Cryptanalysis of Compact-LWE

Jonathan Bootle[1] and Mehdi Tibouchi[2]

[1] University College London
jonathan.bootle.14@ucl.ac.uk
[2] NTT Secure Platform Laboratories
tibouchi.mehdi@lab.ntt.co.jp

Abstract. As an invited speaker of the ACISP 2017 conference, Dongxi Liu recently introduced a new lattice-based encryption scheme (joint work with Li, Kim and Nepal) designed for lightweight IoT applications, and announced plans to submit it to the NIST postquantum competition. The new scheme is based on a variant of standard LWE called Compact-LWE, but is claimed to achieve high security levels in considerably smaller dimensions than usual lattice-based schemes. In fact, the proposed parameters, allegedly suitable for 138-bit security, involve the Compact-LWE assumption in dimension only 13.

In this note, we show that this particularly aggressive choice of parameters fails to achieve the stated security level. More precisely, we show that ciphertexts in the new encryption scheme can be decrypted using the public key alone with $> 99.9\%$ probability in a fraction of a second on a standard PC, which is not quite as fast as legitimate decryption, but not too far off.

Keywords: Compact-LWE, lattice-based cryptography, cryptanalysis, lattice reduction, IoT.

1 Introduction

Lattice-based cryptography stands out as one of the main candidates for constructing quantum-secure cryptographic primitives, thanks to its versatility (almost all cryptography, from encryption and signatures all the way to predicate encryption and FHE, can be instantiated under lattice assumptions) and its strong postquantum security guarantees (such as worst-case to average-case reductions) [Pei15]. However, early provably secure lattice-based schemes, such as Regev's encryption scheme from standard LWE [Reg05], tended to be quite inefficient, due to the large key sizes needed to achieve security, and to a lesser extent the need to sample from distributions like discrete Gaussians, which is difficult to do in a secure and efficient manner.

As a result, there has been a movement towards increasingly optimized lattice-based schemes, with the goal of making lattice-based cryptography a viable alternative to current RSA and discrete logarithm-based deployments, preferably even on low-end and constrained devices. Those schemes are often

based on variants of the LWE problem with much smaller key sizes (such as Ring-LWE [LPR10]; one could also see NTRU-based constructions as belonging to this category, although NTRU itself predates LWE-based cryptography) and incorporate a range of speed-ups for practical implementations. For encryption in particular, recent proposals include NewHope [ADPS17], NTRU Prime [BCLvV16], Lizard [CKLS16] and Kyber [BDK+17]. They have been implemented on a wide range of platforms, and although the security guarantees they offer may not be quite as strong as standard LWE-based schemes, they are very conservatively designed, and their security claims are unlikely to be falsified short of spectacular advances in the analysis of lattice problems.

On the other hand, in the quest for faster lattice-based cryptography, more exotic variants of LWE and more aggressive parameter choices have also been considered in the literature, and occasionally been broken. This includes a collection of more or less artificial parameter choices for Ring-LWE [Pei16], LWE variants with very small matrix entries [HM17], so-called "overstretched" variants of NTRU [ABD16, KF17] and more.

A recent example of a particularly aggressive parameter choice for a scheme based on a non-standard LWE variant is the Compact-LWE encryption scheme of Liu, Li, Kim and Nepal [LLKN17], which was awarded an invited talk slot at the ACISP 2017 conference [Liu17], and which the authors plan to enter into the NIST postquantum competition. The scheme is designed for IoT applications, and does achieve rather impressive performance on low-cost embedded microcontrollers. However, it does so based on the use of surprisingly small parameters; in particular, the computations are carried out in dimension 13. According to the authors' analysis, the scheme should nevertheless offer 138 bits of security, due to their underlying assumption being immune to usual attacks against lattice-based constructions.

Our contributions. Analyzing the security of LWE variants is important, particularly when they are proposed for use in very concrete real world settings, as is the case for Liu et al.'s Compact-LWE assumption and the corresponding encryption scheme. Unfortunately, our analysis reveals that the security claims of the proposed scheme are overly optimistic.

