationships

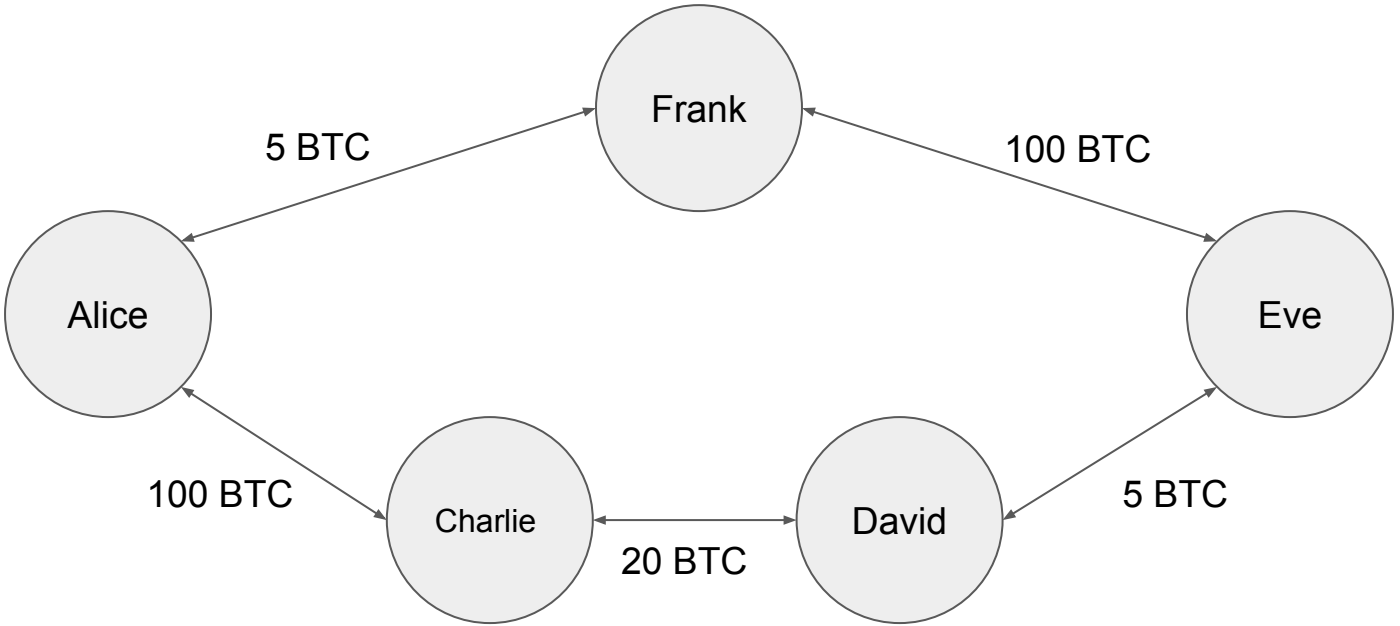
In every direct trust relationship, there is an out-of-bound cost for breaking the trust promise:

- Consequences in the underlying friendship?
- Breach of an actual legal contract?

