Cryptographic decoding of the Leech lattice

Alex van Poppelen

Supervisors:
Dr. Léo Ducas (CWI)
Dr. Gerard Tel (Utrecht University)

September 20, 2016

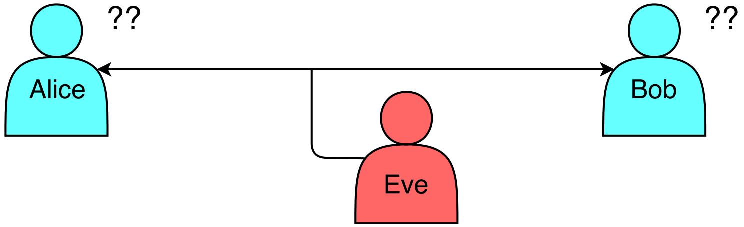
Table of Contents

Introduction

Encryption

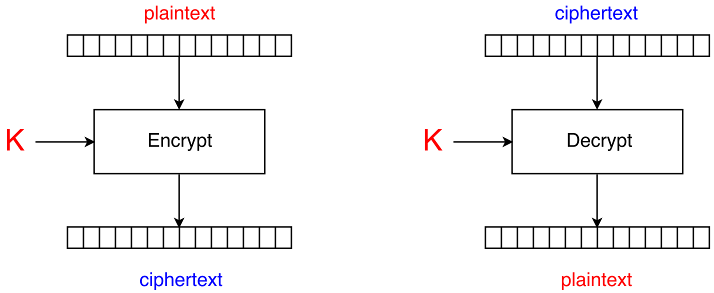
Problem:

- Two parties, Alice and Bob, wish to communicate
- Eve is listening in on the channel
- Alice and Bob want to keep their communication confidential



Symmetric key encryption

Symmetric key encryption works as follows:



- Uses the same key K for encryption and decryption
- Minimal bandwidth overhead
- Typically very efficient

Public key encryption

Public key (asymmetric) encryption consists of three algorithms:

Key generator: Produces a public key P and a private key S.

Public key encryption

Public key (asymmetric) encryption consists of three algorithms:

Key generator: Produces a public key P and a private key S.
Encryption: Uses P to hide a message M in a ciphertext C.

Public key encryption

Public key (asymmetric) encryption consists of three algorithms:

Key generator: Produces a public key P and a private key S.
Encryption: Uses P to hide a message M in a ciphertext C.
Decryption: Uses S to recover M from C.

Public key encryption

Public key (asymmetric) encryption consists of three algorithms:

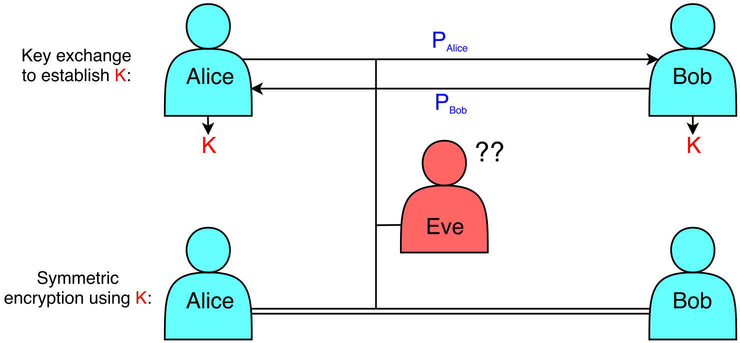
Key generator: Produces a public key P and a private key S.
Encryption: Uses P to hide a message M in a ciphertext C.
Decryption: Uses S to recover M from C.

- Ciphertexts are large (lots of bandwidth overhead)
- Less efficient than symmetric key encryption
- However, does **not** require a pre-shared secret

Encryption in practice

We combine the two to get best of both worlds.

Public key encryption is used to establish a shared secret (called a key exchange in this context), then data is encrypted using a symmetric key algorithm.



Quantum computing

Quantum computing uses quantum-mechanical phenomena for computation.

Instead of only 0s and 1s, quantum bits can exist in a superposition of two states, only “collapsing” when measured. They can also be “entangled”.

Quantum computing

Quantum computing uses quantum-mechanical phenomena for computation.