More precisely, plaintexts in Compact-LWE encryption are masked by a low-weight linear combination of the vectors in the public key (essentially a subset sum). Due to the very low dimension of the problem, we find that it is easy to recover the coefficients of this subset sum given only a ciphertext and the corresponding public key. Based on experiments using the Sage computer algebra software on a desktop PC, we find that our algorithm correctly decrypts a ciphertext with the public key alone in a fraction of a second with $> 99.9\%$ success rate. The source code for the entire attack is provided as an appendix to this paper.

2 Preliminaries

2.1 Notation

The authors of [LLKN17] denote by $\mathbb{Z}_\ell$ denote the set of integers $\{0, 1, \ldots, \ell-1\}$, for any positive integer ℓ. Although this notation can be regarded as questionable, we use it in this note for the sake of consistency with the original paper.

2.2 The Compact-LWE Encryption Scheme

Liu et al. [LLKN17] propose an encryption scheme based a variant of the LWE problem in which the errors are scaled by a fixed secret value, and the sample vectors $\boldsymbol{a}$ have small coefficients. The underlying hardness assumption, called Decision Compact-LWE, is exactly the semantic security of the encryption scheme, so we omit its definition, and simply describe the encryption scheme itself directly.

Public parameters. The public parameters of the scheme are given the tuple of positive integers $pp = (q, n, m, t, w, b)$, which should satisfy the following constraints:

$$n + 1 < m < n^2, \quad 2b(b \log_2 b + 1) < q \quad \text{and} \quad 2 \log_2 b < n.$$

Key generation. Sample $\boldsymbol{s}$ uniformly at random from $\mathbb{Z}_q^n$ and choose sk, r, p from $\mathbb{Z}_q$ subject to the following constraints:

$$t \leq p, \quad sk \cdot (t-1) + wrp < q, \quad b < r,$$

and the integers sk, p, q are pairwise coprime. The private key is then $\mathbf{K} = (\boldsymbol{s}, sk, r, p)$.

To construct the public key, sample m vectors $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_m$ uniformly at random from $\mathbb{Z}_b^n$, and m noise values $e_1, \ldots, e_m$ uniformly at random from $\mathbb{Z}_r$. Compute the corresponding Compact-LWE samples as follows:

$$(\boldsymbol{a}_i, b_i) = \left(\boldsymbol{a}_i, \langle \boldsymbol{a}_i, \boldsymbol{s} \rangle + sk_q^{-1} \cdot p \cdot e_i \bmod q\right),$$

where $sk_q^{-1} \in \mathbb{Z}_q$ denotes the multiplicative inverse of sk modulo q. The public key $\mathbf{PK}$ then consists of the collection of all pairs $(\boldsymbol{a}_i, b_i)$ for $1 \leq i \leq m$.

We note that the key generation algorithm is not completely well-defined by the above (and hence by the original paper [LLKN17]), since the precise distribution of (sk, r, p) is not specified. In our experiments, we generate them as follows: r is first picked uniformly at random such that $2 \leq r < q/(wt)$; then p is sampled uniformly among integers coprime to r such that $t \leq p < q/(rw)$; and finally, sk is sampled uniformly among integers coprime to r and p such that $1 \leq sk < (q - wrp)/(t-1)$. However, other distributions should have little or no impact on the effectiveness of our attack.

Table 1. Parameters proposed by Liu et al. [LLKN17] for their Compact-LWE encryption scheme.

q	t	m	w	n	b
2^{32}	2^{16}	74	86	13	16

Encryption. Let v be a value from the plaintext space $\mathbb{Z}_t$. The encryption algorithm produces a ciphertext $\boldsymbol{c} = \mathrm{Enc}(\mathbf{PK}, v)$ in $\mathbb{Z}_q^{n+1}$ as follows.

