UNIVERSITY OF BIRMINGHAM

Saber: a Post-Quantum Lattice-Based Protocol

Andrea Basso (Saber team)

PQCifris, 24 agosto 2020

In brevissimo

Cos’è Saber

Saber è un protocollo crittografico

- post-quantum, cioè resistente agli attacchi quantistici
- basato sui reticoli e sul problema del Mod-LWR

In pratica,

- offre buone performance software e ottime performance hardware
- ha costi di comunicazione ridotti
- è flessibile e usabile praticamente

La crittografia è un problema dopo l'altro

Ogni protocollo crittografico si basa su un problema computazionale che è facile da generare e difficile da risolvere.

Alcuni esempi classici sono:

- la fattorizzazione degli interi (RSA)
- il logaritmo discreto (Diffie-Hellman)

Nella crittografia basata sui reticoli, le famiglie di problemi sono:

- NTRU
- Learning With Errors

Learning With Errors (LWE) [Reg05]

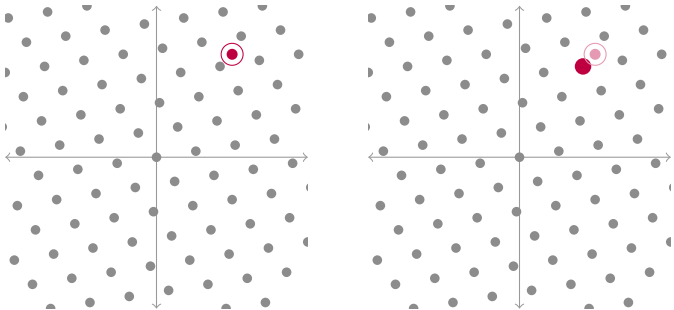
Il problema LWE chiede di ottenere il vettore $\mathbf{s} \in \mathbb{Z}_q^n$, conosciuti la matrice $A \in \mathbb{Z}_q^{n \times n}$ e il vettore $\mathbf{b}$, dove

$$\mathbf{b} = \underbrace{\begin{bmatrix} a_{0,0} & a_{0,1} & \dots & a_{0,n-1} \\ a_{1,0} & a_{1,1} & \dots & a_{1,n-1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n-1,0} & a_{n-1,1} & \dots & a_{n-1,n-1} \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} s_0 \\ s_1 \\ \vdots \\ s_{n-1} \end{bmatrix}}_{\mathbf{s}} + \underbrace{\begin{bmatrix} e_0 \\ e_1 \\ \vdots \\ e_{n-1} \end{bmatrix}}_{\mathbf{e}}.$$

Cos'è un reticolo (*lattice*)

Data una base $B = \{\mathbf{b}_0, \mathbf{b}_1, \ldots, \mathbf{b}_m\} \in \mathbb{Z}^n$, un reticolo $\mathcal{L}$ è definito come

$$\mathcal{L} = \{\sum \alpha_i \mathbf{b}_i \mid \alpha_i \in \mathbb{Z}\}.$$



I grafici sono stati gentilmente creati da [Jea16].

Ring Learning With Errors (RLWE)

Se la matrice A è scelta accuratamente, il prodotto $A\mathbf{s}$ si riduce ad una moltiplicazione tra polinomi in un anello.

Per esempio, se la matrice A è della forma

$$A = \begin{bmatrix} a_0 & a_1 & \dots & a_{n-1} \\ a_1 & a_2 & \dots & -a_0 \\ \vdots & \vdots & \ddots & \vdots \\ a_{n-1} & -a_0 & \dots & -a_{n-2} \end{bmatrix},$$

calcolare $A\mathbf{s}$ è equivalente a calcolare $a(x)s(x)$ sull'anello $\mathbb{Z}_q[x]/\langle x^n + 1\rangle$, dove

$$a(x) = a_0 + a_1 x + \dots a_{n-1}x^{n-1}, \qquad s(x) = s_1 + s_2 x + \dots + s_n x^{n-1}.$$

Module Learning With Errors (MLWE)

Esiste una variante sui moduli, dove i valori di A e $\mathbf{s}$ sono polinomi.

Con rango 3, la matrice A è della forma

$$A = \begin{bmatrix} a_{00}(x) & a_{10}(x) & a_{20}(x) \\ a_{01}(x) & a_{11}(x) & a_{21}(x) \\ a_{02}(x) & a_{12}(x) & a_{22}(x) \end{bmatrix}.$$

Il Module Learning With Errors rappresenta una via di mezzo tra il Learning with Errors puro e il Ring Learning with Errors.

