Malicious prover can craft malformed proofs and see whether verifier accepts or not $\Longrightarrow$ Given verifier's challenge, prover can construct proofs of arbitrary statements

but verifier cannot participate...

To get reusable soundness in the designated-verifier model, we need to refresh the challenge (different challenge for each statement)

Idea: replace PKE with ABE: different "randomness" can be used to check proofs of different statements

Let n be the length of the statement

Let $F : \mathcal{K} \times ([\lambda] \times \{0,1\}^n) \to \{0,1\}$ be a PRF $\leftarrow$ will be used to derive challenge bit for statement x (for each index $1, \dots, \lambda$)

Let $k \xleftarrow{R} \mathcal{K}$. Define the function

$$f_{i^*}(x, i, b) := \begin{cases} 0 & \text{if } F(k, (i, x)) = b \text{ and } i = i^* \\ 1 & \text{otherwise} \end{cases}$$

Setup(1^λ): sample $k \xleftarrow{R} \mathcal{K}$ and $(mpk, msk) \leftarrow$ ABE.Setup(1^λ)

sample $sk_i \leftarrow$ ABE.KeyGen(msk_i, f_i) for each $i \in [\lambda]$

output $pk = mpk$

$sk = (sk_1, \dots, sk_\lambda, k)$

Prove(pk, x, w): construct first message $\sigma_1, \dots, \sigma_\lambda$ of λ copies of the Σ-protocol

for each σ_i, compute responses $z_i^{(0)}$ and $z_i^{(1)}$ to be the responses associated with challenge bit 0 and 1

compute $ct_i^{(b)} \leftarrow$ ABE.Encrypt$(mpk, (x, i, b), z_i^{(b)})$

output $\pi = (\sigma_1, \dots, \sigma_\lambda, ct_1^{(0)}, ct_1^{(1)}, \dots, ct_\lambda^{(0)}, ct_\lambda^{(1)})$

Verify(sk, x, π): evaluate $b_i \leftarrow F(k, (i, x))$ for each $i \in [\lambda]$

compute response $z_i^{(b_i)} \leftarrow$ ABE.Decrypt$(sk_i, ct_i^{(b_i)})$

check that $(\sigma_i, b_i, z_i^{(b_i)})$ is valid for each $i \in [\lambda]$

Completeness. By definition, $f_i(x, i, b) = 0$ when $b_i = F(k, (i, x))$ so verifier is able to recover $z_i^{(b_i)}$ for each $i \in [\lambda]$.
Completeness follows from completeness of the underlying HVZK.

Zero-Knowledge: Follows by a similar argument as before.

For all $i, j \in [\lambda]$, $f_i(x, j, b) = 1$ when $b = 1 - F(k, (i, x))$

$\hookrightarrow$ By ABE security, $ct_i^{(1-b_i)}$ is computationally indistinguishable from encryption of all-zeroes string

Soundness: Ideally, want to argue that k is hidden to adversary $\Longrightarrow$ uniform, independent challenge $b_1, \dots, b_\lambda \xleftarrow{R} \{0,1\}$ used to check each statement x

$\hookrightarrow$ Verifier rejection attack no longer works: randomness associated with statement x is independent of randomness associated with statement x^*

Problem: Adversary gets to query the verification oracle which invokes ABE.Decrypt$(sk_i, \cdot)$ on an adversarially-chosen ciphertext

$\hookrightarrow$ Output of decryption oracle could leak information about sk_i (which contains information about the PRF key)

To address this, we need to make sure that oracle access to Decrypt$(sk_f, \cdot)$ does not leak information about f other than whether decryption succeeded or not (i.e., whether $f(x) = 0$ or $f(x) = 1$)

Easy to achieve this property with lattice-based ABE scheme:

Recall structure of ABE ciphertexts

$s^T A$ + error

$s^T [B_1 - x_1 G \mid \cdots \mid B_\ell - x_\ell G]$ + error

$s^T p + \mu \cdot \lfloor \frac{q}{2} \rfloor$ + error

We will also include x as part of the ciphertext

$[A \mid B_f] \cdot T_f = G$ where $\|T_f\|$ is small

Let secret key for f be a trapdoor T_f for $[A \mid B_f]$ where $B_f = [B_1 \mid \cdots \mid B_\ell] \cdot H_f$

To decrypt, we

$$(s^T [B_1 - x_1 G \mid \cdots \mid B_\ell - x_\ell \cdot G] + \text{error}) H_{f,x} \approx s^T (B_f - f(x) \cdot G)$$
$$= s^T B_f \quad \text{when} \quad f(x) = 0$$

Given $s^T [A \mid B_f]$ + error and T_f, can solve LWE and recover the secret key s and the error

↳ Given s, μ, can recover the error from the ABE ciphertext

Decryption outputs message μ only if errors are sufficiently small (i.e., such that decryption with sk_f always outputs μ)

Key observation: if errors small enough such that Decrypt$(sk_f, ct) \to \mu$ (regardless of which trapdoor we used), then we can also implement decryption using a trapdoor for A

Namely if $A \cdot T_A = G$ and A is short, we can again recover the LWE secret s and the errors (and implement the same size checks)

In other words: decryption introduces a ciphertext validity check with the guarantees:

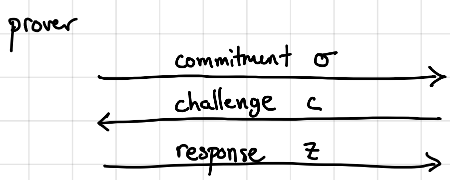
1) If validity check passes, then decryption with sk_f can be simulated by decryption with msk
2) If validity check does not pass, then decryption always outputs $\perp$

Takeaway: Decrypt$(sk_f, \cdot)$ hides information about f other than value of $f(x)$

Validity check passing / not passing depends only on ciphertext, not decryption key

With this "function-hiding" property, we can appeal to PRF security to argue that independent randomness is used to check proofs of each statement — can now reduce soundness to soundness of underlying Σ-protocol

What about public verification? Let's recall an approach in the random oracle model:



To obtain a NIZK in the ROM, we can derive

$$c \leftarrow H(x, \sigma) \quad \text{where } H \text{ is modeled as a random oracle}$$

Random oracle functions as verifier's randomness — challenge is determined only after prover has selected a commitment and is unpredictable a priori

Can we remove the random oracle?

↳ In practice: instantiate random oracle with a cryptographic hash function (e.g., SHA-256)

does not admit a proof of security to a property like collision-resistance

Can we identify a sufficient condition for security?