Zeus: Analyzing Safety of Smart Contracts

Sukrit Kalra, Seep Goel, Mohan Dhawan @ IBM Research – India
Subodh Sharma @ IIT Delhi

Take Away

- Smart contracts are buggy
  - Faithful execution ensured by consensus
  - Correctness and Fairness not guaranteed
- Zeus is a framework enabling verification of smart contracts
  - Compatible with both Hyperledger Fabric and Ethereum with few changes
    - Can plugin any backend verifier
  - Works at scale
    - Study over 22.4K Solidity contracts (1524 unique)
    - Around 94% contracts (> $500M) vulnerable to correctness issues
  - Sound with low verification overhead
    - Zero false negatives, low false alarm rate
    - Takes under 1 min to analyze 97% contracts

Buggy Smart Contracts

- TheDAO bug
  - Loss of ~$60M in Ethereum blockchain
  - Resulted in a hard fork, which is often unacceptable
- Prior work (CCS'16) shows almost 1/3 contracts are buggy
  - Tons of bug reports on forums
  - Independent audit of Solidity contracts revealed 1 bug per 100 lines
- Cannot be easily edited to patch bugs
  - Manual auditing is error-prone
    - Natural language contracts easier for non-programmers to understand
  - Prior work is neither sound nor complete and has many false alarms
  - Formal verification is necessary

Correctness and Fairness

- Correctness (Verification)
  - Adherence to best practices to avoid common but buggy coding paradigms
  - Determined by low-level code analysis
- Fairness (Validation)
  - Contract does exactly what it claims
  - Determined by high-level policy enforcement

Motivating Example

```
while (Balance > (depositors[index].Amount*115/100)
                              && index < Total_investors) {
    if (depositors[index].Amount != 0) {
        payment = depositors[index].Amount * 115/100;
        depositors[index].EtherAddress.send(payment);
        Balance -= payment;
        Total_Paid_Out += payment;
        depositors[index].Amount = 0;
    } break;
}
```

- Correctness: Arithmetic operations can overflow
- Fairness: `index` is never incremented

Example 1: Integer Overflow

```
uint payout = balance/participants.length;
for (var i = 0; i < participants.length; i++)
   participants[i].send(payout);
```

