Managing data for Fintech: Consensus, Privacy and Regulations

Wei Xu

Institute for Interdisciplinary Information Sciences
Tsinghua University

About Me - Wei Xu 徐葳

- Tsinghua & Penn (B.S) 1999-2003
- Berkeley (M.S. and Ph.D.) 2003 - 2010
  - Advisors: David Patterson and Armando Fox
- Google 2010 - 2013
- Joined Tsinghua in 2013
  - Assist. Prof. and Assist. Dean @ Institute for Interdisciplinary Information Sciences (IIIS)
- Research Area
  - Distributed systems + Machine learning
  - Interdisciplinary "Big data" Applications, esp. Fintech

"Fintech" is about different things in China vs. in the US

- In the US: more efficiency

- Investors --------(many many brokers) ------- > users of the fund
- Payment
  - A business school case study 10 years ago vs. now
- Tax filing
- Compliance
- ……

"Fintech" is about different things in China vs. in the US

- In China: All about new business models
- "Internet finance"
  - Internet insurance
  - 10 cents per insurance policy (delivery fees)
- Credit scores from alternative data
  - Phone records, SMS messages, location tracking
- Mobile payment

- Innovating the traditional economy
  - e-commerce + payment + loans
  - Supply chain financing

Differences in Fintech development

US: navigating through
the mature market

China: flying into the unknown, fast

Significant challenges in the infrastructure of Fintech in China

Problems:

- Trust
- Risk management
- Privacy
- Regulations
- Data monopoly

Consequences:

"Policy Risk"

aka. Blaming the government

P2P lending is popular in China

- 8643 Internet loan companies as of June 2017
- Loan balance: 960.8bn RMB (155bn USD)

- No central credit bureau like Experian
  - The government one only serves banks, not loan companies

- Many regulations came out since last year
  - Preventing "a systematic financial risk"

Outline

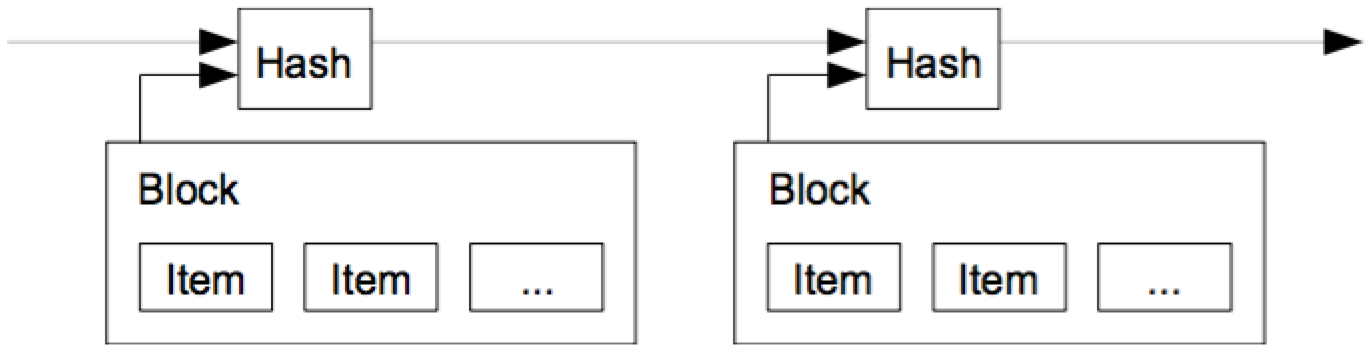
- A fast consensus protocol for consortium block chains

- Privacy preserving data mining framework

- Privacy + regulations, how to balance them?

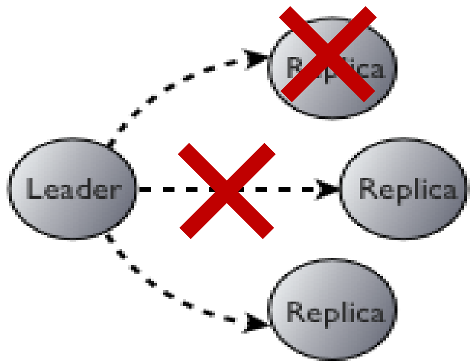
Block chain

- A fully decentralized database
- Maintains a continuously growing log
- “Distributed ledger”



The key challenge of block chain (and all distributed storage systems)

- Replication and Consensus



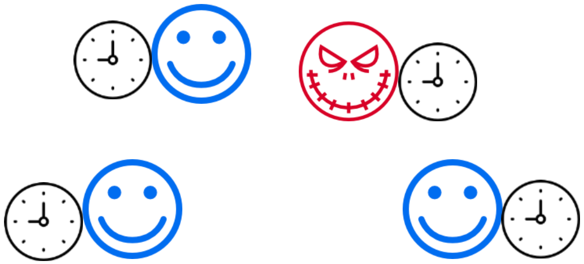
Correctness

Liveness

Unfortunately: Impossible - Fischer-Lynch-Paterson (FLP) impossibility results

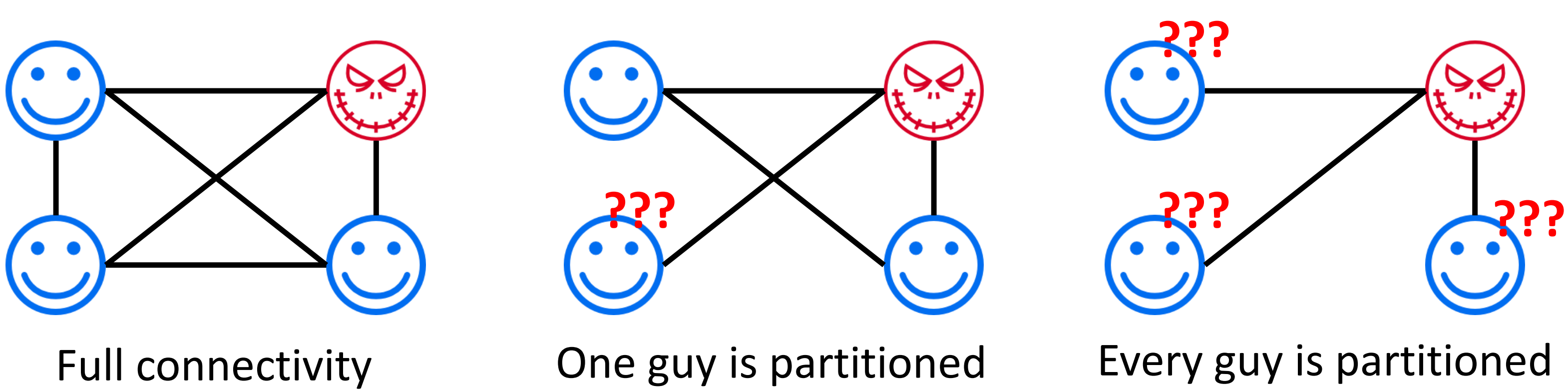
Application Scenarios

- Consortium Blockchain
  - A fixed group of *N* players
  - Trusted PKI (Public Key Infrastructure)
  - Keeps Safety and Liveness
- Partially-synchronized global clock
  - Drift <= seconds
- Asynchronous network
  - Messages could be delayed and/or dropped
  - Eventual connectivity assumption for liveness

