TFHE Parameter Setup and Applications

Jakub Klemsa

EURECOM,
Sophia-Antipolis, France

Mikulášská kryptobesídka
Prague, September 8-9, 2022

Line-Up

1. Introduction to (T)FHE
2. TFHE Parameter Setup
   - Practical FHE?
   - TFHE Parameter Setup
3. TFHE in the Scenes
   - Practical Application: Arithmetics over Encrypted Data
   - Results of Benchmarks

Introduction to (T)FHE

$\Rightarrow$ previous talk

TFHE Parameter Setup

Practical FHE?

Fully Homomorphic Encryption

+ **privacy-preserving** data processing,
+ prospective **applications**: healthcare, finance, ...

? **not deployed** massively
  - (used to be) restrictively impractical.

Vaguely: "**PRACTICAL**" FHE

- design scheme, implement basic functionality,

! most schemes (incl. TFHE): **maaany** parameters & choices,
  - num. parameters, order of operations, alg. choice, ...
  - mutually entangled,

⇒ to make (T)FHE practical, **find The Setup**, the best one.

Practical FHE?

Fully Homomorphic Encryption

+ **privacy-preserving** data processing,
+ prospective **applications**: healthcare, finance, ...
? **not deployed** massively
  - (used to be) restrictively impractical.

Vaguely: "**PRACTICAL**" FHE

- design scheme, implement basic functionality,

! most schemes (incl. TFHE): **maaany** parameters & choices,
  - num. parameters, order of operations, alg. choice, ...
  - mutually entangled,

⇒ to make (T)FHE practical, **find The Setup**, the best one.

Practical FHE?

Fully Homomorphic Encryption

+ **privacy-preserving** data processing,
+ prospective **applications**: healthcare, finance, . . .
? **not deployed** massively
  - (used to be) restrictively impractical.

Vaguely: "**PRACTICAL**" FHE

- design scheme, implement basic functionality,

! most schemes (incl. TFHE): **maaany** parameters & choices,
  - num. parameters, order of operations, alg. choice, . . .
  - mutually entangled,

⇒ to make (T)FHE practical, **find The Setup**, the best one.

Practical FHE?

Fully Homomorphic Encryption

+ **privacy-preserving** data processing,
+ prospective **applications**: healthcare, finance, ...

? **not deployed** massively
  - (used to be) restrictively impractical.

Vaguely: "**PRACTICAL**" FHE

- design scheme, implement basic functionality,

! most schemes (incl. TFHE): **maaany** parameters & choices,
  - num. parameters, order of operations, alg. choice, ...
  - mutually entangled,

⇒ to make (T)FHE practical, **find The Setup**, the best one.

Practical FHE?

Fully Homomorphic Encryption

+ **privacy-preserving** data processing,
+ prospective **applications**: healthcare, finance, ...

? **not deployed** massively

- (used to be) restrictively impractical.

Vaguely: "**PRACTICAL**" FHE

- design scheme, implement basic functionality,

! most schemes (incl. TFHE): **maaany** parameters & choices,

- num. parameters, order of operations, alg. choice, ...
- mutually entangled,

⇒ to make (T)FHE practical, **find The Setup**, the best one.

Practical FHE?

Fully Homomorphic Encryption

+ **privacy-preserving** data processing,
+ prospective **applications**: healthcare, finance, . . .

? **not deployed** massively

- (used to be) restrictively impractical.

Vaguely: "**PRACTICAL**" FHE

- design scheme, implement basic functionality,

! most schemes (incl. TFHE): **maaany** parameters & choices,

- num. parameters, order of operations, alg. choice, . . .
- mutually entangled,

⇒ to make (T)FHE practical, **find The Setup**, the best one.

TFHE Parameter Setup: Objectives

Which setup is "**better**"?

- ? **fast** evaluation:
  - – resource-intensive ⇒ costly (electricity, tailored HW, ...),
- ? **cheap** evaluation:
  - – slow,
- ? error rate, bit-security, plaintext space size, # of additions, ...

