Introduction

Daniel Huynh

- Graduate from Polytechnique (X2016) and X-HEC Data Science for Business
- Former intern at Microsoft France in the office of the CTO
- CEO of Mithril Security, a software solution to secure company external data sharing


MACHINE LEARNING
CRYPTOGRAPHY
PRIVACY PRESERVING ML
imgflip.com

Our agenda for this Common Track

Context and cloud data threads

Core mitigation techniques

- Confidential Computing (CC)
- Homomorphic Encryption (HE)
- Secure Multi-Party Computing (MPC)

Other techniques to consider (if time permits)

- Differential Privacy (DP)

Context and cloud data threads

Data breaches need to be taken seriously

LifeLabs cyberattack one of 'several wake-up calls' for e-health security and privacy
December 19, 2019

"What's interesting about the large push for electronic patient health-care information that you put online is that a lot of these organizations are not designed to withstand attacks."

Canada's Desjardins reports costs of $53M related to data breach
August 13, 2019

"Unauthorized use of internal data by an employee led to breach of personal information including social insurance number, address and details of banking habits."

Anthem is warning consumers about its huge data breach. Here's a translation.
March 6, 2015

"The indications are that they gained access to Anthem's data by stealing the network credentials of at least five employees with high-level IT access."

Data breaches are costly

7.27% decrease
Is the average price of a company disclosing a data breach[1]

$3.92M avg.
The Cost of a data breach to an enterprise. (Up 12% in 5 years).[2]

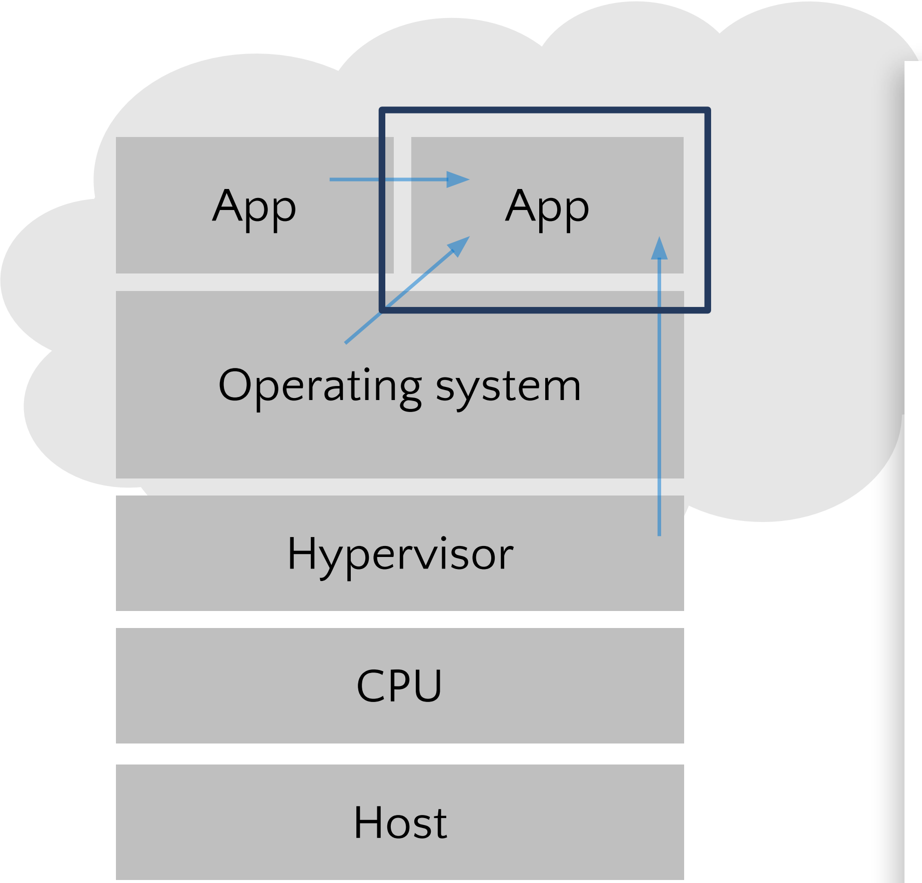
$6.5M avg.
The healthcare cost of a breach (over 60% more than other industries in the study).[2]

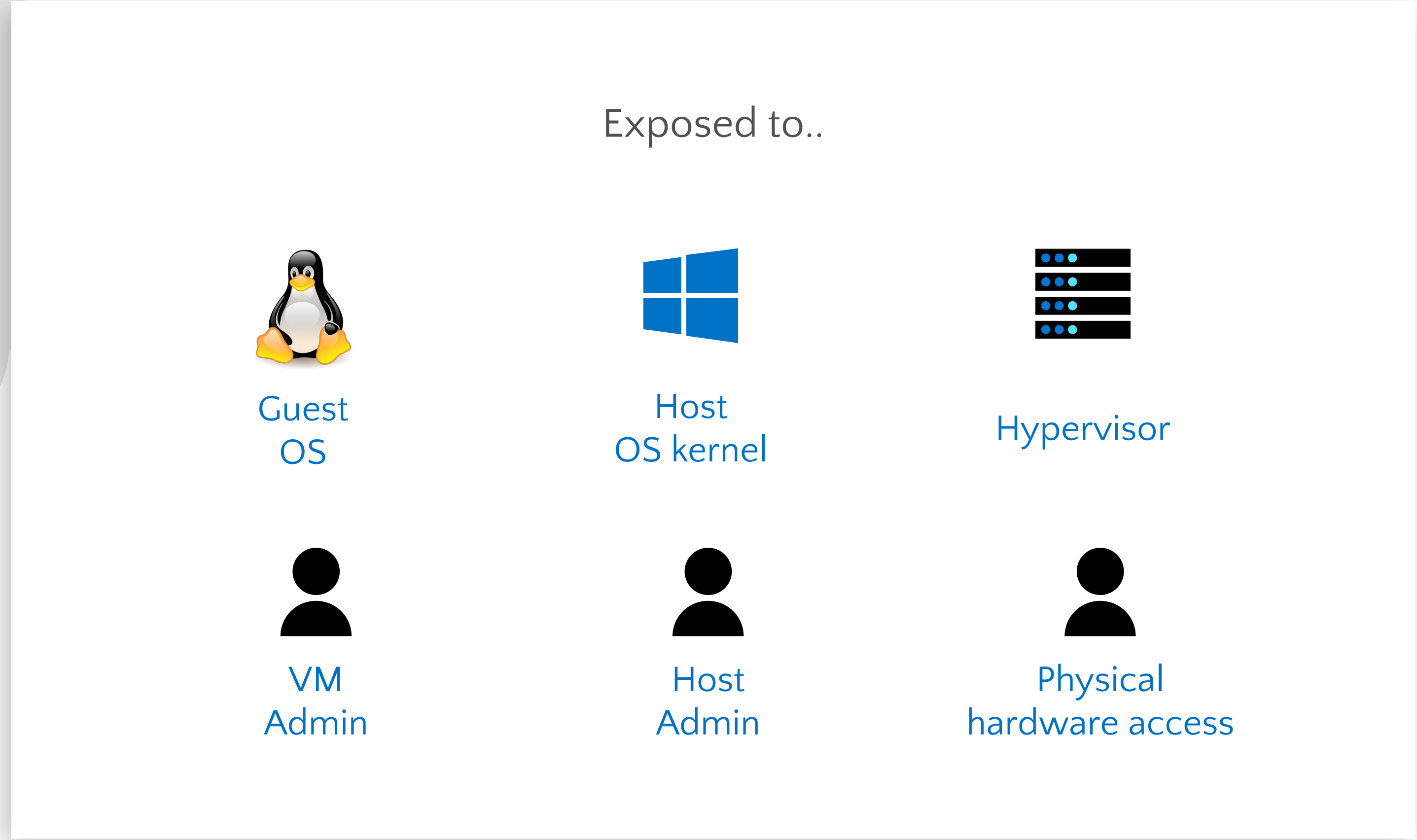
$42M
Projected in losses for 1 million records losses to cost[2]

$388M
Projected losses to cost >50 million records[2]

Sources: 1. Comparitech; 2. IBM

Why do we need to also protect "data in use"?





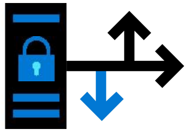
Confidential Computing (CC)

For your security, regulatory, and compliance needs

The ability to store, transport, and act on compute workloads without compromising privacy of data and intellectual property

How to control the (most sensitive) data throughout its complete lifecycle?

Confidential Computing technologies are helping us evolve from computing in the clear to protecting data while in use, reducing the need for trust in HW/SW stacks & operators

Existing Encryption		New
		
Data at rest	**Data in transit**	**In use**
Encrypt inactive data when stored in blob storage, database, etc.	Encrypt data that is flowing between untrusted public or private networks	Protect/Encrypt data that is in use, while in RAM and during computation
The industry moved from disks in the clear to encrypted disks, with managed keys	Evolved from browsing/moving data in the clear (HTTP), to encrypting data (HTTPS / TLS)	Evolving from computing in the clear to Trusted Execution Environments (TEEs), like Intel SGX

Who do you need to trust?

App software vendor

Trusted today on-premises
Use open source or build own software.

Infrastructure provider

Trust a cloud provider, or
Manage own on-premises datacenters.

Hardware vendor

Trusted today on-premises
Build own hardware.

Attack surface

Application runtime & dependencies, many open source packages

Other tenants running on the host, infrastructure services

Observable side channels in hardware

Software TCB

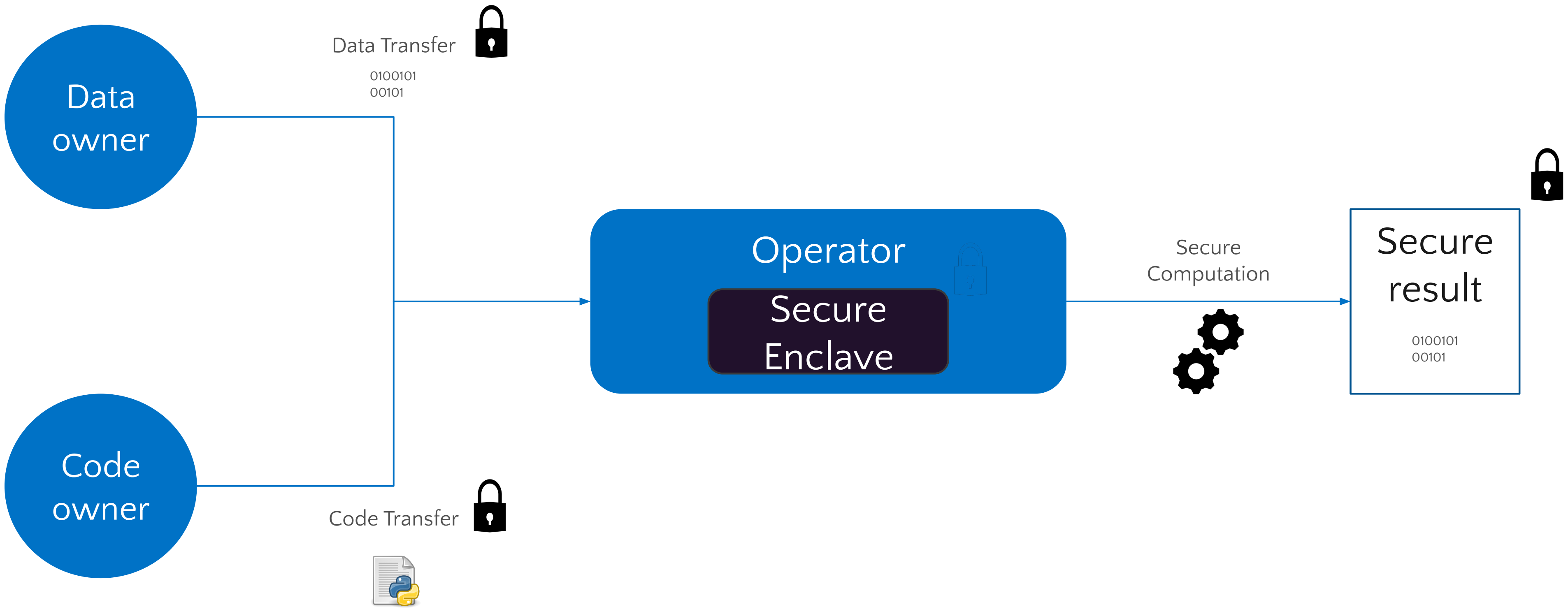
- Application
- VM OS
- Hypervisor
- Host OS
- Hardware

Trusted actors

- App Admin
- VM Admin
- Host Admin
- Physical hardware access

High level view of Confidential Computing with Intel SGX

Confidential computing allows execution of **code** we do not see on **data** we do not see



