Pi Squared

Actors and Blockchains, Together

Xiaohong Chen
CTO, **Pi Squared**

Grigore Rosu
UIUC
Founder & CEO, **Pi Squared**, Founder **Runtime Verification**

2008

Bitcoin: A Peer-to-Peer Electronic Cash System

Satoshi Nakamoto
satoshin@gmx.com
www.bitcoin.org

Abstract. A purely peer-to-peer version of electron
payments to be sent directly from one party to anot
financial institution. Digital signatures provide part
benefits are lost if a trusted third party is still required
We propose a solution to the double-spending problem
The network timestamps transactions by hashing the
hash-based proof-of-work, forming a record that cannc
the proof-of-work. The longest chain not only serves
events witnessed, but proof that it came from the lar
long as a majority of CPU power is controlled by nod
attack the network, they'll generate the longest chain
network itself requires minimal structure. Messages
basis, and nodes can leave and rejoin the network
proof-of-work chain as proof of what happened while

1. Introduction

Commerce on the Internet has come to rely almost exclusiv
trusted third parties to process electronic payments. Whil
most transactions, it still suffers from the inherent we
Completely non-reversible transactions are not really possil
avoid mediating disputes. The cost of mediation incre
minimum practical transaction size and cutting off the pos
and there is a broader cost in the loss of ability to mal
reversible services. With the possibility of reversal, the ne
be wary of their customers, hassling them for more informa
A certain percentage of fraud is accepted as unavoidable.
can be avoided in person by using physical currency, but nc
over a communications channel without a trusted party.

What is needed is an electronic payment system based o
allowing any two willing parties to transact directly with ea
third party. Transactions that are computationally imprac
from fraud, and routine escrow mechanisms could easily b
this paper, we propose a solution to the double-spending pr
timestamp server to generate computational proof of the ch
system is secure as long as honest nodes collectively
cooperating group of attacker nodes.

2013

ETHEREUM: A SECURE DECENTRALISED GENERALI
SHANGHAI VERSION efc5f9a – 20

DR. GAVIN WOOD
FOUNDER, ETHEREUM & PARITY
GAVIN@PARITY.IO

Abstract. "The blockchain" paradigm when coupled with cryptographically-se
utility through a number of projects, with Bitcoin being one of the most notabl
a simple application on a decentralised, but singleton, compute resource. We
singleton machine with shared-state.
Ethereum implements this paradigm in a generalised manner. Furthermore
each with a distinct state and operating code but able to interact through a
We discuss its design, implementation issues, the opportunities it provides an

1. Introduction

With ubiquitous internet connections in most places
of the world, global information transmission has become
incredibly cheap. Technology-rooted movements like Bit-
coin have demonstrated through the power of the default,
consensus mechanisms, and voluntary respect of the so-
cial contract, that it is possible to use the internet to
make a decentralised value-transfer system that can be
shared across the world and virtually free to use. This
system can be said to be a very specialised version of a
cryptographically secure, transaction-based state machine.
Follow-up systems such as Namecoin adapted this origi-
nal "currency application" of the technology into other
applications, albeit rather simplistic ones.

Ethereum is a project which attempts to build the gen-
eralised technology; technology on which all transaction-
based state machine concepts may be built. Moreover it
aims to provide to the end-developer a tightly integrated
end-to-end system for building software on a hitherto un-
explored compute paradigm in the mainstream: a trustful
object messaging compute framework.

1.1. Driving Factors.

There are many goals of this
project; one key goal is to facilitate transactions between
consenting individuals who would otherwise have no means
to trust one another. This may be due to geographical
separation, interfacing difficulty, or perhaps the incompati-
bility, incompetence, unwillingness, expense, uncertainty,
inconvenience, or corruption of existing legal systems. By
specifying a state-change system through a rich and unam-
biguous language, and furthermore architecting a system
such that we can reasonably expect that an agreement will
be thus enforced autonomously, we can provide a means
to this end.

