



PALOMA: Binary Separable Goppa-based KEM

2024.02.28.

Dong-Chan Kim, Chang-Yeol Jeon, Minji Kim,
Dong Hyun Park, Dong Hyeon Kim, Yeonghyo Kim

Future cryptography Design Lab.
Kookmin University, Korea

<https://sites.google.com/kookmin.ac.kr/fdl>

ALICE



BOB



ALICE with key **K**



BOB with key **K**



ALICE with key **K**



$C \leftarrow \text{Encrypt}(\mathbf{K}, P)$

BOB with key **K**



$P \leftarrow \text{Decrypt}(\mathbf{K}, C)$

ALICE with key **K**



$C \leftarrow \text{Encrypt}(K, P)$

BOB with key **K**



$P \leftarrow \text{Decrypt}(K, C)$



How can we share the key **K**?



New Directions in Cryptography

Invited Paper

WHITFIELD DIFFIE AND MARTIN E. HELLMAN, MEMBER, IEEE

Abstract—Two kinds of contemporary developments in cryptography are examined. Widening applications of teleprocessing have given rise to a need for new types of cryptographic systems, which minimize the need for secure key distribution channels and supply the equivalent of a written signature. This paper suggests ways to solve these currently open problems. It also discusses how the theories of communication and computation are beginning to provide the tools to solve cryptographic problems of long standing.

I. INTRODUCTION

WE STAND TODAY on the brink of a revolution in cryptography. The development of cheap digital hardware has freed it from the design limitations of mechanical computing and brought the cost of high grade cryptographic devices down to where they can be used in such commercial applications as remote cash dispensers and computer terminals. In turn, such applications create

The best known cryptographic problem is that of privacy: preventing the unauthorized extraction of information from communications over an insecure channel. In order to use cryptography to insure privacy, however, it is currently necessary for the communicating parties to share a key which is known to no one else. This is done by sending the key in advance over some secure channel such as private courier or registered mail. A private conversation between two people with no prior acquaintance is a common occurrence in business, however, and it is unrealistic to expect initial business contacts to be postponed long enough for keys to be transmitted by some physical means. The cost and delay imposed by this key distribution problem is a major barrier to the transfer of business communications to large teleprocessing networks.

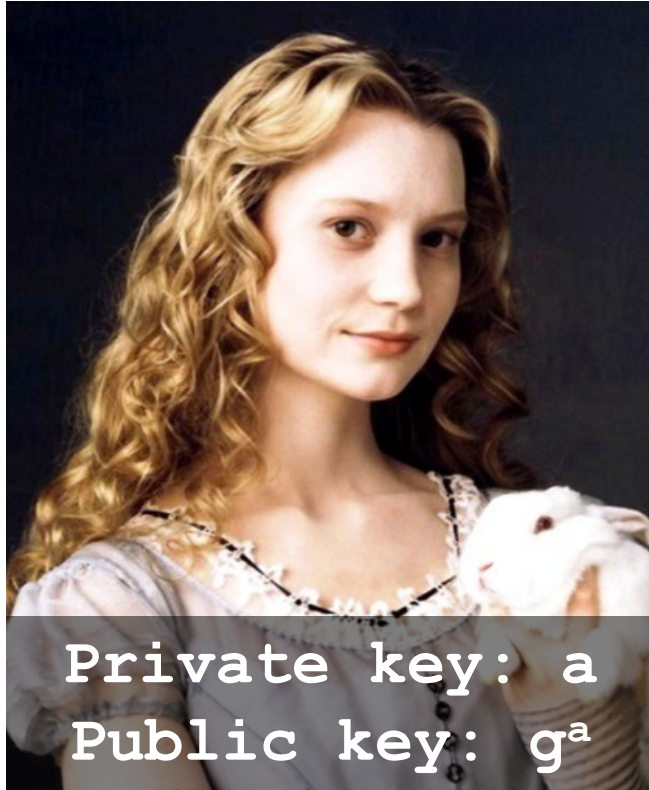
Section III proposes two approaches to transmitting keying information over public (i.e., insecure) channels without compromising the security of the system. In a *public key cryptosystem* enciphering and deciphering are



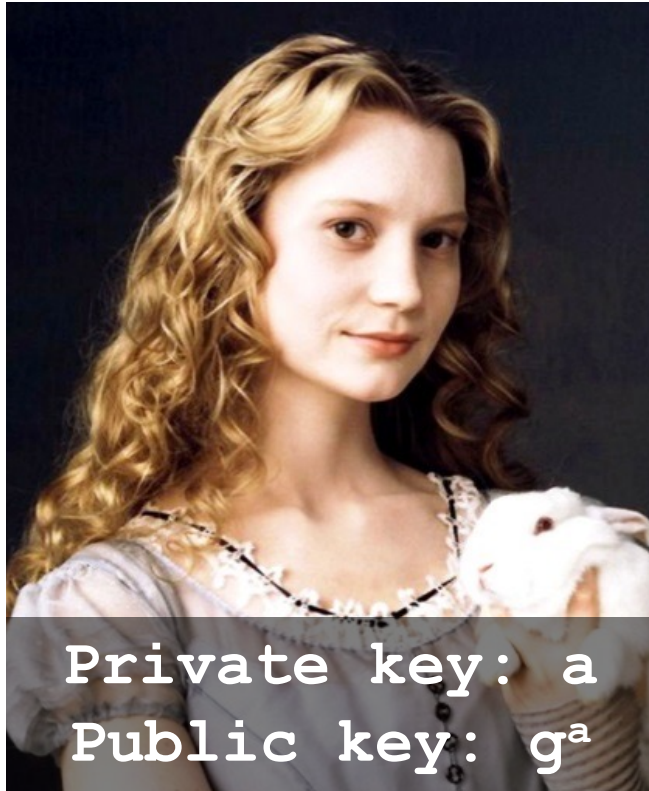
Martin E. Hellman

Whitfield Diffie

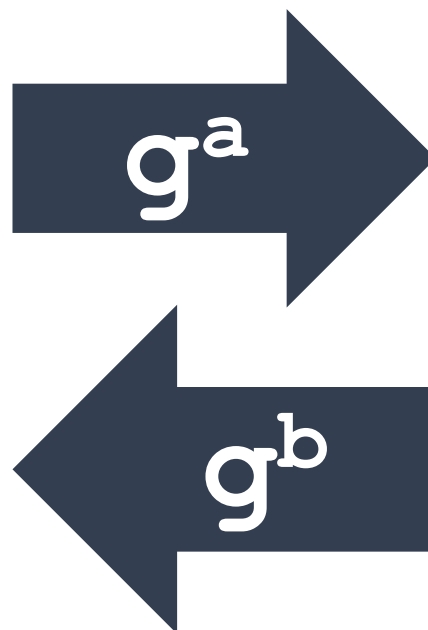
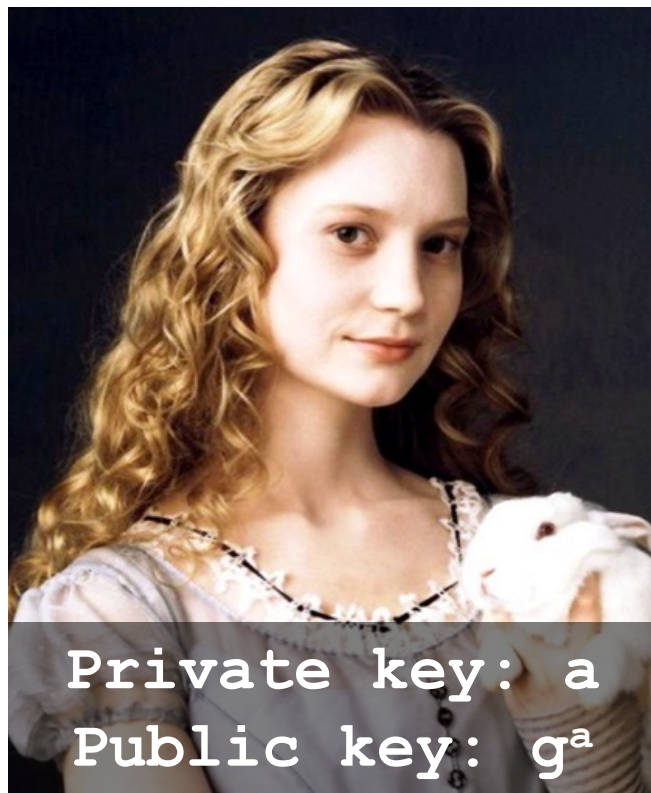
Diffie-Hellman Key Exchange (1976)



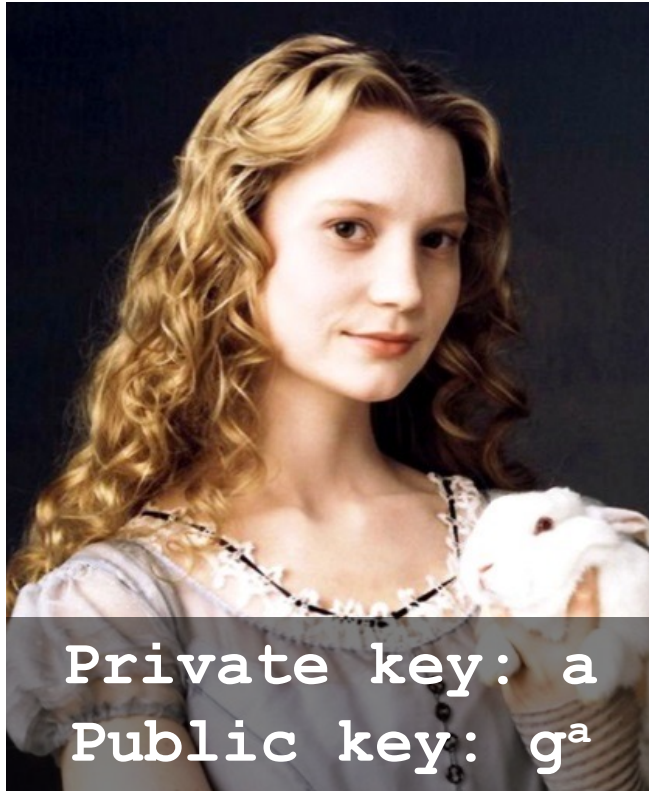
Diffie-Hellman Key Exchange (1976)



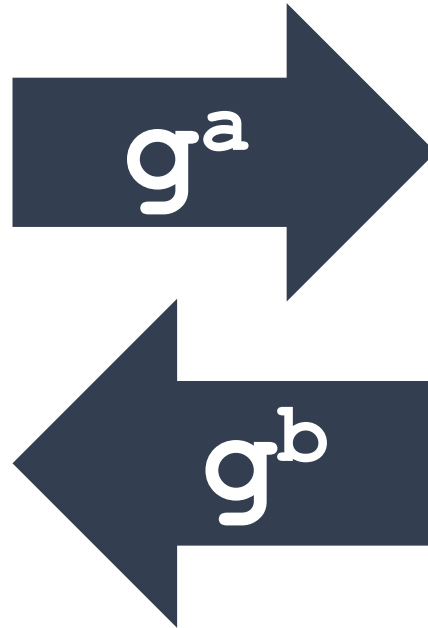
Diffie-Hellman Key Exchange (1976)



Diffie-Hellman Key Exchange (1976)

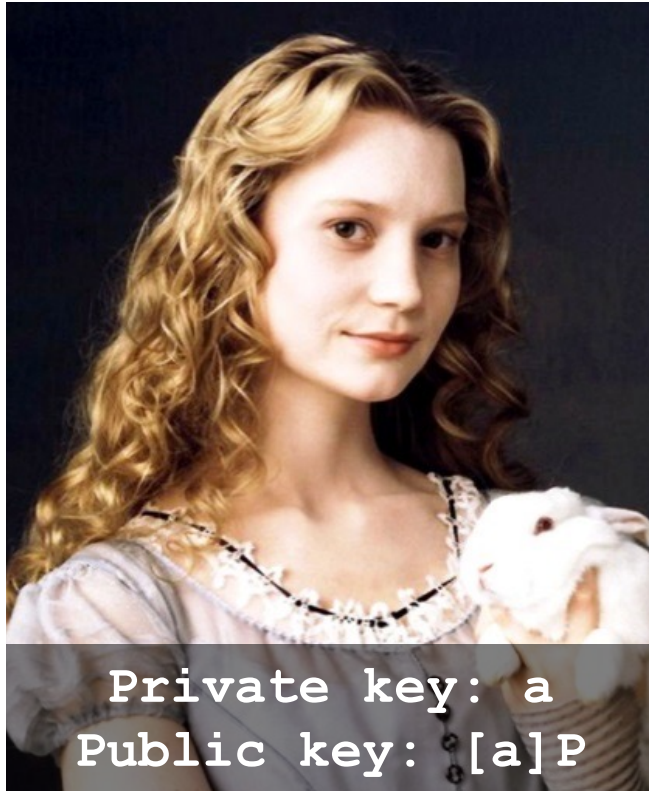


$$K = (g^b)^a$$

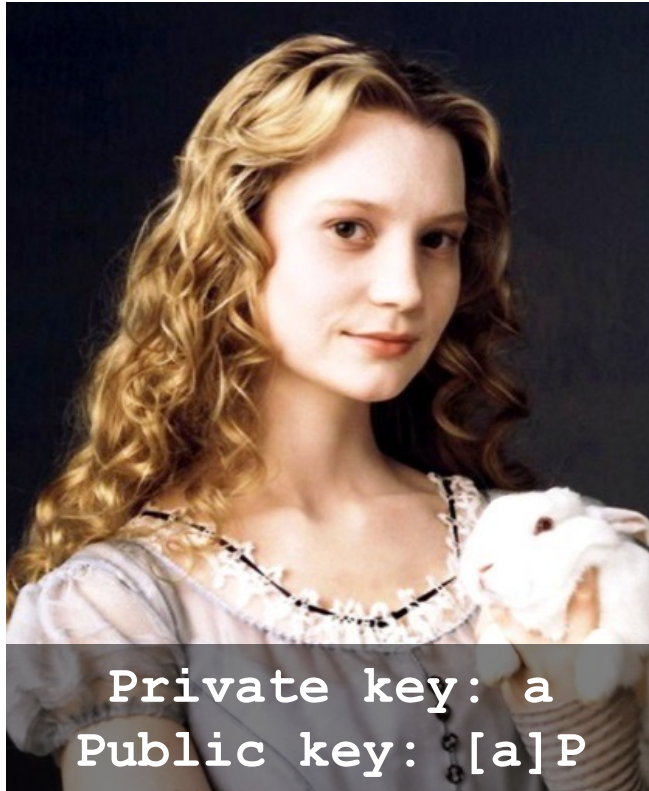


$$K = (g^a)^b$$

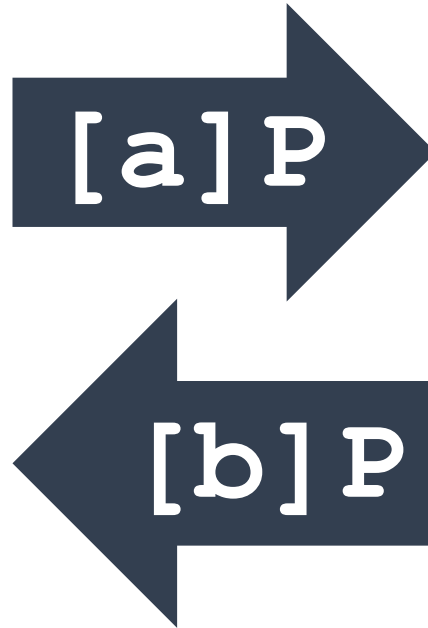
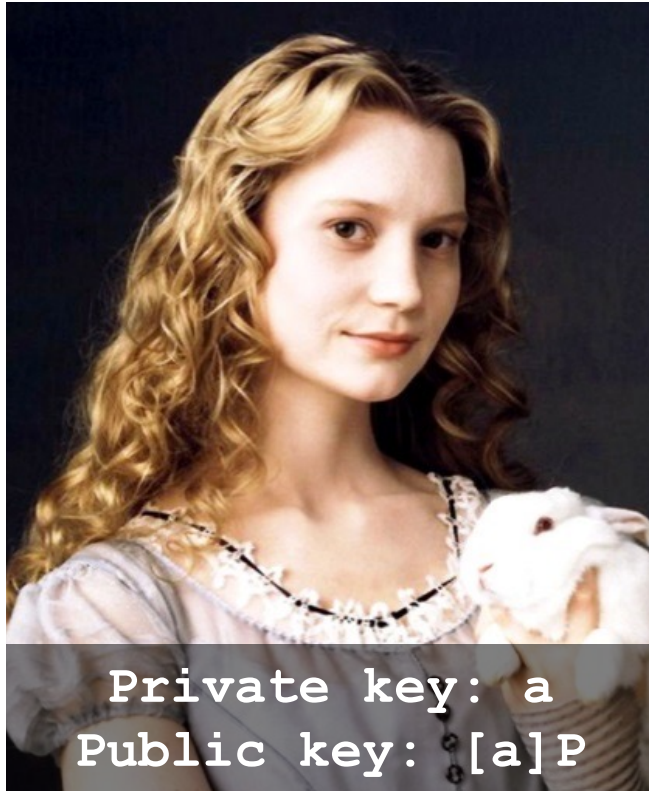
Elliptic Curve based Diffie-Hellman Key Exchange (1985)



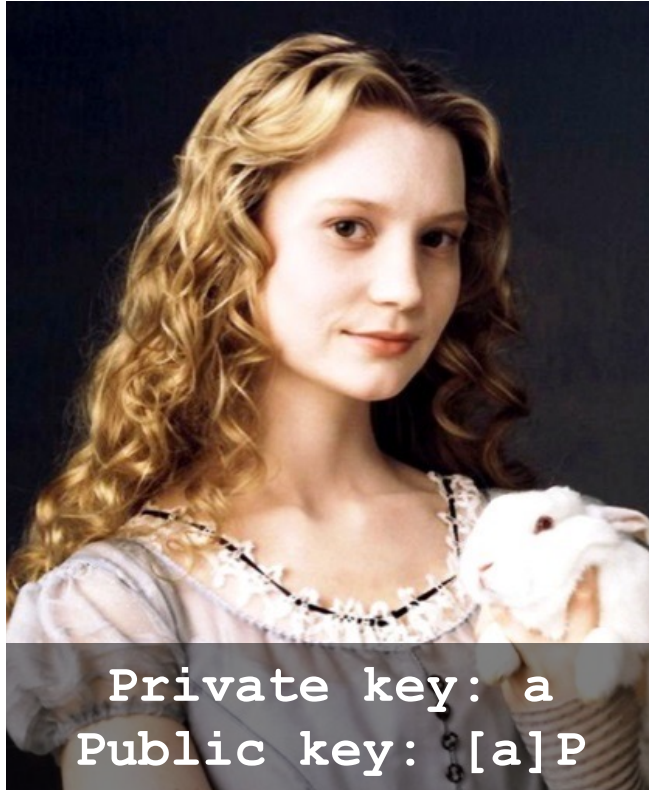
Elliptic Curve based Diffie-Hellman Key Exchange (1985)



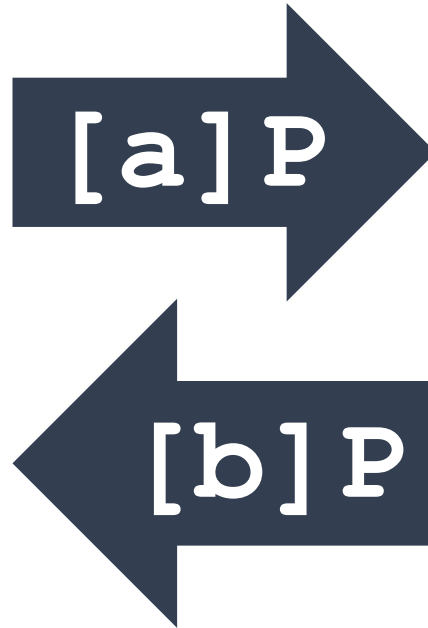
Elliptic Curve based Diffie-Hellman Key Exchange (1985)



Elliptic Curve based Diffie-Hellman Key Exchange (1985)

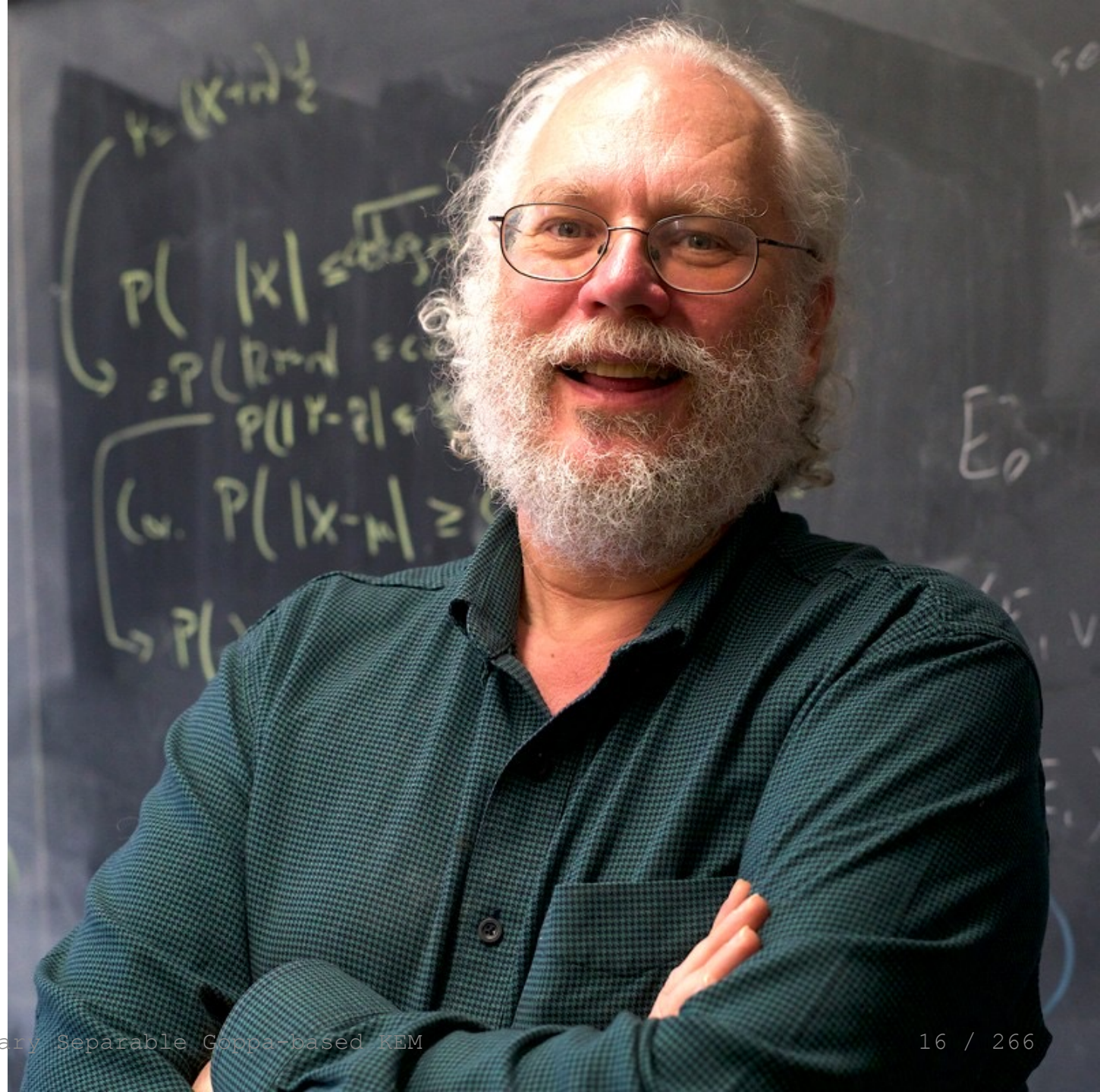


$$K = [a] ([b]P)$$



$$K = [b] ([a]P)$$

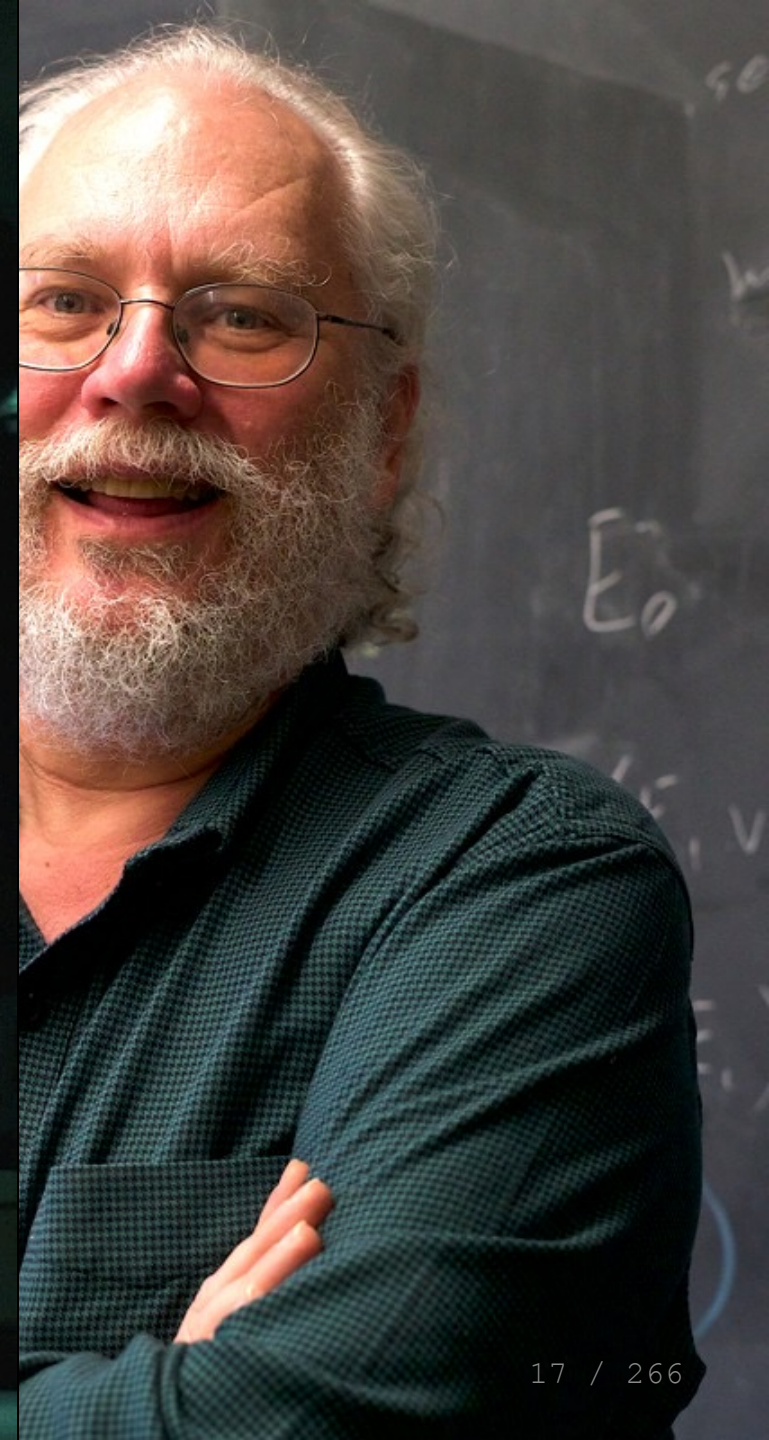
Shor Alg. (1994)



Shor Alg.



IFP/DLP



Shor Alg.

We need a new cipher
based on a different problem,
not relying on IFP/DLP,
to replace DH and ECDH!

IFP/DLP

KEM (Key Encapsulation Mechanism)

Public Key



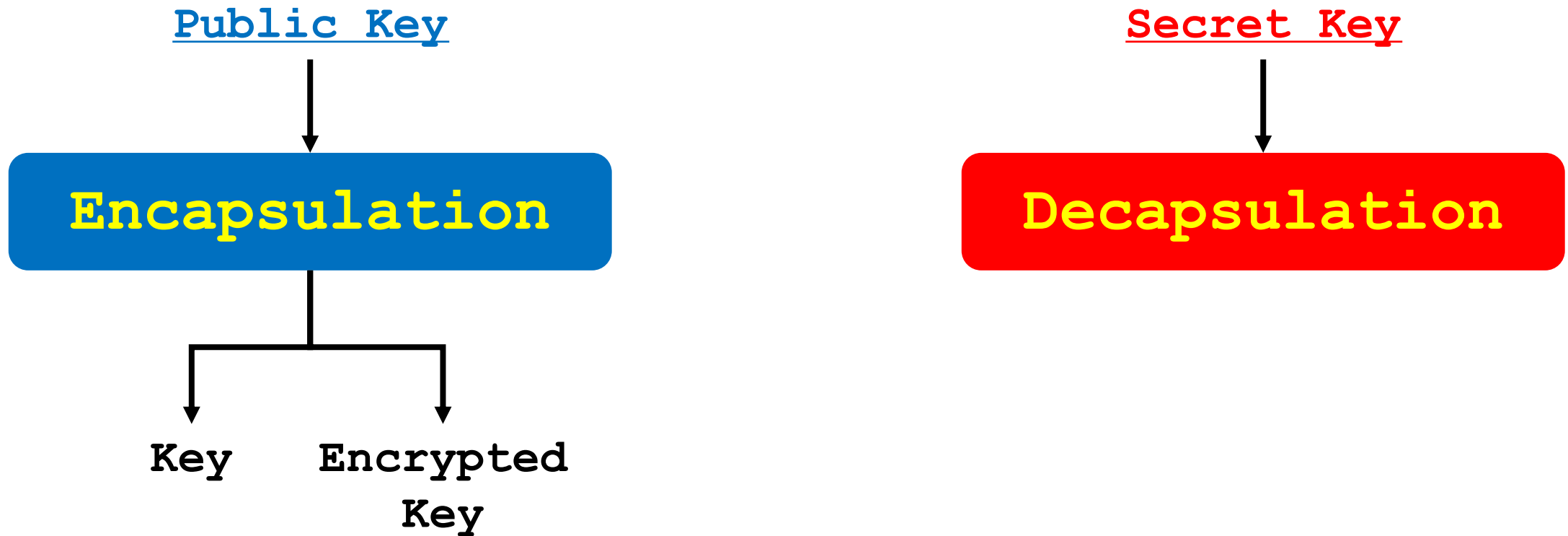
Encapsulation

Secret Key

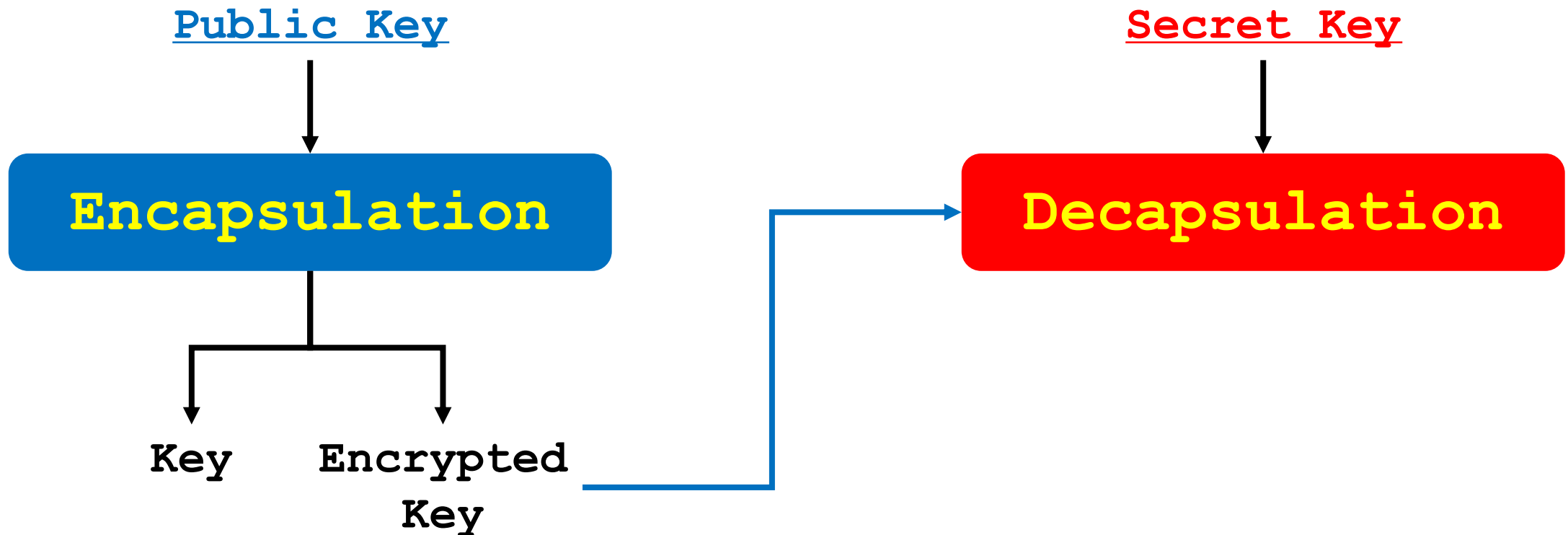


Decapsulation

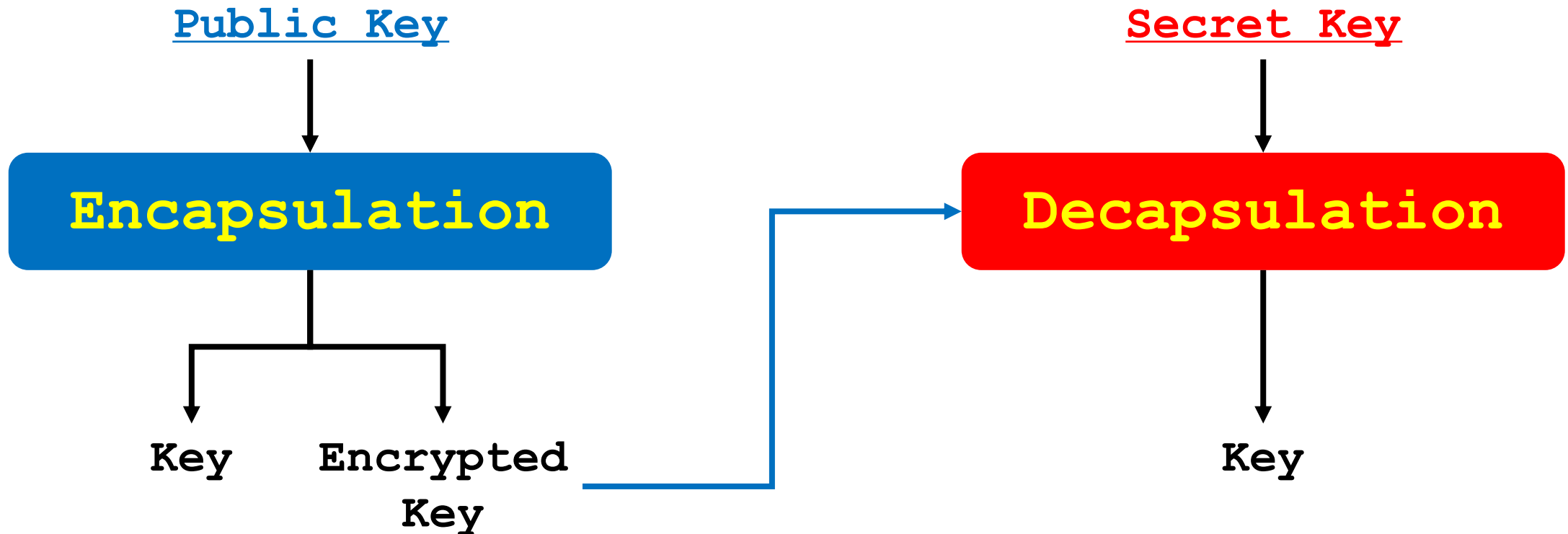
KEM (Key Encapsulation Mechanism)