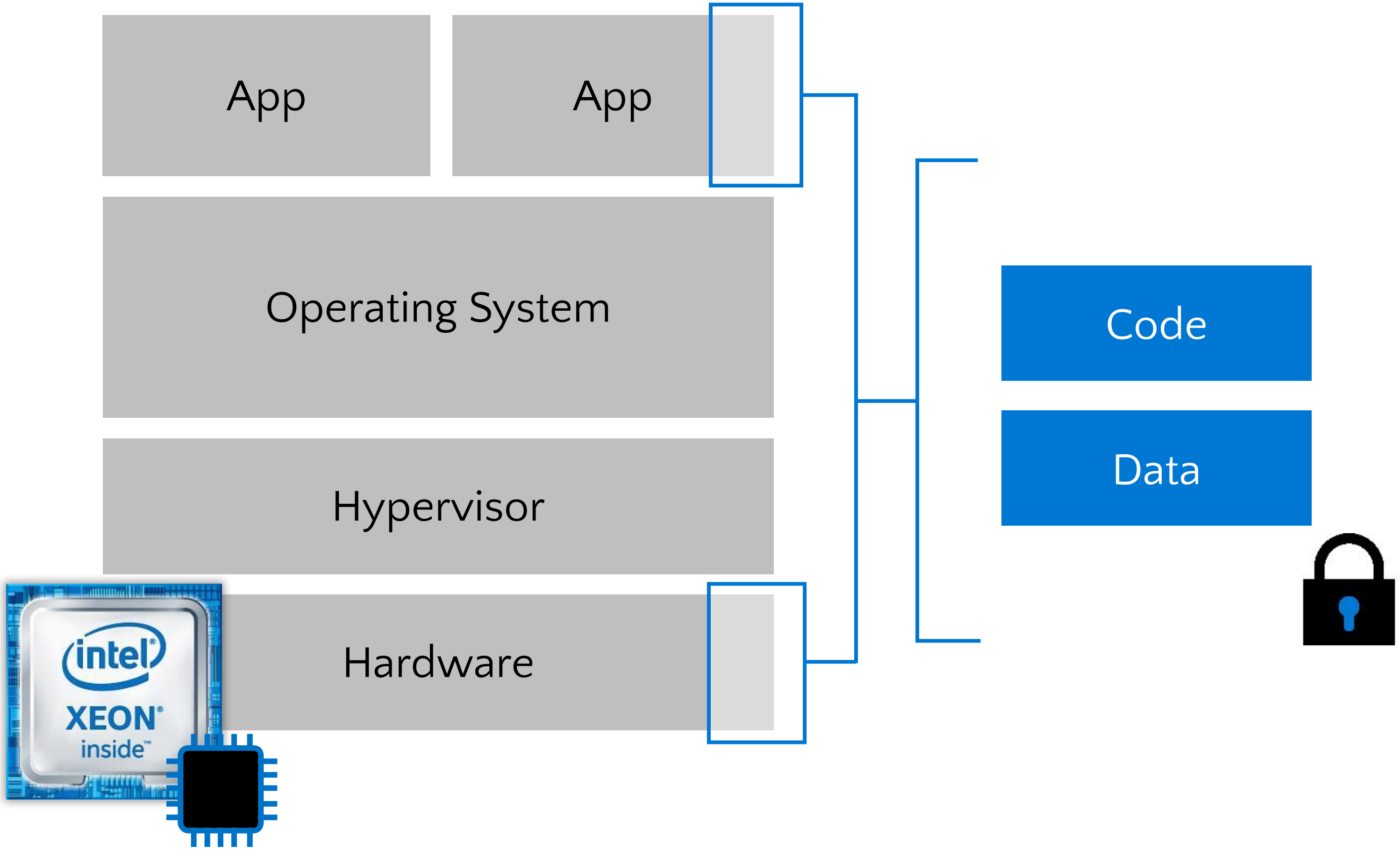
How Intel SGX hardware protects data in use

Intel SGX Goal: Minimize attack surface to CPU

New hardware architecture

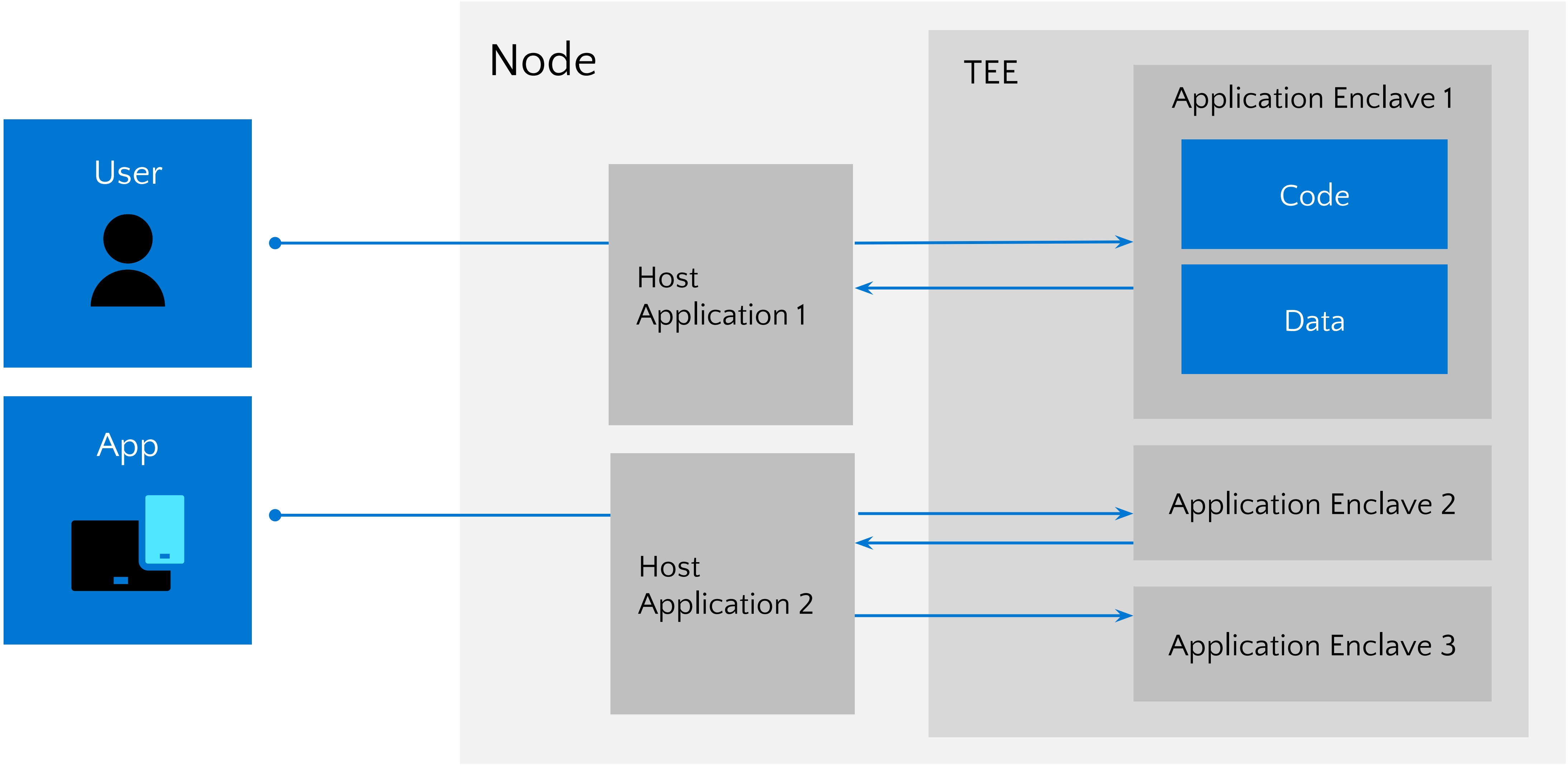
New instructions to set aside private regions ("protected containers") of code and data

Data is only ever in the clear within the protected memory and encrypted if pushed off of CPU



An enclave-based application model

Development experience and software

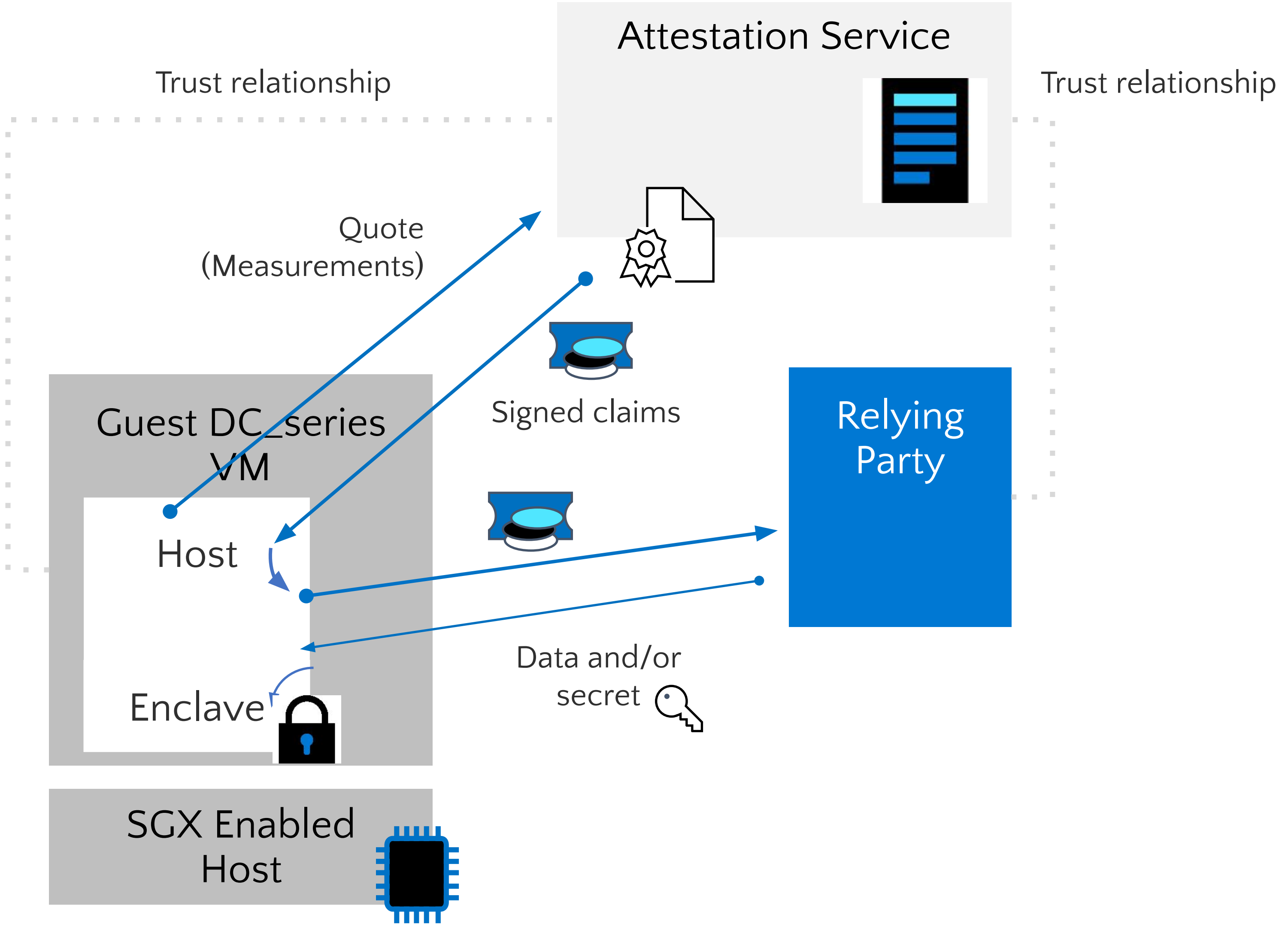


Azure Attestation Service

Azure Confidential Computing

Service for Confidential Computing

1. Quote provide proofs:
   - Code runs in genuine SGX enclave
   - Enclave version and owner is as expected
   - Arbitrary enclave supplied data is as expected
2. Attestation service attests to hardware, rooted in Intel chain of trust, and issues signed claims as a security token
3. Relying party, e.g. another enclave, a cloud service, etc. is presented with security token with certs chained to CPU
4. Security token is used to release data and/or secrets to the application enclave



Confidential computing can help

Use cases

1

Make your applications more secure by leveraging confidential computing VMs and the Open Enclave SDK

2

Enable Customer workloads, confidential blockchain, secrets storage and processing, analytics, ML training and inferencing, datastores, IoT

3

Preserve privacy and sensitive customer data across organizations with multi-party dataset analytics and machine learning, both training and inference.