Learning With Rounding (LWR)

Di ogni problema, esiste anche la versione *derandomizzata* dove gli errori sono introdotti da un'operazione di *rescaling* e arrotondamento.

Se q è il modulo e $p < q$, il vettore $\mathbf{b}$ è dato da

$$\mathbf{b} = \left\lfloor \frac{p}{q} A\mathbf{s} \right\rceil .$$

Panorama dei problemi LWE

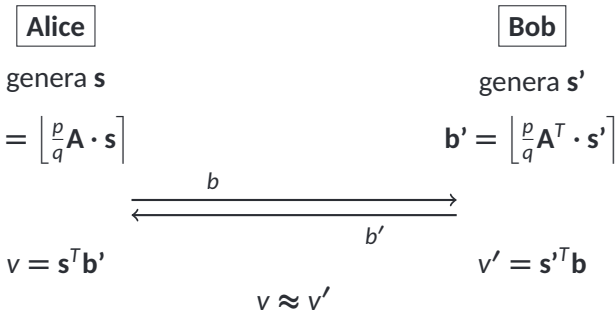
	Base	**Ring**	**Module**
Random error	Learning with Errors (LWE)	Ring Learning with Errors (RLWE)	Module Learning with Errors (MLWE)
Rounding	Learning with Rounding (LWR)	Ring Learning with Rounding (RLWR)	**Module Learning with Rounding (MLWR)**

Sono equivalenti?

In teoria: ci sono riduzioni in tutte le direzioni, ma non sempre strette

In pratica: non esistono (al momento) attacchi che sfruttano le strutture aggiuntive

Come creare un Key-Exchange basato sul Mod-LWR



Perché $v \approx v'$

Alice ottiene $v = s^T \left\lfloor \frac{p}{q} A^T \cdot s' \right\rceil = \frac{p}{q} s^T A^T \cdot s' + s^T e$, mentre

Bob ottiene $v' = s'^T \left\lfloor \frac{p}{q} A \cdot s \right\rceil = \frac{p}{q} s'^T A \cdot s + s'^T e' = \frac{p}{q} s^T A^T \cdot s' + s'^T e'$.

Saber — Key generation [DKSRV18]

Due schemi: Public-Key Encryption (PKE) e Key Encapsulation Mechanism (KEM).
Tre varianti: LightSaber, Saber e FireSaber (con liv. di sicurezza di AES128, AES192, AES256).

I polinomi sono in $\mathcal{R}_q := \mathbb{Z}_q[x]/\langle x^{256}+1\rangle$ o in $\mathcal{R}_p$, dove $q = 2^{13}$ e $p = 2^{10}$.

Algorithm: Key-generation di Saber.PKE

$\boldsymbol{A} \leftarrow \mathcal{U}(\mathcal{R}_q^{3\times 3})$ // $\boldsymbol{A}$ è una matrice unif. random 3×3 con 9 polinomi

$\mathbf{s} \leftarrow \beta_4(\mathcal{R}_q^{3\times 1})$ // $\mathbf{s}$ è un vettore di tre polinomi con coeff. in $[-4,4]$

$\boldsymbol{b} = \left\lfloor \frac{p}{q}\boldsymbol{A}^T \cdot \mathbf{s} \right\rceil$ // $\boldsymbol{b}$ è un vettore di tre polinomi in $\mathcal{R}_p$

return $pk := (\boldsymbol{A}, \boldsymbol{b})$; $sk := \mathbf{s}$

Saber — Encryption

Introduciamo il parametro T per comprimere il messaggio criptato e ridurre i costi di comunicazione. T rappresenta il numero di bit più significativi che vengono inviati. In Saber, $T = 4$.

Algorithm: Encryption di Saber.PKE

input : $pk = (\boldsymbol{A}, \boldsymbol{b})$, messaggio $m \in \mathcal{R}_2$

$\mathbf{s'} \leftarrow \beta_4(\mathcal{R}_q^{3\times 1})$ // $\mathbf{s'}$ è un vettore di tre polinomi con coeff. in $[-4, 4]$

$\boldsymbol{b'} = \left\lfloor \frac{p}{q}\boldsymbol{A} \cdot \mathbf{s'} \right\rceil$ // $\boldsymbol{b'}$ è un vettore di tre polinomi in $\mathcal{R}_p$

