From Naor–Yung to CCA-secure PRCs

A simple compiler for CCA2-PRC in the standard model

Nico Döttling · Antoine Joux · Venkata Koppula
Mahesh Sreekumar Rajasree · Hendrik Waldner

University of Warsaw · November 2026

AI content has a provenance problem

Generate

Text · images · audio · video

Share

Content leaves the model

Edit

Noise and deliberate changes

Can a hidden watermark survive edits and still be detected?

Pseudorandomness meets error correction

Error correction
Recover from bounded noise

+

Pseudorandomness
Codewords look uniform





Public-key pseudorandom code
Anyone encodes. A secret key recovers the message.

Christ–Gunn · CRYPTO 2024

Public-key PRC: three algorithms

SETUP

$\mathrm{Setup}(1^\lambda)$

↓

$(\mathrm{pk}, \mathrm{sk})$

ENCODE

$\mathrm{Encode}(\mathrm{pk}, m; r)$

↓

$c \in \Sigma^n$

DECODE

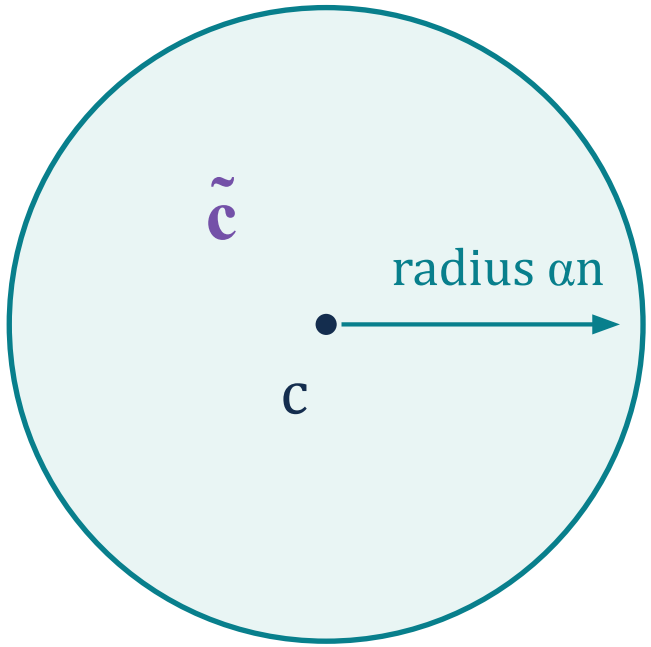
$\mathrm{Decode}(\mathrm{sk}, \tilde{c})$

↓

m or $\perp$

Everyone can encode. Only the secret-key holder decodes.

Robust decoding is a Hamming ball



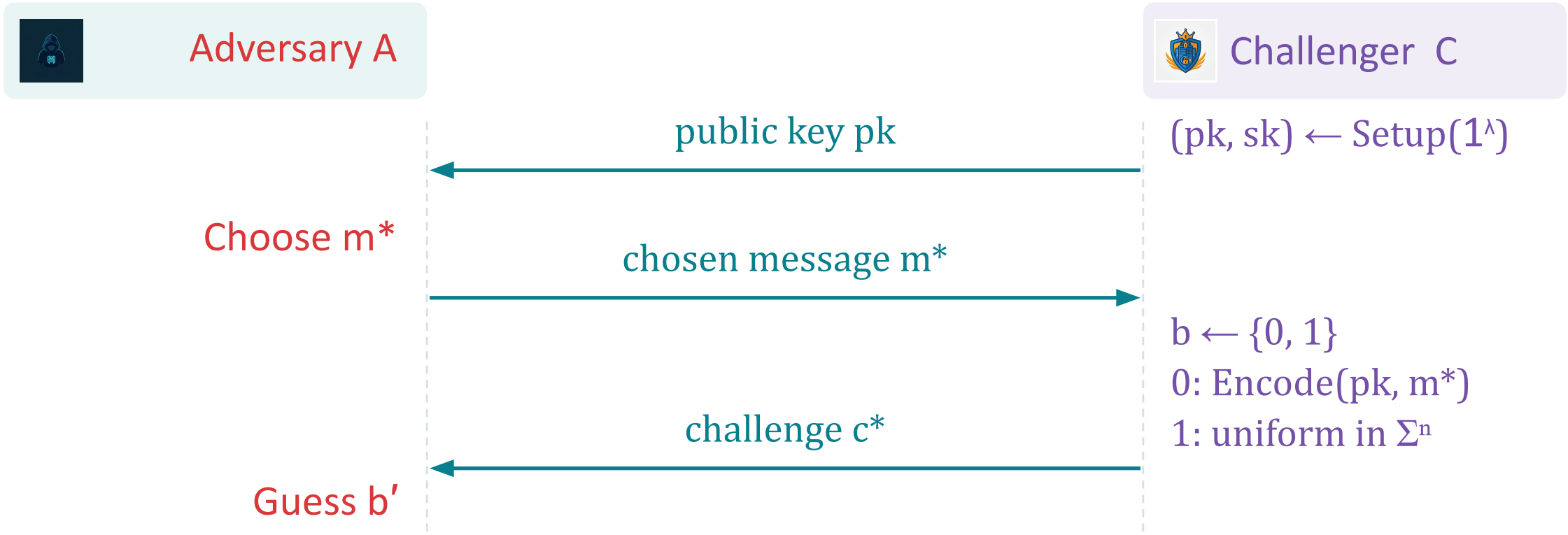
$$c = \mathrm{Encode}(pk, m; r)$$

$$\mathrm{Ham}(c, \tilde{c}) \leq \alpha n$$

$$\mathrm{Decode}(sk, \tilde{c}) = m$$

Bounded corruption preserves the encoded message.

CPA pseudorandomness: real or random?



No efficient adversary distinguishes an encoding from uniform.

Adaptive robustness allows chosen coins

Reveal pk

The adversary sees the public key.

Choose the attack

Output $(m, r, \tilde{c})$.
Set $c = \mathrm{Encode}(pk, m; r)$.

