ERROR REPORT

LWE Implementations

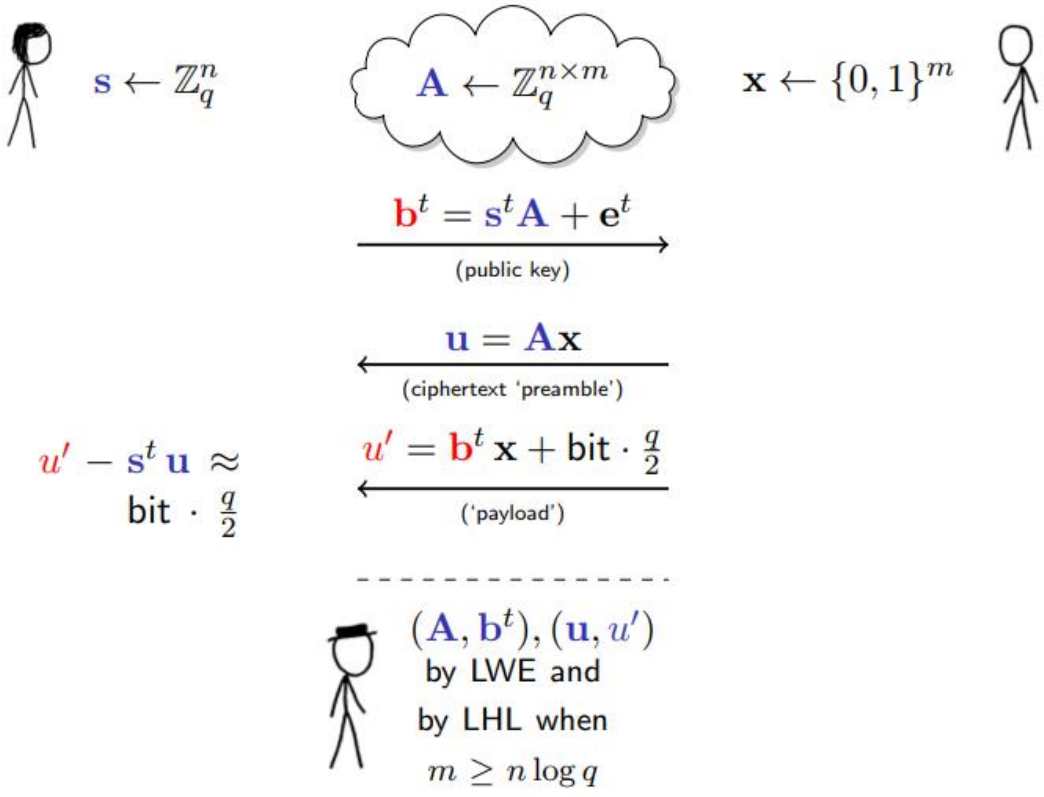
Abstract

Includes the issues found in Public Key Cryptosystem and Dual Cryptosystem implementations.

Contents

1.0 Public Key Cryptosystem

1.1 Notation and Operation


$\mathbf{s} \leftarrow \mathbb{Z}_q^n$
$\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}$
$\mathbf{x} \leftarrow \{0,1\}^m$
$\mathbf{b}^t = \mathbf{s}^t\mathbf{A} + \mathbf{e}^t$
(public key)
$\mathbf{u} = \mathbf{A}\mathbf{x}$
(ciphertext 'preamble')
$u' - \mathbf{s}^t\mathbf{u} \approx$ bit $\cdot \frac{q}{2}$
$u' = \mathbf{b}^t\mathbf{x} +$ bit $\cdot \frac{q}{2}$
('payload')
$(\mathbf{A}, \mathbf{b}^t), (\mathbf{u}, u')$
by LWE and
by LHL when
$m \geq n \log q$

Source: Lattice-Based Crypto & Applications, Bar-Ilan University, Israel 2012

1.2 Issue

For the public-key cryptosystem when using q < 2^{16} (65 536), when encrypting and decrypting bit 0, the accuracy is around 90% to 100%. For bit 1, 100% accuracy is achieved.

1.3 Observations for bit 0

For q = 1024 out of 1000 tests,

Encryption and Decryption work 95.6% of the time.

Max eTx = 187

Max bTxMAX = 1023

Error instances = 44

1.3.1 Scenario 1

When “bTx” is close to “q” and “sTu mod q” is close to 0, the results are wrong.

