Faster homomorphic comparison operations for BGV and BFV

Ilia Iliashenko
imec-COSIC, KU Leuven,
Belgium

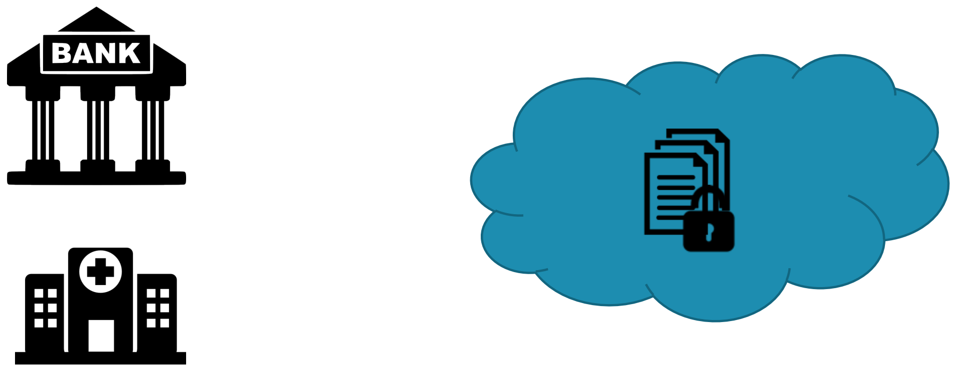
Vincent Zucca
DALI, Université de Perpignan Via Domitia,
LIRMM, Université Montpellier,
France

Privacy Enhancing Technologies Symposium

July 12, 2021

Our data is kept in the cloud

Our data is kept in the cloud



How to work with encrypted data in the cloud?

Security

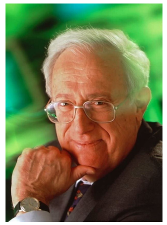
Functionality

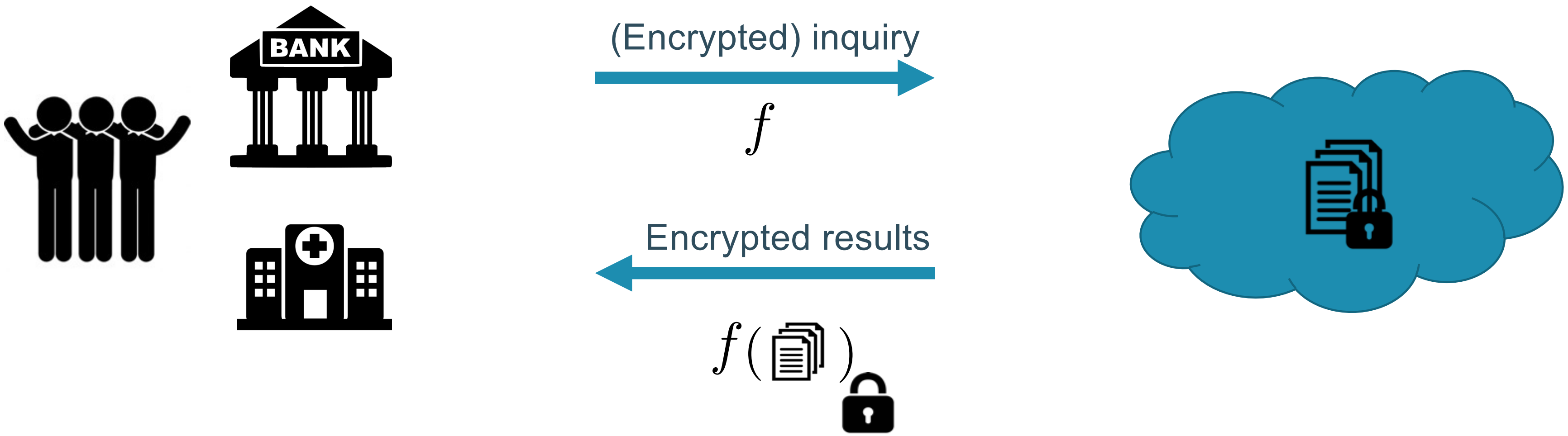
Homomorphic encryption

1978

ON DATA BANKS AND PRIVACY HOMOMORPHISMS

Ronald L. Rivest
Len Adleman
Michael L. Dertouzos





Fully/somewhat homomorhic encryption

$$f\left(Ctxt(m)\right) = Ctxt(f'(m))$$

Fully/somewhat homomorhic encryption

$$f\left(Ctxt(m)\right) = Ctxt(f'(m))$$

- **Fully HE:** $f'(X)$ is any computable function

Fully/somewhat homomorhic encryption

$$f\big(Ctxt(m)\big) = Ctxt(f'(m))$$

- **Fully HE:** $f'(X)$ is any computable function
- **Somewhat HE:** $f'(X)$ is any arithmetic circuit of bounded depth

Fully/somewhat homomorhic encryption

$$f\left(Ctxt(m)\right) = Ctxt(f'(m))$$

- **Fully HE:** $f'(X)$ is any computable function
- **Somewhat HE:** $f'(X)$ is any arithmetic circuit of bounded depth

More efficient in practice

Many useful functions are not arithmetic

- Trigonometric functions
- Sigmoid/step functions
- Comparison functions:
  - logical predicates “is equal”, “is less than”
  - $\max(x, y)$, $\min(x, y)$
  - $\text{argmax}(x_1, \dots, x_n), \text{argmin}(x_1, \dots, x_n)$

