Multiparty Computation from Somewhat Homomorphic Encryption

Ivan Damgård[1] Valerio Pastro[1] Nigel Smart[2] Sarah Zakarias[1]

[1]Aarhus University

[2]Bristol University

November 9, 2011

Multiparty Computation

The problem

- n parties: $P_1, \ldots, P_n$
- for all i P_i has private input x_i
- a function $f : (x_1, \ldots, x_n) \mapsto (y_1, \ldots, y_n)$

Outcome

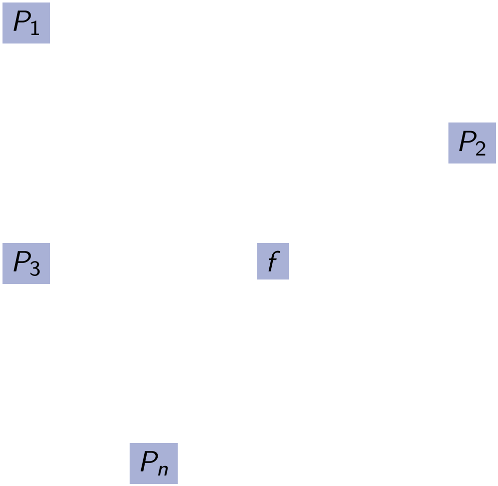
- for all i y_i to be delivered to P_i
- no more info revealed

Applications – Examples

- The millionaire problem [Yao82]:
  $n = 2$,
  $x_i = P_i$'s income,
  $f(x_1, x_2) = (b, b)$, where $x_b = \max\{x_1, x_2\}$
- Keywords search
- Set intersection
- Auctions (e.g. the sugar beet auction, Denmark 2008)
- Dominik's dating problem
- ...

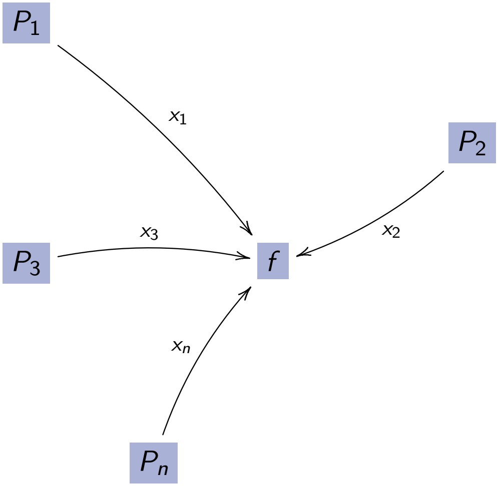
Multiparty Computation – Ideal

The ideal solution: A trusted party!



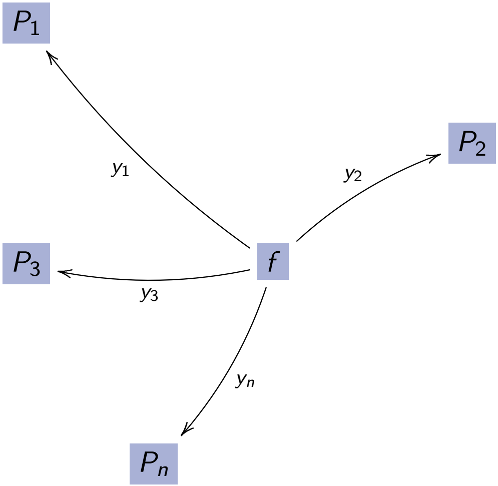
Multiparty Computation – Ideal

Players send their inputs..



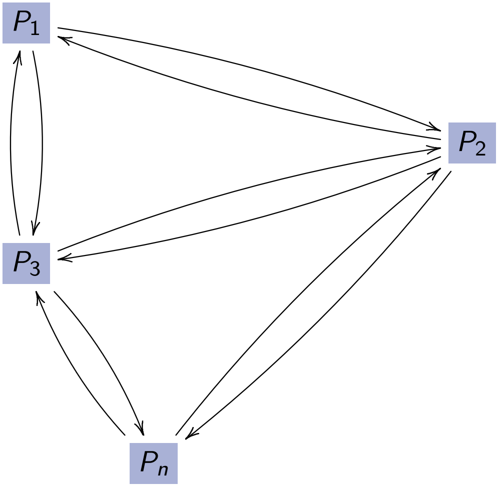
Multiparty Computation – Ideal

..and get their result.