Three abstraction layers for enclaves

Development experience and software

New/Refactor

Understand fundamentals, create a trusted portion of your application to run inside the enclave/TEE with “the building blocks”

Open Enclave SDK (OE SDK)

Confidential Consortium Framework (CCF)

“Lift and shift” model

Application just works inside secure hardware with no/minimal changes

Secure containers/LibOS Strategy

Open source, such as SGX-LKL

Leverage one of the ISVs available in the Marketplace (e.g. Fortanix, Anjuna)

Confidential workloads using PaaS services

It’s taken care of by services already used

SQL Azure DB Always Encrypted

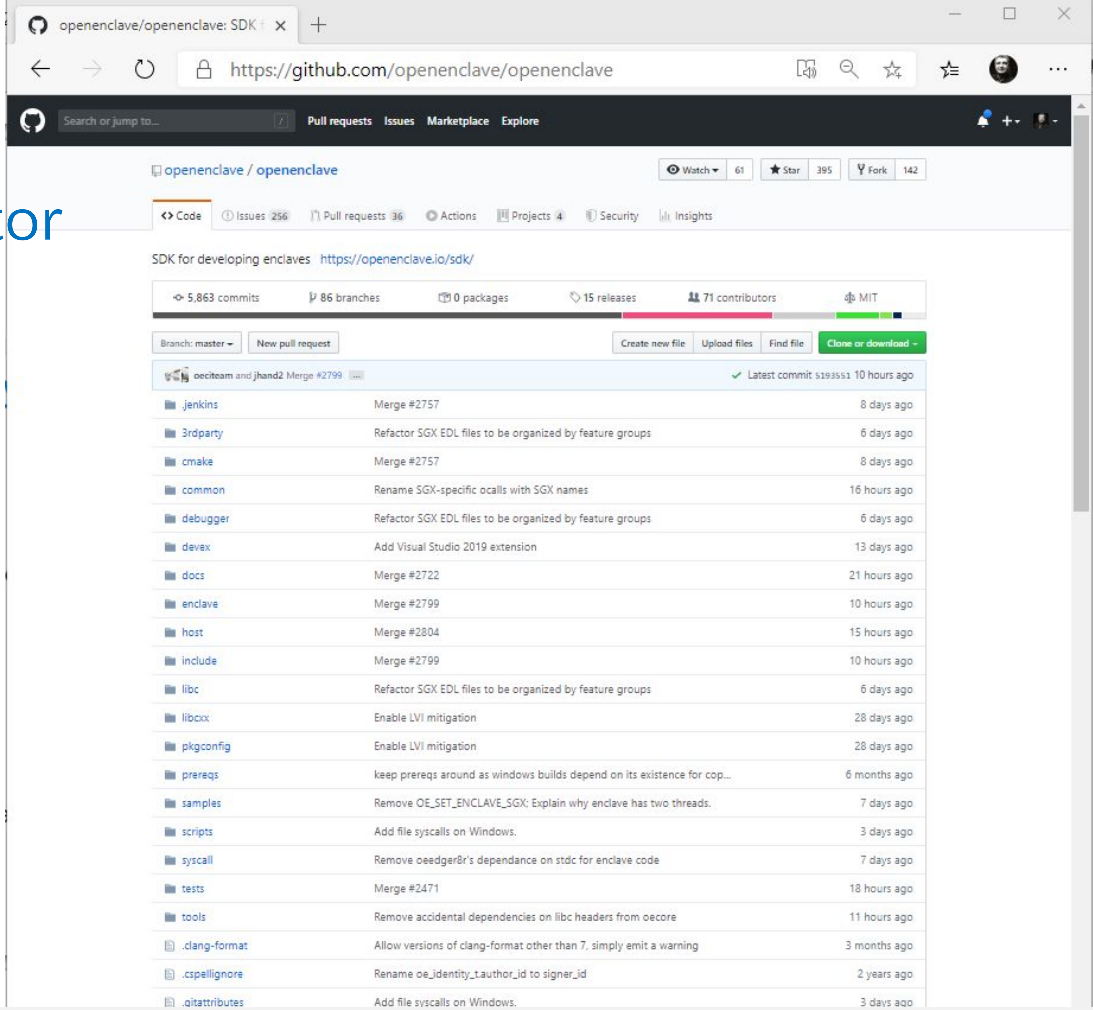
Confidential ML

Confidential AKS

Confidential Blockchain

Open Enclave SDK

Development experience and software: New/Refactor



Driving towards a consistent API surface around enclaving abstraction, supporting portability across enclave types and flexibility in architecture

Open sourced by Microsoft in October 2018 under the MIT license

https://github.com/openenclave/openenclave

More than 6,000 commits and 16 versions since the original release date

More than 70 contributors, 150 forks, 400 stars and 1,200 pull-requests

Very active project with 25-100 commits per week

Core functionalities

- Enclave creation and management
- Expressions of enclave measurement and identity
- Mechanisms for defining call-ins and call-outs
- and the data marshalling associated with them
- System primitives exposed by enclave runtime,
- such as thread and memory management
- Sealing functions
- Attestation functions
- Runtime and cryptographic library support

Extending support to:

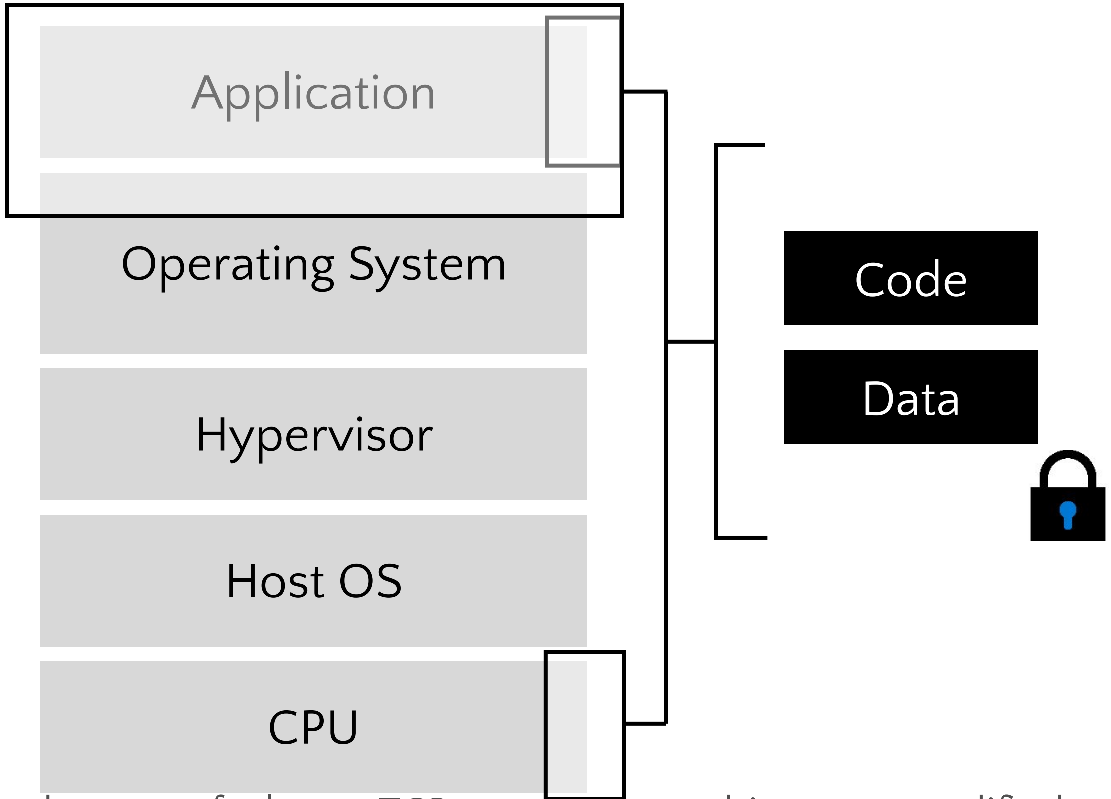
- Operating System: Linux, Windows, OP-TEE OS*
- Trusted Execution Environments: Intel SGX, Arm TrustZone
- Runtime Libraries: C/C++
- Cryptographic Libraries: mBedTLS

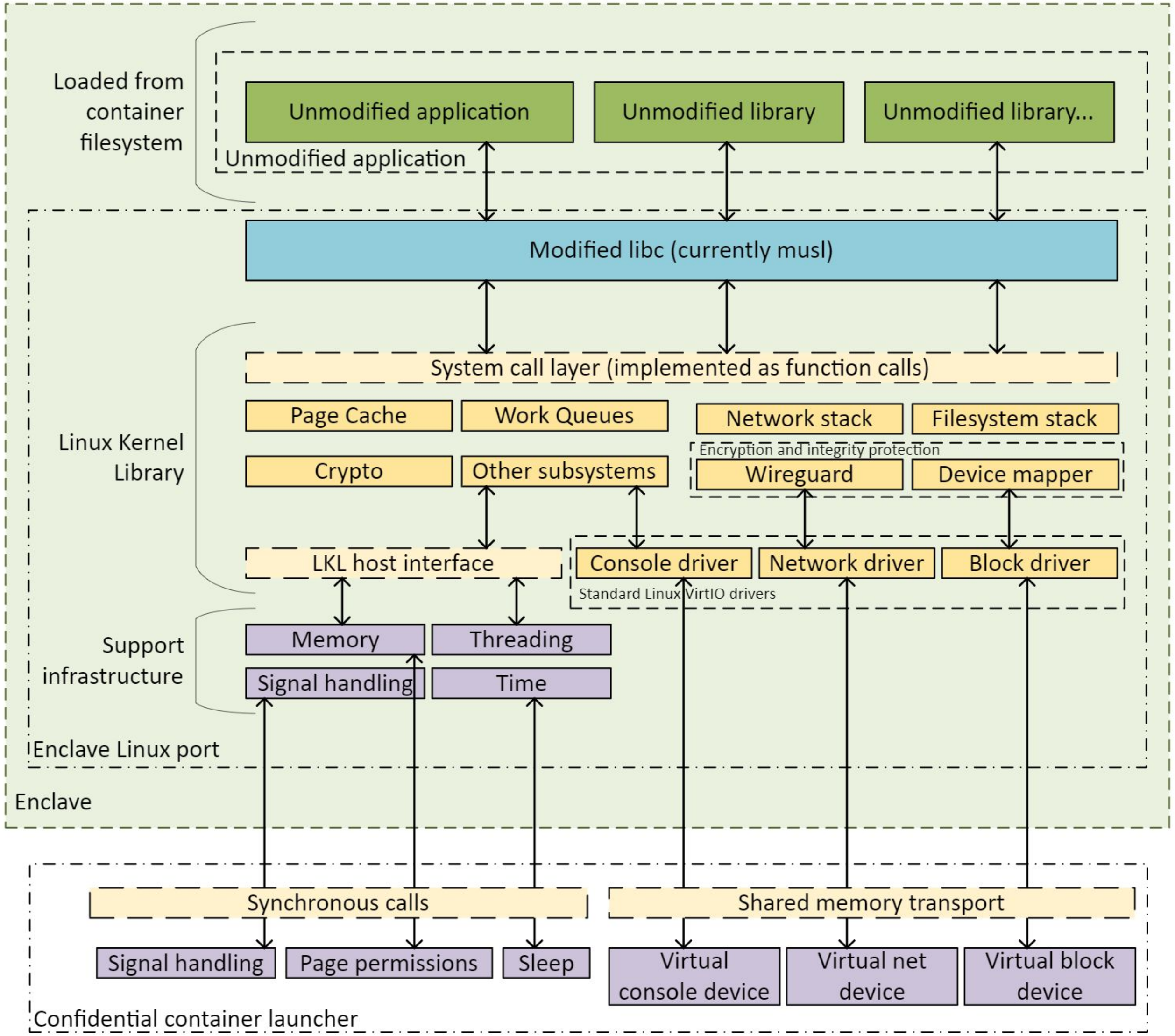
*OP-TEE OS: https://www.op-tee.org/

Make your code confidential without changing a line!

Development experience and software: “Lift and shift” model

Confidential Linux containers with SGX-LKL





At the cost of a larger TCB, we can run arbitrary unmodified applications, such as TensorFlow

https://github.com/lsds/sgx-lkl/tree/oe_port

Demo

Image classification with Graphene SGX

Use Confidential Computing with Kubernetes

Development experience and software

An additional layer of data protection for the Kubernetes workloads with the code running on the CPU with secure hardware enclaves

Using the Open Enclave SDK for confidential computing in code with ACC DC_series VMs

More info: Bringing confidential computing to Kubernetes

The oe-sgx device plugin (alpha) surfaces the usage of Intel SGX's Encrypted Page Cache (EPC) RAM as a schedulable resource for Kubernetes

Allow to schedule pods and containers to use the Open Enclave SDK, and to have access to a TEE by defining a limit on the specific EPC memory that is advertised to the Kubernetes scheduler by the device plugin

See Using SGX with Kubernetes (AKS_Engine)

