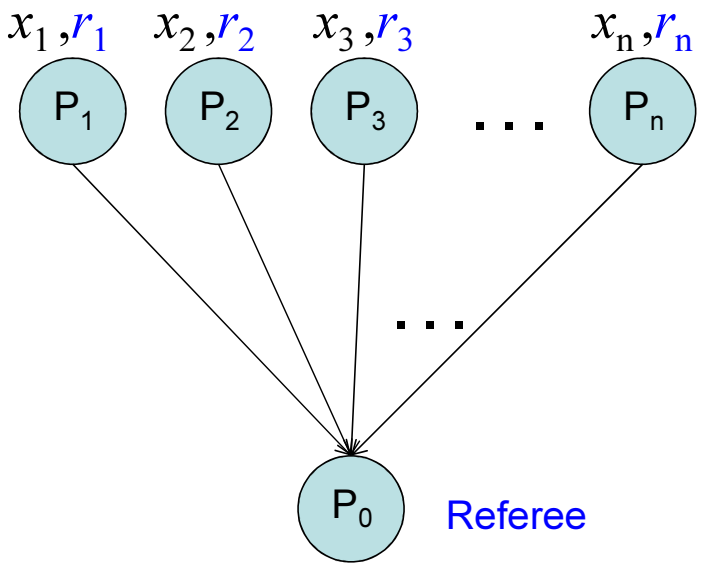
Distributed Obfuscation and Non-Interactive Secure Multiparty Computation

Amos Beimel (BGU)
Ariel Gabizon (Technion)
Yuval Ishai (Technion)
Eyal Kushilevitz (Technion)
Sigurd T. Meldgaard (Aarhus)
Anat Paskin-Cherniavsky (UCLA)

Non-Interactive MPC – Basic Setting

- Goal: non-interactively compute $f(x_1,\ldots,x_n)$.
- Setup: $(r_1,\ldots,r_n) \in_R R$.
- Generic attack: $\{P_0\} \cup T$ can learn not only $f(x)$ but also $f(x')$, for every x' obtained from x by T modifying their inputs.
- T-robustness: that is all $\{P_0\} \cup T$ can learn.
- *t*-robustness, full-robustness
- Focus: Semi-honest, info-theoretic security



Example: 0-robust OR

Fix $p>n$. All arithmetic is over GF[p].

Input: P_i is given $x_i \in \{0,1\}$.
Output: P_0 gets $\vee x_i$.

Preprocessing:
– Pick random $r \in_R$ GF[p]\{0} and random z_i's such that $\Sigma z_i = 0$
– Each P_i is given r, z_i.

Protocol:
1. Each P_i sends $m_i = r \cdot x_i + z_i$ to P_0.
2. P_0 computes $\Sigma m_i = r \cdot \Sigma x_i + \Sigma z_i = r \cdot \Sigma x_i$ which determines the OR.

0-robust but not 1-robust

Example: Fully-robust OR

Input: P_i is given $x_i \in \{0,1\}$.
Output: P_0 gets $\vee x_i$.

Preprocessing:
– Pick random vectors $r_{1,0}, r_{1,1}, \ldots, r_{n,0}, r_{n,1} \in_R (F_2)^{2n}$ with no linear dependency except that $\Sigma r_{i,0} = 0$.
– Each P_i is given $r_{i,0}, r_{i,1}$.

Protocol:
1. Each P_i sends $m_i = r_{i,xi}$ to P_0.
2. P_0 outputs 0 iff messages are linearly dependent.

Fully-robust: by choice of randomness

Remarks and Extensions

Meaningful for a single f, or a class F (and f may be hidden).

Multi-output MPC: each player sends a message to each player. Obtained by n invocations of basic protocol.

Malicious adversary: define ideal model (to handle rushing); deal via standard methods (unconditional one-time MAC).

Correlated Randomness: useful setup; CRS and, e.g., [Bea95,BDOZ11,IKMOP13]

An Abstract Formulation

NIMPC for function f consists of:

1. joint probability distribution $(r_1,....,r_n) \in_R R$
2. n encoding functions $m_i = \mathrm{Enc}_i(x_i,r_i)$

Correctness: from $\{m_i\}_{i\in[n]}$ can efficiently recover $f(x_1,\ldots,x_n)$.