⇒ need **cost model** & finding **balance**.

TFHE Parameter Setup: Objectives

Which setup is "**better**"?

- ? **fast** evaluation:
  - – resource-intensive ⇒ costly (electricity, tailored HW, . . .),
- ? **cheap** evaluation:
  - – slow,
- ? error rate, bit-security, plaintext space size, # of additions, . . .

⇒ need **cost model** & finding **balance**.

TFHE Parameter Setup: Objectives

Which setup is "**better**"?

- ? **fast** evaluation:
  - – resource-intensive ⇒ costly (electricity, tailored HW, ...),
- ? **cheap** evaluation:
  - – slow,
- ? error rate, bit-security, plaintext space size, # of additions, ...

⇒ need **cost model** & finding **balance**.

TFHE Parameter Setup: Objectives

Which setup is "**better**"?

- ? **fast** evaluation:
  - – resource-intensive $\Rightarrow$ costly (electricity, tailored HW, ...),
- ? **cheap** evaluation:
  - – slow,
- ? error rate, bit-security, plaintext space size, # of additions, ...

$\Rightarrow$ need **cost model** & finding **balance**.

TFHE Parameter Setup: Assumptions & Inputs

Assumptions:

1. cost model
   ⇒ optimize wrt **fast evaluation**,
2. platform (CPU)
   ⇒ get **expected cost** of operations,
3. order of op's, algorithms, ...
   ⇒ search only **parameters**,

Inputs provided:

- error rate,
- bit-security,
- plaintext space size,
- # of additions.

TFHE Parameter Setup: Assumptions & Inputs

Assumptions:

1. cost model
   - ⇒ optimize wrt **fast evaluation**,
2. platform (CPU)
   - ⇒ get **expected cost** of operations,
3. order of op's, algorithms, ...
   - ⇒ search only **parameters**,

Inputs provided:

- error rate,
- bit-security,
- plaintext space size,
- # of additions.

TFHE Parameter Setup: Assumptions & Inputs

Assumptions:

1. cost model
   $\Rightarrow$ optimize wrt **fast evaluation**,
2. platform (CPU)
   $\Rightarrow$ get **expected cost** of operations,
3. order of op's, algorithms, ...
   $\Rightarrow$ search only **parameters**,

Inputs provided:

- error rate,
- bit-security,
- plaintext space size,
- # of additions.

TFHE Parameter Setup: Assumptions & Inputs

Assumptions:

1. cost model
   - $\Rightarrow$ optimize wrt **fast evaluation**,
2. platform (CPU)
   - $\Rightarrow$ get **expected cost** of operations,
3. order of op's, algorithms, ...
   - $\Rightarrow$ search only **parameters**,

Inputs provided:

- error rate,
- bit-security,
- plaintext space size,
- # of additions.

TFHE Parameter Setup: Finding Parameters

Finding actual TFHE parameters

- given inputs, find **8 parameters**:
  - dimension n, polynomial degree $N \leftrightarrow$ plaintext size, ...

⇒ **technical** optimization task

- Klemsa, J.: *Hitchhiker's Guide to a Practical Automated* TFHE *Parameter Setup*. 1st FHE.org conference '22 (poster),
- experimental tool[1].

Finding inputs?

- more high-level (wrt application),

⇒ exp. results for variety of inputs.

[1] https://gitlab.eurecom.fr/fakub/tfhe-param-testing/

TFHE Parameter Setup: Finding Parameters

Finding actual TFHE parameters

- given inputs, find **8 parameters**:
  - dimension n, polynomial degree $N \leftrightarrow$ plaintext size, ...

⇒ **technical** optimization task
  - Klemsa, J.: *Hitchhiker's Guide to a Practical Automated* TFHE *Parameter Setup*. 1st FHE.org conference '22 (poster),
  - experimental tool[1].

Finding inputs?

- more high-level (wrt application),

⇒ exp. results for variety of inputs.

