Privacy Preserving Technologies and Open Source Projects: Challenges and Opportunities

Wenhui Zhang , Bytedance Inc.
Chinmay Shah , OpenMined Org.
Callis Ezenwaka, Barewave

Privacy Preserving Technologies and Open Source Projects: Challenges and Opportunities

- Review the current privacy preserving technologies
- Summarize the popular open source projects
- Review both current and in progress standard/regulations/law
- Discuss challenges and opportunities for production

Privacy Preserving for Federated Learning

Growing Demand of Privacy Enhancing Computing

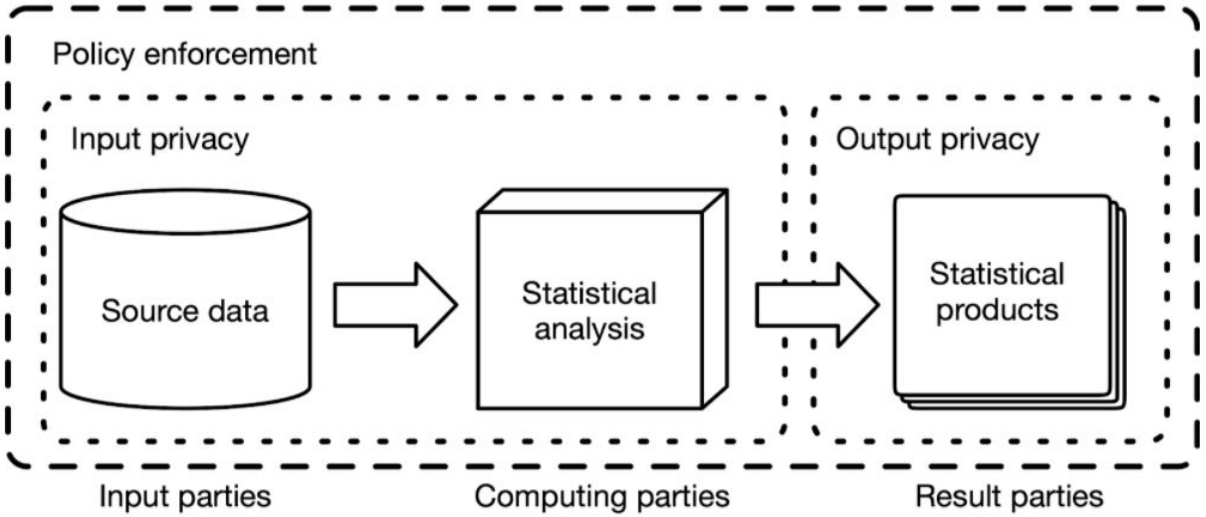
“By 2025, 60% of large organizations will use one or more privacy enhancing computation techniques in analytics, business intelligence or cloud computing. ”

Source: Gartner

Increasing Need for Federated Learning

Privacy Goals for Federated Learning
(e.g. Statistics and Machine Learning)

No data leak, constrain the sensitive data to authorized entities.



Source: UN Handbook for Privacy Preserving Techniques

AKRAINO

Open Source Frameworks for Federated Learning

1. Federated AI Technology Enabler, Webank Fin tech ,
   https://github.com/OpenFederatedLearning/FATE
   https://github.com/OpenFederatedLearning/eggroll
2. FedML, USC, https://github.com/OpenFederatedLearning/FedML
3. Fedlearner, Bytedance, https://github.com/OpenFederatedLearning/fedlearner
4. Harmonia, AL Labs Taiwan, https://github.com/OpenFederatedLearning/harmonia
5. Openfl, Intel, https://github.com/OpenFederatedLearning/openfl
6. PaddleFL, Baidu, https://github.com/OpenFederatedLearning/PaddleFL
7. PySyft, OpenMined, https://github.com/OpenFederatedLearning/PySyft
8. Tensorflow Federated, Google, https://github.com/OpenFederatedLearning/federated
9. 9NFL, Jingdong, https://github.com/OpenFederatedLearning/9nfl

AKRAINO

Privacy Preserving Techniques

Privacy Preserving Techniques

Privacy preserving technique allows sharing sensitive personal information while preserving users' privacy. Below are the technologies used in Federated Learning:

1. Anonymization (weakest)
2. Differential Privacy
3. Fully Homomorphic Encryption (FHE)
4. Zero Knowledge Proof
5. Secure Multi-Party Computation (MPC)
6. Confidential Computing through TEE (e.g. SGX and TrustZone)

