Power Analysis Resistance of RLWE encryption

Andrés Felipe Valencia and Francesco Regazzoni
valena@usi.ch and regazzoni@alari.ch

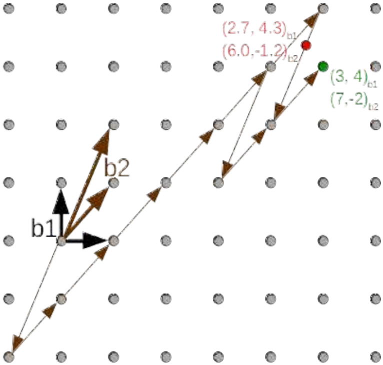
ALaRI Advanced Learning and Research Institute
Università della Svizzera italiana (USI)

Outline

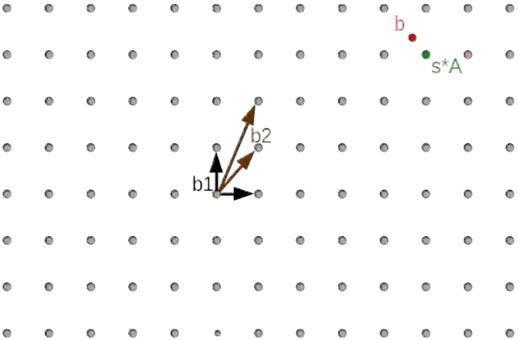
- Lattice-based cryptography
- Physical attacks
- RLWE encryption
- Approaches to protect RLWE against power attacks

Lattice-based cryptography

- A lattice L is a discrete set of points in the space $\mathbb{R}^n$ with periodic structure. Foundations problems are Shortest Vector Problem and Closest Vector Problem



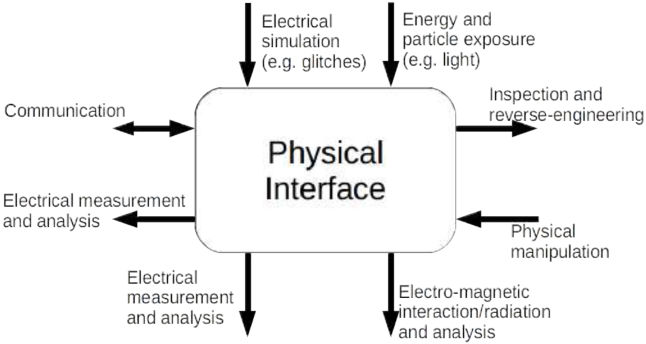
Simple intuition for a crypto system



Lattice-based schemes

- Bliss (Signature scheme)
- Bliss-B (Signature scheme)
- NTRU (Encryption scheme)
- RLWE (Encryption scheme)
- New Hope (Key exchange protocol)
- YASHE (Homomorphic encryption)

Physical attacks



- Timing analysis
- Power analysis
- Fault attacks

NIST requires side channel resistance

- "Schemes that can be made resistant to side-channel attack at minimal cost are more desirable than those whose performance is severely hampered by any attempt to resist side-channel attacks"

[1] http://csrc.nist.gov/groups/ST/post-quantum-crypto/documents/call-for-proposals-final-dec-2016.pdf

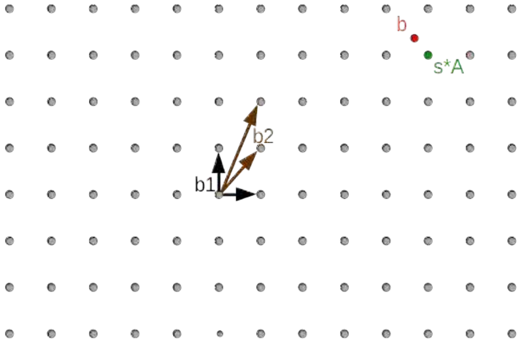
Learning With Errors (LWE)

Find $s \in Z_Q^N$, given $A = \begin{bmatrix} \vdots & & \vdots \\ a_1 & \dots & a_m \\ \vdots & & \vdots \end{bmatrix}$; $b^t = s^t A + e$

- LWE problem is equivalent to lattices problems

Learning With Errors (LWE)

$$\text{Find } s \in Z_Q^N, \text{ given } A = \begin{bmatrix} \vdots & & \vdots \\ a_1 & \dots & a_m \\ \vdots & & \vdots \end{bmatrix};\ b^t = s^t A + e$$