[1] https://gitlab.eurecom.fr/fakub/tfhe-param-testing/

TFHE Parameter Setup: Finding Parameters

Finding actual TFHE parameters

- given inputs, find **8 parameters**:
  - dimension n, polynomial degree $N \leftrightarrow$ plaintext size, ...

⇒ **technical** optimization task
  - Klemsa, J.: *Hitchhiker's Guide to a Practical Automated* TFHE *Parameter Setup*. 1st FHE.org conference '22 (poster),
  - experimental tool[1].

Finding inputs?

- more high-level (wrt application),

⇒ exp. results for variety of inputs.

[1] `https://gitlab.eurecom.fr/fakub/tfhe-param-testing/`

TFHE Parameter Setup: Implementation & Results

Experimental tool[2] – generate & evaluate TFHE parameters.

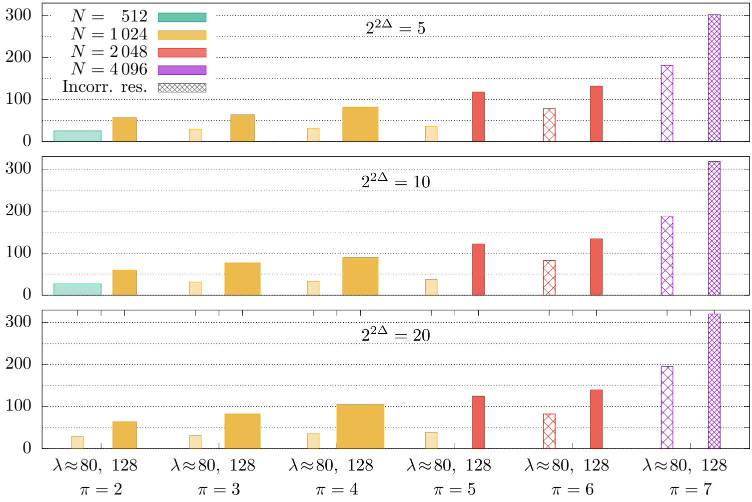


Figure: Bootstrapping times [ms] for various scenarios using Concrete [1].

[2] https://gitlab.eurecom.fr/fakub/tfhe-param-testing

TFHE in the Scenes

Practical Application: Arithmetics over Encrypted Data

Motivation: TFHE param's for parallel arithmetics

- no suitable param's in Concrete `v0.1`
- suitable param's from demo[3] improved by 39% (time),
- comparison of 6 algorithms for parallel addition
  - Klemsa, J., Önen, M.: *Parallel Operations over* TFHE-*Encrypted Multi-Digit Integers*. 12th ACM CODASPY '22 [2].

Current state – Concrete `v0.2-beta`

- different order of operations,
- \+ many hard-coded, optimized parameter sets,
- \+ also implements arithmetics – let's compare

[3] Demo Z/8Z, `https://github.com/zama-ai/demo_z8z`, now obsolete.

Practical Application: Arithmetics over Encrypted Data

Motivation: TFHE param's for parallel arithmetics

- no suitable param's in Concrete `v0.1`
- suitable param's from demo[3] improved by 39% (time),
- comparison of 6 algorithms for parallel addition
  - Klemsa, J., Önen, M.: *Parallel Operations over* TFHE-*Encrypted Multi-Digit Integers*. 12th ACM CODASPY '22 [2].

Current state – Concrete `v0.2-beta`

- different order of operations,
- \+ many hard-coded, optimized parameter sets,
- \+ also implements arithmetics – let's compare

[3] Demo Z/8Z, `https://github.com/zama-ai/demo_z8z`, now obsolete.

Practical Application: Arithmetics over Encrypted Data

Motivation: TFHE param's for parallel arithmetics

- no suitable param's in Concrete `v0.1`
- suitable param's from demo[3] improved by 39% (time),
- comparison of 6 algorithms for parallel addition
  - Klemsa, J., Önen, M.: *Parallel Operations over* TFHE-*Encrypted Multi-Digit Integers*. 12th ACM CODASPY '22 [2].