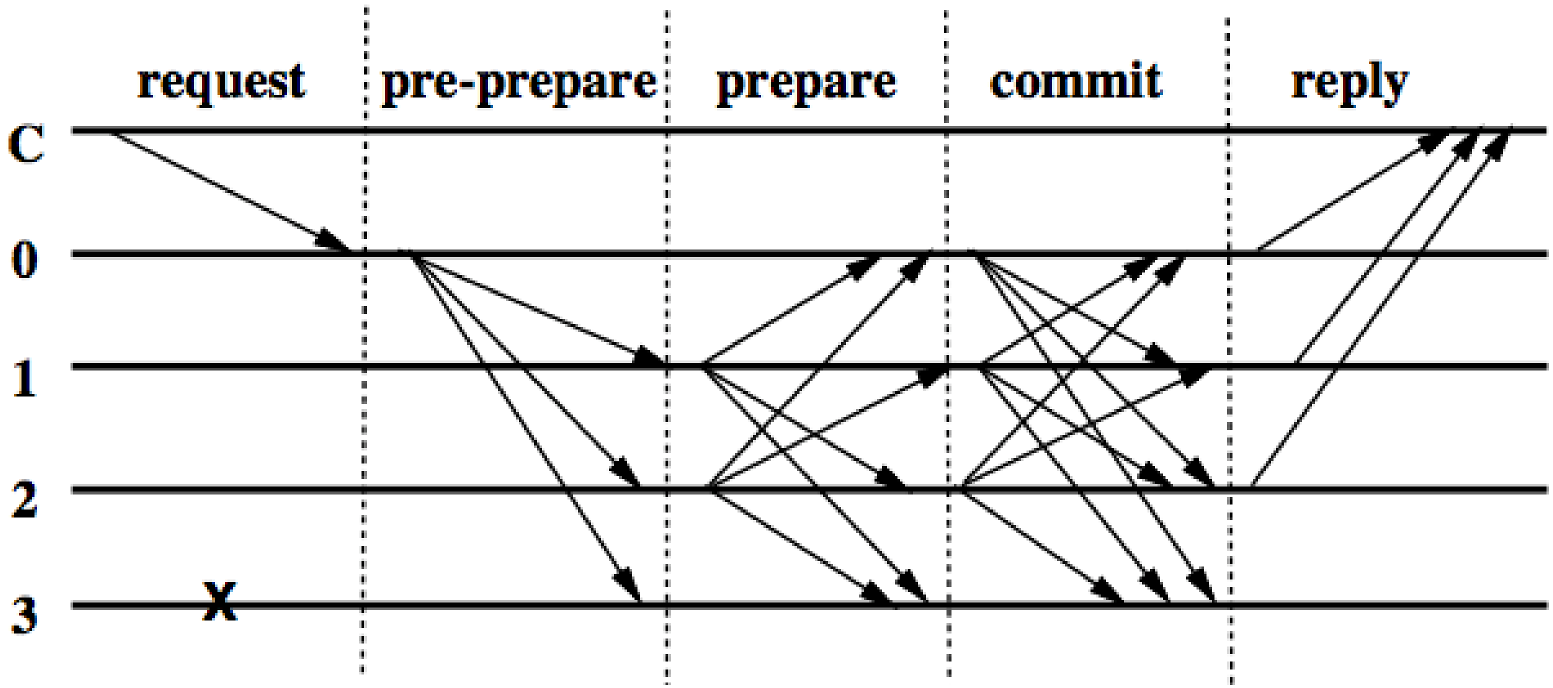


Adversarial Model

- Out of the *N* players
- At most $f < N/3$ are *malicious* (Byzantine failures), others are *honest players*
- Adversaries control the network
  - can partition players adaptively and immediately



Standard Solution: PBFT



Assumptions and performance of different protocols

	OUR WORK	PBFT	Algorand	Honeybadger
Network Assumptions	Asynchronous network	Asynchronous network	Weakly synchronous network	Asynchronous network
Adversarial Model	Adaptive attack	Static attack	Adaptive attack	Adaptive attack
Scalability	140 nodes 5000 tps 45s latency	64 nodes 1700 tps 1.8s latency	50k nodes 360 tps 22s latency	104 nodes 2000 tps 300s latency

Major Challenges and Our Solutions

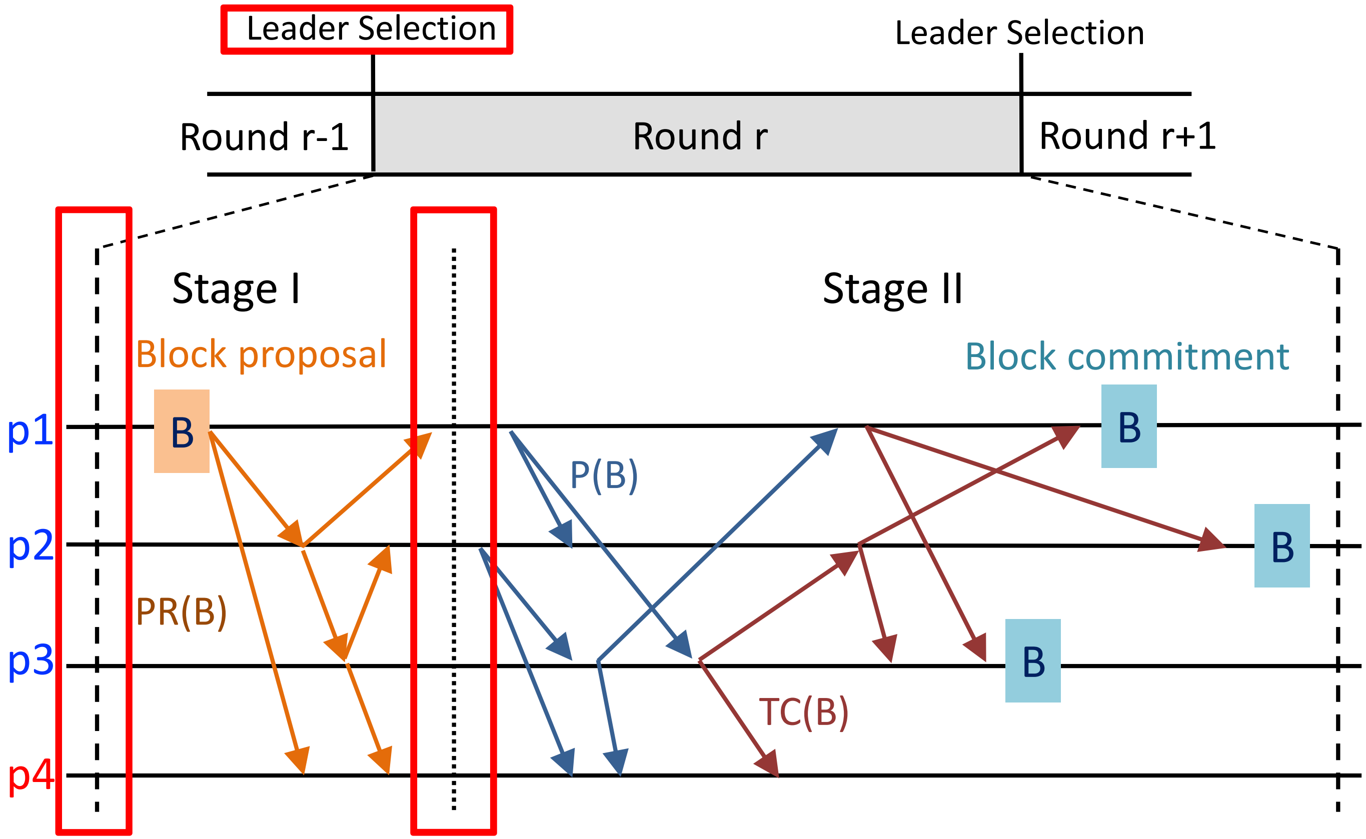
- **Problem 1:** The leader is vulnerable under adaptive attacks (DDoS)
- **Solution 1:** Secret leader selection and one message per view (protocol layer)

