MACHINE LEARNING MEETS CRYPTOLOGY

SETTING THE SCENE

- Assume we are some members of a **hospital**
- We want to **predict** the **probability** our patient develops **type II diabetes** from measurements such as their **age, gender, blood sugar level**
- How can we do this?
- We can train **a machine learning model** on pre-existing data

SETTING THE SCENE

Age	Gender	Blood (sugar) level	Has type 2 diabetes?
…	…	…	…
…	…	…	…
…	…	…	…

ALGORITHM

.
.
.

DESIRED PATIENT'S **AGE, GENDER, BLOOD SUGAR LEVEL**

TRAINED MACHINE LEARNING **MODEL**

PREDICTION OF HAVING TYPE 2 DIABIETES

SETTING THE SCENE

- However, we may not have the **time** to train the model nor the **expertise** (e.g., some algorithms may be better suited than others)
- **Solution?**
- Ask a **third party** to provide an already trained machine learning model fit for our desires

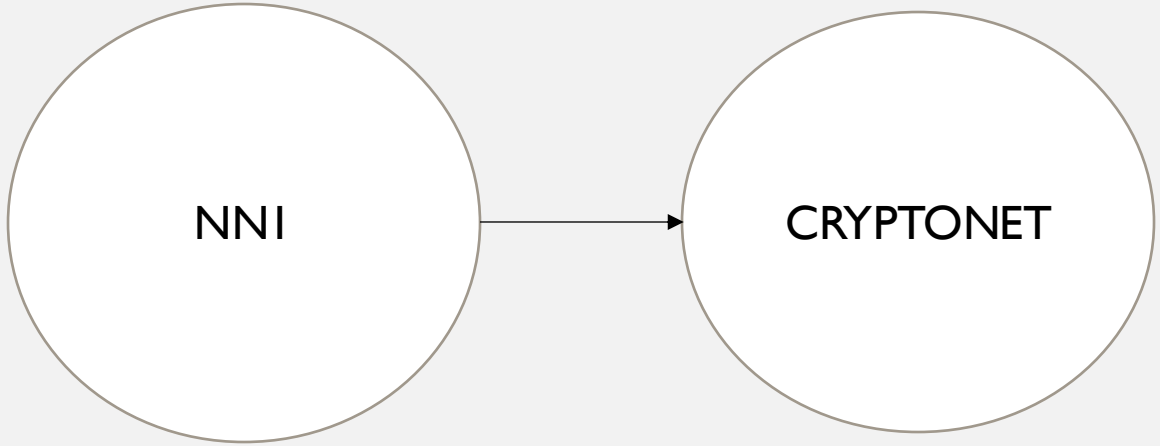
SETTING THE SCENE

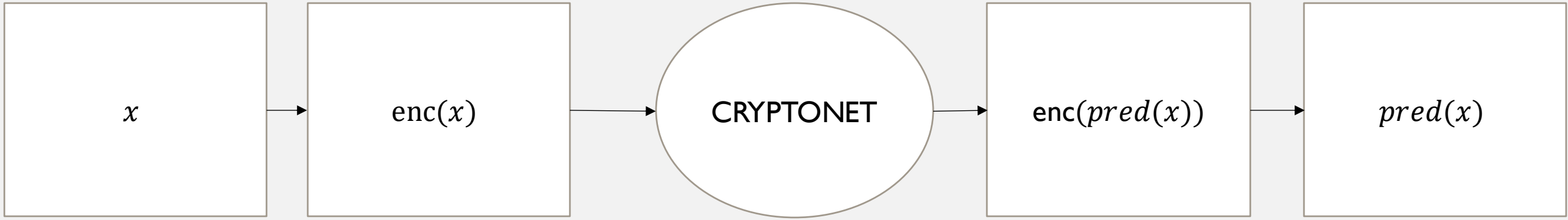
- Is **it safe/ethical/legal** to give **sensitive patient information** to a **third-party?**
- **Solution: CryptoNets**
- Theory of slides related to CryptoNets is from (Gilad-Bachrach, 2016)

CRYPTONETS BRIEF CONCEPT

- A **CryptoNet** is a **neural network** that can be applied to **encrypted data**
- Now onwards we assume the following: we have a trained neural network which we refer to as **NN1** (e.g., which can take **as inputs age, gender,…** and returns as an **output probability** of developing **type II diabetes**)
- We also assume this neural network is hosted by a **third-party**
- A CryptoNet gives us the ability to do as follows

CRYPTONETS BRIEF CONCEPT





CRYPTONETS – STRUCTURE OF NNI

There is always an **input** and an **output** layer

There can be $n \geq 0$ **hidden layers**

Each **node** is a **function** of the nodes with an **edge** into this node from the **previous layer** (excluding the input nodes)

For example, **common functions** used in neural networks are:

