Secure Multi-Party Sorting and Applications

Kristján Valur Jónsson[1] Gunnar Kreitz[2] Misbah Uddin[2]

[1]Reykjavik University

[2]KTH—Royal Institute of Technology

ACNS 2011

KTH Computer Science and Communication

Collaborating on Intrusion Detection

- Traditionally, everyone runs their own IDS
- Can only observe and make decisions on local traffic
- We could gain by collaborating on IDS data
  - Detect low-volume attacks
  - Pre-emptively block attackers
  - Data suggests attacks are coordinated across targets [KKK05]

Today's Threats

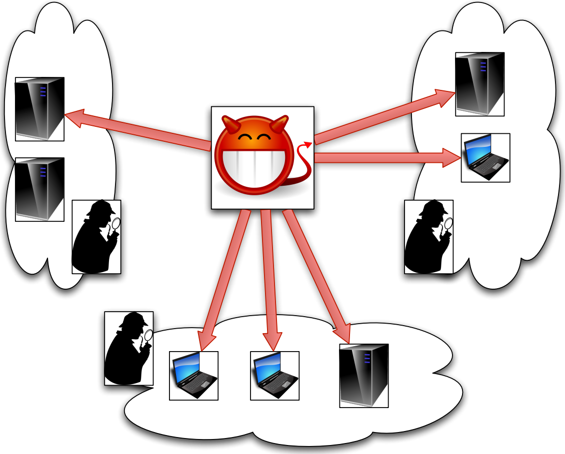
KTH Computer Science and Communication

IDS Data is Sensitive

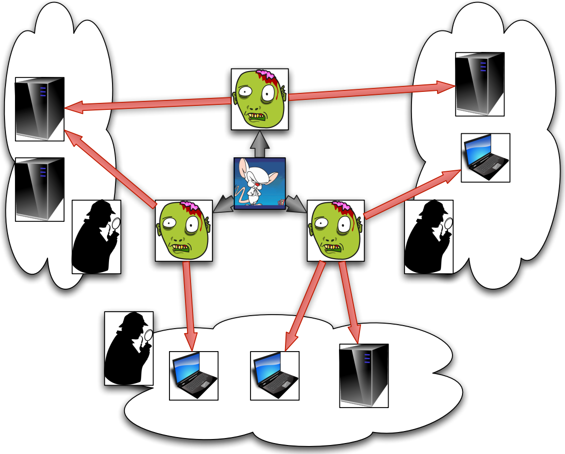
- Issues with publishing IDS data:
  - May reveal what we're looking for
  - Attacker can binary search for detection threshold
  - May reveal information on internal network
- So mostly, we just run our own IDSs (if at all)

KTH Computer Science and Communication

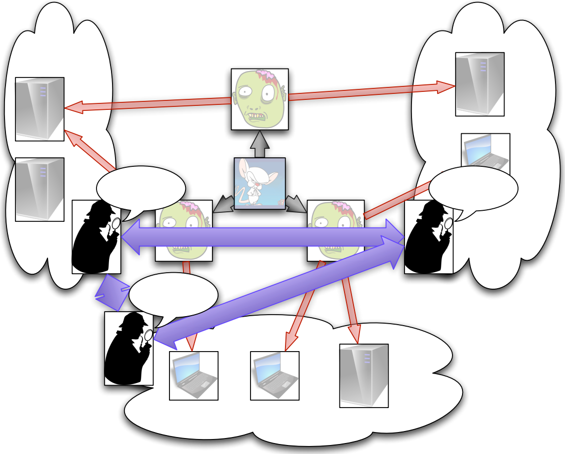
The Story in Pictures



The Story in Pictures



The Story in Pictures



Example IDS Application

- Detecting low-intensity, distributed scans
  - Each participating IDS counts number of SSH login attempts from each IP
  - Sum counter for each IP
- Only reveal top X or everyone above threshold to maintain privacy