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: oe-deployment
spec:
  selector:
    matchLabels:
      app: oe-app
  replicas: 1
  template:
    metadata:
      labels:
        app: oe-app
    spec:
      containers:
      - name: <image_name>
        image: <image_reference>
        command: <exec>
        resources:
          limits:
            openenclave.io/sgx_epc_MiB: 64
```

Multi-party machine learning use case

Goals

Data and Model/IP confidentiality from other parties and/or cloud provider

Control integrity of ML code running against data set from malicious parties

Solution

Run ML modelling process (data input, pre-processing, model training, etc.) in a trusted execution environments

Decrypt data only within a TEE to protect confidentiality at centralized point

Run code that is attestable at centralized point

Properties

Confidentiality and integrity of data & models (from other parties)
Transparency (only certain ML code can access the data)
Auditability (logs of activities)

Protocols

Agree on machine learning algorithm/code to use
Upload code
Verify enclave setup (hardware and software identity)
Send data and keys to system
Receive computed output

Multi-party machine learning:
https://www.microsoft.com/en-us/research/wp-content/uploads/2016/07/paper.pdf

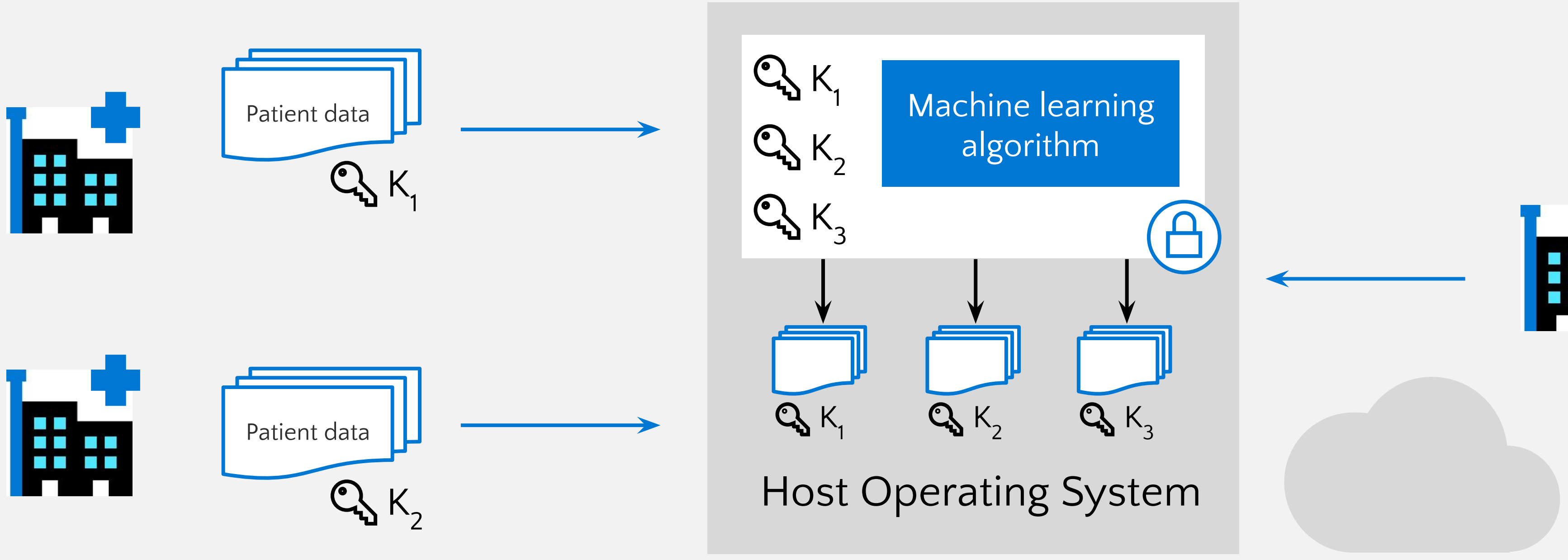
Healthcare scenarios

Development experience and software: Confidential workloads (using PaaS services)

Partnered health facilities contribute private patient health data sets to train a ML model

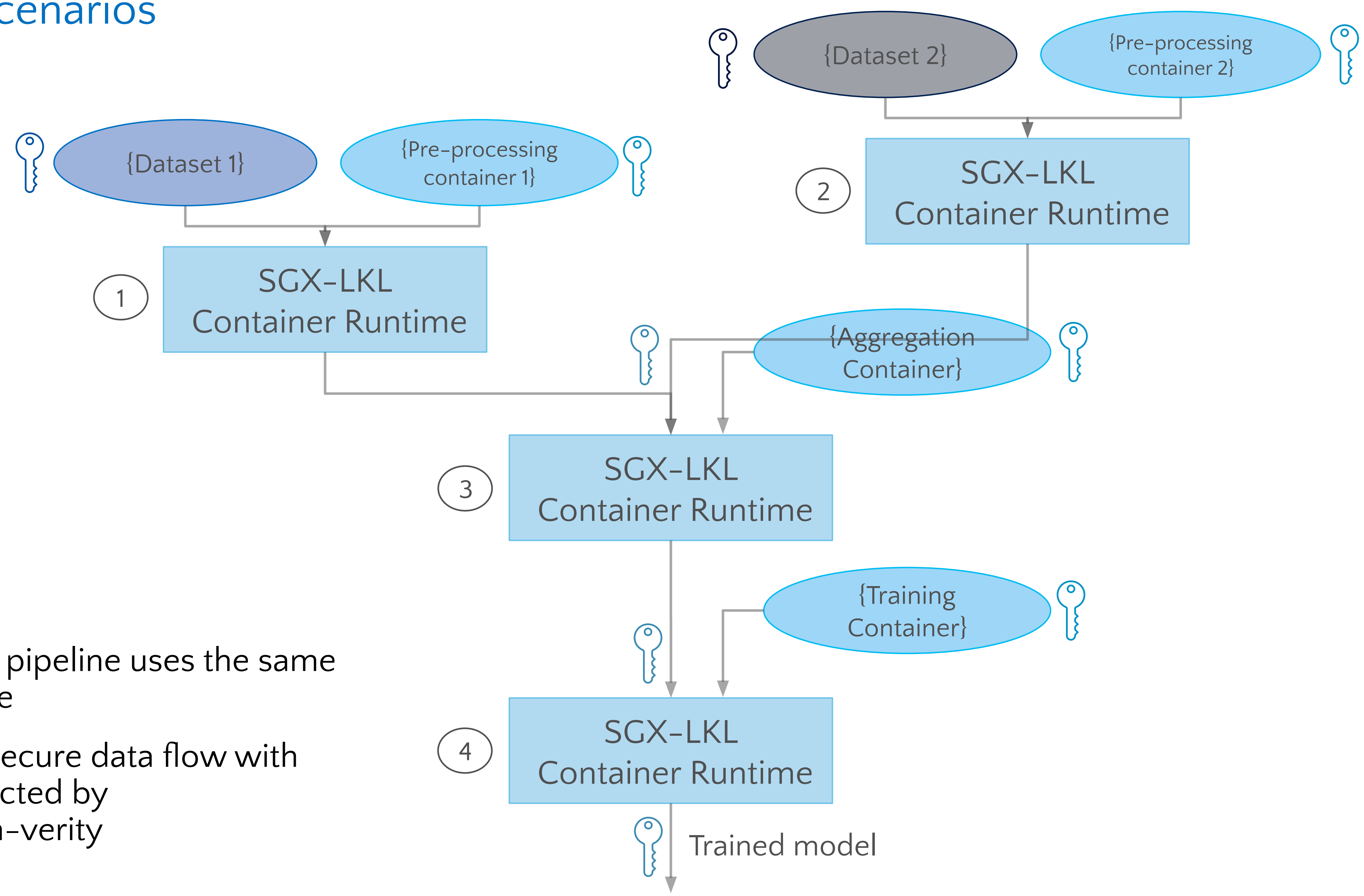
Each facility only sees their respective data sets (aka no one, not even cloud provider, can see all data or trained model, if necessary)

All facilities benefit from using trained model



Confidential ML pipelines for Multi Party machine learning

Healthcare scenarios



Every node of the pipeline uses the same LKL runtime image

Edges represent secure data flow with disk images protected by dm-crypt and dm-verity

To go beyond...

Azure confidential computing and Confidential ML links

Azure confidential computing solution page:
https://azure.microsoft.com/en-us/solutions/confidential-compute/

Azure confidential computing:
https://azure.microsoft.com/en-us/blog/azure-confidential-computing/

Ignite Mechanics Video with Mark Russinovich:
https://youtu.be/Qu6sP0XDMU8

Azure Confidential Compute (Virtual Machine):
https://azuremarketplace.microsoft.com/en-us/marketplace/apps/microsoft-azure-compute.acc-virtual-machine-v2?tab=Overview

Open Enclave SDK page:
https://openenclave.io/sdk/

Confidential ML:
https://aka.ms/AIShow/ConfidentialML

Responsible AI resources:
https://www.microsoft.com/en-us/ai/responsible-ai-resources

Azure responds to COVID-19:
https://azure.microsoft.com/en-us/blog/azure-responds-to-covid19/

To go beyond…

Confidential Computing @ Microsoft Research (MSR)

Reference material on Intel SGX use from

Multi-party machine learning:
https://www.microsoft.com/en-us/research/wp-content/uploads/2016/07/paper.pdf

SQL Server with Haven:
https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/osdi2014-haven.pdf

Map/reduce with VC3:
https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/vc3-oakland2015.pdf

Preventing enclave information leaks:
https://people.eecs.berkeley.edu/~rsinha/research/pubs/pldi2016.pdf

Using side-channel page faults to extract JPG images:
https://www.microsoft.com/en-us/research/wp-content/uploads/2017/06/atc17-final230.pdf

Homomorphic Encryption (HE)

Homomorphic Encryption (HE)

New type of encryption

Allows (un)limited computation on encrypted data

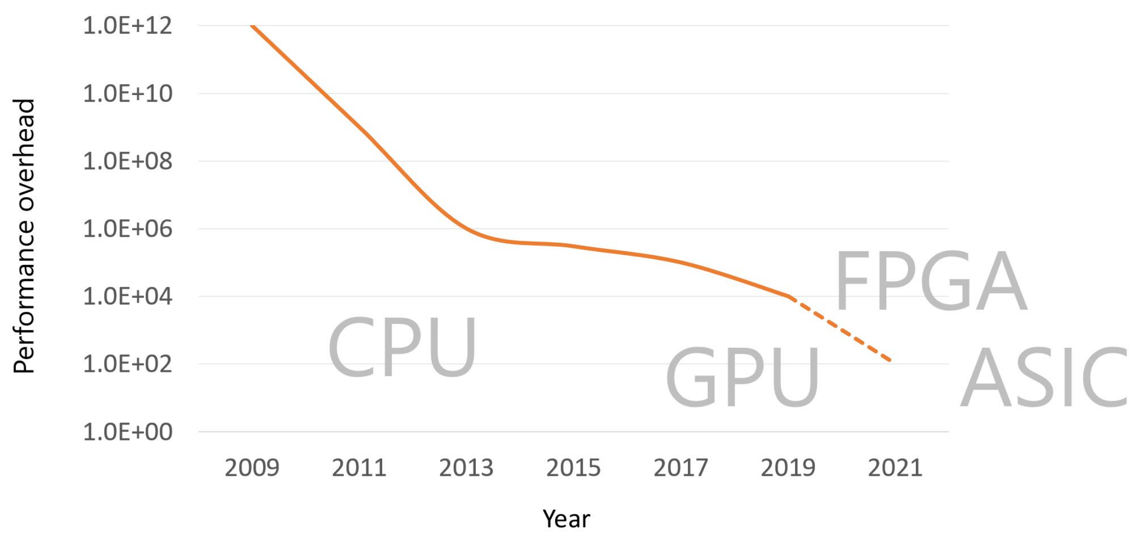
Enables outsourcing of data storage/processing, i.e. data enters / changes / leaves untrusted networks – always encrypted

- No information about the data is revealed to the untrusted network
- Results can only be revealed by data owner
- Think persistent cloud storage with computation (in Azure) : statistical computations, ML inference, secure search

Can be either public-key of private-key encryption

Post-quantum secure

Performance Overhead



"Full homomorphic" encryption

$D_K(C_K(\mathrm{n}) + C_K(\mathrm{m})) = \mathrm{n} + \mathrm{m}$

$D_K(C_K(\mathrm{n}) \times C_K(\mathrm{m})) = \mathrm{n} \times \mathrm{m}$

Some benefits

There is no hardware element in the TCB (so you don't have to trust hardware)

This type of encryption is applicable for some applications, but… its extra cost is still intolerable for general-purpose treatments

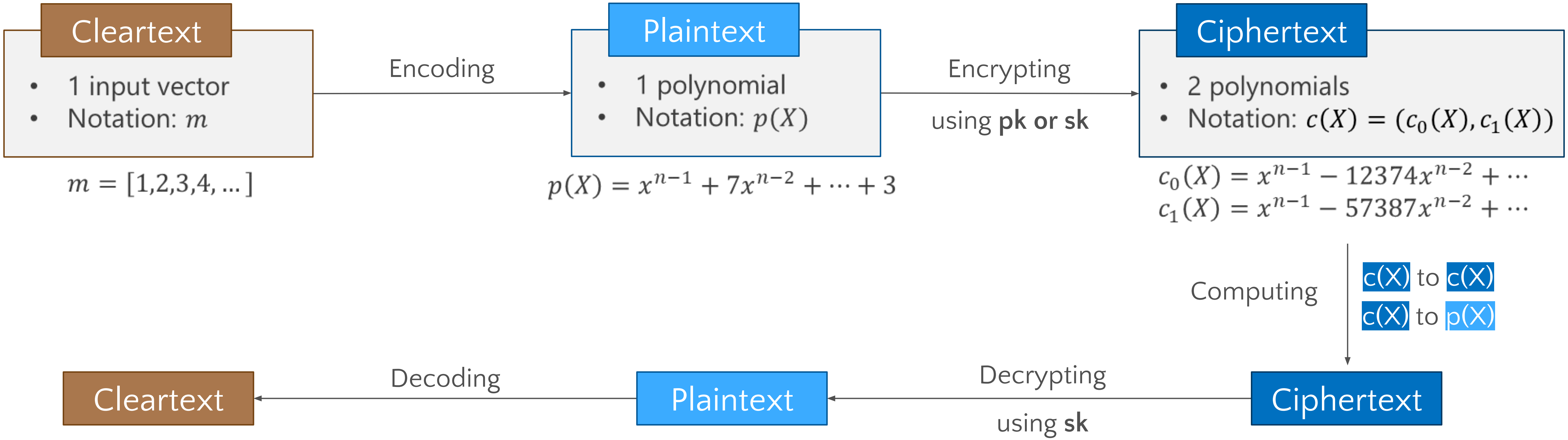
Invented in 2009 by Craig Gentry, @ Stanford university

Originally, 100 trillion times slower to make calculations on encrypted data than on unencrypted data

2 million times faster in 2013

Homomorphic Encryption

High level view



Notes & Key points

example

- Homomorphic operations operate with polynomials
- Microsoft SEAL can work both using
  - ✔ symmetric encryption (one secret key for encryption and decryption)
  - ✔ asymmetric encryption (one public key for encryption and one secret key for decryption
- The encoding operation is also a homomorphism

Homomorphic Encryption

Encoding and decoding with SEAL: the CKKS encoder

CKKS encoder

Principle

- **Input:** one vector of size
- **Principle:**
  - Encode by computing the Lagrangian polynomial interpolation on specific values (the roots of $X^n + 1$)
  - Decode by evaluating on the same values
- **Vector view:**

$$m = [m_0, m_1, \dots, m_{\frac{n}{2}}, m_{n/2}] \rightarrow \quad p(X) = [p_0, p_1, \dots, p_{n-1}]$$

Example

$$m = [3 + 4i, 2 - i] \quad \rightarrow \quad p(X) = 45X^3 + 160X^2 + 91X + 160$$

Another view

- **Considering:** $\mathrm{R} = (r_0, r_1, \dots, r_{n-1})$ the roots of $X^n + 1$
- **Principle:**

