Concrete Hardness of LWE & Signature, Encryption Schemes for SNARKS

References

- LWE Hardness
  1. MRS15: https://eprint.iacr.org/2015/046
  2. AFG14: https://eprint.iacr.org/2013/602.pdf
  3. LP11: https://eprint.iacr.org/2010/613
  4. MR09: https://www.cims.nyu.edu/~regev/papers/pqc.pdf
  5. Reg09: http://dl.acm.org/citation.cfm?id=1060603
  6. GN08: ftp://ftp.di.ens.fr/pub/users/pnguyen/Euro08.pdf
- BKZ2.0
  7. CN11: http://link.springer.com/chapter/10.1007%2F978-3-642-25385-0_1
  8. PS13: https://eprint.iacr.org/2013/630.pdf
- Signature
  9. BG14: https://eprint.iacr.org/2013/838
- Encryption
  10. LP11: https://eprint.iacr.org/2010/613

root-Hermite factor

The quality of a basis output by a lattice reduction algorithm is characterised by the Hermite factor δ_0^n, which is defined such that the shortest non-zero vector $\mathbf{b}_0$ in the output basis has the following property: $\|\mathbf{b}_0\| = \delta_0^n \operatorname{vol}(L)^{1/n}$. We may also refer to δ_0 itself, and call it the root-Hermite factor. We call its logarithm to base 2 the log root-Hermite factor.

- root-hermits factor/bit-security Table for BKZ2.0 [CN11, PS13]
-

bit-security	**80**	**128**	**256**
root-Hermite factor	1.0081	1.0067	1.0055

LWE Definition

Definition 1 (LWE [Reg09]). *Let n, q be positive integers, χ be a probability distribution on $\mathbb{Z}$ and $\mathbf{s}$ be a secret vector in $\mathbb{Z}_q^n$. We denote by $L_{\mathbf{s},\chi}$ the probability distribution on $\mathbb{Z}_q^n \times \mathbb{Z}_q$ obtained by choosing $\mathbf{a} \in \mathbb{Z}_q^n$ uniformly at random, choosing $e \in \mathbb{Z}$ according to χ and considering it in $\mathbb{Z}_q$, and returning $(\mathbf{a}, c) = (\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$.*
Decision-LWE *is the problem of deciding whether pairs $(\mathbf{a}, c) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ are sampled according to $L_{\mathbf{s},\chi}$ or the uniform distribution on $\mathbb{Z}_q^n \times \mathbb{Z}_q$.*
Search-LWE *is the problem of recovering $\mathbf{s}$ from $(\mathbf{a}, c) = (\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ sampled according to $L_{\mathbf{s},\chi}$.*

- An instance of LWE(n,a,q), where aq = sigma is the width parameter of the discrete Gaussian
- Application involves as many as m samples. Why no m? The hardness of the LWE problem itself is essentially independent of the number of samples [Reg10]

Strategies to Attack LWE

- Via SIS: Lattice reduction + Distinguish [MR09, LP11]

- Via BDD: Given a basis of a lattice, a target vector, and a bound on the distance from the target to the lattice, find a lattice vector within that bound of the target vector. Solve BDD would solve Search-LWE [LP11]

- Reduce BDD to u-SVP (unique SVP) [AFG14]

Via SIS

- In MR09, LP11

Lemma 6 ([LP11]). *Given an LWE instance characterised by n, α, q and a vector* $\mathbf{v}$ *of length* $\|\mathbf{v}\|$ *in the scaled dual lattice* $L = \{\mathbf{w} \in \mathbb{Z}_q^m \mid \mathbf{w}\mathbf{A} \equiv 0 \mod q\}$*, the advantage of distinguishing* $\langle \mathbf{v}, \mathbf{e} \rangle$ *from random in* $\mathbb{Z}_q$ *is close to* $\exp(-\pi \cdot (\|\mathbf{v}\| \cdot \alpha)^2)$.

- In LP11, MRS15

Corollary 2. *To obtain a probability ϵ of success in solving an LWE instance parametrised by n, q and α via the SIS strategy, we require a vector* $\mathbf{v}$ *with* $\|\mathbf{v}\| = \frac{1}{\alpha}\sqrt{\ln(\frac{1}{\epsilon})/\pi}$.

let $f(\epsilon)$ denote $\sqrt{\ln(\frac{1}{\epsilon})/\pi}$.

