RSA Conference 2019

San Francisco | March 4 – 8 | Moscone Center

BETTER.

SESSION ID: CRYP-T08 Homomorphic Encryption

New Techniques for Multi-value Input Homomorphic Evaluation and Applications

Sergiu Carpov and Malika Izabachène and Victor Mollimard

Homomorphic Encryption

- Publicly operate on ciphertexts :
  - Correspondence between operations in the encrypted and in the clear domain.

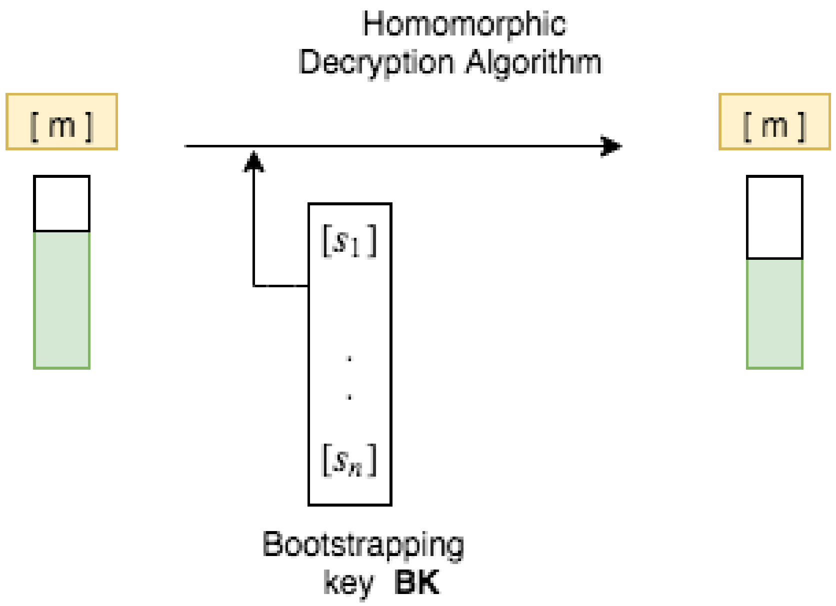
- Fully Homomorphic encryption
  - Allows to evaluate an arbitrary function over encrypted inputs.
  - In particular, Boolean circuits by composing elementary gate operations :

$$[\, b_1 \,] , [\, b_2 \,] \rightarrow \quad [\, b_1 \wedge b_2 \,] \;, [\neg\, b_1 \,] , [\, b_1 \vee b_2 \,]$$

Many applications : Cloud computation, Delegation of computation over sensitive date, Encrypted prediction processing

Somewhat HE to FHE

- Noise growth management using a refreshing technique
- Gentry's Bootstrapping [G09]



Amortized bootstrapping cost per gate is high
Focus on reducing this cost

FHEW-based Fast Bootstrapping

- [AP14] : achieve bootstrapping based on LWE with polynomial noise.
- [DM15] : Gate Bootstrapping for binary gate in ≈ 1sec. + extension.
- [CGGI16]/[CGGI17] : Gate Boostraping for MUX gate in ≈0.1sec.

