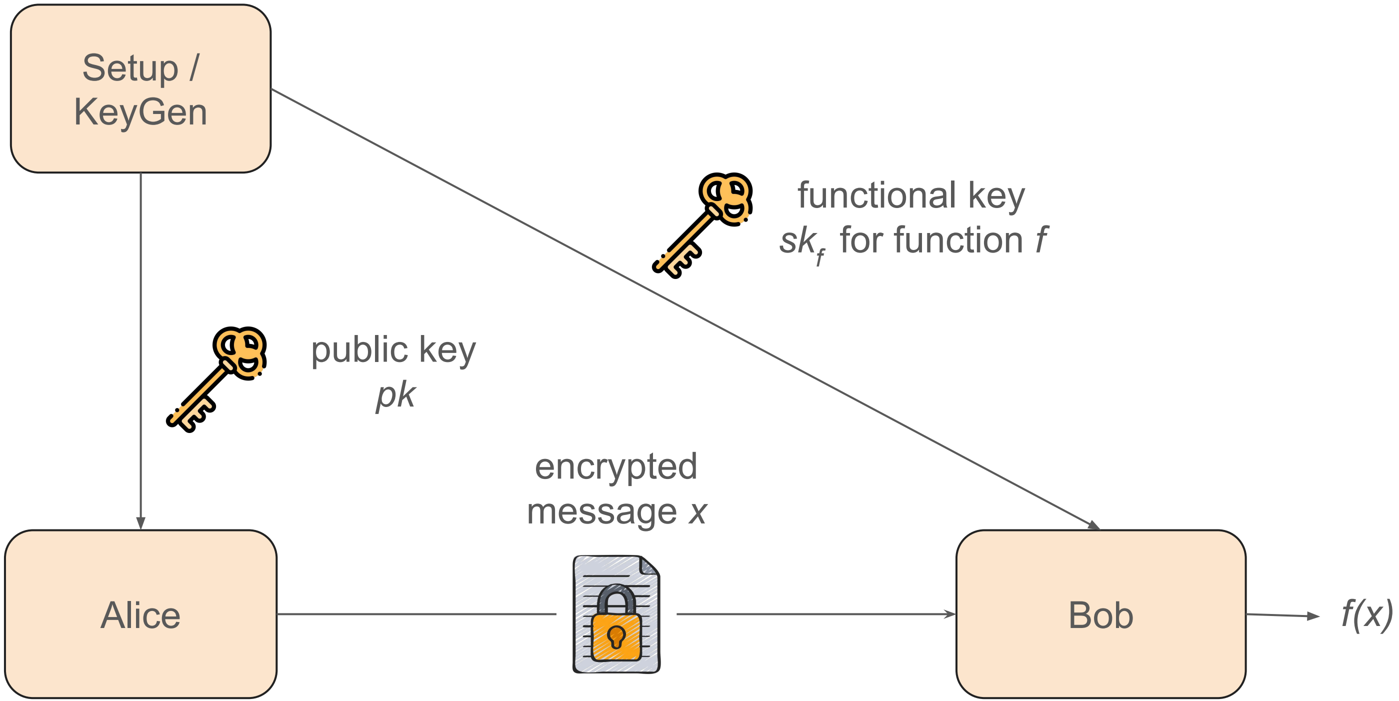
Efficient Lattice-Based Inner-Product Functional Encryption

by Jose Maria Bermudo Mera[1], Angshuman Karmakar[1], **Tilen Marc**[2], and Azam Soleimanian[3]

[1] imec-COSIC, KU Leuven, Leuven, Belgium
[2] Faculty of Mathematics and Physics, University of Ljubljana and XLAB d.o.o., Ljubljana, Slovenia
[3] Equipe Grace, LIX, École Polytechnique, Palaiseau and INRIA, Saclay, France

Functional encryption



Security Notion (informal)

- An adversary should not be able to distinguish between an encryption of arbitrary two messages x^0, x^1
- The encryption should remain indistinguishable even if the adversary has access to functional keys sk_f for any function f for which $f(x^0) = f(x^1)$
- Selective vs. adaptive security

Generality vs. efficiency

- General designs for arbitrary functions, equivalence to iO
- Limited functions, emphasis on efficiency:
    - **inner-product** (linear) functions and quadratic functions
    - security based on well established assumptions: DDH, DCR, **LWE**
    - multi-client setting, decentralization, function hiding

Our work: **Efficient** and practical **RLWE** based (quantumly secure) FE schemes for **inner-product** functions with selective and adaptive security.

