

Fully Homomorphic Encryption with Chosen-Ciphertext Security from LWE

Rupeng Yang

UOW

Zuoxia Yu

UOW

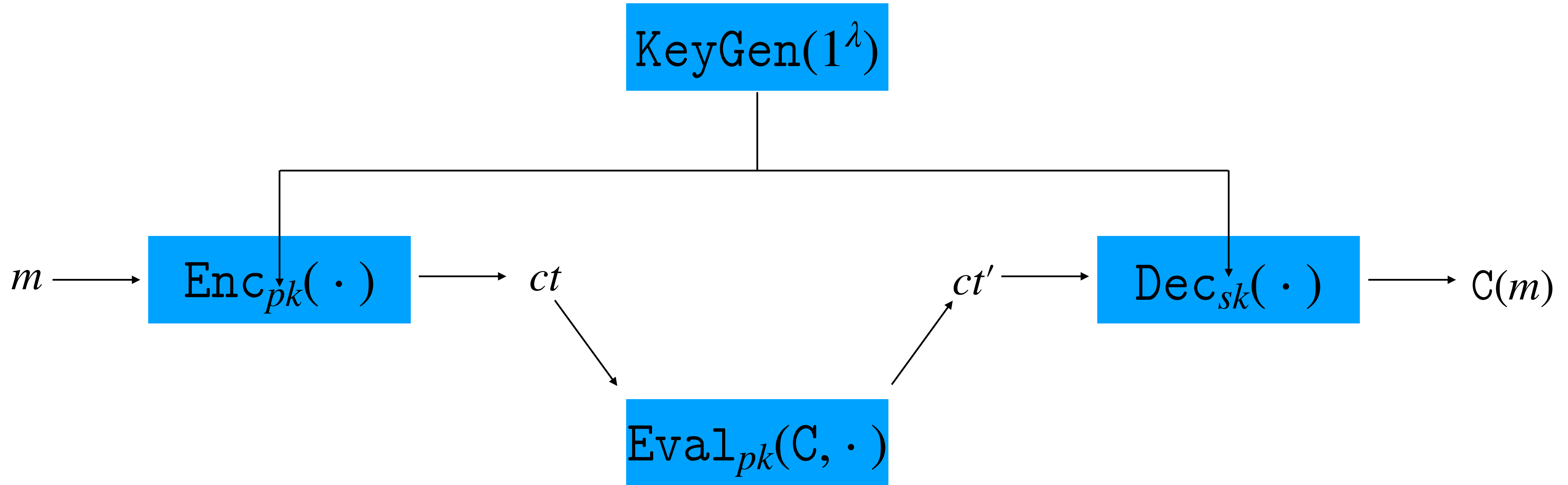
Willy Susilo

UOW

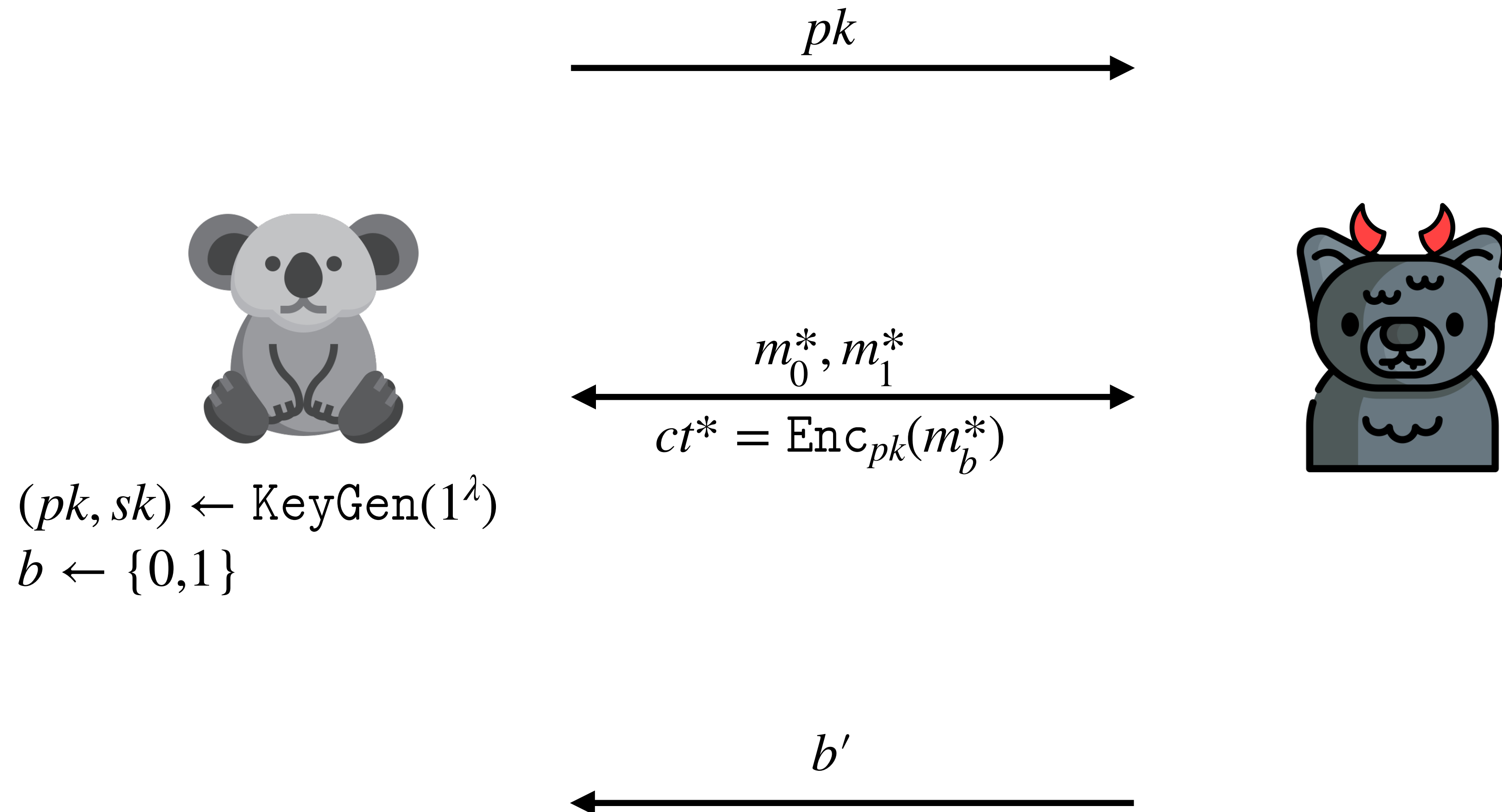


UNIVERSITY OF WOLLONGONG, AUSTRALIA

Fully Homomorphic Encryption



IND-CPA Security of FHE



Why FHE Is Useful

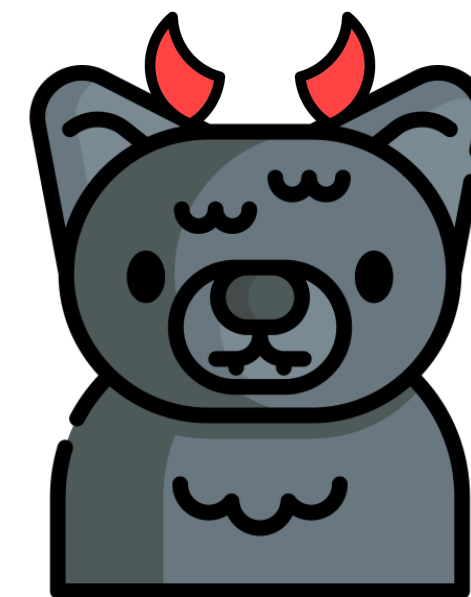


X

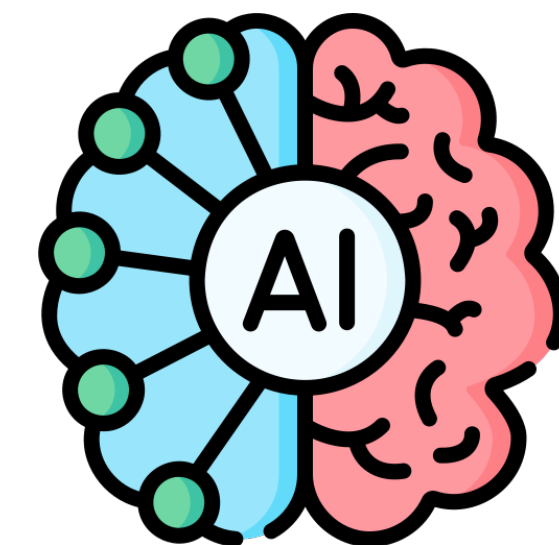
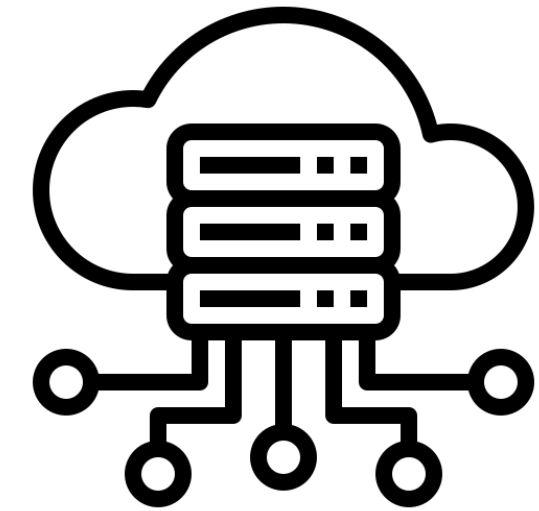
$pk, \text{Enc}_{pk}(X)$