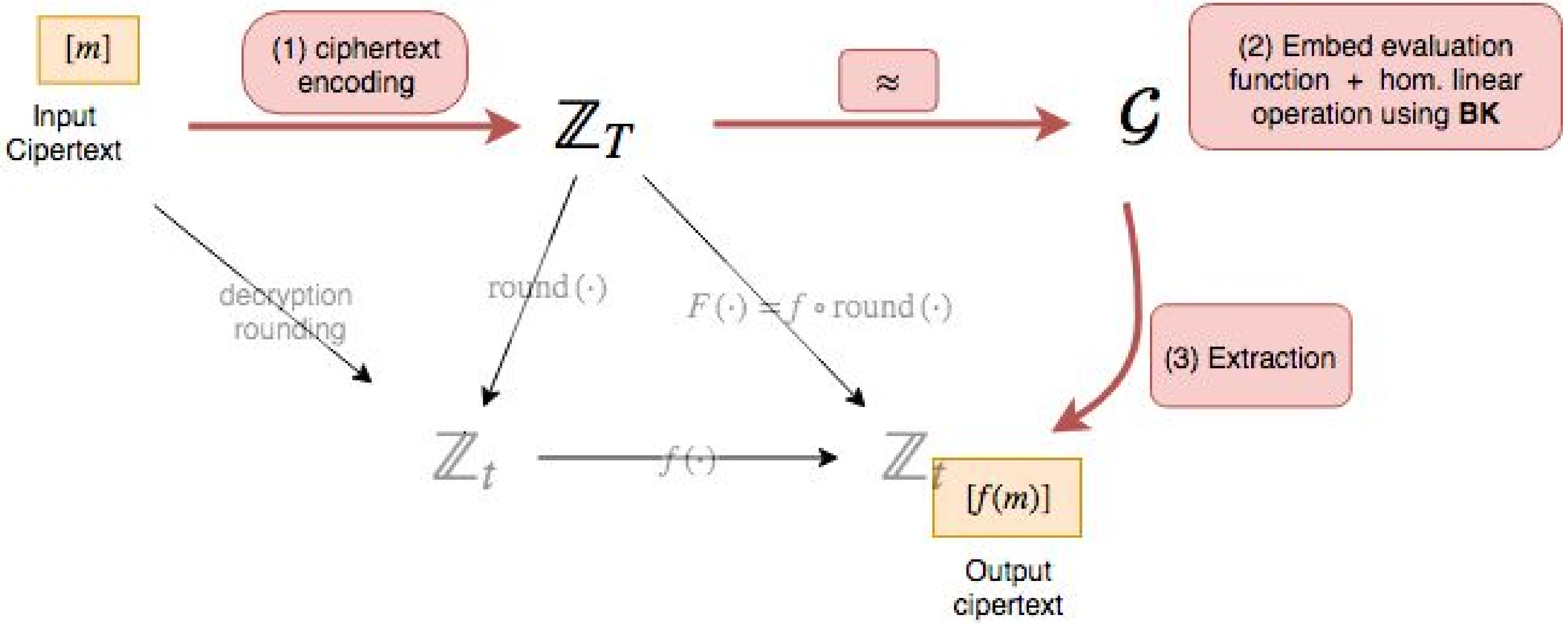
  + arithmetic function via weighted automata.
- [BR15], [BDF18] : extension to larger gates (6-bits input,6-bits output in ≈10sec.).
- [MS18] : improve the amortized bootstrapping cost.
- This work : analysis of the FHEW-based bootstrapping structure.

  optimization of the Bootstrapping for larger gates, application to hom. circuits

  ⇒6-bits input, 6-bits output in ≈1.57sec.

FHEW-based Bootstrapping [DM15]
([BR15],[CGGI16],[BDF18], our work)

Input : a LWE ciphertext of m , description of f, public parameters= (**BK**, …).
Output : a LWE ciphertext of $f(m)$.



TFHE

- T = module of reals modulo 1.
- **Secret key : s** $\in \{0,1\}^n$
- **Encryption :** $c = (\mathbf{a}, b = m + \mathbf{a} \cdot \mathbf{s} + \mathit{noise}) \in T^{n+1}$ with $\mathbf{a} \in T^n$ random.
- **Decryption :** Round $\varphi = b - \mathbf{a} \cdot \mathbf{s}$ to the nearest element in message space.

Learning with errors assumption :

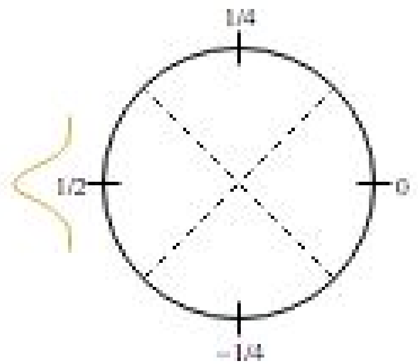
$(\mathbf{a}, b)$ indistinguishable from random in T^{n+1}

TFHE

- T = module of reals modulo 1.
- **Secret key :** $\mathbf{s} \in \{0,1\}^n$
- **Encryption :** $c = (\mathbf{a},\, b = m_i + \mathbf{a} \cdot \mathbf{s} + \textit{noise}) \in T^{n+1}$ with $\mathbf{a} \in T^n$ random.
- **Decryption :** Round $\varphi = b - \mathbf{a} \cdot \mathbf{s}$ to the nearest element in message space.

