Privacy-Preserving Collaboration Using Cryptography

Dr. Emily Shen

Briefing to the Federal Privacy Council
5 May 2020

LINCOLN LABORATORY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Need for Privacy-Preserving Collaboration

Social Good

Commercial Value

National Security

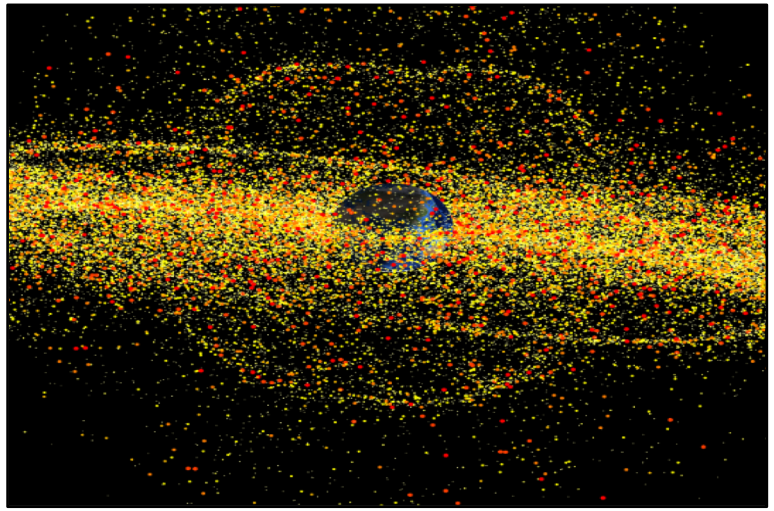
Privacy-Preserving Collaboration for National Security

Humanitarian Target Deconfliction

Parties: humanitarian aid orgs, military

Private inputs: locations of sites, targets

Desired output: locations in common



Satellite Collision Prediction

Parties: international satellite operators

Private inputs: maneuver schedules

Desired output: potential collisions

Collaborative Cyber Threat Analysis

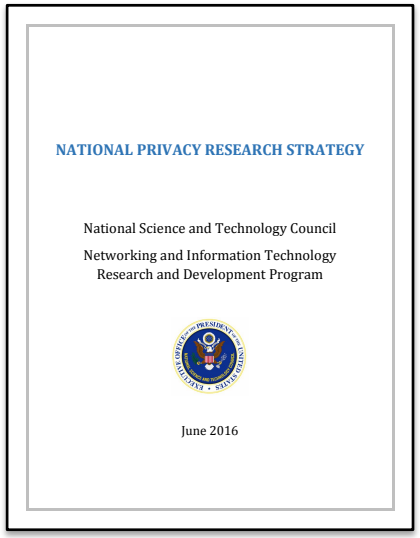
Parties: companies, government

Private inputs: observed threats

Desired output: aggregate threat info

National Need for Privacy-Preserving Technology

National Privacy Research Strategy, 2016


NATIONAL PRIVACY RESEARCH STRATEGY

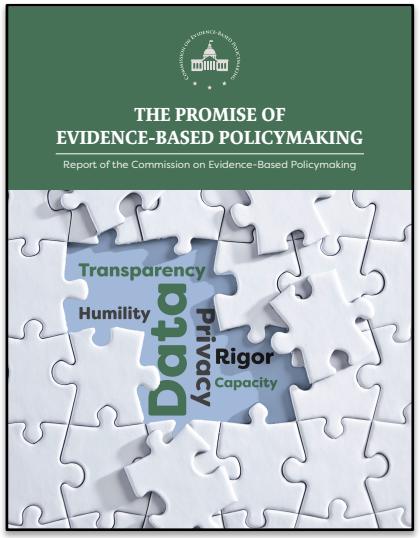
National Science and Technology Council

Networking and Information Technology Research and Development Program

June 2016

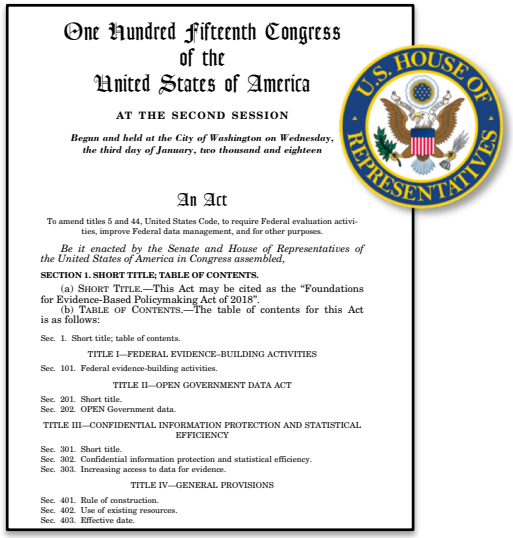
"How can *privacy-enhancing cryptographic technologies* be developed to scale?"

Report of the Commission on Evidence-Based Policymaking, 2017



"*Secure Multiparty Computation* may allow […] combin[ing] data and conduct[ing] analyses without directly accessing or storing information."

Foundations for Evidence-Based Policymaking Act of 2018


One Hundred Fifteenth Congress of the United States of America

AT THE SECOND SESSION

Begun and held at the City of Washington on Wednesday, the third day of January, two thousand and eighteen

An Act

To amend titles 5 and 44, United States Code, to require Federal evaluation activities, improve Federal data management, and for other purposes.

Be it enacted by the Senate and House of Representatives of the United States of America in Congress assembled,

SECTION 1. SHORT TITLE; TABLE OF CONTENTS.

(a) SHORT TITLE.—This Act may be cited as the "Foundations for Evidence-Based Policymaking Act of 2018".

(b) TABLE OF CONTENTS.—The table of contents for this Act is as follows:

Sec. 1. Short title; table of contents.

TITLE I—FEDERAL EVIDENCE-BUILDING ACTIVITIES

Sec. 101. Federal evidence-building activities.

TITLE II—OPEN GOVERNMENT DATA ACT

Sec. 201. Short title.
Sec. 202. OPEN Government data.

TITLE III—CONFIDENTIAL INFORMATION PROTECTION AND STATISTICAL EFFICIENCY

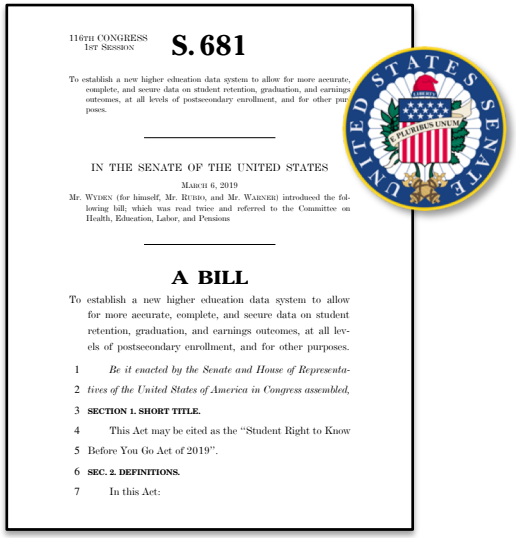
Sec. 301. Short title.
Sec. 302. Confidential information protection and statistical efficiency.
Sec. 303. Increasing access to data for evidence.

TITLE IV—GENERAL PROVISIONS

Sec. 401. Rule of construction.
Sec. 402. Use of existing resources.
Sec. 403. Effective date.

"facilitate data sharing, enable data linkage, and *develop privacy enhancing techniques*"

Student Right to Know Before You Go Bill, 2019


116TH CONGRESS 1ST SESSION

S. 681

To establish a new higher education data system to allow for more accurate, complete, and secure data on student retention, graduation, and earnings outcomes, at all levels of postsecondary enrollment, and for other purposes.

IN THE SENATE OF THE UNITED STATES

MARCH 6, 2019

Mr. WYDEN (for himself, Mr. RUBIO, and Mr. WARNER) introduced the following bill; which was read twice and referred to the Committee on Health, Education, Labor, and Pensions

A BILL

To establish a new higher education data system to allow for more accurate, complete, and secure data on student retention, graduation, and earnings outcomes, at all levels of postsecondary enrollment, and for other purposes.

Be it enacted by the Senate and House of Representatives of the United States of America in Congress assembled,

SECTION 1. SHORT TITLE.

This Act may be cited as the "Student Right to Know Before You Go Act of 2019".

