NTNU | Norwegian University of Science and Technology

VERIFIABLE DECRYPTION FROM LEARNING WITH ROUNDING

Thomas Haines, **Emil August Hovd Olaisen**,
Peter Browne Rønne, and Tjerand Silde

NIST WPEC — September 26, 2024

Abstract

- Briefly describe verifiable decryption.
- Define n-party distributed decryption, and how this can create verifiable decryption.
- Talk about how learning with rounding (LWR) can create a 2-party scheme.

Verifiable Decryption

- A system that enables a prover with the secret key sk in a Public Key Encryption (PKE) scheme to demonstrate that a ciphertext c decrypts to a given message m using that key.
- The protocol is a zero-knowledge proof of knowledge. It should not leak info about the secret key, nor be open to forgery.
- They play an important role in E-voting schemes and other privacy enhancing applications.

Our contributions

- We generalize the framework from Gjøsteen et al [GHM+22]. They only considered 2-party distributed decryption.
- Using learning with rounding we introduce a post-quantum verifiable decryption scheme which has smaller proof size than Lyubashevsky et al. [LNS21], assuming we are decrypting more than 155 ciphertexts.

n-Party Distributed Decryption

Given a PKE scheme with algorithms $\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec}$ we define the algorithms of n-party distributed decryption:

The dealer algorithm ($\mathsf{Deal}(\mathsf{pk}, \mathsf{sk})$) outputs the secret key shares $\{\mathsf{sk}_i\}_{i=1}^{n}$ and additional auxiliary data aux

The verify algorithm ($\mathsf{Verify}(\mathsf{pk}, \mathsf{aux}, i, \mathsf{sk}_i)$) outputs either *yes* or *no*

The player algorithm ($\mathsf{Play}(\mathsf{sk}_i, c)$) outputs a decryption share ds_i

The reconstruction algorithm ($\mathsf{Rec}(c, \{\mathsf{ds}_i\}_{i=1}^n)$) outputs either an error $\perp$ or a message m.

Correctness

A distributed decryption protocol is **correct** if on input message m and pk with $c = \mathsf{Enc}(\mathsf{pk}, m)$, we have that all $(\{\mathsf{sk}_i\}_{i=1}^n, \mathsf{aux})$ generated by the dealer algorithm Deal satisfies $\mathsf{Verify}(\mathsf{pk}, \mathsf{aux}, i, \mathsf{sk}_i) = 1$ for $1 \leq i \leq n$, and that

$$\mathsf{Rec}(c, \{\mathsf{Play}(\mathsf{sk}_i, c)\}_{i=1}^n) = m.$$

Verifiable Decryption from Distributed Decryption

How does verifiable decryption follow? Suppose we want to prove that $m = \mathsf{Dec}(c, \mathsf{sk})$.

1. The prover runs Deal α times to create the key shares $\{\mathsf{sk}_{i,k}\}_{i=1}^{n}, \mathsf{aux}_k$ for $1 \leq k \leq \alpha$, they commit to these shares. They also generate $\mathsf{ds}_{i,k} = \mathsf{Play}(\mathsf{sk}_{i,k}, c)$ and send decryption share and auxiliary data.

2. The verifier sends back a vector $\phi \in \{1, 2, \ldots n\}^{\alpha}$.

3. The prover sends back the secret key shares $\mathsf{sk}_{i,k}$ unless $i \neq \phi[k]$.

4. For all $1 \leq i \leq n, 1 \leq k \leq \alpha$ the verifier checks if $\mathsf{Rec}(c, \{\mathsf{ds}_{i,k}\}_{i=1}^{n}) = m$. They also check if $\mathsf{Play}(\mathsf{sk}_{i,k}, c) = \mathsf{ds}_{i,k}$ and if $\mathsf{Verify}(\mathsf{pk}, \mathsf{aux}_k, i, \mathsf{sk}_{i,k})$ holds true whenever $i \neq \phi[k]$.

