

Security and Privacy in Cloud Computing

Ragib Hasan
Johns Hopkins University
en.600.412 Spring 2010

Lecture 6
03/22/2010

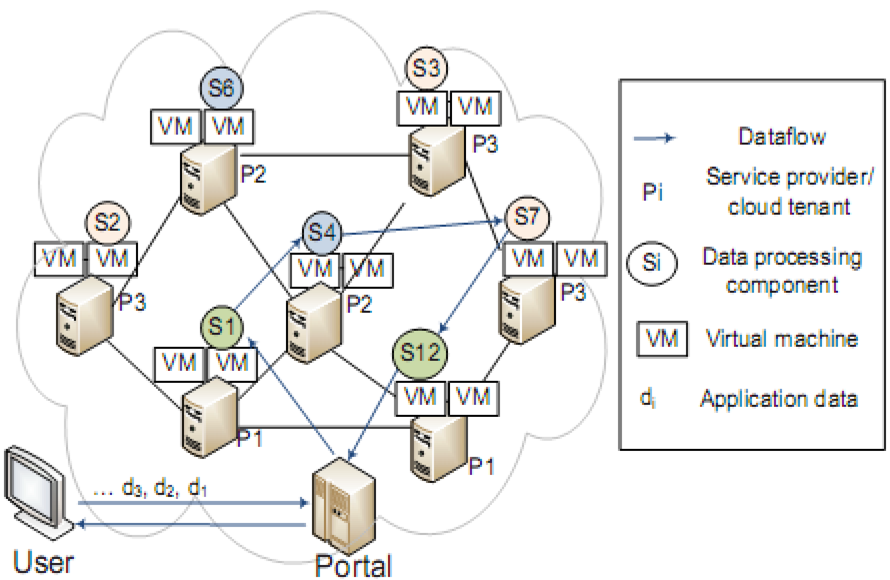
Verifying Computations in a Cloud

Scenario

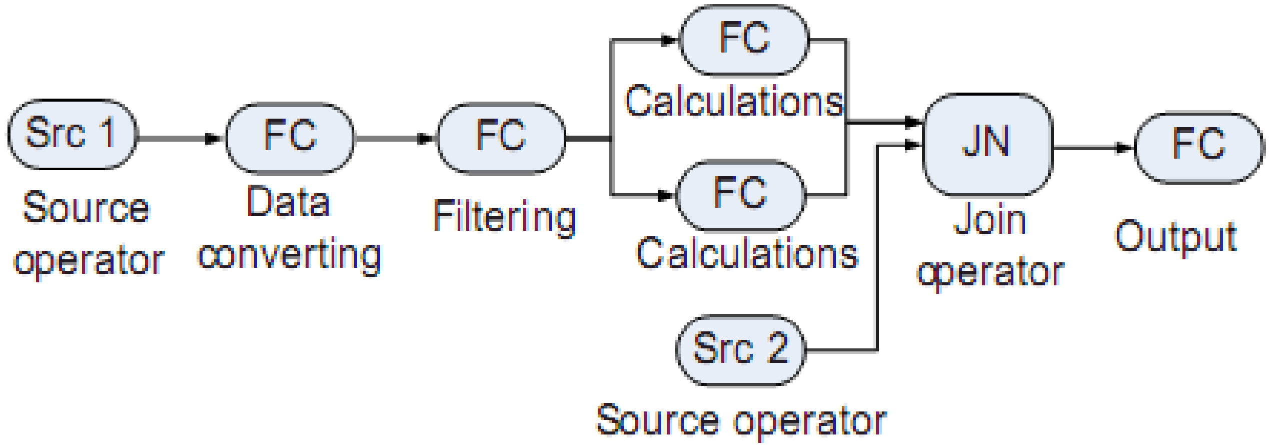
User sends her data processing job to the cloud.

Clouds provide dataflow operation as a service (e.g., MapReduce, Hadoop etc.)

Problem: Users have no way of evaluating the correctness of results



DataFlow Operations



Properties

High performance, in-memory data processing

Each node performs a particular function

Nodes are mostly independent of each other

Examples

MapReduce, Hadoop, System S, Dryad

How do we ensure DataFlow operation results are correct?

Goals

- To determine the malicious nodes in a DataFlow system
- To determine the nature of their malicious action
- To evaluate the quality of output data

Du et al., **RunTest: Assuring Integrity of Dataflow Processing in Cloud Computing Infrastructures**, AsiaCCS 2010

Possible Approaches

- Re-do the computation
- Check memory footprint of code execution
- Majority voting
- Hardware-based attestation
- Run-time attestation

RunTest: Randomized Data Attestation

Idea

- For some data inputs, send it along multiple dataflow paths
- Record and match all intermediate results from the matching nodes in the paths
- Build an attestation graph using node agreement
- Over time, the graph shows which node misbehave (always or time-to-time)

Attack Model

- Data model:
  - Input deterministic DataFlow (i.e., same input to a function will always produce the same output)
  - Data processing is stateless (e.g., selection, filtering)
- Attacker:
  - Malicious or compromised cloud nodes
  - Can produce bad results always or some time
  - Can collude with other malicious nodes to provide same bad result