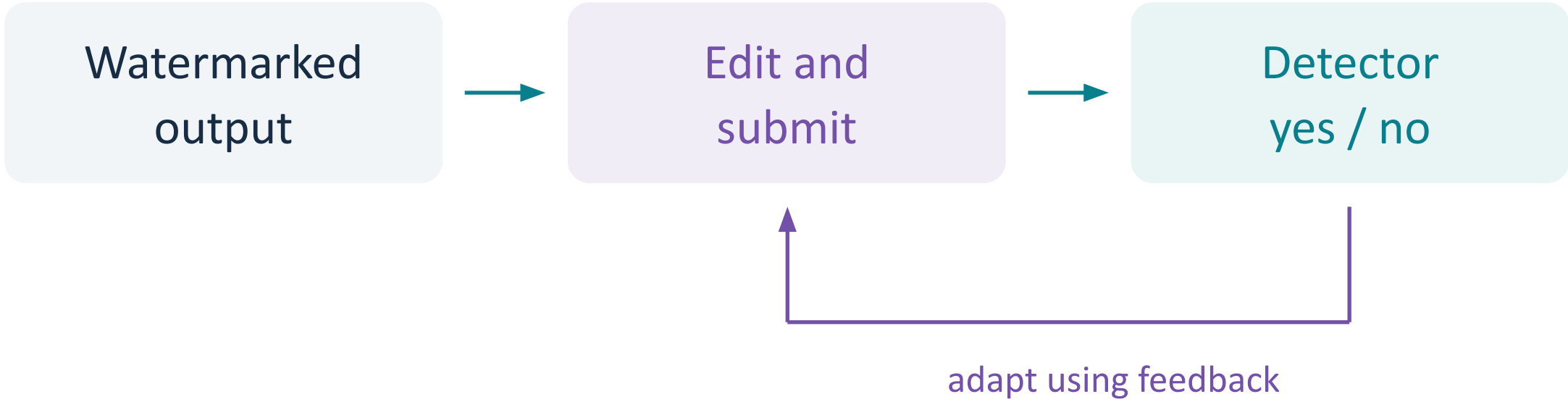
Test decoding

The adversary wins if both conditions hold:

$$\mathrm{Ham}(c, \tilde{c}) \leq \alpha n \quad \text{and} \quad \mathrm{Decode}(sk, \tilde{c}) \neq m$$

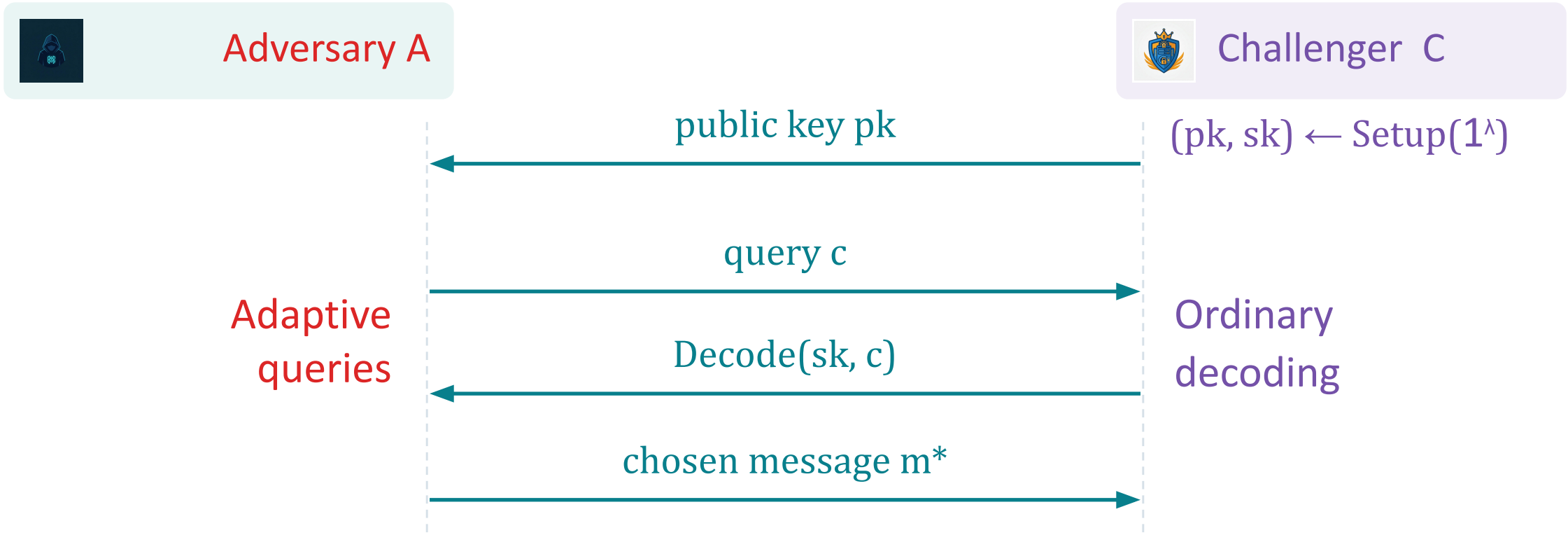
Adaptive robust if any adversary succeeds negligibly.

Detector access creates an adaptive attack



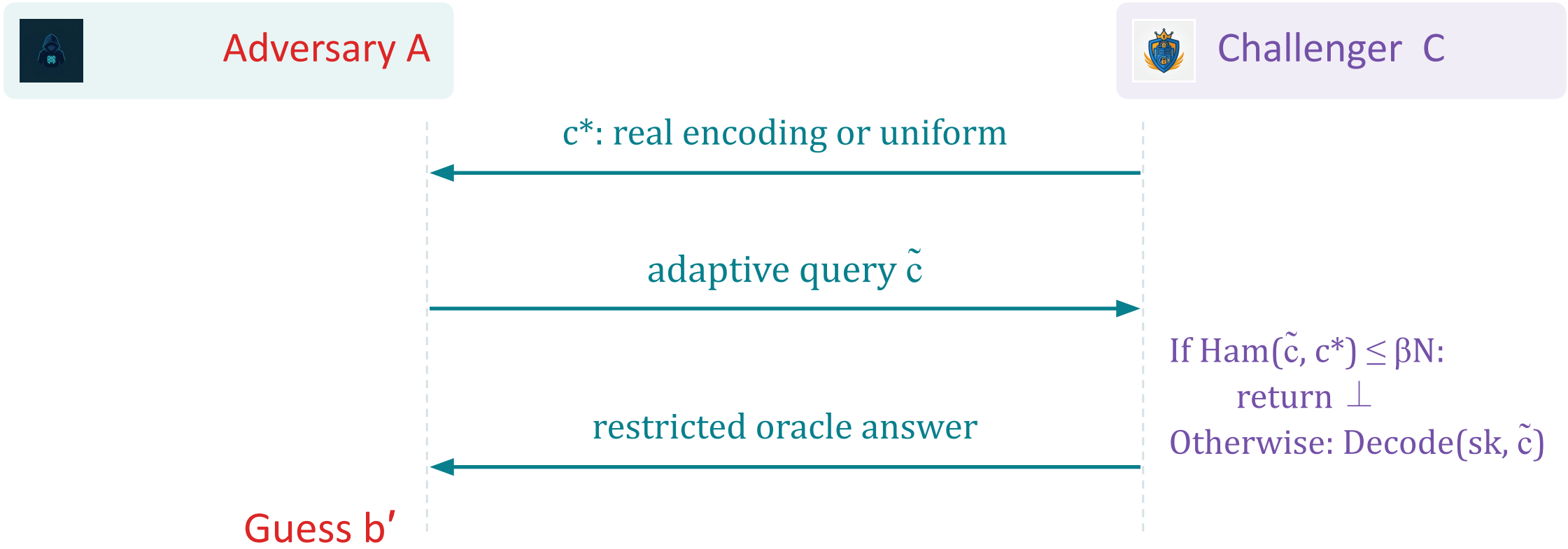
CPA pseudorandomness alone gives no decoding-oracle guarantee.