Example : $\mathcal{M} = \left\{0, \frac{1}{4}, -\frac{1}{4}, \frac{1}{2}\right\} \mathit{mod}\ 1$ and $m = \frac{1}{2}\ \mathit{mod}\ 1$

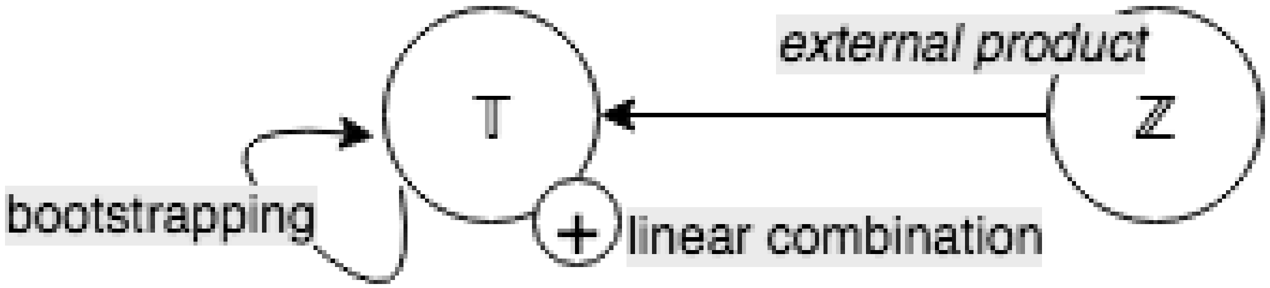
1. Compute $\varphi = m + \textit{noise}$
2. Choose $\mathbf{a} \in T^n$ random
3. Return the ciphertext $(\mathbf{a}, \mathbf{a} \cdot \mathbf{s} + \varphi)$



TFHE Homomorphic Operations

- TLWE Sample : (n+1) torus scalars.
- TRLWE Sample : k+1 torus polynomials of degree N.

- Operations in T : addition, external multiplication with integer elements.



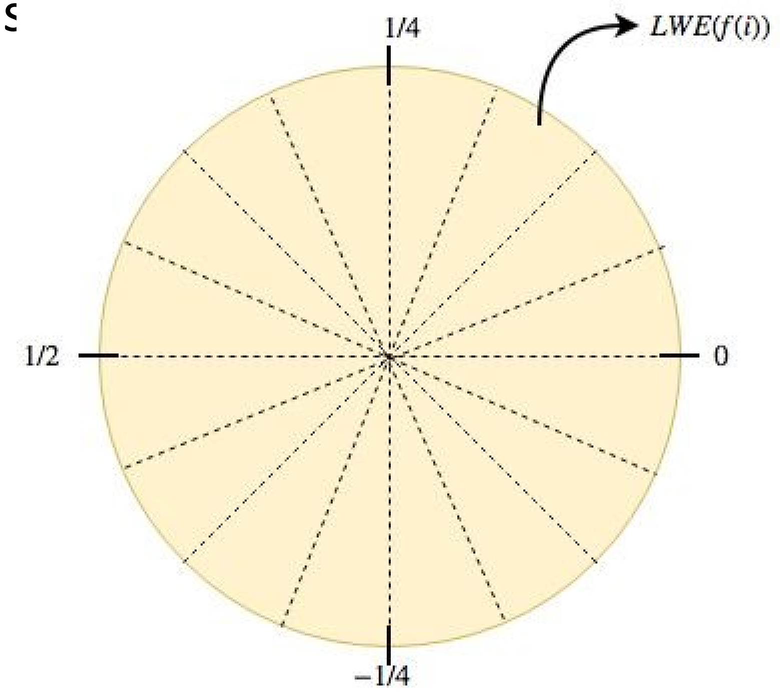
TFHE Bootstrapping for evaluating $f : Z_t \rightarrow Z_t$

Step 1 :

1. Round c=(**a**,b) in a discrete space of size 2N.
2. Encode f as a polynomial TV_F modulo X^N+1 where $f = \mathrm{F} \circ$ round.

Step 2 :

1. Homomorphically rotate the polynomial by b- **a** · **s** positions



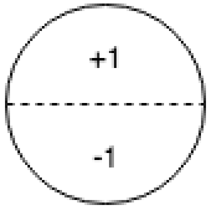
Step 3 :

