Juan R. Troncoso, co-founder & CEO

juan@tuneinsight.com

RSA uses large finite (abelian) groups
$G = (\mathbb{Z}/n\mathbb{Z})^x$ (2048 bits, 4096 bits,...)
To speed things up:

- **Elliptic curve crypto** uses smaller groups, whose operations are more expensive.
- **Lattice cryptography** uses larger groups, but whose operations are much cheaper.

Lattice-based cryptography is the use of conjectured hard problems on point lattices in $\mathbb{R}^n$ as the foundation for secure cryptographic systems.
Features:

- Apparent resistance to *quantum* attacks (in contrast with most number-theoretic cryptography) [Sho97]
- High asymptotic efficiency and parallelism
- Security under *worst-case* intractability assumptions [Ajt96]
- Versatile and powerful cryptographic objects (FHE [Gen09], ABE [BGG+14], Code obfuscation [GGH+13]…)

Main Milestones in Lattice Cryptography

1982: First use of lattices in cryptanalysis (LLL): knapsack cryptosystems

1996: First crypto schemes based on hard lattice problems: NTRU, Ajtai-Dwork, GGH,...

2009: Fully-Homomorphic Encryption on Lattices

2012: Leveled cryptosystems

What is a Lattice?

A lattice is the set of all integer linear combinations of (linearly independent) basis vectors

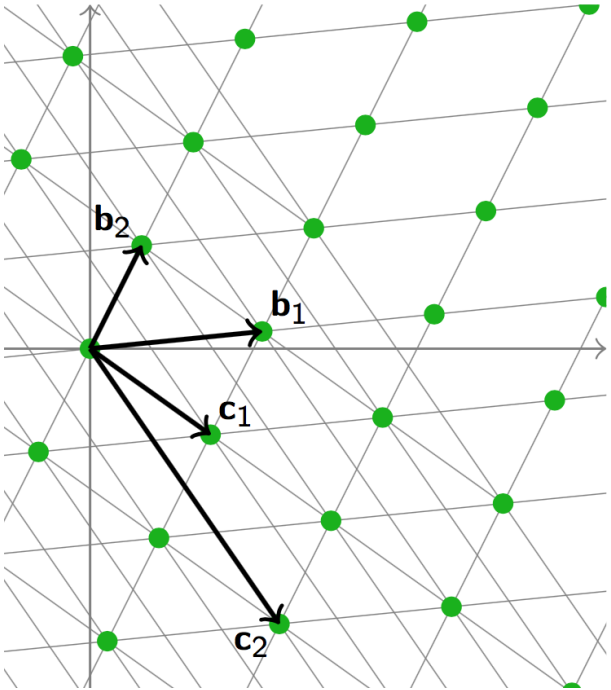
$$\boldsymbol{B} = \{\boldsymbol{b}_1, \boldsymbol{b}_2, \dots, \boldsymbol{b}_n\} \subset \mathbb{R}^n$$

$$\mathcal{L} = \sum_{i=1}^{n} \boldsymbol{b}_i \cdot \mathbb{Z} = \{\boldsymbol{B}\boldsymbol{x} : \boldsymbol{x} \in \mathbb{Z}^n\}$$

The same lattice has many different bases:

$$\mathcal{L} = \sum_{i=1}^{n} \boldsymbol{c}_i \cdot \mathbb{Z}$$

Lattice: discrete additive subgroup of $\mathbb{R}^n$



Simple Example (Preliminary Homomorphic Encryption)

Good bases and bad bases: GGH (Goldreich, Goldwasser, Halevi) family

Two lattice bases

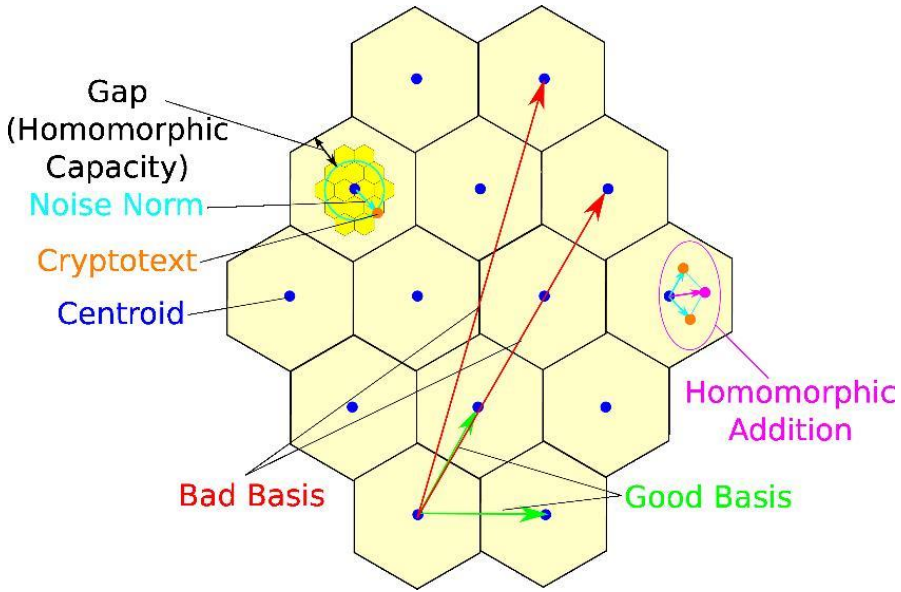
- "Good" basis ($\boldsymbol{B}$, private key)
- "Bad" basis ($\boldsymbol{H}$, public key, Hermite Normal Form)

Encryption of m: $\mathbf{c} = E(m) = \boldsymbol{v} + \boldsymbol{n}[m]$ (lattice point + noise)

Decryption: $D(\boldsymbol{c})$: $\hat{\boldsymbol{v}} = \boldsymbol{B}[\boldsymbol{B}^{-1}\boldsymbol{c}]$

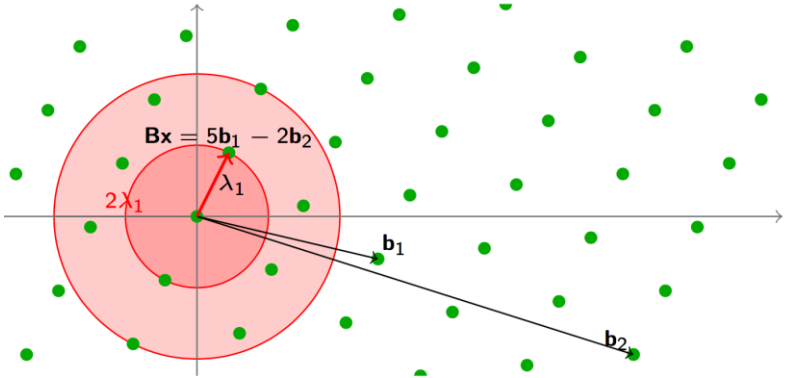
Homomorphism:

$$\boldsymbol{c}_1 + \boldsymbol{c}_2 = (\boldsymbol{v}_1 + n[m_1]) + (\boldsymbol{v}_2 + n[m_2])$$
$$= \boldsymbol{v}_3 + n[m_1 + m_2]$$

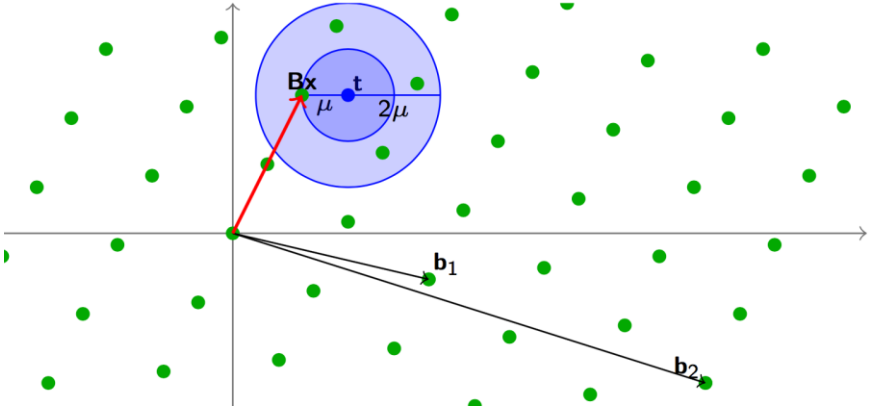


Base Lattice Problems (ex: SVP, CVP)

Shortest Vector Problem SVP_γ
Given a lattice $\mathcal{L}(\boldsymbol{B})$, find a (nonzero) lattice vector $\boldsymbol{Bx}, \boldsymbol{x} \in \mathbb{Z}^k$ of length (at most) $\|\boldsymbol{Bx}\| \leq \gamma\lambda_1$



Closest Vector Problem CVP_γ
Given a lattice $\mathcal{L}(\boldsymbol{B})$ and a target point $\boldsymbol{t}$, find a lattice vector $\boldsymbol{Bx}$ within distance $\|\boldsymbol{Bx} - \boldsymbol{t}\| \leq \gamma\mu$



Ring-LWE distribution: For an $s \in R_q$ (the secret), the ring-LWE distribution $A_{s,\chi}$ over $R_q \times R_q$ is sampled by choosing $a \in R_q$ uniformly at random, $e \leftarrow \chi$, and outputting

$$(a, b = s \cdot a + e \bmod q)$$

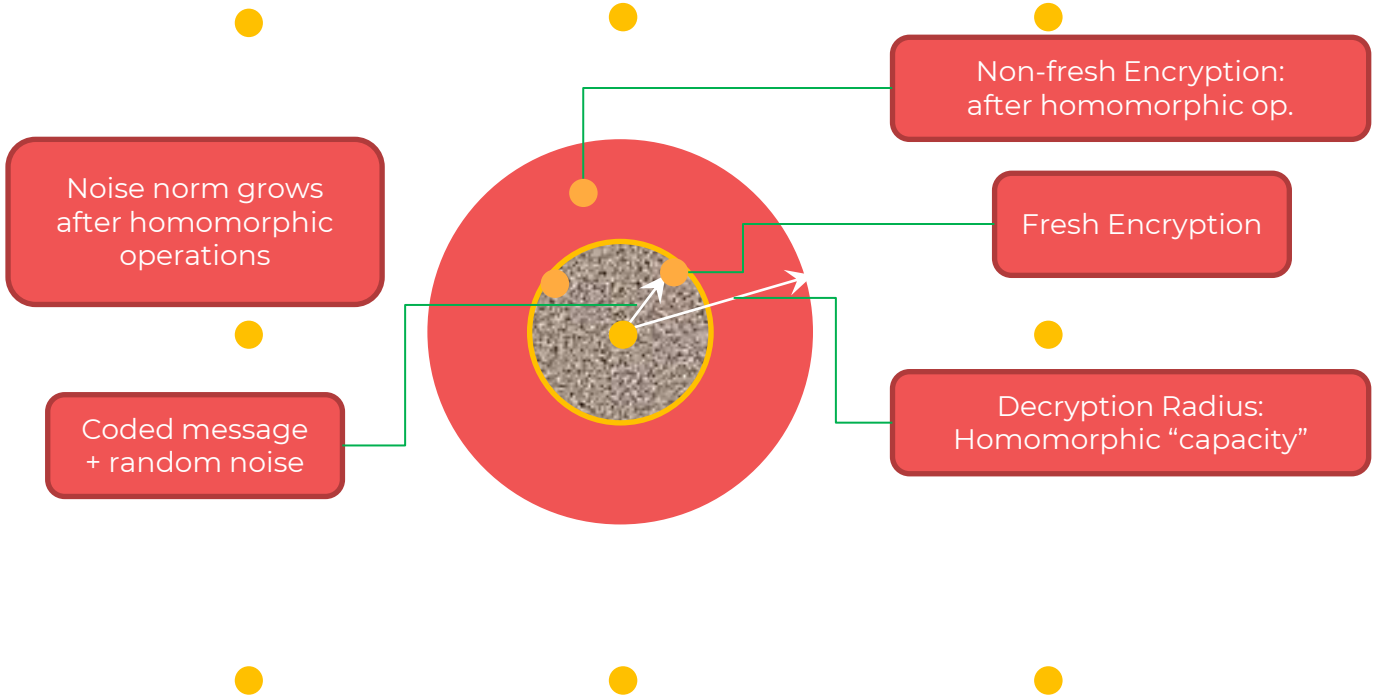
Decision-R-LWE: Given m independent samples $(a_i, b_i) \in R_q \times R_q$ where every sample is distributed according to either:

- $A_{s,\chi}$ for a uniformly random $s \in R_q$ (fixed for all samples),
- The uniform distribution

Distinguish which is the case (with non-negligible advantage)

Normal form: secret from $s \leftarrow \chi$
More efficient than LWE (smaller m and FFT-like polynomial products)
Reduction of $RLWE_{q,\chi,m}$ to quantum SVP_γ [LPR10]

Noise management is essential in homomorphic cryptosystems



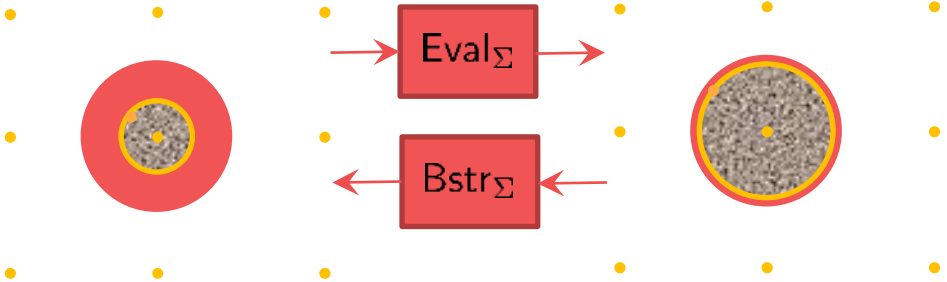
How to build Homomorphic Cryptosystems from R-LWE (Somewhat vs Fully HE)

FHE

KeyGen_Σ
Enc_Σ
Dec_Σ
Eval_Σ
Bstr_Σ HD

$$c = \mathsf{Eval}_\Sigma(pk, f, (c_1, \ldots, c_n)) \equiv \mathsf{Enc}_\Sigma(pk, f(m_1, \ldots, m_n))$$

Only valid when f is of depth < L



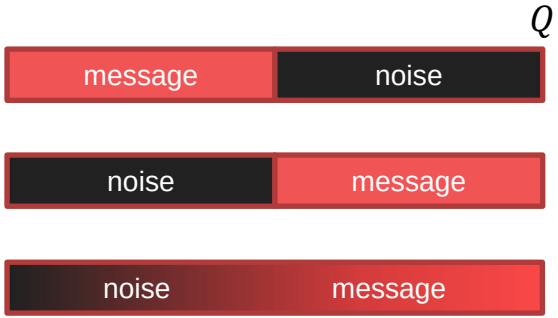
If Dec (squashed) has depth < L

$$\mathsf{Bstr}_\Sigma(pk, [\![m]\!]_{pk}) = \mathsf{Eval}_\Sigma(pk, \mathsf{Dec}_\Sigma, [\![m]\!]_{pk})$$

Common characteristics of modern RLWE cryptosystems:

- Cyclotomic polynomial $f(x) = x^n + 1$, n power of two
- Ciphertext modulus $\mathrm{Q} = \prod q_i$
- Ring $\mathrm{R_Q} = \mathbb{Z}_Q / (f(x))$
- Error distribution χ with power $\|\chi\| < B$
- Plaintext modulus $\ll Q$, scale factor Δ
- Key generation:
- Secret key: $s \leftarrow \chi$
- Public key is an RLWE sample: E.g., $(a_0 = -(a_1 s + e), a_1)$, with $a_1 \leftarrow R_Q$, $e \leftarrow \chi$
- Encryptions are vectors of polynomials in R_Q, with the encoded message
- The decryption function is of the form

$$\sum_{i=0}^{v} c_i s^i$$



Efficiently using lattice cryptosystems: packing in the coefficient domain

All the encryptions over RLWE work with polynomials of degree d
Each coefficient is a plaintext slot in $\mathbb{Z}_Q$: $\boldsymbol{a} \equiv a \equiv \sum_{i=0}^{d-1} a_i x^i$
SIMD homomorphic operations:

Polynomial addition

$$E(\boldsymbol{a}) + E(\boldsymbol{b}) = E\left(\sum_{i=0}^{d-1} a_i x^i\right) + E\left(\sum_{i=0}^{d-1} b_i x^i\right) = \mathrm{E}(\sum_{i=0}^{d-1} (a_i + b_i) x^i)$$

Polynomial multiplication (modular polynomial $f(x) = x^d + 1$)

$$E(\boldsymbol{a}) \cdot E(\boldsymbol{b}) = E\left(\sum_{i=0}^{d-1} a_i x^i\right) \cdot E\left(\sum_{i=0}^{d-1} b_i x^i\right) = \mathrm{E}\left(\sum_{i=0}^{d-1}\left(\sum_{j=0}^{i}\left(a_j \cdot b_{i-j}\right)\right)x^i - \sum_{i=0}^{d-2}\left(\sum_{j=i+1}^{d-1}\left(a_j \cdot b_{d+i-j}\right)\right)x^i\right)$$

Nega-cyclic homomorphic convolution: $E(c') = E(a') \cdot E(b')$

After decryption: $c = \sum_{i=0}^{d-1} c_i' (-1)^{-\frac{i}{d}} x^i \Longrightarrow c = \sum_{i=0}^{d-1}\left(\sum_{j=0}^{i}\left(a_j \cdot b_{i-j}\right)\right)x^i + \sum_{i=0}^{d-2}\left(\sum_{j=i+1}^{d-1}\left(a_j \cdot b_{d+i-j}\right)\right)x^i$

Efficiently using lattice cryptosystems: packing in the slot domain

Use an automorphism as message coding that switches domain
Equivalent to an NTT (Number Theoretic Transform) or DFT (Discrete Fourier Transform)
Ex. DFT (for inputs $x, X \in \mathbb{C}^d$)

$$DFT[x] = X_k = \sum_{n=0}^{d-1} x_n \cdot e^{-\frac{j2\pi kn}{d}}$$

$$DFT^{-1}[X] = x_n = \frac{1}{d} \sum_{k=0}^{d-1} X_k \cdot e^{\frac{j2\pi kn}{d}}$$

Important properties:

- Circular convolution: $DFT^{-1}[X \cdot Y]_n = \sum_{i=0}^{n} x_i y_{n-i} + \sum_{i=n+1}^{d-1} x_i y_{d+n-i} = x_n \circledast y_n$
- Duality: $DFT[x \cdot y]_k = \frac{1}{d}\left(\sum_{i=0}^{n} X_i Y_{n-i} + \sum_{i=n+1}^{d-1} X_i Y_{d+n-i}\right) = \frac{1}{d} X_k \circledast Y_k$
- Parceval's theorem: $\sum_{n=0}^{d-1} x_n y_n^* = \frac{1}{d} \sum_{k=0}^{d-1} X_k Y_k^*$

Homomorphic operations become component-wise when the message is in the slot domain

Function evaluation: polynomial approximations

Ring operations are additions and products

Non-polynomial functions have to be:
- approximated by a polynomial
- run on universal gates nand / xor with binary arithmetic

Let $f(x)$: $[a, b] \subset R \rightarrow R$, with $c \in [a, b]$
- **Taylor approximation**: Error bounded, but not uniform in $[a, b]$
  Preferred when input distribution is denser around c (e.g., Gaussian)

- **Least-squares approximation**: minimizes average square error in $[a, b]$
  Preferred when input is uniformly distributed and high homomorphic capacity

- **Chebyshev approximation**: Bounded maximum error, converges with d to the minimax polynomial that minimizes this maximum approximation error $[a, b]$.
  Preferred to avoid overflows and for numerical stability

Practical example: evaluating a Logistic regression under encryption with Tune Insight's Python cryptolib

Evaluation of a Logistic regression prediction

$$y_i = \mu\left(\beta_0 + \sum_{j=1}^{k} a_{i,j}\beta_j\right)$$

For a dataset with l records and k features

Practical example: evaluating a Logistic regression under encryption with Tune Insight's Python cryptolib

1. Parameterization and cryptosystem instantiation

```
from tuneinsight.cryptolib.hefloat import hefloat

# Parameterization: scale/precision and circuit depth
log_scale = 45                                  # Fixed-point arithmetic floating point scaling factor in bits (log2(Delta))
levels = 7                                      # Circuit depth
log_qi = [log_scale+5] + levels*[log_scale]     # 5 additional bits for the lowest level, to account for plaintext growth
log_pi = [log_scale+5]                          # Auxiliary module used for relinearization (usually, at least of the same size as the lowest level q0)

# In order to generate an instance of the cryptosystem, the RLWE ring degree is automatically chosen to ensure at least 128-bit of security
# A context stores the scheme cryptographic parameters and a key generator
context = hefloat.new_context(log_qi = log_qi, log_default_scale= log_scale, log_pi = log_pi)

#Print some information about the cryptographic parameters
print(f'Log2 N: {context.parameters.log_n()}')
print(f'Log2 Moduli Chain: Q{log_qi} + P{log_pi}')
print(f'Log2 QP: {context.parameters.log_q() + context.parameters.log_p()}')
print(f'Log2 Slots: {context.parameters.log_slots()}')
print(f'Available Depth: {levels}')
```

```
Log2 N: 14
Log2 Moduli Chain: Q[50, 45, 45, 45, 45, 45, 45, 45] + P[50]
Log2 QP: 414.9999999903431
Log2 Slots: 13
Available Depth: 7
```

Practical example: evaluating a Logistic regression under encryption with Tune Insight's Python cryptolib

2. Key generation

```
# Generate a fresh secret key
sk = context.new_secret_key()

# Instantiate an evaluator with a relinearization key
# The relinearization key is at public-evaluation key required to ensure ciphertext x ciphertext compactness
# The resulting evaluator object contains only public information and can be freely shared
evaluator = context.new_evaluator(context.new_relinearization_key(sk))
```

3. Polynomial approximation of the activation function

```
import numpy.polynomial.chebyshev as chebyshev
import numpy as np

# Expected interval of the encrypted values after the scalar product
a = -12
b = 12

# Interpolates the Sigmoid in the interval [-12, 12] and returns the coefficients
# for the Chebyshev approximation polynomial in the Chebyshev basis
coeffs = chebyshev.chebinterpolate(lambda x: 1/(1+np.exp(-((b-a)/2 * x + (b+a)/2))), 63)
```

Practical example: evaluating a Logistic regression under encryption with Tune Insight's Python cryptolib

