Scalable Privacy-Preserving Computing with High Numerical Precision

Dimitar Jetchev

Chief Technology Officer
Inpher Inc./Sarl
The biggest obstacle to using advanced data analysis isn't skill base or technology; it's plain old access to the data.

- Edd Wilder-James
  Google TensorFlow
in transit (https://)

at rest (on disk)

in use (in memory)
Opportunities beyond data security.

- Monetize analytics (without exposing data)
- Access private data sources to train AI
- Regulatory Compliance (GDPR, PDPA)
- Cross-industry collaboration
- Secure Cloud Computing
ING Belgium Sees Opportunities for ‘Secret’ Sharing of Encrypted Data

Zero-knowledge computing would let companies analyze encrypted information without revealing any secret information.
High-Precision Use Cases
Iridium 33 and Kosmos-2251 Satellite Collision

- Collision - 2009
- 11,700 m/s
- 789 km above Syberia
- More than 2000 debris
- ISS special maneuvers
High-Precision Privacy-Preserving Compute

- Predicting collisions of satellites
- Satellite trajectories are private
- Satellite operators nonetheless perform conjunction analysis
- Need to evaluate non-linear functions with high numerical precision
Detecting Rare Events with Private Data
*see FC’18 paper: High-Precision Privacy-Preserving Real-Valued Function Evaluation

Problem: detect a rare event via a classification algorithm (e.g., a fraudulent activity or a rare market event)

- 1 out of 10,000 bank customers performs fraudulent activity
- A classifier predicting always negative has 99.99% accuracy

<table>
<thead>
<tr>
<th>PREDICTED 1</th>
<th>PREDICTED 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE 1</td>
<td>TP = 0</td>
</tr>
<tr>
<td>TRUE 0</td>
<td>FP = 0</td>
</tr>
</tbody>
</table>

accuracy \approx 1, \text{ recall} = 0, \text{ F1-score} = 0

For classifying rare events, numerical accuracy does matter!
Secure Multiparty Computation
Secret Shared Data
Reveal Secret Shared Data
Multiplications

6 \times 7
Multiplications

\[ \times 7 \]
Multiplications
Multiplications

X
Beaver Multiplication

Goal

- Given $x, y \in G_1, G_2$, compute $x \times y \in G_3$
  - where $\times : G_1 \times G_2 \rightarrow G_3$ is bilinear and public
Beaver Multiplication

Goal

- Given $x, y \in G_1, G_2$, compute $x \times y \in G_3$
  - where $\times: G_1 \times G_2 \rightarrow G_3$ is bilinear and public

Offline phase (independent on the data value)

- Secret share $(\lambda, \mu, \lambda \times \mu)$ for random $\lambda, \mu$
Beaver Multiplication

Goal
- Given \( x, y \in G_1, G_2 \), compute \( x \times y \in G_3 \)
  - where \( \times : G_1 \times G_2 \rightarrow G_3 \) is bilinear and public

Offline phase (independent on the data value)
- Secret share \( (\lambda, \mu, \lambda \times \mu) \) for random \( \lambda, \mu \)

Online phase
- Jointly reveal \( a = x + \lambda \)
- Jointly reveal \( b = y + \mu \)
- Now, everything becomes linear:
  \[ x \times y = a \times b - \lambda \times b - a \times \mu - \lambda \times \mu \]
- And never ever re-use the same triplet twice!
Customer-hosted Analyst Platform (cloud or on-prem)

Inpher-hosted XOR Service (never exposed to data)

Data Source 1

Data Source 2

Data Source n...

Analyst submits operations to XOR Service and selects data sources
Customer-hosted Analyst Platform (cloud or on-prem) 

Inpher-hosted XOR Service (never exposed to data)

frontend (API, UI) 

compiler 

trusted dealer 

gear 

Data Source 1 

Data Source 2 

Data Source n...

Analyst submits operations to XOR Service and selects data sources
Customer-hosted Analyst Platform
(cloud or on-prem)

Inpher-hosted XOR Service
(never exposed to data)

Data Source 1

Data Source 2

Data Source n...