$$p(r_i) = \Delta m_i \quad \text{for} \quad \mathrm{i} \in \left[\!\left[0, {}^{n}/_{2}\right]\!\right]$$

- **Vector view:**

$$\text{with } p(X) = \begin{matrix} p_0 \\ p_1 \\ \dots \\ p_{\frac{n}{2}-1} \end{matrix}, \quad p(R) = \Delta m = \begin{matrix} \Delta m_0 \\ \Delta m_1 \\ \dots \\ \Delta m_{\frac{n}{2}-1} \end{matrix}$$

because $p(R) = [p(r_0), p(r_1), \dots, p(r_{\frac{n}{2}-1})]$

Notes & Key points

About the CKKS encoder

- You can embed **at most** ${}^{n}/_{2}$ numbers in one message
- m can contain **floats or complex numbers**
- Coefficients of $p(X)$ are computed **modulo Q**

About all encoders:

- $\boldsymbol{n}$ **is always a power of 2**
- $\boldsymbol{p(X)}$ **has always a degree of n-1**

Homomorphic Encryption

Encrypting and decrypting

Step 1

Key generation

- **Secret key:** s_k *(randomly chosen)*
- **Public key:** $p_k = (b, a)$
  - *a another sampled polynomial*
  - $b = -as_k + e$
  - *e some random noise*

Step 2

Encryption

Considering $c = (c_0, c_1)$

- $c_0 = b + p = -as_k + e + p$
- $c_1 = a$

Thus, a ciphertext is structured as followed:

Mask removable with s_k	e	p

$\log(Q)\ bits$

Step 3

Decryption

Considering p as the encoded message m,

$$\begin{cases} c_0 = -as_k + p + e \\ c_1 = a \end{cases}$$

so

$$c_0 + c_1 s_k = -as_k + e + p + as_k$$

$$\boxed{c_0 + c_1 s_k = e + p}$$

Notes & Key points

- Remember that s_k, a, b, c_0, c_1, p and e are **polynomials**
- **Decryption is not exact.** The requirement for a correct decryption is $e \ll p$
- Each computation **increases the noise**. Once it reaches a certain point, correct decryption will be impossible.
- The **security** of the encryption relies on the hardness of finding s_k from b : this is called the *Ring Learning With Errors* problem, and it is **quantum secure**.

Microsoft SEAL

Computation : Native operations cheat sheet

Scenarios

Addition

$c(X) \leftrightarrow c(X)$ or $c(X) \leftrightarrow p(X)$

Performance cost: $\vartheta(l \times n)$

Multiplication

$c(X) \leftrightarrow c(X)$ or $c(X) \leftrightarrow p(X)$

Performance cost: $\vartheta(l \times n \log n)$ in BFV, $\vartheta(ln)$ in CKKS

Rescaling (CKKS)

Performed on $c(X)$ only

Performance cost: $\vartheta(l \times n \log n)$

Foundation

Rotation

Performed on $c(X)$ only

Performance cost: $\vartheta(n \times \log n \, logQ)$

Relinearization

Performed on $c(X)$ only

Performance cost: $\vartheta(l^2 \times n \log n)$

Modulus switching

Performed on $c(X)$ only

Performance cost: $\vartheta(l \times n)$ in BFV, $\vartheta(l \times nlogn)$ in CKKS

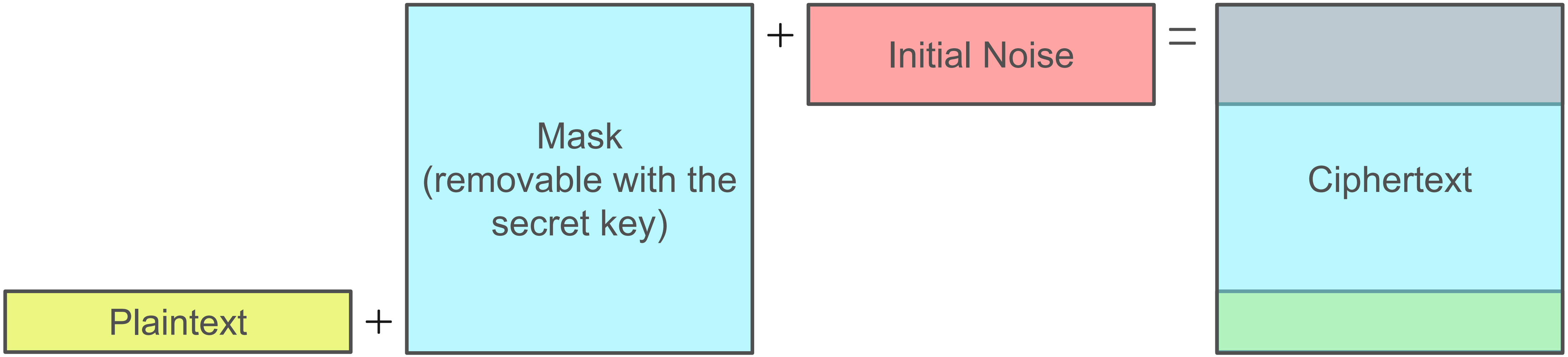
Note: l is the computational depth of the circuit

Homomorphic Encryption (HE) considerations

Anatomy of a ciphertext : Noise

The noise term is essential for homomorphic encryption schemes

Note also big message expansion



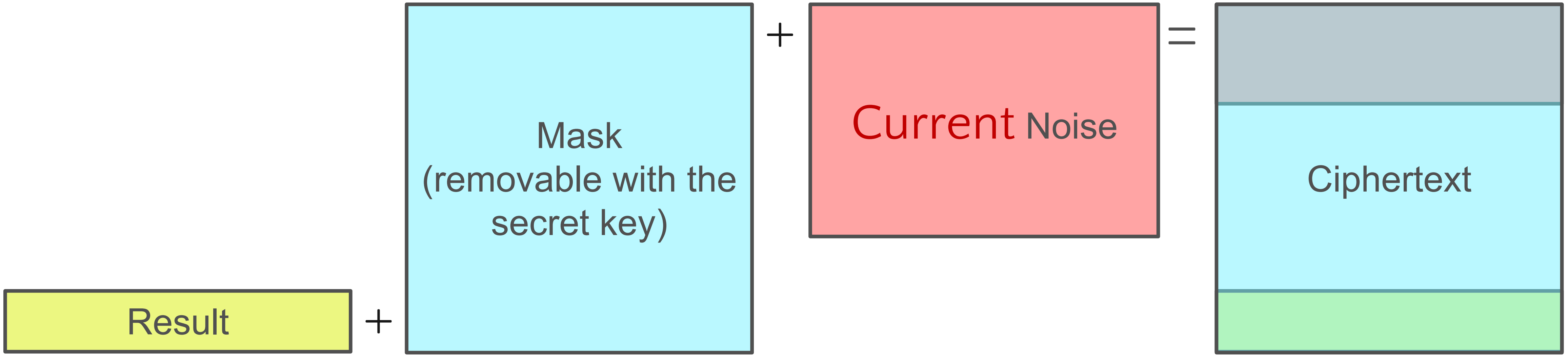
- Initial noise (of a fresh encryption) is small in terms of coefficients' size
- Independently of the initial noise, there is always a big message expansion : it is part of the security basis of this type of encryption

Homomorphic Encryption (HE) considerations

Anatomy of a ciphertext : after computation

After each level, noise increases

Note how message stays same size due to modular arithmetic

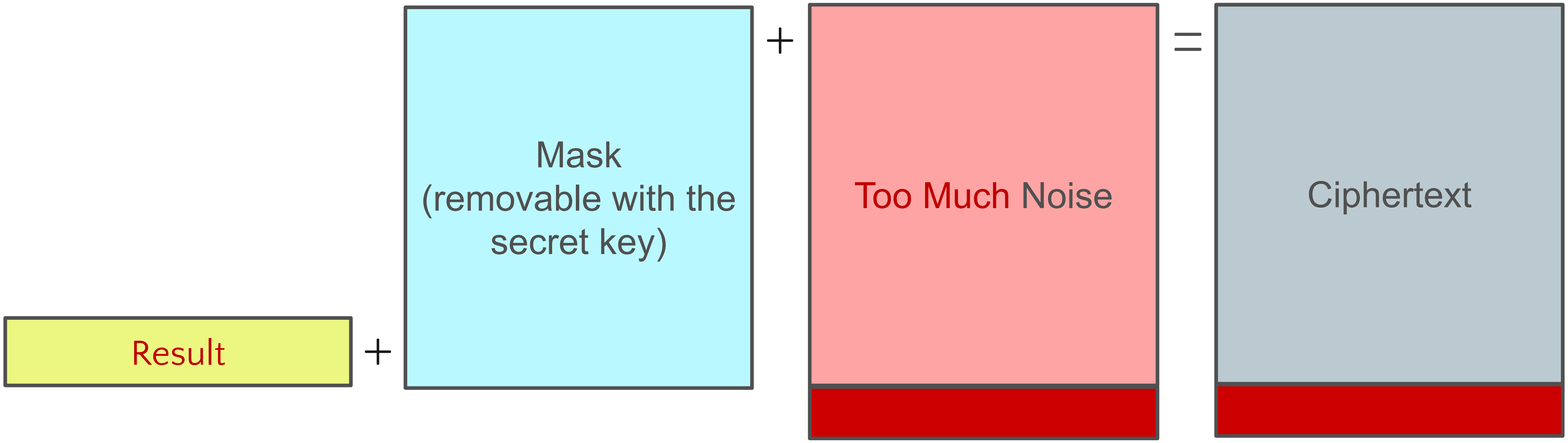


Homomorphic Encryption (HE) considerations

Anatomy of a ciphertext : Noise overflow

At some level, noise completely destroys the message

This is why only limited computation is possible

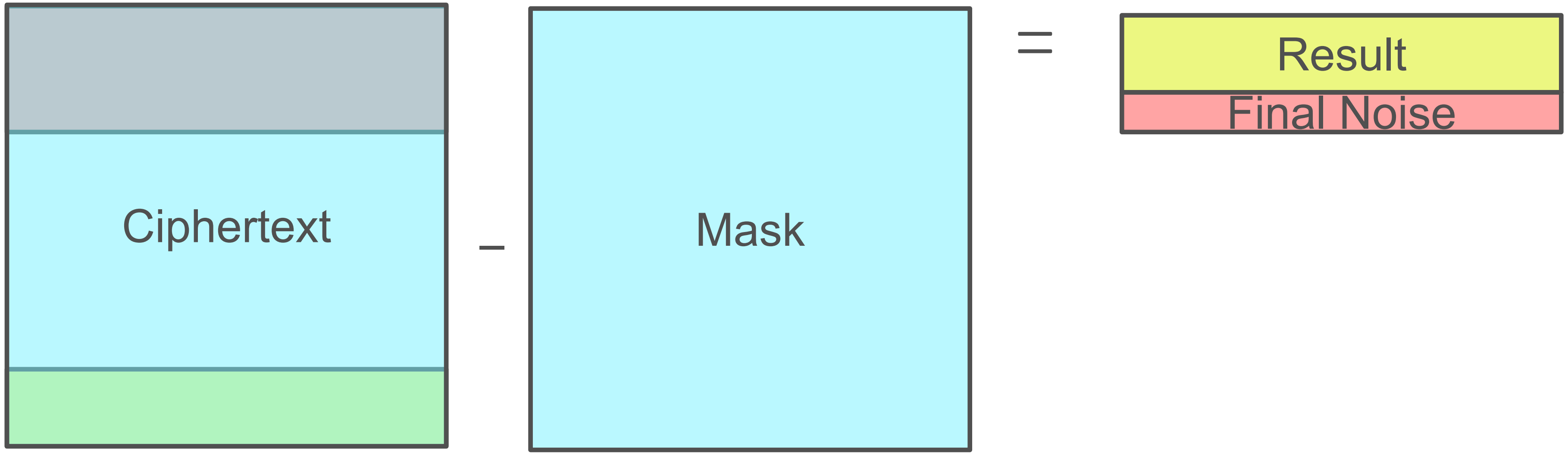


Homomorphic Encryption (HE) considerations

Anatomy of a ciphertext : decryption

Decrypting is removing the mask

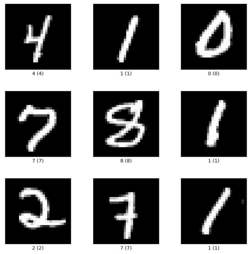
Note that there is a residual noise that must be managed



Microsoft homomorphic encryption

CryptoNets performance break-through

A breakthrough in 2016: evaluates neural net predictions on encrypted data



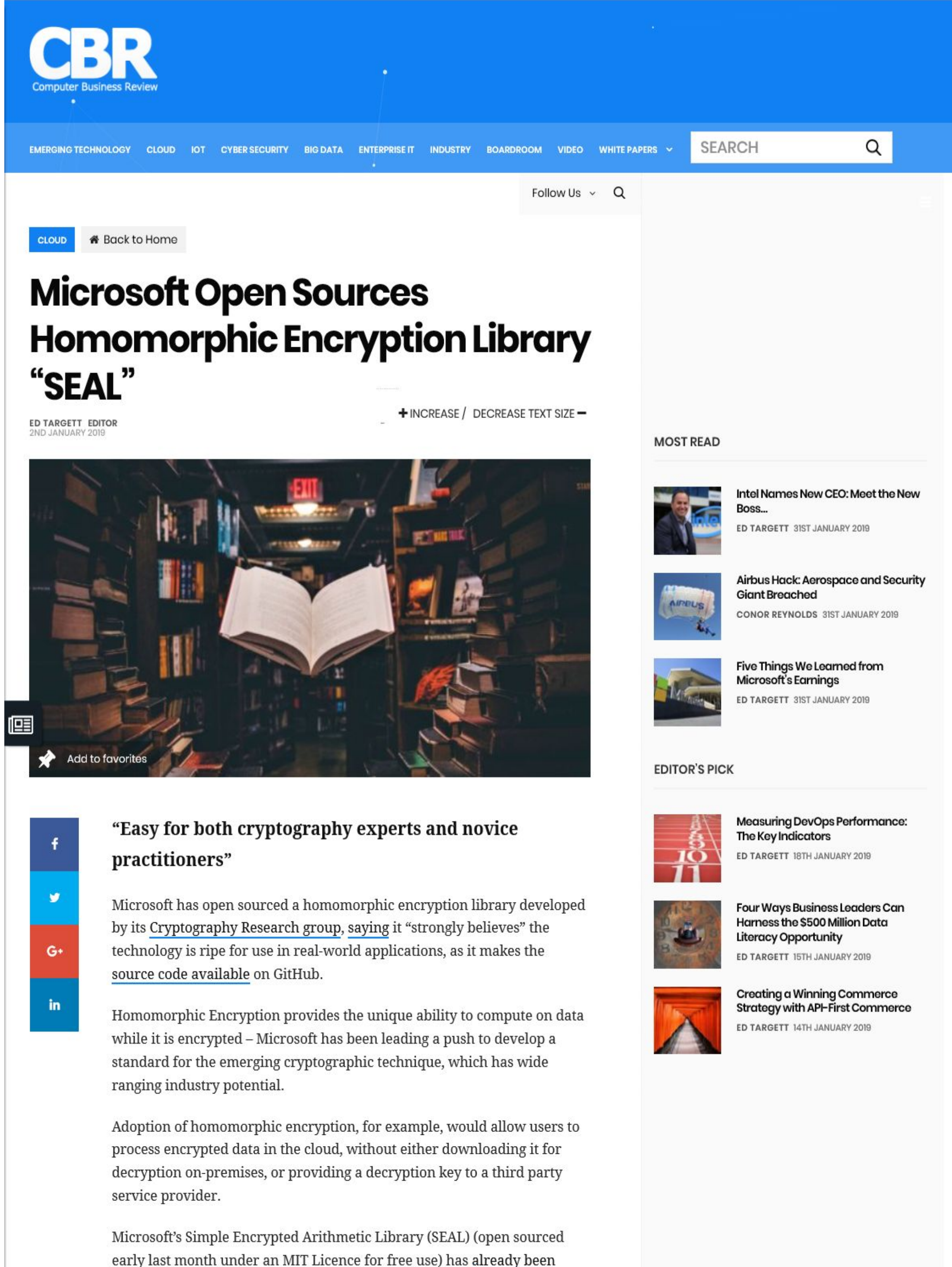
Simple Encrypted Arithmetic Library (SEAL)

See next slide

Widely adopted by research teams worldwide

Standardization in progress w/ community

http://sealcrypto.org



CBR
Computer Business Review

EMERGING TECHNOLOGY CLOUD IOT CYBER SECURITY BIG DATA ENTERPRISE IT INDUSTRY BOARDROOM VIDEO WHITE PAPERS SEARCH

Follow Us

CLOUD Back to Home

Microsoft Open Sources Homomorphic Encryption Library "SEAL"

ED TARGETT EDITOR
2ND JANUARY 2019

+INCREASE / DECREASE TEXT SIZE −

Add to favorites

"Easy for both cryptography experts and novice practitioners"

Microsoft has open sourced a homomorphic encryption library developed by its Cryptography Research group, saying it "strongly believes" the technology is ripe for use in real-world applications, as it makes the source code available on GitHub.

Homomorphic Encryption provides the unique ability to compute on data while it is encrypted – Microsoft has been leading a push to develop a standard for the emerging cryptographic technique, which has wide ranging industry potential.

Adoption of homomorphic encryption, for example, would allow users to process encrypted data in the cloud, without either downloading it for decryption on-premises, or providing a decryption key to a third party service provider.

Microsoft's Simple Encrypted Arithmetic Library (SEAL) (open sourced early last month under an MIT Licence for free use) has already been

MOST READ

Intel Names New CEO: Meet the New Boss...
ED TARGETT 31ST JANUARY 2019

Airbus Hack: Aerospace and Security Giant Breached
CONOR REYNOLDS 31ST JANUARY 2019

Five Things We Learned from Microsoft's Earnings
ED TARGETT 31ST JANUARY 2019

EDITOR'S PICK

Measuring DevOps Performance: The Key Indicators
ED TARGETT 18TH JANUARY 2019