CCA2 game: before the challenge



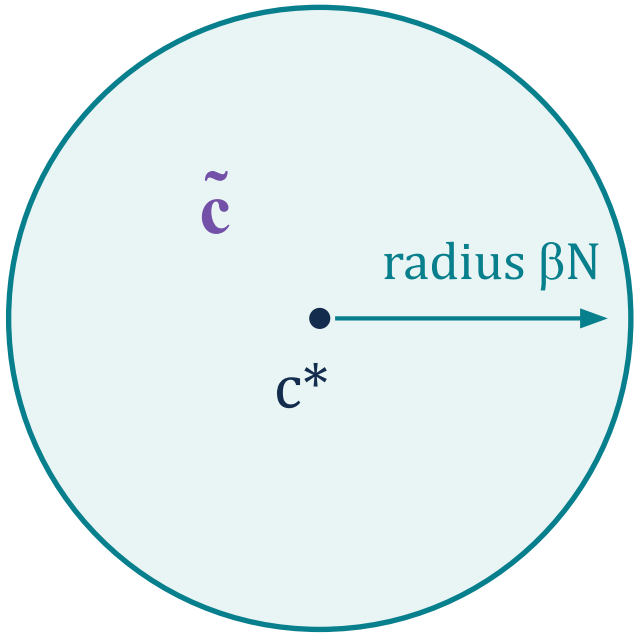
Before the challenge, the adversary may query the decoder freely.

CCA2 game: after the challenge



The post-challenge oracle excludes the entire ball B(c*, βN).

Why exclude a whole Hamming ball?



Flip one symbol of c*.

A robust decoder must still recover the same message.

Rejecting only c* leaves this attack open.

For PRCs, the CCA exclusion is a region, not a single point.

CCA-secure PRCs: the landscape

CONSTRUCTION	SECURITY	MODEL	COMPILER
Ideal PRCs · 2025	CCA2	Random oracle	FO-style
Döttling et al. · 2026	CCA2	Standard	Hinting PRG

Can we simplify the standard-model compiler?

This work	**CCA2**	**Standard**	**CCA-PKE + PRG**

The earlier compiler has many moving parts

Hinting PRG

Equivocal commitments

One-time signatures

PRF + public-key encryption

Auxiliary error correction

Our target: a simpler structure with only a factor-two loss in radius.

The compiler theorem

INPUTS

1. CPA-pseudorandom, adaptively α-robust PRC
2. Almost perfectly correct IND-CCA2 PKE
3. Secure PRG

Adaptively α/2-robust and α/2-CCA2-pseudorandom
Codeword length: 2n · Standard model · Black-box