$\text{Eval}_{pk}(C, ct)$

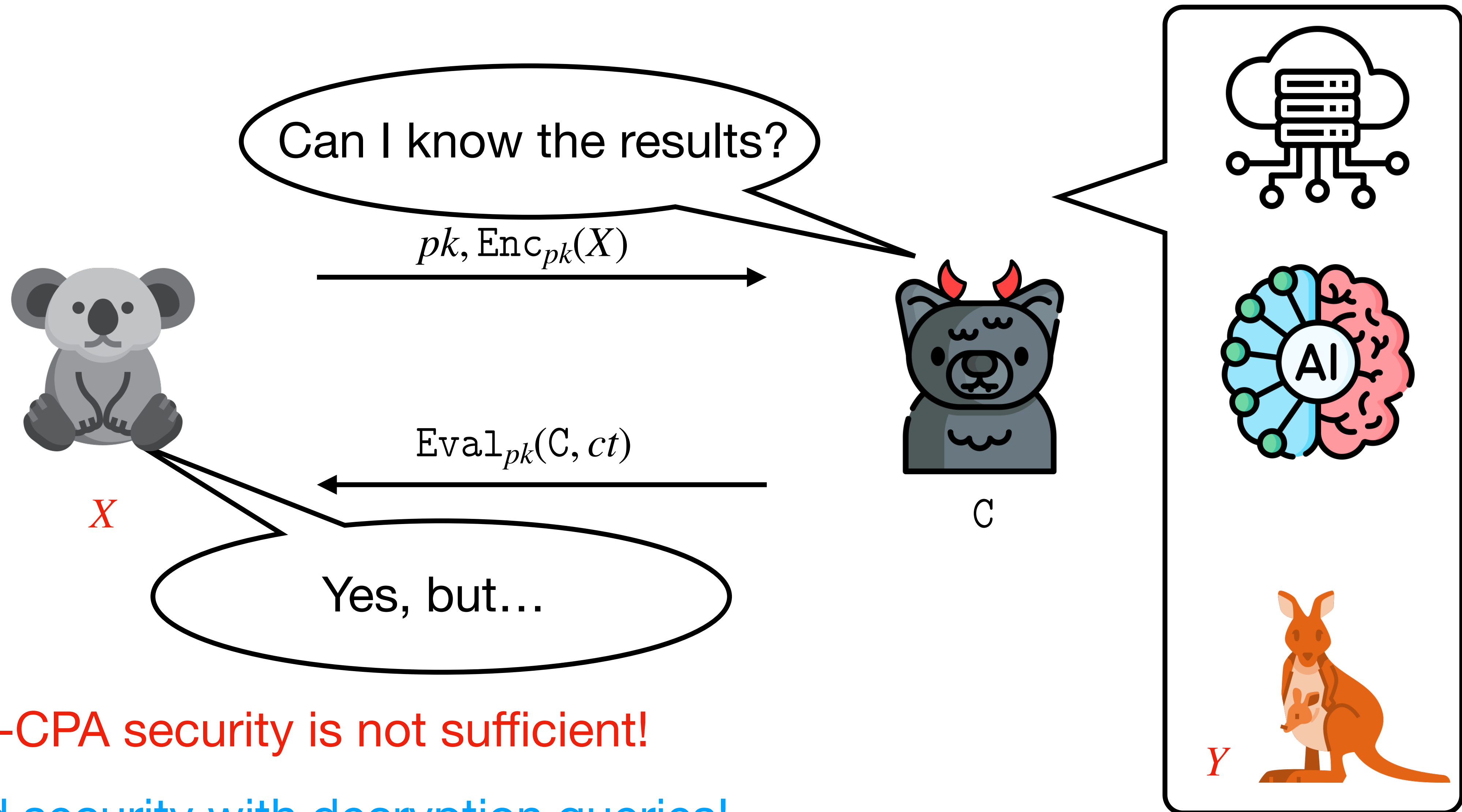


C



Y

Why FHE Is Useful



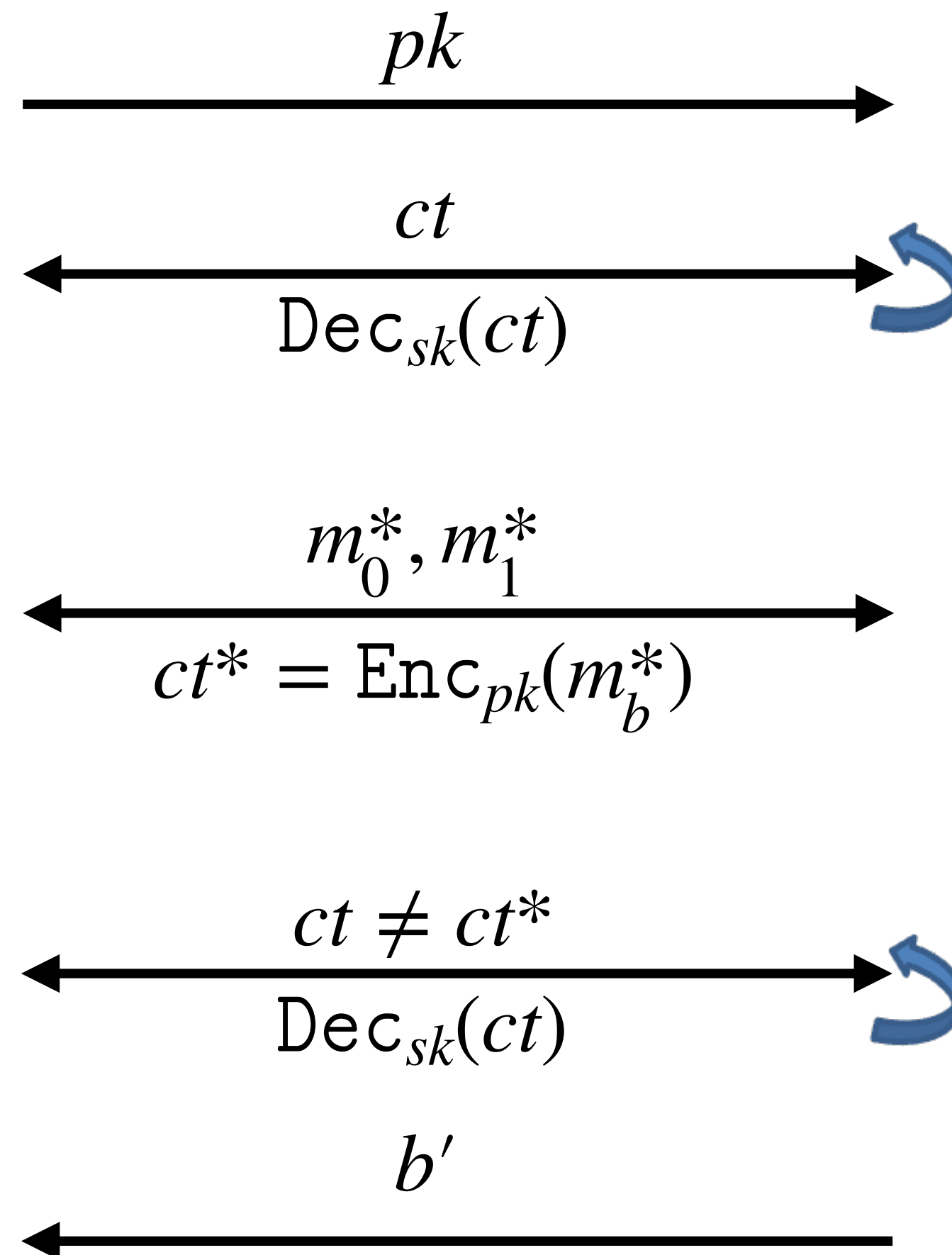
The IND-CPA security is not sufficient!

We need security with decryption queries!

IND-CCA2 Security of FHE



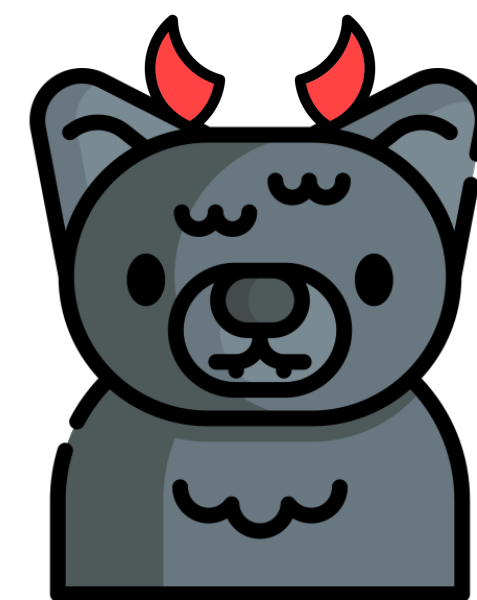
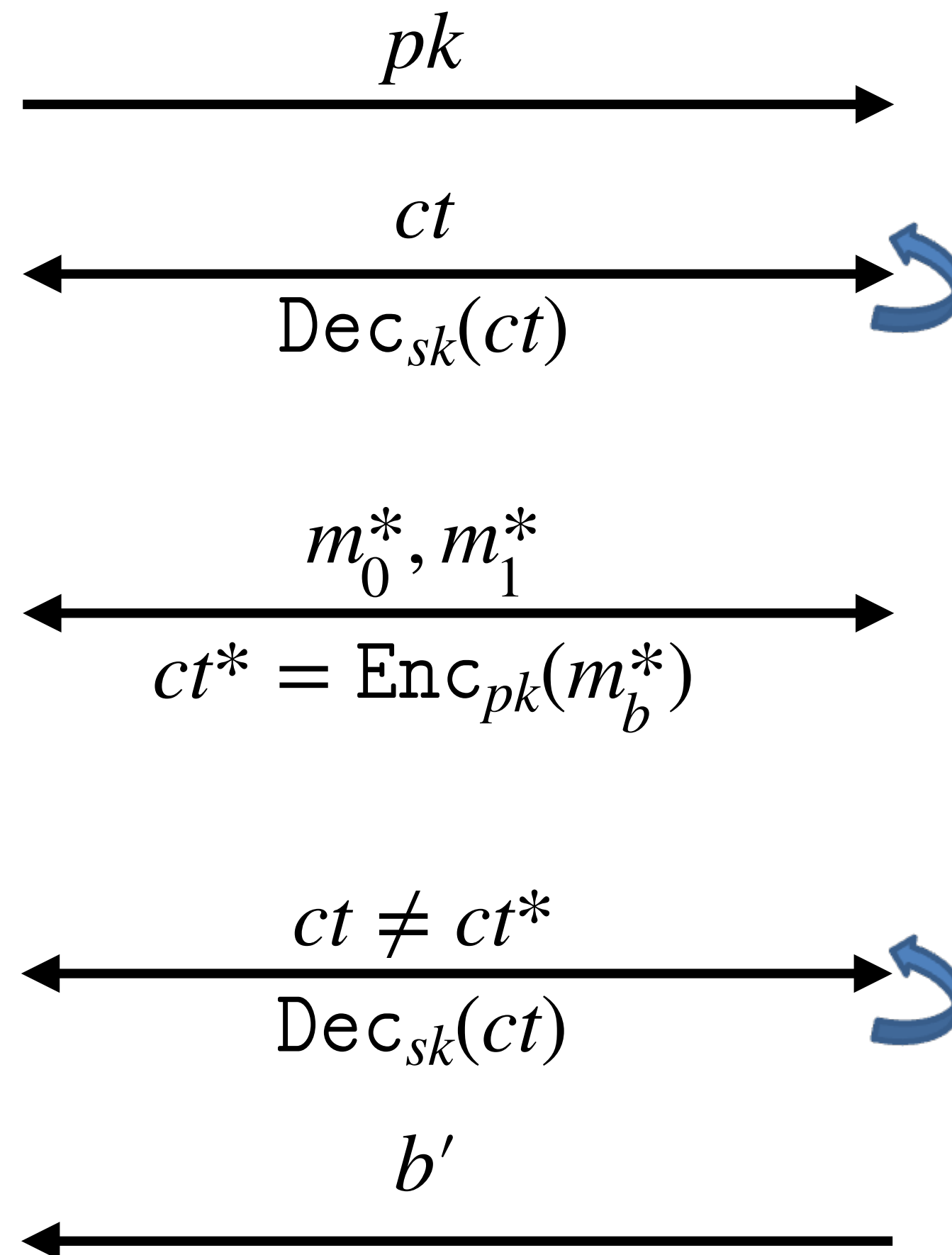
$(pk, sk) \leftarrow \text{KeyGen}(1^\lambda)$
 $b \leftarrow \{0,1\}$



