TRUESET: Faster Verifiable Set Computations

Ahmed E. Kosba[†], Dimitrios Papadopoulos[‡], Charalampos Papamanthou[†]
Mahmoud F. Sayed[†], Elaine Shi[†], Nikos Triandopoulos[§‡]

† University of Maryland, College Park
‡ Boston University
§ RSA Laboratories

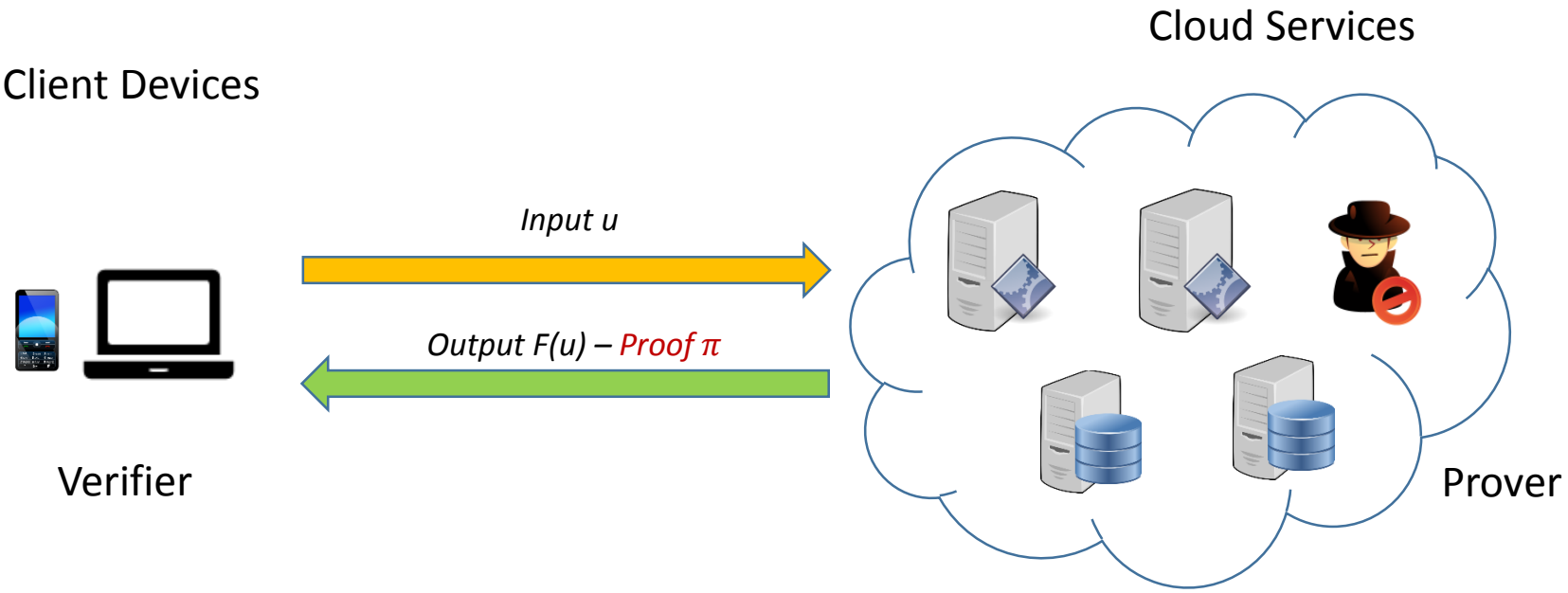


USENIX Security'14
August 22nd, 2014

RSA

BOSTON UNIVERSITY

Outsourcing of Storage and Computations



- Integrity/Correctness Concerns ➡ Verifiable Computation (VC)

- Making VC practical

  Short Proof - Short Verification Time - Short Proof Computation Time

  ⬆

  Not there yet!!

Verifiable Set Operations

Applications

SQL Join Queries

```
SELECT UNIVERSITY.id
FROM UNIVERSITY JOIN CS
ON UNIVERSITY.id = CS.id
```

Jaccard index

$$\texttt{Similarity} = \frac{|A \cap B|}{|A \cup B|}$$

- The proof computation time is very high for current generic VC systems.
  - It can take 100+ seconds to produce a proof for an intersection of two 256-element sets.
- **TRUESET** provides orders of magnitude better performance
  - More than 100x Speed-up achieving < 1 second in the above case.