Operations compiled into a ‘circuit’ and distributed as a binary
Customer-hosted Analyst Platform (cloud or on-prem)

Inpher-hosted XOR Service (never exposed to data)

Offline Phase: Random triplets are generated and distributed
Customer-hosted Analyst Platform (cloud or on-prem)

Inpher-hosted XOR Service (never exposed to data)

Online Phase: Data sources secretly compute with random numbers
Customer-hosted Analyst Platform (cloud or on-prem)

Inpher-hosted XOR Service (never exposed to data)

Partial results sent to Analyst Platform to construct final output
Arithmetic with Real Numbers
### Floating-point representation of the reals

- Addition is generally inefficient compared to multiplication
- Boolean circuit approach is slow in MPC

### Fixed-point representation

- Overflows often blindly destroy the result

### Fixed-point arithmetic with native FP backend

**Pros**
- Overflows do not destroy the result
- **float128** addition and multiplication: native on some hardware

**Cons**
- Small impact on security
- Statistical masking instead of perfect masking
Statistical Masking

Masking $x = \pm 1, \sigma = 1$.  

Masking $x = \pm 1, \sigma = 100$. 
Disadvantages of Real Arithmetic with Floating Point Backend

➢ Large-depth arithmetic circuits require multi-precision floating-point numbers (masking sizes grow with depth)

➢ **Computational security** as opposed to **information-theoretic security**

➢ Memory overhead in both offline and online phases.

➢ Matrix operations over multi-precision floating types may be expensive
### Representing Real Numbers

#### Floating-point representation
- \( x = m \cdot 2^\tau \), \( m \in 2^{-\rho} \cdot \mathbb{Z} \), \( 1/2 \leq |m| \leq 1 \)
- \( \tau = \lceil \log_2 |x| \rceil \) - exponent; data dependent, not MPC-friendly, not public

#### Fixed-point representation
- \( x = m \cdot 2^\tau \) with \( m \in 2^{-\rho} \cdot \mathbb{Z} \), \( 0 \leq |m| < 1 \)
- \( \tau \) is public, MPC-friendly
- Might **overflow** if \( \tau \) is too small
- Might **underflow** if \( \tau \) is too large

#### Addition is more subtle than you might think!
- Given \((m_1, \tau_1)\) and \((m_2, \tau_2)\) as well as \(\tau\)
- Compute \( m \cdot 2^\tau = m_1 \cdot 2^{\tau_1} + m_2 \cdot 2^{\tau_2} \) with a numerical window of size \(\rho\)
- Addition requires right-shift and roundings (non-linear operations)
Overflows and Underflows

The class of a value is known in compile/runtime, but not the value itself.

Plaintext Overflow

We know that $3 \in \mathcal{P}_{2,-2}$, but neither $3 + 3$, nor $3 \times 3$ is in $\mathcal{P}_{2,-2}$, i.e.,

- $\mathcal{P}_{\text{pmsb, plsb}}$ is closed neither under addition nor under multiplication.
- Compute (at compile time) a sharp upper bound $\text{pmsb}$ for the result.

Plaintext Underflow

We know $x = 1 \times 10^{-3}, y = 6 \times 10^{-3} \in \mathcal{P}_{0,-14}$.

- If resulting class is still $\mathcal{P}_{0,-14}$, we lose the result

$$0.0010 \times 0.0060 = 0.0000$$

- Need for computing a suitable $\text{plsb}$ at compile time
Casting Between Classes and Arithmetic Operations

➢ One of the same real number can be represented with different parameters
  ○ mantissa, exponent, numerical window

➢ Need methods to cast from one representation to another

➢ Casts are key for implementing addition and multiplication
Fourier Approximation of Real-Valued Functions
Approximation of Non-Linear Functions

\[ \text{sig}(t) = \frac{1}{1+e^{-t}} \]
Deficiency of Polynomial Approximation

Limitations of polynomial approximation

- Evaluation requires multiple rounds
- Overflows are difficult to manage
- Approximation diverges quickly outside of the domain
- Uniform approximation fails (Runge's phenomenon)
Chebyshev Polynomials and Polynomial Approximations

**Chebyshev Polynomials**

- Polynomial $T_n$ of degree $n$ such that $T_n(\cos \theta) = \cos n\theta$
- Optimal solution to the Runge phenomenon in degree $n$
Naive Fourir Approximation

Naive approach via Fourier series

\[ a_n = \frac{1}{2B} \int_{-B}^{+B} f(x) e^{\frac{\pi inx}{B}} \, dx \]

- Consider \( f(x) = \sum_{k=-n}^{n} a_n e^{\frac{\pi inx}{B}} \)