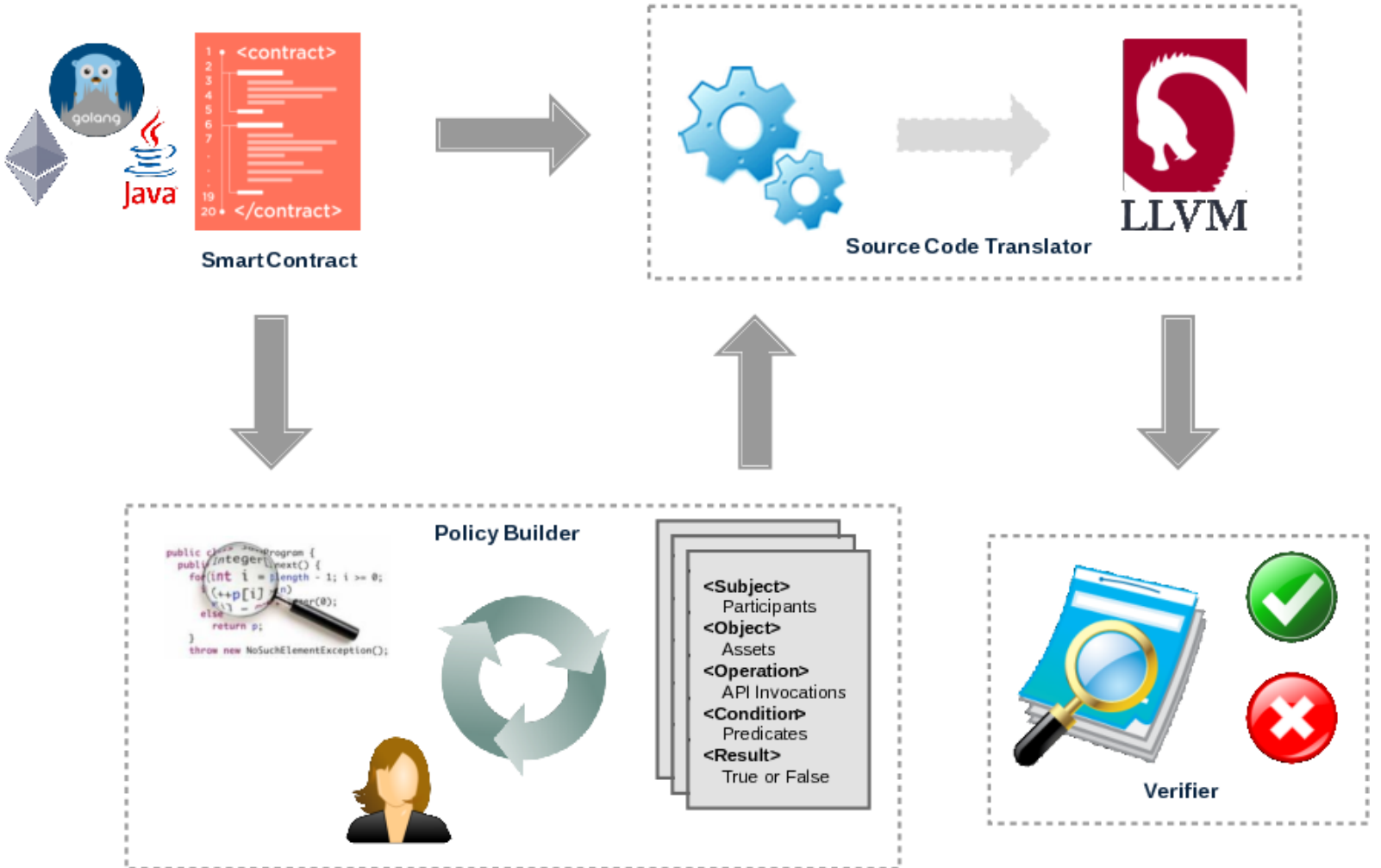
- Solidity is strongly typed
  - Implicit extending of signed/unsigned integers is forbidden
- Over 20 different scenarios for integer overflow or underflow

Example 2: Unfair Auction

```
function placeBid(uint auctionId) returns (bool success) {
    Auction a = auctions[auctionId];
    if (a.currentBid >= msg.value) throw;
    uint bidIdx = a.bids.length++;
    Bid b = a.bids[bidIdx];
    b.bidder = msg.sender;
    b.amount = msg.value;
    ...
    BidPlaced(auctionId, b.bidder, b.amount);
    return true;
}
```

- An auction can be “with reserve” or “without reserve”
- In “with reserve”, seller can bid only if disclosed to participants

Zeus


golang
Java
<contract>
</contract>
SmartContract
Source Code Translator
LLVM
Policy Builder
<Subject>
Participants
<Object>
Assets
<Operation>
API Invocations
<Condition>
Predicates
<Result>
True or False
Verifier

Policy Builder

<Subject, Object, Operation, Condition, Result>

```
<Subject> msg.sender </Subject>
<Object> a.seller </Object>
<Operation trigger=”pre”> placeBid </Operation>
<Condition> a.seller != msg.sender </Condition>
<Result> True </Result>
```

- Leverage user assistance to build a XACML styled policy specification
- Zeus runs a taint analysis pass over the contract code with sources as contract and runtime defined global variables and sinks as invocations to external APIs

Source Code Translator

- Source code (GO / Solidity) to LLVM bitcode
  - Use `llgo` for GO to LLVM translation
  - Built our own Solidity to LLVM translator
- Ensuring soundness
  - Correctly reason about all execution orders
  - Need to traverse all possible paths
- Modeling syntax
  - Inheritance, Arrays, External functions
- Handling LLVM Optimizations