IND-CCA2 Security of FHE



$(pk, sk) \leftarrow \text{KeyGen}(1^\lambda)$
 $b \leftarrow \{0,1\}$

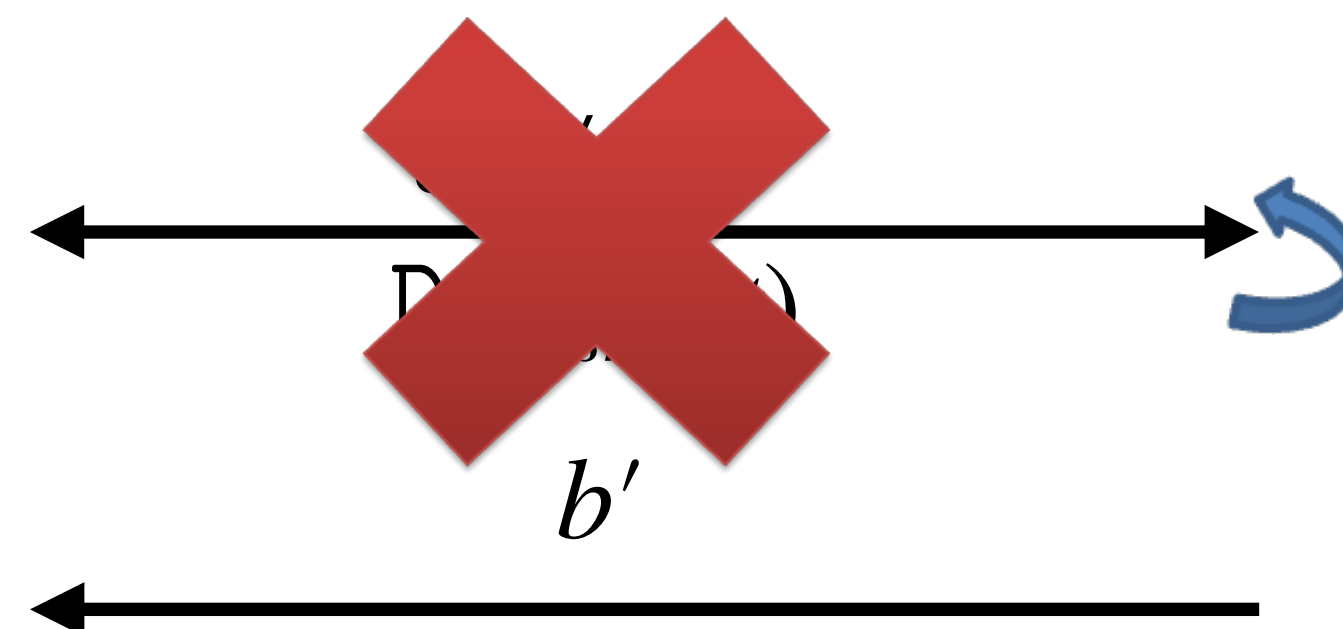
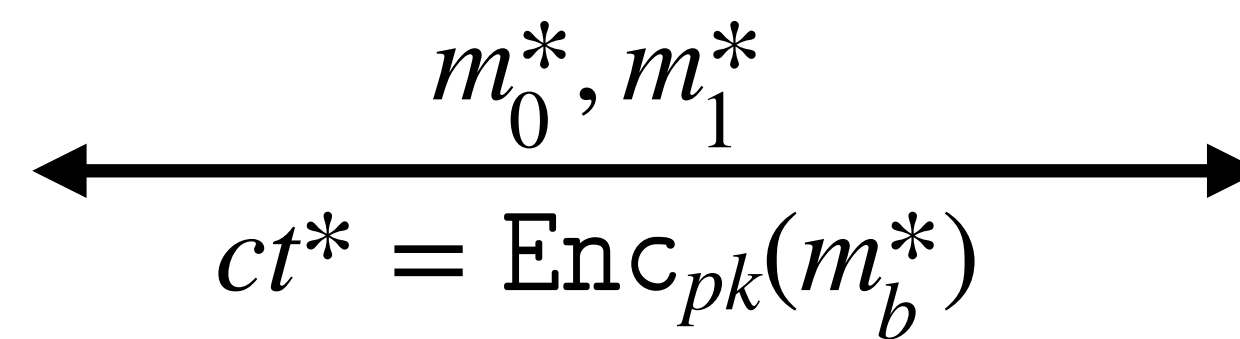
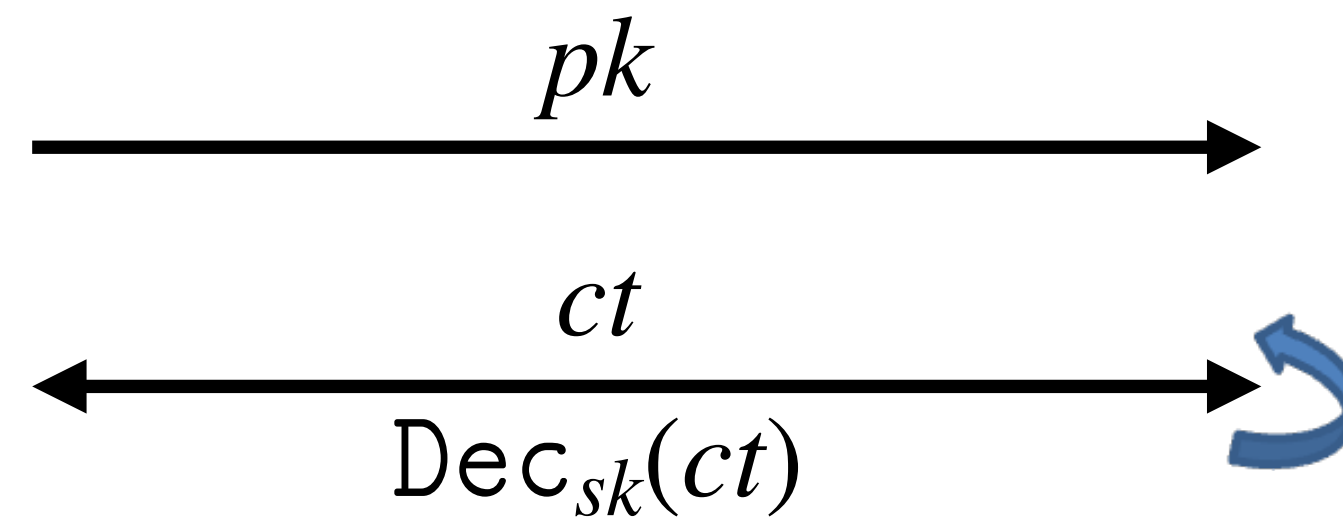


However, ct here can be evaluated from ct^* ; so, there is no hope to achieve IND-CCA2 security for FHE!

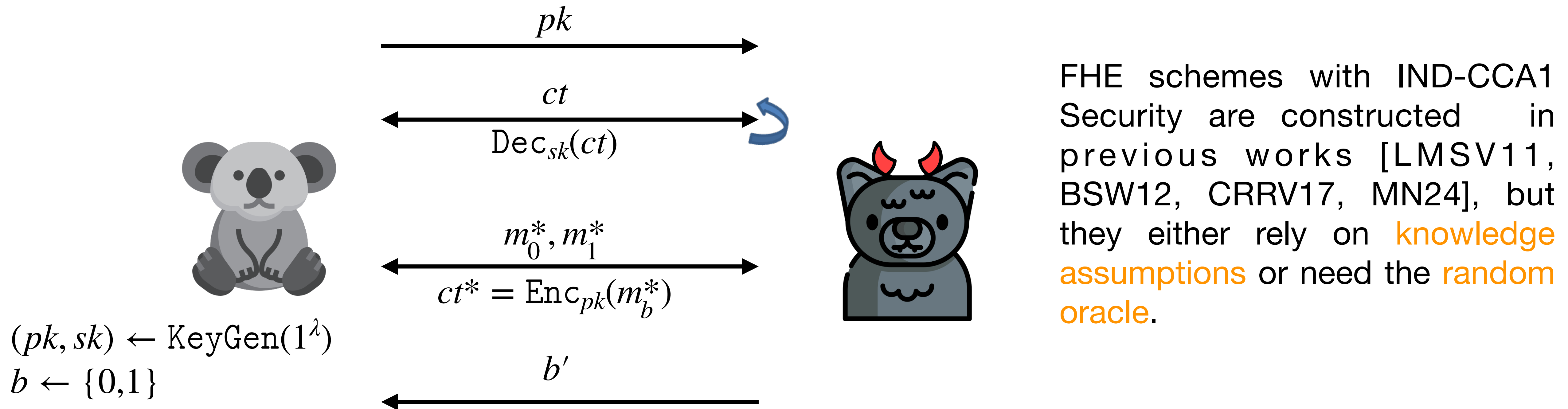
IND-CCA1 Security of FHE



$(pk, sk) \leftarrow \text{KeyGen}(1^\lambda)$
 $b \leftarrow \{0,1\}$



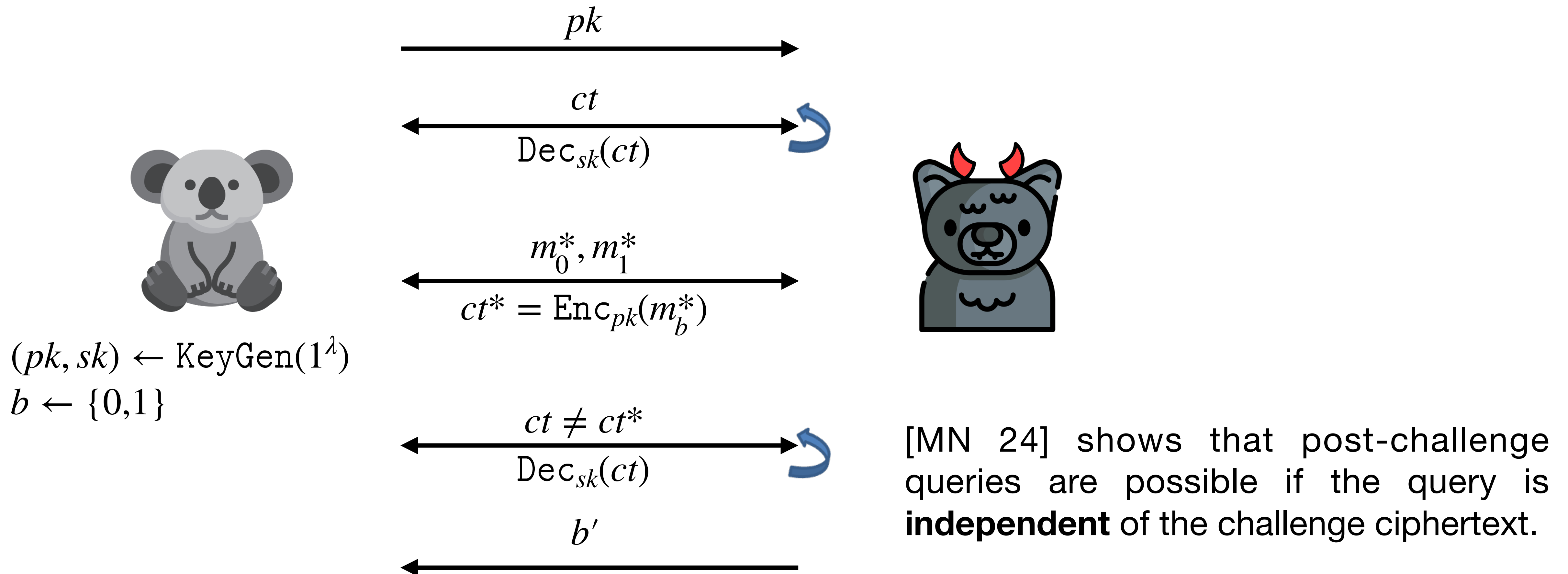
IND-CCA1 Security of FHE



FHE schemes with IND-CCA1 Security are constructed in previous works [LMSV11, BSW12, CRRV17, MN24], but they either rely on **knowledge assumptions** or need the **random oracle**.

- [LMSV 11] Jake Loftus, Alexander May, Nigel P Smart, and Frederik Vercauteren. On CCA-secure somewhat homomorphic encryption. In SAC, 2011.
- [BSW 12] Dan Boneh, Gil Segev, and Brent Waters. Targeted malleability: homomorphic encryption for restricted computations. In ITCS, 2012.
- [CRRV 17] Ran Canetti, Srinivasan Raghuraman, Silas Richelson, and Vinod Vaikuntanathan. Chosen-ciphertext secure fully homomorphic encryption. In PKC, 2017.
- [MN 24] Mark Manulis and Jérôme Nguyen. Fully homomorphic encryption beyond IND-CCA1 security: Integrity through verifiability. In EUROCRYPT, 2024.

