Building Algorithm-Hiding FHE Systems from Exotic Number Representations

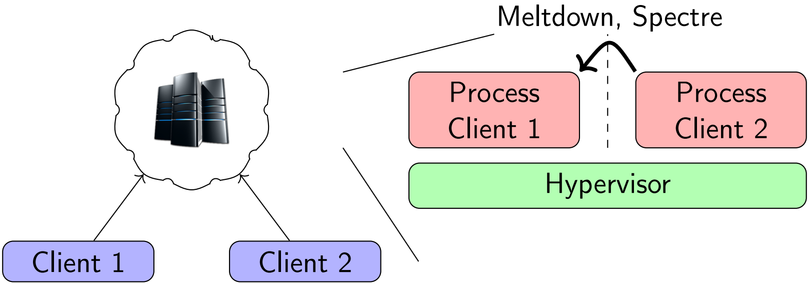
P. Martins[1] L. Sousa[1]

[1]INESC-ID
Instituto Superior Técnico, Univ. Lisboa

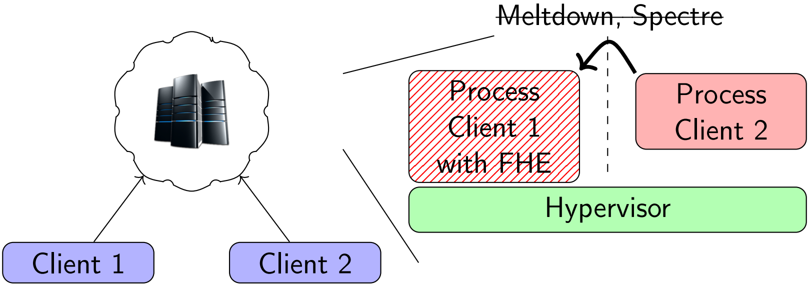
Workshop on Randomness and Arithmetics for Cryptography on Hardware

Table of Contents

Motivation


Meltdown, Spectre
Process Client 1
Process Client 2
Hypervisor
Client 1
Client 2

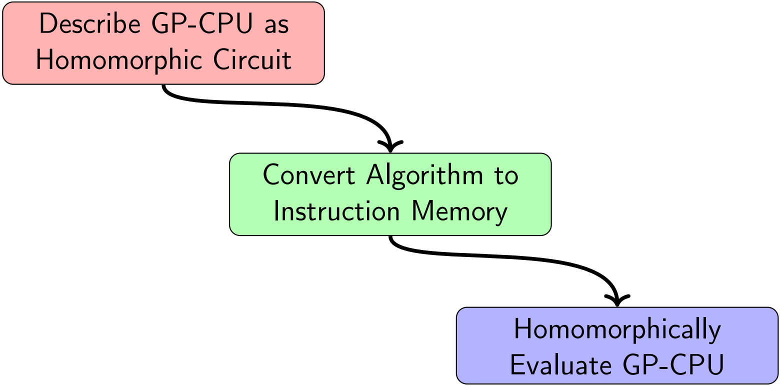
Motivation



- Data disclosure is prevented
- What about algorithm disclosure?

Table of Contents

Solution #1



M. Brenner, J. Wiebelitz, G. von Voigt, M. Smith, Secret program execution in the cloud applying homomorphic encryption, in: IEEE DEST 2011, pp. 114–119. doi:10.1109/DEST.2011.5936608.

Solution #1

- The evaluator does not know which instruction is being executed
- All the CPU circuitry needs to be evaluated at each cycle
- Including memory accesses, ALU operations, etc

Solution #1

- The evaluator does not know which instruction is being executed
- All the CPU circuitry needs to be evaluated at each cycle
- Including memory accesses, ALU operations, etc

$\Rightarrow$ Impractical

BGV

- **Ring**: $R = \mathbb{Z}[X]/(\phi_m(X))$
  $\phi_m(X)$ is a cyclotomic polynomial of degree $\varphi(m)$
- **Ciphertexts**: $\boldsymbol{c_0} + \boldsymbol{c_1} Y \in R_q[Y]$
- **Decryption**: $[\boldsymbol{c_0} + \boldsymbol{c_1}\boldsymbol{s}]_q = [[\boldsymbol{m}]_2 + 2\boldsymbol{v}]_q$
  $\boldsymbol{m} \in R_2$
- **Addition**: $(\boldsymbol{c_0} + \boldsymbol{c_0'}) + (\boldsymbol{c_1} + \boldsymbol{c_1'})Y$
  evaluated at $Y = \boldsymbol{s}$ leads to $\approx [[\boldsymbol{m} + \boldsymbol{m}']_2 + 2(\boldsymbol{v} + \boldsymbol{v}')]_q$

