From

# HOMOMORPHIC ENCRYPTION

to Privacy-Preserving Image Classification in the Cloud

Dr. Matthias Minihold • 20.08.2021
Virtual @ QPQ Technology Seminar, Dublin

# Overview

# Impact of Quantum Computing on IT Security—Overview

## Goals of Cryptography within IT Security

- Confidentiality (A speaks in private with B)
- Authenticity (A knows it is B where data originates)
- Integrity (A can verify that the data is unmodified and complete)
- Non-repudiation (B cannot deny sending signed data)

## Effects of Grover's and Shor's quantum algorithms in cryptanalysis

- Symmetric Ciphers (AES, ...): security level halved by Grover's algorithm;
  $\exists c \in \mathbb{R}\ \forall n \in \mathbb{N} : \mathcal{O}\left(c^n\right) \xrightarrow{\text{Grover}} \mathcal{O}\left(c^{\frac{n}{2}}\right) = \mathcal{O}\left(\sqrt{c^n}\right)$,
- Encryption (RSA, ECC) and signatures (RSA, (EC)DSA): broken by Shor's algorithm;
  $\exists c \in \mathbb{R}\ \forall n \in \mathbb{N} : \mathcal{O}\left(c^n\right) \xrightarrow{\text{Shor}} \mathcal{O}\left(n^c\right)$.

Implementation and integration issues lead to delayed migration to post-quantum crypto.

## Computing on Encrypted Data Practice—Law Perspective

### $\approx 50$ Years Data Protection Regulations: Timeline for the EU

1970 *Hessian Data Protection Regulation* privacy law (Hesse),
1986 Overhauled $2^{nd}$ version for public authorities (in Germany),
1995 Adapt & blue-print natural person's EU Data Protection Directive,
2016 Superseded by EU's General Data Protection Regulation (GDPR),
2018 GDPR is enforceable since May 2018 granting basic protection,
2021 Prominent coverage of fines issued due to GDPR all over Europe.

Any *'free'* Cloud-service means *user data* is the *product*.

## Computing on Encrypted Data Theory—Theoretical Perspective

Let $n \in \mathbb{N}$ denote the security parameter. Typically $> 80$ bit post-quantum security level.

### (Public-Key) Encryption Scheme $S$

Given an encryption (resp. decryption) function $\mathsf{Enc}_{\mathsf{pk}} : \mathcal{M} \to \mathcal{C}$ (resp. $\mathsf{Dec}_{\mathsf{sk}} : \mathcal{C} \to \mathcal{M}$) with secret-key–public-key pair $(\mathsf{sk}, \mathsf{pk}) \xleftarrow{\$} \mathsf{Gen}(1^n)$; we call it private-key, if $\mathsf{sk} = \mathsf{pk}$, and require all algorithms to be efficiently computable ($\mathrm{PPT}$).

For all plaintexts $m \in \mathcal{M}$, and all key-pairs $(\mathsf{sk}, \mathsf{pk}) \in \mathcal{K}$ we have

$\Pr[\mathsf{Dec}_{\mathsf{sk}}(\mathsf{Enc}_{\mathsf{pk}}(m)) = m] = 1 - \mathrm{negl}(n)$, holds with overwhelming probability ('w.o.p.').

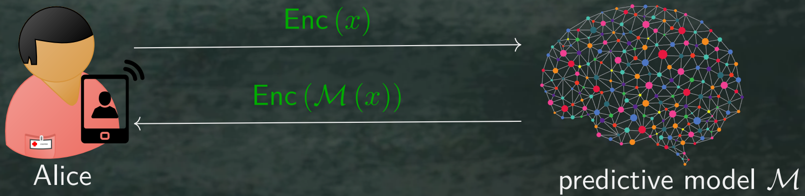
### Evaluating a Function $f$ on Encrypted Data

Let $S = (\mathsf{Gen}(1^n), \mathsf{Enc}(.), \mathsf{Dec}(.))$ be a (public-key) encryption scheme:

$\mathsf{Eval}(f, \mathsf{Enc}_{\mathsf{pk}}(m)) = c \in \mathcal{C}$, such that w.o.p. $\mathsf{Dec}_{\mathsf{sk}}(c) = f(m)$ holds.

