SecLab
COMPUTER SECURITY LABORATORY

THE OHIO STATE UNIVERSITY

TensorCrypt: Repurposing Neural Networks for Efficient Cryptographic Computation

Xin Jin[1], Shiqing Ma[2], Zhiqiang Lin[1]

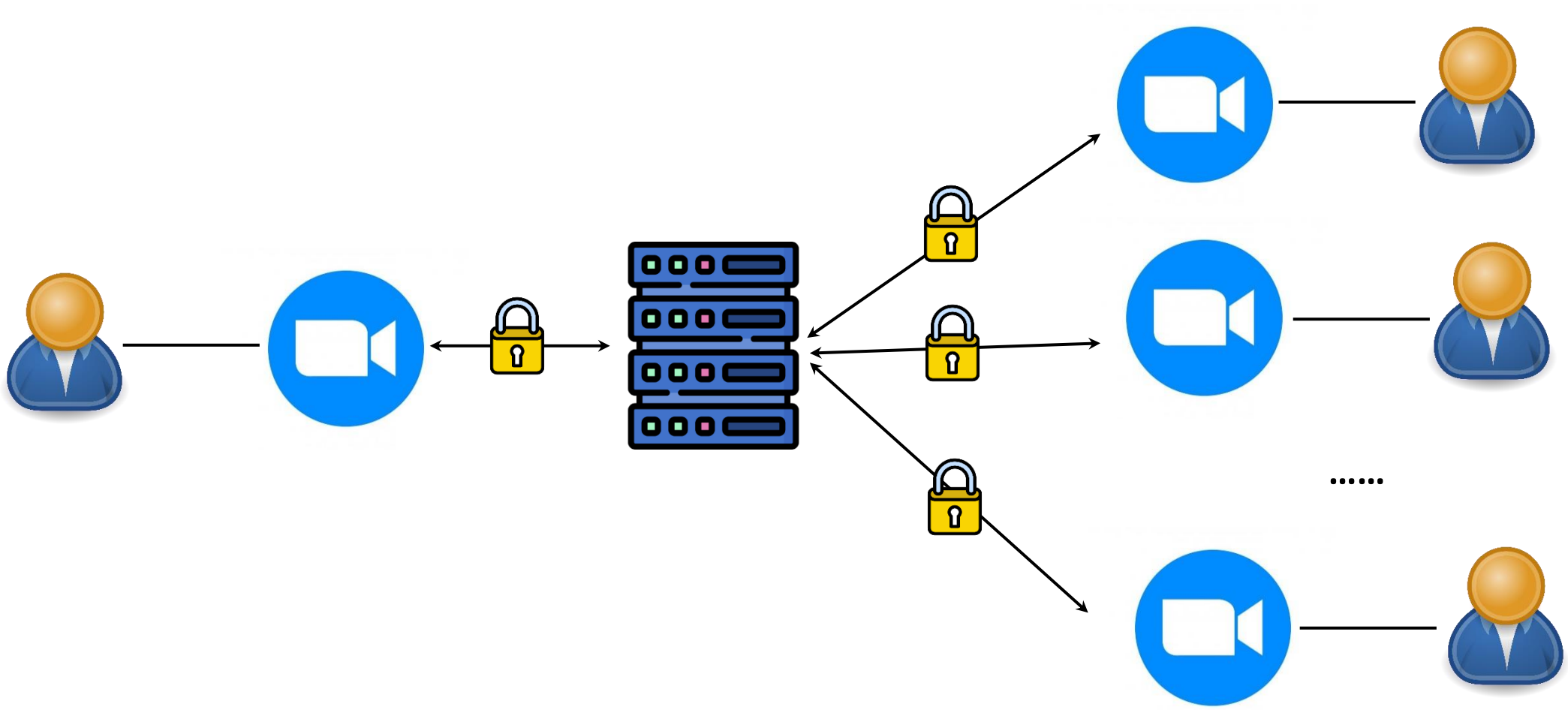
[1]The Ohio State University

[2]University of Massachusetts Amherst

NDSS 2025

University of Massachusetts Amherst

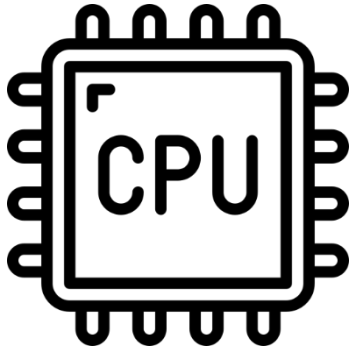
Accelerating cryptographic computation is demanding



Accelerating cryptographic computation is challenging

CPU-based Acceleration, e.g.,

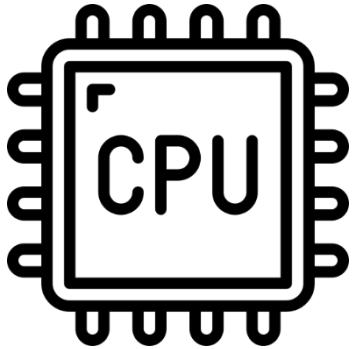
- Intel AES-NI (Gueron, 2010)



Accelerating cryptographic computation is challenging

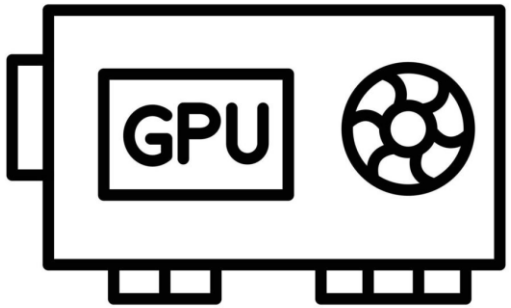
CPU-based Acceleration, e.g.,

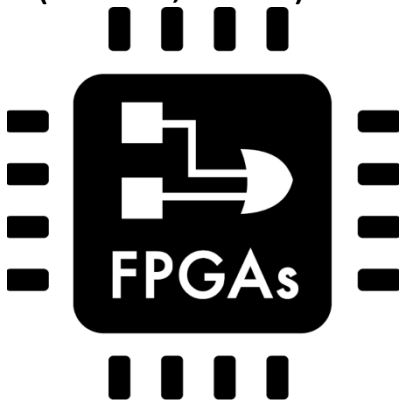
- Intel AES-NI (Gueron, 2010)



Hardware accelerator-based Acceleration, e.g.,

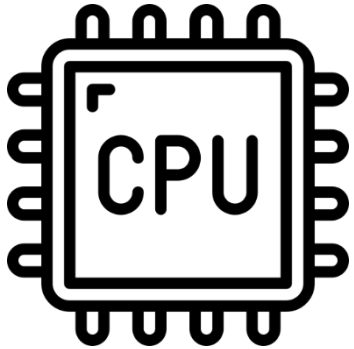
- MemShield (Santucci, 2020)
- FPGA cipher implementation (Kumar, 2015)



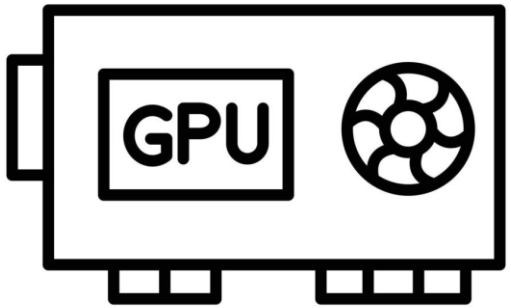


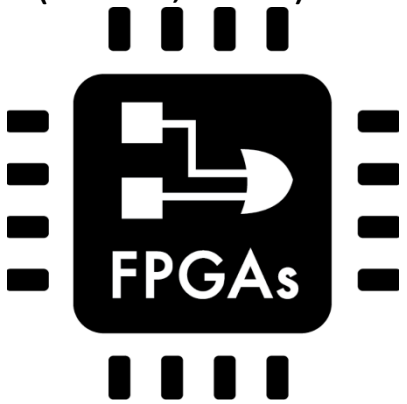
Accelerating cryptographic computation is challenging