Ring definition

- $\mathbb{Z}$=set of integers
- $\mathbb{Z}_Q$ =set of integers module q
- $\mathbb{Z}_Q^N$ =set of vectors of size n where every component is in $\mathbb{Z}_Q$
- $R_Q = \mathbb{Z}_Q^N/(x^N + 1)$=Ring of vectors in $\mathbb{Z}_Q^N$ module $(x^N + 1)$

Ring Learning With Errors (RLWE)

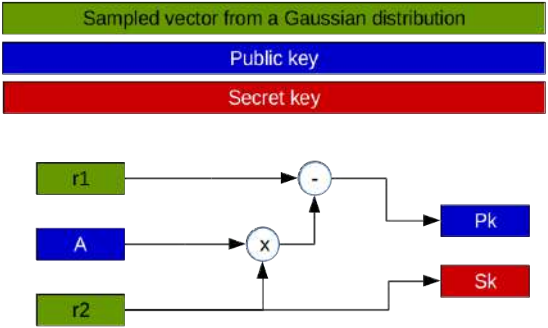
Find $s \in Z_Q^N$, given $A = \begin{bmatrix} \vdots & & \vdots \\ a_1 & \dots & a_m \\ \vdots & & \vdots \end{bmatrix}$; $b^t = s^t A + e$

- We moved from standard lattices to lattices in a ring
- Then the matrix A becomes a vector
- Key size is reduced
- Performance is improved (By using more mathematical tools)

Definition

RLWE (Ring Learning With Errors) encryption is a cryptosystem based on the Learning With Errors problem on Ring. It is parameterized by the length N, an integer Q and a distribution with variance σ

RLWE – Key generation



Number Theoretic Transform (NTT)

- The Number Theoretic Transform is a Fourier transform performed in a ring instead of $\mathbb{C}$

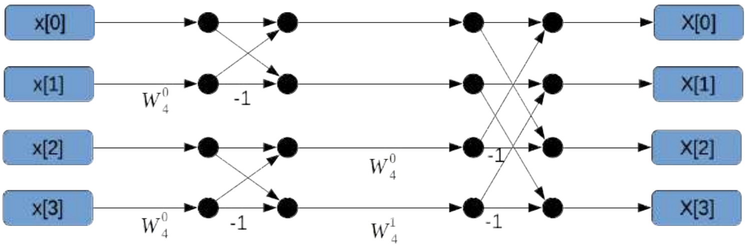
NTT algorithm

Require: Vector x of N components

function $NTT(x)$
    $A \leftarrow Bitreverse(a)$
    for m=2 to N by m=2*m **do**
        $\omega = \omega_n^{n/m}; \omega = 1;$
        for j=0 to m/2 -1 **do**
            for k=0 to N-1 by m **do**
                $t = \omega * X[k+j+m/2]; \quad u = X[k+j];$
                $X[k+j] = u+t; \quad X[k+j+m/2] = u-t;$
            end for
        end for
    end for
    return Vector X of N components
end function

RLWE – NTT procedure

- W_N^j = J-th power of the N-th primitive root of unity
- x vector in time domain
- X vector in NTT domain

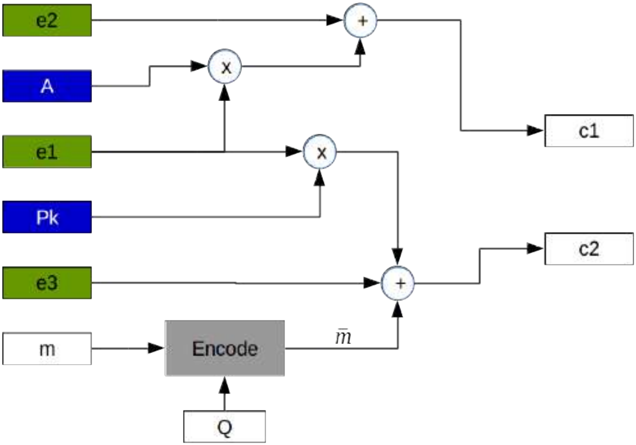


Number Theoretic Transform (NTT)

