Salsa Picante:
a machine learning attack on LWE with binary secrets

Cathy Li [1] Jana Sotáková [2,*] Emily Wenger [3] Mohamed Malhou [1] Evrard Garcelon [4] Francois Charton [1] Kristin Lauter [1]

[1]Meta AI

[2]QuSoft and U of Amsterdam [*]work done while at Meta AI

[3]U Chicago

[4]ENSAE - CREST

AICRYPT, April 22, 2023

Learning with errors

For an integer modulus q, write $\mathbb{Z}_q := \mathbb{Z}/q\mathbb{Z}$.

Learning with errors

For an integer modulus q, write $\mathbb{Z}_q := \mathbb{Z}/q\mathbb{Z}$.

Learning with errors

Dimension n, integer modulus q. Secret $\mathbf{s} \in \mathbb{Z}_q^n$.
Error distribution χ (take χ centered Gaussian with $\sigma = 3.2$).

Learning with errors

For an integer modulus q, write $\mathbb{Z}_q := \mathbb{Z}/q\mathbb{Z}$.

Learning with errors

Dimension n, integer modulus q. Secret $\mathbf{s} \in \mathbb{Z}_q^n$.
Error distribution χ (take χ centered Gaussian with $\sigma = 3.2$).
Consider noisy inner products $b = \mathbf{a} \cdot \mathbf{s} + e$ for $\mathbf{a} \leftarrow \mathbb{Z}_q^n$ and e sampled from χ.

Learning with errors

For an integer modulus q, write $\mathbb{Z}_q := \mathbb{Z}/q\mathbb{Z}$.

Learning with errors

Dimension n, integer modulus q. Secret $\mathbf{s} \in \mathbb{Z}_q^n$.
Error distribution χ (take χ centered Gaussian with $\sigma = 3.2$).
Consider noisy inner products $b = \mathbf{a} \cdot \mathbf{s} + e$ for $\mathbf{a} \leftarrow \mathbb{Z}_q^n$ and e sampled from χ.

The **Learning with Errors** problem is to recover $\mathbf{s}$ given m pairs $(\mathbf{a}_j, b_j)$.

Learning with errors

For an integer modulus q, write $\mathbb{Z}_q := \mathbb{Z}/q\mathbb{Z}$.

Learning with errors

Dimension n, integer modulus q. Secret $\mathbf{s} \in \mathbb{Z}_q^n$.
Error distribution χ (take χ centered Gaussian with $\sigma = 3.2$).
Consider noisy inner products $b = \mathbf{a} \cdot \mathbf{s} + e$ for $\mathbf{a} \leftarrow \mathbb{Z}_q^n$ and e sampled from χ.

The **Learning with Errors** problem is to recover $\mathbf{s}$ given m pairs $(\mathbf{a}_j, b_j)$.

binary secret: $\mathbf{s} \in \{0, 1\}^n$

Salsa - first machine learning attack on LWE

Salsa [WCCL22] trains transformers $\mathcal{M}$ on the pairs $(\mathbf{a}, b)$ to approximate the mapping

$$\mathbf{a} \mapsto b \approx \mathbf{a} \cdot \mathbf{s}$$

SALSA - first machine learning attack on LWE

SALSA [WCCL22] trains transformers $\mathcal{M}$ on the pairs $(\mathbf{a}, b)$ to approximate the mapping

$$\mathbf{a} \mapsto b \approx \mathbf{a} \cdot \mathbf{s}$$

Main idea

If the transformer learns, it must have learned information about the secret.

Salsa - first machine learning attack on LWE

Salsa [WCCL22] trains transformers $\mathcal{M}$ on the pairs $(\mathbf{a}, b)$ to approximate the mapping

$$\mathbf{a} \mapsto b \approx \mathbf{a} \cdot \mathbf{s}$$

Main idea

If the transformer learns, it must have learned information about the secret.