Z. Brakerski, C. Gentry, V. Vaikuntanathan, (Leveled) Fully Homomorphic Encryption Without Bootstrapping, ACM Trans. Comput. Theory 6 (3) (2014) 13:1–13:36

BGV

- **Multiplication**: $(\boldsymbol{c_0} + \boldsymbol{c_1} Y) \times (\boldsymbol{c'_0} + \boldsymbol{c'_1} Y) =$
$$\mathtt{ct}_{mult,0} + \mathtt{ct}_{mult,1} Y + \mathtt{ct}_{mult,2} Y^2$$
evaluated at $Y = \boldsymbol{s}$ leads to $\approx [[\boldsymbol{m} \times \boldsymbol{m}']_2 + 2\boldsymbol{v}'']_q$

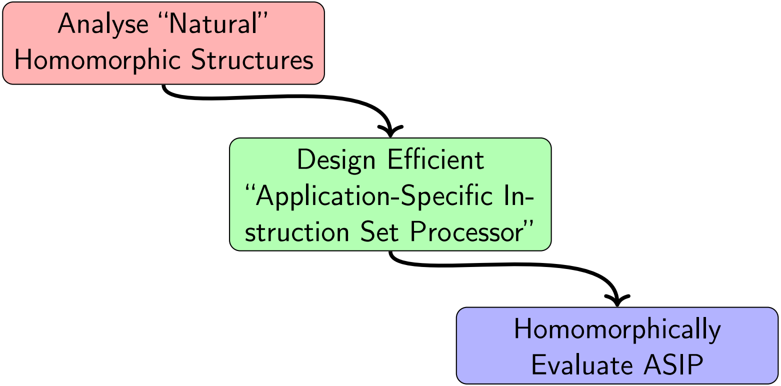
- **Relinearisation**: Multiply $\mathtt{ct}_{mult,2}$ by pseudo-encryption of $\boldsymbol{s}^2$ and add to $(\mathtt{ct}_{mult,0}, \mathtt{ct}_{mult,1})$

- **Modulus-switching**:

$$\begin{aligned}
\boldsymbol{\delta}_i &\leftarrow 2 \cdot [-\mathtt{ct}_{\mathtt{mult},i}/2]_{q/q'} \text{ for } i = 0, 1 \\
\mathtt{ct} &\leftarrow \Big(\left[q'/q \cdot (\mathtt{ct}_{\mathtt{mult},0} + \boldsymbol{\delta}_0)\right]_{q'}, \\
&\qquad \left[q'/q \cdot (\mathtt{ct}_{\mathtt{mult},1} + \boldsymbol{\delta}_1)\right]_{q'} \Big)
\end{aligned}$$

Table of Contents

Proposed Solution



P. Martins, L. Sousa, A methodical FHE-based cloud computing model, in Future Generation Computer Systems, Volume 95, 2019, pp. 639-648, doi:10.1016/j.future.2019.01.046.

"Natural" Homomorphic Structure #1

- Binary plaintext space

$$\mathcal{P} = \mathbb{Z}[X]/(\phi_m(X), 2)$$

with $\phi_m = F_0 \times \ldots \times F_{l-1} \bmod 2$

- Exploit factorisation to encrypt multiple bits in a single ciphertext
- Bits $m_0, \ldots, m_{l-1}$ are encoded as

$$m_i = m(x) \bmod (F_i(x), 2) \, \forall_{0 \leq i < l}$$

- Hom. additions and multiplications operate on them in parallel

“Natural” Homomorphic Structure #1

- Represent $x \in [0,1]$ as $x_0, \ldots, x_{l-1} \in \{0,1\}$ s.t.

$$P(x_i = 1) = x$$

- Batch-encrypt $x_0, \ldots, x_{l-1}$
- Coefficient-wise multiplications and scaled additions

$$z_i = x_i \wedge y_i \Rightarrow z = xy$$

$$z_i = ((1 \oplus s_i) \wedge x_i) \oplus (s_i \wedge y_i) \Rightarrow z = (1-s)x + sy$$

P. Martins, L. Sousa, A Stochastic Number Representation for Fully Homomorphic Cryptography, in: 2017 IEEE SiPS, 2017, pp. 1–6. doi:10.1109/SiPS.2017.8109973.