Two classical routes to CCA security

FUJISAKI–OKAMOTO

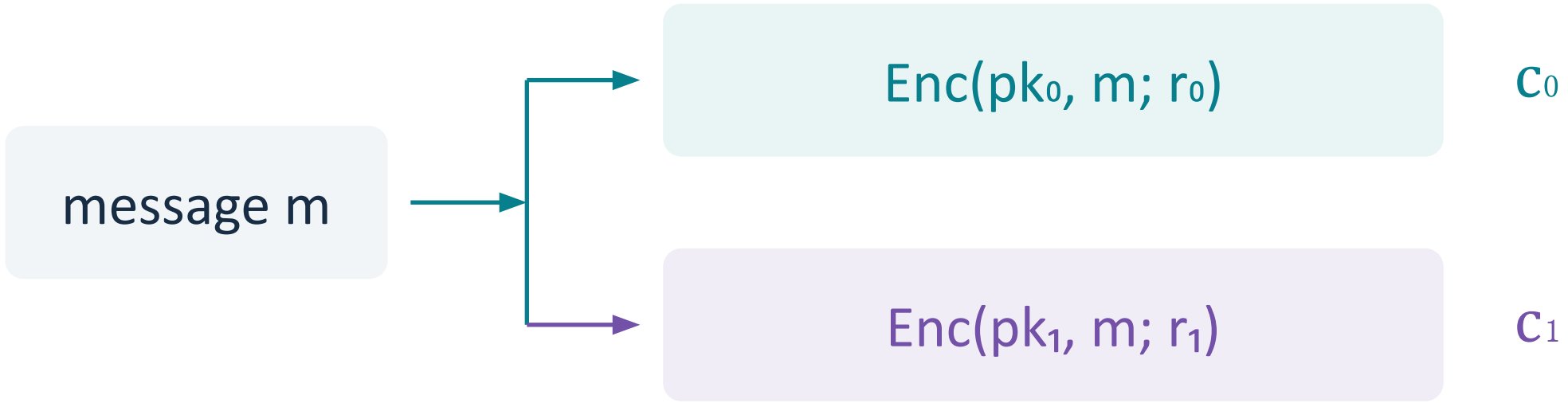
Re-encryption check
Random oracle model

NAOR–YUNG

Two encryptions + proof
Standard model

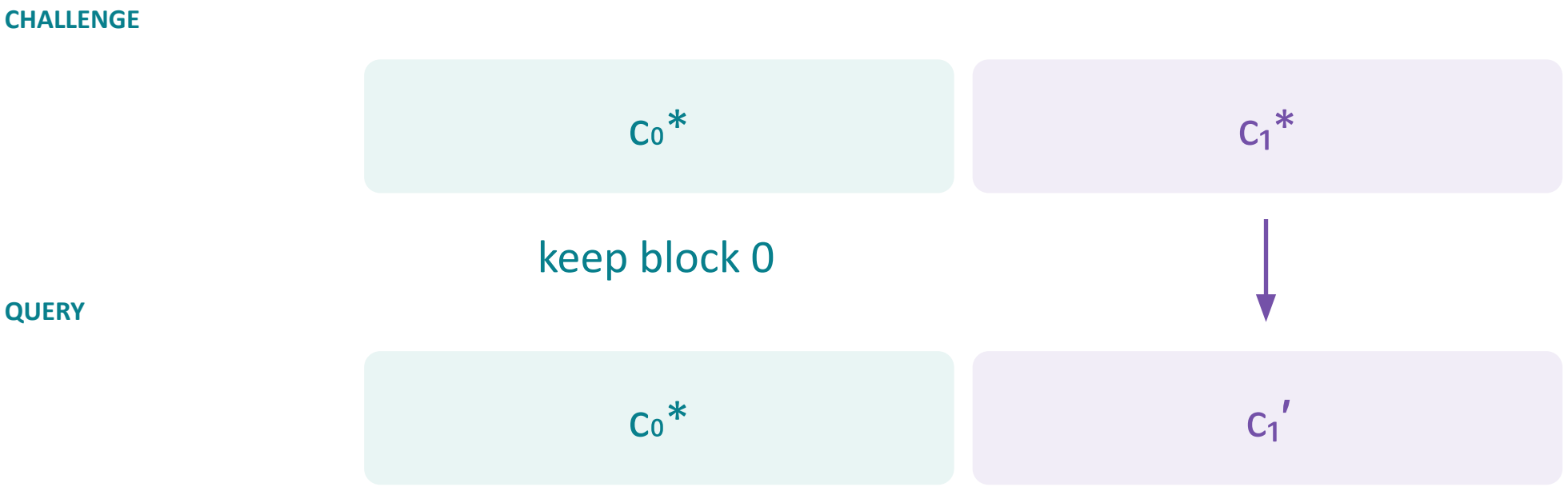
The two-key structure is the starting point for our story.

Naor–Yung: encrypt twice



Two independent keys create two possible decryption paths.

Double encryption alone is malleable



A decoder that trusts only block 0 reveals the challenge message.

Naor–Yung adds a consistency proof

$c_0 = \mathrm{Enc}(pk_0, m; r_0)$

$c_1 = \mathrm{Enc}(pk_1, m; r_1)$

π

Both ciphertexts encrypt the same message.

The proof binds the two decryption paths together.

The proof must survive simulation

Simulate the challenge proof π^*

→

Can the adversary prove a new false statement?

Simulation soundness rules out new false statements after simulation.

The Naor–Yung proof switches paths

Hybrid 0 (Real)

c_0: Enc(pk_0, m_0)

c_1: Enc(pk_1, m_0)

Proof π: Real NIZK

Decryption key:

sk_0

Hybrid 1 (Sim Proof)

c_0: Enc(pk_0, m_0)

c_1: Enc(pk_1, m_0)

Proof π*: Simulated

Decryption key:

sk_0

Hybrid 2 (Change c_1)

c_0: Enc(pk_0, m_0)

c_1: Enc(pk_1, m_1)

Proof π*: Simulated

Decryption key:

sk_0

Hybrid 3 (Switch Key)

c_0: Enc(pk_0, m_0)

c_1: Enc(pk_1, m_1)

Proof π*: Simulated

Decryption key:

sk_1

Hybrid 4 (Change c_0)

c_0: Enc(pk_0, m_1)

c_1: Enc(pk_1, m_1)

Proof π*: Simulated

Decryption key:

sk_1

Finally, revert simulation back to real proof → Reaches original NY ciphertext encrypting m_1.

Our target is CCA-secure error correction

NY works even if we replace NIZK with CCA scheme.

CLASSICAL TARGET

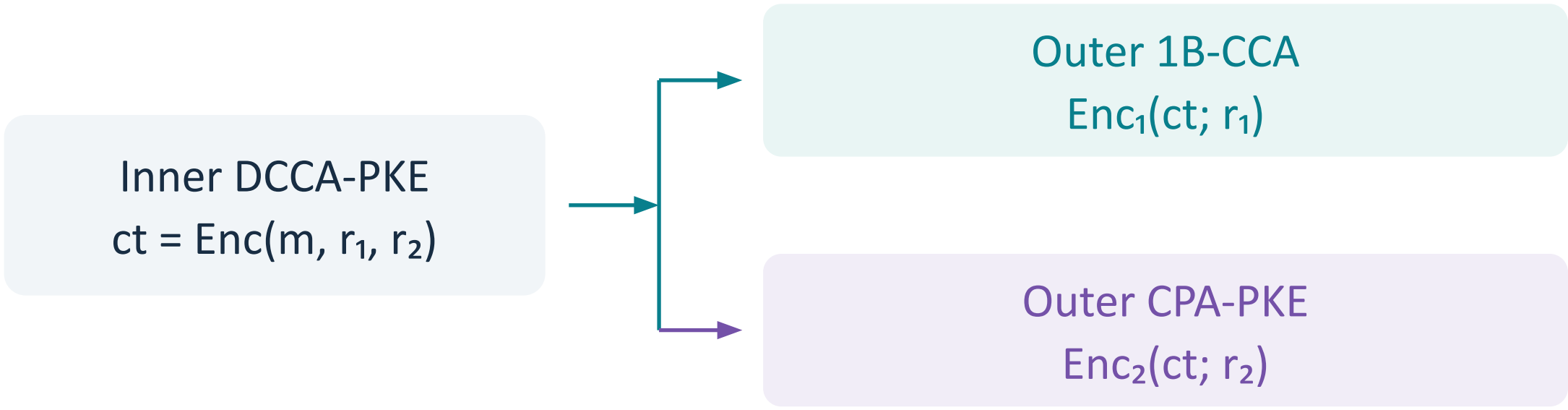
CPA-PKE ⟶ CCA-PKE
using a consistency proof

OUR TARGET

CPA-PRC ⟶ CCA-PRC
using CCA-PKE inside

CCA-PKE is an input primitive to this compiler.

HLW: 1 Bit CCA is complete



Hohenberger–Lewko–Waters · EUROCRYPT 2012

The inner plaintext carries the randomness for both outer layers.

Recover the witness, then verify both blocks

Decrypt

Recover the inner ct.

→

Recover

Read m, r_1, r_2.

→

Re-encrypt

Check both outer blocks.

Encrypted randomness enables a private consistency check.

For PRCs, consistency must tolerate noise

HLW
Outer ciphertexts
Exact equality check

Our adaptation
Outer PRC codewords
Hamming-distance check

Encode the same inner CCA-PKE ciphertext under two PRC keys.