Dealings in this proposed system would have several
attributes not often found in the real world. The incorrupt-
ibility of judgement, often difficult to find, comes naturally
from a disinterested algorithmic interpreter. Transparency,
or being able to see exactly how a state or judgement came
about through the transaction log and rules or instructional
codes, never happens perfectly in human-based systems
since natural language is necessarily vague, information

is often lackir
shake.
Overall, w
can be guarar
viduals, syste
do so with ab
and how thos

1.2. Previou

kernel of this
evolved in ma
chain with a
unlimited inte
changed.
Dwork and
usage of a cry
ture ("proof-o
signal over the
as a spam de
of currency, b
a basic data
allowing a rec
having to rely
system in a si
The first e
strong econon
numurthy et
used to keep
"consumers"
"suppliers" for
by the proof-o
and a ledger
couldn't be co
spoof paymen
ery. Five year
such proof-of-
scope. The fr
widely adopte
Other proj
introduced nu
to the protoc
Primecoin, di
sought to take
tocol and repu

2020

Avalanche Platform
2020/06/30

Kevin Sekniqi, Daniel Laine, Stephen Buttolph

Abstract. This paper provides an architectural overview of the
codenamed `Avalanche Borealis`. For details on the economics
guide the reader to the accompanying token dynamics paper [2].
Disclosure: The information described in this paper is prelimina
Furthermore, this paper may contain "forward-looking statemen

Git Commit: 7497e4a4ba0a1ea2dc2a111bc6d

1 Introduction

This paper provides an architectural overview of the `Avalanche` p
differentiators of the platform: the engine, the architectural model

1.1 Avalanche Goals and Principles

`Avalanche` is a high-performance, scalable, customizable, and sec
broad use cases:

- Building application-specific blockchains, spanning permissio
  deployments.
- Building and launching highly scalable and decentralized appl
- Building arbitrarily complex digital assets with custom rules,

[1] Forward-looking statements generally relate to future events or our
limited to, `Avalanche`'s projected performance; the expected develo
of its vision and growth strategy; and completion of projects that
otherwise under consideration. Forward-looking statements represen
only as of the date of this presentation. These statements are not
reliance should not be placed on them. Such forward-looking statem
risks, which may cause actual performance and results in future peri
expressed or implied herein. `Avalanche` undertakes no obligation to
forward-looking statements are our best prediction at the time they a
will prove to be accurate, as actual results and future events could

