

Holocron

Threshold K-PKE Encryption & Decryption from FHE

Yongsoo Song (Seoul National University)

Presented at NIST TCPT3

30th Sep 2026

K-PKE (Crystals-Kyber)

- **Post-Quantum Public Key Encryption** underlying the NIST-standardized **ML-KEM** (FIPS-203)
- Based on the **Module Learning With Errors** (MLWE) assumption
- **IND-CPA (K-PKE)** and -CCA variants



Holocron Team

- 12 members @ Seoul, Korea
- Samsung SDS
  - Kyoohyung Han, Byeonghak Lee, Jihoon Kwon, Sangyub Lee, Hyojin Yoon, Jihoon Cho
- Seoul National University
  - Intak Hwang, Jaejun Ko, Seonhong Min, Yongsoo Song
- Hansung University
  - Siwoo Eum, Hwajeong Seo



NIST Category N2 Specification

- Thresholdization of K-PKE.Encrypt
  - **Encryption** algorithm of K-PKE
  - Receives **secret-shared** input message
  - Outputs a **standard** K-PKE ciphertext

- Thresholdization of K-PKE.Decrypt
  - **Decryption** algorithm of K-PKE
  - Receives a **standard** K-PKE ciphertext
  - Outputs the message



NIST Category N2 Specification

- Provable security against **active (malicious)** adversaries

- Optional Assumptions:
  - Secure against adaptive/mobile corruptions
  - **Security-with-abort**
  - **Trusted Setup**



Threshold FHE

- Fully Homomorphic Encryption (**FHE**)
  - **Arbitrary circuit evaluation** on **encrypted data**.
  - Conventionally for **two-party computation** (2PC)

- (t-out-of-n) **Threshold FHE** (ThFHE)
  - FHE secret is **distributed** to n parties.
  - Can be decrypted only in the presence of $\geq t$ parties.

- Universal Thresholdizer (UT) via ThFHE [BGG+18]
  - **Equivalent functionality** from FHE
  - Preserving the K-PKE ciphertext format



Notations

- $R = \mathbb{Z}[X]/(X^d + 1)$
- $R_q = R/qR$
- $\mathcal{U}(\mathcal{S})$: Uniform distribution over $\mathcal{S}$.
- β_η : A distribution over R where each coefficient is drawn from $\sum_1^\eta \mathcal{U}(\mathbb{B}) - \sum_1^\eta \mathcal{U}(\mathbb{B})$



Setup

- **K-PKE** setup
  - Let $\mathsf{dk}, \mathsf{ek} \leftarrow \mathsf{Kyber.KeyGen}(1^\lambda)$.
  - Output $\mathsf{ek} \in R_q^{k \times (k+1)}$.
- **FHE** setup
  - Let $(\mathsf{sk}_{\mathrm{enc}}, \mathsf{pk}_{\mathrm{enc}}) \leftarrow \mathsf{CKKS.KeyGen}(1^\lambda)$.
  - Let $(\mathsf{sk}_{\mathrm{dec}}, \mathsf{pk}_{\mathrm{dec}}) \leftarrow \mathsf{BGV.KeyGen}(1^\lambda)$.
  - Switching key: $\mathsf{dk}_{\mathrm{FHE}} \leftarrow \mathsf{BGV.Enc}(\mathsf{sk}_{\mathrm{dec}}; \mathsf{dk})$
  - Secret-share $\mathsf{sk}_{\mathrm{enc}}$ and $\mathsf{sk}_{\mathrm{dec}}$ in the t-out-of-n manner.
  - Publish $(\mathsf{pk}_{\mathrm{enc}}, \mathsf{pk}_{\mathrm{dec}})$



Setup

The setup can be done by...

- Assuming a **trusted key dealer**,
- Or, by running distributed key generation (DKG)
  - DKG for BGV/CKKS: [RST+22], [MHP+26]
  - A similar approach can be applied to K-PKE as well.



Encryption Protocol

- Plain Algorithm
  1) Sample $\mathbf{r} \leftarrow \beta_\eta^k$ and $\mathbf{e} \leftarrow \beta_\eta^{k+1}$.
  2) Compute $(v, \mathbf{u}) = \mathbf{r}^\top \cdot \mathsf{ek} + \mathbf{e} + (\lfloor q/2 \rceil \cdot m, \mathbf{0}) \pmod q$.
  3) Output $\mathsf{ct} = (\lfloor 2^{d_v}/q \cdot v \rceil, \lfloor 2^{d_u}/q \cdot \mathbf{u} \rceil)$.
- Naïve UT computes **everything** homomorphically.
- In this case,
  - Sampling from β_η, and modulus switching.
  - Extremely **expensive** to evaluate inside FHE.



