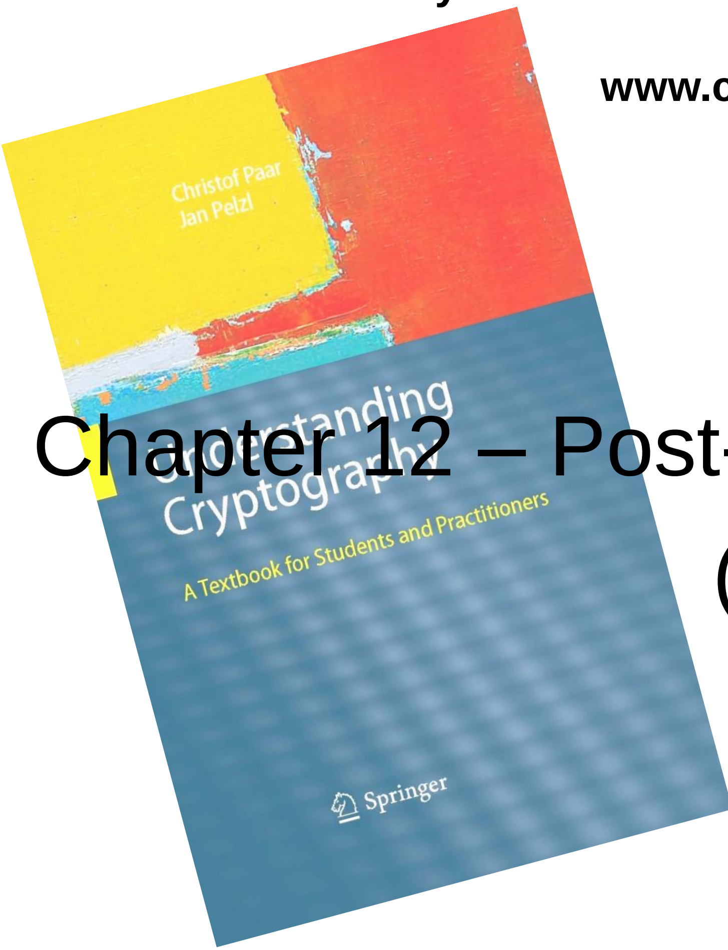
Understanding Cryptography

by Christof Paar, Jan Pelzl and Tim Güneysu

www.crypto-textbook.com



Chapter 12 – Post-Quantum Cryptography (PQC)

These slides were originally prepared by Christof Paar, Jan Pelzl and Tim Güneysu. Later, they were modified by Tomas Fabsic for purposes of teaching I-ZKRY at FEI STU.

Homework till 2.12.2024

- Read Section 13.1
- Read Section 12.1.
- Read Section 12.2. until p. 399 where the subsection about Ring-LWE starts
- Solve the problem from the exercise set no. 10 and submit it to AIS by **2.12.2024 23:59**.

Homework till 8.12.2024

- Read the remainder of Section 12.2
- Read Section 12.3 until p.414 (including p.414)
- Solve problems from the EXTRA exercise set no. 11 and submit them to AIS by **8.12.2024 23:59**. Please, note that the **deadline is on Sunday** (not on Monday, as usual).

Homework for week 13

- Read Section 12.3 until p.417 (including the top half of p.417).

Some legal stuff (sorry): Terms of Use

- The slides can used free of charge. All copyrights for the slides remain with Christof Paar and Jan Pelzl.
- The title of the accompanying book “Understanding Cryptography” by Springer and the author’s names must remain on each slide.
- If the slides are modified, appropriate credits to the book authors and the book title must remain within the slides.
- It is not permitted to reproduce parts or all of the slides in printed form whatsoever without written consent by the authors.

Content of this Chapter

- Introduction
- Lattice-Based Cryptography
- Code-Based Cryptography

Content of this Chapter

- **Introduction**
- Lattice-Based Cryptography
- Code-Based Cryptography

Quantum computing

- A quantum computer is a machine that operates on **qubits** instead of classical bits.
- Roughly speaking, a single qubit |q> is a state of memory that is not as discrete as we know it from conventional bits, which can take the two values 0 or 1.
- Rather, a qubit is a fuzzy memory element that can also represent values “in between” the two corresponding bounds |0> and |1>.
- The overlap between these bounds is characterized by coefficients or so-called amplitudes α and β.
- This allows a qubit to be represented as a scaled combination of the two bounds like |q> = α|0> + β|1>.
- We say that the qubit q is in a **superposition** of the basis states |0> and |1>.

