2024. 05. 14. Paper Presentation Talk

Optimizing Homomorphic Evaluation Circuits by Program Synthesis and Term Rewriting

Presented by Geon Park

KAIST

Programming Systems Laboratory

2024. 05. 14. Paper Presentation Talk

Optimizing Homomorphic Evaluation Circuits by Program Synthesis and Term Rewriting

Presented by Geon Park

Original paper by DongKwon Lee, Woosuk Lee, Hakjoo Oh, and Kwangkeun Yi

KAIST
Programming Systems Laboratory

Motivation

- Homomorphic Encryption, by example

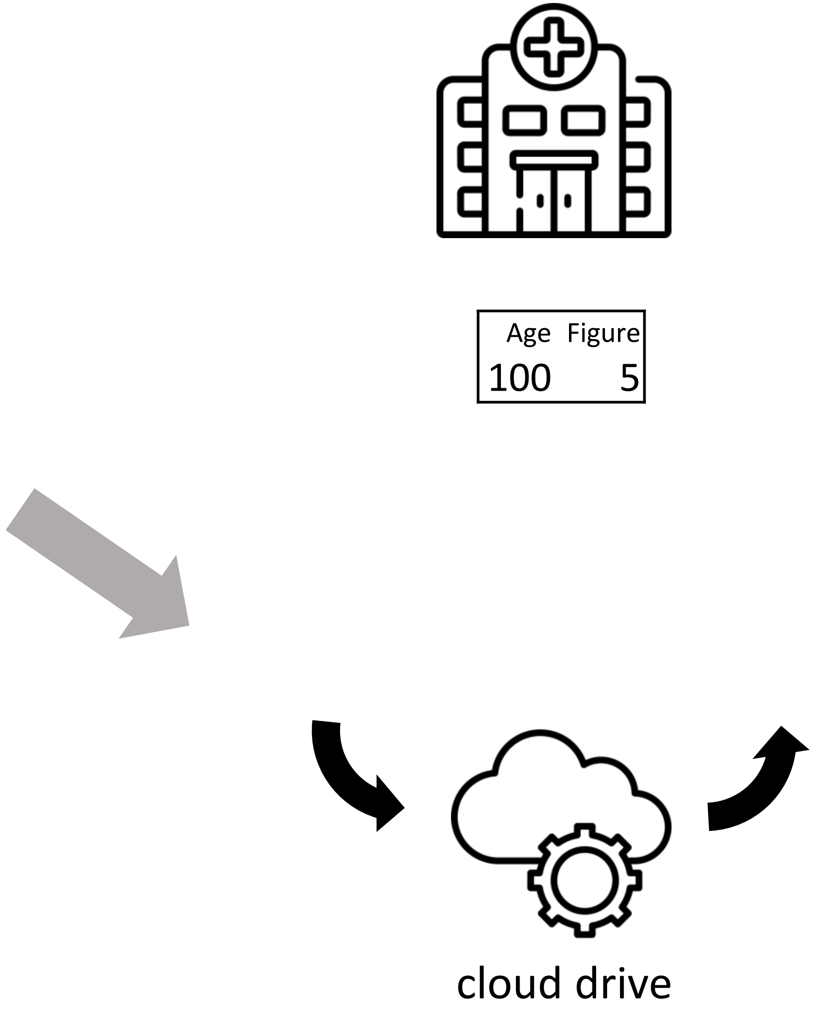
Motivation

- Homomorphic Encryption, by example

Age	Figure
100	5

Motivation

- Homomorphic Encryption, by example



Motivation

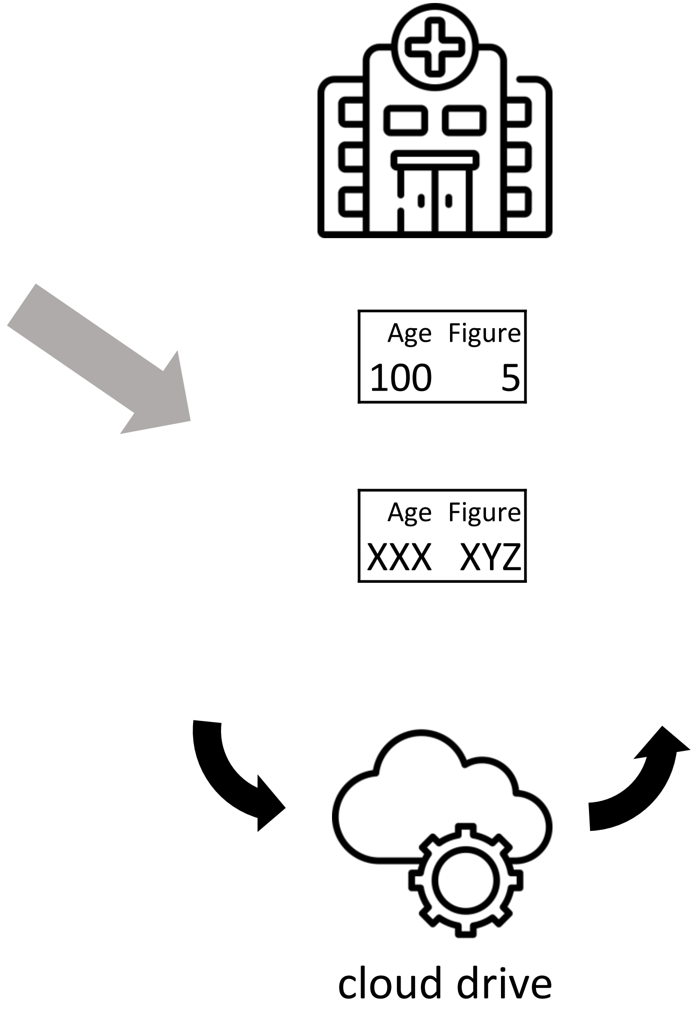
- Homomorphic Encryption, by example
  - Data should not be leaked online

Age	Figure
100	5

cloud drive

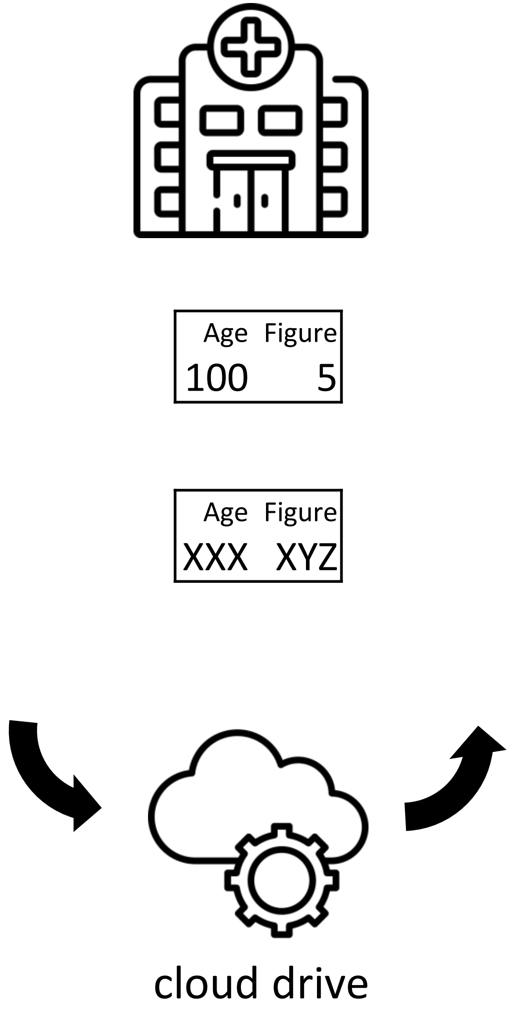
Motivation

- Homomorphic Encryption, by example
  - Data should not be leaked online



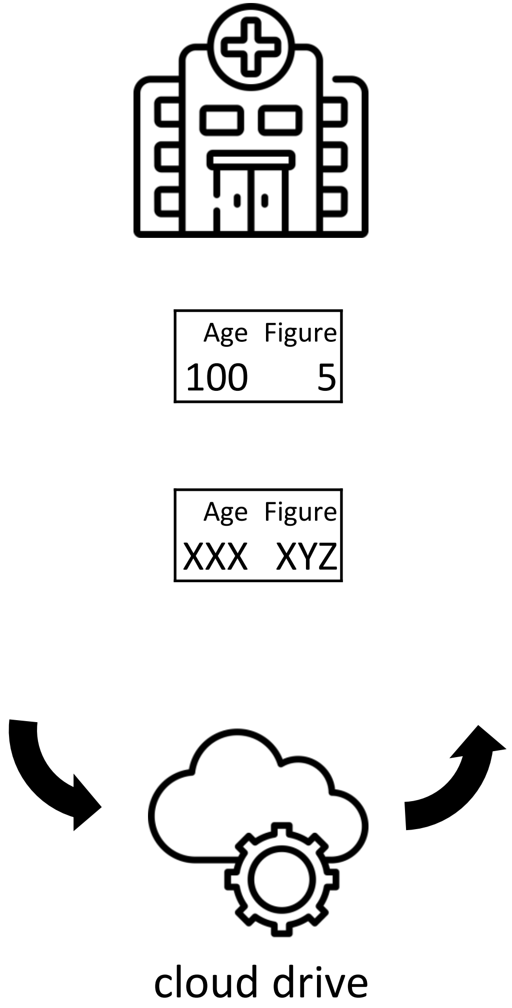
Motivation

- Homomorphic Encryption, by example
  - Data should not be leaked online
  - Want to execute program on a cloud drive



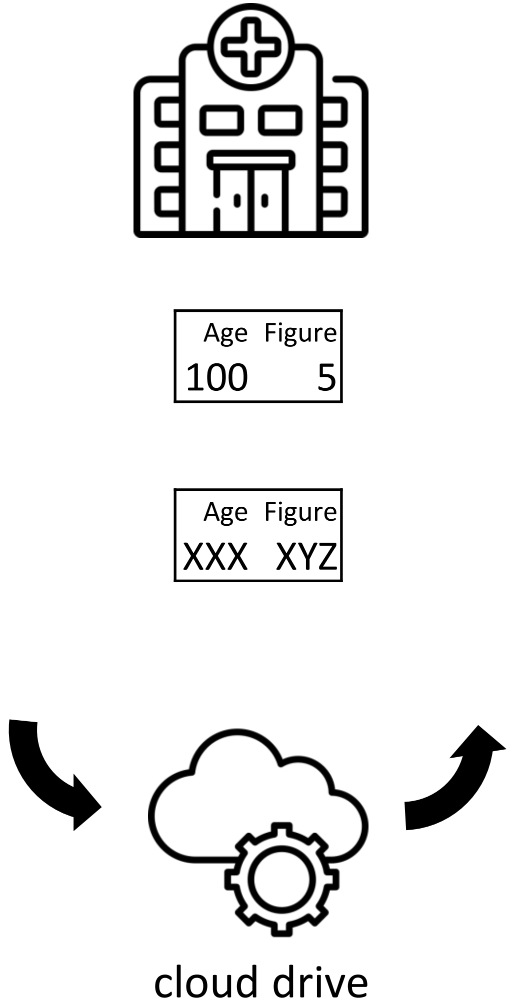
Motivation

- Homomorphic Encryption, by example
  - Data should not be leaked online
  - Want to execute program on a cloud drive
  - Q: Can we do operation on encrypted data?



Motivation

- Homomorphic Encryption, by example
  - Data should not be leaked online
  - Want to execute program on a cloud drive
  - Q: Can we do operation on encrypted data?
  - A: Yes, for RSA encryption scheme



Motivation

- Homomorphic Encryption, by example
    - Data should not be leaked online
    - Want to execute program on a cloud drive
    - Q: Can we do operation on encrypted data?
    - A: Yes, for RSA encryption scheme
        - Decrypting operation (×) result gives same result as done for plaintext

Age	Figure
100	5

Age	Figure
XXX	XYZ

cloud drive

*Kihong Heo, “CS348 Information Security – Lecture 7”, 2022

Motivation

- Homomorphic Encryption, by example
  - Data should not be leaked online
  - Want to execute program on a cloud drive
  - Q: Can we do operation on encrypted data?
  - A: Yes, for RSA encryption scheme
    - Decrypting operation (×) result gives same result as done for plaintext
    - Cloud can't decrypt personal data(100) from encrypted data(100^7 % 707)

Age	Figure
100	5

Age	Figure
XXX	XYZ

Age
100
Figure
5

(707, 7)

↑

lock selected by RSA scheme

Age(Enc.)
100^7 % 707
Figure(Enc.)
5^7 % 707

cloud drive

*Kihong Heo, "CS348 Information Security – Lecture 7", 2022

Motivation

- Homomorphic Encryption, by example
  - Data should not be leaked online
  - Want to execute program on a cloud drive
  - Q: Can we do operation on encrypted data?
  - A: Yes, for RSA encryption scheme
    - Decrypting operation (×) result gives same result as done for plaintext
    - Cloud can't decrypt personal data(100) from encrypted data(100^7 % 707)

Age	Figure
100	5

Age	Figure
XXX	XYZ

Age
100
Figure
5

(707, 7)

↑

lock selected by RSA scheme

Age(Enc.)
100^7 % 707
Figure(Enc.)
5^7 % 707

(707, 343)

↑

open key selected by RSA scheme

cloud drive

*Kihong Heo, "CS348 Information Security – Lecture 7", 2022

Motivation

- Homomorphic Encryption, by example
  - Data should not be leaked online
  - Want to execute program on a cloud drive
  - Q: Can we do operation on encrypted data?
  - A: Yes, for RSA encryption scheme
    - Decrypting operation (×) result gives same result as done for plaintext
    - Cloud can't decrypt personal data(100) from encrypted data($100^7 \% 707$)

Age	Figure
100	5

Age	Figure
XXX	XYZ

Age
100
Figure
5

(707, 7)

↑

lock selected by RSA scheme

Age(Enc.)
$100^7 \% 707$
Figure(Enc.)
$5^7 \% 707$

(707, 343)

↑

open key selected by RSA scheme

$(\text{Age}_{\text{(Enc.)}} \times \text{Figure}_{\text{(Enc.)}})^{343} \% 707$

$= 100 \times 5$

$= \text{Age} \times \text{Figure}$

(Works because $100 \times 5 < 707$)

cloud drive

*Kihong Heo, "CS348 Information Security – Lecture 7", 2022

Fully Homomorphic Encryption

- Can do + and ×

Fully Homomorphic Encryption

- Can do + and ×
- With + and ×, can do much of all computation

Fully Homomorphic Encryption

- Can do + and ×
- With + and ×, can do much of all computation
- First discovered in 2009*
  - Use ideals in algebraic number fields