1. Extract the constant term which encrypts $f(m)$.
2. Switch the ciphertext back to the original key.

Mutli-value Bootstrapping – Test Polynomial Factorization

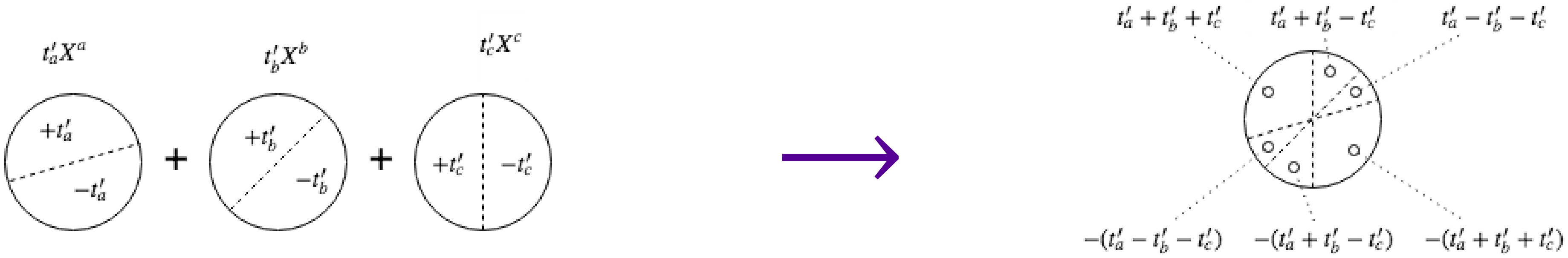
- First-phase test polynomial : divides the torus circle in two parts.

$$TV^{(0)}$$

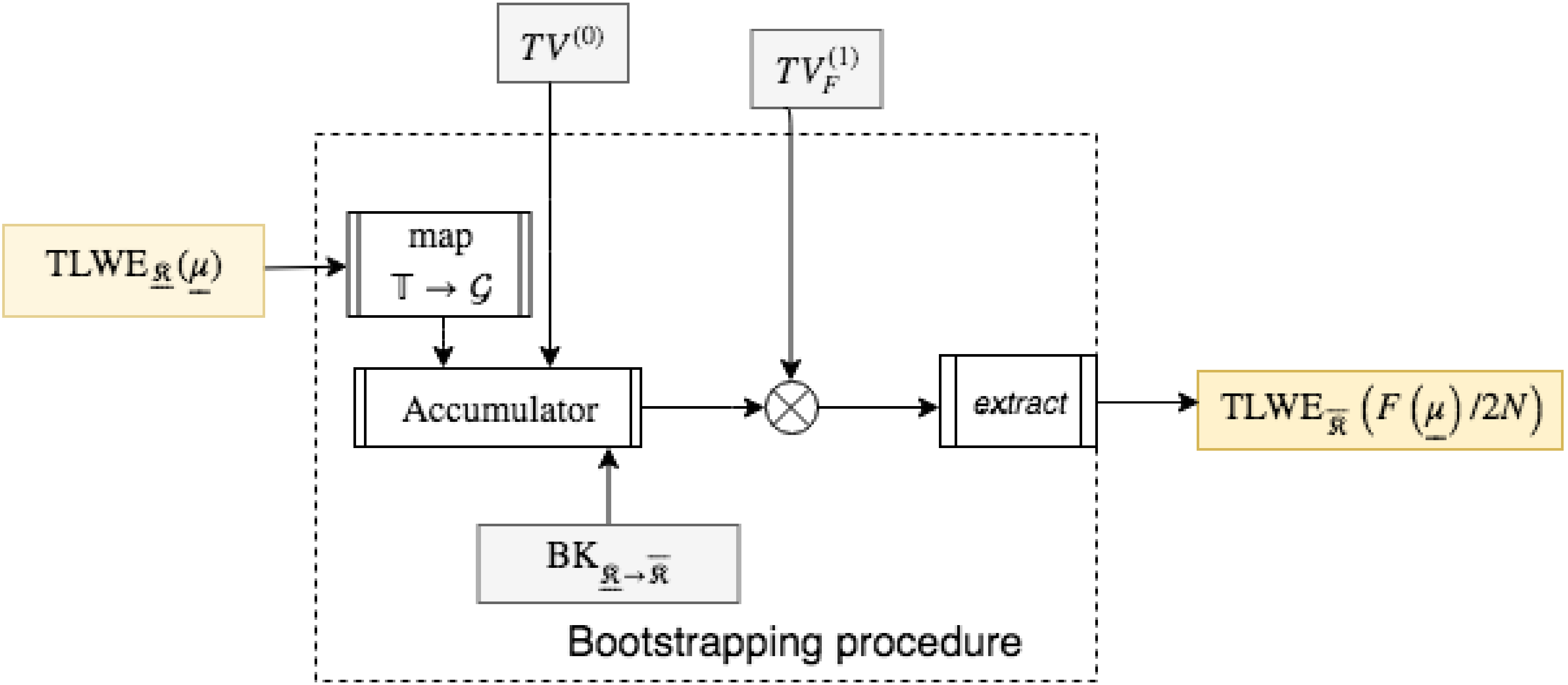


- Second-phase test polynomial : builds a linear combination of previous half-circles.