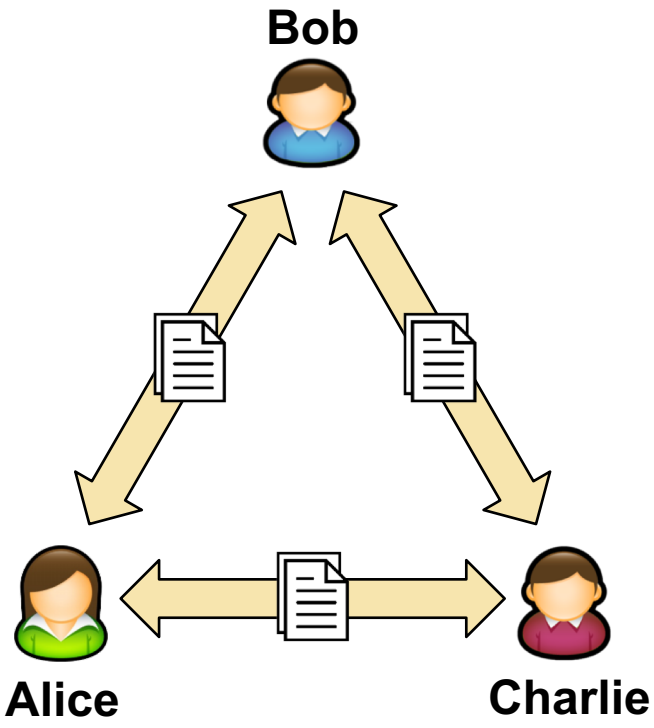
SEC. 2. DEFINITIONS.

In this Act:

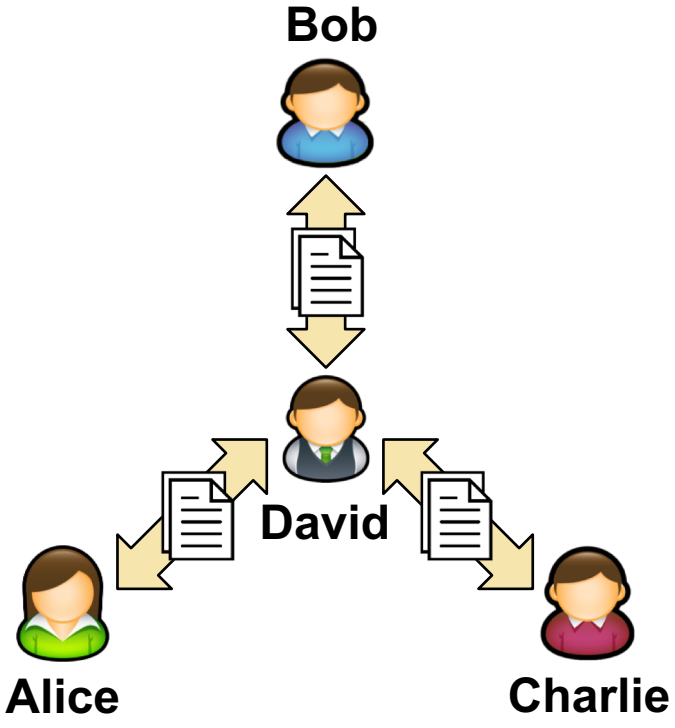
"the Commissioner shall use *secure multiparty computation technologies*"

Current Approaches to Collaboration

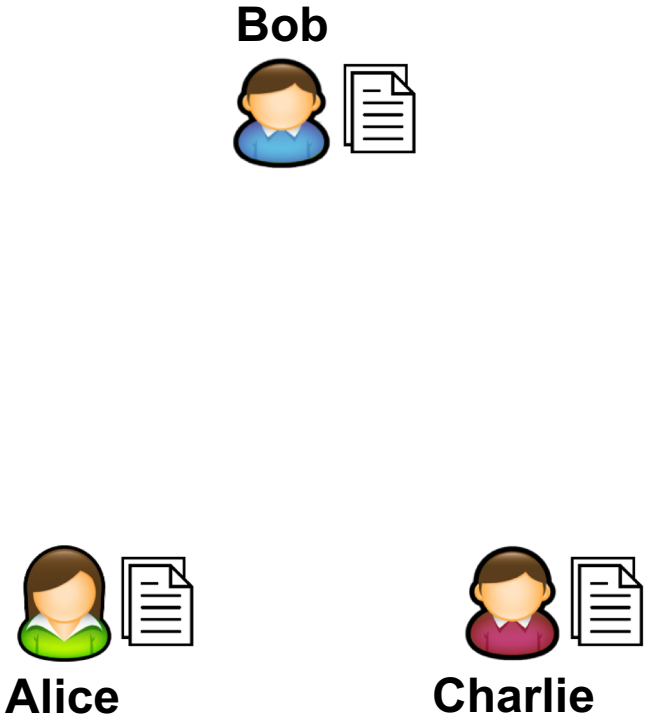
Share with each other



Share with an external party



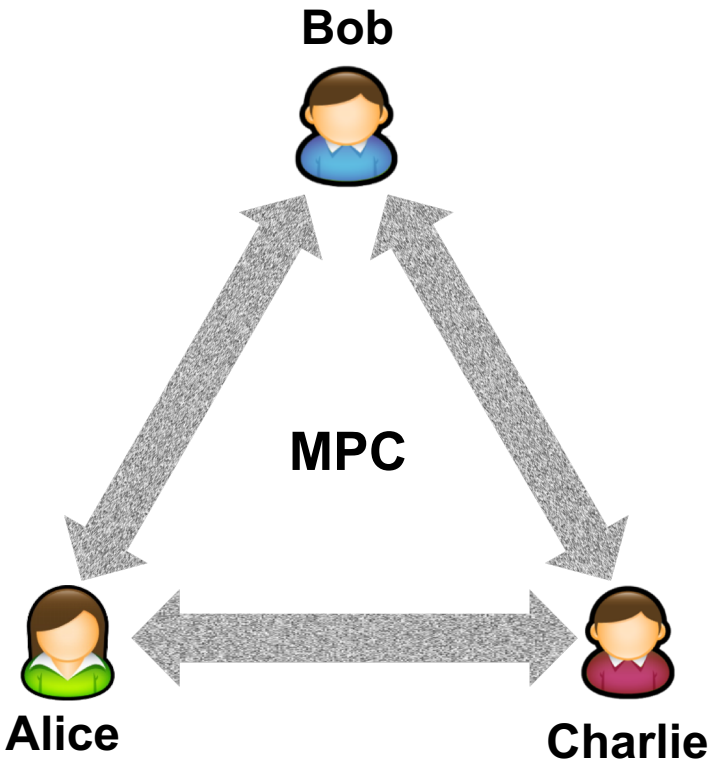
Don't share; don't learn results



Currently collaboration requires giving data to trusted parties, accepting security and privacy risks

Secure Multi-Party Computation (MPC)

- Goal: replace trusted party with technology
- Requirements:
  - Correctness: everyone learns correct result of computation
  - Security: no one learns anything beyond result
- Secure multi-party computation (MPC) provides correctness and security without a trusted party
  - For any computation
  - For any number of parties
- MPC invented in 1987, initially purely theoretical
- In past decade, MPC implemented and becoming practical for many applications



Secure MPC allows computing joint results without giving input data to any trusted party

Outline

- Motivation
- Secure Multi-Party Computation (MPC)
- Lincoln MPC Framework
- Prototypes and Deployments
- Summary

How MPC Works (1)

Secret-Share Inputs

- **Split each input into random *secret shares***
- **Send a distinct share to each party**

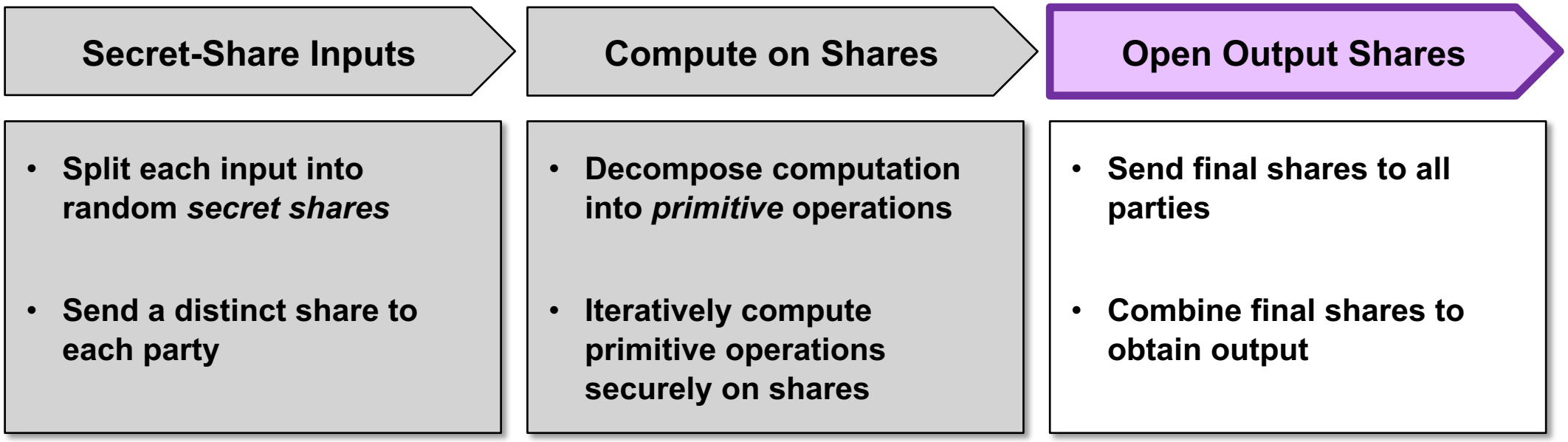
How MPC Works (2)