- Gibbs phenomenon is an issue
Privacy-Preserving Evaluation of Fourier Series

### Fourier Triplets

- Fourier series can be evaluated in MPC
- \( h(x) = \sum_{i=-n}^{n} a_k e^{ikx} \Rightarrow \text{triplets replaced} \ (\lambda, e^{\pm i\lambda}, \ldots, e^{\pm in\lambda}) \)

### Periodic extensions on larger intervals

To evaluate a function \( f \) on an interval \([-B, B]\):
- Extend \( f \) to a periodic function on \([-2B, 2B]\)
- Consider the Fourier series of the extension
Rapidly Convergent Uniform Approximation

\( f: [-\pi/2, \pi/2] \rightarrow \mathbb{R} \) - not necessarily smooth or periodic (square-integrable)

**The approximation problem**

Given \( n \), define

\[
G_n = \left\{ g(x) = \frac{a_0}{2} + \sum_{k=1}^{n} a_k \sin kx + \sum_{k=1}^{n} b_k \cos kx \right\}.
\]

of \( 2\pi \)-periodic functions. Consider

\[
g_n(x) = \arg\min_{g \in G_n} \| f - g \|_{L^2_{[-\pi/2, \pi/2]}}.
\]

- Coefficients with respect to the standard basis are numerically unstable (diverge as \( n \rightarrow \infty \))
- Coefficients with respect to a basis of Chebyshev polynomials (first and second kind) is stable; exponential convergence
Building Logistic Regression Models

Requires a privacy-preserving evaluation of sigmoid function:

\[ h_\theta(x) = \frac{1}{1 + e^{-\theta \cdot x}}, \]

as well as minimization of the cost function:

\[ J(\theta) = \frac{1}{N} \sum_{i=1}^{N} \text{cost}(h_\theta(x^{(i)}, y^{(i)})) \],

where

\[ \text{cost}(h_\theta(x, y)) = -y \log(h_\theta(x)) - (1 - y) \log(1 - h_\theta(x)). \]
Logistic Regression - 1M x 50 - 3 Players

<table>
<thead>
<tr>
<th></th>
<th>[FC2018]</th>
<th>Xor v2.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abs precision</td>
<td>2.3e-5</td>
<td>1.7e-8</td>
</tr>
<tr>
<td>Offline Time</td>
<td>04:06:43</td>
<td>00:36:40</td>
</tr>
<tr>
<td>Offline RAM</td>
<td>276G</td>
<td>6.87G</td>
</tr>
<tr>
<td>Triplets pp.</td>
<td>51.2G</td>
<td>92.0G</td>
</tr>
<tr>
<td>Online (100 MBps)</td>
<td>03:18:19</td>
<td>00:24:30</td>
</tr>
<tr>
<td>Online RAM</td>
<td>192G</td>
<td>10.1G</td>
</tr>
<tr>
<td>Online Comm pp.</td>
<td>20G</td>
<td>66.8</td>
</tr>
</tbody>
</table>
Methods for Compiling Privacy-Preserving Programs
Why a Compiler?

AUTOMATION AND STATIC ANALYSIS

➢ Computation requires auxiliary random masking data (offline phase)
➢ Random data generated from distributions with specific parameters
➢ \Longrightarrow \text{ need for static analysis of distributions (statistical calculator)} (compile time)
➢ Optimizations (memory/communication)

CONSTRUCTS SPECIFIC TO MPC

➢ Type checking, ANF, SSA (standard phases of compilation)
➢ Inline function definitions
➢ Unroll for loops with bounded number of iterations
def solve(A: Matrix, b: Vector): Vector {
    var nrows: Int = xor.rows(A);
    var ncols: Int = xor.cols(A);

    var P: Matrix = xor.orthrand(nrows, ncols, -6);
    var Q: Matrix = xor.orthrand(nrows, ncols, -6);

    var PAQ: Matrix = P * A * Q;
    var Pb: Vector = P * b;

    xor.reveal(PAQ);
    xor.reveal(Pb);
    var r: Vector = xor.publicSolve(PAQ, Pb);
    return Q * r;
}

def linreg(y: Vector, X: Matrix): Vector {
    var A: Matrix = xor.transpose(X) * X;
    var b: Vector = xor.transpose(X) * y;
    return solve(A, b);
}