Choose w indices $i_1, \ldots, i_w$ in $\{1, \ldots, m\}$ uniformly and independently at random (in particular, they are not necessarily distinct), and let:

$$(\boldsymbol{a}, b) = \sum_{k=1}^{w} \left(\boldsymbol{a}_{i_k}, b_{i_k}\right)$$

be the sum of the corresponding Compact-LWE samples from the public key $\mathbf{PK}$. Then, output the ciphertext $\boldsymbol{c}$ given by:

$$\boldsymbol{c} = (\boldsymbol{a}, v - b \bmod q).$$

Decryption. Given a ciphertext $\boldsymbol{c} = (\boldsymbol{a}, x)$, the decryption algorithm recovers the corresponding plaintext $\mathrm{Dec}(\boldsymbol{K}, \boldsymbol{c}) = v \in \mathbb{Z}_t$ as follows:

$$v = sk_p^{-1} \cdot \Big(sk \cdot \big(\langle \boldsymbol{a}, \boldsymbol{s} \rangle + x\big) \bmod q\Big) \bmod p,$$

where sk_p^{-1} denotes the multiplicative inverse of sk modulo p.

Proposed parameters. The authors of [LLKN17] propose to instantiate their schemes with the parameters given in Table 1. As noted in the introduction, the most remarkable aspect of those parameters is the extremely small dimension $n = 13$ in which the computations are carried out. This makes the scheme quite fast and compact, but raises concerns regarding security, which the next section will show are well-warranted.

3 Attack on Compact-LWE Encryption

In this section, we describe our attack on the encryption scheme of §2.2. More precisely, we show that it is possible to decrypt ciphertexts using only the information contained in the public key.

As we have have seen, ciphertexts are of the form $(\boldsymbol{a}, v - b \bmod q)$ where $(\boldsymbol{a}, b)$ is the sum of w randomly chosen elements (Compact-LWE samples) from the public key. To decrypt, it suffices to recover the correct linear combination of public key elements used to compute the ciphertext. Now, the plaintext v is

small (it satisfies $0 \leq v < t$), whereas the ciphertext mask value b is a full-size value in $\mathbb{Z}_q$. Therefore, one can try to decrypt a ciphertext $(\boldsymbol{a}, x)$ by looking for a vector of coefficients $\boldsymbol{u} = (u_1, \ldots, u_m)$ such that:

$$\boldsymbol{a} = \sum_{i=1}^{m} u_i \boldsymbol{a}_i \text{ in } \mathbb{Z}^n, \quad x \text{ is close to } -\sum_{i=1}^{m} u_i b_i \text{ modulo } q,$$

and the vector $\boldsymbol{u}$ is small (in fact, the correct linear combination satisfies $u_i \geq 0$ for all i and $\sum u_i = w$). The problem of finding such a vector $\boldsymbol{u}$ can be expressed as a lattice problem.

Attack strategy. More precisely, denote by $A \in \mathbb{Z}^{m \times n}$ the matrix whose rows are the public key vectors $\boldsymbol{a}_i$, and $\boldsymbol{b} \in \mathbb{Z}^m$ the column vector of the b_i's. Then, we can consider the lattice $L \subset \mathbb{Z}^{m+n+2}$ generated by the rows of the following matrix, which depends only on the ciphertext $(\boldsymbol{a}, x)$, the public key $\mathbf{PK}$ and the public parameters pp:

$$M = M(pp, \mathbf{PK}, \boldsymbol{a}, x) = \begin{pmatrix} 1 & \mathbf{0} & \kappa \boldsymbol{a} & x \\ \mathbf{0} & tI_m & -\kappa A & \boldsymbol{b} \\ 0 & \mathbf{0} & \mathbf{0} & q \end{pmatrix}$$

where κ is some suitably large constant, say $\kappa = q$. Now if $\boldsymbol{u} \in \mathbb{Z}^m$ is the vector of coefficients used to construct the ciphertext $(\boldsymbol{a}, x)$, i.e. $(\boldsymbol{a}, x) = (\boldsymbol{u}^T A, v - \langle \boldsymbol{u}, \boldsymbol{b} \rangle \bmod q)$, then the following vector $\widetilde{\boldsymbol{u}}$ belongs to the lattice L:

$$\widetilde{\boldsymbol{u}} = (1, tu_1, \ldots, tu_m, 0, \ldots, 0, v) = (1, t\boldsymbol{u}, \mathbf{0}, v).$$