Many useful functions are not arithmetic

- Trigonometric functions
- Sigmoid/step functions
- Comparison functions:
  - logical predicates “is equal”, “is less than”
  - $\max(x, y)$, $\min(x, y)$
  - $\operatorname{argmax}(x_1, \dots, x_n), \operatorname{argmin}(x_1, \dots, x_n)$

Computation complexity in HE

Cheap operations	Expensive operations
Plaintext + ciphertext	Ciphertext * ciphertext
Ciphertext + ciphertext	
Plaintext * ciphertext	

Computation complexity in HE

Cheap operations	Expensive operations
Plaintext + ciphertext	Ciphertext * ciphertext
Ciphertext + ciphertext	
Plaintext * ciphertext	

$P(X) = a_0 + a_1 X + a_2 X^2 + \cdots + a_{n-1} X^{n-1}$, a_i's are public.

Computation complexity in HE

Cheap operations	Expensive operations
Plaintext + ciphertext	Ciphertext * ciphertext
Ciphertext + ciphertext	
Plaintext * ciphertext	

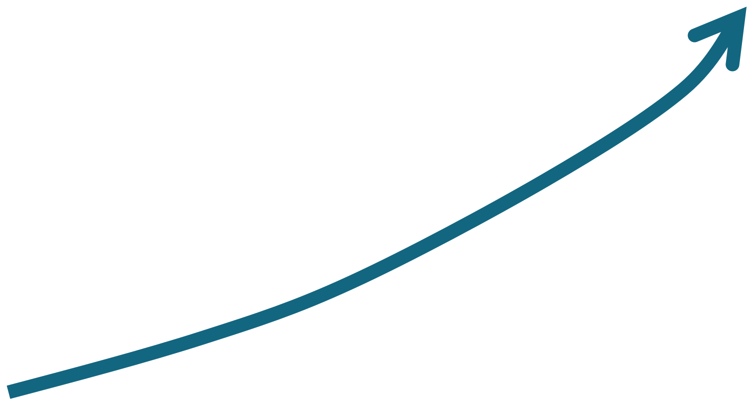
$P(X) = a_0 + a_1 X + a_2 X^2 + \cdots + a_{n-1} X^{n-1}$, a_i's are public.

$P(ctxt) = a_0 + a_1 \cdot ctxt + a_2 \cdot ctxt^2 + \cdots + a_{n-1} \cdot ctxt^{n-1}$

Scalar multiplications are cheap, non-scalar ones are expensive.

Computation complexity in HE

Encryption parameters



Non-scalar multiplicative depth

Context

HE schemes

Arithmetic	HE schemes
Bit-wise	FHEW, TFHE
Integers	BGV, BFV
Approximate (fixed-point)	CKKS/HEAAN

HE schemes

Arithmetic	SHE/FHE schemes
Bit-wise	FHEW, TFHE
Integers	BGV, BFV
Approximate (fixed-point)	CKKS/HEAAN

BGV and BFV can

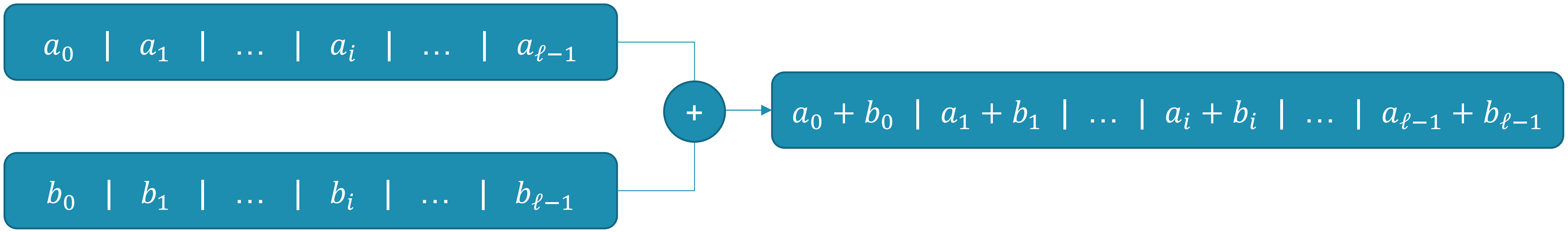
- evaluate arithmetic circuits
- encode data as elements of $\mathbb{F}_{p^d}$

Plaintext space

$$\mathbb{F}_{p^d}^{\ell}$$

Plaintext space

$$\mathbb{F}_{p^d}^{\ell}$$



Parallel (SIMD) operations on ℓ slots!

Possibility to add, multiply, rotate, select the different slots.

Plaintext encoding of large integers

- Decompose an integer a in base $p' \leq p$: $a = \sum a_i p'^i$

a_0	a_1	$\ldots$	a_i	$\ldots$	a_{r-1}

Each a_i is also an element of $\mathbb{F}_p$

Plaintext encoding of large integers

- Decompose an integer a in base $p' \leq p$: $a = \sum a_i p'^i$

a_0 | a_1 | … | a_i | … | a_{r-1}

Each a_i is also an element of $\mathbb{F}_p$

- Every group of d digits can be mapped to an element of $\mathbb{F}_{p^d}$

$\mathbb{F}_p^d$ $\qquad\qquad$ $\mathbb{F}_{p^d}$

a_{id} | … | $a_{i(d+1)-1}$ $\longleftrightarrow$ $b_i = a_{id} + a_{id+1}X + \cdots a_{i(d+1)-1}X^{d-1}$