*C. Gentry, "Fully homomorphic encryption using ideal lattices", STOC, 2009

Fully Homomorphic Encryption

- Can do + and ×
- With + and ×, can do much of all computation
- First discovered in 2009*
  - Use ideals in algebraic number fields
- Emerging **AI privacy problem** needs it

*C. Gentry, "Fully homomorphic encryption using ideal lattices", STOC, 2009

Fully Homomorphic Encryption

- Can do + and ×
- With + and ×, can do much of all computation
- First discovered in 2009*
  - Use ideals in algebraic number fields
- Emerging **AI privacy problem** needs it
  - AI runs online, and it sees your data

*C. Gentry, "Fully homomorphic encryption using ideal lattices", STOC, 2009

Fully Homomorphic Encryption

- Can do + and ×
- With + and ×, can do much of all computation
- First discovered in 2009*
  - Use ideals in algebraic number fields
- Emerging **AI privacy problem** needs it
  - AI runs online, and it sees your data

Apr 25, 2023

*C. Gentry, “Fully homomorphic encryption using ideal lattices”, STOC, 2009

Fully Homomorphic Encryption

- Can do + and ×
- With + and ×, can do much of all computation
- First discovered in 2009*
  - Use ideals in algebraic number fields
- Emerging **AI privacy problem** needs it
  - AI runs online, and it sees your data

AI and Privacy: The privacy concerns surrounding AI, its potential impact on personal data

Last Updated: Apr 25, 2023, 08:31:00 PM IST

FOLLOW US SHARE FONT SIZE SAVE PRI

Apr 25, 2023

*C. Gentry, "Fully homomorphic encryption using ideal lattices", STOC, 2009

*https://economictimes.indiatimes.com/news/how-to/ai-and-privacy-the-privacy-concerns-surrounding-ai-its-potential-impact-on-personal-data/articleshow/99738234.cms?from=mdr

Fully Homomorphic Encryption

- Can do + and ×
- With + and ×, can do much of all computation
- First discovered in 2009*
  - Use ideals in algebraic number fields
- Emerging **AI privacy problem** needs it
  - AI runs online, and it sees your data

AI and Privacy: The privacy concerns surrounding AI, its potential impact on personal data

Last Updated: Apr 25, 2023, 08:31:00 PM IST

FOLLOW US SHARE FONT SIZE SAVE

Apr 25, 2023

2023-11-28

*C. Gentry, "Fully homomorphic encryption using ideal lattices", STOC, 2009

*https://economictimes.indiatimes.com/news/how-to/ai-and-privacy-the-privacy-concerns-surrounding-ai-its-potential-impact-on-personal-data/articleshow/99738234.cms?from=mdr

Fully Homomorphic Encryption

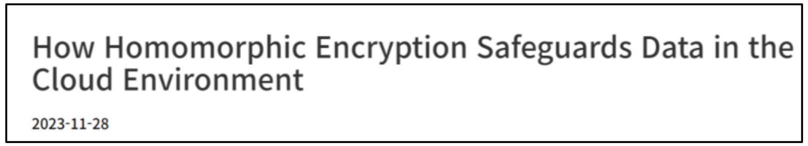
- Can do + and ×
- With + and ×, can do much of all computation
- First discovered in 2009*
    - Use ideals in algebraic number fields
- Emerging **AI privacy problem** needs it
    - AI runs online, and it sees your data


AI and Privacy: The privacy concerns surrounding AI, its potential impact on personal data

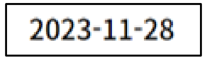
Last Updated: Apr 25, 2023, 08:31:00 PM IST

FOLLOW US SHARE FONT SIZE SAVE


Apr 25, 2023


How Homomorphic Encryption Safeguards Data in the Cloud Environment

2023-11-28


2023-11-28

*C. Gentry, “Fully homomorphic encryption using ideal lattices”, STOC, 2009

*https://economictimes.indiatimes.com/news/how-to/ai-and-privacy-the-privacy-concerns-surrounding-ai-its-potential-impact-on-personal-data/articleshow/99738234.cms?from=mdr

*https://ahha.ai/2023/11/28/en-homomorphic/

Fully Homomorphic Encryption

- Can do + and ×
- With + and ×, can do much of all computation
- First discovered in 2009*
  - Use ideals in algebraic number fields
- Emerging **AI privacy problem** needs it
  - AI runs online, and it sees your data

AI and Privacy: The privacy concerns surrounding AI, its potential impact on personal data

Last Updated: Apr 25, 2023, 08:31:00 PM IST

Apr 25, 2023

How Homomorphic Encryption Safeguards Data in the Cloud Environment

2023-11-28

2023-11-28

3 days ago

*C. Gentry, "Fully homomorphic encryption using ideal lattices", STOC, 2009

*https://economictimes.indiatimes.com/news/how-to/ai-and-privacy-the-privacy-concerns-surrounding-ai-its-potential-impact-on-personal-data/articleshow/99738234.cms?from=mdr

*https://ahha.ai/2023/11/28/en-homomorphic/

Fully Homomorphic Encryption

- Can do + and ×
- With + and ×, can do much of all computation
- First discovered in 2009*
  - Use ideals in algebraic number fields
- Emerging **AI privacy problem** needs it
  - AI runs online, and it sees your data

AI and Privacy: The privacy concerns surrounding AI, its potential impact on personal data

Last Updated: Apr 25, 2023, 08:31:00 PM IST

Apr 25, 2023

How Homomorphic Encryption Safeguards Data in the Cloud Environment

2023-11-28

2023-11-28

Cyber Security

Can fully homomorphic encryption solve AI's privacy problem?

3 days ago

*C. Gentry, "Fully homomorphic encryption using ideal lattices", STOC, 2009

*https://economictimes.indiatimes.com/news/how-to/ai-and-privacy-the-privacy-concerns-surrounding-ai-its-potential-impact-on-personal-data/articleshow/99738234.cms?from=mdr

*https://ahha.ai/2023/11/28/en-homomorphic/

*https://aijourn.com/can-fully-homomorphic-encryption-solve-ais-privacy-problem/

Need of general optimizing technique

- The computational model, called **circuits**, are of great interest

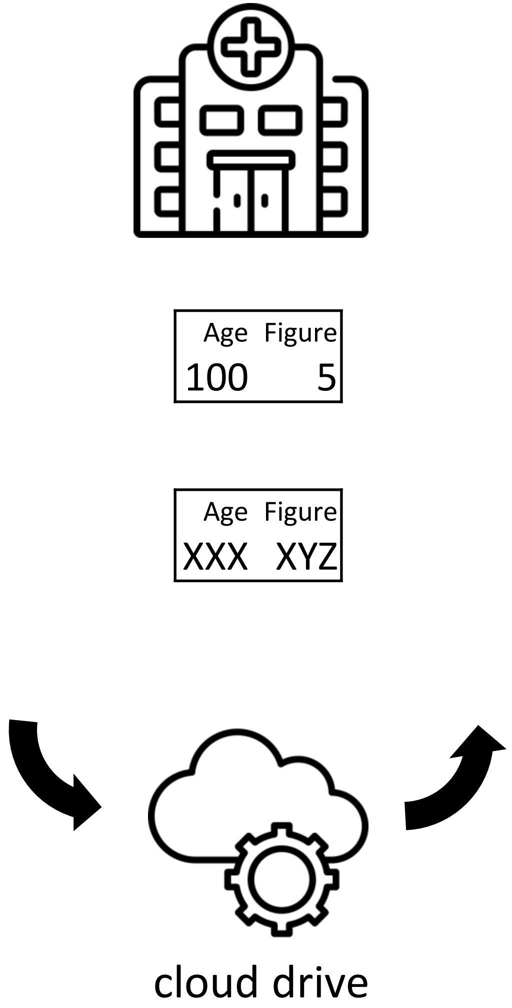
Age	Figure
100	5

Age	Figure
XXX	XYZ

cloud drive

Need of general optimizing technique

- The computational model, called **circuits**, are of great interest
- Problem : FHE require lots of **calculation cost**



Need of general optimizing technique

- The computational model, called **circuits**, are of great interest
- Problem : FHE require lots of **calculation cost**
  - Will be shown in killer example

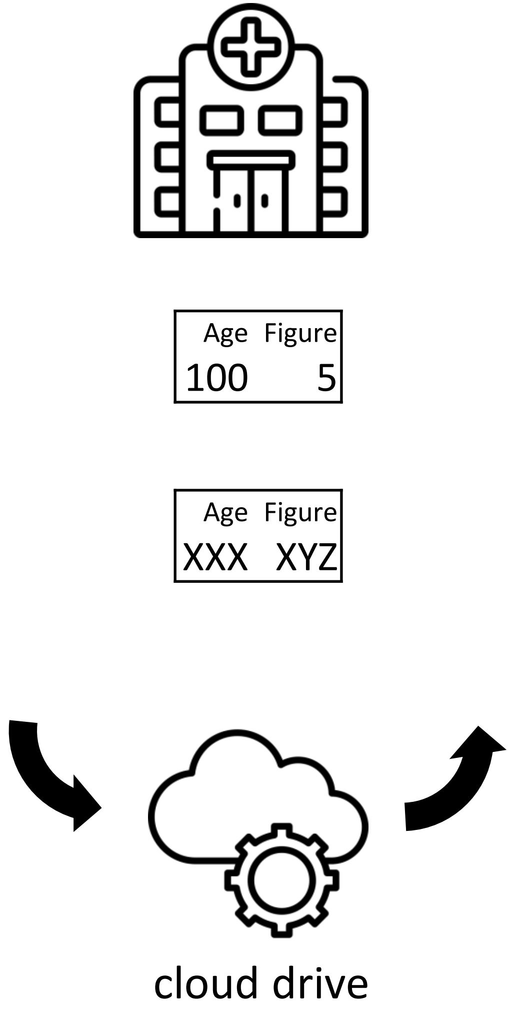
Age	Figure
100	5

Age	Figure
XXX	XYZ

cloud drive

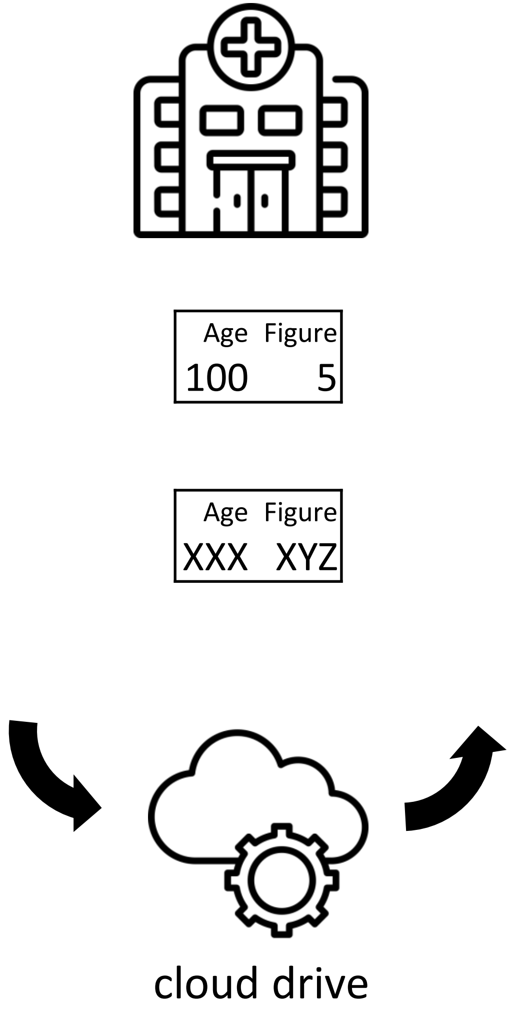
Need of general optimizing technique

- The computational model, called **circuits**, are of great interest
- Problem : FHE require lots of **calculation cost**



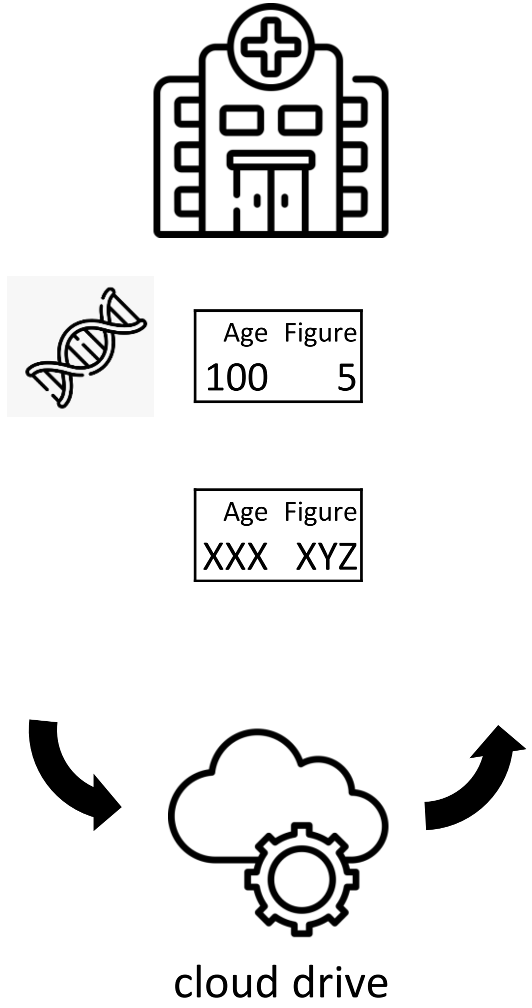
Need of general optimizing technique

- The computational model, called **circuits**, are of great interest
- Problem : FHE require lots of **calculation cost**
- One Solution : Domain-specific optimizations



Need of general optimizing technique

- The computational model, called **circuits**, are of great interest
- Problem : FHE require lots of **calculation cost**
- One Solution : Domain-specific optimizations



Need of general optimizing technique

- The computational model, called **circuits**, are of great interest
- Problem : FHE require lots of **calculation cost**
- One Solution : Domain-specific optimizations

Age	Figure
100	5

Age	Figure
XXX	XYZ

Some specific algorithm in biology

cloud drive

Need of general optimizing technique

- The computational model, called **circuits**, are of great interest
- Problem : FHE require lots of **calculation cost**
- One Solution : Domain-specific optimizations