Recover the secret:

1. *Direct recovery:* if $\mathbf{e}_i$ is the i-th standard vector and $K \leftarrow \mathbb{Z}_q$, then

$$\mathcal{M}(K\mathbf{e}_i) \approx (K\mathbf{e}_i) \cdot s = \begin{cases} K & s_i = 1, \\ 0 & s_i = 0. \end{cases}$$

Salsa - first machine learning attack on LWE

Salsa [WCCL22] trains transformers $\mathcal{M}$ on the pairs $(\mathbf{a}, b)$ to approximate the mapping

$$\mathbf{a} \mapsto b \approx \mathbf{a} \cdot \mathbf{s}$$

Main idea

If the transformer learns, it must have learned information about the secret.

Recover the secret:

1. *Direct recovery:* if $\mathbf{e}_i$ is the i-th standard vector and $K \leftarrow \mathbb{Z}_q$, then

$$\mathcal{M}(K\mathbf{e}_i) \approx (K\mathbf{e}_i) \cdot s = \begin{cases} K & s_i = 1, \\ 0 & s_i = 0. \end{cases}$$

2. *Distinguisher*: for $(\mathbf{a}, b)$ and LWE sample,

$$\mathcal{M}(\mathbf{a} + K\mathbf{e}_i) \approx (\mathbf{a} + K\mathbf{e}_i) \cdot s = \mathbf{a} \cdot s + K\mathbf{e}_i \cdot s \approx b \longleftrightarrow s_i = 0$$

Drawbacks of Salsa

1. Only succeeds in recovering small dimensions ≤ 128 and Hamming weights (≤ 3 for $n = 128$);

Drawbacks of SALSA

1. Only succeeds in recovering small dimensions ≤ 128 and Hamming weights (≤ 3 for $n = 128$);
2. requires millions of samples $(\mathbf{a}, b)$ to train transformers;

Drawbacks of SALSA

1. Only succeeds in recovering small dimensions ≤ 128 and Hamming weights (≤ 3 for $n = 128$);
2. requires millions of samples $(\mathbf{a}, b)$ to train transformers;
3. only handles (R)LWE with binary secrets.

Salsa Picante

Salsa Picante [LSW+23] is a big improvement on Salsa [WCCL22]:

Salsa Picante

Salsa Picante [LSW$^+$23] is a big improvement on Salsa [WCCL22]:

- highest dimensions and Hammight weights recovered:

Dimension	80	150	200	256	300	350
log q	7	13	17	23	27	32
highest h	9	13	22	31	33	60*

Salsa Picante's highest recovered secret Hamming weights h

Salsa Picante

Salsa Picante [LSW+23] is a big improvement on Salsa [WCCL22]:

- highest dimensions and Hammight weights recovered:

Dimension	80	150	200	256	300	350
$\log q$	7	13	17	23	27	32
highest h	9	13	22	31	33	60*

Salsa Picante's highest recovered secret Hamming weights *h*

- Novel mechanisms:

Salsa Picante

Salsa Picante [LSW+23] is a big improvement on Salsa [WCCL22]:

- highest dimensions and Hammight weights recovered:

Dimension	80	150	200	256	300	350
$\log q$	7	13	17	23	27	32
highest h	9	13	22	31	33	60*

Salsa Picante's highest recovered secret Hamming weights h

- Novel mechanisms:
  - $\star$ reduced number of samples required (we take $m = 4n$);

Salsa Picante

Salsa Picante [LSW$^+$23] is a big improvement on Salsa [WCCL22]:

- highest dimensions and Hammight weights recovered:

Dimension	80	150	200	256	300	350
$\log q$	7	13	17	23	27	32
highest h	9	13	22	31	33	60*

Salsa Picante's highest recovered secret Hamming weights h

- Novel mechanisms:
  - $\star$ reduced number of samples required (we take $m = 4n$);
  - $\star$ (costly) pre-processing step to change the distribution of $(\mathbf{a}, b)$;

Salsa Picante

Salsa Picante [LSW+23] is a big improvement on Salsa [WCCL22]:

- highest dimensions and Hammight weights recovered:

Dimension	80	150	200	256	300	350
$\log q$	7	13	17	23	27	32
highest h	9	13	22	31	33	60*

Salsa Picante's highest recovered secret Hamming weights h

- Novel mechanisms:
  - ⋆ reduced number of samples required (we take $m = 4n$);
  - ⋆ (costly) pre-processing step to change the distribution of $(\mathbf{a}, b)$;
  - ⋆ better secret recovery

Preprocessing step in PICANTE

PICANTE starts from $m = 4n$ original samples $(\mathbf{a}_j, b_j)$.

