SHAFT: Secure, Handy, Accurate, and Fast Transformer Inference

Artifact Evaluated
NDSS SYMPOSIUM
Available
Functional
Reproduced

Andes Y. L. Kei, Sherman S. M. Chow
Department of Information Engineering
The Chinese University of Hong Kong

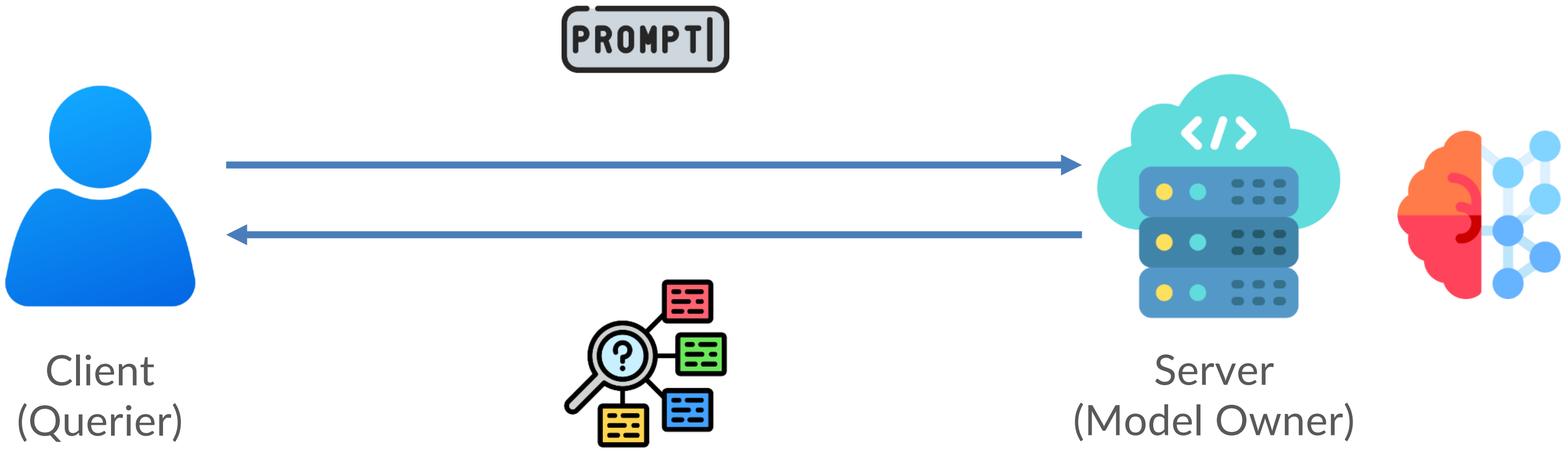
NDSS SYMPOSIUM/2025
Internet Society

Background

- **Transformer**: the current "standard" architecture for large language models (LLMs)

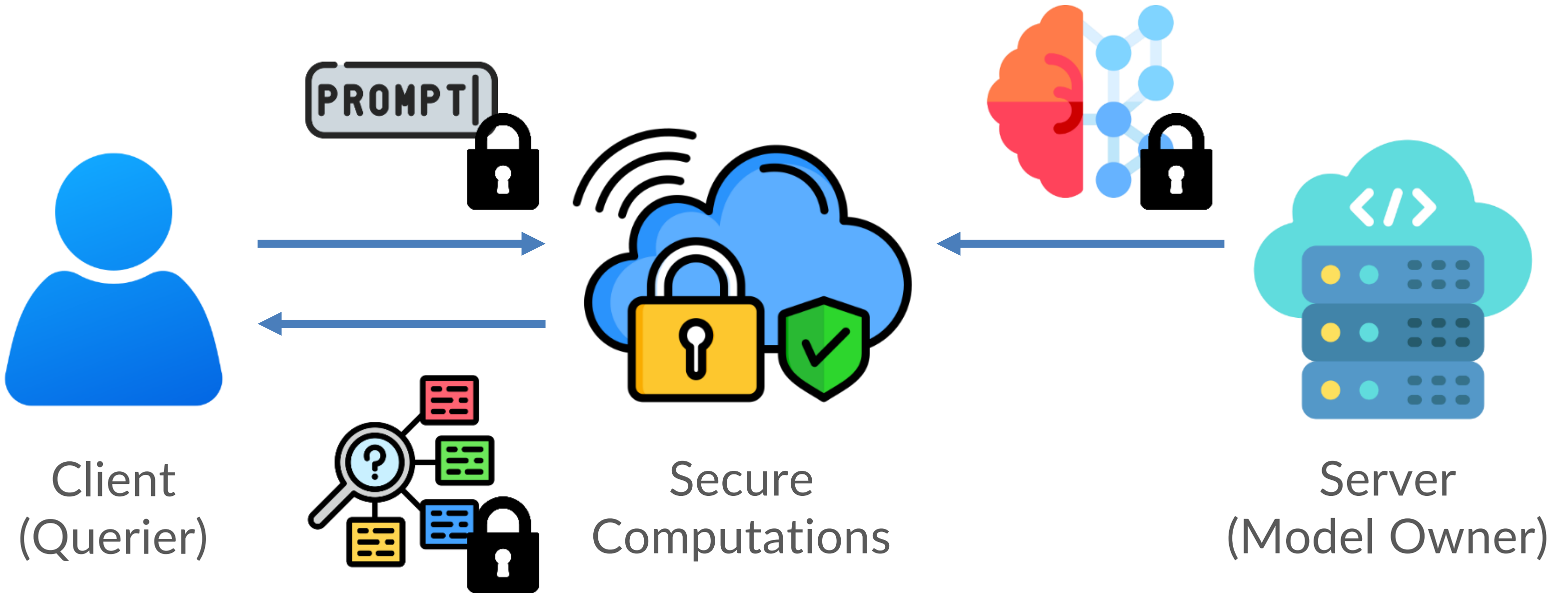
Background

- **Transformer**: the current "standard" architecture for large language models (LLMs)
- **Privacy concerns** arise due to the increasing adoption of LLMs (*e.g.*, ChatGPT)



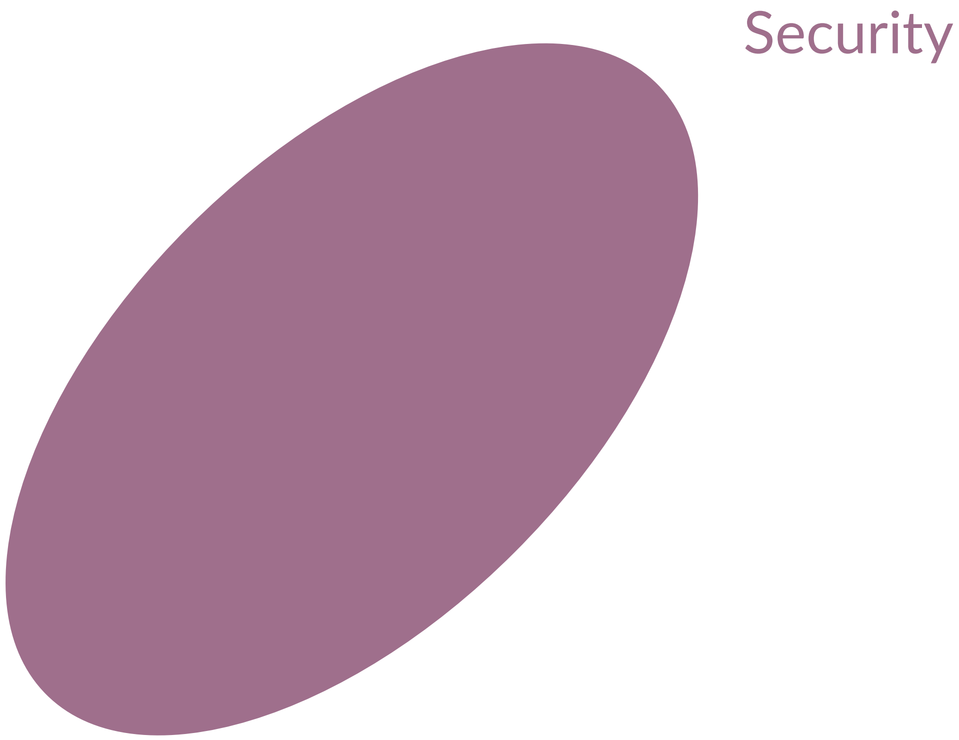
Background

- **Transformer**: the current "standard" architecture for large language models (LLMs)
- **Privacy concerns** arise due to the increasing adoption of LLMs (*e.g.*, ChatGPT)
- **Private inference**: performs model inference without leaking the ***model*** or ***query***