CPU-based Acceleration, e.g.,
- Intel AES-NI (Gueron, 2010)



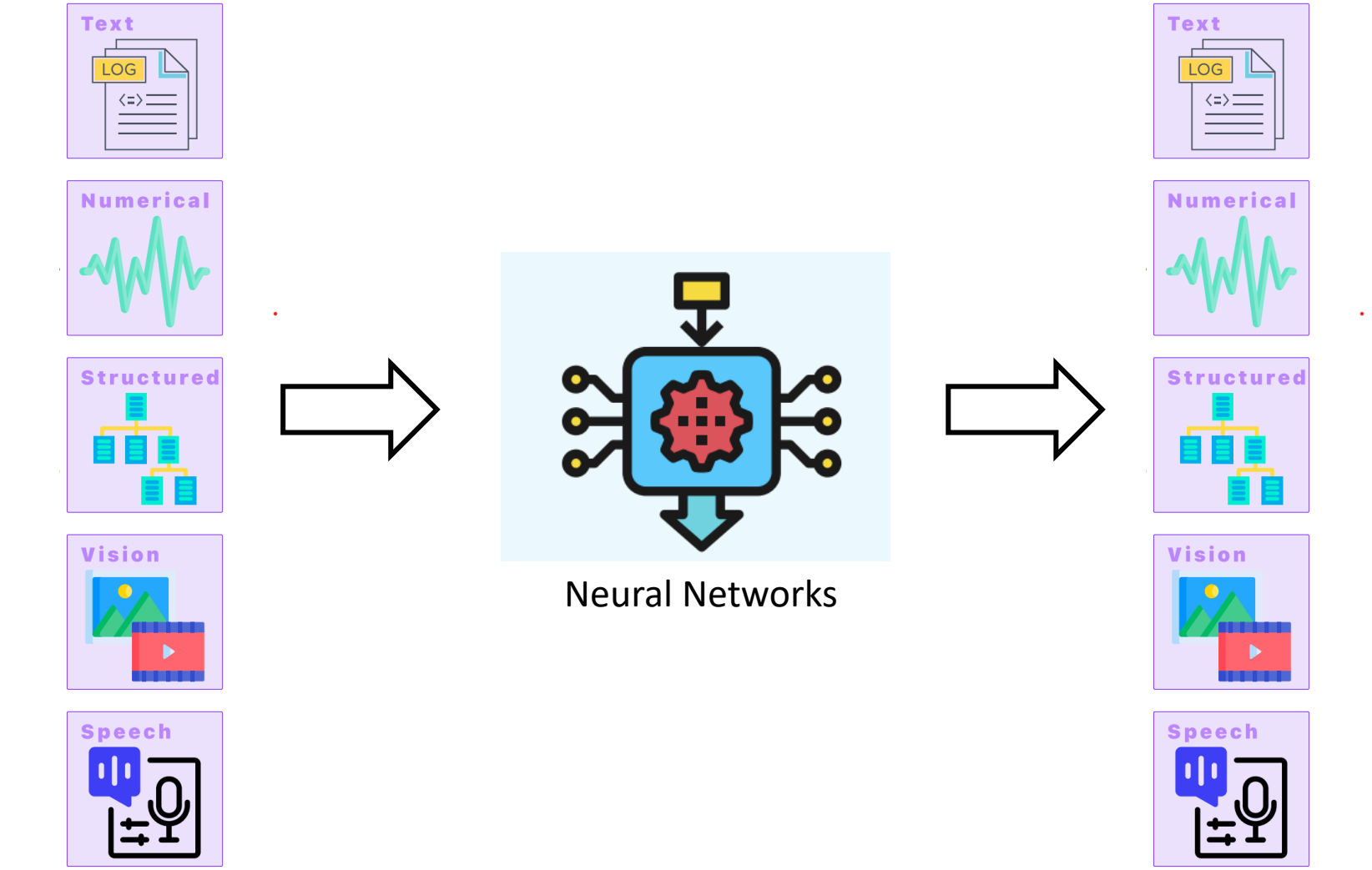
Hardware accelerator-based Acceleration, e.g.,
- MemShield (Santucci, 2020)
- FPGA cipher implementation (Kumar, 2015)





Current acceleration approaches require ***diverse software and hardware sets***, complicating the design, implementation, debugging, and deployment.

Neural networks are promising



Neural networks are supported on diverse software and hardware stacks

Hardware Accelerators

NVIDIA GPU

Google TPU

Neural networks are supported on diverse software and hardware stacks

Hardware Accelerators

NVIDIA GPU

Google TPU

Hardware Platforms

Cloud Server

Desktop

Smartphone

IoT Device

Neural networks are supported on diverse software and hardware stacks

Hardware Accelerators

NVIDIA GPU

Google TPU

Hardware Platforms

Cloud Server

Desktop

Smartphone

IoT Device

AI Frameworks

PyTorch

TensorFlow

Neural networks are supported on diverse software and hardware stacks

Hardware Accelerators

NVIDIA GPU

Google TPU

Hardware Platforms

Cloud Server

Desktop

Smartphone

IoT Device

AI Frameworks

PyTorch

TensorFlow

AI Compilers

tvm

OpenXLA

Neural networks are supported on diverse software and hardware stacks

Hardware Accelerators

NVIDIA GPU

Google TPU

Hardware Platforms

Cloud Server

Desktop

Smartphone

IoT Device

AI Frameworks

PyTorch

TensorFlow

AI Compilers

tvm

OpenXLA

Programming Languages of AI Libraries

python™

Java

Swift

Neural networks are supported on diverse software and hardware stacks

Hardware Accelerators

NVIDIA GPU

Google TPU

Hardware Platforms

Cloud Server

Desktop

Smartphone

IoT Device

AI Frameworks

PyTorch

TensorFlow

AI Compilers

tvm

OpenXLA

Programming Languages of AI Libraries

python™

Java

Swift

Neural networks are ***Turing Complete***. (Siegelmann, 1992)

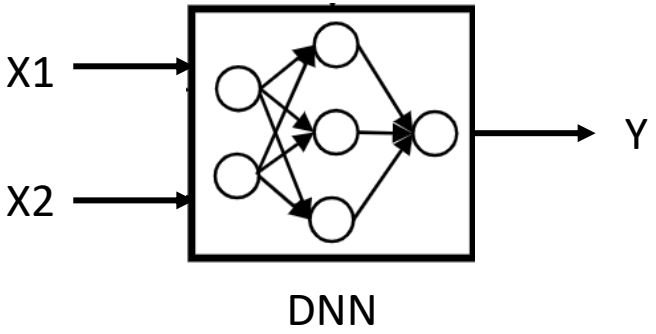
Learning cryptographic computation is extremely challenging

Examples in basic math

Learning cryptographic computation is extremely challenging

Examples in basic math

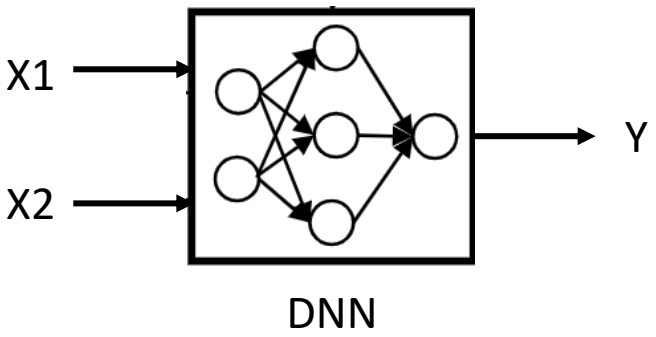
Integer addition within 100:
Y = X1 + X2