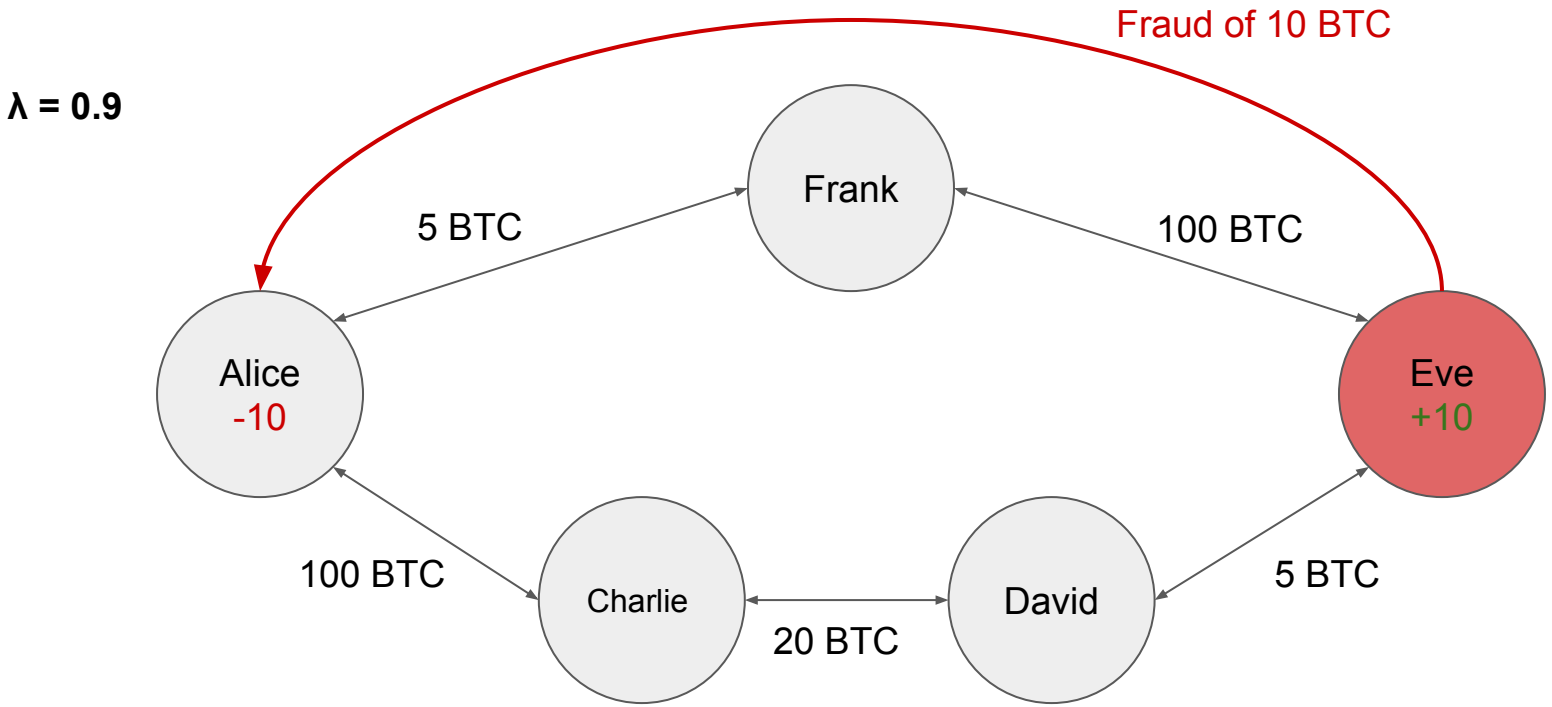
If the out-of-bound cost is smaller than the value of the trust relationship in the system, then the trust relationship is wrong.