Age	Figure
100	5

Age	Figure
XXX	XYZ

Optimization technique*

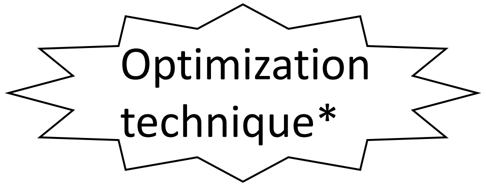
Some specific algorithm in biology

cloud drive

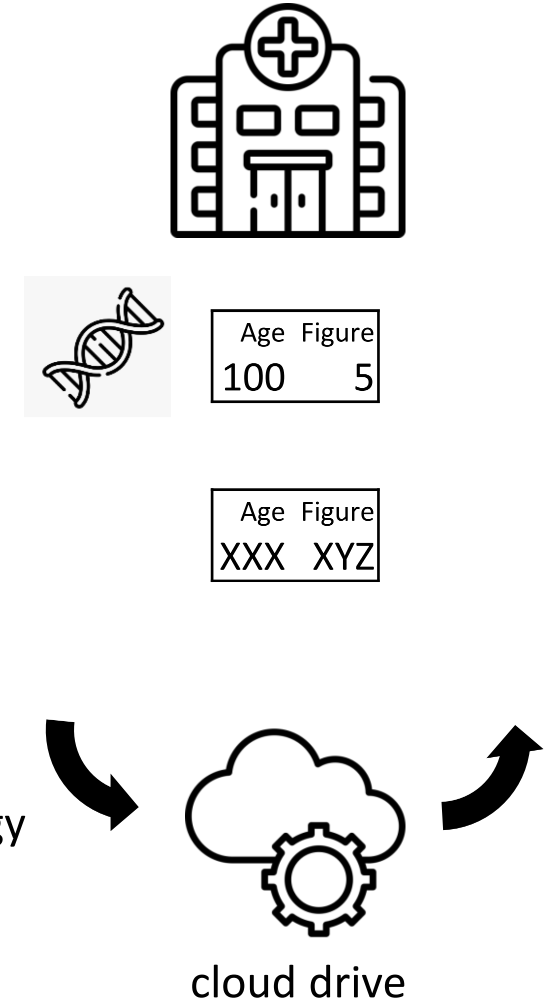
*Jung Hee Cheon et.al., "Homomorphic Computation of Edit Distance", Financial Cryptography and Data Security, 2015

Need of general optimizing technique

- The computational model, called **circuits**, are of great interest
- Problem : FHE require lots of **calculation cost**
- One Solution : Domain-specific optimizations
  - But, require an expertise in cryptography



Some specific algorithm in biology



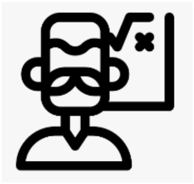
*Jung Hee Cheon et.al., “Homomorphic Computation of Edit Distance”, Financial Cryptography and Data Security, 2015

Need of general optimizing technique

- The computational model, called **circuits**, are of great interest
- Problem : FHE require lots of **calculation cost**
- One Solution : Domain-specific optimizations
  - But, require an expertise in cryptography

Age	Figure
100	5

Age	Figure
XXX	XYZ



Optimization technique*

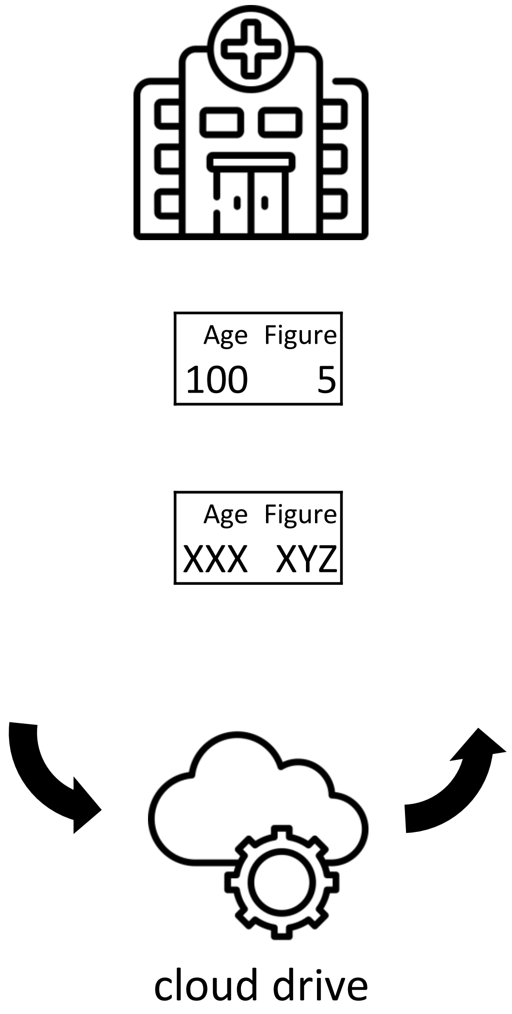
Some specific algorithm in biology

cloud drive

*Jung Hee Cheon et.al., “Homomorphic Computation of Edit Distance”, Financial Cryptography and Data Security, 2015

Need of general optimizing technique

- The computational model, called **circuits**, are of great interest
- Problem : FHE require lots of **calculation cost**
- One Solution : Domain-specific optimizations
  - But, require an expertise in cryptography
- Dream : Make an FHE compiler



Need of general optimizing technique

- The computational model, called **circuits**, are of great interest
- Problem : FHE require lots of **calculation cost**
- One Solution : Domain-specific optimizations
  - But, require an expertise in cryptography
- Dream : Make an FHE compiler
  - Compiler will equip circuit-to-circuit optimization

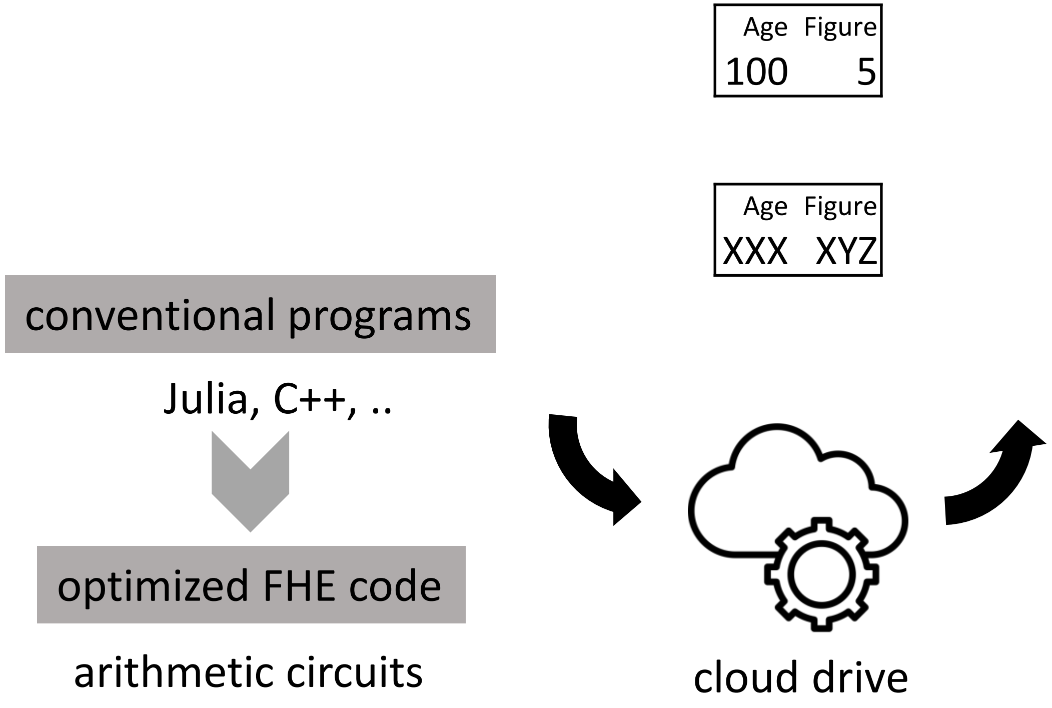
Age	Figure
100	5

Age	Figure
XXX	XYZ

cloud drive

Need of general optimizing technique

- The computational model, called **circuits**, are of great interest
- Problem : FHE require lots of **calculation cost**
- One Solution : Domain-specific optimizations
  - But, require an expertise in cryptography
- Dream : Make an FHE compiler
  - Compiler will equip circuit-to-circuit optimization

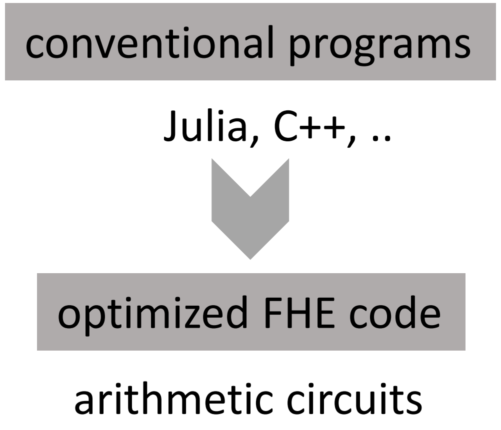


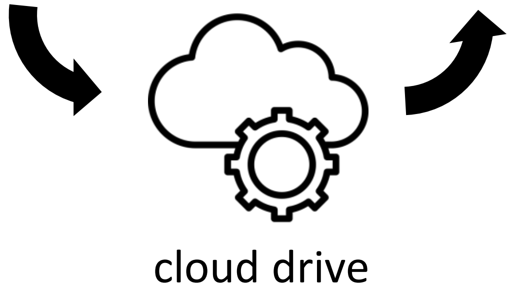
Need of general optimizing technique

- The computational model, called **circuits**, are of great interest
- Problem : FHE require lots of **calculation cost**
- One Solution : Domain-specific optimizations
  - But, require an expertise in cryptography
- Dream : Make an FHE compiler
  - Compiler will equip circuit-to-circuit optimization
  - But, current FHE compilers still favor FHE-friendly programs
  - Also, only hand-written optimization rules yet

Age	Figure
100	5

Age	Figure
XXX	XYZ





Idea

- Idea: Implement a general optimization technique

Idea

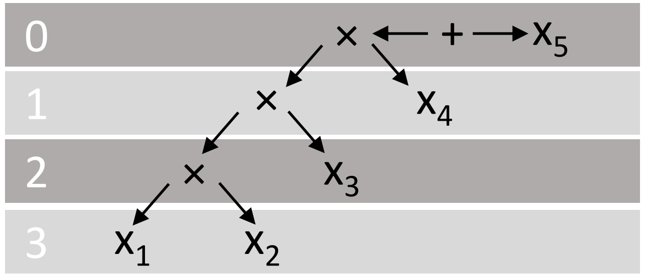
- Idea: Implement a general optimization technique
- **Reduce multiplicative(×) depth** which is the main overhead of program

Idea

- Idea: Implement a general optimization technique
- **Reduce multiplicative(×) depth** which is the main overhead of program

circuit of × depth 3

$c(x_1,x_2,x_3,x_4,x_5) = ((x_1 \times x_2) \times x_3) \times x_4 + x_5$

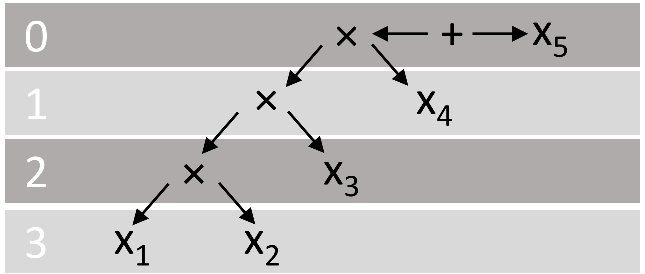


Idea

- Idea: Implement a general optimization technique
- **Reduce multiplicative(×) depth** which is the main overhead of program

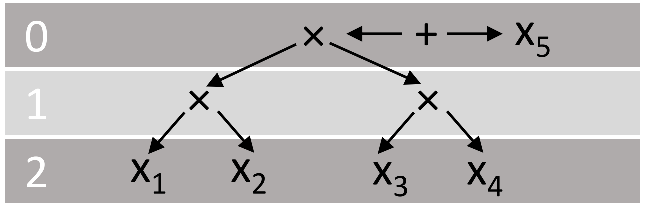
circuit of × depth 3

$c(x_1,x_2,x_3,x_4,x_5) = ((x_1 \times x_2) \times x_3) \times x_4 + x_5$



circuit of × depth 2

$c'(x_1,x_2,x_3,x_4,x_5) = (x_1 \times x_2) \times (x_3 \times x_4) + x_5$

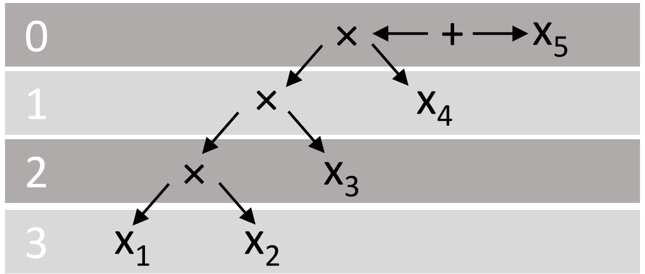


Idea

- Idea: Implement a general optimization technique
- **Reduce multiplicative(×) depth** which is the main overhead of program

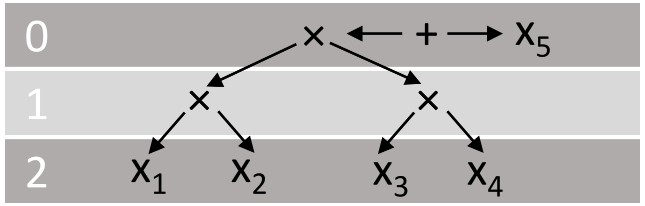
circuit of × depth 3

$c(x_1,x_2,x_3,x_4,x_5) = ((x_1 \times x_2) \times x_3) \times x_4 + x_5$



circuit of × depth 2

$c'(x_1,x_2,x_3,x_4,x_5) = (x_1 \times x_2) \times (x_3 \times x_4) + x_5$



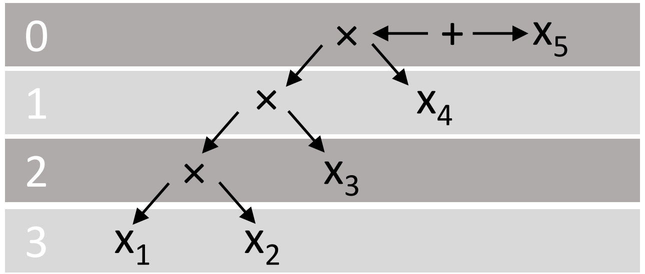
- **Syntax-guided synthesis** to find depth-decreasing rules

Idea

- Idea: Implement a general optimization technique
- **Reduce multiplicative(×) depth** which is the main overhead of program

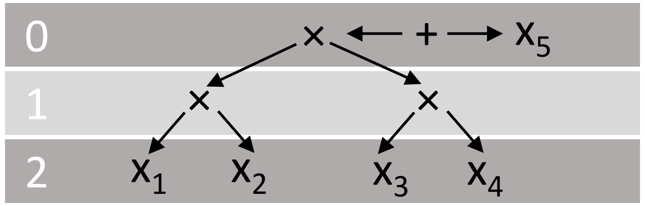
circuit of × depth 3

$c(x_1,x_2,x_3,x_4,x_5) = ((x_1 \times x_2) \times x_3) \times x_4 + x_5$



circuit of × depth 2

$c'(x_1,x_2,x_3,x_4,x_5) = (x_1 \times x_2) \times (x_3 \times x_4) + x_5$



- **Syntax-guided synthesis** to find depth-decreasing rules
- Then, can apply rule generally

Appealing result: A new best attempt

1. Speedup

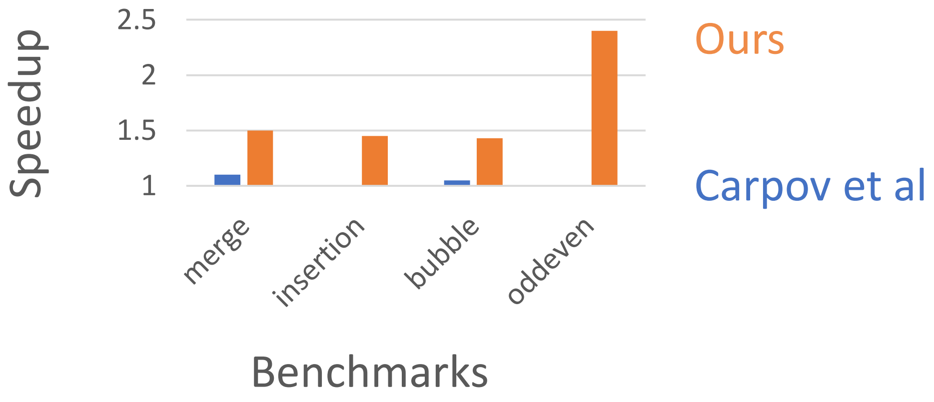
For 4 sorting benchmarks, average 2x speedup, outperforming state-of-the-art FHE compiler (1.1x)

*Dongkwon Lee et al., Optimizing homomorphic evaluation circuits by program synthesis and term rewriting, PLDI, 2020.

