Workshop on Blockchain and its Applications - 2018
IIIT Allahabad, India

# Zero Knowledge Proofs and its applications in Blockchain

(August 19, 2018)

Shubham Sahai Srivastava

C3i Center, IIT Kanpur.
Department of Computer Science and Engineering, IIT Kanpur.

# Blockchain and Privacy?

- Blockchain: immutability, censorship-resistance, and open and permissionless
- Bitcoin (Blockchain 1.0): pseudo-anonymity, public ledger, patterns, de-anonymization … transactions
- Ethereum (Blockchain 2.0): smart contracts, DApp … data
- Fabric (Blockchain 3.0): Privacy and channels, … trust?
- Do we need to continue transacting in the open to enjoy the benefits?
- Mixers, ring signatures, partially homomorphic…

# Zero Knowledge Proofs

The general idea of proof systems and zero knowledge

- First proposed in 1985 by MIT researchers: Shafi Goldwasser, Silvio Micali, and Charles Rackoff
- Decades of research, but generating excitement lately!
- Potential to redefine the concept of online privacy.

# Let’s Begin …

Bob is color blind!

# Let’s Begin …


# Let’s Begin …


# Let's Begin …

If Alice is cheating, then probability of fooling Bob :

$$\left(\frac{1}{2}\right)^n$$

For n = 15 :

0.0000305

# In the end …

Alice Convinces Bob, that with very high probability :
“She has **different** coloured  hats.”

However, Bob has no idea :
“What are the **colors** of the hats?”

# Zero Knowledge Proofs

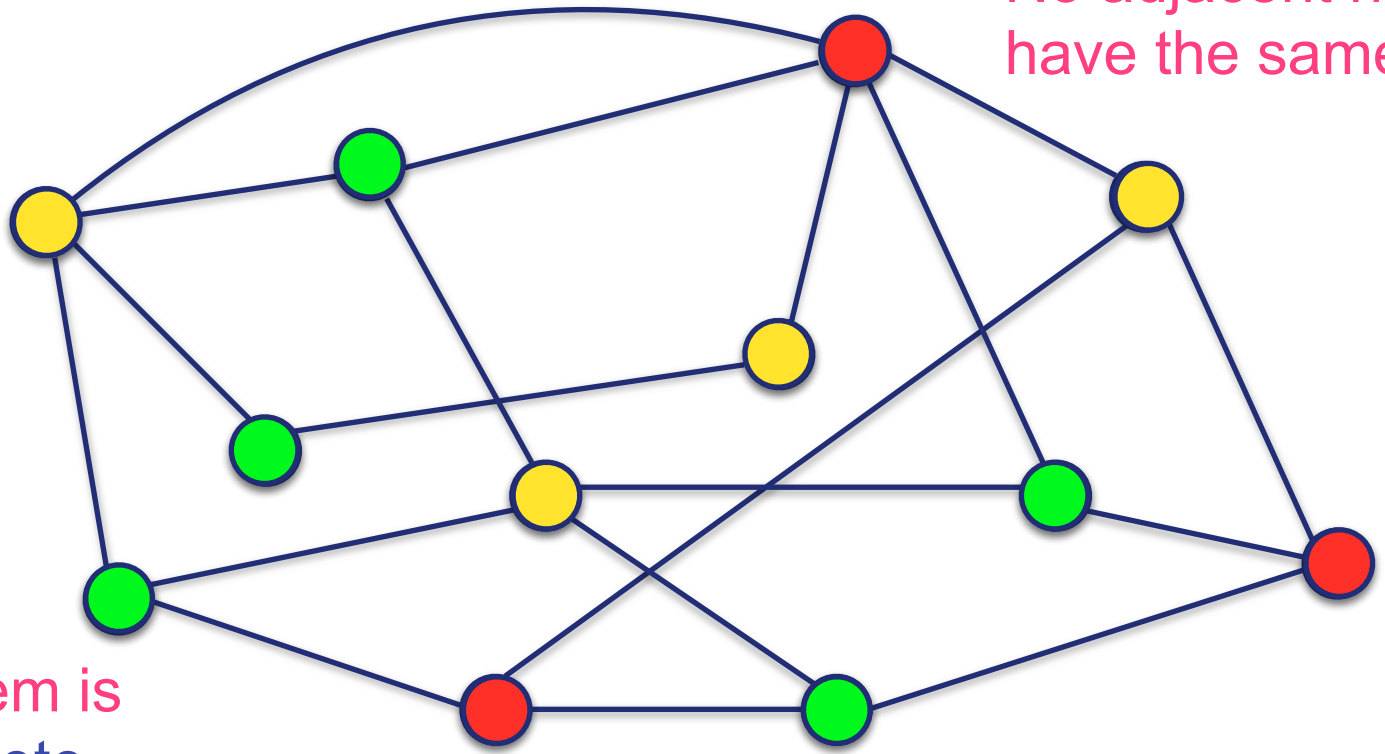
- A method by which one party: **“Prover”**, can convince another party: **“Verifier”**; about the **“truthfulness”** of a statement.
- The protocol conveys nothing apart from the veracity of the statement.