Instead of only 0s and 1s, quantum bits can exist in a superposition of two states, only “collapsing” when measured. They can also be “entangled”.

New algorithms possible! Often probabilistic.

This is very exciting. But for cryptographers, and people who depend on cryptography (everyone), it is also very **worrying**.

Quantum algorithms

Grover's algorithm

Search algorithm with generic acceleration from $O(N)$ to $O(\sqrt{N})$ evaluations.

Result: Double key size for symmetric encryption.

Shor's algorithm

Solves integer factorization and discrete logarithm problems in polynomial time.

Result: All crypto deployed today is broken.

We need new public key crypto resistant to quantum algorithms!

Post-quantum cryptography

Nobody knows when sufficiently large quantum computers will become reality. Could be 5 years, could be 30 years, could be never.

If quantum computing is so far away, why is it important now?

Post-quantum cryptography

Nobody knows when sufficiently large quantum computers will become reality. Could be 5 years, could be 30 years, could be never.

If quantum computing is so far away, why is it important now?

- Cryptanalysis needs a lot of time to mature
- Implementations need time to mature
- Forward secrecy: protect current data for the long term

Post-quantum cryptography

Nobody knows when sufficiently large quantum computers will become reality. Could be 5 years, could be 30 years, could be never.

If quantum computing is so far away, why is it important now?

- Cryptanalysis needs a lot of time to mature
- Implementations need time to mature
- Forward secrecy: protect current data for the long term

NSA and many other institutions want a transition to post-quantum crypto.

Lattice-based cryptography

Lattice-based cryptography is a promising candidate for post-quantum security.

Pros:

- Resistant (so far) to quantum attacks
- Strong theoretical foundation (worst-case hardness)
- Algorithmically simple, highly parallelizable
- Fast
- Broad diversity of cryptographic possibilities

Cons:

- Current protocols require significantly more bandwidth than those in use today

Improving the bandwidth

Can we do better?

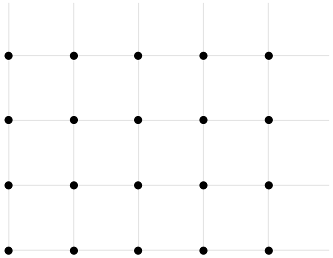
Lattices

What is a lattice?

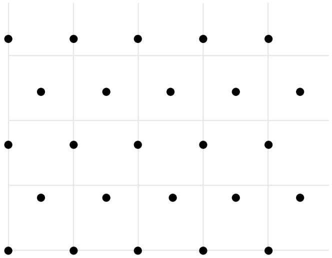
Definition (Lattice).

A lattice is an additive subgroup $\mathcal{L}$ of a finite dimensional Euclidean vector space $\mathbb{R}^m$ such that $\mathcal{L}$ is discrete.

$\mathbb{Z}^2$:



A_2:



Informally, a lattice is just a repeating grid of points in a finite dimensional vector space!

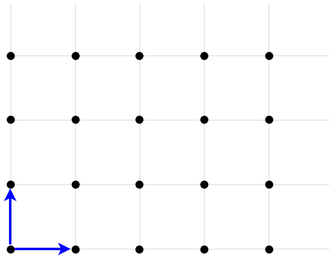
Lattice basis

Definition (Lattice basis).

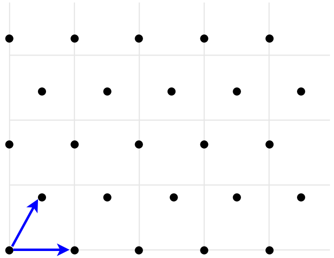
Given n linearly independent vectors $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n \in \mathbb{R}^m$ as a $m \times n$ matrix $\boldsymbol{B}$, we can define the lattice

$$\mathcal{L} = \mathcal{L}(\boldsymbol{B}) := \boldsymbol{B} \cdot \mathbb{Z}^k = \left\{ \sum_{i=1}^{k} z_i \boldsymbol{b}_i : z_i \in \mathbb{Z} \right\}.$$

$\mathbb{Z}^2$:

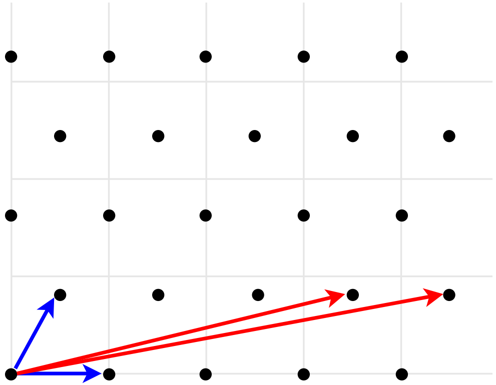


A_2:



Lattice basis

Lattice bases are not made equal! Short = good.