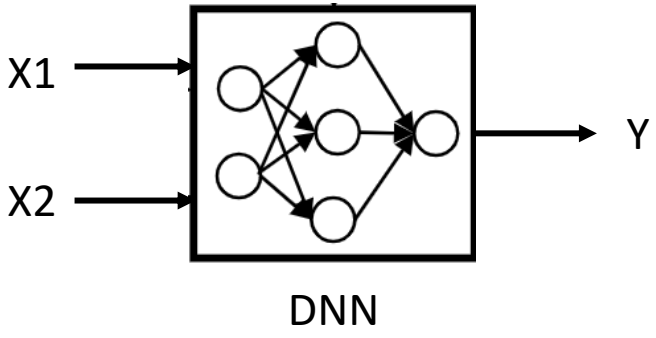
Learning cryptographic computation is extremely challenging

Examples in basic math

Integer addition within 100:
Y = X1 + X2



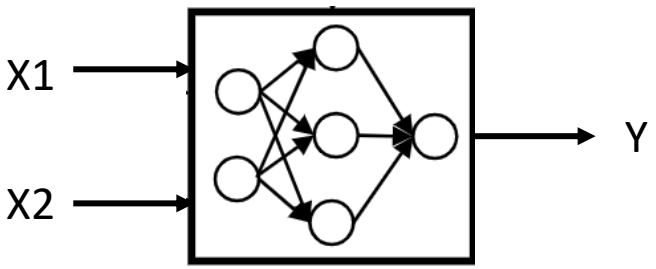
Floating point addition within 100:
Y = X1 + X2



Learning cryptographic computation is extremely challenging

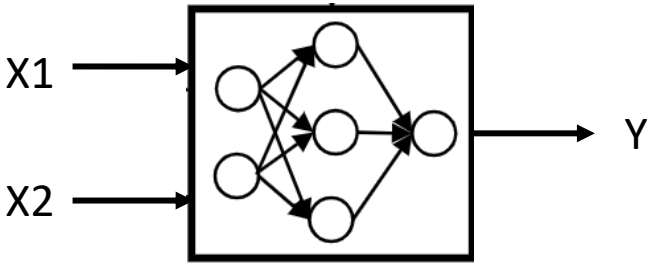
Examples in basic math

Integer addition within 100:
Y = X1 + X2

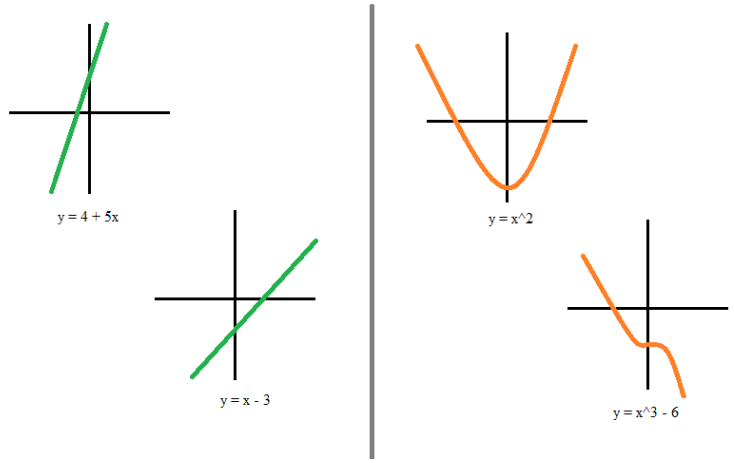


DNN

Floating point addition within 100:
Y = X1 + X2



DNN

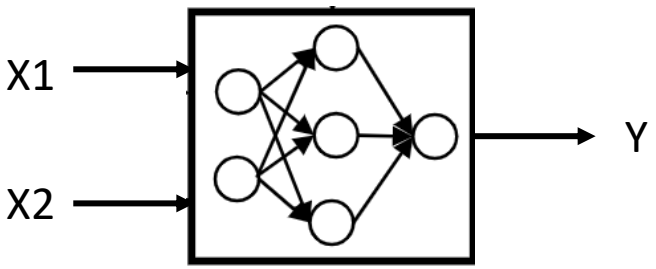


Linear v.s. Non-linear

Learning cryptographic computation is extremely challenging

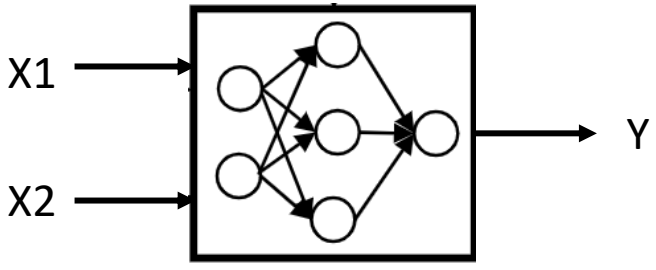
Examples in basic math

Integer addition within 100:
Y = X1 + X2

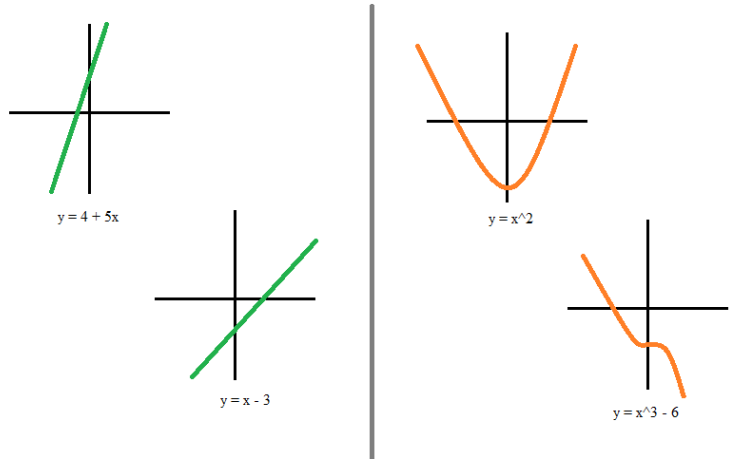


DNN

Floating point addition within 100:
Y = X1 + X2



DNN



Linear v.s. Non-linear

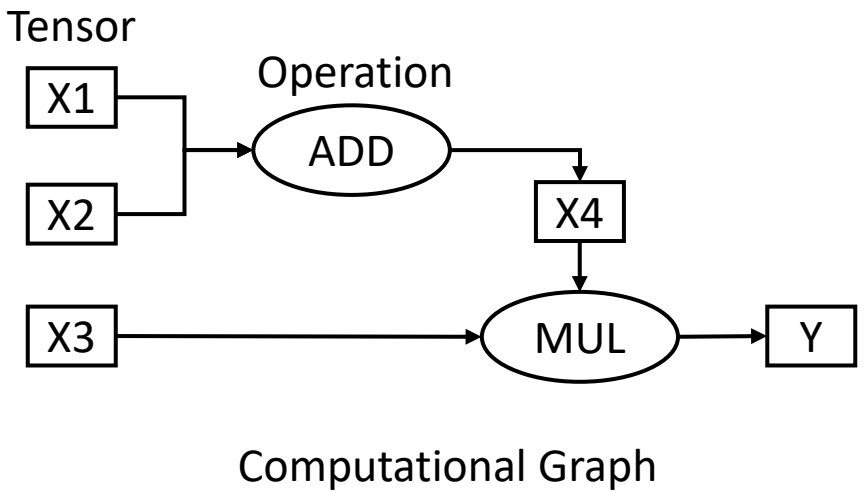
Learning complex computations requires (1) a massive volume of data, (2) NN architecture with high non-linearity, and (3) extensive training efforts.

