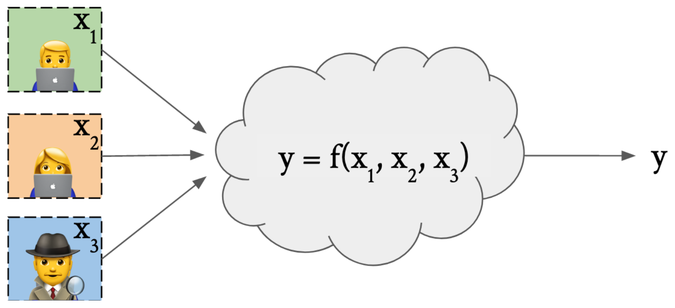
Multiparty homomorphic encryption and the Lattigo library

Christian Mouchet[1], Jean-Philippe Bossuat[2], Juan R. Troncoso-Pastoriza[2] and Jean-Pierre Hubaux[1]

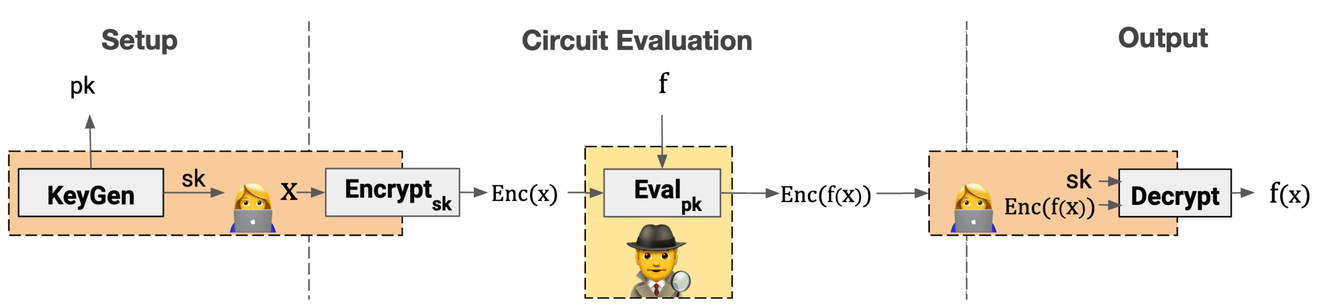
[1]École polytechnique fédérale de Lausanne [2]Tune Insight SA.

The secure multiparty computation problem



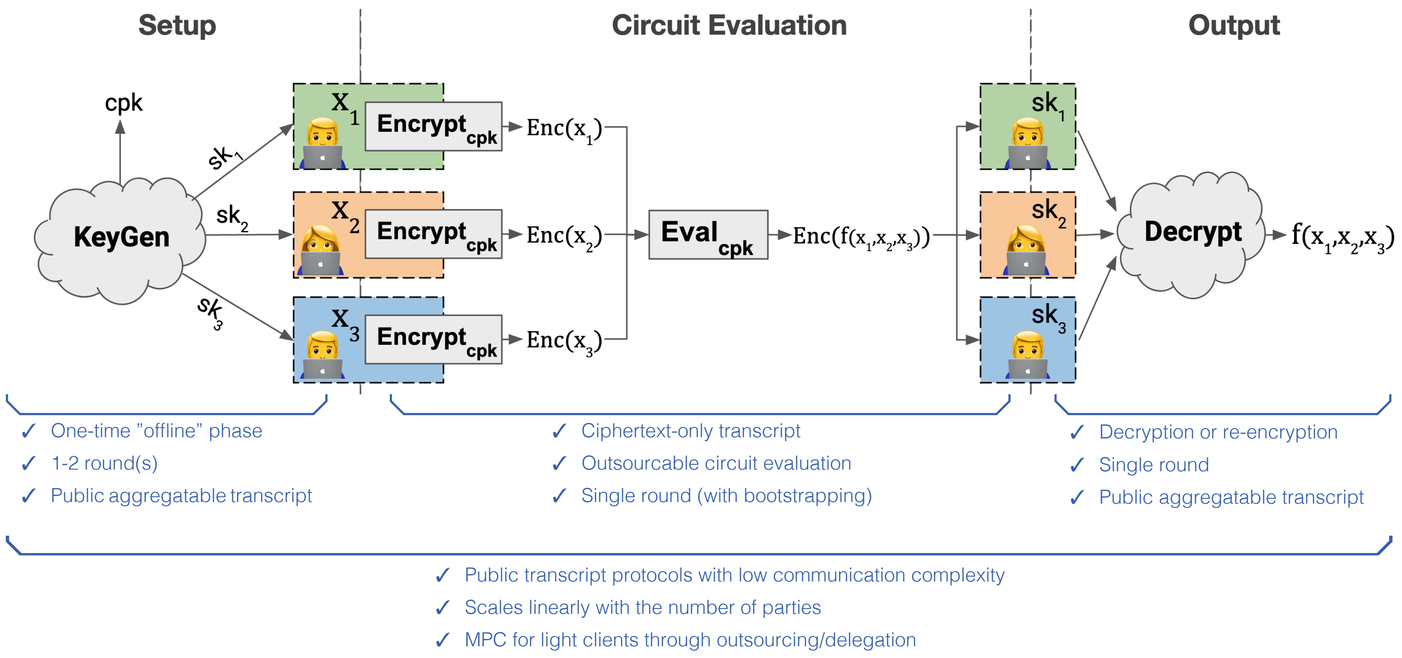
- In the secure multiparty computation (MPC) problem, a group of parties seeks to compute a joint function over their private inputs, without revealing more than the final result.
- The MPC problem is general and can model a broad range of application from privacy-preserving statistics to secure federated learning.
- There exists several cryptographic protocols realizing MPC under a variety of models. But they often come with high performance costs.