Appealing result: A new best attempt

1. Speedup

For 4 sorting benchmarks, average 2x speedup, outperforming state-of-the-art FHE compiler (1.1x)

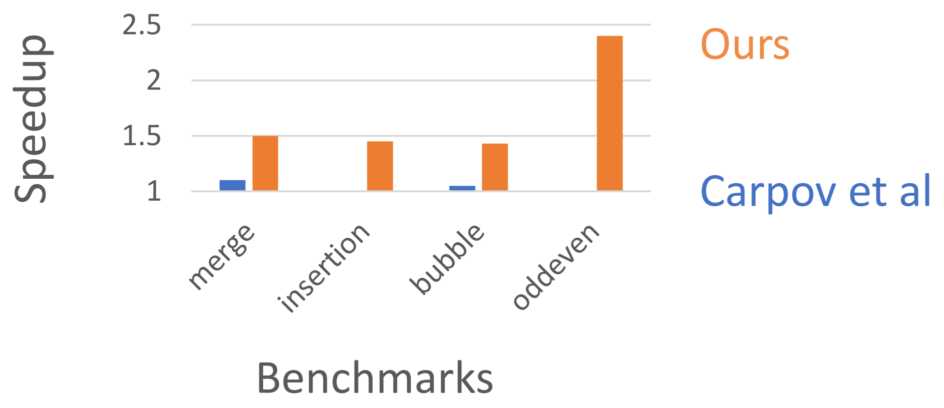


*Dongkwon Lee et al., Optimizing homomorphic evaluation circuits by program synthesis and term rewriting, PLDI, 2020.

Appealing result: A new best attempt

1. Speedup

For 4 sorting benchmarks, average 2x speedup, outperforming state-of-the-art FHE compiler (1.1x)



2. Decreasing multiplicative (×) depth

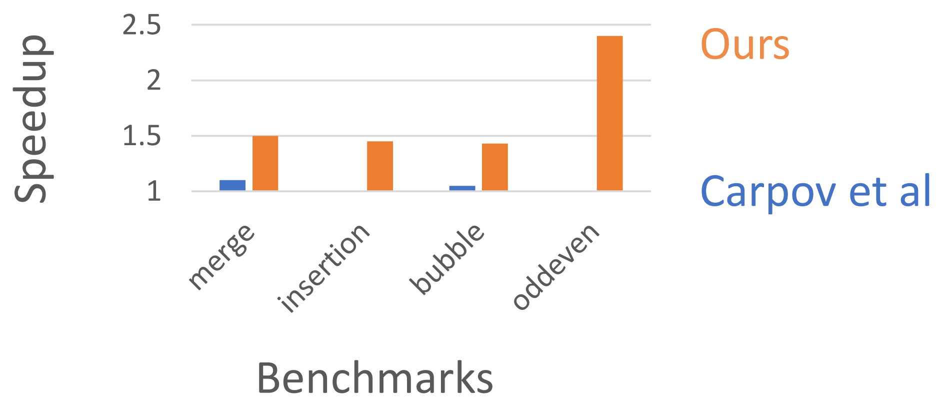
For 4 sorting benchmarks, average 20% decrease, outperforming state-of-the-art FHE compiler (4.4%)

*Dongkwon Lee et al., Optimizing homomorphic evaluation circuits by program synthesis and term rewriting, PLDI, 2020.

Appealing result: A new best attempt

1. Speedup

For 4 sorting benchmarks, average 2x speedup, outperforming state-of-the-art FHE compiler (1.1x)



2. Decreasing multiplicative (×) depth

For 4 sorting benchmarks, average 20% decrease, outperforming state-of-the-art FHE compiler (4.4%)

	Original	**Carpov et al**	**Ours**
merge	45	41	36
insertion	45	45	36
bubble	45	41	36
oddeven	25	25	20

*Dongkwon Lee et al., Optimizing homomorphic evaluation circuits by program synthesis and term rewriting, PLDI, 2020.

Killer example – why ×?

Killer example – why ×?

- Observe multiplicative depth is the main overhead
  - The noise which stacks for each operation, grows much higher in × than +

Killer example – why ×?

- Observe multiplicative depth is the main overhead
  - The noise which stacks for each operation, grows much higher in × than +

FHE scheme to encode 1 bit (m)

Killer example – why ×?

- Observe multiplicative depth is the main overhead
  - The noise which stacks for each operation, grows much higher in × than +

FHE scheme to encode 1 bit (m)

- m, + and × are on $Z_2=\{0, 1\}$
- secret key : *p*, random number : *q*, *r* ($r \ll p$)

plaintext

$$m \xrightarrow{\text{Enc}} pq + 2r + m$$

Killer example – why ×?

- Observe multiplicative depth is the main overhead
  - The noise which stacks for each operation, grows much higher in × than +

FHE scheme to encode 1 bit (m)

- m, + and × are on $Z_2=\{0, 1\}$
- secret key : *p*, random number : *q*, *r* ($r \ll p$)

plaintext

$m \xrightarrow{\text{Enc}} pq + 2r + m$

$((c \bmod p) \bmod 2) \xleftarrow{\text{Dec}} c$

Killer example – why ×?

- Observe multiplicative depth is the main overhead
  - The noise which stacks for each operation, grows much higher in × than +

FHE scheme to encode 1 bit (m)

- m, + and × are on $Z_2=\{0, 1\}$
- secret key : *p*, random number : *q*, *r* (r ≪ p)

plaintext

$m \xrightarrow{\text{Enc}} pq + 2r + m$

$((c \bmod p) \bmod 2) \xleftarrow{\text{Dec}} c$

$m \xleftarrow{\text{Dec}} pq + 2r + m$

Killer example – why ×?

- Observe multiplicative depth is the main overhead
  - The noise which stacks for each operation, grows much higher in × than +

FHE scheme to encode 1 bit (m)

- m, + and × are on $Z_2=\{0, 1\}$
- secret key : *p*, random number : *q*, *r* (r ≪ p)

plaintext

$$m \xrightarrow{\text{Enc}} pq + 2r + m$$

$$((c \bmod p) \bmod 2) \xleftarrow{\text{Dec}} c$$

$$m \xleftarrow{\text{Dec}} pq + 2r + m \qquad (2r + m < 2r + 2 \ll p)$$

$$\begin{aligned} \therefore \mathrm{Dec}(pq + 2r + m) &= (((2r + m) \bmod p) \bmod 2) \\ &= ((2r + m) \bmod 2) \\ &= m \end{aligned}$$

Killer example – why ×?

- Observe multiplicative depth is the main overhead
  - The noise which stacks for each operation, grows much higher in × than +

FHE scheme to encode 1 bit (m)

- m, + and × are on $Z_2=\{0, 1\}$
- secret key : *p*, random number : *q*, *r* ($r \ll p$)

plaintext

$m \xrightarrow{\text{Enc}} pq + 2r + m$

$((c \bmod p) \bmod 2) \xleftarrow{\text{Dec}} c$

$m \xleftarrow{\text{Dec}} pq + 2r + m$ ($2r + m < 2r + 2 \ll p$)

$\therefore \text{Dec}(pq + 2r + m) = (((2r + m) \bmod p) \bmod 2)$
$= ((2r + m) \bmod 2)$
$= m$

$c_1 + c_2 = p(q_1+q_2) + 2(r_1+r_2) + m_1 + m_2$

$c_1 \times c_2 =$

Killer example – why ×?

- Observe multiplicative depth is the main overhead
  - The noise which stacks for each operation, grows much higher in × than +

FHE scheme to encode 1 bit (m)

- m, + and × are on $Z_2=\{0, 1\}$
- secret key : *p*, random number : *q*, *r* ($r \ll p$)

plaintext

$m \xrightarrow{\text{Enc}} pq + 2r + m$

$((c \bmod p) \bmod 2) \xleftarrow{\text{Dec}} c$

$m \xleftarrow{\text{Dec}} pq + 2r + m$ $(2r + m < 2r + 2 \ll p)$

$$\therefore \mathrm{Dec}(pq + 2r + m) = (((2r + m) \bmod p) \bmod 2)$$
$$= ((2r + m) \bmod 2)$$
$$= m$$

noise after (mod p)

$c_1 + c_2 = p(q_1+q_2) + 2(r_1+r_2) + m_1 + m_2$

$c_1 \times c_2 =$

Killer example – why ×?

- Observe multiplicative depth is the main overhead
  - The noise which stacks for each operation, grows much higher in × than +

FHE scheme to encode 1 bit (m)

- m, + and × are on $Z_2=\{0, 1\}$
- secret key : *p*, random number : *q*, *r* ($r \ll p$)

plaintext

$m \xrightarrow{\text{Enc}} pq + 2r + m$

$((c \bmod p) \bmod 2) \xleftarrow{\text{Dec}} c$

$m \xleftarrow{\text{Dec}} pq + 2r + m$ $\quad (2r + m < 2r + 2 \ll p)$

$$\therefore \mathrm{Dec}(pq + 2r + m) = (((2r + m) \bmod p) \bmod 2) = ((2r + m) \bmod 2) = m$$

noise after (mod p)

$c_1 + c_2 = p(q_1+q_2) + 2(r_1+r_2) + m_1 + m_2$

$c_1 \times c_2 = p(pq_1q_2 + \ldots) + 2(2r_1r_2 + r_1m_2 + r_2m_1) + m_1m_2$

Killer example – why ×?

- Observe multiplicative depth is the main overhead
  - The noise which stacks for each operation, grows much higher in × than +

FHE scheme to encode 1 bit (m)

- m, + and × are on Z_2={0, 1}
- secret key : *p*, random number : *q*, *r* ($r \ll p$)

plaintext

$$m \xrightarrow{\text{Enc}} pq + 2r + m$$

$$((c \bmod p) \bmod 2) \xleftarrow{\text{Dec}} c$$

$$m \xleftarrow{\text{Dec}} pq + 2r + m \quad (2r + m < 2r + 2 \ll p)$$

$$\therefore \text{Dec}(pq + 2r + m) = (((2r + m) \bmod p) \bmod 2)$$
$$= ((2r + m) \bmod 2)$$
$$= m$$

noise after (mod p)

$$c_1 + c_2 = p(q_1+q_2) + 2(r_1+r_2) + m_1 + m_2$$

$$c_1 \times c_2 = p(pq_1q_2 + \ldots) + 2(2r_1r_2 + r_1m_2 + r_2m_1) + m_1m_2$$

noise after (mod p)

Structure

- Step 1 - Offline Learning (with training circuits)

Training set

Rule

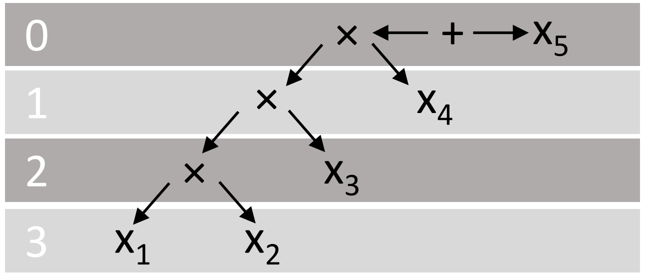
Structure

- Step 1 - Offline Learning (with training circuits)

Training set

- $((x_1 \times x_2) \times x_3) \times x_4 + x_5$
- …

Rule $\emptyset$



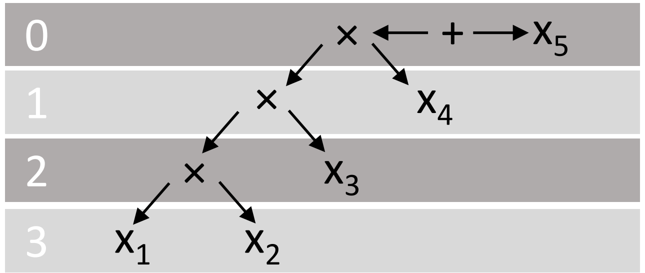
Structure

- Step 1 - Offline Learning (with training circuits)

Training set

- $((x_1 \times x_2) \times x_3) \times x_4 + x_5$
- …

Rule $\emptyset$



nodes of path that leads to deepest part

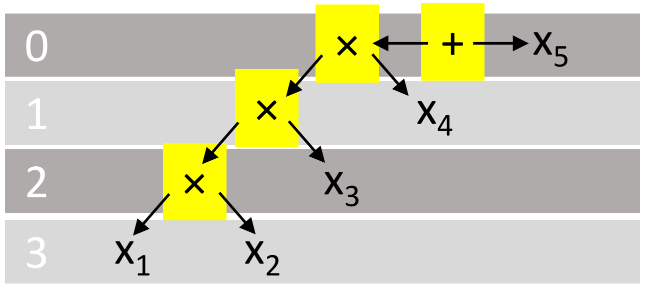
Structure

- Step 1 - Offline Learning (with training circuits)