Plaintext encoding of large integers

- Decompose an integer a in base $p' \leq p$: $a = \sum a_i p'^i$

$a_0 \mid a_1 \mid \ldots \mid a_i \mid \ldots \mid a_{r-1}$

Each a_i is also an element of $\mathbb{F}_p$

- Every group of d digits can be mapped to an element of $\mathbb{F}_{p^d}$

$\mathbb{F}_p^r$ $\quad\longleftrightarrow\quad$ $\mathbb{F}_{p^d}^{r/d}$

$a_0 \mid a_1 \mid \ldots \mid a_{r-1}$ $\quad\longleftrightarrow\quad$ $b_0 \mid b_1 \mid \ldots \mid b_{r/d-1}$

Computations over $\mathbb{F}_p$

Equality function over $\mathbb{F}_p$:

$$\mathrm{EQ}_{\mathbb{F}_p}(x, y) = 1 - (x - y)^{p-1} = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{otherwise} \end{cases}$$

Computations over $\mathbb{F}_p$

Equality function over $\mathbb{F}_p$:

$$\mathrm{EQ}_{\mathbb{F}_p}(x, y) = 1 - (x - y)^{p-1} = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{otherwise} \end{cases}$$

Lagrange Interpolation

Every function $f\colon \mathbb{F}_p^n \to \mathbb{F}_p$ can be interpolated by a unique polynomial of degree at most $p - 1$ in each variable

$$P_f(X_1, \ldots, X_n) = \sum_{\boldsymbol{a} \in \mathbb{F}_p^n} f(\boldsymbol{a}) \prod_{i=1}^{n} \mathrm{EQ}_{\mathbb{F}_p}(X_i, a_i)$$

Homomorphic comparison algorithm [TLW+20]

Input: two encrypted integers $\boldsymbol{x}$ and $\boldsymbol{y}$ encoded into $\mathbb{F}_{p^d}^{\ell}$

x_0	x_1	…	x_i	…	$x_{\ell-1}$

y_0	y_1	…	y_i	…	$y_{\ell-1}$

Homomorphic comparison algorithm [TLW+20]

Input: two encrypted integers $\boldsymbol{x}$ and $\boldsymbol{y}$ encoded into $\mathbb{F}_{p^d}^{\ell}$

$x_0 \mid x_1 \mid \ldots \mid x_i \mid \ldots \mid x_{\ell-1}$ $\quad$ $y_0 \mid y_1 \mid \ldots \mid y_i \mid \ldots \mid y_{\ell-1}$

1. Extract digits from $\mathbb{F}_p$

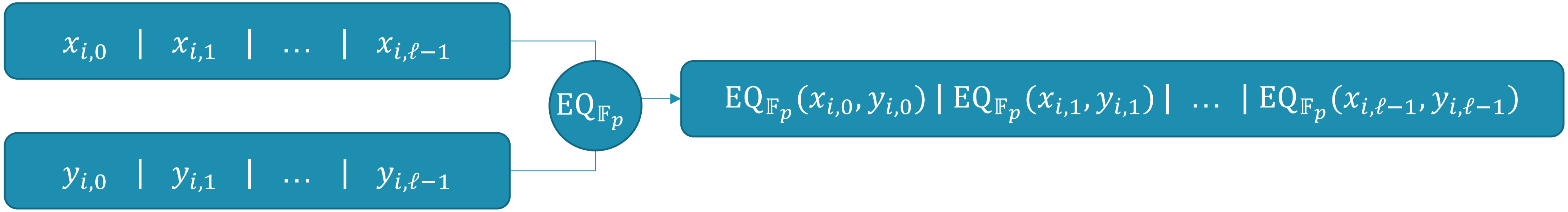
$x_{0,0} \mid x_{1,0} \mid \ldots \mid x_{\ell-1,0}$ $\quad$ $y_{0,0} \mid y_{1,0} \mid \ldots \mid y_{\ell-1,0}$

…

$x_{0,d-1} \mid x_{1,d-1} \mid \ldots \mid x_{\ell-1,d-1}$ $\quad$ $y_{0,d-1} \mid y_{1,d-1} \mid \ldots \mid y_{\ell-1,d-1}$

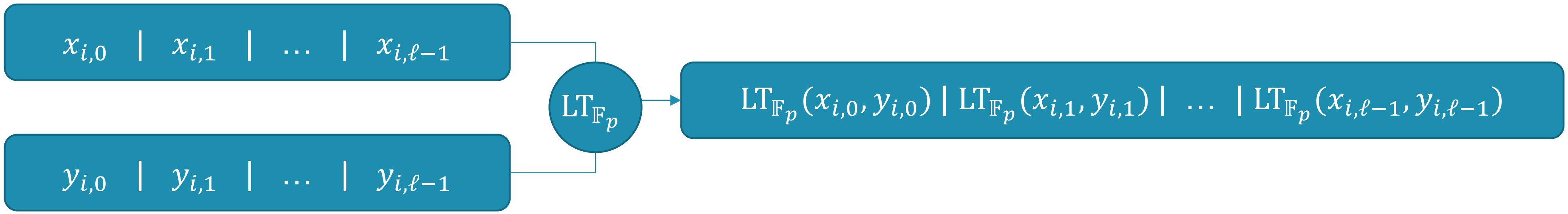
Homomorphic comparison algorithm [TLW+20]

2. Compare corresponding digits by computing the equality function



Homomorphic comparison algorithm [TLW+20]