Verifiable Computation

Approaches:

- Secure hardware based
- Replication based
- Cryptography based

BCGTV [Ben-sasson et al, Crypto'13]
Pinocchio [Parno et al, IEEE S&P'13]
Pantry [Braun et al, SOSP'13]

Characteristics:

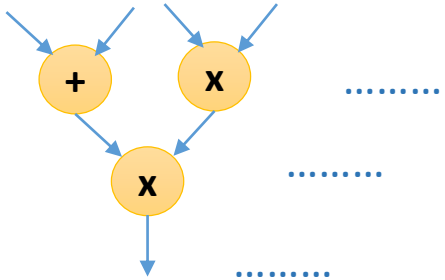
- Compact Constant-size Proof, e.g. 288 bytes for Pinocchio
- Short Verification Time: O(size of IO)
- High Proof Computation Time

```
void func(struct Input* in,
struct Output* out){

    /* subset of C */

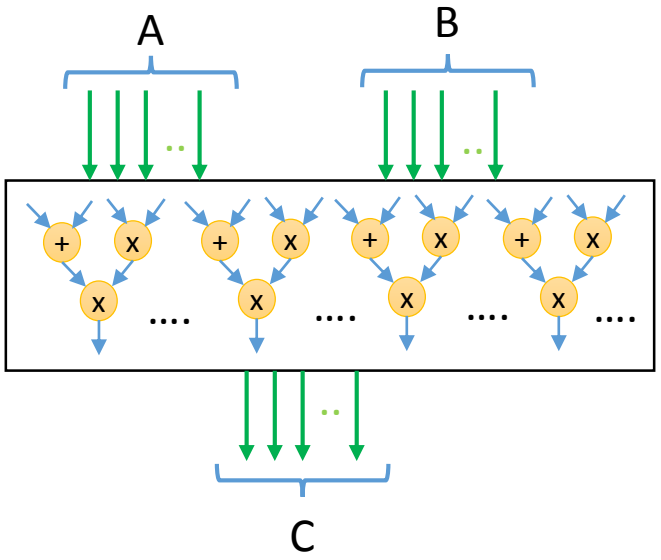
}
```