Rectified linear value: $x_3 = \max(0, x_1)$

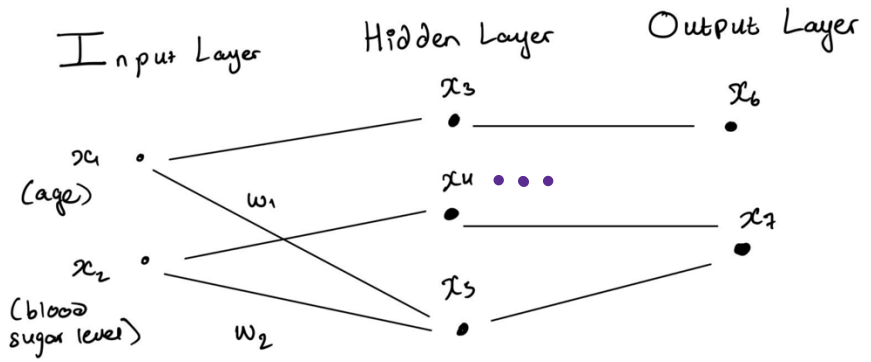
Sigmoid: $x_6 = \frac{1}{1+e^{-x_3}}$

Convolution: $x_5 = w_1 x_1 + w_2 x_2$

Mean Pooling: $x_7 = \frac{x_1+x_2}{2}$

Max Pooling: $x_7 = \max(x_1, x_2)$

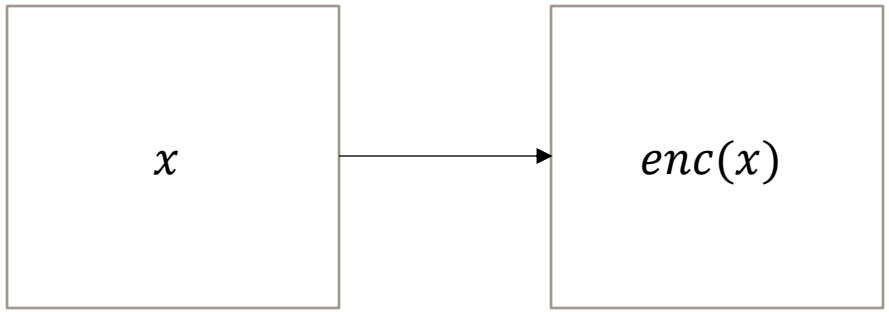
NNI



CRYPTONETS – ENCRYPTION SCHEME

$enc(x)$ is usually a levelled homomorphic encryption scheme

- Levelled Homomorphic Encryption
- Homomorphic Encryption, Fully Homomorphic Encryption
- Rings and Groups



GROUPS AND RINGS

- A **group** is a **set** G with an **operation** $*$

$(\mathbb{Z}, +)$ is a group

$(\mathbb{Z}, *)$ is not a group

Group $\forall\, a, b, c \in G$	
Closure	$a * b \in G$
Commutative	$a * b = b * a$
Associative	$(a * b) * c = a * (b * c)$
Identity	$\forall a \in G\ \exists\, e \in G\ s.t. a * e = e * a = a$
Inverse	$\forall a \in G\ \exists\, b \in G\ s.t. a * b = b * a = \mathrm{e}$

GROUPS AND RINGS

- A **ring** a **set** R with **operation** $(+,*)$
- A **ring** is an **abelian** group under $+$
- Under $*$ **closure** and **associativity** hold (commutativity can hold but not necessary)
- Distributivity: $a * (b + c) = a * b + a * c$

$(\mathbb{Z}, +,*)$ is a ring

Set of odd integers is not a ring

Group $\forall a, b, c \in G$	
Closure	$a * b \in G$
Commutative	$a * b = b * a$
Associative	$(a * b) * c = a * (b * c)$
Identity	$\forall a \in G \; \exists \, e \in G \; s.t. \, a * e = e * a = a$
Inverse	$\forall a \in G \; \exists \, b \in G \; s.t. \, a * b = b * a$

HOMOMORPHIC ENCRYPTION, FULLY HOMOMORPHIC ENCRYPTION

- Let $\mathbf{enc}: E \to D$ denote the encryption function
- Let $m_1, m_2 \in E$

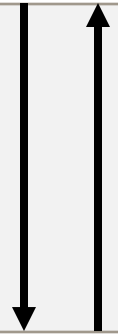
Homomorphic	Fully Homomorphic
Group homomorphism i.e., $(E,*_E), (D,*_D)$ are **groups**	Ring homomorphism i.e., $(E, +_E,*_E), (D, +_D,*_D)$ are **rings**
$enc(m_1 *_E m_2) = enc(m_1) *_D enc(m_2)$	$enc(m_1 +_E m_2) = enc(m_1) +_D enc(m_2)$, $enc(m_1 *_E m_2) = enc(m_1) *_D enc(m_2)$

HOMOMORPHIC ENCRYPTION, FULLY HOMOMORPHIC ENCRYPYION

$m_1, m_2 \in E$

$m_1 + m_2 \in E$

$enc(m_1 + m_2) \in D$



$enc(m_1) \in D$

$enc(m_2) \in D$

$enc(m_1) + enc(m_2) \in D$

LEVELLED HOMOMORPHIC ENCRYPTION

- Let $\mathbf{enc}: \boldsymbol{E} \rightarrow \boldsymbol{D}$ denote the encryption function
- Let $\boldsymbol{m_1, m_2} \in \boldsymbol{E}$

Fully Homomorphic
Ring homomorphism i.e., $(\boldsymbol{E}, +_{\boldsymbol{E}}, *_{\boldsymbol{E}}), (\boldsymbol{D}, +_{\boldsymbol{D}}, *_{\boldsymbol{D}})$ are **rings**
$enc(m_1 +_E m_2) = enc(m_1) +_D enc(m_2),$ $enc(m_1 *_E m_2) = enc(m_1) *_D enc(m_2)$

Levelled Homomorphic
Ring homomorphism i.e., $(\boldsymbol{E}, +_{\boldsymbol{E}}, *_{\boldsymbol{E}}), (\boldsymbol{D}, +_{\boldsymbol{D}}, *_{\boldsymbol{D}})$ are **rings**
As long as $\mathrm{enc}(m_1), \mathrm{enc}(m_2)$ encrypt with small enough errors $enc(m_1 +_E m_2) = enc(m_1) +_D enc(m_2),$ $enc(m_1 *_E m_2) = enc(m_1) *_D enc(m_2)$

Know in advance complexity of arithmetic circuit applied to the data
Efficiency's sake

AN EXAMPLE – SETUP AND KEY GENERATION

Give as an example the encryption scheme used in [1]