- **Problem 2:** Poor scalability
- **Solution 2:** Multi-signature and gossip (implementation layer)

Protocol Design Choices

- 2-phase protocol like PBFT
  - To achieve fast commitment for normal scenarios (happy path)
- Clock-based synchronous protocol
  - To deal with rounds with 0 or >1 potential leaders
- Use multi-signatures to reduce message sizes

Clock-based Byzantine Agreement Protocol



Secret Leader Selection

Cryptographic sortation mechanism [Micali. 2017]

- Selected as a potential leader if

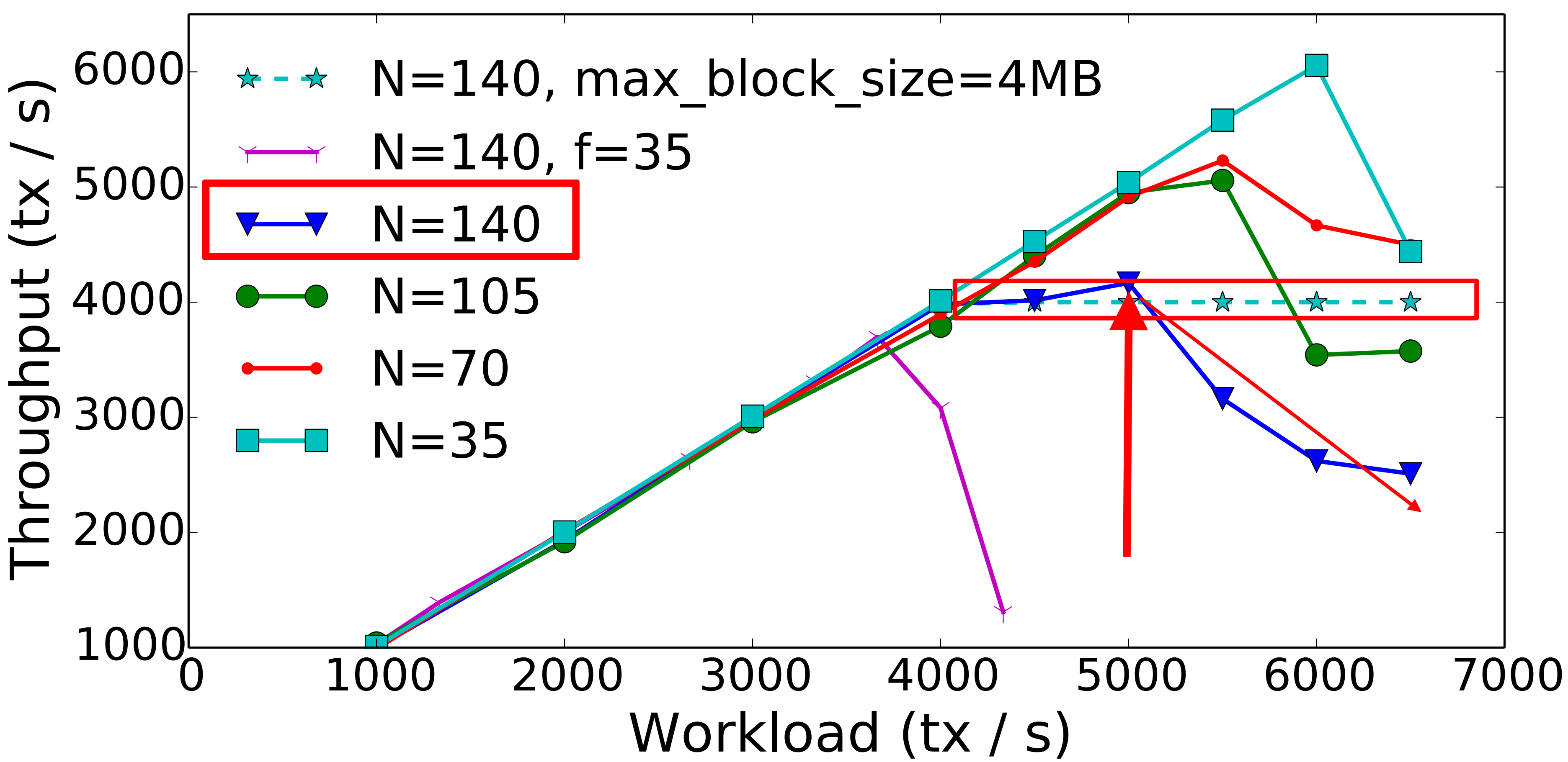
$$H(sign_{sk}(round)) < D$$

- Each player secretly knows whether she is a potential leader
  - Prevent DDOS attacks targeting the leader
- Easy proof of leader role: just a signature