def main() {
    var X: Matrix = xor.input("X");
    var y: Vector = xor.input("y");
    var theta: Vector = linreg(y, X);
    xor.output(theta, "thetas");
}
def solve(A: Matrix, b: Vector): Vector {
    var nrows: Int = xor.rows(A);
    var ncols: Int = xor.cols(A);
    var P: Matrix = xor.orthrand(nrows, ncols, -6);
    var Q: Matrix = xor.orthrand(nrows, ncols, -6);
    var PAQ: Matrix = P * A * Q;
    var Pb: Vector = P * b;
    xor.reveal(PAQ);
    xor.reveal(Pb);
    var r: Vector = xor.publicSolve(PAQ, Pb);
    return Q * r;
}

def linreg(y: Vector, X: Matrix): Vector {
    var A: Matrix = xor.transpose(X) * X;
    var b: Vector = xor.transpose(X) * y;
    return solve(A, b);
}

def main() {
    var X: Matrix = xor.input("X");
    var y: Vector = xor.input("y");
    var theta: Vector = linreg(y, X);
    xor.output(theta, "thetas");
}
<table>
<thead>
<tr>
<th>Index</th>
<th>Phase</th>
<th>Brief</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>parse</td>
<td>Parses the program.</td>
</tr>
<tr>
<td>1</td>
<td>namer</td>
<td>Enters symbols for all top-level functions.</td>
</tr>
<tr>
<td>2</td>
<td>typer</td>
<td>Checks that the program is well typed, and performs some basic type inference.</td>
</tr>
<tr>
<td>3</td>
<td>anf</td>
<td>Transforms the program in A-normal form.</td>
</tr>
<tr>
<td>4</td>
<td>ssa</td>
<td>Transforms the tree into its Static Single-Assignment (SSA) form.</td>
</tr>
<tr>
<td>5</td>
<td>inline</td>
<td>Inlines all functions called by the main function into its body.</td>
</tr>
<tr>
<td>6</td>
<td>tuple-elimination</td>
<td>Eliminates tuples creation and projections.</td>
</tr>
<tr>
<td>7</td>
<td>copy-propagation</td>
<td>Removes Intermediate variables from assign-chains, and other unused variables.</td>
</tr>
<tr>
<td>8</td>
<td>constant-fold</td>
<td>Performs basic partial evaluation and folds constants through the program.</td>
</tr>
<tr>
<td>9</td>
<td>dimension-checking</td>
<td>Checks the dimensions of all matrices, and ensures that the operations are valid.</td>
</tr>
<tr>
<td>10</td>
<td>desugaring</td>
<td>Transforms a user-level program into one that operates on MPC-level primitives only.</td>
</tr>
<tr>
<td>11</td>
<td>visibility</td>
<td>Tracks and sets the visibility of all values.</td>
</tr>
<tr>
<td>12</td>
<td>plaintext-param</td>
<td>Computes plaintext parameters pMsb and plsB.</td>
</tr>
<tr>
<td>13</td>
<td>mask-resolution</td>
<td>Resolves masking parameters from plaintext parameters.</td>
</tr>
<tr>
<td>14</td>
<td>builtin-params-resolution</td>
<td>Computes extra parameters specific to builtin apps.</td>
</tr>
<tr>
<td>15</td>
<td>codegen</td>
<td>Generates a compiled-program.bin that can be executed by an appropriate backend.</td>
</tr>
</tbody>
</table>
Linear Regression - Assembly

: ...  
6: CreateContainer(V6, FlMR<2,9,7,-43>);
7: BeaverMod(PriV1, PriV1, V6, AW=(29,-20), BW=(29,-20), W=(9,-40), Pairing=4);
8: CreateContainer(V8, FlMR<2,15,13,-37>);
9: BeaverMod(PriV1, PriV3, V8, AW=(35,-20), BW=(35,-20), W=(15,-40), Pairing=4);
10: {
11: CreateContainer(V11, FlMR<2,2,0,-6>);
12: RandomOrthogonalMatrix(V11);
13: CreateContainer(V13, FlMR<2,2,0,-6>);
14: RandomOrthogonalMatrix(V13);
15: CreateContainer(V15, FlMR<2,14,12,-38>);
16: BeaverMod(PriV11, PriV6, V15, AW=(52,-6), BW=(20,-38), W=(14,-44), Pairing=3);
17: CreateContainer(V17, FlMR<2,15,13,-37>);
18: BeaverMod(PriV15, PriV13, V17, AW=(21,-37), BW=(52,-6), W=(15,-43), Pairing=3);
19: CreateContainer(V19, FlMR<2,16,14,-36>);
20: BeaverMod(PriV11, PriV8, V19, AW=(52,-6), BW=(22,-36), W=(16,-42), Pairing=3);
21: Reveal(V17);
22: Reveal(V19);
23: CreateContainer(V23, FlMR<2,1,26,24,-26>);
24: PublicSolve(V17, V19, V23);
25: CreateContainer(V25, FlMR<2,1,27,25,-25>);
26: BeaverMod(PriV13, PubV23, V25, AW=(52,-6), BW=(33,-25), W=(27,-31), Pairing=3);
27: } 
: ...
Linear Regression - Assembly

