Multiparty Computation from Somewhat Homomorphic Encryption

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

[1]Aarhus University

[2]Bristol University

August 22, 2012

Our work: What is it?

An(other) MPC protocol:

- Active security
- Dishonest majority
- Computational security
- Universally composable

Previous work (examples):

- Early construction [CLOS02]
- “MPC in the Head” approach [IKOS07, IPS08]
- Preprocessing model [DO10, BDOZ11, NNOB12]

Notation

[BDOZ11]: (BeDOZa)

"Semi-Homomorphic Encryption and Multiparty Computation"

Notation

[BDOZ11]: (BeDOZa)

"Semi-Homomorphic Encryption and Multiparty Computation"

SPDZ: (SPeeDZ) $\leftarrow$ This talk!

"Multiparty Computation from Somewhat Homomorphic Encryption"

SPDZ Old Techniques – The Preprocessing Model

2-phases approach

Preprocessing	$\Longrightarrow$	Online
Shared randomness generation (public key crypto required)	$\Longrightarrow$	Evaluation of f using preprocessed data

SPDZ Old Techniques – The Preprocessing Model

2-phases approach

Preprocessing	$\Longrightarrow$	Online
Shared randomness generation (public key crypto required)	$\Longrightarrow$	Evaluation of f using preprocessed data

Features:

- Preprocessing: independent of f
- Online phase: very fast – no PKE!

Digression on [BDOZ11]'s Online Phase

Computation: on additive secret sharing

$$\text{Secret } x = x_1 + \cdots + x_n, \qquad x_i \longrightarrow P_i$$

Security: information theoretic MACs on shares

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

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

$$[x] := \left(x_i, \quad \left(\mathrm{MAC}^j(x_i)\right)_{j=1, j\neq i}^n, \quad \left(\left(\alpha_j^i, \beta_{x,j}^i\right)\right)_{j=1, j\neq i}^n \right)_{i=1,\ldots,n}$$

Computation with Secret Sharing and MACs

How to compute $[x + y]$ from $[x]$ and $[y]$?

Very easy! $P_i : x_i + y_i, \qquad \mathrm{MAC}^j(x_i) + \mathrm{MAC}^j(y_i), \qquad \beta^i_{x,j} + \beta^i_{y,j}$

Computation with Secret Sharing and MACs

How to compute $[x+y]$ from $[x]$ and $[y]$?

Very easy! $P_i : x_i + y_i, \qquad \mathrm{MAC}^j(x_i) + \mathrm{MAC}^j(y_i), \qquad \beta^i_{x,j} + \beta^i_{y,j}$

How to compute $[x \cdot y]$ from $[x]$ and $[y]$?

Using [Bea91]: easy if players have a “multiplicative triple” $[a], [b], [a \cdot b]$:

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

$$[z] = [a \cdot b] - \varepsilon \cdot [b] - \delta \cdot [a] + \varepsilon \cdot \delta.$$

$[z]$ equals $[x \cdot y]$:

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

Summary on the Online Phase

Computation

Linear secret sharing and MACs	$\rightarrow$	$[x + y]$: locally add
Multiplicative triples	$\rightarrow$	$[x \cdot y]$: add and reconstruct

Summary on the Online Phase

Computation

Linear secret sharing and MACs $\rightarrow$ $[x + y]$: locally add
Multiplicative triples $\rightarrow$ $[x \cdot y]$: add and reconstruct

Security

Secret sharing inputs $\rightarrow$ privacy
MACs (on shares) $\rightarrow$ authenticity

Summary on the Online Phase

Computation

Linear secret sharing and MACs $\rightarrow$ $[x+y]$: locally add
Multiplicative triples $\rightarrow$ $[x \cdot y]$: add and reconstruct

Security

Secret sharing inputs $\rightarrow$ privacy
MACs (on shares) $\rightarrow$ authenticity

Data needed per secret

One secret $\rightarrow$ n shares $\rightarrow$ n MACs (and keys) per share $\rightarrow$
$\rightarrow$ $O(n^2)$ field elements per secret.

Lowering the amount of data needed?

The Catch

In [BDOZ11], MACs on *shares* to authenticate *secret*.