# Machine Learning as a Service (MLaaS)

User submits Enc $(x)$ and recovers Enc $(\mathcal{M}(x))$; the encrypted prediction.




- ✓ Privacy — input & output data is encrypted (user has only key)
- ◆ Efficiency — is a central practical issue

**Goal of PhD-Thesis:** FHE–DiNN — fast homomorphic evaluation of neural networks ✓

## Recapitulation: Homomorphisms and Fully Homomorphic Encryption (FHE)

Remarkably, FHE can evaluate any function $f$ on encrypted inputs $c$.

**FHE means "$\forall f: \; f \circ \mathsf{FHE.Enc}_{\mathsf{pk}} \mathrel{\widehat{=}} \mathsf{FHE.Enc}_{\mathsf{pk}} \circ f$"**

Let $(\mathsf{FHE.Gen}, \mathsf{FHE.Enc}, \mathsf{FHE.Dec}, \mathsf{FHE.Eval})$ be an (IND-CPA–secure public-key) encryption scheme with compact ciphertexts $\mathcal{C}$.
If for any computable function $f \in \mathcal{F}$ and all plaintexts $m_1, m_2 \in \mathcal{M}$,

$$(f \circ \mathsf{FHE.Enc}_{\mathsf{pk}})(m_1, m_2) = \overbrace{f([m_1]_{\mathsf{pk}}, [m_2]_{\mathsf{pk}})}^{f(c_1,c_2)=c} \stackrel{!}{=} \overbrace{[f(m_1, m_2)]_{\mathsf{pk}}}^{c' \in \mathcal{C}}$$
$$= (\mathsf{FHE.Enc}_{\mathsf{pk}} \circ f)(m_1, m_2),$$

holds with $f(m_1, m_2) = m_3 \in \mathcal{M} \subseteq \mathcal{C}$, then it is an FHE scheme.

Actually, w.o.p. $\mathsf{FHE.Dec}_{\mathsf{sk}}(c) = \mathsf{FHE.Dec}_{\mathsf{sk}}(c') \in \mathcal{M}$ must match!

## FHE — 'The Holy Grail of Cryptography' [Mic10]

### $\approx 40$ Years of FHE: Timeline

1978 Adleman, Dertouzos, and Rivest mention private homomorphisms

2009 Gentry's *theoretical breakthrough* construction: $1^{\text{st}}$ generation

2012 Brakerski, Gentry, and Vaikuntanathan (BGV)'s *simpler* $2^{\text{nd}}$ gen.

2013 Gentry, Sahai, and Waters (GSW)'s *efficient*: $3^{\text{rd}}$ generation

2016 Chillotti, Gama, Georgieva, and Izabachène (CGGI)'s *efficient implementation*: TFHE

2021 FHE schemes' & applications' *practical breakthrough*?

# Definitions: From LWE to TLWE and TGSW

## LWE assumption (over the Torus)

Given a secret $\mathbf{s} \xleftarrow{\$} \{0,1\}^n$, it is hard to distinguish between $(\mathbf{a}, b)$, where $\mathbf{a} \xleftarrow{\$} \mathbb{T}^n$ and $b = \langle \mathbf{s}, \mathbf{a} \rangle + e \in \mathbb{T}$, with $e \leftarrow \chi$, and $(\mathbf{u}, v) \xleftarrow{\$} \mathbb{T}^{n+1}$.