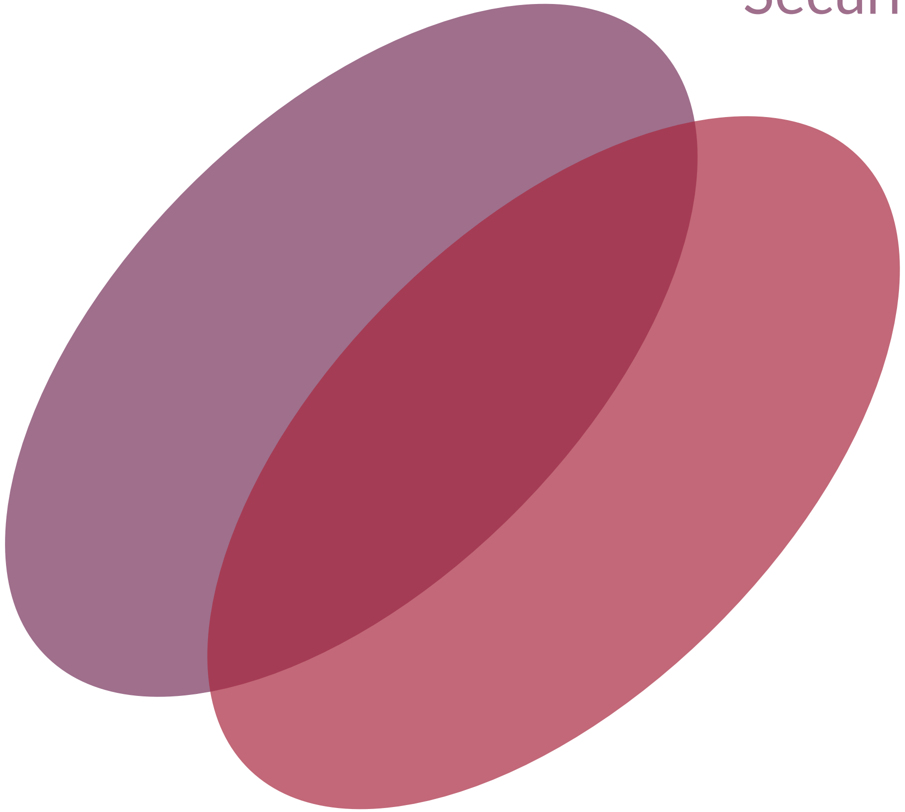


L. Ng, **S. Chow**. "GForce: GPU-Friendly Oblivious and Rapid Neural Network Inference." *Usenix Security* 2021.

Desirable Properties for Private Transformer Inference



Desirable Properties for Private Transformer Inference

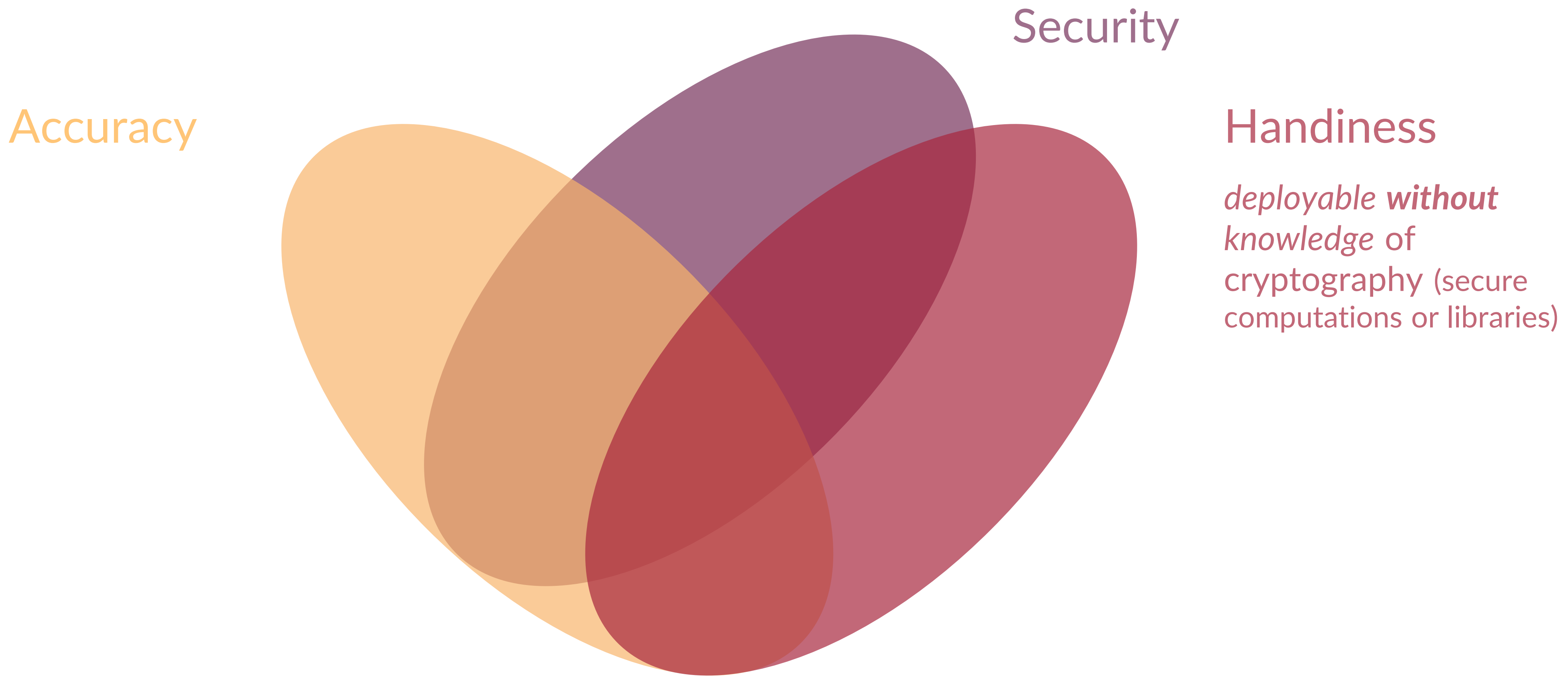


Security

Handiness

deployable ***without*** *knowledge* of cryptography (secure computations or libraries)

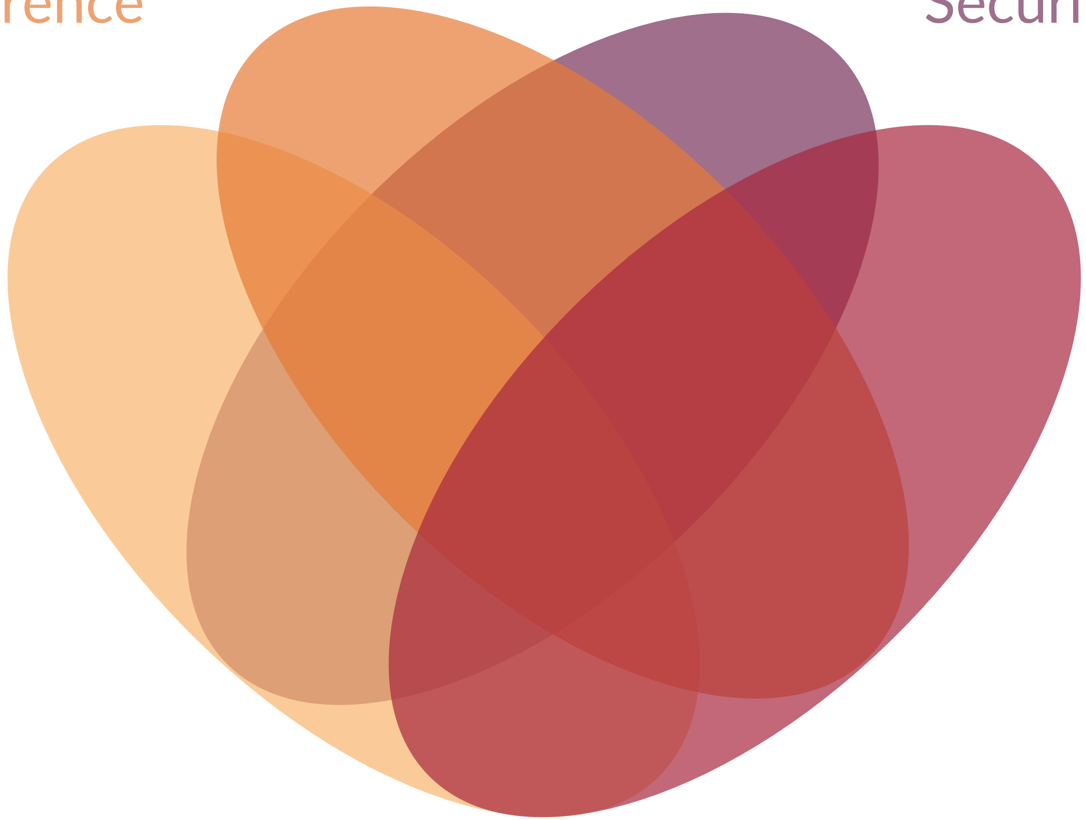
Desirable Properties for Private Transformer Inference



Desirable Properties for Private Transformer Inference

Fast Inference

Accuracy



Security

Handiness

deployable ***without*** *knowledge* of cryptography (secure computations or libraries)