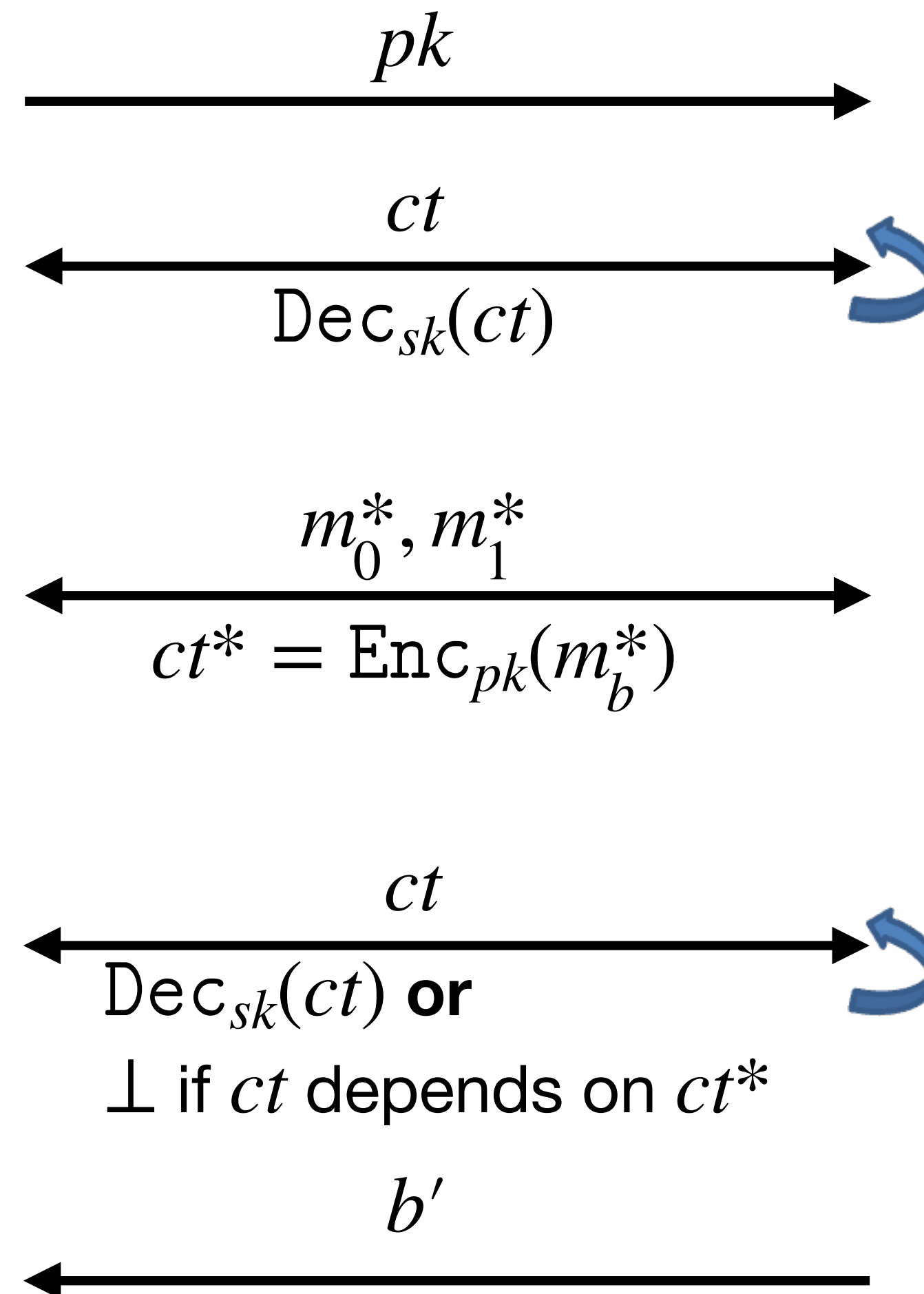
Beyond CCA1 Security of FHE



VCCA Security of FHE



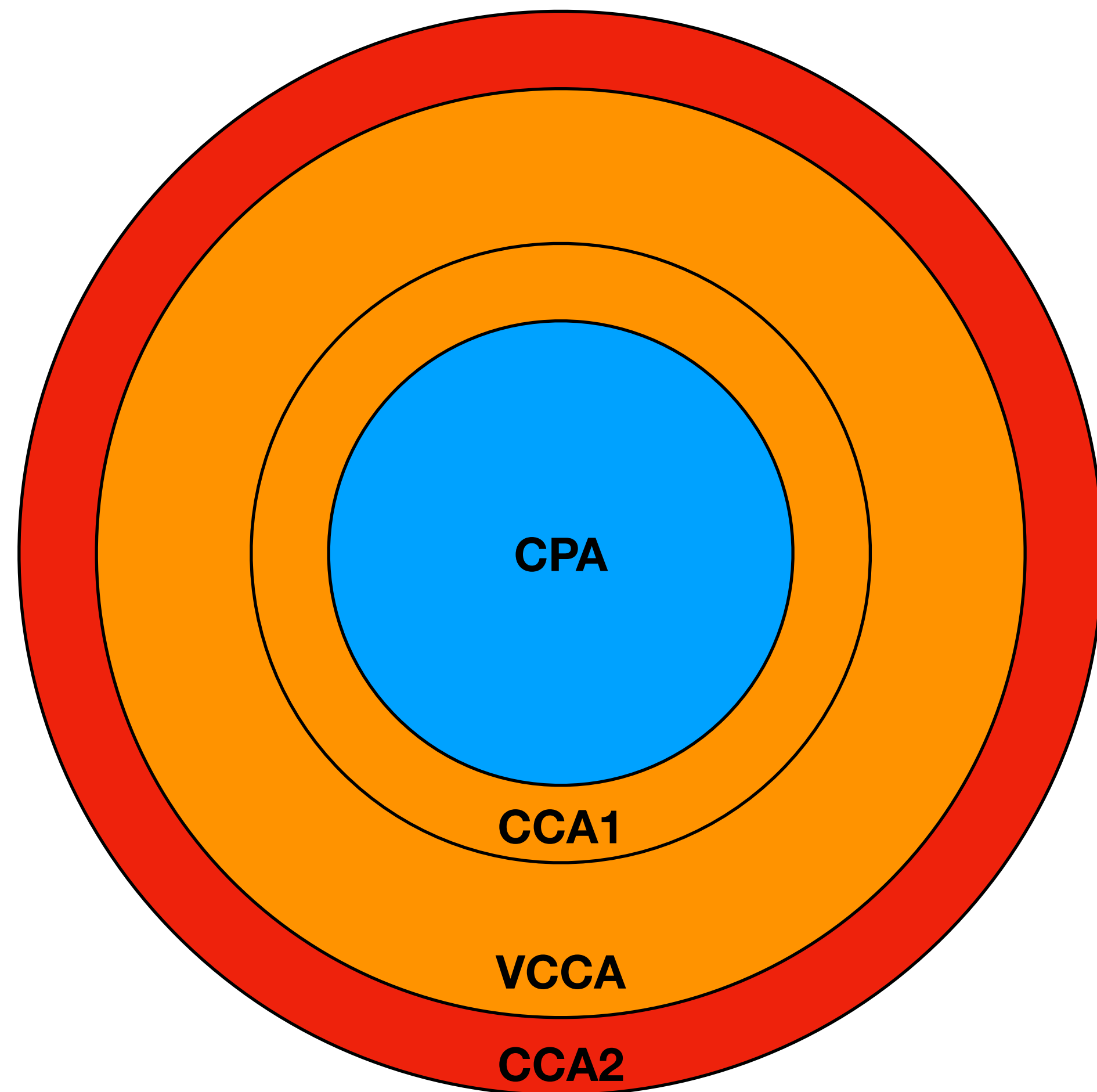
$(pk, sk) \leftarrow \text{KeyGen}(1^\lambda)$
 $b \leftarrow \{0,1\}$






How to check if ct depends on ct^* ?

- Definition: Assume that is possible
- Construction: Extract all details of the computation that outputs ct ; that is possible if the computation is proved by a **ZK-SNARK**.

FHE with Decryption Queries



 = known from standard assumption
 = known from non-standard assumption
 = Impossible

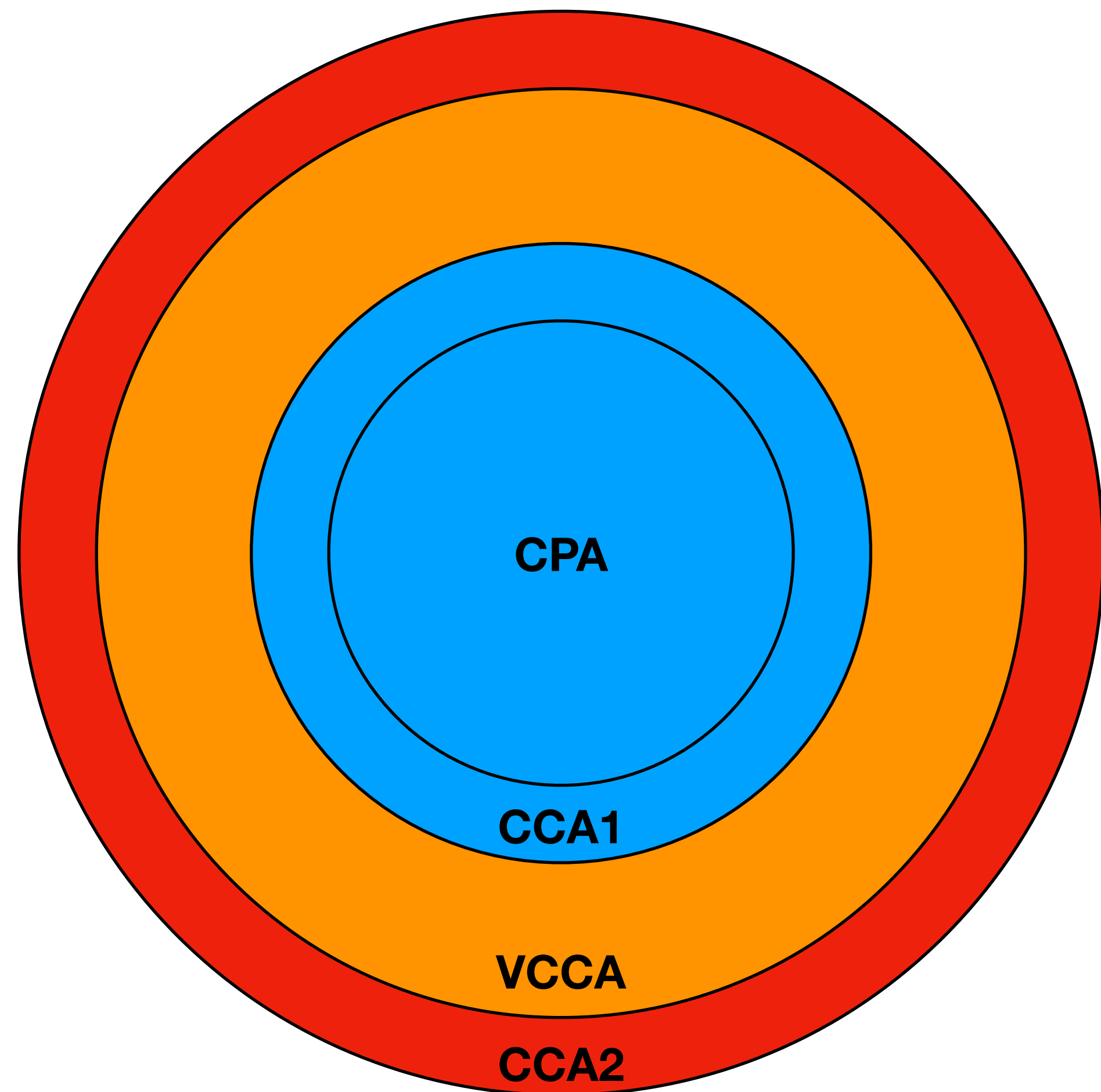
State-of-the-art:

- FHE with CPA security is possible from standard **(circular-secure) LWE** assumptions.
- But if we hope to have security with decryption queries, we must rely on non-standard assumptions such as **knowledge assumptions** or need the **random oracle**.

