KU LEUVEN

Leuvenshtein: Efficient FHE-based Edit Distance Computation with Single Bootstrap per Cell

Wouter Legiest[1], Jan-Pieter D'Anvers[1], Bojan Spasic[2], Nam-Luc Tran[2] and Ingrid Verbauwhede[1]

[1]COSIC – KU Leuven [2]Swift

Overview

TFHE

Edit Distance

Equality Checking

Results & Conclusion

Overview

TFHE

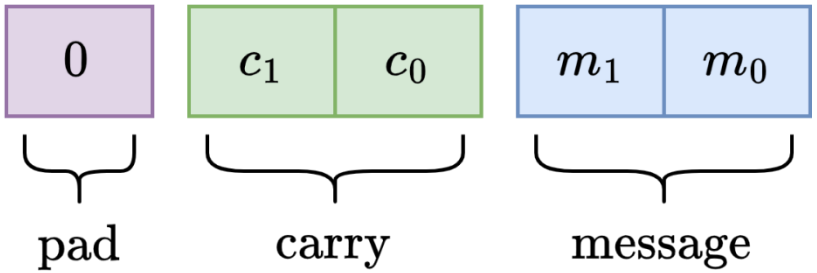
Edit Distance

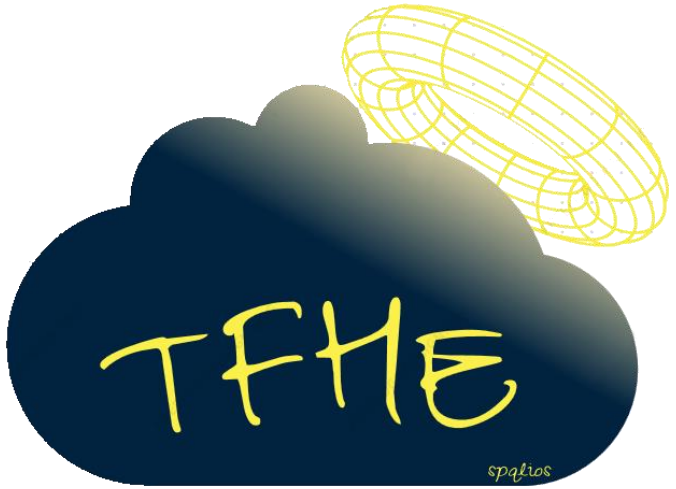
Equality Checking

Results & Conclusion

Torus Fully Homomorphic Encryption (TFHE)