Training set

- $((x_1 \times x_2) \times x_3) \times x_4 + x_5$
- …

Rule ∅



nodes of path that leads to deepest part

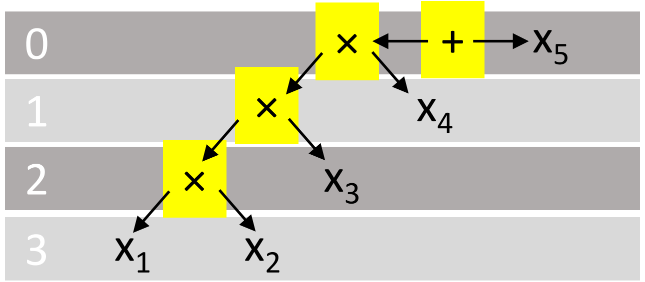
Structure

- Step 1 - Offline Learning (with training circuits)

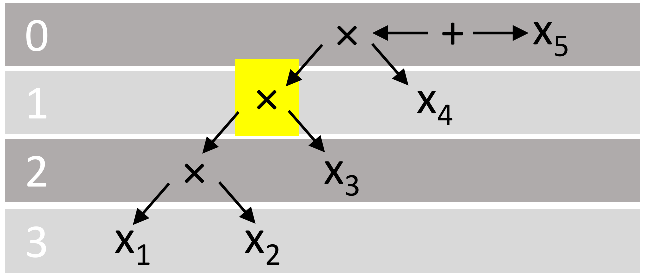
Training set

- $((x_1 \times x_2) \times x_3) \times x_4 + x_5$
- …

Rule $\emptyset$



nodes of path that leads to deepest part



select any node

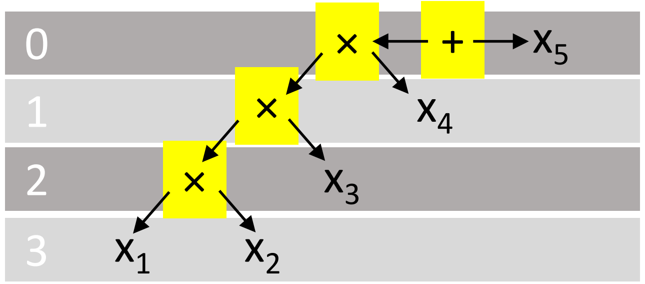
Structure

- Step 1 - Offline Learning (with training circuits)

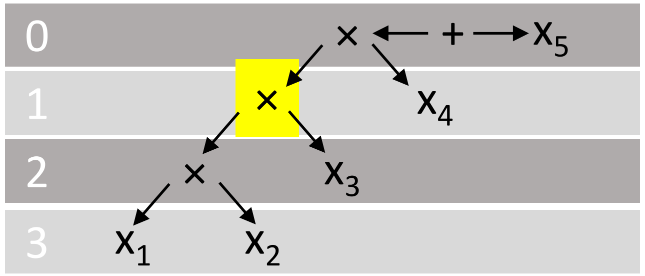
Training set

- $((x_1 \times x_2) \times x_3) \times x_4 + x_5$
- …

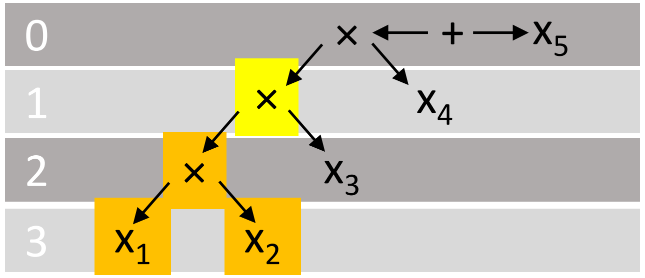
Rule $\emptyset$



nodes of path that leads to deepest part



select any node



select left or right (one with larger depth)

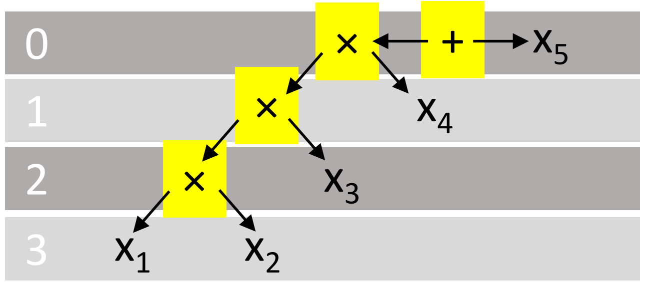
Structure

- Step 1 - Offline Learning (with training circuits)

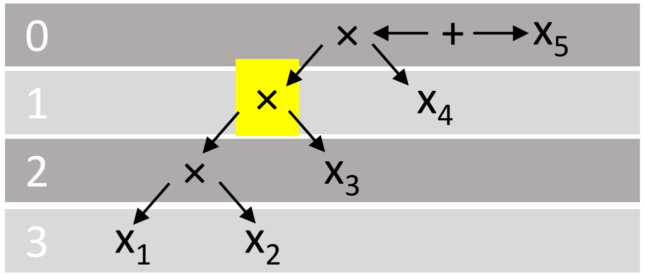
Training set

- $((x_1 \times x_2) \times x_3) \times x_4 + x_5$
- …

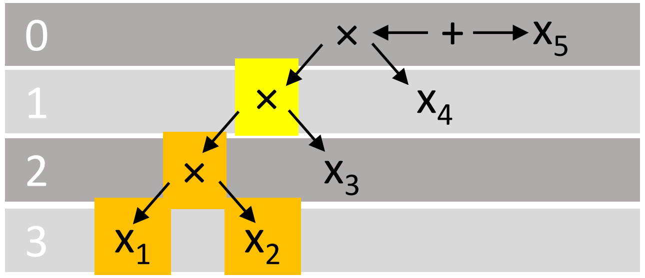
Rule $\emptyset$



nodes of path that leads to deepest part



select any node



select left or right (one with larger depth)

1. $\forall x_1,x_2.x_1 \times x_2 \Leftrightarrow r'(x_1,x_2)$
2. $\text{depth}(r') < \text{depth}(x_1 \times x_2)$

synthesize a rewrite rule if possible

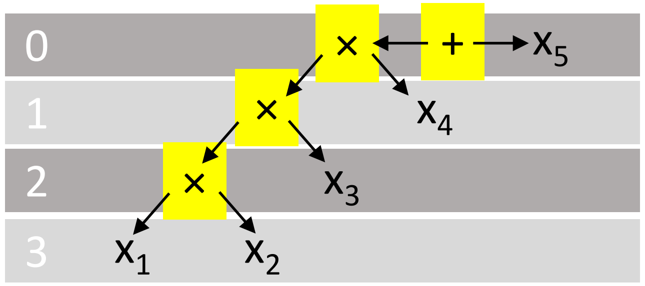
Structure

- Step 1 - Offline Learning (with training circuits)

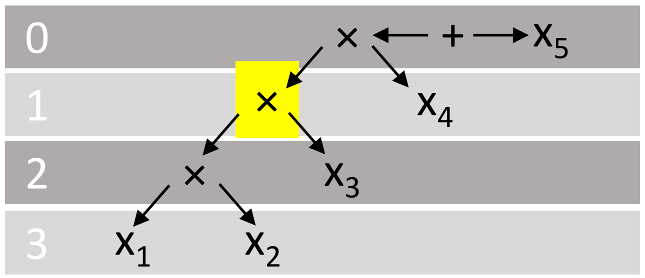
Training set

- $((x_1 \times x_2) \times x_3) \times x_4 + x_5$
- …

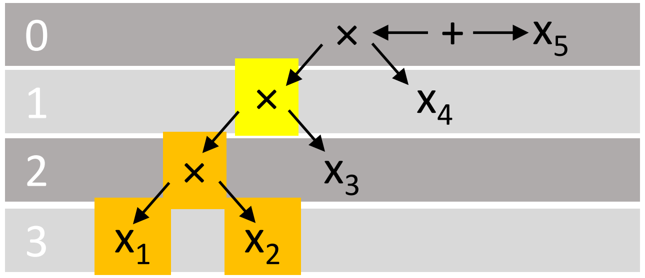
Rule $\emptyset$



nodes of path that leads to deepest part



select any node



select left or right (one with larger depth)

1. $\forall x_1, x_2 . x_1 \times x_2 \Leftrightarrow r'(x_1, x_2)$
2. $\mathrm{depth}(r') < \mathrm{depth}(x_1 \times x_2)$

synthesize a rewrite rule if possible

$\perp$

(no rewrite rule)

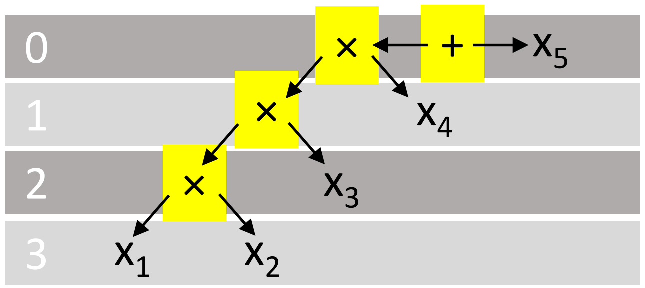
Structure

- Step 1 - Offline Learning (with training circuits)

Training set

- $((x_1 \times x_2) \times x_3) \times x_4 + x_5$
- ...

Rule $\emptyset$



nodes of path that leads to deepest part

select any node

select left or right (one with larger depth)

synthesize a rewrite rule if possible

(found rewrite rule)

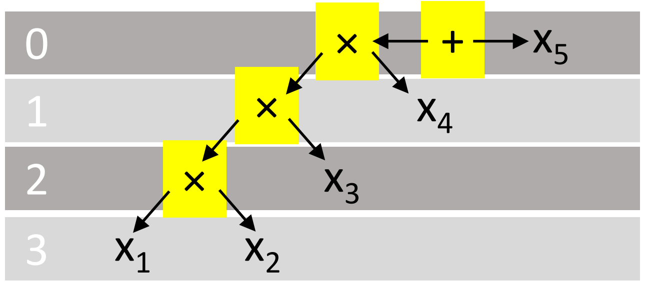
Structure

- Step 1 - Offline Learning (with training circuits)

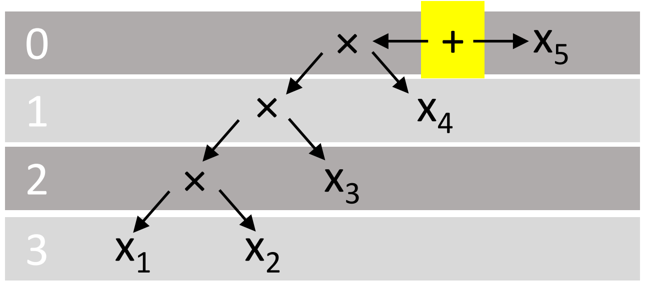
Training set

- $((x_1 \times x_2) \times x_3) \times x_4 + x_5$
- ...

Rule Ø



nodes of path that leads to deepest part



select any node

select left or right (one with larger depth)

synthesize a rewrite rule if possible

(found rewrite rule)

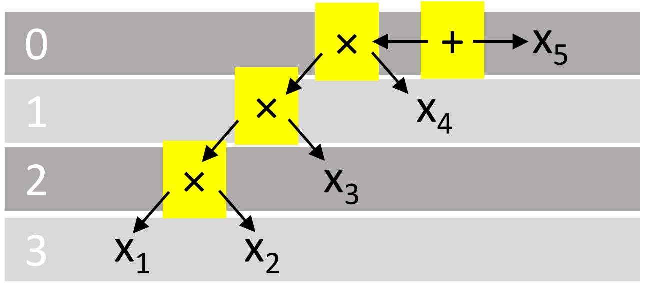
Structure

- Step 1 - Offline Learning (with training circuits)

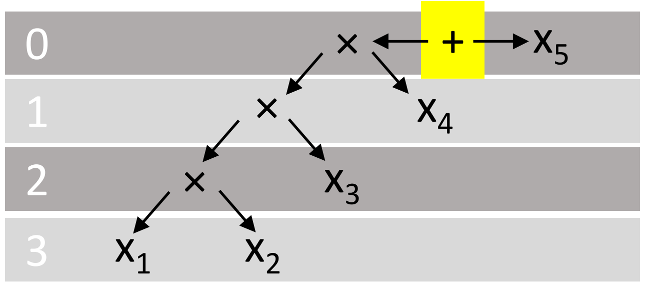
Training set

- $((x_1 \times x_2) \times x_3) \times x_4 + x_5$
- ...

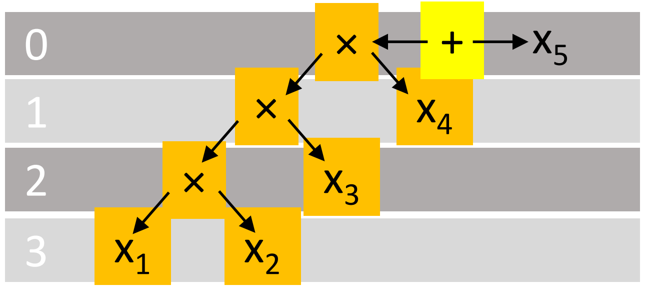
Rule $\emptyset$



nodes of path that leads to deepest part



select any node



select left or right (one with larger depth)

synthesize a rewrite rule if possible

(found rewrite rule)

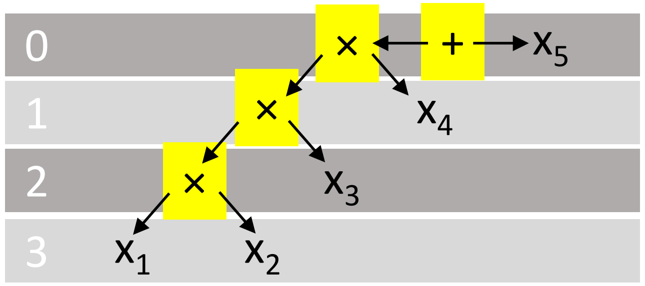
Structure

- Step 1 - Offline Learning (with training circuits)

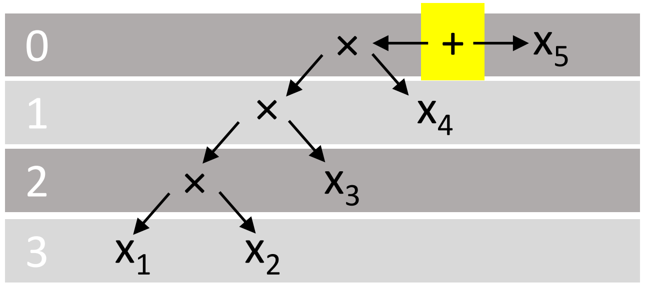
Training set

- $((x_1 \times x_2) \times x_3) \times x_4 + x_5$
- …

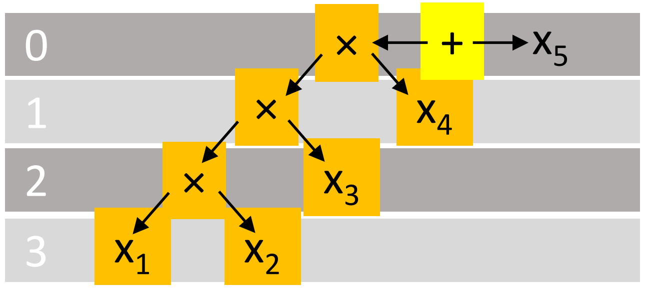
Rule $\emptyset$



nodes of path that leads to deepest part



select any node



select left or right (one with larger depth)