2020

INPUT | OUTPUT

IOHK | WHY WE ARE BUILDING CARDANO | 06/28/2017

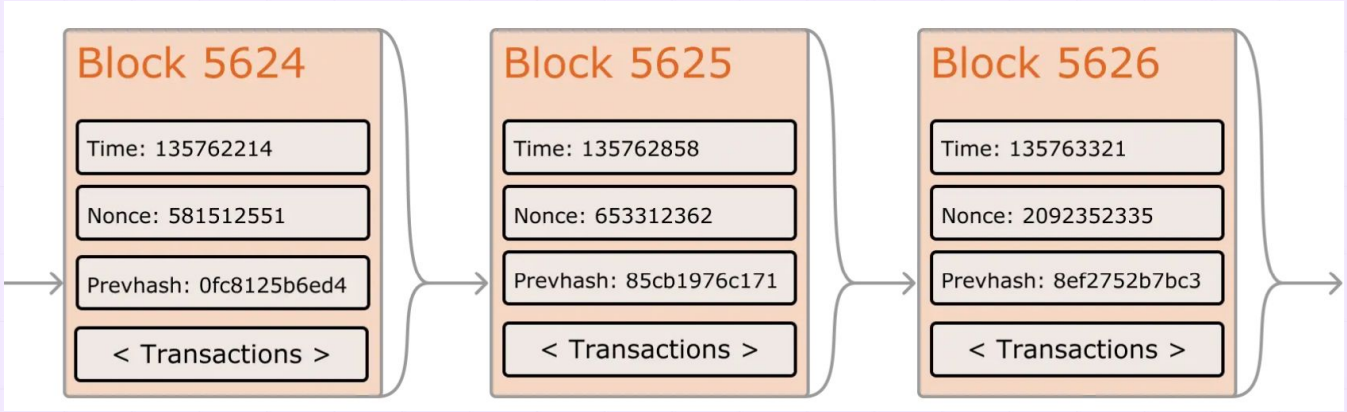
WHY WE ARE BUILDING CARDANO

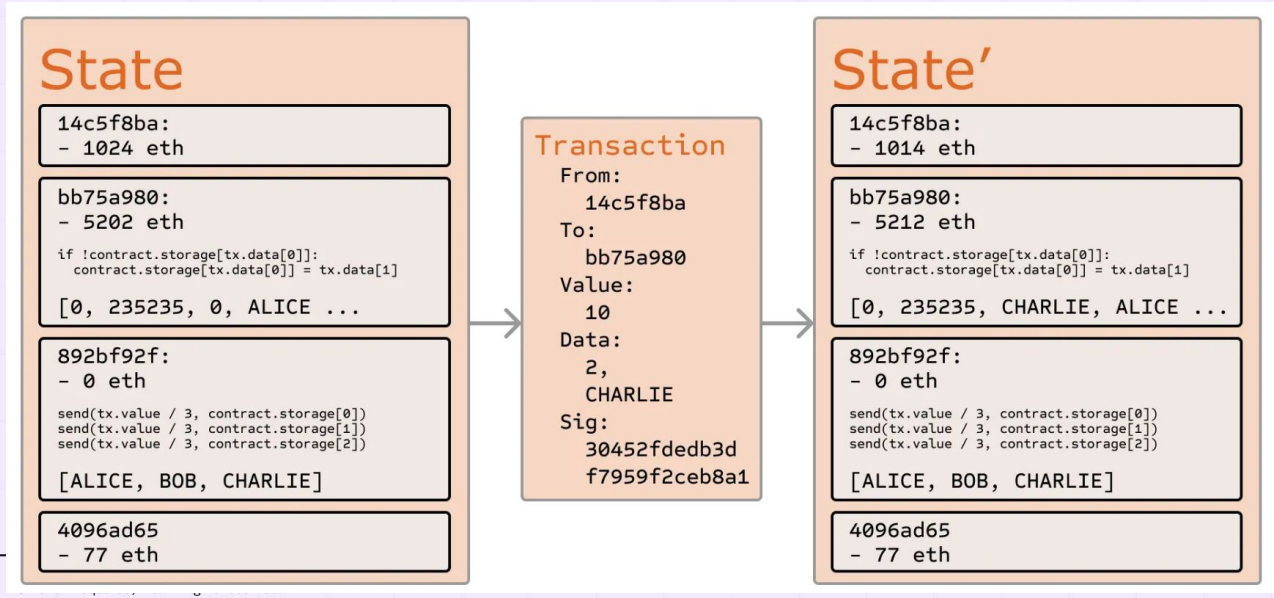
A Subjective Approach

CHARLES HOSKINSON
<Charles.Hoskinson@iohk.io>
<C3A6 5E46 7B54 77DF 3C4C 9790 4D22 B3CA 5B32 FF66>

Pi Squared

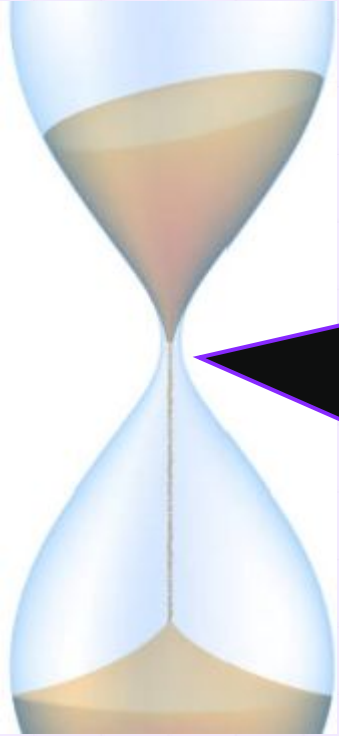
Blockchain





Pi Squared

Blockchains' Pain Point



Imposes total order on all transactions!

Blockchains' major pain point: do not scale, ie slow and expensive.