Preprocessing step in Picante

Picante starts from $m = 4n$ original samples $(\mathbf{a}_j, b_j)$.
Natural ideal: take linear combinations to get more samples:

Preprocessing step in PICANTE

PICANTE starts from $m = 4n$ original samples $(\mathbf{a}_j, b_j)$.
Natural ideal: take linear combinations to get more samples:

+ cheaply get as many samples as we want,

Preprocessing step in PICANTE

PICANTE starts from $m = 4n$ original samples $(\mathbf{a}_j, b_j)$.
Natural ideal: take linear combinations to get more samples:

+ cheaply get as many samples as we want,
− the error blows up,

Preprocessing step in PICANTE

PICANTE starts from $m = 4n$ original samples $(\mathbf{a}_j, b_j)$.
Natural ideal: take linear combinations to get more samples:

+ cheaply get as many samples as we want,
- the error blows up,
+ by controling the linear combinations, we can enforce different distribution of $\mathbf{a}$

Preprocessing step in PICANTE

PICANTE starts from $m = 4n$ original samples $(\mathbf{a}_j, b_j)$.
Natural ideal: take linear combinations to get more samples:

+ cheaply get as many samples as we want,
− the error blows up,
+ by controling the linear combinations, we can enforce different distribution of $\mathbf{a}$

Desired distribution?

SALSA observed that if entries of $\mathbf{a}$ are smaller than q, the transformers learn better.

Preprocessing step in PICANTE II.

Goal:

Get samples $(\mathbf{a}, b)$ with $\mathbf{a}$ with smaller entries.

Preprocessing step in PICANTE II.

Goal:

Get samples $(\mathbf{a}, b)$ with $\mathbf{a}$ with smaller entries.

PICANTE's solution:

Apply lattice-reduction algorithms to reduce the norm of the samples, and hence the coordinates.

Preprocessing step in Picante II.

Goal:

Get samples $(\mathbf{a}, b)$ with $\mathbf{a}$ with smaller entries.

Picante's solution:

Apply lattice-reduction algorithms to reduce the norm of the samples, and hence the coordinates.

Take n out of the m samples, put the $\mathbf{a}$'s in a matrix $\mathbf{A}$ and apply BKZ to:

$$\begin{bmatrix} \omega \cdot \mathbf{1}_n & \mathbf{A}_{n\times n} \\ 0 & q \cdot \mathbf{1}_n \end{bmatrix},$$

where $\omega = 15$ is controling the error/coefficients of the linear combinations.

Preprocessing step in PICANTE II.

Goal:

Get samples $(\mathbf{a}, b)$ with $\mathbf{a}$ with smaller entries.

PICANTE's solution:

Apply lattice-reduction algorithms to reduce the norm of the samples, and hence the coordinates.

Take n out of the m samples, put the $\mathbf{a}$'s in a matrix $\mathbf{A}$ and apply BKZ to:

$$\begin{bmatrix} \omega \cdot \mathbf{1}_n & \mathbf{A}_{n\times n} \\ 0 & q \cdot \mathbf{1}_n \end{bmatrix},$$

where $\omega = 15$ is controling the error/coefficients of the linear combinations.

Repeat this until have enough samples (≈ 4 million) for training the transformers.

Effect of the pre-processing

Dimension $n = 150$ with $q = 6421$ ($\log q = 13$).

Effect of the pre-processing

Dimension $n = 150$ with $q = 6421$ ($\log q = 13$).