Verifier

- `assert` statements used to generate verification conditions as Constrained Horn Clauses (CHCs)
- CHCs are fed to SMT solvers which determine if the `assert` can fail or not
- Verification is fast since CHCs are discharged efficiently

Implementation

- Hyperledger Fabric
  - Leveraged `llgo` for translation
  - Mockstub takes in string input, while Zeus' verification is atop integers
    - Need to modify contract code to take in integers
  - Need to manually insert `assert` statements for verification checks
- Ethereum
  - Built Solidity to LLVM translator
  - Automatically inserts conditions to be checked as `assert` statements
- Compatibility with other verifiers
  - Tested with Seahorn and SMACK

Evaluation (Ethereum)

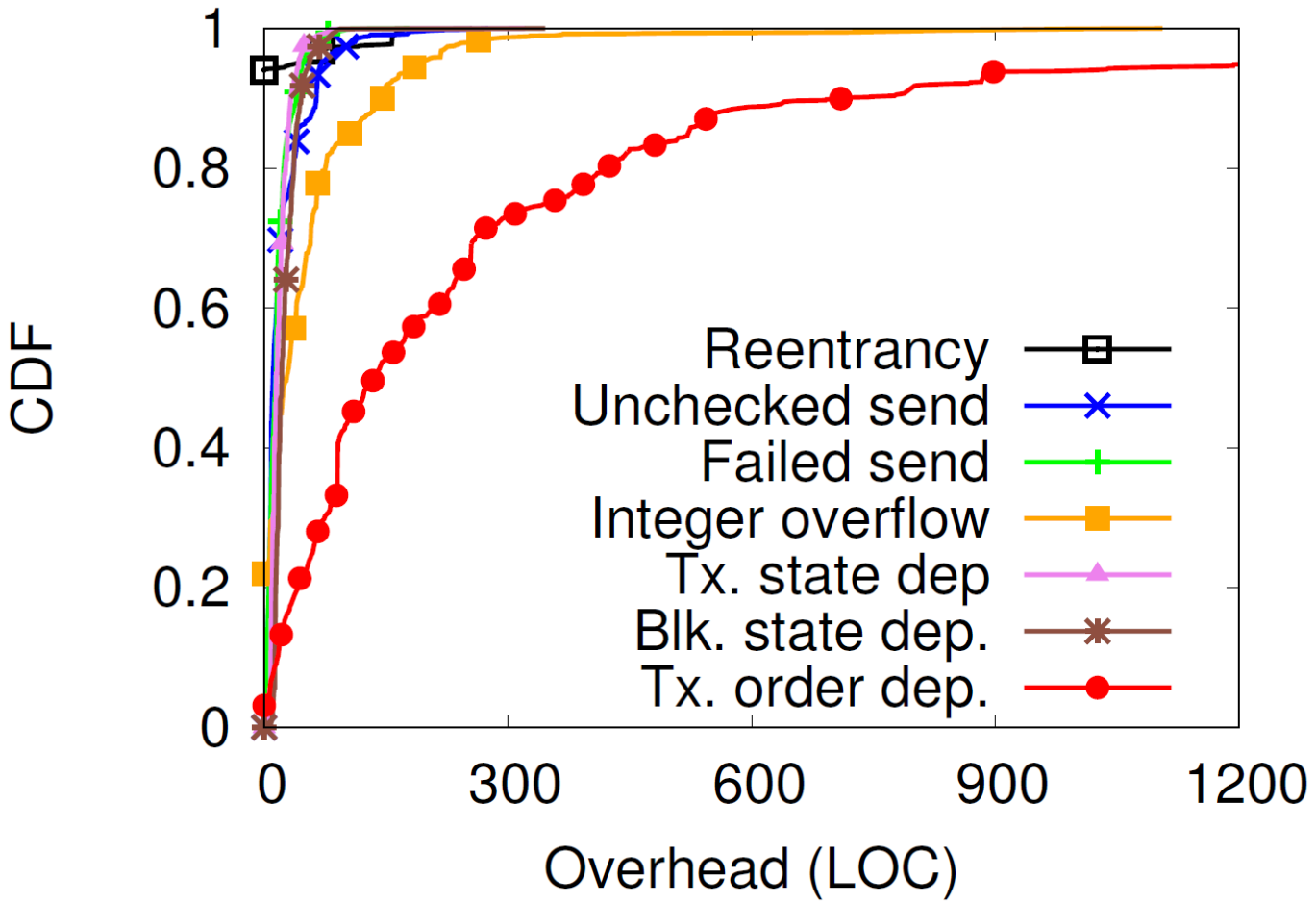
- Data set
  - Study over 22.4K Solidity contracts at unique addresses
  - 1524 contracts with unique source code