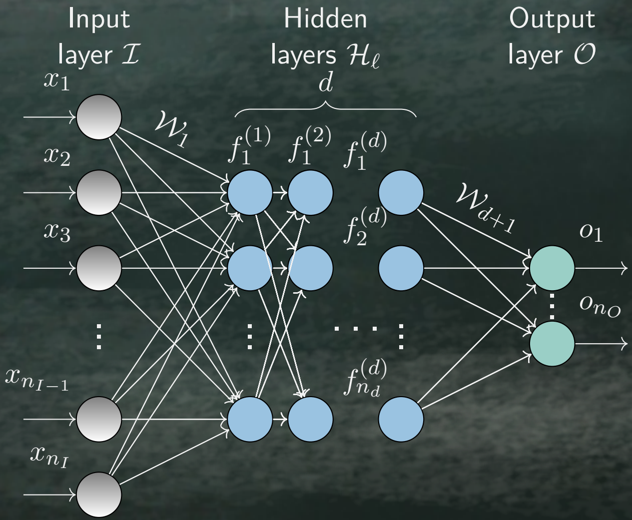
To define polynomial and matrix generalizations, we set:

- $\mathbb{B} := \{-1, 1\}$, $\mathbb{B}[X]/(X^N + 1)$, polynomials of deg $< N = 1024$,
- $\mathbb{T} := \mathbb{R}/\mathbb{Z}$, with torus-polynomials $\mathbb{T}_N[X] := \mathbb{T}[X]/(X^N + 1)$,
- $\mathbb{T}_N[X]^k := \mathbb{T}[X]^k/(X^N + 1)$, tuples of torus-polynomials, $k \geq 1$.

## TLWE/TGSW Sample

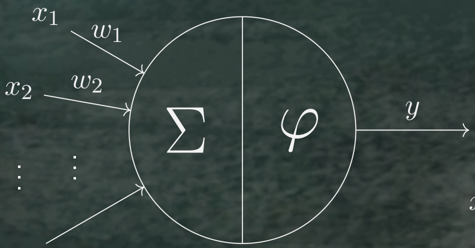
Let $\mathbf{s} \xleftarrow{\$} \mathbb{B}[X]^k/(X^N + 1)$, a vector of $k \geq 1$ polynomials, and message $m \in \mathbb{T}_N[X]^k$. $(\mathbf{a}, b) \in \mathbb{T}_N[X]^{k+1}$ is a TLWE Sample, if $a \xleftarrow{\$} \mathbb{T}_N[X]^k$, $b = \mathbf{a} \cdot \mathbf{s} + m + e$, with Gaussian-noise $e \leftarrow \chi_\alpha, \alpha > 0$ at $\mathbf{a} \cdot \mathbf{s} + m$. A TGSW Sample is a list of $\ell \geq 1$ TLWE Samples or a $(k + 1 \times \ell)$-matrix.

# Deep Feed-Forward Neural Network with $n_{\mathcal{I}} : n_1 : \cdots : n_d : n_{\mathcal{O}}$–topology



## Close-up on Neuron

Computation for every neuron:




$x_i, w_i, y \in \mathbb{R}$

$$y = \varphi\left(\sum_i w_i\, x_i\right),$$

where $\varphi$ is an *activation function*.

## FHE-friendly Discretized Neural Networks

**Goal:** *FHE-friendly* model of neural network: $x_i, w_i, y \in \mathbb{Z}$.

### Definition (DiNN)

A neural network whose layers have inputs in $\{-I, \ldots, I\} \subseteq \mathbb{Z}$, weights in $\{-W, \ldots, W\} \subseteq \mathbb{Z}$, for $I, W, O \in \mathbb{N}$, and each neuron's activation function maps the weighted sum to integer values in $\{-O, \ldots, O\} \subseteq \mathbb{Z}$.

1. Not restrictive as it seems as, e.g., binarized NNs perform well;
2. trade-off between size and performance;
3. conversion is straight-forward.

### Main impediment: non-linear functions

Applying the non-linear activation function after linear layer.