Attack Model (scenarios)

Parameters

- b_i = probability of providing bad result
- c_i = probability of providing the same bad result as another malicious node

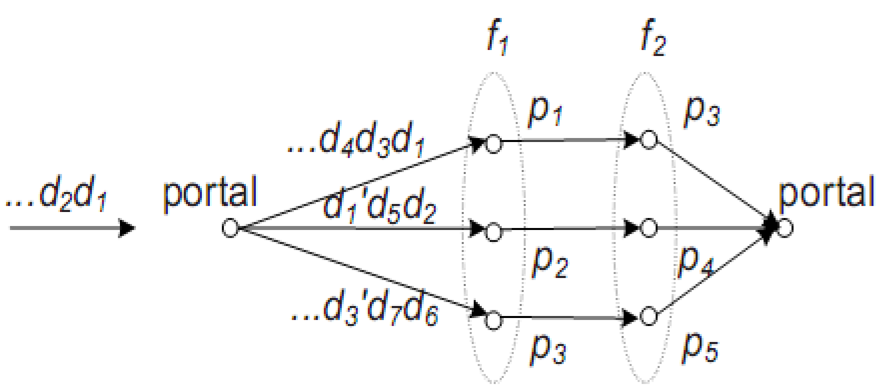
Attack scenarios

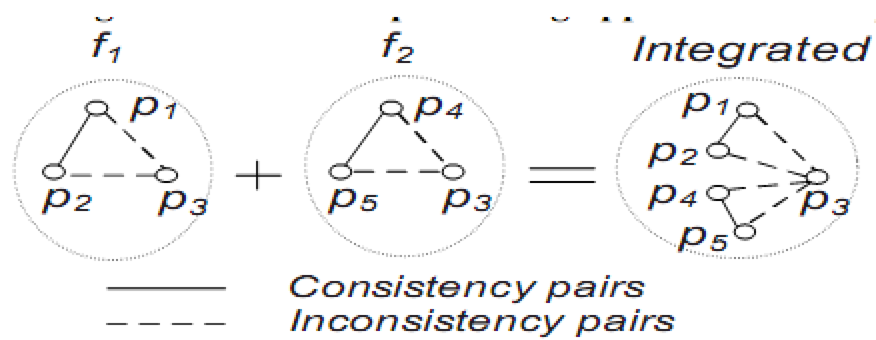
- **NCAM**: $b_i = 1, c_i = 0$
- **NCPM**: $0 < b_i < 1, c_i = 0$
- **FTFC**: $b_i = 1, c_i = 1$
- **PTFC**: $0 < b_i < 1, c_i = 0$
- **PTPC**: $0 < b_i < 1, 0 < c_i < 1$

Integrity Attestation Graph

Definition:

- Vertices: Nodes in the DataFlow paths
- Edges: Consistency relationships.
- Edge weight: fraction of consistent output of all outputs generated from same data items

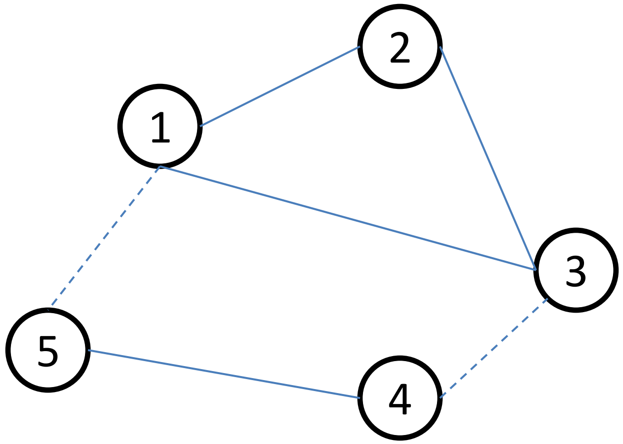




Consistency Clique

Complete subgraph of an attestation graph which has

- 2 or more nodes
- All nodes always agree with each other (i.e., all edge weights are 1)



How to find malicious nodes

Intuitions

- Honest nodes will always agree with each other to produce the same outputs, given the same data
- Number of malicious nodes is less than half of all nodes

Finding Consistency Cliques: BK Algorithm

Goal: find the **maximal clique** in the attestation graph

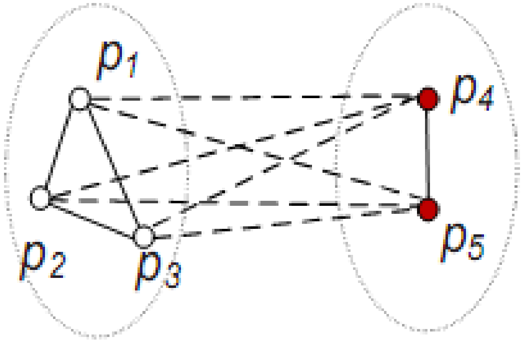
Technique:

Apply Bron-Kerbosch algorithm to find the maximal clique(s) (see better example at Wikipedia)