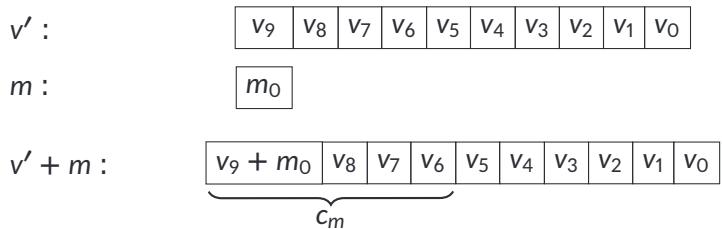
$v' = \boldsymbol{b}^T \mathbf{s'}$ // v' è un polinomio in $\mathbb{Z}_p[x]$

$c_m = \left\lfloor \frac{T}{p}v' + \frac{T}{2}m \right\rceil$

return $c := (c_m, \boldsymbol{b'})$

Message packing e compressione

Dato che $v' \in \mathbb{Z}_p[x]$ e $p = 2^{10}$, ogni coefficiente può essere rappresentato da 10 bit.
Quindi per ogni coefficiente abbiamo che:



Saber — Decryption

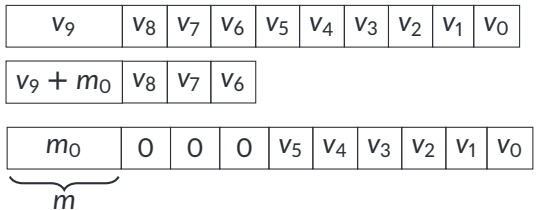
Algorithm: Decryption di Saber.PKE

input : $c = (c_m, \boldsymbol{b'})$, $sk = \mathbf{s}$

$v = \boldsymbol{b'}^T \mathbf{s}$

$m = \left\lfloor \frac{2}{q}(v - \frac{p}{T}c_m) \right\rceil$

return m



Failure is (not) an option

È possibile che la decrittazione fallisca se gli errori introdotti sono troppo grandi.

Se denotiamo con **e** ed **e'** gli errori introdotti dall'arrotondamento, cioè abbiamo

$$\left\lfloor \frac{p}{q}\mathbf{A}^T \cdot \mathbf{s} \right\rceil = \frac{p}{q}\mathbf{A}^T \cdot \mathbf{s} + \mathbf{e}, \qquad \left\lfloor \frac{p}{q}\mathbf{A} \cdot \mathbf{s'} \right\rceil = \frac{p}{q}\mathbf{A} \cdot \mathbf{s'} + \mathbf{e'},$$

ed $e_r \in_R [-p/2T, p/2T]$ (errore introdotto dalla compressione), si può calcolare che la decrittazione fallisce solo se

$$\mathbf{s'}^T\mathbf{e} - \mathbf{s}^T\mathbf{e'} + e_r \bmod p \geq p/4.$$

È un problema?

Esistono attacchi che sfruttano questa proprietà [DGJ+19], ma la probabilità di decryption failure è negligibile (2^{-136}).

Adaptive attacks e Saber.KEM

Se lo stesso vettore segreto è riutilizzato, Saber.PKE è vulnerabile agli adaptive attacks.

Esiste Saber.KEM, un protocollo basato su Saber.PKE utilizzando la trasformazione di Fujisaki-Okamato, che permette di riutilizzare lo stesso vettore segreto.

La trasformazione di Fujisaki-Okamoto [HHK17]

FO trasforma un protocollo di crittazione a chiave pubblica (sicuro CPA) in un Key Encapsulation Mechanism (sicuro CCA).
Definisce un algoritmo di encapsulation e decapsulation che impedisce gli adaptive attacks, assicurandosi che tutti gli input siano generati casualmente.

Le caratteristiche di Saber

Module Learning with Rounding

- Flessibilità (si può variare il rango del modulo mantenendo lo stesso grado dei polinomi)
- Non serve generare gli errori
- Minore bandwidth/costi di comunicazione

Distribuzione binomiale per i vettori segreti

- Più flessibile e più facile da implementare di quella uniforme

Le caratteristiche di Saber

Potenze di due

- Generazione della matrice *A* semplificata
- Riduzioni modulari "gratuite"
- Arrotondamento semplice da implementare
- Non si può usare l'NTT
  - In software, Karatsuba e Toom-Cook
  - In hardware, soluzioni ad-hoc che sfruttano i coefficienti segreti piccoli