Multiparty Computation – Real

The trusted party: useful?



Multiparty Computation – Dealing with Players

Ideal scenario $\Rightarrow$ concrete protocol?

The setup – Real world

- n parties: $P_1, \ldots, P_n$
- for all i P_i has private input x_i
- f replaced by interaction between players and local computation

Outcome

- for all i y_i to be delivered to P_i
- no more info revealed

Multiparty Computation – Those Annoying Players

Some players may cheat (to get more info)!
Secure Protocol? Real world indistinguishable from Ideal world.
Adversarial entity who controls dishonest players.

Adversarial Behavior

Dishonest players *follow* the protocol:	Passive Adversary
Dishonest players *deviate* from the protocol:	Active Adversary

Security Requirements

$$View(P_i)_{Ideal} \equiv_{Stat/Comp} View(P_i)_{Real}$$

in presence of passive/active Adversary

Our Target

Construction of a protocol for:

- Secure Multiparty Computation
- Active Adversary
- Dishonest Majority (P_i honest, for all $j \neq i$, P_j controlled by the Adversary)

Modern Approaches – High Level

2-phases approach

Preprocessing	$\Rightarrow$	Online
Players generate some shared randomness, independently of f (public key crypto required).	$\Rightarrow$	Previous data used to evaluate of f (seen as an arithmetic circuit).

Online phase: very fast – no PKE!

Modern Approaches – High Level

Fully Homomorphic Encryption [Gen09]

Use an encryption scheme $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ such that for *any* arithmetic circuit C:

$$\mathsf{Dec}_{sk}(C'(\mathsf{Enc}_{pk}(m_1), \dots, \mathsf{Enc}_{pk}(m_n))) = C(m_1, \dots, m_n),$$

where C' acts as C on encrypted data.

If so, $\mathsf{Enc}_{pk_i}(y_i) = \mathsf{Enc}_{pk_i}(f_i(x_1, \dots, x_n)) = f_i(\mathsf{Enc}_{pk_i}(x_1), \dots, \mathsf{Enc}_{pk_i}(x_n))$.
Drawback: FHE is impractical (nowadays)!

Our Approach

Take the best of the two previous methods!
2-phases approach with Somewhat Homomorphic Encryption.

Somewhat Homomorphic Encryption Scheme

An encryption scheme $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ such that:

$$\mathsf{Dec}_{sk}(C'(\mathsf{Enc}_{pk}(m_1), \ldots, \mathsf{Enc}_{pk}(m_n))) = C(m_1, \ldots, m_n),$$

where C is an arithmetic circuit in a specific set S.

In our case: $S =$ circuits of mult depth one.
Further requirement: a *distributed decryption*.

Our Approach – Showing off

1. (much) More practical than the FHE-approach.
2. Preprocessing phase: similar to [BDOZ11], but less protocols needed.
3. Online phase: Better scalability ($O(n)$ vs $O(n^2)$) mults to compute a secure mult)

Note: msgs in $(\mathbb{F}_{p^k})^s$: a vector space of dim s over a field of size p^k..
..but for simplicity we set $s = 1$ (more details later!)

Online Phase – Digression

Suppose $x, y \in \mathbb{F}_{p^k}$. We write $[x], [y]$ if x, y are additively secret shared among the players:

$$x = \sum_{i=1}^{n} x_i, \qquad y = \sum_{i=1}^{n} y_i, \qquad P_i \text{ has } x_i, y_i.$$

Easy to compute $[x + y]$:
P_i locally computes $a_i = x_i + y_i$.

$$\sum_{i=1}^{n} a_i = \sum_{i=1}^{n} (x_i + y_i) = x + y.$$

Addition: easy.