Good basis $\rightarrow$ bad basis : Easy,
Bad basis $\rightarrow$ good basis : Hard.

SVP

Definition (Shortest Vector Problem).

Given an arbitrary basis $\boldsymbol{B}$ for a lattice, find a shortest nonzero lattice vector.

This problem is very well-studied.

Variations (esp. approximation versions) form the foundation for lattice cryptography.

Learning with errors

Learning with errors

Definition (LWE distribution).

For a vector $\boldsymbol{s} \in \mathbb{Z}_q^n$, and an error distribution χ, the LWE distribution $A_{\boldsymbol{s},\chi}$ is sampled by outputting

$$(\boldsymbol{a}, b = \langle \boldsymbol{s}, \boldsymbol{a} \rangle + e \mod q)$$

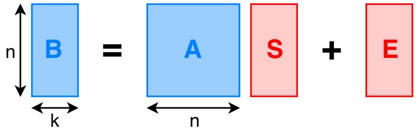
where $\boldsymbol{a} \in \mathbb{Z}_q^n$ is uniformly random, and e is sampled from χ.

Search problem:
Given m samples from $A_{\boldsymbol{s},\chi}$, find $\boldsymbol{s}$.

Decision problem:
Given m samples from $A_{\boldsymbol{s},\chi}$, or m uniformly random samples, distinguish which is the case.

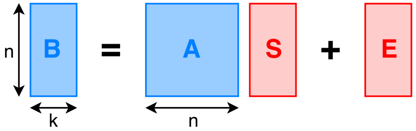
LWE cryptosystem: key generation and encryption

Key generation: Sample two short matrices $\boldsymbol{S}, \boldsymbol{E} \in \mathbb{Z}^{n \times k}$ from χ. $\boldsymbol{S}$ is the private key, and used to calculate the public key $\boldsymbol{B}$.

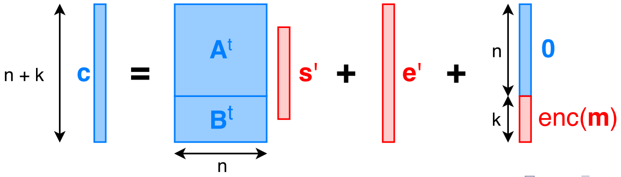


LWE cryptosystem: key generation and encryption

Key generation: Sample two short matrices $\boldsymbol{S}, \boldsymbol{E} \in \mathbb{Z}^{n \times k}$ from χ. $\boldsymbol{S}$ is the private key, and used to calculate the public key $\boldsymbol{B}$.



Encryption: Sample two short vectors $\boldsymbol{s}'$ and $\boldsymbol{e}'$. To encrypt a binary message $\boldsymbol{m} \in \{0, 1\}^l$, the sender generates ciphertext $\boldsymbol{c}$.

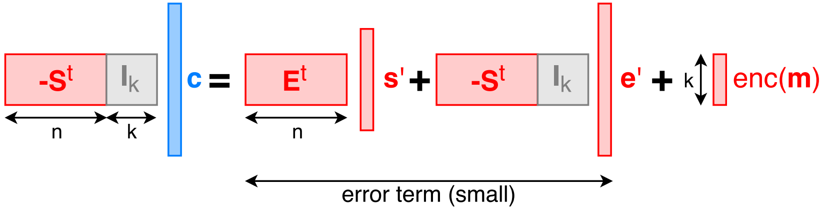


LWE cryptosystem: decryption

Decryption: To decrypt $\boldsymbol{c}$, the receiving party calculates

$$\begin{bmatrix} -\boldsymbol{S}^t & \boldsymbol{I}_k \end{bmatrix} \cdot \boldsymbol{c}.$$

Most terms cancel out, recovering enc($\boldsymbol{m}$) plus an error term. All components of this error term are sampled from χ, meaning they are short.