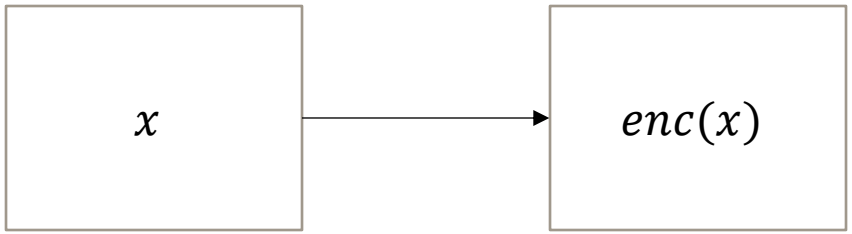
For example: $n = 3, t = 6$

$x^6 + 10x = 1 + 4x$

NOTE: not every element in R_q^n is invertible

So, steps 1, 2 and 3 are iterated until corresponding f has an inverse

$n = 3\,, t = 6$
$x^6 + 10x = 1 + 4x$
Do example on board

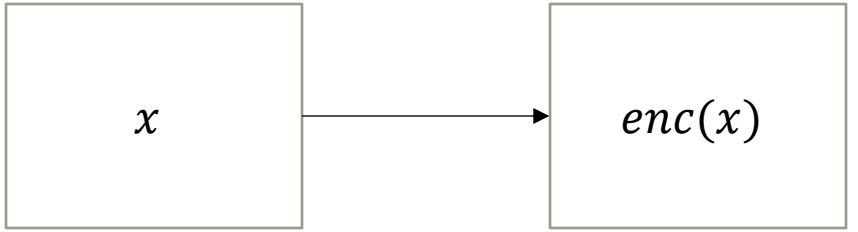


$enc: R_t^n \rightarrow R_q^n$
where $R_t^n = \mathbb{Z}_t[x]/(x^n + 1)$

KEY GENERATION:
1. $f', g \in R_q^n$
2. PRIVATE KEY: $f = f't + 1$
3. PUBLIC KEY: $h = tgf^{-1}$

AN EXAMPLE – ENCRYPTION SCHEME

$e, s \in R_q^n$ are **noise** with **small absolute coefficients**



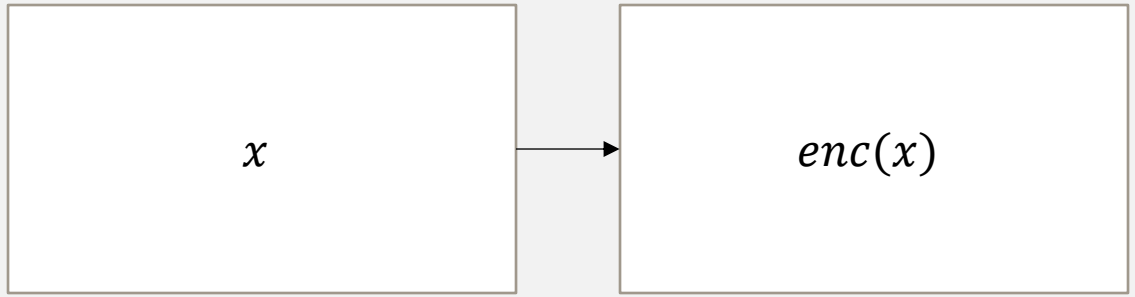
For $m \in R_t^n$

$$\varphi(m) = c = \left(\left\lfloor \frac{q}{t} \right\rfloor m + e + hs\right) \bmod q$$

$$m = \left\lceil \frac{t}{q} fc \right\rfloor \bmod t$$

AN EXAMPLE – ENCRYPTION SCHEME

- Last box **decrypts** to $m_1 m_2$ under the **secret key** f^2
- Can **modify** the result so instead **decrypts** to the **secret key** f by a process called **relinearisation**

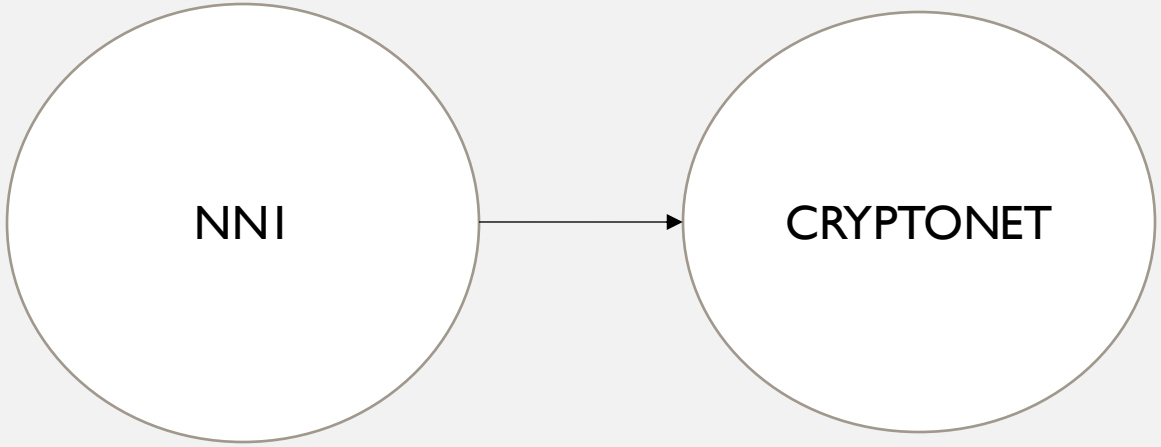


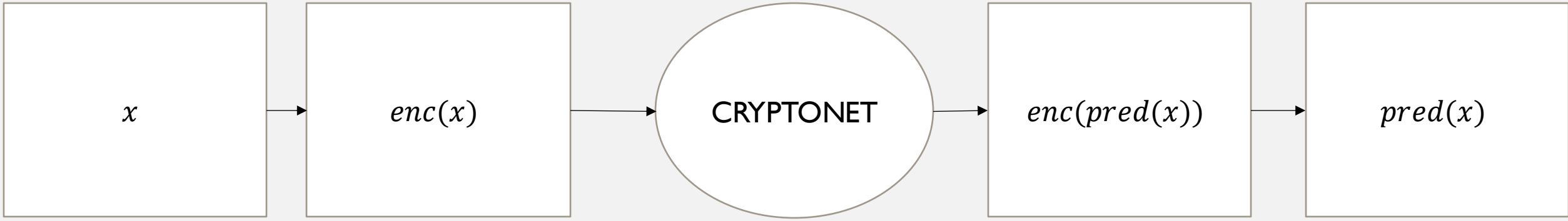
Let $m_1, m_2 \in R_t^n = \mathbb{Z}_t[x]/(x^n + 1)$
Let $c_1 = enc(m_1), c_2 = enc(m_2)$