Online Phase

Multiplication? Not as easy as addition!
Want to compute $[x \cdot y]$ from $[x], [y]$.
Using [Bea91]: easy if players have a "multiplicative triple" $[a], [b], [a \cdot b]$:

1. Compute $[x + a], [y + b]$ (easy).
2. Reconstruct $\varepsilon = x + a, \delta = y + b$
3. Compute
$$[z] = [a \cdot b] - \varepsilon \cdot [b] - \delta \cdot [a] + \varepsilon \cdot \delta.$$

$[z]$ is a secret sharing of $x \cdot y$:

$$\begin{aligned} z &= a \cdot b - \varepsilon \cdot b - \delta \cdot a + \varepsilon \cdot \delta \\ &= a \cdot b - (x + a) \cdot b - (y + b) \cdot a + (x + a) \cdot (y + b) \\ &= xy \end{aligned}$$

Online Phase

Security? MACs!

Message Authentication Codes (à la [BDOZ11])

$$\mathrm{MAC}^j(x_i) \quad = \quad \alpha_i^j \quad \cdot \quad x_i \quad + \quad \beta_{x,i}^j$$

$$P_i \qquad\qquad\qquad\qquad P_j$$

We require P_i to have: $x_i, \quad \left\{\mathrm{MAC}^j(x_i)\right\}_{j=1, j\neq i}^n, \quad \left\{\left(\alpha_j^i, \beta_{x,j}^i\right)\right\}_{j=1,j\neq i}^n$

Above situation: $[x]$ ("bracket notation").

Notice: each player has $O(n)$ MACs, $O(n)$ keys for each secret value.

Result: for each secret value $O(n^2)$ keys and MACs to insure security.

Summary

$$\left.\begin{array}{c}\text{Multiplicative Triples}\\ \text{Additive Secret Sharing}\\ \text{MACs}\end{array}\right\} \Longrightarrow \text{Secure MPC.}$$

How to obtain multiplicative triples? Preprocessing!

Preprocessing Phase

Target: generate $[a], [b], [c]$ with $c = ab$.

Setup

1. Generate keys for the SHE scheme
2. Generate the α_j^i's (first half of the MACs' keys)
3. Broadcast $\mathsf{Enc}_{pk}(\alpha_j^i)$
4. Invoke a Zero-Knowledge Proof of Knowledge (Π_{ZKPoPK}) on $(\mathsf{Enc}_{pk}(\alpha_j^i), \alpha_j^i)$

Setup: independent from values to generate.

Preprocessing Phase

Triples

Getting $a \cdot b + r$:

1. P_i generates uniform values $a_i, b_i, r_i \in \mathbb{F}_{p^k}$
2. P_i generates uniform values $\beta^i_{a,j}, \beta^i_{b,j}, \beta^i_{r,j} \in \mathbb{F}_{p^k}$
3. P_i computes and broadcasts encryptions of all the above values
4. P_i Invokes Π_{ZKPoPK} on the above ciphertexts
5. local comp.: get $\mathrm{Enc}_{pk}(a), \mathrm{Enc}_{pk}(b), \mathrm{Enc}_{pk}(r)$

$$\text{E.g.:} \quad \mathrm{Enc}_{pk}(a) = \mathrm{Enc}_{pk}\left(\sum_{j=1}^{n} a_j\right) \leftarrow \sum_{j=1}^{n} \mathrm{Enc}_{pk}(a_j)$$

6. local comp.: get $\mathrm{Enc}_{pk}(r + a \cdot b) \leftarrow \mathrm{Enc}_{pk}(r) + \mathrm{Enc}_{pk}(a) \cdot \mathrm{Enc}_{pk}(b)$
7. agreement on decrypting: everyone gets $a \cdot b + r$

Preprocessing Phase

Triples

from $a \cdot b + r$ to $[c] = [a \cdot b]$ & MACs on it:

8. P_1 sets $c_1 \leftarrow (r + c) - r_1$, P_i sets $c_i \leftarrow -r_i$, for $(i \neq 1)$
9. All players compute $\mathsf{Enc}_{pk}(c_1) \leftarrow \mathsf{Enc}_{pk}(r + c, \mathbf{0}) - \mathsf{Enc}_{pk}(r_1)$
10. All players set $\mathsf{Enc}_{pk}(c_i) \leftarrow -\mathsf{Enc}_{pk}(r_i)$, for $(i \neq j)$
11. P_i computes encryptions on MACs for a_j (sim. b_j, c_j):

$$\mathsf{Enc}_{pk}(\mathsf{MAC}^i(a_j)) \leftarrow \mathsf{Enc}_{pk}(\alpha^i_j) \cdot \mathsf{Enc}_{pk}(a_j) + \mathsf{Enc}_{pk}(\beta^i_{a,j})$$

Not Happy with the Current Online Phase?

As said, $[x]$ means $O(n^2)$ keys and MACs to compute securely.

$$[x] = \left((x_i)_{i=1}^{n}, \left(\mathsf{MAC}^j(x_i)\right)_{i,j=1}^{n}, \left((\alpha_j^i, \beta_{x,j}^i)\right)_{i,j=1}^{n} \right)$$

- Additive secret sharing of x
- MACs on shared values
- Keys for the MACs

MACs on *shares* $\Rightarrow$ Authentication on *secret values*.
Why not MACs on *secret values*?

There you go

Assuming α obtained by the players in bracket notation $[\alpha]$,

$$\langle x \rangle := (\delta, (x_i)_{i=1}^{n}, (\gamma(x)_i)_{i=1}^{n})$$

- δ: a public value (dependent of x)
- additive secret sharing of x
- additive secret sharing of $\gamma(x) = \alpha \cdot (x + \delta)$ (MAC on x)

Note: "partial openings" during computation (value reconstructed, MAC not reconstructed), in order to keep α secret!

Note: MACs not reconstructed during computation $\Rightarrow$ values may be incorrect.

Usage – Sketch

Preproc.:	Generate $[\alpha]$
	Generate $[x]$'s
	Compute $[\alpha \cdot x]$'s – killing one bracket-triple
	Set $\langle x \rangle \leftarrow (0, (x_i)_{i=1}^n, ((\alpha \cdot x)_i)_{i=1}^n)$ for all x's
Add.:	As in bracket notation! (local addition)
Mult.:	Using [Bea91], but partially opening $\langle x \rangle - \langle a \rangle, \langle y \rangle - \langle b \rangle$
Output:	Generate comb. of MACs of opened values,
	Commit, reconstruct the key,
	Comb. was valid? $\Rightarrow$ output.

Usage – Output

Setting: $\langle y \rangle = (\delta, (y_i)_{i=1}^n, (\gamma(y)_i)_{i=1}^n)$ to be output to P_h,
$\langle a_j \rangle = \left(\delta_j, (a_{j,l})_{l=1}^n, (\gamma(a_j)_l)_{l=1}^n\right)$, $1 \leq j \leq T$ opened.

Output

1. Public values $e_1, \ldots, e_T \in \mathbb{F}_{p^k}$ are generated
2. Players compute $a \leftarrow \sum_j e_j \cdot a_j$
3. P_i commits to $\gamma_i \leftarrow \sum_j e_j \gamma(a_j)_i, y_i, \gamma(y)_i$
4. $[\alpha]$ is reconstructed
5. P_i opens γ_i
6. Players check $\alpha \left(a + \sum_j e_j \cdot \delta_j\right) = \sum_i \gamma_i$
7. Commitments to $y_i, \gamma(y)_i$ are opened to P_h
8. P_h computes $y \leftarrow \sum_i y_i$ and checks $\alpha(y + \delta) = \sum_i \gamma(y)_i$

1. Introduction
2. Online Phase
3. Preprocessing
4. An Improved Online Phase
5. Concrete Scheme
6. Benchmarks

Packing Stuff

In this talk: how to squeeze messages into one value.
More details on the cryptoscheme? Check the paper!