# Main Idea: Activation While Bootstrapping FHE

Combine necessary refreshing with desirable activation function:

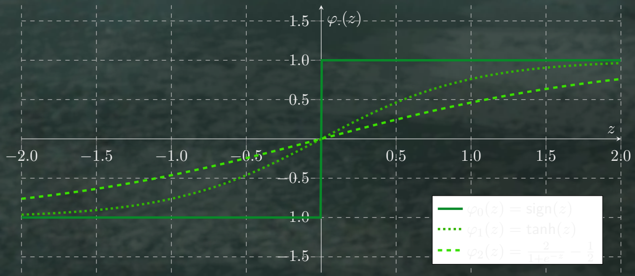
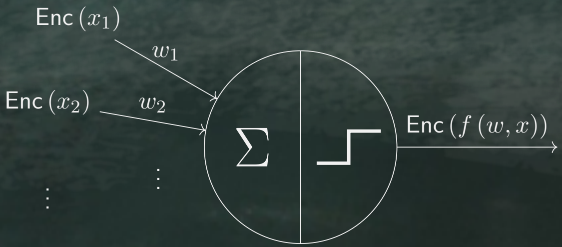



Figure: Several neural network activation functions and our choice $\varphi_0$.

$$\mathsf{Enc}\,(z) \to \mathsf{Enc}\,(f\,(z)) \to \ldots$$

## Close-up on a single neuron: two steps




Each neuron computes Enc $(f(w, x))$, *e.g.* Enc (sign $(\langle w, x \rangle)$):

1. Compute inner product $\sum_i w_i \text{Enc}(x_i)$ (linearly homomorphic)
2. Bootstrap encryption of activated result (fully homomorphic)

## Torus Fully Homomorphic Encryption (TFHE)

We use Torus Fully Homomorphic Encryption framework on $\mathbb{T} := \mathbb{R}/\mathbb{Z}$.

### Security Assumption underlying TFHE and FHE–DiNN

Hardness of Learning with Errors (LWE) on $\mathbb{T}$:

$$(\mathbf{a},\ \langle \mathbf{s}, \mathbf{a} \rangle + e \mod 1) \overset{c}{\approx} (\mathbf{a},\ \mathbf{u}) \in \mathbb{T}^{n+1},$$

where $e \leftarrow \chi_\alpha$, $\mathbf{s} \leftarrow_\$ \mathbb{B}^n$, $\mathbf{a}, \mathbf{u} \leftarrow_\$ \mathbb{T}^n$ with error parameter $\alpha$.

We also use other torus-based schemes allowing performance increase:

- TLWE (for encrypting polynomials $\mathbb{T}[X]$)
- TGSW ('matrix TLWE'; roughly equivalent to GSW construction)

## Novel TFHE-Adaptations for Fast DiNN Inference

1. Combining implementations of `Bootstrapping` and `Activation`
2. Reducing bandwidth usage by `Packing` ciphertexts
3. Moving boostrapping operation order, i.e., when to do a `Keyswitch`
4. `Reparametrizing` message space between neural network layers
5. Optimizing alternative implementation of `BlindRotate`

**Goal** `Packing`: encrypt polynomial $\mathbb{T}[X]$ instead of $\mathbb{T}$ scalars: $x(X) = \sum_i x_i X^i \in \mathbb{T}[X]$ a ciphertext.

**Idea** Redefine and pack (clear) weights in hidden layers: $w(X) := \sum_i w_i X^{-i}$.

**Effect** Constant term of $x(X) \cdot w(X) \in \mathbb{T}[X]$ is $\sum_i w_i x_i \in \mathbb{T}$.

# Novel TFHE-Adaptations for Fast DiNN Inference

1. Combining implementations of `Bootstrapping` and `Activation`
2. Reducing bandwidth usage by `Packing` ciphertexts
3. Moving boostrapping operation order, i.e., when to do a `Keyswitch`
4. `Reparametrizing` message space between neural network layers
5. Optimizing alternative implementation of `BlindRotate`

**Goal** Reduce LWE dimension, ensuring security level, to optimize memory, efficiency, bootstrapping–key's size, final noise, and the number of expensive external products.