Category	#Contracts	Lines of Code	
		Source	LLVM
DAO	140	2.8	24.3
Game	244	23.3	609.2
Token	290	25.2	385.9
Wallet	72	10.8	105.9
Misc	778	47.6	924.3
Total	**1524**	**109.7**	**2049.6**

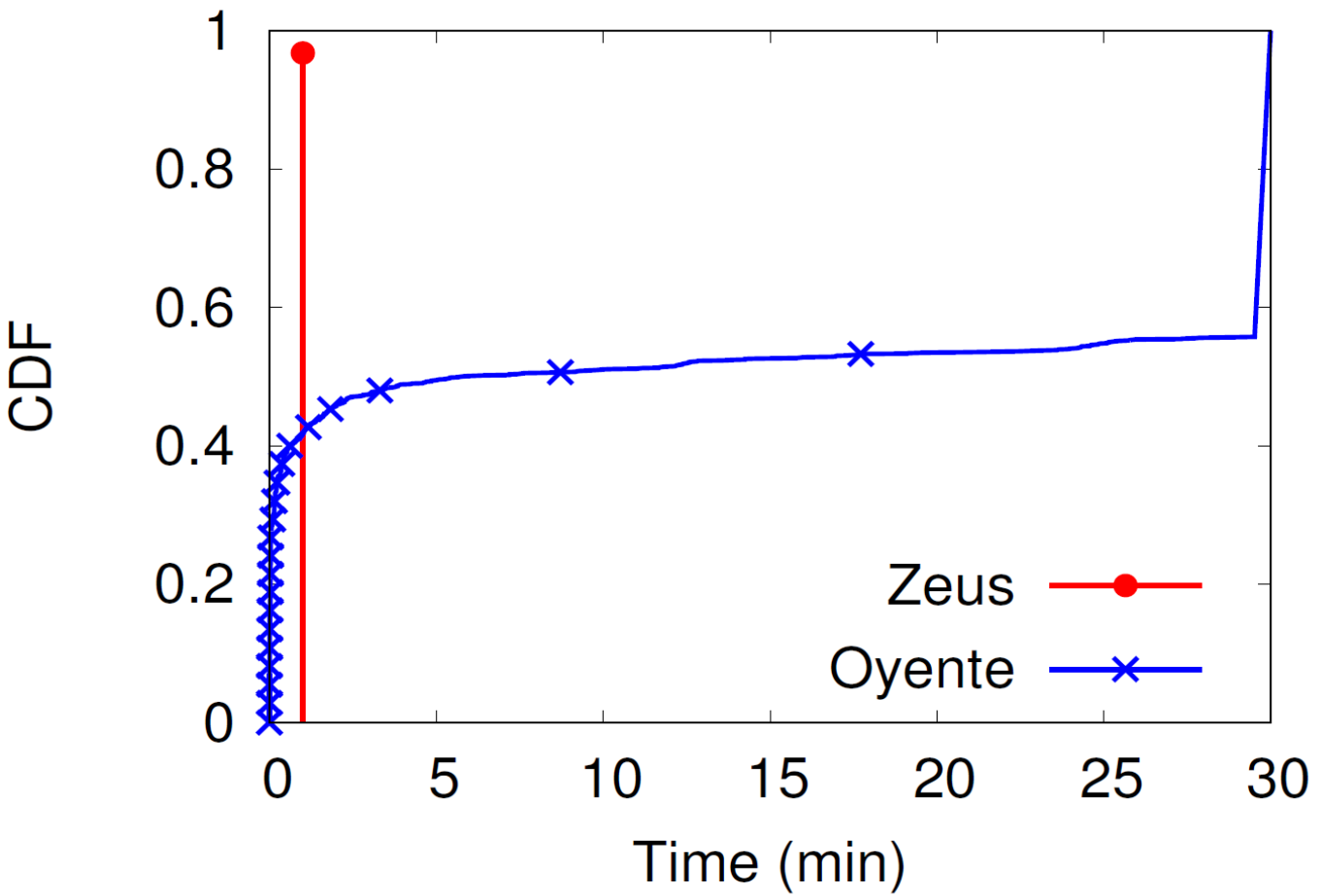
Correctness: Comparison w/ Oyente

Bugs	Zeus							Oyente						
	Safe	Unsafe	No Result	Timeout	False +ve	False -ve	% Alarms	Safe	Unsafe	No Result	Timeout	False +ve	False -ve	% Alarms
Reentrancy	1428	54	7	25	0	0	0.00	548	265	226	485	254	51	31.24
Unchecked Send	1191	324	5	4	3	0	0.20	1066	112	203	143	89	188	7.56
Failed Send	1068	447	3	6	0	0	0.00							
Integer Overflow	378	1095	18	33	40	0	2.72							
Tx. State Dependence	1513	8	0	3	0	0	0.00							
Block State Dependence	1266	250	3	5	0	0	0.00	798	15	226	485	2	84	0.25
Tx. Order Dependence	894	607	13	10	16	0	1.07	668	129	222	485	116	158	14.20

Overhead: Verification Checks



Overhead: Verification Time



Take Away

- Smart contracts are buggy
  - Faithful execution ensured by consensus
  - Correctness and Fairness not guaranteed
- Zeus is a framework enabling verification of smart contracts
  - Compatible with both Hyperledger Fabric and Ethereum with few changes
    - Can plugin any backend verifier
  - Works at scale
    - Study over 22.4K Solidity contracts (1524 unique)
    - Around 94% contracts (> $500M) vulnerable to correctness issues
  - Sound with low verification overhead