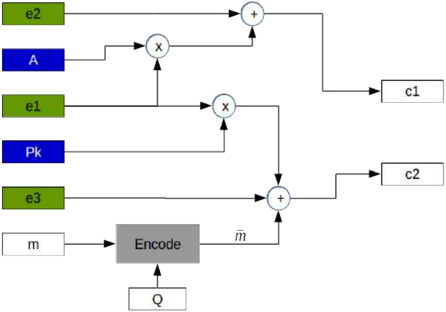
- The Number Theoretic Transform is a Fourier transform performed in a ring instead of $\mathbb{C}$

- It speed up the RLWE encryption because it allows to perform the polynomial multiplication with complexity $\mathcal{O}(n \log n)$
- It requires to make N a power of 2, and $Q = 1 \; mod \; 2 * N$ a prime

RLWE – Encryption

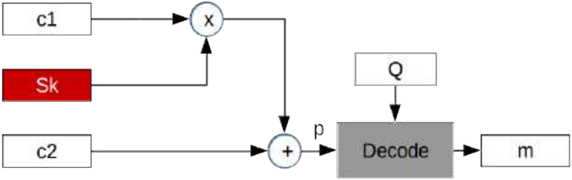


RLWE – Encryption

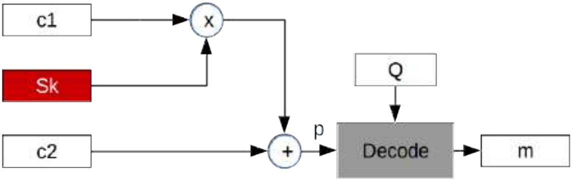


1: **function** $Encode(m)$
2: $\bar{m} \leftarrow m \cdot \lfloor Q/2 \rfloor$
3: **return** $\bar{m}$
4: **end function**

RLWE – Decryption



RLWE – Decryption



function $Decode(p)$

$$m = \begin{cases} 1 & p \in [\lfloor \frac{Q}{4} \rfloor, \lfloor \frac{3*Q}{4} \rfloor) \\ 0 & otherwise \end{cases}$$

return m

end function

Approach 1

- Take advantange of the linearity of multiplication and INTT operation
- Divide the key in two random shares to avoid correlation between intermediate values and the key

[2] https://eprint.iacr.org/2015/724.pdf

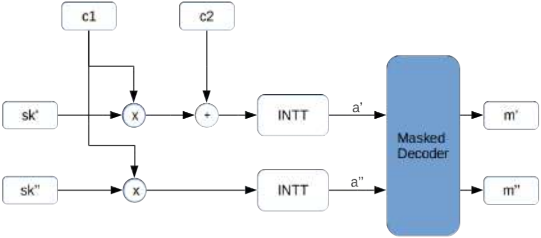
Masked decoder

$$sk = sk' + sk'' \quad (1)$$

$$m \leftarrow Decode(INTT(c_1 \cdot sk) + c_2) \quad (2)$$

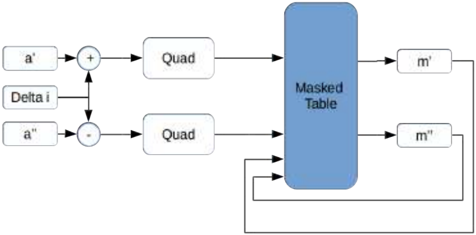
$$INTT(c_1 \cdot sk + c_2) = INTT(c_1 \cdot sk' + c_2) + INTT(c_1 \cdot sk'') \quad (3)$$

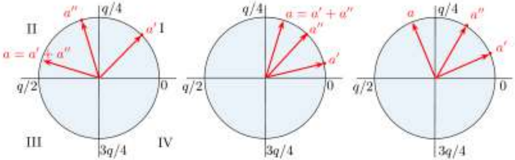
$$m = m' + m'' \quad (4)$$



[2] https://eprint.iacr.org/2015/724.pdf

Masked Decoder





[2] https://eprint.iacr.org/2015/724.pdf

Approach 1

- Take advantange of the linearity of multiplication and INTT operation
- Divide the key in two random shares to avoid correlation between intermediate values and the key

- The decode function is not linear and it has to be modified
- Area increases around 20% (FPGA synthesis)
- Maximum frequency is reduce in 20% (FPGA synthesis)

