The Second IEEE International Conference on Trust, Privacy and Security in Intelligent Systems, and Applications

Revisiting Secure Computation using Functional Encryption: Opportunities and Research Directions

Runhua Xu[+] and James Joshi[*]

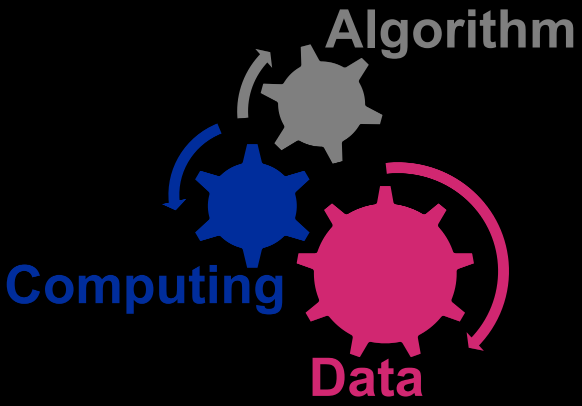
[+] *AI Security and Privacy, IBM Research – Almaden Research Center*
[*] *University of Pittsburgh & NSF* [#]

IBM

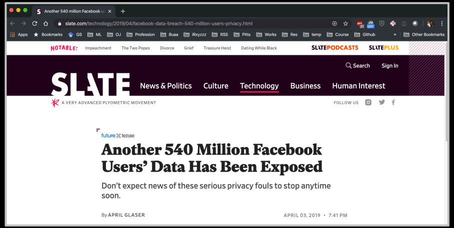
[#]This work was performed while James Joshi was serving as a Program Director at NSF; the work represents the authors’ views and not that of NSF’s.

Background

Privacy-Preserving Data Processing



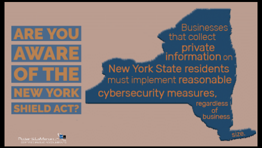
DATA
concerns of data leakage
regulation



China Internet Security Law
June 1, 2017

GDPR
May 25, 2018

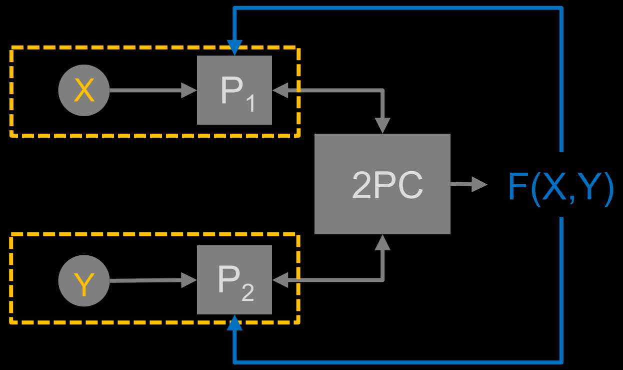
HIPAA
August 21, 1996



CCPA
January 1, 2020

NY SHIELD
March 21, 2020

Background
Secure Computation



Secure Multi-party Computation (SMC)
Multi-Party Computation (MPC)
Secure Function Evaluation (SFE)

Secure Two-party Computation (2PC)
Andrew Yao
1980s
“Protocols for secure computation”
“How to generate and exchange secrets”

APPLICATION
secure search
secure auction
privacy-preserving biometric authentication
privacy-preserving machine learning

Secure Computation
Typical Approaches and Techniques Stack

Garbled Circuits
+
Oblivious Transfer

Transmission Overhead

Homomorphic Enc

Preprocessing Model
Pure Fully HE

Computation Cost

Mixed Protocol

GC, HE
GMW, Secret Sharing

Functional Enc

similar to HE
computation over
ciphertext

Generality v.s. Practicality

Computation Efficiency

Scalability

Practicality

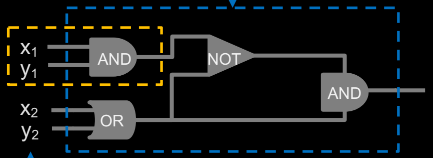
Threat Model

Transmission Efficiency

Existing Secure Computation
Generic SMC using Garbled Circuits

F (X,Y)

Alice: $x \rightarrow (x_1, x_2)^{01}$
Bob: $y \rightarrow (y_1, y_2)^{01}$



Alice

AND - Truth Table

x_1	y_1	AND
0	0	0
0	1	0
1	0	0
1	1	1

Keys

$x_1^0 \rightarrow w_{x_1}^0$
$x_1^1 \rightarrow w_{x_1}^1$
$y_1^0 \rightarrow w_{y_1}^0$
$y_1^1 \rightarrow w_{y_1}^1$
$g_{AND}^0 \rightarrow w_{g_{AND}}^0$
$g_{AND}^1 \rightarrow w_{g_{AND}}^1$

