Practical Private Aggregation

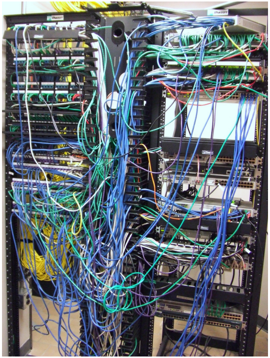
Gunnar Kreitz Mads Dam Douglas Wikström

KTH – Royal Institute of Technology
{gkreitz,mfd,dog}@kth.se

NordSec 2010

KTH Computer Science and Communication

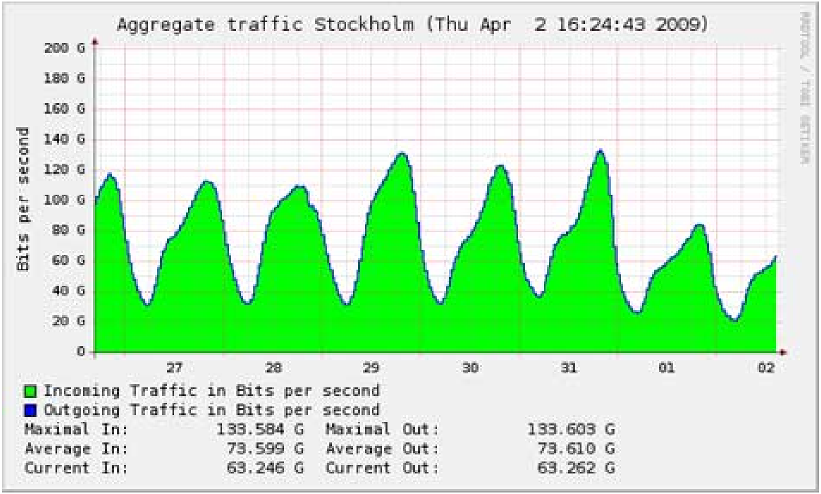
Monitoring Networks



- Networks of today are complex
- Need monitoring to
  - Detect and prevent attacks
  - Detect and prevent problems
  - Troubleshoot
  - ...

KTH Computer Science and Communication

Monitoring to Troubleshoot



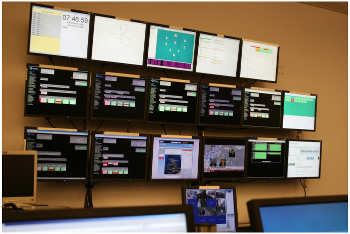
KTH Computer Science and Communication

Private Aggregation

We don't want to share:

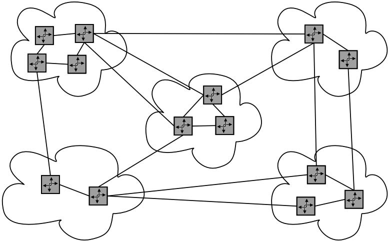
- traffic information
- intrusion detection alerts
- real-time error condition status
- capacity and consumption in a power grid

Monitoring Aggregates



- Summation to compute:
  - Total number of network flows
  - Average packet loss (two summations)
- Disjunction (Boolean Or) to compute:
  - Any failure alerts?
- Maximum to compute:
  - Highest load

Motivating Scenario



- Monitoring real-time network data
- Measurements by a few hundred routers
- Aggregates are shared between five ISP:s
- No ISP wants the others to have access to its data
- ISP:s trust each other to not lie about measurements

KTH Computer Science and Communication

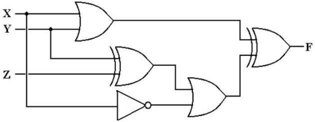
A Trusted Third Party

- Is there someone we all trust?
- Can send measurements to the Trusted Third Party
- She performs computation and tells everyone result
- Given a Trusted Third Party, problem is easy

KTH Computer Science and Communication

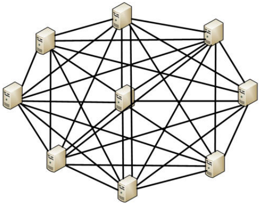
Sometimes There is no Trusted Third Party

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. $\lfloor n/2 \rfloor$)
- But ...