**Idea** Bootstrap = SampleExtract ∘ BlindRotate ∘ KeySwitch

**Effect** Less noise; size $n < N$ is used only for bootstrapping

## Novel TFHE-Adaptations for Fast DiNN Inference

1. Combining implementations of `Bootstrapping` and `Activation`
2. Reducing bandwidth usage by `Packing` ciphertexts
3. Moving boostrapping operation order, i.e., when to do a `Keyswitch`
4. `Reparametrizing` message space between neural network layers
5. Optimizing alternative implementation of `BlindRotate`

**Goal** Dynamically change the message space to reduce errors.

**Idea** For $I_\ell$, an upper bound on the sum in layer $\ell + 1$, define:

$$\texttt{testvector}(X) = t(X) := \frac{1}{2I_\ell + 1} \sum_{i=0}^{N-1} X^i.$$

**Effect** Less slices, hence less inaccurate decisions when rounding.

# Novel TFHE-Adaptations for Fast DiNN Inference

1. Combining implementations of `Bootstrapping` and `Activation`
2. Reducing bandwidth usage by `Packing` ciphertexts
3. Moving boostrapping operation order, i.e., when to do a `Keyswitch`
4. `Reparametrizing` message space between neural network layers
5. Optimizing alternative implementation of `BlindRotate`

We unfold the loop for computing $X^{\langle \mathbf{s},\mathbf{a} \rangle}$ in `BlindRotate`.

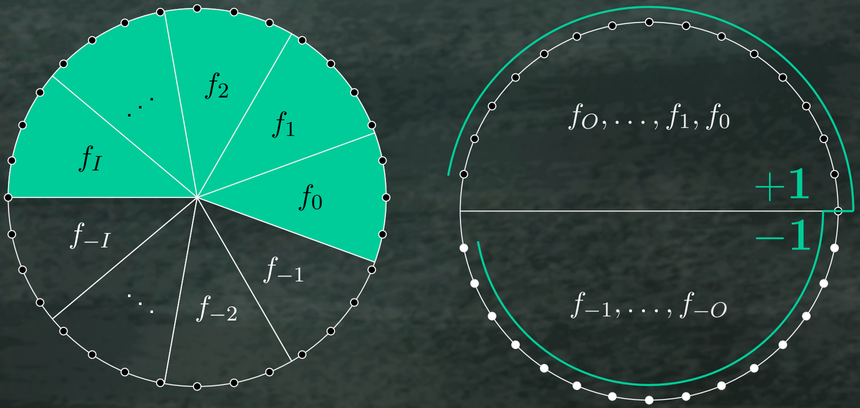
**Goal** Trade-off off-line pre-processing for on-line speed.

**Idea** Windowed processing & using algebraic keys-relations.

**Effect** Larger bootstrapping key traded for faster execution.

# Extending the TFHE Framework for Fast Bootstrapping

...with anti-periodic $f : \mathbb{W}_I \to \mathbb{W}_O$, mapping input slots to outputs:



# Moving the bootstrapping operation order

## Bootstrap

Bootstrapping-to-sign comprises 3 algorithms, given bk, ksk, $t(X)$, and an $N$-dim. LWE sample $\mathbf{c} = (\mathbf{a}, b) = \text{LWE}_{\mathbf{s},\alpha}(m)$ of message $m$ under key $\mathbf{s}$:

- **BlindRotate:** $(\text{TGSW})^n \times (n - \text{LWE}) \times$ TLWE $\rightarrow$ TLWE
  Rotates the *wheel*, i.e. computes $X^{b - \langle \mathbf{s}, \mathbf{a} \rangle} \cdot t(X)$.
- **SampleExtract:** TLWE $\rightarrow N$-LWE
  Extracts $N$-LWE sample $\mu_0$ of message $\mu \in \mathbb{T}_N[X]$.
- **KeySwitch:** $(n - \text{LWE})^n \times N$-LWE $\rightarrow n$-LWE
  Returns a $n$-LWE sample under $\mathbf{s}'$ of $b - \langle \mathbf{s}, \mathbf{a} \rangle$.

Reversing the two LWE schemes of sizes $n < N$ improves run-time.

## Fast Fourier Transform (FFT)

Think of $\mathbf{x} = \mathsf{Enc}_{\mathsf{pk}}(\mathbf{p}) \in \mathbb{T}$ as an TLWE encrypted pixel (or a whole picture packed into one input ciphertext $\mathbf{x} = \mathsf{Enc}_{\mathsf{pk}}\left(\sum_i p_i X^i\right) \in \mathbb{T}[X]$), and $\mathbf{w}$ as public (or company) known weights per neuron.
We pre-compute the Fourier transform $\widehat{\mathbf{w}} = \mathcal{F}_{2N}(\mathbf{w})$ of $\mathbf{w}$ off-line.