Accelerating Computation via Computational Graphs

Key insight: neural networks are computational graphs

Accelerating Computation via Computational Graphs

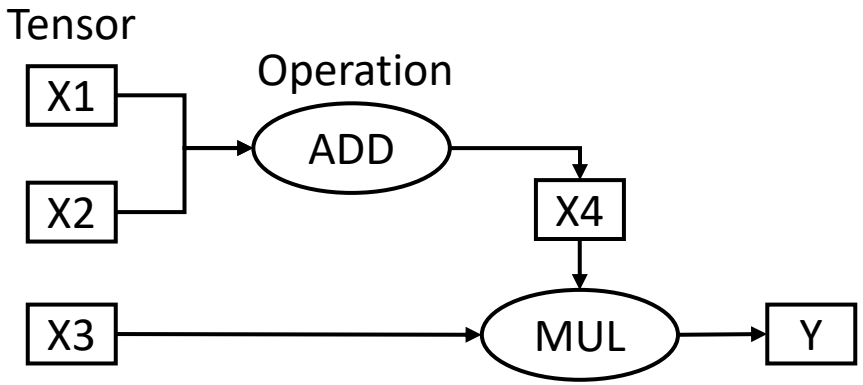
Key insight: neural networks are computational graphs



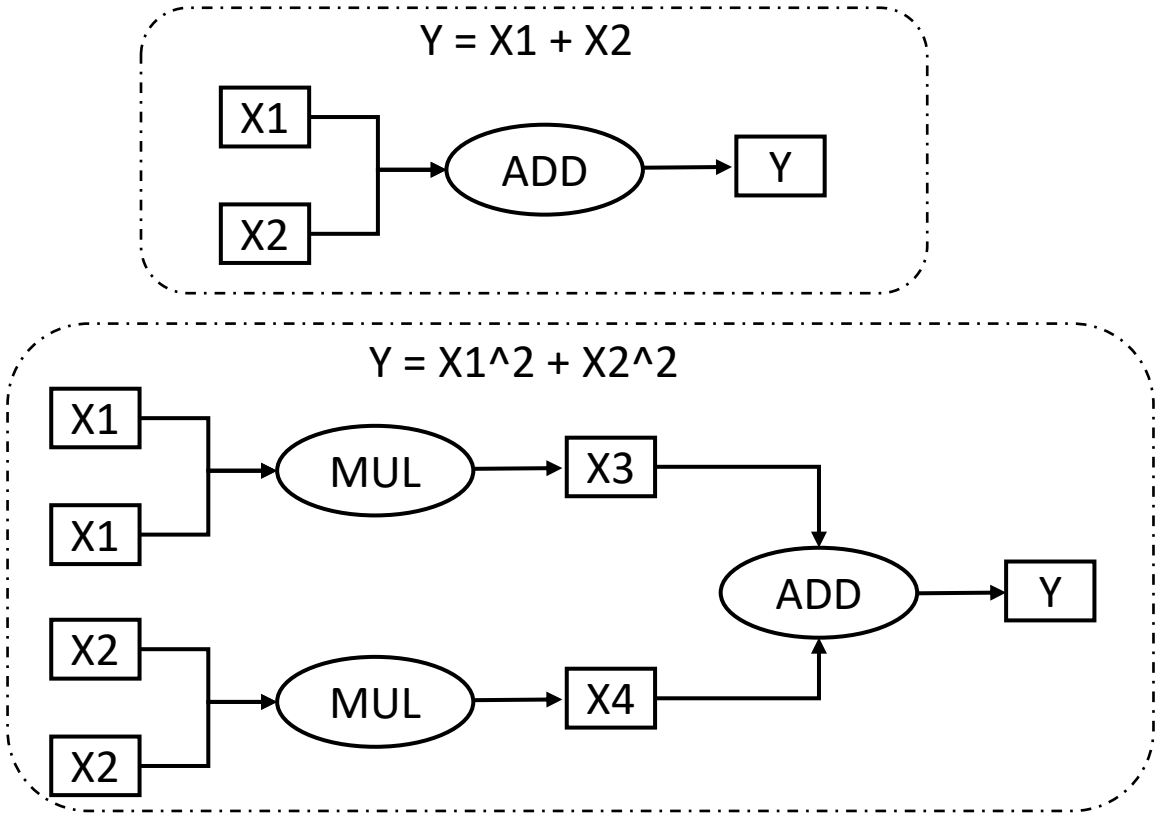
Computational Graph

Accelerating Computation via Computational Graphs

Key insight: neural networks are computational graphs



Computational Graph



Computational Graphs to Accelerate Computations

Domain-specific Language (DSL)

```
⟨program⟩      P ::= input(arg); s*; output(v)
⟨statement⟩    s ::= x := e | x := y ⟨op⟩ z
                 | if (v) : s1* else : s2*
                 | while (x>0) : s*
⟨expr⟩         e ::= v | x | x[v] | x[vi:vj]
⟨operator⟩    op ::= + | − | * | / | XOR | ...
⟨⟩              v ∈ Value
⟨⟩          x,y,z ∈ Identifier;
```

Cryptographic DSL

```
⟨NN⟩           M ::= input(arg); l*; output(v)
⟨layer⟩        l ::= x := add(·) | sub(·) | mul(·)
                 | bitShift(·) | bitAnd(·)
                 | slice(·) | lookup(·)
                 | cond(v, l1*, l2*) | loop(x>0, l*)
⟨layer-add⟩  add(v) ::= (+ v)
⟨layer-loop⟩ loop(x>0, l*) ::= (l[0] ∘ l[1] ∘ ...)
⟨⟩             arg, x, v ∈ Tensor;
```

Neural Network DSL

Semantic Rules

Definitions: $\sigma, \phi \in Store: Variable \rightarrow Value$; σ for cryptography (CRYPT) DSL, ϕ for neural network (NN) DSL.

Evaluation context rules:

$$E_s ::= E_s;s \mid [\cdot]_s \mid x{:=}[\cdot]_e \mid x{:=}[[\cdot]_e] \mid x{:=}[[\cdot]_e : e] \mid x{:=}[v : [\cdot]_e] \mid x{:=}[\cdot]_e \langle op \rangle e \mid x{:=}v \langle op \rangle [\cdot]_e \mid \textbf{if } [\cdot]_e : s_1 \textbf{ else } : s_2 \mid \textbf{while } [\cdot]_e : s^*,$$

$$E_l ::= E_l;l \mid [\cdot]_l \mid x{:=}[\cdot]_e \mid x{:=}[[\cdot]_e] \mid x{:=}[[\cdot]_e{:}e] \; x{:=}[e{:}[\cdot]_e] \mid x{:=}l([\cdot]_e, e) \mid x{:=}l(v, [\cdot]_e)$$

Expression Rule: $\sigma : e \xrightarrow{e} v$ [E-CRYPT] $\quad \phi : e \xrightarrow{e} v$ [E-NN]

$\sigma : v \xrightarrow{e} v$ [E-CONST-CRYPT] $\quad \phi : v \xrightarrow{e} v$ [E-CONST-NN]

$\sigma : x \xrightarrow{e} \sigma(x)$ [E-VAR-CRYPT] $\quad \phi : x \xrightarrow{e} \phi(x)$ [E-VAR-NN]

Statement/Layer Rule: $\sigma, s \xrightarrow{s} \sigma', s'$ [STMT-CRYPT] $\quad \phi, l \xrightarrow{l} \phi', l'$ [STMT-NN]