**(Zero Knowledge)**

- A zero knowledge proof must satisfy :
    - **Completeness** : Honest Prover can convince an honest Verifier.
    - **Soundness** : Cheating Prover can convince an honest Verifier with negligible probability.
    - **Zero Knowledge** : Verifier learns nothing apart from the veracity of the statement.

# Graph 3-Coloring (G3C)

No adjacent nodes have the same color.



The problem is NP-Complete

# G3C : Problem …

Given a Graph G with **n** vertices and **m** edges

G is 3-colorable!

I know the colouring

Really?
**Prove** it?

# G3C : Protocol …

**Alice:** Choose random assignment of 3 colors and color the graph, locks them in a box and sends to Bob

**Bob:** Chooses an edge (e) randomly from the graph

Request keys for the 2 boxes corresponding to e

**Bob:** If the colors in the boxes are different, colouring of the edge is correct.

# G3C : Protocol …

**Alice:** Choose random assignment of 3 colors and color the graph

**Bob:** Chooses an edge (e) randomly from the graph

If these two colors are different:
"The edge chosen was correctly coloured"

Repeat the same experiment again!

# G3C: Repeat! How many times ?

If Alice is cheating, then probability of fooling Bob :

$$\left(\frac{m-1}{m}\right)^{m^2}$$

For m = 7 :

0.000524

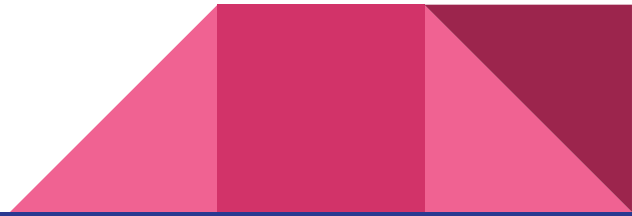
# G3C: Is it a Zero Knowledge Proof?

It can be shown that the protocol satisfies:

- **Completeness:** Honest Prover can convince an honest Verifier.
- **Soundness:** Cheating Prover can convince an honest Verifier with negligible probability.
- **Zero Knowledge:** Verifier learns nothing apart from the veracity of the statement.

# Application of ZKP in Blockchains?

- The proofs seen so far are **interactive**
- Not suitable for blockchain applications. **Why?**
- Desirable properties from a ZKP variant to be usable in blockchain:
    - Non-interactiveness
    - Small in size
    - Fast verification



# zk-SNARK

“it’s really mind boggling”
- Sergey Brin
(Blockchain Summit-2018)

“

**Z**ero **K**nowledge

- **S**uccinct
- **N**on-interactive
- **AR**guments of
- **K**nowledge

# Non Interactive Zero Knowledge Proofs

- No interaction required between the Prover and the Verifier
- Impossible in standard cryptographic model
  [Goldreich and Oren; 1993]
- Possible in **common reference string** and **random oracle model.**
- Common reference string can yield computational zero knowledge.

# zk-SNARKs

- No interaction with Prover required
- A proof once generated can be verified by anyone
- To establish this, we need to establish some shared "reference string"
- This is done during Setup Phase…generates toxic waste
- For every function (circuit) we need to run the Setup Phase

# What can we do with it?

I know a ‘x’, such that :

$x^3 + x + 5 == 35$

Really?
Why don’t you
**prove** it?