AKRAINO

Federated Learning Stack

Client

Server

Anonymization

Differential Privacy

Zero Knowledge Proof

Fully Homomorphic Encryption

Secure Multi-Party Computation

Trusted Execution Environment

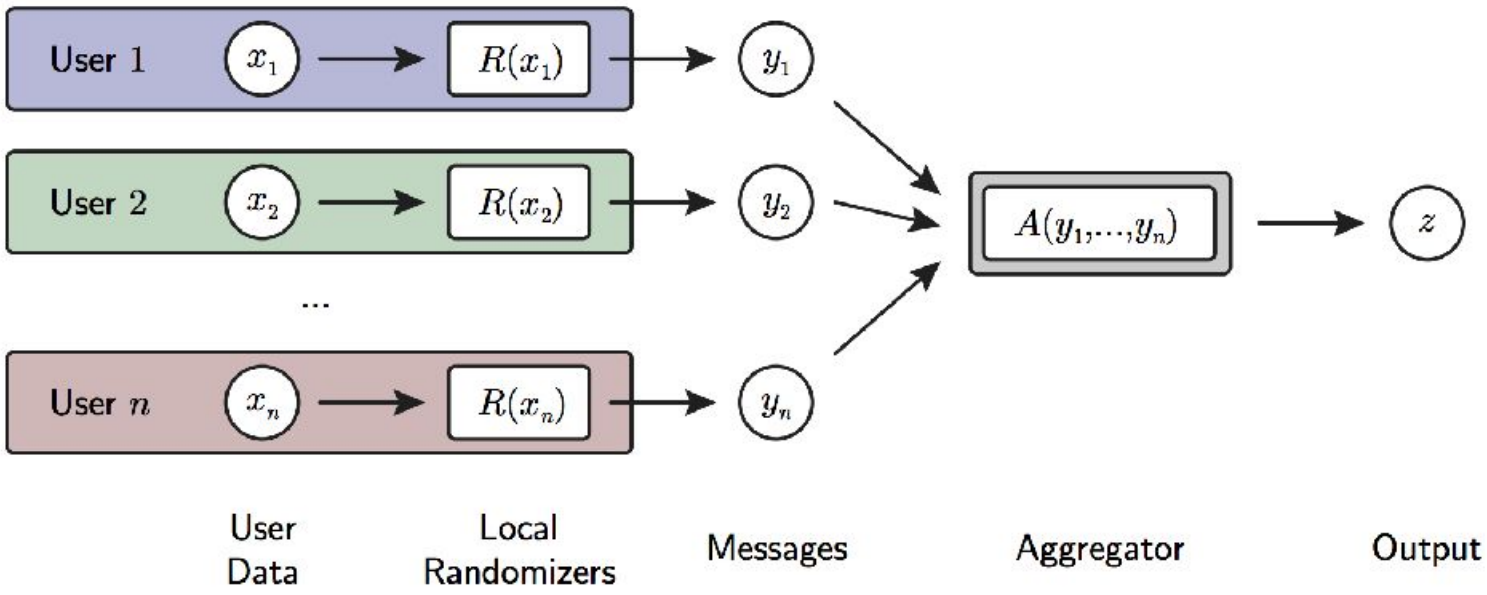
Anonymization

The goal of anonymization is to anonymize data so that the data could not be associated with any one individual.

- Data Generalization (achieve k-anonymity and l-diversity):
  - k-anonymity: General a data property, so that in the group the data belongs to, at least k-1 individuals who have the same properties.
  - l-diversity: Anonymized data set for its' sensitive attributes.
- Adding Noise to Data:
  - By adding mathematical noise to data, make it difficult to ascertain whether any one individual is part of a data set.

Differential Privacy

Differential Privacy provides output privacy for statistics and databases.



Open Source Projects of Differential Privacy

1. Diffprivlib, built and supported by IBM, is also designed for data scientists, but is slightly more focused on machine learning tasks. http://www.bipr.net/diffpriv/
2. Google has Tensorflow privacy, a library for training machine learning models with privacy for training data. https://github.com/tensorflow/privacy
3. Uber use have SQL-diff for dataflow analysis & differential privacy for SQL queries. https://github.com/uber-archive/sql-differential-privacy
4. PING by Microsoft, is a SQL query engine with differential privacy preserved. https://www.microsoft.com/en-us/research/project/privacy-integrated-queries-ping/
5. PSI is a framework for pipelined differential privacy analysis, http://psiprivacy.org/static/about/index.html
6. PipelineDP is a framework developed by OpenMined, for performing Differentially private data aggregation https://pipelinedp.io/
7. OpenDP is a community-supported set of tools designed for data scientists. It includes implementations of many of the tools we have discussed in this series. https://github.com/OpenFederatedLearning/smartnoise-samples

Fully Homomorphic Encryption

Homomorphic Encryption cryptosystem is a cryptosystem whose decryption is a morphism.

```
Decrypt(a*b) = Decrypt(a) * Decrypt(b)
```

Homomorphic Encryption cryptosystem allows operate on ciphertexts without decryption.

It ensures end-to-end semantically secure, which is ensuring security against honest but curious adversaries.

Different from confidential computing, FHE takes a software-based data encryption/protection.

Since FHE does not perform computational processing in Trusted Execution Environment (TEE), and unauthorized access or modification of data and application code during processing might occur. Thus, FHE does not support application code integrity nor code confidentiality.

Fully Homomorphic Encryption

1. Pre-FHE, where operations are limited for unbounded homomorphic encryption operations
   - RSA cryptosystem (unbounded number of modular multiplications)
   - ElGamal cryptosystem (unbounded number of modular multiplications)
   - Goldwasser–Micali cryptosystem (unbounded number of exclusive or operations)
   - Benaloh cryptosystem (unbounded number of modular additions)
   - Paillier cryptosystem (unbounded number of modular additions)
   - Sander-Young-Yung system (after more than 20 years solved the problem for logarithmic depth circuits)
   - Boneh–Goh–Nissim cryptosystem (unlimited number of addition operations but at most one multiplication)
   - Ishai-Paskin cryptosystem (polynomial-size branching programs)
2. First-generation FHE, based on lattice model, however "limited to evaluating low-degree polynomials over encrypted data" [3] .
   - Marten van Dijk, Craig Gentry, Shai Halevi and Vinod Vaikuntanathan (idea lattice)
3. Second-generation FHE, based on RLWE and NTRU related problem.
   - The Brakerski-Gentry-Vaikuntanathan (BGV, 2011) scheme.
   - NTRU-based scheme by Lopez-Alt, Tromer, and Vaikuntanathan (LTV, 2012).
   - The Brakerski/Fan-Vercauteren (BFV, 2012) scheme,on Brakerski's scale-invariant cryptosystem.
   - The NTRU-based scheme by Bos, Lauter, Loftus, and Naehrig (BLLN, 2013),building on LTV and Brakerski's scale-invariant cryptosystem;
4. Third-generation FHE
   - Craig Gentry, Amit Sahai, and Brent Waters (GSW), on building FHE schemes that avoids an expensive "relinearization" step in homomorphic multiplication.
   - FHEW (2014), ring variants of the GSW cryptosystem
   - TFHE (2016), ring variants of the GSW cryptosystem
   - CKKS scheme, focuses on machine learning, conducts efficient rounding operations in encrypted state.

Open Source Projects of Fully Homomorphic Encryption

Name	BGV	BFV	FHEW	TFHE	CKKS	CKKS Bootstrapping
Concrete				Yes		
EVA					Yes	
FHEW			Yes	Yes		
FV-NFLlib		Yes				
HeaAn					Yes	Yes
HElib	Yes				Yes	
Lattigo		Yes			Yes	Yes
Microsoft SEAL		Yes			Yes	
NuFHE				Yes		
PALISADE	Yes	Yes	Yes	Yes	Yes	
TFHE				Yes		
$\Lambda \circ \lambda$	Yes	Yes				

Zero Knowledge Proof

Zero Knowledge Proofs provides the ability to prove honest computation without revealing inputs.

A zero-knowledge proof has three salient properties:

- **Completeness:** If the statement is true and both the prover and the verifier follow the protocol; the verifier will accept the proof.
- **Soundness:** If the statement is false, and the verifier follows the protocol; the verifier will not be convinced by the proof.
- **Zero-knowledge:** If the statement is true and the prover follows the protocol; the verifier will not learn any confidential information from the interaction with the prover except that the statement is true.