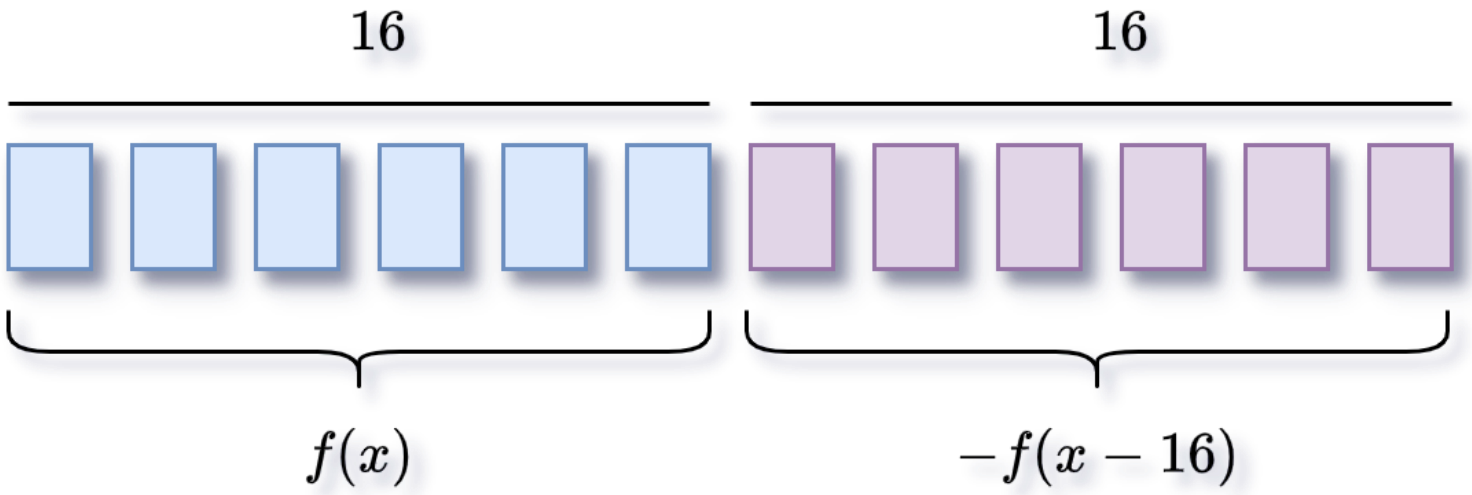
- Boolean-branch FHE





- Using the 4b model
  - Best trade-off: Space and Complexity

TFHE: Negative Look-up Table



- Arbitrary LUT evaluation during bootstrap

Overview

TFHE

EDIT DISTANCE

EQUALITY CHECKING

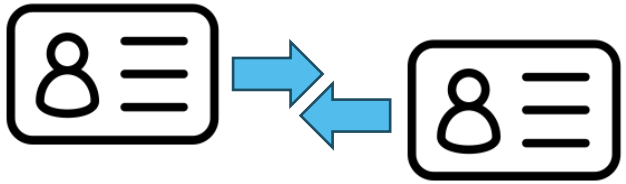
RESULTS & CONCLUSION

String similarity scoring

- Score the similarity between string
  - The lower the score, the more similar

- Basis for:

DNA Matching



Name matching

OCR Correction

String comparison

- How close are two strings?
  - Natural measure
- “John Doe” vs. “Jon Doe”
  - Hamming distance: 6 difference

J	O	H	N		D	O	E
J	O	N		D	O	E	

  - Edit distance: 1 difference

J	O	H	N		D	O	E
J	O		N		D	O	E

Edit distance

- “John Doe” vs. “Jon Doe”
  - Edit distance: 1 difference

J	O	H	N		D	O	E
J	O		N		D	O	E

- Minimal number of operations
  - Deletion
  - Insertion
  - Substitution

J	O	H	N		D	O	E
J	O		N		D	O	O

J	O		N		D	O	E
J	O	H	N		D	E	O

How to compute: Wagner Fisher

- Build up matrix for each substring

		s	a	b	b	a	t	h
	0	1	2	3				
s	1	**0**	1	2				
a	2	1	**0**	1				
b	3	2	1	**0**				
a								
t								
o								
n								

ṣắčắʧộŋ → ṣắččắʧḥ

$$D[i,j] = \begin{cases} D[i-1,j-1] & \text{if } a_i = b_j \\ 1 + \min(D[i-1,j], D[i,j-1], D[i-1,j-1]) & \text{otherwise.} \end{cases}$$