3. Compare corresponding digits by computing the less-than function



$$\mathrm{LT}_{\mathbb{F}_p}(x, y) = \begin{cases} 1 \text{ if } \; x < y \\ 0 \text{ otherwise} \end{cases}$$

Homomorphic comparison algorithm [TLW+20]

4. Compute the lexicographical order

$$\mathrm{LT}_{\mathbb{F}_p^d}(x_i, y_i) = \sum_{j=0}^{d-1} \mathrm{LT}_{\mathbb{F}_p}\left(x_{j,i}, y_{j,i}\right) \prod_{k=j+1}^{d-1} \mathrm{EQ}_{\mathbb{F}_p}(x_{k,i}, y_{k,i}),$$

$$\mathrm{EQ}_{\mathbb{F}_p^d}(x_i, y_i) = \prod_{j=0}^{d-1} \mathrm{EQ}_{\mathbb{F}_p}(x_{j,i}, y_{j,i})$$

Homomorphic comparison algorithm [TLW+20]

4. Compute the lexicographical order

$$\mathrm{LT}_{\mathbb{F}_p^d}(x_i, y_i) = \sum_{j=0}^{d-1} \mathrm{LT}_{\mathbb{F}_p}\left(x_{j,i}, y_{j,i}\right) \prod_{k=j+1}^{d-1} \mathrm{EQ}_{\mathbb{F}_p}(x_{k,i}, y_{k,i}),$$

$$\mathrm{EQ}_{\mathbb{F}_p^d}(x_i, y_i) = \prod_{j=0}^{d-1} \mathrm{EQ}_{\mathbb{F}_p}(x_{j,i}, y_{j,i})$$

$$\mathrm{LT}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=0}^{\ell-1} \mathrm{LT}_{\mathbb{F}_p^d}(x_i, y_i) \prod_{j=i+1}^{\ell-1} \mathrm{EQ}_{\mathbb{F}_p^d}(x_j, y_j)$$

Contributions

Core part of integer comparison

4. Compute the lexicographical order

$$\mathrm{LT}(x_i, y_i) = \sum_{j=0}^{d-1} \mathrm{LT}_{\mathbb{F}_p}(x_{j,i}, y_{j,i}) \prod_{k=j+1}^{d-1} \mathrm{EQ}_{\mathbb{F}_p}(x_{k,i}, y_{k,i}),$$

$$\mathrm{EQ}(x_i, y_i) = \prod_{j=0}^{d-1} \mathrm{EQ}_{\mathbb{F}_p}(x_{j,i}, y_{j,i})$$

$$\mathrm{LT}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=0}^{\ell-1} \mathrm{LT}(x_i, y_i) \prod_{j=i+1}^{d-1} \mathrm{EQ}(x_j, y_j)$$

How to compute $\mathrm{LT}_{\mathbb{F}_p}(x, y)$: bivariate method

Let $x, y \in [0, p - 1]$.

How to compute $\mathrm{LT}_{\mathbb{F}_p}(x, y)$: bivariate method

Let $x, y \in [0, p-1]$.
$\mathrm{LT}_{\mathbb{F}_p}(x, y)$ is defined by the following lookup table

x \ y	0	1	2	3	…	$p-1$
0	0	1	1	1	…	1
1	0	0	1	1	…	1
2	0	0	0	1	…	1
3	0	0	0	0	…	1
…	…	…	…	…	…	…
$p-1$	0	0	0	0	…	0

How to compute $\mathrm{LT}_{\mathbb{F}_p}(x, y)$: bivariate method

Let $x, y \in [0, p-1]$.
$\mathrm{LT}_{\mathbb{F}_p}(x, y)$ is defined by the following lookup table

x \ y	0	1	2	3	…	$p-1$
0	0	1	1	1	…	1
1	0	0	1	1	…	1
2	0	0	0	1	…	1
3	0	0	0	0	…	1
…	…	…	…	…	…	…
$p-1$	0	0	0	0	…	0

$$P_{\mathrm{LT}_{\mathbb{F}_p}}(X, Y) = \sum_{a=0}^{p-2} \mathrm{EQ}_{\mathbb{F}_p}(X, a) \sum_{b=a+1}^{p-1} \mathrm{EQ}_{\mathbb{F}_p}(Y, b)$$

How to compute $\mathrm{LT}_{\mathbb{F}_p}(x, y)$: bivariate method

Let $x, y \in [0, p-1]$.
$\mathrm{LT}_{\mathbb{F}_p}(x, y)$ is defined by the following lookup table

x \ y	0	1	2	3	…	$p-1$
0	0	1	1	1	…	1
1	0	0	1	1	…	1
2	0	0	0	1	…	1
3	0	0	0	0	…	1
…	…	…	…	…	…	…
$p-1$	0	0	0	0	…	0

$$P_{\mathrm{LT}_{\mathbb{F}_p}}(X, Y) = \sum_{a=0}^{p-2} \mathrm{EQ}_{\mathbb{F}_p}(X, a) \sum_{b=a+1}^{p-1} \mathrm{EQ}_{\mathbb{F}_p}(Y, b)$$

$3p - 5$ non-scalar multiplications [TLW+20]

How to compute $\mathrm{LT}_{\mathbb{F}_p}(x, y)$: univariate method

Let $x, y \in [0, p/2)$ for odd p.