Desirable Properties for Private Transformer Inference

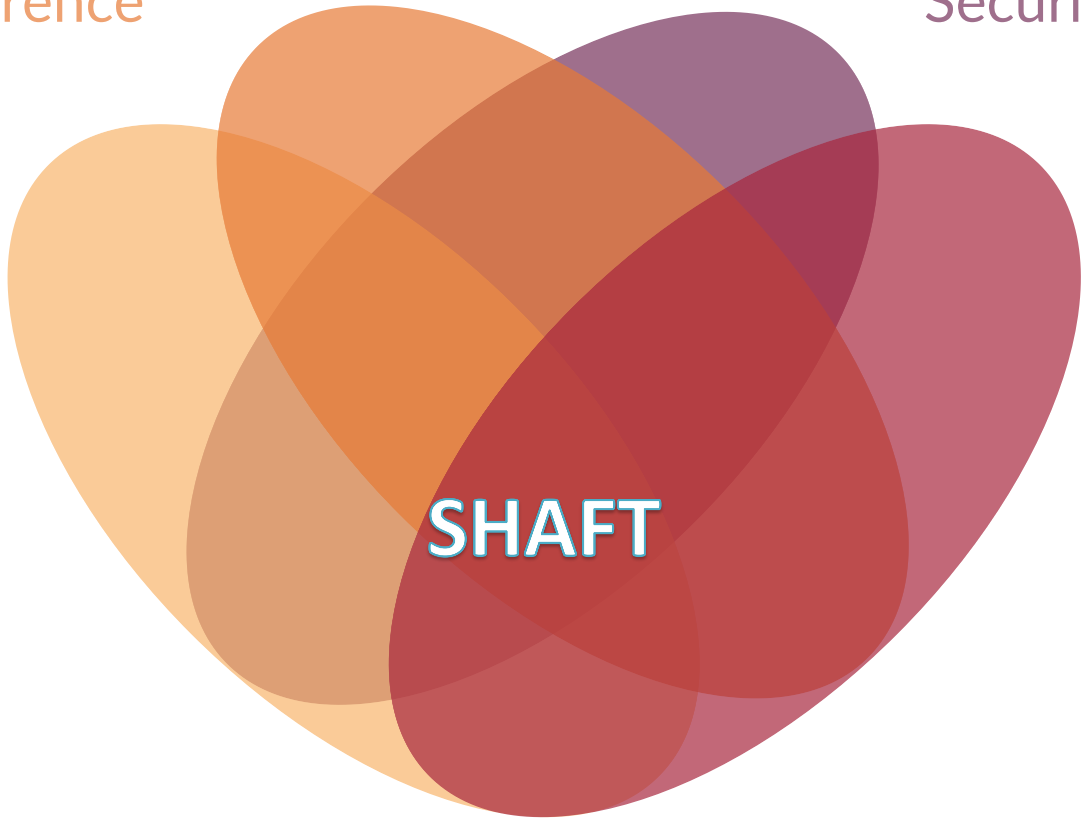
Fast Inference

Security

Accuracy

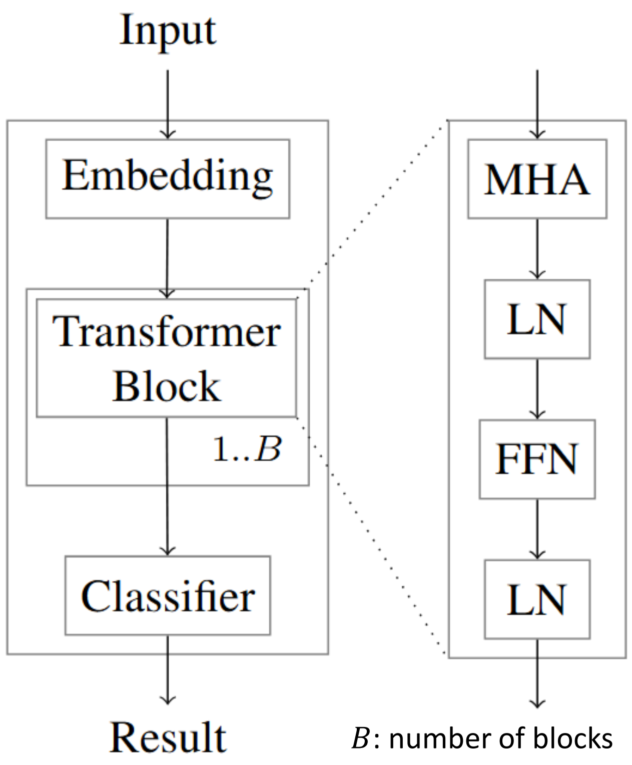
Handiness

*deployable **without** knowledge* of cryptography (secure computations or libraries)



Challenges in Private Transformer Inference

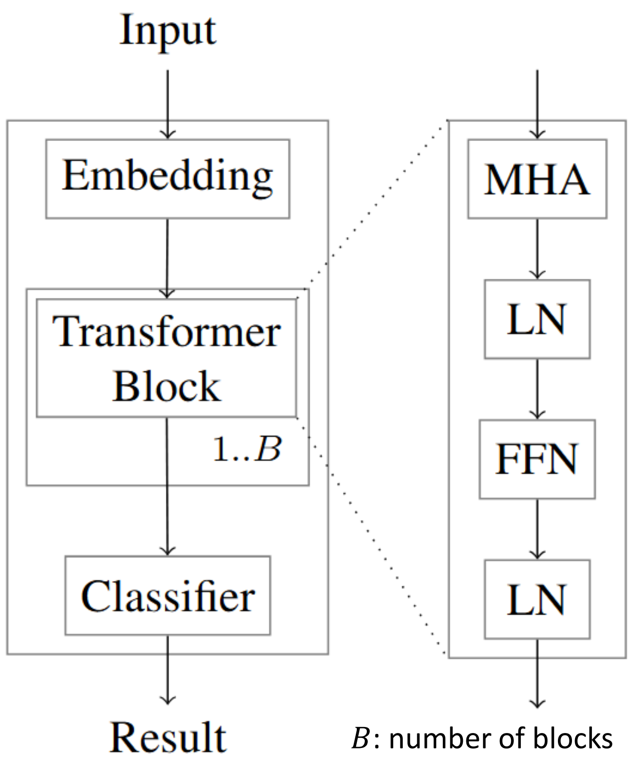
- Embedding
  - One-hot vector conversion
- Multi-head attention (MHA)
  - **Softmax**
- Feed-forward network (FFN)
  - **Gaussian error linear unit (GELU)**
- Layer normalization (LN)
  - Inverse square root



Note: other **linear** operations/layers within the above can be easily realized.

Challenges in Private Transformer Inference

- Embedding
  - One-hot vector conversion
- Multi-head attention (MHA)
  - **Softmax**
- Feed-forward network (FFN)
  - **Gaussian error linear unit (GELU)**
- Layer normalization (LN)
  - Inverse square root
- More challenging than (convolutional) neural networks
  - **Expensive secure softmax** in **every** MHA layer
  - **Sophisticated secure GELU** activation in **every** FFN layer



Note: other **linear** operations/layers within the above can be easily realized.

Private Transformer Inference Frameworks

Framework	Core Techniques	Accuracy	Efficiency
MPCFormer (ICLR ’23)	Rough Approximations		✓✗
SIGMA (PETS ’24)			
BumbleBee (NDSS ’25)			
SHAFT (Ours)			

D. Li et al. “MPCFormer: Fast, performant and private transformer inference with MPC.” *ICLR* 2023.
K. Gupta et al. “SIGMA: Secure GPT inference with function secret sharing.” *PETS* 2024.
W. Lu et al. “BumbleBee: Secure two-party inference framework for large transformers.” *NDSS* 2025.

Private Transformer Inference Frameworks

Framework	Core Techniques	Accuracy	Efficiency
MPCFormer (ICLR ’23)	Rough Approximations		✓✗
SIGMA (PETS ’24)			
BumbleBee (NDSS ’25)			
SHAFT (Ours)			

- Newer works applied **recent paradigm** or improved secure **linear** protocols

D. Li et al. “MPCFormer: Fast, performant and private transformer inference with MPC.” *ICLR* 2023.
K. Gupta et al. “SIGMA: Secure GPT inference with function secret sharing.” *PETS* 2024.
W. Lu et al. “BumbleBee: Secure two-party inference framework for large transformers.” *NDSS* 2025.

Private Transformer Inference Frameworks

Framework	Core Techniques	Accuracy	Efficiency
MPCFormer (ICLR ’23)	Rough Approximations		✓✗
SIGMA (PETS ’24)	Lookup Tables from FSS	✓	✓✗
BumbleBee (NDSS ’25)			
SHAFT (Ours)			

- Newer works applied **recent paradigm** or improved secure **linear** protocols
  - SIGMA uses **function secret sharing (FSS)** to reduce running time

D. Li et al. “MPCFormer: Fast, performant and private transformer inference with MPC.” *ICLR* 2023.
K. Gupta et al. “SIGMA: Secure GPT inference with function secret sharing.” *PETS* 2024.
W. Lu et al. “BumbleBee: Secure two-party inference framework for large transformers.” *NDSS* 2025.

Private Transformer Inference Frameworks

Framework	Core Techniques	Accuracy	Efficiency
MPCFormer (ICLR ’23)	Rough Approximations		✓✗
SIGMA (PETS ’24)	Lookup Tables from FSS	✓	✓✗
BumbleBee (NDSS ’25)	RLWE-based Homomorphic Multiplication	✓	✓✗
SHAFT (Ours)			

- Newer works applied **recent paradigm** or improved secure **linear** protocols
  - SIGMA uses **function secret sharing (FSS)** to reduce running time
  - BumbleBee optimizes homomorphic **matrix multiplication** to save communication

D. Li et al. “MPCFormer: Fast, performant and private transformer inference with MPC.” *ICLR* 2023.
K. Gupta et al. “SIGMA: Secure GPT inference with function secret sharing.” *PETS* 2024.
W. Lu et al. “BumbleBee: Secure two-party inference framework for large transformers.” *NDSS* 2025.