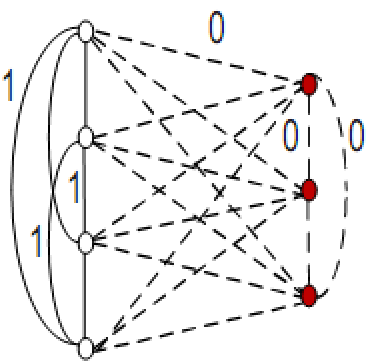
Any node not in a maximal clique of size k/2 is a malicious node

Note: BK algorithm is NP-Hard

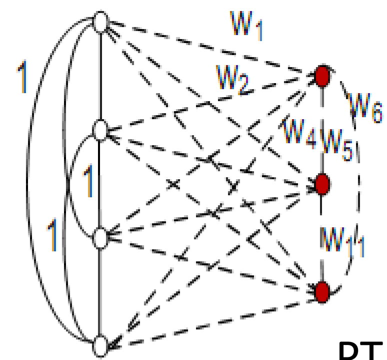
Authors proposed 2 optimizations to make it run quicker



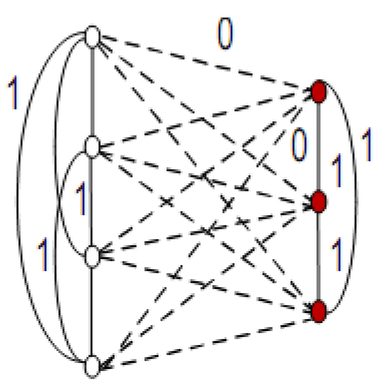
Identifying attack patterns



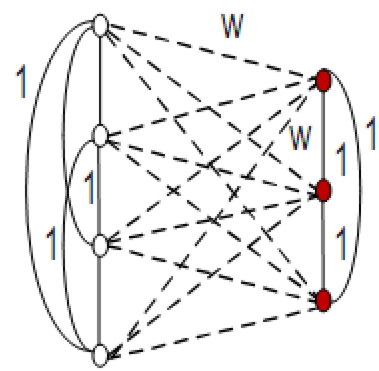
NCAM



PTFC/NCPM



FTFC



PTFC

Inferring data quality

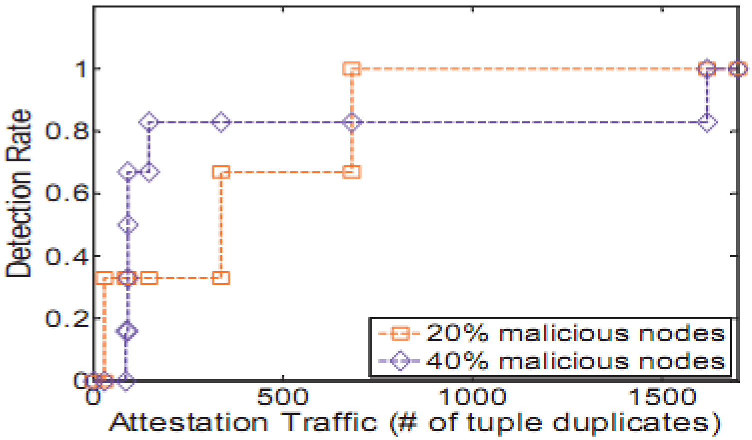
Quality = $1 - (c/n)$

- where
  - n = total number of unique data items
  - c = total number of duplicated data with inconsistent results

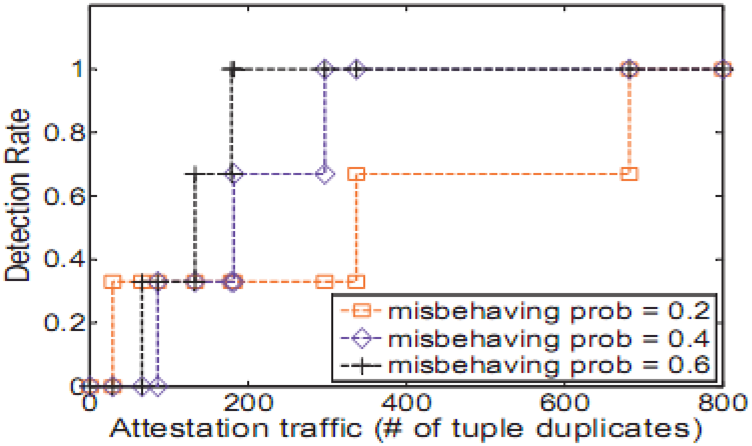
Evaluation

- Extended IBM System S
- Experiments:
  - Detection rate
  - Sensitivity to parameters
  - Comparison with majority voting

Evaluation



NCPM (b=0.2, c=0)



Different misbehavior probabilities

Discussion

- Threat model
- High cost of Bron-Kerbosch algorithm ($O(3^{n/3})$)
- Results are for building attestation graphs per function
- Scalability
- Experimental evaluation

Further Reading

TPM Reset Attack http://www.cs.dartmouth.edu/~pkilab/sparks/

Halderman et al., **Lest We Remember: Cold Boot Attacks on Encryption Keys, USENIX Security 08,** http://citp.princeton.edu/memory/