The Problem:

*How to construct FHE schemes with CCA1 security (or even more) from **(circular-secure) LWE** assumptions in the standard model?*

Our Results



The Problem:

*How to construct FHE schemes with CCA1 security (or even more) from **(circular-secure) LWE** assumptions in the standard model?*

Our Results:

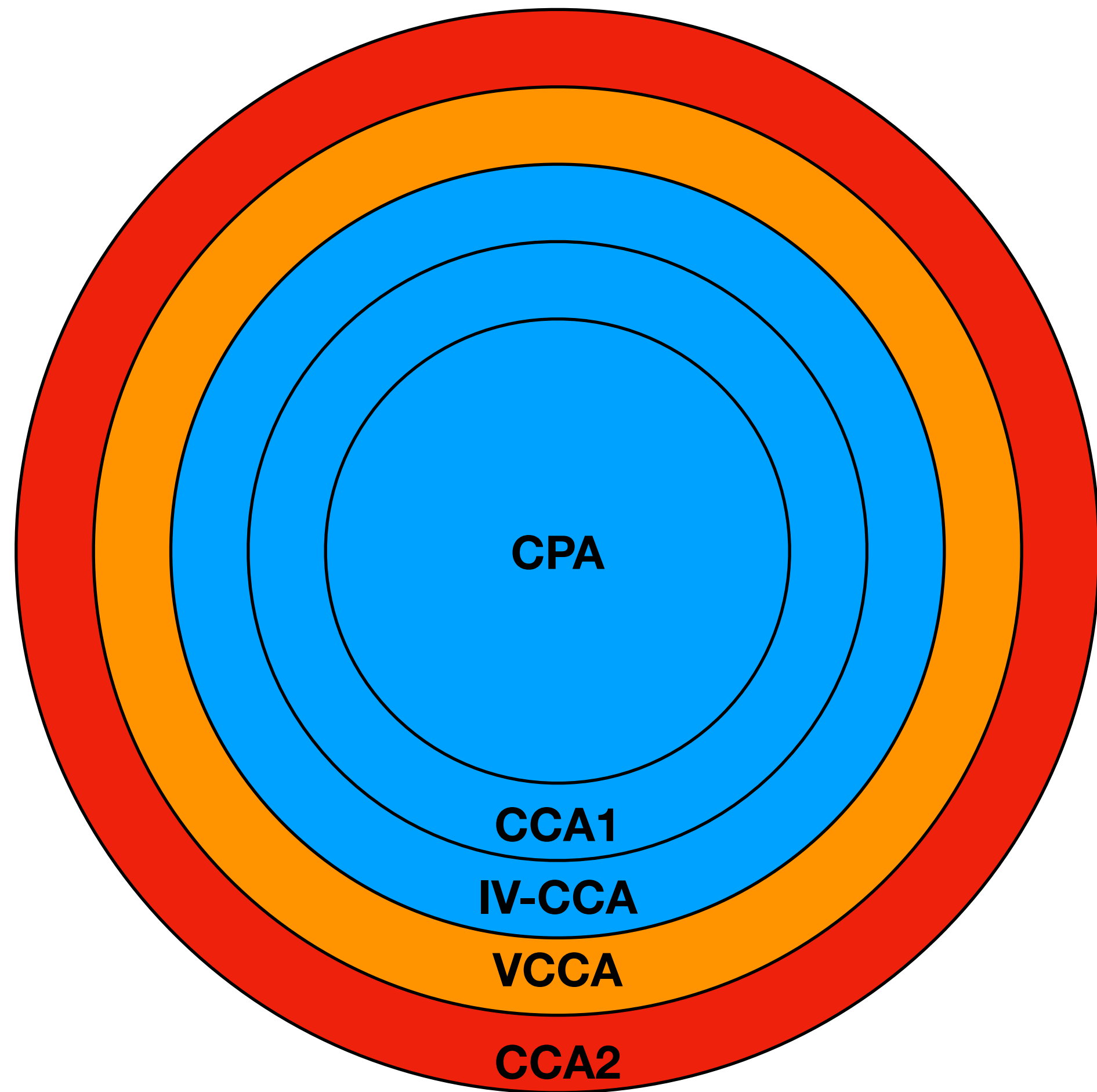
1. We show how to construct FHE with IND-CCA1 security from **circular-secure LWE**.




CPA
= known from standard assumption

CCA1
= known from non-standard assumption

CCA2
= Impossible

Our Results



 = known from standard assumption
 = known from non-standard assumption
 = Impossible

The Problem:

How to construct FHE schemes with CCA1 security (or even more) from [\(circular-secure\) LWE](#) assumptions in the standard model?

Our Results:

1. We show how to construct FHE with IND-CCA1 security from [circular-secure LWE](#).
2. We define a new security called input-verifiable CCA (IV-CCA) security and construct FHE with IV-CCA security from [circular-secure LWE](#).
 - The IV-CCA security is strictly stronger than the CCA1 security, but is weaker than the VCCA security.

Our Results

- **We construct FHE schemes with IND-CCA1 security from circular-secure LWE.**
- We define a new security notion called IV-CCA security, which lies between IND-CCA1 security and VCCA security.
- We construct FHE schemes with IV-CCA security from circular-secure LWE.

Warmup: The Naor-Yung Paradigm

KeyGen(1^λ) :

$(sk_1, pk_1) \leftarrow \text{FHE}.\text{KeyGen}(1^\lambda)$

$(sk_2, pk_2) \leftarrow \text{FHE}.\text{KeyGen}(1^\lambda)$

$crs \leftarrow \text{ZK}.\text{Setup}(1^\lambda)$

$PK = (pk_1, pk_2, crs)$

$SK = sk_1$

Enc(PK, m) :

$c_1 = \text{FHE}.\text{Enc}(pk_1, m)$

$c_2 = \text{FHE}.\text{Enc}(pk_2, m)$

$\pi \leftarrow \text{ZK}.\text{Prove}(crs,$
 $c_1 \text{ \& } c_2 \text{ encrypt } m)$

$CT = (c_1, c_2, \pi)$

Dec(SK, CT) :

If π is valid:

Output $\text{FHE}.\text{Dec}(sk_1, ct_1)$

Output \perp

Warmup: The Naor-Yung Paradigm

KeyGen(1^λ) :	Enc(PK, m) :	Dec(SK, CT) :	Eval($PK, C, \overrightarrow{CT} = (\vec{c}_1, \vec{c}_2, \vec{\pi})$) :
$(sk_1, pk_1) \leftarrow \text{FHE}.\text{KeyGen}(1^\lambda)$	$c_1 = \text{FHE}.\text{Enc}(pk_1, m)$	If π is valid:	$c'_1 = \text{FHE}.\text{Eval}(pk_1, C, \vec{c}_1)$
$(sk_2, pk_2) \leftarrow \text{FHE}.\text{KeyGen}(1^\lambda)$	$c_2 = \text{FHE}.\text{Enc}(pk_2, m)$	Output $\text{FHE}.\text{Dec}(sk_1, ct_1)$	$c'_2 = \text{FHE}.\text{Eval}(pk_2, C, \vec{c}_2)$
$crs \leftarrow \text{ZK}.\text{Setup}(1^\lambda)$	$\pi \leftarrow \text{ZK}.\text{Prove}(crs,$	Output \perp	$\pi' \leftarrow ???$
$PK = (pk_1, pk_2, crs)$	$c_1 \ \& \ c_2 \text{ encrypt } m)$		
$SK = sk_1$	$CT = (c_1, c_2, \pi)$		

Warmup: The Naor-Yung Paradigm

KeyGen(1^λ) :	Enc(PK, m) :	Dec(SK, CT) :	Eval($PK, C, \vec{CT} = (\vec{c}_1, \vec{c}_2, \vec{\pi})$) :
$(sk_1, pk_1) \leftarrow \text{FHE.KeyGen}(1^\lambda)$	$c_1 = \text{FHE.Enc}(pk_1, m)$	If π is valid:	$c'_1 = \text{FHE.Eval}(pk_1, C, \vec{c}_1)$
$(sk_2, pk_2) \leftarrow \text{FHE.KeyGen}(1^\lambda)$	$c_2 = \text{FHE.Enc}(pk_2, m)$	Output $\text{FHE.Dec}(sk_1, ct_1)$	$c'_2 = \text{FHE.Eval}(pk_2, C, \vec{c}_2)$
$crs \leftarrow \text{ZK.Setup}(1^\lambda)$	$\pi \leftarrow \text{ZK.Prove}(crs,$	Output \perp	$\pi' \leftarrow ???$
$PK = (pk_1, pk_2, crs)$	$c_1 \ \& \ c_2 \text{ encrypt } m)$		
$SK = sk_1$	$CT = (c_1, c_2, \pi)$		

The problem: How to generate the proof for an *evaluated* ciphertext?

- The direct generation needs the randomness of the encryption as a witness. 😞😞😞
- We can prove that the two ciphertext components are evaluated from ciphertexts encrypting the same message using the same circuit. 😊
- But the proof size will usually be a polynomial in the circuit size. 😞😞😞
- We can use a SNARG to generate a succinct proof 😊😊😊
 - The SNARG can prove succinct proof whose size is independent of the statement/witness size
 - But a general SNARG relies on either **random oracle or non-falsifiable assumptions** 😞

Warmup: The Naor-Yung Paradigm

KeyGen(1^λ) :	Enc(PK, m) :	Dec(SK, CT) :	Eval($PK, C, \vec{CT} = (\vec{c}_1, \vec{c}_2, \vec{\pi})$) :
$(sk_1, pk_1) \leftarrow \text{FHE}.\text{KeyGen}(1^\lambda)$	$c_1 = \text{FHE}.\text{Enc}(pk_1, m)$	If π is valid:	$c'_1 = \text{FHE}.\text{Eval}(pk_1, C, \vec{c}_1)$
$(sk_2, pk_2) \leftarrow \text{FHE}.\text{KeyGen}(1^\lambda)$	$c_2 = \text{FHE}.\text{Enc}(pk_2, m)$	Output $\text{FHE}.\text{Dec}(sk_1, ct_1)$	$c'_2 = \text{FHE}.\text{Eval}(pk_2, C, \vec{c}_2)$
$crs \leftarrow \text{ZK}.\text{Setup}(1^\lambda)$	$\pi \leftarrow \text{ZK}.\text{Prove}(crs,$	Output \perp	$\pi' \leftarrow ???$
$PK = (pk_1, pk_2, crs)$	$c_1 \ \& \ c_2 \text{ encrypt } m)$		
$SK = sk_1$	$CT = (c_1, c_2, \pi)$		

The problem: How to generate the proof for an *evaluated* ciphertext?

- The direct generation needs the randomness of the encryption as a witness. 😞😞😞
- We can prove that the two ciphertext components are evaluated from ciphertexts encrypting the same message using the same circuit. 😊
- But the proof size will usually be a polynomial in the circuit size. 😞😞😞
- We can use a SNARG to generate a succinct proof 😊😊😊
 - The SNARG can prove succinct proof whose size is independent of the statement/witness size
 - But a **general** SNARG relies on either **random oracle or non-falsifiable assumptions** 😞

The Main Gap: Succinct Proof for Correctness of Evaluated Ciphertexts from LWE

The problem: How to generate the succinct proof for the correctness of an *evaluated* ciphertext?

- Correctness of a ciphertext: The two ciphertext components are evaluated from ciphertexts encrypting the same message using the same circuit.