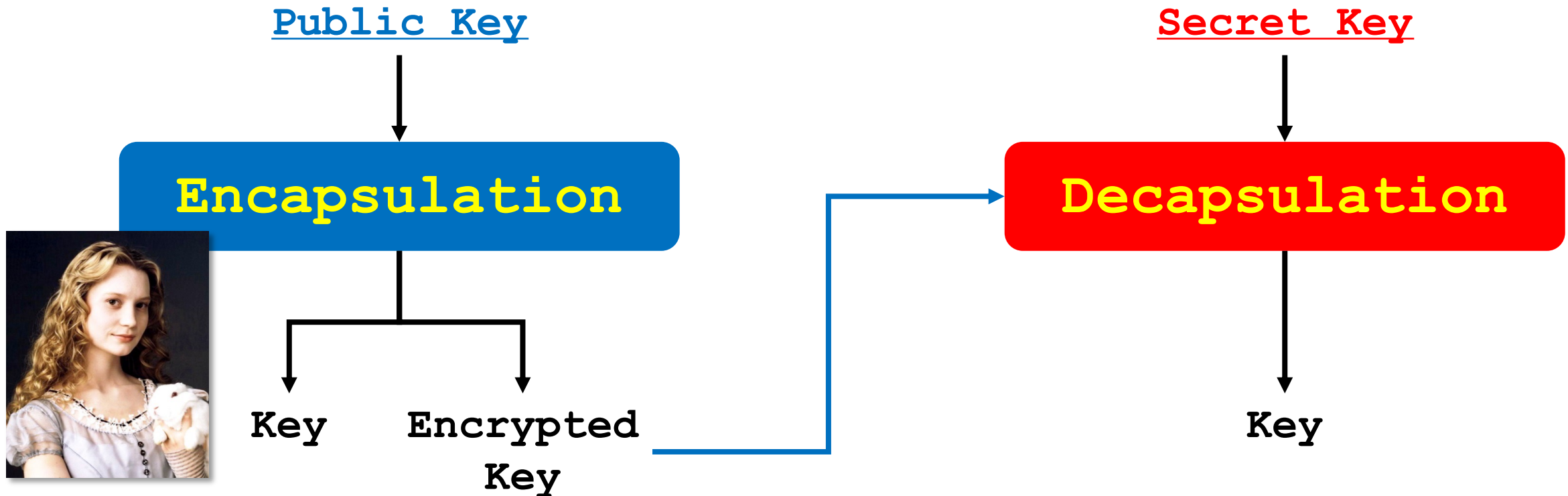
KEM (Key Encapsulation Mechanism)



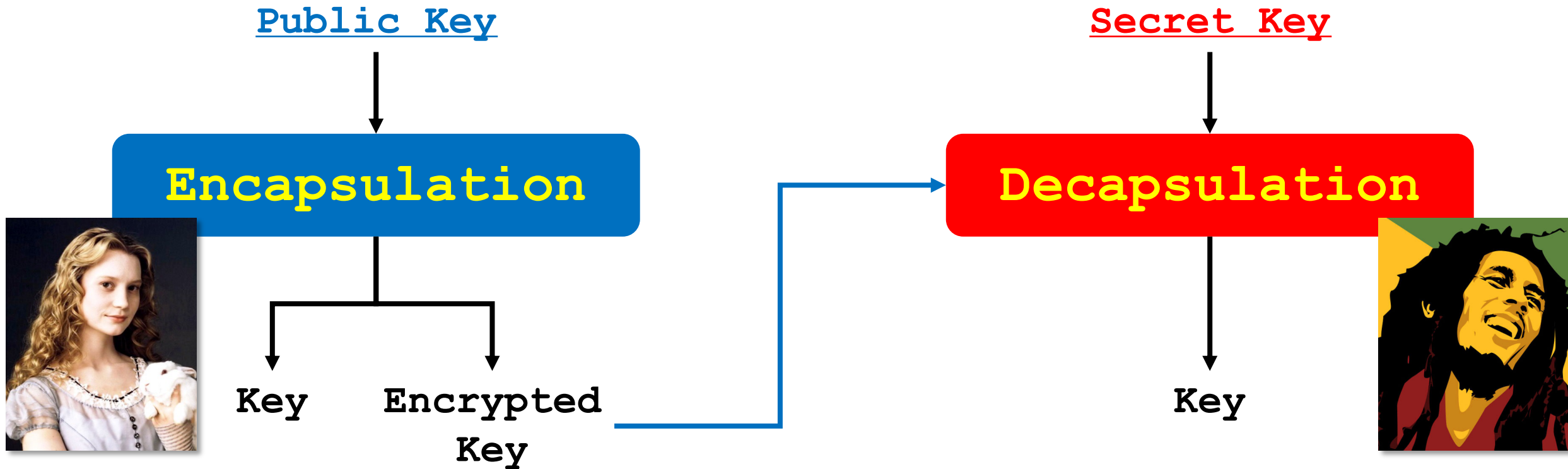
KEM (Key Encapsulation Mechanism)



KEM (Key Encapsulation Mechanism)



KEM (Key Encapsulation Mechanism)



PALOMA MENSAJERA



PALOMA MENSAJERA



PALOMA MENSAJERA



Please Always Laugh Out loud, My Amigo!

KpqC Competition Round 1

(November 2022 ~ November 2023)

- [KpqC-bulletin board](#) : The kpqc-bulletin Google group for any official comments on the first round candidate algorithms (To send a post, refer to [here](#).)
- Email kpqcrypto@gmail.com for any administrative questions.

Public-key Encryption and Key-establishment Algorithms

* : Principal submitter

Algorithm	Algorithm Information	Submitters
IPCC	Document Implementation package Website	Jieun Ryu Yongbhin Kim Seungtai Yoon Ju-Sung Kang Yongjin Yeom*
Layered ROLLO-I	Document Implementation package Website	Chanki Kim* Young-Sik Kim
NTRU+	Document Implementation package	Jonghyun Kim Jong Hwan Park *
PALOMA	Document Implementation package	Dong-Chan Kim* Chang-Yeol Jeon Yeonghyo Kim Minji Kim

2024.02.28. @ KpqC Winter Camp

PALOMA: Binary Separable Goppa-based KEM¹

Dong-Chan Kim Chang-Yeol Jeon Yeonghyo Kim Minji Kim

Future cryptography Design Lab., Kookmin University

dckim@kookmin.ac.kr

2022

¹This work is submitted to ‘Korean Post-Quantum Cryptography Competition’ (www.kpqc.or.kr).



CBCrypto 2023

International Workshop on Code-Based Cryptography

Lyon, France, April 22-23, 2023



Code-based cryptography is the area of research that focuses on the study of cryptosystems based on error-correcting codes, following the seminal work of McEliece and Niederreiter in the late 1970s - early 1980s. These systems have shown no vulnerabilities to quantum attackers and this research branch is widely regarded as one of the most promising in the so-called area of Post-Quantum Cryptography. Current efforts in code-based cryptography are directed at producing fast, secure and efficient schemes. Research in this area has also been fostered by the recent [NIST's Post-Quantum Standardization call](#).

The goal of this two-day event is to promote this research area to an increasingly larger audience. Besides bringing together the existing community, in fact, CBCrypto aims at providing an opportunity to extend the range of participation to researchers approaching this area for the first time, or simply interested in knowing more about it. The program includes invited talks, contributed talks and dedicated discussion sessions.



CBCrypto 2023

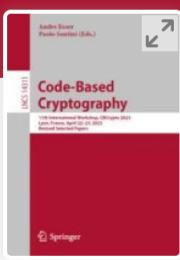
International Workshop on Code-Based Cryptography

Lyon, France, April 22-23, 2023



Code-based cryptography is the area of research that focuses on the study of cryptosystems based on error-correcting codes, following the seminal work of McEliece and Niederreiter. It is widely regarded as one of the most promising directions directed at producing future-proof cryptography [call](#).

The goal of this two-day workshop is to provide CBCrypto aims at providing a platform for researchers in knowing more about it.



Code-Based Cryptography Workshop

↳ CBCrypto 2023: **Code-Based Cryptography** pp 144–173 | [Cite as](#)

[Home](#) > [Code-Based Cryptography](#) > [Conference paper](#)

PALOMA: Binary Separable Goppa-Based KEM

[Dong-Chan Kim](#) , [Chang-Yeol Jeon](#), [Yeonghyo Kim](#) & [Minji Kim](#)

Conference paper | [First Online: 28 October 2023](#)

PALOMA: Binary Separable Goppa-based KEM

Dong-Chan Kim, Chang-Yeol Jeon, Yeonghyo Kim, and Minji Kim

Future cryptography Design Lab., Kookmin University, Korea
{dckim,chjeon96,yh17,minji0022}@kookmin.ac.kr

Abstract. In this paper, we propose PALOMA, a new code-based key encapsulation mechanism, which is designed by combining an NP-hard SDP(Syndrome Decoding Problem)-based trapdoor with a binary separable Goppa code and FO(Fujisaki-Okamoto) transformation. Cryptographic schemes based on an SDP defined with a binary Goppa code have not been found to be vulnerable to critical attacks, and the FO transformation ensures IND-CCA2 security in the ROM(Random Oracle Model). The combination is highly regarded in cryptographic communities for its strong security guarantees. PALOMA has a public key size of approximately 300KB or more due to its SDP-based trapdoor nature. Furthermore, the key generation process, which involves generating the parity-check matrix of the scrambled Goppa code, is relatively slow compared to other post-quantum ciphers. However a primary role of post-quantum cryptography is to serve as an alternative to current cryptosystems that are vulnerable to quantum computing attacks. Therefore, in post-quantum cryptography, ensuring strong security guarantees is more important than efficiency. Consequently, we have designed PALOMA with a focus on conservative security guarantees, while ensuring that there is no significant degradation in application quality.

Keywords: code-based key encapsulation mechanism · Binary separable Goppa code · Patterson decoding · post-quantum cryptography

1 Introduction

In this paper, we propose PALOMA, a new code-based KEM(Key Encapsulation Mechanism), which has the following features:

- Trapdoor based on SDP with a binary separable Goppa code.
- IND-CCA2-secure KEM based on FO transformation.
- Parameter sets that ensure security strengths of 128, 192, and 256-bit.

1.1 Trapdoor

Syndrome Decoding Problem. SDP is a problem of finding the preimage vector with a specific Hamming weight for a given random binary parity-check matrix and a syndrome. In 1978, SDP was proven to be NP-hard because it is equivalent to the 3-dimensional matching problem[3,15]. The McEliece and Niederreiter



CBCrypto 2023

International Workshop on Code-Based Cryptography

Lyon, France, April 22-23, 2023



PALOMA: Binary Separable Goppa-based KEM

Dong-Chan Kim, Chang-Yeol Jeon, Yeonghyo Kim, and Minji Kim

Future cryptography Design Lab., Kookmin University, Korea
 {dckim,chjeon96,yh17,minji0022}@kookmin.ac.kr

Abstract. In this paper, we propose PALOMA, a new code-based key encapsulation mechanism, which is designed by combining an NP-hard SDP(Syndrome Decoding Problem)-based trapdoor with a binary separable Goppa code and FO(Fujisaki-Okamoto) transformation. Cryptographic schemes based on an SDP defined with a binary Goppa code have not been found to be vulnerable to critical attacks, and the FO transformation ensures IND-CCA2 security in the ROM(Random Oracle Model). The combination is highly regarded in cryptographic communities for its strong security guarantees. PALOMA has a public key size of approximately 300KB or more due to its SDP-based trapdoor nature. Furthermore, the key generation process, which involves generating the parity-check matrix of the scrambled Goppa code, is relatively slow compared to the post-quantum schemes where a primary goal of post-quantum cryptography is to design an alternative to current cryptosystems that are vulnerable to quantum computing attacks. Therefore, in post-quantum cryptography, ensuring strong security guarantees is more important than efficiency. Consequently, we have designed PALOMA with a focus on conservative security guarantees, while ensuring that there is no significant degradation in communication quality.

Keywords: code-based key encapsulation mechanism · Binary separable Goppa code · Patterson decoding · post-quantum cryptography

1 Introduction

In this paper, we propose PALOMA, a new code-based KEM(Key Encapsulation Mechanism), which has the following features:

- Trapdoor based on SDP with a binary separable Goppa code.
- IND-CCA2-secure KEM based on FO transformation.
- Parameter sets that ensure security strengths of 128, 192, and 256-bit.

1.1 Trapdoor

Syndrome Decoding Problem. SDP is a problem of finding the preimage vector with a specific Hamming weight for a given random binary parity-check matrix and a syndrome. In 1978, SDP was proven to be NP-hard because it is equivalent to the 3-dimensional matching problem[3,15]. The McEliece and Niederreiter

No changes to the specification

Enhanced security proof

Code-Based Cryptography Workshop

↳ CBCrypto 2023: **Code-Based Cryptography** pp 144–173 | [Cite as](#)



[Home](#) > [Code-Based Cryptography](#) > [Conference paper](#)

PALOMA: Binary Separable Goppa-Based KEM

[Dong-Chan Kim](#) , [Chang-Yeol Jeon](#), [Yeonghyo Kim](#) & [Minji Kim](#)

Conference paper | [First Online: 28 October 2023](#)

KpqC Competition Round

Selected Algorithms from the KpqC Competition Round 1

(December 7, 2023)

After nearly a year of evaluation, the KpqC team is pleased to announce the algorithms that will advance to the KpqC Competition Round 2.
The following are eight Round 2 candidates.

Digital Signature	PKE/KEM
AlMer	NTRU+
HAETAE	PALOMA
MQ-Sign	REDOG
NCC-Sign	SMAUG+TIGER (merged)

We would like to thank all the Round 1 submission teams for their efforts in this competition. They did have an impact on the maturation of the PQC fields in South Korea during Round 1.

We also want to thank the public reviewers for their interests in and comments on the KpqC algorithms. Better outcomes were achieved as a result of public scrutiny and analysis.

Lastly, we would like to thank all the people who assisted us and gave constructive comments to enhance the KpqC Round 1.

PALOMA: Binary Separable Goppa-based KEM¹

(KpqC Round 2 Proposal)

Dong-Chan Kim[†], Chang-Yeol Jeon, Minji Kim
Dong Hyun Park, Dong Hyeon Kim, Yeonghyo Kim

Future cryptography Design Lab., Kookmin University

dckim@kookmin.ac.kr

February 23, 2024



¹This work is submitted to ‘Korean Post-Quantum Cryptography Competition’ (www.kpqc.or.kr).

KpqC Competition Round

Selected Algorithms from the KpqC Competition Round 1

(December 7, 2023)

Two changes to the specification
Enhanced security proof
Two developers have joined

Digital Signature	Public Key Encryption
AI-Mer	NTRU+
MA	CRYSTALS-Dilithium
MQ-Sign	REDUC
NCC-Sign	SMAUG+TIGER (merged)

We would like to thank all the Round 1 submission teams for their efforts in this competition. They did have an impact on the maturation of the PQC fields in South Korea during Round 1.

We also want to thank the public reviewers for their interests in and comments on the KpqC algorithms. Better outcomes were achieved as a result of public scrutiny and analysis.

Lastly, we would like to thank all the people who assisted us and gave constructive comments to enhance the KpqC Round 1.

PALOMA: Binary Separable Goppa-based KEM¹

(KpqC Round 2 Proposal)

Dong-Chan Kim[†], Chang-Yeol Jeon, Minji Kim
Dong Hyun Park, Dong Hyeon Kim, Yeonghyo Kim

Future cryptography Design Lab., Kookmin University

dckim@kookmin.ac.kr

February 23, 2024



¹This work is submitted to 'Korean Post-Quantum Cryptography Competition' (www.kpqc.or.kr).

Round 1 Ver. vs. Round 2 Ver.

	1R	2R	Reason
GenPermMat	$\mathbf{P} \leftarrow \prod_{j=0}^{n-1} \mathbf{P}_{j,l_j}$	$\mathbf{P} \leftarrow [u_{l_0} \mid \cdots \mid u_{l_{n-1}}]$	Ensuring that \mathbf{P} is sampling from a uniform distribution
Secret key sk	$(L, g(X), \mathbf{S}^{-1}, r_{\widehat{c}})$	$(L, g(X), \mathbf{S}^{-1}, r_{\widehat{c}}, r)$	A 256-bit string r , which is independent of $(L, g(X), \mathbf{S}^{-1}, r_{\widehat{c}})$, is added for implicit rejection.

Dong-Chan Kim, Chang-Yeol Jeon,
Yeonghyo Kim, Minji Kim



Dong-Chan Kim, Chang-Yeol Jeon,
Minji Kim, **Dong Hyun Park**,
Dong Hyeon Kim, Yeonghyo Kim

Code-based KEM PALOMA based on:

1. **Syndrome Decoding Problem** (NP-hard),
2. **Niederreiter-type** Trapdoor (syndrome scrambling),
3. **Binary Separable** (**not irreducible**) **Goppa** Code, (no critical attack so far) and
 - almost constant-time random generation of a Goppa code
 - extended Patterson decoding
4. **Fujisaki-Okamoto** Structure
 - proven to be IND-CCA2-secure in ROM

Contents

1. Preliminaries (ECC / SDP / code-based trapdoor / binary Goppa code)
2. The design rationale of PALOMA
3. Specification (trapdoor / KEM / parameter sets)
4. Evaluation of security (trapdoor / KEM)
5. Performance (data size / speed)
6. Comparison between PALOMA and Classic McEliece

Contents

- 1. Preliminaries** (ECC / SDP / code-based trapdoor / binary Goppa code)
2. The design rationale of PALOMA
3. Specification (trapdoor / KEM / parameter sets)
4. Evaluation of security (trapdoor / KEM)
5. Performance (data size / speed)
6. Comparison between PALOMA and Classic McEliece

CODE

CODE

from Coding Theory



CODE

from Coding Theory (not Cryptography)



CODE

from Coding Theory

(not Cryptography)

a.k.a. Error-Correcting Code



Error-Correcting Code



Error-Correcting Code



HELLO



Error-Correcting Code



HELLO



HELL!

Error-Correcting Code



HELLO



Error-Correcting Code



HELLO



HHHEEEELLLLLLOOOO



Error-Correcting Code



HELLO



HHHEEEELLLLLLOOO HHHEEEELLLLLLO!O



Error-Correcting Code



HELLO



HHHEEEELLLLLLOOO



HHHEEEELLLLLLO!O



HELLO



Error-Correcting Code



Message

HELLO



HHHEEEELLLLLLLOOO



HHHEEEELLLLLLLO!O



HELLO



Error-Correcting Code



Message

HELLO



HHHEEEELLLLLLOO

Codeword



HELLO



HHHEEEELLLLLLO!O

Error-Correcting Code



Message

HELLO

Encode



HHHEEEELLLLLLOO

Codeword



HELLO



HHHEEEELLLLLLO!O

Error-Correcting Code



Message

HELLO



Encode

Codeword

HHHEEEELLLLLLLOO



Decode

HELLO

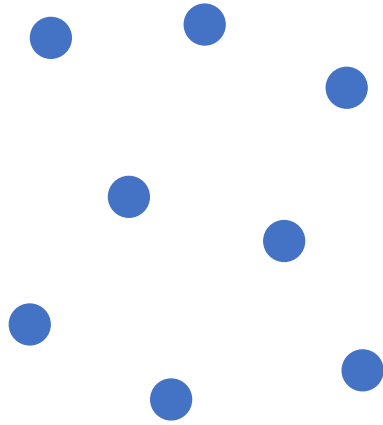


HHHEEEELLLLLLLO!O



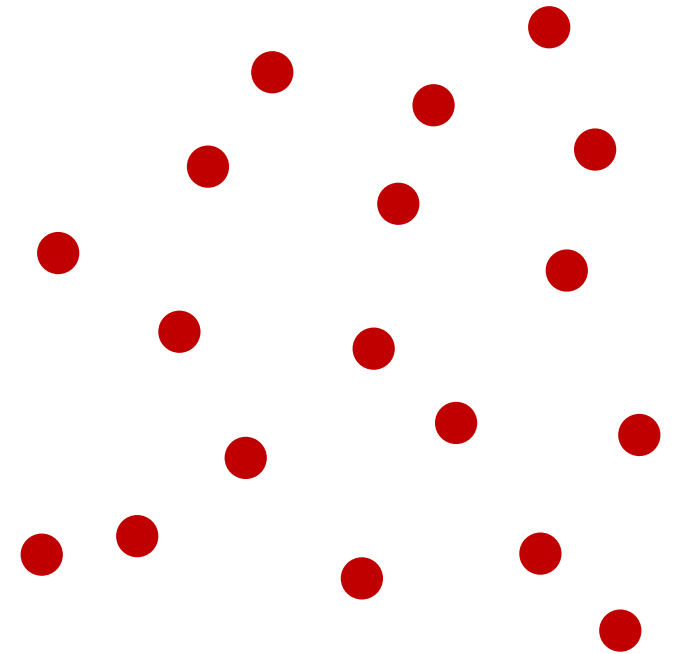
Encoding

$$\mathbb{F}_2^k = \{0, 1\}^k$$



Messages

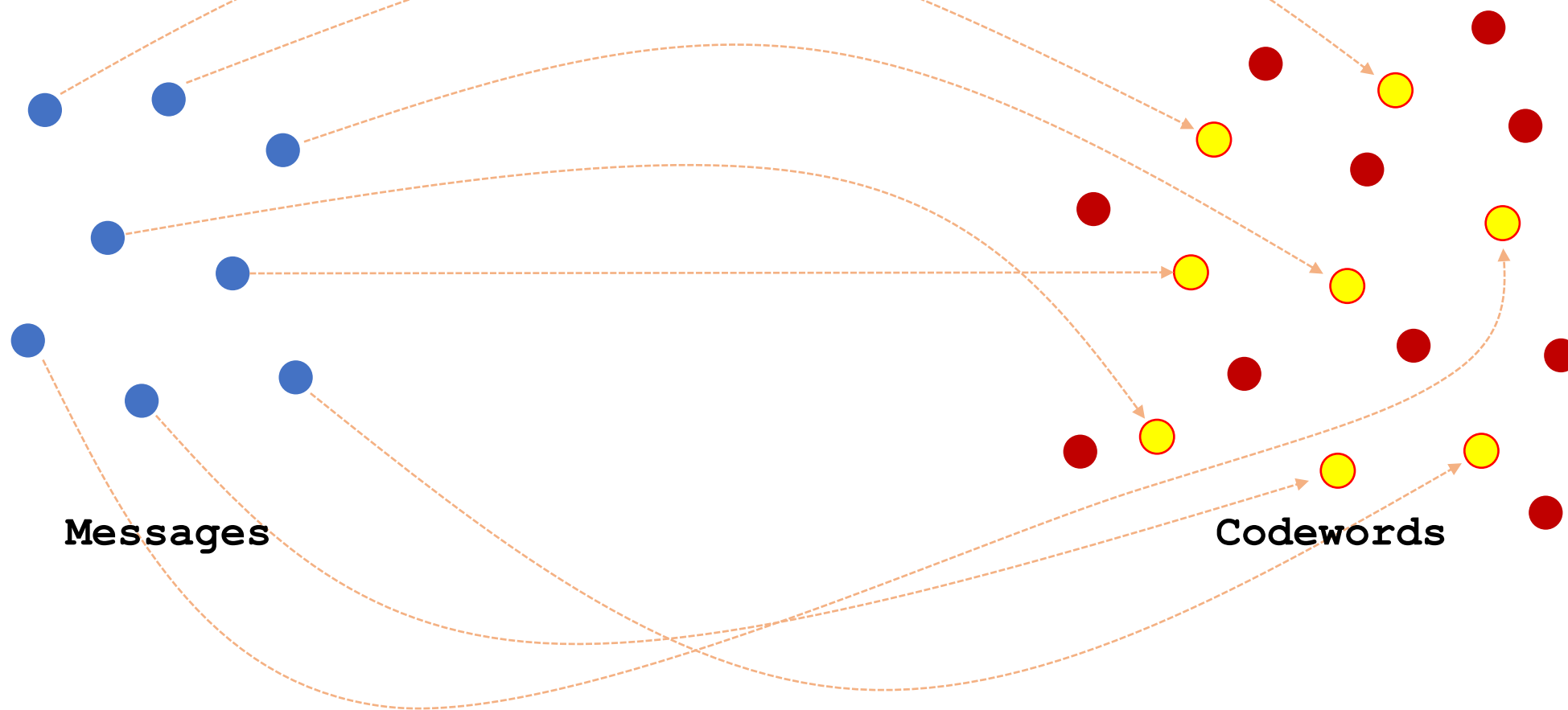
$$\mathbb{F}_2^n = \{0, 1\}^n$$



Encoding

$$\mathbb{F}_2^k = \{0,1\}^k$$

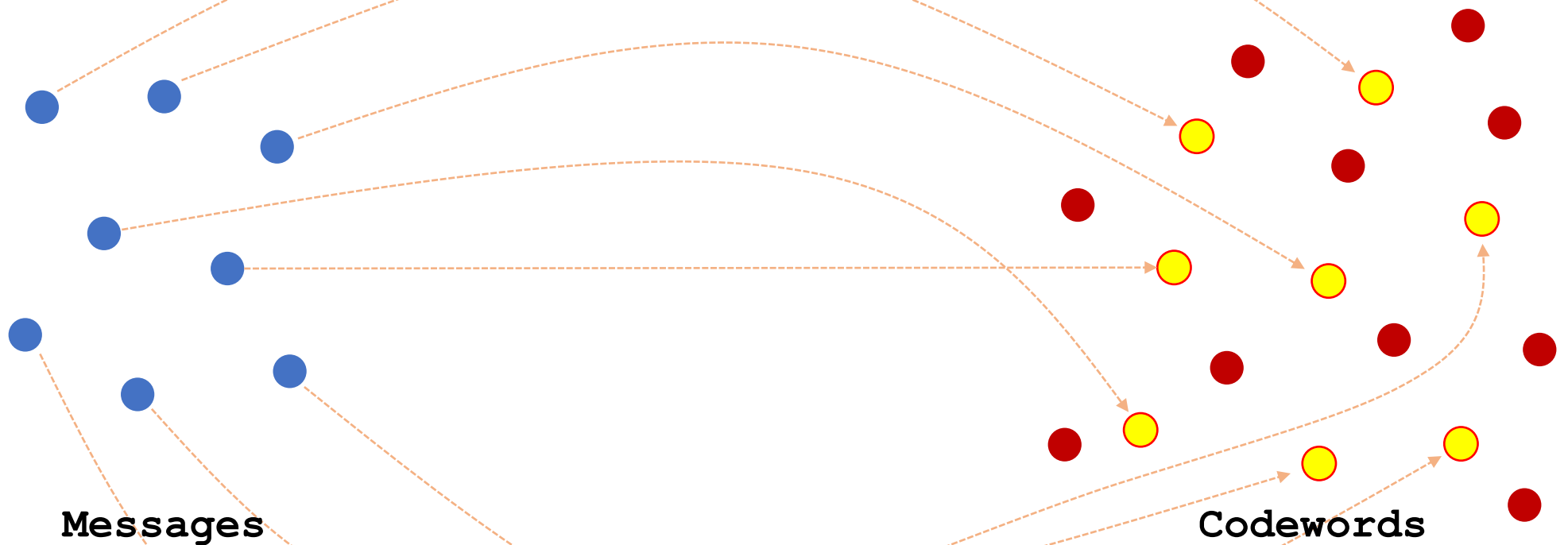
$$\mathbb{F}_2^n = \{0,1\}^n$$



Encoding

$$\mathbb{F}_2^k = \{0,1\}^k$$

$$\mathbb{F}_2^n = \{0,1\}^n$$



Linear Code
= k -dim. subspace of \mathbb{F}_2^n

Linear Code $C = [n, k]_q$

length = n

dimension = k

defined over F_q

Linear Code $C = [10, 5]_2$

```
[1 0 0 1 1 1 0 0 1 1]
[1 0 0 1 0 1 0 0 0 0]
[1 0 0 1 1 1 1 0 0 0]
[0 1 1 0 0 1 0 0 1 1]
[1 1 0 1 0 1 1 0 0 1]
```

Parity-check matrix H

Linear Code $C = [10, 5]_2$

1	0	0	1	1	1	0	0	1	1
1	0	0	1	0	1	0	0	0	0
1	0	0	1	1	1	1	0	0	0
0	1	1	0	0	1	0	0	1	1
1	1	0	1	0	1	1	0	0	1

Parity-check matrix H

$[c_0]$
 $[c_1]$
 $[c_2]$
 $[c_3]$
 $[c_4]$
 $[c_5]$
 $[c_6]$
 $[c_7]$
 $[c_8]$
 $[c_9]$

codeword

=

$[0]$
 $[0]$
 $[0]$
 $[0]$
 $[0]$

Linear Code $C = [10, 5]_2$

1	0	0	1	1	1	0	0	1	1
1	0	0	1	0	1	0	0	0	0
1	0	0	1	1	1	1	0	0	0
0	1	1	0	0	1	0	0	1	1
1	1	0	1	0	1	1	0	0	1