1. $\forall x_i.(((x_1 \times x_2)) \times x_3) \times x_4) \Leftrightarrow r'(x_1,x_2,x_3,x_4)$
2. $\mathrm{depth}(r') < \mathrm{depth}(((x_1 \times x_2)) \times x_3) \times x_4)$

synthesize a rewrite rule if possible

(found rewrite rule)

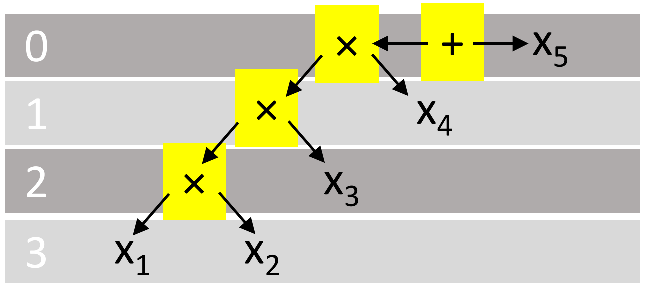
Structure

- Step 1 - Offline Learning (with training circuits)

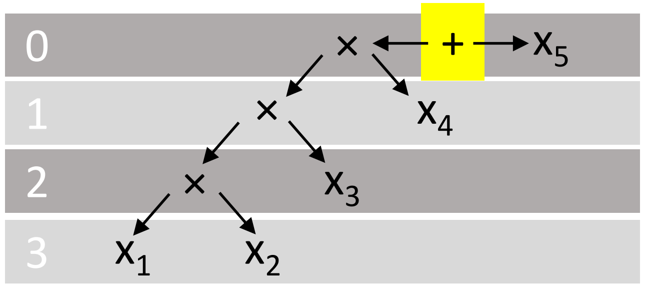
Training set

- $((x_1 \times x_2) \times x_3) \times x_4 + x_5$
- ...

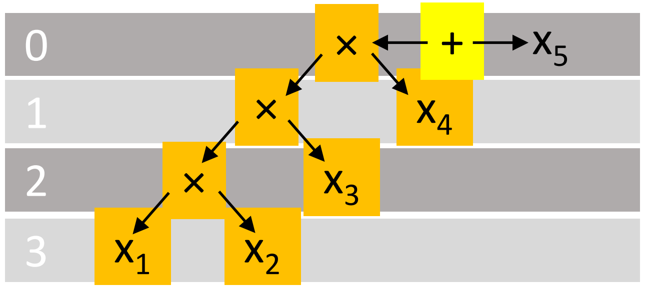
Rule $\emptyset$



nodes of path that leads to deepest part



select any node



select left or right (one with larger depth)

1. $\forall x_i.(((x_1 \times x_2)) \times x_3) \times x_4) \Leftrightarrow r'(x_1,x_2,x_3,x_4)$
2. $\mathrm{depth}(r') < \mathrm{depth}(((x_1 \times x_2)) \times x_3) \times x_4)$

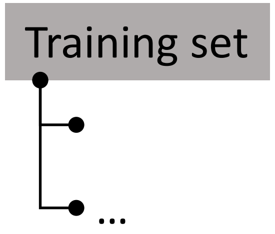
synthesize a rewrite rule if possible

$r' = (x_1 \times x_2) \times (x_3 \times x_4)$

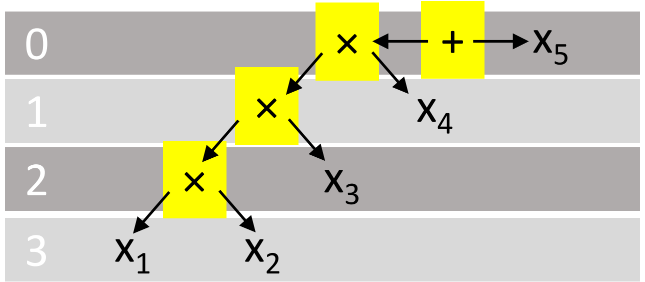
(found rewrite rule)

Structure

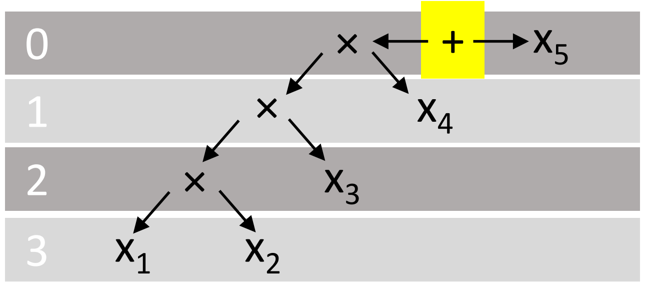
- Step 1 - Offline Learning (with training circuits)



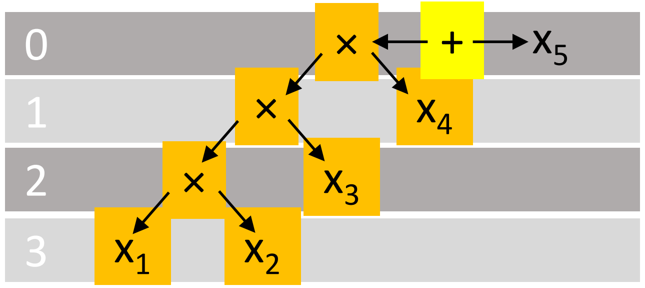
Rule Ø



nodes of path that leads to deepest part



select any node



select left or right (one with larger depth)

1. $\forall x_i.(((x_1 \times x_2)) \times x_3) \times x_4) \Leftrightarrow r'(x_1,x_2,x_3,x_4)$
2. $\text{depth}(r') < \text{depth}(((x_1 \times x_2)) \times x_3) \times x_4)$

synthesize a rewrite rule if possible

$r' = (x_1 \times x_2) \times (x_3 \times x_4)$

(found rewrite rule)

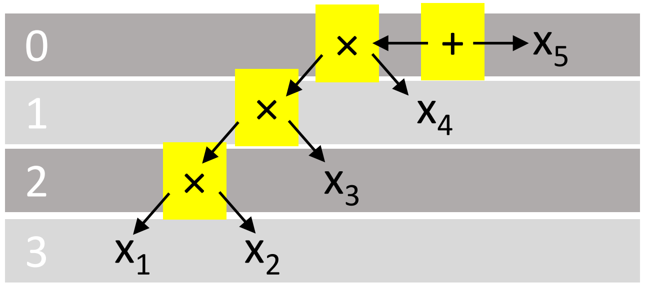
Structure

- Step 1 - Offline Learning (with training circuits)

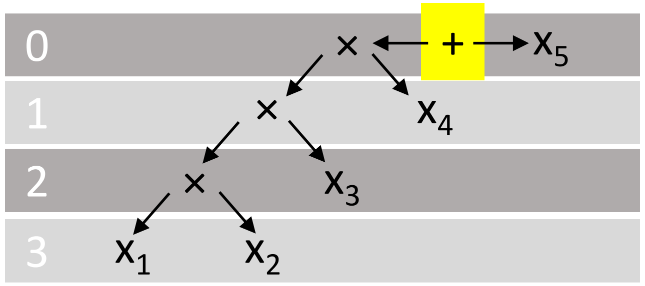
Training set

- $(x_1 \times x_2) \times (x_3 \times x_4) + x_5$
- …

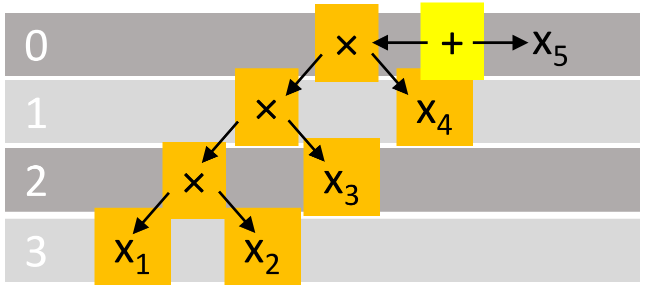
Rule $\emptyset$



nodes of path that leads to deepest part



select any node



select left or right (one with larger depth)

1. $\forall x_i.(((x_1 \times x_2)) \times x_3) \times x_4) \Leftrightarrow r'(x_1,x_2,x_3,x_4)$
2. $\mathrm{depth}(r') < \mathrm{depth}(((x_1 \times x_2)) \times x_3) \times x_4)$

synthesize a rewrite rule if possible

$r' = (x_1 \times x_2) \times (x_3 \times x_4)$

(found rewrite rule)

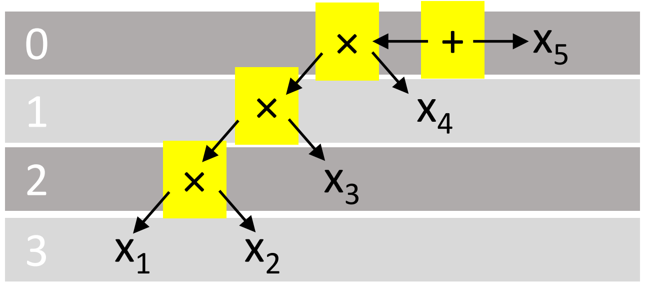
Structure

- Step 1 - Offline Learning (with training circuits)

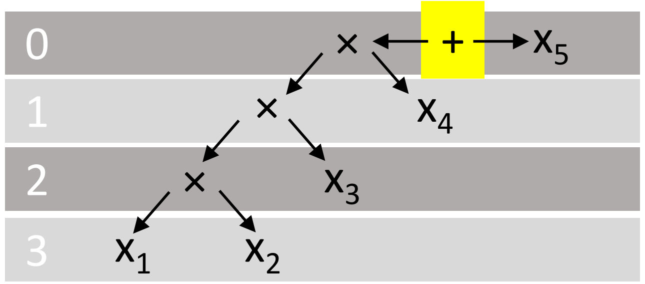
Training set

- $(x_1 \times x_2) \times (x_3 \times x_4) + x_5$
- …

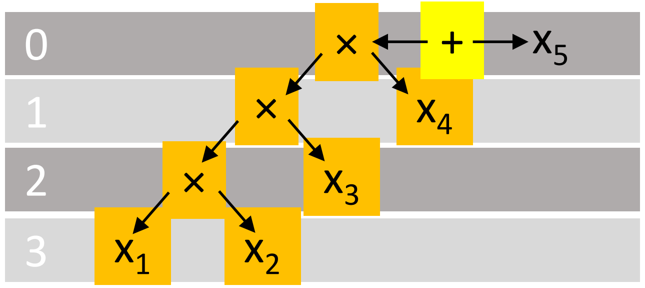
Rule



nodes of path that leads to deepest part



select any node



select left or right (one with larger depth)

1. $\forall x_i.(((x_1 \times x_2)) \times x_3) \times x_4) \Leftrightarrow r'(x_1,x_2,x_3,x_4)$
2. $\mathrm{depth}(r') < \mathrm{depth}(((x_1 \times x_2)) \times x_3) \times x_4)$

synthesize a rewrite rule if possible

$r' = (x_1 \times x_2) \times (x_3 \times x_4)$

(found rewrite rule)

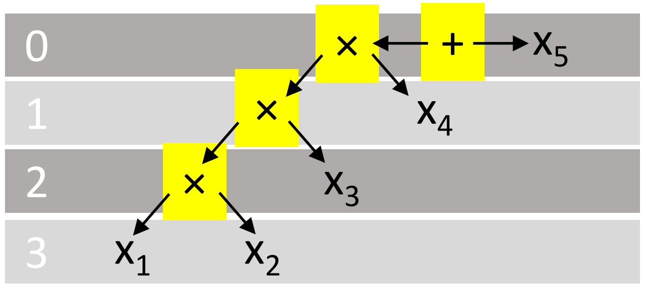
Structure

- Step 1 - Offline Learning (with training circuits)
  - Simplify each rule by changing sub-formula to new variable, then running SAT solver

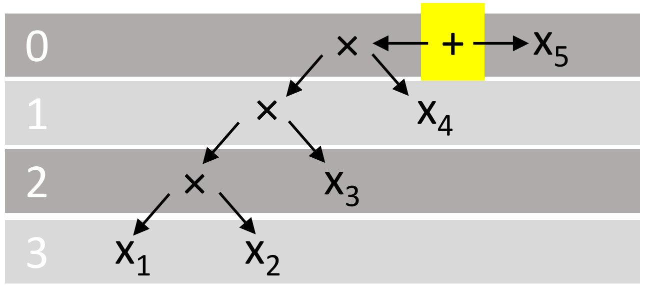
Training set

- $(x_1 \times x_2) \times (x_3 \times x_4) + x_5$
- …

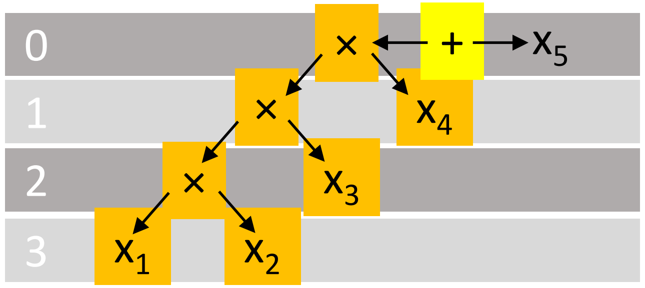
Rule 1 rule (simplified)



nodes of path that leads to deepest part



select any node



select left or right (one with larger depth)

1. $\forall x_i.(((x_1 \times x_2)) \times x_3) \times x_4) \Leftrightarrow r'(x_1,x_2,x_3,x_4)$
2. $\mathrm{depth}(r') < \mathrm{depth}(((x_1 \times x_2)) \times x_3) \times x_4)$

synthesize a rewrite rule if possible

$r' = (x_1 \times x_2) \times (x_3 \times x_4)$

(found rewrite rule)

Structure

- Step 1 - Offline Learning (with training circuits)
  - Simplify each rule by changing sub-formula to new variable, then running SAT solver

Training set

- $(x_1 \times x_2) \times (x_3 \times x_4) + x_5$
- …

Rule 1 rule (simplified)

nodes of path that leads to deepest part

select any node

select left or right (one with larger depth)

synthesize a rewrite rule if possible

(found rewrite rule)