Encryption Protocol

- Instead, delegate FHE-unfriendly operations to users
  - **Sampling** and **modulus switching**
- Our idea:
  - Each party P_i, $i \in \mathcal{I}$. **samples** a randomness ω_i
  - From $\{\omega_i\}_{i\in\mathcal{I}}$, **homomorphically generate** $\{(\mathbf{r}_i, \mathbf{e}_i)\}_{i\in\mathcal{I}}$ s.t.

$$\mathbf{r} = \sum_{i\in\mathcal{I}} \mathbf{r}_i \quad \text{and} \quad \mathbf{e} = \sum_{i\in\mathcal{I}} \mathbf{e}_i.$$

  - Each i-th party outputs $\mathsf{ct}_i = \mathbf{r}_i^\top \cdot \mathsf{ek} + \mathbf{e}_i + (\lfloor q/2 \rceil \cdot m_i, \mathbf{0})$
  - **Aggregate** $(v, \mathbf{u}) = \sum_{i\in\mathcal{I}} \mathsf{ct}_i = \mathbf{r}^\top \cdot \mathsf{ek} + \mathbf{e} + (\lfloor q/2 \rceil \cdot m, \mathbf{0})$.
  - **Publicly** compute $\mathsf{ct} = (\lfloor 2^{d_v}/q \cdot v \rceil, \lfloor 2^{d_u}/q \cdot \mathbf{u} \rceil)$.