In addition: new results on lattices, compiler to (decentralized, identity based) multi-client IPFE, optimized implementation

RLWE based IPFE

- Setup:

$$\mathrm{pk} = (a, \{\mathrm{pk}_i\}), \mathrm{pk}_i = as_i + e_i, a \leftarrow R_q, s_i, e_i \leftarrow D_{\sigma_1}$$

- Encrypt(x):

$$\mathrm{ct}_0 = ar + f, \mathrm{ct}_i = \mathrm{pk}_i r + f_i + \lfloor q/K \rfloor x_i 1_R, r, f \leftarrow D_{\sigma_2}, f_i \leftarrow D_{\sigma_3}$$

- KeyGen(y):

$$\mathrm{sk}_y = \sum y_i s_i$$

- Decrypt:

$$\langle x, y \rangle \lfloor q/K \rfloor 1_R + \mathrm{noise} = (\sum y_i \mathrm{ct}_i) - \mathrm{ct}_0 \mathrm{sk}_y$$

Security challenges

- Adversary should not be able to distinguish an encryption of x^0, x^1 even knowing functional keys for functions with $f(x^0) = f(x^1)$
- Functional keys reveal more information about the underlying RLWE problem than desired

Recall, RLWE problem asks for $a, u \leftarrow \mathcal{R}_q, s \leftarrow \chi, e \leftarrow \chi$ to distinguish

$(a, as + e)$ and (a, u)

A stronger, **multi-hint extended RLWE** is needed, asking to distinguish

$(a, as + e, (r_i, f_i, r_i s + g_i, f_i e + h_i)_{i \in [l]})$ and $(a, u, (r_i, f_i, r_i s + g_i, f_i e + h_i)_{i \in [l]})$

where r_i, f_i, g_i, h_i are sampled from a small distribution

Security challenges

- a simple modification of the scheme for the **adaptive** security

$$\mathrm{pk}_i = \sum_{j=1}^{m} a_j s_{ij}, a_j \leftarrow R_q, s_{ij} \leftarrow D_\sigma$$

- mhe-RLWE not sufficient, need for **Leftover-Hash lemma** in rings and **Complexity Leveraging**

Efficiency and implementation

- We carefully craft the parameters to not lose efficiency
- Ring setting leads to smaller keys, faster operations
- Batching: multiple vectors can be encrypted in parallel allowing SIMD type of calculations on encrypted data

	$\lvert \mathsf{mpk} \rvert$	$\lvert \mathsf{msk} \rvert$	$\lvert \mathsf{ct} \rvert$	$\lvert \mathsf{sk}_f \rvert$
ALS16 [7]	$O(n^2 \log^2 q + \ell n \log q)$	$O(\ell n \log^2 q)$	$O(n \log q^2 + \ell \log q)$	$O(n \log^2 q)$
ABDP15 [4]	$O((n+\ell) n \log^2 q)$	$O(\ell n \log q)$	$O((n+\ell) \log q)$	$O(n \log q)$
RLWE-FE	$O(\ell n \log q)$	$O(\ell n \log q)$	$O(\ell n \log q)$	$O(n \log q)$

	Setup	Encryption	KeyGen	Decryption
ALS16 [7]	$O(\ell n^2 \log q)$	$O(n^2 \log q + \ell n)$	$O(\ell n \log q)$	$O(n \log q + \ell)$
ABDP15 [4]	$O(\ell n^2 \log q)$	$O((\ell + n) n \log q)$	$O(\ell n)$	$O(\ell + n)$
RLWE-FE	$O(\ell n \log n)$	$O(\ell n \log n)$	$O(\ell n)$	$O(\ell n + n \log n)$

Efficiency and implementation

- The primes are chosen to support NTT multiplication and fast modular reduction
- For correctness the primes are required to be very large
  - We provided a CRT based RNS implementation
- Further, we combine Cooley-Tukey and Gentleman-Sande NTT
  - Removes the requirement of rearrangement between NTT and INTT steps
- The Gaussian sampling is implemented in two steps
  - First, samples are generated from a small Gaussian distribution
  - Second, these samples are combined to generate samples from Gaussian distributions of arbitrary standard deviation
  - Both of these steps are performed in constant-time to eliminate timing attacks.