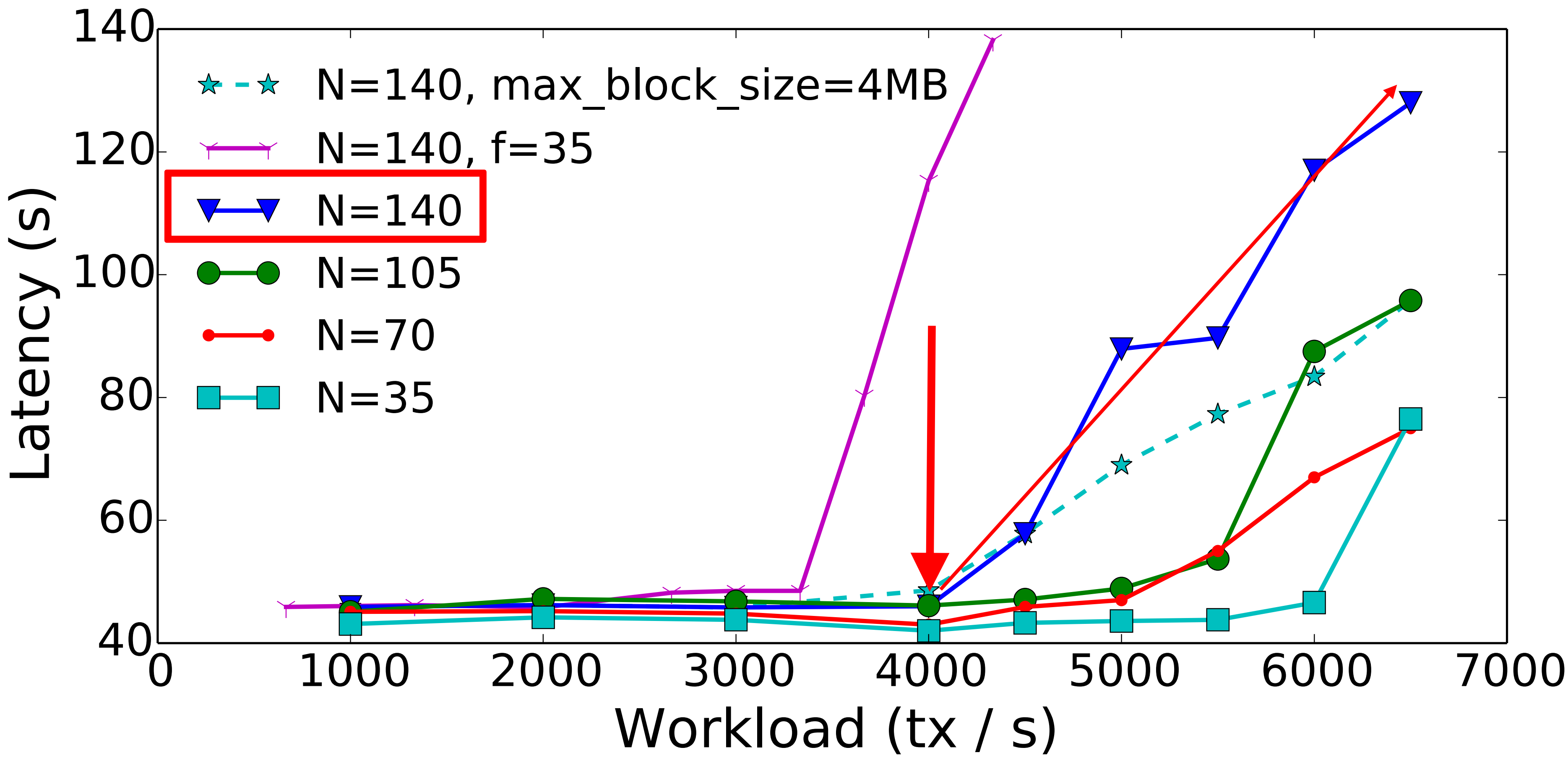
Evaluation Setup

- Java implementation with
  - grpc-java for communication and JPBC for cryptography.
- Launch 140 instances from 14 regions on AWS.
- Each region:
  - 1x t2.2xlarge
  - 4x t2.xlarge
  - 5x t2.medium
- 30 seconds round time with 5 seconds for stage II
- 250-byte transaction size (similar to bitcoin)

Evaluation result: high Throughput



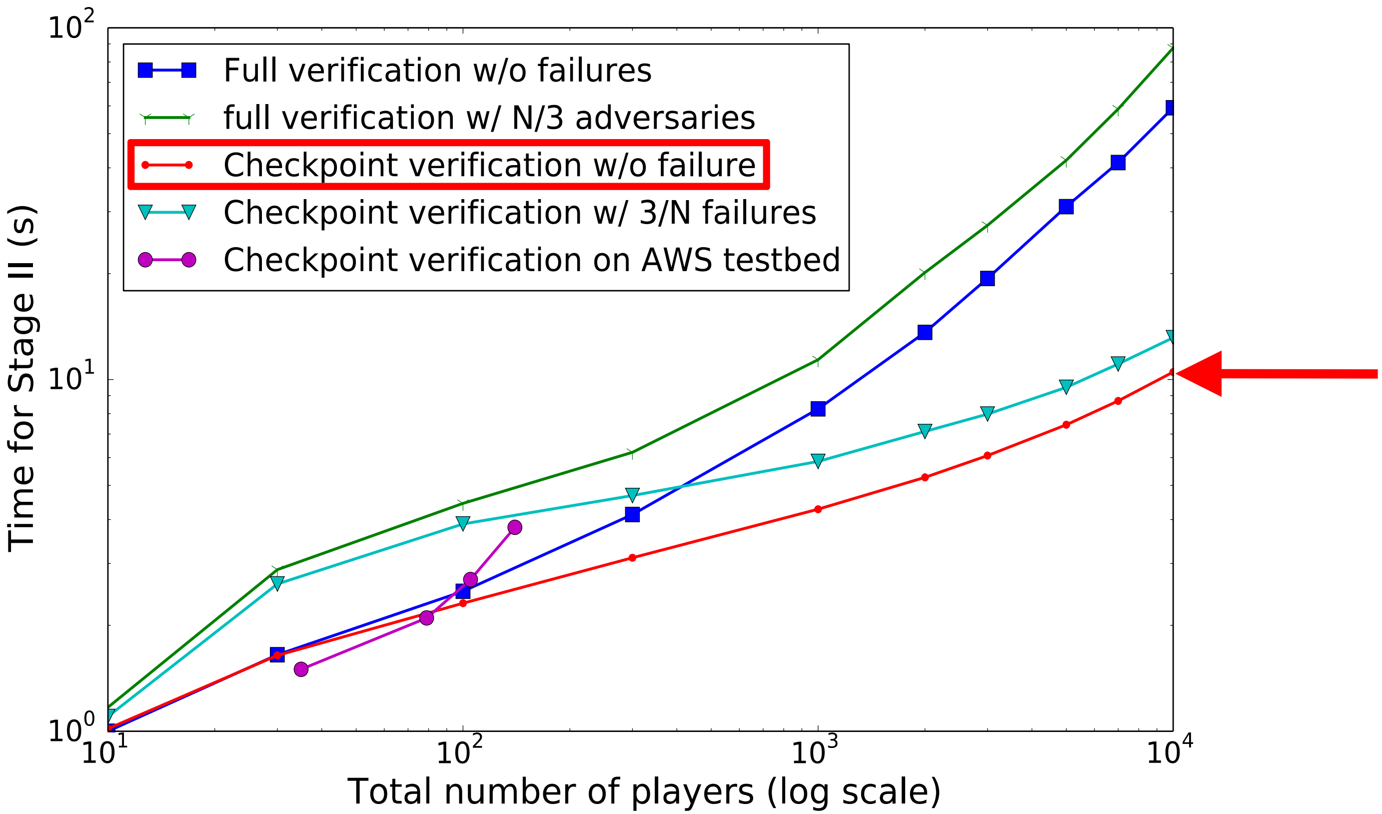
Evaluation result: Low block commit time



Good Scalability

10K Nodes:
Algorand: 12s for agreement
Gosig: 11s for agreement

Simulation Setup:
- ~300ms end-to-end latency
- 0.625ms verifying time per signer.



