Introduction to Cryptography
Part VI: Encryption based on LWE

Clémence Chevignard & André Schrottenloher

Inria Rennes
Team CAPSULE

Context

Recap

The public-key cryptosystems we have seen so far:

- **Public-key encryption:** RSA, ElGamal
- **Digital signatures:** RSA FDH

are based on:

Recap

The public-key cryptosystems we have seen so far:

- **Public-key encryption:** RSA, ElGamal
- **Digital signatures:** RSA FDH

are based on:

- the **RSA assumption** (RSA)
- the **Decisional Diffie-Hellman** assumption in well-chosen Abelian groups (safe-prime, ECC)

These assumptions:

- have been here for a long time
- are **well understood**
- are **trusted**
- allow quite **efficient** schemes (key sizes, computation time, etc.)

Other schemes

There exists **many other schemes**, depending on **different security assumptions**, some **well understood**, others less (recall: $\text{trust}(t) = \int_0^t \text{cryptanalysis } dt$).

They have been around for as long as RSA / Dlog (ex.: McEliece code-based cryptosystem from 1978), but received **less attention** and did not compete well with them.

With a few exceptions (e.g., hash-based signature standards), anything else than RSA / Dlog was purely theoretical research **until quite recently.**

Quantum computing

Quantum computing is a computational model which is equivalent to Turing machines regarding calculability, but apparently not (we don't have proof) regarding complexity.

- Initiated in the 80s with the prospect of **simulating a complex quantum mechanical system** with a "controlled" one

$\Longrightarrow$ e.g., to understand protein folding

- Could it also be used to speed up **classical computations?**

Deutsch, "Quantum theory, the Church-Turing principle and the universal quantum computer", Proc. R. Soc. Lond. 1985

Shor's algorithm

- In the 1990s, the quantum computing model was well-defined and people started finding **quantum speedups** on some computing tasks.

Shor's algorithm

- In the 1990s, the quantum computing model was well-defined and people started finding **quantum speedups** on some computing tasks.

1994: Shor's algorithm

- Factorization of n-bit integers in $\mathcal{O}\left(n^3\right)$ operations
- Solving n-bit Dlog instances **in any Abelian group** in $\mathcal{O}\left(n^3\right)$ operations

Shor's algorithm

- In the 1990s, the quantum computing model was well-defined and people started finding **quantum speedups** on some computing tasks.

1994: Shor's algorithm

- Factorization of n-bit integers in $\mathcal{O}(n^3)$ operations
- Solving n-bit Dlog instances **in any Abelian group** in $\mathcal{O}(n^3)$ operations

1996: Grover's algorithm

- Solve an exhaustive search problem in the $\sqrt{\cdot}$ of the classical time
- Ex.: n-bit preimage search in $\mathcal{O}(2^{n/2})$

Consequences for cryptography

Quantum computing **is not all-powerful**, but performs surprisingly well for some crypto problems.

Public-key

- Shor's algorithm **completely breaks** RSA and Dlog-based crypto (about 10^9 operations required to factor 2048-bit RSA).

$\Longrightarrow$ need to develop **post-quantum** crypto based on other assumptions.

Secret-key

- Grover's algorithm reduces the generic security levels and the costs of some attacks
- Other attacks occur but they are typically less spectacular than Shor

$\Longrightarrow$ need to patiently evaluate security against quantum attackers.

Post-quantum cryptography

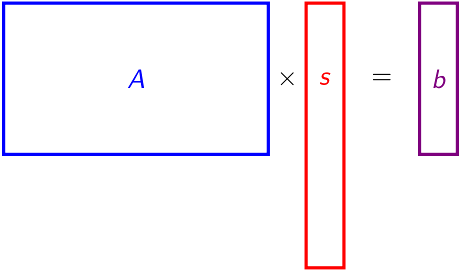
Post-quantum crypto = crypto that remains secure in the presence of a quantum adversary.

This is not science-fiction:

- IBM already possesses quantum computer with a few hundreds qubits.
- The first (NIST) standards for PKE and signatures were completed last year, deployment is ongoing.

Bad-LWE

A bad PKE (do not use it!)



- Choose a public matrix $\mathsf{A} \in \mathbb{Z}_{\mathsf{q}}^{\ell \times \mathsf{n}}$ at random
- Choose $\mathsf{s} \in \mathbb{Z}_{\mathsf{q}}^{\mathsf{n}}$ at random: our private key
- Let $(\mathsf{A}, b := \mathsf{A}\mathsf{s})$ be our public key

Still a bad PKE (do not use it!)

KeyGen:

- Private key: random $s \in \mathbb{Z}_q^n$
- Public key: random matrix A, $As = (a_i \cdot s) := (b_i)$

Encrypt $m \in \{0, 1\}$:

- Pick a random vector $r \in \{0, 1\}^\ell$
- Return $c_1, c_2 := rA, (m + r \cdot b)$

Decrypt $c = (c_1, c_2) \in \mathbb{Z}_q^{n+1}$:

- Return $m = c_2 - c_1 \cdot s$

Still a bad PKE (do not use it!)

KeyGen:

- Private key: random $s \in \mathbb{Z}_q^n$
- Public key: random matrix A, $As = (a_i \cdot s) := (b_i)$

Encrypt $m \in \{0, 1\}$:

- Pick a random vector $r \in \{0, 1\}^\ell$
- Return $c_1, c_2 := rA, (m + r \cdot b)$

Decrypt $c = (c_1, c_2) \in \mathbb{Z}_q^{n+1}$:

- Return $m = c_2 - c_1 \cdot s$

$$
\begin{aligned}
c_2 - c_1 \cdot s &= (m + r \cdot b) - (rA) \cdot s \\
&= m + (r \cdot b) - r(\underbrace{(A \cdot s)}_{=b}) = m
\end{aligned}
$$

Why is this broken?