Multi-Party Computation Issues



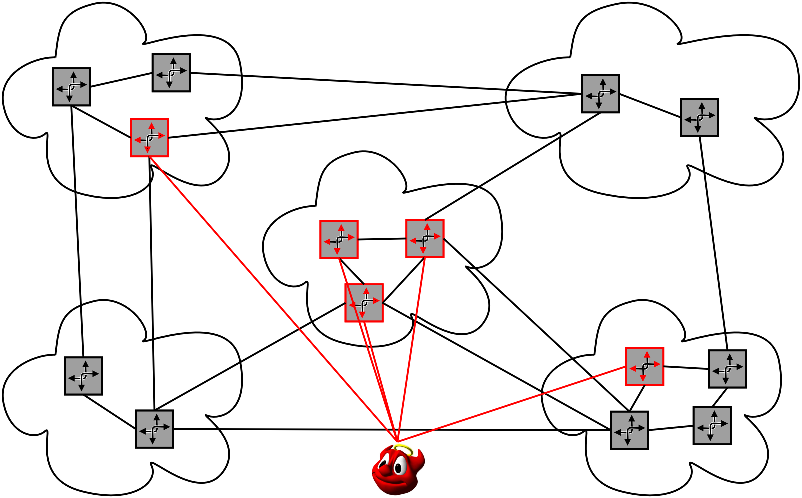
- General purpose protocols are often unacceptably slow
- Most MPC protocols assume they're run on full mesh networks
- . . . with private channels between each pair of parties
- If we're monitoring routers, we can't assume routing works
  - So we only want to use direct links

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

The Adversary



How Powerful is Our Adversary?

- Two main models of adversary's evilness:
  - Passive (*Honest-but-curious*): follows protocol but tries to deduce more information
  - Active (*Byzantine*): arbitrary deviations from protocol
- In this talk, all adversaries are passive

Image credit: OpenBSD `http://www.openbsd.org/art2.html`

How Powerful is Our Adversary?

- Two main models of adversary's power:
  - Computational Security: Probabilistic polynomial time
  - Information-Theoretic Security: Unlimited computation time
- In this talk, we consider both notions

Our Contribution

- Study **efficient** multi-party computation in **partial-mesh networks**
- Three protocols:
  - Information-theoretically secure protocol for summation
  - Computationally secure protocol for disjunction
  - Information-theoretically secure protocol for disjunction
- Computing maximum can be done by repeated disjunction

KTH Computer Science and Communication

Our Goal

Photo by Jon McGovern `http://www.flickr.com/photos/jonmcgovern/2673202881/`, CC BY NC 2.0
Photo by Toban Black `http://www.flickr.com/photos/tobanblack/3169340252/`, CC BY NC 2.0

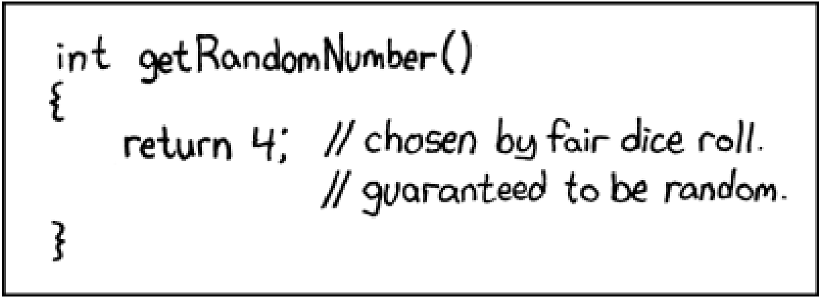
Private Summation

$$\sum$$

- Protocol for summation modulo 2^{64} (more generally, in an Abelian group)
- Extremely efficient and simple protocol
- Adds an overhead of a single round with two messages per link compared to non-private summation
- Private if adversary does not corrupt a separator of the network
- Similar to [Chaum88] and [ChorKushilevitz93]

KTH Computer Science and Communication

The Most Expensive Operation

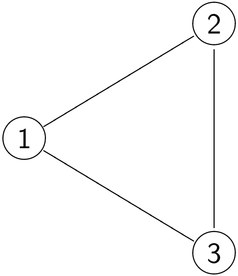


KTH Computer Science and Communication

Private Summation (cont'd)