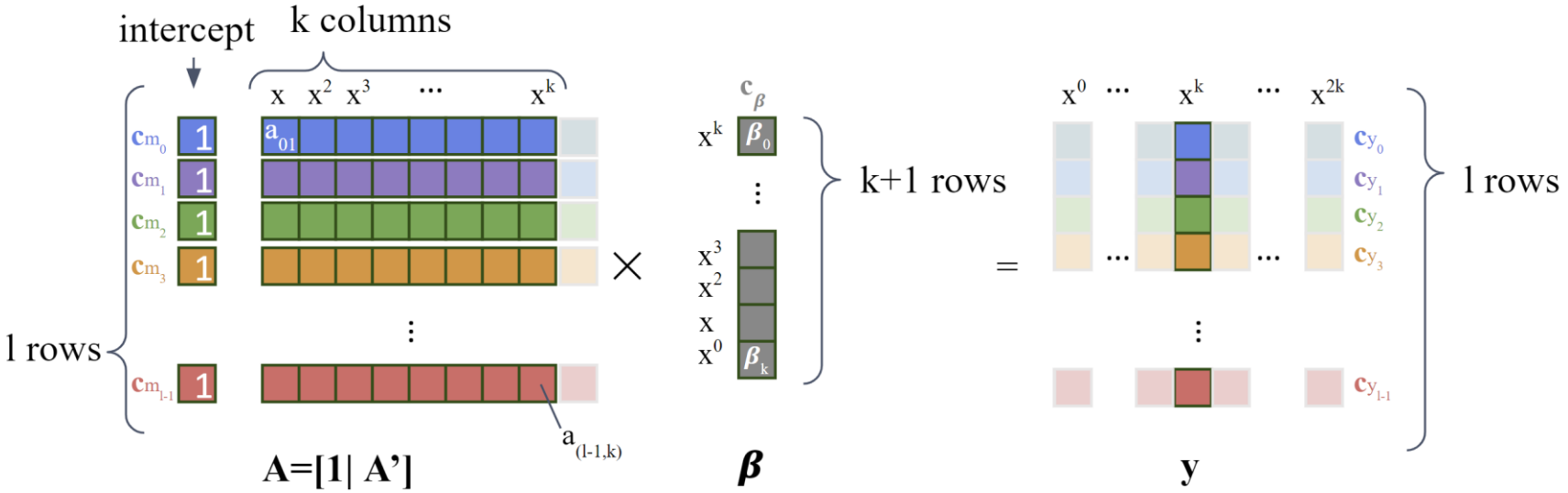
4. Synthetic data generation

```
## Synthetic data generation:
# Number of samples to process in parallel (available plaintext slots that one encryption can hold)
batch_size = context.slots()

# Number of features (k=200)
features = 200

# Generate random data in [-0.5, 0.5]. This is the matrix A'
data = np.random.rand(batch_size, features)-0.5

# Generate random regression weights in [0, 1]. These represent beta_i, i=1,...,k
weights = np.random.rand(features, 1)

# Generate random bias (intercept coefficient) in [0, 1]. This represents beta_0
bias = np.random.rand(1)
```

Practical example: evaluating a Logistic regression under encryption with Tune Insight's Python cryptolib

5. Packed encryption of all inputs

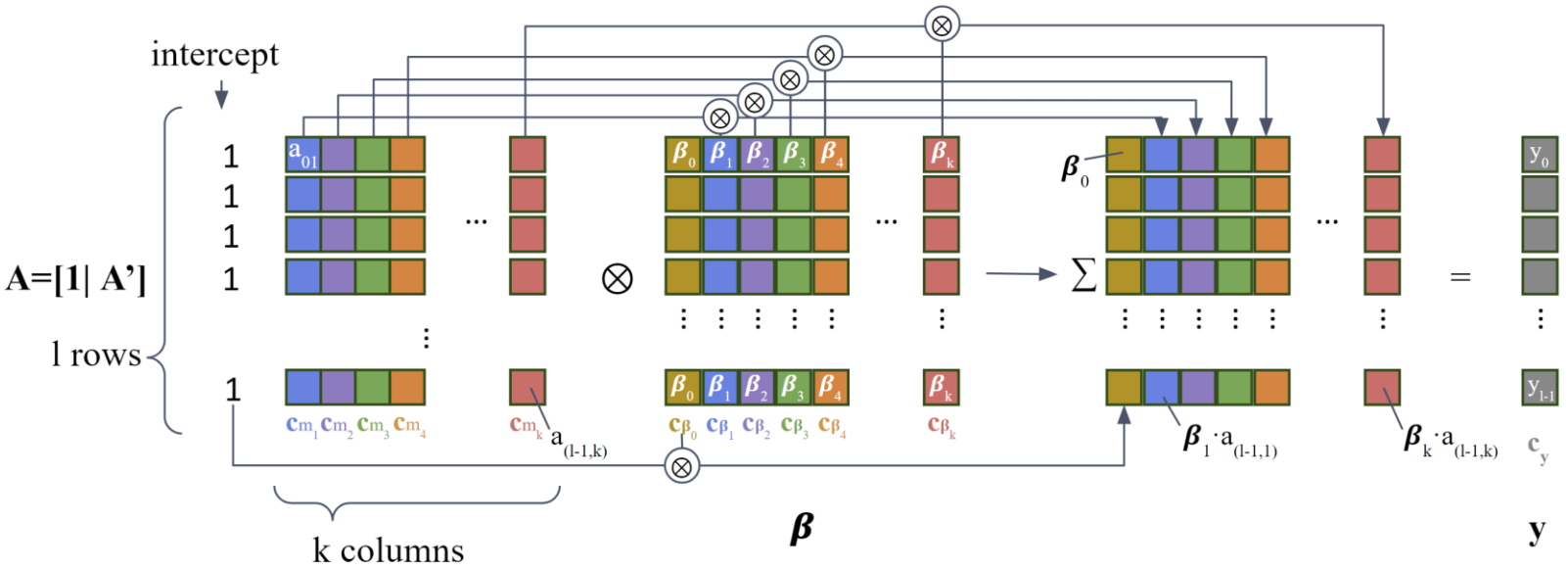
Option 1: horizontal packing



Practical example: evaluating a Logistic regression under encryption with Tune Insight's Python cryptolib

5. Packed encryption of all inputs

Option 2: vertical packing



Practical example: evaluating a Logistic regression under encryption with Tune Insight's Python cryptolib

5. Packed (batched) encryption of all inputs

```
# This optional parameter defines whether the input vectors will be encoded in the coefficients domain (if batched=False)
# or in the slots domain (if batched=True). The latter is the default behavior, and it enables component-wise homomorphic operations
# (additions and products)
batched = True

# The encrypt function can receive a two-dimensional matrix as input, in which case it encrypts each row of the input matrix in one ciphertext.
# Therefore, we transpose the input A', in order to encrypt each column of A' in one ciphertext.
# We need to explicitly make a copy to ensure a correct memory
# alignment when passing C pointers of arrays to the Go wrapper.
# The function returns an object that stores a vector of ciphertexts.
encrypted_data = context.encrypt(data.transpose().copy(), sk, batched)

# As for the regression coefficients, we encrypt each of the weights replicated in all slots of the corresponding ciphertext.
# For this, we apply repetition coding (with tile) and pass the resulting matrix as input to the encrypt function, so that each row is encrypted in a separate ciphertext.
# The result is an object that stores a vector of ciphertexts, each containing one regression coefficient replicated in all its slots.
encrypted_weights = context.encrypt(np.tile(weights, (1, batch_size))* 2/(b-a), sk, batched)

# The intercept coefficient or bias is also encrypted in its own ciphertext, with the same repetition coding as all the other regression coefficients
encrypted_bias = context.encrypt(np.tile(bias, (1, batch_size))* 2/(b-a) + (-a-b)/(b-a), sk, batched)
```

Practical example: evaluating a Logistic regression under encryption with Tune Insight's Python cryptolib

6. Homomorphic evaluation of the model prediction under encryption

```
# Encrypted evaluation of data @ weights computed as np.sum(data.transpose() * np.tile(bias, (1, batch_size)), axis=0)
# This is faster, but equivalent, to doing evaluator.sum(evaluator.mul(encrypted_data, encrypted_weights), axis=0)
encrypted_scalar_product = evaluator.scalar_product(encrypted_data, encrypted_weights)

# Encrypted evaluation of data @ weights + bias
encrypted_scalar_product_plus_bias = evaluator.add(encrypted_bias, encrypted_scalar_product)
```

```
# Encrypted evaluation of sigmoid(data @ weights + bias)
encrypted_prediction = evaluator.polynomial(encrypted_scalar_product_plus_bias, coeffs=coeffs, basis="Chebyshev")
```

Practical example: evaluating a Logistic regression under encryption with Tune Insight's Python cryptolib

7. Decryption of results

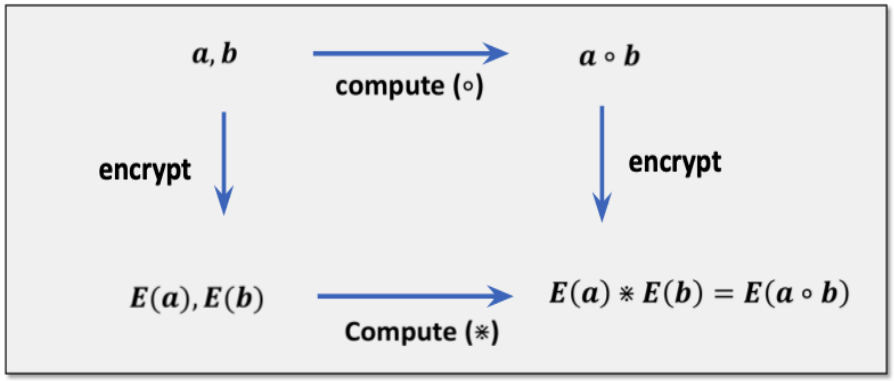
```
# Decrypts the values
prediction = context.decrypt(encrypted_prediction, sk)[:, :batch_size]
```

8. Accuracy comparison with the clear-text process

```
from math import log
# Finally, we evaluate the plaintext circuit
clear_target = 1/(np.exp(-(data @ weights + bias))+1)

# And compare with the decrypted result
print(f'Obtained: {prediction}')
print(f'Clear_tg: {clear_target.transpose()}')
print(f'Precision as -log2(avg_l2(obtained-clear_tg))): {-log(np.sqrt(np.sum((prediction-clear_target.transpose())**2))/batch_size, 2)}')
```

```
Obtained: [[0.87919843 0.17177785 0.22382661 ... 0.99488169 0.06266606 0.45500863]]
Clear_tg: [[0.87919843 0.17177785 0.22382656 ... 0.99488167 0.06266607 0.45500862]] Precision as -log2(avg_l2(obtained-clear_tg))):
31.818855691640856
```

Recap on Homomorphic Encryption



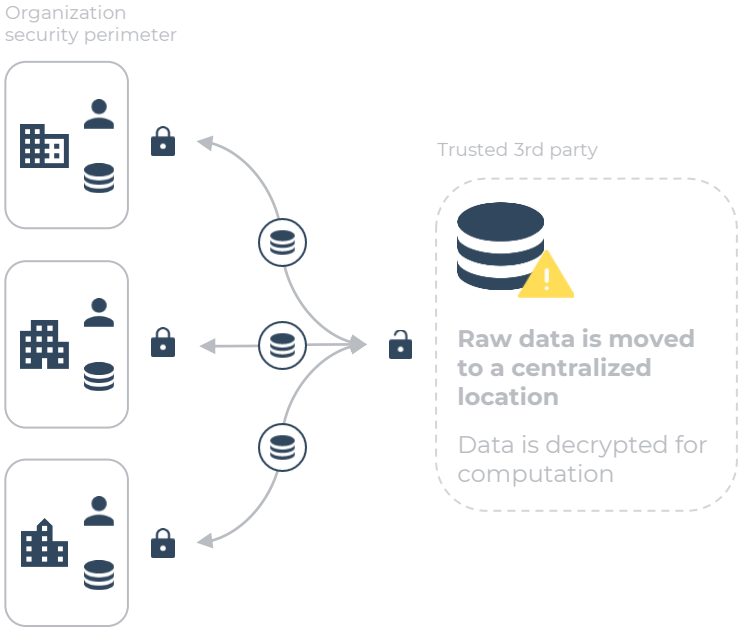
1. Put your gold in a locked box.
2. Keep the key.
3. Let your jeweler work on it through a glove box.
4. Unlock the box when the jeweler is done!