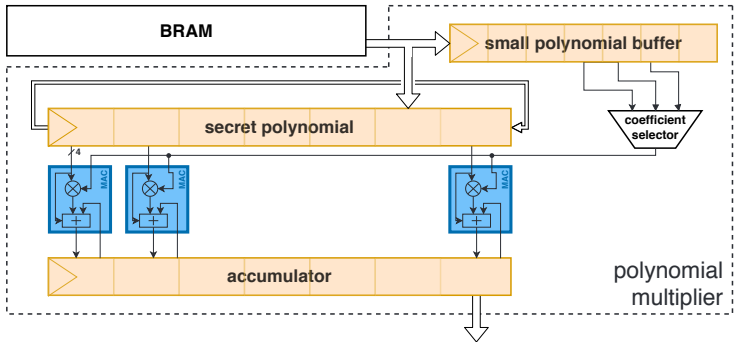
Moltiplicazioni non NTT

L'NTT (Number Theoretic Transform) permette di calcolare il prodotto tra polinomi in un campo $\mathbb{Z}_q$ in $\mathcal{O}(n \log n)$, più efficacemente degli altri algoritmi.
Però richiede che q sia primo e che $q \equiv 1 \bmod 2n$.

Quindi Saber non può usare l'NNT.
Questo è un grosso argomento di ricerca.

In software, approccio ricorsivo con Karatsuba/Toom-Cook.
In hardware, approccio ad-hoc.

Implementazione hardware ad alte perfomance [RB20]



Calcola encapsulation/decapsulation in 26.5/32.1 μs, cioè ~38k/31k operazioni al secondo, con un consumo d'area di ~24k LUTs, ~10k flip-flops, nessun DSP e 2 BRAM tiles (su un FPGA Ultrascale+).

Attacchi side-channel...

Esistono attacchi che ottengono le chiavi tramite

- timing attacks
- electromagnetic (EM) / power attacks
- fault attacks

La competizione NIST

La side-channel security sarà un fattore decisivo per il round 3 della competizione NIST.

...e le contromisure

Masking

Dividi i valori segreti in due componenti casuali tali che

$$s = s' + s''$$ *(per le operazioni lineari)*

o tali che

$$s = s' \oplus s''$$ *(per le operazioni bitwise)*

Saber

- Software: pubblicata di recente un'implementazione resistente agli attacchi side-channel (overhead: 2.5x) [BDK+20]
- Hardware: ci stiamo lavorando...

Conclusione

Saber è un protocollo

- sicuro,
- veloce,
- flessibile.

Scelte di design meno convenzionali:

- variante con l'arrotondamento dei problemi LWE,
- moduli che sono potenze di due.

Nuovi argomenti da ricercare:

- sicurezza side-channel,
- algoritmi di moltiplicazioni meno convenzionali.

References I

[BDK+20] Michiel Van Beirendonck, Jan-Pieter D'Anvers, Angshuman Karmakar, Josep Balasch, and Ingrid Verbauwhede.

A side-channel resistant implementation of saber.

Cryptology ePrint Archive, Report 2020/733, 2020.

https://eprint.iacr.org/2020/733.

[DGJ+19] Jan-Pieter D'Anvers, Qian Guo, Thomas Johansson, Alexander Nilsson, Frederik Vercauteren, and Ingrid Verbauwhede.

Decryption failure attacks on ind-cca secure lattice-based schemes.

In Dongdai Lin and Kazue Sako, editors, *Public-Key Cryptography – PKC 2019*, pages 565–598, Cham, 2019. Springer International Publishing.

References II

[DKSRV18] Jan-Pieter D'Anvers, Angshuman Karmakar, Sujoy Sinha Roy, and Frederik Vercauteren.

Saber: Module-LWR Based Key Exchange, CPA-Secure Encryption and CCA-Secure KEM, volume 10831, page 282–305.

Springer International Publishing, 2018.

[HHK17] Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz.

A Modular Analysis of the Fujisaki-Okamoto Transformation.

In Yael Kalai and Leonid Reyzin, editors, *Theory of Cryptography - 15th International Conference, TCC 2017, Baltimore, MD, USA, November 12-15, 2017, Proceedings, Part I*, volume 10677 of *Lecture Notes in Computer Science*, pages 341–371. Springer, 2017.

[Jea16] Jérémy Jean.

TikZ for Cryptographers.

https://www.iacr.org/authors/tikz/, 2016.

References III

[RB20] Sujoy Sinha Roy and Andrea Basso.

High-speed instruction-set coprocessor for lattice-based key encapsulation mechanism: Saber in hardware.

Cryptology ePrint Archive, Report 2020/434, 2020.

https://eprint.iacr.org/2020/434.

[Reg05] Oded Regev.

On lattices, learning with errors, random linear codes, and cryptography.

In *Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing*, STOC '05, page 84–93, New York, NY, USA, 2005. Association for Computing Machinery.