Private Transformer Inference Frameworks

Framework	Core Techniques		Accuracy	Efficiency
MPCFormer (ICLR ’23)	Rough Approximations			✓✗
SIGMA (PETS ’24)	Lookup Tables from FSS		✓	✓✗
BumbleBee (NDSS ’25)	RLWE-based Homomorphic Multiplication		✓	✓✗
SHAFT (Ours)	**ODE**	**Fourier Series**	✓	✓

- Newer works applied **recent paradigm** or improved secure **linear** protocols
  - SIGMA uses **function secret sharing (FSS)** to reduce running time
  - BumbleBee optimizes homomorphic **matrix multiplication** to save communication

- Observation: ***non-linear*** function approximations remain **under**explored

D. Li et al. “MPCFormer: Fast, performant and private transformer inference with MPC.” *ICLR* 2023.
K. Gupta et al. “SIGMA: Secure GPT inference with function secret sharing.” *PETS* 2024.
W. Lu et al. “BumbleBee: Secure two-party inference framework for large transformers.” *NDSS* 2025.

Private Transformer Inference Frameworks

Framework	Security	Handiness	Accuracy	Efficiency
MPCFormer (ICLR ’23)	✓✗			✓✗
SIGMA (PETS ’24)	✓		✓	✓✗
BumbleBee (NDSS ’25)	✓		✓	✓✗
SHAFT (Ours)	✓			

Private Transformer Inference Frameworks

Framework	Security	Handiness	Accuracy	Efficiency
MPCFormer (ICLR ’23)	✓✗			✓✗
SIGMA (PETS ’24)	✓		✓	✓✗
BumbleBee (NDSS ’25)	✓		✓	✓✗
SHAFT (Ours)	✓			✓

- SHAFT outperforms two recent works in **efficiency**
  - vs. SIGMA: **reduces communication** by 25-41% with **similar running time**
  - vs. BumbleBee: 4.6-**5.3× faster** on LAN, 2.9-**4.4× faster** on WAN

OS: Ubuntu 20.04.

CPU: Intel Xeon Gold 5318Y, GPU: two NVIDIA A40, RAM: 256 GB.

Models: BERT-base, BERT-large, GPT-2, ViT-base.

Private Transformer Inference Frameworks

Framework	Security	Handiness	Accuracy	Efficiency
MPCFormer (ICLR '23)	✓✗			✓✗
SIGMA (PETS '24)	✓		✓	✓✗
BumbleBee (NDSS '25)	✓		✓	✓✗
SHAFT (Ours)	✓	✓	✓	✓

- SHAFT outperforms two recent works in **efficiency**
  - vs. SIGMA: **reduces communication** by 25-41% with **similar running time**
  - vs. BumbleBee: 4.6-**5.3× faster** on LAN, 2.9-**4.4× faster** on WAN
- SHAFT achieves **comparable accuracy to plaintext inference**

OS: Ubuntu 20.04.

CPU: Intel Xeon Gold 5318Y, GPU: two NVIDIA A40, RAM: 256 GB.

Models: BERT-base, BERT-large, GPT-2, ViT-base.

Datasets: QNLI, CoLA, SST-2 from the GLUE benchmark.

Private Transformer Inference Frameworks

Framework	Security	Handiness	Accuracy	Efficiency
MPCFormer (ICLR ’23)	✓✗			✓✗
SIGMA (PETS ’24)	✓		✓	✓✗
BumbleBee (NDSS ’25)	✓		✓	✓✗
SHAFT (Ours)	✓	✓	✓	✓

- For **handy** deployment of secure inference, we offer an **open-source** framework
  - **PyTorch-like APIs** smoothly integrate with the **Hugging Face** transformer library

PyTorch Hugging Face

L. Ng, **S. Chow**. “SoK: Cryptographic Neural-Network Computation.” *S&P* 2023.

Importing Hugging Face Transformers

- Existing works like CrypTen (NeurIPS '21) allow importing **simple** models
  - but **lack support** for **transformer-specific** layers (*e.g.*, GELU)
- We implement **code conversion** of these layers

```
from transformers import AutoModelForSequenceClassification
import crypten as ct
# (standard) data loading and preprocessing omitted
model = AutoModelForSequenceClassification.from_pretrained("user/bert-base-cased-qnli")
ct.init()
model_ss = ct.nn.from_pytorch(model, dummy_data).encrypt().cuda()
data_ss = ct.cryptensor(data).cuda()
output_ss = model_ss(data_ss)
output = output_ss.get_plain_text()
```

B. Knott et al. "CrypTen: Secure multi-party computation meets machine learning." *NeurIPS* 2021.

Our Technical Novelties

1. First **constant-round** private **softmax** protocol for transformers
   - Prior works need **logarithmic** rounds (in input length m) for **numerical stability**
   - We guarantee the same in **constant** rounds by uniquely combining **ordinary differential equation (ODE)** and **input clipping**

Our Technical Novelties

1. First **constant-round** private **softmax** protocol for transformers
   - Prior works need **logarithmic** rounds (in input length m) for **numerical stability**
   - We guarantee the same in **constant** rounds by uniquely combining **ordinary differential equation (ODE)** and **input clipping**

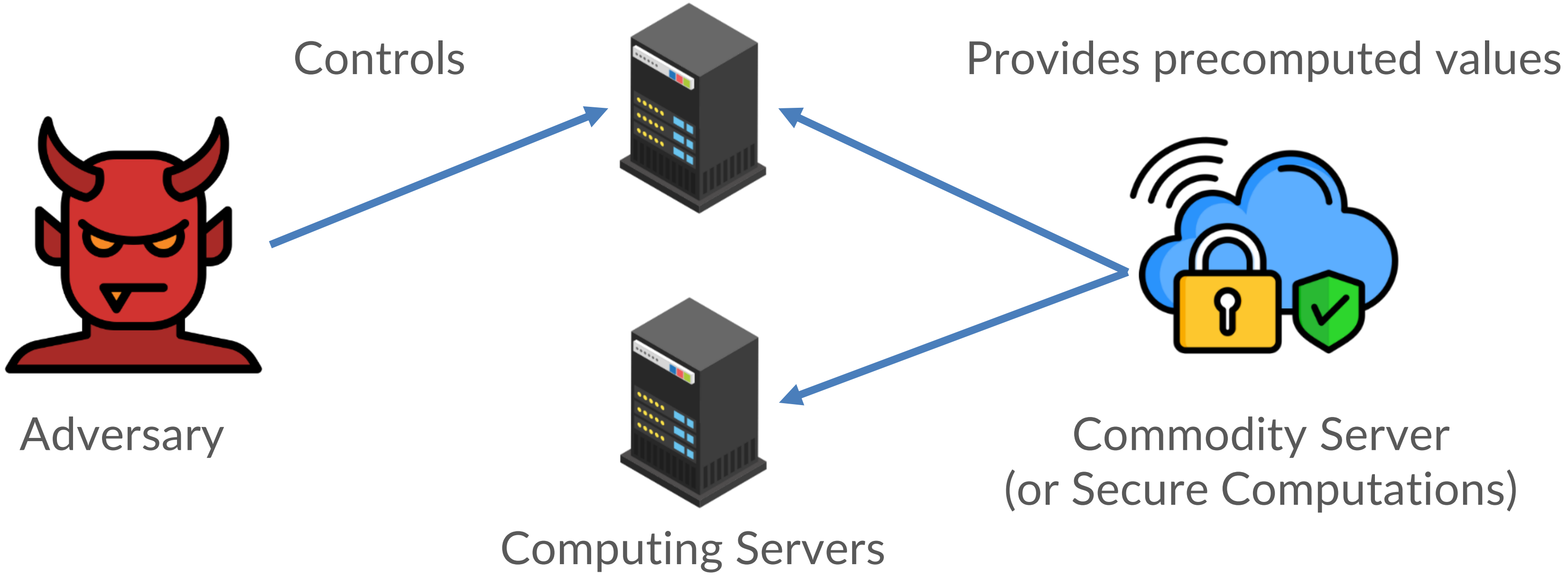
2. A **precise** and **efficient** private **GELU** protocol
   - We design a **GELU characterization** for **Fourier series** approximation
   - **Reduce** round complexity from **two** (S&P ’24) to **one**

hundreds of thousands of softmax & GELU in transformers

Q. Pang et al. “BOLT: Privacy-preserving, accurate and efficient inference for transformers.” *S&P* 2024.

Security Model: 2-Party Outsourced Setting w/ Precomputation

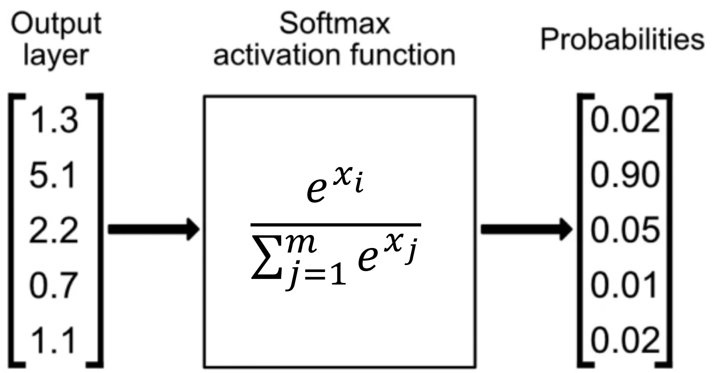
- Well-established 2-party setting, *e.g.*, NDSS ’09
- Model and query are **secret-shared** to two servers
- Adversary: **semi-honest**, controls **one** of the two servers
- **Commodity server** can be replaced by **2-party computation** between servers



S. Chow, J.-H. Lee, L. Subramanian. “Two-Party Computation Model for Privacy-Preserving Queries over Distributed Databases.” *NDSS* 2009.