Figure from Prof. Kristin Lauter ("Private AI: Machine Learning on Encrypted Data", 2021)

Homomorphic encryption enables computations directly on encrypted data: "compute on the data without seeing the data"

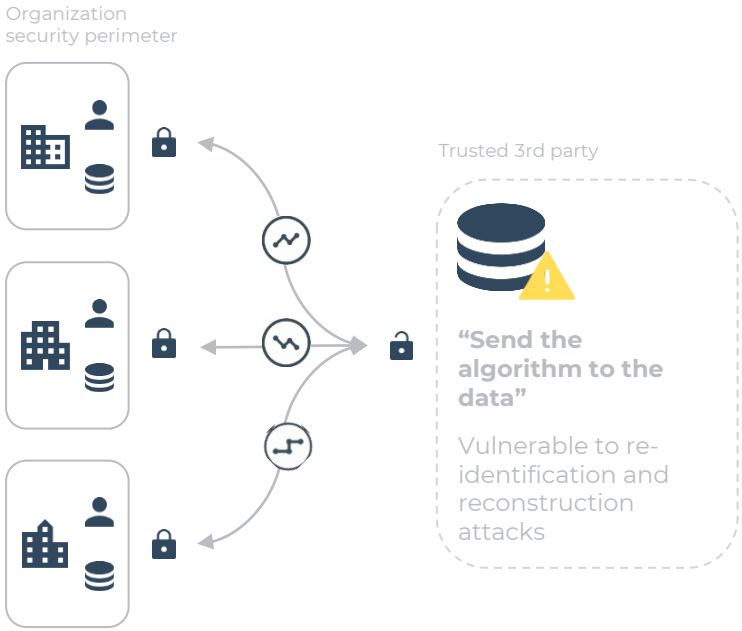
...but what happens if the raw data cannot be moved or centralized?

Data collaborations: Centralized approach



- **Single point of failure at the central database**
- **Individual sites lose control over their data**
- **Not always feasible across jurisdictions**

Data collaborations: Federated Learning



- **Requires trust on the aggregation server**
- **Vulnerable to re-identification and reconstruction attacks**

- B. Hitaj, G. Ateniese, and F. Perez-Cruz. Deep models under the GAN: Information leakage from collaborative deep learning. In ACM CCS, 2017.
- Z. Wang, M. Song, Z. Zhang, Y. Song, Q. Wang, and H. Qi. Beyond inferring class representatives: User-level privacy leakage from federated learning. In IEEE INFOCOM, 2019.
- L. Zhu, Z. Liu, and S. Han. Deep leakage from gradients. In NIPS. 2019.
- L. Melis, C. Song, E. De Cristofaro, and V. Shmatikov. Exploiting unintended feature leakage in collaborative learning. In IEEE S&P, 2019.
- M. Nasr, R. Shokri, and A. Houmansadr. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In IEEE S&P, 2019.

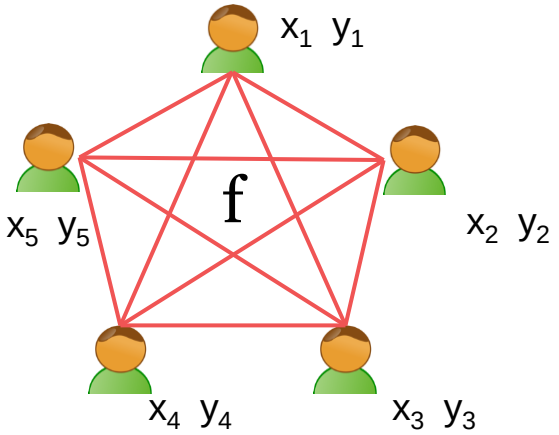
Secure Multiparty Computation (SMC)

Problem statement:

A set of players $\mathcal{P} = \{P_1, P_2, \dots, P_N\}$ would like to compute a function $f(x_1, x_2, \dots, x_N) = (y_1, y_2, \dots, y_N)$ of their joint inputs.

Requirements:

1. *Privacy*
   No party should learn anything more than its prescribed output
2. *Correctness*
   Each party is guaranteed that the output that it receives is correct



Realization:

An (interactive) multiparty cryptographic protocol

MHE (Multiparty Homomorphic Encryption)

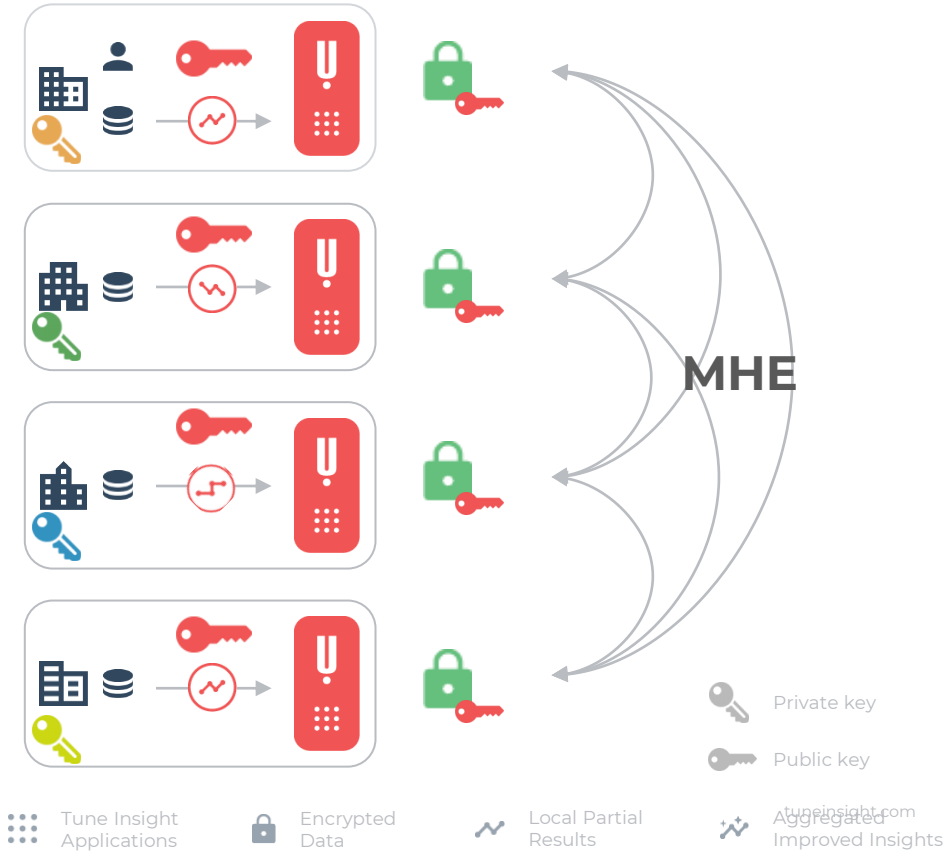
Combination of:

HE encrypted computation → **Security and Privacy**

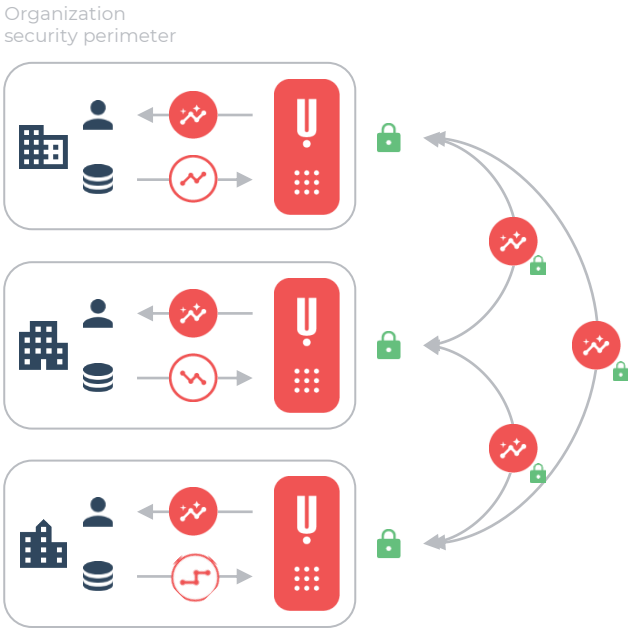
SMC collective key → **Trust Distribution**

FL Iterative local & collective training → **Data Minimization**

- ✓ **Policy enforcement embedded in the protocol**
- ✓ **Raw data does not move**
- ✓ **Computation is encrypted end-to-end**



Data collaborations: Secure and distributed approach



- Minimization of transfers
- Always aggregated & encrypted data
- Computation over encrypted data
- Controlled computation

Tune Insight Applications

Encrypted Data

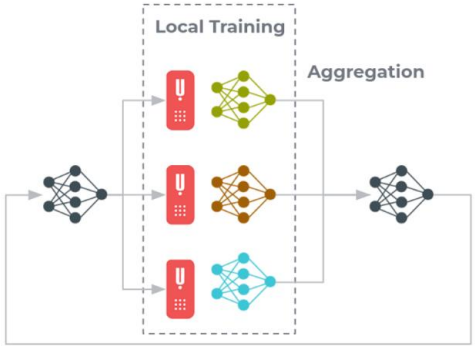
Local Partial Results

Aggregated Collective Insights

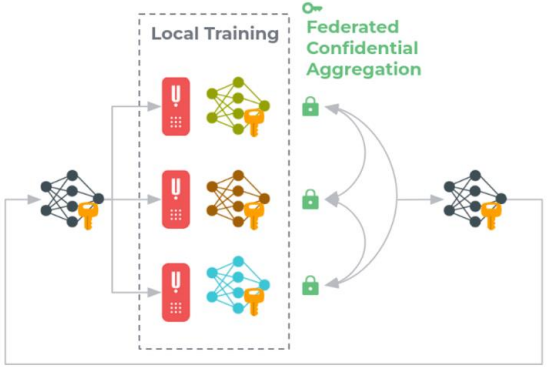
Practical example: training mortality models on D_2 dataset with federated data using Tune Insight's platform and Python SDK

Comparison of three scenarios

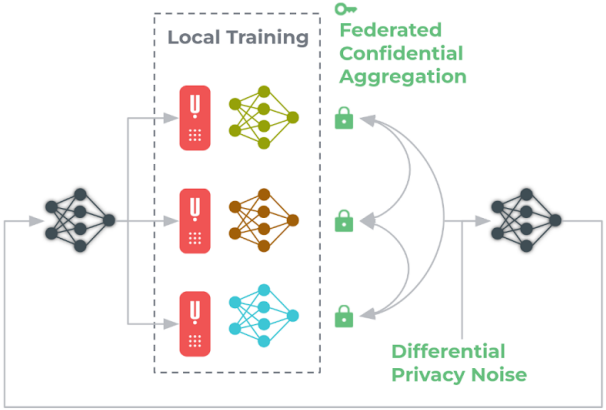
Federated Learning



Encrypted Federated Learning



Hybrid Federated Learning



Practical example: training mortality models on D_2 dataset with federated data using Tune Insight's platform and Python SDK

1. Model parameter definition (Cox and Logistic Regression)

```
task_id_cox = 'mortality_cox'
task_def_cox = {
    "n_inputs": 11,
    "n_classes": 2,
    "model": {
        "type": "cox",
        "pretrained": False,
    },
    "shuffle_data": True,
    "balance_train_classes": True,
    "batch_size": 2048,
    "drop_last_batch": True,
    "loss_criteria": "cross_entropy_loss"
}
```

```
task_id_logreg = 'mortality_logreg'
task_def_logreg = {
    "n_inputs": 11,
    "n_classes": 2,
    "model": {
        "type": "logreg",
        "pretrained": False,
    },
    "shuffle_data": True,
    "balance_train_classes": True,
    "batch_size": 2048,
    "drop_last_batch": True,
    "loss_criteria": "cross_entropy_loss"
}
```

Practical example: training mortality models on D_2 dataset with federated data using Tune Insight's platform and Python SDK

Secure Federated Learning workflow: Training parameters

```
learning_params_federated_secure = models.HybridFLLearningParams(
    fl_rounds = 2,
    local_epochs = 1,
    num_workers = 8,
    batch_size = 2048,
    learning_rate = 0.01,
    momentum = 0.9,
    gradient_clipping = 0.1,
    epsilon = 1,
    delta = 0.001,
)
```