$$TV_F^{(1)} = t'_a X^a + t'_b X^b + t'_c X^c$$

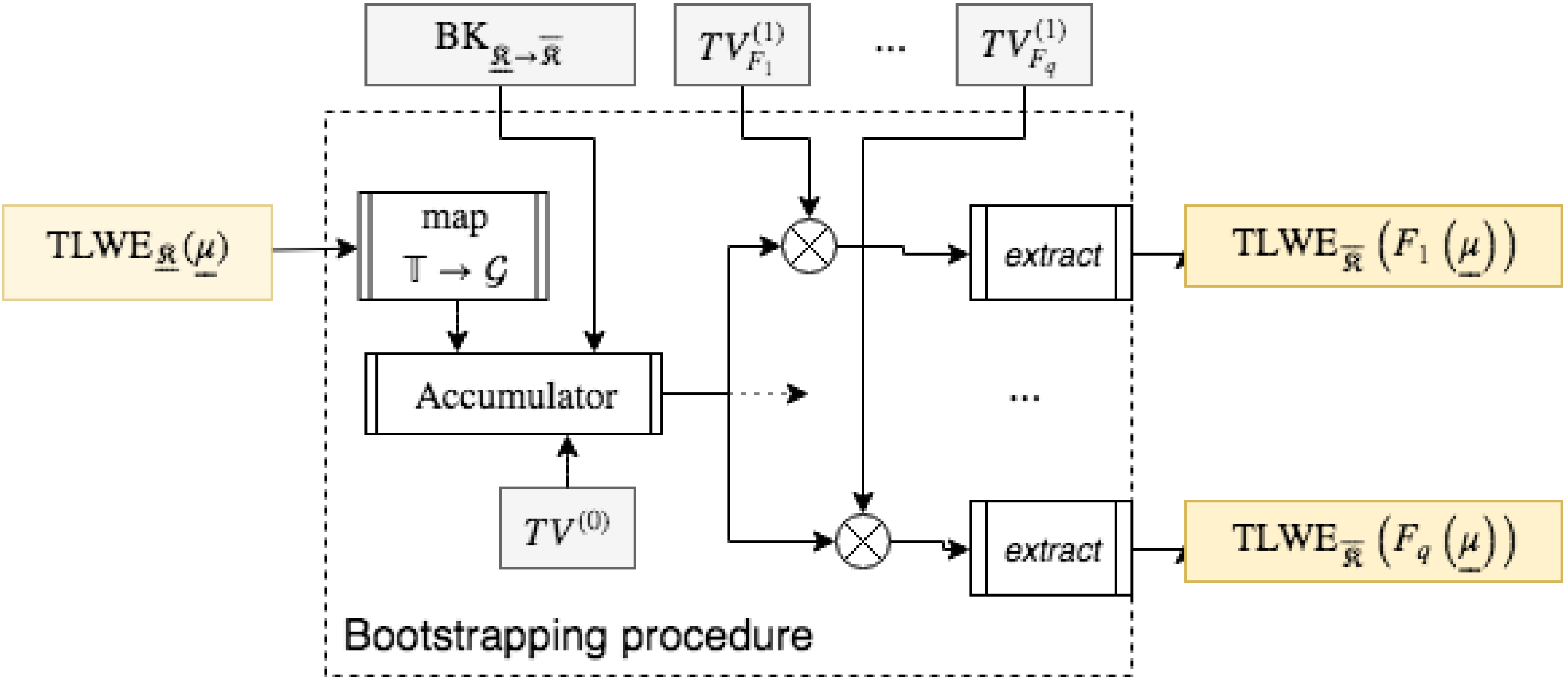


Optimized multi-value Bootstrapping



Multi-output version

- Evaluate several functions $F_1,\ldots, F_q$ on the same input.