# Lets work out this example…

$$x^3 + x + 5 == 35$$

Function:

```
def qeval(x):
    y = x**3
    return x + y + 5
```

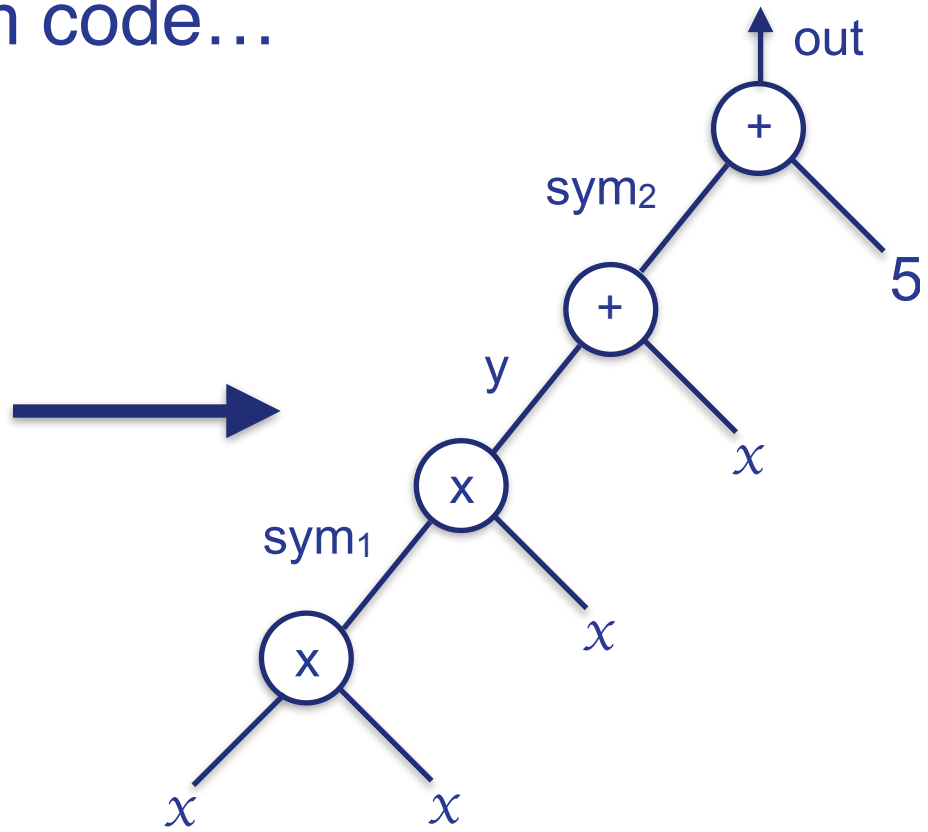
Flattened Code:

$$\begin{aligned} sym_1 &= x * x \\ y &= sym_1 * x \\ sym_2 &= y + x \\ \sim out &= sym_2 + 5 \end{aligned}$$

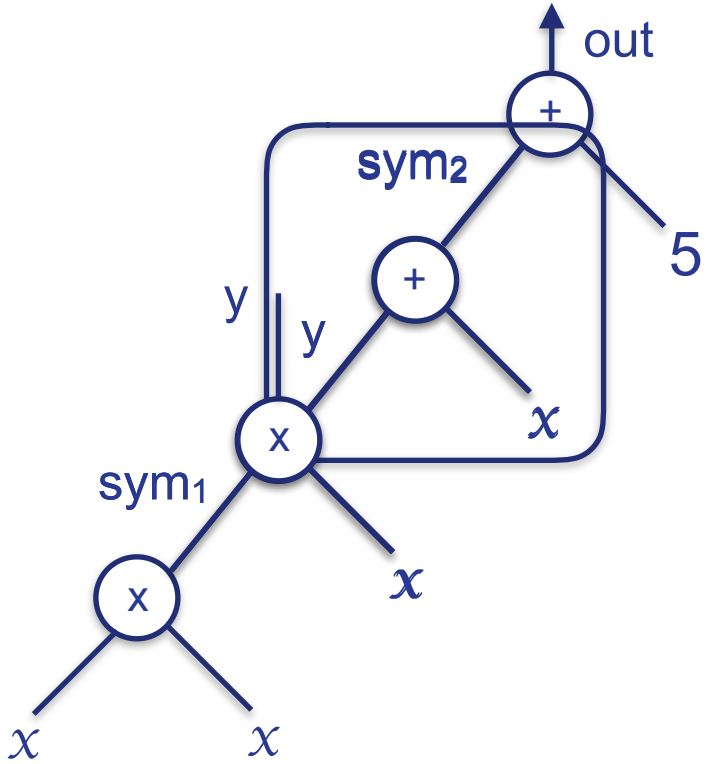
# Arithmetic Circuit from code…

Flattened Code:

$sym_1 = x * x$
$y = sym_1 * x$
$sym_2 = y + x$
$\sim out = sym_2 + 5$



# Constraints on variables …




Every gate in the circuit, imposes a constraint on its **inputs** and **outputs**.

x + sym1 = sym2

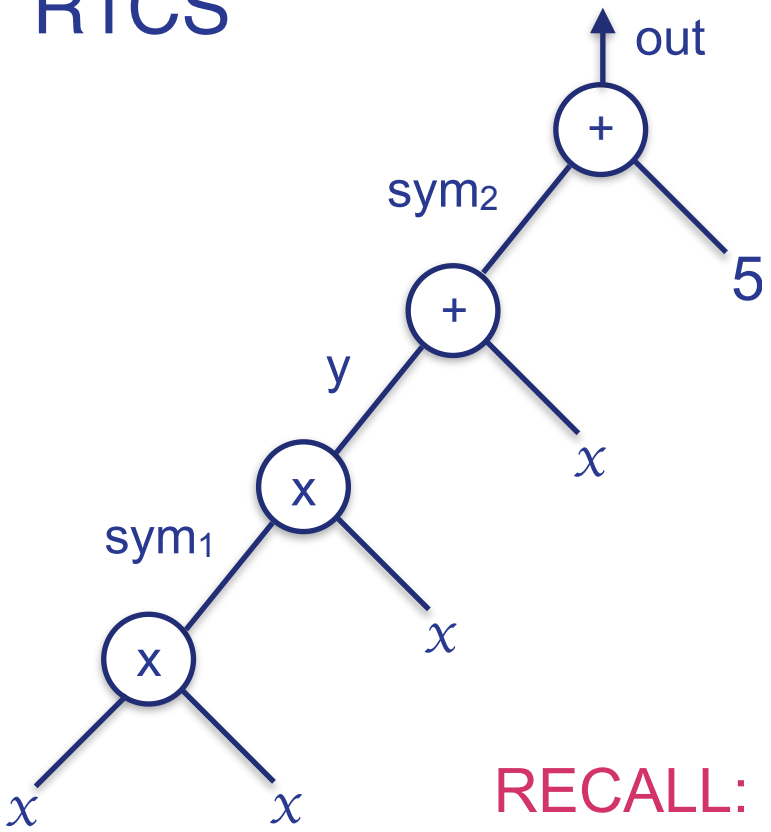
# Rank 1 Constraint System (R1CS)

- Sequence of 3 vectors (a,b,c) and a solution vector s, which satisfy:

$$s.a * s.b - s.c = 0$$

- Represent constraint corresponding to each gate as a R1CS
- The vectors will be:
    - **a** : Left input to the gate
    - **b** : Right input to the gate
    - **c** : Output of the gate
    - **s** : Solution vector

# R1CS



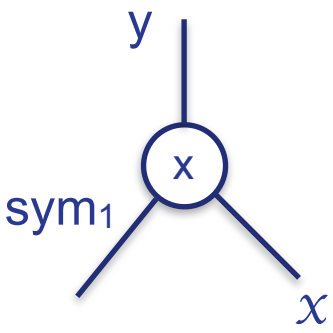

Fix a variable ordering amongst all the variables of the circuit

[ ONE, $x$, OUT, $sym_1$, y, $sym_2$ ]

RECALL: Every gate corresponds to a constraint

# R1CS …

[ ONE, $x$, OUT, $sym_1$, y, $sym_2$ ]




$$a = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \quad b = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad c = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \quad s = \begin{bmatrix} 0 \\ 3 \\ 0 \\ 5 \\ 15 \\ 0 \end{bmatrix}$$