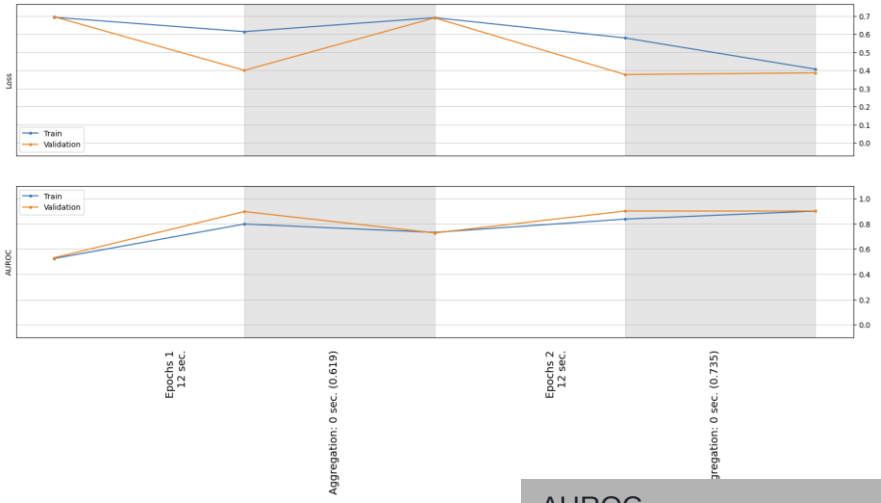
Running the computation

```
hybrid_fl = project.new_hybrid_fl()
hybrid_fl.max_timeout = 300 * 60 * time.second
_ = hybrid_fl.create_from_params(task_id=task_id_cox, learning_params=learning_params_federated_secure, task_def=task_def_cox)
```

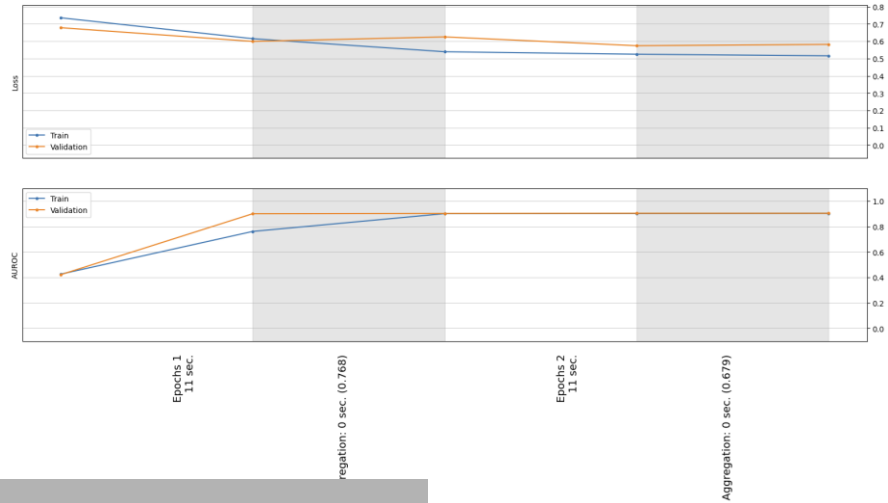
Practical example: training mortality models on D_2 dataset with federated data using Tune Insight's platform and Python SDK

Training performance

Cox Regression



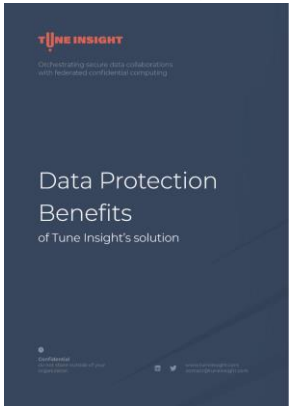
Logistic Regression



AUROC		
	Cox regression	Logistic regression
Local	0.916300	0.905600
Federated	0.916500	0.904500
Secure Federated	0.900600	0.904500

*"**Technical solutions such as multiparty homomorphic encryption (MHE)** that combine these three technical measures while still allowing for the possibility to query and analyse encrypted data without decrypting it have significant potential to **provide effective security measures that facilitate cross-borders transfers of personal data in high-risk settings**."*
Compagnucci et al., Supplementary Measures and Appropriate Safeguards for International Transfers of Personal Data after Schrems II (February 23, 2022). https://ssrn.com/abstract=4042000

Contact us for a full analysis of the platform benefits and risk minimization, addressing the relevant GDPR recitals.



Article 25 Data protection by design and by default	Article 32 Security of processing	Article 33 Breach notification to supervisory authority
Article 34 Breach communication to the data subject	Article 35 Data protection impact assessment	Article 46 Transfers subject to appropriate safeguards

Other applications of secure federated analytics

TUNE INSIGHT

Hospitals & Pharma

Collective survival analysis in oncology

Lab reference data

Train image classifiers in dermatology

Insurance & Re-Insurance

Train collective risk models

Cross-vertical collaboration (Value-Based Healthcare)

groupemutuel

+ others

Cyber Security

Cross-organization alert enrichment

Collective threat intelligence models

Private search of IoCs/alerts

Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

armasuisse
Science and Technology

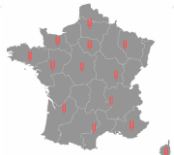
+ others

Financial Services

Collaborative analytics

Sensitive data pooling, AML-CFT

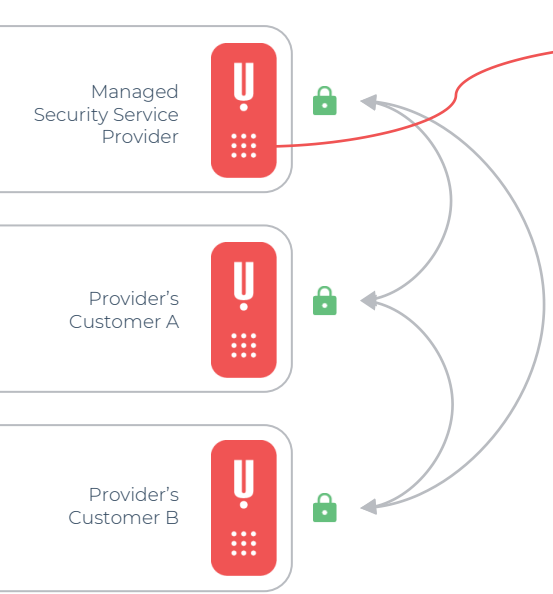
ACPR
BANQUE DE FRANCE



Participated in the Tech Sprint organized by ACPR on Confidential Data Pooling for AML-CFT in 2022

Confidential Collaborative Analytics and Machine Learning

With Tune Insight, organizations can collaborate on their most sensitive cybersecurity data to collectively better defend against cyber attacks

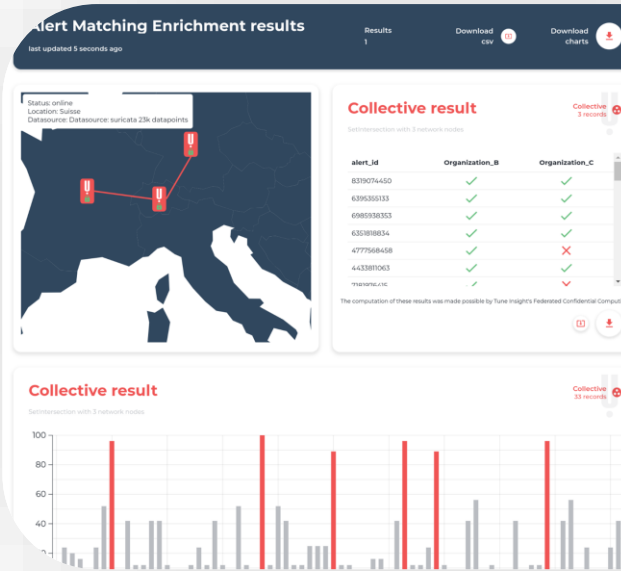


Example Cybersecurity

Connecting the dots between events across customers happens in the analyst's head

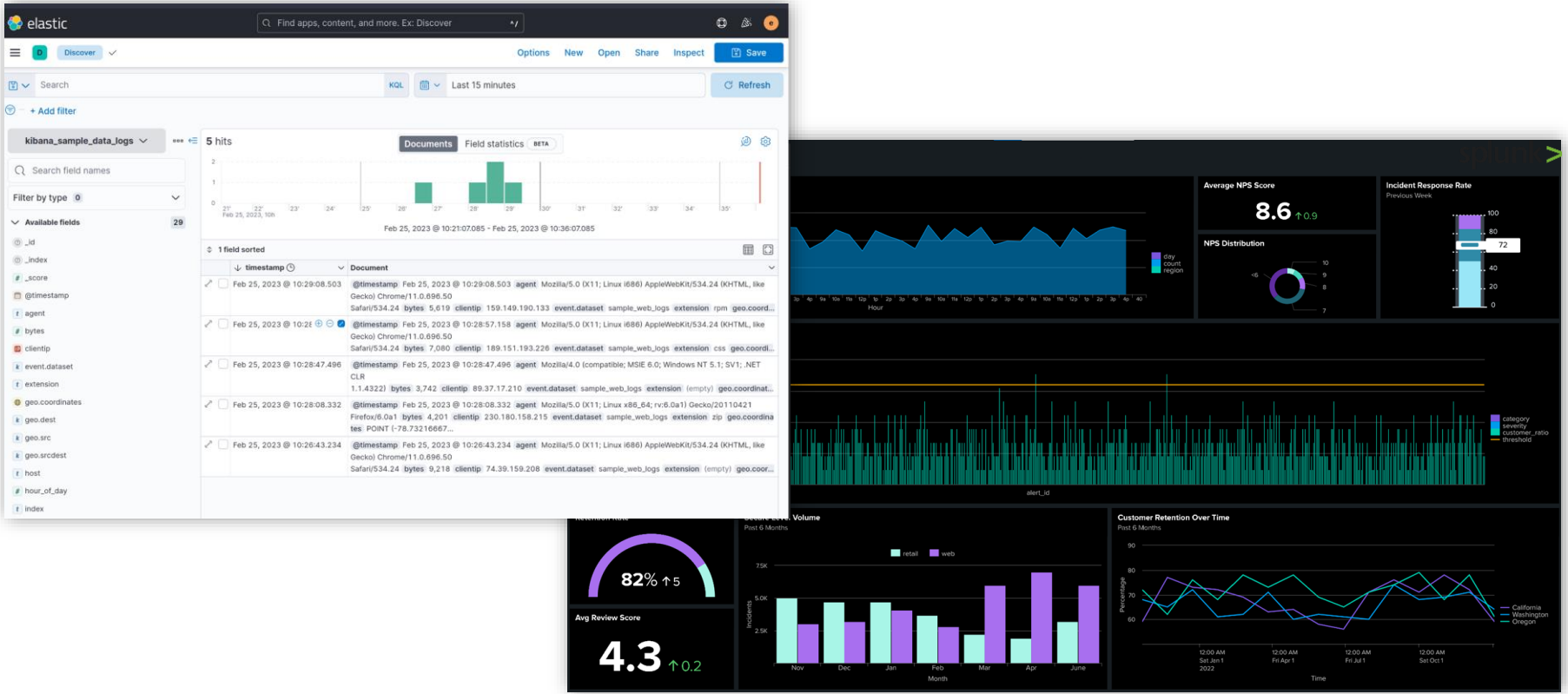
With Tune Insight, MSSPs can automate collective alerts enrichment across customers to reduce false positives and save time