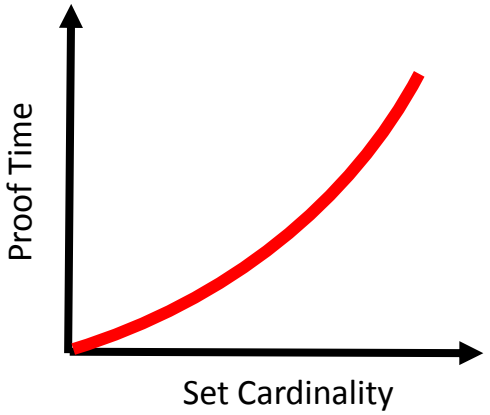


Each individual operation is mapped to a set of gates or constraints

Arithmetic Representation of Set Operations is **Expensive**



Arithmetic Set Circuit



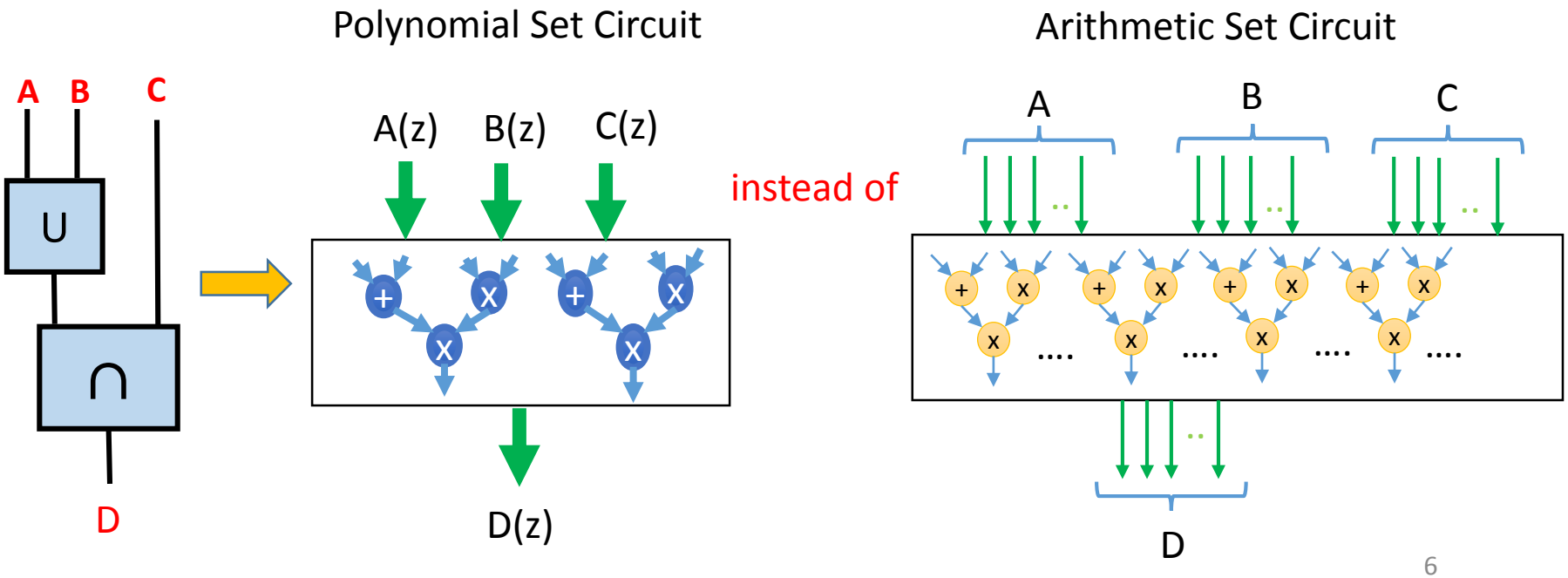
- Another challenge: Have to account for the worst-case set size during proof computation.

TRUESET

Goals:

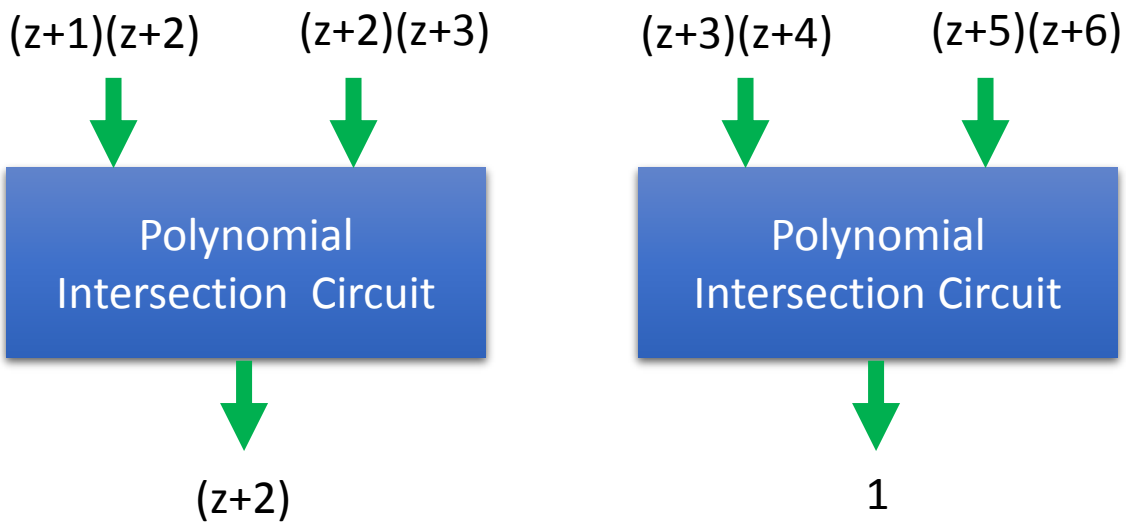
- Reduce proof computation time for set operations
- Achieve input-specific running time for the prover
- Retain the expressiveness of previous techniques

Main Idea:



Sets as Polynomials

- Represent a set A = { a_1, a_2, …, a_n} by an n-degree polynomial A(z) = $(z+a_1)(z+a_2)$ .. $(z+a_n)$



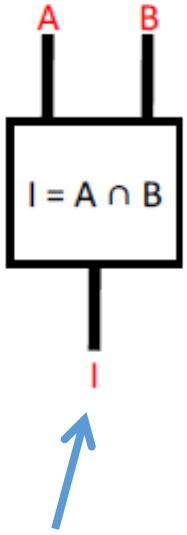
Two Primary Advantages:

- The circuit size is constant for set operations.
- The effort correlates with the degrees of the polynomials on the wires.