Our SHE scheme

A variant of [BV11],

- with distributed decryption,
- specialized for parallel operations on multiple data.

Plaintexts live in $(\mathbb{F}_{p^k})^s$,
while ciphertexts in $(A_q)^3$ (for a convenient algebra A_q).

Packing Stuff – Choose your Angle

First task: thinking of $\mathbf{m} \in (\mathbb{F}_{p^k})^s$ as an element in A_q.
$F = \Phi_m \in \mathbb{Z}[X]$: cyclotomic polynomial of degree $N = \phi(m)$.

Choice of m?

Such that F mod p factors into at least s irreducible factors, each with degree divisible by k.

Concretely: $F \bmod p = f_1 \cdots f_{s'} \in \mathbb{F}_p[X]$, $\deg(f_i) = k_i \cdot k$.

Packing Stuff – The Final Deal

Facts

- $\mathbb{F}_p[X]/(f_i)$ is an extension field of $\mathbb{F}_{p^k}$
- $\mathbb{F}_p[X]/(f_i)$ is a direct summand of $\mathbb{F}_p[X]/(F)$
- $\mathbb{Z}^N$ projects onto $\mathbb{F}_p[X]/(F)$
- for large q: computation on elements in $\mathbb{Z}^N$ with small infinity norm can be thought as in $A_q := (\mathbb{Z}/q\mathbb{Z})[X]/(F)$

Encoding Messages?

$$\mathbf{m} \in (\mathbb{F}_{p^k})^s \hookrightarrow \bigoplus_{i=1}^{s'} \mathbb{F}_p[X]/(f_i) \xrightarrow{\sim} \mathbb{F}_p[X]/(F) \hookrightarrow \mathbb{Z}^N \twoheadrightarrow A_q$$

Preprocessing – the Numbers

Comparison to previous work:

- u: security parameter
- κ: size of encryption

	[BDOZ11]	Our work
Encryption Type	Semi-Homomorphic	SHE, mult. depth 1
ZKPoPK amortized complexity	$O(\kappa + u)$ bits	$O(\kappa + u)$ bits
Correct Mult. amortized complexity	$O(\kappa \cdot u)$ bits	0
offline benchmark (2-party case)	2-4sec (Paillier 1024-bit)	8msec (sec.: RSA 1024-bit*)

*: using a SHE scheme based on [BV11].

Online – the Numbers

Comparison to previous work:

- n: #players
- m_f: #multiplications in the circuit to compute

	[BDOZ11]	Our work
Complexity for one secure mult	$O(n^2)$ $\mathbb{F}_p$-mults	$O(n)$ $\mathbb{F}_p$-mults
Preprocessed data needed	$\Theta(m_f \cdot n^2)$	$O(m_f \cdot n)$

http://eprint.iacr.org/2011/535.pdf

THANKS

Rikke Bendlin, Ivan Damgård, Claudio Orlandi, and Sarah Zakarias.
Semi-homomorphic encryption and multiparty computation.
In Kenneth G. Paterson, editor, *EUROCRYPT*, volume 6632 of *Lecture Notes in Computer Science*, pages 169–188. Springer, 2011.

Donald Beaver.
Efficient multiparty protocols using circuit randomization.
In Joan Feigenbaum, editor, *CRYPTO*, volume 576 of *Lecture Notes in Computer Science*, pages 420–432. Springer, 1991.

Zvika Brakerski and Vinod Vaikuntanathan.
Fully homomorphic encryption from ring-lwe and security for key dependent messages.
In Phillip Rogaway, editor, *CRYPTO*, volume 6841 of *Lecture Notes in Computer Science*, pages 505–524. Springer, 2011.

Craig Gentry.
Fully homomorphic encryption using ideal lattices.
In Michael Mitzenmacher, editor, *STOC*, pages 169–178. ACM, 2009.