Developed frontend and backend integrations

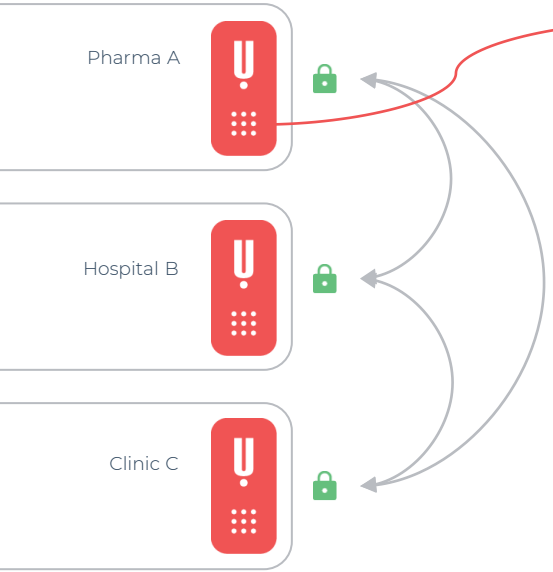


Cyber: Integration with existing platforms and dashboards

Use case: enriching alerts with data from multiple parties, integrated in the organization's existing tools and workflows



Based on the same core technology, we address similar problems in other verticals

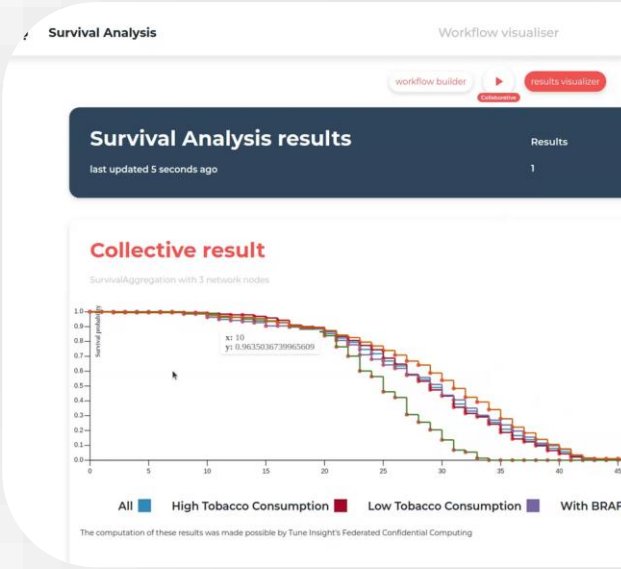


Example Healthcare

Relying only on their own data, hospitals and clinics lack representative datasets to provide personalized care

With Tune Insight, they can collaborate with others to recommend precision treatments without moving or disclosing any raw patient data, and include private players in the collaboration

Developed frontend and backend integrations



Collaborations in Financial Services

Challenges in Fraud Detection and AML[2]

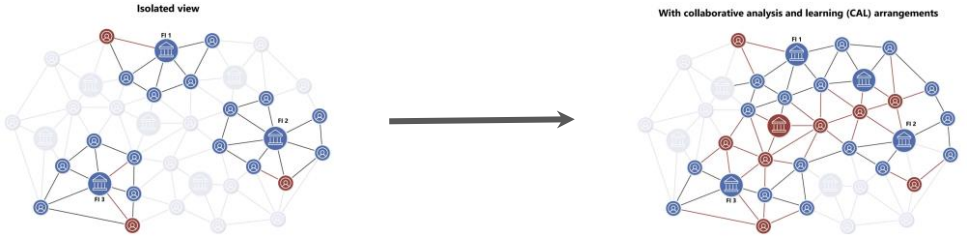
- *Isolated view*
- Data interoperability
- *Data protection and privacy*

WORLD ECONOMIC FORUM

"PETs can fundamentally change the nature of data sharing in financial services, creating new value for customers and addressing institutions' most pressing problems in a way that is acceptable to customers, regulators, and society at large."[1]

WEF. September 2019

1 https://www.weforum.org/whitepapers/the-next-generation-of-data-sharing-in-financial-services-using-privacy-enhancing-techniques-to-unlock-new-value/



Traditional Siloed Rule-Based

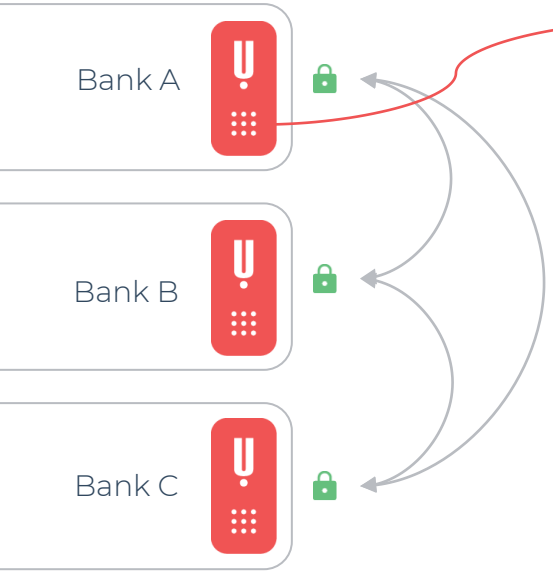
- Traditional rule-based systems can result in high false-positives and false-negatives (90%-95% of the generated alerts are FP)[2]
- ML not fully effective when data from multiple sources is not available (siloed views)

Collaborative Analysis and Learning

- Cross-border ML monitoring can reduce FPs 75% vs rule-based siloed.[2]
- PET-enabled CAL with machine learning-based network analysis appears to reduce the number of FPs by up to 80% compared with the siloed rule-based method.

2 https://www.bis.org/about/bisih/topics/fmis/aurora.htm

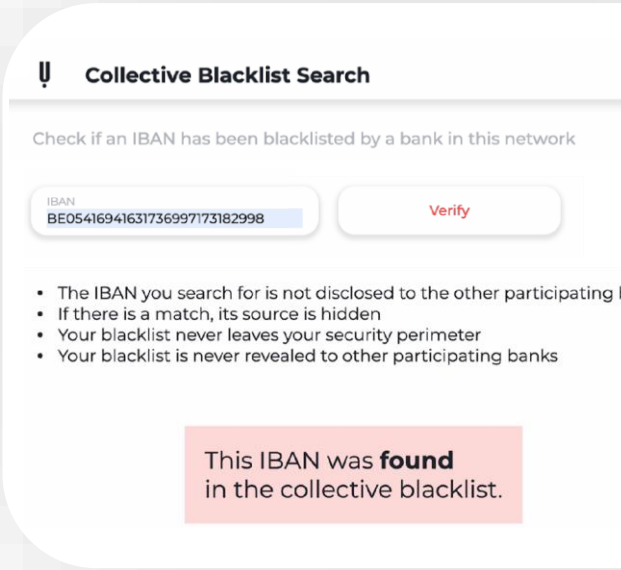
Based on the same core technology, we address similar problems in other verticals



Example Financial Sector

Problem:
Customer data cannot be shared
Effective fraud detection requires collaborations

Solution:
Blacklist matching and training of fraud models without moving or disclosing customer data



Collective IBAN Usage Monitoring using Private Search

In this notebook, we showcase *Tune Insight's Collective Private Search* through a simple use case: A group of three independent financial institutions hold databases of transactions and analysts want to query the usage of specific IBANs over a period of time across all financial institutions.

In practice, responding to such query would require the institutions to share or centralize their transactional data and view the query from the user, affecting confidentiality and privacy across stakeholders.

Using our *Collective Private Search* distributed computation, the analyst's query can effectively be treated while ensuring:

- The transactions databases are not shared across institutions.
- The analyst's query is not visible to any of the institutions.

These guarantees are made possible through the use of a Private Information Retrieval (PIR) protocol, a cryptographic primitive that allows a user to retrieve information from a server without revealing which information they are retrieving. The PIR protocol is implemented securely using homomorphic encryption.

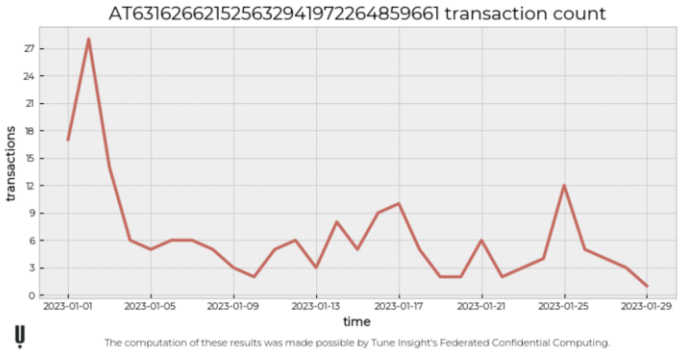
In [12]:
```
# Valid Query
searched_iban = "AT6316266215256329419722648596 61"
result = private_search.query(searched_iban)
display(result)
```

	2023-01-01	2023-01-02	2023-01-03	2023-01-04	2023-01-05	2023-01-06	2023-01-07	2023-01-08	2023-01-09	2023-01-10	...	2023-01-20	2023-01-21	2023-01-22	2023-01-23	2023-01-24	2023-01-25	2023-
0	17	28	14	6	5	6	6	5	3	2	...	2	6	2	3	4	12	

1 rows × 29 columns

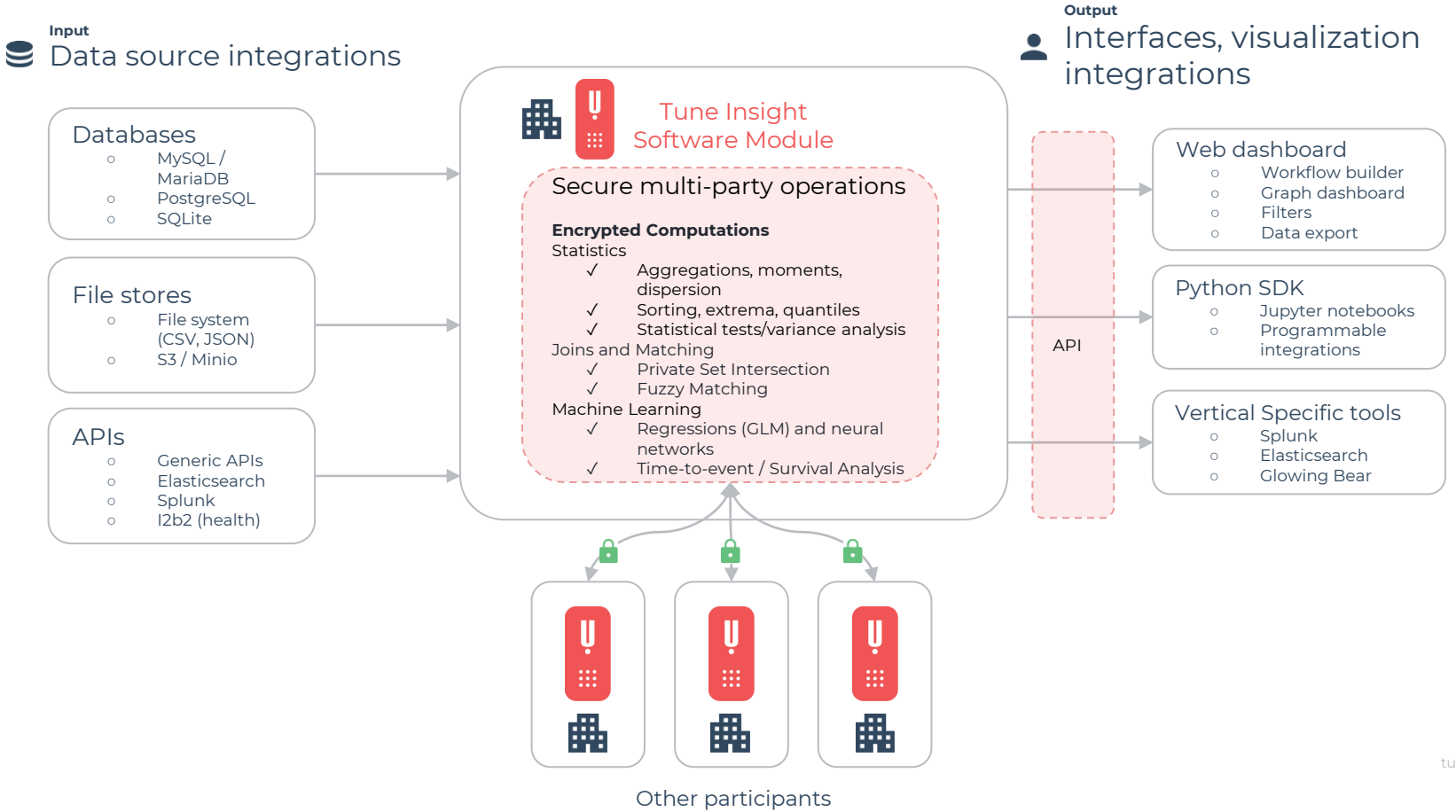
The results can then be plotted using the library or used in further processing.