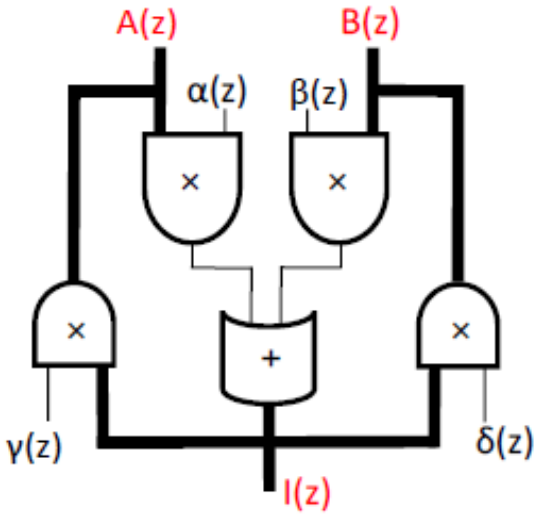
How to build O(1) circuits for set operations?

Efficient Set Circuits

- Intersection Gate



GCD(A, B)



I(z) = GCD(A(z), B(z)) **iff** there exists polynomials $\alpha(z), \beta(z), \gamma(z), \delta(z)$ such that

$$\Rightarrow \alpha(z)A(z) + \beta(z)B(z) = I(z)$$

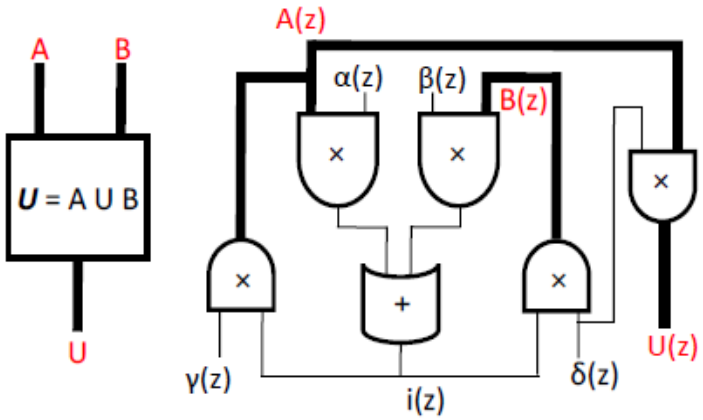
$$\Rightarrow \gamma(z)I(z) = A(z)$$

$$\Rightarrow \delta(z)I(z) = B(z)$$

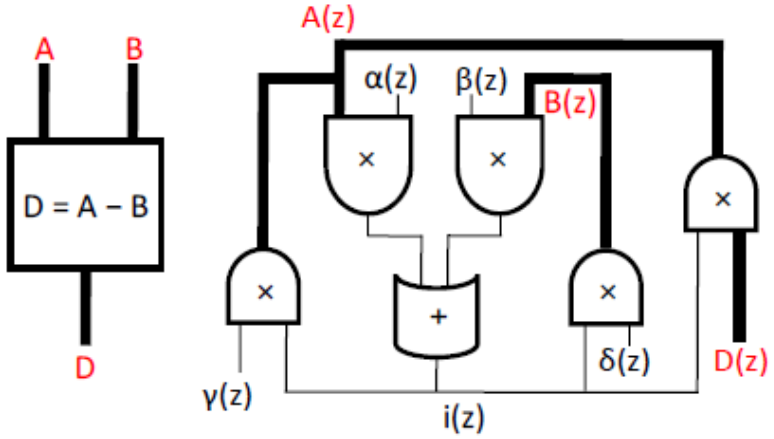
- The witness polynomials can be calculated by the Extended Euclidean algorithm for polynomials.