Parity-check matrix H

$[c0]$
$[c1]$
$[c2]$
$[c3]$
$[c4]$
$[c5]$
$[c6]$
$[c7]$
$[c8]$
$[c9]$

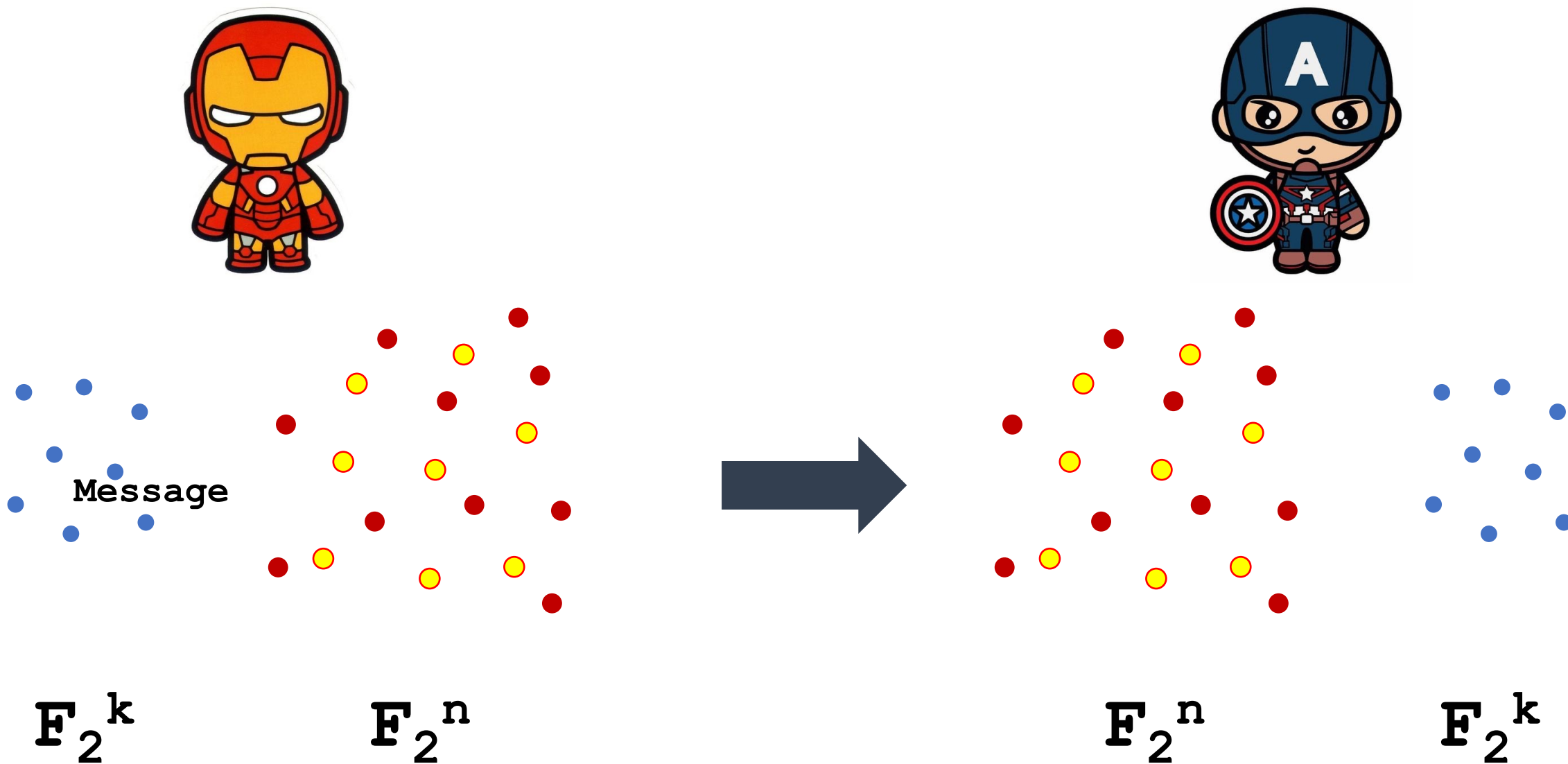
codeword

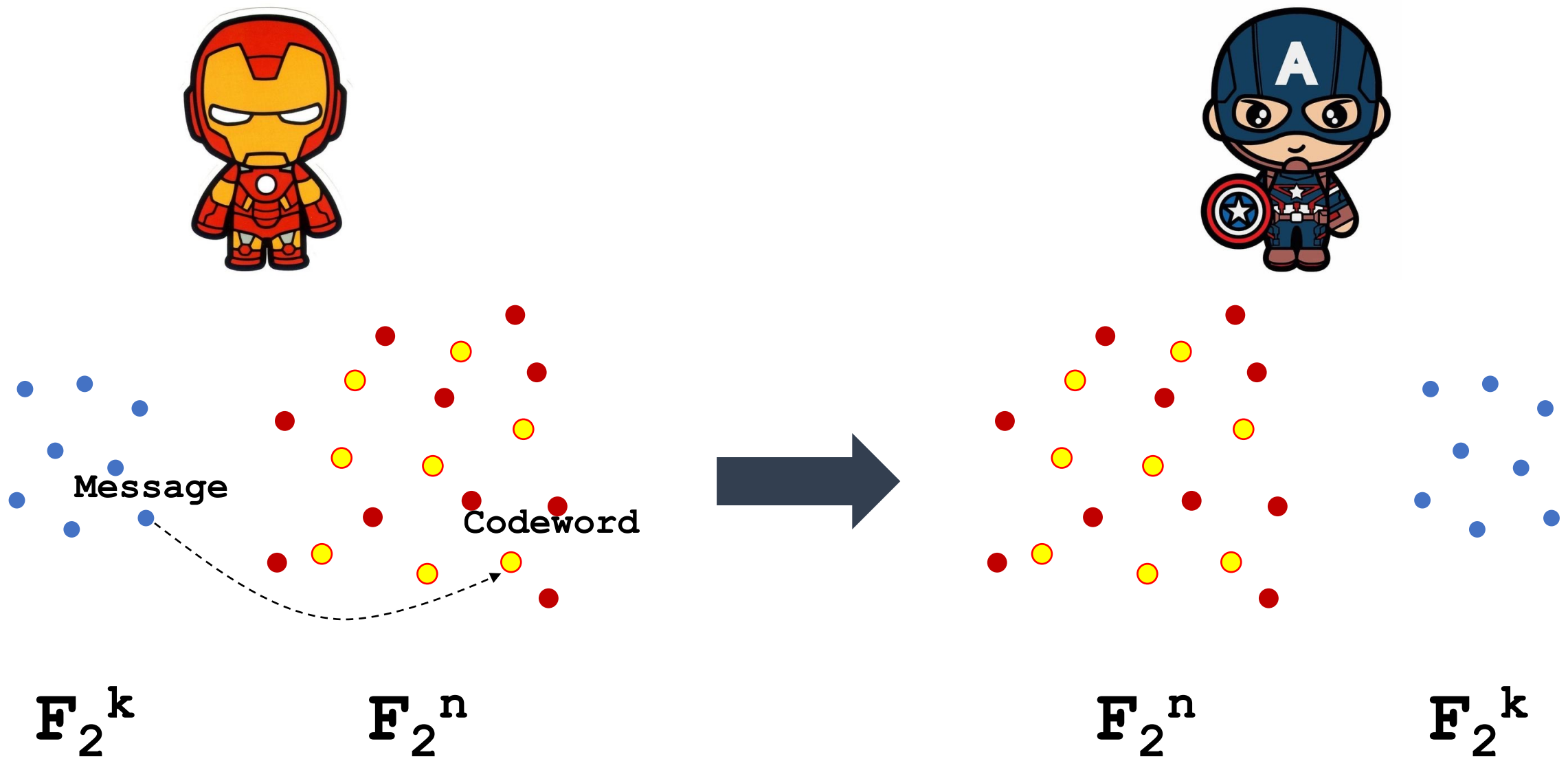
=

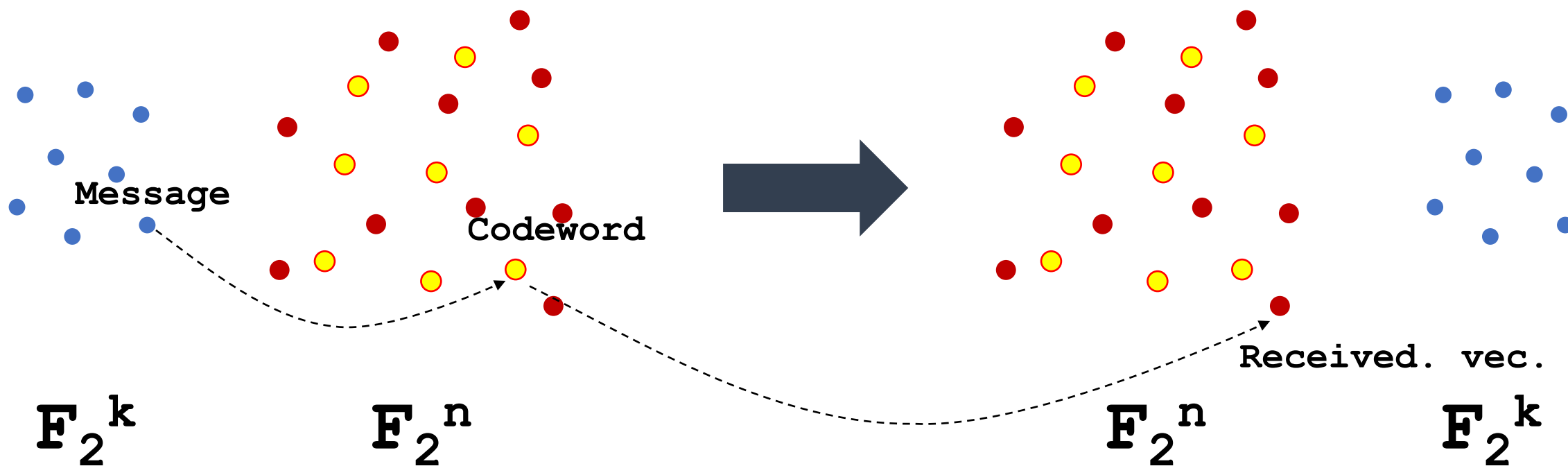
$[0]$
$[0]$
$[0]$
$[0]$
$[0]$

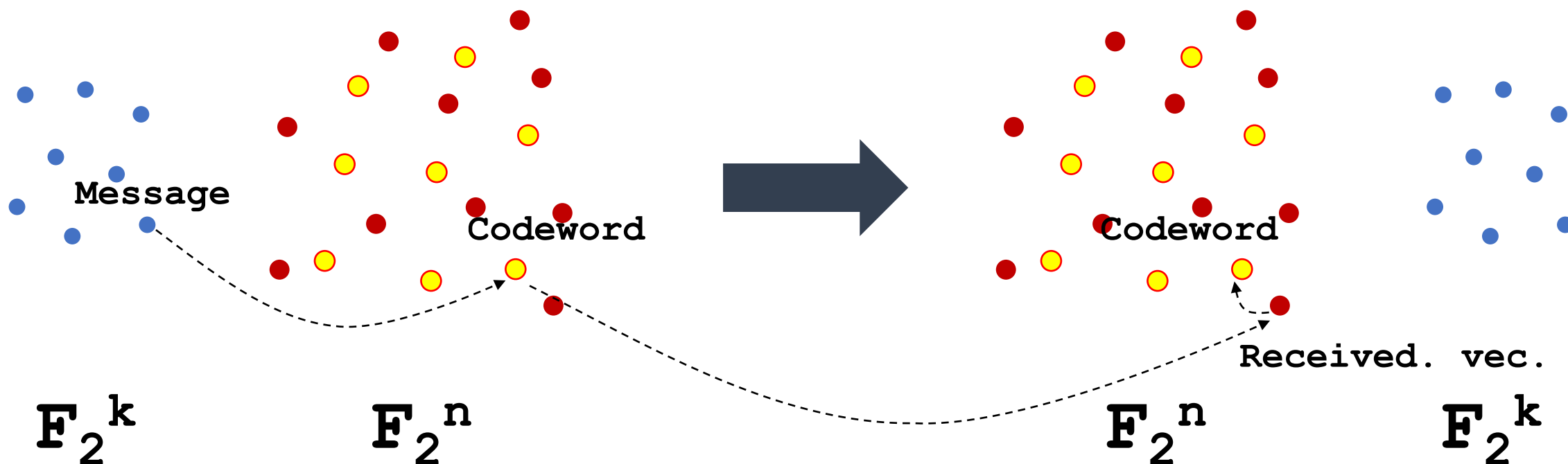
(0, 0, 0, 0, 0, 0, 0, 0, 0, 0)	0
(0, 1, 1, 0, 1, 0, 0, 0, 1, 0)	4
(0, 1, 1, 0, 1, 1, 0, 1, 0, 0)	5
(0, 0, 0, 0, 0, 0, 1, 0, 1, 1)	3
(0, 0, 1, 1, 1, 0, 0, 0, 1, 0)	5
(0, 1, 0, 1, 0, 0, 0, 0, 1, 1)	5
(0, 1, 0, 1, 0, 1, 0, 0, 0, 1)	4
(0, 0, 1, 1, 1, 1, 0, 0, 0, 1)	6
(0, 1, 1, 0, 0, 0, 0, 1, 0, 1)	5
(0, 0, 0, 0, 0, 1, 0, 1, 0, 0)	3
(0, 0, 0, 0, 0, 1, 1, 1, 1, 1)	6
(0, 1, 1, 0, 0, 0, 1, 1, 1, 0)	6
(0, 1, 0, 1, 1, 0, 1, 1, 1, 0)	6
(0, 0, 1, 1, 0, 0, 0, 1, 1, 0)	4
(0, 0, 1, 1, 0, 1, 1, 0, 1, 0)	5
(0, 1, 0, 1, 1, 1, 1, 0, 0, 0)	5
(1, 1, 1, 1, 0, 0, 1, 1, 1, 0)	8
(1, 0, 0, 1, 1, 1, 1, 1, 0, 0)	6
(1, 0, 0, 1, 1, 0, 1, 0, 1, 0)	5
(1, 1, 1, 1, 0, 0, 0, 1, 0, 0)	5
(1, 1, 0, 0, 1, 1, 1, 0, 1, 1)	7
(1, 0, 1, 0, 0, 1, 1, 0, 0, 1)	5
(1, 0, 1, 0, 0, 0, 0, 1, 1, 1)	6
(1, 1, 0, 0, 1, 0, 1, 1, 0, 1)	6
(1, 0, 0, 1, 0, 1, 0, 1, 0, 1)	5
(1, 1, 1, 1, 1, 1, 0, 1, 1, 1)	9
(1, 1, 1, 1, 1, 0, 0, 0, 0, 1)	6
(1, 0, 0, 1, 0, 0, 0, 0, 0, 1)	4
(1, 0, 1, 0, 1, 1, 0, 0, 0, 0)	4
(1, 1, 0, 0, 0, 1, 0, 0, 1, 0)	4
(1, 1, 0, 0, 0, 0, 0, 0, 1, 0)	3
(1, 0, 1, 0, 1, 0, 0, 0, 1, 1)	5

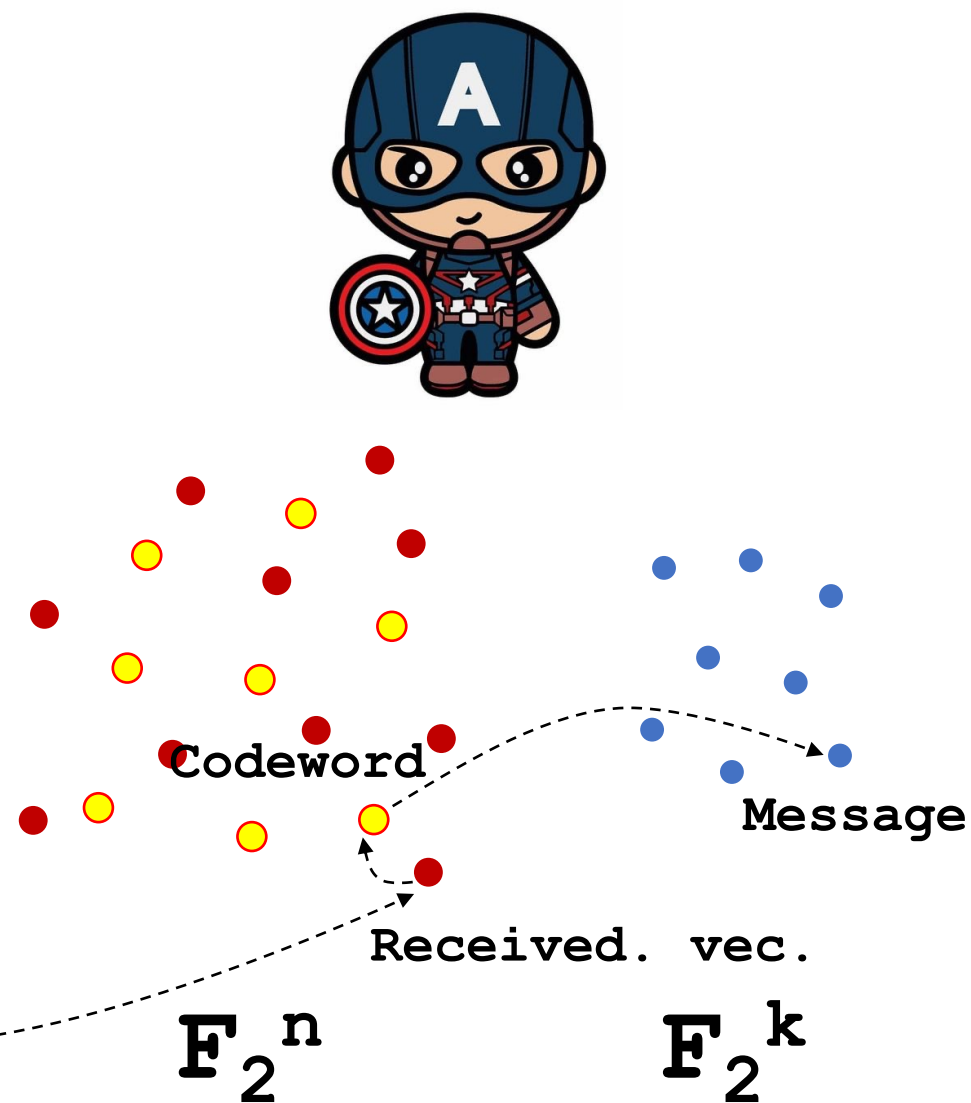
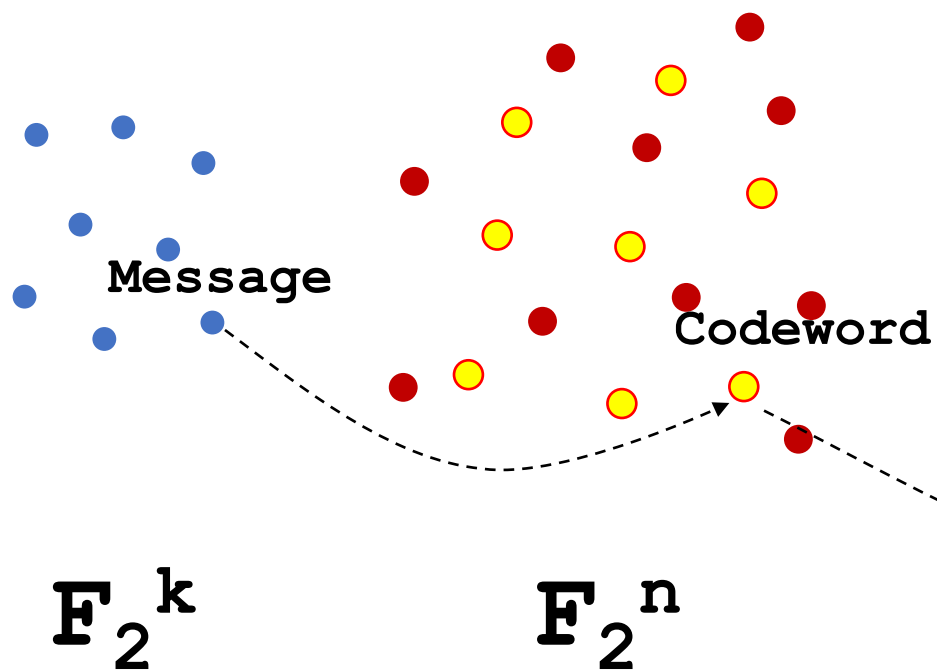
All codewords

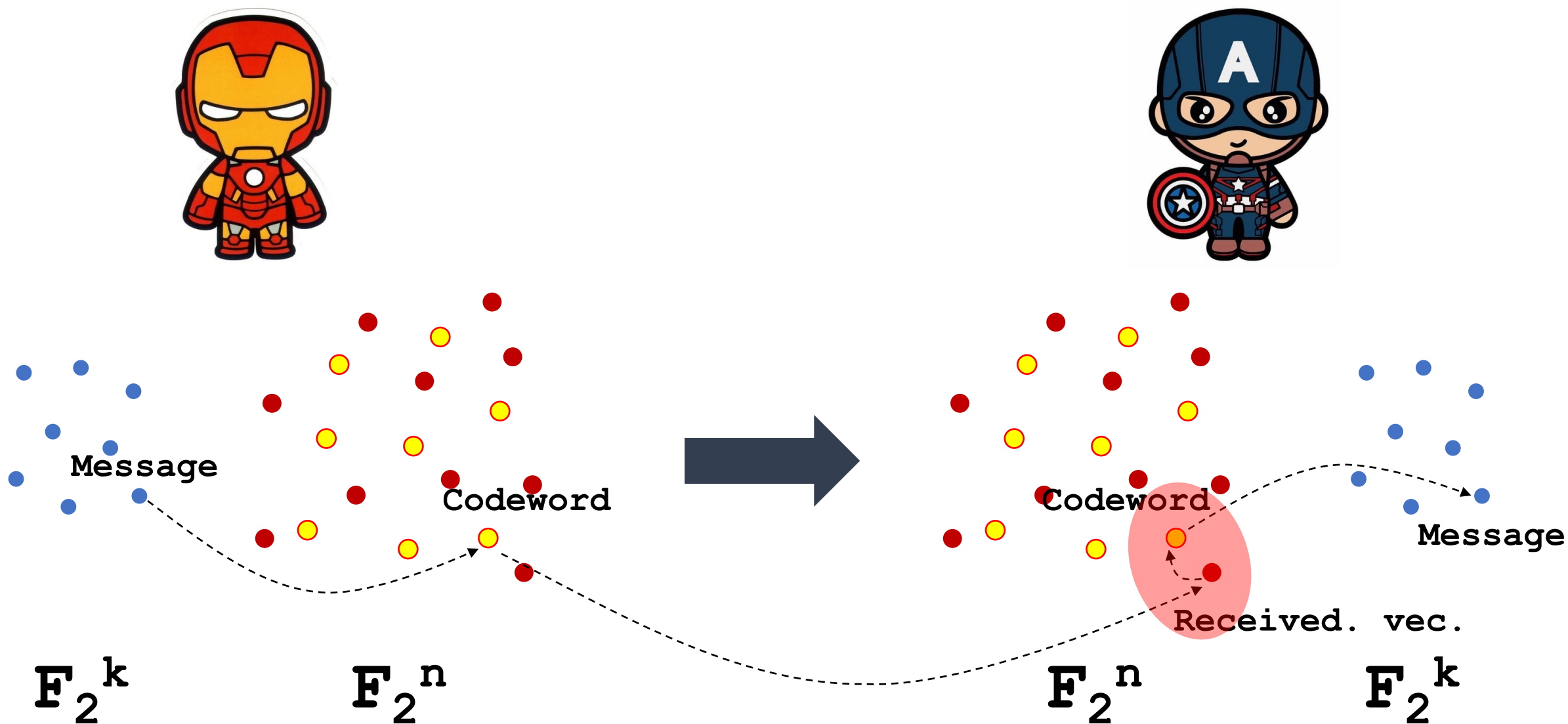


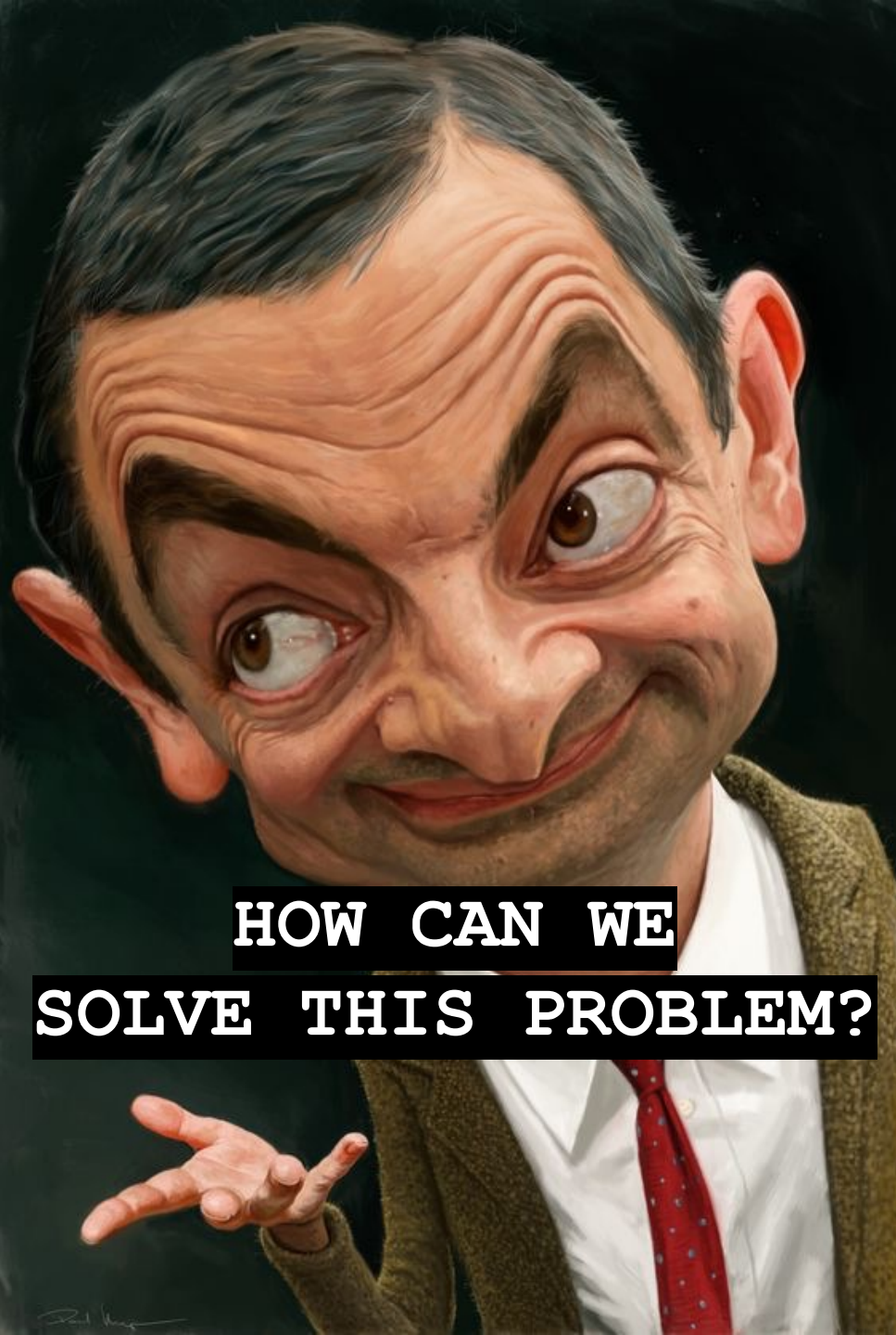
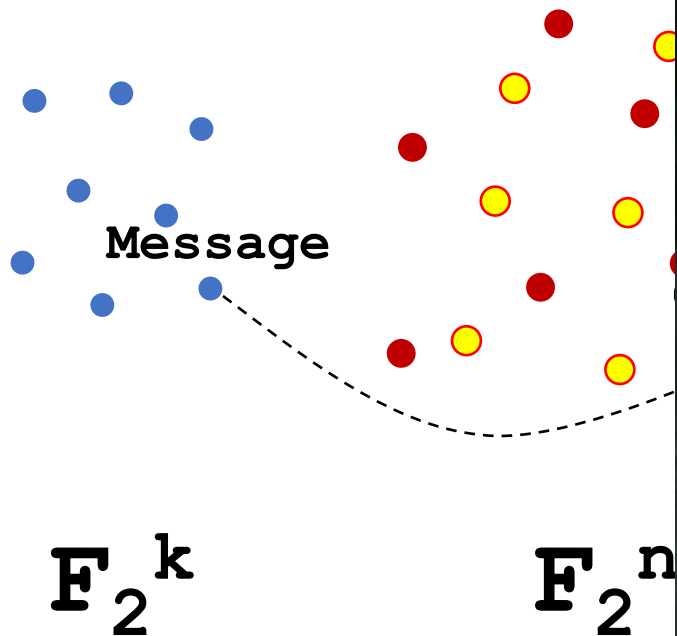




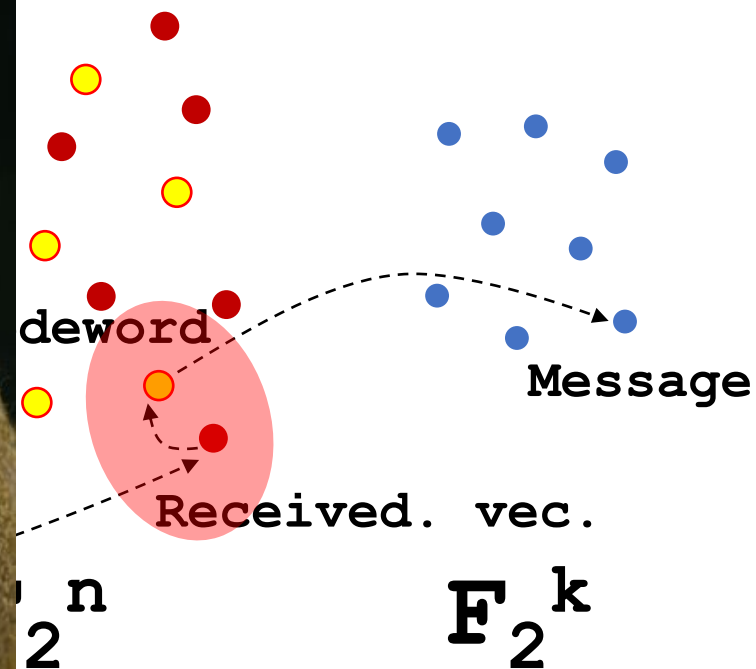








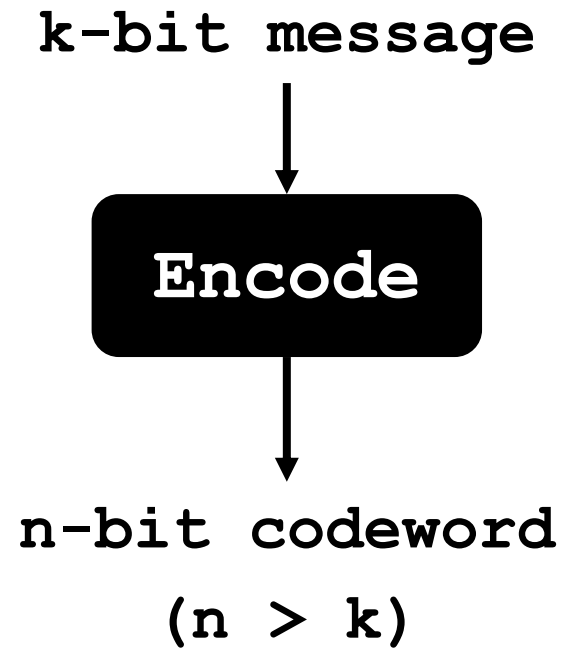
HOW CAN WE
SOLVE THIS PROBLEM?