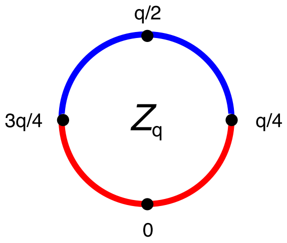
Encoding and decoding

How do we encode a binary message into the ciphertext?

Ciphertext coordinates are modulo q. Suppose we want to encode 1 bit per coordinate.

Encoding: $\{0,1\} \to \mathbb{Z}_q$.
Map $0 \to 0$, and $1 \to \frac{q}{2}$.

Decoding: $\mathbb{Z}_q \to \{0,1\}$.
Map ranges $[0, \frac{q}{4})$ and $[\frac{3q}{4}, q) \to 0$,
and $[\frac{q}{4}, \frac{3q}{4}) \to 1$.



Error must be bounded by $\frac{q}{4}$ in each coordinate!

Decoding

Decoding all bits in k coordinates can be seen as decoding a lattice. Specifically, the scaled integer lattice, $\frac{q}{2}\mathbb{Z}_q^k$.

Decoding

Decoding all bits in k coordinates can be seen as decoding a lattice. Specifically, the scaled integer lattice, $\frac{q}{2}\mathbb{Z}_q^k$.

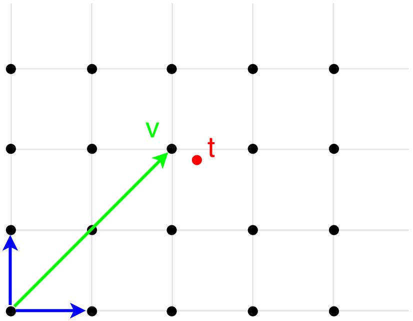
What is lattice decoding in general? Can we use other lattices?

Leech lattice decoding

Lattice decoding = CVP

Definition (Closest Vector Problem).

Given an arbitrary basis $\boldsymbol{B}$ for a lattice and a target point $\boldsymbol{t} \in \mathbb{R}^n$, find a lattice vector $\boldsymbol{v}$ such that the distance between $\boldsymbol{v}$ and $\boldsymbol{t}$ is minimal among lattice vectors.



Lattice codes

What kind of lattice encoding is best for the cryptosystem (or error-correction in general)?

Lattice codes

What kind of lattice encoding is best for the cryptosystem (or error-correction in general)?

Requirements:

- Efficient to decode (CVP is very hard in general)
- Dense (encode more bits in fewer coordinates)
- Large minimal distance (tolerate higher error)

Lattice codes

What kind of lattice encoding is best for the cryptosystem (or error-correction in general)?

Requirements:

- Efficient to decode (CVP is very hard in general)
- Dense (encode more bits in fewer coordinates)
- Large minimal distance (tolerate higher error)

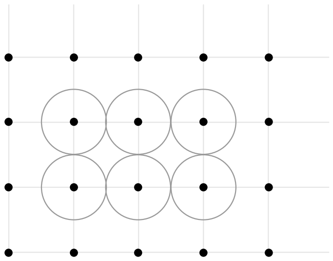
Our error distribution is Gaussian in each coordinate. In multiple dimensions, this distribution looks spherical.

- Efficient error correction relates to dense sphere packings.

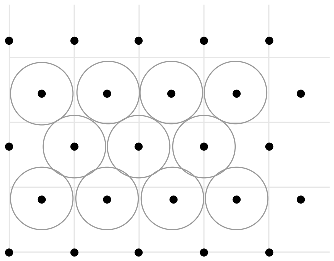
Sphere packing

Very difficult problem in general, lots of research!

$\mathbb{Z}^2$:



A_2:



Same minimal distance between points, but A_2 is significantly denser (0.91 vs 0.79)!

The Leech lattice

The **Leech lattice** Λ_{24} is the densest sphere packing in 24 dimensions.