Source: UN Handbook for Privacy Preserving Techniques https://docs.google.com/document/d/1GYu6UJI81jR8LgooXVDsYk1s6FlM-SbOvo3oLHglFhY/edit#

Usecase of Zero Knowledge Proof at Edge

- checking that taxes have been properly paid by some company or person;
- checking that a given loan is not too risky;
- checking that data is retained by some record keeper (without revealing or transmitting the data);
- checking that an airplane has been properly maintained and is fit to fly.

In many of the above auditing and compliance checking scenarios the underlying computation is a data analysis algorithm. Thus zero knowledge enables proofs that a given output is the output of a correct data analysis on some sensitive input data.

Source: UN Handbook for Privacy Preserving Techniques https://docs.google.com/document/d/1GYu6UJI81jR8LgooXVDsYk1s6FIM-SbOvo3oLHglFhY/edit#

AKRAINO

Open Source Projects of Zero Knowledge Proof

- SNARKs (Succinct Non-Interactive Argument of Knowledge
  1. Zokrates a great SNARK domain specific language (DSL) for generating proofs and validating them on Ethereum https://github.com/ZeroKnowlegeProof/ZoKrates
  2. Bellman Rust implementation https://github.com/ZeroKnowlegeProof/bellman
  3. Snarky OCaml implementation (DSL) https://github.com/ZeroKnowlegeProof/snarky
  4. LIbsnark C++ https://github.com/ZeroKnowlegeProof/libsnark
  5. Iden3's Circum (DSL) & SnarkJS Javascript Implementation https://github.com/ZeroKnowlegeProof/circom
  6. Republic Protocol's zksnark-rs (DSL) Rust implementation https://github.com/ZeroKnowlegeProof/zksnark-rs
  7. DIZK Java Distributed system https://github.com/ZeroKnowlegeProof/dizk
  8. Go-SNARK zkSNARK library implementation in Go https://github.com/ZeroKnowlegeProof/go-snark-study
- STARKs
  1. C++ implementation https://github.com/ZeroKnowlegeProof/libSTARK
- Bulletproofs
  1. Benedikt's Bunz Java implementation https://github.com/ZeroKnowlegeProof/BulletProofLib

Secure Multi-Party Computation

- Garbled Circuit(Andrew Chi-Chih Yao, FOCS'86) -> high bandwidth, high latency
- Secret Sharing(Shamir and Blakley 1979) -> low computation, low communication
- Oblivious Transfer (Rabin 1981) -> sent n msg, receiver receive one of them, could also be used in private set intersection, private information retrieval

Open Source Projects of Secure Multi-Party Computation

- ABY/ABY3 - 2PC/3PC with secret sharing and garbled circuits https://github.com/encryptogroup/ABY https://github.com/ladnir/aby3
- BatchDualEx - 2PC with garbled circuits https://github.com/osu-crypto/batchDualEx
- Carbyne Stack - MPC with Kubernetes, Istio, and Knative https://carbynestack.io/
- CrypTen, Facebook, MPC in PyTorch https://github.com/facebookresearch/CrypTen
- EMP-toolkit - 2PC and MPC with garbled circuits https://github.com/emp-toolkit
- Fancy-Garbling - 2PC with arithmetic garbled circuits in Rust https://github.com/spaceships/fancy-garbling
- FRESCO - MPC supporting TinyTables or SPDZ protocols http://fresco.readthedocs.io/en/latest/
- HoneyBadgerMPC - confidentiality layer for blockchains for output delivery https://github.com/initc3/HoneyBadgerMPC
- JIFF - JavaScript https://github.com/multiparty/jiff/
- MPyC - BGW https://www.win.tue.nl/~berry/mpyc/
- Obliv-C - 2PC with garbled circuits; secure against semi-honest adversaries. http://oblivc.org/
- Obliv-Java - Faithful reimplementation of Java using Obliv-C.https://github.com/Calctopia-OpenSource/jdk10u
- Rosetta - 3PC TensorFlow https://github.com/LatticeX-Foundation/Rosetta/
- MOTION - Mixed-Protocol MPC framework supporting full-threshold boolean and arithmetic GMW and BMR https://github.com/encryptogroup/MOTION
- MP-SPDZ - SPDZ, SPDZ2k, MASCOT, Overdrive, BMR garbled circuits, Yao's garbled circuits, and computation based on three-party replicated secret sharing as well as Shamir's secret sharing https://github.com/data61/MP-SPDZ
- Sharemind - 2PC or 3PC with secret sharing https://sharemind.cyber.ee/
- Tf-encrypted - 3PC with secret sharing on TensorFlow-based applications. https://github.com/tf-encrypted/tf-encrypted

Confidential Computing through TEE

Confidential Computing leverages hardware-based Trusted Execution Environments (TEE) to protect data in use.

It preserves: (defined by CCC)

- “Data confidentiality: Unauthorized entities cannot view data while it is in use within the TEE.”
- “Data integrity: Unauthorized entities cannot add, remove, or alter data while it is in use within the TEE.”
- “Code integrity: Unauthorized entities cannot add, remove, or alter code executing in the TEE.”

Source: Confidential Computing Consortium

AKRAINO

Open Source Projects of Confidential Computing

- Projects on Intel SGX:
  1. SGX SDK with samples: https://github.com/intel/linux-sgx
  2. Gramine Build: https://gramine.readthedocs.io/en/latest/devel/building.html
  3. Open Enclave SDK: https://github.com/openenclave/openenclave
  4. MesaTEE, https://github.com/ConfidentialComputing/incubator-teaclave
  5. Fortanix TEE, https://github.com/ConfidentialComputing/rust-sgx
  6. Enclave Development Platform (EDP), Fortanix https://edp.fortanix.com/
  7. Google Aslyo TEE, https://asylo.dev/
  8. Anjuna Redis: https://docs.anjuna.io/apps/redis/installing.html
  9. Anjuna: https://docs.anjuna.io/anjuna-runtime/anjuna-documentation/latest/index.html.
  10. EGo, Edgeless https://www.ego.dev/
  11. MarbleRun, Edgeless https://marblerun.sh/
  12. EdgelessDB, Edgeless https://www.edgeless.systems/products/edgelessdb/