[2] https://eprint.iacr.org/2015/724.pdf

Approach 2

- Use the homomorphic addition property of RLWE encryption
- It avoids the modification in the decoder

[3] https://www.esat.kuleuven.be/cosic/publications/article-2633.pdf

Working principle

- Given $Decryption(c_1, c_2) = (m)$ and $Decryption(c'_1, c'_2) = (m')$ then $Decryption(c_1 + c'_1, c_2 + c'_2) = m \oplus m'$
- In the decoding phase a random message m' is generated and encrypted in (c'_1, c'_2)
- Then $Decryption(c_1 + c'_1, c_2 + c'_2)$ is performed.
- The output is $(m', m \oplus m')$

Approach 2

- Use the homomorphic addition property of RLWE encryption
- It avoids the modification in the decoder

- It needs an encryption step in the decryption phase, and encryption is 2.8 times slower than the decryption
- Decryption failure rate increases

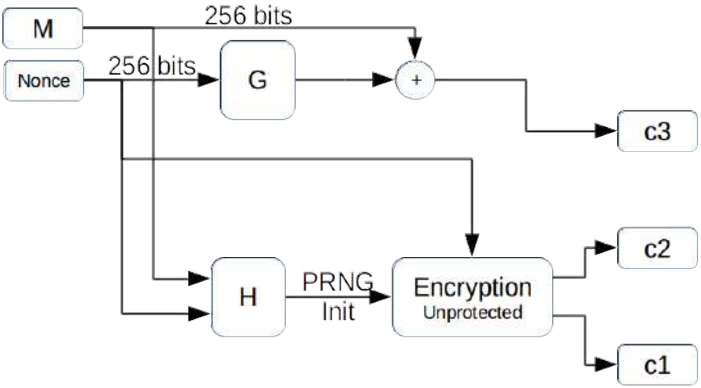
[3] https://www.esat.kuleuven.be/cosic/publications/article-2633.pdf

Approach 3

- Use the principle of dividing on shares the vulnerables variables
- Implement the Fujisaki-Okamoto tranformation (Targhi-Unruh variant) to achived CCA2 protection

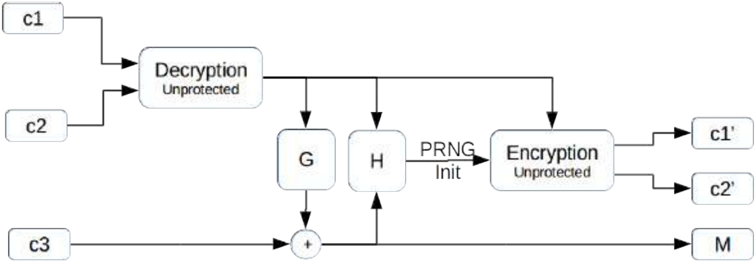
[4] https://eprint.iacr.org/2016/1109.pdf

Encryption phase for RLWE-CCA2



[4] https://eprint.iacr.org/2016/1109.pdf

Decryption phase for RLWE-CCA2



4 https://eprint.iacr.org/2016/1109.pdf

Approach 3

- Use the principle of dividing on shares the vulnerables variables
- Implement the Fujisaki-Okamoto tranformation (Targhi-Unruh variant) to achived CCA2 protection

- It needs an encryption step in the decryption phase
- The decryption in around 5 times slower than in the unprotected version

[4] https://eprint.iacr.org/2016/1109.pdf

Sort of comparison

Approach 1	LUTs/FFs/DSPs	Cycles/Time(μs)
Unprotected	1713 / 830 / 1	2.8k / 23.5
Protected	2014 / 959 / 1	7.5k/75.2

	Cycle count	Dynamic memory	Platform
Approach 1	2,070,952	15,284 bytes	Virtex-II FPGA
Approach 2	3,661,431	15,412 bytes	ARM Cortex-M4
Approach 3	2,931,411	19,380 bytes	Cortex-M4F

[4] https://eprint.iacr.org/2016/1109.pdf

Learned leason

- RLWE can be masked splitting the key and using the linearity of NTT domain
- Intermediate values can be hidden using the homomorphic addition property
- Splitting variables in shares can be used for all sensitive values

Questions

Thank you for your attention