- |v| = f(epsilon)/a

Via SIS

- In LP11, MRS15

Lemma 7. *Let an LWE instance be parametrised by n, α, q. Any lattice reduction algorithm achieving log root-Hermite factor*

$$\log \delta_0 = \frac{\log^2\left(\alpha \frac{1}{f(\epsilon)}\right)}{4n \log q}$$

can distinguish $L_{\mathbf{s},\chi}$ with probability ϵ.

Proof. With high probability $\mathrm{vol}(L) = q^n$ and by definition the Hermite factor is $\delta_0^m = \frac{\|\mathbf{v}\|}{\mathrm{vol}(L)^{\frac{1}{m}}}$ so we have $\|\mathbf{v}\| = \delta_0^m q^{\frac{n}{m}}$. On the other hand, we require $\|\mathbf{v}\| = \frac{1}{\alpha} f(\epsilon)$ by Corollary 2. By Section 3.3 the optimal subdimension m which minimises the quantity $\delta_0^m q^{\frac{n}{m}}$ is $m = \sqrt{\frac{n \log q}{\log \delta_0}}$. Since we assume we can choose any number of samples m, we always choose to use this optimal subdimension. Rearranging with this value of m, we obtain $\log \delta_0 = \frac{\log^2\left(\frac{1}{\alpha} f(\epsilon)\right)}{4n \log q} = \frac{\log^2\left(\alpha \frac{1}{f(\epsilon)}\right)}{4n \log q}$ as our desired log root-Hermite factor. □

Via BDD

- Basic way to solve BDD is Babai's Nearest Plane algorithm.
- [LP11] proposes an improved algorithm upon it.
- Idea: lattice-reduction algorithm + apply BDD-solver
    1. Apply BKZ2.0 to get the reduced basis with certain root-Hermite factor value, say delta=1.01
    2. Apply LP-BDD solver on the reduced basis
    3. As long as the time cost in step 2 < in step 1. Stop
- Problem: Hard to optimize all the parameters. No analysis regarding the bit-security. So far, we only have running time by experiments. [LP11]

Reduce BDD to u-SVP

Albrecht, Fitzpatrick and Göpfert [AFG14] consider the complexity of solving LWE by reducing BDD to uSVP (unique Shortest Vector Problem). Formally, the γ-uSVP problem is as follows: given a lattice L such that $\lambda_2(L) > \gamma\lambda_1(L)$, find a shortest nonzero vector in L.

- In practice, an Algorithm solving HSVP will solve u-SVP instances where the gap [GN08]

$\lambda_2(L) > \tau\delta_0^m\lambda_1(L)$ with some probability depending on τ

Reduce BDD to u-SVP

Lemma 8 (Lemma 2 in [AFG14]). *Let* $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$*, let* $\alpha q > 0$ *and let* $\epsilon' > 1$*. Let* $\mathbf{e} \in \mathbb{Z}_q^m$ *such that each component is drawn from* χ *and considered mod* q*. Under the assumption that* $\lambda_1(L(\mathbf{A})) \geq \sqrt{\frac{m}{2\pi e}}\mathrm{vol}(L)^{1/m}$ *and that the rows of* $\mathbf{A}$ *are linearly independent over* $\mathbb{Z}_q$*, we can create an embedding lattice with* λ_2/λ_1*-gap greater than*

$$\frac{\min\{q, \frac{q^{1-\frac{n}{m}}\Gamma(1+\frac{m}{2})^{\frac{1}{m}}}{\sqrt{\pi}}\}}{\frac{\epsilon' s\sqrt{m}}{\sqrt{\pi}}} \approx \frac{\min\{q, q^{1-\frac{n}{m}}\sqrt{\frac{m}{2\pi e}}\}}{\frac{\epsilon' s\sqrt{m}}{\sqrt{\pi}}}$$

with probability greater than $1 - \left(\epsilon' \cdot \exp\left(\frac{1-\epsilon'^2}{2}\right)\right)^m$.

- We require a gap of size approximately $\frac{\lambda_2}{\lambda_1} = \frac{q^{1-\frac{n}{m}}\sqrt{\frac{1}{2e}}}{\epsilon' \alpha q}$
- which needs to be > tau delta^m

Reduce BDD to u-SVP

- In [MRS15]

Lemma 9. *Given an LWE instance characterised by* n, α, q. *Any lattice reduction algorithm achieving log root-Hermite factor*

$$\log \delta_0 = \frac{\log^2\left(\tau\alpha\sqrt{2e}\right)}{4n\log q}$$

solves LWE with success probability greater than

$$\epsilon_\tau \cdot \left(1 - \left(\epsilon' \cdot \exp\left(\frac{1-\epsilon'^2}{2}\right)\right)^m\right)$$

for some $\epsilon' \approx 1$ *and some fixed* $\tau \leq 1$, *and* $0 < \epsilon_\tau < 1$ *as a function of* τ.