How to compute $\mathrm{LT}_{\mathbb{F}_p}(x, y)$: univariate method

Let $x, y \in [0, p/2)$ for odd p.

$$\mathrm{LT}_{\mathbb{F}_p}(x, y) = \mathrm{IsNegative}_{\mathbb{F}_p}(x - y, 0)$$

$x - y$	$-\frac{p-1}{2}$	…	-3	-2	-1	0	1	2	3	…	$\frac{p-1}{2}$
$\mathrm{IsNegative}_{\mathbb{F}_p}$	1	…	1	1	1	0	0	0	0	…	0

How to compute $\mathrm{LT}_{\mathbb{F}_p}(x, y)$: univariate method

Let $x, y \in [0, p/2)$ for odd p.

$$\mathrm{LT}_{\mathbb{F}_p}(x, y) = \mathrm{IsNegative}_{\mathbb{F}_p}(x - y, 0)$$

$x - y$	$-\frac{p-1}{2}$	…	-3	-2	-1	0	1	2	3	…	$\frac{p-1}{2}$
$\mathrm{IsNegative}_{\mathbb{F}_p}$	1	…	1	1	1	0	0	0	0	…	0

$$Q_{\mathrm{LT}_{\mathbb{F}_p}}(X - Y) = \sum_{a=-(p-1)/2}^{-1} \mathrm{EQ}_{\mathbb{F}_p}(X - Y, a)$$

How to compute $\mathrm{LT}_{\mathbb{F}_p}(x, y)$: univariate method

Let $x, y \in [0, p/2)$ for odd p.

$$\mathrm{LT}_{\mathbb{F}_p}(x, y) = \mathrm{IsNegative}_{\mathbb{F}_p}(x - y, 0)$$

$x - y$	$-\frac{p-1}{2}$	…	-3	-2	-1	0	1	2	3	…	$\frac{p-1}{2}$
$\mathrm{IsNegative}_{\mathbb{F}_p}$	1	…	1	1	1	0	0	0	0	…	0

$$Q_{\mathrm{LT}_{\mathbb{F}_p}}(X - Y) = \sum_{a=-(p-1)/2}^{-1} \mathrm{EQ}_{\mathbb{F}_p}(X - Y, a)$$

$\sqrt{2p - 2} + \mathcal{O}(\log p)$ non-scalar multiplications [PS73,SFR20]

Bivariate method

$$P_{\mathrm{LT}_{\mathbb{F}_p}}(X,Y) = \sum_{a=0}^{p-2} \mathrm{EQ}_{\mathbb{F}_p}(X,a) \sum_{b=a+1}^{p-1} \mathrm{EQ}_{\mathbb{F}_p}(Y,b)$$

Our results:

- $P_{\mathrm{LT}_{\mathbb{F}_p}}(X,Y)$ has total degree p and not $2p-2$

Bivariate method

$$P_{\mathrm{LT}_{\mathbb{F}_p}}(X,Y) = \sum_{a=0}^{p-2} \mathrm{EQ}_{\mathbb{F}_p}(X,a) \sum_{b=a+1}^{p-1} \mathrm{EQ}_{\mathbb{F}_p}(Y,b)$$

Our results:

- $P_{\mathrm{LT}_{\mathbb{F}_p}}(X,Y)$ has total degree p and not $2p-2$
- $P_{\mathrm{LT}_{\mathbb{F}_p}}(X,Y) = Y(X-Y)(X+1)f(X,Y)$

$$f(X,Y) = \sum_{i=0}^{(p-3)/2} f_i(X)Z^i$$

with $Z = Y(X-Y)$ and $\deg f_i(X) = p-3-2i$

Bivariate method

$$P_{\mathrm{LT}_{\mathbb{F}_p}}(X,Y) = \sum_{a=0}^{p-2} \mathrm{EQ}_{\mathbb{F}_p}(X,a) \sum_{b=a+1}^{p-1} \mathrm{EQ}_{\mathbb{F}_p}(Y,b)$$

Our results:

- $P_{\mathrm{LT}_{\mathbb{F}_p}}(X,Y)$ has total degree p and not $2p-2$
- $P_{\mathrm{LT}_{\mathbb{F}_p}}(X,Y) = Y(X-Y)(X+1)f(X,Y)$

$$f(X,Y) = \sum_{i=0}^{(p-3)/2} f_i(X)Z^i$$

with $Z = Y(X-Y)$ and $\deg f_i(X) = p-3-2i$

Non-scalar multiplications:

$2p-6 < 3p-5$ [TLW+20]

Univariate method

$$Q_{\mathrm{LT}_{\mathbb{F}_p}}(X - Y) = \sum_{a=-(p-1)/2}^{-1} \mathrm{EQ}_{\mathbb{F}_p}(X - Y, a)$$

Our results:

- $Q_{\mathrm{LT}_{\mathbb{F}_p}}(X - Y) = \frac{p+1}{2}(X - Y)^{p-1} + (X - Y)g((X - Y)^2)$ with $\deg g = (p - 3)/2$.