Setup algorithm

GENERATE

$$(pk_0, sk_0), (pk_1, sk_1) \leftarrow PRC.KeyGen(),$$
$$(pk_{CCA}, sk_{CCA}) \leftarrow PKE.KeyGen()$$

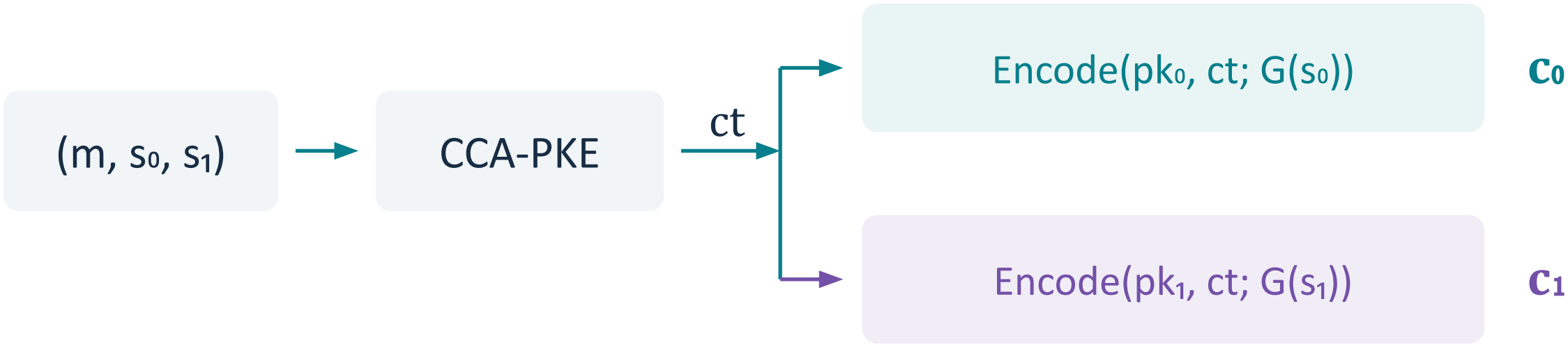
PUBLIC KEY

$$pk := (pk_{CCA}, pk_0, pk_1)$$

SECRET KEY

$$sk := (sk_{CCA}, sk_0)$$

Encoding: one ciphertext, two codewords



Transmit $c = (c_0, c_1) \in \Sigma^{2n}$.

The encoder in three steps

SAMPLE

$$s_0, s_1 \leftarrow \{0,1\}^{\lambda}$$

ENCRYPT

$$ct \leftarrow \mathrm{PKE.Enc}(pk_{CCA}, (m, s_0, s_1))$$

ENCODE

$$c_b := \mathrm{Encode}(pk_b, ct; G(s_b)),\ b \in \{0,1\}$$

All sampling is independent; PKE.Enc uses fresh encryption coins.

Decode one block, then recover the seeds

RECEIVED WORD

$\tilde{c} = (\tilde{c}_0, \tilde{c}_1)$

DECODE BLOCK 0

$\hat{ct} = \mathrm{Decode}(sk_0, \tilde{c}_0)$

DECRYPT $\hat{CT}$

Recover (m, s_0, s_1)

$$\text{Reconstruct } \hat{c}_b := \mathrm{Encode}(pk_b, \hat{ct}; G(s_b)) \text{ for } b \in \{0, 1\}.$$

Reject immediately if decoding, decryption, or parsing fails.

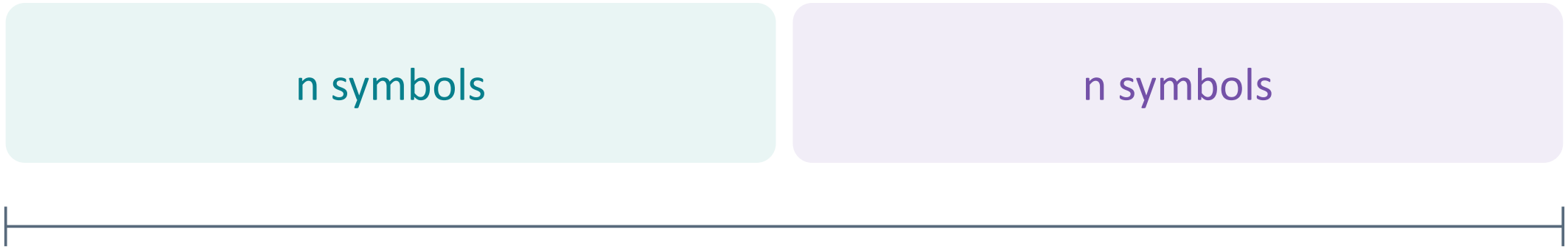
The acceptance test is global



$$\text{Accept m iff } \mathrm{Ham}(\hat{c}, \tilde{c}) \le t, \quad t = \lfloor \alpha n \rfloor$$

One block supplies c_t. Both blocks must pass the consistency check.

Twice the length, half the relative radius



$$N = 2n$$

$$t = \lfloor \alpha n \rfloor = \lfloor (\alpha/2) N \rfloor$$

The second block supplies a proof path; the error budget remains $\lfloor \alpha n \rfloor$.

Robustness: project the error budget

$$\mathrm{Ham}(c, \tilde{c}) \le \lfloor \alpha n \rfloor \Rightarrow \mathrm{Ham}(c_0, \tilde{c}_0) \le \lfloor \alpha n \rfloor$$

Base robustness

Recover the inner ct.

PKE correctness

Recover m and both seeds.

Global check

Reconstruction equals c.

If the compiled decoder fails, the first base PRC violates robustness.