Secret-Share Inputs → **Compute on Shares**

- Split each input into random *secret shares*
- Send a distinct share to each party

- Decompose computation into *primitive* operations
- Iteratively compute primitive operations securely on shares

How MPC Works (3)

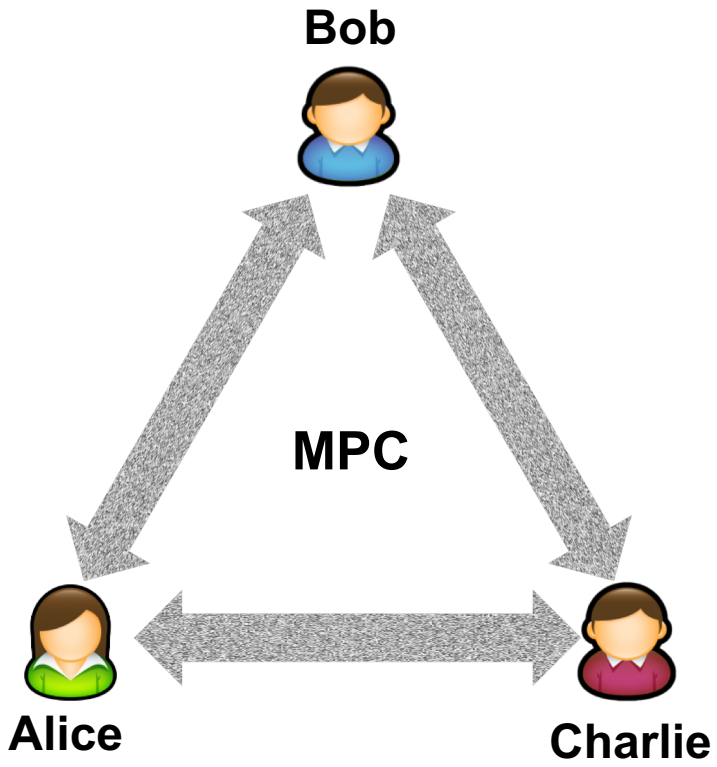


Example: MPC to Compute Average Salary

- Alice, Bob, Charlie want to securely compute average of their salaries
- Compute sum of salaries using MPC, then divide by 3

Inputs:
Monthly Salaries

Alice		$8K
Bob		$12K
Charlie		$10K



Output:
Sum of Salaries

$30K

Secret Sharing Inputs (1)

Secret-Share Inputs → **Compute on Shares** → **Open Output Shares**

	Inputs: Monthly Salaries	Secret Shares (mod 100)	Output: Sum of Salaries
Alice	$8K	16	
Bob	$12K	6	
Charlie	$10K	86	

$$8 - (16 + 6) \pmod{100}$$
$$= -14 \pmod{100} = 86$$

Secret Sharing Inputs (2)

Secret-Share Inputs → **Compute on Shares** → **Open Output Shares**

	Inputs: Monthly Salaries		Secret Shares (mod 100)		Output: Sum of Salaries
Alice		$8K	16	59	22
Bob		$12K	6	33	89
Charlie		$10K	86	20	99
			8	12	10

Computing on Shares

Secret-Share Inputs → **Compute on Shares** → Open Output Shares

	Inputs: Monthly Salaries	Secret Shares (mod 100)	Output: Sum of Salaries
Alice	 $8K	16 + 59 + 22 = 97	
Bob	 $12K	6 + 33 + 89 = 28	
Charlie	 $10K	86 + 20 + 99 = 5	

Opening Output Shares

	Inputs: Monthly Salaries		Secret Shares (mod 100)						Output: Sum of Salaries
Alice		$8K	16	+	59	+	22	=	97
Bob		$12K	6	+	33	+	89	=	28
Charlie		$10K	86	+	20	+	99	=	5

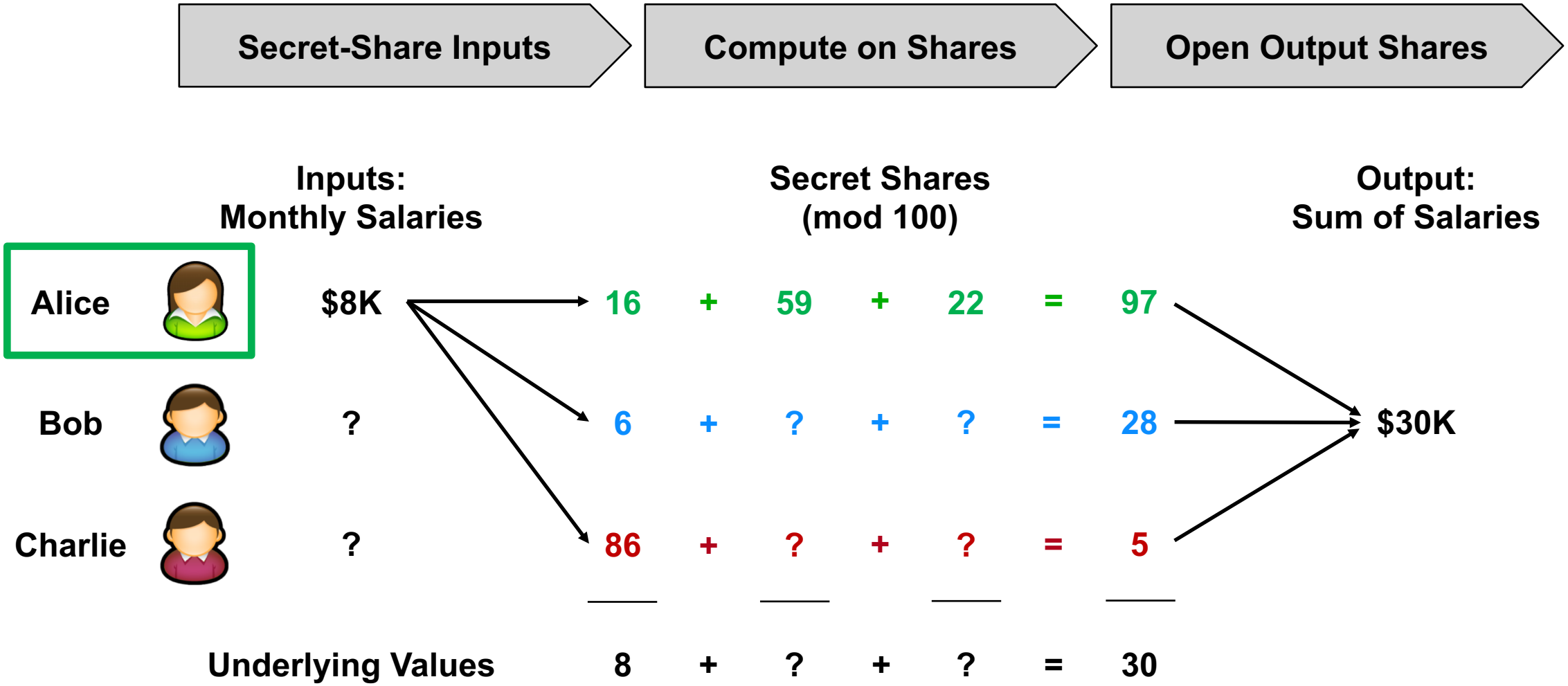
Correctness

Secret-Share Inputs → Compute on Shares → Open Output Shares

	Inputs: Monthly Salaries		Secret Shares (mod 100)						Output: Sum of Salaries
Alice		$8K	16	+	59	+	22	=	97
Bob		$12K	6	+	33	+	89	=	28
Charlie		$10K	86	+	20	+	99	=	5
	Underlying Values		8	+	12	+	10	=	30