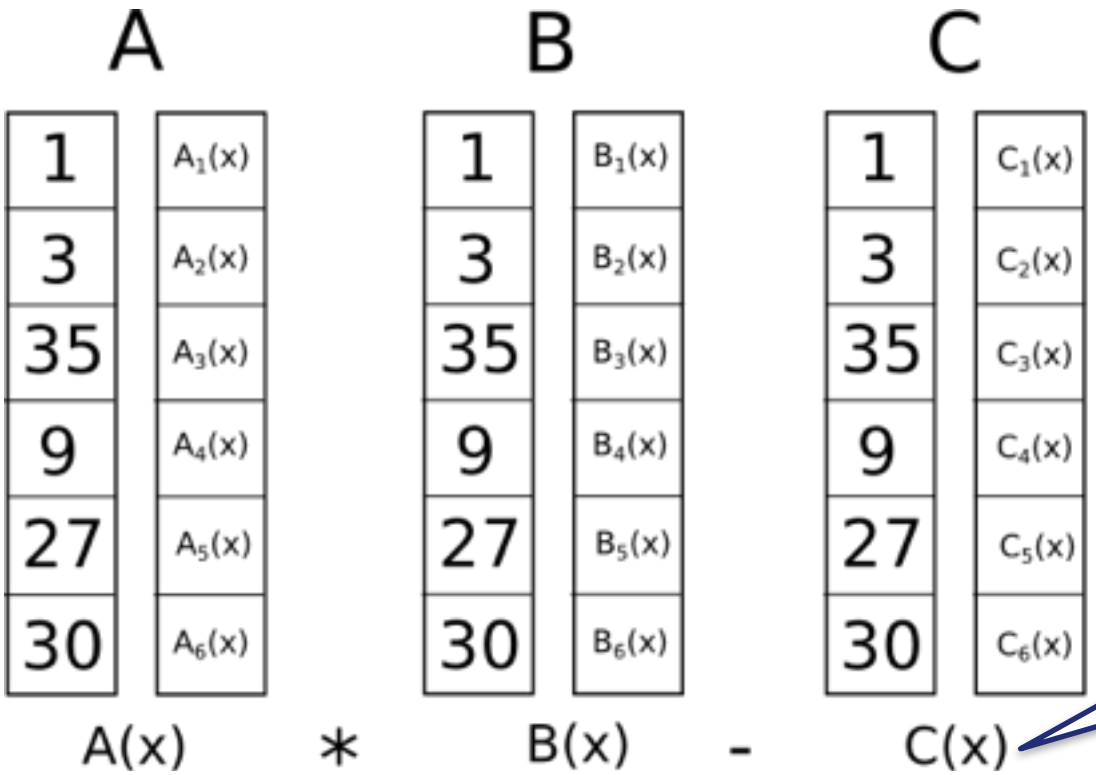
Represent all the gates as R1CS

Easy to see s.a * s.b - s.c = 0 holds

# R1CS

- Goal is to come up with s, that satisfies all the R1CS simultaneously.
- To verify: Solve each R1CS equation corresponding to a solution vector.
- Can we do better?

# Quadratic Arithmetic Programs (QAPs)




$$A(x) = \Sigma\, s_i\, A_i(x)$$

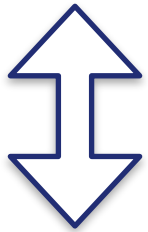
$$B(x) = \Sigma\, s_i\, B_i(x)$$

$$C(x) = \Sigma\, s_i\, C_i(x)$$

Should be 0 for all x corresponding to circuit gates.

# QAP …

$$A(x) * B(x) - C(x) = 0, \quad \forall x \in [1, |Gates|]$$



A(x) * B(x) - C(x) is divisible by $\prod_{i=1}^{|Gates|}(x-i)$

Target Polynomial (T(x))

# QAP …

$$A(x) * B(x) - C(x) = H(x) * T(x)$$

$$A(x) = \Sigma\, s_i\, A_i(x)$$

$$B(x) = \Sigma\, s_i\, B_i(x)$$

$$C(x) = \Sigma\, s_i\, C_i(x)$$

If we know the solution vector (**s**), then:

- We know: $s_i$, $\forall i \in [1, |Vars|]$
- For a given **e**, can compute : A(e), B(e), C(e)
- T(x) is public, so we can compute H(e)

# Proof

- The evaluation point e is chosen randomly by the *Verifier* and sent to the *Prover*.
- A(e), B(e), C(e), H(e) is the desired proof.
- The point e is send in an “hidden” form.
- Polynomials have to be evaluated “blindly”.
- Elliptic Curve pairings are used as a underlying “hiding scheme”.