$\sigma : x := y[v] \xrightarrow{s} \sigma[x \rightarrow \sigma(y[v])],$ skip [LOOKUP-CRYPT] $\quad \phi : x := \textbf{lookup}(y, v) \xrightarrow{l} \phi[x \rightarrow \phi(y[v])],$ skip [LOOKUP-NN]

$\sigma : x := y[v_i : v_j] \xrightarrow{s} \sigma[x \rightarrow [\sigma(y[v_i]), \ldots, \sigma(y[v_j])]],$ skip [SLICE-CRYPT] $\quad \phi : x := \textbf{slice}(y, [v_i : v_j]) \xrightarrow{l} \phi[x \rightarrow [\phi(y[v_i]), \ldots, \phi(y[v_j])]],$ skip [SLICE-NN]

$\sigma : x := y + z \xrightarrow{s} \sigma[x \rightarrow \sigma(y) + \sigma(z)],$ skip [OP-ADD-CRYPT] $\quad \phi : x := \textbf{add}(y, z) \xrightarrow{l} \phi[x \rightarrow \phi(y) + \phi(z)],$ skip [OP-ADD-NN]

$\sigma : \textbf{if}(v) : s_1 : \textbf{else} : s_2 \xrightarrow{s} \sigma, s_1$, if $v = true$ [IF-T-CRYPT] $\quad \phi : \textbf{cond}(v, l_1, l_2) \xrightarrow{l} \phi, l_1$, if $v = true$ [IF-T-NN]

$\sigma : \textbf{while } (x > 0) : s_x^* \xrightarrow{s} s_x^*, s,$ if $x > 0$ [LOOP-E-T-CRYPT] $\quad \phi : \textbf{loop}(x > 0, l_x^*) \xrightarrow{l} l_x^*, s$, if $x > 0$ [LOOP-T-NN]

Global Rules:

$$\frac{\sigma : e \xrightarrow{e} v}{\sigma, E[e]_e \rightarrow \sigma, E[v]_e}$$ [G-EXPR-CRYPT] $\quad \frac{\phi : e \xrightarrow{l} v}{\phi, E[e]_e \rightarrow \phi, E[v]_e}$ [G-EXPR-NN]

$$\frac{\sigma : s \xrightarrow{s} \sigma', s'}{\sigma, E[s]_s \rightarrow \sigma', E[s']_s}$$ [G-STMT-CRYPT] $\quad \frac{\phi : l \xrightarrow{l} \phi', l'}{\phi, E[l]_l \rightarrow \phi', E[l']_l}$ [G-STMT-NN]

Transformation Rules

Statement Transformation: $s \xrightarrow{ctx} l$

$$x := y[v] \xrightarrow{ctx} \boxed{x := \textbf{lookup}(y, v)} \quad \text{[T-LOOKUP]}$$

$$x := y[v_i : v_j] \xrightarrow{ctx} \boxed{x := \textbf{slice}(y, [v_i : v_j])} \quad \text{[T-SLICE]}$$

$$x := y + z \xrightarrow{ctx} \boxed{x := \textbf{add}(y, z)} \quad \text{[T-OP-ADD]}$$

$$x := y \hat{} z \xrightarrow{ctx} \boxed{x := \textbf{bitXOR}(y, z)} \quad \text{[T-OP-XOR]}$$

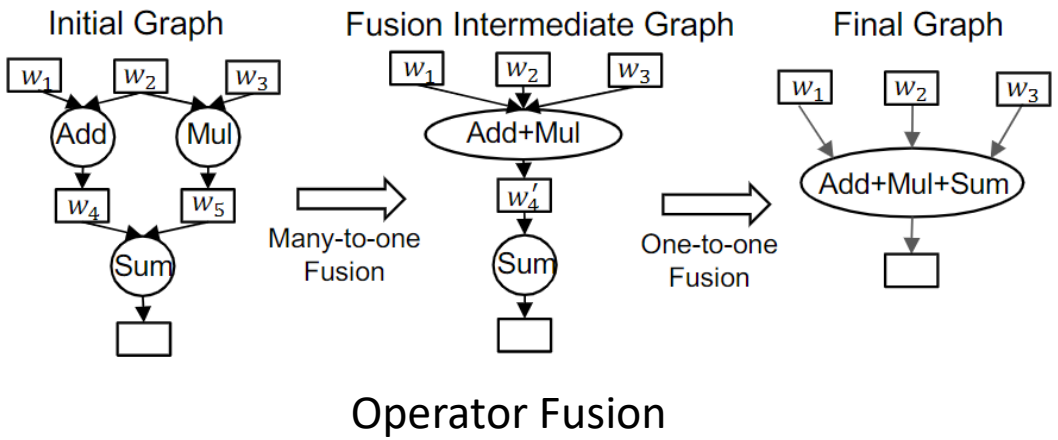
$$\frac{s_1 \xrightarrow{ctx} l_1 \quad s_2 \xrightarrow{ctx} l_2}{\textbf{if}(v) : s_1^* \ \textbf{else} : s_2^* \xrightarrow{ctx} \boxed{\textbf{cond}(v, l_1^*, l_2^*)}} \quad \text{[T-IF]}$$

$$\frac{s_x \xrightarrow{ctx} l_x}{\textbf{while}(x > 0) : s_x^* \xrightarrow{ctx} \boxed{\textbf{loop}(x > 0, l_x^*)}} \quad \text{[T-LOOP]}$$

Program Transformation: $P \xrightarrow{ctx} M$

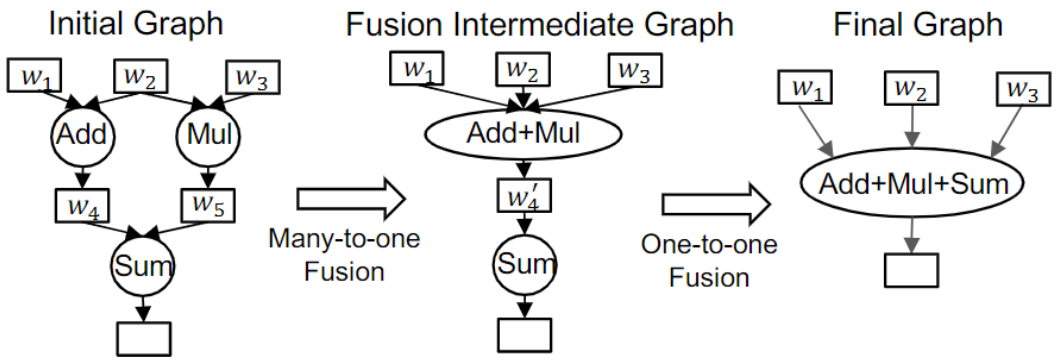
$$\frac{s \xrightarrow{ctx} l}{\textbf{input}(arg);\ s^*;\ \textbf{output}(v) \xrightarrow{ctx} \boxed{\textbf{input}(arg); l^*; \textbf{output}(v)}} \quad \text{[T-INPUT-ASGN]}$$