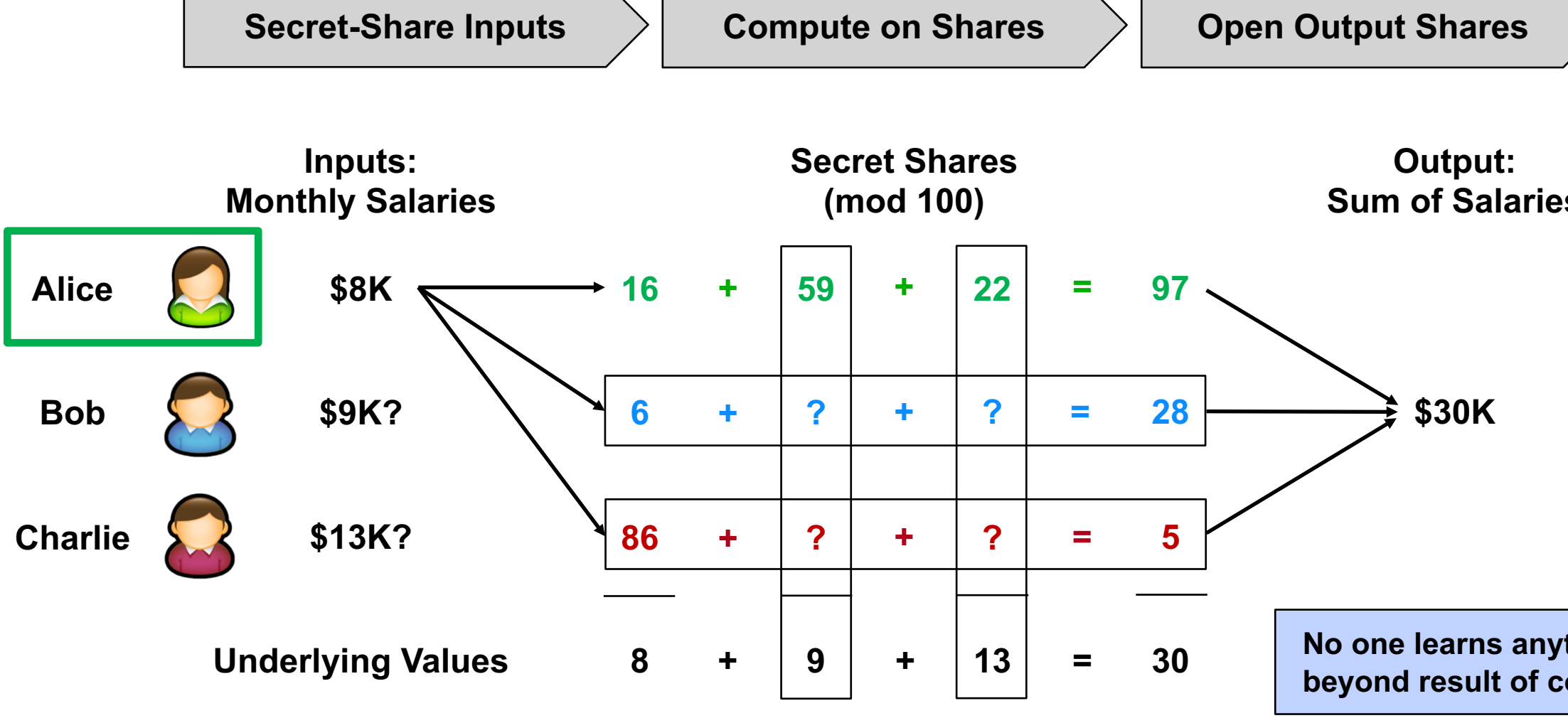
Everyone learns correct result of computation

Security (1)


Secret-Share Inputs
Compute on Shares
Open Output Shares
Inputs:
Monthly Salaries
Secret Shares
(mod 100)
Output:
Sum of Salaries
Alice
Bob
Charlie
$8K
?
?
16 + 59 + 22 = 97
6 + ? + ? = 28
86 + ? + ? = 5
$30K
Underlying Values
8 + ? + ? = 30

Security (2)

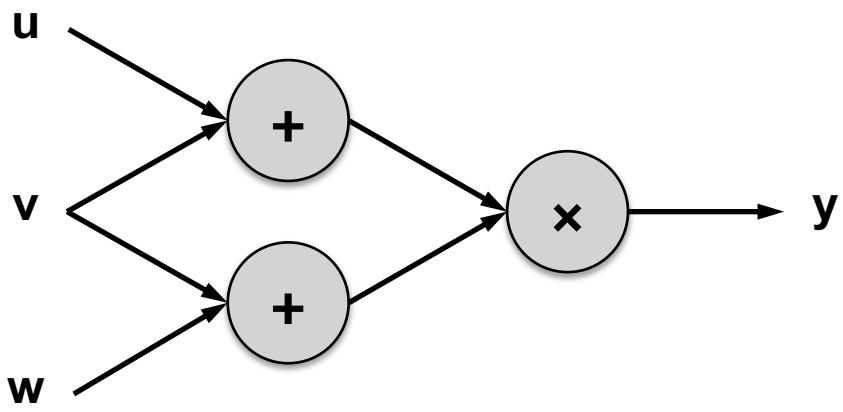
Secret-Share Inputs → Compute on Shares → Open Output Shares



No one learns anything beyond result of computation

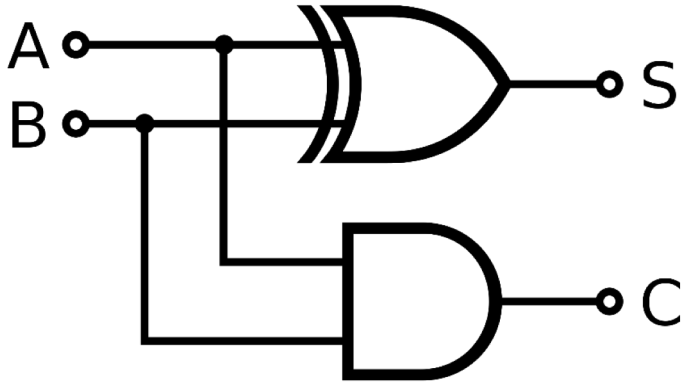
MPC for Any Function

MPC for Arithmetic Computation



MPC for add, multiply primitives over integers can securely compute any function!

MPC for Boolean Computation



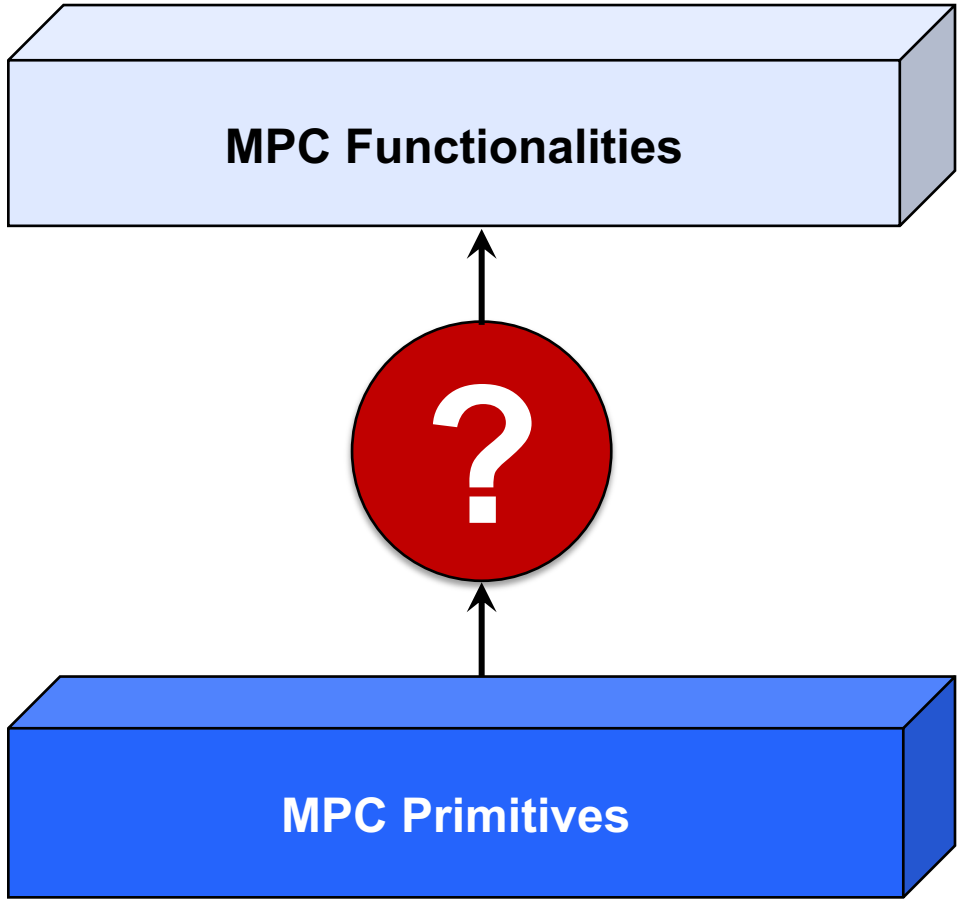
MPC for XOR, AND primitives over bits can securely compute any function!