### Convolution and Efficient (FFT) Multiplication

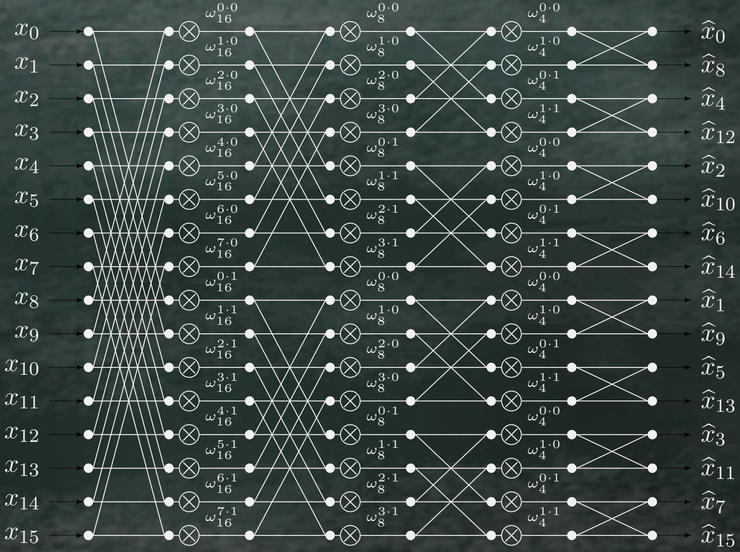
Let $N, I \in \mathbb{N}$ be powers of 2, for instance $N = 1024, I = 2^{32}$. The input polynomial $\mathbf{x} \in \mathbb{T}_N[X]$ and the weights are embedded in the first components of vectors as $w_j \in \mathbb{Z}, x_j \in \mathbb{W}_I \subseteq \mathbb{T}, 0 \leq j < N$, then using the fast Fourier transform allows efficient computation of the multisum:

$$(\mathcal{F}_N(\mathbf{x}))_m = (\mathcal{F}_{\frac{N}{2}}((\mathbf{x}_{2j})_{0\leq j<\frac{N}{2}}))_{\frac{m}{2}} + \omega_{\frac{N}{2}}^{m} \cdot (\mathcal{F}_{\frac{N}{2}}((\mathbf{x}_{2j+1})_{0\leq j<\frac{N}{2}}))_{\frac{m}{2}+1},$$

$$\mathcal{F}_N(\mathbf{x} * \mathbf{w}) = \mathcal{F}_N(\mathbf{x}) \cdot \mathcal{F}_N(\mathbf{w}) \in \mathbb{C},$$

$$(\mathbf{x} * \mathbf{w}) \equiv \mathcal{F}_N^{-1}(\mathcal{F}_N(\mathbf{x}) \cdot \mathcal{F}_N(\mathbf{w})) \mod 1.$$

# Speeding-up the Processing: FFT Data-Flow $\widehat{\mathbf{x}} = \mathbb{F}_{2N}(\mathbf{x})$




FFT's divide-and-conquer strategy for power-of-2 lengths; $2N = 16$.

# Digit Recognition & Classification in the Cloud

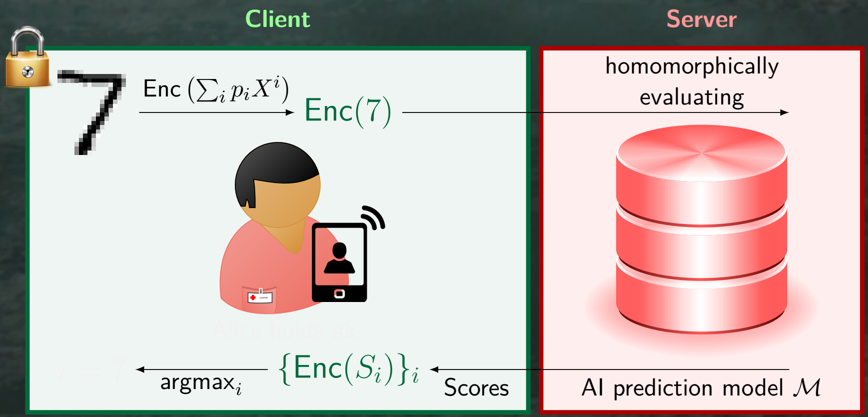
We showcase a solution to the problem of *digit recognition*.