: ... 
6: CreateContainer(V6, FlMR<2,2,9,7,-43>);
7: BeaverMod(PriV1, PriV1, V6, AW=(29,-20), BW=(29,-20), W=(9,-40), Pairing=4);
8: CreateContainer(V8, FlMR<2,1,15,13,-37>);
9: BeaverMod(PriV1, PriV3, V8, AW=(35,-20), BW=(35,-20), W=(15,-40), Pairing=4);
10: {
11: CreateContainer(V11, FlMR<2,2,2,0,-6>);
12: RandomOrthogonalMatrix(V11);
13: CreateContainer(V13, FlMR<2,2,2,0,-6>);
14: RandomOrthogonalMatrix(V13);
15: CreateContainer(V15, FlMR<2,2,14,12,-38>);
16: BeaverMod(PriV11, PriV6, V15, AW=(52,-6), BW=(20,-38), W=(14,-44), Pairing=3);
17: CreateContainer(V17, FlMR<2,2,15,13,-37>);
18: BeaverMod(PriV15, PriV13, V17, AW=(21,-37), BW=(52,-6), W=(15,-43), Pairing=3);
19: CreateContainer(V19, FlMR<2,1,16,14,-36>);
20: BeaverMod(PriV11, PriV8, V19, AW=(52,-6), BW=(22,-36), W=(16,-42), Pairing=3);
21: Reveal(V17);
22: Reveal(V19);
23: CreateContainer(V23, FlMR<2,1,26,24,-26>);
24: PublicSolve(V17, V19, V23);
25: CreateContainer(V25, FlMR<2,1,27,25,-25>);
26: BeaverMod(PriV13, PubV23, V25, AW=(52,-6), BW=(33,-25), W=(27,-31), Pairing=3);
27: }
: ...
Linear Regression - Assembly

... CreateContainer(V6, FlMR<2,2,9,7,-43>);
6: BeaverMod(PriV1, PriV1, V6, AW=(29,-20), BW=(29,-20), W=(9,-40), Pairing=4);
7: CreateContainer(V8, FlMR<2,1,15,13,-37>);
8: BeaverMod(PriV1, PriV3, V8, AW=(35,-20), BW=(35,-20), W=(15,-40), Pairing=4);
9: 
10: {
11: CreateContainer(V11, FlMR<2,2,2,0,-6>);
12: RandomOrthogonalMatrix(V11);
13: CreateContainer(V13, FlMR<2,2,2,0,-6>);
14: RandomOrthogonalMatrix(V13);
15: CreateContainer(V15, FlMR<2,2,14,12,-38>);
16: BeaverMod(PriV11, PriV6, V15, AW=(52,-6), BW=(20,-38), W=(14,-44), Pairing=3);
17: CreateContainer(V17, FlMR<2,2,15,13,-37>);
18: BeaverMod(PriV15, PriV13, V17, AW=(21,-37), BW=(52,-6), W=(15,-43), Pairing=3);
19: CreateContainer(V19, FlMR<2,1,16,14,-36>);
20: BeaverMod(PriV11, PriV8, V19, AW=(52,-6), BW=(22,-36), W=(16,-42), Pairing=3);
21: Reveal(V17);
22: Reveal(V19);
23: CreateContainer(V23, FlMR<2,1,26,24,-26>);
24: PublicSolve(V17, V19, V23);
25: CreateContainer(V25, FlMR<2,1,27,25,-25>);
26: BeaverMod(PriV13, PubV23, V25, AW=(52,-6), BW=(33,-25), W=(27,-31), Pairing=3);
27: }
28: ...