Indeed, we have:

$$(1, \boldsymbol{u}, \alpha) \cdot M = \big(1, t\boldsymbol{u}, \kappa(\boldsymbol{a} - \boldsymbol{u}^T A), x + \langle \boldsymbol{u}, \boldsymbol{b} \rangle + \alpha q\big) = \big(1, t\boldsymbol{u}, \mathbf{0}, v + (\alpha + \beta) q\big)$$

where β is the quotient in the Euclidean division of $(x + \langle \boldsymbol{u}, \boldsymbol{b} \rangle)$ by q. By choosing $\alpha + \beta = 0$, we obtain that $\widetilde{\boldsymbol{u}} \in L$ as desired.

Thus, the correct vector $\boldsymbol{u}$ corresponds to a vector in the lattice L, which is moreover relatively short: all of its coefficients are bounded by a small multiple of t, and are in particular a lot smaller than q. Conversely, consider a lattice vector $\widetilde{\boldsymbol{u}'} \in L$ whose first coefficient is 1, and satisfying $\|\widetilde{\boldsymbol{u}'}\| < q/2$. Clearly, $\widetilde{\boldsymbol{u}'}$ must be of the form $(1, \boldsymbol{u}', \alpha') \cdot M$ for some $\boldsymbol{u}' \in \mathbb{Z}^m$ and $\alpha' \in \mathbb{Z}$. Thus:

$$\widetilde{\boldsymbol{u}'} = \big(1, t\boldsymbol{u}', \kappa(\boldsymbol{a} - (\boldsymbol{u}')^T A), x + \langle \boldsymbol{u}', \boldsymbol{b} \rangle + \alpha' q\big),$$

and we must have $\boldsymbol{a} - (\boldsymbol{u}')^T A = \mathbf{0}$, since otherwise the vector $\widetilde{\boldsymbol{u}'}$ would have coefficients of absolute value at least $\kappa = q$, contradicting the bound on the norm. As a result, the vector $\boldsymbol{u}'$ must be of the form $\boldsymbol{u} + \boldsymbol{z}$, where $\boldsymbol{z}$ is in the left kernel of the matrix $A \in \mathbb{Z}^{m \times n}$. This gives:

$$\widetilde{\boldsymbol{u}'} = \big(1, t\boldsymbol{u} + t\boldsymbol{z}, \mathbf{0}, v + \langle \boldsymbol{z}, \boldsymbol{b} \rangle \bmod^* q\big)$$

where we denote by mod^* the centered modulo operator (the last coefficient is necessarily of that form due to the constraint on the norm of $\widetilde{\boldsymbol{u}'}$. Note furthermore that since $\boldsymbol{b} = A\boldsymbol{s} + sk_q^{-1} \cdot p \cdot \boldsymbol{e} \bmod q$, we have:

$$\langle \boldsymbol{z}, \boldsymbol{b} \rangle \equiv \boldsymbol{z}^T A\boldsymbol{s} + sk_q^{-1} \cdot p\langle \boldsymbol{z}, \boldsymbol{e} \rangle \equiv sk_q^{-1} \cdot p\langle \boldsymbol{z}, \boldsymbol{e} \rangle \pmod q.$$