Encrypted Truth Table

x_1	y_1	AND	Encrypted
0	0	0	$E_{w_{x_1}^0}(E_{w_{y_1}^0}(w_{g_{AND}}^0))$
0	1	0	$E_{w_{x_1}^0}(E_{w_{y_1}^1}(w_{g_{AND}}^0))$
1	0	0	$E_{w_{x_1}^1}(E_{w_{y_1}^0}(w_{g_{AND}}^0))$
1	1	1	$E_{w_{x_1}^1}(E_{w_{y_1}^1}(w_{g_{AND}}^1))$

Garbled Table

Garbled Circuit
$E_{w_{x_1}^0}(E_{w_{y_1}^0}(w_{g_{AND}}^0))$
$E_{w_{x_1}^1}(E_{w_{y_1}^1}(w_{g_{AND}}^1))$
$E_{w_{x_1}^1}(E_{w_{y_1}^0}(w_{g_{AND}}^0))$
$E_{w_{x_1}^0}(E_{w_{y_1}^1}(w_{g_{AND}}^0))$

Alice sends $w_{x_1}^b$ ($b \in \{0,1\}$ based on value of x_1) and garbled table to Bob

Bob

If Bob knows $w_{y_1}^b$,

it can decrypt to get AND gate result $w_{g_{AND}}^{\{0,1\}}$
then compute next NOT gate
secure evolution issue solved

How to let Bob knows key $w_{y_1}^b$
without leaking y_1 to Alice?

Alice **Bob**

$y_1^0 \rightarrow w_{y_1}^0$
$y_1^1 \rightarrow w_{y_1}^1$

Oblivious Transfer

y_1^b

$w_{y_1}^b$

Suppose that $y_1^b = 1$
Bob can only learn $w_{y_1}^0$
Alice cannot know which $w_{y_1}^0$ or $w_{y_1}^1$ is chosen

Existing Secure Computation
SMC Derived from Homomorphic Encryption

General description of homomorphic encryption

$$pk, sk \leftarrow KGen(1^\lambda)$$
$$C_{HE} \leftarrow \{Enc_{pk}(m_1), \dots, Enc_{pk}(m_n)\}$$
$$C_{HE}^f \leftarrow Eval_{pk}(f, C_{HE})$$
$$f(m_1, \dots, m_n) \leftarrow Dec_{sk}(C_{HE}^f)$$

Types

- partially HE: one type of gate, e.g., addition or multiplication
- somewhat HE: two types of gates, but only for subset of circuits
- leveled fully HE: arbitrary circuits of bounded (pre-determined) depth
- fully HE: arbitrary circuits of unbounded depth

Preprocessing model approach

- Offline Process:
    - trusted dealer: provides raw materials for the computation (somewhat HE)
- Online Process:
    - use only inexpensive primitives to evaluate a function

Pure fully HE approach

- Directly adopt the fully HE
    - each party encrypt their input
    - all party perform a distributed decryption on C_{HE}^f

Existing Secure Computation
Achieving MC in Mixed-Manner

General Idea: evaluate operations according to their best efficient representations

- additions and multiplications
    - has efficient representation as an arithmetic circuit
    - use HE approach
- comparison operations
    - has efficient representation as a Boolean circuit
    - use GC+OT approach

Typical Proposals

- TASTY framework
    - compiler a function using HE, GC, OT, etc.
- ABY framework
    - arithmetic sharing + boolean sharing + garbled circuits
- ABY^3 framework
    - ABY in the three-party setting for privacy-preserving ML
- Chameleon framework
    - ABY with a semi-honest third-party for preprocessing arithmetic triples
        - previously, OT used
    - Can handle signed fixed-point numbers

Secure Computation
using Emerging Functional Encryption

General description of Functional Encryption

$$pk, msk \leftarrow Setup(1^\lambda)$$
$$dk_f \leftarrow KGen(msk)$$
$$C_{FE} \leftarrow \{Enc_{pk}(m_1), \dots, Enc_{pk}(m_n)\}$$
$$f(m_1, \dots, m_n) \leftarrow Dec_{dk_f}(C_{HE}^f)$$

$$\boxed{\begin{array}{c} pk, sk \leftarrow KGen(1^\lambda) \\ C_{HE} \leftarrow \{Enc_{pk}(m_1), \dots, Enc_{pk}(m_n)\} \\ C_{HE}^f \leftarrow Eval_{pk}(f, C_{HE}) \\ f(m_1, \dots, m_n) \leftarrow Dec_{sk}(C_{HE}^f) \end{array}}$$