Open Source Projects of Confidential Computing

- Projects on RISCV TEE
  1. Penglai MPU (sPMP) https://penglai-enclave.systems/
  2. Keystone, https://keystone-enclave.org/
  3. OpenTitan, ePMP for RoT chain https://docs.opentitan.org/sw/device/silicon_creator/mask_rom/docs/memory_protection/
  4. ibex core supports ePMP , and Seagate is using ePMP on their cores even before it was ratified. https://github.com/lowRISC/ibex
  5. SiFive provides cores with PMP, similar to IOPMP. https://forums.sifive.com/t/pmp-registers-and-user-mode/3448
  6. HexFive developed multizone, https://github.com/ConfidentialComputing/multizone-sdk
  7. OP-TEE, RISC-V version https://archive.fosdem.org/2021/schedule/event/tee_teep/
  8. RISC-V AP-TEE, https://github.com/riscv-admin/trusted-computing/tree/main/specifications/AP-TEE

AKRAINO

Summary

Technology	Performance	Generability	Security	Description	Maturity	Suggested for Production
Anonymization	High	High	Medium	depends on noise level and data	High	Yes
Differential Privacy	High	Low	Medium	depends on noise level and data	Increasing	No
Fully Homomorphic Encryption	Low	Medium	High	high cost of computation and low cost of communication	Increasing	No
Zero Knowledge Proof	Low	Low	High	used in secure authentication protocols	Increasing	Yes
Secure Multi-Party Computation	Low to Medium	High	High	high cost of computation and communication	High	No
Confidential Computing through TEE	High	High	medium to high	need to trust hardware providers	Increasing, expect to mature in about one year	Yes

Privacy Preserving Standards, Regulations and Laws

Current Standards

ISO/IEC 29101:2013 (Information technology – Security techniques – Privacy architecture framework) is one of the oldest standards efforts that handles secure computing. It presents architectural views for information systems that process personal data and show how Privacy Enhancing Technologies such as secure computing, but also pseudonymisation, query restrictions and more could be deployed to protect Personally Identifiable Information.

ISO/IEC 29101 pre-dates the European General Data Protection Regulation (GDPR), so it does not include all the latest knowledge on secure computing and its role in regulation. For example, it is unaware of the view of anonymised processing and using secure computing might actually not be processing in the sense of the law.

ISO/IEC 19592-1:2016 (Information technology – Security techniques – Secret sharing – Part 1: General) focuses on the general model of secret sharing and the related terminology. It introduces properties that secret sharing schemes could have, e.g. the homomorphic property that is a key aspect for several MPC systems.

ISO/IEC 19592-2:2017 (Information technology – Security techniques – Secret sharing – Part 2: Fundamental mechanisms) introduces specific schemes. It starts with the classic ones like Shamir and replicated secret sharing. All schemes are systematically described using the terms and properties from Part 1. There were originally plans to have more parts for this standard that would describe MPC paradigms, but work has not started yet.

Source: UN Handbook for Privacy Preserving Techniques https://docs.google.com/document/d/1GYu6UJI81jR8LgooXVDsYk1s6FlM-SbOvo3oLHglFhY/edit#

Standards in Progress

ISO/IEC 18033-6 (Information technology security techniques – Encryption algorithms – Part 6: Homomorphic encryption) is a standard on homomorphic encryption schemes

Given the more conservative nature of ISO/IEC when it comes to encryption schemes, it is attempting to focus on the ones with multiple known industrial uses. However, as it is still work in progress, it is unclear how it will turn out in the end.

The **Homomorphic Encryption Standardization Initiative**[22] is an open standardisation initiative for fully homomorphic encryption with participants from industry, government and academia. The initiative attempts to build broad community agreement on security levels, encryption parameters, encryption schemes, core library API, and eventually the programming model, with the goal of driving adoption of this technology.

ISO/IEC 20889 (Privacy enhancing data de-identification terminology and classification of techniques) is another project that approaches privacy technologies a bit differently. This project will result in a standard that describes ways to turn identifiable data into de-identified data. Here, the choices include various noise-based techniques, cryptographic techniques and more.

Source: UN Handbook for Privacy Preserving Techniques https://docs.google.com/document/d/1GYu6UJI81jR8LgooXVDsYk1s6FIM-SbOvo3oLHglFhY/edit#

Privacy Preserving: Challenges and Opportunities to Use in Production

Challenges in Production

1. Reduce computation and communication overhead on encrypted data for resource limited nodes/parties, for example, data size after FHE expands to 1 to 4 orders of magnitude. The bottlenecks lie in the node/party which has the smallest computation resource and network resource
2. Federated learning requires all nodes/parties to be online, perform collaborative and synchronized compute and communication. Thus avoiding straggler-nodes/parties and synchronization is important.
3. Interconnectivity between different nodes/parties is important. Due to different privacy preserving algorithms, communication between nodes/parties require a common protocol.
4. Avoiding reinviting wheels, reducing overhead for deployment cluster with different cloud providers. Share common APIs for parties who stores their data on various platforms.

Future Work in Production

1. Optimization on algorithms, de-couple computation modules, re-orchestration of calculation modules.
2. Reduce communication overhead by reducing communication frequency, and communication data size.
3. Tracing and identification of resource limited nodes/parties, define the traces to be collected.
4. Integrity preserving tracing and logging system for federated learning platforms.
5. Using C++/C and compiler optimization for accelerating algorithm implementation.
6. Using hardware acceleration (GPU, TPU, FPGA, ASIC) for acceleration of new encryption technologies and privacy preserving related protocols.
7. Pipeline scheduling for data consumption, data read/write, data encryption/decryption, data transportation, computation and storage.
8. Safety and security measurements and grading for the federated learning platforms.
9. Automatic evaluation and standardization of federated learning platforms for its performance.
10. Enhance explanatory of federated learning.

PipelineDP OpenMined

Wenhui Zhang — Bytedance Inc — wenhui.zhang@bytedance.com
Abinav Ravi Venkatakrishnan — deepc GmbH — subramathreya@gmail.com
Chinmay Shah — Openmined — cs@chinmayshah.xyz

OpenMined Org

OpenMined (https://www.openmined.org/) is a group of open-source developers that is focused on developing **privacy preserving technologies** tools.
OpenMined's mission is to make **privacy preserving machine learning (PPML)** accessible.

AKRAINO

Differential Privacy

Who are using differential Privacy?