Concrete LWE Hardness

- Two approaches to analyze the bit security of LWE

1. lwe_sis(n,a,q): via SIS [MR09, LP11]
2. lwe_gap(n,a,q): BDD reduce to u-SVP [AFG14, MRS15]
3. Set epsilon=0.1, tau=0.4 [LP11, GN08]
4. Given LWE(n,a,q), calculate delta = max {lwe_sis(n,a,q), lwe_gap(n,a,q)}

BG14/Telsa Signature

Algorithm 1 Key generation

INPUT: $n, m, k, q, \sigma_S, \sigma_E$
OUTPUT: $\mathbf{A}, \mathbf{T}$

$\mathbf{A} \leftarrow \mathbb{Z}_q^{m \times n}$
$\mathbf{S} \leftarrow D_S^{n \times k}$
$\mathbf{E} \leftarrow D_E^{m \times k}$
if $|\mathbf{E}_{i,j}| > 7\sigma_E$ for any (i, j) **then**
    Restart
end if
$\mathbf{T} \equiv \mathbf{AS} + \mathbf{E} \pmod q$
return $\mathbf{A}, \mathbf{T}$

Algorithm 2 Signing

INPUT:
$\mu, \mathbf{A}, \mathbf{T}, \mathbf{S}, D_y, D_z, d, w, \sigma_E, H, F, M$
OUTPUT: $(\mathbf{z}, c)$

$\mathbf{y} \leftarrow D_y^n$
$\mathbf{v} \equiv \mathbf{Ay} \pmod q$
$c = H(\lfloor \mathbf{v} \rceil_d, \mu)$
$\mathbf{c} = F(c)$
$\mathbf{z} = \mathbf{y} + \mathbf{Sc}$
$\mathbf{w} \equiv \mathbf{Az} - \mathbf{Tc} \pmod q$
if $|[\mathbf{w}_i]_{2^d}| > 2^{d-1} - 7w\sigma_E$ **then**
    Restart
end if
return $(\mathbf{z}, c)$ with probability $\min\left(D_z^n(\mathbf{z})/(M \cdot D_{y,\mathbf{Sc}}^n(\mathbf{z})), 1\right)$

Algorithm 3 Verifying

INPUT: $\mu, \mathbf{z}, c, \mathbf{A}, \mathbf{T}, \ell, B, d, H, F$
OUTPUT: Accept or Reject

$\mathbf{c} = F(c)$
$\mathbf{w} \equiv \mathbf{Az} - \mathbf{Tc} \pmod q$
$c' = H(\lfloor \mathbf{w} \rceil_d, \mu)$
if $c' = c$ and $\|\mathbf{z}\|_\ell \leq B$ **then**
    return "Accept"
else
    return "Reject"
end if

Verify Cost of BG14

- compute w: m*n+m*k
- check |z|<=B: n log(B)
- random oracle: a sequence of

80 bit-secuity Ajtai hash until the output size <=512

+ SHA256.

Algorithm 3 Verifying

INPUT: $\mu, \mathbf{z}, c, \mathbf{A}, \mathbf{T}, \ell, B, d, H, F$

OUTPUT: Accept or Reject

$\mathbf{c} = F(c)$

$\mathbf{w} \equiv \mathbf{Az} - \mathbf{Tc} \pmod{q}$

$c' = H\left(\lfloor \mathbf{w} \rceil_d,\, \mu\right)$

if $c' = c$ and $\|\mathbf{z}\|_\ell \leq B$ **then**

    return "Accept"

else

    return "Reject"

end if

Security proof of BG14

Theorem 3. *Let q be prime. Let the parameters be chosen such that $B, 2^d \geq 14\alpha q$ and*

$$q^m \geq (4B+1)^n (2^{d+1})^m 2^\kappa. \tag{6}$$

Suppose equation (2) holds. Let $D_y = [-B, B]$ with the uniform distribution and let $\boldsymbol{S}, \boldsymbol{E}$ have entries chosen from discrete Gaussian distributions with standard deviation $\sigma_S = \sigma_E = \alpha q$. Let A be a forger against the signature scheme in the random oracle model that makes h hash queries, s sign queries, runs in time t and succeeds with probability δ. Then A can be turned into an algorithm that solves (n, m, q, α)-decisional-LWE, running in time approximately $2t$, and with success probability at least

$$\min_{0<\delta'<\delta} \max \left\{ |\delta - \delta'|, \frac{\delta'}{h}\left(\frac{\delta'}{h} - \frac{1}{2^\kappa}\right) + O\left(\frac{s(s+h)}{2^\kappa} + \frac{m+n}{2^{140}}\right) \right\}.$$

Parameter Configuration

1. set q= q_snark, k=kappa=256

2. start from the small n.