$$c_1 + c_2 = \left\lfloor \frac{q}{t} \right\rfloor (m_1 + m_2) + (e_1 + e_2) + h(s_1 + s_2)$$
$$= enc(m_1 + m_2)$$

$$\frac{t}{q} c_1 c_2 = \left\lfloor \frac{q}{t} \right\rfloor m_1 m_2 + e' + h^2 s_1 s_2$$

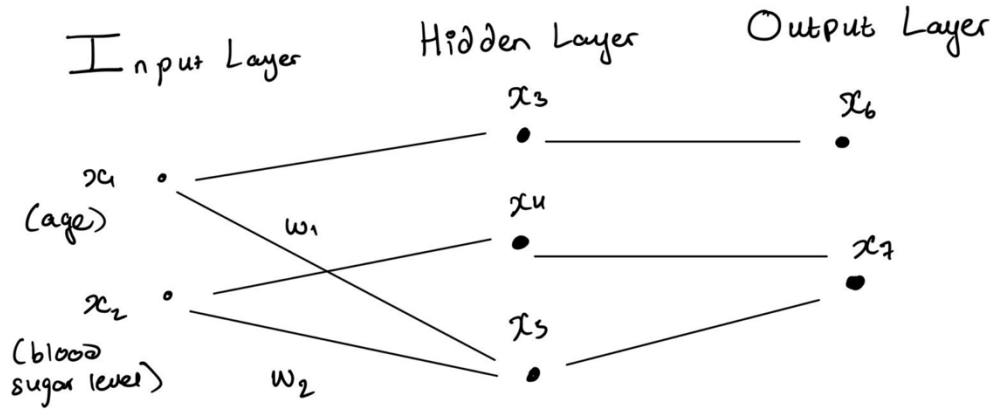
CRYPTONETS





ENCODINGS / DECODINGS

- **NN1** does computations on $\mathbb{R}$, taking as input $\mathbb{R}$
- The **encryption scheme** mentioned takes in as **input** $R_t^n = \mathbb{Z}_t[x]/(x^n + 1)$
- Hence need a mapping from one to the other
- Encoding: $\mathbf{E}$: $\mathbb{R} \rightarrow R_t^n$
- Decoding: D: $R_t^n \rightarrow \mathbb{R}$



EXAMPLES

For $a \in \mathbb{R}$, $E(a) = \lfloor a \rceil \ mod\ t$ (i.e., the constant polynomial)

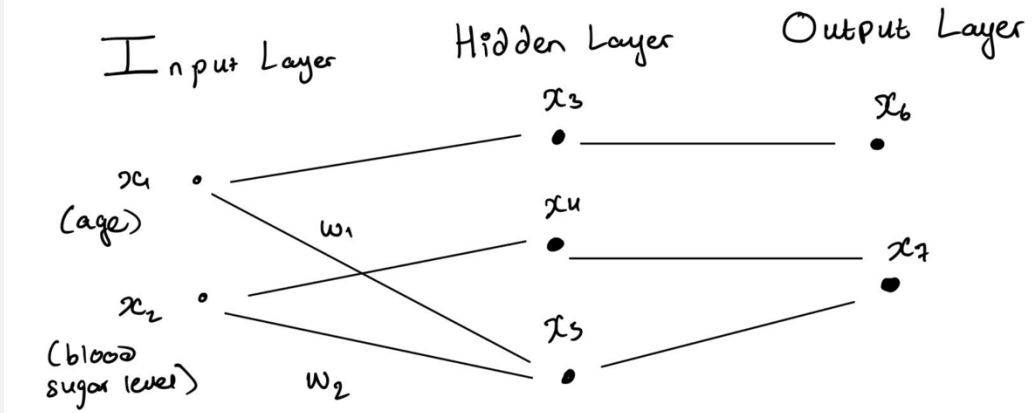
For $a \in \mathbb{R}$ → Fixed precision → Convert to binary → Convert to polynomial binomial expansion → Evaluate at 2 to get back a

Choose t such that $x^n + 1 = \prod(x - \alpha_i)$ → $R_t^n \approx \mathbb{Z}^{\times n}$

Can take n integer values → Pack into a single polynomial → Encrypt this polynomial → Unpack output to give n values

ENCODINGS / DECODINGS

- NN1 does computations on $\mathbb{R}$
- CryptoNets do computations in R_q^n
- Could convert w_1, w_2 to elements in R_t^n using previous examples
- Easier way exists - plain operations