Outline

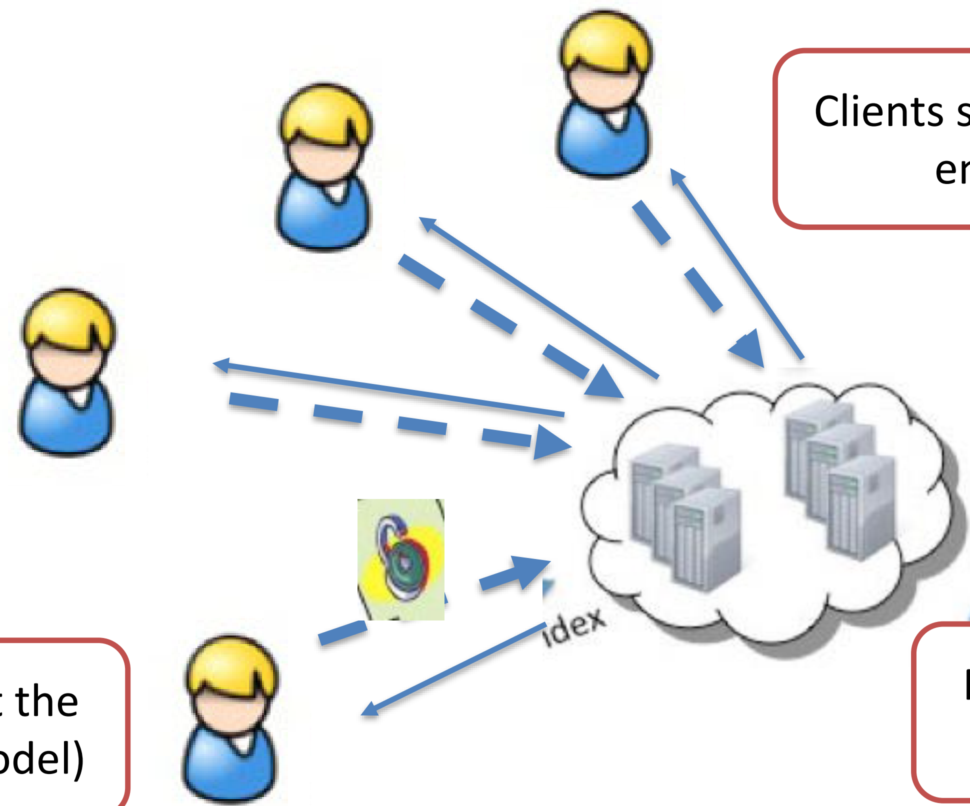
- A fast consensus protocol for consortium block chains

➢ Privacy preserving data mining framework

- Privacy + regulations, how to balance them?

Application 1:
Privacy-Preserving Data Mining

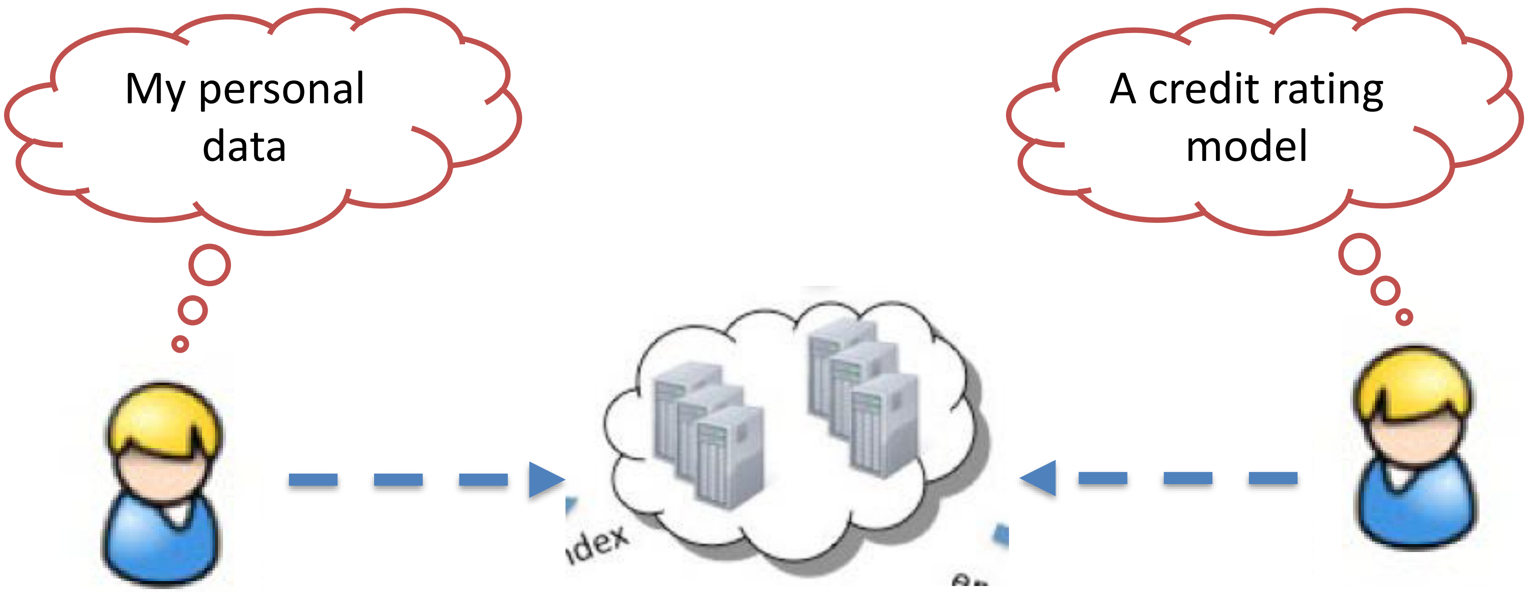
Clients send non-sensitive / encrypted data



Participants get the result (e.g. a model)

Middleman gets no information

Application 2: Private model inference



Middleman gets no information

Existing Solutions

- Garbled circuit (*Yao 1986*)
  - Sends random circuits

Expensive communication

- Fully homomorphic encryption (*Gentry 2009*)
  - Sends encrypted data

Expensive computation

- Differential privacy (*Dwork 2006*)
  - Sends data with noise

Very inaccurate results

- Secret sharing
  - Shares the data among different parties, so no single person learns about the data

Limited set of operations

Key features of our solution

- Familiar Python, automatic program optimizations
- Support different security frameworks
- Use fix-number computation for real numbers, greatly improves performance
- New efficient secret sharing operations

Plain Python APIs and code

- Dynamic matrix shape
- Easy to port legacy code
- Automatic code rewriting and optimizations
  - E.g. vector

```
xy = [x[i] * y[i] for i in range(
    dimension)]

for _ in range(round_cnt):
  for i in range(len(x) / batch):
    start = i * batch
    end = (i+1) * batch
    grads = initGrad(dimension + 1)
    wTx = privpy.dot(x[start:end], w
    [:-1].trans()) + w[-1]
    coeff = logistic(-y[start:end] *
    wTx)
    coexy = privpy.mulv(coeff, xy[start
    :end])
    coey = coeff * y[start:end]
    for j in range(batch):
      grads += privpy.append(coexy[j],
    coey[j])
    w += eta * grads / batch

print w.reveal()
```