Homomorphic Lookup Table

- A boolean Lookup Table (LUT) $f\colon Z_2^r \rightarrow Z_2^q$

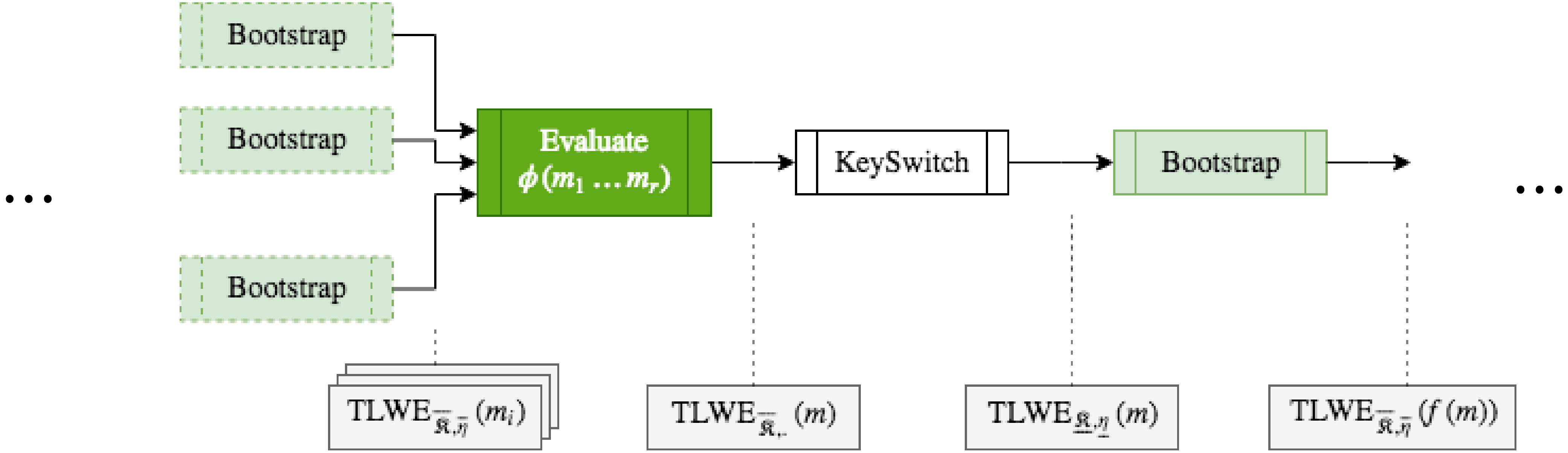
Consider the case q=1

$$\Leftrightarrow \mathrm{F} \circ \varphi \text{ where } \mathrm{F} : Z_{2^r} \longrightarrow Z_2 \text{ and } \varphi : Z_2^r \longrightarrow Z_{2^r} \quad \text{s.t. } \varphi(m_1, \ldots, m_q) = \sum m_i 2^i.$$

- Homomorphic evaluation of the function f :

1. Encode m_j as $\frac{j}{2^{r+1}}$ for $j \in Z_{2^r}$, encode outputs as $\frac{j}{2^{r+1}}$ for $j \in Z_2$ on the half circle.
2. Multi-value Bootstrapping with $TV^0 = \sum X_i$ and TV_F^1 with small norm.

Homomorphic Circuits



Implementation for r=6

Encryption Parameters (for 128 bits of security) :

- TLWE : $\mathrm{n} = 803, \quad \alpha_{LWE} = 2^{-20} \quad \Rightarrow \quad 6.3kB$
- TRLWE: $\mathrm{N} = 2^{14}, \alpha_{TRWE} = 2^{-50} \quad \Rightarrow \quad 256kB$
- TRGSW: $\mathrm{B_g} = 2^6, \ l = 2^3 \quad \Rightarrow \quad 2MB$

Key Parameters (for 128 bits of security) **:**

- LWE key : $\mathrm{n} = 803, \ \mathrm{h} = 63$
- BK $< 2GB$ and KS $\approx 6GB$ generated in 66sec. both

Running time : *Multi-value Bootstrapping with 6-bit inputs, 6bits-outputs runs in 1.57 sec on a single core of an Intel E3-1240 processor running at 3.50GHz.*

Summary

- Optimize the multi-value input Bootstrapping
  - Split factorization method for the test polynomial.
  - Large gate homomorphic evaluation.
  - Multi-output evaluation on the same input.

- Application to homomorphic circuit
  - Implementation of 6-to-6 look-up-table in 1.57 sec (vs a$\approx$ 10sec in [BDF18]).
  - Only 0.05 sec. more for additonal 128 outputs on the same 6 input bits.

Conclusion

- Other applications (hints in the paper):
  - Optimization of the circuit bootstrapping of [CGGI17] : invoke the gate bootstrapping main subroutine once rather than p times.
  - Activation function in neural network homomorphic evaluation : where f is a threshold function.

- Further Improvements ?
  - Other possible factorization instanciations than splitting TV as TV_0 and TV_F^1 ?

- Implement other applications where evaluating f using the Multi-value Bootstrapping could be efficient.