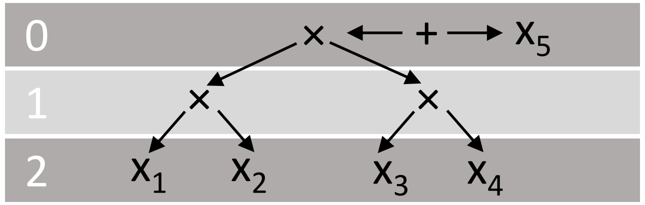
Structure

- Step 1 - Offline Learning (with training circuits)
  - Simplify each rule by changing sub-formula to new variable, then running SAT solver

Training set

- $(x_1 \times x_2) \times (x_3 \times x_4) + x_5$
- …

Rule 1 rule (simplified)



nodes of path that leads to deepest part

start again…

select any node

select left or right (one with larger depth)

synthesize a rewrite rule if possible

(found rewrite rule)

Structure

- Step 1 - Offline Learning (with training circuits)
  - Simplify each rule by changing sub-formula to new variable, then running SAT solver

Training set

- $(x_1 \times x_2) \times (x_3 \times x_4) + x_5$
- …

Rule 1 rule (simplified)

Structure

- Step 1 - Offline Learning (with training circuits)
  - Simplify each rule by changing sub-formula to new variable, then running SAT solver

Training set

- $(x_1 \times x_2) \times (x_3 \times x_4) + x_5$
- …

Rule 1 rule (simplified)

Structure

- Step 1 - Offline Learning (with training circuits)
  - Simplify each rule by changing sub-formula to new variable, then running SAT solver

Training set

- $(x_1 \times x_2) \times (x_3 \times x_4) + x_5$
- …

Rule 1 rule : $((x_1 \times x_2) \times x_3) \times x_4 \Leftrightarrow (x_1 \times x_2) \times (x_3 \times x_4)$

Structure

- Step 1 - Offline Learning (with training circuits)
  - Simplify each rule by changing sub-formula to new variable, then running SAT solver

Training set

- $(x_1 \times x_2) \times (x_3 \times x_4) + x_5$
- …

Rule

1 rule : $((x_1 \times x_2) \times x_3) \times x_4 \Leftrightarrow (x_1 \times x_2) \times (x_3 \times x_4)$

$((x_1 \times x_2) \times x_3) \times x_4 \Leftrightarrow (x_1 \times x_2) \times (x_3 \times x_4)$ Choose any that's good enough

Structure

- Step 1 - Offline Learning (with training circuits)
  - Simplify each rule by changing sub-formula to new variable, then running SAT solver

Training set

- $(x_1 \times x_2) \times (x_3 \times x_4) + x_5$
- …

Rule

1 rule : $((x_1 \times x_2) \times x_3) \times x_4 \Leftrightarrow (x_1 \times x_2) \times (x_3 \times x_4)$

$((x_1 \times x_2) \times x_3) \times x_4 \Leftrightarrow (x_1 \times x_2) \times (x_3 \times x_4)$

$(a \times x_3) \times x_4 \Leftrightarrow (x_1 \times x_2) \times a$

Structure

- Step 1 - Offline Learning (with training circuits)
  - Simplify each rule by changing sub-formula to new variable, then running SAT solver

Training set

- $(x_1 \times x_2) \times (x_3 \times x_4) + x_5$
- …

Rule

1 rule : $((x_1 \times x_2) \times x_3) \times x_4 \Leftrightarrow (x_1 \times x_2) \times (x_3 \times x_4)$

$((x_1 \times x_2) \times x_3) \times x_4 \Leftrightarrow (x_1 \times x_2) \times (x_3 \times x_4)$

$(a \times x_3) \times x_4 \Leftrightarrow (x_1 \times x_2) \times a$ NO!

Structure

- Step 1 - Offline Learning (with training circuits)
  - Simplify each rule by changing sub-formula to new variable, then running SAT solver

Training set

- $(x_1 \times x_2) \times (x_3 \times x_4) + x_5$
- …

Rule 1 rule : $((x_1 \times x_2) \times x_3) \times x_4 \Leftrightarrow (x_1 \times x_2) \times (x_3 \times x_4)$

$((x_1 \times x_2) \times x_3) \times x_4 \Leftrightarrow (x_1 \times x_2) \times (x_3 \times x_4)$

$(a \times x_3) \times x_4 \Leftrightarrow (x_1 \times x_2) \times a$ NO!

$((x_1 \times x_2) \times x_3) \times x_4 \Leftrightarrow (x_1 \times x_2) \times (x_3 \times x_4)$ Choose any that's good enough

Structure

- Step 1 - Offline Learning (with training circuits)
  - Simplify each rule by changing sub-formula to new variable, then running SAT solver

Training set

- $(x_1 \times x_2) \times (x_3 \times x_4) + x_5$
- …

Rule

1 rule : $((x_1 \times x_2) \times x_3) \times x_4 \Leftrightarrow (x_1 \times x_2) \times (x_3 \times x_4)$

$((x_1 \times x_2) \times x_3) \times x_4 \Leftrightarrow (x_1 \times x_2) \times (x_3 \times x_4)$

$(a \times x_3) \times x_4 \Leftrightarrow (x_1 \times x_2) \times a$ NO!

$((x_1 \times x_2) \times x_3) \times x_4 \Leftrightarrow (x_1 \times x_2) \times (x_3 \times x_4)$

$a \times x_4 \Leftrightarrow a \times (x_3 \times x_4)$ NO!

…

Structure

- Step 1 - Offline Learning (with training circuits)
  - Simplify each rule by changing sub-formula to new variable, then running SAT solver

Training set

- $(x_1 \times x_2) \times (x_3 \times x_4) + x_5$
- …

Rule

1 rule : $((x_1 \times x_2) \times x_3) \times x_4 \Leftrightarrow (x_1 \times x_2) \times (x_3 \times x_4)$

$((x_1 \times x_2) \times x_3) \times x_4 \Leftrightarrow (x_1 \times x_2) \times (x_3 \times x_4)$

$(a \times x_3) \times x_4 \Leftrightarrow (x_1 \times x_2) \times a$ NO!

$((x_1 \times x_2) \times x_3) \times x_4 \Leftrightarrow (x_1 \times x_2) \times (x_3 \times x_4)$

$a \times x_4 \Leftrightarrow a \times (x_3 \times x_4)$ NO!

…

Simplified Rule:
$((x_1 \times x_2) \times x_3) \times x_4 \Leftrightarrow (x_1 \times x_2) \times (x_3 \times x_4)$

Structure

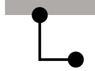
- Step 2 - Online Optimization
  - Find possible substitution that matches circuit to rule's (LHS)
  - Apply only when depth decreases

Rule 1 rule : $((x_1 \times x_2) \times x_3) \times x_4 \Leftrightarrow (x_1 \times x_2) \times (x_3 \times x_4)$

Structure

- Step 2 - Online Optimization
  - Find possible substitution that matches circuit to rule’s (LHS)
  - Apply only when depth decreases

Circuit



Rule 1 rule : $((x_1 \times x_2) \times x_3) \times x_4 \Leftrightarrow (x_1 \times x_2) \times (x_3 \times x_4)$

Structure

- Step 2 - Online Optimization
  - Find possible substitution that matches circuit to rule's (LHS)
  - Apply only when depth decreases

Circuit

Rule 1 rule : $((x_1 \times x_2) \times x_3) \times x_4 \Leftrightarrow (x_1 \times x_2) \times (x_3 \times x_4)$

Possible substitution

Structure

- Step 2 - Online Optimization
  - Find possible substitution that matches circuit to rule's (LHS)
  - Apply only when depth decreases

Circuit

$((((y_1 \times y_2) \times (y_3 \times y_4)) \times y_5) \times y_6) + y_7$

Rule 1 rule : $((x_1 \times x_2) \times x_3) \times x_4 \Leftrightarrow (x_1 \times x_2) \times (x_3 \times x_4)$

Possible substitution

Structure

- Step 2 - Online Optimization
  - Find possible substitution that matches circuit to rule's (LHS)
  - Apply only when depth decreases

Circuit

$((((y_1 \times y_2) \times (y_3 \times y_4)) \times y_5) \times y_6) + y_7$

Rule 1 rule : $((x_1 \times x_2) \times x_3) \times x_4 \Leftrightarrow (x_1 \times x_2) \times (x_3 \times x_4)$

Possible substitution

$((x_1 \times x_2) \times x_3) \times x_4$

Structure

- Step 2 - Online Optimization
  - Find possible substitution that matches circuit to rule's (LHS)
  - Apply only when depth decreases

Circuit

$((((y_1 \times y_2) \times (y_3 \times y_4)) \times y_5) \times y_6) + y_7$

Rule 1 rule : $((x_1 \times x_2) \times x_3) \times x_4 \Leftrightarrow (x_1 \times x_2) \times (x_3 \times x_4)$

Possible substitution

$((x_1 \times x_2) \times x_3) \times x_4$

$((((y_1 \times y_2) \times (y_3 \times y_4)) \times y_5) \times y_6) + y_7$

Structure

- Step 2 - Online Optimization
  - Find possible substitution that matches circuit to rule's (LHS)
  - Apply only when depth decreases

Circuit

$((((y_1 \times y_2) \times (y_3 \times y_4)) \times y_5) \times y_6) + y_7$

Rule

1 rule : $((x_1 \times x_2) \times x_3) \times x_4 \Leftrightarrow (x_1 \times x_2) \times (x_3 \times x_4)$

Possible substitution

$((x_1 \times x_2) \times x_3) \times x_4$

$(((\underbrace{(y_1 \times y_2)}_{x_1} \times \underbrace{(y_3 \times y_4)}_{x_2}) \times \underbrace{y_5}_{x_3}) \times \underbrace{y_6}_{x_4}) + y_7$

Structure

- Step 2 - Online Optimization
  - Find possible substitution that matches circuit to rule's (LHS)
  - Apply only when depth decreases

Circuit

$((((y_1 \times y_2) \times (y_3 \times y_4)) \times y_5) \times y_6) + y_7$

Rule

1 rule : $((x_1 \times x_2) \times x_3) \times x_4 \Leftrightarrow (x_1 \times x_2) \times (x_3 \times x_4)$

Possible substitution

$((x_1 \times x_2) \times x_3) \times x_4 \qquad \Leftrightarrow (x_1 \times x_2) \times (x_3 \times x_4)$

$((((y_1 \times y_2) \times (y_3 \times y_4)) \times y_5) \times y_6) + y_7 \qquad \Leftrightarrow ((y_1 \times y_2) \times (y_3 \times y_4)) \times (y_5 \times y_6)$ depth : 4 → 3 OK!

Evaluation

- Benchmark: 25 FHE applications

Evaluation

- Benchmark: 25 FHE applications
  - 9 open-source FHE compiler benchmarks (medical diagnosis, search, …)

Evaluation

- Benchmark: 25 FHE applications
    - 9 open-source FHE compiler benchmarks (medical diagnosis, search, …)
    - 4 privacy-preserving sorting algorithms

Evaluation

- Benchmark: 25 FHE applications
  - 9 open-source FHE compiler benchmarks (medical diagnosis, search, …)
  - 4 privacy-preserving sorting algorithms
  - 26 homomorphic bitwise operations

Evaluation

- Benchmark: 25 FHE applications
  - 9 open-source FHE compiler benchmarks (medical diagnosis, search, …)
  - 4 privacy-preserving sorting algorithms
  - 26 homomorphic bitwise operations
  - 25 more benchmarks from EPFL

Evaluation

- Benchmark: 25 FHE applications
  - 9 open-source FHE compiler benchmarks (medical diagnosis, search, …)
  - 4 privacy-preserving sorting algorithms
  - 26 homomorphic bitwise operations
  - 25 more benchmarks from EPFL
  - rule out 39 which are already depth-optimal or too large

Evaluation

- Benchmark: 25 FHE applications
  - 9 open-source FHE compiler benchmarks (medical diagnosis, search, …)
  - 4 privacy-preserving sorting algorithms
  - 26 homomorphic bitwise operations
  - 25 more benchmarks from EPFL
  - rule out 39 which are already depth-optimal or too large
- Comparison: state-of-the-art methods
  - Carpov et al.: use hand-written rewrite rules

Evaluation

- Benchmark: 25 FHE applications
  - 9 open-source FHE compiler benchmarks (medical diagnosis, search, …)
  - 4 privacy-preserving sorting algorithms
  - 26 homomorphic bitwise operations
  - 25 more benchmarks from EPFL
  - rule out 39 which are already depth-optimal or too large
- Comparison: state-of-the-art methods
  - Carpov et al.: use hand-written rewrite rules
- Evaluation method: cross validation

Evaluation

- Benchmark: 25 FHE applications
  - 9 open-source FHE compiler benchmarks (medical diagnosis, search, …)
  - 4 privacy-preserving sorting algorithms
  - 26 homomorphic bitwise operations
  - 25 more benchmarks from EPFL
  - rule out 39 which are already depth-optimal or too large
- Comparison: state-of-the-art methods
  - Carpov et al.: use hand-written rewrite rules
- Evaluation method: cross validation
- Q1. Effectiveness & Comparison
- Q2. How is it affected to the selection of training set?

Result

Q1. Effectiveness & Comparison

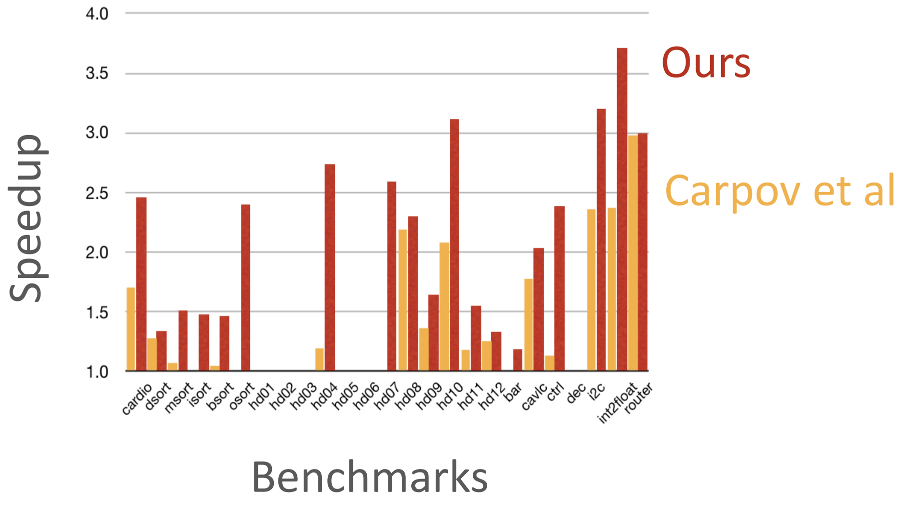
A1. Average 2.05x faster than state-of-the-art method

*Dongkwon Lee et al., Optimizing homomorphic evaluation circuits by program synthesis and term rewriting, PLDI, 2020.