CRYPTONETS SUMMARY

$x_1, x_2, x_3 \in \mathbb{R}$

$E(x_1), E(x_2), E(x_3) \in R_t^n = \mathbb{Z}_t[x]/(x^n + 1)$

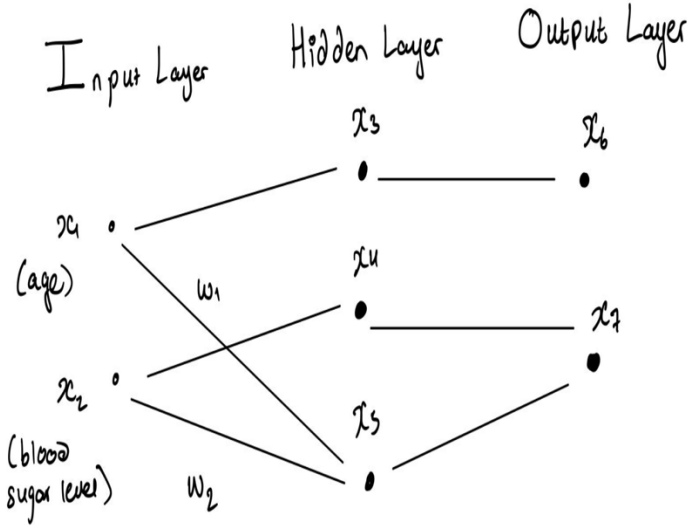
$enc\big(E(x_1)\big), enc\big(E(x_2)\big), enc\big(E(x_3)\big) \in R_q^n$

COMPUTATIONS

$enc\big(pred(x)\big) \in R_q^n$

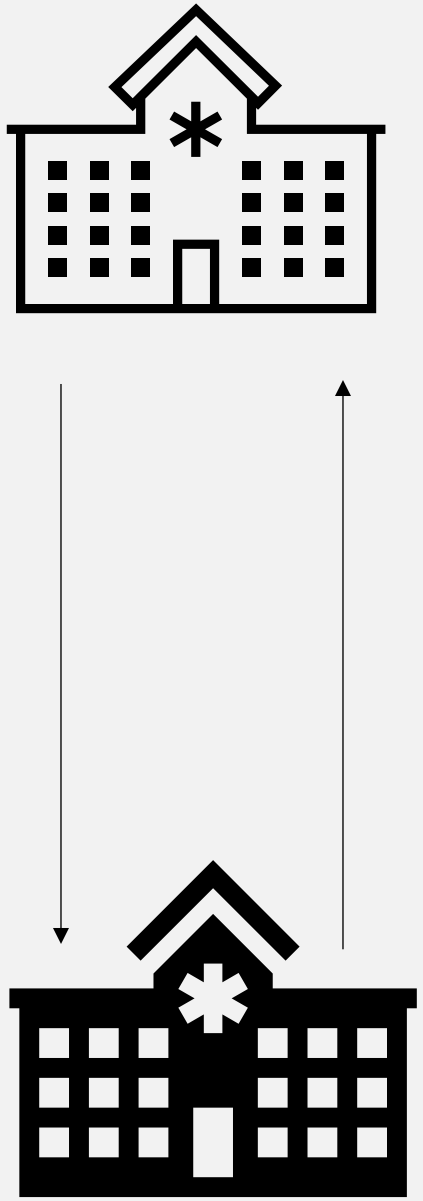
$pred(x) \in R_t^n$

$D(pred(x)) \in \mathbb{R}$



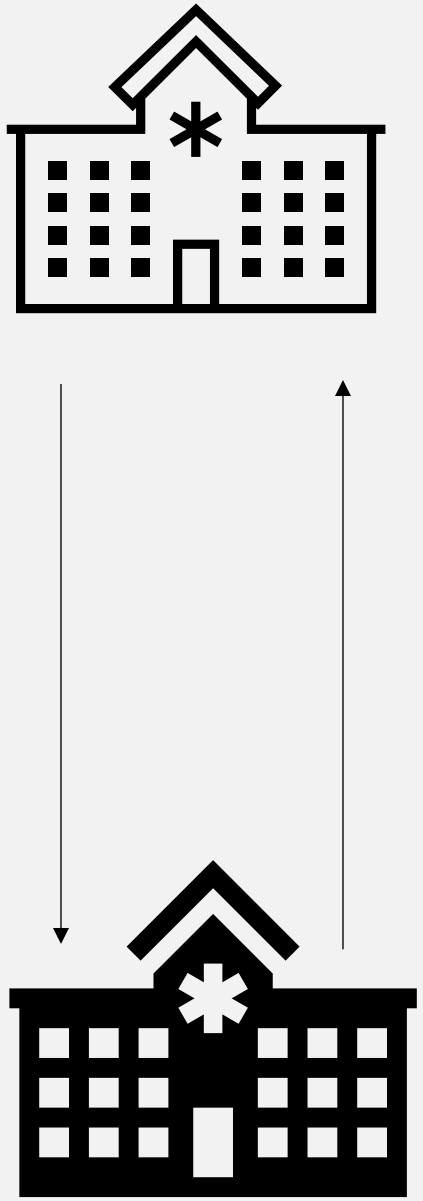
SETTING THE SCENE II

- **Two hospitals** want to determine which one of them is **'better'** than the other (e.g., from a combination of mortality rates, patient satisfaction and more)
- But they do not want to reveal these details to one another
- They both only want to obtain the information of which hospital is better
- Solution?



SETTING THE SCENE II

- Secure Multi-Party Computation **(SMPC)**
- Can **execute a protocol** by sending bits of information back and forth
- The only **new information** then obtained is which **hospital** is **better**
- In general: some function that needs computing on private inputs and SMPC allows information to be exchanged to compute just the output of the function



SECURE MULTI-PARTY COMPUTATION - GENERALISED

- Theory of this slide and following slides inferred from (Zhou, 2024)
- **Parties $P_1, \dots, P_n$** holding **confidential** input data $x_1, \dots, x_n$
- The parties want to jointly calculate a function of the form $y = f(x_1, \dots, x_n)$
- **All parties** should **receive y** however **no additional information** should be disclosed to any parties and outsiders other than what was known before
- y can be calculated satisfying the above requirements by **executing a protocol**
- We refer to this as Secure Multi-Party Computation **(SMPC)**
- Algorithmic tools to implement SMPC: **secret sharing schemes, homomorphic encryption, garbled circuits etc.**