Private Softmax with Private Max()

- $\mathbf{Softmax}(\vec{x})_i = e^{x_i} / \sum_j e^{x_j}$
  - Converts values in a vector into probabilities

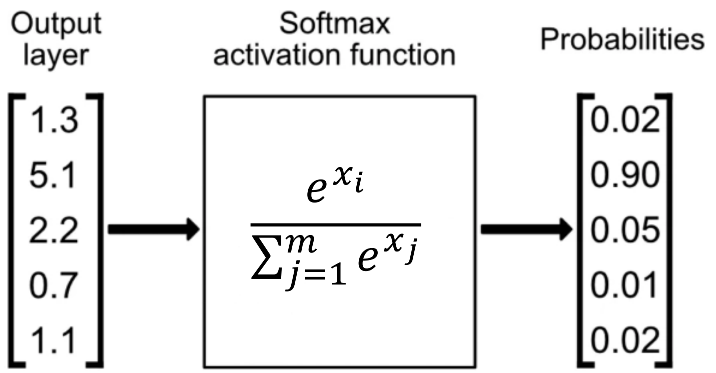


m: Input length

https://medium.com/towards-data-science/softmax-activation-function-explained-a7e1bc3ad60

Private Softmax with Private Max()

- $\textbf{Softmax}(\vec{x})_i = e^{x_i} / \sum_j e^{x_j}$
  - Converts values in a vector into probabilities
- Basic idea: evaluates a sequence of e^x and $1/x$
  - Problem: e^x and $1/x$ **overflow easily**

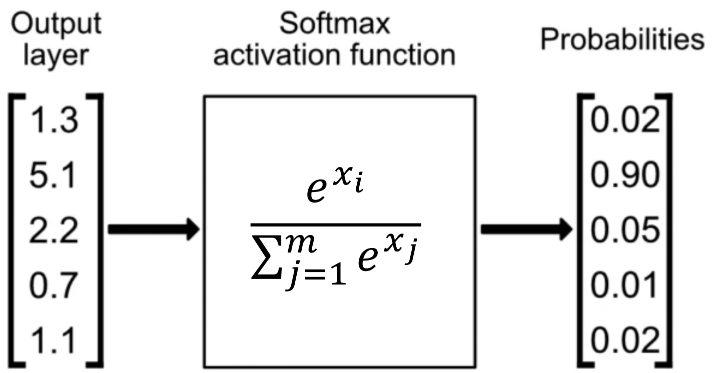


m: Input length

https://medium.com/towards-data-science/softmax-activation-function-explained-a7e1bc3ad60

Private Softmax with Private Max()

- $\textbf{Softmax}(\vec{x})_i = e^{x_i} / \sum_j e^{x_j}$
  - Converts values in a vector into probabilities
- Basic idea: evaluates a sequence of e^x and $1/x$
  - Problem: e^x and $1/x$ **overflow easily**
- Typical solution: computes $\text{Softmax}(\vec{x} - \max(\vec{x}))$

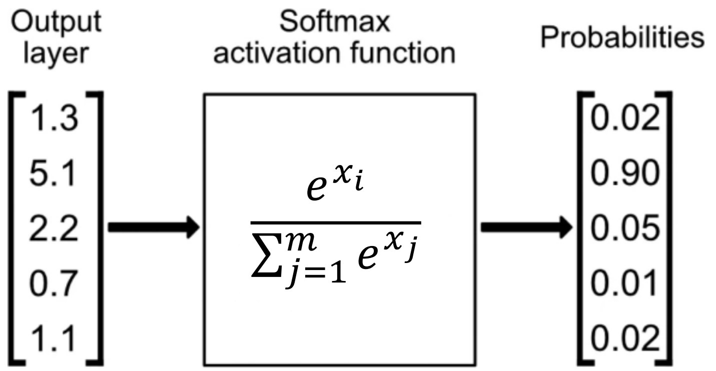


m: Input length

https://medium.com/towards-data-science/softmax-activation-function-explained-a7e1bc3ad60

Private Softmax with Private Max()

- $\mathbf{Softmax}(\vec{x})_i = e^{x_i} / \sum_j e^{x_j}$
  - Converts values in a vector into probabilities
- Basic idea: evaluates a sequence of e^x and $1/x$
  - Problem: e^x and $1/x$ **overflow easily**
- Typical solution: computes $\mathrm{Softmax}(\vec{x} - \max(\vec{x}))$



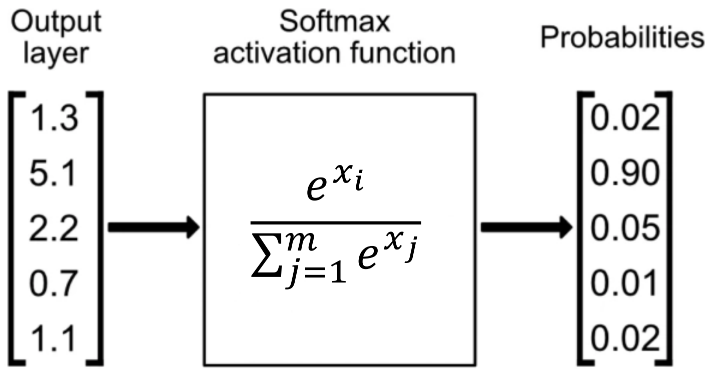
m: Input length

https://medium.com/towards-data-science/softmax-activation-function-explained-a7e1bc3ad60

- **Avoids overflows** without affecting correctness

Private Softmax with Private Max()

- $\textbf{Softmax}(\vec{x})_i = e^{x_i} / \sum_j e^{x_j}$
    - Converts values in a vector into probabilities
- Basic idea: evaluates a sequence of e^x and $1/x$
    - Problem: e^x and $1/x$ **overflow easily**
- Typical solution: computes $\text{Softmax}(\vec{x} - \max(\vec{x}))$



m: Input length

https://medium.com/towards-data-science/softmax-activation-function-explained-a7e1bc3ad60

- **Avoids overflows** without affecting correctness

- Secure evaluation of maximum requires **logarithmic** rounds

Numerically-Unstable Private Softmax with ODE (ACSAC '23)

- Let t be the number of iterations, m be the input length
- **ODE approximation** of $\mathrm{Softmax}(\vec{x})$:
  - Initial guess: $\vec{y}_0 = \vec{1}/m$
  - Iterative updates: $\vec{y}_i = \vec{y}_{i-1} + \frac{1}{t}(\vec{x} - \langle \vec{x}, \vec{y}_{i-1} \rangle \vec{1}) * \vec{y}_{i-1}$

Inner product
All-one vector
Entry-wise product

- Total **2**t rounds (2 per iteration)
- Needs **large** t (*e.g.*, 128) for **unbounded** $\vec{x}$ in transformers
  - Correctness **requires** $\max(\vec{x}) - \min(\vec{x}) \leq t$

Y. Zheng, Q. Zhang, **S. Chow**, Y. Peng, S. Tan, L. Li, S. Yin. "Secure softmax/sigmoid for machine-learning computation." *ACSAC* 2023.

Our Private Softmax with Input-Clipped ODE

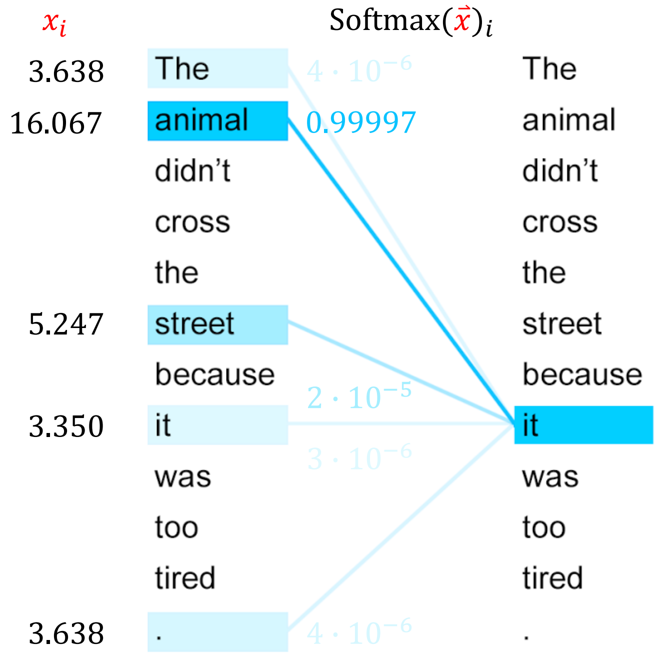
- Key idea: **clips** input to a **pre-defined** range $[a, b]$
  - $t = b - a$ ensures correctness, even with **small** t
  - **Constant round**!
- Why clipping matters in $\mathrm{Softmax}(\vec{x})_i = e^{x_i} / \sum_j e^{x_j}$?