Some Known SNARG for Special Languages from LWE:

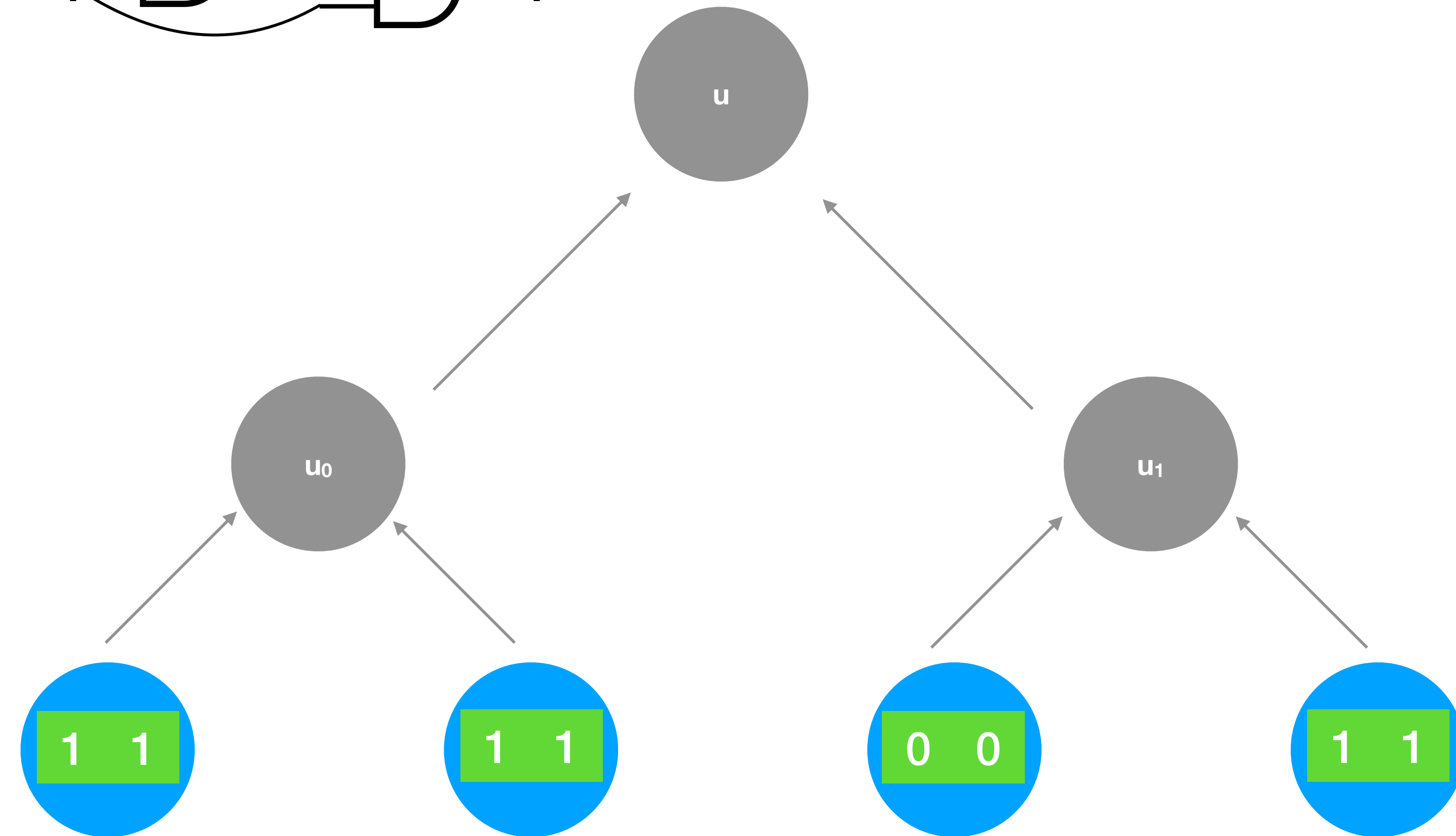
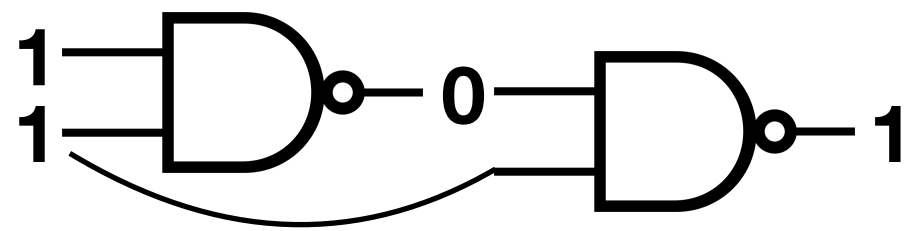
- Index Batch NP
- Language P
- Monotone Policy batch NP

However, they are not able to prove the correctness of the evaluated ciphertexts....

So, we need to design a new SNARG for the correctness of FHE ciphertexts.

- We will start with SNARG for index batch NP (A.K.A., Index BARG), because it is also the main building block for the remaining two special SNARGs.
 - It proves: $C(1, w_1) = 1 \wedge C(2, w_2) = 1 \wedge \dots \wedge C(n, w_n) = 1$
 - The proof size: $\log n \cdot |C| \leq \lambda \cdot |C|$

The First Attempt



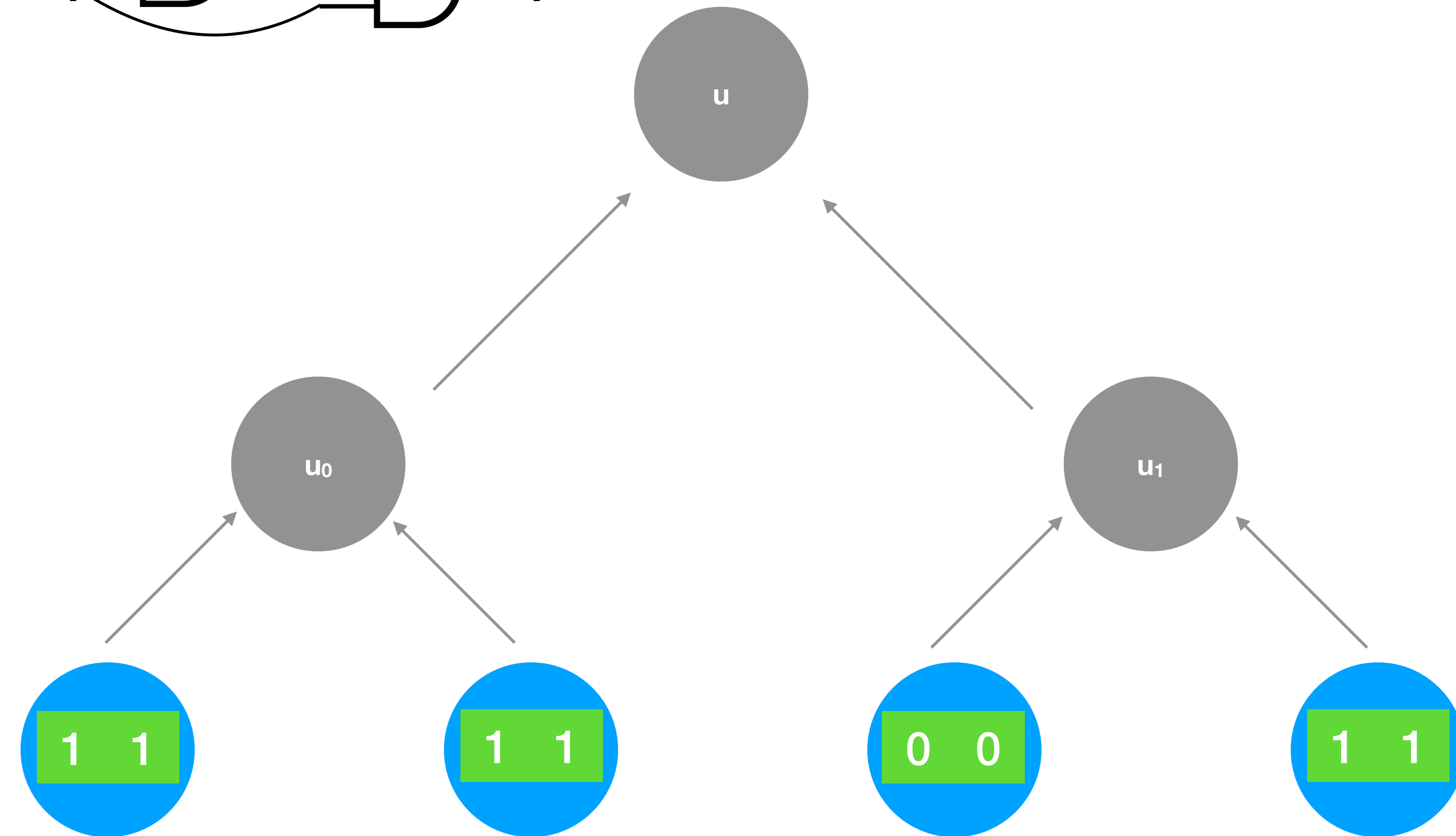
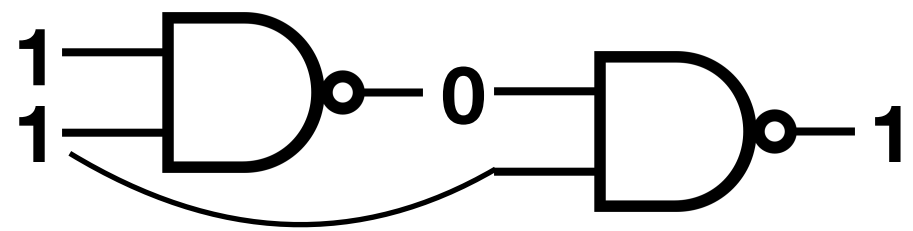
How to generate the proof?

1. Accumulate all ciphertexts in a Merkle-tree.
2. For each index i :
 1. If $i \leq 2$: Prove that
 - (1) the ciphertext has a valid ZK proof;
 - (2) the ciphertext is accumulated in the tree.
 2. If $i > 2$: Prove that
 - (1) it is generated from two *priori* ciphertexts.
 - (2) the ciphertext is accumulated in the tree.

We can generate succinct proof for step 2 using a BARG.

The construction is inspired by [CJJ21]

The First Attempt



How to generate the proof?

1. Accumulate all ciphertexts in a Merkle-tree.
2. For each index i :
 1. If $i \leq 2$: Prove that
 - (1) the ciphertext has a valid ZK proof;
 - (2) the ciphertext is accumulated in the tree.
 2. If $i > 2$: Prove that
 - (1) it is generated from two *priori* ciphertexts.
 - (2) the ciphertext is accumulated in the tree.

We can generate succinct proof for step 2 using a BARG.

The proof can prove the correctness of the final ciphertext if **all statements** are guaranteed to be correct.

However, ...

The Challenge

Security of the above construction relies on *global soundness*, i.e., **all statements** can be guaranteed to be correct.



The Challenge

Known Index BARG from LWE only has local soundness, i.e., only **one (hidden) statement** is guaranteed to be correct.



We need to know which statement to check

Security of the above construction relies on *global soundness*, i.e., **all statements** can be guaranteed to be correct.



