# Designs for practical SHE schemes based on RLWR

M. Bolboceanu, A. Costache, E. Hales, R. Player, M. Rosca, R. Titiu
(with thanks to A. Costache and R. Player for their slides!)

# What is RLWR?

# To start: what is RLWE?

Let N be a power of 2, and q a modulus.

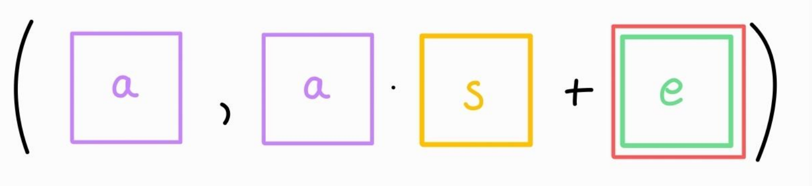
Consider also the ring $R_q = \mathbb{Z}_q[X]/(X^N+1)$

$$\left( \boxed{a} , \boxed{a} \cdot \boxed{s} + \boxed{e} \right) \in R_q \times R_q$$

Decision problem: is (a,b) formed like this, or uniformly random?
Search problem: what is the secret s?

# How is RLWR different?

- No Gaussian errors
- Achieve this through rounding operation on a.s.



# What is RLWR?

Rounding operation

$$\lfloor x \rceil_{q,p} := \left\lfloor \frac{p}{q} x \right\rceil$$

Let N be a power of 2, and q a modulus.

Consider also the ring $R_q = \mathbb{Z}_q[X]/(X^N + 1)$

$$\left( \boxed{a} \, , \left\lfloor \boxed{a} \cdot \boxed{s} \right\rceil_{q,p} \right) \in R_q \times R_p$$

Decision problem: is (a,b) formed like this, or uniformly random?

Search problem: what is the secret s?

# Motivation for considering RLWR

- Avoids need for Gaussian sampling
  - Side channels
  - Cost
- Easy to implement rounding
- Potential for improved bandwidth
- LWR is used by several NIST candidates (Lizard, Round5, Saber)

# How might we make SHE schemes from LWR?

# Initial attempt: BFV analogue [CS17]

Method: Swap out everything from BFV for rounded versions

uniform random bits

$$(\mathsf{ct}_0, \mathsf{ct}_1) = \left( \sum_{k=1}^{\ell} \boxed{r_k} \cdot \underline{v_k}, \boxed{\Delta_p} \cdot \underline{m} + \sum_{k=1}^{\ell} \boxed{r_k} \cdot \underline{w_k} \right) \in R_q \times R_p$$

Public key Pairs:

$(v_k, w_k) \in R_q \times R_p$

$=$

$(v_k, \lceil v_k s \rfloor_{q,p})$

scale parameter

$\Delta_p = \lfloor \frac{p}{t} \rceil$

message

$m \in R_t$

Costache, A., & Smart, N. P. (2017). Homomorphic encryption without gaussian noise. *Cryptology ePrint Archive*. https://eprint.iacr.org/2017/163.pdf

# There’s a problem…

When we multiply two ciphertexts,ct=($ct_0$,$ct_1$) and ct’=($ct_0$’,$ct_1$’), we do a tensor product, and it is not clear what ring each component should be in

$\boxed{R_q}$

$\boxed{R_p}$

$$\left(\boxed{\mathtt{ct}_0}\boxed{\mathtt{ct}_0'}, \boxed{\mathtt{ct}_0}\boxed{\mathtt{ct}_1'} + \boxed{\mathtt{ct}_0'}\boxed{\mathtt{ct}_1}, \boxed{\mathtt{ct}_1}\boxed{\mathtt{ct}_1'}\right) = (c_2, c_1, c_0)$$