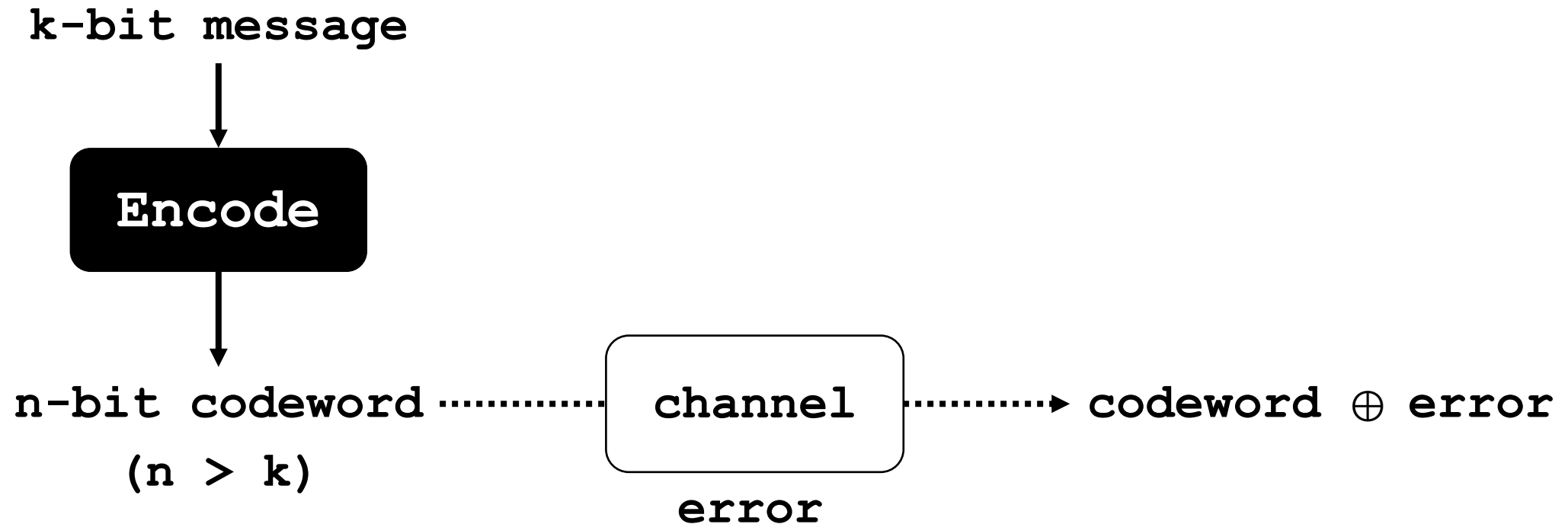
Error-Correcting Code

k -bit message

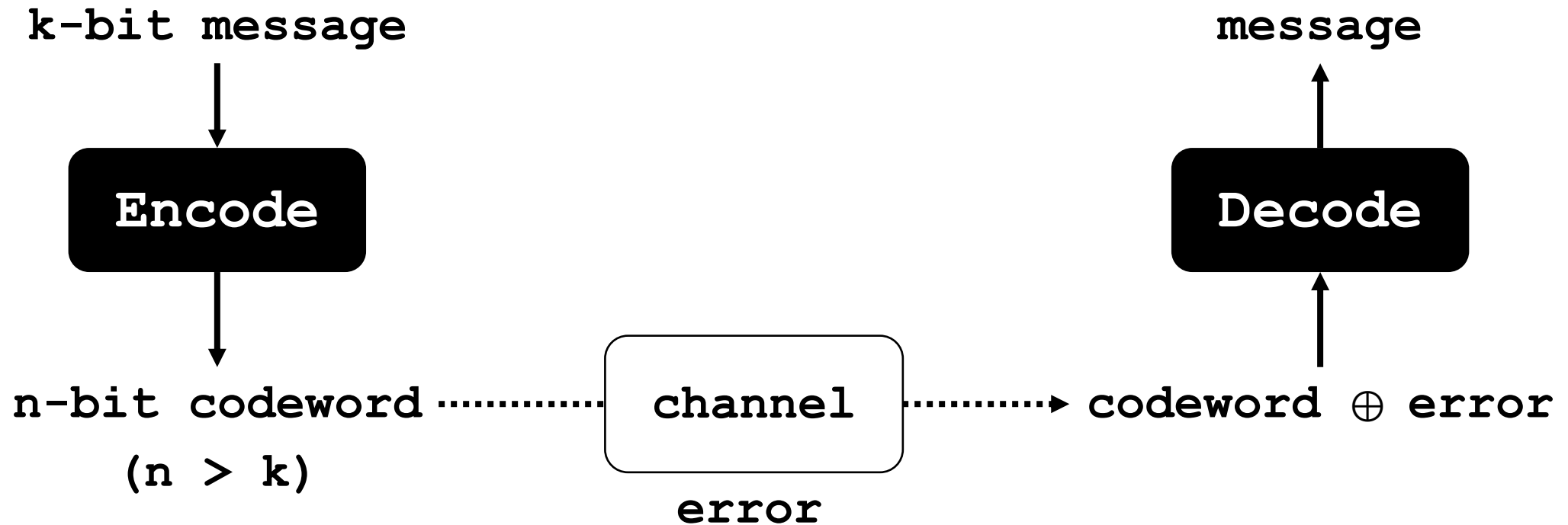
Error-Correcting Code



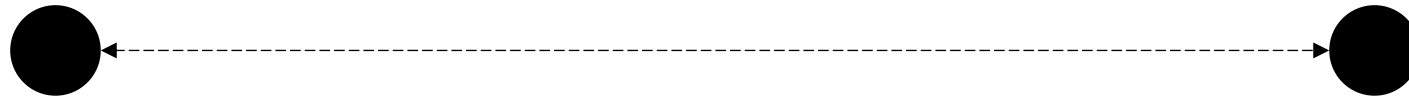
Error-Correcting Code



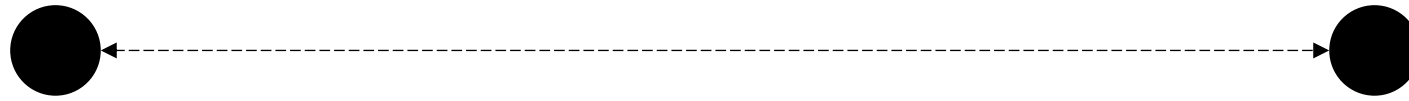
Error-Correcting Code



Distance between Two Vectors



Distance between Two Vectors



Hamming distance

$$d_H(\mathbf{x}, \mathbf{y}) := w_H(\mathbf{x} - \mathbf{y})$$

Hamming weight

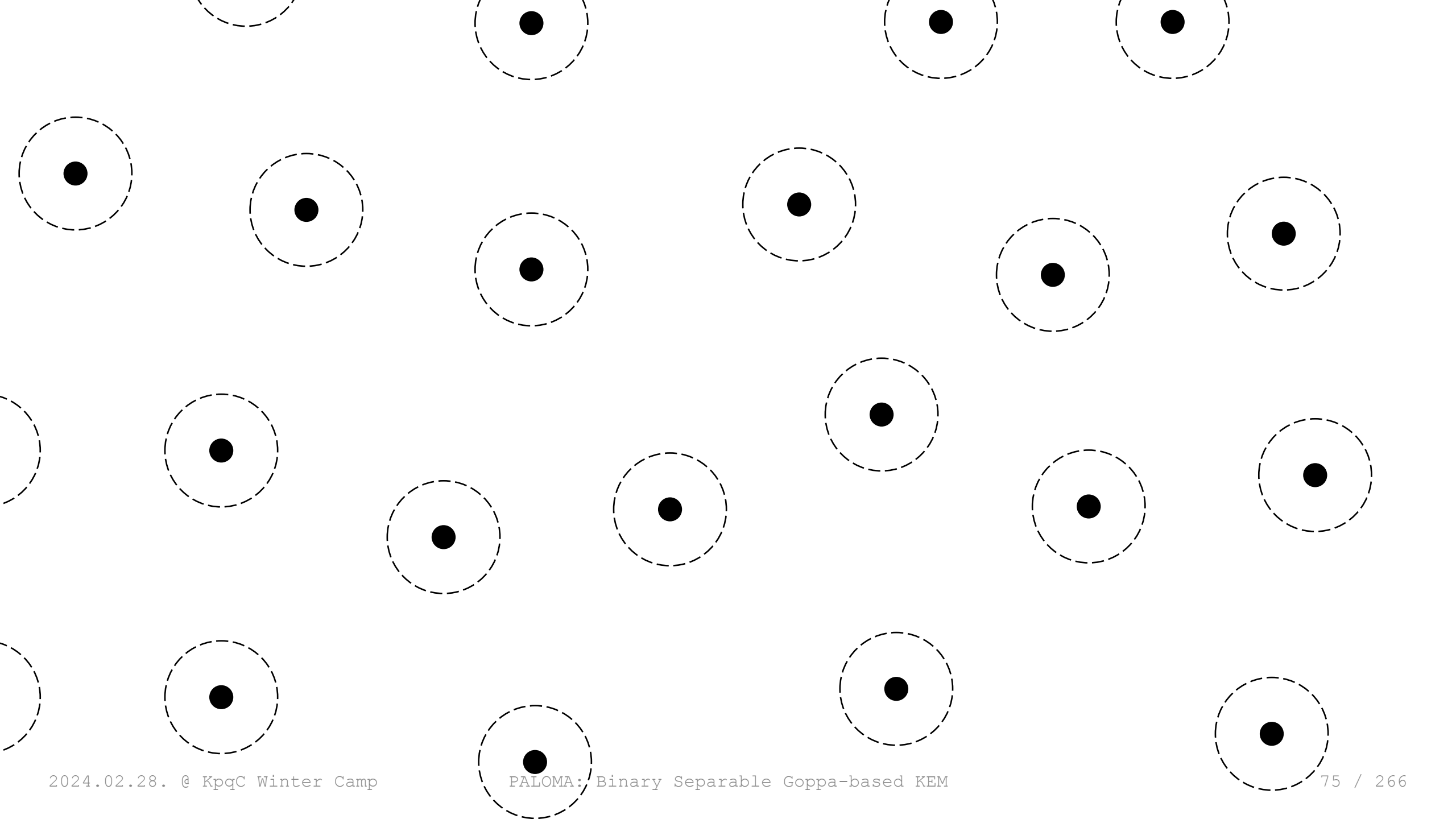
Distance between Two Vectors

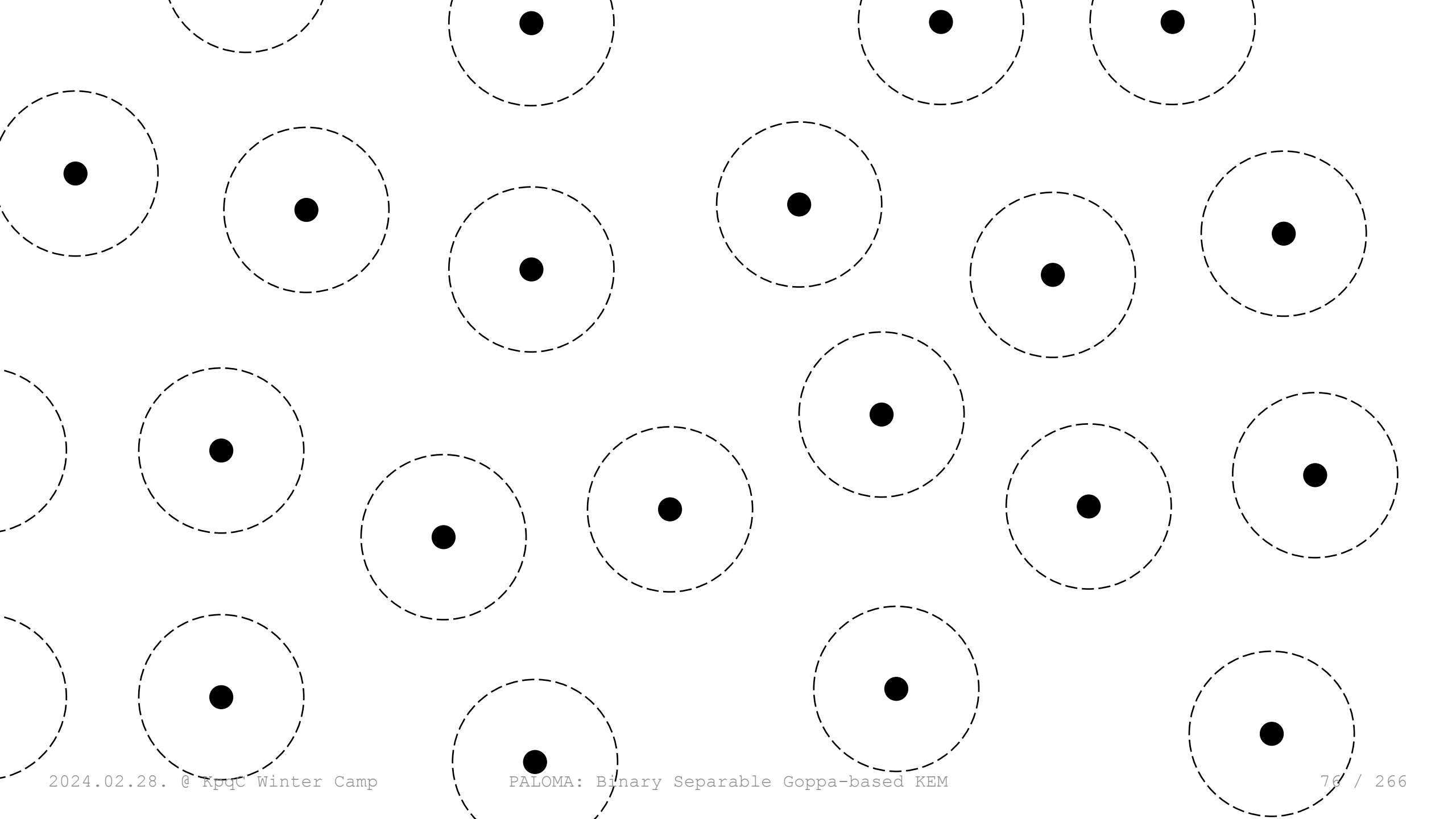
$$\mathbf{x} = (1, 0, 0, 1, 1, 0)$$

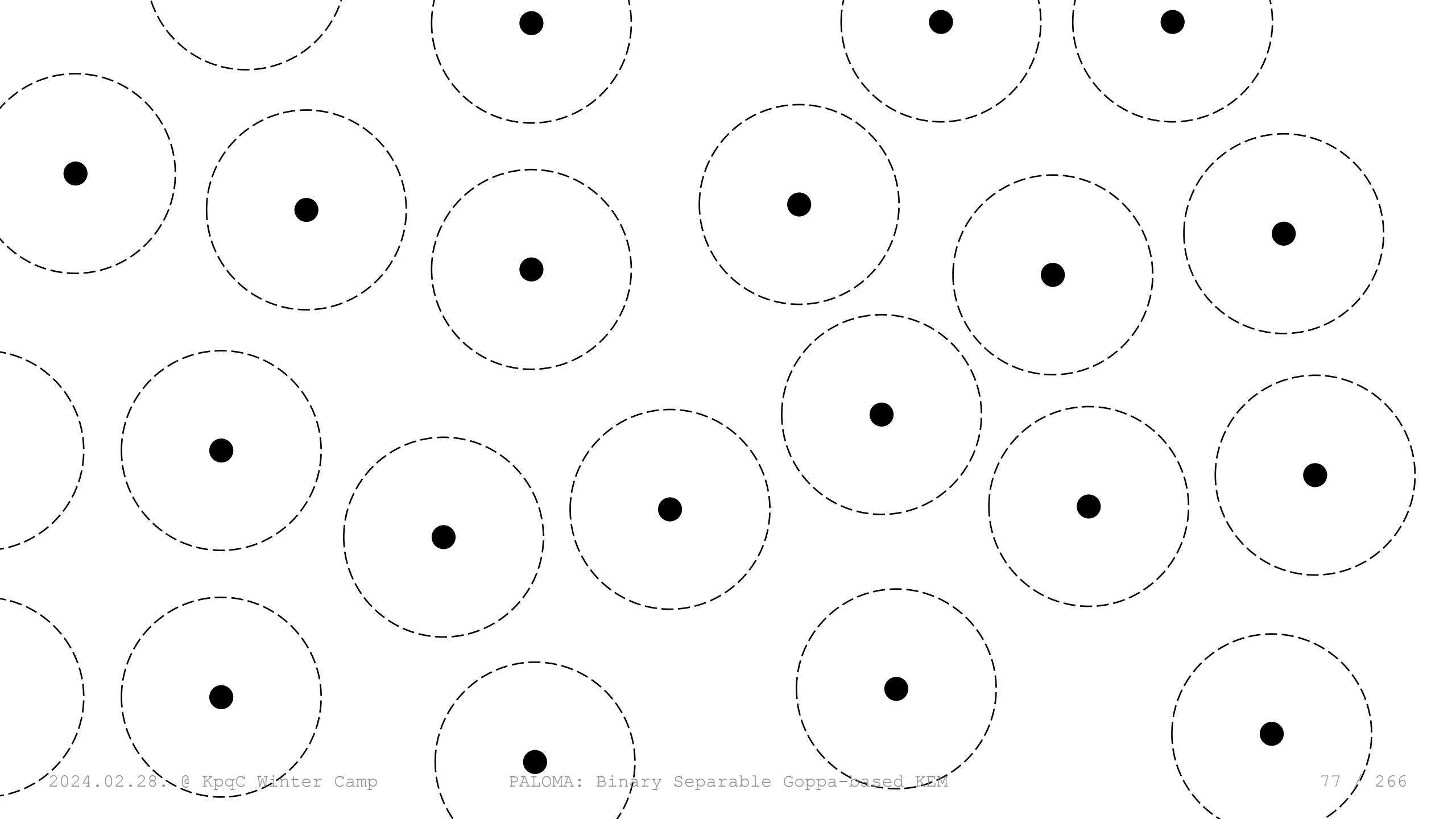
$$\mathbf{y} = (0, 1, 1, 1, 0, 0)$$

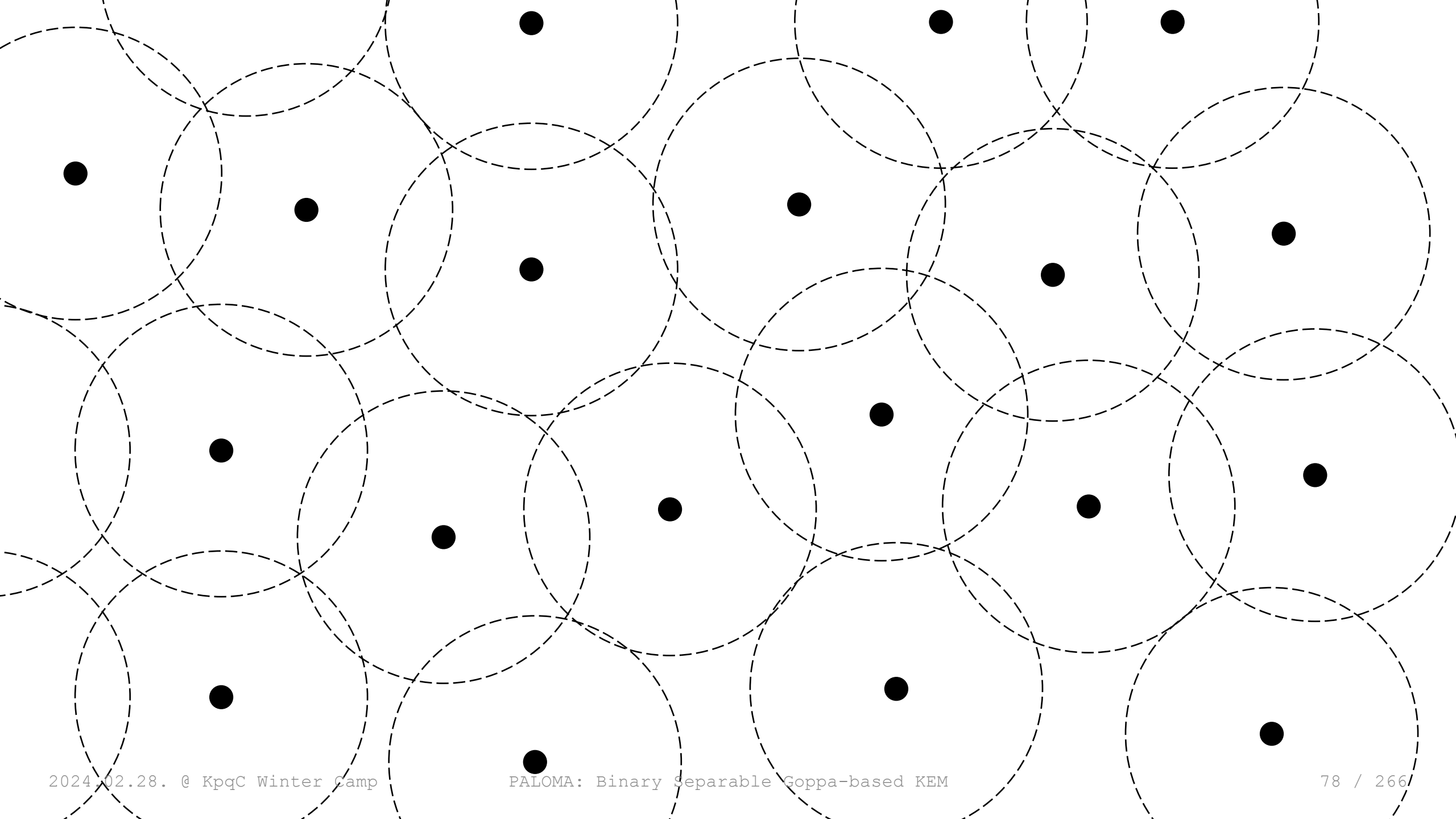
$$d_H(\mathbf{x}, \mathbf{y}) = 4$$

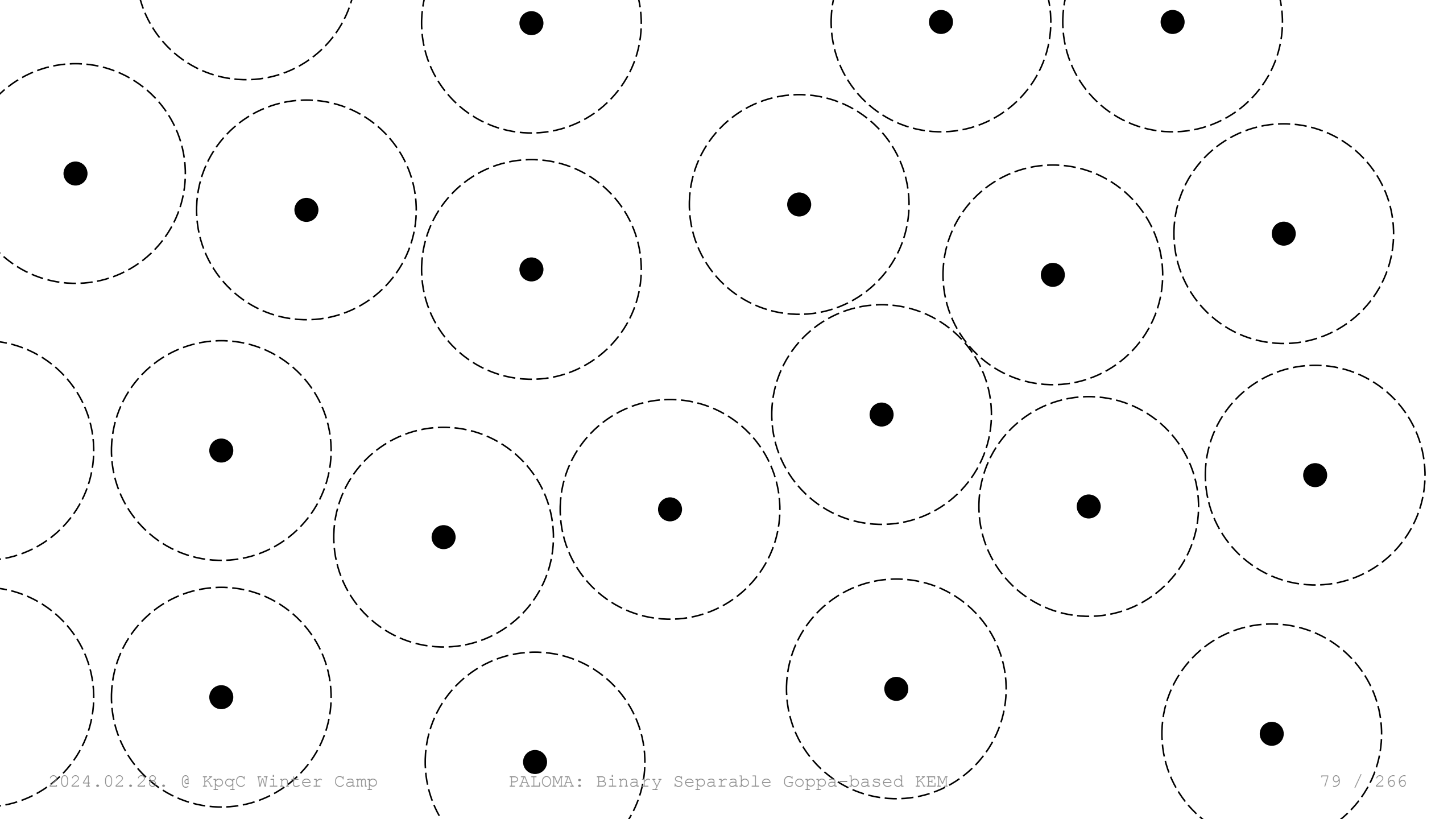
Codewords in \mathbb{F}_2^n

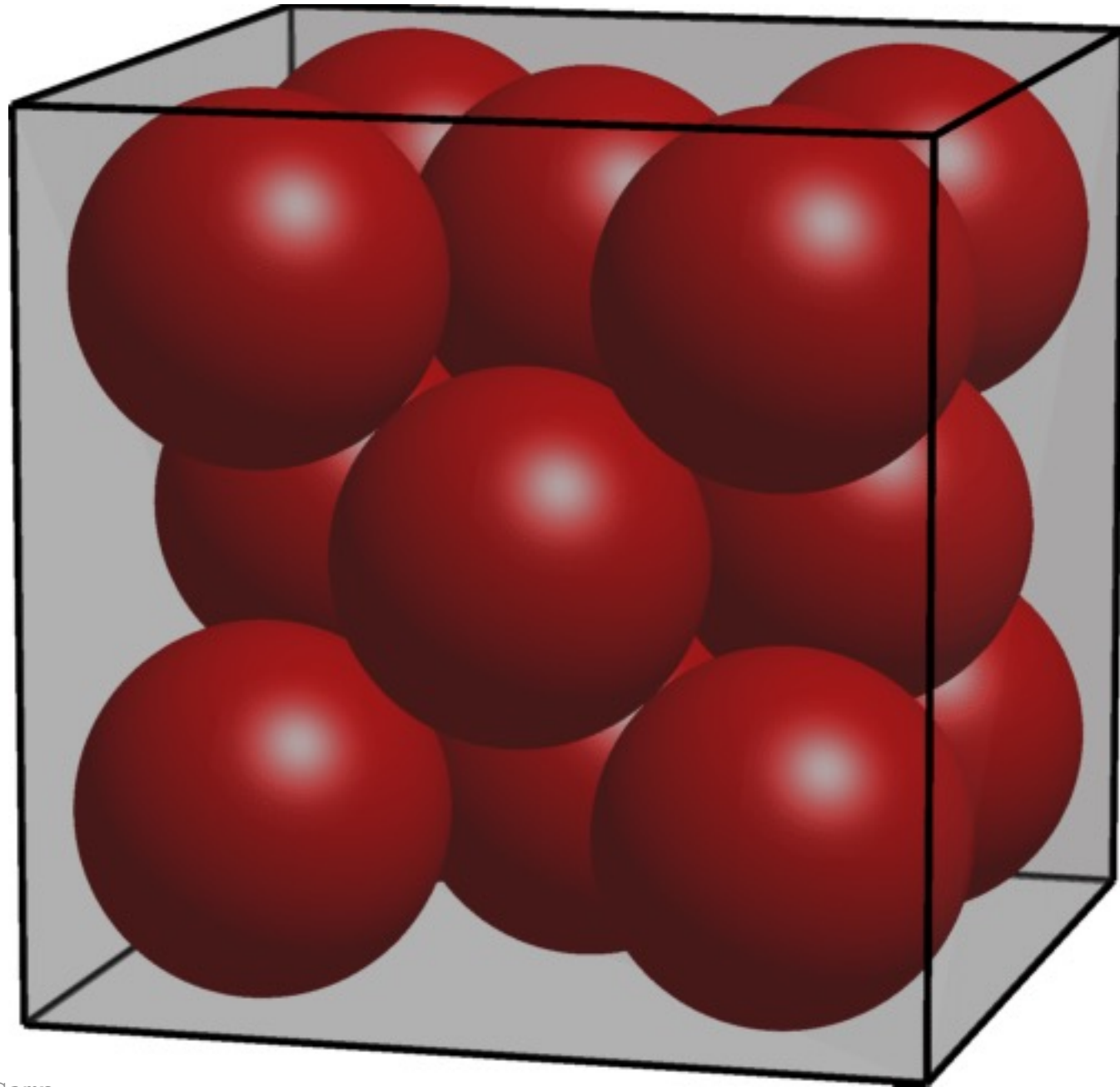


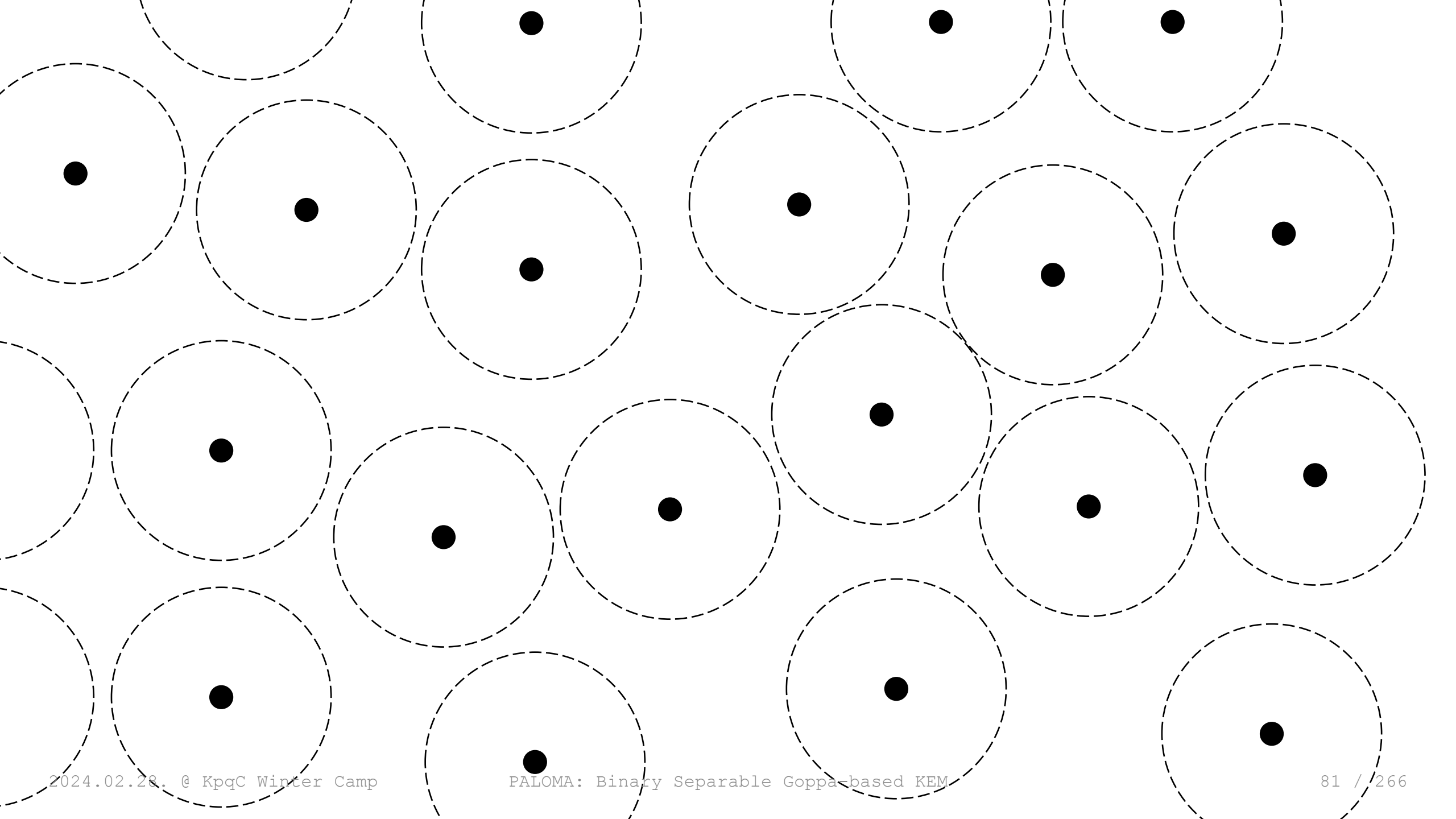


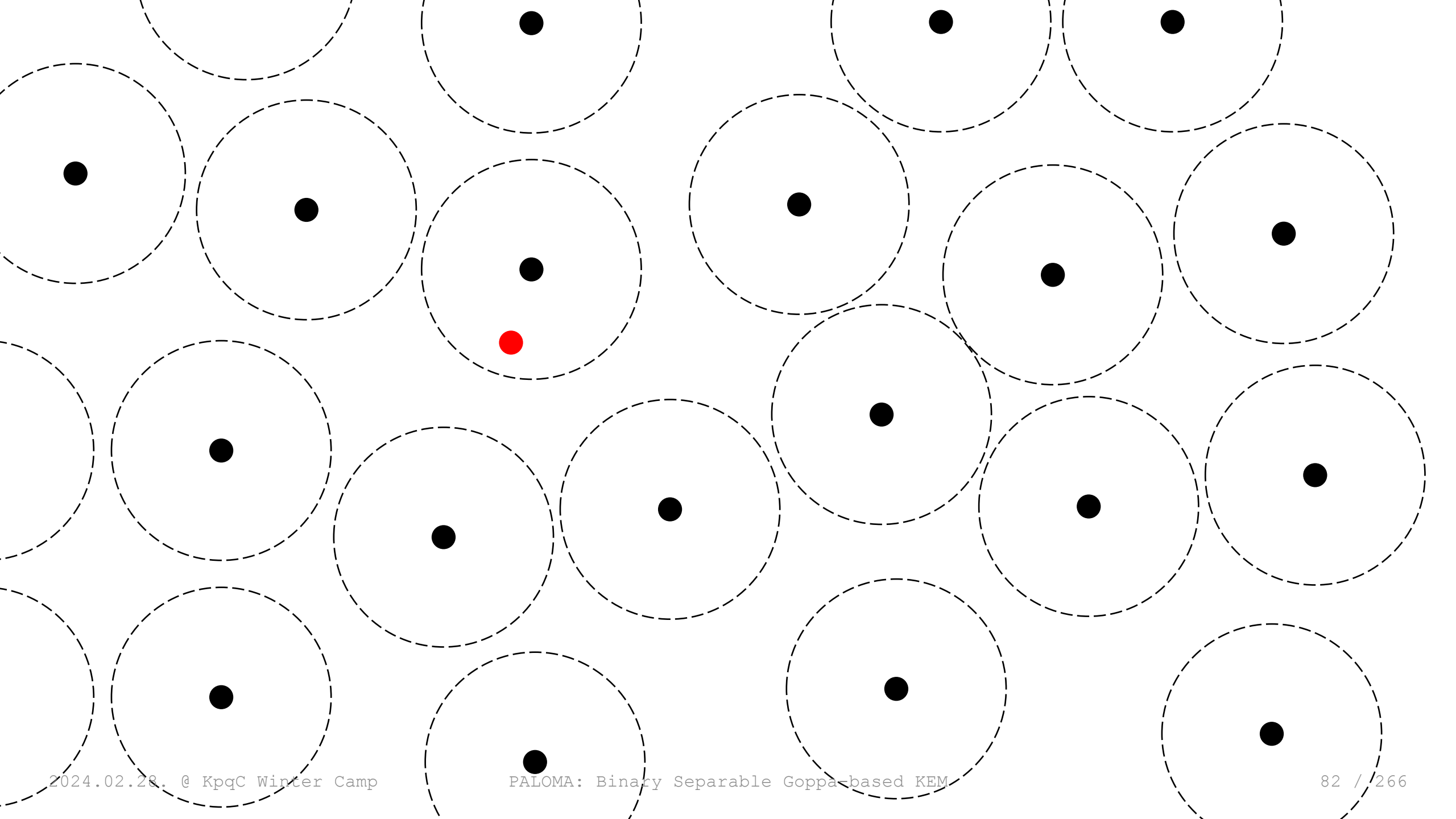


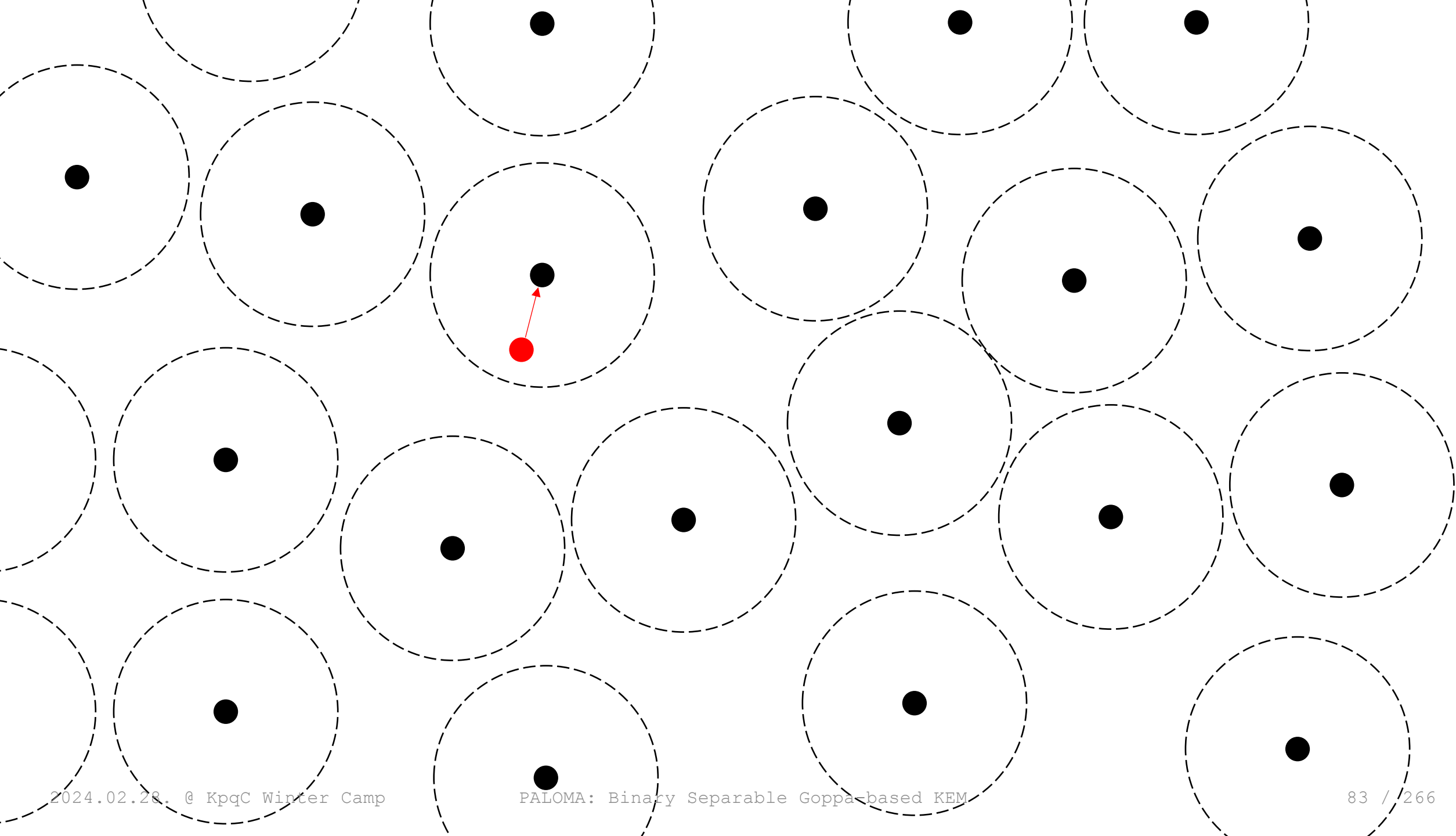


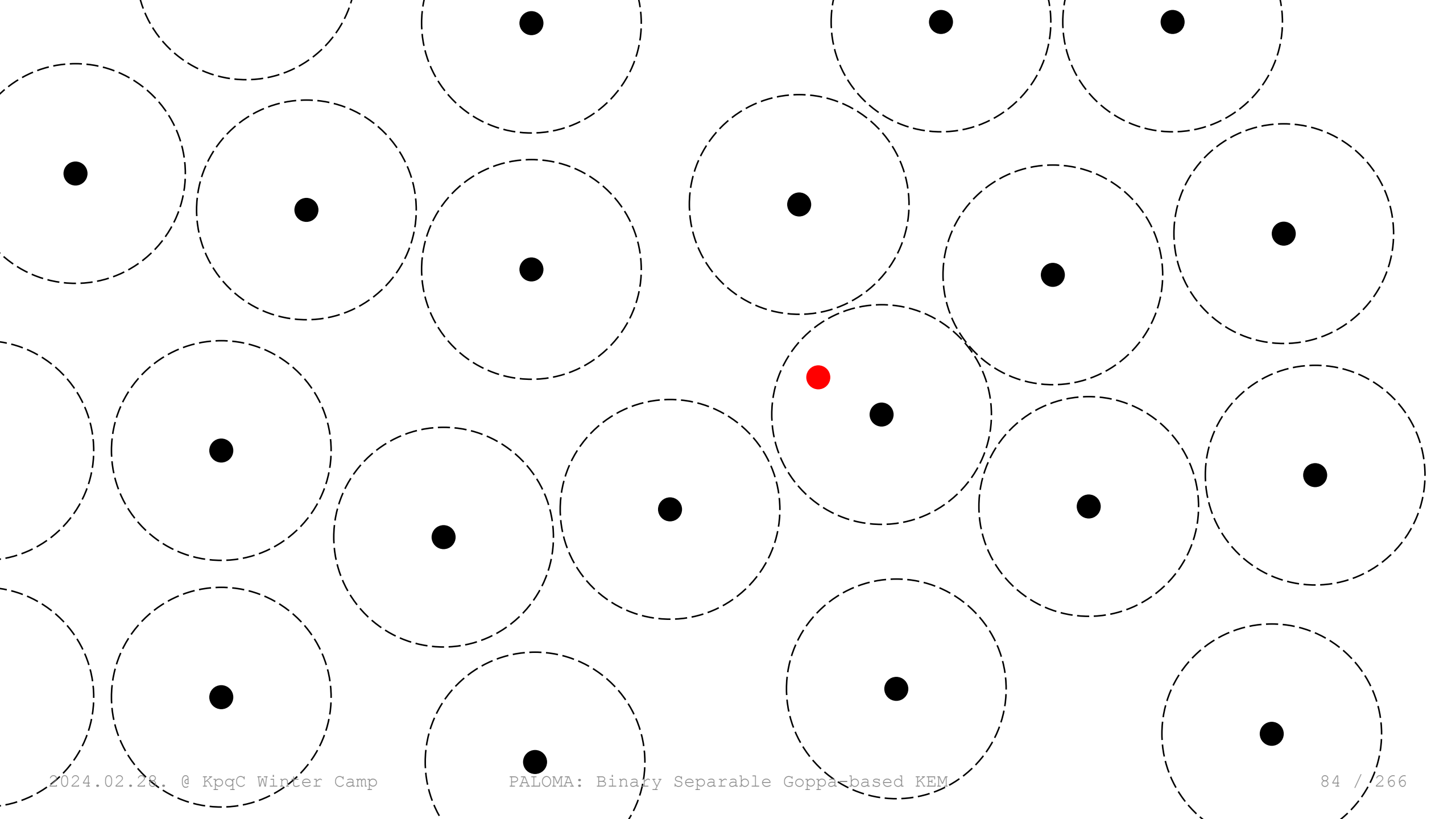


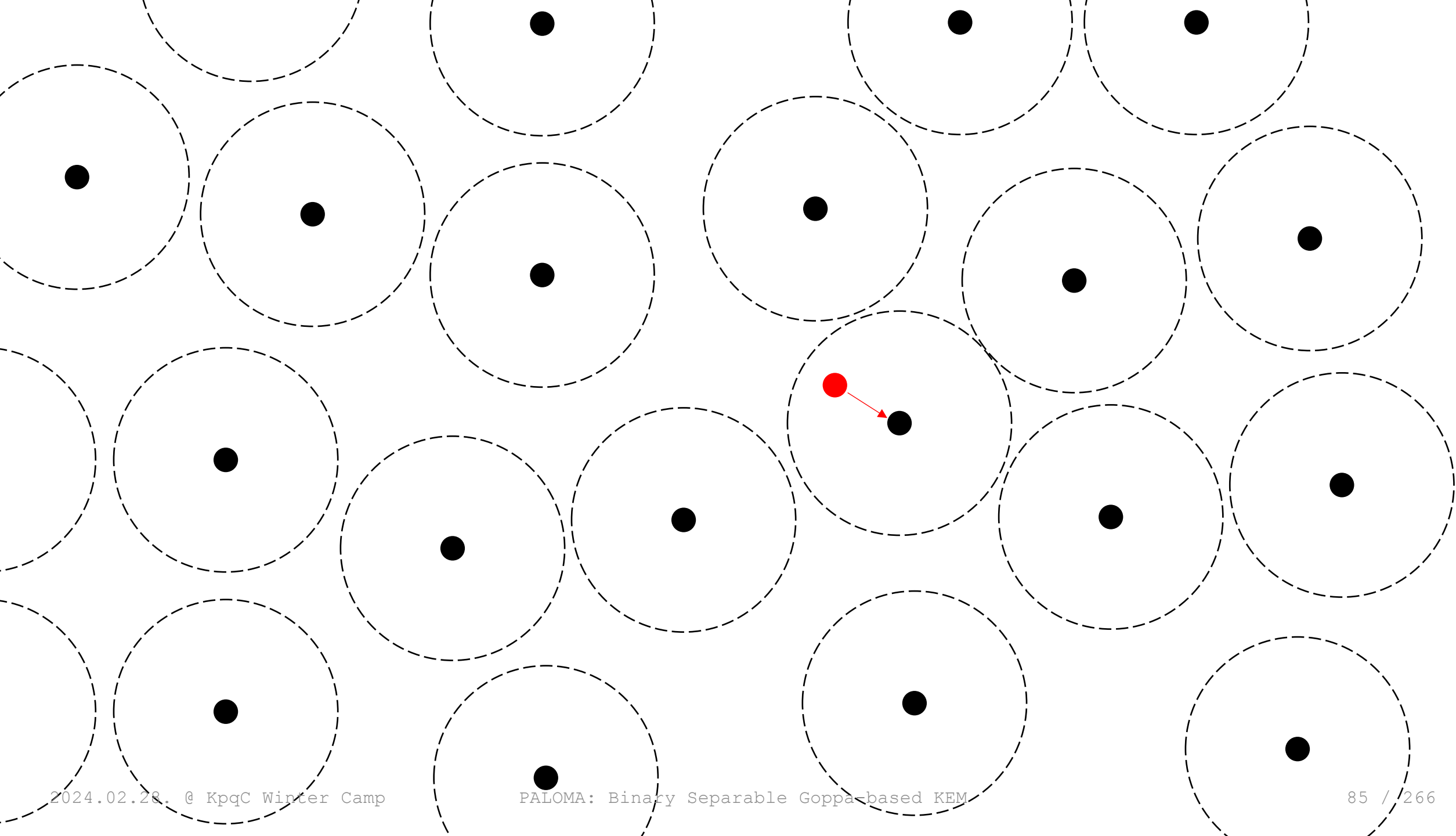


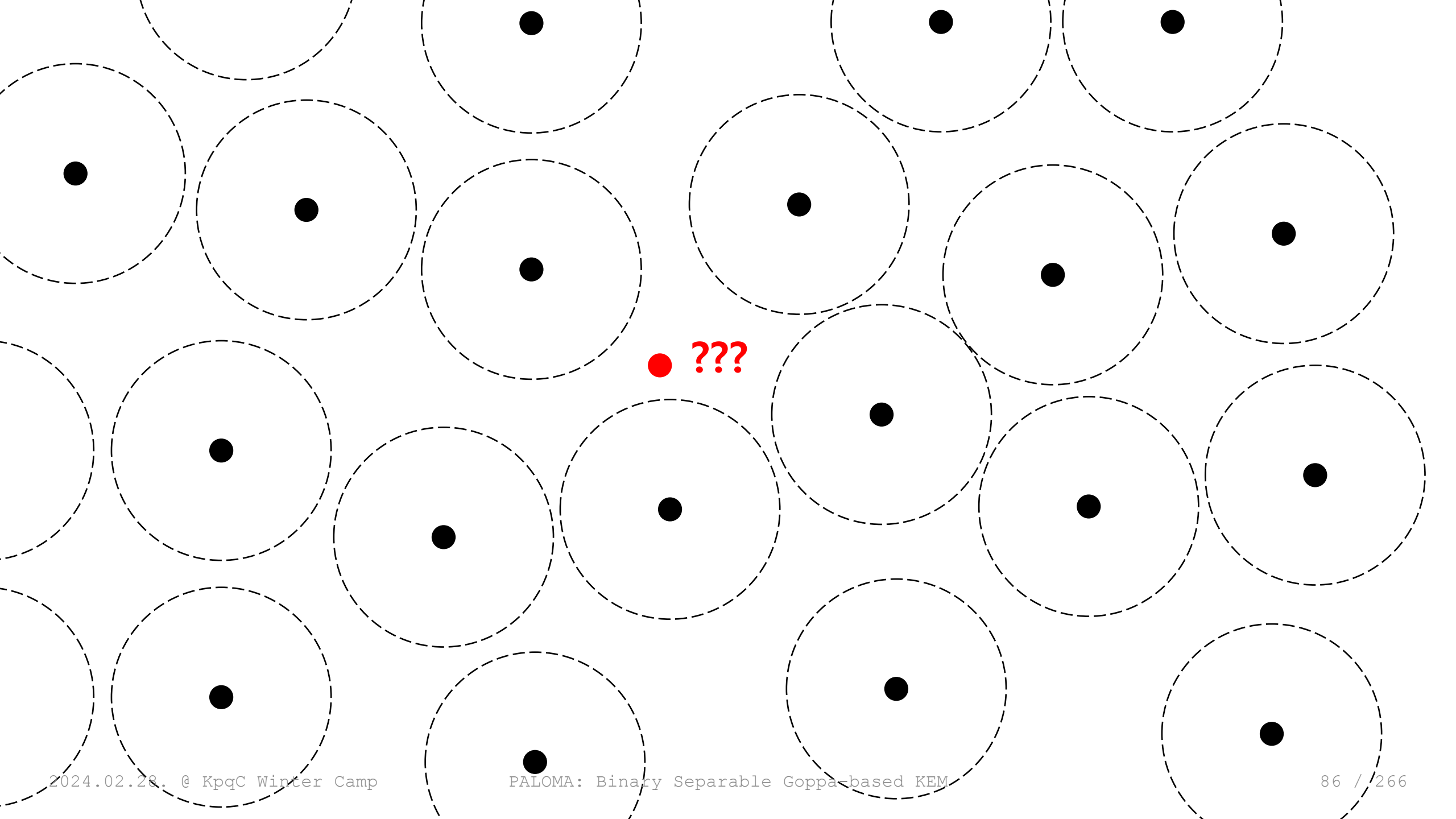


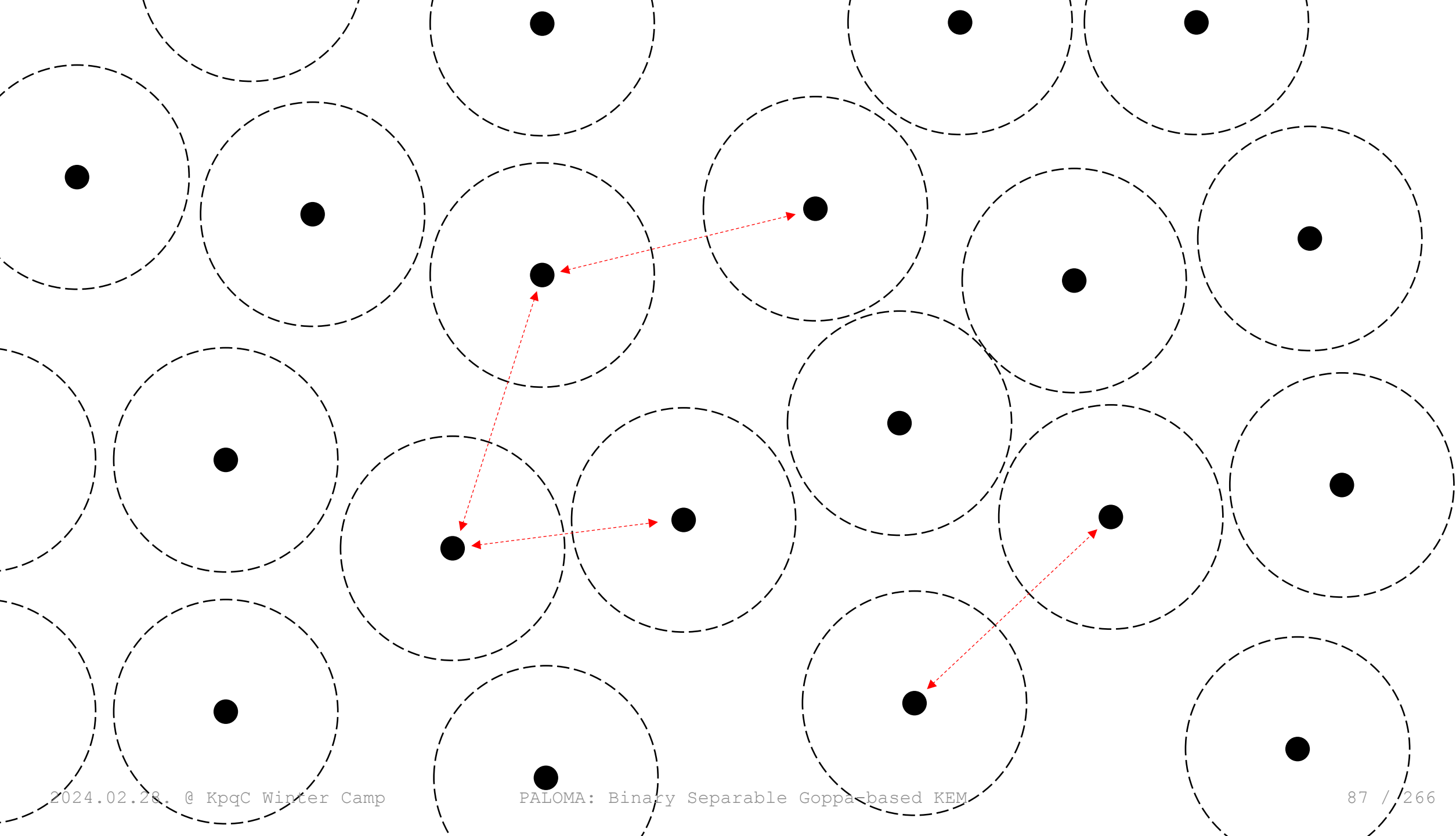




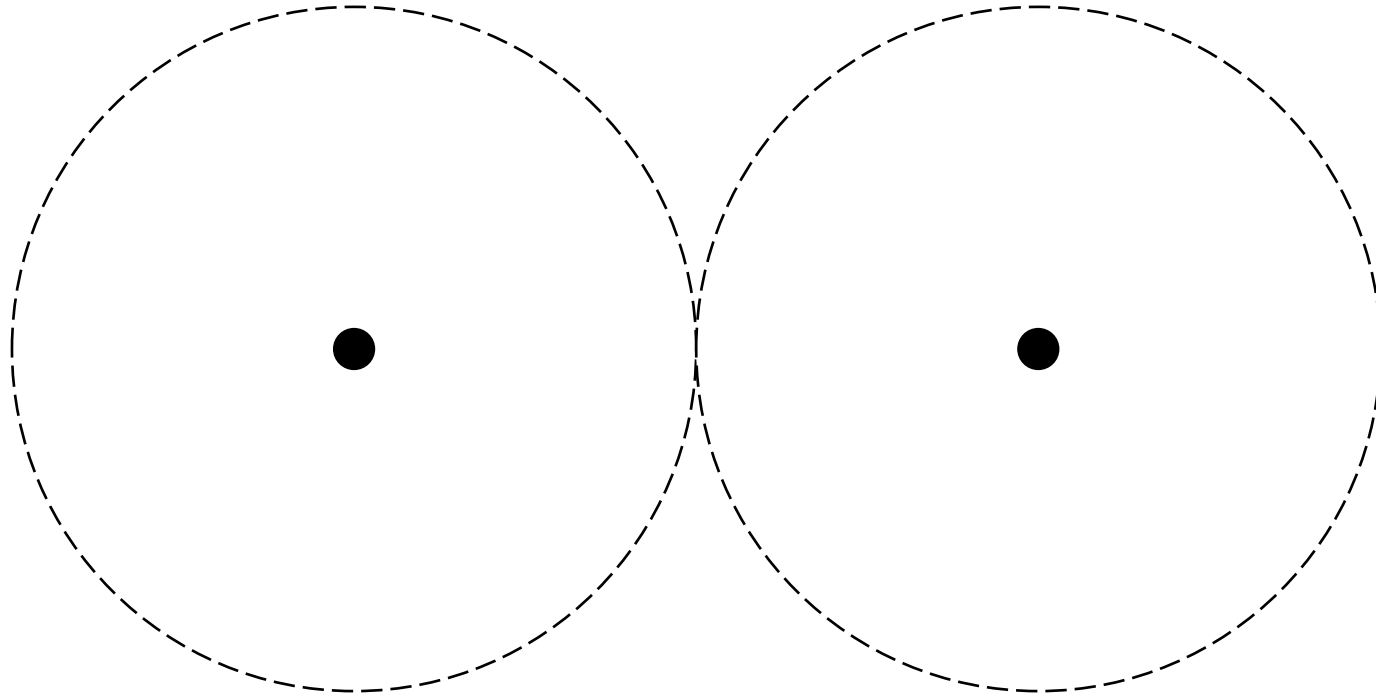




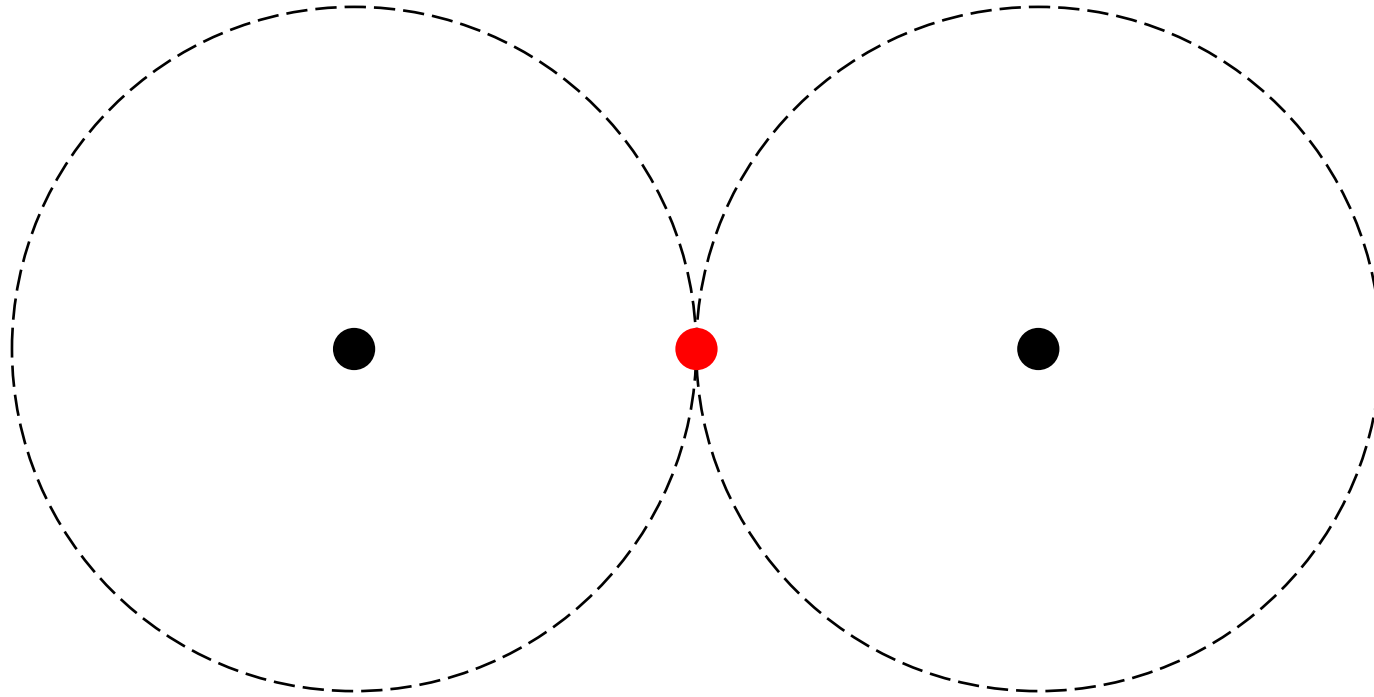




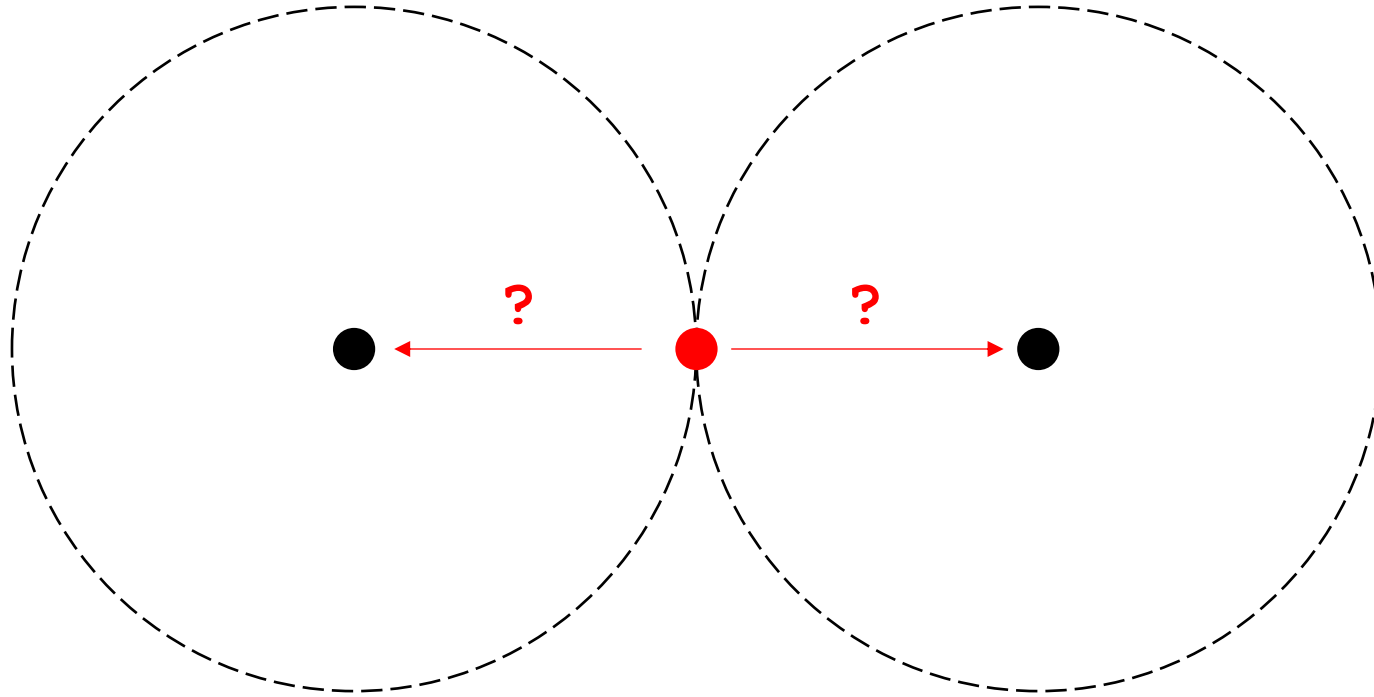
Linear Code $C = [n, k, d \geq 2t+1]_q$



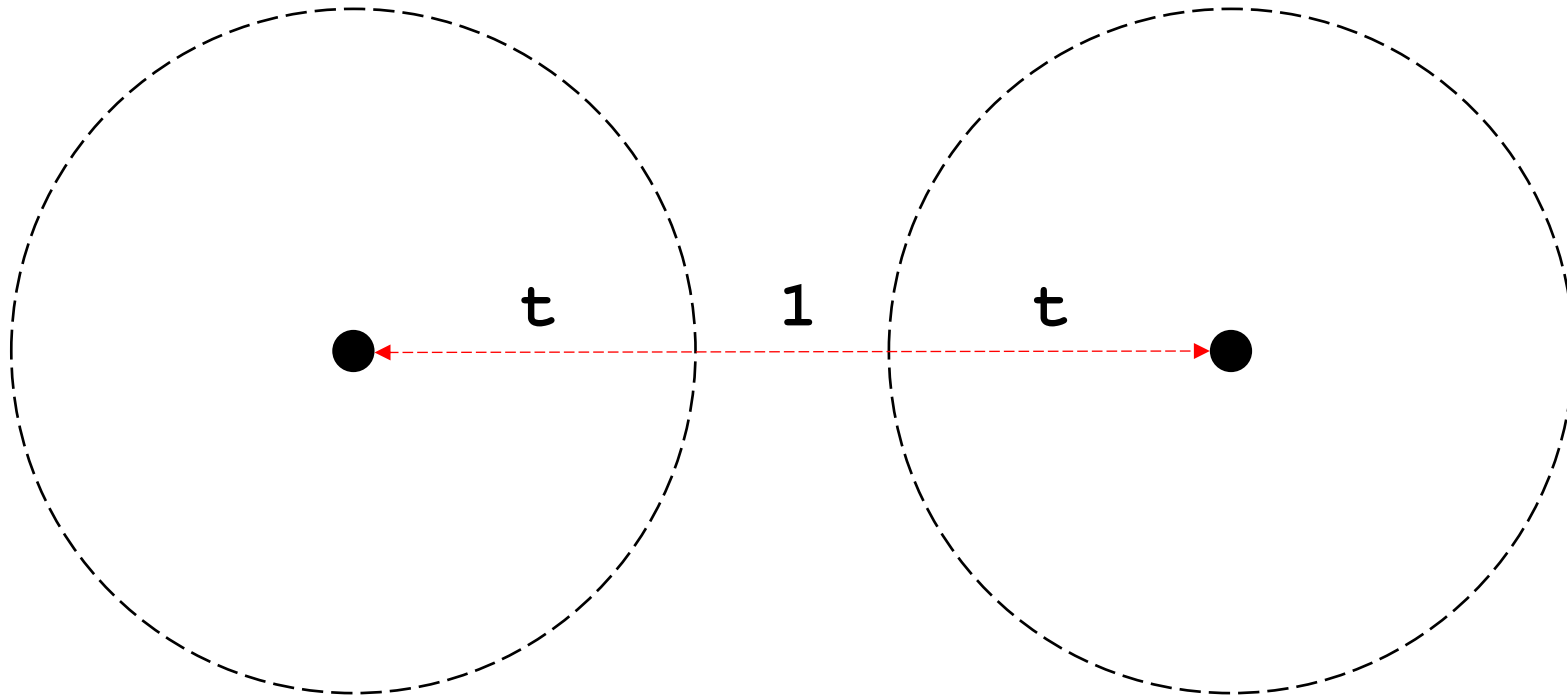
Linear Code $C = [n, k, d \geq 2t+1]_q$



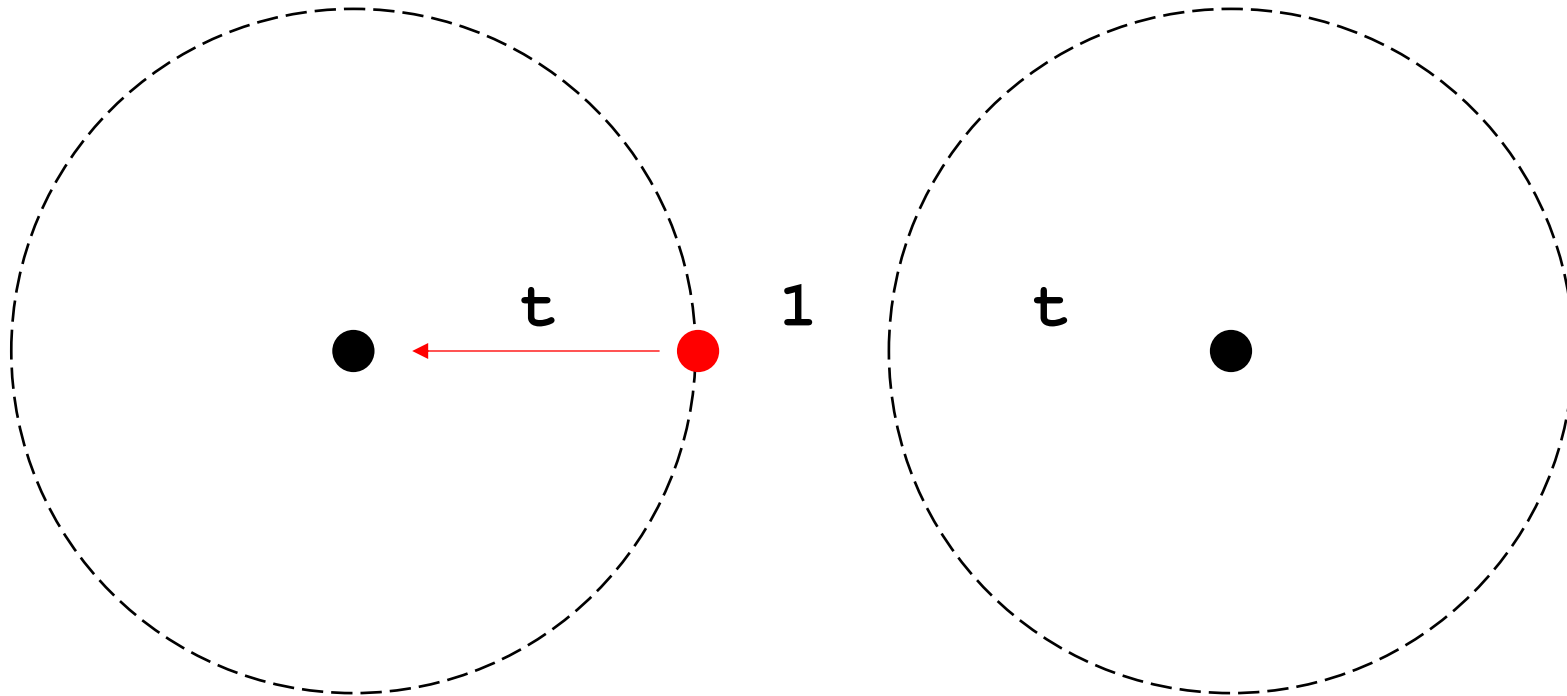
Linear Code $C = [n, k, d \geq 2t+1]_q$



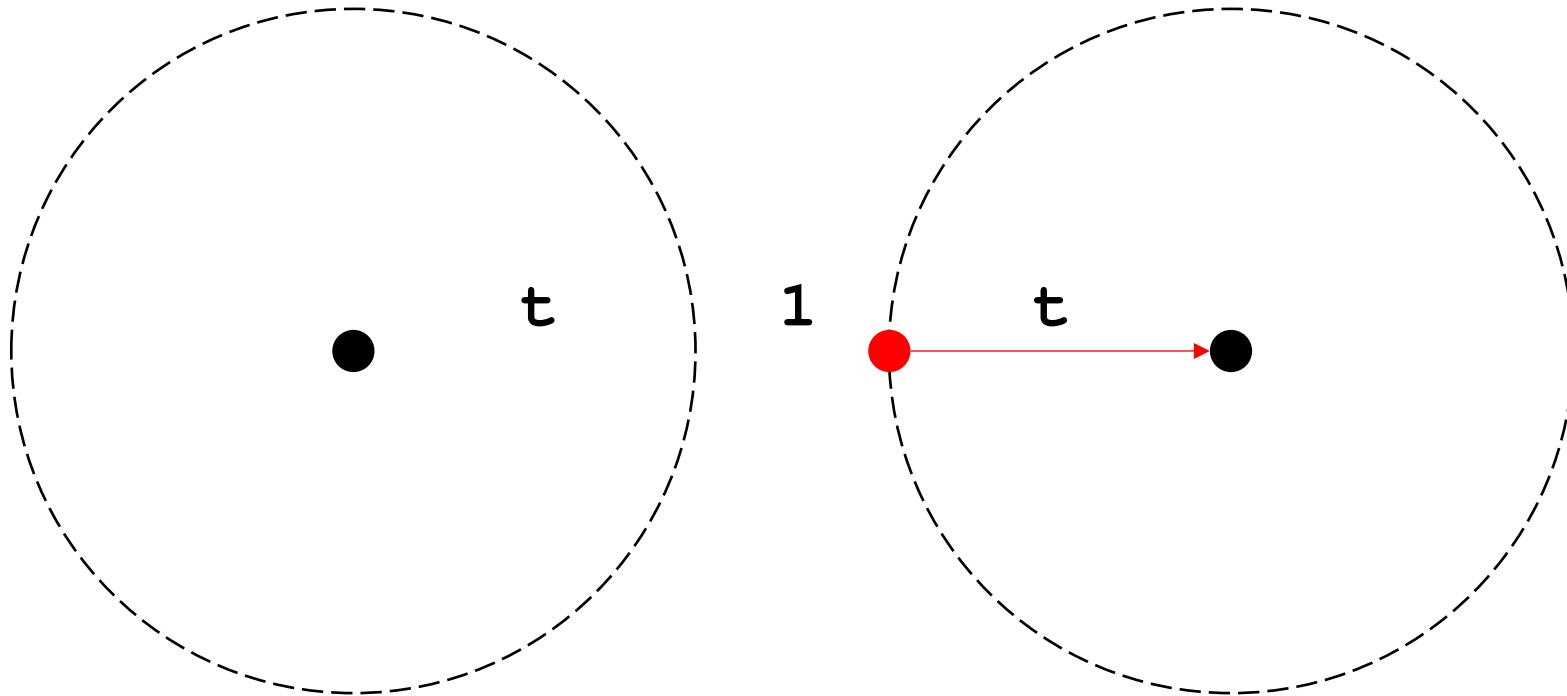
Linear Code $C = [n, k, d \geq 2t+1]_q$



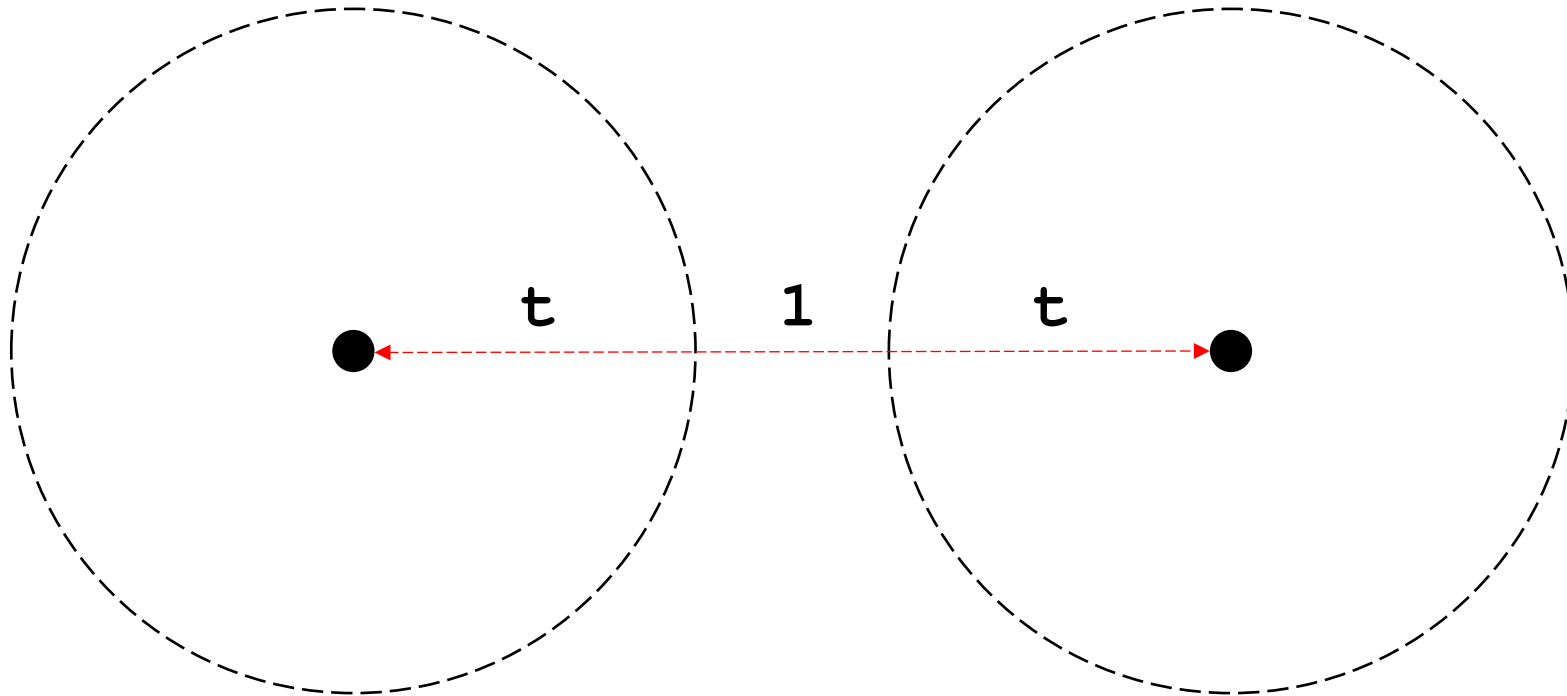
Linear Code $C = [n, k, d \geq 2t+1]_q$



Linear Code $C = [n, k, d \geq 2t+1]_q$



Linear Code $C = [n, k, d \geq 2t+1]_q$



Minimum distance

$$d = \min_{\substack{x, y \in C \\ x \neq y}} d_H(x, y) \geq 2t + 1$$

Linear Code $C = [n, k, d \geq 2t+1]_q$

length = n

dimension = k

t -error correctable

defined over F_q

Linear Code $C = [10, 5, 3 \geq 2 \cdot 1 + 1]_2$

[1	0	0	1	1	1	0	0	1	1]
[1	0	0	1	0	1	0	0	0	0]
[1	0	0	1	1	1	1	0	0	0]
[0	1	1	0	0	1	0	0	1	1]
[1	1	0	1	0	1	1	0	0	1]

Parity-check matrix H

[c0]
[c1]
[c2]
[c3]
[c4]
[c5]
[c6]
[c7]
[c8]
[c9]

Codeword

[0]
[0]
[0]
[0]
[0]

(0, 0, 0, 0, 0, 0, 0, 0, 0, 0)	0
(0, 1, 1, 0, 1, 0, 0, 0, 1, 0)	4
(0, 1, 1, 0, 1, 1, 0, 1, 0, 0)	5
(0, 0, 0, 0, 0, 1, 0, 1, 1, 0)	3
(0, 0, 1, 1, 1, 0, 0, 1, 0, 1)	5
(0, 1, 0, 1, 0, 0, 0, 1, 1, 1)	5
(0, 1, 0, 1, 0, 1, 0, 0, 0, 1)	4
(0, 0, 1, 1, 1, 1, 0, 0, 0, 1)	6
(0, 1, 1, 0, 0, 0, 1, 0, 1, 1)	5