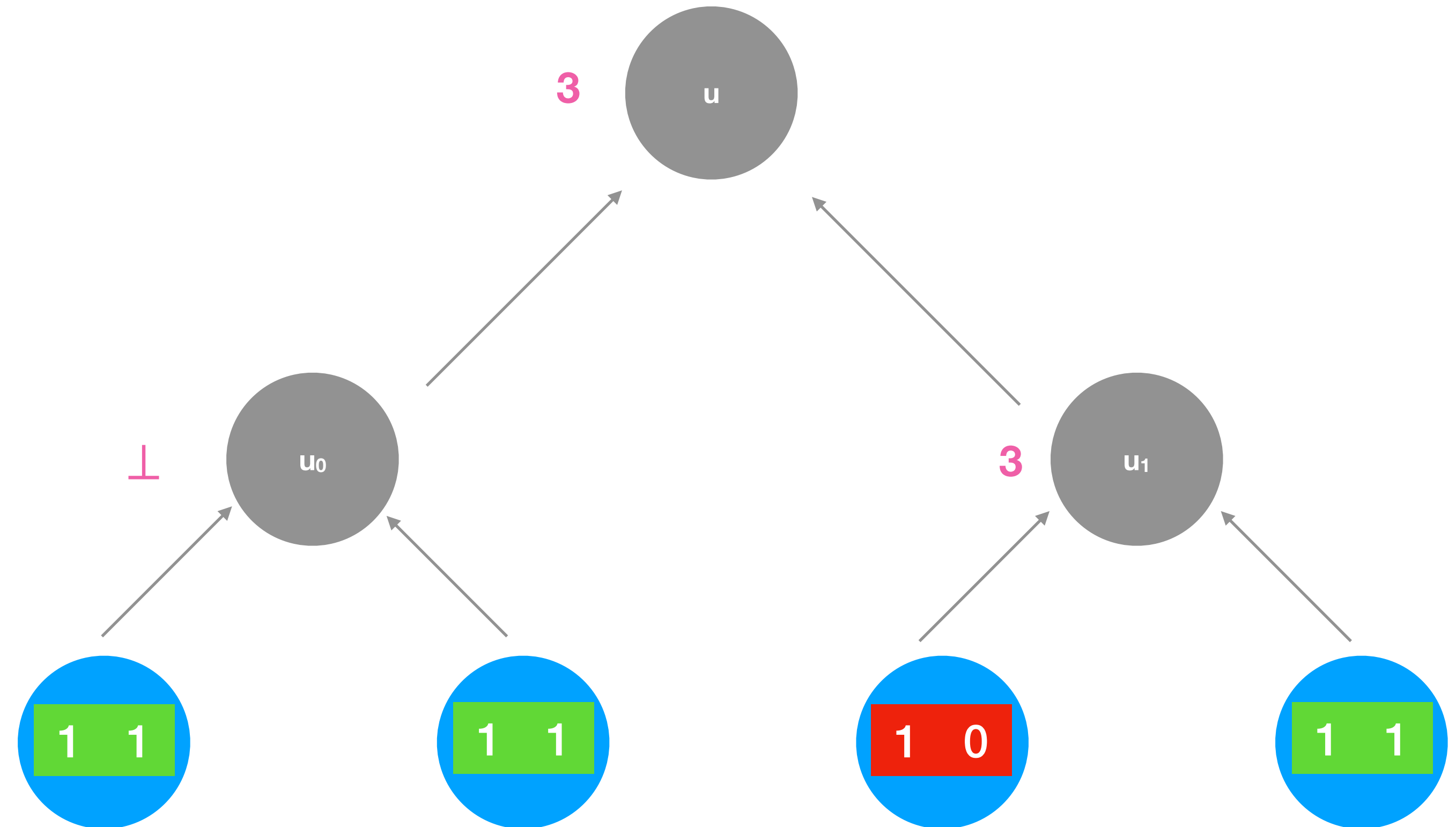
Our Solution

We need to know which statement to check

The Merkle tree checks if each ciphertext is valid and record the invalid ones. Then it copies (one of) the invalid ciphertext into its parent node.

The check needs both secret keys and we need to protect the secret key (and enabling the checking) by encrypting them using another FHE.

Then the challenger (in the proof) can use the secret key of the additional FHE as a trapdoor to know which ct is invalid.



Our Solution

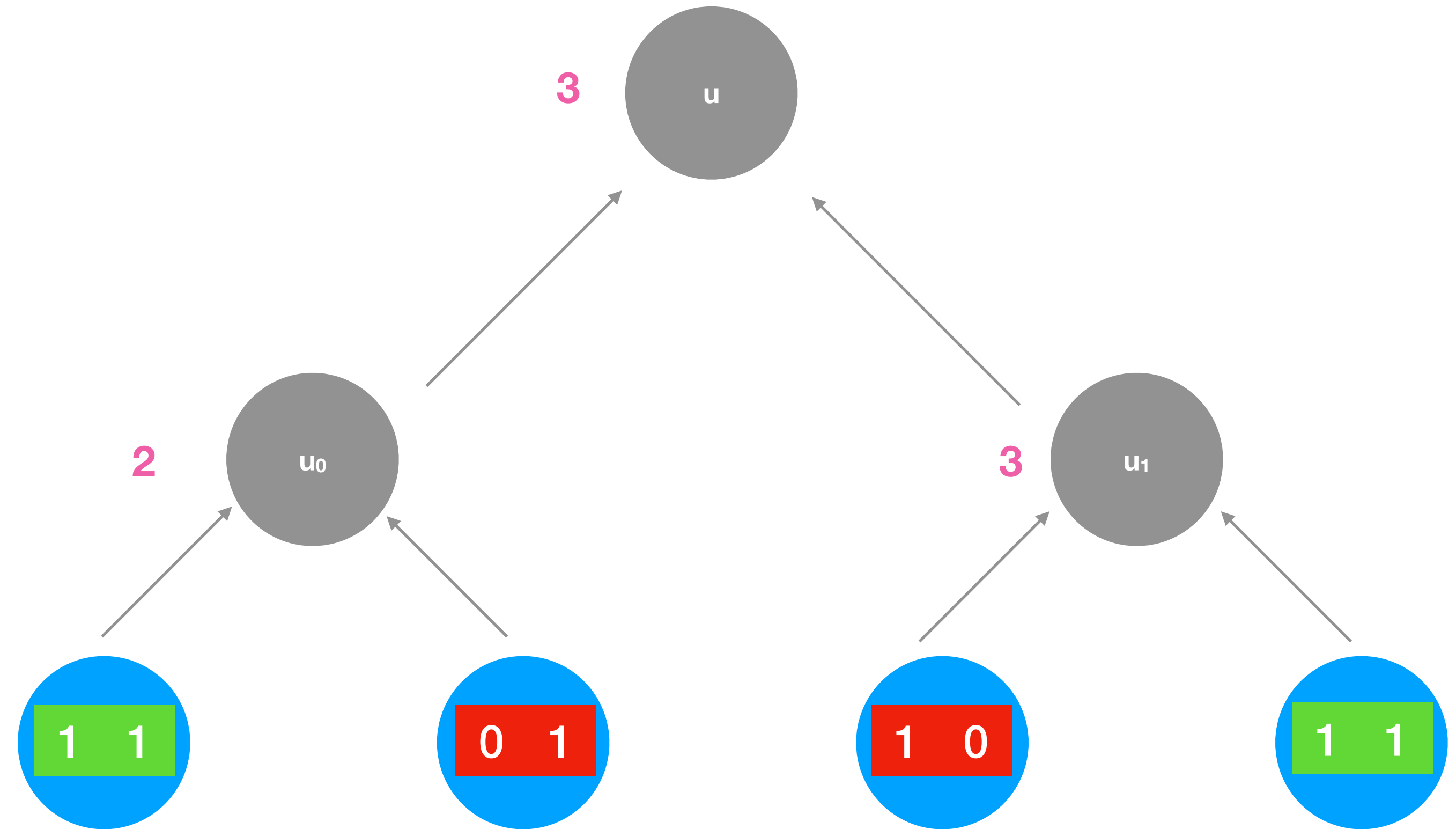
We need to know which statement to check

The current modification only checks if an invalid ct exists, but the invalid ct may not be led by an invalid statement at the current possible. **Maybe this is due to an invalid ciphertext at a priori position?**

We can use two Merkle Trees with check and guarantee that:

(1) either the current statement is wrong;
(2) or the wrong statement appears before.

This compress the space of possible wrong statement iteratively.



The construction is inspired by [BBK+ 23]

[BBK+23] Zvika Brakerski, Maya Farber Brodsky, Yael Tauman Kalai, Alex Lombardi, and Omer Paneth. SNARGs for monotone policy batch NP. In CRYPTO, pages 252–283. Springer, 2023.

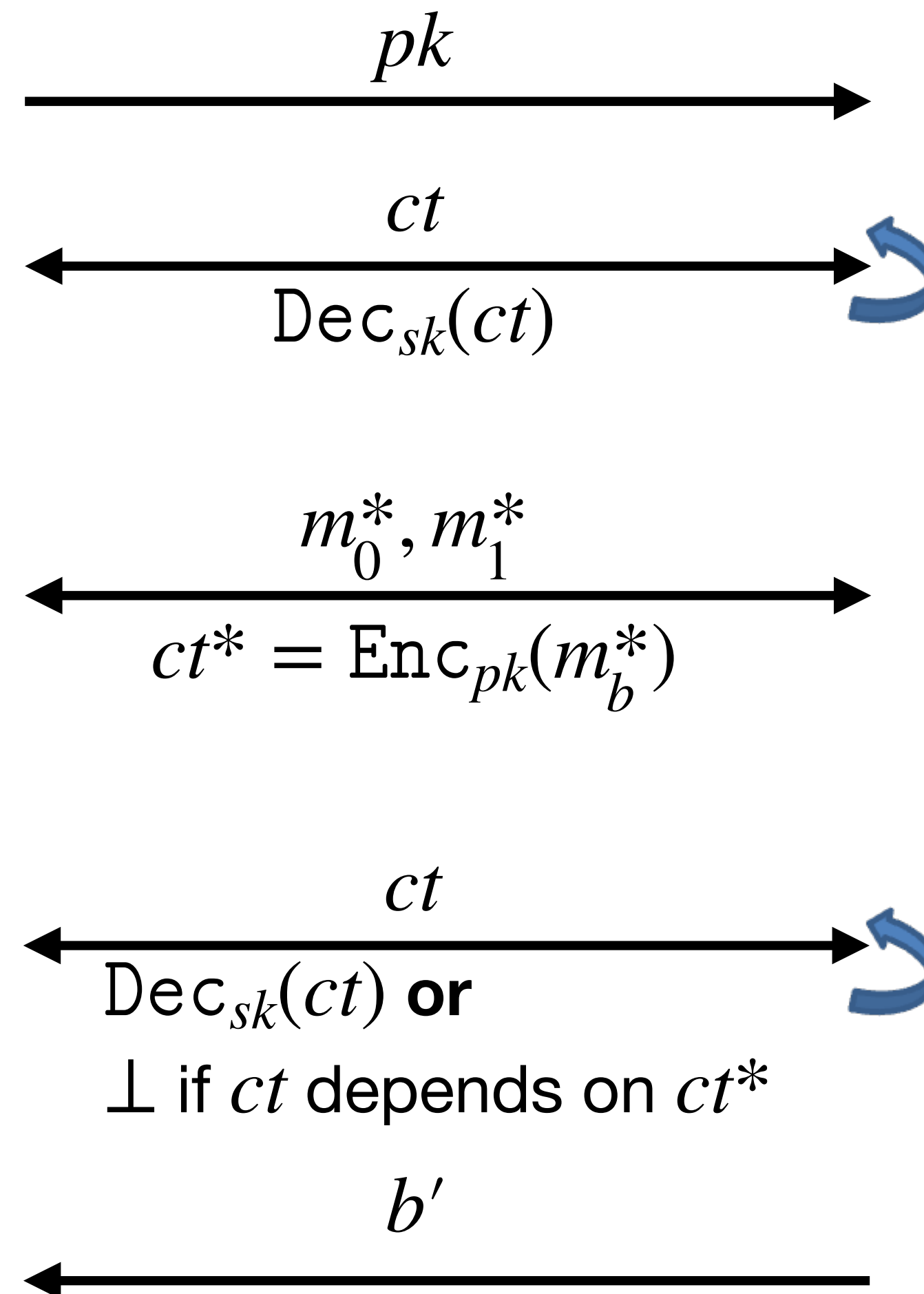
Our Results

- We construct FHE schemes with IND-CCA1 security from (circular-secure) LWE.
- **We define a new security notion called IV-CCA security, which lies between IND-CCA1 security and VCCA security.**
- **We construct FHE schemes with IV-CCA security from (circular-secure) LWE.**