- Apple
- Microsoft
- Google
- LinkedIn
- Uber
- US Census Bureau

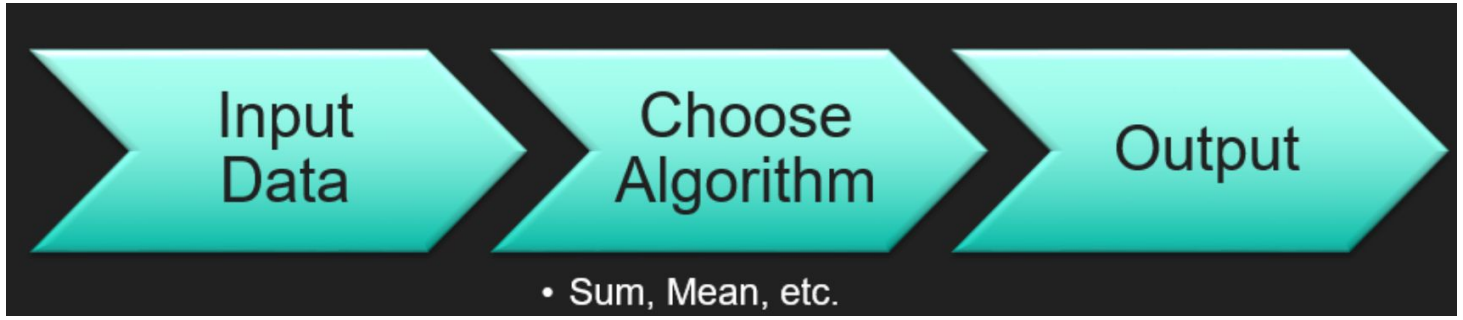
Example Application using differential Privacy?

- Google Chrome (RAPPOR)
- Google Maps
- COVID Mobility Report
- COVID-19 Vaccination Insights

AKRAINO

Differential Privacy Based Machine Learning Process

Original Machine Learning Process



Differential Privacy Based Machine Learning Process

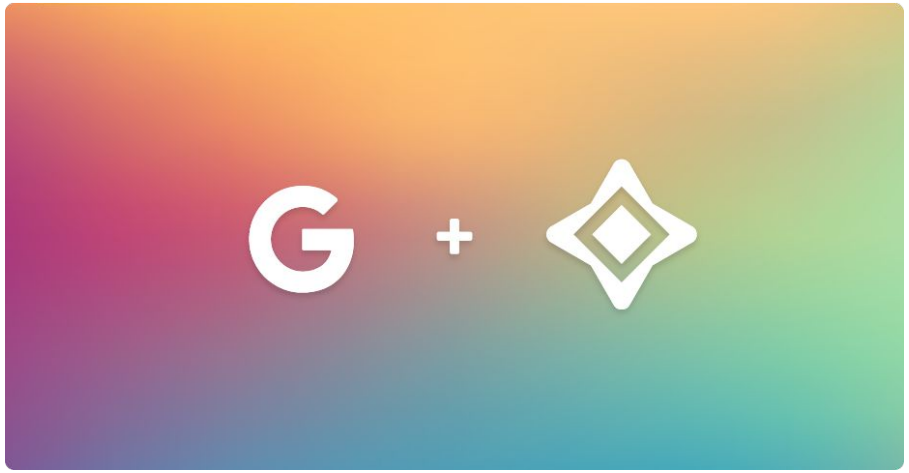
AKRAINO

PipelineDP

One of the most important **privacy preserving technologies** in the **privacy preserving technologies** ecosystem is **Differential privacy.**

PipelineDP (https://pipelinedp.io/) is an open source tool to build **differential privacy** on private data pipelines.

Based on **PyDP** to provide provide **output privacy.**



https://blog.openmined.org/announcing-pipelinedp-an-api-for-applying-differential-privacy-in-production/

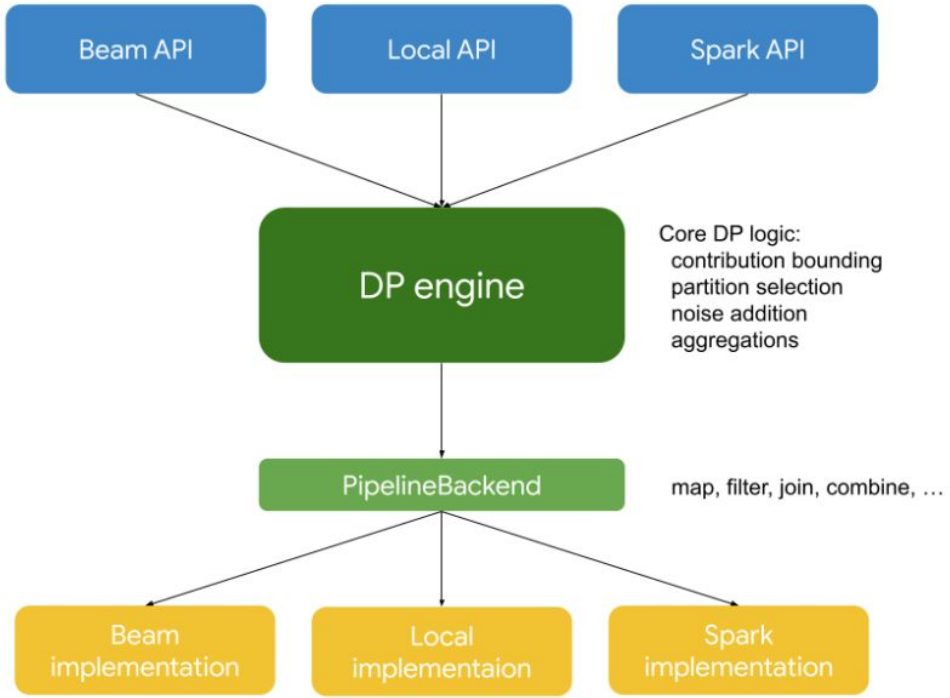
AKRAINO

PipelineDP Architecture

PipelineDP process large datasets using batch processing, supports **Apache Beam, Apache Spark etc.**