Our Private Softmax with Input-Clipped ODE

- Key idea: **clips** input to a **pre-defined** range $[a, b]$
  - $t = b - a$ ensures correctness, even with **small** t
  - **Constant round**!
- Why clipping matters in $\mathrm{Softmax}(\vec{x})_i = e^{x_i} / \sum_j e^{x_j}$?
  - Attention layers map **word relevance** to probabilities

Our Private Softmax with Input-Clipped ODE

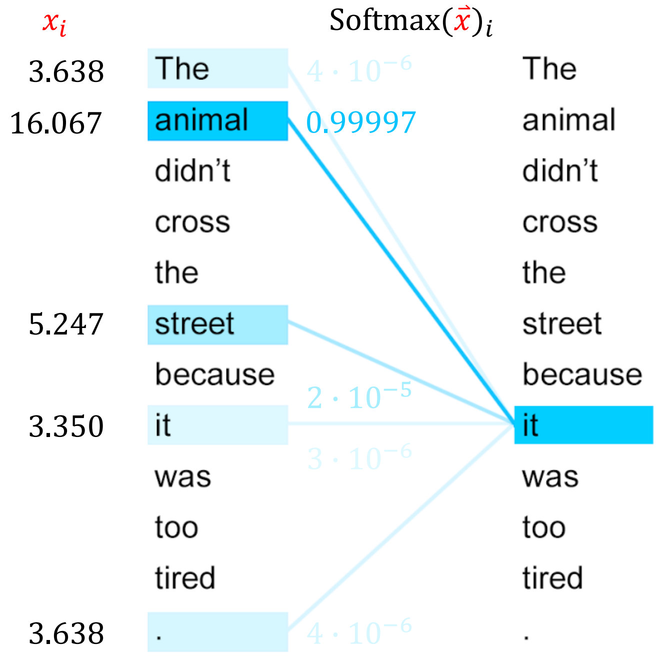
- Key idea: **clips** input to a **pre-defined** range $[a, b]$
  - $t = b - a$ ensures correctness, even with **small** t
  - **Constant round**!
- Why clipping matters in $\text{Softmax}(\vec{x})_i = e^{x_i} / \sum_j e^{x_j}$?
  - Attention layers map **word relevance** to probabilities
  - Words usually relate to *only a few* others in a sentence



https://research.google/blog/transformer-a-novel-neural-network-architecture-for-language-understanding

Our Private Softmax with Input-Clipped ODE

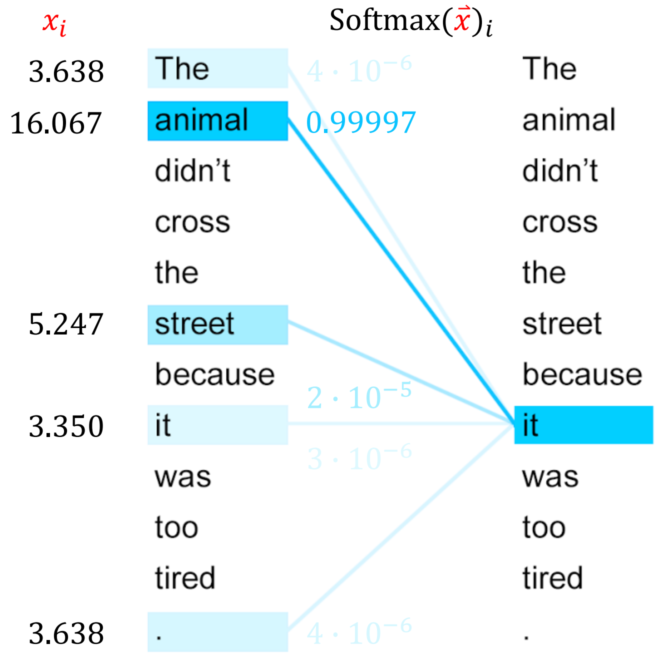
- Key idea: **clips** input to a **pre-defined** range $[a, b]$
  - $t = b - a$ ensures correctness, even with **small** t
  - **Constant round**!
- Why clipping matters in $\text{Softmax}(\vec{x})_i = e^{x_i} / \sum_j e^{x_j}$?
  - Attention layers map **word relevance** to probabilities
  - Words usually relate to *only a few* others in a sentence
  - **Most** x_j's are **small**, but **a few** large x_j's **dominate** the sum



https://research.google/blog/transformer-a-novel-neural-network-architecture-for-language-understanding

Our Private Softmax with Input-Clipped ODE

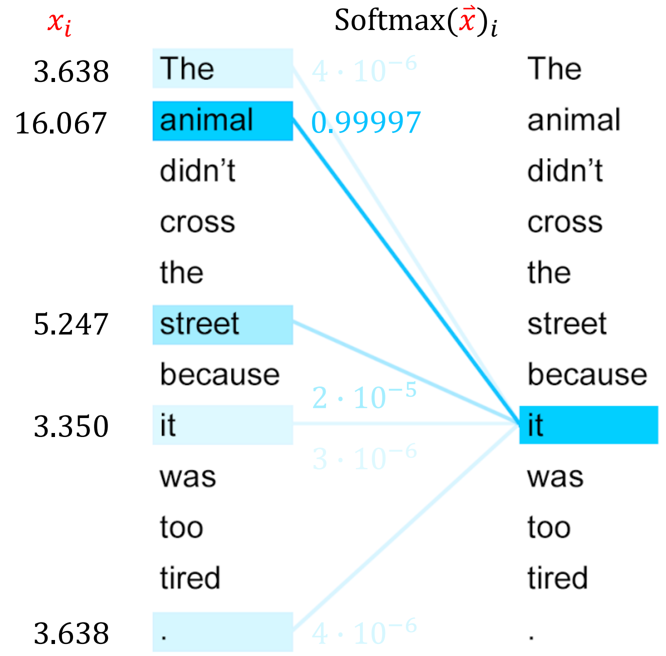
- Key idea: **clips** input to a **pre-defined** range $[a, b]$
  - $t = b - a$ ensures correctness, even with **small** t
  - **Constant round**!
- Why clipping matters in $\mathrm{Softmax}(\vec{x})_i = e^{x_i} / \sum_j e^{x_j}$?
  - Attention layers map **word relevance** to probabilities
  - Words usually relate to *only a few* others in a sentence
  - **Most** x_j's are **small**, but **a few** large x_j's **dominate** the sum
  - Clipping large x_j aggressively can cause **significant** errors



https://research.google/blog/transformer-a-novel-neural-network-architecture-for-language-understanding

Our Private Softmax with Input-Clipped ODE

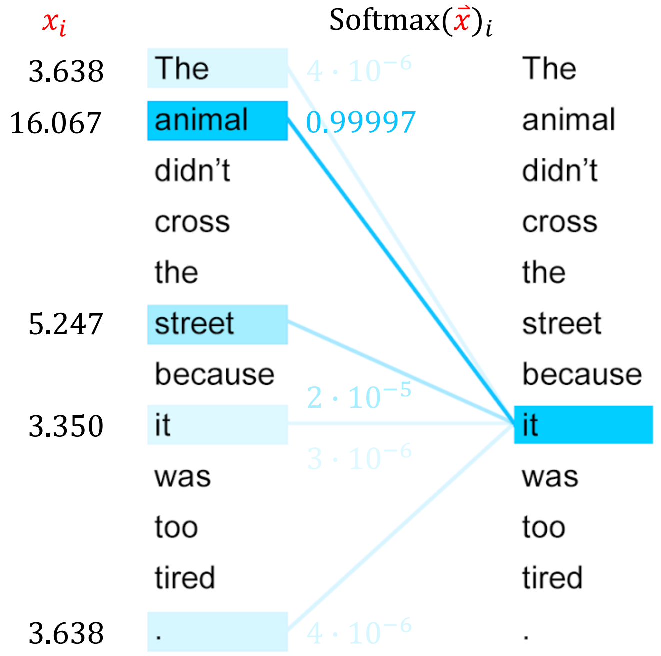
- Key idea: **clips** input to a **pre-defined** range $[a, b]$
  - $t = b - a$ ensures correctness, even with **small** t
  - **Constant round**!
- Why clipping matters in $\text{Softmax}(\vec{x})_i = e^{x_i} / \sum_j e^{x_j}$?
  - Attention layers map **word relevance** to probabilities
  - Words usually relate to *only a few* others in a sentence
  - **Most** x_j's are **small**, but **a few** large x_j's **dominate** the sum
  - Clipping large x_j aggressively can cause **significant** errors
- How to select a and b?



https://research.google/blog/transformer-a-novel-neural-network-architecture-for-language-understanding

Our Private Softmax with Input-Clipped ODE

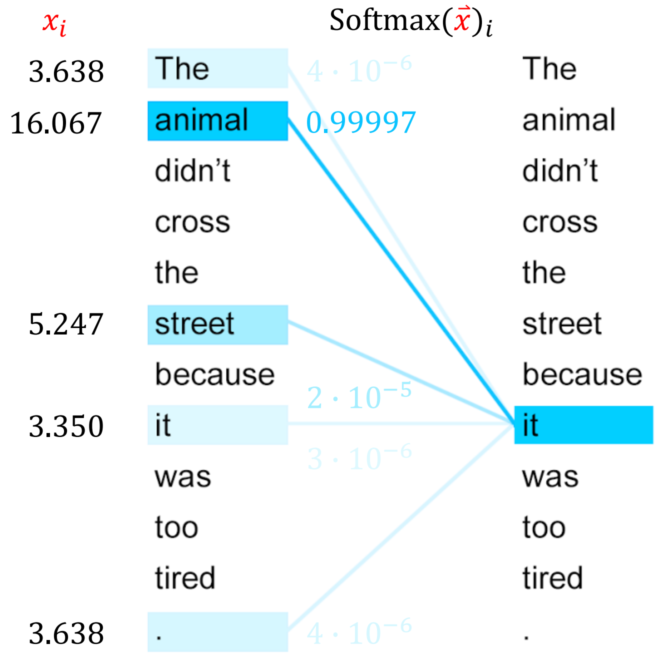
- Key idea: **clips** input to a **pre-defined** range $[a, b]$
  - $t = b - a$ ensures correctness, even with **small** t
  - **Constant round**!
- Why clipping matters in $\text{Softmax}(\vec{x})_i = e^{x_i} / \sum_j e^{x_j}$?
  - Attention layers map **word relevance** to probabilities
  - Words usually relate to *only a few* others in a sentence
  - **Most** x_j's are **small**, but **a few** large x_j's **dominate** the sum
  - Clipping large x_j aggressively can cause **significant** errors
- How to select a and b?
  - Sets a **larger positive** b to **minimize errors** from *large* x_j's



https://research.google/blog/transformer-a-novel-neural-network-architecture-for-language-understanding