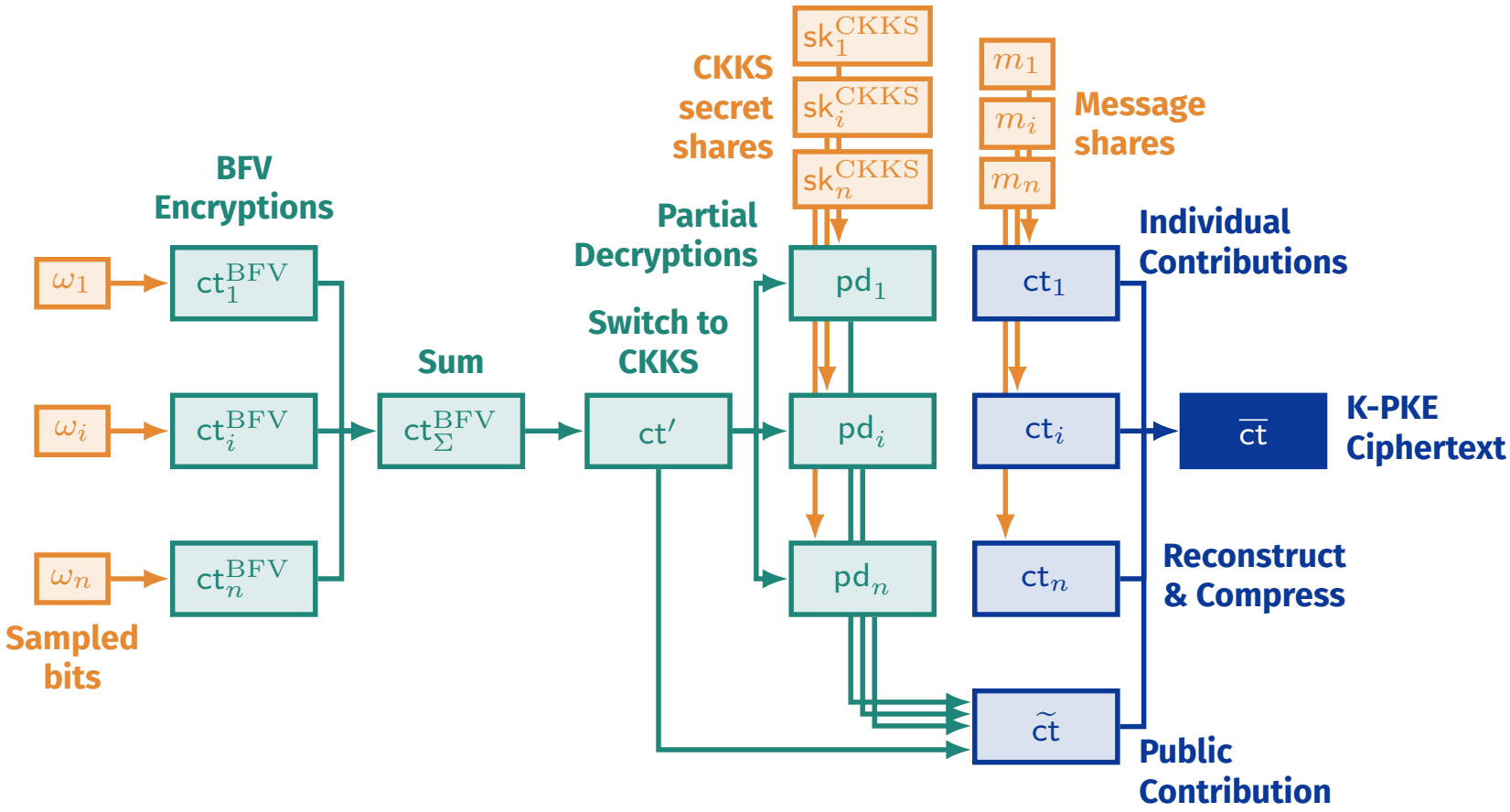


Encryption Protocol (Semi-honest)

1) P_i broadcasts a BFV encryption of $\omega_i \leftarrow \mathbb{Z}_2^{(4k+2)d\eta}$
2) Aggregate to compute $\omega = \sum_{1 \le i \le n} \omega_i \in \mathbb{Z}_2^{(4k+2)d\eta}$
3) Scheme switching to discrete CKKS [BKSS24]
   - Obtain an encryption of $[\omega]_{\mathbb{Z}} \in \{0,1\}^{(4k+2)d\eta}$
4) Distributed decryption
   - Each P_i computes a partial decryption masked by a large smudging noise $f_i \in \mathbb{Z}^{(4k+2)d\eta}$. This forms additive shares of $[\omega]_{\mathbb{Z}}$: a public f_0 and $\{f_i\}_{i \in \mathcal{I}}$
   - P_i outputs K-PKE enc. of m_i with randomness f_i
5) Aggregate K-PKE Ciphertexts
   - Publicly compute K-PKE enc. using the public f_0
   - Aggregate all to output the K-PKE enc. of $m = \sum_{1 \le i \le n} m_i$



Encryption Protocol



Decryption Protocol

- Plain Algorithm
  1) Parse $\mathsf{ct} = (\overline{v}, \overline{\mathbf{u}})$
  2) Compute $v^{(p)} = \left\lfloor \frac{p}{2^{d_v}} \cdot v \right\rceil$ and $\mathbf{u}^{(p)} = \left\lfloor \frac{p}{2^{d_u}} \cdot \mathbf{u} \right\rceil$
  3) Output

$$\left\lfloor \frac{2}{p} \cdot \left(v^{(p)} - \mathsf{dk}^\top \cdot \mathbf{u}^{(p)} \bmod p \right) \right\rceil$$

- Apply UT, but with BGV.
  - i.e., Homomorphically compute the decryption formula, using the encryption of dk.



Decryption Protocol (Semi-honest)

1) Given a ciphertext $(\overline{v}, \overline{\mathbf{u}})$, publicly compute

$$v^{(p)} = \left\lfloor \frac{p}{2^{d_v}} \cdot v \right\rceil \quad \text{and} \quad \mathbf{u}^{(p)} = \left\lfloor \frac{p}{2^{d_u}} \cdot \mathbf{u} \right\rceil.$$