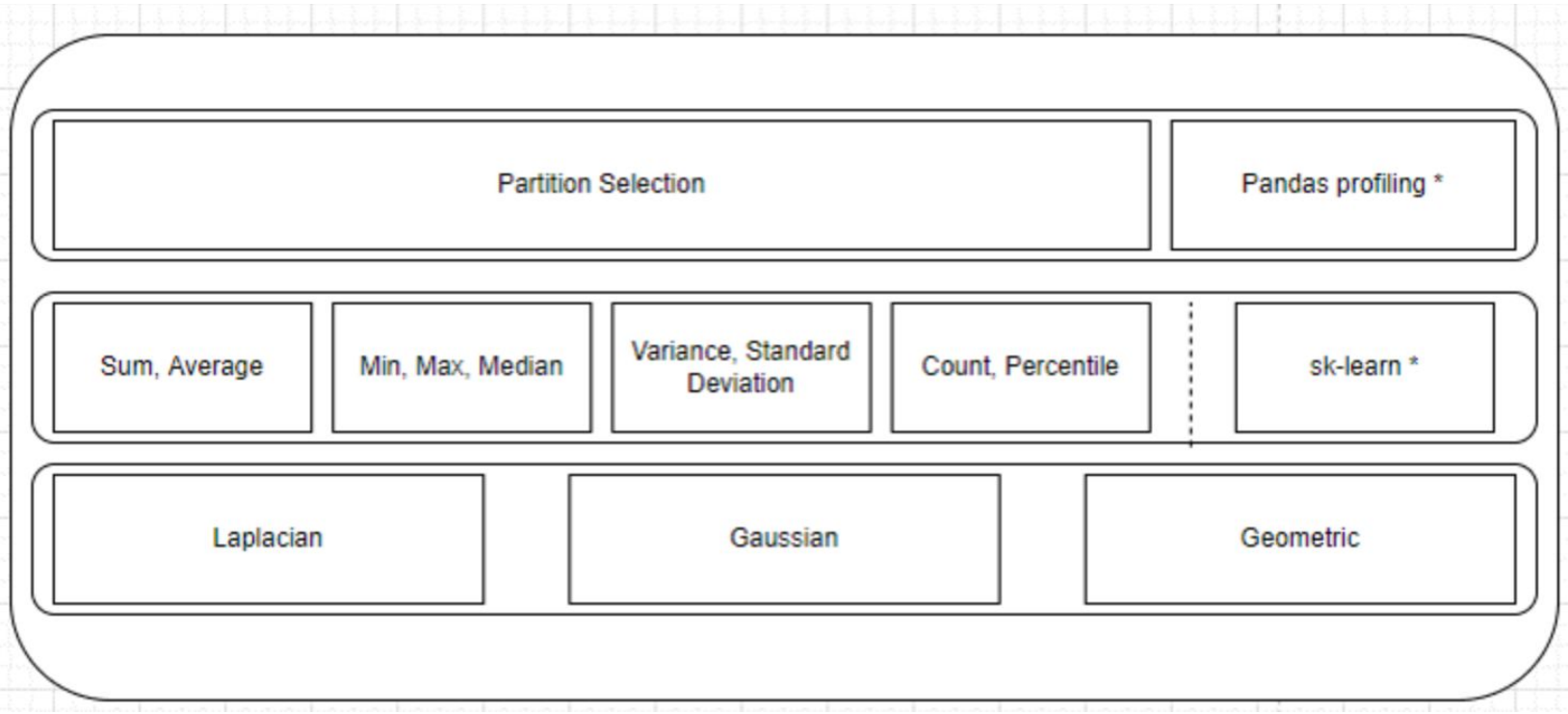
DPEngine is the heart of PipelineDP. Developers can use DPEngine directly, there are also developer-facing APIs that resemble regular (non-private) APIs of the popular frameworks.

PipelineBackend is an abstraction for low-level data processing operations (map, join, combine, filter, etc.). It enabled connection with data located in **Apache Beam, Apache Spark or even locally**.



AKRAINO

DPEngine



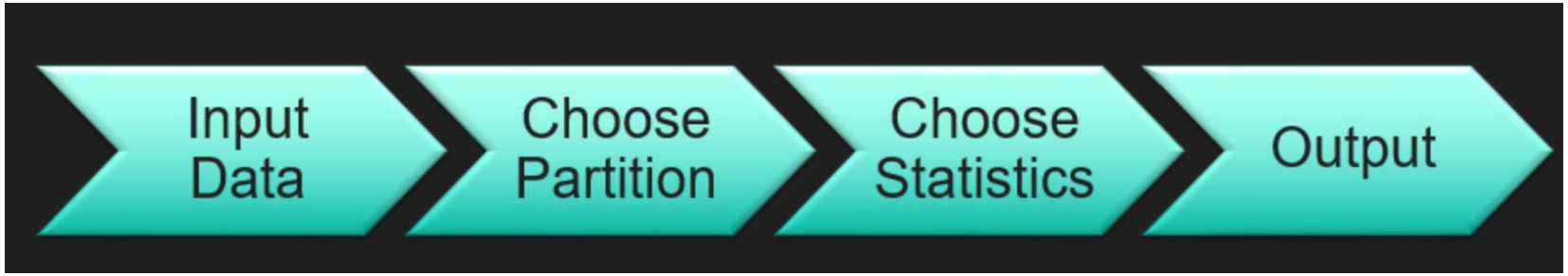
- Encapsulates the complexities of differential privacy, such as:
  - protecting outliers and rare categories,
  - generating safe noise,
  - privacy budget accounting.
- Supports many standard computations, such as count, sum and average.

Ref: https://github.com/google/differential-privacy/blob/main/common_docs/Differential_Privacy_Computations_In_Data_Pipelines.pdf

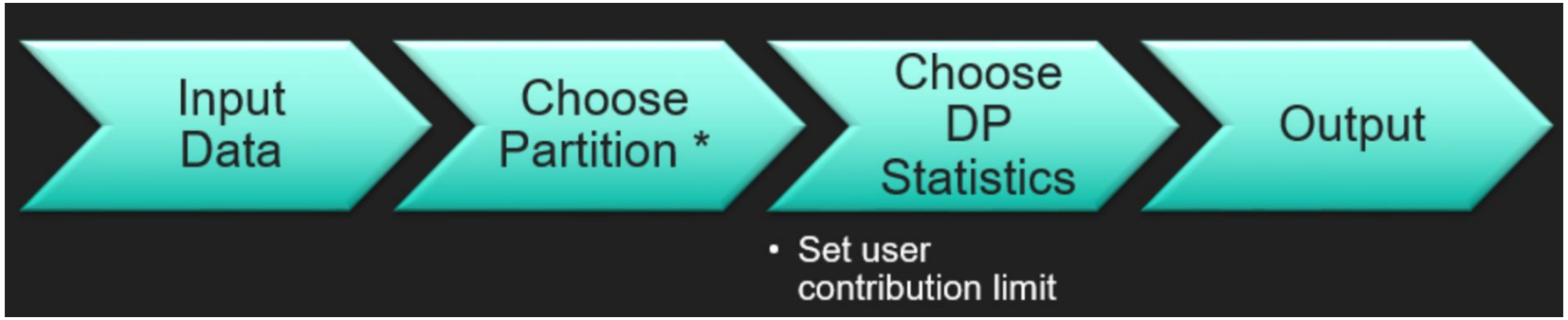
AKRAINO

Data Pipeline with Differential Privacy

Original Data Pipeline



Data Pipeline with Differential Privacy



AKRAINO

How to Use

1.Install Python and run on the command line `pip install pipeline-dp pyspark absl-py`