Our Private Softmax with Input-Clipped ODE

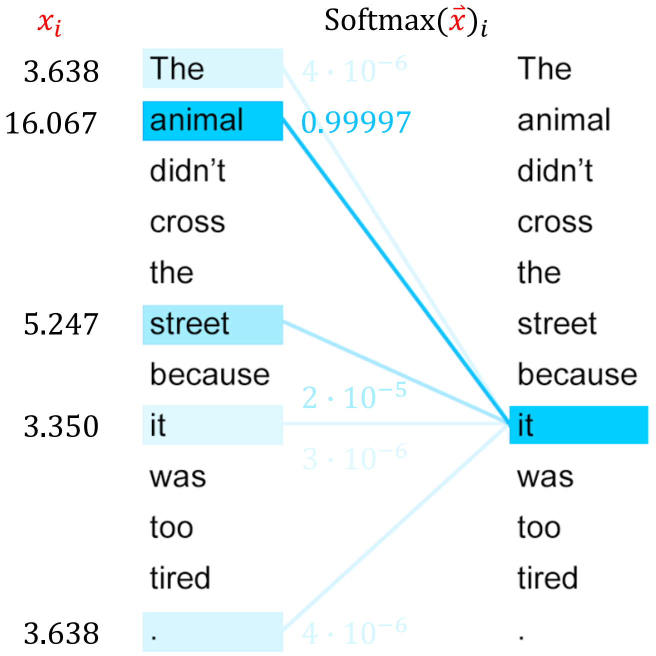
- Key idea: **clips** input to a **pre-defined** range $[a, b]$
  - $t = b - a$ ensures correctness, even with **small** t
  - **Constant round**!
- Why clipping matters in $\mathrm{Softmax}(\vec{x})_i = e^{x_i} / \sum_j e^{x_j}$?
  - Attention layers map **word relevance** to probabilities
  - Words usually relate to *only a few* others in a sentence
  - **Most** x_j's are **small**, but **a few** large x_j's **dominate** the sum
  - Clipping large x_j aggressively can cause **significant** errors
- How to select a and b?
  - Sets a **larger positive** b to **minimize errors** from *large* x_j's
  - Chooses a **slightly negative** a to **include** most *small* x_j's



https://research.google/blog/transformer-a-novel-neural-network-architecture-for-language-understanding

Our Private Softmax with Input-Clipped ODE

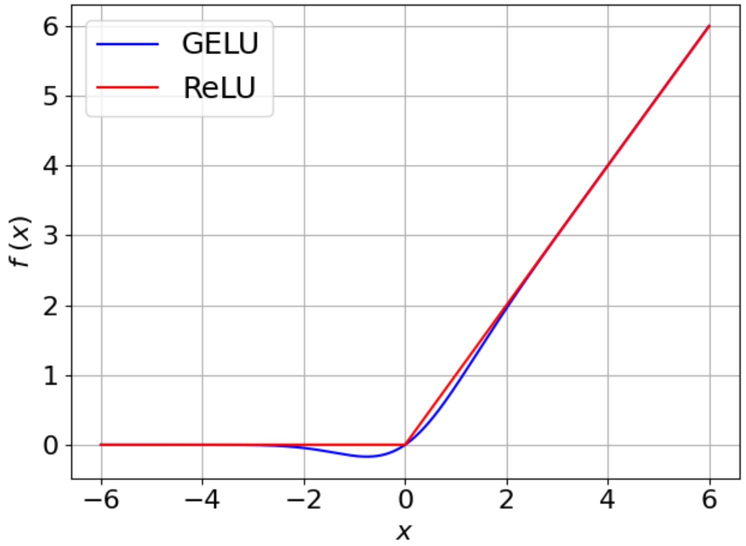
- Key idea: **clips** input to a **pre-defined** range $[a, b]$
  - $t = b - a$ ensures correctness, even with **small** t
  - **Constant round**!
- Why clipping matters in $\text{Softmax}(\vec{x})_i = e^{x_i} / \sum_j e^{x_j}$?
  - Attention layers map **word relevance** to probabilities
  - Words usually relate to *only a few* others in a sentence
  - **Most** x_j's are **small**, but **a few** large x_j's **dominate** the sum
  - Clipping large x_j aggressively can cause **significant** errors
- How to select a and b?
  - Sets a **larger positive** b to **minimize errors** from *large* x_j's
  - Chooses a **slightly negative** a to **include** most *small* x_j's
  - $t = 16, a = -4, b = 12$ in all our experiments



https://research.google/blog/transformer-a-novel-neural-network-architecture-for-language-understanding

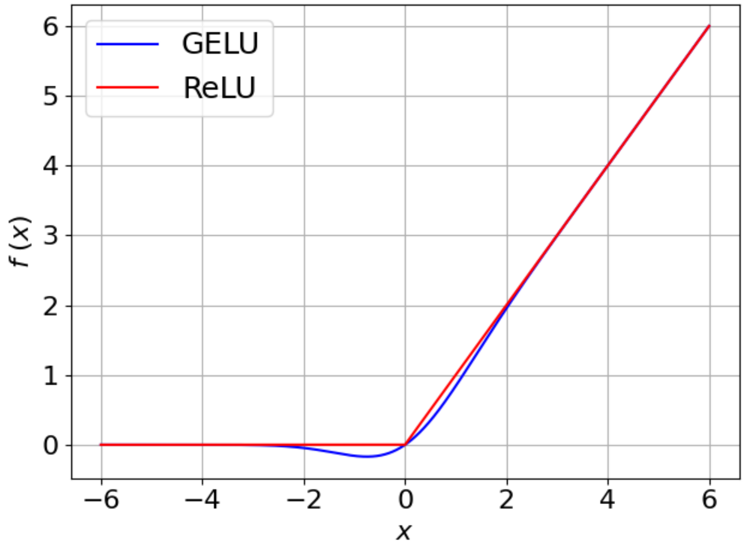
Private GELU with (Piecewise) Polynomial

- $\mathbf{GELU}(x) = 0.5x\left(1 + \mathrm{Erf}\left(x/\sqrt{2}\right)\right)$, $\mathrm{Erf}(x) = (2/\sqrt{x}) \int_0^x e^{-u^2}\, du$
- Standard approach:
  - Idea: $\mathrm{GELU}(x)$ is close to $\mathrm{ReLU}(x) = \max(x, 0)$ when $|x|$ is **relatively large**
  - Approximates $\mathrm{GELU}(x)$ for x near 0 with **polynomial(s)**
  - Sets $\mathrm{GELU}(x) = \mathrm{ReLU}(x)$ for larger $|x|$



Private GELU with (Piecewise) Polynomial

- $\mathbf{GELU}(x) = 0.5x\left(1 + \mathrm{Erf}\left(x/\sqrt{2}\right)\right)$, $\mathrm{Erf}(x) = \left(2/\sqrt{x}\right)\int_0^x e^{-u^2}\,du$
- Standard approach:
  - Idea: $\mathrm{GELU}(x)$ is close to $\mathrm{ReLU}(x) = \max(x, 0)$ when $|x|$ is **relatively large**
  - Approximates $\mathrm{GELU}(x)$ for x near 0 with **polynomial(s)**
  - Sets $\mathrm{GELU}(x) = \mathrm{ReLU}(x)$ for larger $|x|$
- State-of-the-art: a **degree-4** polynomial (S&P '24)
  - Secure evaluation requires **two** rounds
  - **Substantial** overheads for transformers with **hundreds of thousands** of GELU



Q. Pang et al. "BOLT: Privacy-preserving, accurate and efficient inference for transformers." *S&P* 2024.

Private GELU with Fourier Series (FS)

- **Fourier Series (FS)**: provide precise approximations of functions
  - with a **sinusoidal** shape for a **bounded** input range

Private GELU with Fourier Series (FS)

- **Fourier Series (FS)**: provide precise approximations of functions
  - with a **sinusoidal** shape for a **bounded** input range
- Securely evaluating an FS takes only **one** round (ACSAC '23)

Y. Zheng, Q. Zhang, **S. Chow**, Y. Peng, S. Tan, L. Li, S. Yin. "Secure softmax/sigmoid for machine-learning computation." *ACSAC* 2023.

Private GELU with Fourier Series (FS)

- **Fourier Series (FS)**: provide precise approximations of functions
  - with a **sinusoidal** shape for a **bounded** input range
- Securely evaluating an FS takes only **one** round (ACSAC '23)
- Problem: GELU is **not** sinusoidal (*i.e.*, **direct** approximation with FS **fails**)

Y. Zheng, Q. Zhang, **S. Chow**, Y. Peng, S. Tan, L. Li, S. Yin. "Secure softmax/sigmoid for machine-learning computation." *ACSAC* 2023.