...
Linear Regression - Assembly

: ...  
6: CreateContainer(V6, FlMR<2,2,9,7,-43>);  
7: BeaverMod(PriV1, PriV1, V6, AW=(29,-20), BW=(29,-20), W=(9,-40), Pairing=4);  
8: CreateContainer(V8, FlMR<2,1,15,13,-37>);  
9: BeaverMod(PriV1, PriV3, V8, AW=(35,-20), BW=(35,-20), W=(15,-40), Pairing=4);  
10: {  
11: CreateContainer(V11, FlMR<2,2,2,0,-6>);  
12: RandomOrthogonalMatrix(V11);  
13: CreateContainer(V13, FlMR<2,2,2,0,-6>);  
14: RandomOrthogonalMatrix(V13);  
15: CreateContainer(V15, FlMR<2,2,14,12,-38>);  
16: BeaverMod(PriV11, PriV6, V15, AW=(52,-6), BW=(20,-38), W=(14,-44), Pairing=3);  
17: CreateContainer(V17, FlMR<2,2,15,13,-37>);  
18: BeaverMod(PriV15, PriV13, V17, AW=(21,-37), BW=(52,-6), W=(15,-43), Pairing=3);  
19: CreateContainer(V19, FlMR<2,1,16,14,-36>);  
20: BeaverMod(PriV11, PriV8, V19, AW=(52,-6), BW=(22,-36), W=(16,-42), Pairing=3);  
21: Reveal(V17);  
22: Reveal(V19);  
23: CreateContainer(V23, FlMR<2,1,26,24,-26>);  
24: PublicSolve(V17, V19, V23);  
25: CreateContainer(V25, FlMR<2,1,27,25,-25>);  
26: BeaverMod(PriV13, PubV23, V25, AW=(52,-6), BW=(33,-25), W=(27,-31), Pairing=3);  
27: }  
: ...
CreateContainer(V6, FlMR<2,2,9,7,-43>);
BeaverMod(PriV1, PriV1, V6, AW=(29,-20), BW=(29,-20), W=(9,-40), Pairing=4);
CreateContainer(V8, FlMR<2,1,15,13,-37>);
BeaverMod(PriV1, PriV3, V8, AW=(35,-20), BW=(35,-20), W=(15,-40), Pairing=4);
{
    CreateContainer(V11, FlMR<2,2,2,0,-6>);
    RandomOrthogonalMatrix(V11);
    CreateContainer(V13, FlMR<2,2,2,0,-6>);
    RandomOrthogonalMatrix(V13);
    CreateContainer(V15, FlMR<2,2,14,12,-38>);
    BeaverMod(PriV11, PriV6, V15, AW=(52,-6), BW=(20,-38), W=(14,-44), Pairing=3);
    CreateContainer(V17, FlMR<2,2,15,13,-37>);
    BeaverMod(PriV15, PriV13, V17, AW=(21,-37), BW=(52,-6), W=(15,-43), Pairing=3);
    CreateContainer(V19, FlMR<2,1,16,14,-36>);
    BeaverMod(PriV11, PriV8, V19, AW=(52,-6), BW=(22,-36), W=(16,-42), Pairing=3);
    Reveal(V17);
    Reveal(V19);
    CreateContainer(V23, FlMR<2,1,26,24,-26>);
    PublicSolve(V17, V19, V23);
    CreateContainer(V25, FlMR<2,1,27,25,-25>);
    BeaverMod(PriV13, PubV23, V25, AW=(52,-6), BW=(33,-25), W=(27,-31), Pairing=3);
}
Compiling to Preserve Privacy - Beyond MPC

- Fully homomorphic encryption (FHE)
- Garbled circuits (GC)
- Multi-party computation (MPC)
- Differential privacy (DP)
- Privacy budget (PB)
- Federated learning (FL)
## Other Applications - Compiling FHE Programs

**Bootstrapping:** an operation used to reduce noise in the plaintext in order to continue processing the ciphertext in a computation

### Ring-LWE-based FHE Schemes

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TFHE</td>
<td>suitable for boolean circuits, automata evaluations</td>
</tr>
<tr>
<td>B/FV</td>
<td>suitable for integer arithmetic, arithmetic circuits</td>
</tr>
<tr>
<td>HEAAN</td>
<td>suitable for vectorized operations with real numbers</td>
</tr>
<tr>
<td>CHIMERA</td>
<td>scheme switching method</td>
</tr>
</tbody>
</table>

### Numerical Parameters in the Context of FHE

- $L$ - level exponent of ciphertext - related to the maximal number of homomorphic operations before performing a bootstrapping
- $\rho$ - numerical window (as in the MPC setting)
- $\tau$ - exponent (only know a sufficiently good bound at compile time)
Thank you!