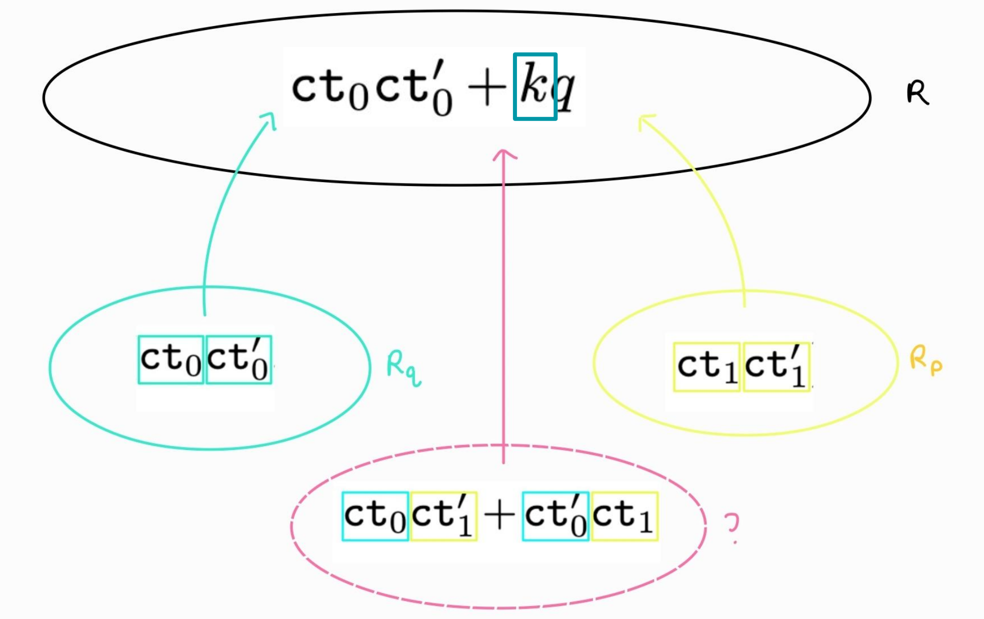
# When we do noise analysis…

We normally lift to R

Luo et al suggest we consider $c_2$ in the ring $R_q$

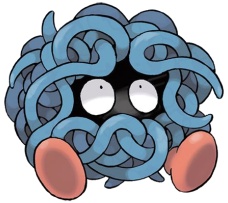
Creates unmanageable noise growth, due to integer polynomial $\boxed{k}$, if we consider in the wrong modulus

When we're done with analysis, where do we map it to?




Luo, F., Wang, F., Wang, K., Li, J., & Chen, K. (2018). LWR-based fully homomorphic encryption, revisited. *Security and Communication Networks*, *2018*.

# The Tangled Modulus problem



# Where should the elements live after multiplication?

In which ring should we consider each element, in order to avoid unmanageable noise?

$$\left(\boxed{\mathtt{ct}_0}\boxed{\mathtt{ct}'_0}, \boxed{\mathtt{ct}_0}\boxed{\mathtt{ct}'_1} + \boxed{\mathtt{ct}'_0}\boxed{\mathtt{ct}_1}, \boxed{\mathtt{ct}_1}\boxed{\mathtt{ct}'_1}\right) = (c_2, c_1, c_0)$$

We define the invariant noise as $N_{\text{mult}}$ in the following equation:

$$\frac{t}{p}\left[\left(\frac{p}{q}\right)^2 c_2 s^2 - \frac{p}{q} c_1 s + c_0\right] = m_{\text{mult}} + N_{\text{mult}} + tG_{\text{mult}}$$

# Possible solutions

$\boxed{\text{Solution 1: Carefully choose modulus and do noise analysis to ensure manageable noise (our work, LPR-style)}}$

Solution 2: Modify the evaluation key to avoid the problem (our work, Regev-style)

Solution 3: Base the RLWR scheme on a different scheme to BFV (LWWC18)

# RLWE LPR variant of BFV interlude

# LPR-style encryption scheme

Key Generation:

$$\mathtt{sk} = s \in R_q$$
$$\mathtt{pk} = (a, as + e) \in R_q \times R_q$$

Encryption:

$$\mathtt{ct} = (c_0, c_1) \in R_q \times R_q, \text{ where}$$

$$c_0 = \mathtt{pk}_0 u + e_0$$
$$c_1 = \mathtt{pk}_1 u + e_1 + \lfloor q/t \rfloor \cdot \mathtt{m} \qquad \mathtt{m} \in R_t$$

Decryption of ct:

$$\mathtt{m}' = \left[ \left\lfloor \frac{t}{q} \left[ (-c_0 s) + c_1 \right]_q \right\rceil \right]_t$$



Lyubashevsky, V., Peikert, C., & Regev, O. (2013). On ideal lattices and learning with errors over rings. *Journal of the ACM (JACM)*, *60*(6), 1-35.