BKZ reduction using *fplll* with blocksize β and LLL-DELTA δ:

δ	-	0.96	0.96	0.99
β	-	16	20	20
$\text{norm}(\mathbf{a})/\text{norm}(\mathbf{a}_{random})$	1	0.669	0.581	0.528
cost per matrix (min)	0	30	54	188
highest h	-	**5**	**8**	**12**

Cost of preprocessing

n	$\log_2(q)$	Cost per matrix CPU hours	Matrices needed	Total cost CPU years
80	7	0.01	34,800	0.05
150	13	3.1	14,600	5.3
200	17	15.9	10,800	19.4
256	23	51.9	8,300	48.1
300	27	105.8	7,100	85.6
350	32	152.0	6,000	105

Table. Resources needed for preprocessing. *Total resources needed to produce* 2^{22} *reduced samples, by reducing* $2^{21}/n$ *matrices. This operation can be run in parallel for each matrix.*

PICANTE

SALSA PICANTE is a major extension of the SALSA attack.

+ much higher dimensions and Hamming weights,

PICANTE

SALSA PICANTE is a major extension of the SALSA attack.

+ much higher dimensions and Hamming weights,
+ linear number of samples ($m = 4n$),

Picante

Salsa Picante is a major extension of the Salsa attack.

+ much higher dimensions and Hamming weights,
+ linear number of samples ($m = 4n$),
- costly preprocessing to generate enough data,

Picante

Salsa Picante is a major extension of the Salsa attack.

- $+$ much higher dimensions and Hamming weights,
- $+$ linear number of samples ($m = 4n$),
- – costly preprocessing to generate enough data,
- $+$ trade-off between the cost of preprocessing and highest h recovered.

Picante

Salsa Picante is a major extension of the Salsa attack.

+ much higher dimensions and Hamming weights,
+ linear number of samples ($m = 4n$),
– costly preprocessing to generate enough data,
+ trade-off between the cost of preprocessing and highest h recovered.

Other aspects:

+ Improved secret recovery: improved distinguisher and a novel cross-attention mechanism

PICANTE

SALSA PICANTE is a major extension of the SALSA attack.

+ much higher dimensions and Hamming weights,
+ linear number of samples ($m = 4n$),
– costly preprocessing to generate enough data,
+ trade-off between the cost of preprocessing and highest h recovered.

Other aspects:

+ Improved secret recovery: improved distinguisher and a novel cross-attention mechanism
? PICANTE compared to other (classical) lattice attacks:

Picante

Salsa Picante is a major extension of the Salsa attack.

+ much higher dimensions and Hamming weights,
+ linear number of samples ($m = 4n$),

– costly preprocessing to generate enough data,

+ trade-off between the cost of preprocessing and highest h recovered.

Other aspects:

+ Improved secret recovery: improved distinguisher and a novel cross-attention mechanism

? Picante compared to other (classical) lattice attacks:

1. preprocessing very costly, but parallelizable,

Picante

Salsa Picante is a major extension of the Salsa attack.

+ much higher dimensions and Hamming weights,
+ linear number of samples ($m = 4n$),
– costly preprocessing to generate enough data,
+ trade-off between the cost of preprocessing and highest h recovered.

Other aspects:

+ Improved secret recovery: improved distinguisher and a novel cross-attention mechanism
? Picante compared to other (classical) lattice attacks:
    1. preprocessing very costly, but parallelizable,
    2. use lattice-reduction algorithm with much weaker parameters than pue lattice attacks.

References I

Cathy Li, Jana Sotáková, Emily Wenger, Mohamed Malhou, Evrard Garcelon, Francois Charton, and Kristin Lauter.
SALSA PICANTE: a machine learning attack on LWE with binary secrets.
Cryptology ePrint Archive, Paper 2023/340, 2023.
https://eprint.iacr.org/2023/340.

Emily Wenger, Mingjie Chen, Francois Charton, and Kristin Lauter.
Salsa: Attacking lattice cryptography with transformers, 2022.
https://arxiv.org/abs/2207.04785.