(0, 0, 0, 0, 1, 0, 1, 0, 1, 0)	3
(0, 0, 0, 0, 1, 1, 1, 1, 1, 1)	6
(0, 1, 1, 0, 0, 1, 1, 1, 0, 1)	6
(0, 1, 0, 1, 1, 0, 1, 1, 1, 0)	6
(0, 0, 1, 1, 0, 0, 1, 1, 0, 0)	4
(0, 0, 1, 1, 0, 1, 1, 0, 1, 0)	5
(0, 1, 0, 1, 1, 1, 1, 0, 0, 0)	5
(1, 1, 1, 1, 0, 1, 1, 1, 1, 0)	8
(1, 0, 0, 1, 1, 1, 1, 1, 0, 0)	6
(1, 0, 0, 1, 1, 0, 1, 0, 1, 0)	5
(1, 1, 1, 1, 0, 0, 1, 0, 0, 0)	5
(1, 1, 0, 0, 1, 1, 1, 0, 1, 1)	7
(1, 0, 1, 0, 0, 1, 1, 0, 0, 1)	5
(1, 0, 1, 0, 0, 0, 1, 1, 1, 1)	6
(1, 1, 0, 0, 1, 0, 1, 1, 0, 1)	6
(1, 0, 0, 1, 0, 1, 0, 1, 0, 1)	5
(1, 1, 1, 1, 1, 1, 0, 1, 1, 1)	9
(1, 1, 1, 1, 1, 0, 0, 0, 0, 1)	6
(1, 0, 0, 1, 0, 0, 0, 0, 0, 1)	4
(1, 0, 1, 0, 1, 1, 0, 0, 0, 0)	4
(1, 1, 0, 0, 0, 1, 0, 0, 1, 0)	4
(1, 1, 0, 0, 0, 0, 0, 1, 0, 0)	3
(1, 0, 1, 0, 1, 0, 0, 1, 1, 0)	5

All codewords96 / 266

Linear Code $C = [10, 5, 3 \geq 2 \cdot 1 + 1]_2$

[1	0	0	1	1	1	0	0	1	1]
[1	0	0	1	0	1	0	0	0	0]
[1	0	0	1	1	1	1	0	0	0]
[0	1	1	0	0	1	0	0	1	1]
[1	1	0	1	0	1	1	0	0	1]

Parity-check matrix H

[0]
[0]
[0]
[0]
[0]
[1]
[0]
[1]
[1]
[0]

Codeword

=

[0]
[0]
[0]
[0]
[0]

(0, 0, 0, 0, 0, 0, 0, 0, 0, 0)	0
(0, 1, 1, 0, 1, 0, 0, 0, 1, 0)	4
(0, 1, 1, 0, 1, 1, 0, 1, 0, 0)	5
(0, 0, 0, 0, 0, 1, 0, 1, 1, 0)	3
(0, 1, 0, 1, 0, 1, 0, 0, 0, 1)	4
(0, 0, 1, 1, 1, 1, 0, 1, 0, 1)	6
(0, 1, 1, 0, 0, 0, 1, 0, 1, 1)	5
(0, 0, 0, 0, 1, 1, 1, 1, 1, 1)	6
(0, 1, 1, 0, 0, 1, 1, 1, 0, 1)	6
(0, 1, 0, 1, 1, 0, 1, 1, 1, 0)	6
(0, 0, 1, 1, 0, 0, 1, 1, 0, 0)	4
(0, 0, 1, 1, 0, 1, 1, 0, 1, 0)	5
(0, 1, 0, 1, 1, 1, 1, 0, 0, 0)	5
(1, 1, 1, 1, 0, 1, 1, 1, 1, 0)	8
(1, 0, 0, 1, 1, 1, 1, 1, 0, 0)	6
(1, 0, 0, 1, 1, 0, 1, 0, 1, 0)	5
(1, 1, 1, 1, 0, 0, 1, 0, 0, 0)	5
(1, 1, 0, 0, 1, 1, 1, 0, 1, 1)	7
(1, 0, 1, 0, 0, 1, 1, 0, 0, 1)	5
(1, 0, 1, 0, 0, 0, 1, 1, 1, 1)	6
(1, 1, 0, 0, 1, 0, 1, 1, 0, 1)	6
(1, 0, 0, 1, 0, 1, 0, 1, 0, 1)	5
(1, 1, 1, 1, 1, 1, 0, 1, 1, 1)	9
(1, 1, 1, 1, 1, 0, 0, 0, 0, 1)	6
(1, 0, 0, 1, 0, 0, 0, 0, 1, 1)	4
(1, 0, 1, 0, 1, 1, 0, 0, 0, 0)	4
(1, 1, 0, 0, 0, 1, 0, 0, 1, 0)	4
(1, 1, 0, 0, 0, 0, 0, 1, 0, 0)	3
(1, 0, 1, 0, 1, 0, 0, 1, 1, 0)	5

All codewords97 / 266

Linear Code $C = [10, 5, 3 \geq 2 \cdot 1 + 1]_2$

[1	0	0	1	1	1	0	0	1	1]
[1	0	0	1	0	1	0	0	0	0]
[1	0	0	1	1	1	1	0	0	0]
[0	1	1	0	0	1	0	0	1	1]
[1	1	0	1	0	1	1	0	0	1]

Parity-check matrix H

[1]
[0]
[0]
[0]
[0]
[1]
[0]
[1]
[1]
[0]

Received. vec.

(0, 0, 0, 0, 0, 0, 0, 0, 0, 0)	0
(0, 1, 1, 0, 1, 0, 0, 0, 1, 0)	4
(0, 1, 1, 0, 1, 1, 0, 1, 0, 0)	5
(0, 0, 0, 0, 0, 1, 0, 1, 1, 0)	3
(0, 0, 1, 1, 1, 0, 0, 1, 0, 1)	5
(0, 1, 0, 1, 0, 0, 0, 1, 1, 1)	5
(0, 1, 0, 1, 0, 1, 0, 0, 0, 1)	4
(0, 0, 1, 1, 1, 1, 0, 0, 1, 1)	6