General Functional Encryption

Decryption party is allowed to request a functional private key

$D_{dk_f}\left(E_{pk}(m_1, \dots, m_n)\right) = f(m_1, \dots, m_n)$ without leaking $m_1, \dots, m_n$ to decryption party

Functional Encryption for specific functionality: FE for Inner-Product

Where the allowed function is inner-product functions:

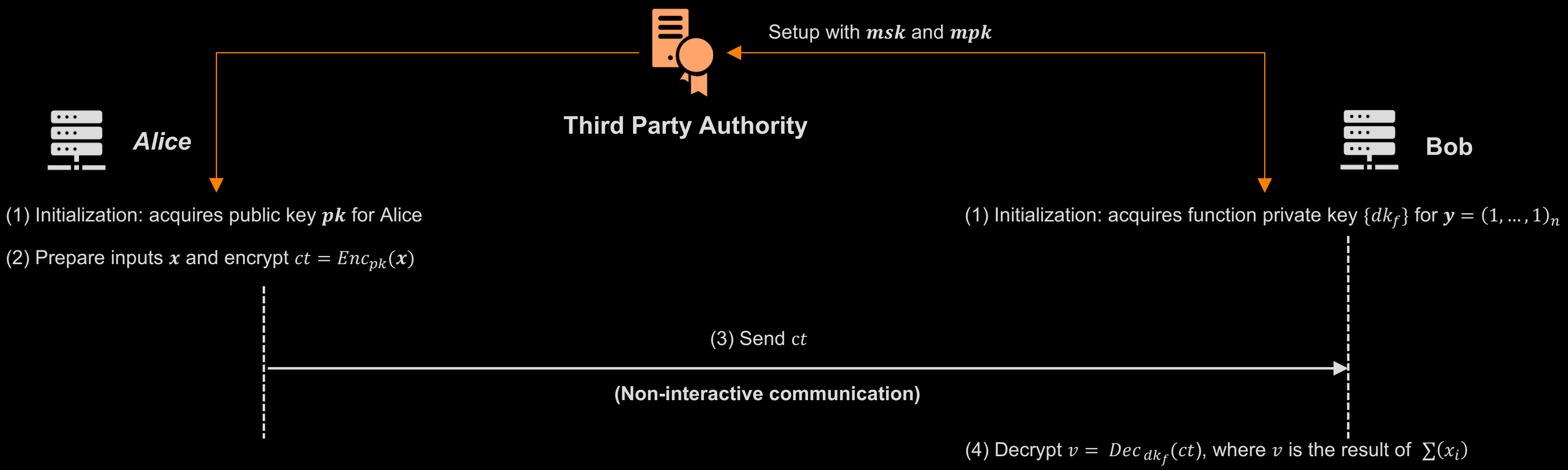
Single Input Functional Encryption for Inner-Product $f_S(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{n} x_i y_i \quad s.t. |\boldsymbol{x}| = |\boldsymbol{y}| = n$

$\boldsymbol{x}$ is from one single party, $\boldsymbol{y}$ is from the decryption party

Multiple Input Functional Encryption for Inner-Product $f_M((\boldsymbol{x_1}, \boldsymbol{x_2}, \dots, \boldsymbol{x_n}), \boldsymbol{y}) = \sum_{i=1}^{n} \sum_{j=1}^{\eta_i} (x_{ij} y_{\sum_{k=1}^{i-1} \eta_k + j}) \quad s.t. |\boldsymbol{x_i}| = \eta_i, |\boldsymbol{y}| = \sum_{i=1}^{n} \eta_i$

Secure Computation
using Emerging Functional Encryption

Secure Two-Party Computation using single-input functional encryption



Secure Computation
using Emerging Functional Encryption

Secure Multi-Party Computation using multi-input functional encryption

Setup with $\boldsymbol{msk}$ and $\boldsymbol{mpk}$

Third Party Authority

Data Sources

$D_1, \ldots, D_n$

Coordinator

(1) Initialization: acquires public key $\{\boldsymbol{pk_i}\}$ for each data source

(2) D_i prepares inputs x_i and encrypt $\{c_i = Enc_{pk_i}(x_i)\}$, respectively

(1) Initialization: acquires function private key $\{dk_f\}$ for $\boldsymbol{y} = (1, \ldots, 1)_n$