Advantage of quantum computing

- With two conventional bits, we can store one out of the four possible states 00, 01, 10 and 11.
- However, two qubits contain a representation of all four possible states at the same time, to be determined by the corresponding amplitudes.
- In general, an n-bit register on a classical computer can hold exactly one state, while an n-qubit register represents 2^n states at the same time.
- Hence, computing with such an n-qubit quantum computer can be significantly more powerful than any n-bit classical computer.

Limitations of quantum computing

- It is a popular misunderstanding that quantum computers will lead to significant gains in performance for all applications.
- In fact, they can accelerate only certain classes of computations and algorithms for which the native superposition of inputs can be efficiently exploited.

Grovers algorithm

- One problem a quantum computer can solve efficiently is searching in an unsorted database with N entries.
- A classical computer needs to iterate through the entries and compare them with the desired value, which needs N steps in the worst case.
- In contrast, Grovers algorithm can be used on a quantum computer to solve the problem in approximately $\sqrt{N}$ steps.

Grovers algorithm and symmetric cryptosystems

- As we have seen in previous chapters, the best known attack against sound symmetric ciphers is an exhaustive key search, cf. Section 3.5.1.
- We recall that at least one known plaintext-ciphertext pair is required.
- This attack is basically the same as searching in an unsorted database:
  - encrypt the known plaintext with all possible keys,
  - retrieve a large database of unsorted values,
  - and then search for the known ciphertext.
- For example, AES with a 128-bit key can be broken with a classical computer in roughly 2^128 steps, assuming we have a plaintext/ciphertext pair.
- With a quantum computer running Grovers algorithm, the same attack is more efficient: It would take only 2^64 steps.

Grovers algorithm and symmetric cryptosystems

- Fortunately, the problem can be solved by increasing the key length of symmetric algorithms.
- In fact, Grover’s algorithm was the main reason why AES was designed with the two key lengths of 192 and 256 bits, in addition to the 128-bit key.

Quantum computer attacks on asymmetric cryptosystems

- Unfortunately, quantum computers pose a much more serious threat to all asymmetric cryptosystems that are currently in use.
- In 1994, Peter Shor published two algorithms for quantum computers that can efficiently solve:
  - Prime factorization
  - Discrete logarithm problem
- Fortunately, large-scale quantum computers that are required to break cryptosystems like RSA and ECC cannot be built currently.
- It is commonly believed that practical attacks running on quantum computers are most likely at least 10–20 years away, possibly much longer.

Why quantum-secure asymmetric cryptosystems are needed NOW

- First reason: **“store now, decrypt later”** attack
- Second reason: the development and the adoption of new asymmetric algorithms take a long time

PQC vs Quantum Cryptography

- **Quantum cryptography** denotes concepts such as quantum key distribution (QKD) for securely exchanging keys over quantum channels, which are built on actual quantum effects.
- **Post-quantum cryptography** (**PQC**), denotes the class of cryptographic algorithms that are designed to run on conventional computers but that are capable of withstanding attacks that use powerful quantum computers.

NIST PQC Standardization Process

- In 2017, NIST issued an open standardization call for quantum-secure asymmetric cryptosystems.
- This process is similar to the AES competition.
- The process is still ongoing, but has already led to the publication of 3 PQC standards:
  - ML-KEM (aka Kyber)
  - ML-DSA (aka Dilithium)
  - SLH-DSA (aka SPHINCS+)