Efficient Set Circuits

- Union and Difference gates can be built similarly.



$$\alpha(z)A(z)+\beta(z)B(z)=i(z)$$
$$\gamma(z)i(z)=A(z)$$
$$\delta(z)i(z)=B(z)$$
$$\delta(z)A(z)=U(z)$$

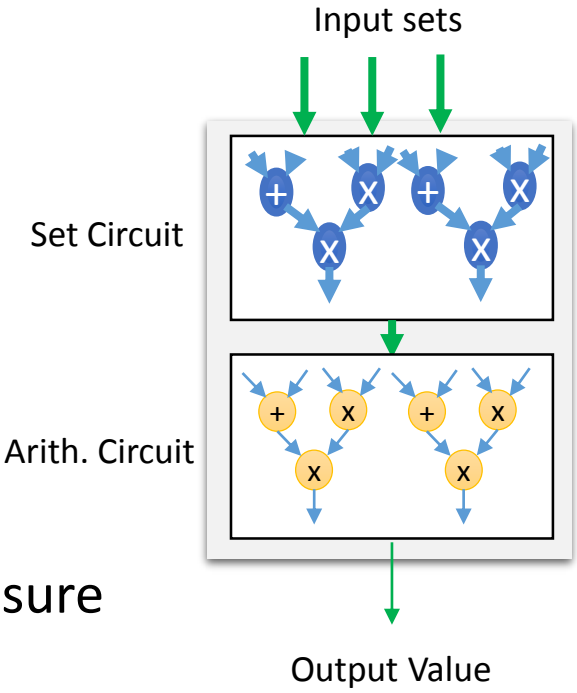


$$\alpha(z)A(z)+\beta(z)B(z)=i(z)$$
$$D(z)i(z)=A(z)$$
$$\delta(z)i(z)=B(z)$$

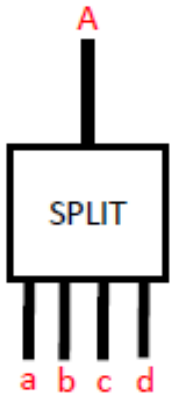
Retaining Expressiveness

- Hybrid Queries:

```
SELECT COUNT(UNIVERSITY.id)
FROM UNIVERSITY JOIN CS
ON UNIVERSITY.id = CS.id
```



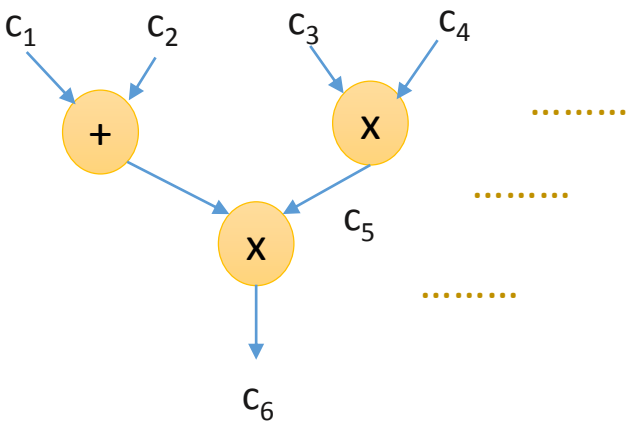
- TrueSet provides a set of useful gates to ensure expressiveness
  - Zero-degree assertion gate.
  - Split and Merge gates.
  - Cardinality gate.



How to build verifiable **polynomial** circuits protocol?

[Gennaro et al. EUROCRYPT'13, Parno et al. IEEE S&P'13]

Quadratic Arithmetic Programs (QAPs)



Equivalent Constraints

$c_5 = c_3.c_4$
$c_6 = c_5.(c_1 + c_2)$
...



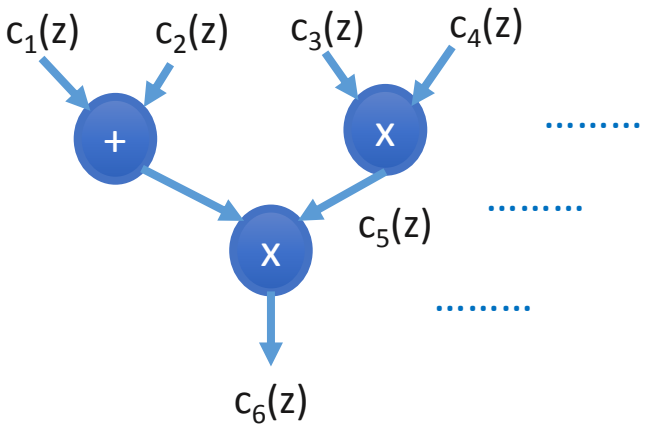
$$(\sum_{k=1}^{m} c_k v_k(x))\,(\sum_{k=1}^{m} c_k w_k(x)) - (\sum_{k=1}^{m} c_k y_k(x)) = t(x)h(x)$$

where

$t(x) = (x - r_1)\,(x - r_2)\,..\,(x - r_d)$

v_k, w_k and y_k are polynomials defined based on the circuit structure.