Basic Idea about Secret Sharing

- semi-honest servers: S_1 and S_2

S_1 S_2

$$u = u_1 + u_2$$

Pick a random number

$u_2 := u - u_1 \pmod{p}$

Secret Sharing — Addition

S$_1$ S$_2$

$$u = u_1 + u_2$$

$$v = v_1 + v_2$$

$$u + v = (u_1+v_1) + (u_2+v_2)$$

Secret shares of u + v

Secret Sharing — Multiplication

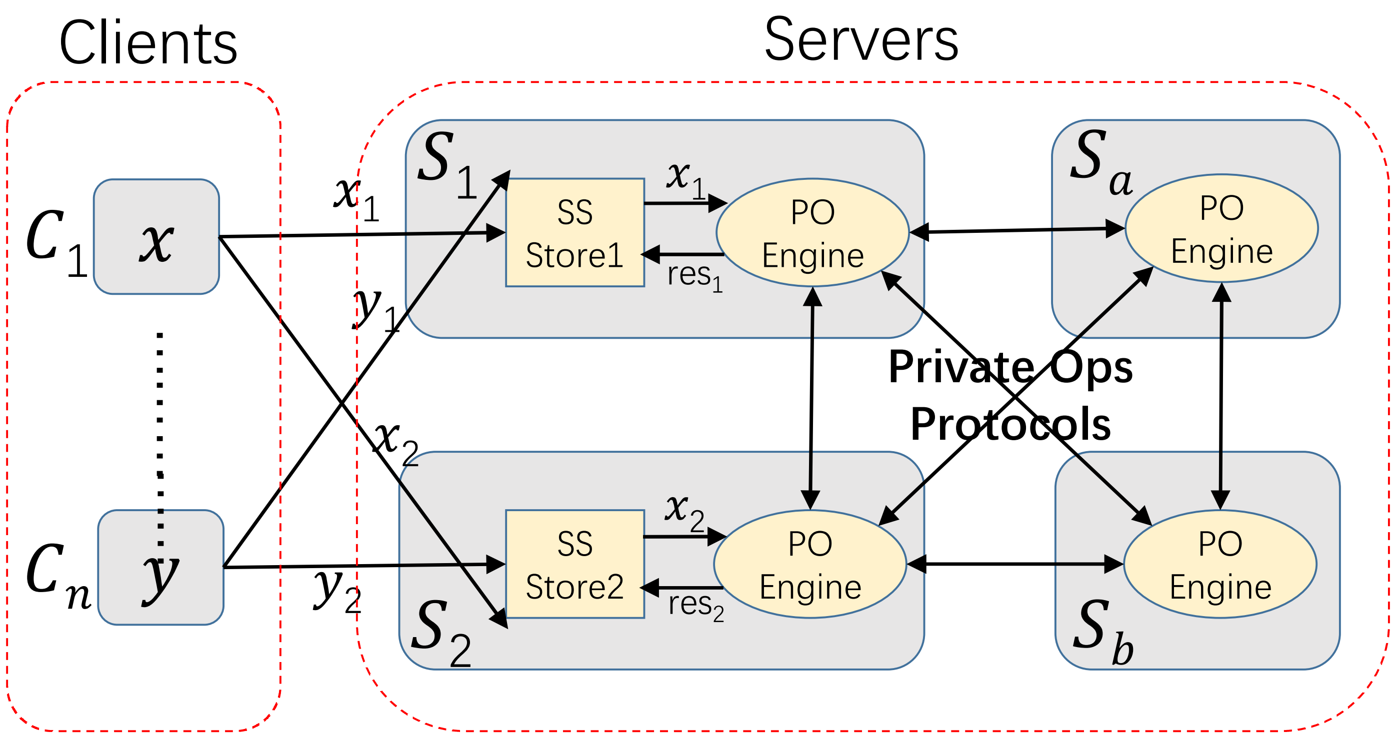
S1

S2

$$u \times v = u_1 \times v_1 + u_2 \times v_2 + u_1 \times v_2 + u_2 \times v_1$$

How to calculate the cross terms?

Our system architecture

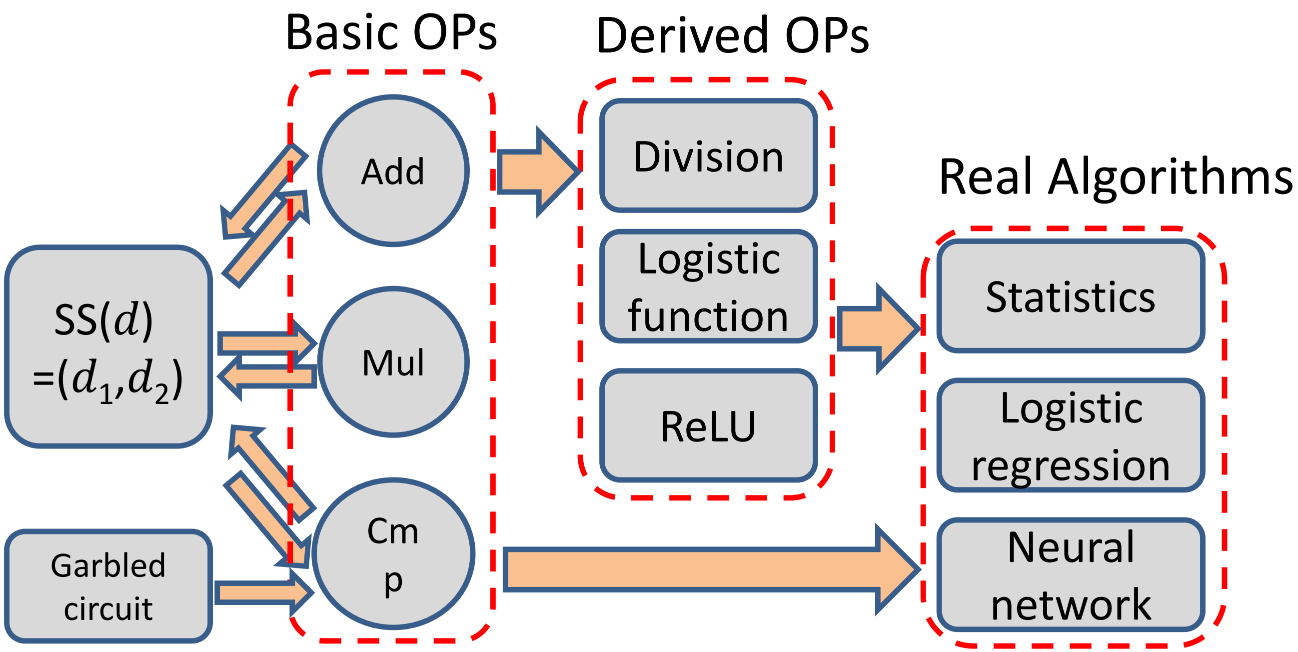


Security Assumptions

- Semi-honest servers
- No server conspiring with other servers to break the protocol

- Common assumption
  - Achievable using random server selection

Implementing real algorithms



Evaluation: basic operators

Throughput (OP/s)

	Obliv-C	HElib	SPDZ	PrivPy
mul	3,930	258	83,073	2,583,158
cmp	78,431	-	20,472	150,125

Efficient real-number multiplication

- **Sharemind: 16×**
- **SecureML: 36 ×**

Evaluation: Machine Learning Algorithms

Algorithm	Dataset	Size per instance	Time per instance (s)
Logistic regression	Adult	124	2.4e-3
K-means (5 clusters)	Credit-card	28	4.56e-3
CNN (LeNet-5)	MNIST	784	0.097

Removing the semi-honest assumption?