Model Optimizations

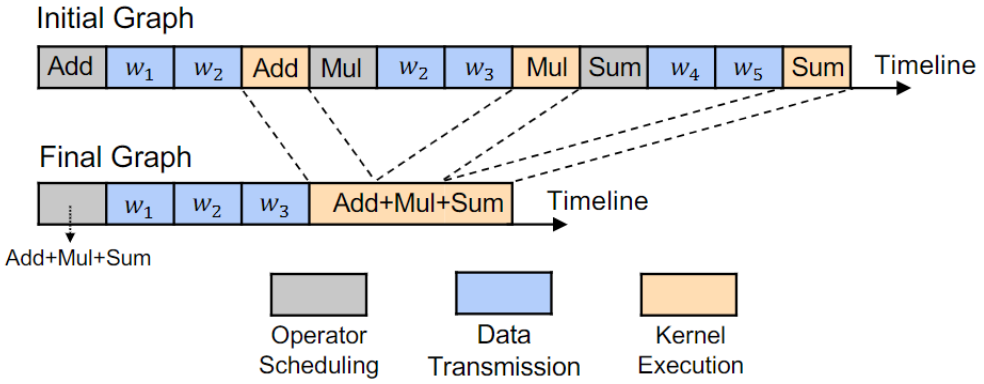


Operator Fusion

Model Optimizations

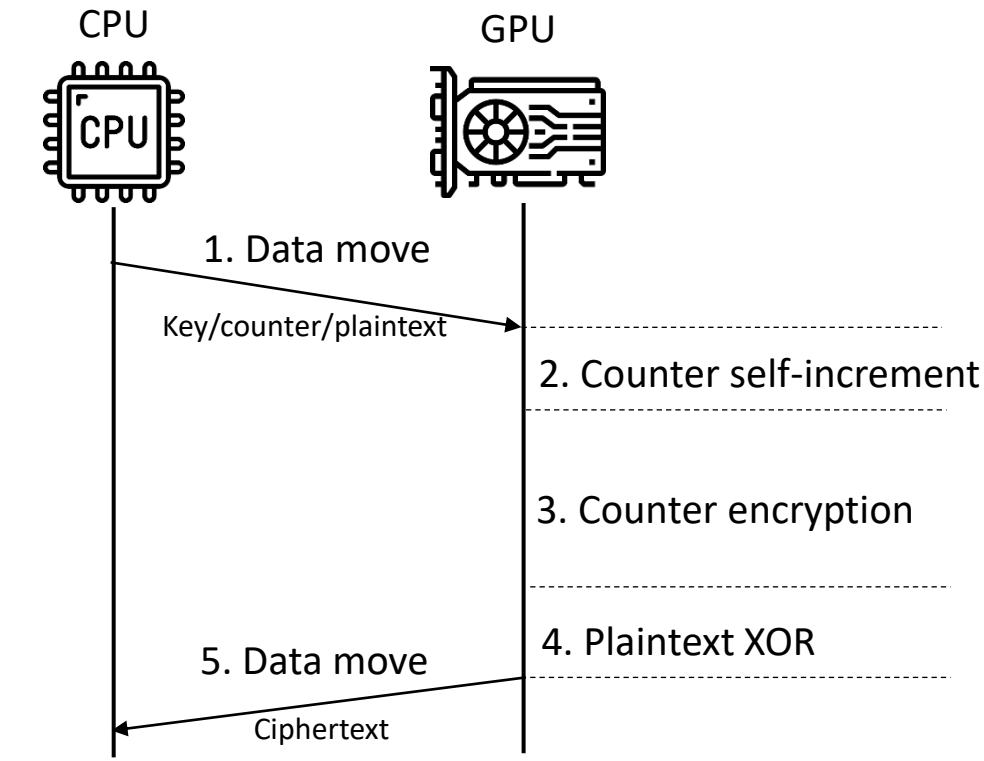


Operator Fusion



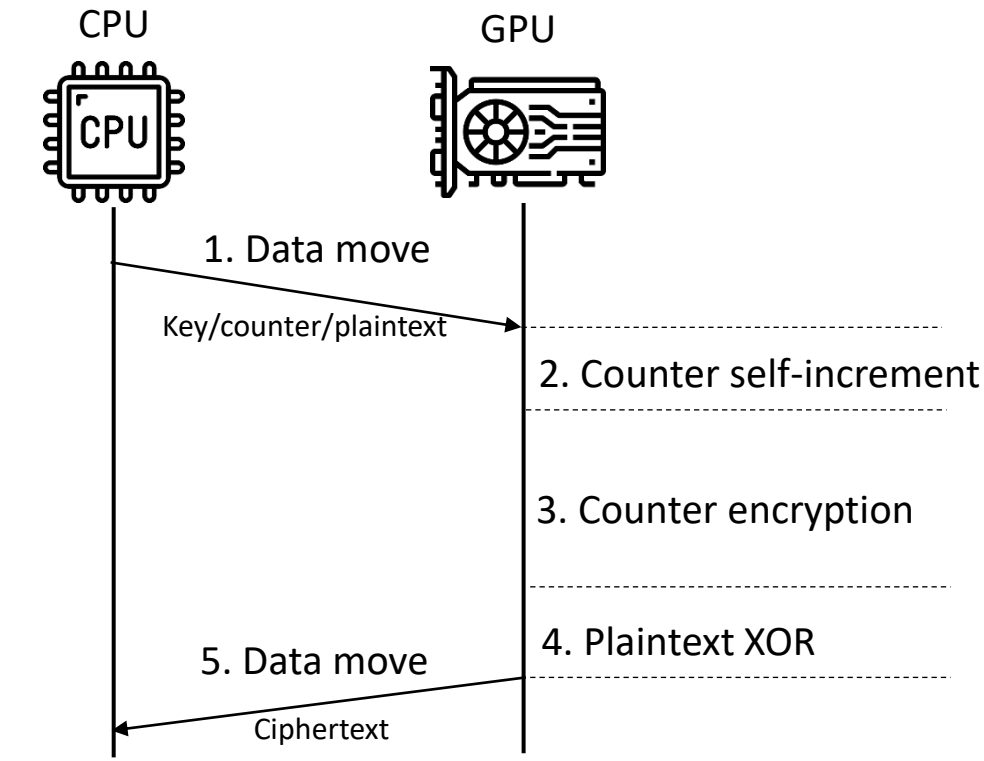
Overhead Analysis

Load-on-use Memory Management

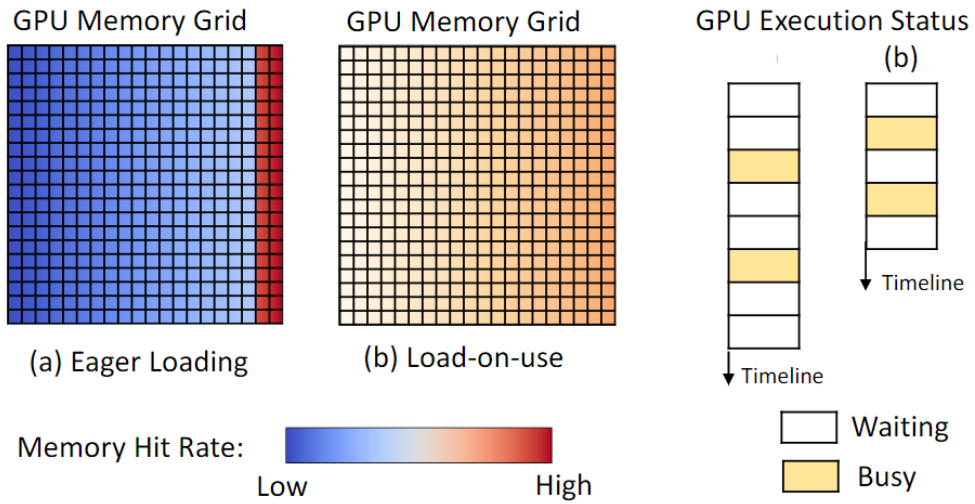


Eager Loading for AES CTR/Chacha/Salsa

Load-on-use Memory Management



Eager Loading for AES CTR/Chacha/Salsa



GPU Memory Usage Comparison

Evaluation Setup

Evaluation target ciphers

- AES: 5 modes, ECB/**CTR**/CBC/CFB/OFB
- Chacha (Chacha20 and AES are the only 2 ciphers for encrypting large volumes of data in TLS 1.3)
- Salsa: variant of Chacha