T-robustness: $\{m_i\}_{i\notin T}, \{r_i\}_{i\in T}$ gives same information on $\{x_i\}_{i\notin T}$ as oracle access to $f_{\{xi\,:\,i\notin T\}}(\{x_i\}_{i\in T})$

Boolean case: just $m_{i,0}$, $m_{i,1}$ for all $i\in[n]$ (where $m_{i,b}=\mathrm{Enc}_i(b,r_i)$).

Motivation

0-robustness ≡ Private Simultaneous Messages (PSM) model [FKN94]

Lots of work on reducing interaction, and eliminating "simultaneous interaction" in secure MPC [HLP11].

Our is completely non-interactive and uses general correlated randomness and gives IT-security.

Best-possible security for Non-Interactive secure computation

Relation with Obfuscation

Obfuscation: randomized mapping from a circuit/program $c \in C$ to some equivalent c' that hides c except I-O relation

NIMPC $\Rightarrow$ Obfuscation:

Use fully robust NIMPC for the universal function for C.

$$U_C(c_1,..., c_m, x_1,..., x_n) = c(x)$$

Get pairs $(a_{1,0}, a_{1,1}),..., (a_{m,0}, a_{m,1})$ and $(b_{1,0}, b_{1,1}),..., (b_{n,0}, b_{n,1})$.

Output $a_{1,c1},...,a_{m,cm}$, and $(b_{1,0}, b_{1,1}),..., (b_{n,0}, b_{n,1})$.

Idea: functionality via correctness; hiding via T-robustness, where T = x-players.

Relation with Obfuscation (cont.)

Differences: NIMPC non-trivial also for single f, for learnable functions, and in information-theoretic setting.

Negative results for obfuscation $\Rightarrow$ negative results for NIMPC.

e.g., no efficient, fully-robust, IT-NIMPC for NC^1 unless PH collapses [GR07].

Relation with Multi-Input Functional Encryption

Functional Encryption (FE): E(x) such that for $f \in F$ can produce SK_f that allows deciphering $f(x)$ but nothing else.

Multi-Input Functional Encryption (MIFE): like FE but f gets multiple ciphertexts.

MIFE stronger than NIMPC: reusable

Recent ind-obfuscation candidate [GGHRSW13] + MIFE from ind-obfuscation [GGGJKLSSZ14] $\Rightarrow$ Computationally secure, fully robust NIMPC under strong assumptions.

In contrast allowing one-time use + correlated randomness meaningful in IT case.

Our Results

Fully robust protocol for all functions, w/communication poly in size of input domain.

Fully robust, efficient protocol for group products
$f_G(x_1,..., x_n) = x_1 \cdot x_2 \cdot ... \cdot x_n$.

t-robust protocol for symmetric functions, w/communication $n^{O(t)}$.

Negative: known PSM and garbling protocols, not even 1-robust.

Some Protocols

Example: SUM (over Abelian group G)

Input: P_i is given $x_i \in G$.
Output: P_0 gets Σx_i.

Preprocessing:
– Pick random $r_1, \ldots, r_{n-1}, r_n \in_R G$ such that $\Sigma r_i = 0$.
– P_i is given r_i.

Protocol:
1. Each P_i computes $m_i = x_i + r_i$ and sends to P_0.
2. P_0 computes $\Sigma m_i = \Sigma x_i + \Sigma r_i = \Sigma x_i$.

Full robustness: by choice of r_i's.

Example: Group Product (possibly non-abelian G)

Input: P_i is given $x_i \in G$.
Output: P_0 gets Πx_i.

Preprocessing:
– Pick random $r_1, \ldots, r_{n-1} \in_R G$, and set $r_0 = r_n = 1$.
– P_i is given, for each $g \in G$, the value $m_{i,g} = (r_{i-1})^{-1} \cdot g \cdot r_i$.

Protocol:
1. Each P_i sends $m_{i,xi}$ to P_0.
2. P_0 computes $\Pi m_{i,xi} = \Pi (r_{i-1})^{-1} \cdot x_i \cdot r_i = \Pi x_i$.