CCA security uses two decoding paths

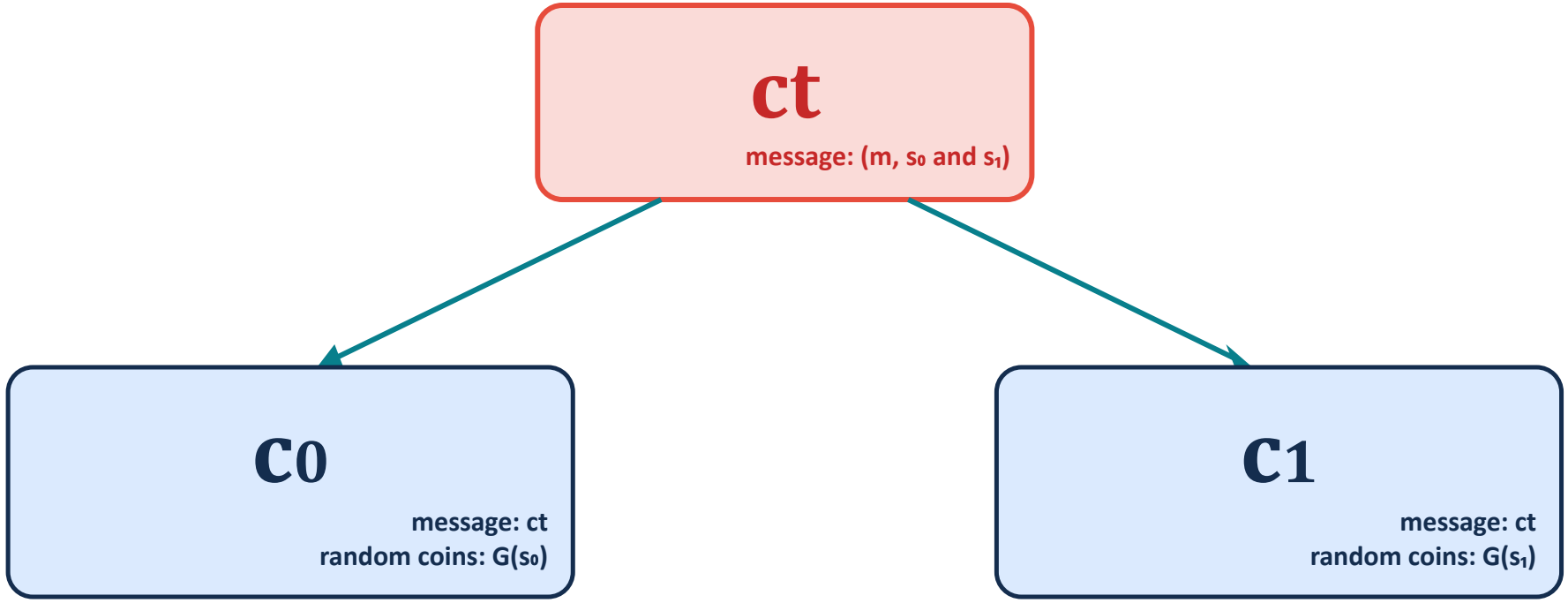
Decode via block 0
Hide challenge block 1



Decode via block 1
Hide challenge block 0

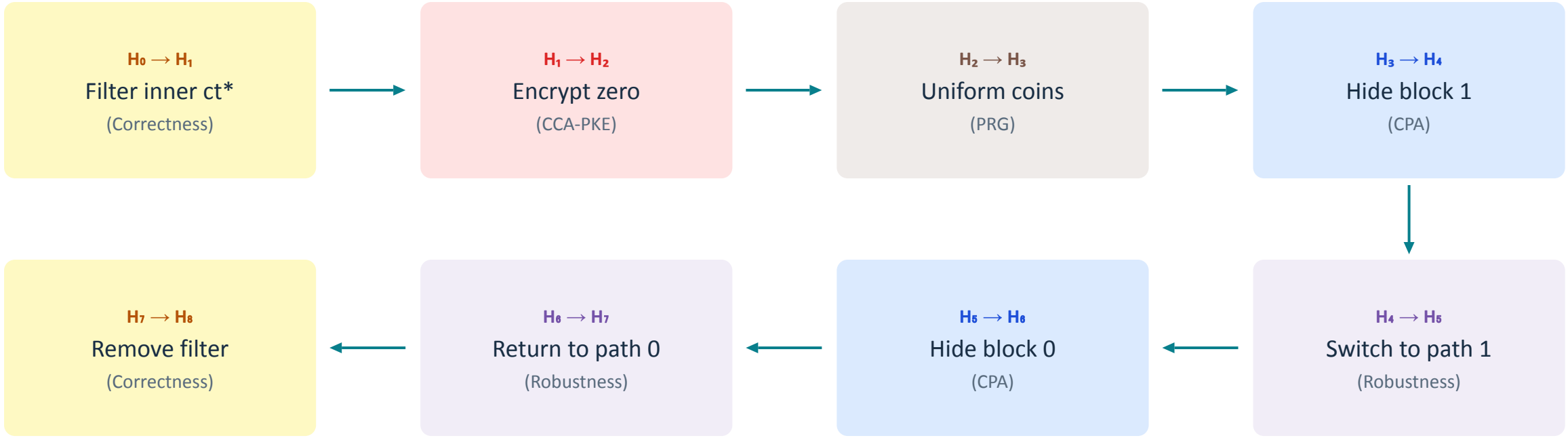
Adaptive robustness makes the two checked decoding paths agree.

The encoder in three steps



All sampling is independent; PKE.Enc uses fresh encryption coins.

Proof roadmap: complete sequence ($H_0 \to H_8$)



H_8 is exactly the uniform-challenge experiment.

$H_0 \longrightarrow H_1$: filter the inner challenge ct*

Unfiltered
Decode_0



Decode_0
Reject inner ct*

Recover ct* + pass the global check
$\Rightarrow$ query lies inside the excluded challenge ball.

On good PKE keys, the filter changes no oracle answer.

$H_1 \longrightarrow H_2$: replace the inner payload

$\mathrm{PKE.Enc}(pk_{CCA},(m^*, s_0^*, s_1^*))$ ⟶ $\mathrm{PKE.Enc}(pk_{CCA}, 0^{\ell})$

The internal filter prevents any decryption query on ct*.

IND-CCA2 security of the inner PKE justifies this transition.

$H_2 \longrightarrow H_3$: replace the challenge coins

$G(s_0^*), G(s_1^*)$ → r_0^*, r_1^*

Independent uniform coins

The inner challenge now encrypts zero, so its payload no longer carries the seeds.

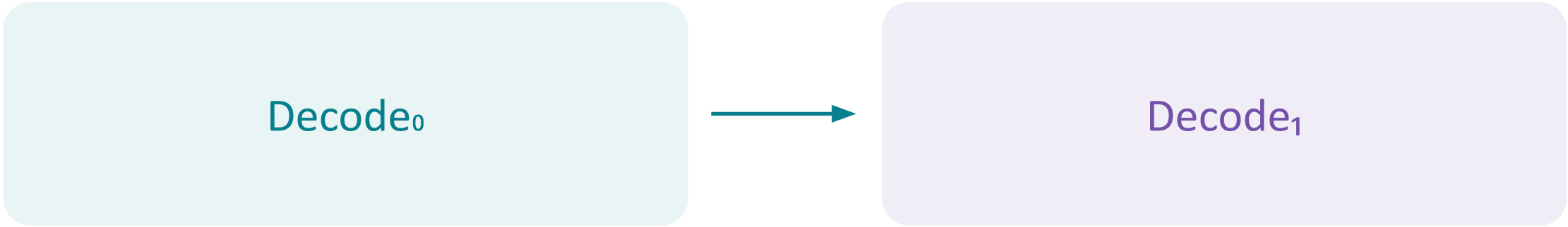
Only the challenge coins change. The decoder still reconstructs using G.