Evaluation Setup

Evaluation target ciphers

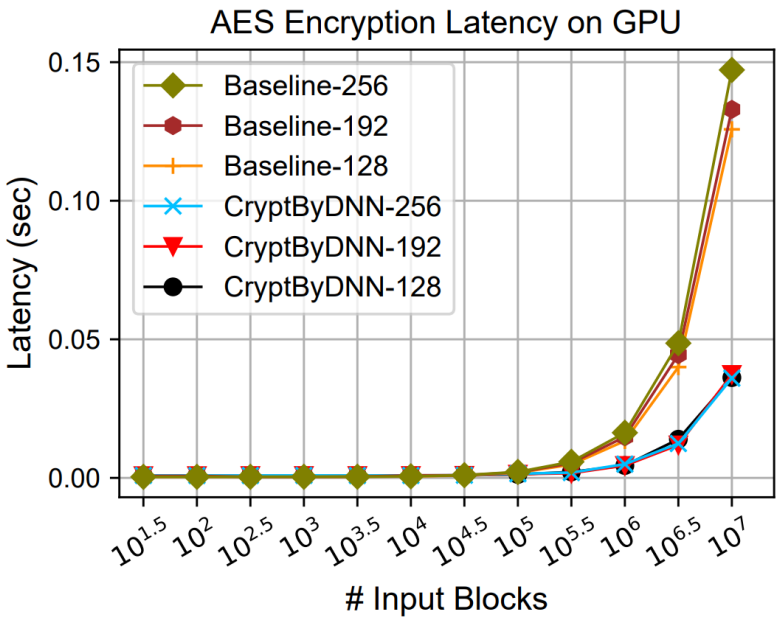
- AES: 5 modes, ECB/**CTR**/CBC/CFB/OFB
- Chacha (Chacha20 and AES are the only 2 ciphers for encrypting large volumes of data in TLS 1.3)
- Salsa: variant of Chacha

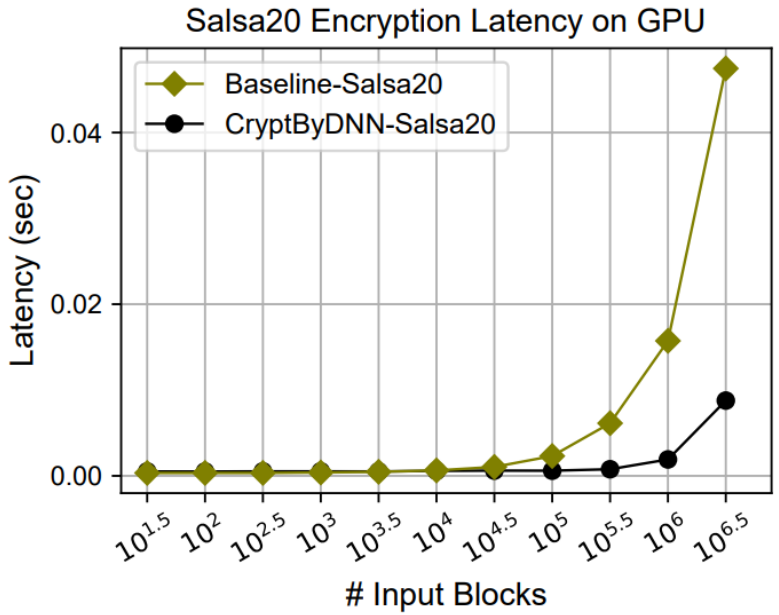
Baselines: GPU-based Cipher Implementations

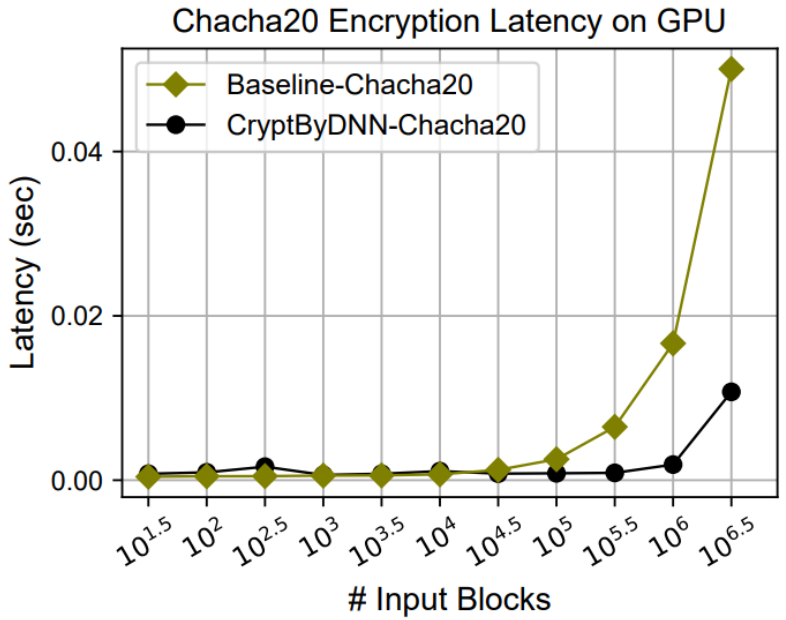
- AES (Tezcan, 2020)
- Chacha20 and Salsa20 (Santucci, 2020)

Effectiveness

TensorCrypt models are up to 5.44 times faster than baselines.