Full robustness: proof in non-Abelian case is involved.

Fully-robust Protocol for all functions

Step 1: Fully robust NIMPC for

$$H = \{h_a \mid h_a(x)=1 \text{ iff } x=a\} \cup \{z(x)=0 \text{ for all } x\}$$

(also hides h itself).

Input: P_i is given $x_i \in \{0,1\}$.
Output: P_0 gets $h(x)$ (for $h \in H$).

Preprocessing:

– Pick random vectors $r_{1,0},r_{1,1},\ldots, r_{n,0},r_{n,1} \in_R (F_2)^{2n}$. If $h=z$ then no linear dependencies; if $h=h_a$ no dependencies except that $\Sigma r_{i,ai} = 0$.

– P_i is given $r_{i,0},r_{i,1}$.

Protocol:

1. Each P_i sends $m_i = r_{i,xi}$ to P_0.
2. P_0 outputs 1 iff messages are linearly dependent.

Note: if $\{x_i\}_{i \notin T}$ consistent with a then T learns a_T.

Fully-robust Protocol for all functions (Step 2)

Idea: for arbitrary $f: X^n \to \{0,1\}$, write $f = \Sigma_{a: f(a)=1} h_a$, and use above protocol for each h_a.

Problems:

1. Reveals number of 1's of f.
2. When $f(x)=1$ not supposed to learn a for which $h_a(x)=1$.

Solutions:

1. Let $h'_a = h_a$ if $f(a)=1$ and $h'_a = z$ otherwise. Write $f = \Sigma_a h'_a$.
2. Randomly permute the order.

Symmetric Functions

$f:\{0,1\}^n \to \{0,1\}$ is symmetric if $f(x_1,..,x_n)=h(\Sigma x_i)$.

We show t-robust protocol w/complexity $n^{O(t)}$.
This talk: t=1.

High level view:

1. For $G=Z_{n+1}$ protocol for $f':G^n \to \{0,1\}$ s.t. $f'(x_1,..,x_n)=h(\Sigma x_i)$.
2. Same as 1, except that input of a designated P_j forced to $\{0,1\}$.
3. Additively sec... Σx_i. robustness allows to learn
   For $j\in[n]$, compute
   restricted to {

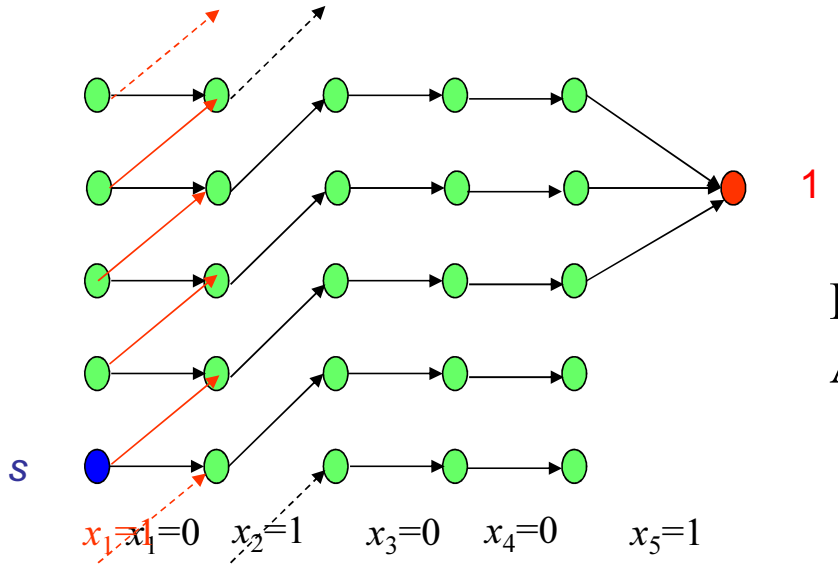
Idea: T=$\{P_0, P_j\}$ (
$h_i(s), h_i(s+1),..., h$

Idea: Run 2 protocols for $f_0(x_2,..,x_n)=f(0,x_2,..,x_n)$ and $f_1(x_2,..,x_n)=f(1,x_2,..,x_n)$
Outputs randomly permuted and masked.
Let P_1 choose the correct execution and un-mask the relevant output.