Prover(pk, $\{c_j, m_j\}_{j=1}^{\tau}$; sk)		Verifier(pk, $\{c_j, m_j\}_{j=1}^{\tau}$)
For each round $k \in [\alpha]$:		
Split sk in n shares $\{\mathsf{sk}_{i,k}\}$		
Commit to each key share $\mathsf{sk}_{i,k}$		
For each key share $i \in [n]$ and each ciphertext $j \in [\tau]$:		
Partially decrypt c_j using $\mathsf{sk}_{i,k}$ to get decryption share $\mathsf{ds}_{i,j,k}$		
$w \leftarrow$ Set of all commitments and decryption shares	$\xrightarrow{w}$	
	$\xleftarrow{\phi}$	$\phi \leftarrow \alpha$ challenges from $[n]$
$z \leftarrow$ All except challenged key shares $\mathsf{sk}_{i,k}$ from ϕ	$\xrightarrow{z}$	
		For each round $k \in [\alpha]$:
		For each $i \in [n]$ and $j \in [\tau]$:
		Verify that key share is correct
		Re-compute decryption share
		Verify reconstructed message

Figure: High-level overview of the verifiable decryption in the head protocol.

Benefits of the Framework

- Only the number of decryption shares increases as the number of ciphertexts increases.
- As a consequence the frameworks is well suited to applications with a large number of ciphertexts such as electronic voting.
- In addition; the framework is ideal for distributes decryption schemes with small decryption shares.
- We achieve this using Learning with Rounding.

Learning with Errors (LWE) and Learning with rounding (LWR)

Let q and n be positive integers, and let Φ be a distribution over $\mathbb{Z}_q^n$, the LWE$_{n,q,\Phi}$ problem is to distinguish the distributions with $A \xleftarrow{\$} \mathbb{Z}_q^{n \times n}$ and $\mathbf{s}, \mathbf{e} \leftarrow \Phi$:

$$(A, A\mathbf{s} + \mathbf{e} \mod q)$$

$$(A, b), b \xleftarrow{\$} \mathbb{Z}_q^n$$

Learning with Errors (LWE) and Learning with rounding (LWR)

Let $p < q$ and n be positive integers, and let Φ be a distribution over $\mathbb{Z}_q^n$, the LWR$_{n,p,q,\Phi}$ problem is to distinguish the distributions with $A \leftarrow\$ \mathbb{Z}_q^{n\times n}$ and $\mathbf{s} \leftarrow \Phi$:

$$(A, (A\mathbf{s} \mod q) \mod p)$$

$$(A, b), b \leftarrow\$ \mathbb{Z}_p^n$$

From Learning with Rounding to 2-party distributed decryption

- Let R_p, R_q be the respective rings $\mathbb{Z}[X]/(p, X^N + 1), \mathbb{Z}[X]/(q, X^N + 1)$

- Suppose we have a message $m \in R_p$ such that:

$$m = ((v - u_0 - u_1) \mod q) \mod p$$

  with $v, u_0, u_1 \leftarrow\$ R_q$ and $t_0 = u_0 \mod p, t_1 = u_1 \mod p$, we can use Lemma 1 [BKS19] to show that:

$$m = ((v - t_0 - t_1) \mod q) \mod p$$

  with high probability.

- We may think of t_0, t_1 as decryption shares $\mathsf{ds}_0, \mathsf{ds}_1$.

Contributions

Verifiable decryption scheme	Encryption scheme	Ciphertext size	Plaintext size	Amortized proof size
Gjøsteen et al. [GHM+22]	BGV	28.2 KB	2048 bits	$(4883/\tau + 1.8)$ MB
Our protocol Π_2	BGV	28.2 KB	2048 bits	$(2691/\tau + 32.8)$ KB
Lyubashevsky et al. [LNS21]	Kyber-512	0.8 KB	256 bits	43.6 KB
Our protocol Π_2	M − LWE	19.9 KB	256 bits	$(3181/\tau + 4.1)$ KB

Table: Amortized comparison between verifiable decryption schemes for $\lambda = 128$.

References

- Elette Boyle, Lisa Kohl, and Peter Scholl.
  Homomorphic secret sharing from lattices without FHE.
  pages 3–33, 2019.
- Kristian Gjøsteen, Thomas Haines, Johannes Müller, Peter B. Rønne, and Tjerand Silde.
  Verifiable decryption in the head.
  pages 355–374, 2022.
- Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler.
  Shorter lattice-based zero-knowledge proofs via one-time commitments.
  pages 215–241, 2021.