Quadratic Polynomial Programs (QPPs)



Equivalent Constraints

$c_5(z) = c_3(z).c_4(z)$
$c_6(z) = c_5(z).(c_1(z) + c_2(z))$
...



$$(\sum_{k=1}^{m} c_k(z)v_k(x))\,(\sum_{k=1}^{m} c_k(z)w_k(x)) - (\sum_{k=1}^{m} c_k(z)y_k(x)) = t(x)h(x,z)$$

Bivariate Polynomial

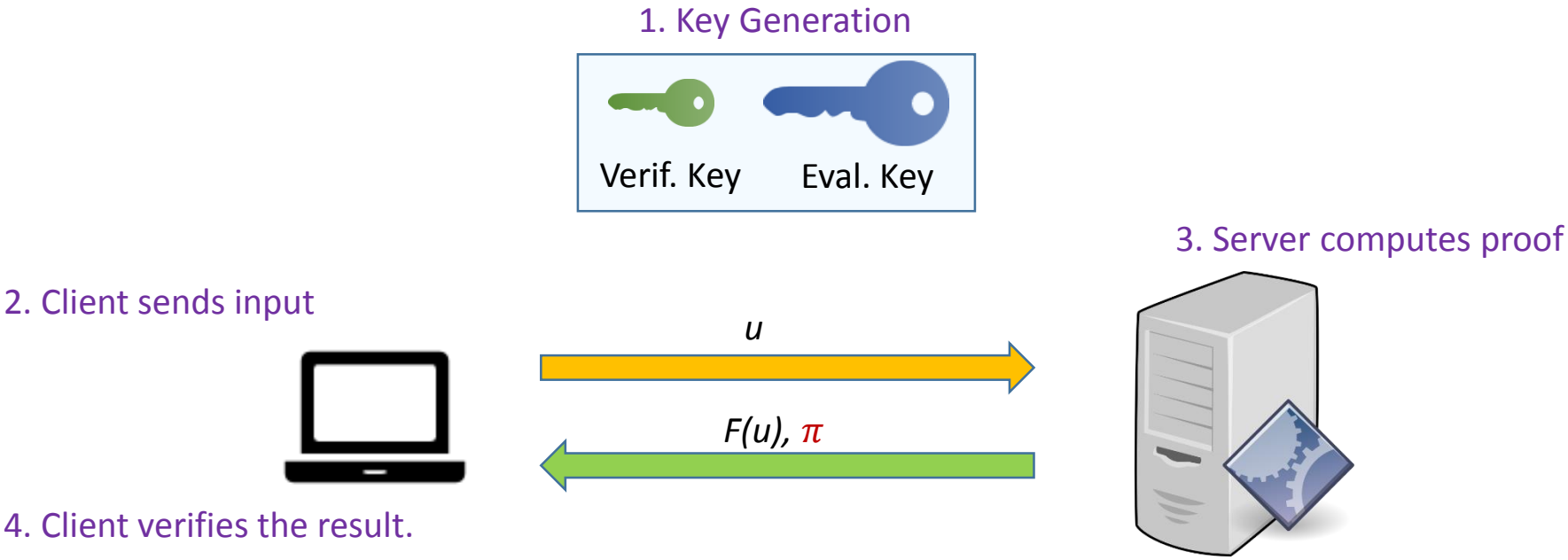
where

$t(x) = (x - r_1)\,(x - r_2)\,..\,(x - r_d)$

v_k, w_k and y_k are polynomials defined based on the circuit structure.