3. start from small d. compute the smallest m,B, a such that the constraint in (6) holds

4. Run LWE(n,a,q) to get delta.

5. If delta<delta_80, compute veriy-cost(n,m,k,a,q,B)

6. return the configuration LWE(n,m,a,q) with the lowest verify-cost

Concrete Configuration

n	m	a	bound checking	computation	random oracle	total cost
14	33	2.44*10-4	3441	8910	33088	45439

Encryption Scheme [LP11]

- $\mathsf{Gen}(\bar{\mathbf{A}}, 1^\ell)$: choose $\mathbf{R}_1 \leftarrow D_{\mathbb{Z},s_k}^{n_1 \times \ell}$ and $\mathbf{R}_2 \leftarrow D_{\mathbb{Z},s_k}^{n_2 \times \ell}$, and let $\mathbf{P} = \mathbf{R}_1 - \bar{\mathbf{A}} \cdot \mathbf{R}_2 \in \mathbb{Z}_q^{n_1 \times \ell}$. The public key is $\mathbf{P}$ (and $\bar{\mathbf{A}}$, if needed), and the secret key is $\mathbf{R}_2$.

  In matrix form, the relationship between the public and secret keys is:

$$[\bar{\mathbf{A}} \quad \mathbf{P}] \cdot \begin{bmatrix} \mathbf{R}_2 \\ \mathbf{I} \end{bmatrix} = \mathbf{R}_1 \bmod q. \tag{3.1}$$

- $\mathsf{Enc}(\bar{\mathbf{A}}, \mathbf{P}, \mathbf{m} \in \Sigma^\ell)$: choose $\mathbf{e} = (\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3) \in \mathbb{Z}^{n_1} \times \mathbb{Z}^{n_2} \times \mathbb{Z}^\ell$ with each entry drawn independently from $D_{\mathbb{Z},s_e}$. Let $\bar{\mathbf{m}} = \mathsf{encode}(\mathbf{m}) \in \mathbb{Z}_q^\ell$, and compute the ciphertext

$$\mathbf{c}^t = [\mathbf{c}_1^t \quad \mathbf{c}_2^t] = [\mathbf{e}_1^t \quad \mathbf{e}_2^t \quad \mathbf{e}_3^t + \bar{\mathbf{m}}^t] \cdot \begin{bmatrix} \bar{\mathbf{A}} & \mathbf{P} \\ \mathbf{I} & \\ & \mathbf{I} \end{bmatrix} \in \mathbb{Z}_q^{1 \times (n_2 + \ell)}. \tag{3.2}$$

  (Note that the first ciphertext component $\mathbf{c}_1^t$ can be precomputed before $\mathbf{P}$ and $\mathbf{m}$ are known.)

- $\mathsf{Dec}(\mathbf{c}^t = [\mathbf{c}_1^t, \mathbf{c}_2^t], \mathbf{R}_2)$: output $\mathsf{decode}(\mathbf{c}_1^t \cdot \mathbf{R}_2 + \mathbf{c}_2^t)^t \in \Sigma^\ell$.

  Using Equation (3.2) followed by Equation (3.1), we are applying decode to

$$[\mathbf{c}_1^t \quad \mathbf{c}_2^t] \cdot \begin{bmatrix} \mathbf{R}_2 \\ \mathbf{I} \end{bmatrix} = (\mathbf{e}^t + [\mathbf{0} \quad \mathbf{0} \quad \bar{\mathbf{m}}^t]) \cdot \begin{bmatrix} \mathbf{R}_1 \\ \mathbf{R}_2 \\ \mathbf{I} \end{bmatrix} = \mathbf{e}^t \cdot \mathbf{R} + \bar{\mathbf{m}}^t,$$

  where $\mathbf{R} = \left[\begin{smallmatrix} \mathbf{R}_1 \\ \mathbf{R}_2 \\ \mathbf{I} \end{smallmatrix}\right]$. Therefore, decryption will be correct as long as each $|\langle \mathbf{e}, \mathbf{r}_j \rangle| < t$, the error threshold of decode. (We give a formal analysis in Section 3.2 below.)