- The protocol does one round of input randomization (*blinding*)
- Then, any (non-private) summation protocol is run on the blinded inputs
- The blinding preserves the sum of the inputs
- Information-theoretically secure

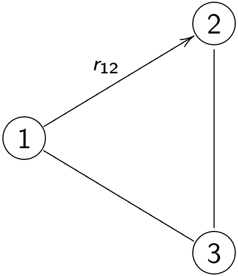
Summation Protocol by Example



$$x_1' = x_1$$
$$x_2' = x_2$$
$$x_3' = x_3$$

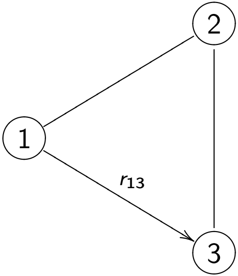
KTH Computer Science and Communication

Summation Protocol by Example



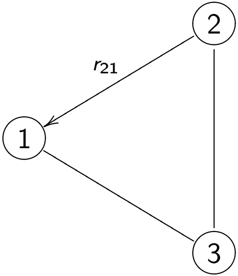
$$x_1' = x_1 - r_{12}$$
$$x_2' = x_2 + r_{12}$$
$$x_3' = x_3$$

Summation Protocol by Example



$$
\begin{aligned}
x_1' &= x_1 - r_{12} - r_{13} \\
x_2' &= x_2 + r_{12} \\
x_3' &= x_3 \qquad\quad + r_{13}
\end{aligned}
$$

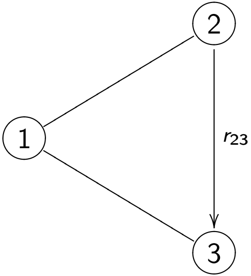
Summation Protocol by Example



$$\begin{aligned} x_1' &= x_1 - r_{12} - r_{13} + r_{21} \\ x_2' &= x_2 + r_{12} \phantom{- r_{13}} - r_{21} \\ x_3' &= x_3 \phantom{+ r_{12}} + r_{13} \end{aligned}$$

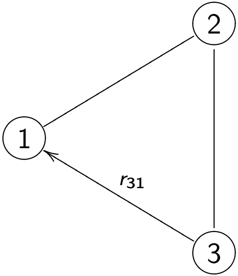
KTH Computer Science and Communication

Summation Protocol by Example



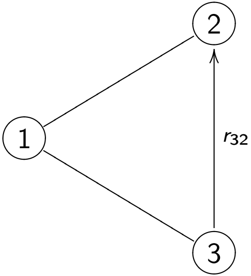
$$
\begin{aligned}
x_1' &= x_1 - r_{12} - r_{13} + r_{21} \\
x_2' &= x_2 + r_{12} \phantom{- r_{13}} - r_{21} - r_{23} \\
x_3' &= x_3 \phantom{+ r_{12}} + r_{13} \phantom{- r_{21}} + r_{23}
\end{aligned}
$$

Summation Protocol by Example



$$
\begin{aligned}
x_1' &= x_1 - r_{12} - r_{13} + r_{21} + r_{31} \\
x_2' &= x_2 + r_{12} - r_{21} - r_{23} \\
x_3' &= x_3 + r_{13} + r_{23} - r_{31}
\end{aligned}
$$

Summation Protocol by Example



$$
\begin{aligned}
x_1' &= x_1 - r_{12} - r_{13} + r_{21} \phantom{- r_{23}} + r_{31} \\
x_2' &= x_2 + r_{12} \phantom{- r_{13}} - r_{21} - r_{23} \phantom{- r_{31}} + r_{32} \\
x_3' &= x_3 \phantom{+ r_{12}} + r_{13} \phantom{- r_{21}} + r_{23} - r_{31} - r_{32}
\end{aligned}
$$

Private Summation Protocol

- Each party P_i with input x_i proceeds as follows:
  1. Send random $r_{i,j}$ to each neighbor P_j
  2. Wait for $r_{j,i}$ from each neighbor P_j
  3. Compute

$$x_i' = x_i + \sum_{P_j \text{neighbor}} r_{j,i} - \sum_{P_j \text{neighbor}} r_{i,j}$$

- We could now publish x_i' and still remain private!