Univariate method

$$Q_{\mathrm{LT}_{\mathbb{F}_p}}(X - Y) = \sum_{a=-(p-1)/2}^{-1} \mathrm{EQ}_{\mathbb{F}_p}(X - Y, a)$$

Our results:

- $Q_{\mathrm{LT}_{\mathbb{F}_p}}(X - Y) = \frac{p+1}{2}(X - Y)^{p-1} + (X - Y)g((X - Y)^2)$ with $\deg g = (p - 3)/2$.
- Non-scalar multiplications: $\sqrt{p-3} + \mathcal{O}(\log p) < \sqrt{2p-2} + \mathcal{O}(\log p)$[SFR20]

Univariate method

$$Q_{\mathrm{LT}_{\mathbb{F}_p}}(X-Y) = \sum_{a=-(p-1)/2}^{-1} \mathrm{EQ}_{\mathbb{F}_p}(X-Y,a)$$

Our results:

- $Q_{\mathrm{LT}_{\mathbb{F}_p}}(X-Y) = \frac{p+1}{2}(X-Y)^{p-1} + (X-Y)g((X-Y)^2)$ with $\deg g = (p-3)/2$.
- Non-scalar multiplications: $\sqrt{p-3} + \mathcal{O}(\log p) < \sqrt{2p-2} + \mathcal{O}(\log p)$[SFR20]
- $\mathrm{EQ}_{\mathbb{F}_p}(\mathrm{X},\mathrm{Y}) = 1 - (X-Y)^{p-1}$ is almost for free

Univariate method

$$Q_{\mathrm{LT}_{\mathbb{F}_p}}(X-Y) = \sum_{a=-(p-1)/2}^{-1} \mathrm{EQ}_{\mathbb{F}_p}(X-Y,a)$$

Our results:

- $Q_{\mathrm{LT}_{\mathbb{F}_p}}(X-Y) = \frac{p+1}{2}(X-Y)^{p-1} + (X-Y)g((X-Y)^2)$ with $\deg g = (p-3)/2$.
- Non-scalar multiplications: $\sqrt{p-3} + \mathcal{O}(\log p) < \sqrt{2p-2} + \mathcal{O}(\log p)$[SFR20]
- $\mathrm{EQ}_{\mathbb{F}_p}(\mathrm{X},\mathrm{Y}) = 1 - (X-Y)^{p-1}$ is almost for free

  $\Longrightarrow$ save $\mathcal{O}((d-1)\log p)$ non-scalar multiplications for the lexicographical order!

Min/max

$$\min(x, y) = y + (x - y)\mathrm{LT}(x, y)$$

Min/max

$$\min(x, y) = y + (x - y)\mathrm{LT}(x, y)$$

- use the univariate approach

$$Q_{\min}(X, Y) = \frac{p+1}{2}(X + Y)^{p-1} + g((X - Y)^2)$$

Min/max

$$\min(x, y) = y + (x - y)\mathrm{LT}(x, y)$$

- use the univariate approach

$$Q_{\min}(X, Y) = \frac{p+1}{2}(X + Y)^{p-1} + g((X - Y)^2)$$

- Saves one multiplicative level.
- Same complexity as for evaluating $\mathrm{LT}_{\mathbb{F}_p}$
- Similar method can be applied to evaluate $\mathrm{ReLU}(x) = \max(x, 0)$

Sorting [CDS+15]

Let $A = [a_0, a_1, \dots, a_{N-1}]$ be an array of numbers

$$A = [5,1,7,2]$$

Sorting [CDS+15]

Let $A = [a_0, a_1, \ldots, a_{N-1}]$ be an array of numbers

$$A = [5,1,7,2]$$

1. Compute the comparison matrix $\boldsymbol{L} = \left\{\mathrm{LT}_{\mathbb{F}_p}\left(a_i, a_j\right)\right\}_{i,j}$

$$\boldsymbol{L} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

Complexity: $N(N-1)/2$ homomorphic comparisons

Sorting [CDS+15]

Let $A = [a_0, a_1, \ldots, a_{N-1}]$ be an array of numbers

$$A = [5,1,7,2]$$

1. Compute the comparison matrix $\boldsymbol{L} = \left\{ \mathrm{LT}_{\mathbb{F}_p}(a_i, a_j) \right\}_{i,j}$

$$\boldsymbol{L} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{pmatrix} \xrightarrow{\text{Sum the rows}} M = [2,0,3,1]$$

Sorting [CDS+15]

Let $A = [a_0, a_1, \dots, a_{N-1}]$ be an array of numbers

$$A = [5,1,7,2]$$

1. Compute the comparison matrix $\boldsymbol{L} = \left\{ \mathrm{LT}_{\mathbb{F}_p}(a_i, a_j) \right\}_{i,j}$

$$\boldsymbol{L} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{pmatrix} \xrightarrow{\text{Sum the rows}} M = [2,0,3,1]$$

Positions in the sorted array

Sorting [CDS+15]

Let $A = [a_0, a_1, \ldots, a_{N-1}]$ be an array of numbers

$$A = [5,1,7,2]$$

$$M = [2,0,3,1]$$

2. Select element with index i in the sorted array A_{sorted}

$$\mathrm{EQ}_{\mathbb{F}_p}(M[j], i) \cdot a_j = \begin{cases} a_j \text{ if } M[j] = i \\ 0 \text{ otherwise} \end{cases}$$

Sorting [CDS+15]

Let $A = [a_0, a_1, \ldots, a_{N-1}]$ be an array of numbers

$$A = [5,1,7,2]$$

$$M = [2,0,3,1]$$

2. Select element with index i in the sorted array A_{sorted}

$$\mathrm{EQ}_{\mathbb{F}_p}(M[j], i) \cdot a_j = \begin{cases} a_j \text{ if } M[j] = i \\ 0 \text{ otherwise} \end{cases}$$

$$A_{sorted}[i] = \sum_j \mathrm{EQ}_{\mathbb{F}_p}(M[j], i) \cdot a_j$$

Sorting [CDS+15]