Encryption Scheme [LP11]

- encode(m) := m* floor(q/2)   cost 1
- decode(c):= 0 if |c|<=floor(q/4) cost log(q)

(apart from draw samples from discrete Gaussian)

- Enc Cost: n1x(n2+1)+1
- Dec Cost: n2

Security proof of LP11

Theorem 3.2. *The cryptosystem from Section 3.1 is CPA-secure, assuming the hardness of decision-*LWE *with modulus q for: (i) dimension n_2 with error distribution $D_{\mathbb{Z},s_k}$, and (ii) dimension n_1 with error $D_{\mathbb{Z},s_e}$.*

Proof. It suffices to show that the entire view of the adversary in an IND-CPA attack is computationally indistinguishable from uniformly random, for any encrypted message $\mathbf{m} \in \Sigma^\ell$. The view consists of $(\bar{\mathbf{A}}, \mathbf{P}, \mathbf{c})$, where $\bar{\mathbf{A}} \in \mathbb{Z}_q^{n_1 \times n_2}$ is uniformly random, $\mathbf{P} \leftarrow \mathsf{Gen}(\bar{\mathbf{A}}, 1^\ell)$, and $\mathbf{c}^t \leftarrow \mathsf{Enc}(\bar{\mathbf{A}}, \mathbf{P}, \mathbf{m})$. First, $(\bar{\mathbf{A}}, \mathbf{P})$ is computationally indistinguishable from uniformly random $(\bar{\mathbf{A}}, \mathbf{P}^*) \in \mathbb{Z}_q^{n_1 \times (n_2+\ell)}$ under assumption (i) in the lemma statement, because $\mathbf{P} = (\bar{\mathbf{A}}^t)^t \cdot (-\mathbf{R}_2) + \mathbf{R}_1$, and $\bar{\mathbf{A}}^t$ is uniform while the entries of both $-\mathbf{R}_2$ and $\mathbf{R}_1$ are drawn from $D_{\mathbb{Z},s_k}$. So the adversary's view is indistinguishable from $(\mathbf{A}, \mathbf{c})$ where $\mathbf{A} = (\bar{\mathbf{A}}, \mathbf{P}^*)$ is uniformly random and $\mathbf{c} \leftarrow \mathsf{Enc}(\mathbf{A}, \mathbf{m})$. Now $(\mathbf{A}, \mathbf{c})$ is also computationally indistinguishable from uniformly random $(\mathbf{A}, \mathbf{c}^*)$ under assumption (ii) in the lemma statement, because $\mathbf{c} = (\mathbf{A}^t \mathbf{e}_1 + \left[\begin{smallmatrix}\mathbf{e}_2 \\ \mathbf{e}_3\end{smallmatrix}\right]) + \left[\begin{smallmatrix}\mathbf{0} \\ \mathbf{m}\end{smallmatrix}\right]$, and $\mathbf{A}$ is uniform while the entries of $\mathbf{e}_1$, $\mathbf{e}_2$, and $\mathbf{e}_3$ are drawn from $D_{\mathbb{Z},s_e}$. $\square$

- Break the LP11 crypto-system would solve an instance of LWE(n1, sigma_k/q, q)

- or, an instance of LWE(n2+1, sigma_e/q, q)

Ahemd’s Subset Sum Circuit

```
@Override
protected int[] buildCircuit() {

    int[] outDigest = new int[dimension];
    Arrays.fill(outDigest, generator.getZeroWire());

    int[] inputWires = padded_ins;

    for (int i = 0; i < dimension; i++) {
        for (int j = 0; j < max_m_length; j++) {
            outDigest[i] = generator.add(outDigest[i], generator.mulConst(
                    inputWires[j], coeffs[i][j], false, ""), "");
        }
    }

    int[] outWires;
    if (!binaryOutput) {
        outWires = outDigest;
    } else {
        outWires = new int[dimension * 256];
        for (int i = 0; i < dimension; i++) {
            int[] bits = generator.split(outDigest[i], 254, "");
            for (int j = 0; j < bits.length; j++) {
                outWires[j + i * 256] = bits[j];
            }
            outWires[bits.length + i * 256] = generator.getZeroWire();
            outWires[bits.length + i * 256 + 1] = generator.getZeroWire();
        }
    }

    return outWires;
}

}
```