Four Ways Business Leaders Can Harness the $500 Million Data Literacy Opportunity
ED TARGETT 15TH JANUARY 2019

Creating a Winning Commerce Strategy with API-First Commerce
ED TARGETT 14TH JANUARY 2019

Microsoft SEAL

Simple Encrypted Arithmetic Library

A state-of-the-art homomorphic encryption library from Microsoft Research (MSR)

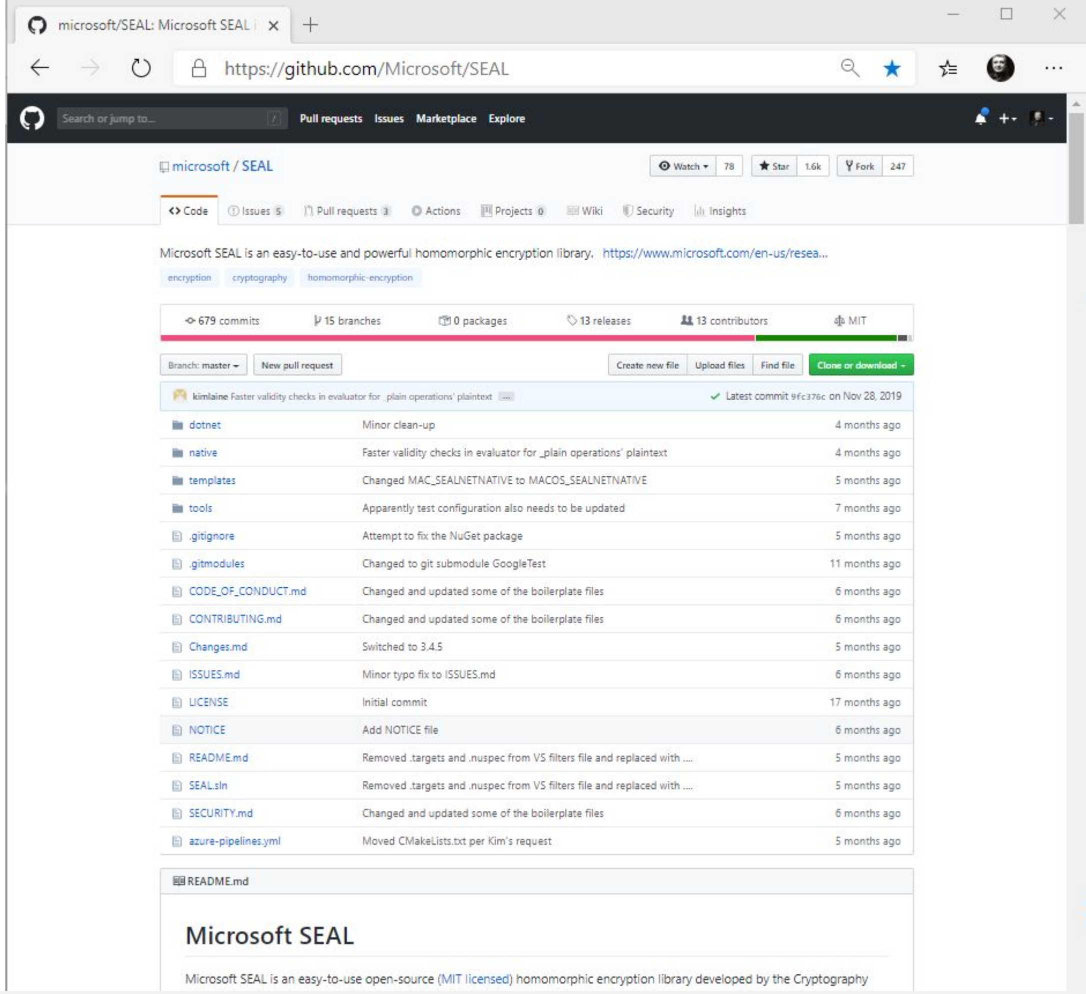
Encryption and decryption: Low-level primitives for encrypted arithmetic (add, subtract, multiply)

GPU/FPGA acceleration (up to +50x perf, resp. up to +120x perf)

SEAL 1.0 released under MSR-LA in 2015

SEAL 3.5.0 released under MIT license in April 2020

https://github.com/microsoft/SEAL/blob/master/Changes.md



Written in C++ 17; includes .NET Standard wrapper for public API (> SEAL 3.2)

- Supporting Azure Functions : serverless Secure Compute

From open source community:

- PyHeal (Python wrappers from Accenture), SEAL-python (yet another wrapper fpr SEAL 3.4.5)
- node-seal (JavaScript wrappers)
- nGraph HE Transformer (from Intel)
- TenSEAL (from OpenMinded)

Support to:

- Operating System: Windows, Linux, macOS, FreeBSD
- Smartphones : Android

Samples' repos on GitHub:

- SEAL-Demo
- algorithms-in-SEAL
- bootcamp
- Etc.

Applications of homomorphic encryption

Implementing Private AI

Implementing neural networks directly against SEAL is very challenging…

CHET is an optimizing compiler for fully-homomorphic neural-network inferencing

ONNX Runtime is Microsoft's open-source inference engine for ONNX, see RendezVous S03R03 "Open Neural Network eXchange (ONNX) De quoi s'agit-il ?"

An integration into ONNX Runtime allows our Private AI efforts to benefit from the large ecosystem around ONNX, which implies to expose capabilities for encrypted inferencing in ONNX Runtime:

- Adding a CHET execution provider
- Extending handling of encrypted inputs and outputs in ONNX Runtime

For models that CHET provides kernels for, compatible ONNX models and data will be provided.

EVA is a generic low-level language for homomorphic encryption

CHET compiles neural networks into EVA programs

EVA programs are optimized and compiled into SEAL API calls

SECLUD is a PaaS offering that will offer powerful but easy-to-understand privacy-preserving services through Azure

Private data lookup, statistics, data collaboration, and ML predictions

One core area of investment is enabling CHET-based inferencing in SECLUD, and thus integrating encrypted ONNX Runtime inferencing

Demo

First steps with TenSEAL

Code available at:
https://github.com/OpenMined/TenSEAL/tree/master/tutorials

Demo

Homomorphic Random Forests

Paper available at:
https://arxiv.org/pdf/2006.08299.pdf

Code available at:
https://github.com/dhuynh95/cryptotree

To go beyond…

On docs.microsoft.com

How to deploy an encrypted inferencing web service

- See Tutorial #3: Deploy an image classification model for encrypted inferencing in Azure Container Instance (ACI)

Medium series of articles on Homomorphic Encryption intro

Series of article on HE from the ground up, with a Python implementation from scratch of CKKS:

- Overview and use cases
- HE landscape and CKKS
- Encoding and decoding in CKKS

To go beyond…

Microsoft Research

Private AI Bootcamp December 2019

With a series of videos on:

- Intro to Homomorphic Encryption,
- Microsoft SEAL (+ bootcamp repo on GitHub),
- Intro to CKKS,
- Techniques in PPML (+ algorithms-in-SEAL repo on GitHub),
- Building Applications with Microsoft SEAL,
- Etc.

Private AI February 2020

Secure Multi-Party Computing (MPC)
Enabling secure collaborative computation

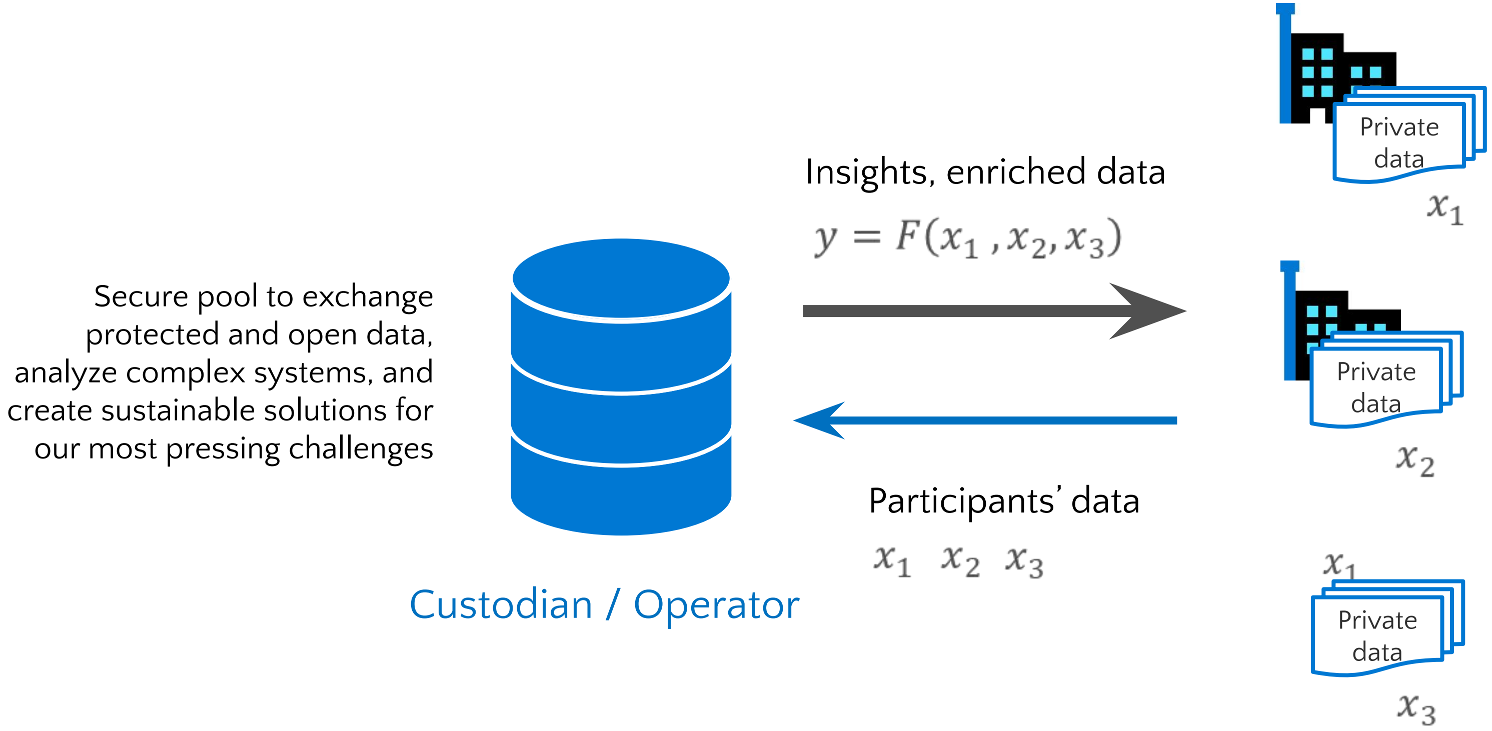
Solving today's complex challenges require greater access and use of trustworthy data to create new insights and facilitate **responsive and responsible action**

MPC has different trade-offs from trusted hardware enclave (CC) and homomorphic encryption (HE)

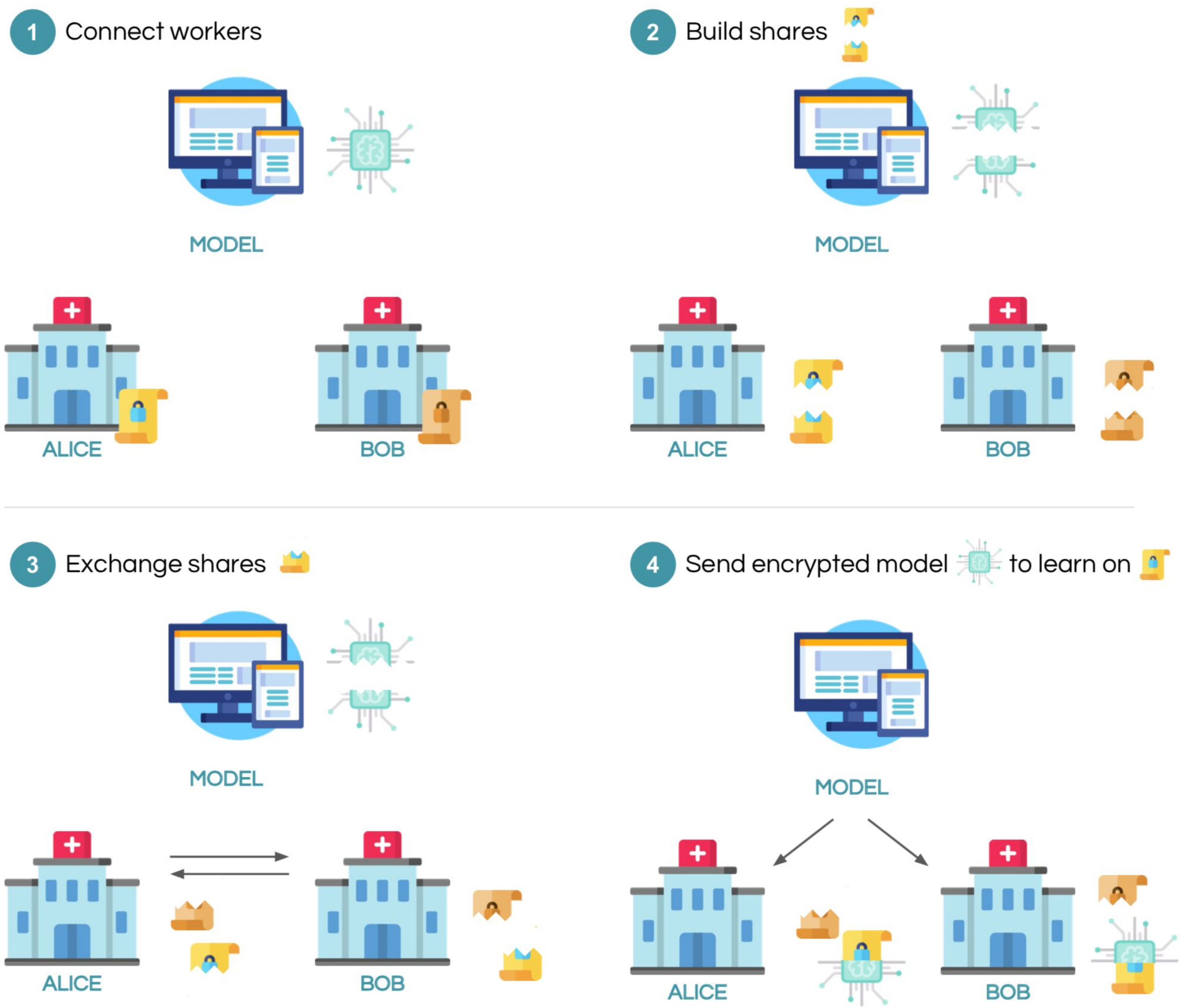
Secure Multi-Party Computation (MPC) pattern

A custodian provides a secure pool that meets requirements for confidentiality and compliance

Input privacy: x_i is not leaked to parties != i



Principle



Source: PySyft from OpenMined
https://github.com/OpenMined/PySyft

Secure MPC principles

Additive shares

- If N people participate in a computation, data is sharded into N shares such that knowing up to N–1 shares provides no information on the input, but only by summing all N shares we can reconstruct the data

Randomized shares

- To provide privacy for each party when data is shared, we sometime need to have a third party providing random masks $\mathbf{e}$ to each party so that they can disclose $\mathbf{y} = \mathbf{x} + \mathbf{e}$, instead of just $\mathbf{x}$ to hide the initial input

Secure computations

- Using shares and random mask, complex operations can be performed such as matrix multiplication, comparisons, MaxPooling, etc.

SMPC principles

Example of multiplication

- Have $\langle x \rangle, \langle y \rangle$, want $\langle x \cdot y \rangle$.
- Use random triple $\langle a \rangle, \langle b \rangle, \langle a \cdot b \rangle$
- Compute and open $\langle x + a \rangle, \langle y + b \rangle$

- Observe:

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

opened

preprocessed

Source: https://simons.berkeley.edu/sites/default/files/docs/15669/simons-spdz.pdf

Secure MPC protocols

EzPC project from Microsoft Research India

SecureNN: Efficient and Private Neural Network Training (2019)

Source: https://www.microsoft.com/en-us/research/uploads/prod/2018/09/securenneprint.pdf

SecureNN is a 3-party SMPC framework with various building blocks such as matrix multiplication, ReLU, convolution, MaxPool, etc. for the training and inference of Neural Networks such that no party learns anything about other people's data