MPC can securely compute any function using arithmetic or Boolean primitives

Outline

- Motivation
- Secure Multi-Party Computation (MPC)
- Lincoln MPC Framework
- Prototypes and Deployments
- Summary

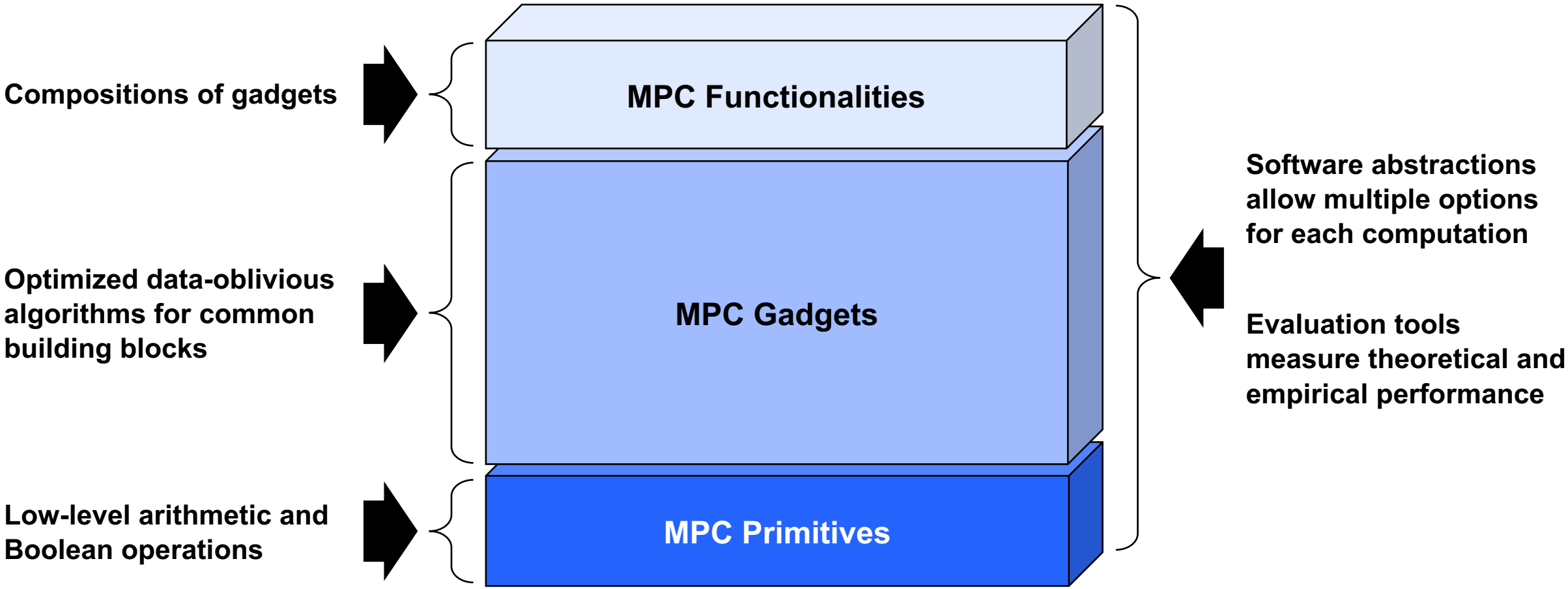
Challenges of Building Complex MPC Functionalities



Practical MPC development requires higher-level building blocks, data-oblivious algorithms, unique efficiency considerations

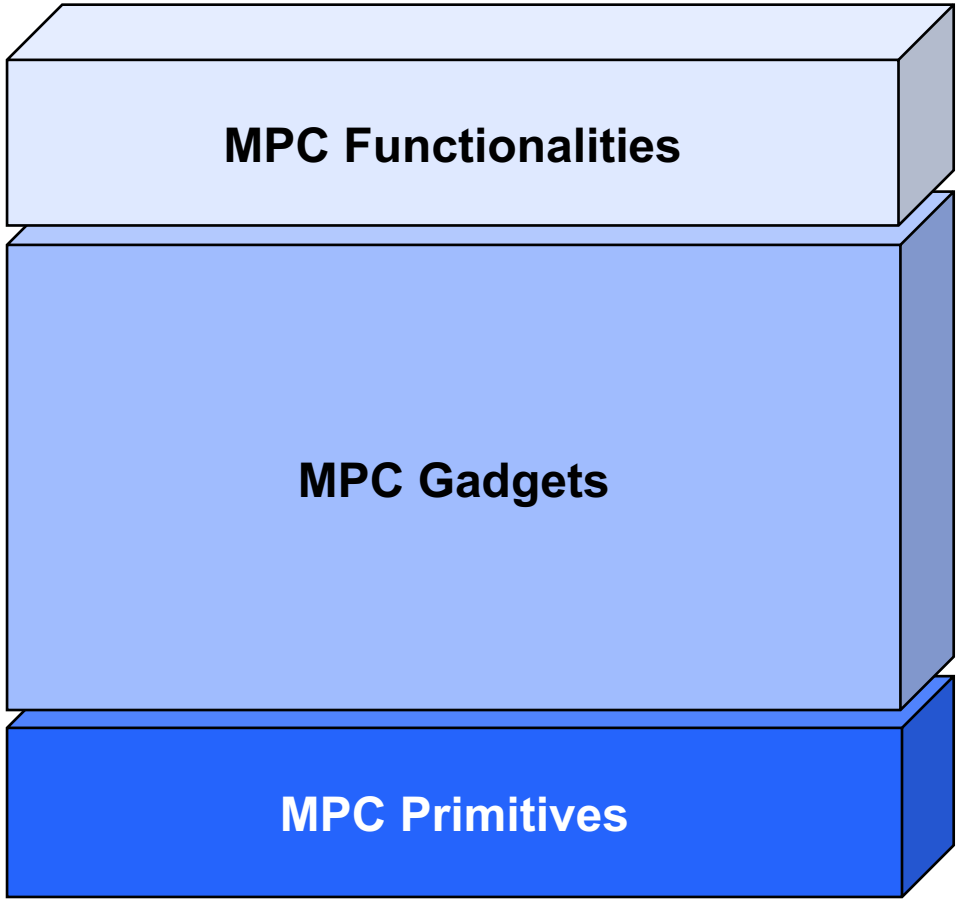
Rapid Assembly of MPC Protocols (RAMP) Framework