Verifiable Polynomial Circuits

- Protocol outline:



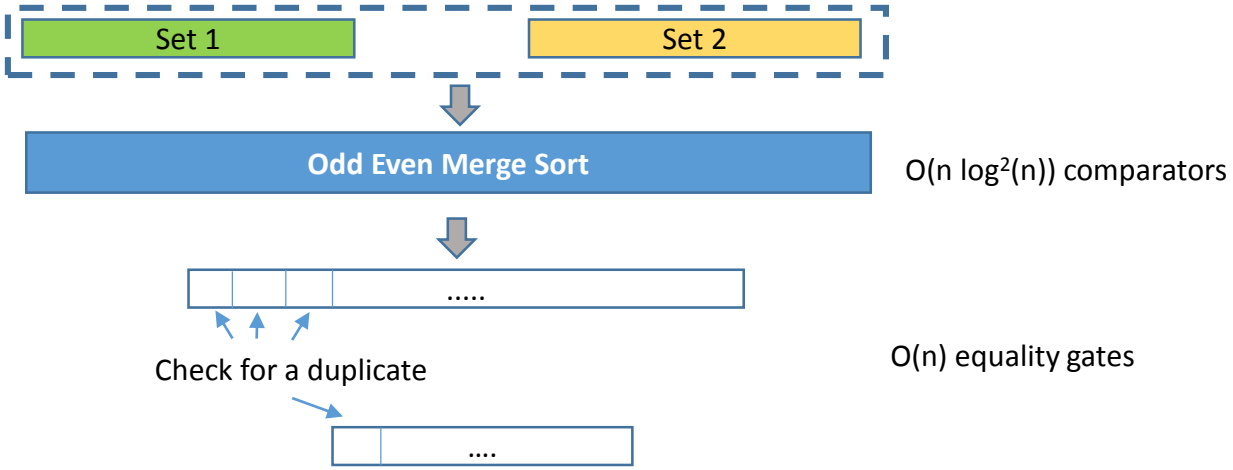
Implementation

- Added support to Pinocchio's C++ implementation to handle verifiable polynomial circuits with loops.
- Used open-source libraries to handle field and crypto operations: NTL and nifty ate-pairing.
- Operations are done in a Field F_p where p is a 254-bit prime. Bit security level is 127.
- Comparison with two Pinocchio implementations:
  - The original executable by Microsoft Research (MS-Pinocchio)
  - An executable that uses the same polynomial and crypto libraries as TrueSet (NTL-ZM Pinocchio)

Evaluation

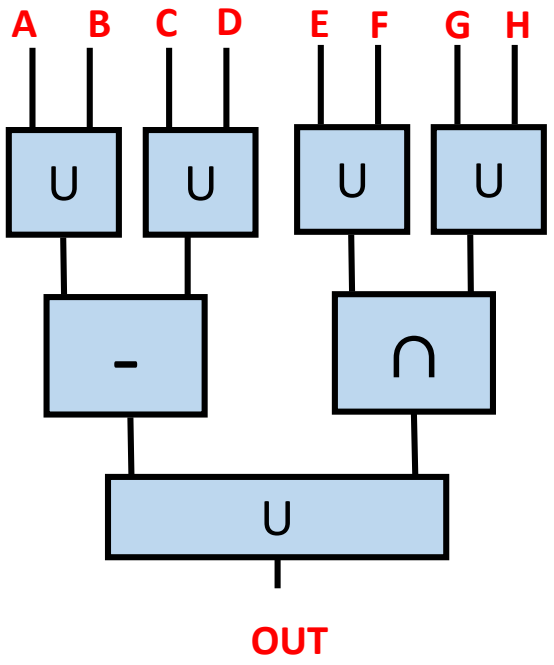
- Comparison:
  - Two variants for Pinocchio set circuit programs:
    - A pair-wise approach requiring $O(n^2)$ equality-check gates.
    - A sorting-network approach requiring $O(n \log^2(n))$ comparator gates.