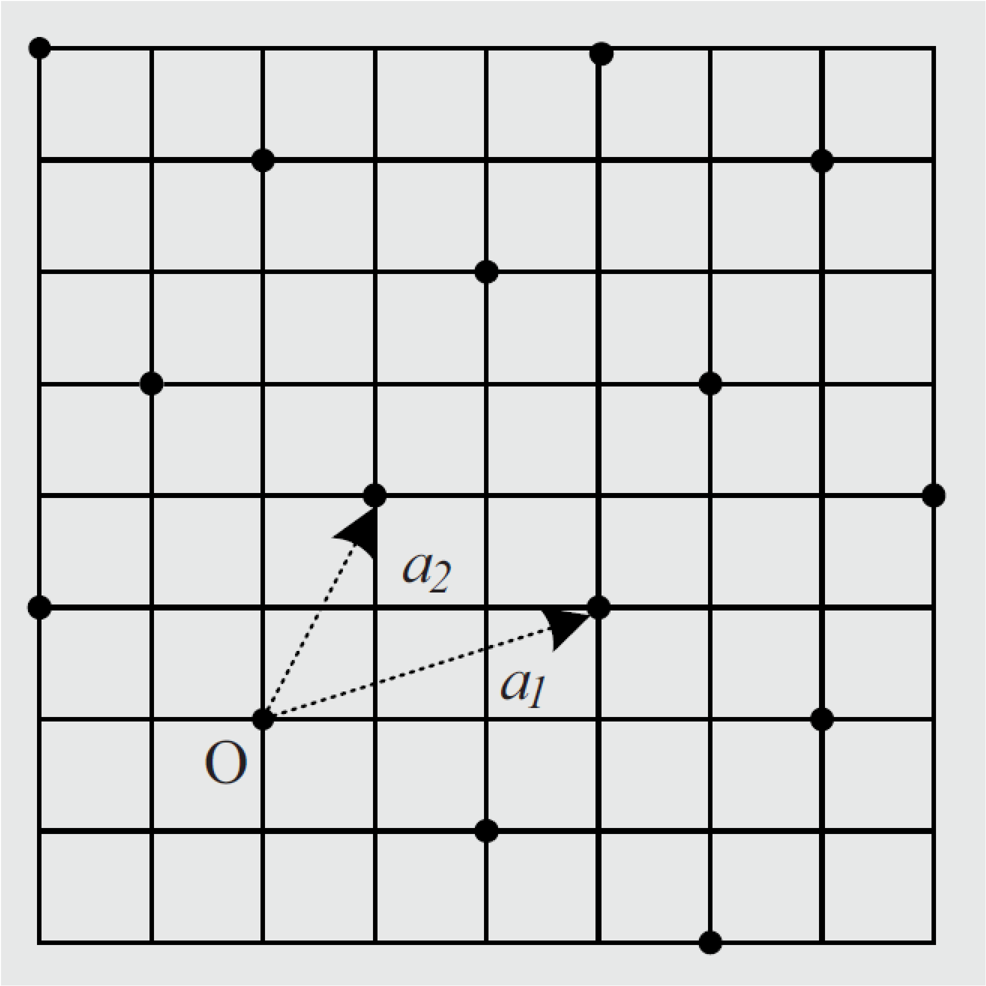
Promising PQC families

PQC Family	Supported Services	Cryptosystems
Lattice-based cryptography	key transport digital signatures	LWE, KYBER, FRODO, DILITHIUM, FALCON
Code-based cryptography	key transport	McEliece, Niederreiter
Hash-based cryptography	digital signatures	MSS, XMSS, LMS, SPHINCS+

Content of this Chapter

- Introduction
- **Lattice-Based Cryptography**
- Code-Based Cryptography

- **What is a lattice?**



Chapter 12 of Understanding Cryptography by Christof Paar, Jan Pelzl and Tim Güneysu