Current state – Concrete `v0.2-beta`

- different order of operations,
- \+ many hard-coded, optimized parameter sets,
- \+ also implements arithmetics – let's compare 😈

[3] Demo Z/8Z, `https://github.com/zama-ai/demo_z8z`, now obsolete.

Parmesan: Parallel ARithMEticS over tfhe ENcrypted data

Parmesan Library[4]

- based on parallel addition (signed binary repre),
- other op's: scalar mul, mul, squaring, signum, maximum, rounding,
- rewritten for Concrete `v0.2`,

Parmesan (exp.)

- signed, unlimited integers,
- \+ parallelization,
- \+ Karatsuba mul (squaring), addition chains for sc. mul (used in ECC),
- − CRT repres.,
- \+ signum, max, round.

Concrete-integer (beta)

- `uint`-like types,
- − parallelization (lim.),
- − naïve arith. alg's,
- \+ CRT repres. (dev),
- \+ bit operations,

[4] https://github.com/fakub/parmesan

Parmesan: Parallel ARithMEticS over tfhe ENcrypted data

Parmesan Library[4]

- based on parallel addition (signed binary repre),
- other op's: scalar mul, mul, squaring, signum, maximum, rounding,
- rewritten for Concrete `v0.2`,

Parmesan (exp.)

- signed, unlimited integers,
- \+ parallelization,
- \+ Karatsuba mul (squaring), addition chains for sc. mul (used in ECC),
- − CRT repres.,
- \+ signum, max, round.

Concrete-integer (beta)

- `uint`-like types,
- − parallelization (lim.),
- − naïve arith. alg's,
- \+ CRT repres. (dev),
- \+ bit operations,

[4] `https://github.com/fakub/parmesan`

Benchmarks

Hardware – multi-threaded environments:

- exp. server with Intel Core i7-7800X (**12 threads**),
  - EURECOM's machine,
- cluster node with 2×AMD EPYC 7543 (**64** + **64 threads**),
  - operated by e-INFRA CZ[5] (Metacentrum, CESNET).

[5] https://www.e-infra.cz/en

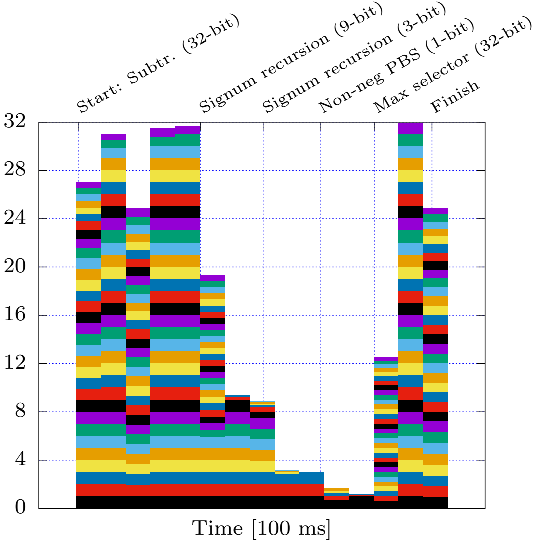
Benchmarks

Hardware – multi-threaded environments:

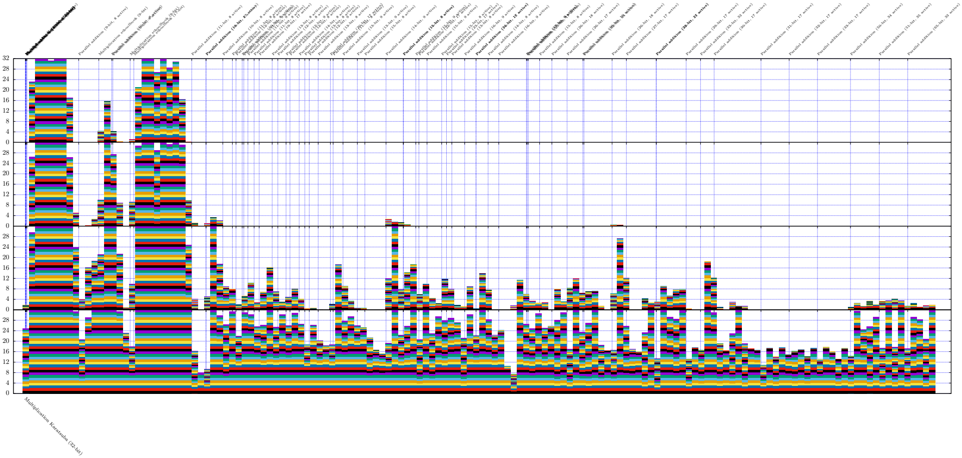
- exp. server with Intel Core i7-7800X (**12 threads**),
  - EURECOM's machine,
- cluster node with 2×AMD EPYC 7543 (**64** + **64 threads**),
  - operated by e-INFRA CZ[5] (Metacentrum, CESNET).