Bitcoin: A Peer-to-Peer Electronic Cash System

Satoshi Nakamoto
satoshin@gmx.com
www.bitcoin.org

Abstract. A purely peer-to-peer version of electronic cash would allow online payments to be sent directly from one party to another without going through a financial institution. Digital signatures provide part of the solution, but the main benefits are lost if a trusted third party is still required to prevent double-spending. We propose a solution to the double-spending problem using a peer-to-peer network. The network timestamps transactions by hashing them into an ongoing chain of hash-based proof-of-work, forming a record that cannot be changed without redoing the proof-of-work. The longest chain not only serves as proof of the sequence of events witnessed, but proof that it came from the largest pool of CPU power. As long as a majority of CPU power is controlled by nodes that are not cooperating to attack the network, they'll generate the longest chain and outpace attackers. The network itself requires minimal structure. Messages are broadcast on a best effort basis, and nodes can leave and rejoin the network at will, accepting the longest proof-of-work chain as proof of what happened while they were gone.

93% of transactions on Ethereum do not need a blockchain!

I need a blockchain !?
[for payments and settlement …]

I need a blockchain !?
[for assets store and transfer …]

I need a blockchain !?
[for permanent record settlement …]

No, you don't !

You need fast, cheap & verifiable payments and settlement.

You need a SET!

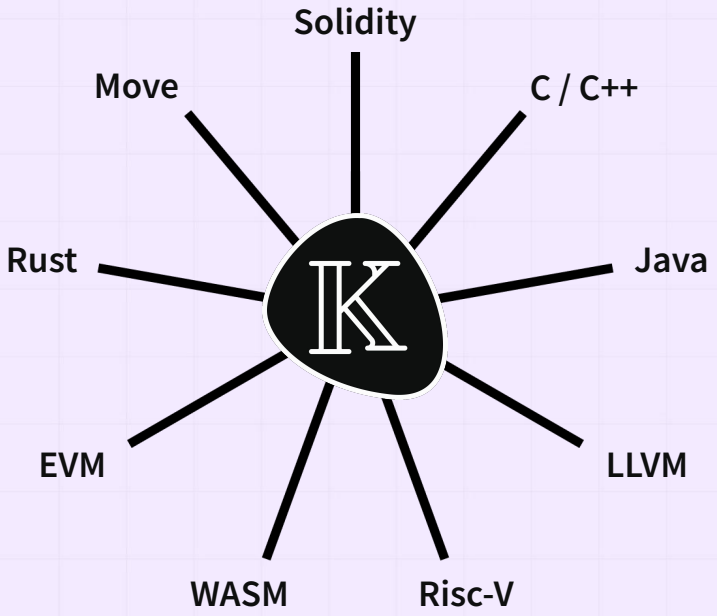
SET

New Web3 Infra Category for Fast, Verifiable Settlement (250k+ TPS, sub-100ms finality)

Ultimate verifiability for payments and
- Blockchains
- AI agents
- Centralized / mobile apps (e.g. trading)

Bring all languages & VMs to Web3 & AI
- Correctly (no assumptions, only proofs)
- Efficiently (verify once, use everywhere)

Fast. All Languages. All Proofs.



Core Pi Squared Technology

(see Appendix and Papers)