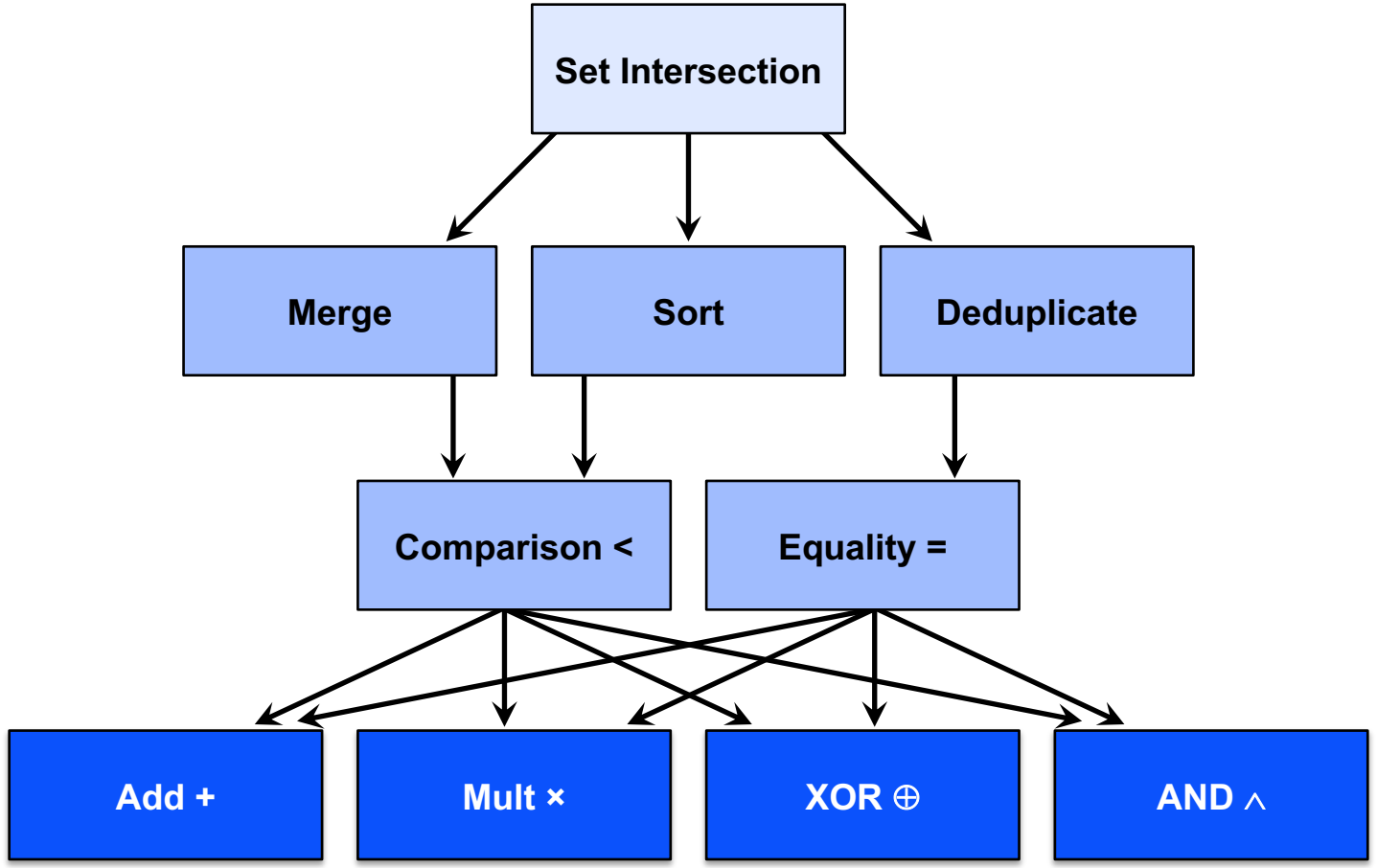
RAMP



Lincoln's RAMP framework enables rapid prototyping of MPC functionalities

RAMP Gadget Approach



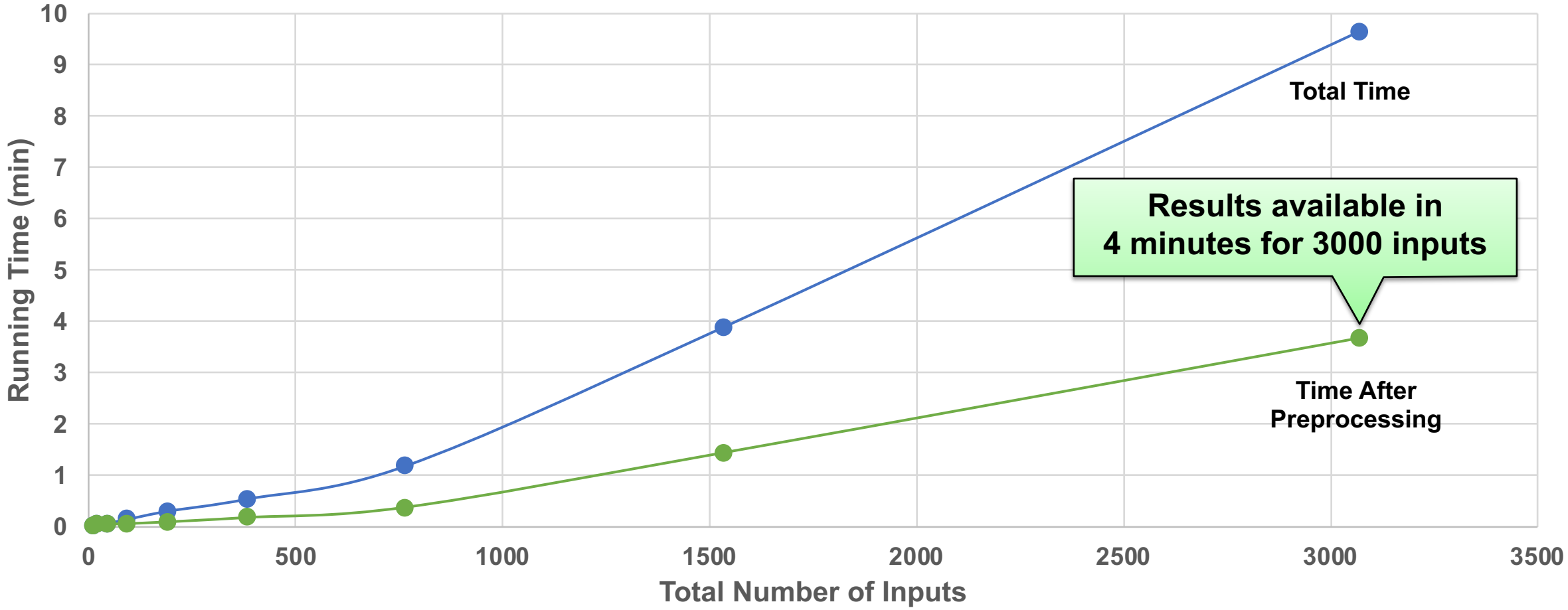


Implementation of MPC for Set Intersection

```
template <class T>
std::vector<T> setIntersection(std::shared_ptr<Context> context,
                               const std::vector<std::vector<T>>& input_lists) {
  auto merged = merge(context, input_lists);
  auto deduplicated = thresholdDeduplicate(context, merged, input_lists.size());
  return sort(context, deduplicated);
}
```

Gadgets abstract thousands of lines of code to make MPC functionalities easy to prototype

Example Performance of MPC for Set Intersection



RAMP Gadget Library

Arithmetic

```
arithmeticSum
arithmeticSumOfList
cumulativeSum
exponentiate
popCount
product
productVectorDifferentSizes
singleRoundAdditionHelper
subtract (secret-secret)
subtract (public-secret)
subtract (secret-public)
sum
```

Boolean operators

```
bgwNot
bgwOr
bgwXor (secret-secret)
bgwXor (public-secret)
bitmaskVectorAnd (bool)
bitmaskVectorAnd (arith)
bitmaskVectorOr (bool)
bitmaskVectorOr (arith)
bitwiseToSingleShare
leastSignificantBit
linearPrefixOr
mux (bool)
mux (arith)
```

Fixed-point arithmetic

```
fixedPointAdd
fixedPointDiv
fixedPointMult
```

Equality

```
equality (bool)
equality (arith)
equalityUsingRandomBitwiseShares
fermatEquality
```

Comparison

```
bitwiseSharesLessThanModulus
bitwiseSharesLessThanPublicValues
comparator
greaterThan (bool)
greaterThan (arith)
greaterThanThreshold
greaterThanWithinHalfField
lessThan (bool)
lessThan (arith)
lessThanWithinHalfField
linearGreaterThan
maskWithBitwiseRandomSharesAndOpen
max
maxWithinHalfField
min
minWithinHalfField
publicValuesLessThanBitwiseShares
treewiseGreaterThan
```

Sort

```
mergeVectorsShortestFirst
mergeVectorsTreePattern
oddEvenMerge
oddEvenMergeSort
sort
sortPairsByFirstElement
```

Set operations

```
deduplicate (bool)
deduplicate (arith)
distinctCountFromIndicators (bool)
distinctCountFromIndicators (arith)
distinctCountOfSorted
setIntersectionCardinality (bool)
setIntersectionCardinality (arith)
thresholdDeduplicate (bool)
thresholdDeduplicate (arith)
thresholdSetIntersection
thresholdSetIntersectionCompaction
vectorizedThresholdVectorEquality
```

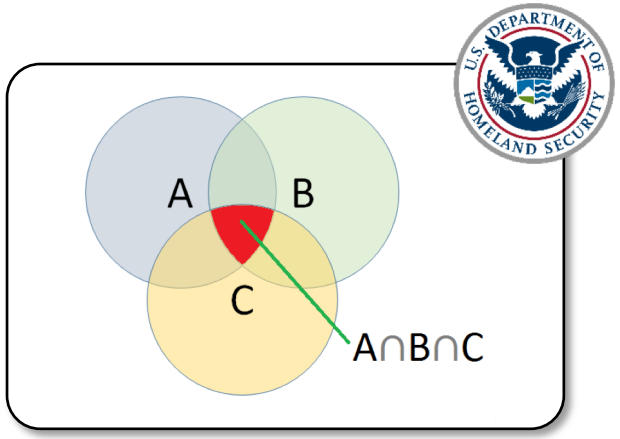
Linear algebra

```
innerProduct
outerProduct
matMult (secret-secret)
matMult (public-secret)
matMult (secret-public)
```

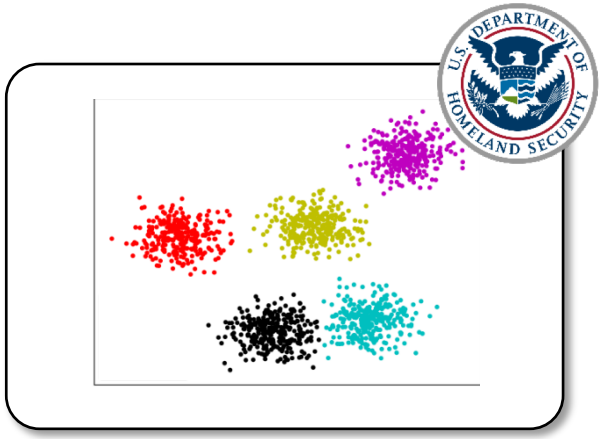
Outline

- Motivation
- Secure Multi-Party Computation (MPC)
- Lincoln MPC Framework
- Prototypes and Deployments
- Summary