2.Run python python run_on_beam.py --input_file=<path to data.txt from 3> --output_file=<...>

```
from absl import app
from absl import flags
import pyspark
import pipeline_dp
from pipeline_dp.private_spark import make_private
from pipeline_dp import SumParams
from common_utils import parse_partition
from common_utils import delete_if_exists
```

How to Use

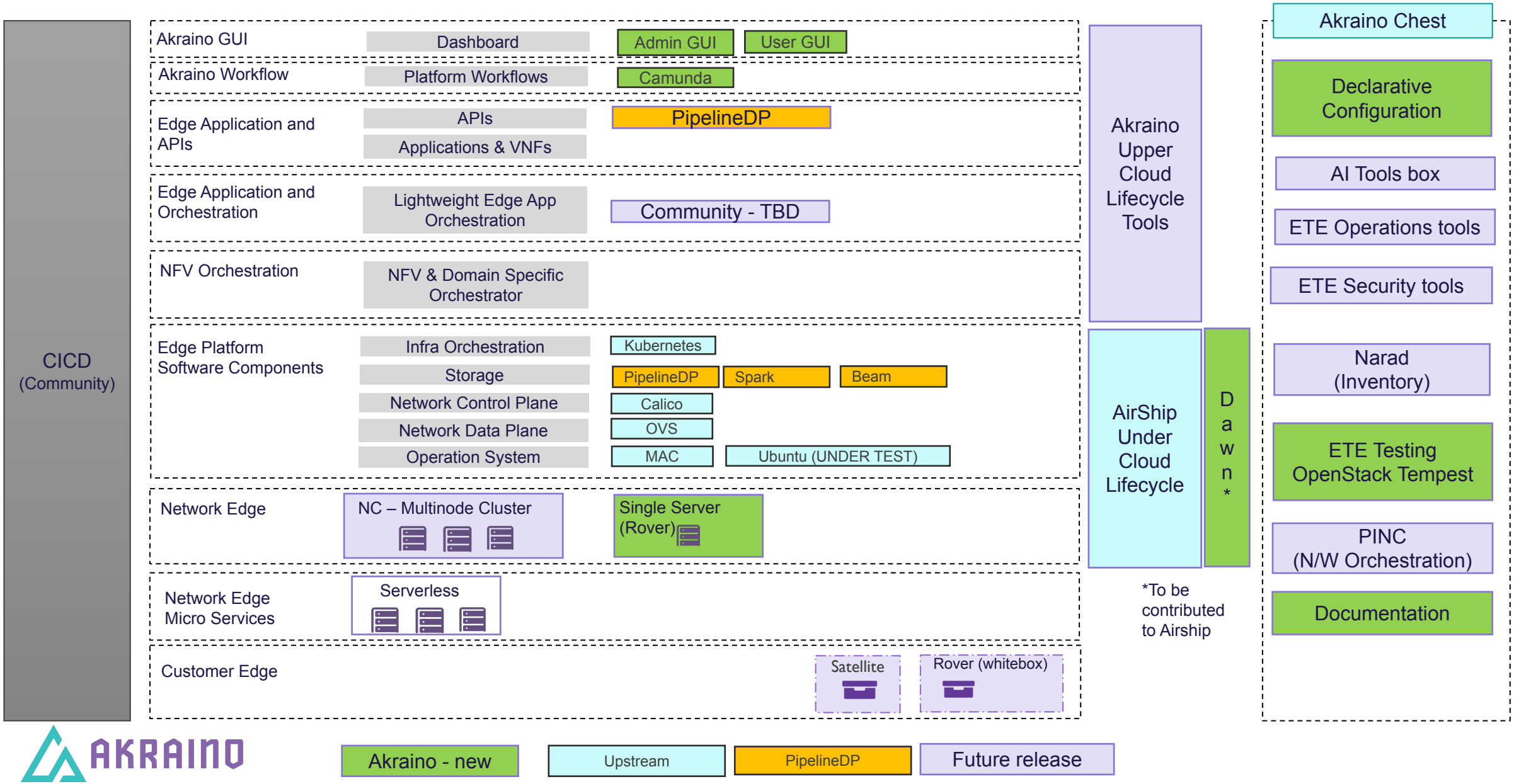
```
master = "local[1]"  # use one worker thread to load the file as 1 partition
conf = pyspark.SparkConf().setMaster(master)
sc = pyspark.SparkContext(conf=conf)
movie_views = sc \
    .textFile(FLAGS.input_file) \
    .mapPartitions(parse_partition)

# Define the privacy budget available for our computation.
budget_accountant = pipeline_dp.NaiveBudgetAccountant(total_epsilon=1,
                                                      total_delta=1e-6)

# Wrap Spark's RDD into its private version
private_movie_views = \
    make_private(movie_views, budget_accountant, lambda mv: mv.user_id)

# Calculate the private sum
dp_result = private_movie_views.sum(
    SumParams(
        # Limits to how much one user can contribute:
        # .. at most two movies rated per user
        max_partitions_contributed=2,
        # .. at most one rating for each movie
        max_contributions_per_partition=1,
        # .. with minimal rating of "1"
        min_value=1,
        # .. and maximum rating of "5"
        max_value=5,
        # The aggregation key: we're grouping by movies
        partition_extractor=lambda mv: mv.movie_id,
        # The value we're aggregating: we're summing up ratings
        value_extractor=lambda mv: mv.rating))

budget_accountant.compute_budgets()

# Save the results
dp_result.saveAsTextFile(FLAGS.output_file)
```