Let $A = [a_0, a_1, \ldots, a_{N-1}]$ be an array of numbers

$$A = [5,1,7,2]$$

$$M = [2,0,3,1]$$

2. Select element with index i in the sorted array A_{sorted}

$$\mathrm{EQ}_{\mathbb{F}_p}(M[j], i) \cdot a_j = \begin{cases} a_j \text{ if } M[j] = i \\ 0 \text{ otherwise} \end{cases}$$

$$A_{sorted}[i] = \sum_j \mathrm{EQ}_{\mathbb{F}_p}(M[j], i) \cdot a_j$$

$$A_{sorted} = [a_1, a_3, a_0, a_2] = [1,2,5,7]$$

Min/max element of array

Let $A = [a_0, a_1, \ldots, a_{N-1}]$ be an array of numbers

- Use the sorting algorithm $\Longrightarrow N(N-1)/2$ homomorphic comparisons ✘

Min/max element of array

Let $A = [a_0, a_1, \dots, a_{N-1}]$ be an array of numbers

- Use the sorting algorithm $\Longrightarrow N(N-1)/2$ homomorphic comparisons ✘
- $N-1$ successive comparisons $\Longrightarrow$ depth too big ✘

Min/max element of array

Let $A = [a_0, a_1, \dots, a_{N-1}]$ be an array of numbers

Use the tournament method to mix both strategies an obtain the best trade-off

a_0

a_1

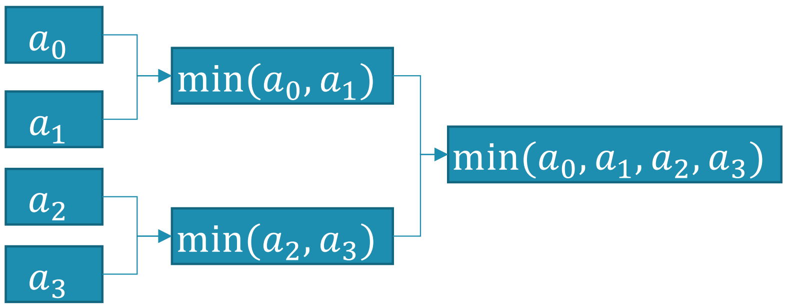
a_2

a_3

Min/max element of array

Let $A = [a_0, a_1, \dots, a_{N-1}]$ be an array of numbers

Use the tournament method to mix both strategies an obtain the best trade-off

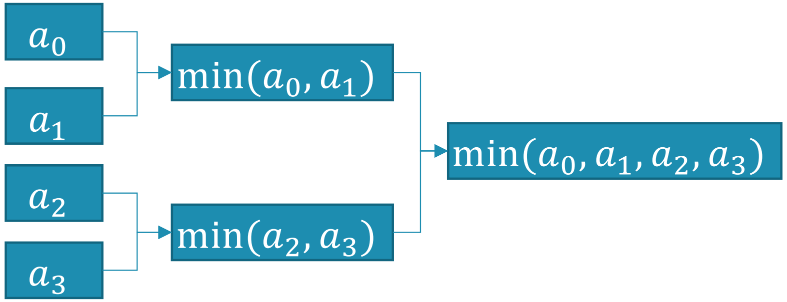


- Use T stages of the tournament method

Min/max element of array

Let $A = [a_0, a_1, \dots, a_{N-1}]$ be an array of numbers

Use the tournament method to mix both strategies an obtain the best trade-off



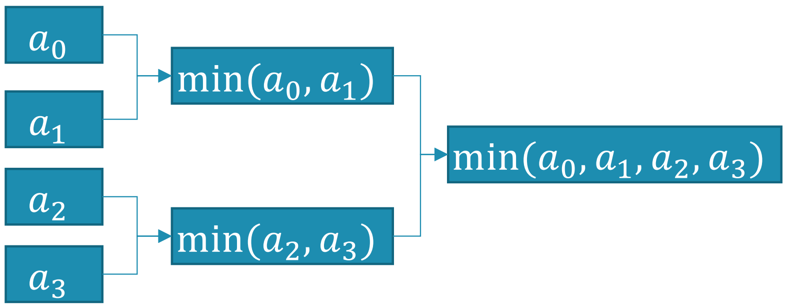
- Use T stages of the tournament method
- Extract the minimum by sorting