VCCA Security of FHE



$(pk, sk) \leftarrow \text{KeyGen}(1^\lambda)$
 $b \leftarrow \{0,1\}$



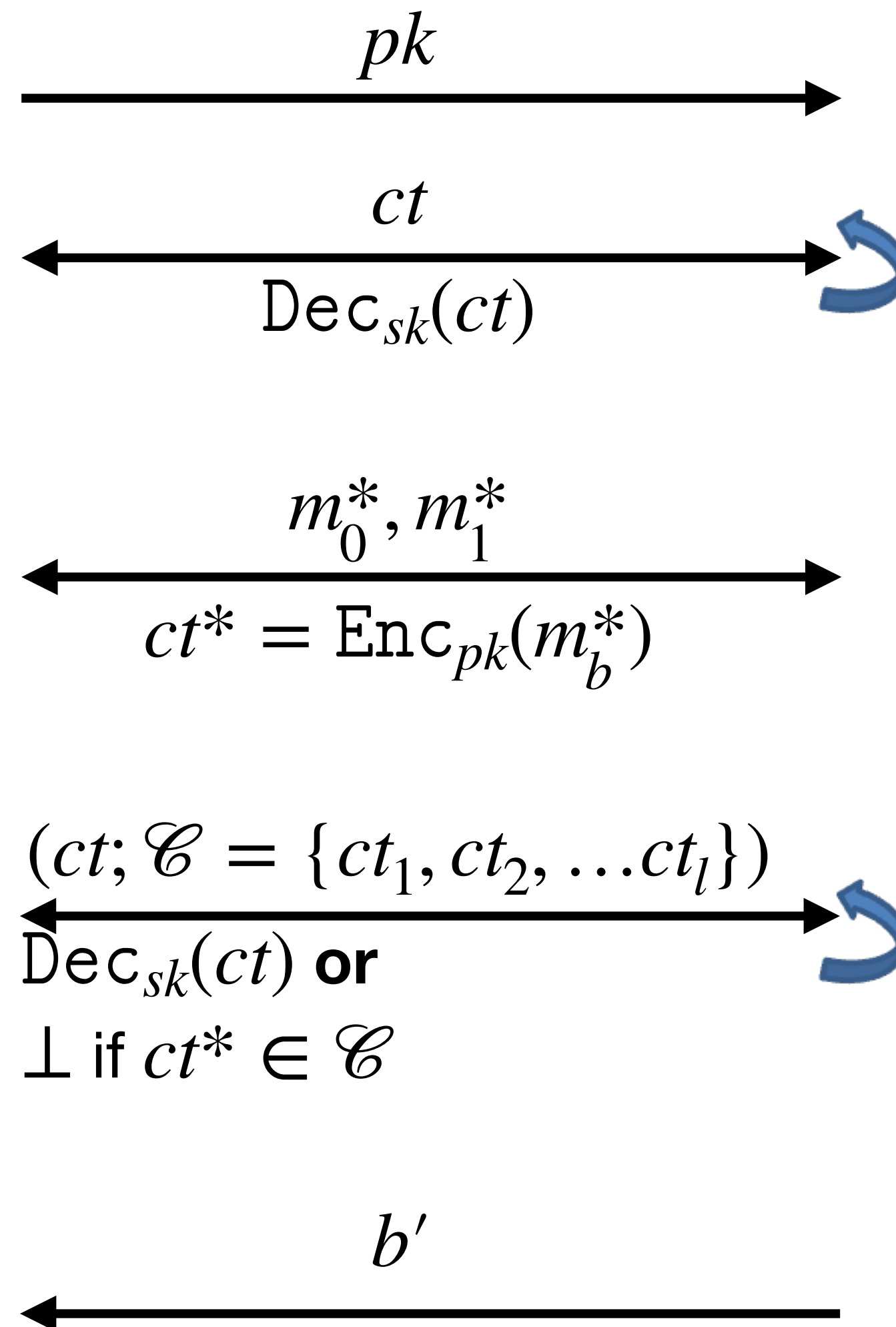
How to check if ct depends on ct^* ?

- Definition: Assume that is possible
- Construction: Extract all details of the computation that outputs ct ; that is possible if the computation is proved by a **ZK-SNARK**.

IV-CCA Security of FHE



$(pk, sk) \leftarrow \text{KeyGen}(1^\lambda)$
 $b \leftarrow \{0,1\}$



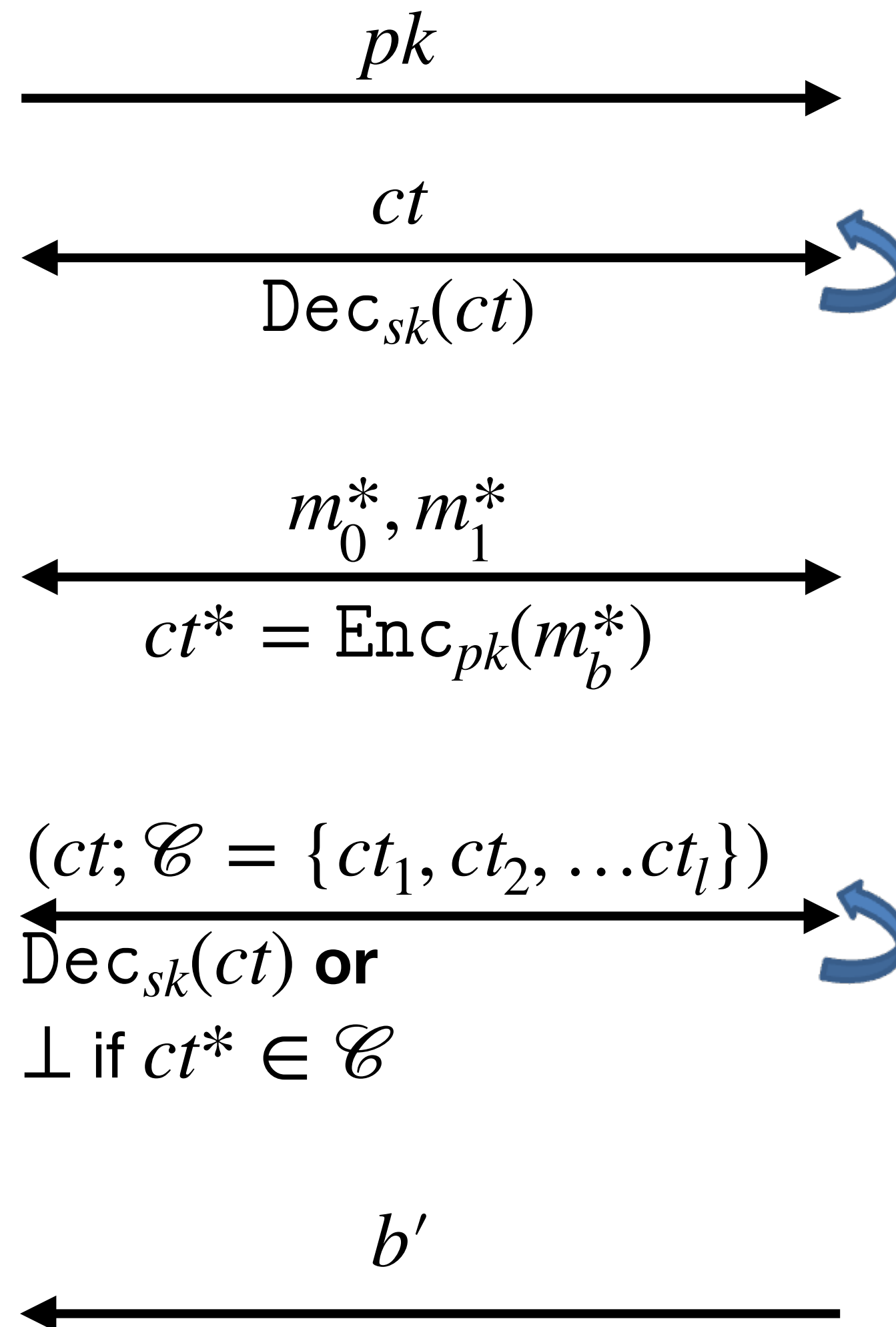
This guarantees that ct is generated from ciphertexts excluding ct^* .

The challenger also needs to ensure that ct is generated from \mathcal{C} . But how to check if ct is generated from \mathcal{C} .

IV-CCA Security of FHE



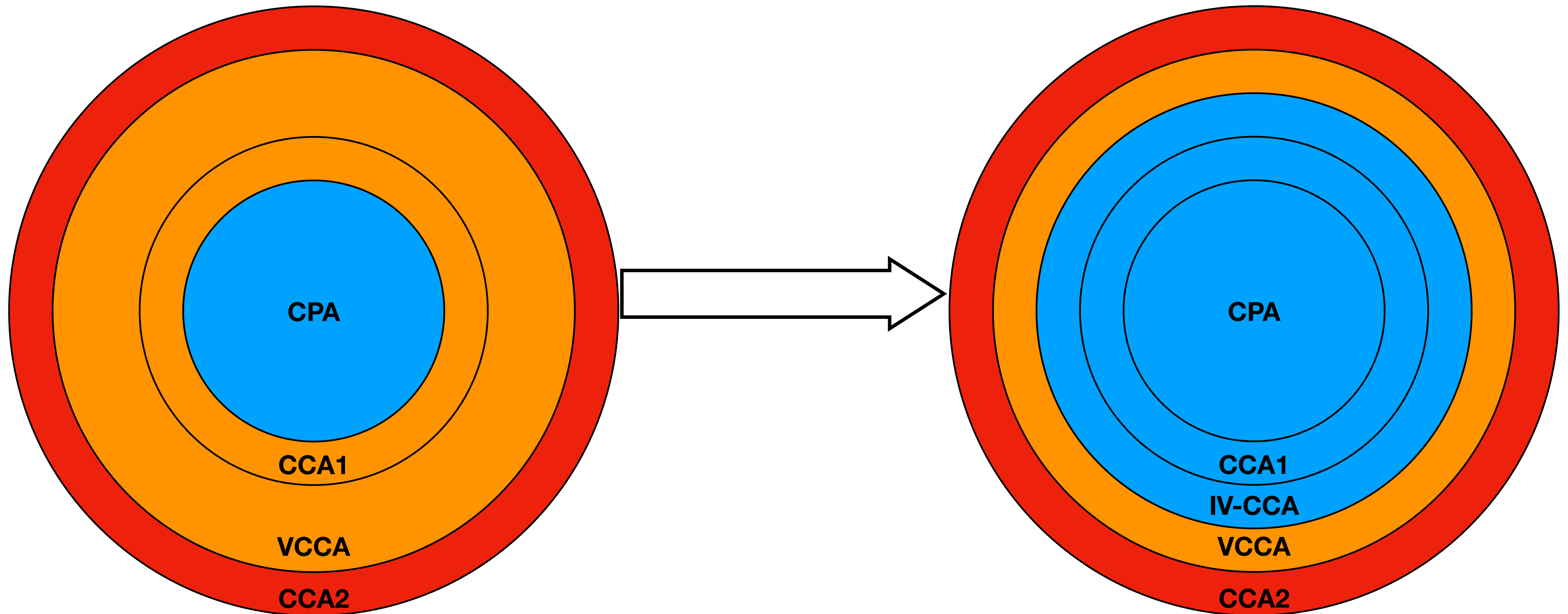
$(pk, sk) \leftarrow \text{KeyGen}(1^\lambda)$
 $b \leftarrow \{0,1\}$



But how to check if ct is generated from \mathcal{C} .

1. Add an additional Merkle tree for *all input ciphertexts*, and checks input ciphertexts in the new tree in the index BARG.
2. Then reconstruct the Merkle tree from \mathcal{C} and compare.

Our Results



Legend:

- = known from standard assumption
- = known from non-standard assumption
- = Impossible

Thanks for your Attention!