[5] `https://www.e-infra.cz/en`

Processor Load (32-bit Maximum, 32 threads)



Processor Load (32-bit Multiplication)



Full size in Parmesan Library supplementary materials.

Operation	$n =$ #bits	Parmesan Circ. depth	Parmesan #PBS	Parmesan Ideal #thr's	Parmesan Eff. [%]	Parmesan 12-thr. [ms]	Parmesan 128-thr. [ms]	Concrete v0.2 12-thr. [ms]	Concrete v0.2 128-thr. [ms]	Sp.-Up 12-thr.	Sp.-Up 128-thr.
PBS	–	–	–	–	–	110	140	–	–	–	–
Add/Sub (amort. for Concrete)	4	2	$2n$	n	100	220	420	270	480	1.2	1.1
	8					330	400	550	820	1.7	2.1
	16					570	390	1 150	1 310	2.0	3.4
	32					990	460	2 270	2 260	2.3	4.9
Scalar Mul #bits = 16, val's of $k \rightarrow$	4 095	2	$2n$	16	100	580	470	16 340	16 750	28	36
	4 096	0	0	–	–	≈ 0	≈ 0	1 720	1 630	$\approx \infty$	$\approx \infty$
	4 097	2	$2n$	16	100	580	450	1 720	1 600	3.0	3.6
	805	6	114	22	86	1 900	1 420	7 560	7 380	4.0	5.2
	3 195	6	114	22	86	1 890	1 370	10 660	10 390	5.6	7.6
Mul (mod 2^n in Concrete)	4	7	40	16	36	950	1 490	1 230	2 080	1.3	1.4
	8	15	176	64	18	3 330	3 290	4 600	5 390	1.4	1.6
	16	15	725	89	54	10 990	6 770	18 580	18 620	1.7	2.8
	32	25	2 575	≤ 278	≥ 37	38 440	15 260	72 160	72 780	1.9	4.8
Squ (mod 2^n in Concrete)	4	5	32	10	64	740	1 020	1 230	2 020	1.7	2.0
	8	11	138	16	78	2 530	2 430	4 620	5 370	1.8	2.2
	16	19	520	64	43	7 990	5 290	18 560	18 490	2.3	3.5
	32	19	1 901	≤ 217	≥ 46	27 250	10 100	72 200	71 060	2.6	7.0
Signum	32	3	11	8	46	390	600	(not implemented)		–	–
Maximum	32	6	109	32	57	1 880	1 370				
Rounding (at 5^{th} bit)	32	4	56	27	52	1 140	1 090				

Takeaway

FHE is not Sci-Fi,

FHE is here NOW!

Thank you for your attention!

Acknowledgements & References

This research was supported by the MESRI-BMBF French-German joint project UPCARE (ANR-20-CYAL-0003-01).

CONCRETE: Concrete Operates oN Ciphertexts Rapidly by Extending TfhE (v0.1.11).
`https://concrete.zama.ai`, 2021.

Jakub Klemsa and Melek Önen.
Parallel Operations over TFHE-Encrypted Multi-Digit Integers.
In *Proceedings of the Twelveth ACM Conference on Data and Application Security and Privacy*, CODASPY '22, page 288–299, New York, NY, USA, 2022. Association for Computing Machinery.