# LPR-style encryption: multiplication

Multiplication of ct=($ct_0$,$ct_1$) and ct'=($ct_0$',$ct_1$'):

$$\mathtt{ct}_{\mathsf{mult}} := \left( \left\lfloor \frac{t}{q} c_0 c_0' \right\rceil,\ \left\lfloor \frac{t}{q} (c_0 c_1' + c_1 c_0') \right\rceil,\ \left\lfloor \frac{t}{q} c_1 c_1' \right\rceil \right) = (d_0, d_1, d_2)$$



Lyubashevsky, V., Peikert, C., & Regev, O. (2013). On ideal lattices and learning with errors over rings. *Journal of the ACM (JACM)*, *60*(6), 1-35.

# LPR-style RLWR scheme

# Our LPR-style SHE scheme from RLWR

Key Generation:

$$\mathtt{sk} = s \in R_r$$
$$\mathtt{pk} = (a, b = \lfloor a \cdot s \rceil_{r,q}) \in R_r \times R_q$$

Encryption: $\mathtt{ct} = (c_0, c_1) \in R_q \times R_p$, where

$$c_0 = \lfloor \mathrm{pk}_0 u \rceil_{r,q}$$
$$c_1 = \lfloor \mathrm{pk}_1 u \rceil_{q,p} + \lfloor p/t \rfloor \cdot \mathtt{m}$$

Decryption:

$$\mathtt{m}' = \left[ \left\lfloor \frac{t}{p} \left( -\frac{p}{q} c_0 s + c_1 \right) \right\rceil \right]_t$$

# Our LPR scheme mult from RLWR

Here's where the elements should live

This combination avoids the unmanageable noise growth

$$\mathsf{ct}_{\mathsf{mult}} := \left( \left[ \left\lfloor \frac{t}{p} c_0 c_0' \right\rceil \right]_{q^2/p}, \left[ \left\lfloor \frac{t}{p} (c_0 c_1' + c_1 c_0') \right\rceil \right]_q, \left[ \left\lfloor \frac{t}{p} c_1 c_1' \right\rceil \right]_p \right)$$

**LPR-style encryption: multiplication**

Multiplication of ct=($c_0$,$c_1$) and ct'=($c_0$',$c_1$'):

$$\mathsf{ct}_{\mathsf{mult}} := \left( \left\lfloor \frac{t}{q} c_0 c_0' \right\rceil, \left\lfloor \frac{t}{q} (c_0 c_1' + c_1 c_0') \right\rceil, \left\lfloor \frac{t}{q} c_1 c_1' \right\rceil \right) = (d_0, d_1, d_2)$$

# Security proof?

# Security proof for regular LPR (from RLWE)

Hyb1 Honestly generated $\mathtt{pk}$ and $\mathtt{ct}$ (Real IND-CPA game)



Lyubashevsky, V., Peikert, C., & Regev, O. (2013). On ideal lattices and learning with errors over rings. *Journal of the ACM (JACM)*, *60*(6), 1-35.

# Security proof for regular LPR (from RLWE)

Hyb1 Honestly generated $\mathtt{pk}$ and $\mathtt{ct}$ (Real IND-CPA game)

$\downarrow$ Decision Ring-LWE

Hyb2 Uniform $\mathtt{pk} = (a, b)$, honestly generated $\mathtt{ct} = (au + e_1, bu + e_2)$



Lyubashevsky, V., Peikert, C., & Regev, O. (2013). On ideal lattices and learning with errors over rings. *Journal of the ACM (JACM)*, *60*(6), 1-35.

# Security proof for regular LPR (from RLWE)

**Hyb1** Honestly generated $\texttt{pk}$ and $\texttt{ct}$ (Real IND-CPA game)

$\downarrow$ Decision Ring-LWE

**Hyb2** Uniform $\texttt{pk} = (a, b)$, honestly generated $\texttt{ct} = (au + e_1, bu + e_2)$

$\downarrow$ Decision Ring-LWE

**Hyb3** Uniform $\texttt{pk}$, uniform $\texttt{ct}$ ($\mathcal{A}$ has no information)



Lyubashevsky, V., Peikert, C., & Regev, O. (2013). On ideal lattices and learning with errors over rings. *Journal of the ACM (JACM)*, *60*(6), 1-35.