Trust In Friends: Example

λ = 0.9



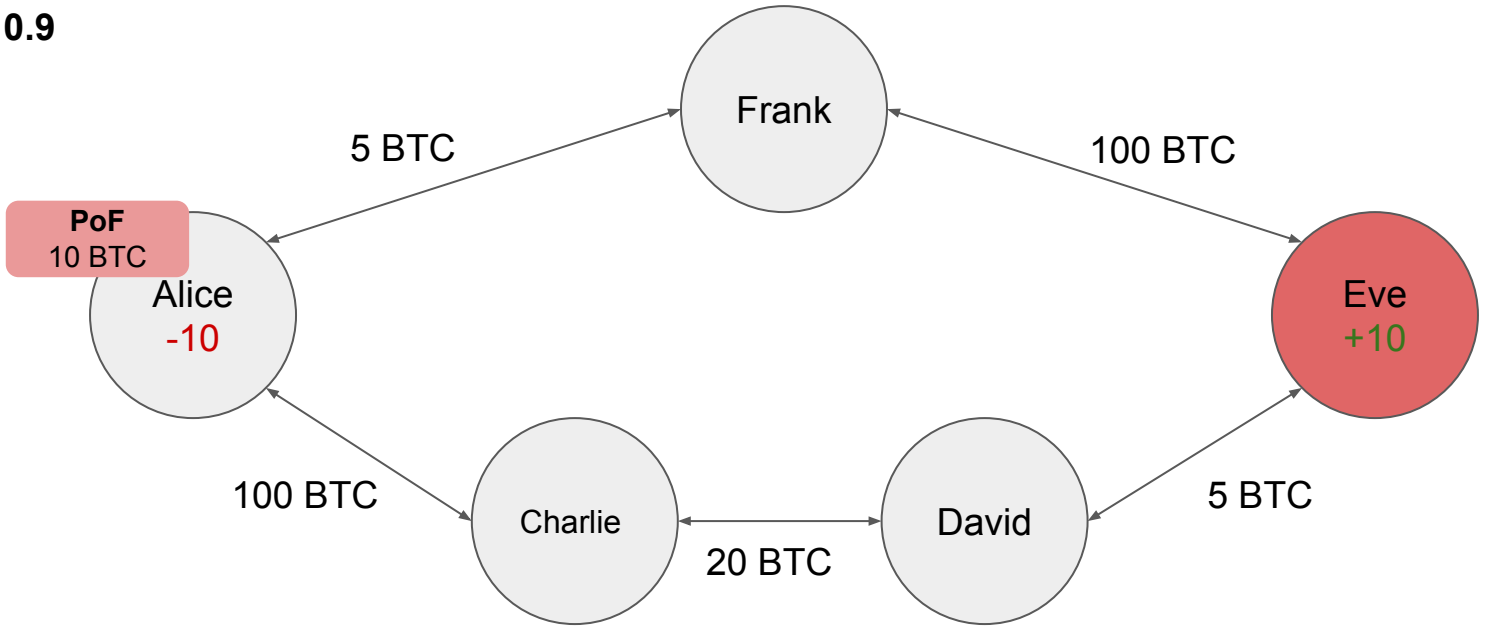
Trust In Friends: Example



Trust In Friends: Example

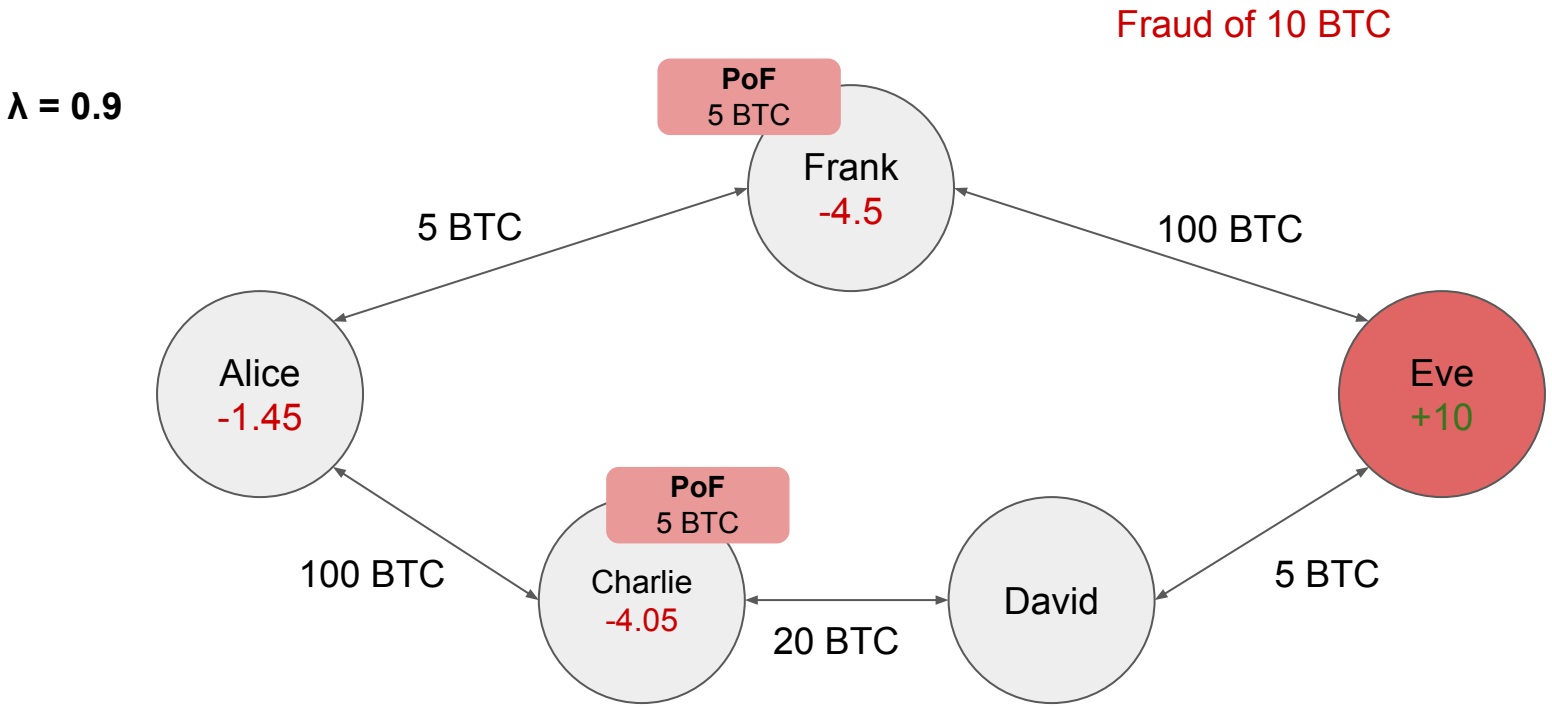
Fraud of 10 BTC

λ = 0.9

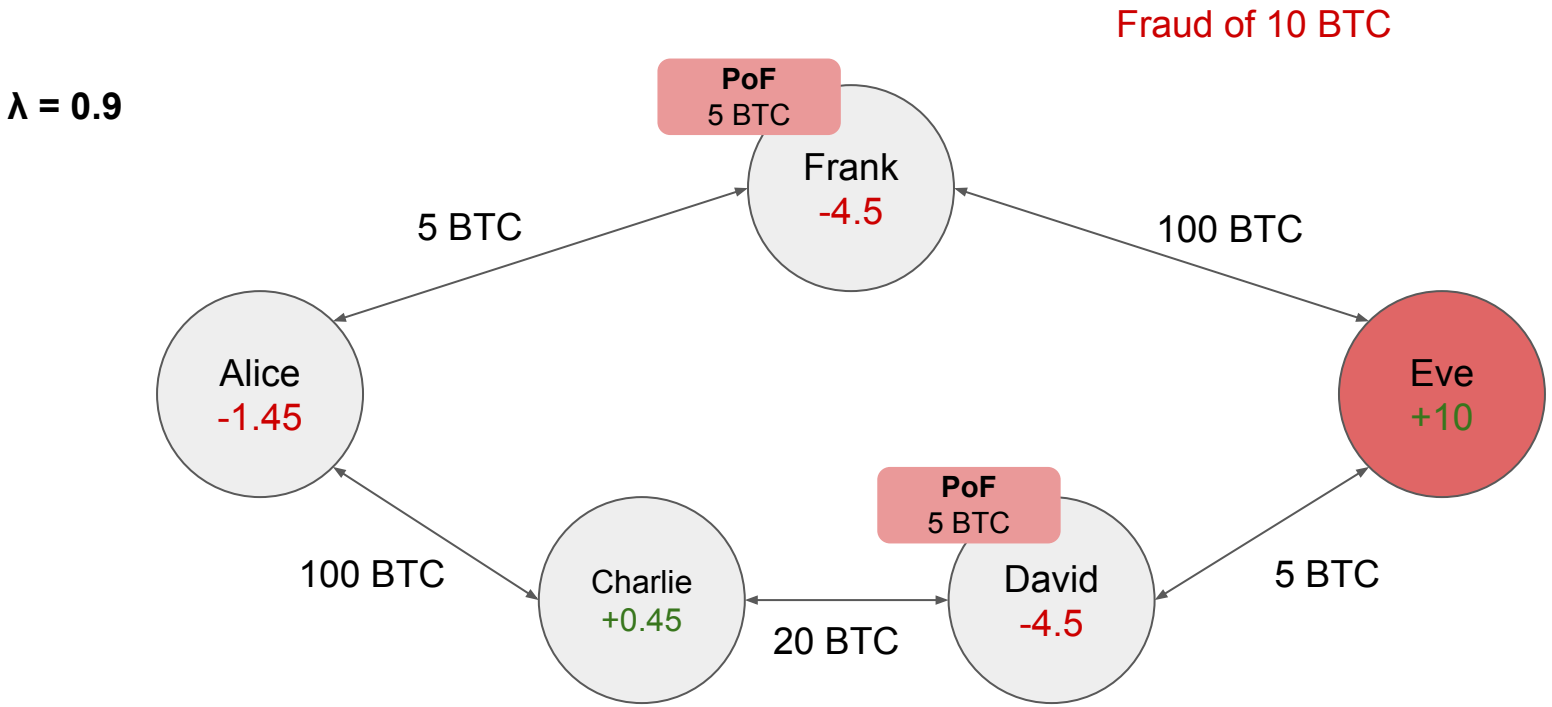


PoF 10 BTC = **PoF** 5 BTC + **PoF** 5 BTC

Trust In Friends: Example



Trust In Friends: Example



Trust In Friends: Intuition

- PoFs move closer to the fraudster in every transaction.
- The victim recovers part of her loss.
  - Higher value of λ => More of the loss will be recovered.
  - Better graph connectivity => More of the loss will be recovered.
- Intermediaries take the risk of selling PoFs forward:
  - They will be rewarded for their risk, depending on closeness to the fraudster and λ.
  - If they know that their trust relationships are correct, then they are undertaking no risk.
- **Fraud is irrational if all direct trust relationships are correct** (proof in the report).

Conclusion

Conclusion

- **TrustIsRisk.js**: The first implementation of a generic, decentralized, sybil-resilient trust inference library.
  - Developed with OpenBazaar in mind.
  - Sybil resilient
  - Performance problems: Need to implement it as an SPV node.
- **Trust In Friends**: A theoretical trust inference system where fraud is irrational.
  - Fraud is irrational if direct trusts are placed correctly.
  - Sybil resilient.
  - Participation incentive.
  - Indisputable fraud assumption: Sometimes not realistic.
  - No implementation.

Questions?