Ex:-

1.

```
[DEBUG] bTx:  1020

[DEBUG] sTu before mod:  920789003

[DEBUG] sTu mod q:  11

[DEBUG] u_:  1020

[DEBUG] eTx:  -15

[DEBUG] recovered:  1

[DEBUG] u_ - sTu:  1009
```

2.

```
[DEBUG] bTx:  984

[DEBUG] sTu before mod:  1088840713

[DEBUG] sTu mod q:  9

[DEBUG] u_:  984

[DEBUG] eTx:  -49

[DEBUG] recovered:  1

[DEBUG] u_ - stu:  975
```

When encrypting: u’ = bTx + bit · q/2

When bit = 0, u’ = bTx

When decrypting u’ = bTx,

bit · q/2 + eTx = u’ – sTu = bTx – sTu

When “bTx” is close to “q” and “sTu mod q” is close to 0, the above value can be greater than q/2. Therefore it is decrypted to bit 1.

1.3.2 Scenario 2

When “bTx” is close to 0 and “sTu mod q” is close to “q”, the results are wrong.

1.

```
[DEBUG] bTx:  16

[DEBUG] sTu before mod:  1199177704

[DEBUG] sTu mod q:  1000

[DEBUG] u_:  16

[DEBUG] eTx:  40

[DEBUG] recovered:  1

[DEBUG] u_ - stu:  -984
```

2.

```
[DEBUG] bTx:  4

[DEBUG] sTu before mod:  1333685190

[DEBUG] sTu mod q:  966

[DEBUG] u_:  4

[DEBUG] eTx:  62

[DEBUG] recovered:  1

[DEBUG] u_ - stu:  -962
```

When encrypting: $u' = bTx + bit \cdot q/2$

When bit = 0, $u' = bTx$

When decrypting $u' = bTx$,

$bit \cdot q/2 + eTx = u' – sTu = bTx – sTu$

When “bTx” is close to 0 and “sTu mod q” is close to “q”, the above value can be smaller than -q/2. Therefore it is decrypted to bit 1 because the absolute value of “u′ – sTu” > bit · q/2.

Because of these 2 reasons encryption and decryption of bit 0 give inaccurate results.

1.4 Example

1.4.1 Key Genaration

q = 15, n = 2, m = 3, e_min = -1, e_max = 1

$eT = \begin{bmatrix} 0 & 1 & -1 \end{bmatrix}$ $\quad bT(before\ mod) = \begin{bmatrix} 87 & 82 & 224 \end{bmatrix}$

Public key

$A = \begin{bmatrix} 5 & 9 & 11 \\ 3 & 0 & 9 \end{bmatrix}$ $\quad bT = \begin{bmatrix} 12 & 7 & 14 \end{bmatrix}$

Private key

$A = \begin{bmatrix} 5 & 9 & 11 \\ 3 & 0 & 9 \end{bmatrix}$ $\quad sT = \begin{bmatrix} 9 & 14 \end{bmatrix}$

1.4.2 Encryption (messege bit = 0)

$x = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$ $\quad u\ (before\ mod) = \begin{bmatrix} 11 \\ 9 \end{bmatrix}$ $\quad u = \begin{bmatrix} 11 \\ 9 \end{bmatrix}$

$u\_(before\ mod) = 224$ $\quad u\_ = 14$

1.4.3 Decryption

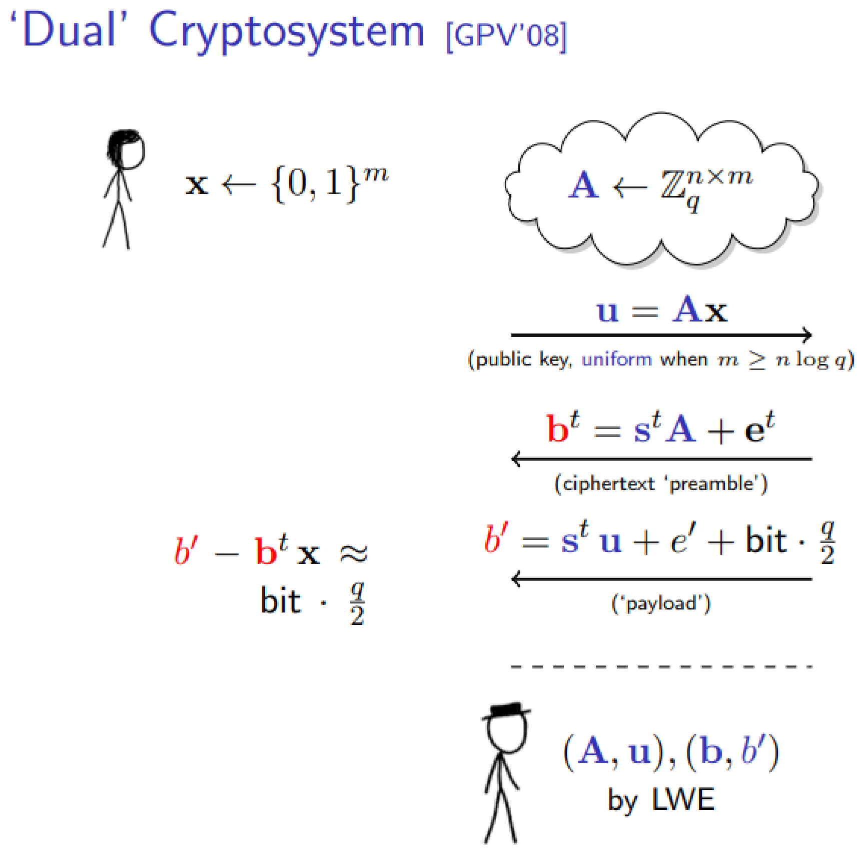
$sTu(before\ mod) = 225.$ $\quad sTu = 0$