How to compute: Myers

- Idea: difference between each matrix cell is max$\{-1, 0, 1\}$

		S	I	T
	0	1	2	3
K	1	1	2	3
I	2	2	1	2
D	3	3	2	2

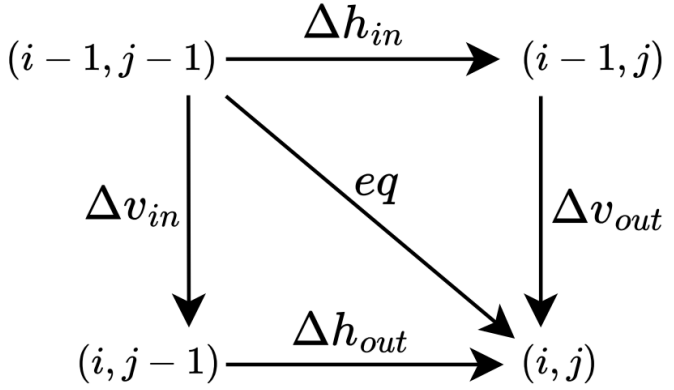
How to compute: Myers

- Idea: difference between each matrix cell is $\max\{-1, 0, 1\}$

$$\Delta v[i,j] = D[i,j] - D[i-1,j],$$
$$\Delta h[i,j] = D[i,j] - D[i,j-1].$$

$$\Delta v_{out} = \min \begin{pmatrix} 1, \\ \Delta v_{in} + 1 - \Delta h_{in}, \\ 1 - \text{EQ} - \Delta h_{in} \end{pmatrix}$$

$$\Delta h_{out} = \min \begin{pmatrix} 1, \\ 1 + \Delta h_{in} - \Delta v_{in}, \\ 1 - \text{EQ} - \Delta v_{in} \end{pmatrix}.$$

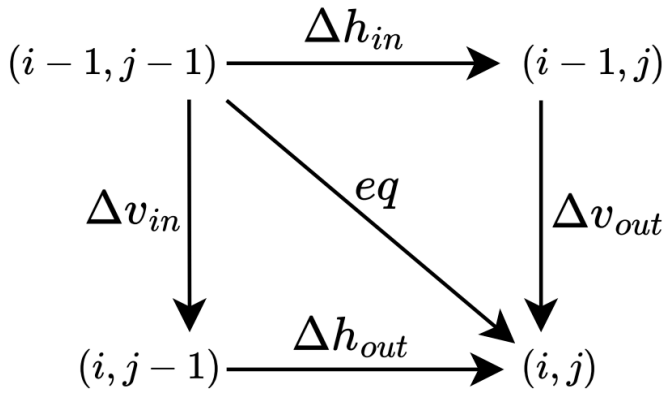


How to compute: Myers

- Idea: difference between each matrix cell is $\max\{-1, 0, 1\}$

$$\Delta v_{out} = \min \begin{pmatrix} 1, \\ \Delta v_{in} + 1 - \Delta h_{in}, \\ 1 - \text{EQ} - \Delta h_{in} \end{pmatrix}$$

$$\Delta h_{out} = \min \begin{pmatrix} 1, \\ 1 + \Delta h_{in} - \Delta v_{in}, \\ 1 - \text{EQ} - \Delta v_{in} \end{pmatrix}.$$



- $\text{EQ} \in \{0, 1\} \quad \Delta h_{in}, \Delta v_{in} \in \{-1, 0, 1\}$
- $\Longrightarrow$ 18 different input
- $\Longrightarrow$ Only have 16 elements in our LUT …

How to compute: Myers

- Use part of the negative space
- Output function: $\mathrm{LUT}_{\min} : 1 + \min(-\mathrm{EQ}, \Delta v_{in}, \Delta h_{in})$

x	Output	x	Output	x	Output	x	Output
0 (0 0000)	0	8 (0 1000)	1	16 (**1** 0000)	0	24 (**1** 1000)	15
1 (0 0001)	0	9 (0 1001)	0	17 (**1** 0001)	0	25 (**1** 1001)	0
2 (0 0010)	0	10 (0 1010)	0	18 (**1** 0010)	0	26 (**1** 1010)	0
3 (0 0011)	0	11 (0 1011)	0	19 (**1** 0011)	0	27 (**1** 1011)	0
4 (0 0100)	1	12 (0 1100)	0	20 (**1** 0100)	15	28 (**1** 1100)	0
5 (0 0101)	1	13 (0 1101)	0	21 (**1** 0101)	15	29 (**1** 1101)	0
6 (0 0110)	0	14 (0 1110)	0	22 (**1** 0110)	0	30 (**1** 1110)	0
7 (0 0111)	1	15 (0 1111)	0	23 (**1** 0111)	15	31 (**1** 1111)	0