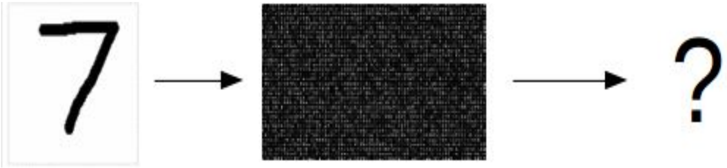
Efficiency and implementation

- An optimized implementation: https://github.com/fentec-project/IPFE-RLWE

Security level	PQ Security	FE Bounds	Gaussian Parameters	Ring Parameters	CRT moduli	Time (ms)
Low	76.3	$B_x : 2$ $B_y : 2$ $\ell : 64$	$\sigma_1 : 33$ $\sigma_2 : 59473921$ $\sigma_3 : 118947840$	$n : 2048$ $\lceil \log q \rceil : 66$	$q_1 : 2^{14} - 2^{12} + 1$ $q_2 : 2^{23} - 2^{17} + 1$ $q_3 : 2^{29} - 2^{18} + 1$	`Setup`:26 `Enc`:16 `KG`:0.27 `Dec`:1
Medium	119.2	$B_x : 4$ $B_y : 16$ $\ell : 785$	$\sigma_1 : 225.14$ $\sigma_2 : 258376412.19$ $\sigma_3 : 516752822.39$	$n : 4096$ $\lceil \log q \rceil : 86$	$q_1 : 2^{24} - 2^{14} + 1$ $q_2 : 2^{31} - 2^{17} + 1$ $q_3 : 2^{31} - 2^{24} + 1$	`Setup`:589 `Enc`:381 `KG`:22 `Dec`:17
High	246.2	$B_x : 32$ $B_y : 32$ $\ell : 1024$	$\sigma_1 : 2049$ $\sigma_2 : 5371330561$ $\sigma_3 : 10742661120$	$n : 8192$ $\lceil \log q \rceil : 101$	$q_1 : 2^{17} - 2^{14} + 1$ $q_2 : 2^{20} - 2^{14} + 1$ $q_3 : 2^{32} - 2^{20} + 1$ $q_4 : 2^{32} - 2^{30} + 1$	`Setup`:1743 `Enc`:1388 `KG`:70 `Dec`:45

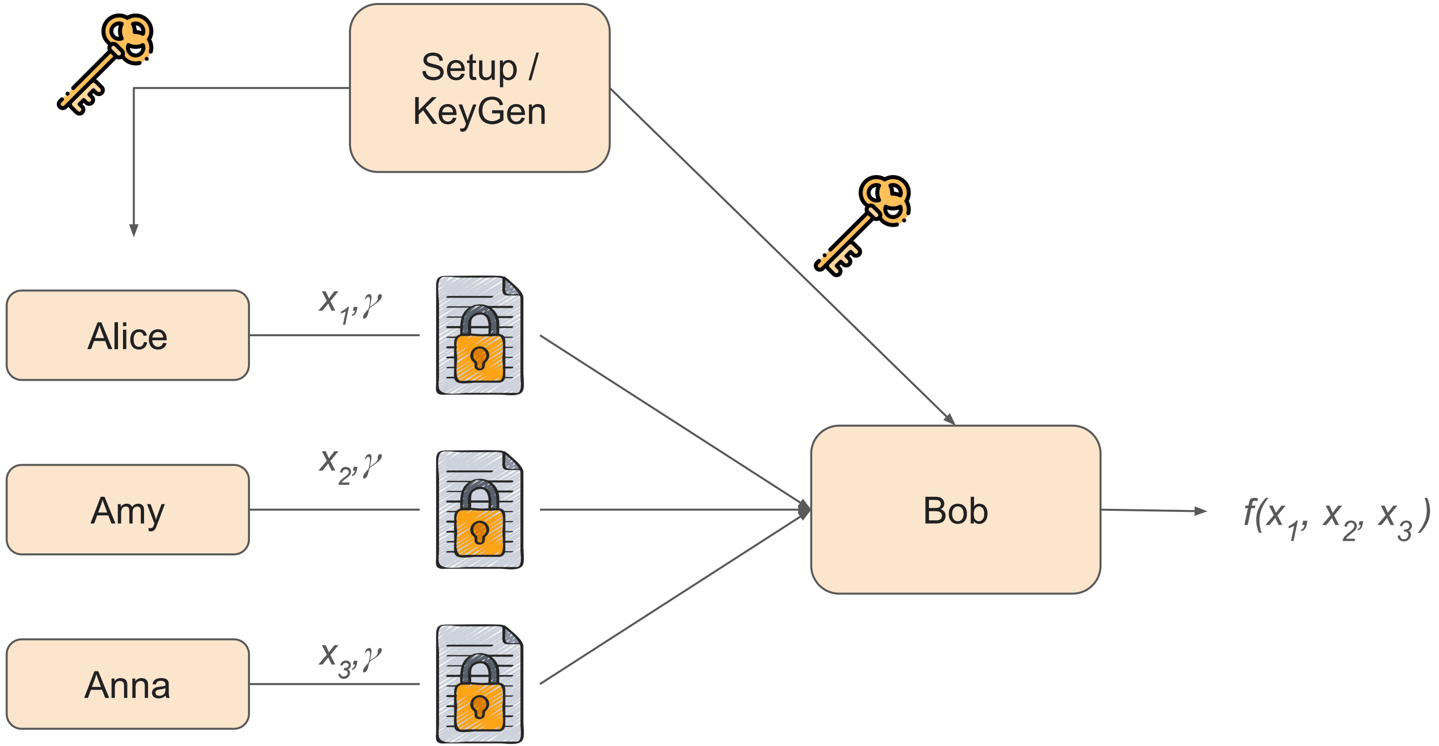
Machine learning on encrypted data

- encrypted images (785-dimensional pixels vectors)



- classifying the content (digit) using logistic regression
- encryption 381ms, model (10 inner-product functions) evaluation 170ms
- batching 4092 images in parallel

(Bonus) Identity based (Decentralized) Multi-Client IPFE



Conclusion

- Efficient RLWE based inner product functional encryption schemes with selective and adaptive security
- Multi-hint extended RLWE problem, Leftover-Hash Lemma for rings
- Compiler to identity-based (decentralized) multi-client scheme
- Optimized implementation in showcase