- Exceptionally dense
- Related to many other branches of mathematics
- Exceptionally structured (symmetries, sublattices)
- Many decoding algorithms available
- Error correction properties extensively studied

Leech lattice encoding in LWE

Restrict Leech lattice points to the following:

$$q\mathbb{Z}^{24} \subset \boldsymbol{T}\Lambda_{24} \subset \mathbb{Z}^{24},$$

where $\boldsymbol{T}$ is some transformation.

- Encode binary message $\boldsymbol{m}$ as Leech lattice point
- Blocks of 24 coordinates
- Use decoding algorithm to decode received point which has error

Leech lattice encoding in LWE

Restrict Leech lattice points to the following:

$$q\mathbb{Z}^{24} \subset \boldsymbol{T}\Lambda_{24} \subset \mathbb{Z}^{24},$$

where $\boldsymbol{T}$ is some transformation.

- Encode binary message $\boldsymbol{m}$ as Leech lattice point
- Blocks of 24 coordinates
- Use decoding algorithm to decode received point which has error

How much can we decrease the bandwidth of the key exchange scheme?

Analysis

We try several parameter sets for both encodings, subject to the following requirements:

- Parameter sets achieve 128 bits of quantum security
- The failure rate of the protocol (error is too large) is very low
- The standard deviation of the error distribution χ is ≥ 1.0

The last requirement ensures combinatorial attacks are not feasible.

Key exchange results

encoding	q	n	σ	k	bits	failure	bandwidth
$\mathbb{Z}_q^k$	2^{12}	568	1.75	128	256	2^{-38}	19.1 KB
	2^{13}	618	1.70	86	258	2^{-38}	18.6 KB
	2^{14}	664	1.65	64	**256**	2^{-38}	**18.2** KB
Λ_{24}	2^{10}	500	1.15	192	288	2^{-37}	17.1 KB
	2^{11}	552	1.10	120	**300**	2^{-38}	**16.3** KB
	2^{12}	574	1.55	96	288	2^{-37}	16.8 KB
	2^{13}	626	1.50	72	288	2^{-37}	16.9 KB
	2^{14}	700	1.00	48	264	2^{-38}	16.7 KB

Key exchange results

Improvement: **10%**.

Significant, but is it worth the extra complexity?

Key exchange results

Improvement: **10%**.

Significant, but is it worth the extra complexity?

Turns out the 24 dimensions of the Leech lattice do not fit well with the requirement to agree on $2^8 = 256$ bits of key material.

What if we only need to agree on 240 bits? We get an improvement of **18%**!

Cryptographic decoder implementation

Leech lattice decoder

CVP Leech lattice decoder by Vardy and Be'ery, 1993.

- Fastest Leech lattice decoder known
- Makes extensive use of the Leech lattice's structure
- Requires 3595 "real" operations in the worst case (ignoring memory addressing, negation, absolute value calls)
- 2955 operations on average

Very complex. Divides the Leech lattice into six component lattices, and combines the cosets of these component lattices using three levels of "glue" codes.

Cryptographic requirements

Implementation must not leak information. This is very difficult. Some **side-channels** include timing, power consumption, cache accesses, error messages.

Requirements:

- Constant time algorithm
- No data-dependent branching
- No data-dependent memory access

Constant time algorithm

For every algorithm step, must do all calculations for all possible cases, regardless of whether necessary.

Leech decoder:

- Many alterations to calculate all branch cases
- Large sorts required list traversing $\rightarrow$ potential leak
- Break up sorting step into many smaller sorts. No more list traversing

Resulting decrease in efficiency was minor.

Branching and memory access

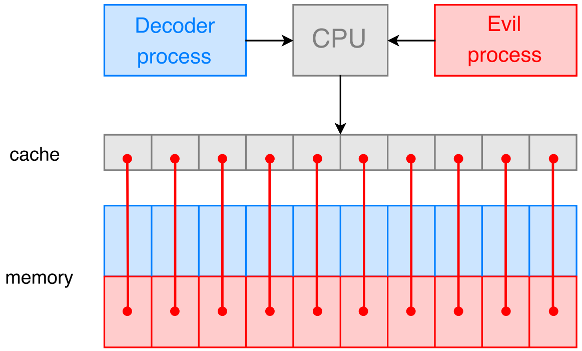
Widely varying compilers and CPU architectures make things difficult.