Details for Step 1

Starting point -- Branching-program based PSM of [IK97]:

Let $M_{i,b}$ corresponds to edges of BP for f labeled by "x_i=b".
Each P_i gets $m_{i,0}=L\cdot M_{i,0}\cdot R + Z_i$, $m_{i,1}=L\cdot M_{i,1}\cdot R + Z_i$, for random Z_i's s.t. $\Sigma Z_i=0$, and random non-singular L,R.
P_0 gets messages $m_{i,xi}$, computes $\Sigma\, m_{i,xi} = L\cdot M_x\cdot R$, where $M_x = \Sigma\, M_{i,xi}$, from which it learns rank$(M_x)=f(x)$.



BP for MAJ_5 function.
Assignment 01001

Details for Step 1 (cont.)

[IK97] not 1-robust:

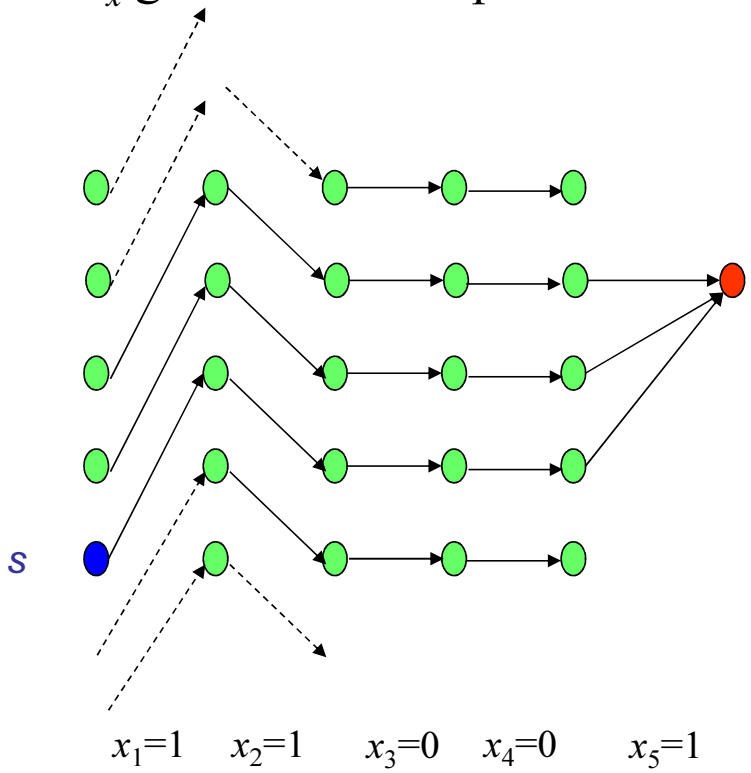
T=$\{P_0,P_i\}$ learn (information on) L,R from P_i's messages $m_{i,0}=L\cdot M_{i,0}\cdot R + Z_i$, $m_{i,1}=L\cdot M_{i,1}\cdot R + Z_i$, by computing

$m_{i,0}-m_{i,1}=L\cdot(M_{i,0}-M_{i,1})\cdot R$.

Using L,R, player P_0 recovers M_x from $L\cdot M_x\cdot R$, which gives all matrices $\{M_{j,xj}\}$ (i.e., all inputs).

Details for Step 1 (cont.)

Idea: [IK97] on B'=randomization of B, via level-by-level shift.
Connect node a at level i to $a+x_i+r_i$ (rather than $a+x_i$), with r_i's random s.t. $\Sigma r_i=0$.
$\Rightarrow M_x$ gives a random path in B' that reveals only shifted truth table.



Randomized BP for MAJ_5.
r_1=1, r_2=-2, r_1=0, r_4=0, r_5=1
(Σr_i=0).

Summary

We introduce the notion of robust NIMPC, with various connections to other primitives.

We present NIMPC protocols for various classes of functions.

Open: more protocols, improved complexity,….

Thank you!