2) Homomorphically evaluate

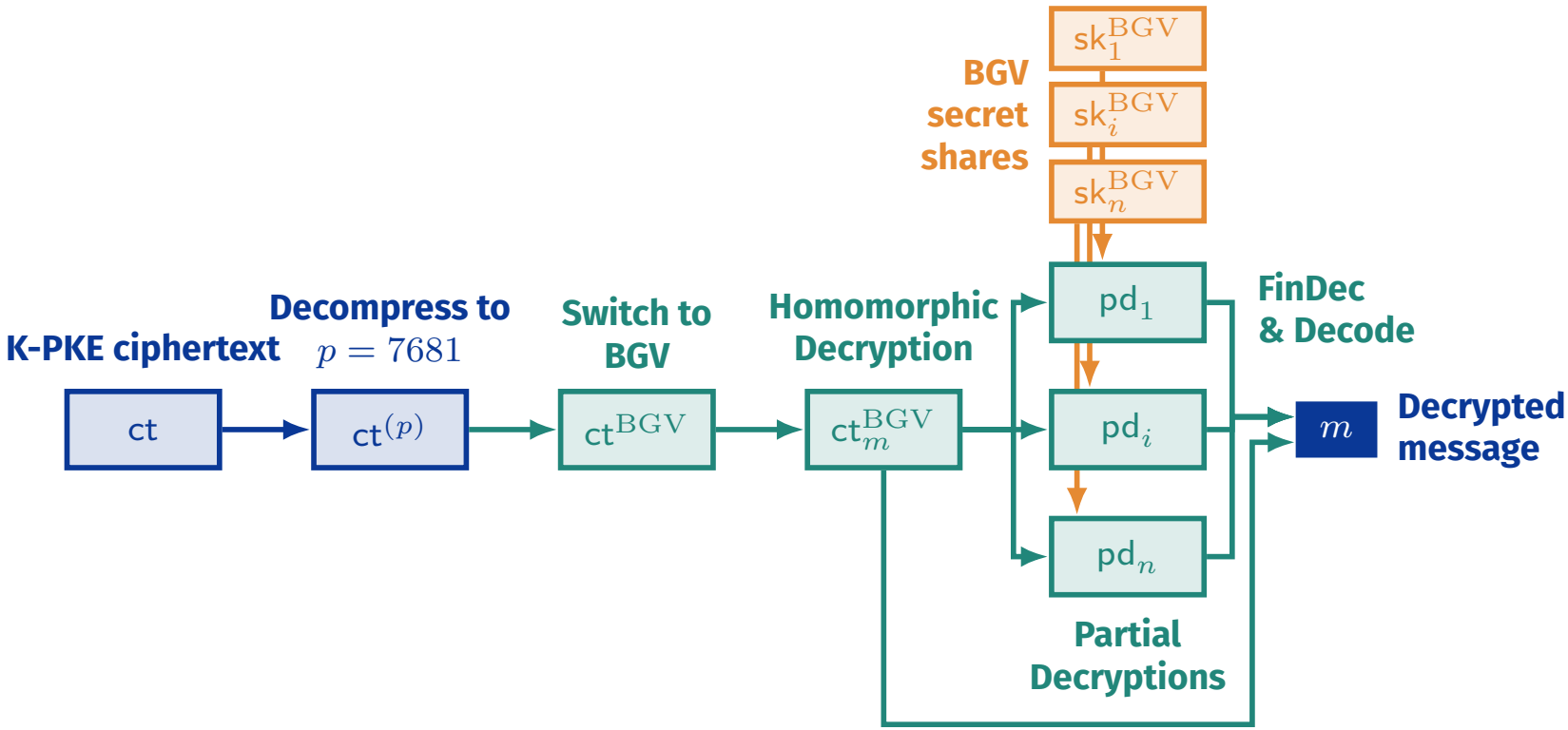
$$m = \left\lfloor \frac{2}{p} \cdot \left(v^{(p)} - \mathsf{dk}^{\top} \cdot \mathbf{u}^{(p)} \bmod p \right) \right\rceil.$$

   - Coefficients-to-Slots linear transformation
   - MSB extraction polynomial evaluation

3) Jointly decrypt the resulting BGV ciphertext.



Decryption Protocol



Malicious Security

- Encryption-Specific
  - Commit-and-reveal for the randomness encryption
- Common (Encryption & Decryption)
  - NIZK to prove the well-formedness of partial decryption



Performance

- Implemented over Lattigo for **ML-KEM-1024**
- Single thread on an Intel Xeon Platinum 8268

Protocol	Time	Communication per party
Encryption	12.13 s	3.5 MiB
Decryption	27.01 s	1.0 MiB

Table: Online (homomorphic computation) benchmark for $n = 16$ and $t = 10$ (semi-honest).



References

[BCD+25] Threshold (Fully) Homomorphic Encryption

[BGG+18] Threshold cryptosystems from threshold fully homomorphic encryption

[BGV12] Fully Homomorphic Encryption without Bootstrapping

[BKSS24] Bootstrapping Small Integers With CKKS

[CKKS17] Homomorphic encryption for arithmetic of approximate numbers

[MHP+26] Distributed key generation for efficient threshold-CKKS

[RST+23] Actively Secure Setup for SPDZ