- **Learning With Errors Problem (LWE)**

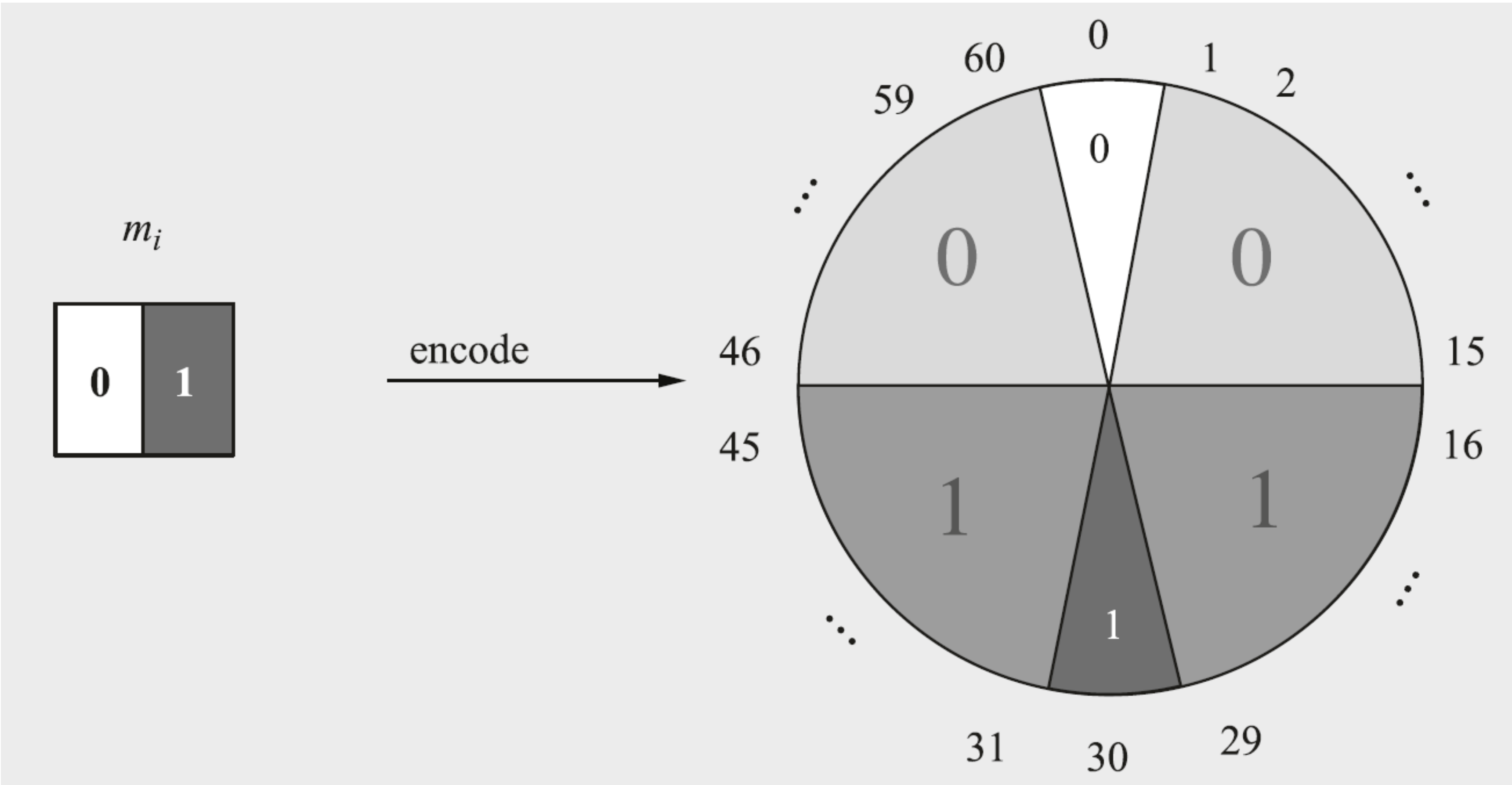
Definition 12.2.2 Learning With Errors Problem (LWE)
Given a set of n basis vectors $\mathbf{a_i} \in \mathbb{Z}_q^m$ *represented by matrix* $\mathbf{A}$ *and a point* $\mathbf{t} \in \mathbb{Z}_q^m$.

The LWE is the problem of determining a set of secret coefficients $\mathbf{s} = (s_1, s_2, \ldots, s_n)$, *with* $s_i \in \mathbb{Z}_q$, *such that:*

$$\mathbf{A} \cdot \mathbf{s} + \mathbf{e} \equiv \mathbf{t} \bmod q$$

where $\mathbf{e}$ *is an unknown error vector consisting of small integers modulo q.*

- **Encoding and Decoding in LWE cryptosystems**



Chapter 12 of Understanding Cryptography by Christof Paar, Jan Pelzl and Tim Güneysu

Simple-LWE Key Generation

Output: public key: $k_{pub} = (\mathbf{t}, \mathbf{A})$ with $\mathbf{t} \in \mathbb{Z}_q^k$ and $\mathbf{A} \in \mathbb{Z}_q^{k \times n}$
private key: $k_{pr} = \mathbf{s} \in \mathbb{Z}_q^n$

1. Choose n random vectors $\mathbf{a}_i \in \mathbb{Z}_q^k$ and combine them in a matrix $\mathbf{A} = (\mathbf{a_1}, \mathbf{a_2}, \ldots, \mathbf{a_n}) \in \mathbb{Z}_q^{k \times n}$.
2. Generate a random secret key $\mathbf{s}$ from "small" integers.
3. Build a random error vector $\mathbf{e}$ from "small" integers.
4. Compute $\mathbf{t} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e}$.
5. Return the public key $k_{pub} = (\mathbf{t}, \mathbf{A})$ and the private key $k_{pr} = \mathbf{s}$.

Simple-LWE Encryption

Input: public key $k_{pub} = (\mathbf{t}, \mathbf{A})$, message $m \in \{0, 1\}$
Output: ciphertext $\mathbf{c} = (\mathbf{c}_{aux}, c_{msg})$ with $\mathbf{c}_{aux} \in \mathbb{Z}_q^n$ and $c_{msg} \in \mathbb{Z}_q$

1. Sample small random integers into vectors $\mathbf{r}, \mathbf{e}_{aux}$ and a value e_{msg}.
2. Encode the message m: $\bar{m} = \mathrm{enc}(m) \in \mathbb{Z}_q$.
3. Compute $\mathbf{c}_{aux} = \mathbf{A}^T \cdot \mathbf{r} + \mathbf{e}_{aux}$.
4. Compute $c_{msg} = \mathbf{t}^T \cdot \mathbf{r} + e_{msg} + \bar{m}$.
5. Return the ciphertext $\mathbf{c} = (\mathbf{c}_{aux}, c_{msg})$.