CrypTFlow: Secure TensorFlow Inference (2020)

Source: https://www.microsoft.com/en-us/research/uploads/prod/2019/09/CrypTFlow.pdf

CrypTFlow is a Secure MPC system with three components:

1. Athos: a TensorFlow end-to-end compiler for SMPC backends
2. Porthos: a state of the art semi-honest SMPC protocol for TensorFlow inference
3. Aramis: a technique to convert semi-honest Secure MPC protocols to MPC protocols with malicious security using hardware integrity solutions such as Intel SGX (see previous Confidential Computing part)

Applications of secure MPC

Auction, Voting, Social research

Distributed signing / Key management

Private set intersection / Location based services

How can two companies find which customers they have in common without revealing their customers to each other?

Collaborative Machine Learning on private data

Huge amount of internal and external interest; high-profile project

(Your favorite computing scenario, with privacy added :-))

As a quick conclusion of this Common Talk

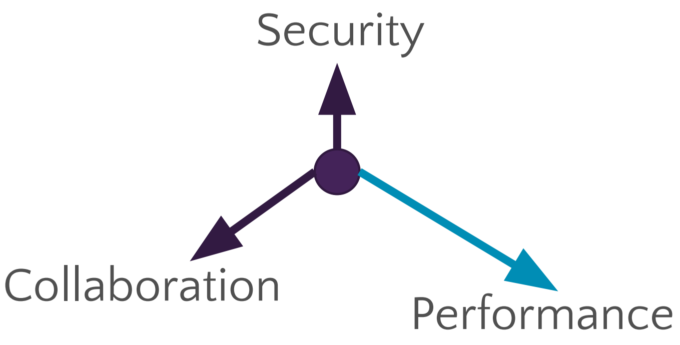
A review and comparison of our three main (mitigation) technologies' topics

	Confidential Computing (CC) (w/ secure hardware)	Homomorphic Encryption (HE)	Secure Multi-Party Computation (MPC)
Performance	-	Compute-bound	Network-bound
Privacy	Trusted Hardware	Encryption	Encryption / Non-collusion
Non-interactive	✔	✔	✘
Cryptographic security	✘ (known attacks)	✔	✔
Pros & Cons	Different Security Model Side-Channel Attacks	High Computation Overhead	High Communication High Number of Interactions

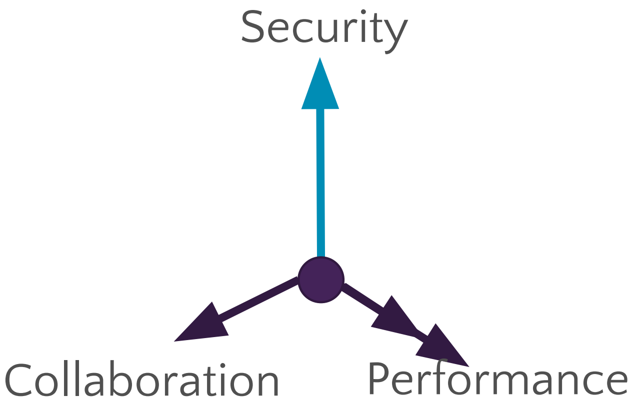
As a conclusion of this Common Talk

A review and comparison of our three main technologies' topics, see also the wrap-up of the workshop for additional considerations

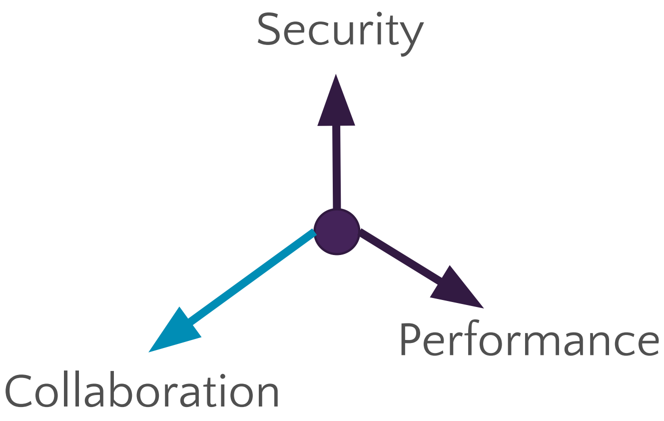
Confidential Computing (CC)
w/ hardware enclaves



Homomorphic Encryption (HE)



Secure Multi Party Computation (MPC)



Q&A

Thanks!

ευχαριστώ Salamat Po متشکرم شكراً Grazie

благодаря ありがとうございます Kiitos Teşekkürler 谢谢

ขอบคุณครับ Obrigado شکریہ Terima Kasih Dziękuję

Hvala Köszönöm Tak Dank u Wel дякую Tack

Mulţumesc спасибо Danke Cảm ơn Gracias

多謝晒 Ďakujem תודה நன்றி Děkuji 감사합니다

https://aka.ms/DataInUseProtectionWS

Differential Privacy (DP)
Protecting privacy while using data

Differential Privacy (DP)

There is a significant amount of data that is inaccessible because of privacy

For most scenarios, data anonymization doesn't work

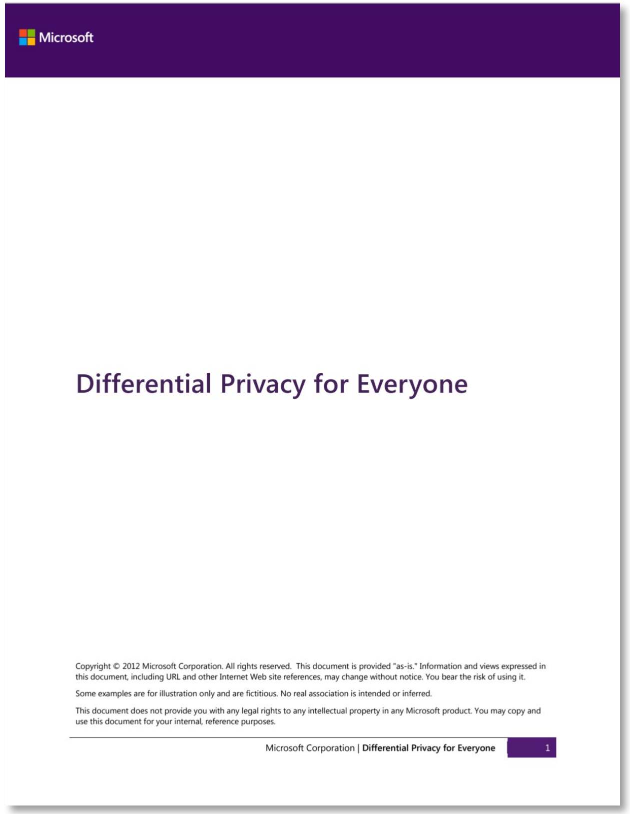
> "Anonymized data isn't"
> – Cynthia Dwork, Microsoft Research (MSR)

Data scientists, analysts, scientific researchers and policy makers often do not need an individual's data

All they need is the signal

Differential privacy provides a way to get insights from the data, but without affecting the privacy of the individuals

A technique that offers strong privacy assurances, preventing data leaks and re-identification of individuals in a dataset



Microsoft

Differential Privacy for Everyone

Microsoft Corporation | **Differential Privacy for Everyone** 1

Differential Privacy:
A Primer for a Non-technical Audience*

(Preliminary version)

Kobbi Nissim†[1], Thomas Steinke[2], Alexandra Wood[3], Mark Bun[2], Marco Gaboardi[4], David R. O'Brien[3], and Salil Vadhan[2]

[1]Department of Computer Science, Georgetown University. kobbi.nissim@georgetown.edu.
[2]Center for Research on Computation and Society, Harvard University. {tsteinke|mbun|salil}@seas.harvard.edu.
[3]Berkman Klein Center for Internet & Society, Harvard University. {awood|dobrien}@cyber.law.harvard.edu.
[4]University at Buffalo, The State University of New York. gaboardi@buffalo.edu.

March 3, 2017

Abstract

This document is a primer on *differential privacy*, which is a formal mathematical framework for guaranteeing privacy protection when analyzing or releasing statistical data. Recently emerging from the theoretical computer science literature, differential privacy is now in initial stages of implementation and use in various academic, industry, and government settings. Using intuitive illustrations and limited mathematical formalism, this document provides an introduction to differential privacy for non-technical practitioners, who are increasingly tasked with making decisions with respect to differential privacy as it grows more widespread in use. In particular, the examples in this document illustrate ways in which social scientists can conceptualize the guarantees provided by differential privacy with respect to the decisions they make when managing personal data about research subjects and informing them about the privacy protection they will be afforded.

Keywords: differential privacy, data privacy, social science research

*This document is the product of a working group of the *Privacy Tools for Sharing Research Data* project at Harvard University (http://privacytools.seas.harvard.edu). The working group discussions were led by Kobbi Nissim. Kobbi Nissim, Thomas Steinke, and Alexandra Wood were the lead authors of this document. Working group members Mark Bun, David R. O'Brien, Marco Gaboardi, Kobbi Nissim, Thomas Steinke, Salil Vadhan, and Alexandra Wood contributed to the conception of the document and to the writing. We thank Caper Gooden, Deborah Hurley, Georgios Kellaris, Daniel Muise, and Michel Reymond for their many valuable comments on earlier versions of this document. This material is based upon work supported by the National Science Foundation under Grant No. 1237235, as well as by the Alfred P. Sloan Foundation.

†Work towards this document was completed while the author was at the Center for Research on Computation and Society at Harvard University.

Differential Privacy (DP)