Lowering the amount of data needed?

The Catch

In [BDOZ11], MACs on *shares* to authenticate *secret*.
Why not MACs on *secret* to authenticate *secret*?

Lowering the amount of data needed?

The Catch

In [BDOZ11], MACs on *shares* to authenticate *secret*.
Why not MACs on *secret* to authenticate *secret*?

Assuming $[\alpha]$ (*one* single value for *all* secrets),

$$\langle x \rangle := (x_1, \ldots, x_n, \quad \gamma(x)_1, \ldots, \gamma(x)_n) \qquad (x_i, \gamma(x)_i) \to P_i$$

$$\begin{aligned} x_1, \ldots, x_n &: \text{ additive secret sharing of } x \\ \gamma(x)_1, \ldots, \gamma(x)_n &: \text{ additive secret sharing of } \gamma(x) = \alpha \cdot x \text{ (MAC on } x\text{)} \end{aligned}$$

Data needed per secret

One secret $\to$ n shares $+$ n shares of a MAC $\to$
$\to O(n)$ field elements per secret.

Does it really work?

Setup

MAC Keys in $[\cdot]$: privately held, different secret $\rightarrow$ different key
MAC Keys in $\langle\cdot\rangle$: $[\alpha]$, unique for all secrets!

Does it really work?

Setup

MAC Keys in $[\cdot]$: privately held, different secret $\rightarrow$ different key
MAC Keys in $\langle\cdot\rangle$: $[\alpha]$, unique for all secrets!

Problem

P_i needs α to check a MAC $\rightarrow$ P_i can later forge MACs!
$\rightarrow$ Gate-by-gate check $=$ insecure

Does it really work?

Setup

MAC Keys in $[\cdot]$: privately held, different secret $\rightarrow$ different key
MAC Keys in $\langle\cdot\rangle$: $[\alpha]$, unique for all secrets!

Problem

P_i needs α to check a MAC $\rightarrow$ P_i can later forge MACs!
$\rightarrow$ Gate-by-gate check $=$ insecure

Solution

- Compute the whole circuit with no checks
- Commit to MACs
- Open $[\alpha]$
- Check MACs

Online – the Numbers

Notation:

- n: # players
- m_f: # multiplications in the circuit C to compute
- $|C|$: Circuit size

	[BDOZ11]	SPDZ
Preprocessed data needed	$\Theta(m_f \cdot n^2)$	$O(m_f \cdot n)$
Complexity (field mults)	$\Omega(\|C\| \cdot n^2)$	$O(\|C\| \cdot n + n^3)$
Amo. timing (64bit prime field)	7.7ms	0.05ms

Note

Preproc. data needed: Optimal up to constant factor.
Complexity: Optimal up to poly-log factors.

High Level Idea

- Generate $a = a_1 + \cdots + a_n$, $b = b_1 + \cdots + b_n$
- Generate and broadcast encryptions $\mathsf{Enc}(a_i)$, $\mathsf{Enc}(b_i)$
- Compute an encryption $\mathsf{Enc}(c)$, where $c = a \cdot b$
- Distribute c_i to P_i, where $c = c_1 + \cdots + c_n$

High Level Idea

- Generate $a = a_1 + \cdots + a_n$, $b = b_1 + \cdots + b_n$
- Generate and broadcast encryptions $\mathsf{Enc}(a_i)$, $\mathsf{Enc}(b_i)$
- Compute an encryption $\mathsf{Enc}(c)$, where $c = a \cdot b$
- Distribute c_i to P_i, where $c = c_1 + \cdots + c_n$

Problems

Does P_i know the plaintext contained in $\mathsf{Enc}(a_i)$, $\mathsf{Enc}(b_i)$?
How to compute $\mathsf{Enc}(c)$?

High Level Idea

- Generate $a = a_1 + \cdots + a_n$, $b = b_1 + \cdots + b_n$
- Generate and broadcast encryptions $\mathsf{Enc}(a_i)$, $\mathsf{Enc}(b_i)$
- Compute an encryption $\mathsf{Enc}(c)$, where $c = a \cdot b$
- Distribute c_i to P_i, where $c = c_1 + \cdots + c_n$

Problems

Does P_i know the plaintext contained in $\mathsf{Enc}(a_i)$, $\mathsf{Enc}(b_i)$?
How to compute $\mathsf{Enc}(c)$?