Code we write as “safe” may be modified or optimized by compiler to be unsafe.

Branching: In general, compiler will remove branches, rather than introduce them (branching is expensive).

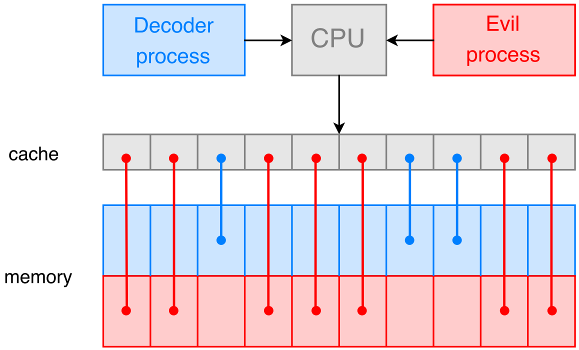
Cache: However, we have little control over how the cache is used.

Cache attack



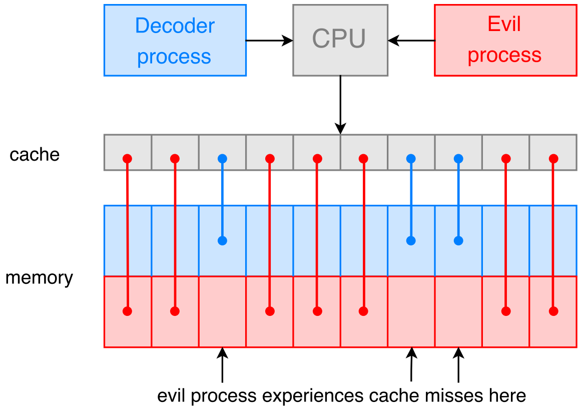
Evil process first overwrites the cache.

Cache attack



Decoder process gets a turn on the CPU. Accesses some data dependent memory, overwriting parts of the cache corresponding to these addresses.

Cache attack



Evil process now tries to access all of its memory. By timing its own memory access calls, can determine when it has experienced a cache miss. A cache miss indicates the decoder accessed memory pointed to by this part of the cache.

Minimization and sorting

Problem:
Get the minimum and maximum of two values: a, b

Solution:

- Extract sign s of $a - b$ (e.g. via a bitshift)
- Construct
$$\min = as + b(s \texttt{ XOR } 1),$$
$$\max = bs + a(s \texttt{ XOR } 1).$$
- Set $a = \min$ and $b = \max$.

This "swap" ensures constant time, and data independent memory access.

Sorting: Use this swap, and constant swap "plan".

Implementation

Unsafe algorithm and safe algorithm (with above modifications and considerations) were implemented in C. Compiled using GCC 5.3.1, run on a 2.4GHz Intel Core i3 M370 processor.

Unsafe: **41.2 $\pm$ 2.3 microseconds**
Safe: **55.4 $\pm$ 0.1 microseconds**

Implementation

Unsafe algorithm and safe algorithm (with above modifications and considerations) were implemented in C. Compiled using GCC 5.3.1, run on a 2.4GHz Intel Core i3 M370 processor.

Unsafe: **41.2 $\pm$ 2.3 microseconds**
Safe: **55.4 $\pm$ 0.1 microseconds**

I did spend more time optimizing the safe version! Lots more room for improvement: algorithm is ripe for vectorization/parallelization. Could cut time in 4, or possibly 8.

Compare to running times of 300 to 1300 microseconds and up for lattice-based key exchanges, and 1400 and up for elliptic curve DH (all on better hardware).

Conclusion

Conclusion

Leech lattice encoding for LWE

- Approximately 10% improvement in key exchange bandwidth
- Leech lattice's dimension is cumbersome in the context of 256 bits, and inhibits better coding gain
- Decoder implementation is fast enough
- It is definitely feasible to make the decoder cryptographically safe

Future work

- Other lattice codes, with more agreeable dimensions?
- Leech lattice as an ideal in a Ring-LWE scheme (more efficient version of LWE), could be a more natural fit?

Thanks for listening!

Questions or comments?