    - Zero false negatives, low false alarm rate
    - Takes under 1 min to analyze 97% contracts

Blockchain @ India Research Lab

Industry Leading Solutions

Blockchain for International Trade

Provide trusted, secure, tamper-proof tracking of trade documents and events

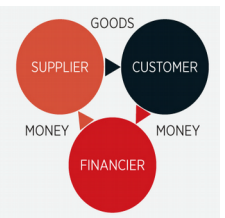
MÆRSK

موانئ دبي العالمية
DP WORLD

جمارك دبي
DUBAI CUSTOMS

Trade/Supply Chain Finance

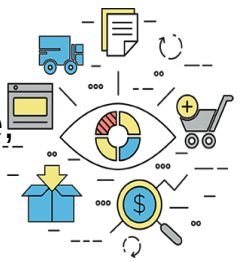
Improve the efficiency and support new models of Financing for Supply Chains and Trade by enabling secure and role-based visibility to key supply chain and trade processes



Mahindra Finance

Supply Chain Visibility

Orchestration of Supply Chains for Controls, Compliance, and Visibility



Academic Research and Open Source Contributions

HYPERLEDGER FABRIC

- Key Contributions to the Data and Ledger Management in Hyperledger Fabric v1 (LevelDB, CouchDB)
- Ongoing research in the area of smart contract verification, performance optimization and scaling, intersection of databases and blockchain.
- Mechanism Design for Blockchain Assisted Supply Chain Collaboration
- Collaborations with multiple leading universities in India

- Co-organized workshop at IDRBT, Hyderabad
- Organized “Emerging Topics in Blockchain Research” workshop at IRL in July 2017
- Numerous external presentations, panels and demos
- 20+ patent applications filed