Simple-LWE Decryption

Input: private key $k_{pr} = \mathbf{s} \in \mathbb{Z}_q^n$, ciphertext $\mathbf{c} = (\mathbf{c}_{aux}, c_{msg})$
Output: message $m \in \{0, 1\}$

1. Return message $m = \mathrm{dec}\left(c_{msg} - \mathbf{s}^T \cdot \mathbf{c}_{aux}\right)$.

Definition 12.2.3 The ring $R_q = \mathbb{Z}_q[x]/(x^n+1)$
The polynomial ring $\mathbb{Z}_q[x]/(x^n+1)$ consists of all polynomials with a maximum degree of $n-1$ with coefficients from $\mathbb{Z}_q$ and n being a power of two, i.e., $n = 2^i$.

The ring operations addition, subtraction and multiplication are performed as regular polynomial arithmetic, with the results being reduced modulo the cyclotomic polynomical x^n+1. All integer coefficients are reduced modulo q.

Definition 12.2.4 Ring-LWE Problem
Let R_q denote the ring $\mathbb{Z}[x]_q/(x^n+1)$, where q is a prime and the positive integer n is a power of two. Given are polynomials $\mathbf{a}$ and $\mathbf{t} \in R_q$.

Ring-LWE is the problem of determining a secret polynomial $s \in R_q$ such that:

$$\mathbf{a}(x) \cdot s(x) + e(x) = \mathbf{t}(x)$$

where the error vector e is a polynomial in the ring R_q with small integer coefficients obtained from a discrete distribution D.

Note:

We use boldface for polynomials with large coefficient values such as $\mathbf{a}(x), \mathbf{t}(x) \in R_q$ while we use plain font for polynomials such as $e(x), s(x)$ which have only small values.

Ring-LWE Key Generation

Output: public key: $k_{pub} = (\mathbf{t}, \mathbf{a})$ and private key: $k_{pr} = s$

1. Choose $\mathbf{a}(x) \in R_q$ from the ring $R_q = \mathbb{Z}[x]_q/(x^n + 1)$.
2. Choose $e(x), s(x) \in R_q$ with coefficients from a set of small integers according to some discrete error distribution D.
3. Compute $\mathbf{t}(x) = \mathbf{a}(x) \cdot s(x) + e(x) \in R_q$.
4. Return the public key $k_{pub} = (\mathbf{t}, \mathbf{a})$ and the private key $k_{pr} = s$.

Ring-LWE Encryption

Input: public key $k_{pub} = (\mathbf{t}, \mathbf{a})$, message $m \in \{0,1\}^n$

Output: ciphertext $\mathbf{c} = (\mathbf{c}_{aux}, \mathbf{c}_{msg})$

1. Choose error polynomials $r(x), e_{aux}(x), e_{msg}(x)$ with coefficients from a set of small integers according to the discrete error distribution D.
2. Write the n message bits m as a message polynomial $m(x)$ and generate the encoded polynomial: $\bar{\mathbf{m}}(x) = \mathrm{enc}(m(x))$.
3. Compute $\mathbf{c}_{aux}(x) = \mathbf{a}(x) \cdot r(x) + e_{aux}(x)$.
4. Compute $\mathbf{c}_{msg}(x) = \mathbf{t}(x) \cdot r(x) + e_{msg}(x) + \bar{\mathbf{m}}(x)$.
5. Return the ciphertext $\mathbf{c} = (\mathbf{c}_{aux}, \mathbf{c}_{msg})$.

Ring-LWE Decryption

Input: private key $k_{pr} = s$, ciphertext $\mathbf{c} = (\mathbf{c}_{aux}, \mathbf{c}_{msg})$

Output: message m

1. Compute $\mathbf{m}'(x) = \mathbf{c}_{msg}(x) - \mathbf{c}_{aux}(x) \cdot s(x)$.
2. Return the decoded message $m = \mathrm{dec}(\mathbf{m}'(x))$.