Homomorphic encryption



- Homomorphic Encryption (HE) enables computation to be performed over encrypted data, without requiring decryption. This enables private-data processing by untrusted entities and has important applications for outsourced architectures.
- In the last decade, the overhead of HE (w.r.t. plaintext computation) has been reduced from six down to three orders of magnitude. Incoming hardware accelerators are expected to further close this gap in the next few years.
- HE provides a direct solution to the MPC problem for two parties.

Multiparty homomorphic encryption



- Multiparty Homomorphic Encryption (MHE [1,2]) techniques extend traditionnal HE to support N-party MPC.
- These schemes enable encryption of messages in such a way that (1) decryption requires the collaboration between the parties and (2) that homomorphic computation are still possible.
- Hence, the parties can use the MHE scheme to encrypt their inputs to the MPC, compute the joint function under encryption and then collectively decrypt the result.
- MHE-based MPC have several advantages over other techniques:
  - ✓The have low communication complexity.
  - ✓They directly benefit from the research in making HE efficient.
  - ✓They are compatible with the paradigms of cloud computing such as thin-clients delegating heavy computation to honest-but-curious service providers.

The Lattigo library

- ✓100% written in Go, as fast as C++
  - ✓Cross-platform (Linux, Darwin, Windows, WASM, ...)
  - ✓Easy builds and dependency management
- ✓Standalone optimized ring arithmetic layer
- ✓Generic RLWE layer
- ✓Complete HE scheme layer
  - ✓Encrypted integer-arithmetic (BGV [3])
  - ✓Encrypted scale-invariant integer-arithmetic (BFV [4])
  - ✓Encrypted complex/float arithmetic (CKKS [5,6])
  - ✓Parameterizable CKKS bootstrapping [7,8]
  - ✓Homomorphic evaluation of lookup tables
- ✓Multiparty extensions layer
  - ✓Multiparty BFV, BGV and CKKS
  - ✓N-out-of-N-threshold [1]
  - ✓t-out-of-N-threshold [2]

Layered architecture

Multiparty layer
Scheme layer
RLWE layer
Ring layer

lattigo/dbfv & dbgv
- Collective key-switching
- Collective refresh
- Collective masked-transform

lattigo/dckks
- Collective key-switching
- Collective refresh
- Collective masked-transform

lattigo/drlwe
- Collective encryption-key generation
- Collective evaluation keys generation
- Generic collective key-switching
- N-out-of-N-threshold encryption [1]
- T-out-of-N-threshold encryption [2]

lattigo/bfv & bgv
- Encrypted integer arithmetic
- Encoding, encryption
- Homomorphic operations
- Linear-transformations evaluation
- Polynomial evaluation

lattigo/ckks
- Encrypted complex/real arithmetic
- Encoding, encryption
- Homomorphic operations
- Linear-transform evaluation
- Polynomial evaluation

ckks/bootstrapping
CKKS Bootstrappping for dense and sparse secret-keys [7,8]

lattigo/rgsw
- Encryption
- Ciphertext
- External product

rgsw/lut
- Blind rotations
- Lookup table generation
- Lookup table evaluation

lattigo/rlwe
- Full-RNS operations
- Generic RLWE key-generation
- Generic RLWE encryption & decryption
- Generic RLWE gadget decomp. & key-switching
- Generic RLWE relinearization
- Generic RLWE automorphisms
- Standard- <-> conjugate-Invariant-ring bridge
- Ring degree switching
- Coefficient extraction and repacking

lattigo/ring
- Power-of-2 cyclotomic-polynomial ring with RNS coefficient-modulus
- Standard and conjugate-invariant NTTs
- RNS basis extension (mod-up)
- RNS division (mod-down & rescaling)
- Efficient Montgomery representation
- Sampling from uniform, Gaussian and ternary distributions

$ go get Lattigo !

[1] Mouchet, C., Troncoso-Pastoriza, J., Bossuat, J.P. and Hubaux, J.P., 2020. Multiparty homomorphic encryption from ring-learning-with-errors, Proceedings on Privacy Enhancing Technologies, 2021, pp. 291–311
[2] Mouchet, C., Bertrand, E. and Hubaux, J.P., 2022. An Efficient Threshold Access-Structure for RLWE-Based Multiparty Homomorphic Encryption. Cryptology ePrint Archive, 2022/780.
[3] Brakerski, Z., Gentry, C. and Vaikuntanathan, V., 2014. (Leveled) fully homomorphic encryption without bootstrapping. ACM Transactions on Computation Theory (TOCT), 6(3), pp.1-36.
[4] Fan, J., Vercauteren, F., 2012. Somewhat practical fully homomorphic encryption. Cryptology ePrint Arch., 2012/144.
[5] Cheon, J.H., Kim, A., Kim, M. and Song, Y., 2017, Homomorphic encryption for arithmetic of approximate numbers. In International conference on the theory and application of cryptology and information security pp. 409-437. Springer, Cham.
[6] Kim, D. and Song, Y., 2018, November. Approximate homomorphic encryption over the conjugate-invariant ring. In International Conference on Information Security and Cryptology (pp. 85-102). Springer, Cham.
[7] Bossuat, J.P., Mouchet, C., Troncoso-Pastoriza, J. and Hubaux, J.P., 2021. Efficient bootstrapping for approximate homomorphic encryption with non-sparse keys. In Annual International Conference on the Theory and Applications of Cryptographic Techniques, pp. 587-617. Springer, Cham.
[8] Bossuat, J.P., Troncoso-Pastoriza, J. and Hubaux, J.P., 2022. Bootstrapping for approximate homomorphic encryption with negligible failure-probability by using sparse-secret encapsulation. In International Conference on Applied Cryptography and Network Security, pp. 521-541. Springer, Cham.