$$x = (\Delta v_{in} + 1) + 3 \cdot (1 + \Delta h_{in}) + 9 \cdot \mathrm{EQ}$$

Conclusion

- Still calculate the complete matrix
- Calculate each cell using only one LUT

Overview

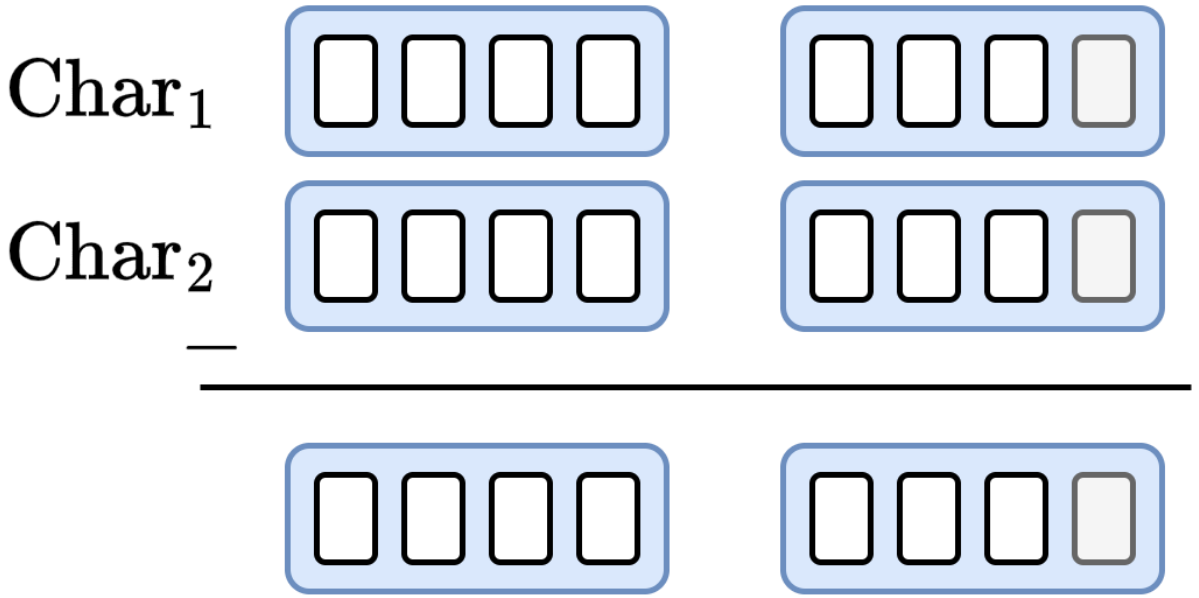
TFHE

Edit Distance

Equality Checking

Results & Conclusion

ASCII: 7 bit Encoding



Overview

TFHE

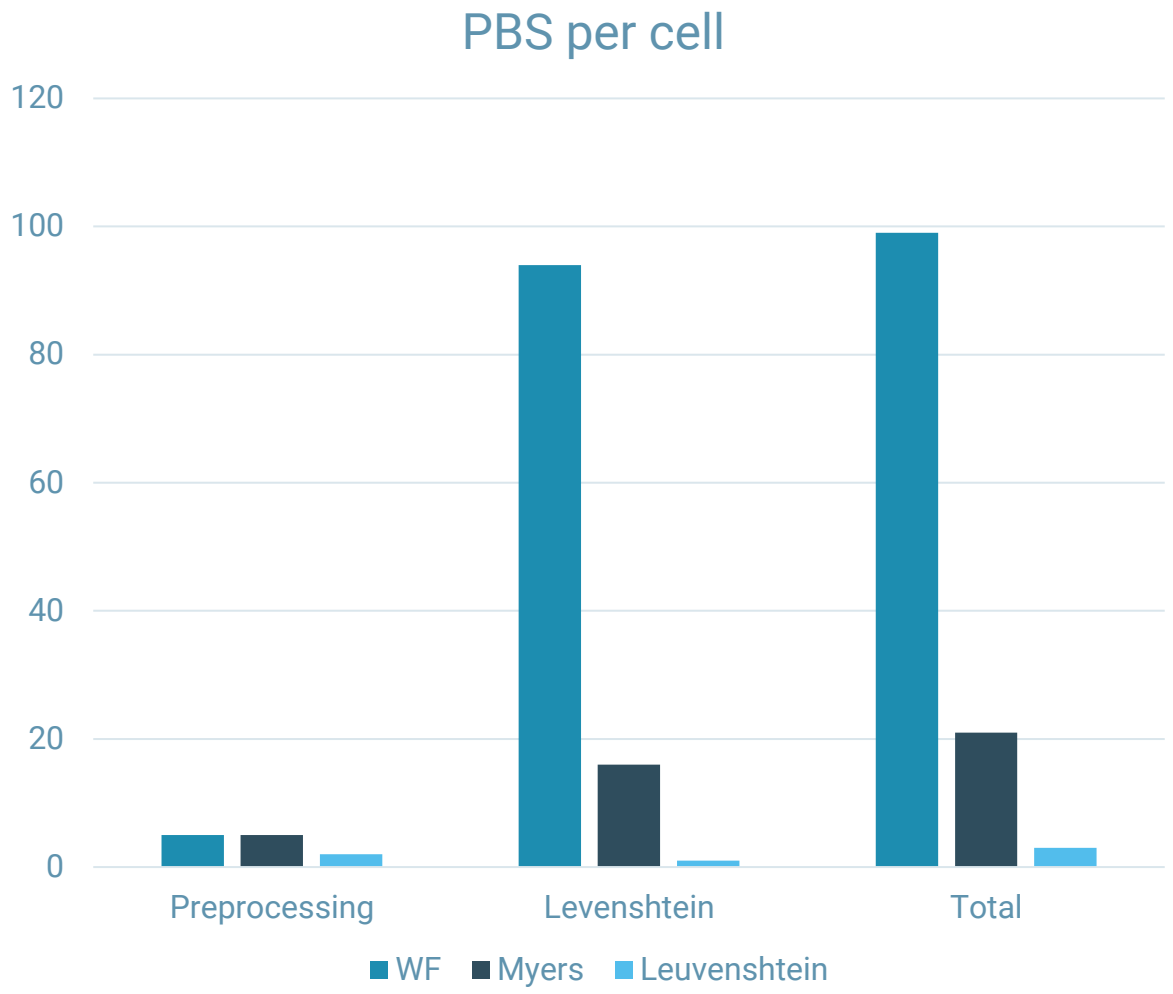
Edit Distance

Equality Checking

Results & Conclusion

Results: Theoretical reduction

- ASCII encoding
- Reducing per cell: $94 \rightarrow 1$ PBS
- Additional optimised equality checking: $5 \rightarrow 2$ PBS
- Reduction of $33\times$ per cell



Result: Practical

	$m = 8$		$m = 100$		$m = 256$	
[CKL15][a]	27.54					
[Zam22a]	241.10	1×	12h 36m	1×	6d 21h[b]	1×
WF	77.59	3.1×	3h 24m	3.7×	22h 19m	7×
Myers	17.81	14×	38m 12s	20×	4h 7m	40×

[a] Using the DGHV scheme, an 80-bit security level and other hardware.
[b] Extrapolated based on 24300 cell calculations.

Conclusion

- $278\times$ faster compared to Concrete compiler
- Still need for manual conversion of algorithms

- Left out of presentation:
  - Skipping part of matrix
  - Optimised equality checking
  - Noise analysis through matrix
  - Comparing unencrypted to encrypted string
  - Approximate calculation