Consider now a *short* vector $\widetilde{\boldsymbol{u}'}$, in the sense that its magnitude is roughly that of $\widetilde{\boldsymbol{u}}$ or smaller. Then, in particular, $\boldsymbol{z}$ should be a very short vector in the left-kernel of A (because $t(\boldsymbol{u} + \boldsymbol{z})$ is short), and we can thus expect $\left|\langle \boldsymbol{z}, \boldsymbol{e} \rangle\right|$ to be small, say less than half of wr, the bound satisfied by $\langle \boldsymbol{u}, \boldsymbol{e} \rangle$ (one expects a better bound because contrary to $\boldsymbol{u}$, $\boldsymbol{z}$ need not have all positive coefficients). Moreover, the last coefficient

$$v' = v + sk_q^{-1} \cdot p\langle \boldsymbol{z}, \boldsymbol{e} \rangle \bmod^* q$$

of $\widetilde{\boldsymbol{u}'}$ should also be small, say less than $q/(2sk)$. In that case, we have $sk \cdot v' \equiv sk \cdot v + p\langle \boldsymbol{z}, \boldsymbol{e} \rangle \pmod q$, and if v is in the first half of the allowed range, i.e. $0 \leq v < (t-1)/2$, the right-hand side is bounded by $sk \cdot (t-1)/2 + wrp/2 < q/2$, implying that the congruence is in fact an equality over $\mathbb{Z}$. In particular, $\langle \boldsymbol{z}, \boldsymbol{e} \rangle$ must be a multiple of sk, and therefore $v = v' \bmod p$. In practice, this relation holds almost all the time even for large values of v, because the scalar product $\langle \boldsymbol{z}, \boldsymbol{e} \rangle$ is usually much smaller than $wr/2$.

The above means that if we can find a short vector $\widetilde{\boldsymbol{u}'}$ in L with its first coefficient equal to 1, we should be able to recover the plaintext up to a possible multiple of p. Moreover, a similar argument shows that *even shorter* vectors in L should be of the form $(0, \boldsymbol{z}', \boldsymbol{0}, \gamma')$ where $\boldsymbol{z}'$ is a very short element in the left-kernel of A and $\gamma' = sk_q^{-1} \cdot p\langle \boldsymbol{z}', \boldsymbol{e} \rangle \bmod^* q$ is a multiple of p.

Description of the attack. Based on the analysis above, we suggest the following heuristic approach to decrypt a given ciphertext $\boldsymbol{c} = (\boldsymbol{a}, x)$: compute the matrix $M(pp, \mathbf{PK}, \boldsymbol{a}, x)$ generating the lattice L as above, and apply the LLL algorithm [LLL82] to obtain a reduced basis $\widetilde{\boldsymbol{u}}_1, \ldots, \widetilde{\boldsymbol{u}}_\ell$. We denote by v_i the last coefficient of $\widetilde{\boldsymbol{u}}_i$ for all i. Then, find the first vector $\widetilde{\boldsymbol{u}}_j$ in that basis whose first coefficient is non zero; it will always be ± 1 so up to a sign change, we can assume that it is 1. Let also g be the gcd of all the v_i's for $i < j$. If $g \geq t$, we have recovered $g = p$ and can therefore return $v_j \bmod g$ as the candidate plaintext. Otherwise, we return v_j directly, since in that case we usually have $g = 0$ and the short basis vectors correspond to short vectors $\boldsymbol{z}$ in the left-kernel of A that are also orthogonal to $\boldsymbol{e}$. This gives the heuristic attack described as Algorithm 1.

Experimental results. We implemented the attack of Algorithm 1 in the computer algebra system SageMath [SM16] using the code provided in Appendix A. The LLL reduction is Sage is carried out using the fplll library [FPL16].

We then ran the attack on a total of 10000 ciphertexts associated with random plaintexts in $\mathbb{Z}_t$, divided into 100 sets of 100 ciphertexts, each set using

Algorithm 1 Decryption attack on Compact-LWE

Input: public parameters $pp = (q, n, m, t, w, b)$, public key $\mathbf{PK} = (A, \boldsymbol{b})$, ciphertext $(\boldsymbol{a}, x)$