# Applications in Blockchain

- Zcash
- Ethereum
- Hyperledger Fabric

# Zcash

- Privacy preserving crypto currency based on zk-SNARKs
- A transaction contains: sender(input txns), amount, reciever(output txn)
- Publish proof that a private transaction follows the rules of the Zcash network, concealing the sender, recipient, and amount (shielded transactions)
- Downside? Trusted Setup (Zcash Trusted Setup Ceremony)

# Zcash: Destroying toxic waste…

The burnt remains of one machine involved in Zcash's trusted setup

Pic Credit : Peter Todd

# Ethereum

- Data Privacy?
- All details about a smart contract are public on the Ethereum blockchain
- fund senders and recipients, all transaction data, all code executed …
- Privacy by secure commitments

# Ethereum (Metropolis)

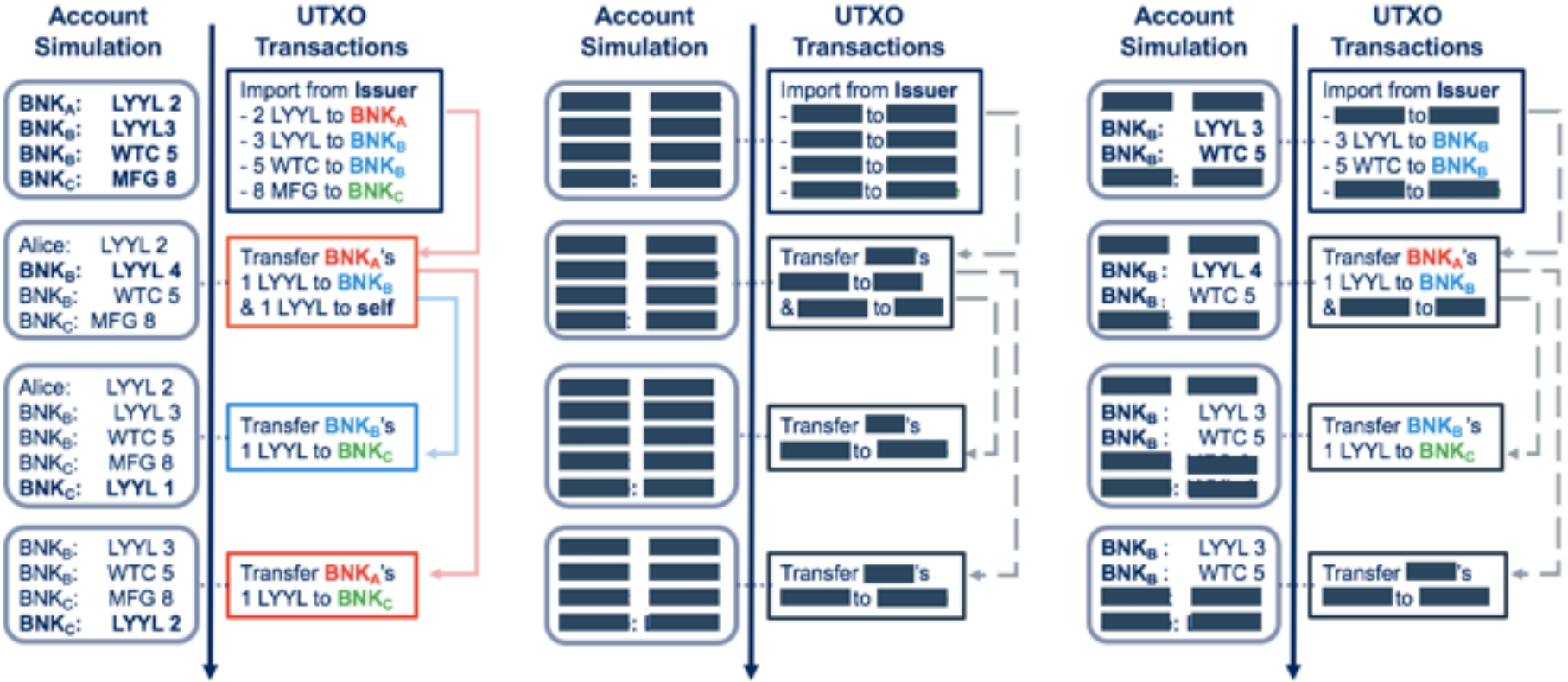
- Ability to verify zk-SNARKs efficiently on-chain.
- Users can store secret off-chain
- Each of these use cases require their own trusted setup
- Contract data has 1-to-1 correspondence with the user
- Can’t achieve autonomous privacy

# Hyperledger Fabric (ZKAT)

- Integrate ZKP into a wider range of applications targeting asset management
- Use of ZKP for privacy-preserving asset management with audit support
- ZKAT is built on top of anonymous authentication mechanisms offered by Identity Mixer.
- Asset Management: issue asset, request transfer of their assets in zero knowledge
- Support Audibility: By assigning auditors to organizations, giving them unrestricted access.

# ZKAT

Pic Credit : IBM Developer Works



# Thank You!