KTH Computer Science and Communication

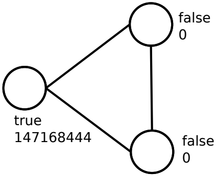
Private Disjunction

$$\bigvee$$

- Computing disjunction (Boolean or) privately is harder
- Our protocols build on summation
- Have negligible risk of giving incorrect output
- We begin with a really simple construction that doesn't work

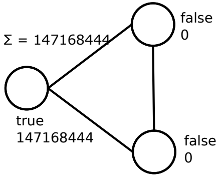
KTH Computer Science and Communication

An Idea That Doesn't Work



- We'll use private summation, with the idea that the sum is non-zero only if at least one party has input `true`
- A party with input `false` uses input 0 for the summation
- A party with input `true` picks random input for the summation
- Sum is random if one or more parties has input `true`, but ...

An Idea That Doesn't Work



- We'll use private summation, with the idea that the sum is non-zero only if at least one party has input `true`
- A party with input `false` uses input 0 for the summation
- A party with input `true` picks random input for the summation
- Sum is random if one or more parties has input `true`, but ...
- a party seeing its own blinded input as output knows (w.h.p.) that it was the only one with input `true`

KTH Computer Science and Communication

Two Protocols for Disjunction

- We propose two mechanisms for patching the idea
- Computationally secure:
  - A party will not know its own blinded input
  - Uses public-key cryptography
- Information-theoretically secure:
  - The sum is never revealed
  - Requires certain structure of the network

KTH Computer Science and Communication

Summary

- Efficient, private, protocols for basic aggregation operations
- Can run in arbitrary networks (most Multi-party Computation protocols assume full mesh)
- Enables aggregation and network monitoring cooperation between competitors

KTH Computer Science and Communication

Future Work

- Security against an active adversary
- Finding efficient protocols for general networks for other problems
- Application in monitoring systems

Maximum

- Given a protocol for disjunction, we can construct a protocol for selecting the maximal element
- Compute disjunction of most significant bit
- Parties who learn their input is less than the maximum proceed with input `false` in further rounds
- Compute disjunction of next-most significant bit
- . . . and so on

Maximum by Repeated Disjunction

Round 0

x_1	1	1	0	1	...
x_2	0	1	1	1	...
x_3	1	1	0	1	...
x_4	1	0	1	0	...
$\bigvee$					

KTH Computer Science and Communication

Maximum by Repeated Disjunction

Round 1

x_1	1	1	0	1	...
x_2	0	1	1	1	...
x_3	1	1	0	1	...
x_4	1	0	1	0	...
$\bigvee$	1				

Maximum by Repeated Disjunction

Round 1

x_1	1	1	0	1	...
x_2	0	0	0	0	...
x_3	1	1	0	1	...
x_4	1	0	1	0	...
$\vee$	1				

KTH Computer Science and Communication

Maximum by Repeated Disjunction

Round 2

x_1	1	1	0	1	...
x_2	0	0	0	0	...
x_3	1	1	0	1	...
x_4	1	0	1	0	...
$\vee$	1	1			

KTH Computer Science and Communication

Maximum by Repeated Disjunction

Round 2

x_1	1	1	0	1	...
x_2	0	0	0	0	...
x_3	1	1	0	1	...
x_4	1	0	0	0	...
$\bigvee$	1	1			

KTH Computer Science and Communication

Maximum by Repeated Disjunction

Round 3

x_1	1	1	0	1	...
x_2	0	0	0	0	...
x_3	1	1	0	1	...
x_4	1	0	0	0	...
$\vee$	1	1	0		

Maximum by Repeated Disjunction

Round 4

x_1	1	1	0	1	...
x_2	0	0	0	0	...
x_3	1	1	0	1	...
x_4	1	0	0	0	...
$\bigvee$	1	1	0	1	

KTH Computer Science and Communication