In [13]:
```
private_search.plot_result(result,f'{searched_iban} transaction count', x_label='time', y_label='transactions',t
```



The computation of these results was made possible by Tune Insight's Federated Confidential Computing.

Product Overview: Tune Insight Software Module



IT Security Assessment

- International security frameworks
  - OWASP Application Security Verification Standard
  - NIST Cybersecurity Framework
- Swiss and domain specific frameworks
  - ICT minimum standard
  - Hospitals and H+ guidelines
- State-of-the-art security technologies
  - OpenID Connect
  - Attribute-based Access Control
  - Key Management Service integration
- Static Application Security Testing
  - Snyk, Trivy, GitHub Dependabot
- Dynamic Application Security Testing
  - Penetration testing by ImmuniWeb

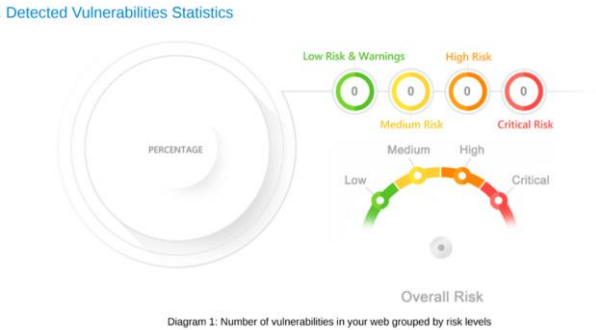


Diagram 1: Number of vulnerabilities in your web grouped by risk levels

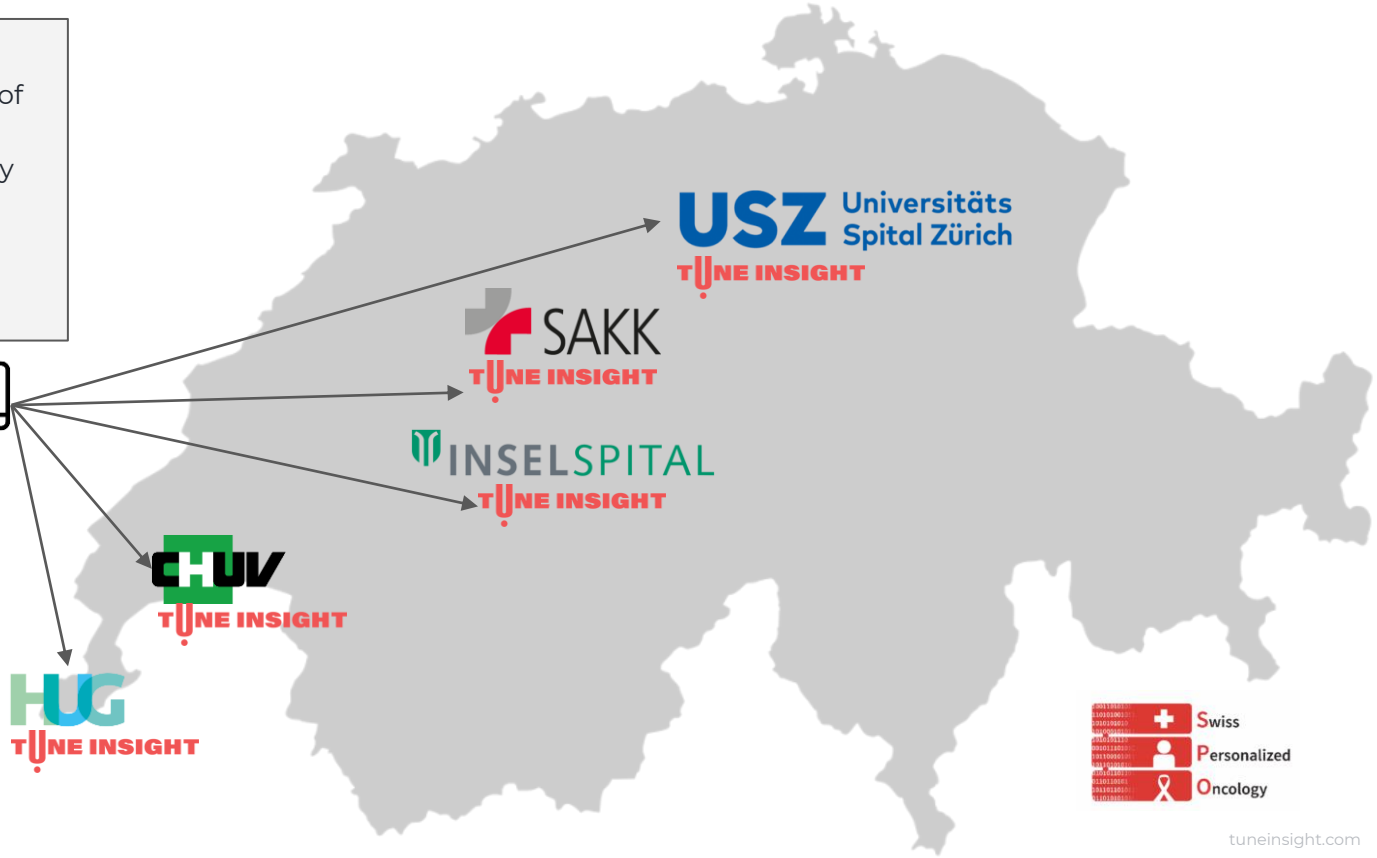
Use case for SPO: Federated analytics platform for research and molecular tumor board

Q1: How many adult cancer patients consenting on reuse of routine data for research with diagnosis of a malignancy on or after 1st January 2015, mutations in BRAF gene and under anti-PD-1 are there?

Explore

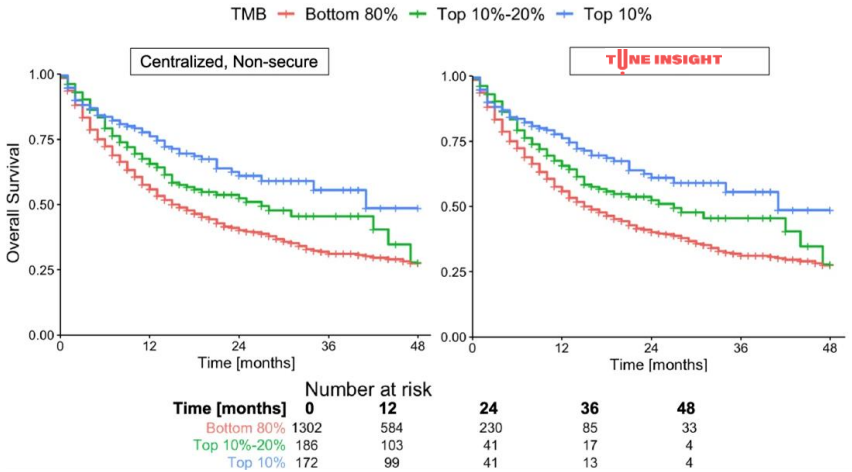
Analysis

Q2: Among these patients, what is the overall survival for patients with and without a mutation on position 600 of the BRAF gene?

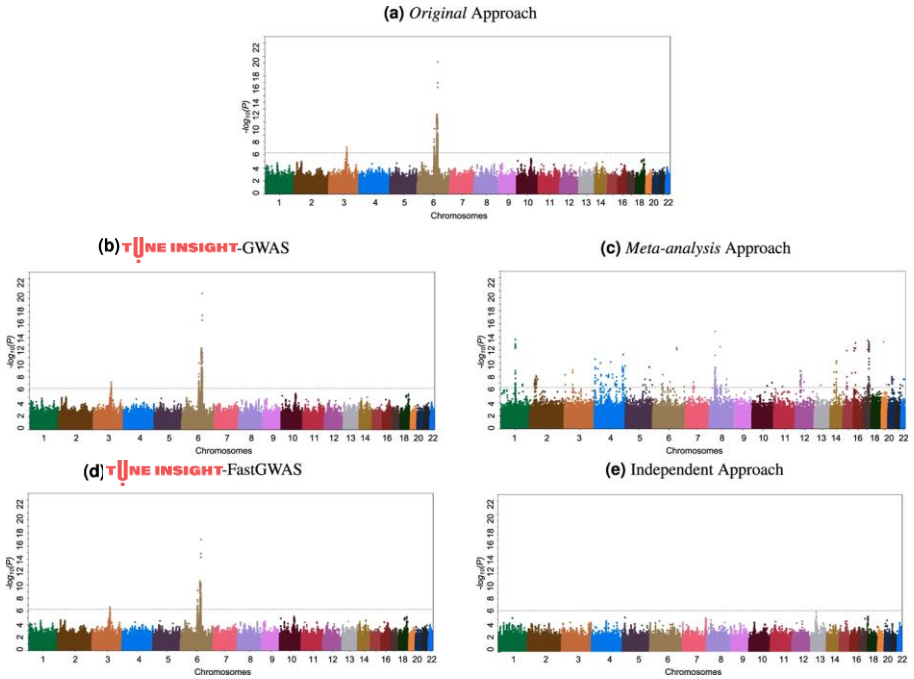


Privacy-Preserving Federated Analytics for Precision Medicine

Survival analysis



GWAS

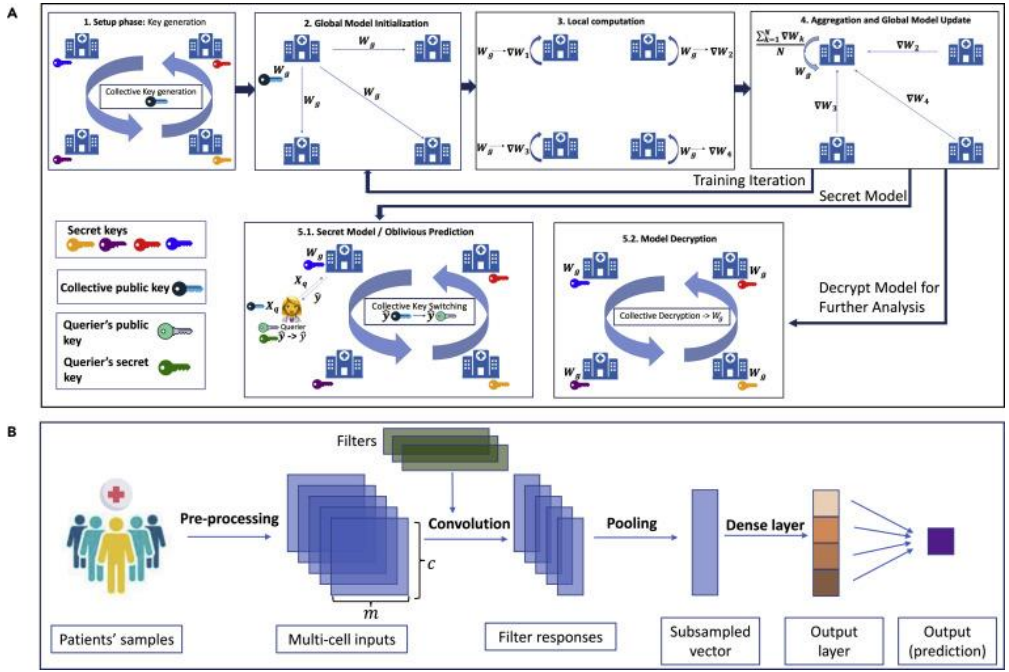


D. Froelicher, J.R. Troncoso-Pastoriza, et al. “Truly privacy-preserving federated analytics for precision medicine with multiparty homomorphic encryption”, Nat Commun 12, 5910 (2021). https://doi.org/10.1038/s41467-021-25972-y