$H_3 \longrightarrow H_4$: hide the second block



The reduction answers decoder queries using sk_0; it never needs sk_1.

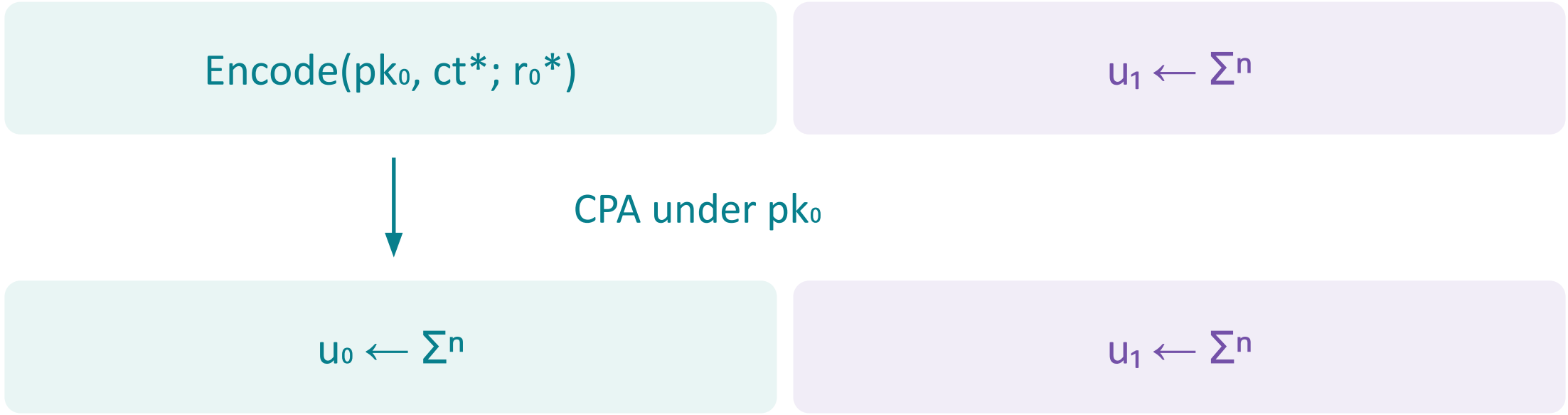
$H_4 \longrightarrow H_5$: switch the decoding path



Acceptance reconstructs a valid pair within distance t.
Each received block is therefore within αn of its reconstruction.

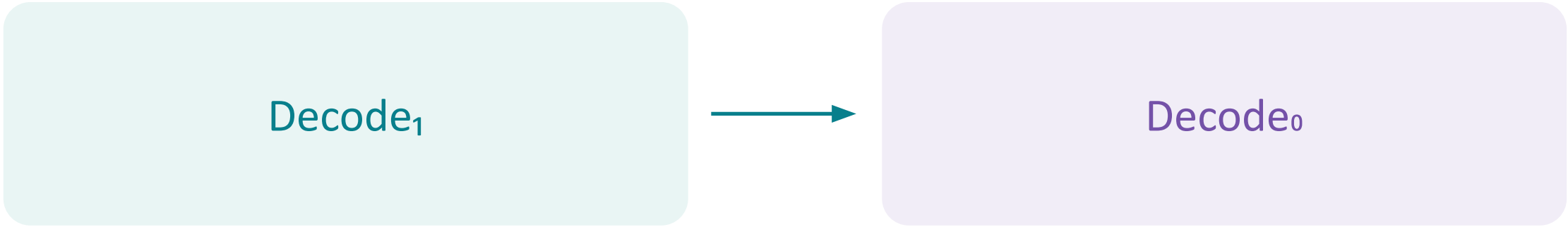
A disagreement yields an adaptive-robustness attack on a base PRC.

$H_5 \longrightarrow H_6$: hide the first block



Now the reduction answers through sk_1 and does not need sk_0.

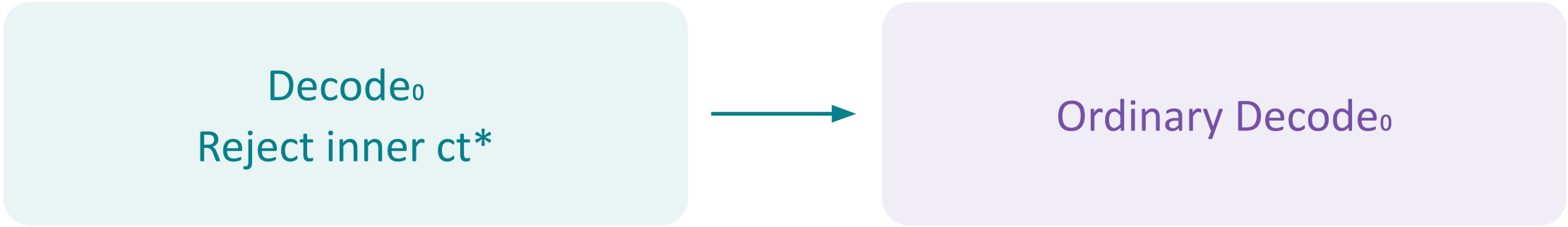
$H_6 \longrightarrow H_7$: restore the actual decoder



Both challenge blocks remain independent and uniform.

Apply the same path-agreement argument from adaptive robustness.

$H_7 \longrightarrow H_8$: remove the hidden filter



The latent ct* is a fresh encryption of zero, independent of the now-uniform challenge.

The only possible difference is a query whose inner decoding hits ct*.

A fresh hidden ciphertext is hard to hit

Bad: some decoder query recovers the latent ct*

$$\Pr[\text{Bad}] \leq q \cdot \mathrm{E}_{pk}\left[\max_{z} \Pr_{r}[\mathrm{Enc}(pk, 0; r) = z]\right]$$

$$= \mathbf{negl}(\lambda)$$

Fresh-ciphertext unpredictability follows from IND-CPA and correctness.