Standard-LWE over $\mathbb{Z}_7$

$$\overset{\mathbf{A}}{\begin{bmatrix} 0 & 6 & 2 & 1 \\ 1 & 5 & 0 & 5 \\ 3 & 3 & 3 & 2 \\ 0 & 1 & 5 & 5 \end{bmatrix}} \cdot \overset{\mathbf{s}}{\begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \end{bmatrix}} + \overset{\mathbf{e}}{\begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \end{bmatrix}} = \overset{\mathbf{t}}{\begin{bmatrix} 3 \\ 4 \\ 1 \\ 5 \end{bmatrix}}$$

Ring-LWE over $\mathbb{Z}_7[x]/x^4+1$

$$\begin{array}{cccc} x^3 & x^2 & x & 1 \\ \hline 6 & 2 & 4 & 0 \end{array} \cdot \begin{array}{c|c} 1 & x^3 \\ 0 & x^2 \\ 0 & x \\ 0 & 1 \end{array} + \begin{array}{c|c} 1 & x^3 \\ 1 & x^2 \\ 0 & x \\ 1 & 1 \end{array} = \begin{array}{c|c} 1 & x^3 \\ 2 & x^2 \\ 5 & x \\ 4 & 1 \end{array}$$

Module-LWE over $\mathbb{Z}_7[x]/x^2+1$

$$\begin{array}{cccc} x & 1 & x & 1 \\ \hline 4 & 1 & 3 & 5 \\ 3 & 5 & 4 & 3 \end{array} \cdot \begin{array}{c|c} 1 & x \\ 0 & 1 \\ 1 & x \\ 1 & 1 \end{array} + \begin{array}{c|c} 1 & x \\ 1 & 1 \\ 1 & x \\ 0 & 1 \end{array} = \begin{array}{c|c} 6 & x \\ 0 & 1 \\ 6 & x \\ 3 & 1 \end{array}$$

Chapter 12 of Understanding Cryptography by Christof Paar, Jan Pelzl and Tim Güneysu

Scheme	Type	Equivalent Security	n	k	q	δ
KYBER-512	Module-LWE	AES-128	256	2	3329	2^{-139}
KYBER-768	Module-LWE	AES-192	256	3	3329	2^{-164}
KYBER-1024	Module-LWE	AES-256	256	4	3329	2^{-174}
NEWHOPE-512	Ring-LWE	AES-128	512	1	12289	2^{-213}
NEWHOPE-1024	Ring-LWE	AES-256	1024	1	12289	2^{-216}
FRODOKEM-640	Standard-LWE	AES-128	640	1	2^{15}	$2^{-138.7}$
FRODOKEM-1340	Standard-LWE	AES-256	1340	1	2^{16}	$2^{-252.5}$

Content of this Chapter

- Introduction
- Lattice-Based Cryptography
- **Code-Based Cryptography**

Coding Theory

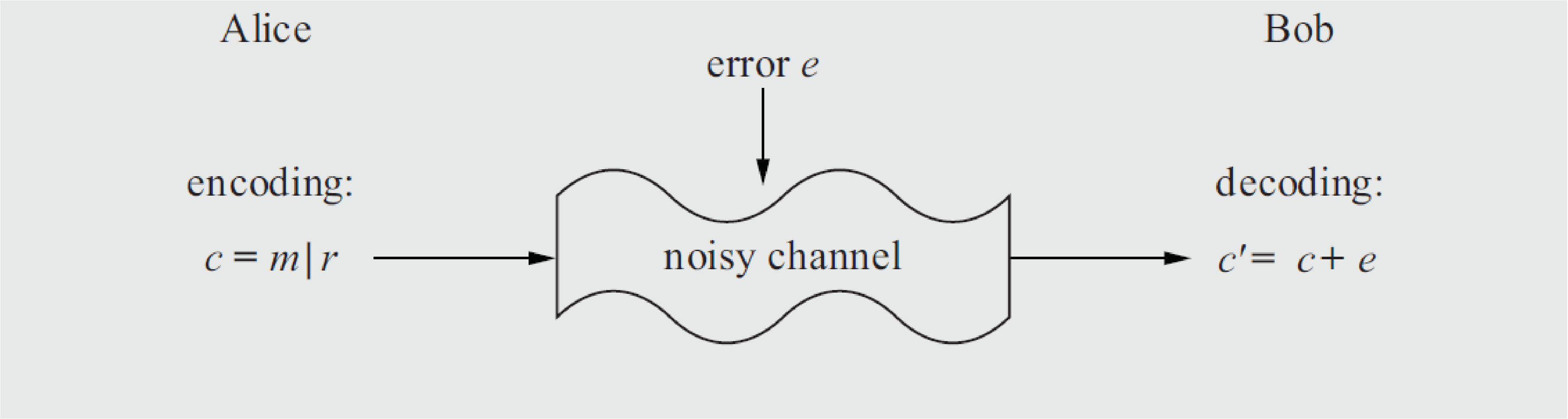


Fig. 12.6 Transfer of a message m over a noisy channel with error-coding

Linear Codes

Alice

Bob

noisy channel

encoding:

$c' = c + e$

decoding:

$$c = m \cdot G \longrightarrow \text{syndrome} \quad s = H \cdot (c')^T$$

$$s \begin{cases} = 0 \longrightarrow \text{no error detected} \\ \neq 0 \longrightarrow \text{error detected} \end{cases}$$

Fig. 12.7 Principle of linear error correction coding

Minimum distance and error correction

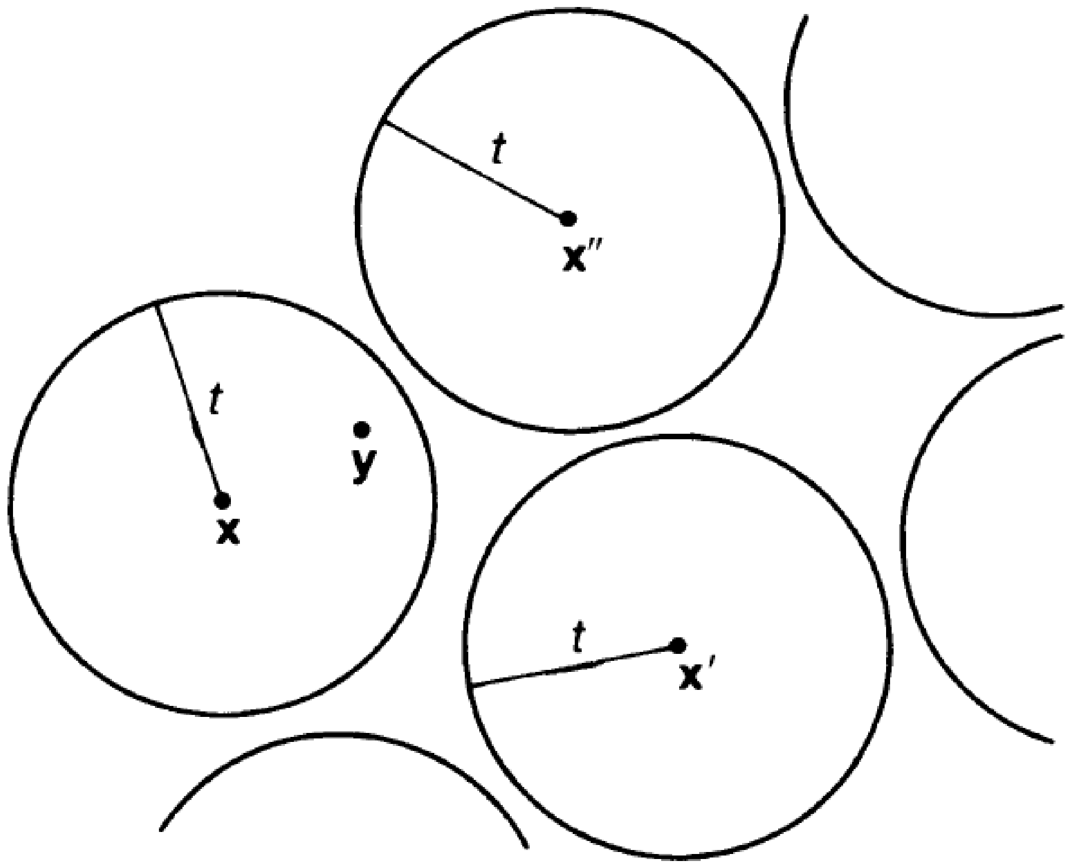
- $d = \min\{HW(c_1 + c_2) : \ c_1, c_2 \in C \ and \ c_1 \neq c_2\}$

For linear codes:

$d = \min\{HW(c) : c \in C \ and \ c \neq \vec{0}\}$

A code can correct

$t = \lfloor (d-1)/2 \rfloor$ errors.



Note: The figure is copied from:

Hill, Raymond. *A first course in coding theory*. Oxford University Press, 1986.

Example 12.7.