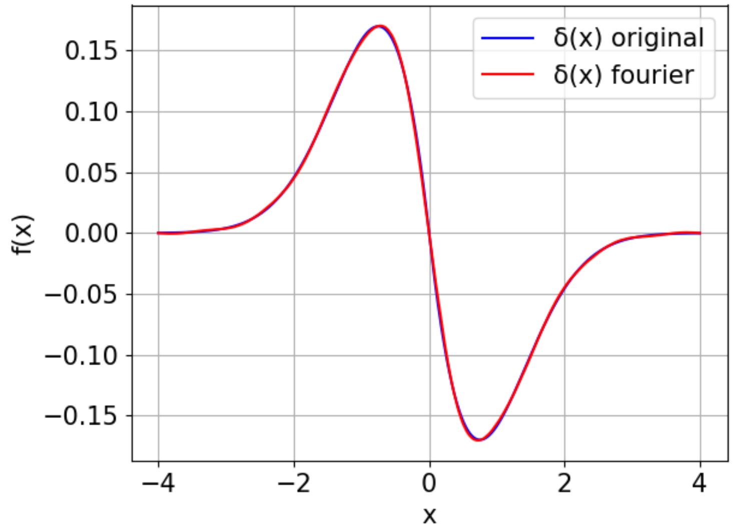
Private GELU with Fourier Series (FS)

- **Fourier Series (FS)**: provide precise approximations of functions
  - with a **sinusoidal** shape for a **bounded** input range
- Securely evaluating an FS takes only **one** round (ACSAC '23)
- Problem: GELU is **not** sinusoidal (*i.e.*, **direct** approximation with FS **fails**)
- Simple solution: approximates $\mathrm{Erf}(x)$ with FS (ACL Findings '24):
  - Recall: $\mathrm{GELU}(x) = 0.5x\left(1 + \mathrm{Erf}\left(x/\sqrt{2}\right)\right)$
  - Requires an **additional** round to **securely multiply** the result by x
  - Increases approximation **error** (when $|x| > 2$)

Y. Zheng, Q. Zhang, **S. Chow**, Y. Peng, S. Tan, L. Li, S. Yin. "Secure softmax/sigmoid for machine-learning computation." *ACSAC* 2023.
J. Luo et al. "SecFormer: Towards fast and accurate privacy-preserving inference for large language models." *ACL (Findings)* 2024.

Our Private GELU with Fourier Series (FS)

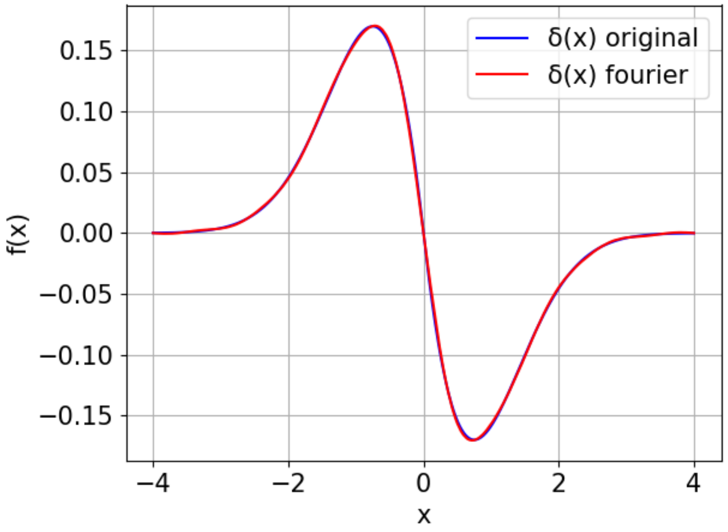
- Goal: **designs a suitable function** for FS approximation
- Our formulation: $\delta(x) = \mathrm{sgn}(x)(\mathrm{GELU}(x) - \mathrm{ReLU}(x))$
  - Modified from **non-sinusoidal** $\mathrm{GELU}(x) - \mathrm{ReLU}(x)$ for table lookup (PETS '24)
  - A **sinusoidal** function for x near 0
  - **Ideal** for accurate FS approximation



K. Gupta et al. "SIGMA: Secure GPT inference with function secret sharing." *PETS* 2024.

Our Private GELU with Fourier Series (FS)

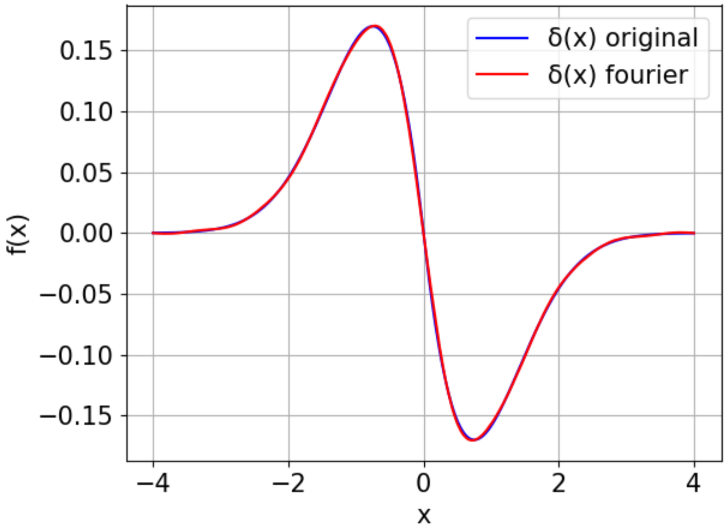
- Goal: **designs a suitable function** for FS approximation
- Our formulation: $\delta(x) = \mathrm{sgn}(x)(\mathrm{GELU}(x) - \mathrm{ReLU}(x))$
  - Modified from **non-sinusoidal** $\mathrm{GELU}(x) - \mathrm{ReLU}(x)$ for table lookup (PETS '24)
  - A **sinusoidal** function for x near 0
  - **Ideal** for accurate FS approximation
- **FS approximation** of $\delta(x)$ for $|x| < 4$:
  - $\delta(x) \approx \sum_{n=1}^{8} \beta_n \sin\left(\frac{n\pi x}{4}\right), \beta_n = \frac{1}{4}\int_{-4}^{4} \delta(u) \sin\left(\frac{n\pi u}{4}\right) du$
  - Coefficients β_n precomputable via numerical integration
- Enforces $\delta(x) \approx 0$ for $|x| \geq 4$



K. Gupta et al. "SIGMA: Secure GPT inference with function secret sharing." *PETS* 2024.

Our Private GELU with Fourier Series (FS)

- Goal: **designs a suitable function** for FS approximation
- Our formulation: $\delta(x) = \mathrm{sgn}(x)(\mathrm{GELU}(x) - \mathrm{ReLU}(x))$
  - Modified from **non-sinusoidal** $\mathrm{GELU}(x) - \mathrm{ReLU}(x)$ for table lookup (PETS '24)
  - A **sinusoidal** function for x near 0
  - **Ideal** for accurate FS approximation
- **FS approximation** of $\delta(x)$ for $|x| < 4$:
  - $\delta(x) \approx \sum_{n=1}^{8} \beta_n \sin\left(\frac{n\pi x}{4}\right), \beta_n = \frac{1}{4}\int_{-4}^{4} \delta(u) \sin\left(\frac{n\pi u}{4}\right) du$
  - Coefficients β_n precomputable via numerical integration
- Enforces $\delta(x) \approx 0$ for $|x| \geq 4$
- GELU characterization: $\mathrm{GELU}(x) = \mathrm{ReLU}(x) + \delta(|x|)$
  - **No extra round** needed: $|x| = 2\mathrm{ReLU}(x) - x$



K. Gupta et al. "SIGMA: Secure GPT inference with function secret sharing." *PETS* 2024.

Other Contributions

1. **First private embedding** protocol **"natively"** taking **indices** as inputs
   - Prior works **assume** inputs are **one-hot vectors**, requiring **extra conversions** by clients
   - Our approach is inspired by **pre-computed one-hot pairs** from Grotto (CCS '23)
   - (unlike Grotto for *spline evaluation*)

2. Extension of our GELU characterization to **other activations**
   - *E.g.*, **sigmoid linear unit/SiLU,** used in the Meta AI's LLaMA model

3. Optimizations for **smaller bitwidth**
   - **Reduces communication** in **mixed-bitwidth** frameworks

K. Storrier et al. "Grotto: Screaming fast $(2 + 1)$-PC for $\mathbb{Z}_{2^n}$ via $(2, 2)$-DPFs." *CCS* 2023.

Final Remarks

- We propose **secure**, **accurate**, and **fast** protocols for **softmax** and **GELU**
- Code: [github.com/andeskyl/SHAFT]
  - Interoperable with **Hugging Face** for **handy transformer** deployment
- Future directions:
  - Private transformer ***fine-tuning/training*** (GPU-TEE co-design?)
  - Security against ***malicious*** adversaries (replicated/authenticated sharing?)
- Contact: {kyl022, sherman}@ie.cuhk.edu.hk

L. Ng, **S. Chow**, A. Woo, D. Wong, Y. Zhao. "Goten: GPU-Outsourcing Trusted Execution of Neural Network Training." *AAAI* 2021.