Solutions

First problem: a ZK-Proof.
Second problem: a very expensive ZK-Proof. . . or?

The Right Encryption Scheme

The Problem:

Given *fresh* $\text{Enc}(a_1), \ldots, \text{Enc}(a_n)$, $\text{Enc}(b_1), \ldots, \text{Enc}(b_n)$, compute:

$\text{Enc}(a)$ $\text{Enc}(b)$

$\text{Enc}(c)$

Where $a_1 + \cdots + a_n = a$, $b_1 + \cdots + b_n = b$, $c = a \cdot b$

Fresh: a ciphertext computed via the encryption algorithm.

The Right Encryption Scheme

The Nicest Solution:

Given *fresh* $\mathsf{Enc}(a_1), \dots, \mathsf{Enc}(a_n)$, $\mathsf{Enc}(b_1), \dots, \mathsf{Enc}(b_n)$, compute:

$$\mathsf{Enc}(a) \leftarrow \sum_i \mathsf{Enc}(a_i), \qquad \mathsf{Enc}(b) \leftarrow \sum_i \mathsf{Enc}(b_i)$$

$$\mathsf{Enc}(c) \leftarrow \mathsf{Enc}(a) \cdot \mathsf{Enc}(b).$$

Where $a_1 + \cdots + a_n = a$, $b_1 + \cdots + b_n = b$, $c = a \cdot b$

Fresh: a ciphertext computed via the encryption algorithm.

Our Abstract Scheme

Somewhat Homomorphic Encryption Scheme

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

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

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

In our case: $S =$ circuits of mult depth one.

Our Concrete Scheme

A variant of Brakerski Vaikuntanathan [BV11] (based on Ring-LWE)

Features of our variant

- computation of circuits of multiplicative depth 1 on ciphertexts,
- distributed decryption,
- specialized for parallel operations on multiple data (SIMD).

Preprocessing – The Numbers

Notation:

- u: security parameter
- κ: size of encryption

	[BDOZ11]	SPDZ
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, sec=80bits)	2-4sec	0.008sec

Summary

SPDZ

- Active security, dishonest majority, preprocessing model
- Online phase:
  - Linear amount of data needed
  - Essentially linear communication complexity
- Preprocessing:
  - Rational use of SHE
  - Fewer ZK protocols, compared to [BDOZ11]
  - Very practical

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

Summary

SPDZ

- Active security, dishonest majority, preprocessing model
- Online phase:
  - Linear amount of data needed
  - Essentially linear communication complexity
- Preprocessing:
  - Rational use of SHE
  - Fewer ZK protocols, compared to [BDOZ11]
  - Very practical

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

THANKS

Rikke Bendlin, Ivan Damgård, Claudio Orlandi, and Sarah Zakarias.
Semi-homomorphic encryption and multiparty computation.
In *EUROCRYPT*, pages 169–188, 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.

Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai.
Universally composable two-party and multi-party secure computation.
In *STOC*, pages 494–503, 2002.

Ivan Damgård and Claudio Orlandi.

Multiparty computation for dishonest majority: From passive to active security at low cost.
In *CRYPTO*, pages 558–576, 2010.

Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai.
Zero-knowledge from secure multiparty computation.
In David S. Johnson and Uriel Feige, editors, *STOC*, pages 21–30. ACM, 2007.

Yuval Ishai, Manoj Prabhakaran, and Amit Sahai.
Founding cryptography on oblivious transfer - efficiently.
In David Wagner, editor, *CRYPTO*, volume 5157 of *Lecture Notes in Computer Science*, pages 572–591. Springer, 2008.

Jesper Buus Nielsen, Peter Sebastian Nordholt, Claudio Orlandi, and Sai Sheshank Burra.
A new approach to practical active-secure two-party computation.
In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO*, volume 7417 of *Lecture Notes in Computer Science*, pages 681–700. Springer, 2012.