“Natural” Homomorphic Structure #1

Require: $B(x) = \sum_{i=0}^{d} \binom{d}{i} b_i x^i (1-x)^{d-i}$
Require: x_0
for $i \in \{0, \ldots, d\}$ **do**
    $b_i^{(0)} := b_i$
end for
for $j \in \{1, \ldots, d\}$ **do**
    for $i \in \{0, \ldots, d-j\}$ **do**
        $b_i^{(j)} := b_i^{(j-1)}(1 - x_0) + b_{i+1}^{(j-1)} x_0$
    end for
end for
return $B(x_0) = b_0^{(d)}$

De Casteljau’s algorithm for the evaluation of a polynomial in Bernstein form

"Natural" Homomorphic Structure #2

- Modify BGV with the following decryption

$$[\boldsymbol{c}_0 + \boldsymbol{c}_1 \boldsymbol{s}]_q = [\boldsymbol{m} + \boldsymbol{v}]_q$$

- A number $x \in \mathbb{R}$ is represented as a polynomial

$$\boldsymbol{x} = \lfloor \Delta x \rceil + \boldsymbol{v}$$

- After multiplications, rescale

$$\mathtt{ct} \leftarrow \left(\left[\left\lfloor q'/q \cdot \mathtt{ct}_{\mathtt{mult},0} \right\rceil \right]_{q'}, \left[\left\lfloor q'/q \cdot \mathtt{ct}_{\mathtt{mult},1} \right\rceil \right]_{q'} \right)$$

J. H. Cheon, A. Kim, M. Kim, Y. Song, Homomorphic Encryption for Arithmetic of Approximate Numbers, Cryptology ePrint Archive, Report 2016/421 (2016).

"Natural" Homomorphic Structure #2

Require: $P(x) = \sum_{i=0}^{d} a_i x^i$
Require: x_0

$s := a_d$
for $i \in \{d-1, \ldots, 0\}$ **do**
    $s := a_i + x_0 s$
end for
return $P(x_0) = s$

Horner's method for the evaluation of a polynomial in power form

ASIP Design

- Approximate continuous functions with Bernstein polynomials through Weierstrass theorem
- If necessary, convert Bernstein polynomials to power form
- Factorise multivariate polynomials into univariate polynomials
- Use de Casteljau algorithm or Horner's method

ASIP Design

Approximate continuous functions with Bernstein polynomials through Weierstrass theorem

$$\beta_{f,k_1,\ldots,k_m}^{(n_1,\ldots,n_m)} = f\left(\frac{k_1}{n_1},\ldots,\frac{k_m}{n_m}\right)$$

$$B_f^{(n_1,\ldots,n_m)}(x_1,\ldots,x_m) = \sum_{\substack{0\le k_l\le n_l\\ l\in\{1,\ldots,m\}}} \beta_{f,k_1,\ldots,k_m}^{(n_1,\ldots,n_m)} \prod_{j=1}^{m} \binom{n_j}{k_j} x_j^{k_j}(1-x_j)^{n_j-k_j}$$

ASIP Design

If necessary, convert Bernstein polynomials to power form

$$x_1^{j_1} \ldots x_m^{j_m} = \sum_{k_1=j_1}^{n_1} \frac{\binom{k_1}{j_1}}{\binom{n_1}{j_1}} \binom{n_1}{k_1} x_1^{k_1} (1 - x_1)^{n_1 - k_1} \times$$

$$\ldots \times \sum_{k_m=j_m}^{n_m} \frac{\binom{k_m}{j_m}}{\binom{n_m}{j_m}} \binom{n_m}{k_m} x_m^{k_m} (1 - x_m)^{n_m - k_m} =$$

$$\sum_{\substack{j_l \le k_l \le n_l \\ l \in \{1,\ldots,m\}}} \prod_{h=1}^{m} \frac{\binom{k_h}{j_h}}{\binom{n_h}{j_h}} \binom{n_h}{k_h} x_h^{k_h} (1 - x_h)^{n_h - k_h}$$