1. Pi² = Proof of Proof — universal verifiability

Zero Knowledge Proof

Mathematical Proof

All Languages
All Proofs

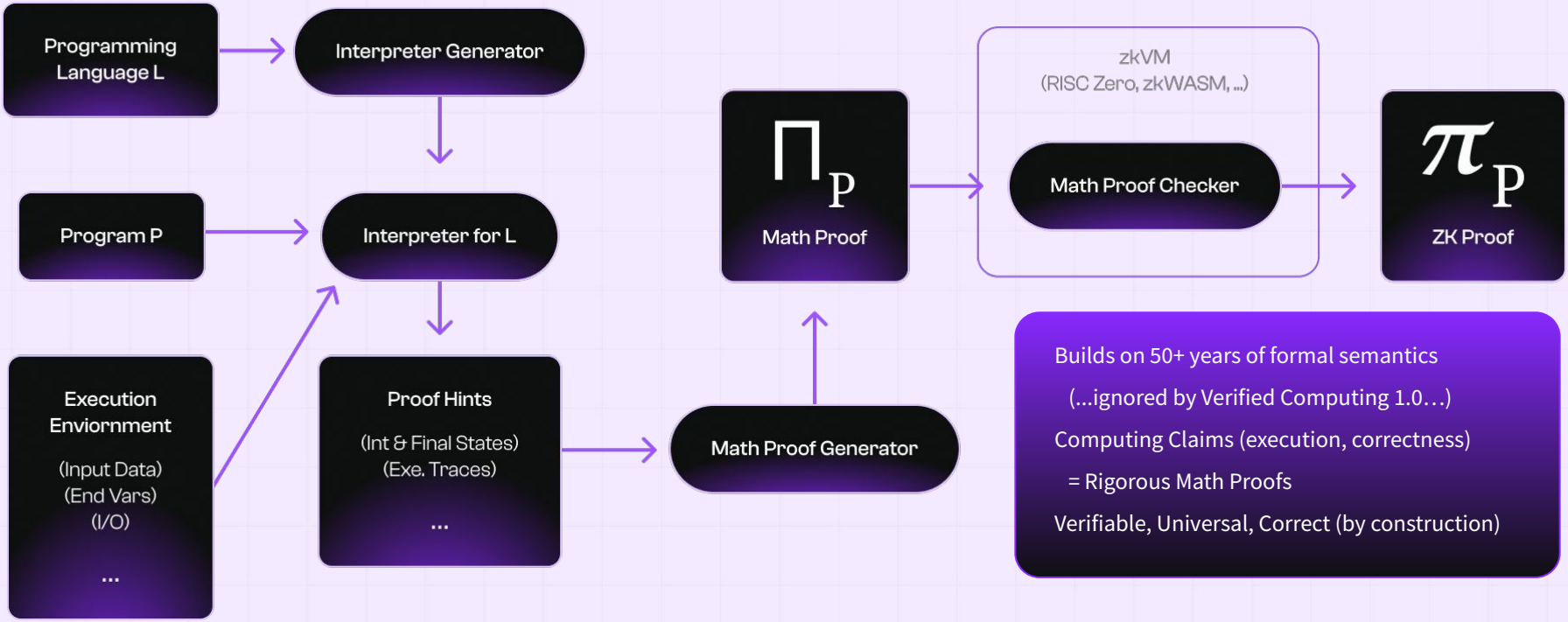
2. FastSet — weak consensus protocol

> 250k TPS
< 100ms finality

Pi Squared

Secret Sauce:

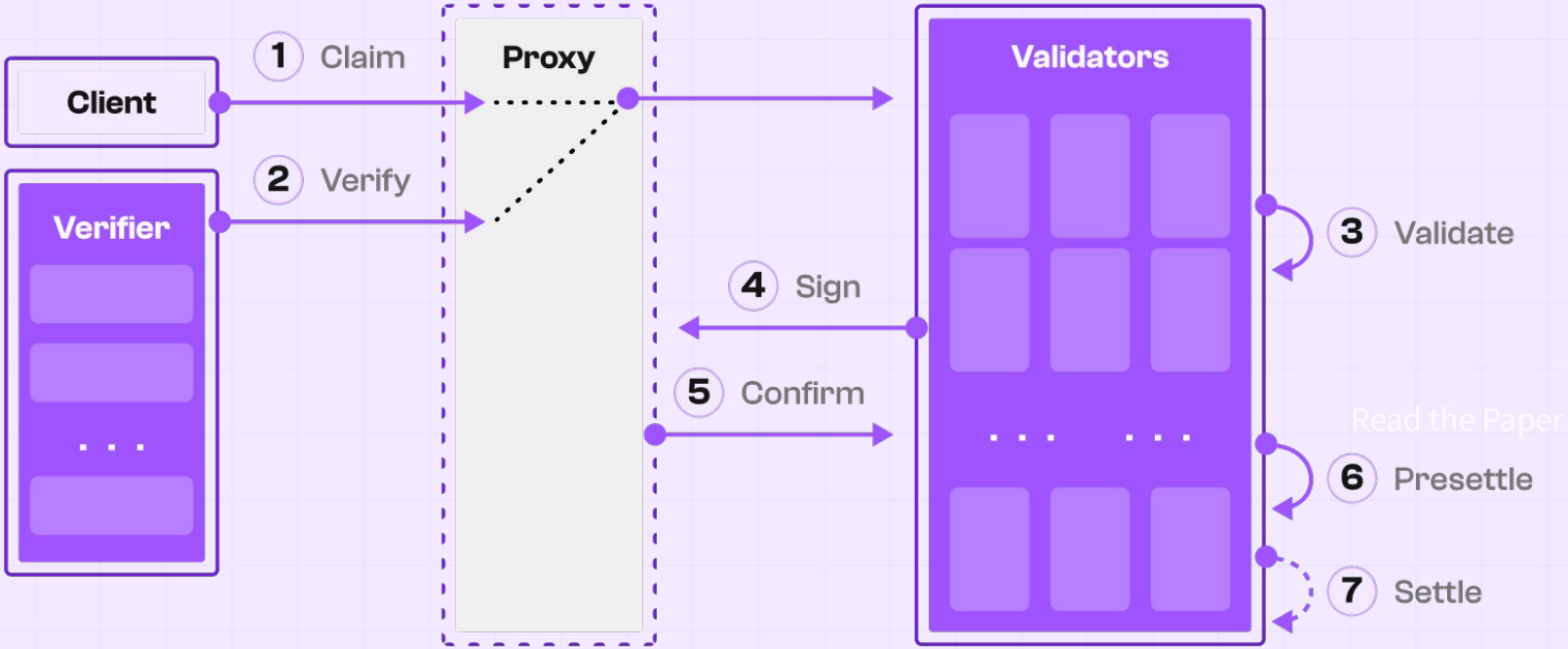
Verifiable Computing 2.0



Pi Squared

FastSet in a Nutshell

Validators = Actors replicating global state



Pi Squared

Example: Voting

```
BASIC_VOTE[MIN]:
    voted : Set;  // users who voted
    constructor {
        voted := {};
    }
    instance constructor {
        not(instance.owner in voted); // guard: must be true
        verify(contract.owner, 1);    // authority verifies voter
        voted.add(instance.owner);
    }
    passed() { size(voted) >= MIN; }
```

$$\texttt{MOTION} = \langle \texttt{contract(BASIC\_VOTE)} \rangle_{\texttt{gov}}$$

All a voter has to do in order to vote is create an instance of this contract:

$$\langle \texttt{instance(MOTION)} \rangle_{\texttt{alice}}$$

$$\langle \texttt{instance(MOTION)} \rangle_{\texttt{bob}}$$

$$\langle \texttt{instance(MOTION)} \rangle_{\texttt{charlie}}$$

...

Paper

Pi Squared

Example: Auctions

```
AUCTION[ITEM,BIDDING_TIME]:
    stopBiddingTime : Int;
    highestBidder : Address;
    highestBid : Int;

    constructor {
        ITEM.transfer_token(contract, 1);
        stopBiddingTime := time + BIDDING_TIME;
        highestBidder := contract.owner;
        highestBid := 0;
    }

    instance bid(amount) {
        time <= stopBiddingTime;
        if (amount > highestBid) {
            transfer(highestBidder, highestBid);
            transfer(contract, amount - highestBid);
            highestBidder := instance;
            highestBid := amount;
        }
    }
```

```
instance withdraw(amount) {
    transfer(instance.owner, amount);
}

end() {
    time > stopBiddingTime;
    ITEM.transfer_token(highestBidder.owner,1);
    transfer(contract.owner, highestBid);
}
```

$\texttt{AliceTicket} = \langle \texttt{contract(AUCTION[ticket\_item,1000])} \rangle_{\texttt{alice}}$

Read the Paper

$\texttt{auction3} = \langle \texttt{instance(AliceTicket)} \rangle_{\texttt{bob}}$

$\langle \texttt{transfer(auction3,100)} \rangle_{\texttt{bob}}$ — bob sends 100 to its instance

$\langle \texttt{auction3.bid(25)} \rangle_{\texttt{bob}}$ — bob sends 25 to AliceTicket (not alice)

$\texttt{tk\_alice} = \langle \texttt{instance(AliceTicket)} \rangle_{\texttt{charlie}}$

$\langle \texttt{transfer(tk\_alice,50)} \rangle_{\texttt{charlie}}$ — charlie sends 50 to its instance

$\langle \texttt{tk\_alice.bid(30)} \rangle_{\texttt{charlie}}$ — sends 25 to auction3 and 5 to AliceTicket

$\langle \texttt{auction3.bid(40)} \rangle_{\texttt{bob}}$ — sends 30 to tk_alice and 10 to AliceTicket