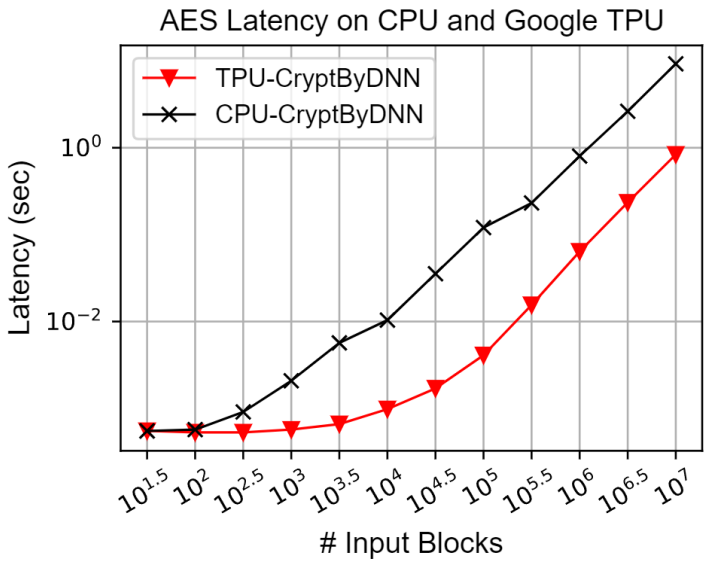


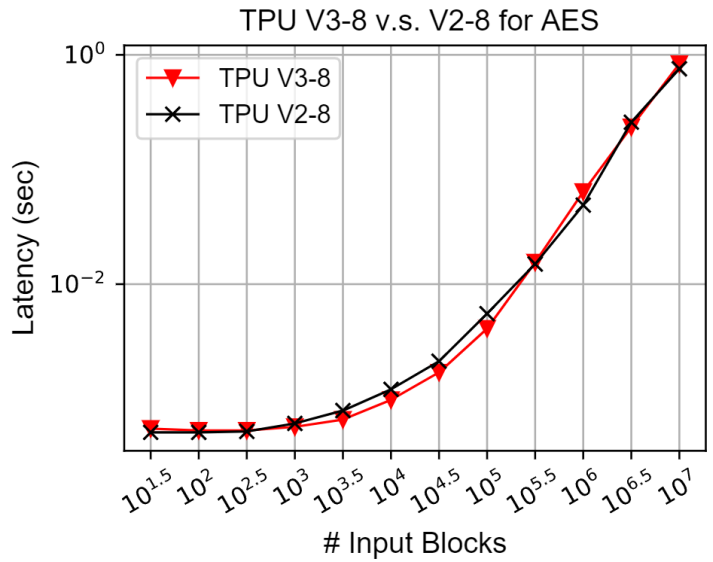




Deployment on Diverse Software and Hardware Stacks

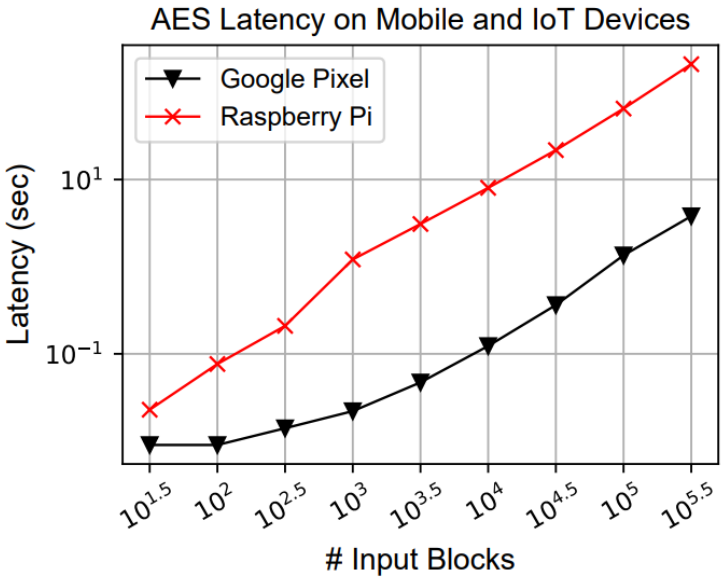
TensorCrypt models on Google TPUs

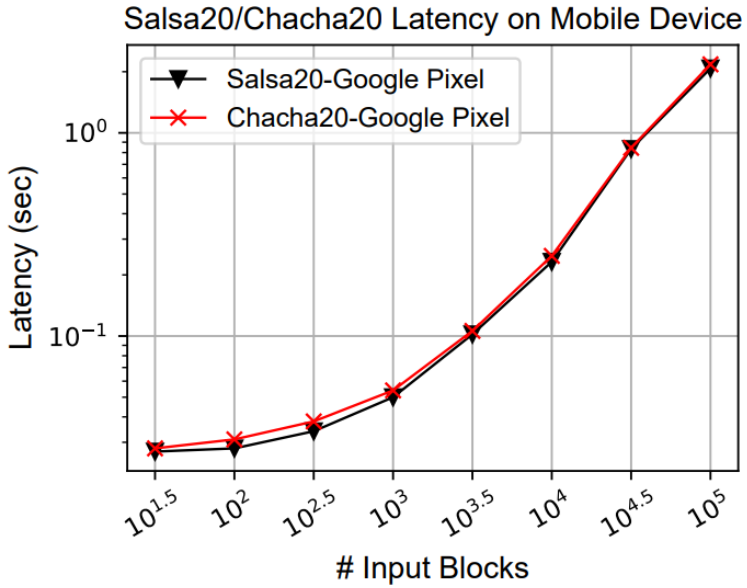




Deployment on Diverse Software and Hardware Stacks

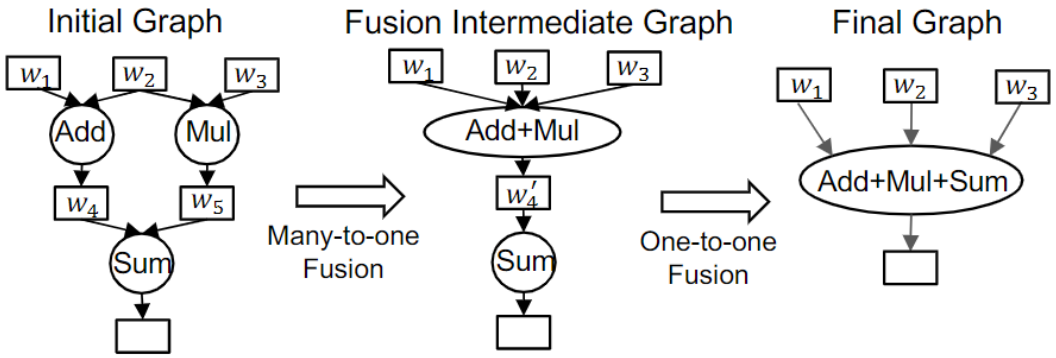
TensorCrypt models on Google Pixel (Smartphone) and Raspberry Pi (IoT)

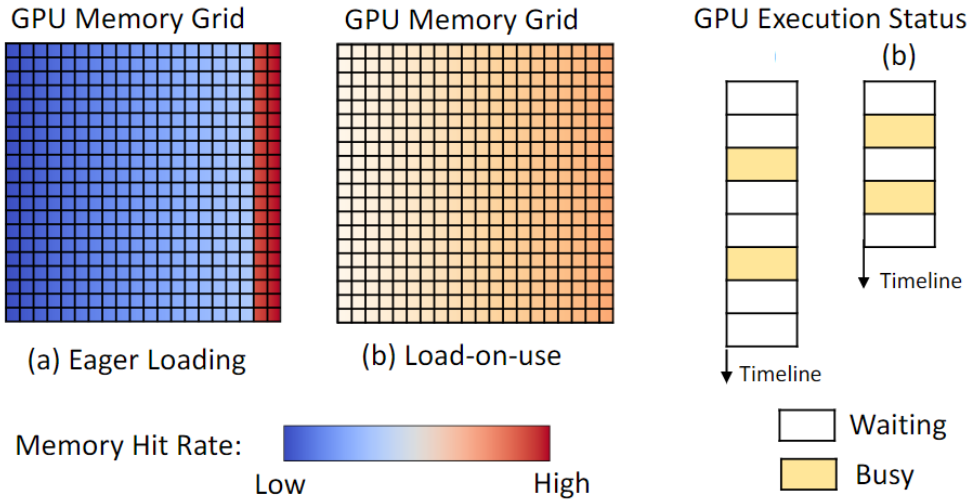




TensorCrypt

- Repurposing neural networks to efficient cryptographic computations
- Program transformation with DSLs and model optimizations
- Advanced encryption and decryption speed with deployment on diverse software and hardware stacks.





References

- Gueron, Shay. "Intel advanced encryption standard (AES) new instructions set." Intel Corporation 128 (2010).
- Santucci, Pierpaolo, et al. "MemShield: GPU-assisted software memory encryption." International Conference on Applied Cryptography and Network Security. Cham: Springer International Publishing, 2020.
- Kumar, Thanikodi Manoj, et al. "A low area high speed FPGA implementation of AES architecture for cryptography application." *Electronics* 10.16 (2021): 2023.
- Siegelmann, Hava T., and Eduardo D. Sontag. "On the computational power of neural nets." Proceedings of the fifth annual workshop on Computational learning theory. 1992.
- Tezcan, Cihangir. "Optimization of advanced encryption standard on graphics processing units." IEEE Access 9 (2021): 67315-67326.

SecLab

COMPUTER SECURITY LABORATORY

THE OHIO STATE UNIVERSITY

TensorCrypt: Repurposing Neural Networks for Efficient Cryptographic Computation

University of Massachusetts Amherst