ASIP Design

Factorise multivariate polynomials into univariate polynomials

$$B_f^{(n_1,\ldots,n_m)}(x_1,\ldots,x_m) = \sum_{k_1=0}^{n_1}\binom{n_1}{k_1}x_1^{k_1}(1-x_1)^{n_1-k_1}\left(\sum_{k_2=0}^{n_2}\binom{n_2}{k_2}x_2^{k_2}(1-x_2)^{n_2-k_2}\right.$$

$$\left.\ldots\left(\sum_{k_m=0}^{n_m}\beta_{f,k_1,\ldots,k_m}^{(n_1,\ldots,n_m)}\binom{n_m}{k_m}x_m^{k_m}(1-x_m)^{n_m-k_m}\right)\ldots\right)$$

$$P(x_1,\ldots,x_m) = \sum_{k_1=0}^{n_1}x_1^{k_1}\left(\sum_{k_2=0}^{n_2}x_1^{k_2}\ldots\left(\sum_{k_m=0}^{n_m}\alpha_{k_1,\ldots,k_m}^{(n_1,\ldots,n_m)}x_m^{k_m}\right)\ldots\right)$$

Proposed Computing Model

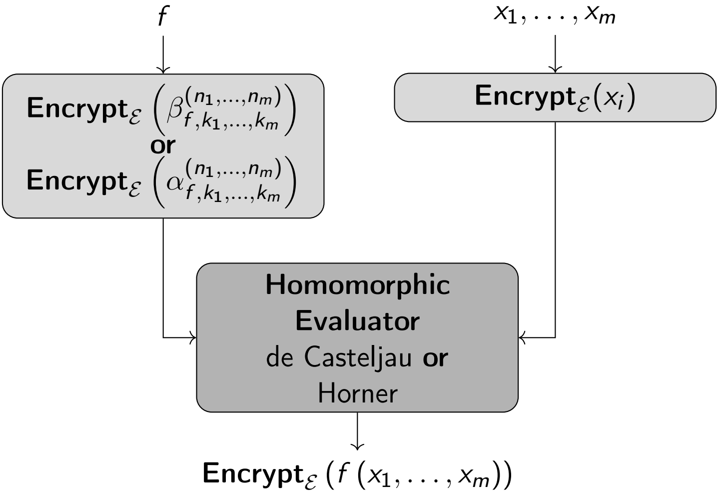


Table of Contents

Example #1

Require: $\boldsymbol{z} \in \mathbb{R}^K$

Sort $(z_1, \ldots, z_K)$ as $(z^{(1)}, \ldots, z^{(K)})$ s.t. $z^{(1)} \geq \ldots \geq z^{(K)}$

$k(\boldsymbol{z}) := \max \left\{ k \in \{1, \ldots, K\} | 1 + kz^{(k)} > \sum_{j \leq k} z^{(j)} \right\}$

$\tau(\boldsymbol{z}) := \frac{\left(\sum_{j \leq k(\boldsymbol{z})} z^{(j)}\right) - 1}{k(\boldsymbol{z})}$

return $\boldsymbol{p}$ s.t. $p_i := \max(0, z_i - \tau(\boldsymbol{z}))$

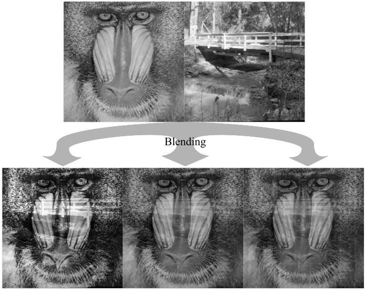
Sparsemax function for mapping scores to probabilities