THREATS TO SMPC

	Semi-honest	**Malicious**	**Covert**
Adversary type	Follows instructions Analyze protocol later to learn information about input of parties	Deviate from protocol	Can deviate from protocol Detected with a minimum fixed probability Can be penalized to deter such actions

SECURITY REQUIREMENTS

Requirement	Definition
Privacy	Each party learns nothing more than their designated output
Correctness	Correct output
Guarantee of output	Output to honest parties not interrupted by corrupted parties
Independence of Input	Input from corrupted party independent from input of honest parties
Robustness	Computation process resists attempts by adversaries to alter outcome or disrupt computation
Verifiability	System detects if adversary deviating from protocol Takes appropriate actions such as terminating the computation or excluding the adversary
Fairness	No party receives their output before others
Probability to catch deviations	Honest parties have a probability to catch corrupted parties on violations of the protocol

TYPES OF SMPC SYSTEMS

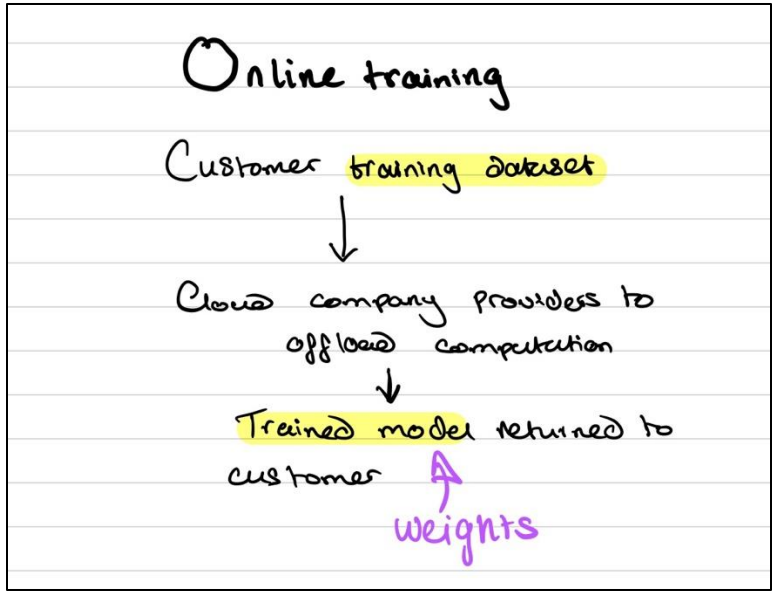
SMPC Type	Data Provider Operation	External Service Operation
Server-side	Data sharing to multiple non-colluding nodes	Most of SMPC operations
Peer-to-peer	Most of SMPC operations	N/A
Server-aided	Most of SMPC operations	Some SMPC operations performed by aiding server

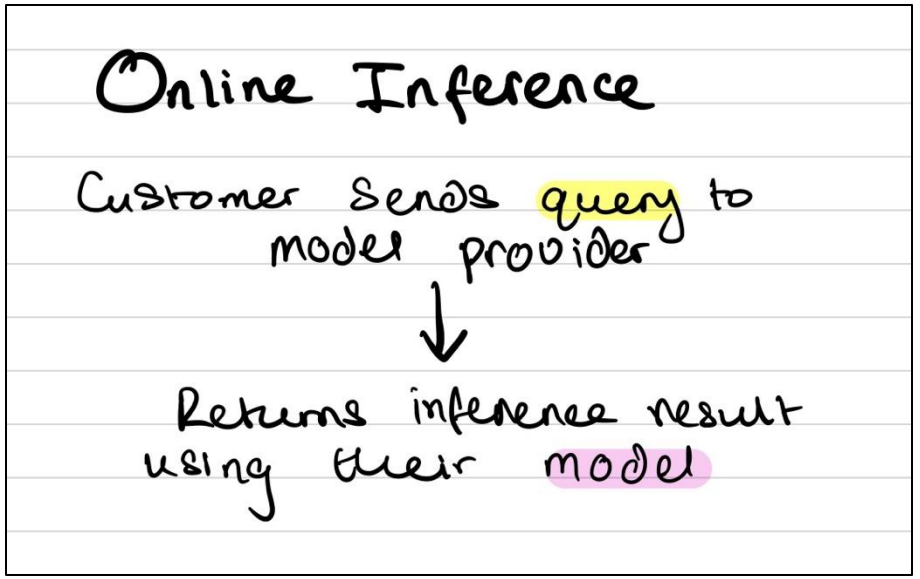
AN EXAMPLE – SECRET SHARING

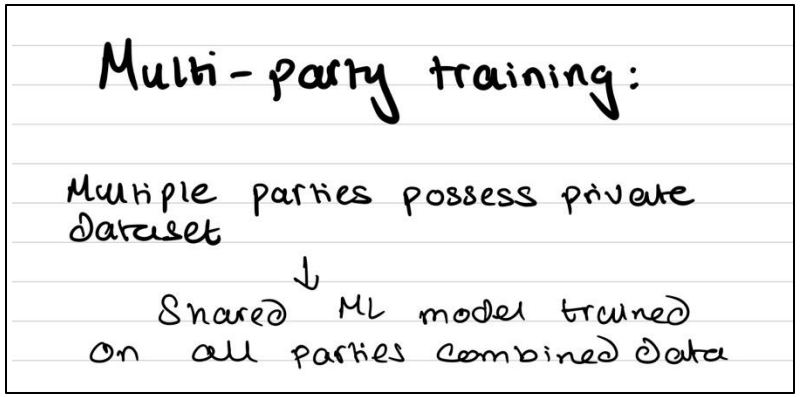
- **Secret sharing**: **secret divided** among a group of parties such that only a **specific subset** can **reconstruct original secret**
- An example is additive secret sharing