# Digit Recognition & Classification in the Cloud

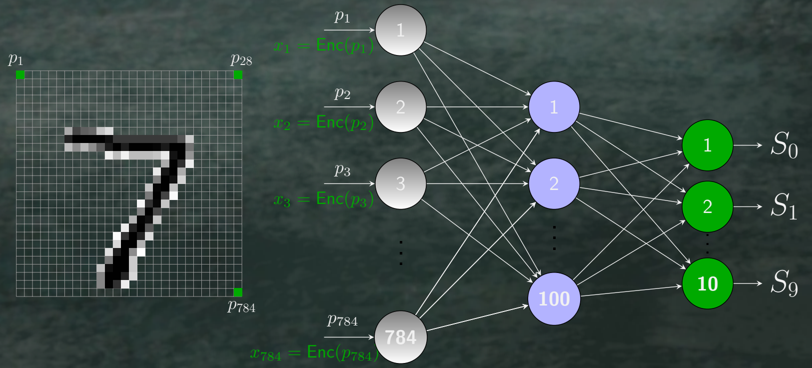
We showcase a solution to the problem of **blind** *digit recognition*.

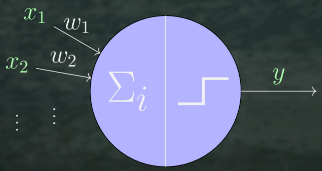

Dataset: MNIST (60 000 images in training set + 10 000 in test set).

# FHE–DiNN: Overview [BMMP18]



# FHE–DiNN: Input Image and 784:100:10–Neural Network



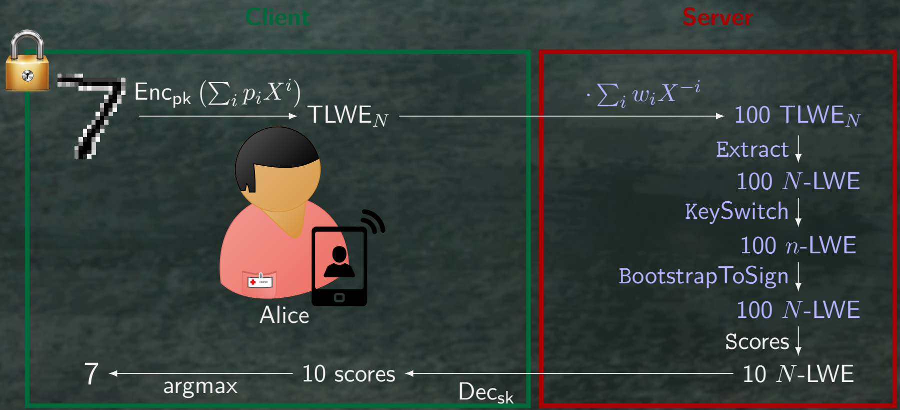




Hidden Neuron (zoomed)

$y = \varphi\left(\Sigma_i\, w_i\, x_i\right)$, $y \in [-O, \dots, O]$

$x_i \in [-I, \dots, I]$, and $w_i \in [-W, \dots W]$.

# FHE–DiNN: Algorithmic Overview [BMMP18]



## FHE–DiNN: Evaluation Formula of our 784:100:10–network

We assume a neural network trained on $D_{\text{train}} = \{(\mathbf{x}^{(i)}, L^{(i)})_i\}$.