# Our RLWR LPR-style security proof

Hyb1 Honestly generated $\mathtt{pk} \in R_r \times R_q$, $\mathtt{ct} \in R_q \times R_p$ (IND-CPA)

# Our RLWR LPR-style security proof

Hyb1 Honestly generated $\mathtt{pk} \in R_r \times R_q$, $\mathtt{ct} \in R_q \times R_p$ (IND-CPA)

$\downarrow$ Decision Ring-LWR

Hyb2 Uniform $\mathtt{pk} = (a, b)$, honestly generated $\mathtt{ct} = (\lfloor au \rceil_{r,q}, \lfloor bu \rceil_{q,p})$

# Our RLWR LPR-style security proof

Hyb1 Honestly generated $\texttt{pk} \in R_r \times R_q$, $\texttt{ct} \in R_q \times R_p$ (IND-CPA)

$\downarrow$ Decision Ring-LWR

Hyb2 Uniform $\texttt{pk} = (a, b)$, honestly generated $\texttt{ct} = (\lfloor au \rceil_{r,q}, \lfloor bu \rceil_{q,p})$

$\downarrow$ 3-moduli Ring-LWR

Hyb3 Uniform $\texttt{pk}$, uniform $\texttt{ct}$ ($\mathcal{A}$ has no information)

# 3-moduli RLWR

Let $a \leftarrow R_r$ and $b \leftarrow R_q$ be chosen uniformly, and let $u$ be a uniformly chosen polynomial with ternary coefficients. The (decisional) 3-moduli Ring-LWR problem with parameters $n$, $r$, $q$, $p$ asks to distinguish between $(a, b, \lfloor au \rceil_{r,q}, \lfloor bu \rceil_{q,p})$ and uniform tuples $(a, b, c, d) \in R_r \times R_q \times R_q \times R_p$.

# Reduction from Ring-LWR to 3-moduli Ring-LWR

Let $p$, $q$, $r$ be integers such that $q|r$ and $pr = q^2$. If there is an efficient algorithm for 3-moduli Ring-LWR with parameters $n$, $r$, $q$, $p$, then there is an efficient algorithm for Ring-LWR with parameters $n$, $r$, $q$.

# Regev-style scheme

# Can we reduce restrictions on p and q?

LPR-style scheme works well for powers of 2 → natural for rounding

LPR-style scheme requires: $q|r$ and $pr = q^2$

We might want to explore alternative schemes to avoid this restriction

Our solution:

- Regev-style scheme
  - Generalise the relinearisation technique to remove the condition $q|r$
  - Provable security for prime p and q with no other restrictions

# Results

# Comparing to BFV asymptotically

| | LPR-style scheme | Regev-style scheme | Prior work [LWWC18] | BFV |
|---|---|---|---|---|
| $\texttt{pk}$ size | $n \log(rq)$ | $\ell n \log(pq)$ | $(\ell + 1) n \log(p)$ | $2n \log(q)$ |
| $\texttt{ct}$ size | $n \log(pq)$ | $n \log(pq)$ | $(\ell + 1) n \log(p)$ | $2n \log(q)$ |
| Security | RLWR | RLWR | RLWR | RLWE |

# Methodology for concrete comparison with BFV

Goal: compare ciphertext sizes between our schemes and BFV, using our proof-of-concept Python implementation

Find parameter set with minimal ciphertext size such that we have:

1) 128 bits security
2) Correct decryption

Tree-shaped arithmetic circuit with depth L

Each level is 8 additions and 1 multiplication

# What is the ciphertext size?

Logarithm to base 2 of the minimal ciphertext size in kilobytes that supports L levels for the specified circuit with plaintext modulus t = 3

Calculated using proof of concept Python implementation

| Scheme | Level $L$ | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 | 18 | 20 | 22 | 24 | 26 | 28 | 30 |
| BFV | 4.17 | 6.21 | 7.86 | 8.27 | 8.58 | 9.92 | 10.14 | 10.34 | 10.50 | 10.65 | 11.86 | 11.99 | 12.10 | 12.21 | 12.31 |
| **LPR-like** | 3.91 | 6.09 | 7.79 | 8.22 | 8.55 | 9.89 | 10.11 | 10.31 | 10.48 | 10.64 | 11.85 | 11.97 | 12.09 | 12.19 | 12.30 |
| **Regev($\ell = 3$)** | 4.09 | 6.17 | 7.83 | 8.25 | 8.57 | 9.91 | 10.13 | 10.33 | 10.50 | 10.65 | 11.85 | 11.98 | 12.10 | 12.20 | 12.30 |

# What is the ciphertext size?

Logarithm to base 2 of the minimal ciphertext size in kilobytes that supports L levels for the specified circuit with plaintext modulus $t = 2^8$