Blueprint Proposal: Data Privacy Blueprint Family

Case Attributes	Description	Informational
Type	Data Privacy Blueprint Family – PipelineDP	
Blueprint Family - Proposed Name	Data Privacy Blueprint Family	
Use Case	Provide FaaS (Function as a Service) for Differential Privacy for Serverless Applications	
Blueprint proposed Name	PipelineDP	
Initial POD Cost (capex)	Unicycle less than $150k: 3 Arm bare metal machines, 1 10G switch	
Scale & Type	For the smallest deployment, this requires 2 Arm bare metal machines. For large deployments, this could span to large number of bare metal machines.	
Applications	Differentially private data aggregation for large scale online education, telemedicine, Hospitals, Govs, Teleco and Schools.	
Power Restrictions	Less than 10Kw	
Infrastructure orchestration	Kubeless Docker 1.13.1 or above and K8s 1.10.2 or above- Container Orchestration OS - MAC/Linux Under Cloud Orchestration - Airship v1.0	
SDN	OVS	
Workload Type	Here are some examples of how to use PipelineDP: • Apache Spark example • Apache Beam example • Framework-free example • Example with all frameworks	

AKRAINO

Data Privacy Blueprint Family – PipelineDP

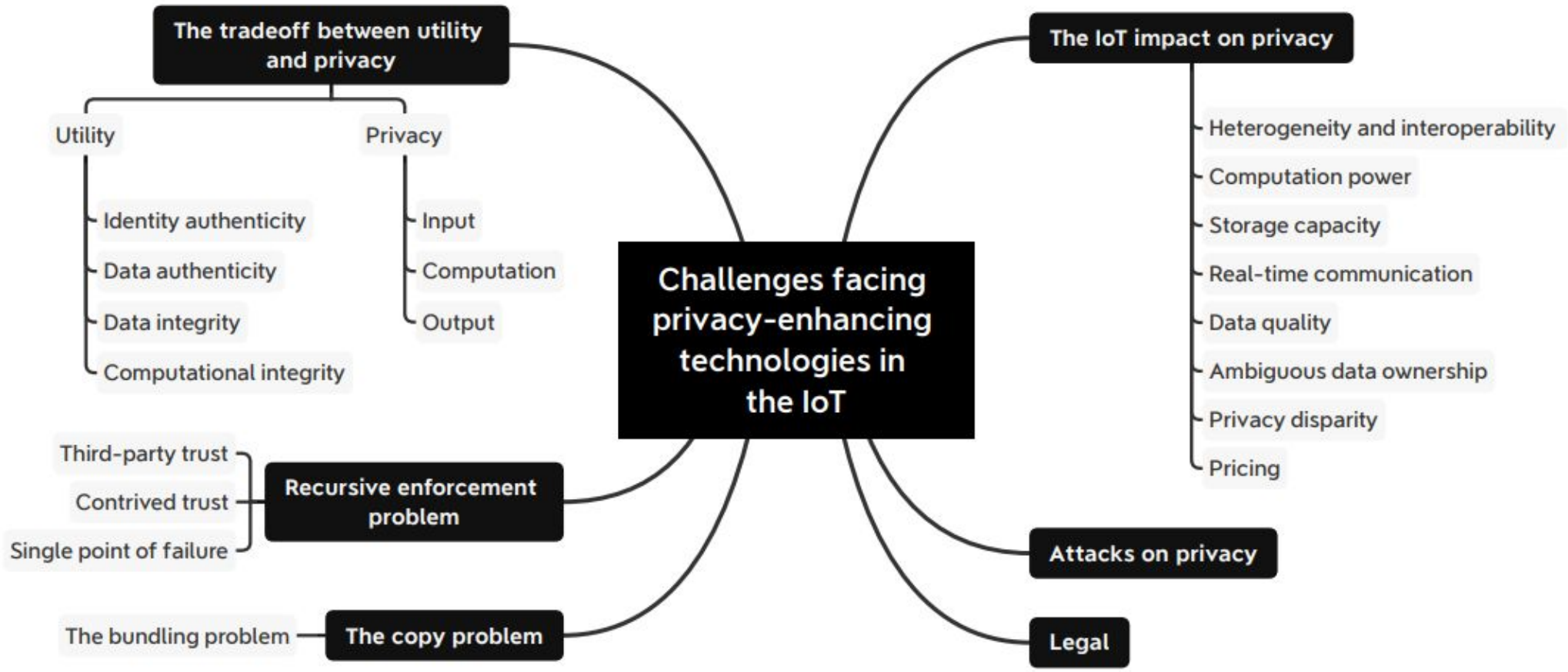


PipelineDP Code

- Github: https://github.com/OpenMined/PipelineDP
- Website: https://pipelinedp.io/
- API: https://pipelinedp.io/api-documentation/index.html
- Utility analysis: https://github.com/OpenMined/PipelineDP/tree/main/utility_analysis
- Proposal: https://wiki.akraino.org/display/AK/OpenMined+PipelineDP

Thanks for listening

Opportunities and Challenges



Source: https://blog.openmined.org/classifying-the-challenges-of-privacy-enhancing-technologies-pets-in-iot-data-markets/

Classification of challenges in Production

- The Narrow Challenges
- The Broad Challenges

The Narrow Challenges

- The Utility and privacy trade-off:
  - A dilemma of preserving the privacy and deriving useful insight from data
  - PETs ensure plausible deniability, but could reduce data authenticity
- The Recursive Enforcement Problem (REP):
  - Manifests itself in a multi-layered supervision structure
  - Issue of trust and cost of third party supervision affects PETs maturity
- The Copy Problem (CP):
  - One could lose control of data as it becomes tradable asset
  - Fear of potentially forgoing the benefits derived from shared data

The Broad Challenges

- The attacks on privacy:
  - Data re-identification
  - Could be any of data forwarding, roles collision, or side channel attacks
- The Legal challenges:
  - Laws appears reactionary rather than proactive
  - Stringent privacy regulation could stifle free markets and innovations
- The IoT impact on privacy
  - Constrained by such factors as context and employed technologies
  - More details on the next slide

The Shortcomings of Edge Computing

- Heterogeneity and Interoperability
- Computation power
- Storage capacity and real-time communication
- Data quality
- Ambiguous data ownership
- Privacy disparity
- Pricing

AKRAINO

Summary of Challenges of Edge Computing

- Establish thresholds that balance the trade-off between data authenticity and privacy
- Maturity of PETs could help reduce the recursive enforcement problem and the copy problem