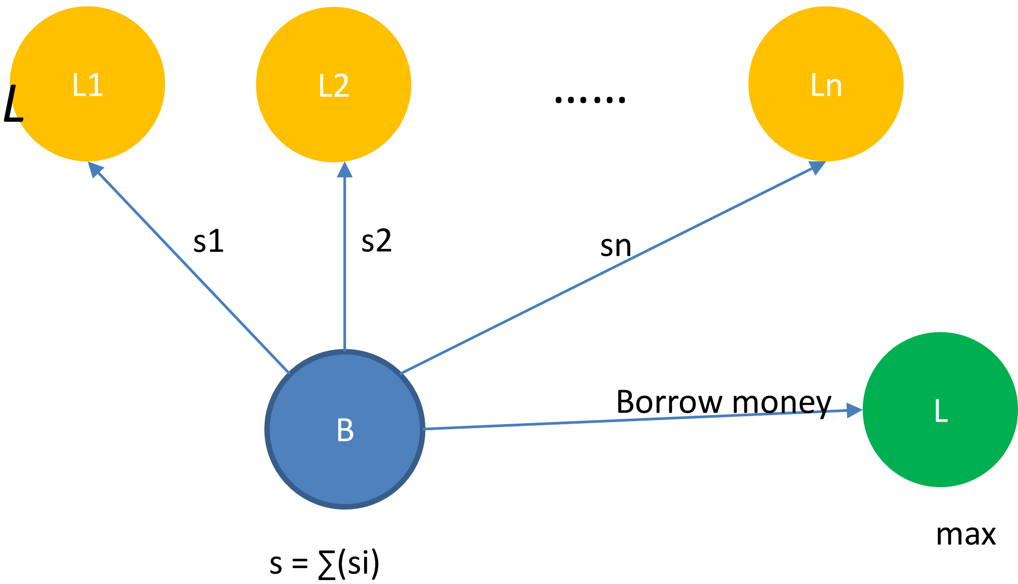
- Expensive
- Doable for certain scenarios

Example application: anti-stacking loans

- Borrower B has balance s_i from lender L_i
- Only B and L_i know s_i

- Now B is applying for a loan from L
- L wants to compute:
  - $s = \sum(s_i) < max?\ 0: 1$

- Only B and L learns about s
- Trust no one
- Do not leak anything



Outline

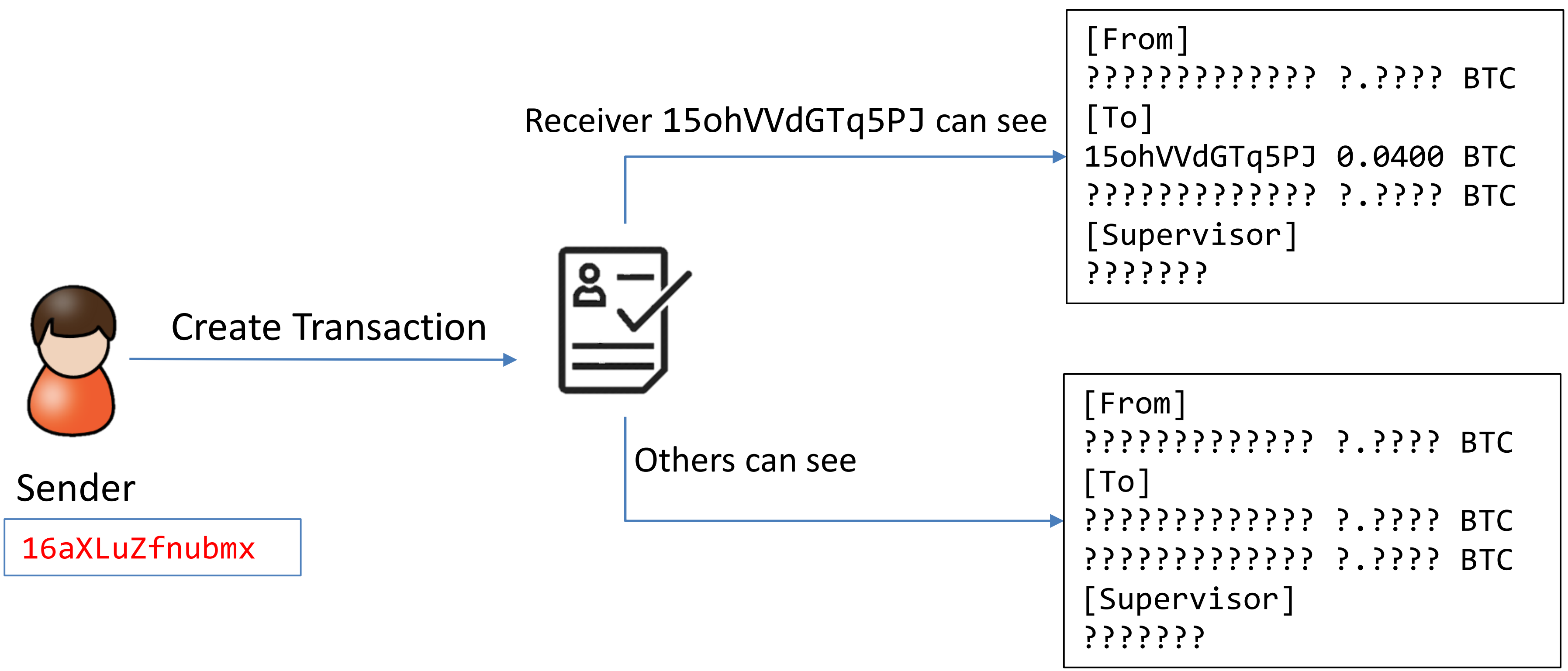
- A fast consensus protocol for consortium block chains

- Privacy preserving data mining framework

➢ Privacy + regulations, how to balance them?

Current block chains do not provide enough privacy

Zcash



Receiver 15ohVVdGTq5PJ can see
[From]
????????????? ?.???? BTC
[To]
15ohVVdGTq5PJ 0.0400 BTC
????????????? ?.???? BTC
[Supervisor]
???????
Create Transaction
Sender
16aXLuZfnubmx
Others can see
[From]
????????????? ?.???? BTC
[To]
????????????? ?.???? BTC
????????????? ?.???? BTC
[Supervisor]
???????

Zerocoin solution: Zero knowledge proof

Instance (public)

Witness (private)