Output: candidate plaintext $v \in \mathbb{Z}$

1: set $\kappa = q$.
2: compute the matrix $M = M(pp, \mathbf{PK}, \boldsymbol{a}, x)$.
3: apply the LLL algorithm to obtain a reduced basis $(\widetilde{\boldsymbol{u}}_1, \ldots, \widetilde{\boldsymbol{u}}_\ell)$ of the lattice generated by the rows of M.
4: for all i, denote by u_i (resp. v_i) the first (resp. the last) component of $\widetilde{\boldsymbol{u}}_i$.
5: let j be the smallest index such that $u_j \neq 0$ $\widetilde{\boldsymbol{u}}_j$ is non zero.
6: let $v = v_j/u_j$
7: compute the greatest common divisor g of the last components of $\widetilde{\boldsymbol{u}}_i$ for $1 \leq i < j$.
8: **if** $g \geq t$, reduce v mod g.
9: **return** v.

a distinct randomly generated key pair: this is the experiment provided by the function call `testsubsetsumdecrypt(100,100)` using the code of Appendix A. In our experiment, 9995 ciphertexts out of 10000 (99.95%) were correctly decrypted, and the attack used an average CPU time of 62 milliseconds per ciphertext, on a single core of a 3.4 GHz Core i7-3770 desktop machine.

Thus, our machine can decrypt a bit over 16 ciphertexts per second without the secret key. This is not quite as many as the ≈ 500 decryptions per second claimed by the authors of [LLKN17] on their target platform *with* the secret key, but not too far off...

4 Conclusion

In this paper, we showed that under the parameters suggested in the paper, ciphertexts of the encryption scheme given in [LLKN17] can be decrypted quickly and efficiently in practice, using only information available in the public parameters of the scheme. In particular, the low value of n which was recommended enabled us to use the LLL algorithm to solve the corresponding problem (essentially a low weight vectorial knapsack) efficiently. The 138-bit security estimate for the suggested parameters of Compact-LWE is thus clearly incorrect.

In fact, it is unlikely that the ways in which Compact-LWE differs from standard LWE-based schemes offer any security advantage. For example, it is easy to see that if the SIS problem can be solved in dimension n, one can easily recover the secret Compact-LWE scaling parameter, and reduce the problem to an LWE variant with an irregular sample matrix, which if anything should be *less* secure than standard LWE, as evidenced by the findings of [HM17].

Although more work may be needed for a precise security evaluation of the proposal, one can already confidently say that Compact-LWE does not look like a strong contender in the upcoming NIST competition.

References

ABD16. Martin R. Albrecht, Shi Bai, and Léo Ducas. A subfield lattice attack on overstretched NTRU assumptions - cryptanalysis of some FHE and graded encoding schemes. pages 153–178, 2016.

ADPS17. Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key exchange - a new hope. In Thorsten Holz and Stefan Savage, editors, *USENIX Security 16*, pages 327–343. USENIX Association, August 2017.

BCLvV16. Daniel J. Bernstein, Chitchanok Chuengsatiansup, Tanja Lange, and Christine van Vredendaal. NTRU prime. Cryptology ePrint Archive, Report 2016/461, 2016. http://eprint.iacr.org/2016/461.

BDK+17. Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, and Damien Stehlé. CRYSTALS – kyber: a CCA-secure module-lattice-based KEM. Cryptology ePrint Archive, Report 2017/634, 2017. http://eprint.iacr.org/2017/634.

CKLS16. Jung Hee Cheon, Duhyeong Kim, Joohee Lee, and Yongsoo Song. Lizard: Cut off the tail! Practical post-quantum public-key encryption from LWE and LWR. Cryptology ePrint Archive, Report 2016/1126, 2016. http://eprint.iacr.org/2016/1126.

FPL16. The FPLLL development team. *fplll, a lattice reduction library*, 2016. Available at https://github.com/fplll/fplll.

HM17. Gottfried Herold and Alexander May. LP solutions of vectorial integer subset sums — cryptanalysis of Galbraith's binary matrix LWE. pages 3–15, 2017.

KF17. Paul Kirchner and Pierre-Alain Fouque. Revisiting lattice attacks on overstretched NTRU parameters. pages 3–26, 2017.