Aggregating Suspsicions

IP	weight
1.1.1.1	5
2.2.2.2	7
3.3.3.3	7

Table: IDS 1

KTH Computer Science and Communication

Aggregating Suspsicions

IP	weight
1.1.1.1	5
2.2.2.2	7
3.3.3.3	7

Table: IDS 1

IP	weight
1.1.1.1	3
4.4.4.4	10
5.5.5.5	20

Table: IDS 2

KTH Computer Science and Communication

Aggregating Suspsicions

IP	weight
1.1.1.1	5
2.2.2.2	7
3.3.3.3	7

Table: IDS 1

IP	weight
1.1.1.1	3
4.4.4.4	10
5.5.5.5	20

Table: IDS 2

IP	weight
1.1.1.1	5
3.3.3.3	2
6.6.6.6	11

Table: IDS 3

Aggregating Suspsicions

IP	weight
1.1.1.1	5
2.2.2.2	7
3.3.3.3	7

Table: IDS 1

IP	weight
1.1.1.1	3
4.4.4.4	10
5.5.5.5	20

Table: IDS 2

IP	weight
1.1.1.1	5
3.3.3.3	2
6.6.6.6	11

Table: IDS 3

IP	weight
1.1.1.1	13
5.5.5.5	20
6.6.6.6	11

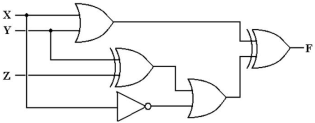
Table: Aggregate top-3

Our Contribution

- Writing down details of sorting networks for MPC sorting
- Implementing sorting — approaching practical speeds
- Aggregation algorithm based on sorting networks

KTH Computer Science and Communication

Secure Multi-Party Computation



- Implements the same functionality as a trusted third party.
- General protocols exist for circuit evaluation (and thus, computing arbitrary functions)
- Secure even in the presence of collusions (up to some limit, e.g. $\lfloor (n-1)/2 \rfloor$)
- Traditionally perceived as very slow

Secure?

What do we mean by security?

- In an ideal world, we have a trusted third party
- We want our protocols to be as secure as the ideal world
- Cheating parties must not:
  - learn more than they do in the ideal world
  - be able to do more than they can in the ideal world

General construction overview

- Represent function to compute as circuit (known to everyone)
- Each party secret shares her input to all others
- For each gate, evaluate gate under secret sharing
- When complete, open output values to all parties

KTH Computer Science and Communication

MPC Programming

- Implementing MPC has become fairly developer-friendly
- A number of platforms exist: FairplayMP, SEPIA, **Sharemind**, VIFF
- Sharemind uses a C-like language, SecreC
- Need to keep a few things in mind when implementing

Private Sorting

- Want sorting operations independent of values we sort
- Most algorithms do not have this property
  - Quick sort comparisons depend on pivot
  - Merge sort makes input-dependent comparisons in merge step
  - ...

KTH Computer Science and Communication

Private Sorting

- A sorting network is a circuit that sorts
- Built from Compare-exchange gates, which sort two elements
- AKS network uses only $\mathcal{O}(n \log n)$ gates, but with large constant
- Odd-even merge sort [Batcher68] uses $\mathcal{O}\left(n \log^2 n\right)$ gates

KTH Computer Science and Communication

Finding Zombies

KTH Computer Science and Communication

Related Work

- Applying Sorting Networks in MPC Setting:
  - Oblivious RAM [DMB-N11]
  - Secure data structures [Toft11]
- Randomized Shell-sort with $\mathcal{O}(n \log n)$ comparisons suitable for MPC [Goodrich10]

KTH Computer Science and Communication

Sorting on its own does not find Zombies

- Sorting is an important building block
- We need more for our motivating IDS application
  - Aggregate reports on same zombie from different IDSs
  - Top-X or thresholds
- Previous solution based on hashing [Many09]

KTH Computer Science and Communication

The Issue of Input Lengths

- When secret sharing, everyone learns how many values each party shares
- In general, we'd like that to be kept secret
- ...so each party can pad their input with `null` values
- How long should the padded inputs be?