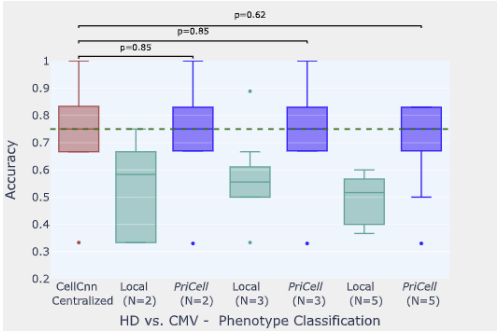
H. Cho, D. Froelicher, J. Chen, M. Edupalli, A. Pyrgelis, J.R. Troncoso-Pastoriza, J.-P. Hubaux, B. Berger. “Secure and Federated Genome-Wide Association Studies for Biobank-Scale Datasets”, bioRxiv, 2022 https://www.biorxiv.org/content/10.1101/2022.11.30.518537v1

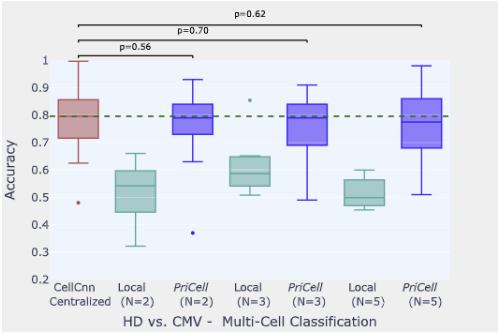
Privacy-Preserving Single-Cell Analysis

System Model



Results

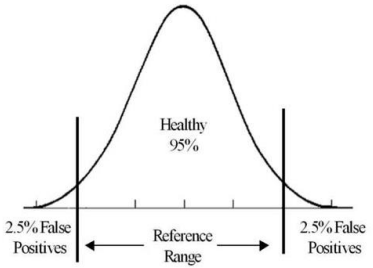




S. Sav, J.P. Bosuat, J.R. Troncoso-Pastoriza, et al. "Privacy-preserving federated neural network learning for disease-associated cell classification." Cell Patterns 2022. https://doi.org/10.1016/j.patter.2022.100487

Use case for Swiss BioRef: real-time personalized lab reference ranges

Q1: What is the reference range for Creatinine (LOINC: 14682-9) for 50y-old male and heart failure?



USZ Universitäts Spital Zürich
TUNE INSIGHT

Schweizer Paraplegiker Zentrum

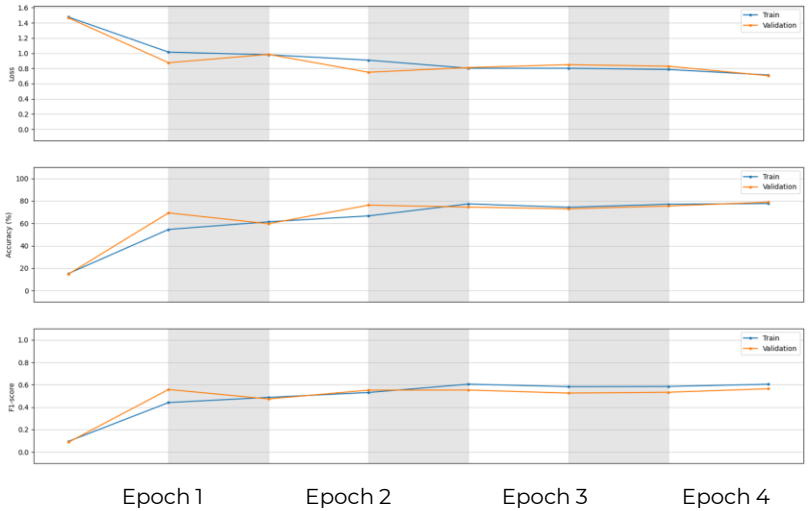
UNIVERSITÄTS-KINDERSPITAL ZÜRICH

INSELSPITAL
TUNE INSIGHT

CHUV
TUNE INSIGHT

HUG
TUNE INSIGHT

Secure Federated Training of Deep Neural Networks on Dermatology Images with combination of HE, MPC, FL, and DP



4 epochs	**Local training baseline**	**Secure federated training**
Nodes	1 node with 10909 samples	3 nodes (~3635 samples each)
Training accuracy	72.16%	77.65%
Training F1-score	0.279431	0.604438
Validation accuracy	72.13%	78.88%
Validation F1-score	0.279364	0.564171
Privacy params	N/A	$\varepsilon = 1.0$, $\delta = 0.0001$
Time overhead	0	~10% (w.r.t. vanilla FL)

Dataset: Fitzpatrick17k, ~30k images (https://github.com/mattgroh/fitzpatrick17k)

Model:

Type: ViT with 4-layers embedding

Size: 5,528,259 parameters, 44.3MB

100 seconds/epoch on a g4dn.2xlarge AWS EC2 instance with a Nvidia T4 GPU (16GB memory)

Selected recent publications from Tune Insight

D. Froelicher et al. “**Scalable and Privacy-Preserving Federated Principal Component Analysis**.” IEEE Symposium on Security and Privacy (2023). https://doi.ieeecomputersociety.org/10.1109/SP46215.2023.00051

H. Cho et al. “**Secure and Federated Genome-Wide Association Studies for Biobank-Scale Datasets**”. Biorxiv, 2022. https://www.biorxiv.org/content/10.1101/2022.11.30.518537v1

S. Sav, J.P. Bosuat, J.R. Troncoso-Pastoriza, et al. “**Privacy-preserving federated neural network learning for disease-associated cell classification**.” Cell Patterns 2022. https://doi.org/10.1016/j.patter.2022.100487

J.P. Bossuat, C. Mouchet, J.R. Troncoso-Pastoriza, J.P. Hubaux. “**Efficient Bootstrapping for Approximate Homomorphic Encryption with Non-Sparse Keys.**” EUROCRYPT 2022

D. Froelicher, J.R. Troncoso-Pastoriza, et al. “**Truly privacy-preserving federated analytics for precision medicine with multiparty homomorphic encryption**.” Nature Communications 12, 5910 (2021). https://doi.org/10.1038/s41467-021-25972-y

J. Scheibner, J.L. Raisaro, J.R. Troncoso-Pastoriza, M. Ienca, J. Fellay, E. Vayena, J.-P. Hubaux. “**Revolutionizing Medical Data Sharing Using Advanced Privacy Enhancing Technologies: Technical, Legal and Ethical Synthesis**.” Journal of Medical Internet Research, 2021. https://pubmed.ncbi.nlm.nih.gov/33629963/

C. Mouchet, J. R. Troncoso-Pastoriza, J.P. Bossuat and J.P. Hubaux. “**Multiparty Homomorphic Encryption from Ring-Learning-with-Errors**.” PETS 2021. IACR ePrint Archive: Report 2020/304, 2020.

D. Froelicher, J. R. Troncoso-Pastoriza, A. Pyrgelis, S. Sav, J. S. Sousa, J.P. Bossuat, and J.P. Hubaux. “**Scalable Privacy-Preserving Distributed Learning**.” PETS 2021. CoRR abs/2005.09532, 2020.

S. Sav, A. Pyrgelis, J.R. Troncoso-Pastoriza, J.-P. Bossuat, J.S. Sousa, J.-P. Hubaux, “**POSEIDON: Privacy-Preserving Federated Neural Network Learning**.” CoRR abs 2009.00349, 2020. NDSS 2021

Lattigo library. **https://github.com/tuneinsight/lattigo**

https://tuneinsight.com

https://linkedin.com/company/tuneinsight

https://twitter.com/tuneinsight

Additional References on Lattice Cryptography

[Ajt96] M. Ajtai. Generating hard instances of lattice problems. Quaderni di Matematica, 13:1-32, 2004. Preliminary version in STOC 1996

[BDOP16] C. Baum, I. Damgård, S. Oechsner and C. Peikert. Efficient Commitments and Zero-Knowledge Protocols from Ring-SIS with Applications to Lattice-based Threshold Cryptosystems. Cryptology ePrint Archive, report 2016/997. 2016

[BGG+14] D. Boneh, C. Gentry, S. Gorbunov, S. Halevi, V. Nikolaenko, G. Segev, V. Vaikuntanathan, D. Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In EUROCRYPT, pp 533-556. 2014

[BGV12] Z. Brakerski, C. Gentry, V. Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. TOCT 6(3):13, 2014. Preliminary version in ITCS 2012

[BLLN13] J. W. Bos, K. Lauter, J. Loftus, M. Naehrig. Improved security for a ring-based fully homomorphic encryption scheme. In IMACC 2013, Oxford, UK, Dec 17-19. LNCS 8308, pp. 45–64. 2013.

Additional References on Lattice Cryptography

[CS15] A. Costache and N.P. Smart. Which Ring Based Somewhat Homomorphic Encryption Scheme is Best? Cryptology ePrint Archive, report 2015/889. 2015

[FV12] J. Fan and F. Vercauteren. Somewhat Practical Fully Homomorphic Encryption. Cryptology ePrint Archive, report 2012/144. 2012

[Gen09] C. Gentry. Fully Homomorphic Encryption using ideal lattices. In STOC, pp. 169-178. 2009

[GGH+13] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, B. Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In FOCS, pp. 40-49, 2013

[HPS96] J. Hoffstein, J. Pipher, J.H. Silverman. NTRU: A Public Key Cryptosystem. NTRU Cryptosystems, Inc. (now Security Innovation)

Additional References on Lattice Cryptography

[LLL82] A.K. Lenstra, H.W. Lenstra, Jr., L. Lovász. Factoring polynomials with rational coefficients. Mathematische Annalen, 261(4):515-534, Dec. 1982

[LNV10] K. Lauter, M. Naehrig, V. Vaikuntanathan. Can homomorphic encryption be practical?. In ACM CCSW '11. ACM, New York, NY, USA, pp. 113-124. 2011

[LP11] R. Lindner and C. Peikert. Better key sizes (and attacks) for LWE-based encryption. In CT-RSA, pp. 319-339. 2011

[LPR10] V. Lyubashevsky, C. Peikert, O. Regev. On Ideal lattices and learning with errors over rings. Journal of the ACM, 60(6):43:1-43:35, Nov 2013. Preliminary version in Eurocrypt 2010

[Pei16] C. Peikert. A Decade of Lattice Cryptography. Cryptology ePrint Archive, report 2015/939. 2015. Last revised 17 Feb 2016

Additional References on Lattice Cryptography

[PTP17] A. Pedrouzo-Ulloa, J. R. Troncoso-Pastoriza, F. Pérez-González. Number Theoretic Transforms for Secure Signal Processing, in IEEE Transactions on Information Forensics and Security, vol. 12, no. 5, pp. 1125-1140, May 2017.

[PR07] C. Peikert and A. Rosen. Lattices that admit logarithmic worst-case to average-case connection factors. In STOC, pp. 478-487. 2007

[RS10] M .Rückert and M. Schneider. Estimating the security of lattice-based cryptosystems. Cryptology ePrint Archive, report 2010/137. 2010

[Sch87] C.-P. Schnorr. A hierarchy of polynomial time lattice basis reduction algorithms. Theor. Comput. Sci., 53:201-224, 1987

[Sho97] P.W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer

[SS13] D. Stehlé and R. Steinfeld. Making NTRUEncrypt and NTRUSign as Secure as Standard Worst-Case Problems over Ideal Lattices. Cryptology ePrint Archive, report 2013/004. 2013