Liu17. Dongxi Liu. Compact-LWE for lightweight public key encryption and leveled IoT authentication. In Josef Pierpzyk and Suriadi Suriadi, editors, *ACISP 17, Part I*, volume 10342 of *LNCS*, page xvi. Springer, Heidelberg, July 2017.

LLKN17. Dongxi Liu, Nan Li, Jongkil Kim, and Surya Nepal. Compact-LWE: Enabling practically lightweight public key encryption for leveled IoT device authentication. Cryptology ePrint Archive, Report 2017/685, 2017. http://eprint.iacr.org/2017/685.

LLL82. Arjen K. Lenstra, Hendrik W. Lenstra, and Laszló Lovász. Factoring polynomials with rational coefficients. *Math. Ann.*, 261(4):515–534, 1982.

LPR10. Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. pages 1–23, 2010.

Pei15. Chris Peikert. A decade of lattice cryptography. Cryptology ePrint Archive, Report 2015/939, 2015. http://eprint.iacr.org/2015/939.

Pei16. Chris Peikert. How (not) to instantiate ring-LWE. pages 411–430, 2016.

Reg05. Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. pages 84–93, 2005.

SM16. The Sage Developers. *SageMath, the Sage Mathematics Software System (Version 8.0)*, 2016. http://www.sagemath.org.

Appendix

A Implementation

```
# Compact-LWE parameters
q=2^32
t=2^16
m=74
w=86
n=13
b=16
R=Integers(q)

# ========================

def keygen():
    s=vector(R, [R.random_element() for _ in range(n)])

    r=randint(2,ceil(q/w/t)-1)

    p=0
    while gcd(p,q)>1:
        p=randint(t,ceil(q/r/w)-1)

    sk=0
    while gcd(sk,q)>1 or gcd(sk,p)>1:
        sk=randint(1,ceil((q-w*r*p)/(t-1))-1)

    return s,r,p,sk

def samplegen(s,r,p,sk):
    A=random_matrix(ZZ,m,n,x=0,y=b)
    k=R(p)/R(sk)
    e=vector(R, [randint(0,r-1) for _ in range(m)])
    v=A*s + k*e
    return A, v.change_ring(ZZ), e

def encrypt(A,v,mu):
    a=vector(R,n)
    x=R(mu)
    for _ in range(w):
        j=randint(0,m-1)
        a+=A[j]
        x-=v[j]

    return a.change_ring(ZZ), x.lift()

# ========================
```

```
def subsetsumdecrypt(A,v,a,x):
    kappa=q

    L=block_matrix(ZZ, \
        [[1, 0, kappa*a.row(), x], \
         [0, t*identity_matrix(m), -kappa*A, v.column()], \
         [0, 0, 0, q]])
    L=L.LLL()

    #index of first non-zero entry in the first column of L
    idx=next((i for i,x in enumerate(L.column(0).list()) if x!=0))
    g=gcd(L[:idx,-1].list())

    cand=L[idx,-1]/L[idx,0]
    if g>t:
        cand=cand%g

    return L, cands

def testsubsetsumdecrypt(trials=100,pairs=1):
    succ=0
    tottime=0.0

    for npair in range(pairs):
        s,r,p,sk=keygen()
        A,v,e=samplegen(s,r,p,sk)

        succnow=0
        for _ in range(trials):
            mu=randint(1,t-1)
            a,x=encrypt(A,v,mu)

            tm=cputime(subprocesses=True)
            mucand=subsetsumdecrypt(A,v,a,x)[1]
            tottime+=float(cputime(tm))
            if mu==mucand:
                succnow+=1
        succ+=succnow
        print "Key pair %d complete. Success rate: %d/%d." % \
                (npair,succnow,trials)

    print "Successful recoveries: %d/%d (%f)." % \
            (succ,trials*pairs,RR(100*succ/trials/pairs))
    print "Average time: %f seconds." % (tottime/trials/pairs)
```