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

$$H = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}$$

d=3

$$t = \lfloor (d-1)/2 \rfloor = 1$$

Messages m	Codewords c
$(0\ 0\ 0\ 0)$	$(0\ 0\ 0\ 0\ 0\ 0\ 0)$
$(0\ 0\ 0\ 1)$	$(0\ 0\ 0\ 1\ 1\ 1\ 1)$
$(0\ 0\ 1\ 0)$	$(0\ 0\ 1\ 0\ 1\ 0\ 1)$
$(0\ 0\ 1\ 1)$	$(0\ 0\ 1\ 1\ 0\ 1\ 0)$
$(0\ 1\ 0\ 0)$	$(0\ 1\ 0\ 0\ 1\ 1\ 0)$
$(0\ 1\ 0\ 1)$	$(0\ 1\ 0\ 1\ 0\ 0\ 1)$
$(0\ 1\ 1\ 0)$	$(0\ 1\ 1\ 0\ 0\ 1\ 1)$
$(0\ 1\ 1\ 1)$	$(0\ 1\ 1\ 1\ 1\ 0\ 0)$
$(1\ 0\ 0\ 0)$	$(1\ 0\ 0\ 0\ 0\ 1\ 1)$
$(1\ 0\ 0\ 1)$	$(1\ 0\ 0\ 1\ 1\ 0\ 0)$
$(1\ 0\ 1\ 0)$	$(1\ 0\ 1\ 0\ 1\ 1\ 0)$
$(1\ 0\ 1\ 1)$	$(1\ 0\ 1\ 1\ 0\ 0\ 1)$
$(1\ 1\ 0\ 0)$	$(1\ 1\ 0\ 0\ 1\ 0\ 1)$
$(1\ 1\ 0\ 1)$	$(1\ 1\ 0\ 1\ 0\ 1\ 0)$
$(1\ 1\ 1\ 0)$	$(1\ 1\ 1\ 0\ 0\ 0\ 0)$
$(1\ 1\ 1\ 1)$	$(1\ 1\ 1\ 1\ 1\ 1\ 1)$

Example 12.7.

$$G = \begin{pmatrix} 1&0&0&0&0&1&1 \\ 0&1&0&0&1&1&0 \\ 0&0&1&0&1&0&1 \\ 0&0&0&1&1&1&1 \end{pmatrix}$$

$$H = \begin{pmatrix} 0&1&1&1&1&0&0 \\ 1&1&0&1&0&1&0 \\ 1&0&1&1&0&0&1 \end{pmatrix}$$

Error e	Syndrome s
$(0\,0\,0\,0\,0\,0\,1)$	$(0\,0\,1)$
$(0\,0\,0\,0\,0\,1\,0)$	$(0\,1\,0)$
$(0\,0\,0\,0\,1\,0\,0)$	$(1\,0\,0)$
$(0\,0\,0\,1\,0\,0\,0)$	$(1\,1\,1)$
$(0\,0\,1\,0\,0\,0\,0)$	$(1\,0\,1)$
$(0\,1\,0\,0\,0\,0\,0)$	$(1\,1\,0)$
$(1\,0\,0\,0\,0\,0\,0)$	$(0\,1\,1)$

d=3

$$t = \lfloor (d-1)/2 \rfloor = 1$$

$$c = m \cdot G$$

$$c' = c + e$$

$$c' \rightarrow\ s = H \cdot (c')^T\ \rightarrow syndrome\ table\ \rightarrow e\ \rightarrow c = c' + e\ \rightarrow m$$

Hard problems in Coding Theory

For larger values of t the syndrome table becomes too large.

randomly generated

Let G be a generator matrix of a binary linear code C.

Suppose that C can correct t errors.

$c = m \cdot G$

$c' = c + e\,, \qquad where\ HW(e) \leq t$

<u>Minimum distance decoding problem (MDD problem)</u>

(problem dekodovania podla minimalnej vzdialenosti)

We are given c', G and t.

We want to compute m.

It is assumed that MDD problem is hard even for quantum computers.

Note: For MDD to be hard, it is important that C is randomly generated. There exist some carefully designed classes of linear codes for which MDD is easy.

MDD vs LWE

MDD	LWE
GF(2)	Z_q
G	A
m	s
e with small HW	e containing only small elements
codeword $c = m \cdot G$	lattice point $A \cdot s$
$c' = m \cdot G + e$	$t = A \cdot s + e$

MDD and SDP

An equivalent problem to MDD:

Syndrome decoding problem (SDP problem)

(problem dekodovania podla syndromu)

Subsection 12.3.2 in the book.

Prominent code-based cryptosystems

- HQC (has a similar construction to the Ring-LWE scheme in 12.2.4)
- BIKE
- Classic McEliece