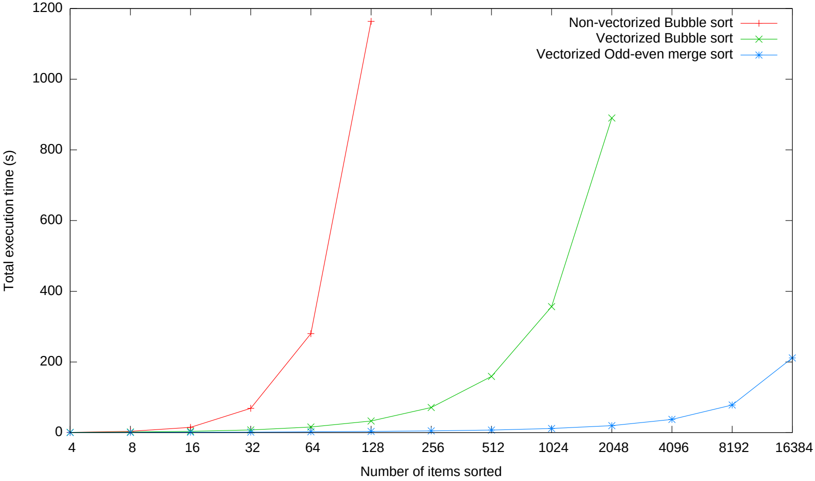
Choices for Input Length Padding

- Trade-off between privacy and speed
- No padding: reveal all input lengths
- Padding to max input length: reveal max input length (but not who has it)
- Padding to output length: reveals nothing

Implementing on Sharemind

- Straightforward implementation of odd-even merge sort and bubble sort
- Bubble sort with and without parallelization
- Ran on three-node cluster for various input lengths
- Passive, information-theoretic security against 1 out of 3 participants

Execution Time for Sorting



Aggregating Suspsicions

- Multiple sources reporting (IP, weight) pairs
- Want to sum up total reported weight for every IP from all reporters
- Reveal top X IPs, or everyone above threshold
- We refer to the general problem as Weighted Set Intersection

KTH Computer Science and Communication

MPC Algorithm for Weighted Set Intersection

- It is cheaper to aggregate when the input is sorted
- Three passes:
  - Sort on IP
  - Aggregate sorted values (similar to merge step of odd-even merge sort)
  - Sort on aggregate weight

KTH Computer Science and Communication

A run-through

IP	weight
1.1.1.1	5
2.2.2.2	7
3.3.3.3	7

Table: IDS 1

IP	weight
1.1.1.1	3
4.4.4.4	10
5.5.5.5	20

Table: IDS 2

IP	weight
1.1.1.1	5
3.3.3.3	2
6.6.6.6	11

Table: IDS 3

KTH Computer Science and Communication

A run-through

IP	weight
1.1.1.1	5
2.2.2.2	7
3.3.3.3	7
1.1.1.1	3
4.4.4.4	10
5.5.5.5	20
1.1.1.1	5
3.3.3.3	2
6.6.6.6	11

Table: Concatenate all inputs

KTH Computer Science and Communication

A run-through

IP	weight
1.1.1.1	5
1.1.1.1	3
1.1.1.1	5
2.2.2.2	7
3.3.3.3	7
3.3.3.3	2
4.4.4.4	10
5.5.5.5	20
6.6.6.6	11

Table: Sort by IP

KTH Computer Science and Communication

A run-through

IP	weight
1.1.1.1	13
(null)	0
(null)	0
2.2.2.2	7
3.3.3.3	9
(null)	0
4.4.4.4	10
5.5.5.5	20
6.6.6.6	11

Table: Aggregate weights

KTH Computer Science and Communication

A run-through

IP	weight
5.5.5.5	20
1.1.1.1	13
6.6.6.6	11
4.4.4.4	10
3.3.3.3	9
2.2.2.2	7
(null)	0
(null)	0
(null)	0

Table: Sort by weight (descending)

KTH Computer Science and Communication

A run-through

IP	weight
5.5.5.5	20
1.1.1.1	13
6.6.6.6	11

Table: Open top 3

KTH Computer Science and Communication

Summary

- Details of using sorting networks in an MPC context
- Close to practical, can sort 2^{14} values in 3.5 minutes
- Aggregation algorithm based on sorting network

KTH Computer Science and Communication