What the simplification buys

BLACK-BOX

Base CPA-PRC

CCA-PKE

PRG

STANDARD MODEL

No random oracle

No NIZK in the compiler

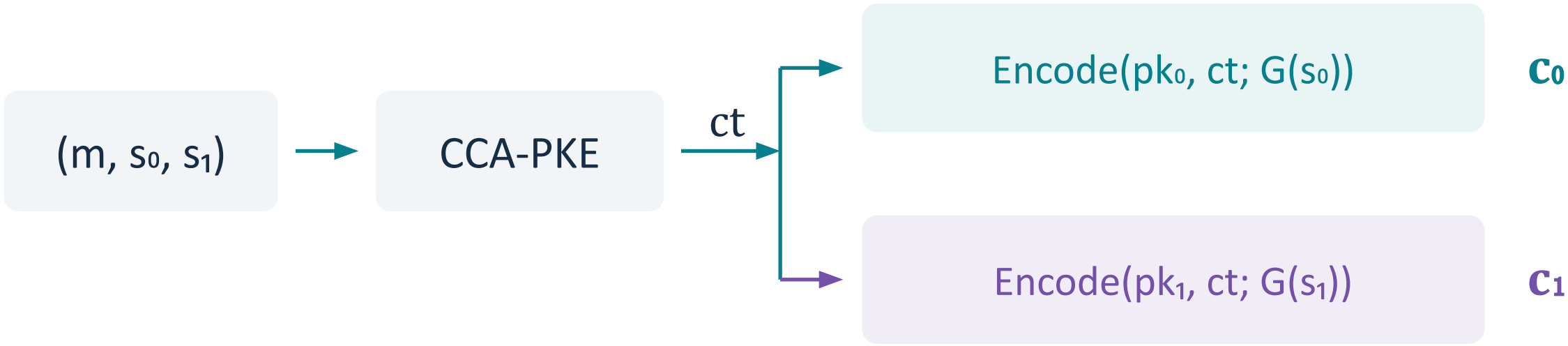
RADIUS A/2

Two codeword blocks

Constant radius retained

A simpler construction and proof; concrete performance remains instantiation-dependent.

The one picture to remember



Recover through one path. Re-encode and check both.

References: PRCs and CCA security

Christ–Gunn · CRYPTO 2024

Pseudorandom Error-Correcting Codes

Alrabiah et al. · STOC 2025

Ideal Pseudorandom Codes

Döttling et al. · CRYPTO 2026

Chosen Ciphertext Secure PRCs in the Standard Model

This manuscript

A Simple Compiler for CCA2-Secure Pseudorandom Codes in the Standard Model

References: the compiler lineage

Naor–Yung · STOC 1990

Public-key cryptosystems provably secure against chosen-ciphertext attacks

Fujisaki–Okamoto · CRYPTO 1999

Secure Integration of Asymmetric and Symmetric Encryption Schemes

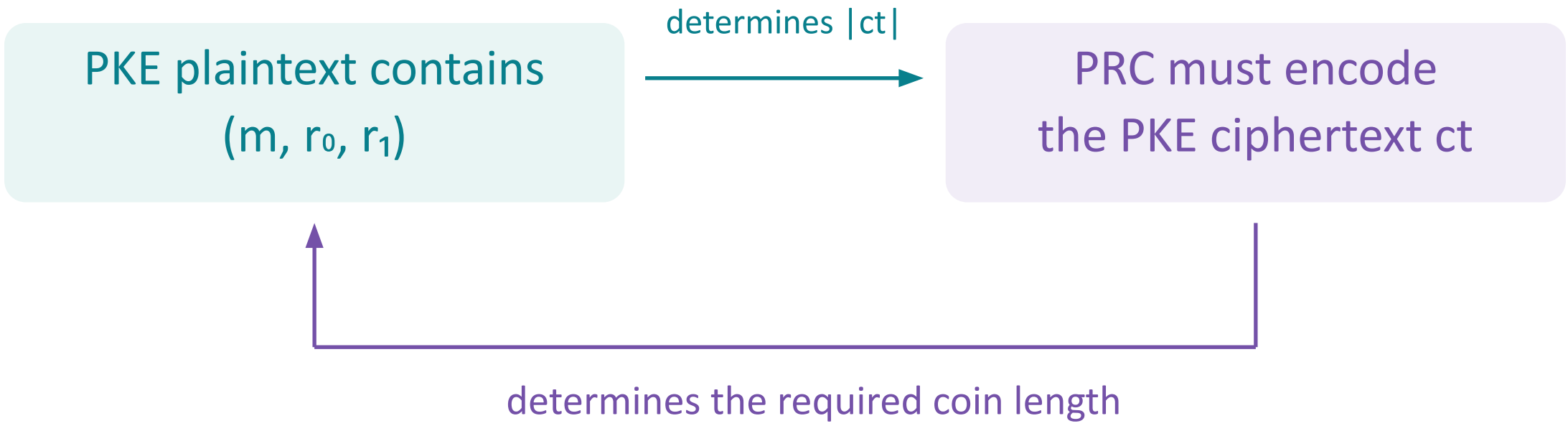
Myers–Shelat · FOCS 2009

Bit Encryption Is Complete

Hohenberger–Lewko–Waters · EUROCRYPT 2012

Detecting Dangerous Queries: A New Approach for Chosen Ciphertext Security

The first attempt creates a length cycle



The direct construction need not have a consistent parameter choice.

Short seeds break the length cycle

Encrypt (m, s_0, s_1)
Each seed has λ bits



Expand only for encoding
$r_0 = G(s_0)$, $r_1 = G(s_1)$

$$|m| + 2\lambda \rightarrow |ct| \rightarrow \text{PRC coin length} \rightarrow \text{PRG output length}$$

Fix the PKE input length first; instantiate the remaining primitives in order.

Why almost perfect correctness?

With overwhelming probability over key generation:

$$\forall\ m, r:\ \mathrm{Dec}(sk, \mathrm{Enc}(pk, m; r)) = m$$

Correctness must hold uniformly for all messages and coins on each good key.

Used when inserting the filter and proving agreement of decoding paths.

The exact adaptive-robustness requirement

$$(m, r, \tilde{c}) \leftarrow A(pk), \quad c := \mathrm{Encode}(pk, m; r)$$

Bad: $\mathrm{Ham}(c, \tilde{c}) \le \alpha n$ and $\mathrm{Decode}(sk, \tilde{c}) \neq m$

Required for both independently keyed base PRC instances.

Only one base secret key is retained

Actual secret key
$(sk_0, sk_{CCA}, pk_0, pk_1)$

Proof setup
Temporarily retains sk_1

sk_1 is discarded by the real setup.

The second decoding path is a proof device, not an extra decoding step.

Why ciphertext unpredictability needs proof

01

Suppose

A fixed plaintext has
a heavy ciphertext atom.

02

Collision test

Two independent
encryptions
then collide noticeably.

03

Contradiction

Correctness separates
messages;
this would violate
IND-CPA.

Lemma 3.5 controls the expected maximum ciphertext probability over keys.