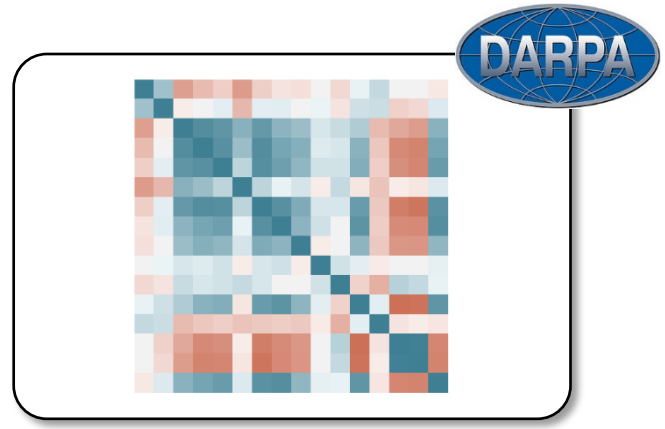
MPC Applications Prototyped



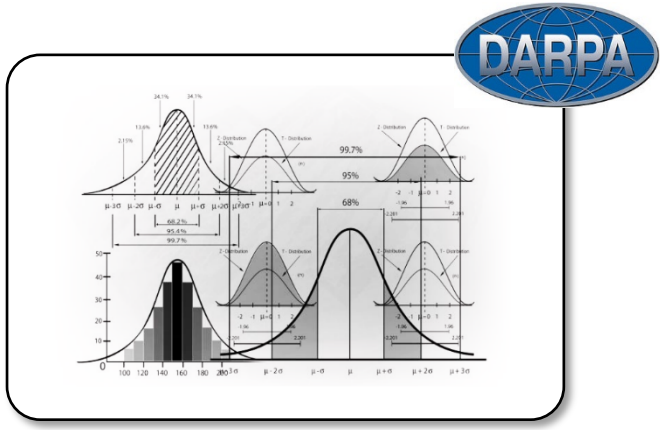
Joining IP blacklists



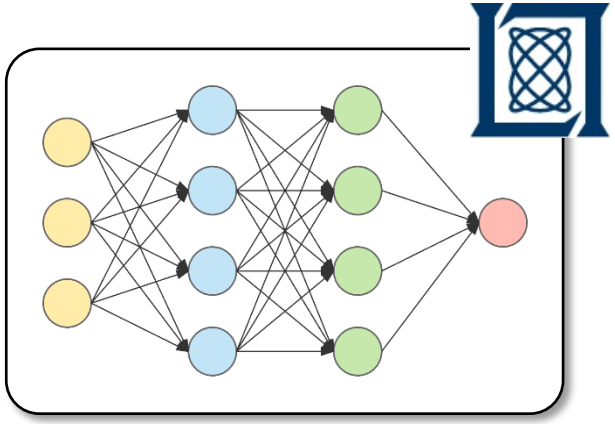
Similarity-based clustering



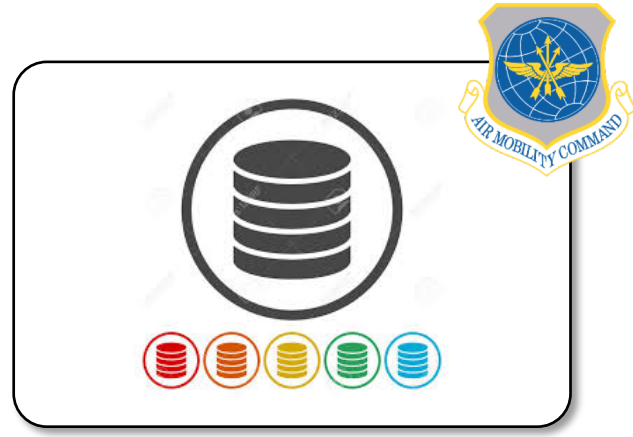
Detecting coordinated attacks



Computing statistical summaries



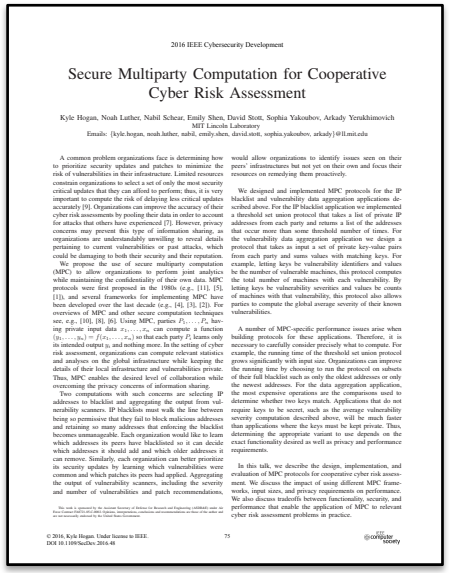
Training neural networks for genomic computation



Merging personnel datasets

Publications and Presentations

IEEE SecDev



2016 IEEE Cybersecurity Development

Secure Multiparty Computation for Cooperative Cyber Risk Assessment

Kyle Hogan, Noah Luther, Nabil Schear, Emilly Shen, David Stott, Sophia Yakoubov, Arkady Yerukhimovich
MIT Lincoln Laboratory
Emails: {kyle.hogan, noah.luther, nabil, emilly.shen, david.stott, sophia.yakoubov, arkady}@ll.mit.edu

A common problem organizations face is determining how to prioritize security updates and patches to minimize the risk of vulnerabilities in their infrastructure. Limited resources constrain organizations to select a set of only the most security critical updates that they can afford to perform; thus, it is very important to compute the risk of delaying less critical updates accurately [9]. Organizations can improve the accuracy of their cyber risk assessments by pooling their data in order to account for attacks that others have experienced [7]. However, privacy concerns may prevent this type of information sharing, as organizations are understandably unwilling to reveal details pertaining to current vulnerabilities or past attacks, which could be damaging to both their security and their reputation.

We propose the use of secure multiparty computation (MPC) to allow organizations to perform joint analytics while maintaining the confidentiality of their own data. MPC protocols were first proposed in the 1980s (e.g., [11], [5], [1]), and several frameworks for implementing MPC have been developed over the last decade (e.g., [4], [3], [2]). For overviews of MPC and other secure computation techniques see, e.g., [10], [8], [6]. Using MPC, parties $P_1, \ldots, P_n$ having private input data $x_1, \ldots, x_n$ can compute a function $(y_1, \ldots, y_n) = f(x_1, \ldots, x_n)$ so that each party P_i learns only its intended output y_i and nothing more. In the setting of cyber risk assessment, organizations can compute relevant statistics and analyses on the global infrastructure while keeping the details of their local infrastructure and vulnerabilities private. Thus, MPC enables the desired level of collaboration while overcoming the privacy concerns of information sharing.

Two computations with such concerns are selecting IP addresses to blacklist and aggregating the output from vulnerability scanners. IP blacklists must walk the line between being so permissive that they fail to block malicious addresses and retaining so many addresses that enforcing the blacklist becomes unmanageable. Each organization would like to learn which addresses its peers have blacklisted so it can decide which addresses it should add and which older addresses it can remove. Similarly, each organization can better prioritize its security updates by learning which vulnerabilities were common and which patches its peers had applied. Aggregating the output of vulnerability scanners, including the severity and number of vulnerabilities and patch recommendations, would allow organizations to identify issues seen on their peers' infrastructures but not yet on their own and focus their resources on remedying them proactively.

We designed and implemented MPC protocols for the IP blacklist and vulnerability data aggregation applications described above. For the IP blacklist application we implemented a threshold set union protocol that takes a list of private IP addresses from each party and returns a list of the addresses that occur more than some threshold number of times. For the vulnerability data aggregation application we design a protocol that takes as input a set of private key-value pairs from each party and sums values with matching keys. For example, letting keys be vulnerability identifiers and values be the number of vulnerable machines, this protocol computes the total number of machines with each vulnerability. By letting keys be vulnerability severities and values be counts of machines with that vulnerability, this protocol also allows parties to compute the global average severity of their known vulnerabilities.

A number of MPC-specific performance issues arise when building protocols for these applications. Therefore, it is necessary to carefully consider precisely what to compute. For example, the running time of the threshold set union protocol grows significantly with input size. Organizations can improve the running time by choosing to run the protocol on subsets of their full blacklist such as only the oldest addresses or only the newest addresses. For the data aggregation application, the most expensive operations are the comparisons used to determine whether two keys match. Applications that do not require keys to be secret, such as the average vulnerability severity computation described above, will be much faster than applications where the keys must be kept private. Thus, determining the appropriate variant to use depends on the exact functionality desired as well as privacy and performance requirements.