Calculated using proof of concept Python implementation

| Scheme | Level $L$ | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 | 18 | 20 | 22 | 24 | 26 | 28 | 30 |
| BFV | 4.58 | 6.63 | 8.27 | 8.68 | 10.06 | 10.32 | 10.54 | 10.73 | 11.95 | 12.11 | 12.24 | 12.37 | 12.48 | 12.59 | 12.69 |
| **LPR-like** | 4.43 | 6.55 | 8.22 | 8.64 | 10.03 | 10.29 | 10.52 | 10.77 | 11.94 | 12.09 | 12.23 | 12.36 | 12.47 | 12.58 | 12.68 |
| **Regev($\ell = 3$)** | 4.52 | 6.60 | 8.25 | 8.67 | 10.04 | 10.31 | 10.53 | 10.72 | 11.95 | 12.10 | 12.24 | 12.36 | 12.48 | 12.59 | 12.69 |

# Parameter selection methodology

- We would like to choose parameters for concrete security using Lattice Estimator
- But Lattice Estimator is built for LWE instances
- Solution: interpret our LWR instances as LWE instances
- Model the implied LWE error distribution as a Gaussian with standard deviation $\sigma = \sqrt{\left((q/p)^2 - 1\right)/12}$ following 'Estimate all the schemes'
- Take p and q powers of 2 for performance
- We choose q/p = 16 to make $\sigma$ = 4.61, which is close to the standard choice for FHE of $\sigma$ = 3.2

Albrecht, M. R., Curtis, B. R., Deo, A., Davidson, A., Player, R., Postlethwaite, E. W., ... & Wunderer, T. (2018). Estimate all the {LWE, NTRU} schemes!. In *Security and Cryptography for Networks: 11th International Conference, SCN 2018*

# Summary

# What did we do?

Designed two SHE schemes based on the RLWR problem, Regev-style and LPR-style

- BFV-like scheme is possible from RLWR
- Comparable parameters to BFV
- Improved ciphertext sizes
- Security analysis

# Next steps…

- Library integration?
- RNS variant?
- Building other things from RLWR?
- [your cool idea here!]

# Thank you!

erin.hales.2018@live.rhul.ac.uk
@erin__hales
Paper coming soon to an eprint near you…

Designed two SHE schemes based on the RLWR problem, Regev-style and LPR-style

- BFV-like scheme is possible from RLWR
- Comparable parameters to BFV
- Improved ciphertext sizes
- Thorough security analysis

# Parameter sizes for LPR-style RLWR scheme

Assuming uniform ternary secret, targeting 128 bits security

| $n$ | $r$ | $q$ | $p$ |
|---|---|---|---|
| $2^{15}$ | $2^{856}$ | $2^{852}$ | $2^{848}$ |
| $2^{14}$ | $2^{425}$ | $2^{421}$ | $2^{417}$ |
| $2^{13}$ | $2^{211}$ | $2^{207}$ | $2^{203}$ |
| $2^{12}$ | $2^{105}$ | $2^{101}$ | $2^{97}$ |
| $2^{11}$ | $2^{52}$ | $2^{48}$ | $2^{44}$ |
| $2^{10}$ | $2^{26}$ | $2^{22}$ | $2^{18}$ |

# Comparing to BFV asymptotically

| | Regev-type scheme | LPR-type scheme | LWWC [54] | BFV [37] |
|---|---|---|---|---|
| Size of pk | $\ell n \log(PQ)$ | $n \log(rq)$ | $(\ell' + 1)n \log(q')$ | $2n \log(q'')$ |
| Size of ct | $n \log(PQ)$ | $n \log(pq)$ | $(\ell' + 1)n \log(p')$ | $2n \log(q'')$ |
| Security | Ring-LWR$_{n,Q,P}$ | Ring-LWR$_{n,r,q}$ | Ring-LWR$_{n,q',p'}$ | Ring-LWE$_{n,\chi,q''}$ |
| Constraints | $P$ and $Q$ prime | $q \mid r$ and $pr = q^2$ | $p' \mid q'$ | N/A |

**Table 1.** A comparison of our Regev-type scheme and our LPR-type scheme with the prior schemes LWWC [54] and BFV [37]. The parameter constraints specified are required for provable security. We may assume $\log(q'') = \log(Q) = \log(r)$, while $q' > p'$ and $p'$ is a polynomial factor larger than $q''$. If the encryption randomness in the Regev-type scheme is sampled from the set of scalars $\{-B/2, \ldots B/2\}$ then $\ell \geq 1/\log(B+1)(n\log(PQ) + 2\lambda - 2)$, while $\ell' > \log(q') + 2\lambda$.