Why is this broken?

Let's do a Chosen-plaintext attack and always encrypt 0. We observe **samples:**

$$\mathrm{r}\mathrm{A}, (\mathrm{r}\mathrm{A}) \cdot \mathrm{s} \tag{1}$$

for unknown r and s.
After enough samples we have $\boldsymbol{R}, \boldsymbol{R}\mathrm{s}$: invert $\boldsymbol{R}$ to find s.

The scheme is broken because linear algebra is easy. How can we complicate it?

The LWE Problem

"Small" distribution: a discrete Gaussian

Definition

Let $s > 0, c \in \mathbb{R}^n, x \in \mathbb{R}^n$, we define

- $\rho_{s,c}(x) = e^{-\pi\|x-c\|^2/s^2}$,
- $D_{s,c}(x) = \rho_{s,c}(x)/s^n$.

$D_{s,c}$ is the density of probability of the Gaussian distribution of center c and variance $ns^2/(2\pi)$ (of parameter s).

Lemma

Let $s > 0$, $\Pr_{x \leftarrow D_s}\left[\|x\| \geq \sqrt{n}s\right] \leq 2^{-n}$.

We use the Discrete Gaussian to generate small numbers in $\mathbb{Z}_q$ (close to 0).

Learning with errors (LWE)

The LWE distribution $D^{LWE}_{n,q,\alpha}(\boldsymbol{s})$ is the discrete distribution over $\mathbb{Z}_q^{n+1}$ obtained by:

- Sample $\boldsymbol{a} \hookleftarrow U(\mathbb{Z}_q^n)$
- e is sampled through a gaussian distribution on $\mathbb{Z}^\ell$.
- Return $(\boldsymbol{a}, (\boldsymbol{a} \cdot \boldsymbol{s}) + e \mod q)$

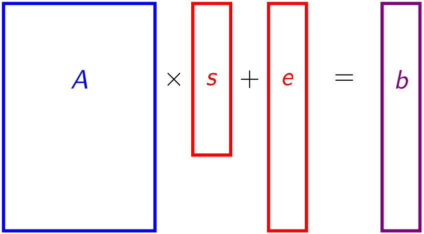
Search-LWE

Let $\boldsymbol{s} \hookleftarrow U(\mathbb{Z}_q^n)$. Given samples from $D^{LWE}_{n,q,\alpha}(\boldsymbol{s})$, find $\boldsymbol{s}$.

Decision-LWE

Let $\boldsymbol{s} \hookleftarrow U(\mathbb{Z}_q^n)$. Distinguish between $D^{LWE}_{n,q,\alpha}(\boldsymbol{s})$ and $U(\mathbb{Z}_q^n \times \mathbb{Z}_q)$.

LWE (ctd.)



With several samples:

- Choose a public matrix $A \in \mathbb{Z}_q^{\ell \times n}$ at random
- Choose $s \in \mathbb{Z}_q^n$ at random **and** a "small" error $e \in \mathbb{Z}_q^\ell$
- Return $A, b := (As + e)$

Theorem
Decision-LWE and search-LWE are equivalent if $q = \mathrm{poly}(n)$.

Regev's PKE

LWE: encryption scheme

Let n, ℓ, q be integers with q prime and $\ell \geq 4(n+1)\log_2 q$ and $\alpha \in]0, 1/(8\ell)[$.

Define:

- **Compress**: decodes an integer mod q into 0 if it's closer to 0 or 1 if it's closer to $\mathsf{q}/2$
- **Decompress**: encodes 0 to 0 and 1 to $\mathsf{q}/2$

KeyGen:

- Private key: random $\mathsf{s} \in \mathbb{Z}_\mathsf{q}^\mathsf{n}$
- Public key: random matrix A, $\mathsf{As} + \mathsf{e} = (\mathsf{a_i} \cdot \mathsf{s} + \mathsf{e_i}) := \mathsf{b_i}$ with e "small" according to discrete Gaussian $D_{\mathbb{Z}^n,\alpha}$

LWE: encryption scheme

Encrypt $m \in \{0,1\}$:

- Pick a random vector $r \in \{0,1\}^\ell$
- Return $c_1, c_2 := rA, (\text{Decompress}(m) + r \cdot b)$

Decrypt $c = (c_1, c_2) \in \mathbb{Z}_q^{n+1}$:

- $m = \text{Compress}(c_2 - c_1 \cdot s)$

Secure

Efficient

Why this works:

$$\begin{aligned} c_2 - c_1 \cdot s &= (\text{Decompress}(m) + r \cdot b) - (rA) \cdot s \\ &= \text{Decompress}(m) + r(As + e) - rAs \\ &= \text{Decompress}(m) + \underbrace{r \cdot e}_{\text{Small}} \end{aligned}$$

Security

Theorem

Regev's PKE is IND-CPA if decisional / search-LWE is hard.

- Is it IND-CCA? No (see TD)
- The scheme is inefficient (1 bit of message only), but serves as a basis for more advanced stuff

Security

Using a Gaussian error, we have a proof that an efficient algorithm for LWE would break a *gap shortest vector problem* on Euclidean lattices.

This is why LWE belongs to **lattice-based crypto**.