In this talk, we describe the design, implementation, and evaluation of MPC protocols for cooperative cyber risk assessment. We discuss the impact of using different MPC frameworks, input sizes, and privacy requirements on performance. We also discuss tradeoffs between functionality, security, and performance that enable the application of MPC to relevant cyber risk assessment problems in practice.

© 2016, Kyle Hogan. Under license to IEEE.
DOI 10.1109/SecDev.2016.48

IEEE HPEC

A Survey of Cryptographic Approaches to Securing Big-Data Analytics in the Cloud

Sophia Yakoubov, Vijay Gadepally, Nabil Schear, Emilly Shen, Arkady Yerukhimovich
MIT Lincoln Laboratory
Lexington, MA 02421
{sophia.yakoubov, vijayg, nabil, emilly.shen, arkady}@ll.mit.edu

***Abstract*—The growing demand for cloud computing motivates the need to study the security of data received, stored, processed, and transmitted by a cloud. In this paper, we present a framework for such a study. We introduce a cloud computing model that captures a rich class of big-data use-cases and allows reasoning about relevant threats and security goals. We then survey three cryptographic techniques – homomorphic encryption, verifiable computation, and multi-party computation – that can be used to achieve these goals. We describe the cryptographic techniques in the context of our cloud model and highlight the differences in performance cost associated with each.**

I. Introduction

In today's data-centric world, big-data processing and analytics have become critical to most enterprise and government applications. Thus, there is a need for an appropriate big-data infrastructure that supports storage and processing on a massive scale.

Cloud computing has become the tool of choice for big-data processing and analytics due to its reduced cost, broad network access, elasticity, resource pooling, and measured service [1]. Cloud computing enables consumers to store and analyze their data using shared computing resources while easily handling fluctuations in the volume and velocity of the data. However, cloud computing comes with risks. The shared compute infrastructure introduces many security concerns not present in more traditional computing architectures. The cloud provider and tenants may be untrusted entities who try to tamper with data storage or computation. These concerns motivate the need for a novel framework for analyzing cloud computing security, as well as for the use of cryptographic tools to address cloud computing security goals. In this paper, we propose a computation model of the cloud for big-data applications, and survey existing cryptographic tools using this model.

Our primary contributions are:

- A general computation model that captures a large class of big-data use-cases in the cloud,
- A description of relevant security threats, and
- A survey of cryptographic tools that address these security threats, and their current performance overhead.

The remainder of this paper is organized as follows: In Section II, we present a model of big-data analytics in the cloud and introduce genomic sequencing as a motivating application. In Section III describes the security goals and the threat model. Then, in Section IV, we describe several cryptographic techniques that can be used to address these security goals in various cloud deployments. Finally, in Section V, we conclude and describe some directions for future research in the area of secure cloud computation.

II. A Model for Big-Data Analytics in the Cloud

In this section, we present a general computational model of the cloud that allows reasoning about a wide variety of big-data analytics applications. In this model, we categorize cloud compute nodes by their roles in the big-data analytics pipeline. Extending the notation of Bogdanov et al. [2], we define the following types of nodes:

- **I** denotes an *input node* supplying raw data for the application. Input nodes include sensors capturing data and machines used to enter client data.
- **C** denotes a *compute node* whose role is to perform the computation for the application. Compute nodes include *ingestion* nodes, which refine the input data to get it ready for analysis, and *enrichment* nodes, which perform the actual analysis.
- **S** denotes a *storage node* whose role is to store data between computations. Storage nodes store both the original inputs and the computation outputs.
- **R** denotes a *result node* which receives the output of some computation, and either makes automated decisions based on that output or conveys the output to a client.

Additionally, X^+ (where $X \in \{\mathbf{I}, \mathbf{C}, \mathbf{S}, \mathbf{R}\}$) denotes a set of one or more (possibly communicating) nodes of type X.

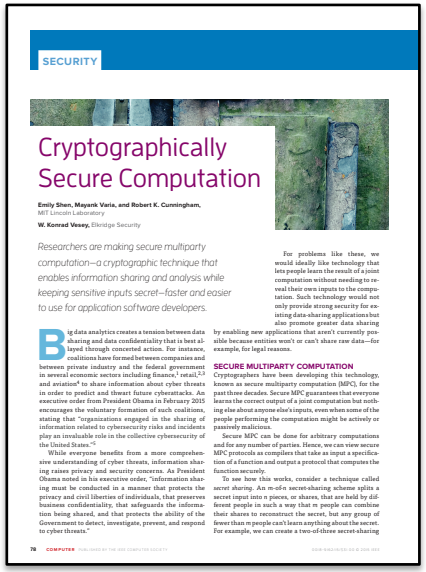
Figure 1 depicts a cloud architecture for big-data analytics using the above terminology. This model can be used to describe a wide variety of big-data applications.

As an example, consider an application in which the cloud stores and correlates sensitive genomics data with the goal of identifying a specific biological sequence, as described by Kepner et al. [3]. In this scenario, the cloud stores and correlates billions of reference genomic sequences. An agency

This work is sponsored by the Assistant Secretary of Defense for Research and Engineering (ASDR&E) under Air Force Contract #FA8721-05-C-0002. Opinions, interpretations, recommendations and conclusions are those of the authors and are not necessarily endorsed by the United States Government.

978-1-4799-6233-4/14/$31.00 ©2014 IEEE

IEEE Computer Magazine



SECURITY

Cryptographically Secure Computation

Emily Shen, Mayank Varia, and Robert K. Cunningham,
MIT Lincoln Laboratory
W. Konrad Vesey, Elkridge Security

Researchers are making secure multiparty computation—a cryptographic technique that enables information sharing and analysis while keeping sensitive inputs secret—faster and easier to use for application software developers.

Big data analytics creates a tension between data sharing and data confidentiality that is best allayed through concerted action. For instance, coalitions have formed between companies and between private industry and the federal government in several economic sectors including finance,[1] retail,[2,3] and aviation[4] to share information about cyber threats in order to predict and thwart future cyberattacks. An executive order from President Obama in February 2015 encourages the voluntary formation of such coalitions, stating that "organizations engaged in the sharing of information related to cybersecurity risks and incidents play an invaluable role in the collective cybersecurity of the United States."[5]

While everyone benefits from a more comprehensive understanding of cyber threats, information sharing raises privacy and security concerns. As President Obama noted in his executive order, "information sharing must be conducted in a manner that protects the privacy and civil liberties of individuals, that preserves business confidentiality, that safeguards the information being shared, and that protects the ability of the Government to detect, investigate, prevent, and respond to cyber threats."

For problems like these, we would ideally like technology that lets people learn the result of a joint computation without needing to reveal their own inputs to the computation. Such technology would not only provide strong security for existing data-sharing applications but also promote greater data sharing by enabling new applications that aren't currently possible because entities won't or can't share raw data—for example, for legal reasons.

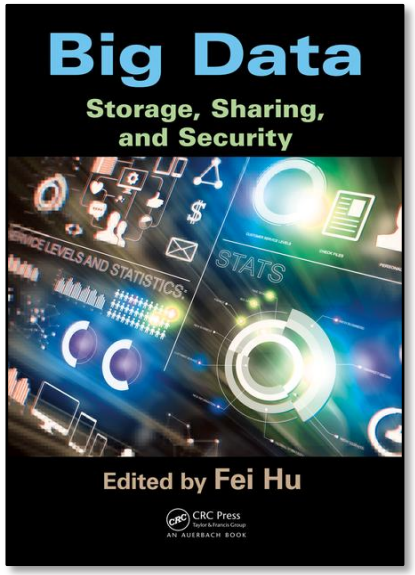
SECURE MULTIPARTY COMPUTATION

Cryptographers have been developing this technology, known as secure multiparty computation (MPC), for the past three decades. Secure MPC guarantees that everyone learns the correct output of a joint computation but nothing else about anyone else's inputs, even when some of the people performing the computation might be actively or passively malicious.

Secure MPC can be done for arbitrary computations and for any number of parties. Hence, we can view secure MPC protocols as compilers that take as input a specification of a function and output a protocol that computes the function securely.

To see how this works, consider a technique called *secret sharing*. An m-of-n secret-sharing scheme splits a secret input into n pieces, or shares, that are held by different people in such a way that m people can combine their shares to reconstruct the secret, but any group of fewer than m people can't learn anything about the secret. For example, we can create a two-of-three secret-sharing

Book Chapter: Cryptography for Big Data Security



DHS S&T Cybersecurity Showcase

Summary

- Secure multi-party computation enables privacy-preserving collaboration
- MPC can securely perform any computation in theory
- Lincoln has developed a rapid prototyping framework to address practical MPC development challenges
- Applying MPC to cyber security and other mission areas
- Ongoing work: exploring additional applications, improving performance and usability, transitioning framework

Contact Information

Dr. Emily Shen
emily.shen@ll.mit.edu