Secret x , n parties → Randomly assign first $n-1$ shares from a finite field F with field size p → Final share is $s_n = (x - \sum_{i=1}^{n-1} s_i) \ (mod\ p)$

$$x = (\sum_{i=1}^{n} s_i) \ (mod\ p)$$


Online training
Customer training dataset
Cloud company provides to offload computation
Trained model returned to customer
weights


Online Inference
Customer sends query to model provider
Returns inference result using their model


Multi-party training:
Multiple parties possess private dataset
Shared ML model trained on all parties combined data

SERVICES

SMPC UTILISSED FOR MACHINE LEARNING

Year	Type of service	Algorithmic tool implementing SMPC	SMPC Type	Security Model	Machine Learning Model
2018	Multi-party training	ElGamal-based encryption protocol	Server-aided	Malicious	Not mentioned
2019	Online training Multi-party training	Paillier scheme	Server-aided	Semi-honest	Autoencoders, ANN
2019	Online inference	Additive sharing	Server-side	Semi-honest	Polynomial regression
2019	Online inference	Additive and binary sharing	Peer-to-peer	Malicious	Decision trees, SVM

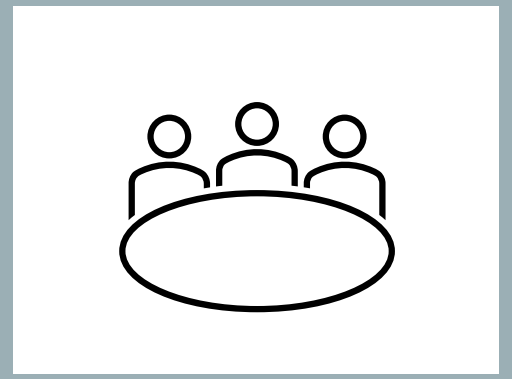
EXAMPLE III – SETTING THE SCENE

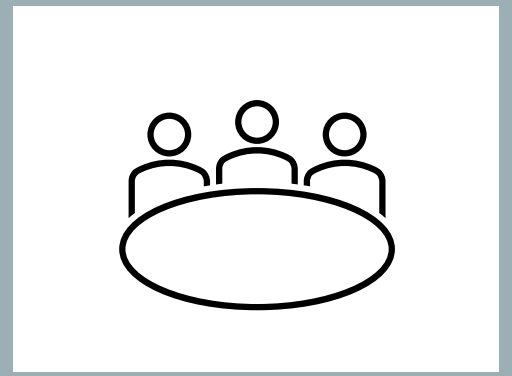
- n parties
- $D_1, \dots, D_n$
- Dataset vertically partitioned
- Each party only has access to a given selection of attributes
- AIM: Find k-nearest neighbors of some input query x_q without disclosing their private data

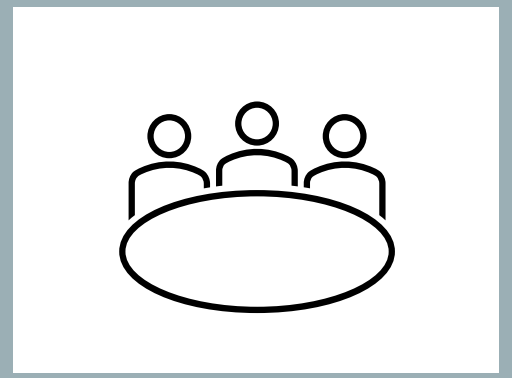
Age	Gender	Blood sugar level

…

.
.
.

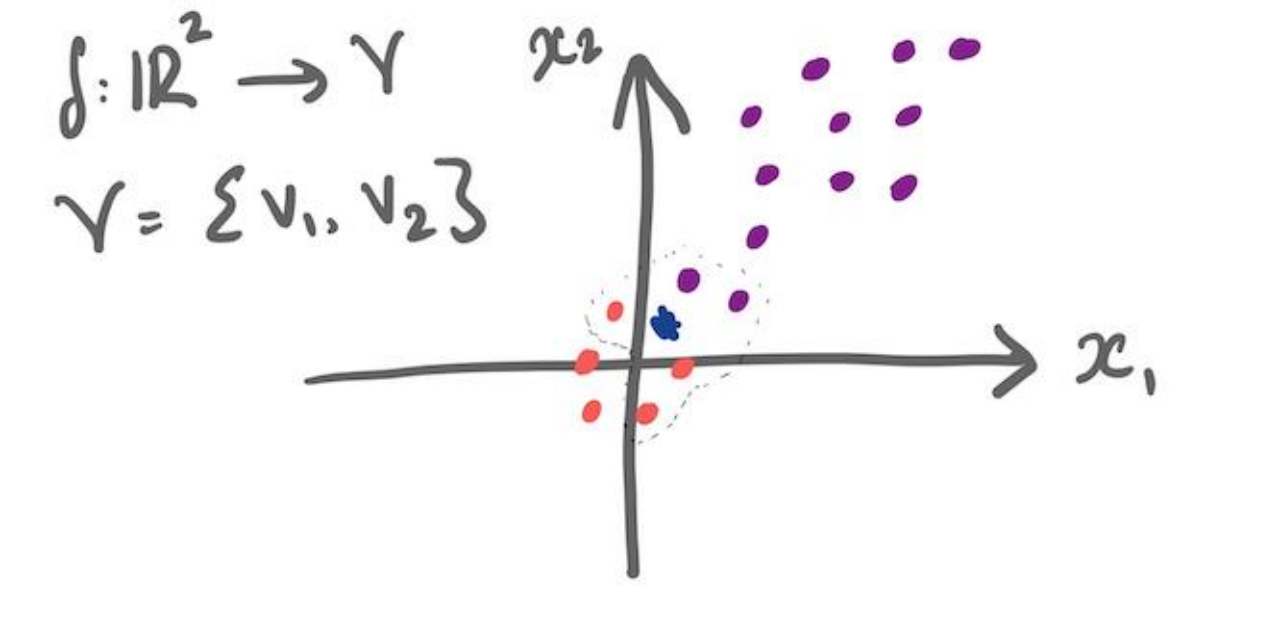






K-NN CLASSIFIERS

- $f: R^n \rightarrow V$, $V = \{v_1, \ldots, v_n\}$
- List of training examples $(x, f(x))$