Bit => u_ - sTu = 14-0 = 14

Closer to 15/2, so the recoverd bit is 1. But the bit we encrypt is 0

2.0 Dual Cryptosystem

2.1 Notation and Operation



‘Dual’ Cryptosystem [GPV’08]

$\mathbf{x} \leftarrow \{0,1\}^m$

$\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}$

$\mathbf{s} \leftarrow \mathbb{Z}_q^n$

$\mathbf{u} = \mathbf{A}\mathbf{x}$

(public key, uniform when $m \geq n \log q$)

$\mathbf{b}^t = \mathbf{s}^t\mathbf{A} + \mathbf{e}^t$

(ciphertext ‘preamble’)

$b' = \mathbf{s}^t\mathbf{u} + e' + \text{bit} \cdot \frac{q}{2}$

(‘payload’)

$b' - \mathbf{b}^t\mathbf{x} \approx \text{bit} \cdot \frac{q}{2}$

$(\mathbf{A}, \mathbf{u}), (\mathbf{b}, b')$

by LWE

Source: Lattice-Based Crypto & Applications, Bar-Ilan University, Israel 2012

2.2 Issue

For the public-key cryptosystem when using q < 150 000, when encrypting and decrypting bit 0, the accuracy is around 90% to 100%. For bit 1, 100% accuracy is achieved.

2.3 Observations for bit 0

For q = 1024 out of 1000 tests,

Encryption and Decryption work 96.2% of the time.

2.3.1 Scenario 1

When “ b’ ” is close to “q” and “bTx mod q” is close to 0, the results are wrong.

Ex:-

1.

```
[DEBUG] b' = sTu + e':  1013

[DEBUG] bTx before mod:  63496

[DEBUG] bTx mod q:  8

[DEBUG] b'-bTx:  1005

[DEBUG] eTx:  -19
```

2.

```
[DEBUG] b' = sTu + e':  974

[DEBUG] bTx before mod:  66590

[DEBUG] bTx mod q:  30

[DEBUG] b'-bTx:  944

[DEBUG] eTx:  -80
```

When encrypting: b’ = sTu + e' + bit · q/2

When bit = 0, b’ = sTu + e'

When decrypting b’ = sTu + e',

bit · q/2 ≈ b’ – bTx

When “ b’ ” is close to “q” and “bTx mod q” is close to 0, the above value can be greater than q/2. Therefore it is decrypted to bit 1.

2.3.2 Scenario 2

When “bTx” is close to 0 and “sTu mod q” is close to “q”, the results are wrong.

Ex:-

1.

```
[DEBUG] b' = sTu + e':  6
[DEBUG] bTx before mod:  60415
[DEBUG] bTx mod q:  1023
[DEBUG] b'-bTx:  -1017
[DEBUG] eTx:  7
iteration = 798
```

2.

```
[DEBUG] b' = sTu + e':  7
[DEBUG] bTx before mod:  64500
[DEBUG] bTx mod q:  1012
[DEBUG] b'-bTx:  -1005
[DEBUG] eTx:  19
iteration = 854
```

When encrypting: b’ = sTu + e' + bit · q/2

When bit = 0, b’ = sTu + e'

When decrypting b’ = sTu + e',

bit · q/2 ≈ b’ – bTx

When “ b’ ” is close to 0 and “bTx mod q” is close to “q”, the above value can be lesser than -q/2. Therefore it is decrypted to bit 1 because the absolute value of “b’ – bTx” > bit · q/2.

Because of these 2 reasons encryption and decryption of bit 0 give inaccurate results.