  Only need to sort $N' = N/2^T$ elements

Min/max element of array

Let $A = [a_0, a_1, \dots, a_{N-1}]$ be an array of numbers

Use the tournament method to mix both strategies an obtain the best trade-off



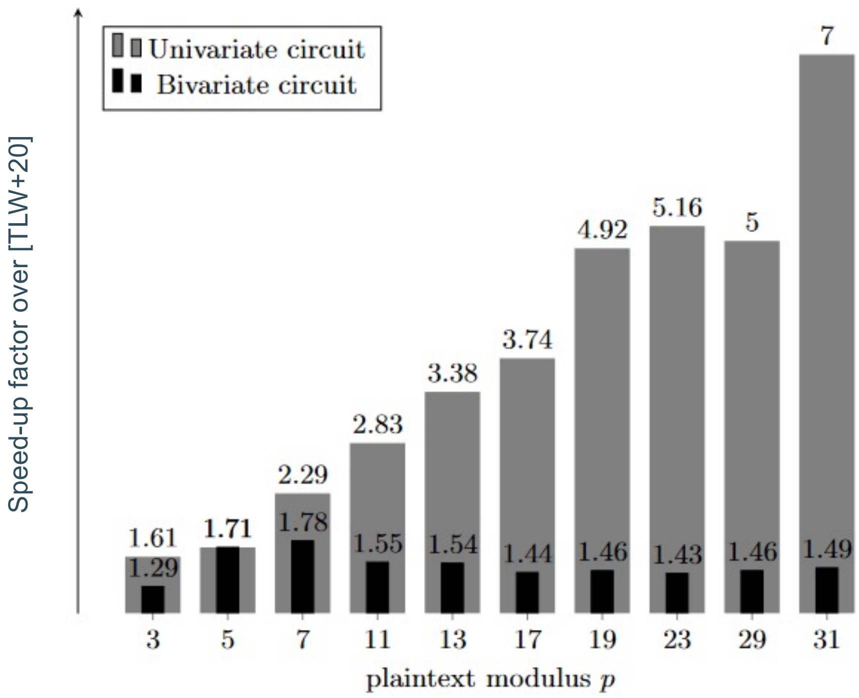
- Use T stages of the tournament method
- Extract the minimum by sorting

  Only need to sort $N' = N/2^T$ elements

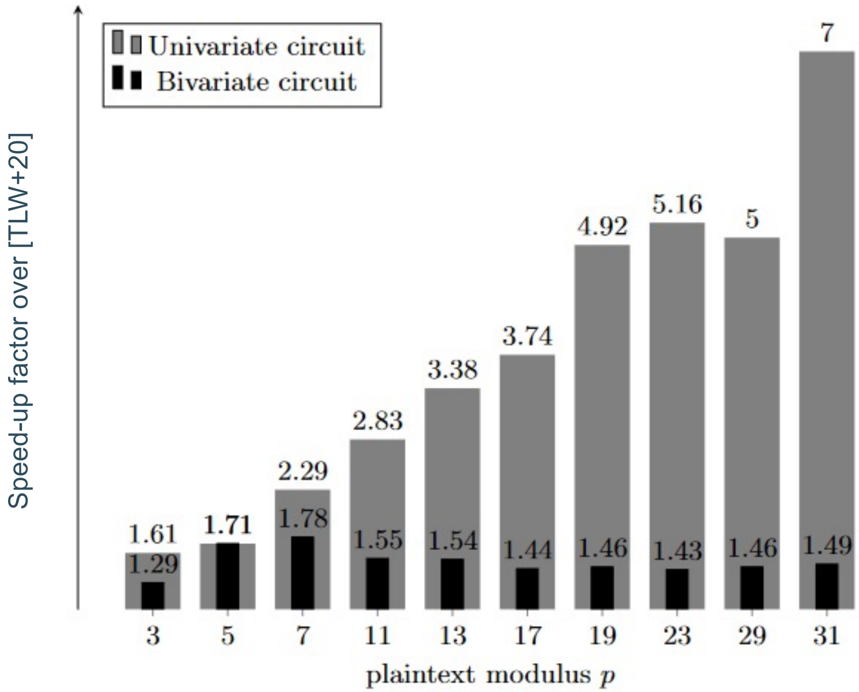
Complexity: $(N - N')$ min + $N'(N' - 1)/2$ less-than functions

Implementation

Less-than function for 64-bits integers



Less-than function for 64-bits integers



Best running time

	Prior work (TLW+20)	This work
p	5	131
Total	24.97s	16.07s
Amortized per integer	36ms	11ms

Sorting N 32-bits integers

N	Total time (s)	Amortized time (ms)	Amortized time [CDS+15]
4	299	64	200
8	1,356	290	944
16	5,700	1,219	4,280
32	23,017	4,922	18,600
64	89,972	19,241	49,700

Time to sort N 32-bits integers with 92 bits of security

Minimum of N 32-bits integers

N	Total time (s)	Amortized time (ms)
2	38	15
4	158	60
8	506	194
16	1,694	649
32	6,440	2,467
64	24,986	9,573

Time to extract the minimum of N 32-bits integers with 121 bits of security

Comparison with other FHE schemes

Bit length	FHE scheme	Bits of Security	Total time (s)	Amortized time (ms)
12	TFHE* CKKS BGV	156 128 126	0.002 127.5 7.09	2.04 1.95 1.23
16	TFHE* CKKS BGV	156 128 126	0.003 297.0 12.11	2.72 4.53 2.10
20	TFHE* CKKS BGV	156 128 126	0.003 373.8 8.66	3.40 5.70 3.01

Timings for the less-than function

* TFHE timings are estimated from [CGG+20]

Conclusions

- Comparison functions over finite fields are fully described.

Conclusions

- Comparison functions over finite fields are fully described.
- We designed 3 times faster circuits to compare 64-bit integers using BGV.
  - Corresponding speed up for sorting and array min/max.

Conclusions

- Comparison functions over finite fields are fully described.
- We designed 3 times faster circuits to compare 64-bit integers using BGV.
  - Corresponding speed up for sorting and array min/max.
- For integer comparison, BGV is
  - as fast as TFHE,
  - slightly faster than CKKS/HEAAN (approximate HE).

Conclusions

- Comparison functions over finite fields are fully described.
- We designed 3 times faster circuits to compare 64-bit integers using BGV.
  - Corresponding speed up for sorting and array min/max.
- For integer comparison, BGV is
  - as fast as TFHE,
  - slightly faster than CKKS/HEAAN (approximate HE).

Future work: other useful functions over rings/fields with effiicient circuits?

Thank you!