(0, 1, 1, 0, 0, 1, 0, 1, 0, 1)	5
(0, 0, 0, 0, 1, 1, 1, 1, 1, 1)	6
(0, 1, 1, 0, 0, 1, 1, 1, 0, 1)	6
(0, 1, 0, 1, 1, 0, 1, 1, 1, 0)	6
(0, 0, 1, 1, 0, 0, 1, 1, 0, 0)	4
(0, 0, 1, 1, 0, 1, 1, 0, 1, 0)	5
(0, 1, 0, 1, 1, 1, 1, 0, 0, 0)	5
(1, 1, 1, 1, 0, 1, 1, 1, 1, 0)	8
(1, 0, 0, 1, 1, 1, 1, 1, 0, 0)	6
(1, 0, 0, 1, 1, 0, 1, 0, 1, 0)	5
(1, 1, 1, 1, 0, 0, 1, 0, 0, 0)	5
(1, 1, 0, 0, 1, 1, 1, 0, 1, 1)	7
(1, 0, 1, 0, 0, 1, 1, 0, 0, 1)	5
(1, 0, 1, 0, 0, 0, 1, 1, 1, 1)	6
(1, 1, 0, 0, 1, 0, 1, 1, 0, 1)	6
(1, 0, 0, 1, 0, 1, 0, 1, 0, 1)	5
(1, 1, 1, 1, 1, 1, 0, 1, 1, 1)	9
(1, 1, 1, 1, 1, 0, 0, 0, 0, 1)	6
(1, 0, 0, 1, 0, 0, 0, 0, 1, 1)	4
(1, 0, 1, 0, 1, 1, 0, 0, 0, 0)	4
(1, 1, 0, 0, 0, 1, 0, 0, 1, 0)	4
(1, 1, 0, 0, 0, 0, 0, 1, 0, 0)	3
(1, 0, 1, 0, 1, 0, 0, 1, 1, 0)	5

All codewords98 / 266

Linear Code $C = [10, 5, 3 \geq 2 \cdot 1 + 1]_2$

[1	0	0	1	1	1	0	0	1	1]
[1	0	0	1	0	1	0	0	0	0]
[1	0	0	1	1	1	1	0	0	0]
[0	1	1	0	0	1	0	0	1	1]
[1	1	0	1	0	1	1	0	0	1]

Parity-check matrix H

[1]
[0]
[0]
[0]
[1]
[0]
[1]
[1]
[0]

Received. vec.

[1]
[1]
[1]
[0]
[1]

Syndrome

(0, 0, 0, 0, 0, 0, 0, 0, 0, 0)	0
(0, 1, 1, 0, 1, 0, 0, 0, 1, 0)	4
(0, 1, 1, 0, 1, 1, 0, 1, 0, 0)	5
(0, 0, 0, 0, 0, 1, 0, 1, 1, 0)	3
(0, 0, 1, 1, 1, 0, 0, 1, 0, 1)	5
(0, 1, 0, 1, 0, 0, 0, 1, 1, 1)	5
(0, 1, 0, 1, 0, 1, 0, 0, 0, 1)	4
(0, 0, 1, 1, 1, 1, 0, 0, 0, 1)	6
(0, 1, 1, 0, 0, 0, 1, 0, 1, 1)	5
(0, 0, 0, 0, 1, 1, 1, 1, 1, 1)	6
(0, 1, 1, 0, 0, 1, 1, 1, 0, 1)	6
(0, 1, 0, 1, 1, 0, 1, 1, 1, 0)	6
(0, 0, 1, 1, 0, 0, 1, 1, 0, 0)	4
(0, 0, 1, 1, 0, 1, 1, 0, 1, 0)	5
(0, 1, 0, 1, 1, 1, 1, 0, 0, 0)	5
(1, 1, 1, 1, 0, 1, 1, 1, 1, 0)	8
(1, 0, 0, 1, 1, 1, 1, 1, 0, 0)	6
(1, 0, 0, 1, 1, 0, 1, 0, 1, 0)	5
(1, 1, 1, 1, 0, 0, 1, 0, 0, 0)	5
(1, 1, 0, 0, 1, 1, 1, 1, 0, 1)	7
(1, 0, 1, 0, 0, 1, 1, 0, 0, 1)	5
(1, 0, 1, 0, 0, 0, 1, 1, 1, 1)	6
(1, 1, 0, 0, 1, 0, 1, 1, 0, 1)	6
(1, 0, 0, 1, 0, 1, 0, 1, 0, 1)	5
(1, 1, 1, 1, 1, 1, 0, 1, 1, 1)	9
(1, 1, 1, 1, 1, 0, 0, 0, 0, 1)	6
(1, 0, 0, 1, 0, 0, 0, 0, 0, 1)	4
(1, 0, 1, 0, 1, 1, 0, 0, 0, 0)	4
(1, 1, 0, 0, 0, 1, 0, 0, 0, 1)	4
(1, 1, 0, 0, 0, 0, 0, 1, 0, 0)	3
(1, 0, 1, 0, 1, 0, 0, 1, 1, 0)	5

Linear Code C = $[10, 5, 3]_{\geq 2 \cdot 1 + 1}_2$

[1	0	0	1	1	1	0	0	1	1]
[1	0	0	1	0	1	0	0	0	0]
[1	0	0	1	1	1	1	0	0	0]
[0	1	1	0	0	1	0	0	1	1]
[1	1	0	1	0	1	1	0	0	1]

Parity-check matrix H

[1]
[0]
[0]
[0]
[0]
[1]
[0]
[1]
[1]
[0]

Received. vec.

```
[1]
[1]
[1]
[0]
[1]
```

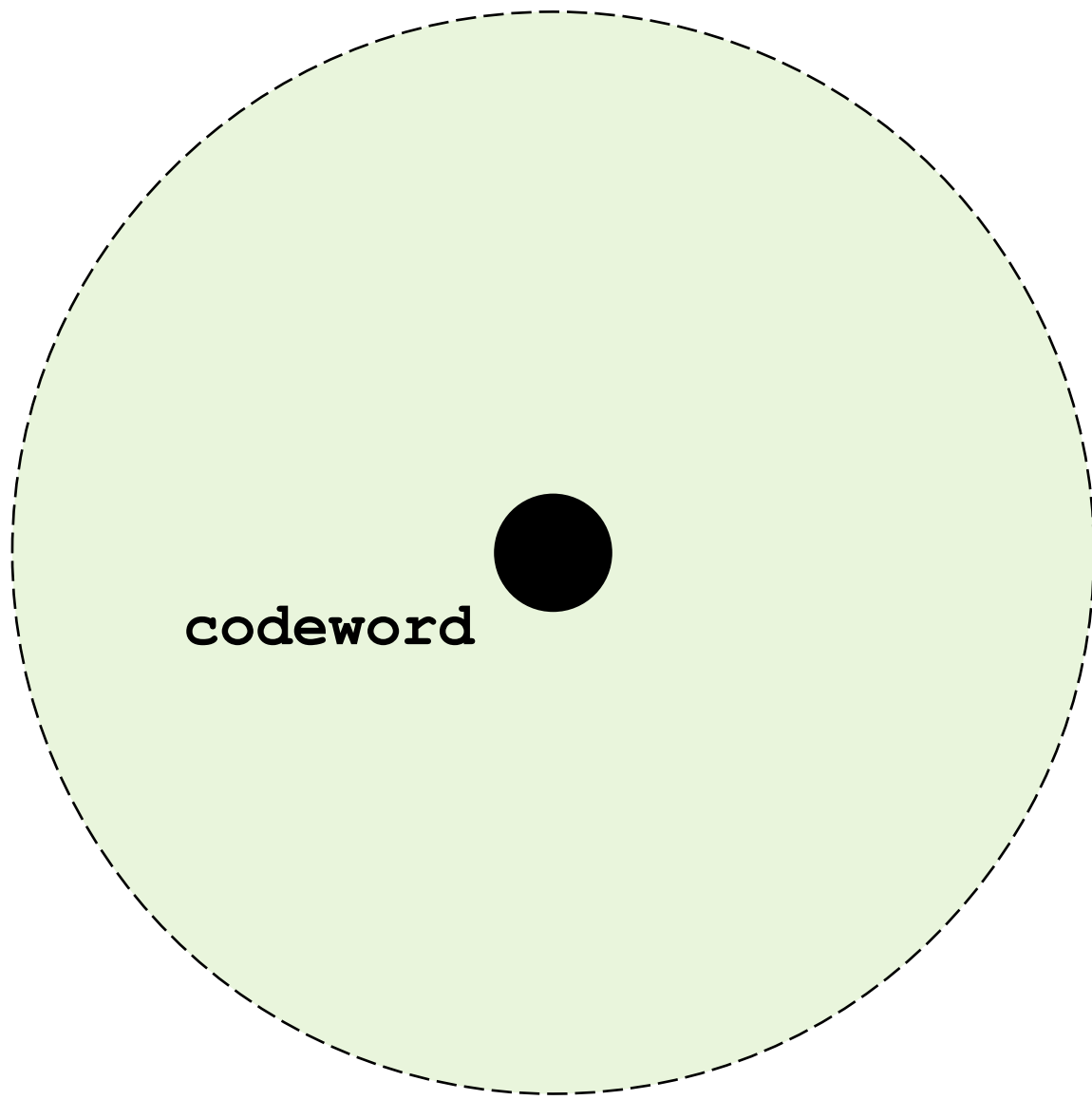
Syndrome

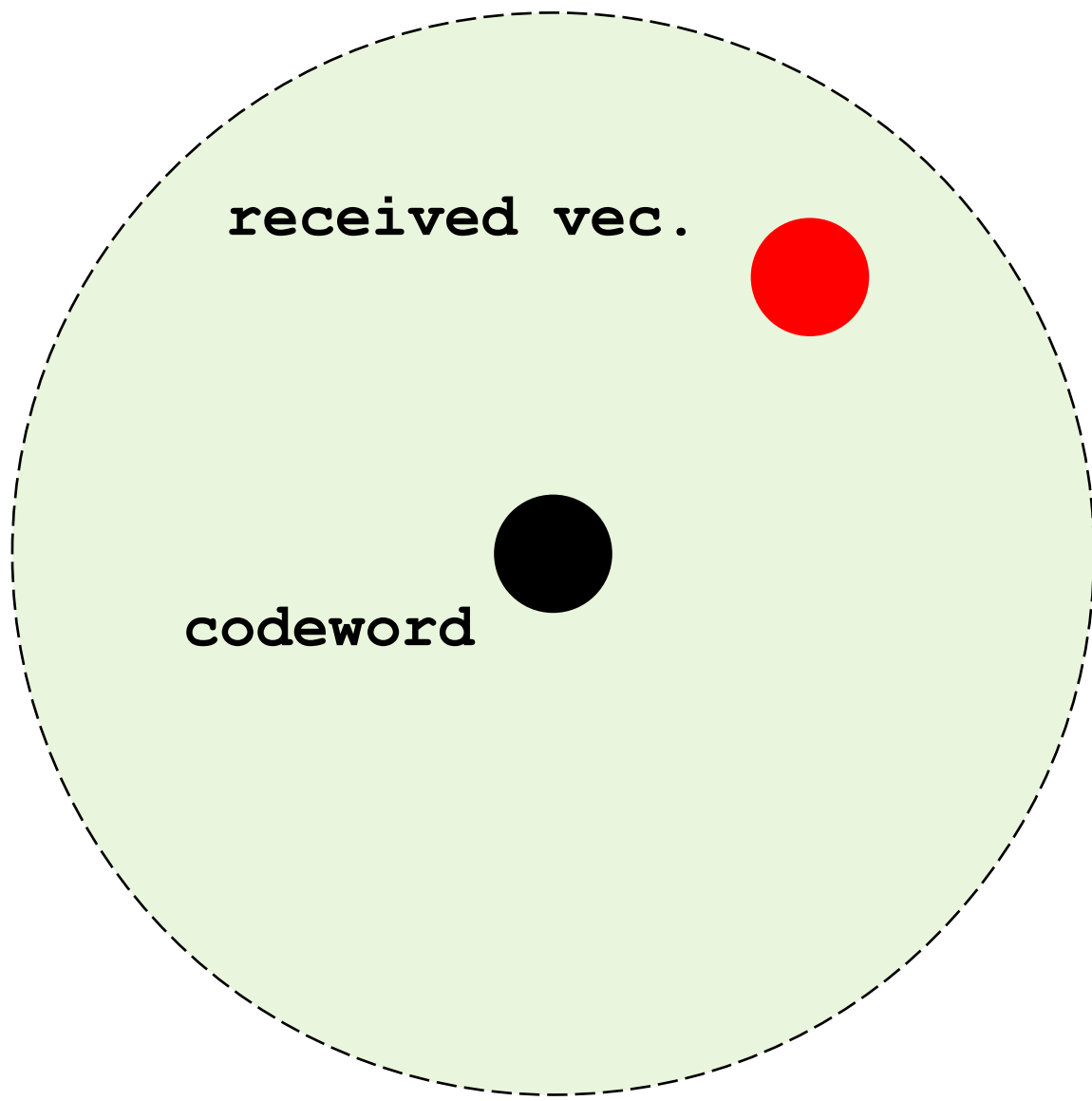
~~(0, 0, 0, 0, 0, 1, 0, 1, 1, 0) 3~~

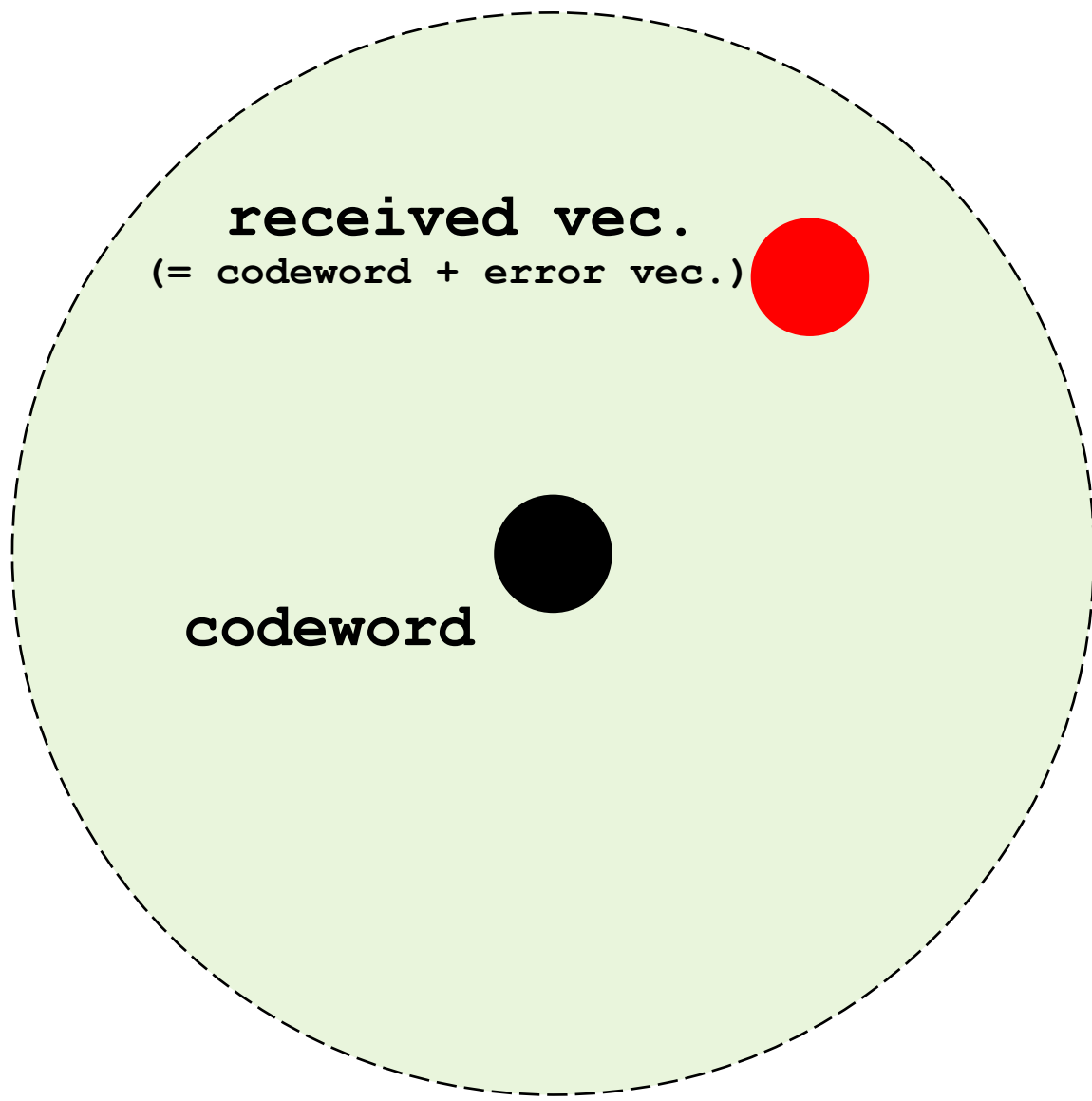
HOW?