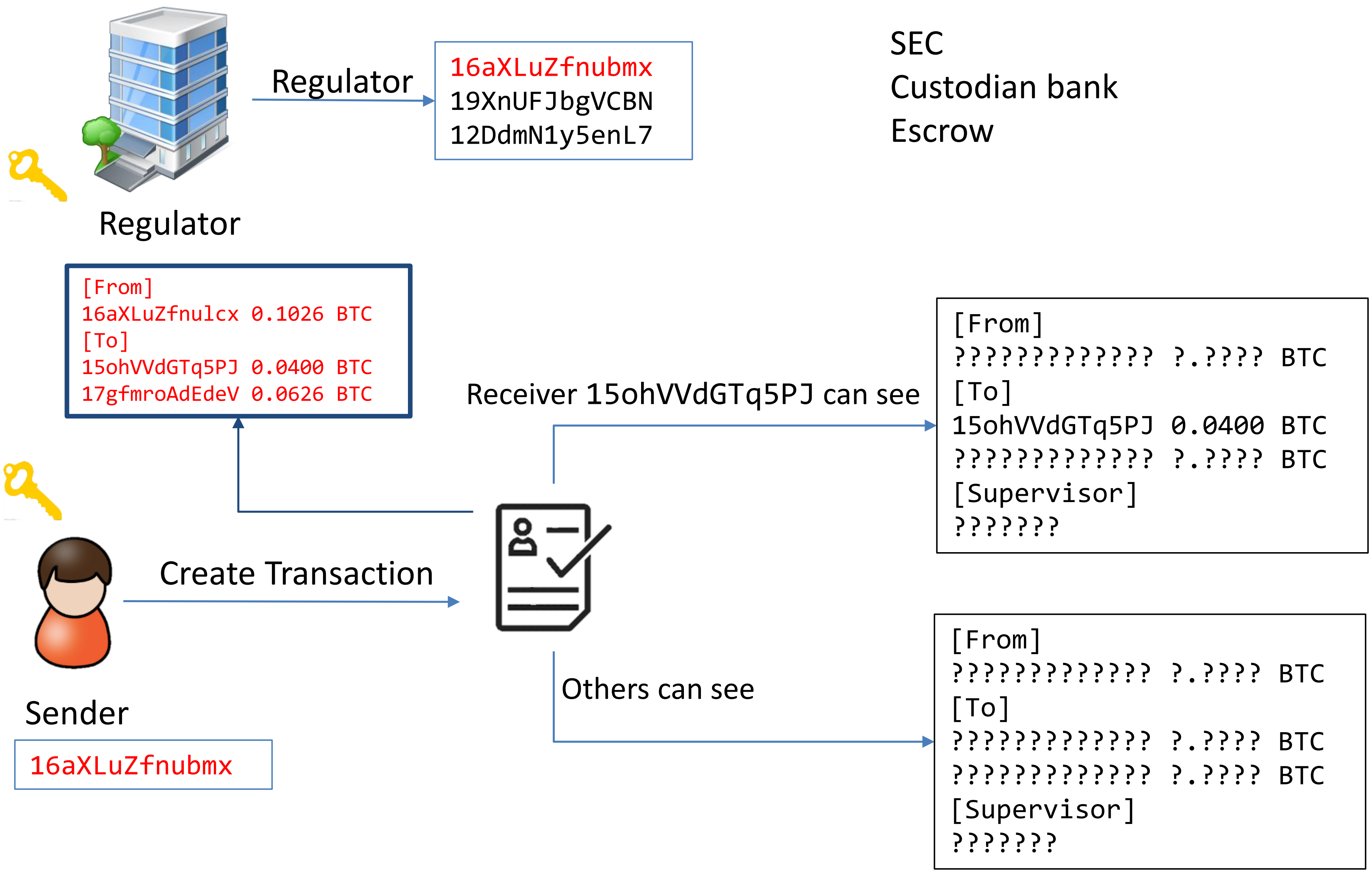
- For an NP statement (*x*, *a*)
  - Instance *x* = (rt, sn_old , cm_new , v_pub , h_Sig , h)
  - Witnesses *a* = (path, coin_old , addr_sk_old, coin_new)

- A zero-knowledge-proof is a string generated from (*x*, *a*)
  - Everyone sees *x* and the proof is convinced that (*x*, *a*) is valid
  - No information about *a* is revealed
  - In other words, one can generate the proof *if and only if* she knows *a*

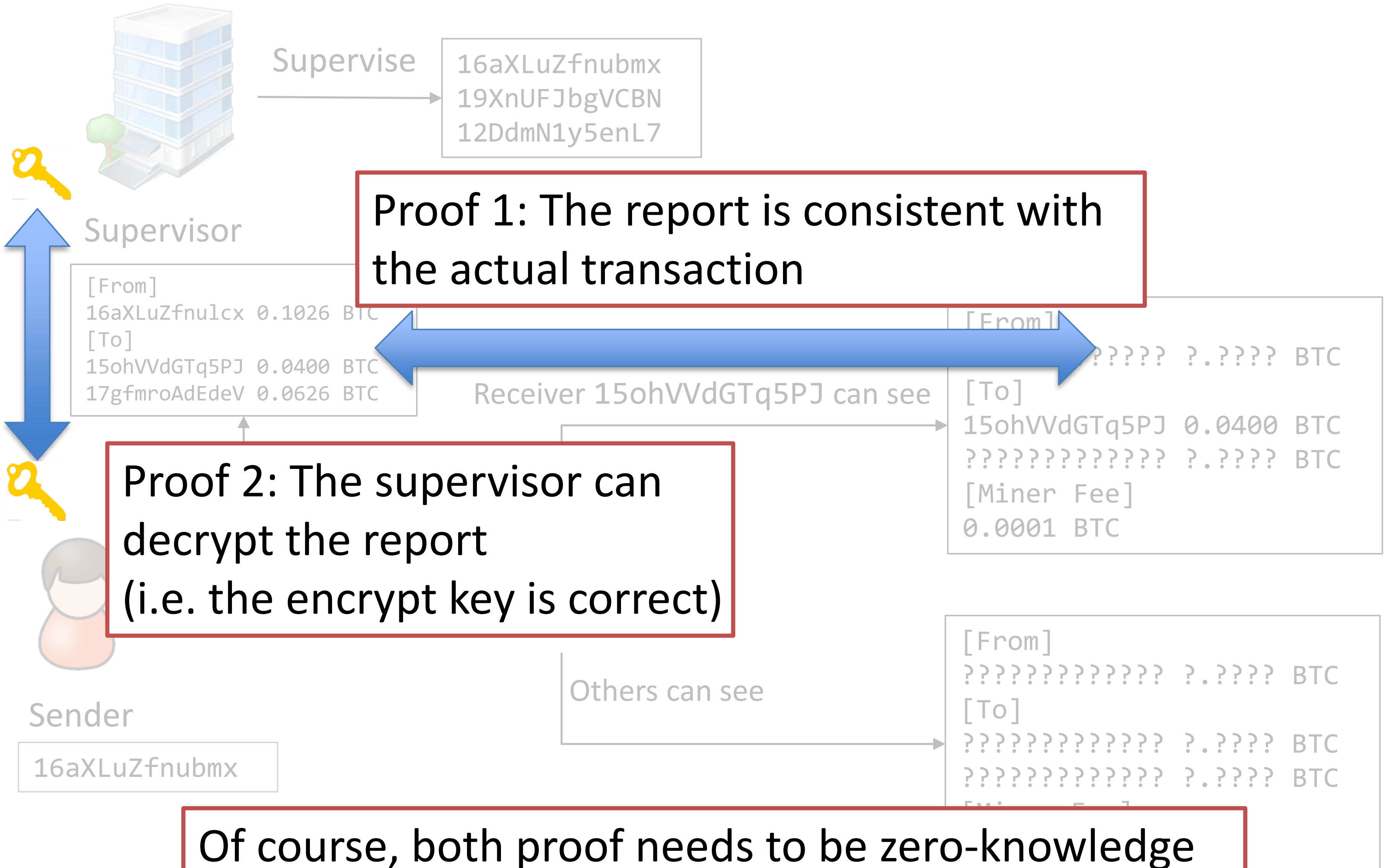
Problem with Zcash?

- Completely anonymous
- Applicable to black market

Adding a regulator – sees everything, but not on the critical parth



Extending the proof to regulator



Proof 1: The report is consistent with the actual transaction

Proof 2: The supervisor can decrypt the report (i.e. the encrypt key is correct)

Of course, both proof needs to be zero-knowledge

Evaluation

Item	Zerocash	My system
Size of proving Key	868 MB	1.68 GB
Size of verifying Key	1.42 KB	2.34 KB
Proving time	211 s	435 s
Verifying time	76.0 ms	87.7 ms
Proof size	288 B	288 B

Summary

- Financial sector in China is more or less a wild west
- The regulations / laws are way behind the technology development

- Need technology solutions

- CS is no longer just helping fintech, the other way around is also true:
  - Many new challenges, new problems
  - BFT, ZKP, GC… all find their application cases

Wei Xu
http://iiis.tsinghua.edu.cn/~xuw
We have openings for faculty, postdoc, visiting students etc.