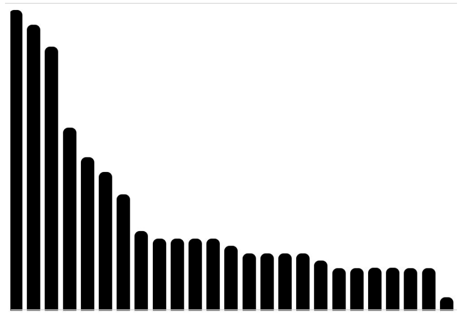
A class of algorithms that facilitate computing and releasing aggregate statistics from sensitive (private) data, while ensuring that privacy is not compromised

Large pool of sensitive data, noisy aggregates are still valuable

Pool of private data

DP algorithm

Aggregate statistics



DP statistics:

- Noisy. Noise conceals contributions of individual users
- Can be safely handed to an all-powerful adversary, with no risk of a privacy leak

Introduced by Cynthia Dwork and her collaborators at MSR in 2006

See Differential Privacy

Gradually came to dominate the area of privacy research

- Many thousands of citations to Cynthia et al. classical papers
- 2017 Gödel prize
- Deployments by major IT companies, US Census Bureau

Two main flavors

1. Local Differential Privacy (telemetry): simple statistics
2. Global Differential Privacy (trusted data collector): more complex aggregates including Machine Learning models

Local Differential Privacy

Setting

Data collector wants to understand global trends in the users' data
Data collector is not trusted, cannot collect raw data

Local DP algorithm

Users carefully randomize their private data before sending it to data collector
Randomization provides plausible deniability
Noise becomes immaterial when data is collected form many users. Global trends are exposed

Usage in telemetry

Facilitates collection of data that cannot be collect in the raw form
Suitable to collect basic statistical aggregates (mean, histogram, frequent items, etc.)
E.g. Telemetry in Windows: see Collecting Telemetry Data Privately

Global Differential Privacy

Setting

Database curator has access to a pool of private users' data

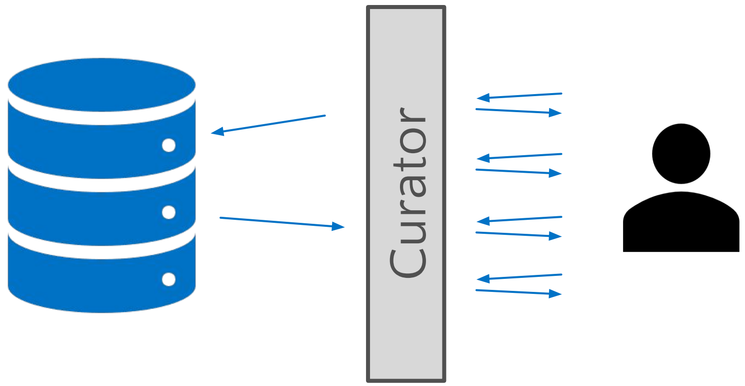
Needs to disclose some aggregate statistics to an analyst

Global DP algorithms

Database curator gives noisy responses to analyst's queries: if the analyst wants f(D), he gets f(D)+noise

Noise is random, and hides a contribution of a single user

Any specific output is roughly as likely to be produced whether or not a single record is removed from the DB



Usage examples

Office: expose the set of frequent e-mail responses, see Assistive AI Makes Replying Easier

LinkedIn (Ad campaign statistics), see LinkedIn's Audience Engagements API: A Privacy Preserving Data Analytics System at Scale

US Census Bureau: every published aggregate statistics for 2020 US Census will use DP

Differential Privacy for Machine Learning and Analytics

Introducing WhiteNoise

An open source project jointly developed by Microsoft, Harvard's Institute for Quantitative Social Science (IQSS) and the School of Engineering and Applied Sciences (SEAS) and

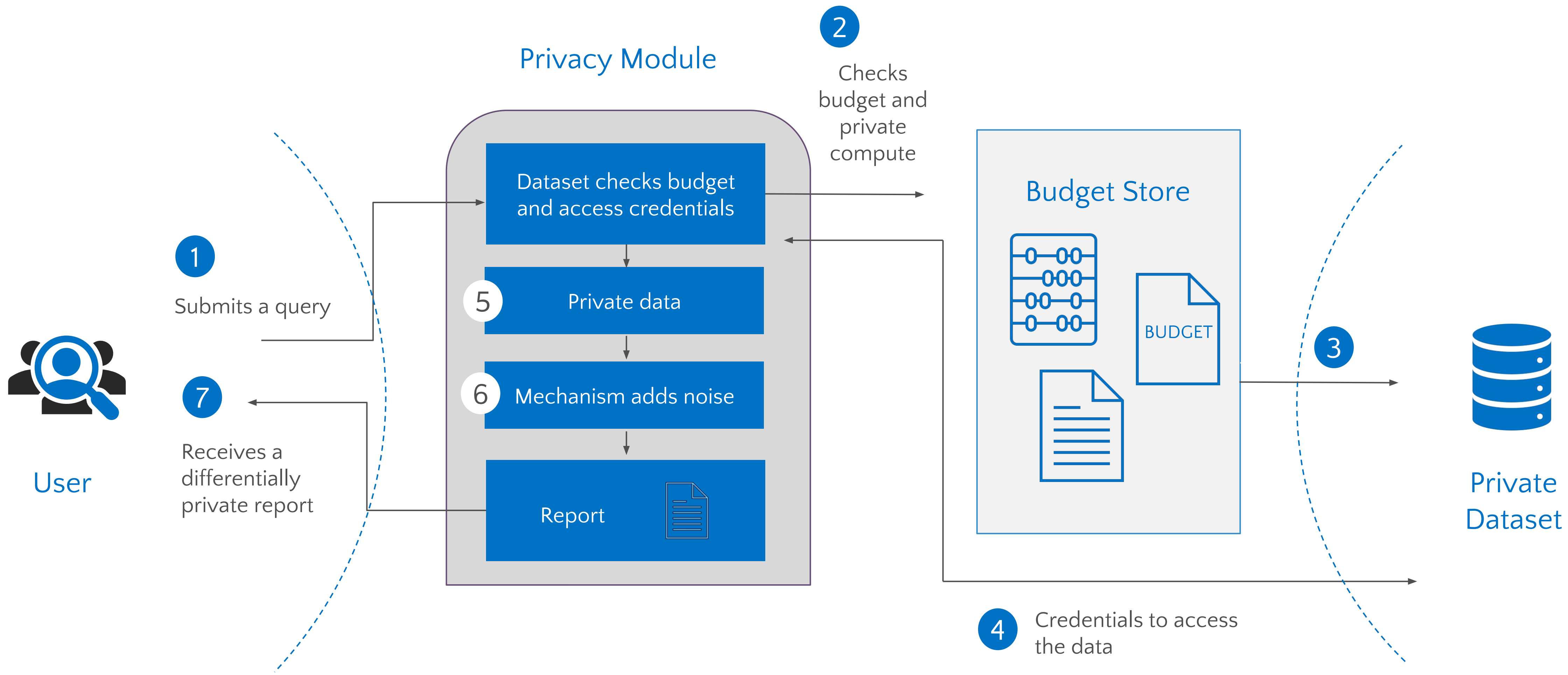
Currently supports scenarios where the data user is trusted by the data owner and wants to ensure that their releases or publications do not compromise privacy

- Future releases will focus on hardened scenarios where the researcher or analyst is untrusted and does not have access to the data directly

https://opendifferentialprivacy.github.io
https://github.com/opendifferentialprivacy

WhiteNoise

WhiteNoise



https://opendifferentialprivacy.github.io
https://github.com/opendifferentialprivacy

WhiteNoise

WhiteNoise

WhiteNoise Core

- A pluggable open source library of differentially private algorithms and mechanisms with implementations based on mature differential privacy research
- APIs for defining an analysis and a validator for evaluating these analyses and composing the total privacy loss on a dataset

WhiteNoise handles managing the budget for each query and adding the appropriate amount of noise-based budget

WhiteNoise System

System components to make it easier to interface with your data systems, including a SQL query dialect and connections to many common data sources, such as PostgreSQL, SQL Server, Spark, Presto, Pandas, and CSV files

WhiteNoise Samples

Example code and notebooks to demonstrate the use of the WhiteNoise platform, teach the properties of differential privacy, and highlight some of the nuances of the WhiteNoise implementation

https://opendifferentialprivacy.github.io
https://github.com/opendifferentialprivacy

WhiteNoise

OpenDP

Building an open source suite of tools for deploying differential privacy

A community effort in partnership with Microsoft to build a trustworthy and open-source suite of differential privacy tools that can be easily adopted by custodians of sensitive data to make it available for research and exploration in the public interest

Incubated by Harvard University's Privacy Tools and Privacy Insights projects (at SEAS and IQSS)

With stakeholders and contributors from across academia, industry, and government

Plan to design, implement, and govern an "OpenDP Commons" that includes a library of differentially private algorithms and other general-purpose tools for use in end-to-end differential privacy systems

https://projects.iq.harvard.edu/opendp

HARVARD UNIVERSITY

To go beyond…

Differential Privacy: series of lecture form Cynthia Dwork (Microsoft Research)

Lecture 1, Lecture 2, Lecture 3, and Lecture 4

WhiteNoise project:

Introducing WhiteNoise: the new differential privacy platform from Microsoft and Harvard's OpenDP
https://cloudblogs.microsoft.com/opensource/2020/05/19/new-differential-privacy-platform-microsoft-harvard-opendp/

Preserve data privacy by using differential privacy and the WhiteNoise package
https://docs.microsoft.com/azure/machine-learning/concept-differential-privacy

Use differential privacy in Azure Machine Learning
https://docs.microsoft.com/azure/machine-learning/how-to-differential-privacy

WhiteNoise: A Platform for Differential Privacy
Https://aka.ms/WhiteNoiseWhitePaper