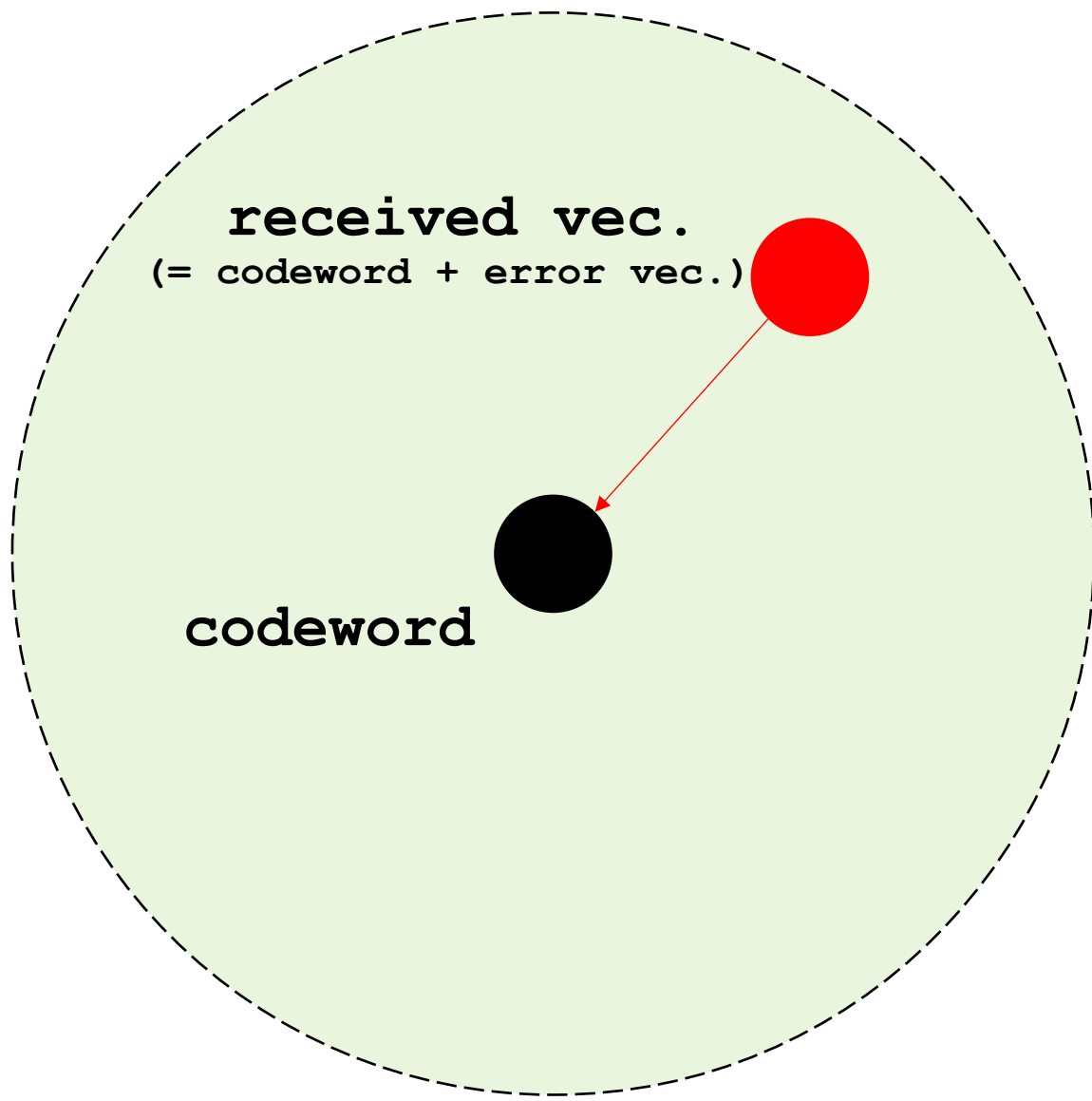
(0, 0, 0, 0, 0, 0, 0, 0, 0, 0)	4
(0, 1, 0, 0, 1, 0, 0, 0, 1, 0)	4
(0, 1, 1, 0, 1, 1, 0, 1, 0, 0)	5
0, 0, 0, 1, 0, 1, 1,	
(0, 1, 0, 1, 0, 1, 0, 0, 0, 1)	4
(0, 0, 1, 1, 1, 1, 0, 0, 1, 1)	6
(0, 1, 1, 0, 0, 0, 1, 0, 1, 1)	5
(0, 0, 0, 0, 1, 0, 1, 0, 0, 1)	3
(0, 0, 0, 0, 1, 1, 1, 1, 1, 1)	6
(0, 1, 1, 0, 0, 1, 1, 1, 0, 1)	6
(0, 1, 1, 0, 1, 1, 1, 1, 0)	6
(0, 0, 0, 1, 1, 0, 1, 1, 0, 0)	4
(0, 0, 1, 1, 0, 1, 1, 0, 1, 0)	5
(0, 1, 0, 1, 1, 1, 1, 0, 0, 0)	5
(1, 1, 1, 1, 0, 1, 1, 1, 1, 0)	8
(1, 0, 0, 1, 1, 1, 1, 1, 0, 0)	6
(1, 0, 0, 1, 1, 0, 1, 0, 1, 0)	5
(1, 1, 1, 1, 0, 0, 1, 0, 0, 0)	5
(1, 1, 0, 0, 1, 1, 1, 0, 1, 1)	7
(1, 0, 1, 0, 0, 1, 1, 0, 0, 1)	5
(1, 0, 1, 0, 0, 0, 1, 1, 1, 1)	6
(1, 1, 0, 0, 1, 0, 1, 1, 0, 1)	6
(1, 0, 0, 1, 0, 1, 0, 1, 0, 1)	5
(1, 1, 1, 1, 1, 1, 0, 1, 1, 1)	9
(1, 1, 1, 1, 1, 0, 0, 0, 0, 1)	6
(1, 0, 0, 1, 0, 0, 0, 0, 1, 1)	4
(1, 0, 1, 0, 1, 1, 0, 0, 0, 0)	4
(1, 1, 0, 0, 0, 1, 0, 0, 1, 0)	4
(1, 1, 0, 0, 0, 0, 0, 1, 0, 0)	3
(1, 0, 1, 0, 1, 0, 0, 1, 1, 0)	5

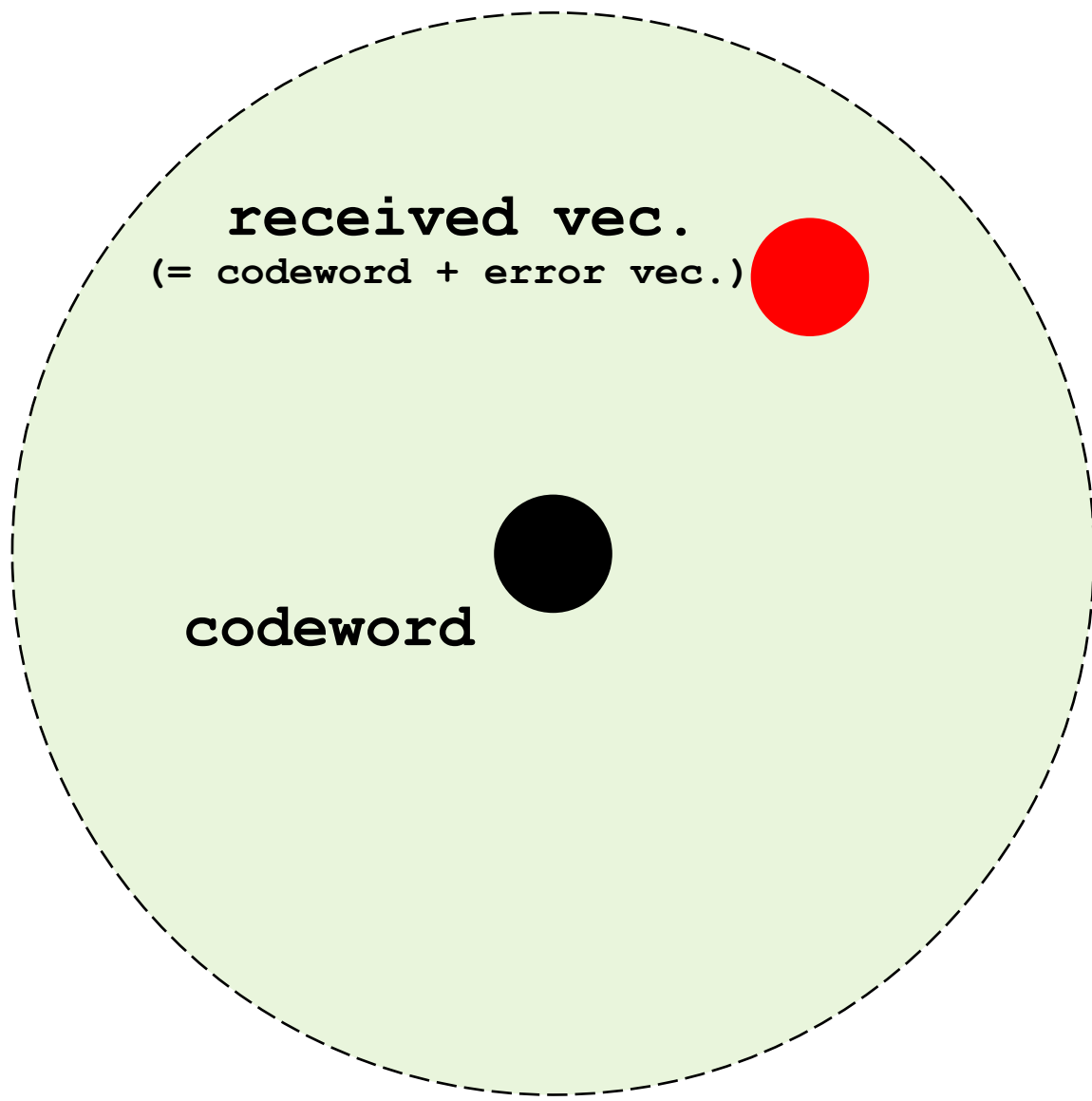
All codewords

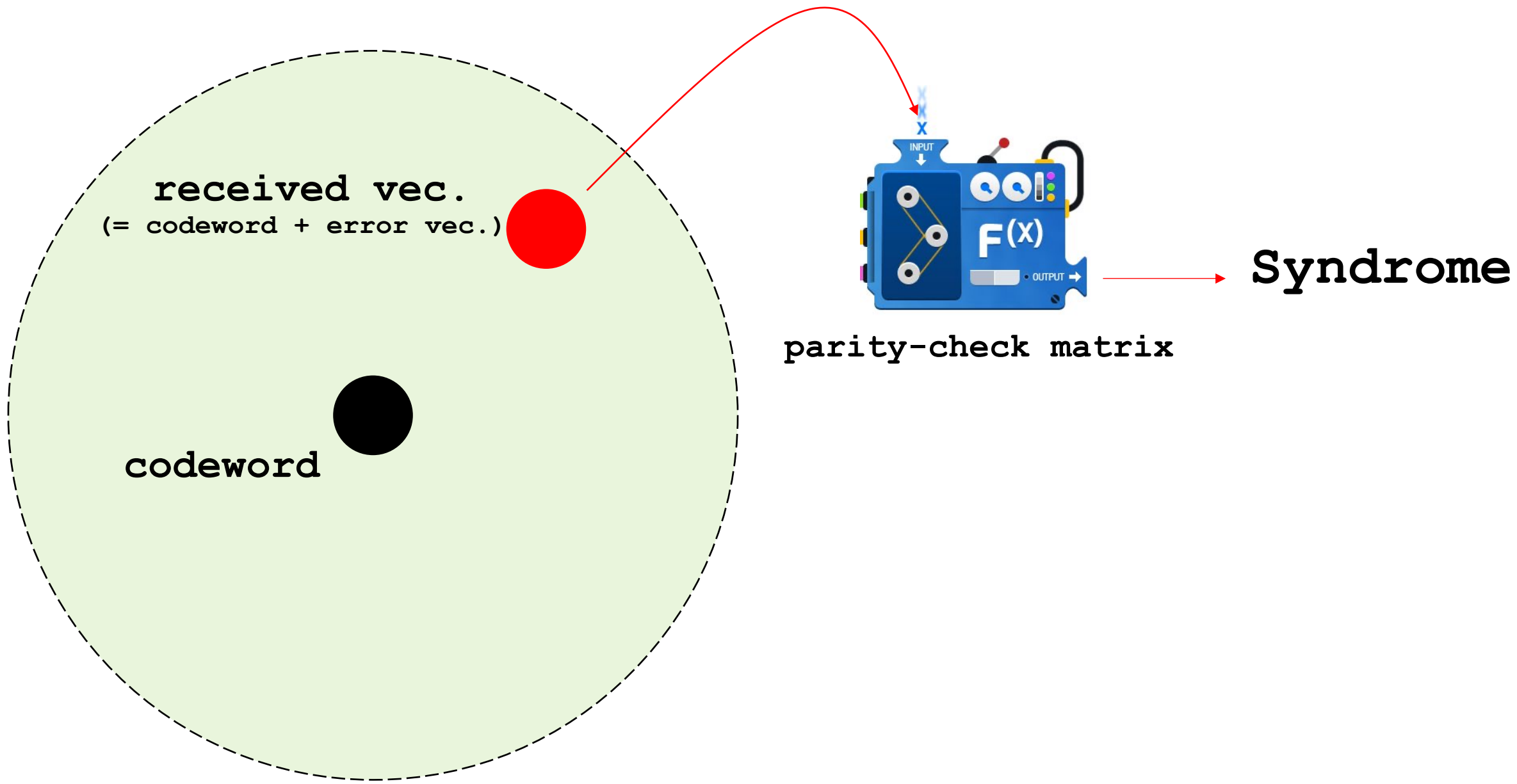


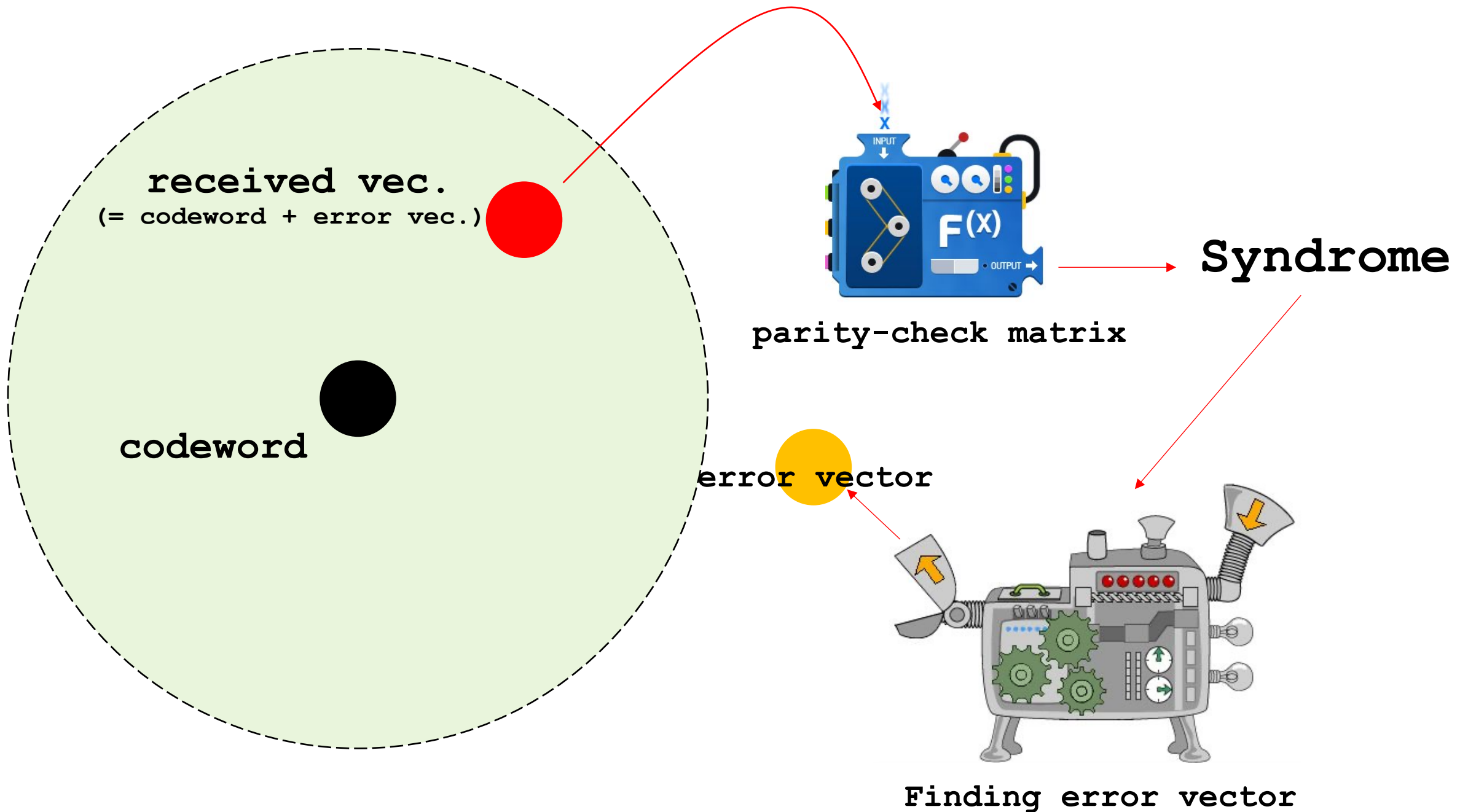


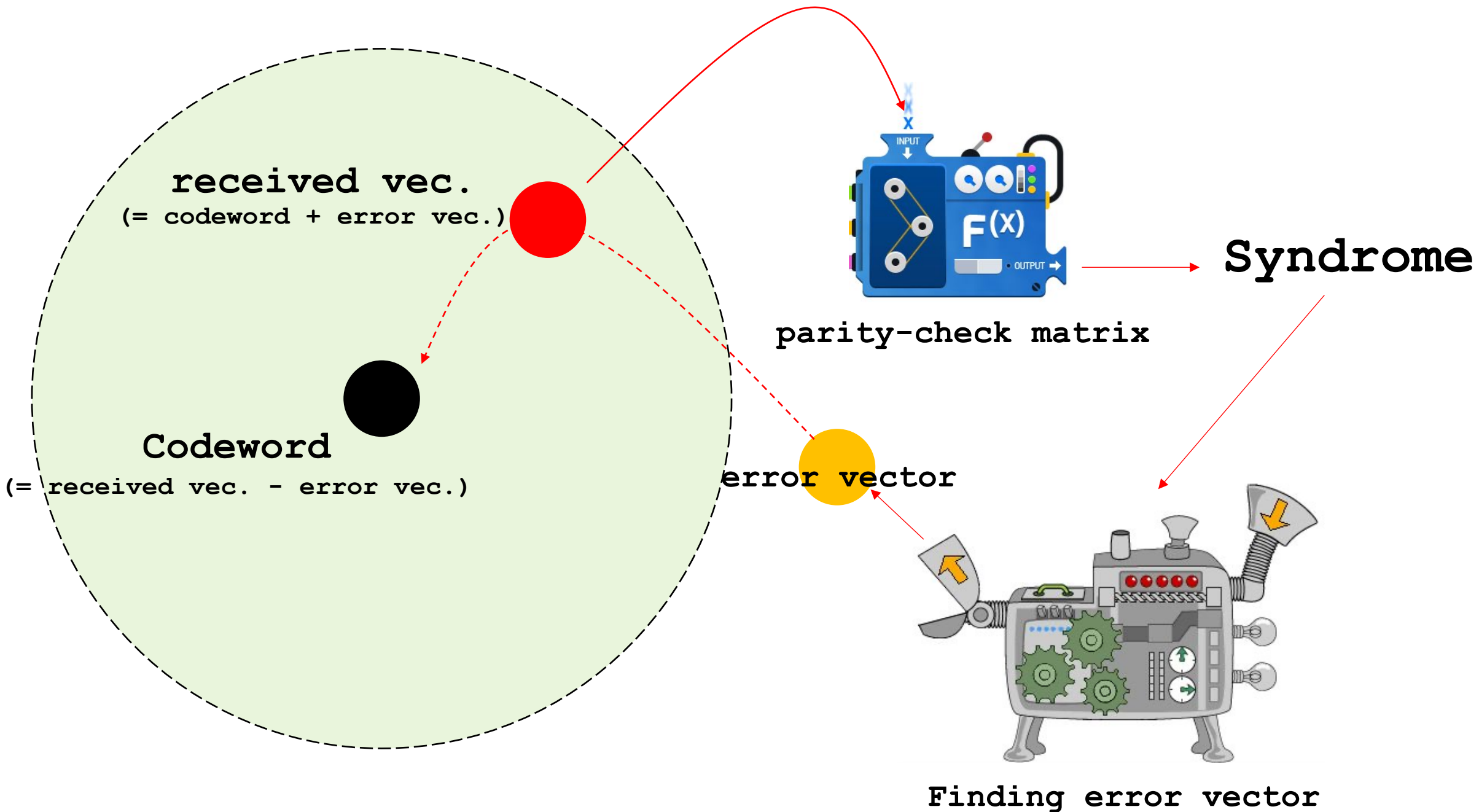


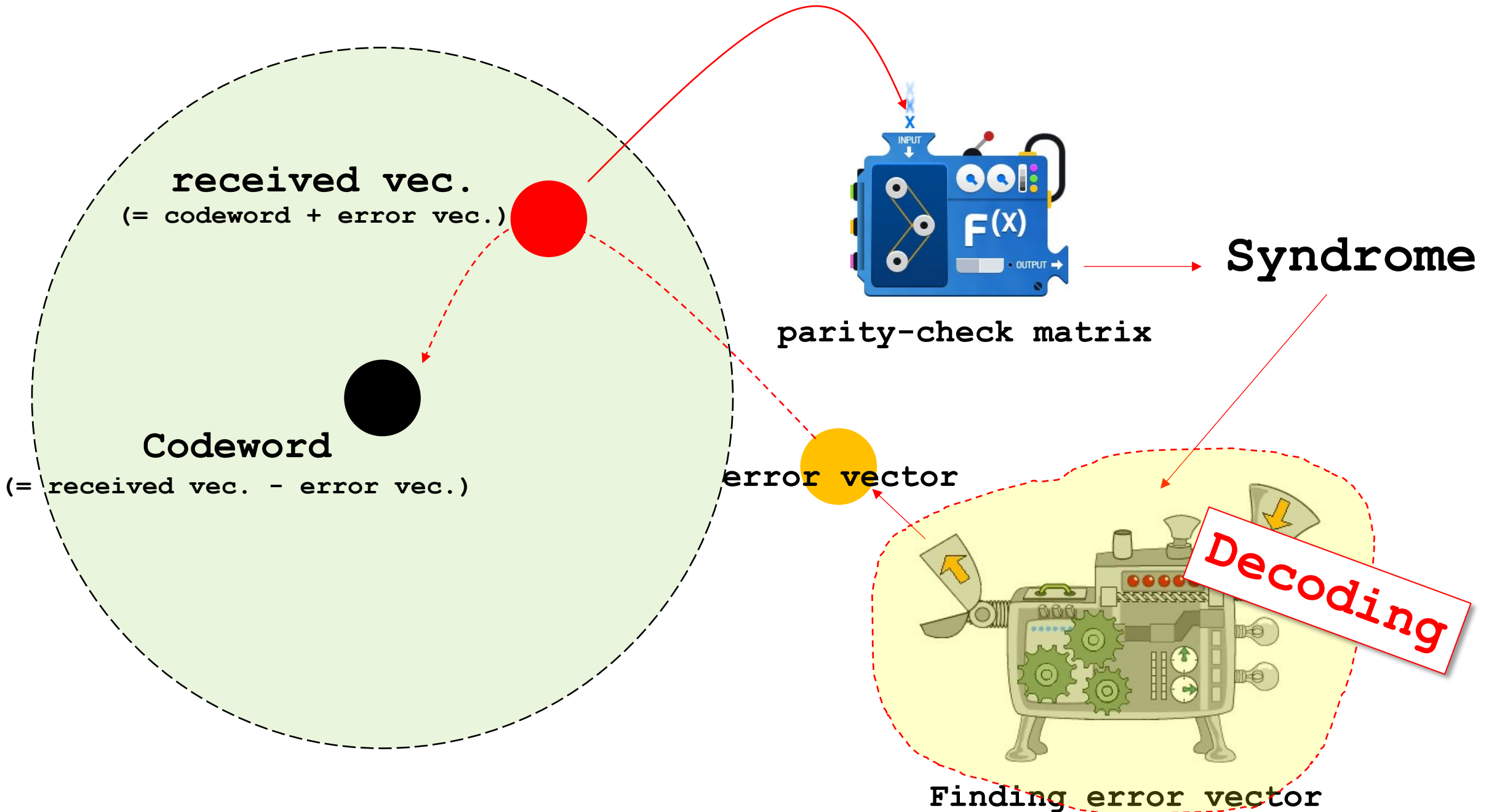






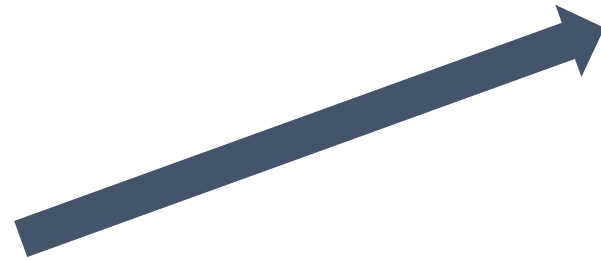






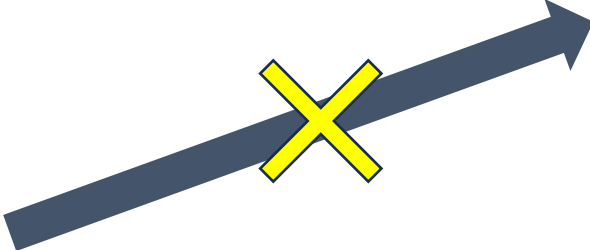
Syndrome

Syndrome

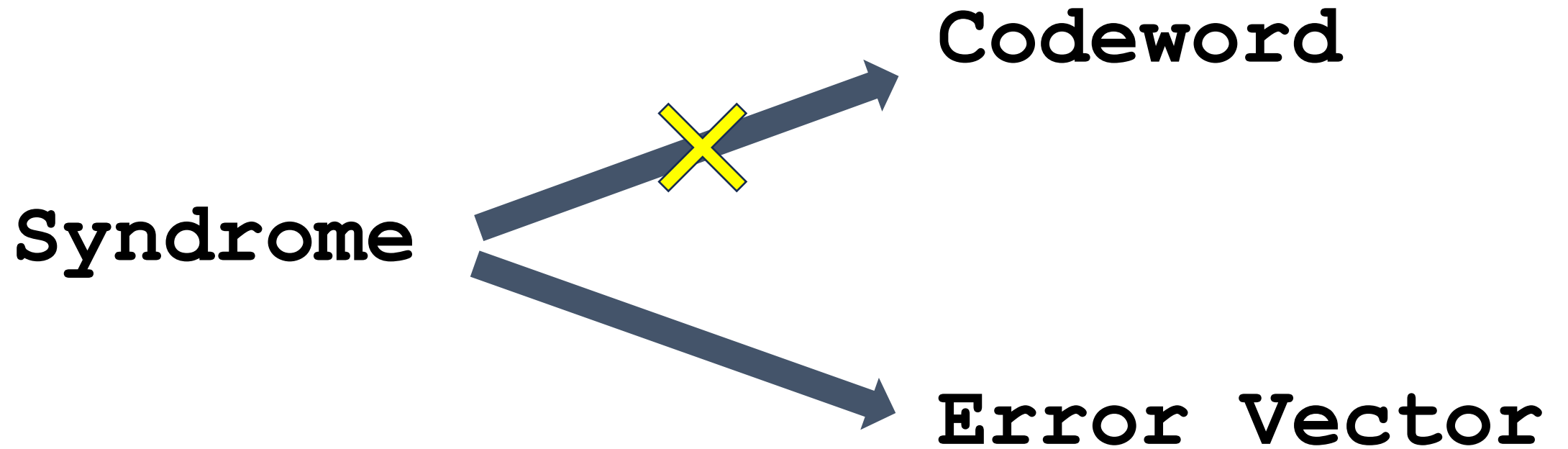


Codeword

Syndrome



Codeword



Linear Code $C = [10, 5, 3 \geq 2 \cdot 1 + 1]_2$

[1	0	0	1	1	1	0	0	1	1]
[1	0	0	1	0	1	0	0	0	0]
[1	0	0	1	1	1	1	0	0	0]
[0	1	1	0	0	1	0	0	1	1]
[1	1	0	1	0	1	1	0	0	1]

Parity-check matrix H

[1]
[0]
[0]
[0]
[1]
[0]
[1]
[1]
[0]

Received. vec.

=

[1]
[1]
[1]
[0]
[1]

Syndrome

$(0, 0, 0, 0, 0, 1, 0, 1, 1, 0)$
3

Linear Code $C = [10, 5, 3 \geq 2 \cdot 1 + 1]_2$

The diagram illustrates the syndrome calculation for a Hamming code. It consists of the following components:

- Parity-check matrix H:** A 5x10 matrix of 0s and 1s.
- codeword:** A vertical column of 10 elements: [0], [0], [0], [0], [0], [1], [0], [1], [1], [0].
- error vec.:** A vertical column of 10 elements: [1], [0], [0], [0], [0], [0], [0], [0], [0], [0]. The first element [1] is highlighted with a red circle.
- Syndrome:** A vertical column of 5 elements: [1], [1], [1], [0], [1].
- Equation:** The codeword plus the error vector equals the syndrome.
- Message:** A red box containing the message (0, 0, 0, 0, 0, 1, 0, 1, 1, 0) 3.

[1]	0	0	1	1	1	0	0	1	1
[1]	0	0	1	0	1	0	0	0	0
[1]	0	0	1	1	1	1	0	0	0
[0]	1	1	0	0	1	0	0	1	1
[1]	1	0	1	0	1	1	0	0	1

Parity-check matrix H

codeword

error vec.

Syndrome

(0, 0, 0, 0, 0, 1, 0, 1, 1, 0) 3

Linear Code $C = [10, 5, 3 \geq 2 \cdot 1 + 1]_2$

1	0	0	1	1	1	0	0	1	1
1	0	0	1	0	1	0	0	0	0
1	0	0	1	1	1	1	0	0	0
0	1	1	0	0	1	0	0	1	1
1	1	0	1	0	1	1	0	0	1

Parity-check matrix H

$\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$

error vec.