Example #1

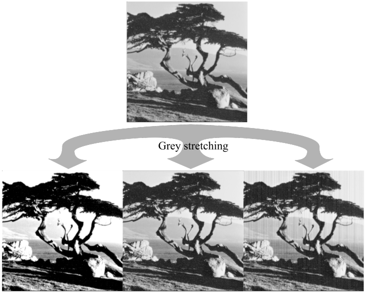
Function	Scheme	# slots	n_1	n_2	m	$\log_2 q$	MAE	Sequential Execution Time [s]	Parallel Execution Time [s]	Speedup
$\text{sparsemax}_1(x_1, 0)$	Fixed-point		5		2^{15}	744	0.0843	0.489	-	-
$\text{sparsemax}_1(x_1, 0)$	Fixed-point		10		2^{15}	744	0.0495	0.689	-	-
$\text{sparsemax}_1(x_1, 0)$	Fixed-point		15		2^{16}	1550	0.0336	9.00	-	-
$\text{sparsemax}_1(x_1, x_2, 0)$	Fixed-point		2	2	2^{15}	744	0.181	0.902	0.543	1.7
$\text{sparsemax}_1(x_1, x_2, 0)$	Fixed-point		3	3	2^{15}	744	0.133	1.57	0.687	2.3
$\text{sparsemax}_1(x_1, x_2, 0)$	Fixed-point		4	4	2^{16}	1550	0.120	20.7	6.87	3.0
$\text{sparsemax}_1(x_1, 0)$	Stochastic	630	5		8191	327	0.104	0.409	0.272	1.5
$\text{sparsemax}_1(x_1, 0)$	Stochastic	1024	10		21845	1440	0.063	16.2	6.40	2.5
$\text{sparsemax}_1(x_1, 0)$	Stochastic	2160	15		55831	2592	0.036	83.0	19.5	4.3
$\text{sparsemax}_1(x_1, x_2, 0)$	Stochastic	630	2	2	8191	327	0.151	0.301	0.254	1.1
$\text{sparsemax}_1(x_1, x_2, 0)$	Stochastic	1024	3	3	21845	985	0.129	9.46	3.58	2.6
$\text{sparsemax}_1(x_1, x_2, 0)$	Stochastic	2160	4	4	55831	2592	0.112	39.6	9.78	4.0

The functions $\text{sparsemax}_1(x_1, 0)$ and $\text{sparsemax}_1(x_1, x_2, 0)$ were approximated and homomorphically evaluated on a i7-5960X, using both a fixed-point approach with Horner's scheme and a stochastic number representation with de Casteljau's algorithm

Example #2



(a) Clear (b) Fixed-point (c) Stochastic



(a) Clear (b) Fixed-point (c) Stochastic

Example #2

System	Encryption [s] Intel / Arm	Filter [s] Intel	Decryption [s] Intel / Arm
Grey Stretching – Fixed-point	52.5 / 685	341	6.9 / 134
Blending – Fixed-point	52.7 / 684	885	5.3 / 88
Grey Stretching – Stochastic	34.5 / 914	1340	61.7 / 1172
Blending – Stochastic	47.7 / 1273	2103	89.4 / 1468
Grey Stretching – Floating-point	324 / 7935	95.9	92.7 / 2630

Average execution time for homomorphic image processing operations on an i7-5960X (Intel) and on a Cortex-A53 (Arm). The last implementation corresponds to an adaption of † to the proposed system. † uses the Paillier cryptosystem

† M. Ziad, A. Alanwar, M. Alzantot, M. Srivastava, CryptoImg: Privacy preserving processing over encrypted images, in: 2016 IEEE CNS, pp. 570–576

Table of Contents

Related Art

Computing Model	Performance	Development Effort	Scope	Privacy
Traditional	Directly exploits CPU architecture	Traditional programming techniques	Supports any application	Vulnerable to attacks like Meltdown and Spectre
PHE libraries	Overhead associated with PHE	Intricate development. Requires strong familiarity with PHE	Limited support of applications	Hides data
FHE w/ application specific circuits	Overhead associated with FHE	Intricate development. Requires strong familiarity with FHE	Supports most applications	Hides data
Proposed model	Limited set of FHE operations	Traditional programming techniques	Continuous functions	Hides data and algorithm
FHE w/ encrypted computer architecture	Impractical	Halting problem may cause development issues	Supports most applications	Hides data and algorithm

Best ▭ Worst

Table of Contents

Conclusion

- Current cloud computing models vulnerable to data and algorithm disclosure
- While FHE prevents data leaking, achieving algorithm secrecy has been impractical so far
- Herein, we focus on a wide range of functions whose approximations can be efficiently evaluated with homomorphic operations
- All approximations are evaluated in the same manner $\Rightarrow$ an evaluator has no way to distinguish them

Thank you!

Any questions?