$x_q \in R^n$ → $x_1, \ldots, x_k \in R^n$ are k nearest instances in training examples to x_q → $\hat{f}(x_q) \leftarrow argmax_{v \in V} \sum_{i=1}^{k} \delta(v, f(x_i))$

THE ISSUE

- Suppose there are **2 parties, 2 attributes** and **4 instances/training examples** and we are looking for **k = 2 nearest neighbors**

Age	**Gender**	$\mathbf{f: R^2 \rightarrow V}$
x_{11}	x_{21}	1
x_{12}	x_{22}	1
x_{13}	x_{23}	2
x_{14}	x_{24}	2

- x_{ij} refers to the jth **instance** of **attribute** i
- **Cannot** use previous slide as **each party** only has **access** to a **portion** of the **training samples**

POSSIBLE SOLUTION

- Theory of following slides inferred from (Zhan, 2005)
- Suppose we have an input query $\mathbf{x_q} = [x_{q1}, x_{q2}]$

Age	Gender	Distance Portion I	Distance Portion II
x_{11}	x_{21}	$s_{11} = x_{11} - x_{q1}$	$s_{12} = x_{21} - x_{q2}$
x_{12}	x_{22}	$s_{21} = x_{12} - x_{q1}$	$s_{22} = x_{22} - x_{q2}$
x_{13}	x_{23}	$s_{31} = x_{13} - x_{q1}$	$s_{32} = x_{23} - x_{q2}$
x_{14}	x_{24}	$s_{41} = x_{14} - x_{q1}$	$s_{42} = x_{24} - x_{q2}$

$\sum_{i=1}^{2} s_{1i}$	$\sum_{i=1}^{2} s_{2i}$	$\sum_{i=1}^{2} s_{3i}$	$\sum_{i=1}^{2} s_{4i}$

THE ISSUE

- **Privacy** of attributes of each party is **compromised!**
- Query request $x_q = [x_{q1}, x_{q2}]$
- Get $d = x_{11} - x_{q1}$
- **Party 2** can then learn the **attribute** of the **first instance** of **party 1** and **repeat** this for **all instances**

THE SOLUTION – HOMOMORPHIC ENCRYPTION

$$enc: E \rightarrow D\ , m_1, m_2 \in E$$
$$enc(m_1 + m_2) = enc(m_1) * enc(m_2)$$

Used **within a protocol** (given in [11]) allows for **distance portions** of **each party** to **remain hidden** whilst also finding **k-nearest neighbors.**

Homomorphic

Group homomorphism i.e., $(E,*_E), (D,*_D)$ are groups

$$enc(m_1 *_E m_2) = enc(m_1) *_D enc(m_2)$$

EXAMPLE II CONCLUDED

- Without loss of generality, one of the parties will have:
- $[\sum_{l=1}^{2} s_{il} - \sum_{l=1}^{2} s_{jl} : i, j \in [1, 4]]$
- A **corresponding map** of **+1/-1**
- For example, **first row, second column** being $-1 \Rightarrow \sum_{l=1}^{2} s_{1l} < \sum_{l=1}^{2} s_{2l}$
- Choose **instances S_i** with the **smallest weights**
- These instances have the **smallest total distance portions** over all **attributes**
- So, we choose **instance 1** and **instance 2** as the **2 nearest neighbors** to x_q

	S_1	S_2	S_3	S_4	Weights
S_1	$+1$	-1	-1	-1	-2
S_2	$+1$	$+1$	-1	$+1$	$+2$
S_3	$+1$	$+1$	$+1$	$+1$	$+4$
S_4	$+1$	-1	-1	$+1$	0

TO CONCLUDE…

- Looked at ways machine learning and cryptology intersect, giving motivating examples for the use of each method
- **CryptoNets**: A NN designed to work over encrypted data and return an encrypted prediction
- **SMPC**: Parties compute a **desired output** from their **inputs**, whilst keeping the **inputs hidden**. Used for many scenarios such as **online inference, online training** and **multi-party training**
- **K-NN Classifier**: **Executes a protocol** which allows for **each party** to know the **k nearest neighbors** of an **input query** x_q from a **collaboration** of their **inputs**, whilst keeping the **inputs hidden (vertically portioned dataset)**

FUTURE CONSIDERATIONS

- Plethora of ML methods which may require private data (dependent upon scenario)
- For example, some broad ideas:
- **Linear regression and non-linear regression** can be performed via **levelled homomorphic encryption** (similar approach to CryptoNets)
- May have some two-dimensional data e.g., $[x_1, x_2]$ and want a third-party service to perform **clustering** on this data **(group similar points together)**
- **Encrypt** the data such that **distance between points preserved**
- Classify a datapoint into which disease it belongs to , an equivalent form of CryptoNets for classification of a datapoint

THANKS SO MUCH! ☺

REFERENCES

- Gilad-Bachrach, R., Dowlin, N., Laine, K., Lauter, K., Naehrig, M. and Wernsing, J., 2016, June. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In International conference on machine learning (pp. 201-210). PMLR.
- Zhan, J.Z., Chang, L. and Matwin, S., 2005. Privacy preserving k-nearest neighbor classification. Int. J. Netw. Secur., 1(1), pp.46-51.
- Zhou, I., Tofigh, F., Piccardi, M., Abolhasan, M., Franklin, D. and Lipman, J., 2024. Secure Multi-Party Computation for Machine Learning: A Survey. *IEEE Access*.