Example Intersection Circuit using a Sorting Network



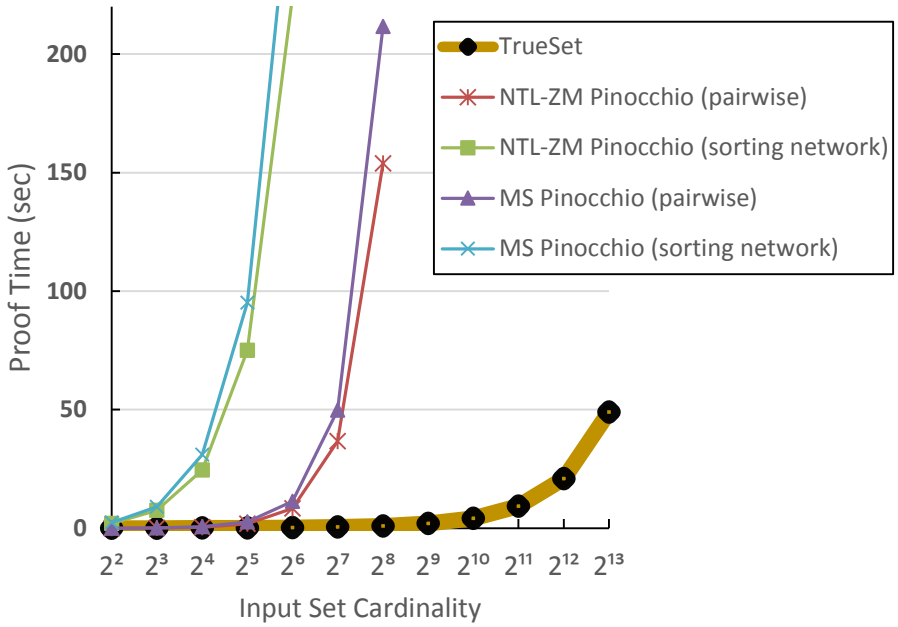
Evaluation

- Set Programs:
  - Single union operation
  - Multi set operations
- The input sets contain random elements from the field F_p.
- For each input set size, a different circuit was produced for Pinocchio alternatives.



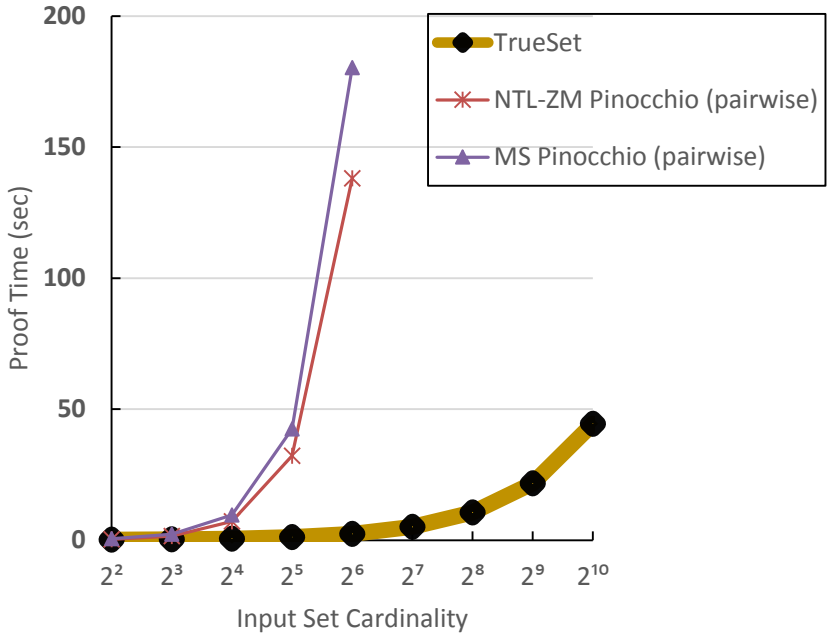
Proof Computation Speedup

Proof Computation – Single Gate



150x improvement when |s| = 256

Proof Computation – Multi-gate



> 50x improvement when |s| = 64

- **More than 90% savings in the evaluation key sizes.**
- **Retain almost similar verification times and verification keys sizes.**

|s| refers to each input set size

Optimizations / Extensions

- Optimizations
  - Bivariate polynomial operations
  - Randomized check for output polynomial
- Case of outsourced sets
  - Usage of Merkle trees and bilinear accumulators.
- TrueSet provides inherent support for multisets, while other approaches will require more complexity.

Conclusions

- **TRUESET** a system that aims at reducing proof computation time for verifiable set computations.

- Modeling set operations as polynomial circuits helped achieve:
  - Much better proof computation time (More than 100x when set size is 256)
  - Great savings (> 90%) in the circuit evaluation key size
  - Input-specific running time for the prover

- Is this practical yet?

Thank You ☺

Questions?

akosba@cs.umd.edu