$\mathcal{M}_{\text{FHE–DiNN}}$ models a weighted recomposition of a TLWE encryption $\mathbf{c}_0$;

$$\begin{cases} \mathbb{T}_N[X]^k \longrightarrow (\mathbb{T}_N[X]^k)^{10} \\ \mathbf{c}_0 \mapsto \vec{\mathbf{c}}_2 = \sum_{\ell_2=1}^{100} \left( \underbrace{\varphi_1 \left( \sum_{\ell_1=1}^{784} (\mathbf{c}_0)_{\ell_1} \cdot (\widehat{\mathbf{w}_{0\to 1}})_{\ell_1} \right)}_{\vec{\mathbf{c}}_1} \right)_{\ell_2} \cdot \left(\widehat{\mathbf{W}_{1\to 2}}\right)_{\ell_2}. \end{cases}$$

The homomorphic evaluation yields 10 samples $\overrightarrow{\mathbf{c}_{\mathcal{O}}}$ as output, encrypting the perceptrons' predicted label likelihoods of an encrypted input digit $\mathbf{c}_{\mathcal{I}}$.

Label $L = \text{argmax}_i \left(\text{Dec}_{\text{sk}}\left(\overrightarrow{\mathbf{c}_{\mathcal{O}}}\right)\right)_i$ is how the model sees the input's depicted digit: $L = \mathcal{M}_{\text{FHE–DiNN}}\left(\mathbf{c}_{\mathcal{I}}\right)$, with $\text{Dec}_{\text{sk}}\left(\mathbf{c}_{\mathcal{I}}\right) \approx \mathbf{x}^{(\mathcal{I})} \in (D_{\text{train}})_{\mathbf{x}}$.

### Main Result of the PhD-Thesis—Scalability

The analysis shows how to bootstrap the most expensive layer, then repeat for arbitrary many hidden neurons arranged in various layers.

# FHE–DiNN Experiments: Practical Performance Neural Networks

Performance metrics on (clear) inputs $x$:

| | Original NN | DiNN + hard_sigmoid | DiNN + sign |
|---|---|---|---|
| **FHE–DiNN** 30 | 94.76% | 93.76% (-1 %) | 93.55% (-1.21%) |
| **FHE–DiNN** 100 | 96.75% | 96.62% (-0.13%) | 96.43% (-0.32%) |

Performance metrics on (encrypted) inputs $\mathsf{Enc}_{\mathsf{pk}}(x)$:

| | Acc. | Disagreements | Total wrong BS | when dis. | Time |
|---|---|---|---|---|---|
| **30** | 93.71% | 273 (105–121) | 3 383/300 000 | 196/273 | 0.515 s |
| **100** | 96.26% | 127 (61–44) | 9 088/1 000 000 | 105/127 | 1.679 s |
| **30 w** | 93.46% | 270 (119–110) | 2 912/300 000 | 164/270 | 0.491 s |
| **100 w** | 96.35 % | 150 (66–58) | 7 452/1 000 000 | 99/150 | 1.640 s |

window size w = 2

# Performance Comparison with Microsoft Cryptonets [DGBL+16]

| | Overall Network | | per Image | | | |
|---|---|---|---|---|---|---|
| | $n_{\mathcal{H}}$ | Accuracy | Eval [s] | $\lvert c \rvert$ [B] | Enc [s] | Dec [s] |
| Cryptonets | 945 | 98.95 % | 570 | 586 M | 122 | 5 |
| *Cryptonets*$^{\star}$ | 945 | 98.95 % | 0.07 | 73.3 k | 0.015 | 0.0006 |
| **FHE–DiNN**30 | 30 | 93.71 % | 0.49 | ≈ 8.2 k | 0.000168 | 0.0000106 |
| **FHE–DiNN**100 | 100 | 96.35 % | 1.64 | ≈ 8.2 k | 0.000168 | 0.0000106 |

Cryptonets$^{\star}$ is amortized per image (accumulating 8192 inferences)

**Experimental Results**

Timing/Image on Intel Core i7-4720HQ CPU @ 2.60GHz: 1.64 [sec].

# Reference

**Practical homomorphic encryption and cryptanalysis.**
Matthias Minihold. PhD Thesis. Bochum, 2019.
https://hss-opus.ub.ruhr-uni-bochum.de/opus4/files/6510/diss.pdf

# Questions?