Result

Q1. Effectiveness & Comparison

A1. Average 2.05x faster than state-of-the-art method



*Used leave-one-out cross validation

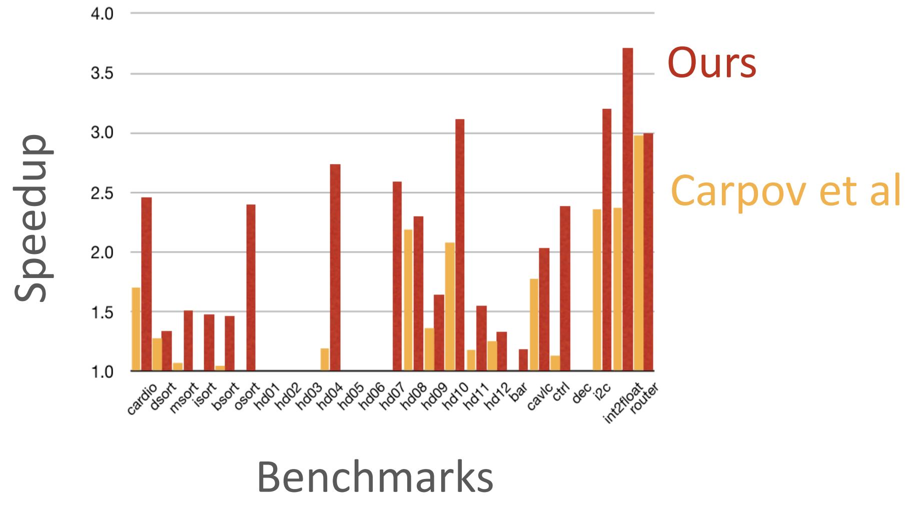
*Dongkwon Lee et al., Optimizing homomorphic evaluation circuits by program synthesis and term rewriting, PLDI, 2020.

Result

Q1. Effectiveness & Comparison

A1. Average 2.05x faster than state-of-the-art method

Average 21.9% reduced depth than state-of-the-art method



*Used leave-one-out cross validation

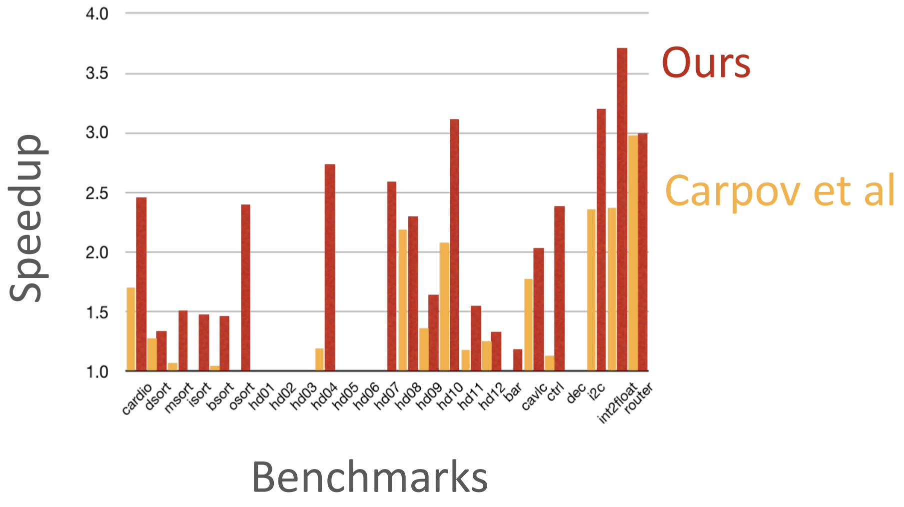
*Dongkwon Lee et al., Optimizing homomorphic evaluation circuits by program synthesis and term rewriting, PLDI, 2020.

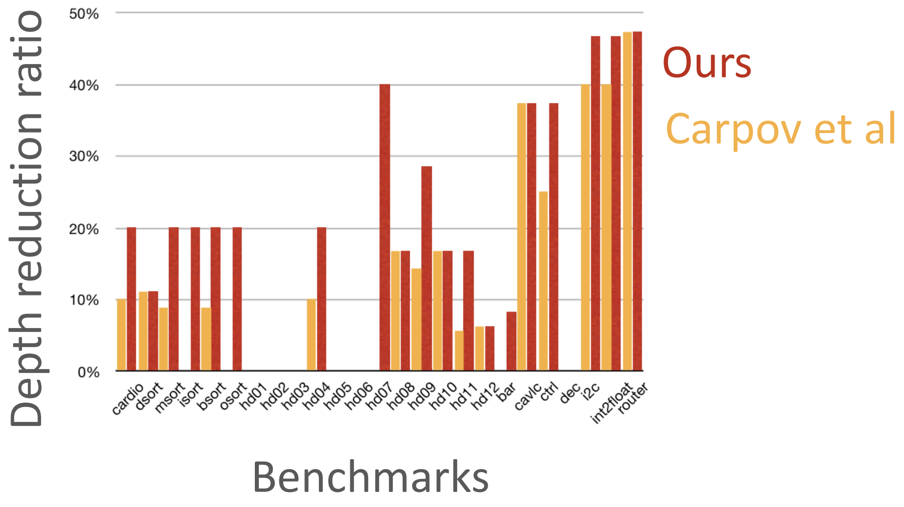
Result

Q1. Effectiveness & Comparison

A1. Average 2.05x faster than state-of-the-art method

Average 21.9% reduced depth than state-of-the-art method





*Used leave-one-out cross validation

*Used leave-one-out cross validation

*Dongkwon Lee et al., Optimizing homomorphic evaluation circuits by program synthesis and term rewriting, PLDI, 2020.

Result

*Dongkwon Lee et al., Optimizing homomorphic evaluation circuits by program synthesis and term rewriting, PLDI, 2020.

Result

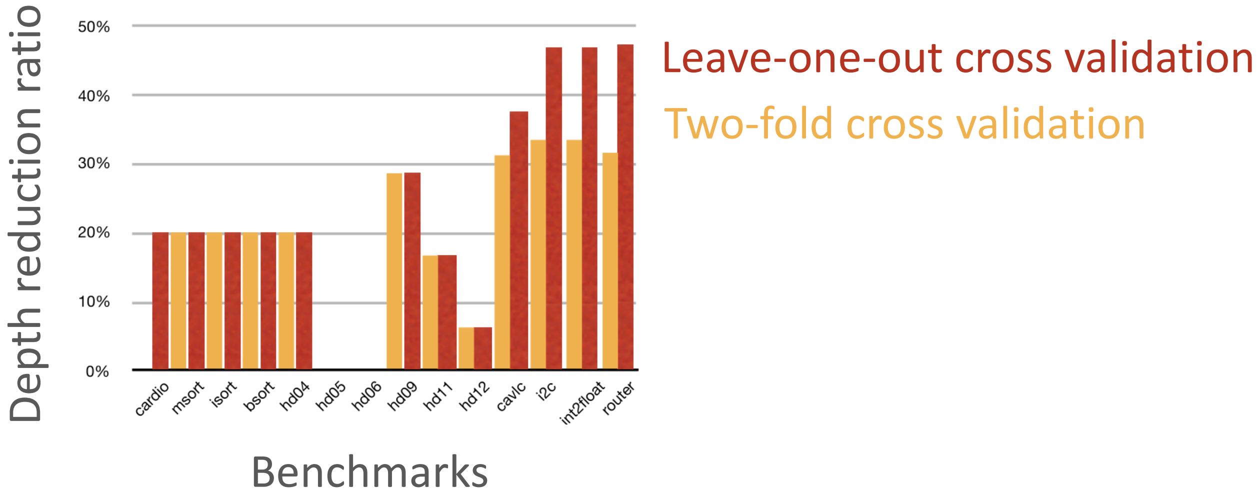
Q2. How is it affected to the selection of training set?
A2. Good performance regardless of selection.

*Dongkwon Lee et al., Optimizing homomorphic evaluation circuits by program synthesis and term rewriting, PLDI, 2020.

Result

Q2. How is it affected to the selection of training set?

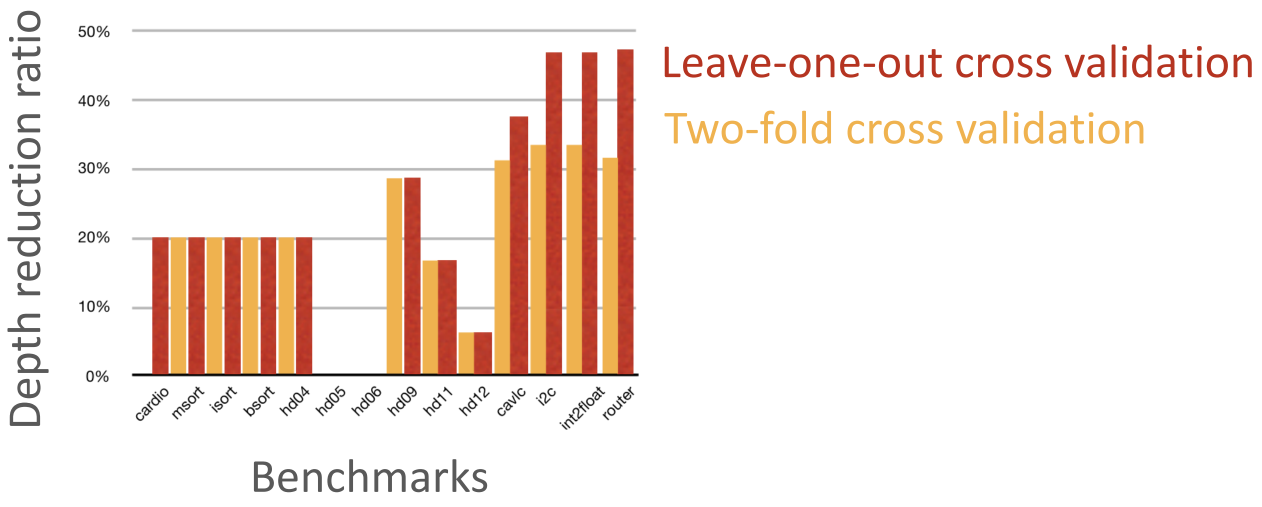
A2. Good performance regardless of selection.



*Dongkwon Lee et al., Optimizing homomorphic evaluation circuits by program synthesis and term rewriting, PLDI, 2020.

Result

Q2. How is it affected to the selection of training set?
A2. Good performance regardless of selection.
So, the rewrite rules (optimizations) are generally applicable



*Dongkwon Lee et al., Optimizing homomorphic evaluation circuits by program synthesis and term rewriting, PLDI, 2020.

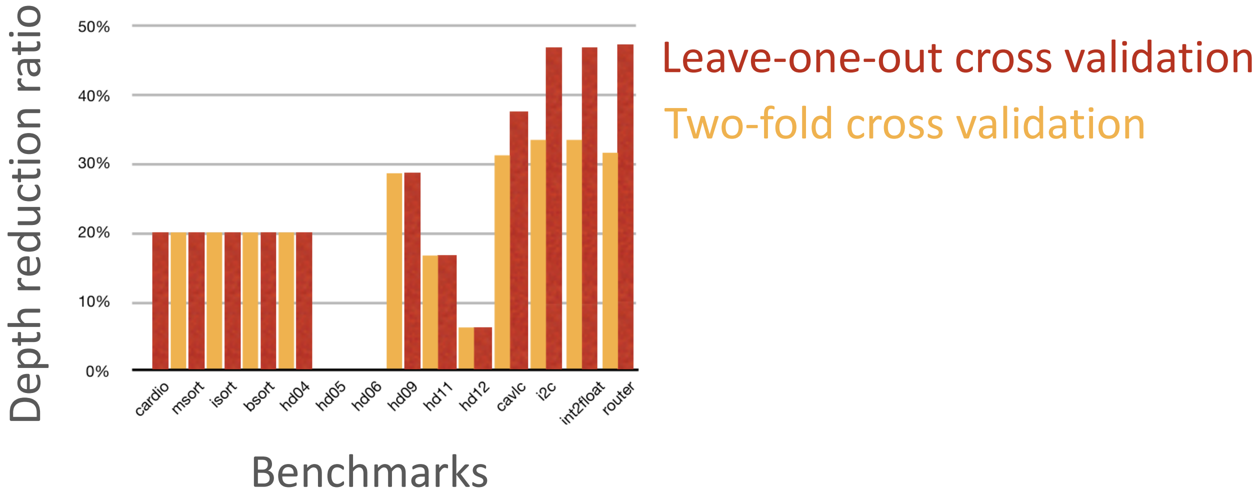
Result

Q2. How is it affected to the selection of training set?

A2. Good performance regardless of selection.

So, the rewrite rules (optimizations) are generally applicable

Learning rewrite rules is meaningful



*Dongkwon Lee et al., Optimizing homomorphic evaluation circuits by program synthesis and term rewriting, PLDI, 2020.

Conclusion

Conclusion

- Fully-homomorphic encryption (FHE) saves privacy in cloud & AI era (now!)
- But it’s slow, hence numerous optimization methods were studied

Conclusion

- Fully-homomorphic encryption (FHE) saves privacy in cloud & AI era (now!)
- But it’s slow, hence numerous optimization methods were studied
- Our novel method is to decrease multiplicative depth

Conclusion

- Fully-homomorphic encryption (FHE) saves privacy in cloud & AI era (now!)
- But it's slow, hence numerous optimization methods were studied
- Our novel method is to decrease multiplicative depth
  - Step 1 - Offline learning with training set
    - Learn rule with each sub-formula of the training set
    - Simplify learnt rule by changing sub-formula to new variable if possible

Conclusion

- Fully-homomorphic encryption (FHE) saves privacy in cloud & AI era (now!)
- But it's slow, hence numerous optimization methods were studied
- Our novel method is to decrease multiplicative depth
  - Step 1 - Offline learning with training set
    - Learn rule with each sub-formula of the training set
    - Simplify learnt rule by changing sub-formula to new variable if possible
  - Step 2 - Online optimization
    - Search for possible substitutions by rewriting

Conclusion

- Fully-homomorphic encryption (FHE) saves privacy in cloud & AI era (now!)
- But it’s slow, hence numerous optimization methods were studied
- Our novel method is to decrease multiplicative depth
  - Step 1 - Offline learning with training set
    - Learn rule with each sub-formula of the training set
    - Simplify learnt rule by changing sub-formula to new variable if possible
  - Step 2 - Online optimization
    - Search for possible substitutions by rewriting
- Resulting circuit 2.05x faster, 21.9% reduced depth than result of SOTA
  - Tested on 25 FHE applications