(3) Send $\{c_i\}$

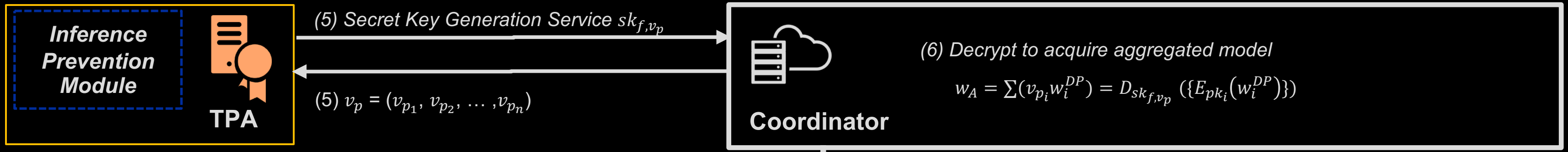
(Non-interactive communication)

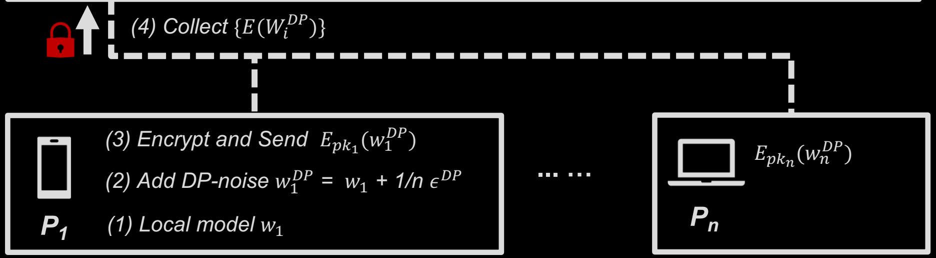
(4) Collect $\{c_i = Enc_{\,pk_i}(x_i)\}$

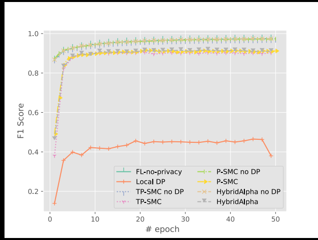
(5) Decrypt $v = \ Dec_{dk_f}(\{c_i\})$, where v is the result of $\sum(x_i)$

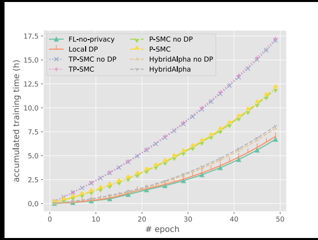
Secure Computation
using Emerging Functional Encryption

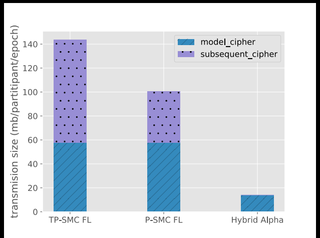
Application: FE-based SC in Privacy-Preserving Federated Learning











Runhua Xu, Nathalie Baracaldo, Yi Zhou, Ali Anwar, and Heiko Ludwig. Hybridalpha: An efficient approach for privacy-preserving federated learning.
In Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security, pages 13–23, 2019a

Secure Computation

FE Approach: Challenges and Directions

Enriching Functionality

Success in FE-based SC in PPFL and PPDL applications
(Inner-Product based FE)

Lack of FE-based SC for more functionalities
Comparison
Max/Min
Degree-n Polynomial Computation

Enhancing Security and Privacy Guarantees

Selective indistinguishability under chosen plaintext attack (IND-CPA)
decisional Diffie-Hellman assumption (DDH)

FE-based SMC
honest-but-curious coordinator
Against an adversary using quantum computing

Increasing Efficiency

Primary Objective in SC

FE-based SC is efficient than HE-based SC
for some functionalities
a challenge in large-scale secure computation scenario

Dynamic, Decentralized, and Threshold Setting

Existing FE-based SC in PPFL/PPDL
(a third-party authority - TPA)

How about removing requirement of TPA?
How about supporting dynamic parties?
How about supporting threshold setting?

Secure Computation

FE Approach: Challenges and Directions

Privacy-Preserving Applications

FE-based PP Federated Learning
(horizontal FL)
FE-based PP Deep Learning
(CNN model)

How about PPFL in vertical setting?
How about PPDL in more types of model?
(RNN, Transformer, Decision Tree, XGboost)

Transparent and Accountable Crypto Infrastructure

Existing FE-based SC in PPFL/PPDL
(a third-party authority – TPA – Critical Infrastructure)

Certificate Authority – Certificate Transparency
TPA – Authority Transparency
Accountable FE-based SMC (coordinator)

Realization and Open-Source Library

HE-based SC open-source libraries
IBM Research – HElib
Microsoft Research – SEAL

a need to establish open-source practical FE libraries

Benchmarks

FE-based SC is a nascent research area

needs benchmarks
to help broader researchers from other communities
to identify FE-based SC solutions.

Thanks

Q&A