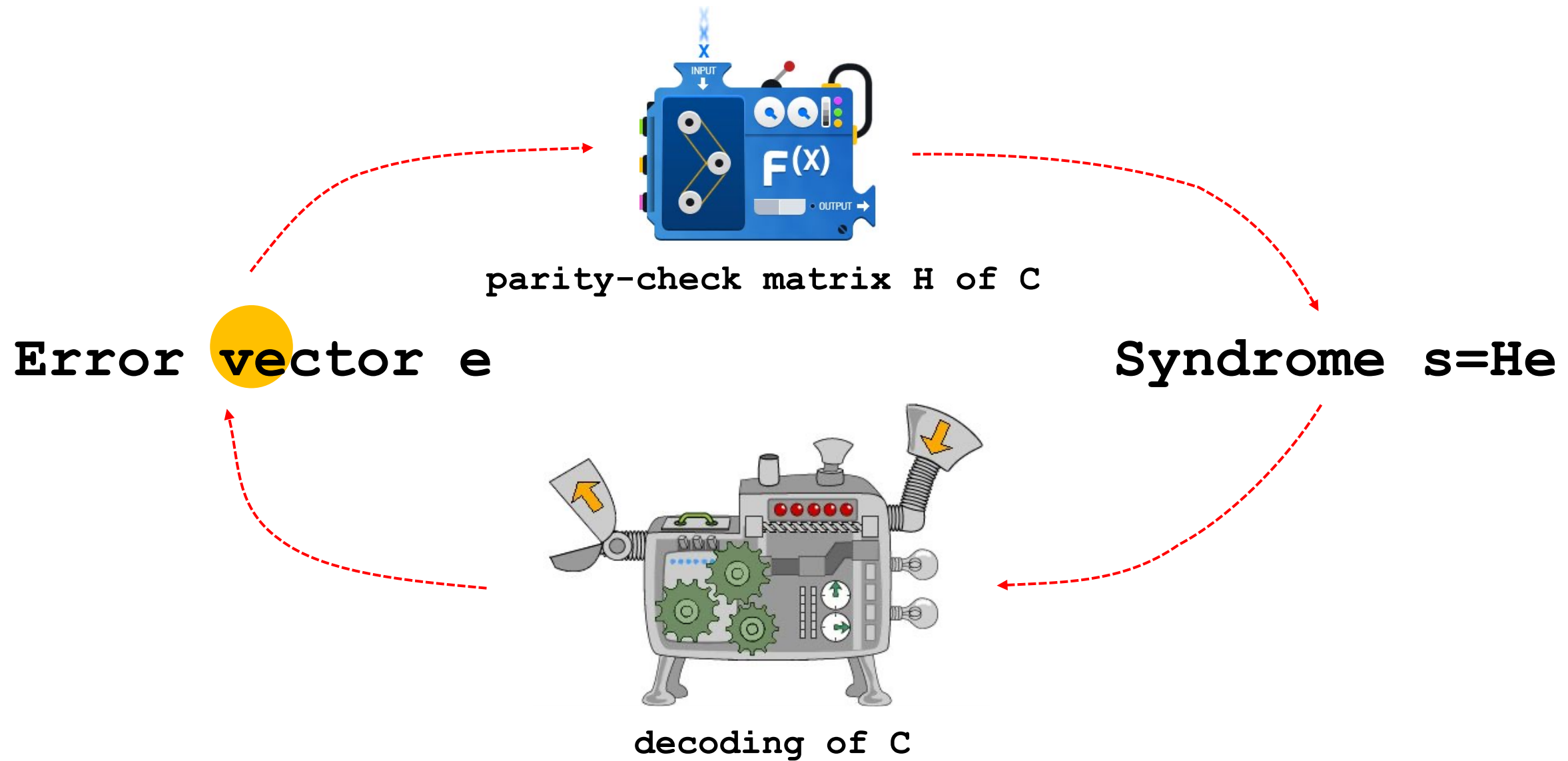
=

$\begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 1 \end{bmatrix}$

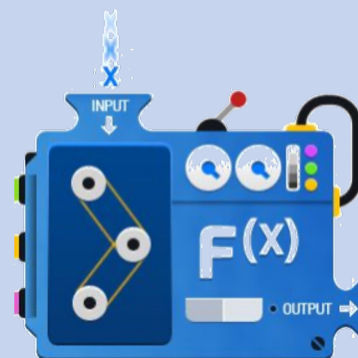
Syndrome

$(0, 0, 0, 0, 0, 1, 0, 1, 1, 0)_3$

$$\begin{aligned}
 s &= Hr \\
 &= H(c+e) \\
 &= Hc + He \\
 &= 0 + He \\
 &= He
 \end{aligned}$$



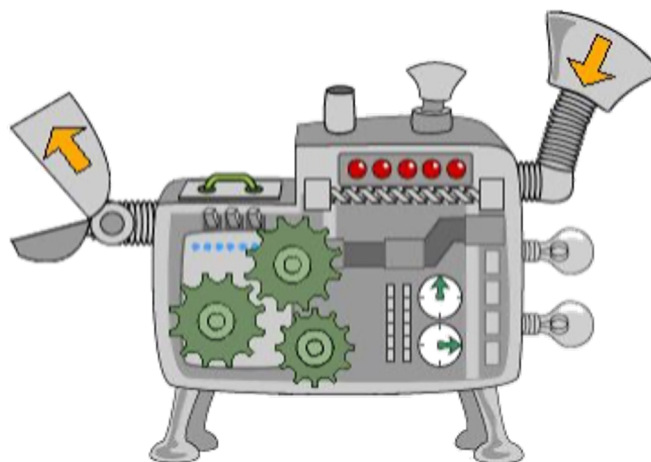
f



parity-check matrix H of C

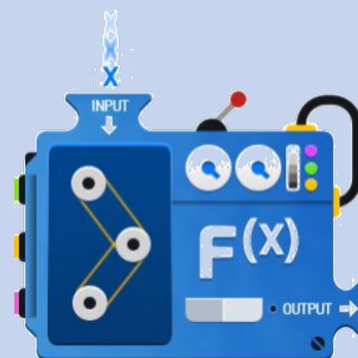
Error **vector** e

Syndrome $s=He$



decoding of C

f

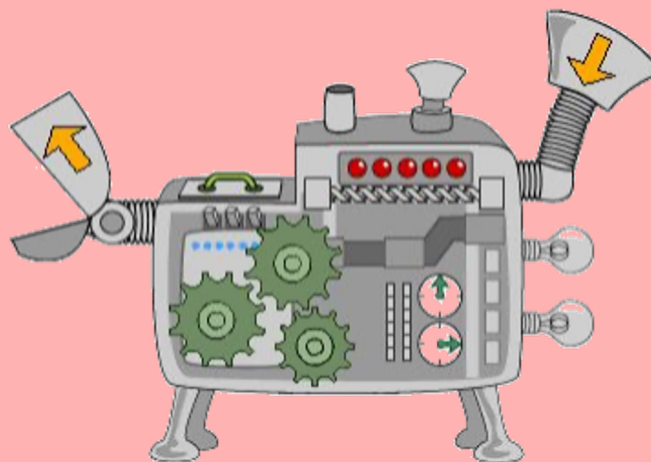


parity-check matrix H of C

Error vector e

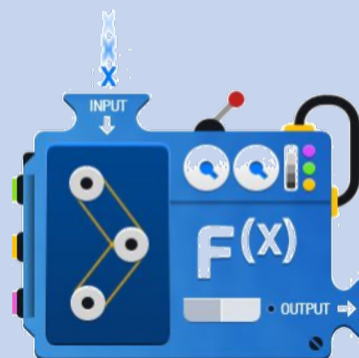
Syndrome $s = He$

f^{-1}



decoding of C

f

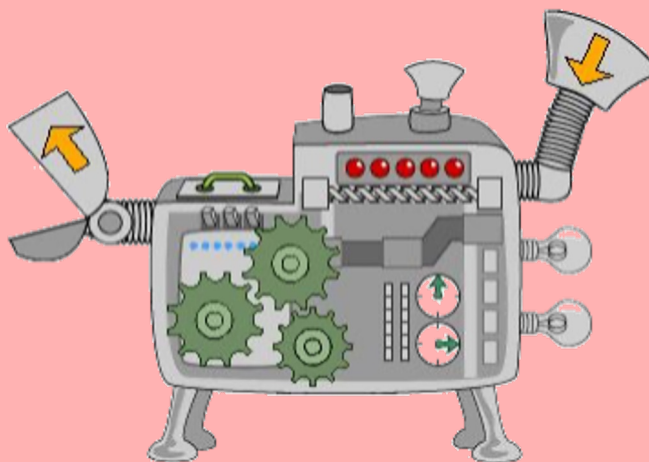


parity-check matrix H of C

Error **vector** e

Syndrome $s=He$

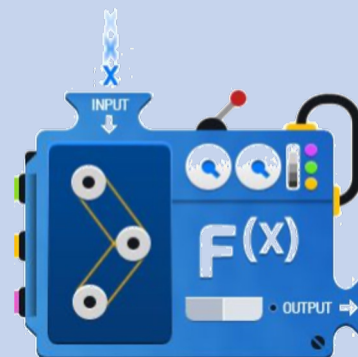
f^{-1}



decoding of C

SDP

f



parity-check matrix H of C

Error vector e

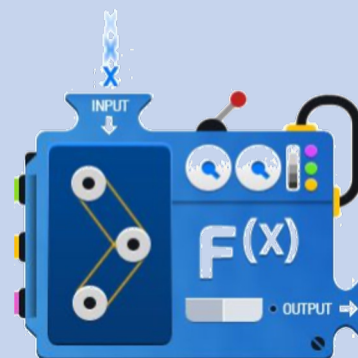
Syndrome $s=He$

If SDP is difficult...

decoding of C

SDP

f



parity-check matrix H of C

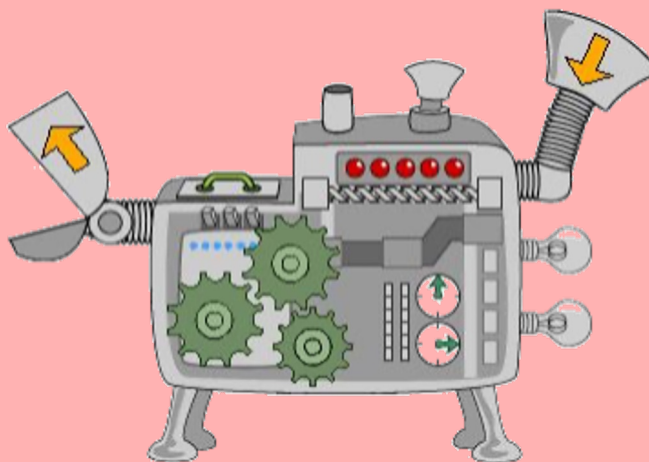
Plaintext

Error vector e

Ciphertext

Syndrome s

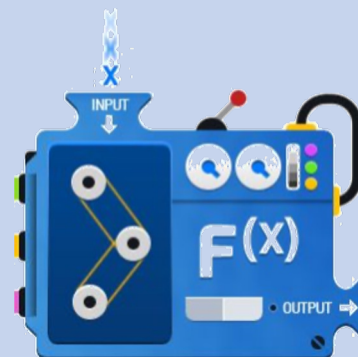
f^{-1}



decoding of C

SDP

f



parity-check matrix H of C

Plaintext

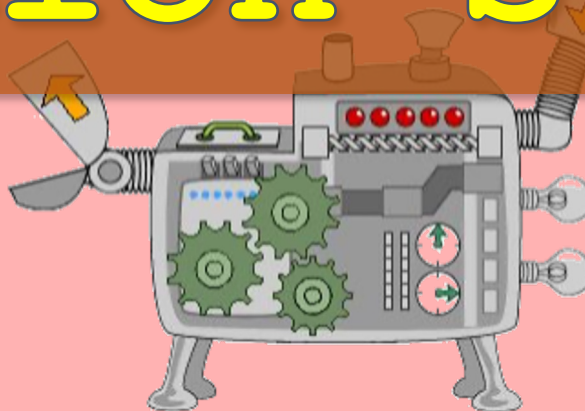
Ciphertext

Error vector e

Syndrome s

Encryption Scheme!!

f^{-1}

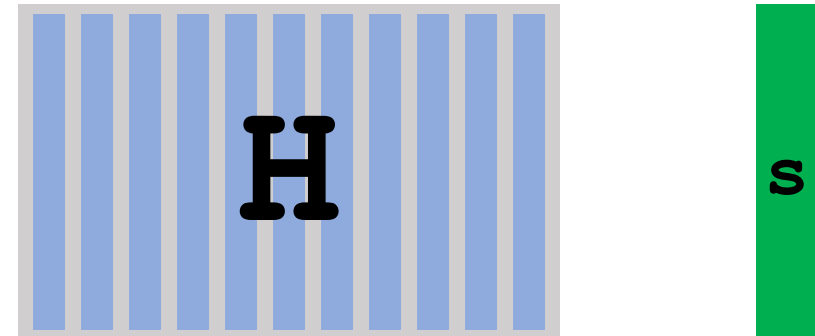


decoding of C

SDP

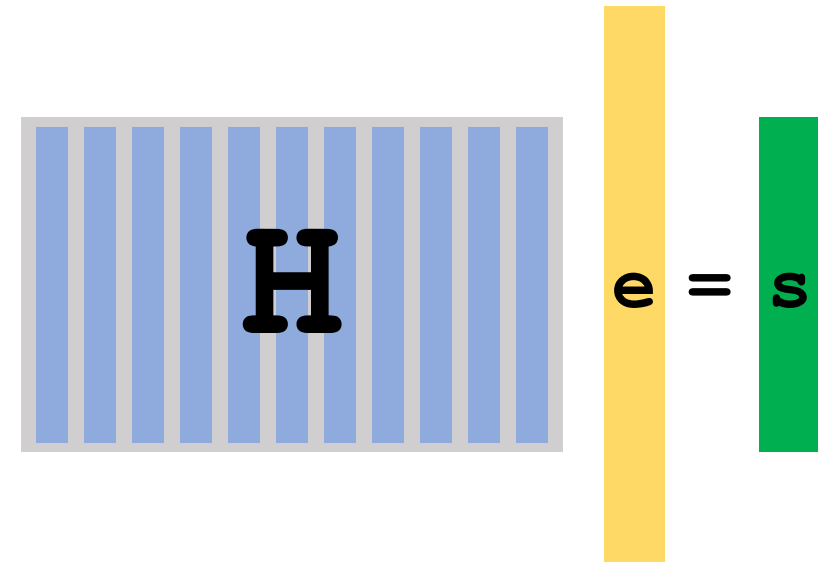
SDP (Syndrome Decoding Problem)

Given a parity-check matrix $H \in \mathbb{F}_2^{(n-k) \times n}$ of a **random binary** linear code $C = [n, k]_2$, a syndrome vector $s \in \mathbb{F}_2^{n-k}$ and a positive integer $w \in \mathbb{Z}_{>0}$, find the error vector $e \in \mathbb{F}_2^n$ such that **$He = s$**



SDP (Syndrome Decoding Problem)

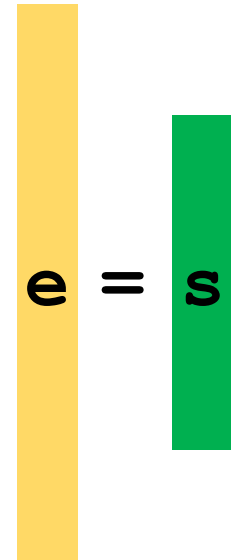
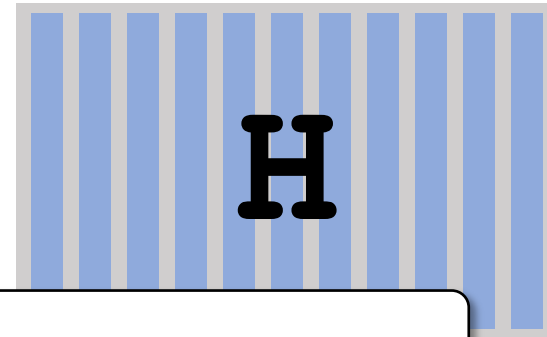
Given a parity-check matrix $H \in \mathbb{F}_2^{(n-k) \times n}$ of a **random binary** linear code $C = [n, k]_2$, a syndrome vector $s \in \mathbb{F}_2^{n-k}$ and a positive integer $w \in \mathbb{Z}_{>0}$, find the error vector $e \in \mathbb{F}_2^n$ such that **$He = s$**



SDP (Syndrome Decoding Problem)

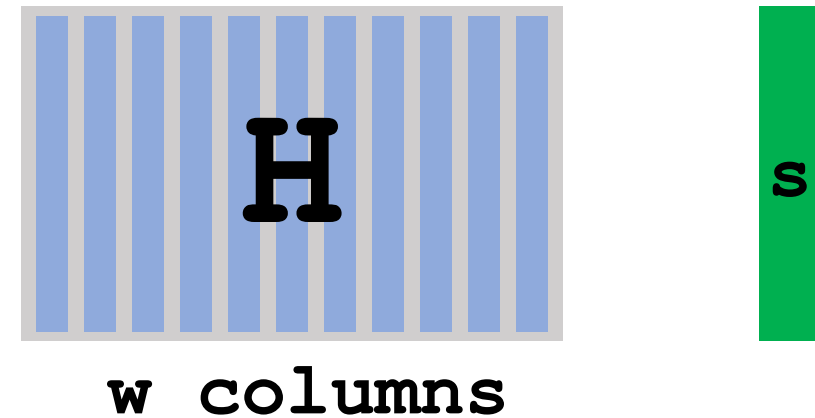
Given a parity-check matrix $H \in \mathbb{F}_2^{(n-k) \times n}$ of a **random binary** linear code $C = [n, k]_2$, a syndrome vector $s \in \mathbb{F}_2^{n-k}$, a positive integer n , find the error vector $e \in \mathbb{F}_2^n$ such that **$He = s$**

Easy Problem



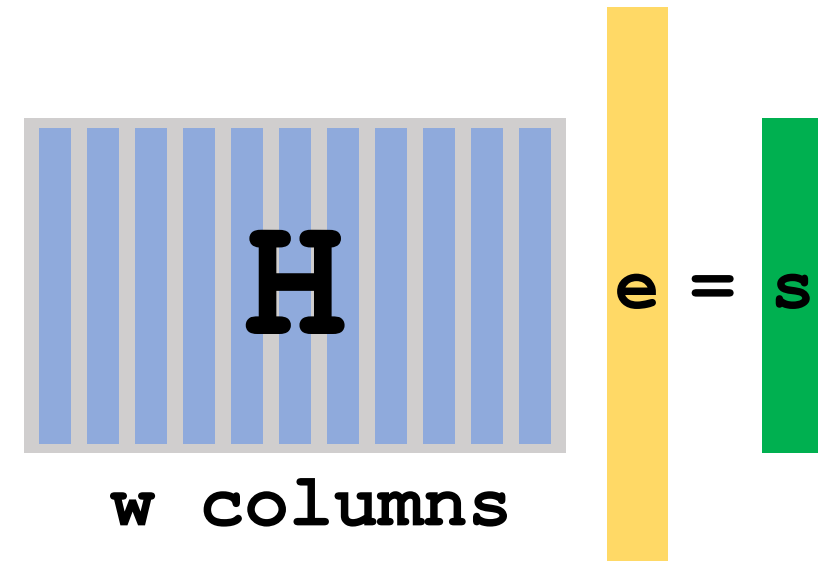
SDP (Syndrome Decoding Problem)

Given a parity-check matrix $H \in \mathbb{F}_2^{(n-k) \times n}$ of a **random binary** linear code $C = [n, k]_2$, a syndrome vector $s \in \mathbb{F}_2^{n-k}$ and a positive integer $w \in \mathbb{Z}_{>0}$, find the error vector $e \in \mathbb{F}_2^n$ such that (1) **$He = s$** and (2) **$wH(e) = w$** .



SDP (Syndrome Decoding Problem)

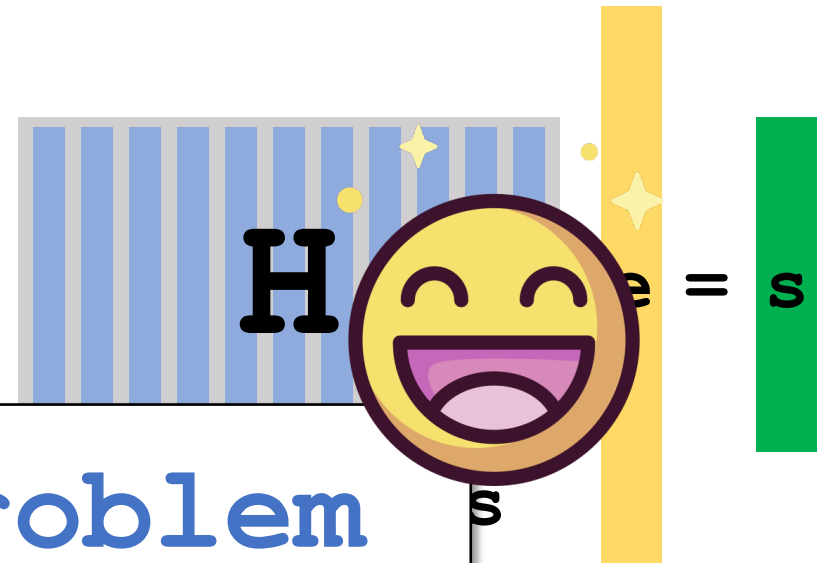
Given a parity-check matrix $H \in \mathbb{F}_2^{(n-k) \times n}$ of a **random binary** linear code $C = [n, k]_2$, a syndrome vector $s \in \mathbb{F}_2^{n-k}$ and a positive integer $w \in \mathbb{Z}_{>0}$, find the error vector $e \in \mathbb{F}_2^n$ such that (1) **$He = s$** and (2) **$wH(e) = w$** .



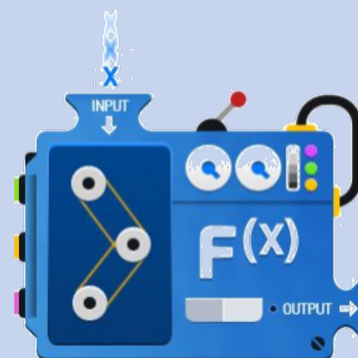
SDP (Syndrome Decoding Problem)

Given a parity-check matrix $H \in \mathbb{F}_2^{(n-k) \times n}$ of a **random binary** linear code $C = [n, k]_2$, a syndrome vector $s \in \mathbb{F}_2^{n-k}$ and a positive integer w , find the error vector $e \in \mathbb{F}_2^n$ such that (1) $He = s$ and (2) $wH(e) = w$.

NP-hard Problem



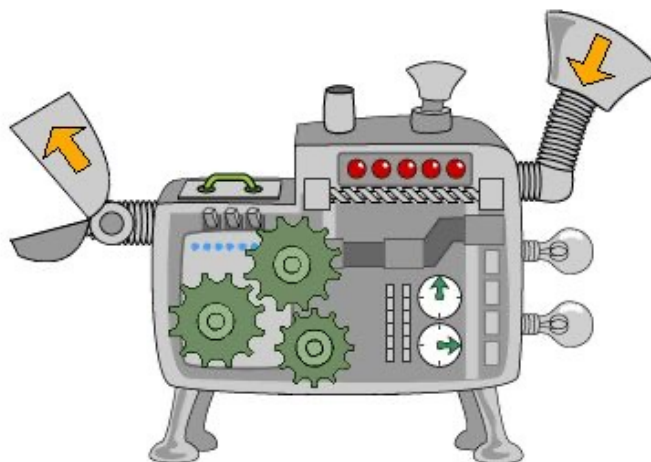
f



parity-check matrix H of C

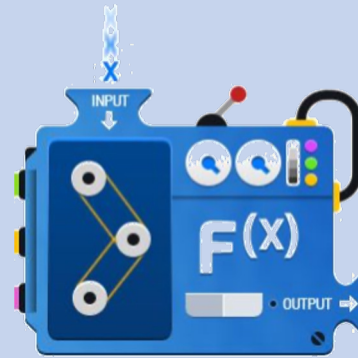
error vector e
($wH(e) = w$)

Syndrome $s = He$



decoding of C

f



parity-check matrix H of C

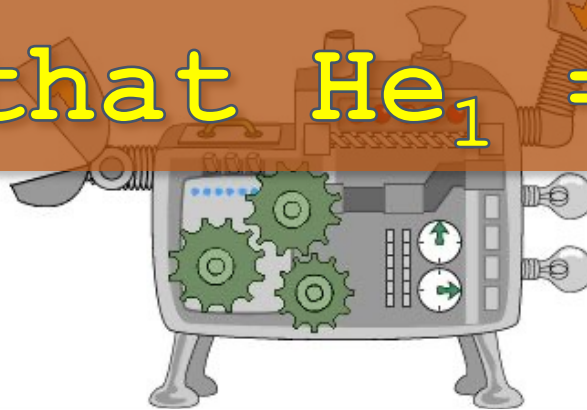
What if...

error vector e

there exists distinct e_1 and e_2

Syndrome $s=He$

such that $He_1 = He_2$?



decoding of C

SDP (Syndrome Decoding Problem)

Given a parity-check matrix $H \in \mathbb{F}_2^{(n-k) \times n}$ of a

For the uniqueness of the root of an SDP, w equals to t when $d_H(C) \geq 2t+1$.

positive integer $w \in \mathbb{Z}_{>0}$,

find the error vector

$e \in \mathbb{F}_2^n$ such that (1) $He = s$

and (2) $wH(e) = w$.

w columns

SDP (Syndrome Decoding Problem)

Given a parity-check matrix $H \in \mathbb{F}_2^{(n-k) \times n}$ of a

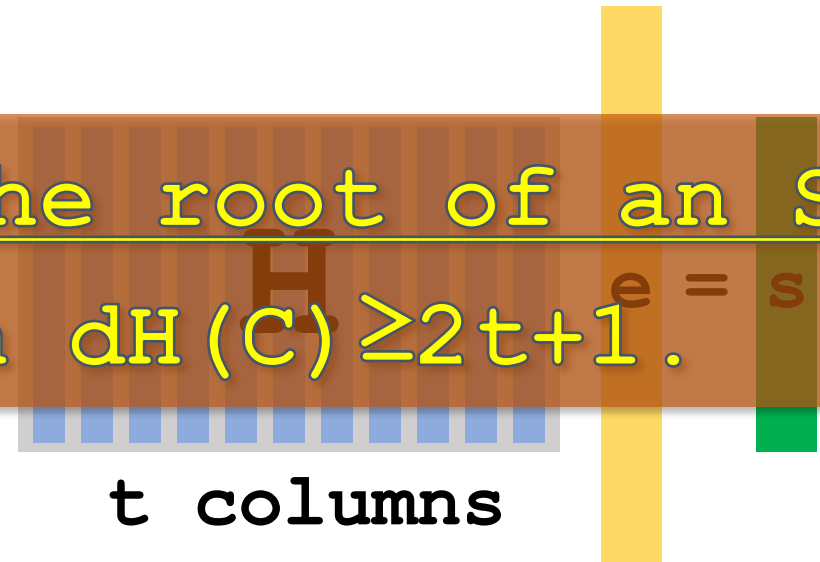
For the uniqueness of the root of an SDP, w equals to t when $d_H(C) \geq 2t+1$.

positive integer $w \in \mathbb{Z}_{>0}$,

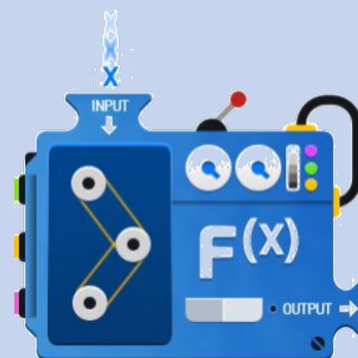
find the error vector

$e \in \mathbb{F}_2^n$ such that (1) $He=s$

and (2) $wH(e)=t$.



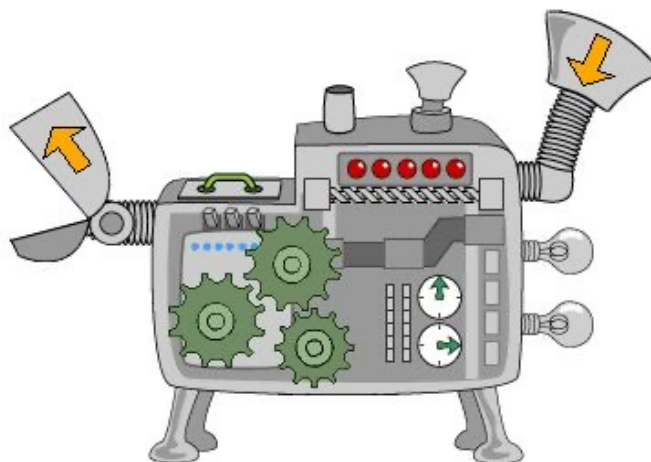
f



parity-check matrix H of C

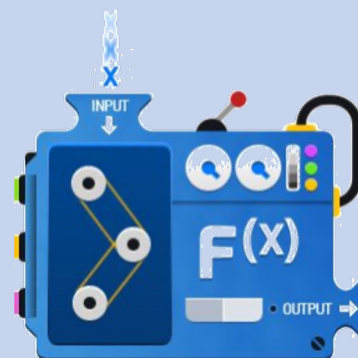
error **vector** e
($wH(e) = t$)

Syndrome $s = He$



decoding of C

f

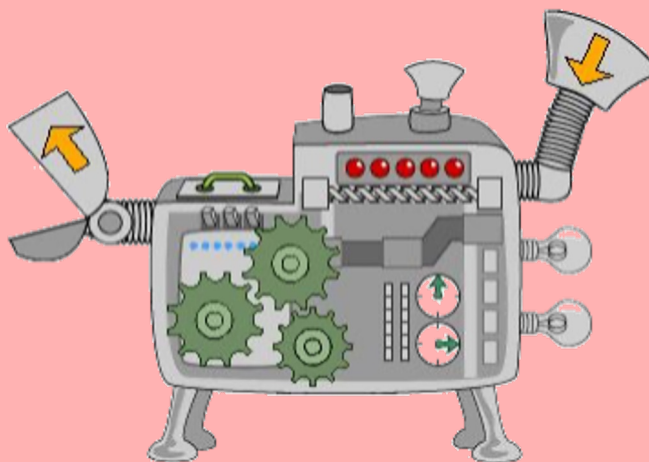


parity-check matrix H of C

error vector e
($wH(e) = t$)

Syndrome $s = He$

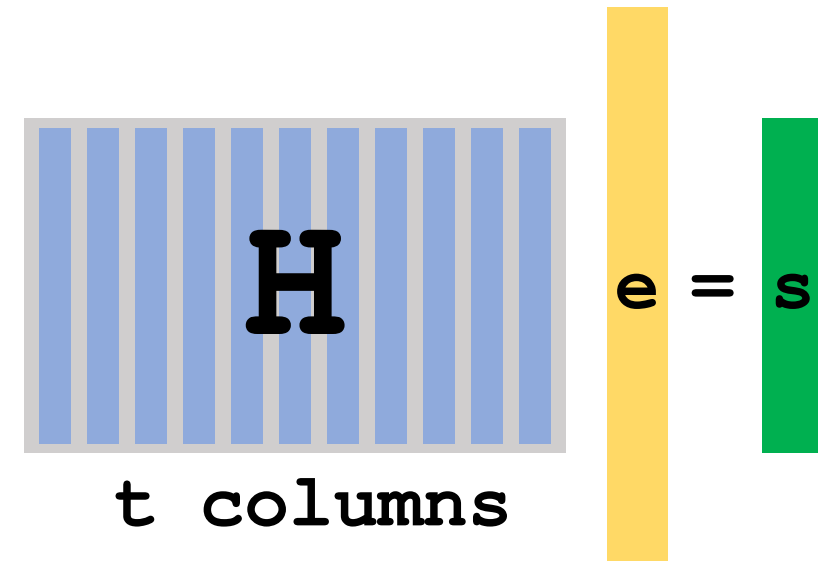
f^{-1}



decoding of C

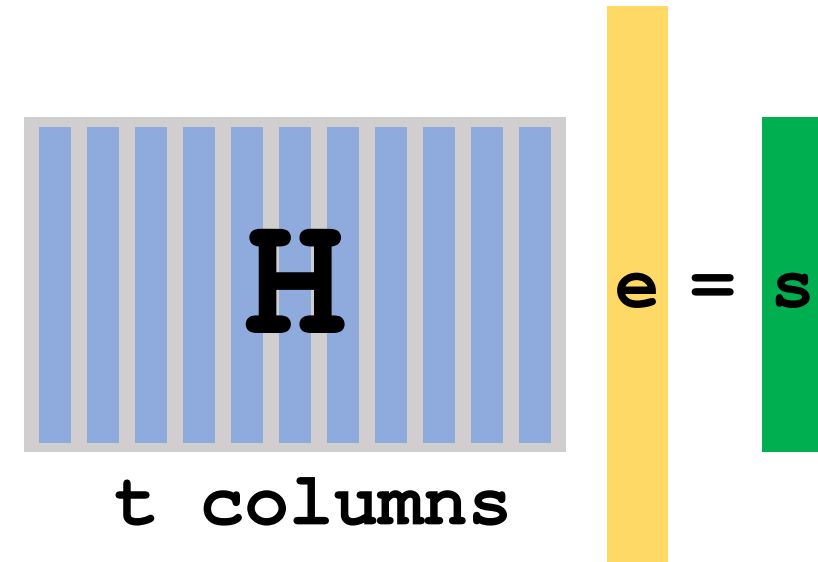
SDP (Syndrome Decoding Problem)

Given a parity-check matrix $H \in \mathbb{F}_2^{(n-k) \times n}$ of a **random binary** linear code $C = [n, k]_2$, a syndrome vector $s \in \mathbb{F}_2^{n-k}$ and a positive integer $w \in \mathbb{Z}_{>0}$, find the error vector $e \in \mathbb{F}_2^n$ such that (1) **$He = s$** and (2) **$wH(e) = t$** .



SDP (Syndrome Decoding Problem)

Given a parity-check matrix $H \in \mathbb{F}_2^{(n-k) \times n}$ of a **random binary** linear code $C = [n, k]_2$, a syndrome vector $s \in \mathbb{F}_2^{n-k}$ and a positive integer $w \in \mathbb{Z}_{>0}$, find the error vector $e \in \mathbb{F}_2^n$ such that (1) **$He = s$** and (2) **$wH(e) = t$** .



Parity-check Matrix H

Good Codes

Parity-check Matrix H

Good Codes

- Efficient decoding

Parity-check Matrix H

Good Codes

- Efficient decoding
- k/n is close to 1
- Many errors can be corrected

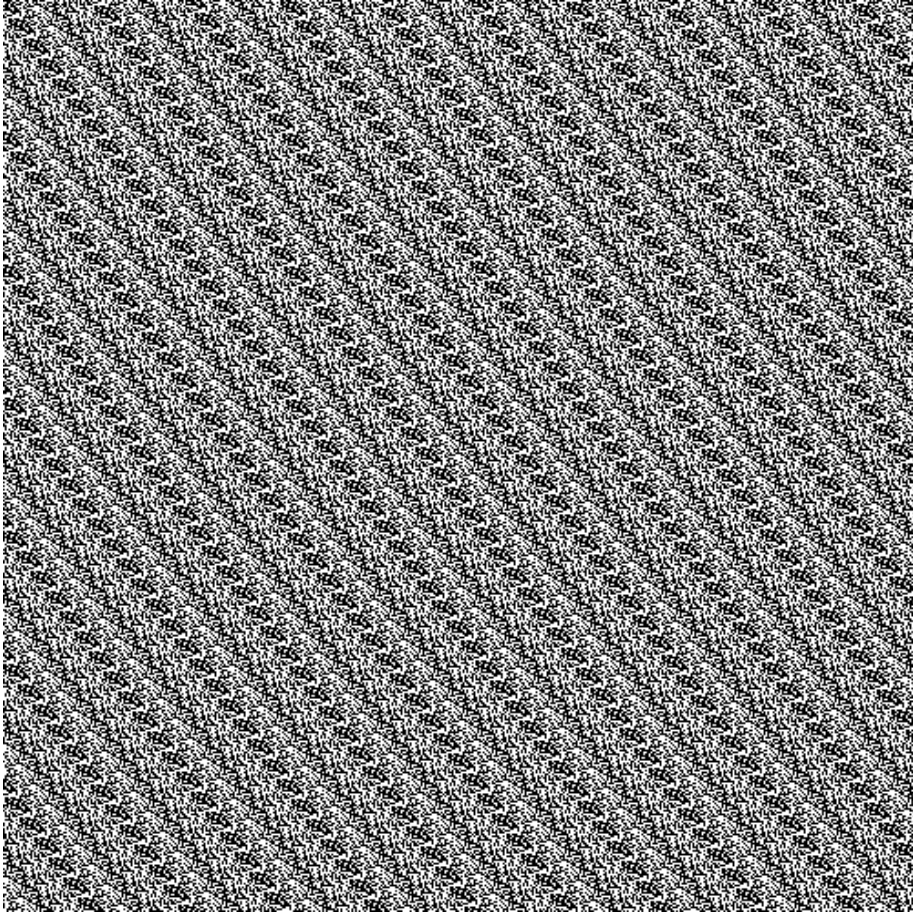
Parity-check Matrix H

Good Codes

- Efficient decoding
- k/n is close to 1
- Many errors can be corrected

BCH, RS, GRS, RM, Goppa, etc.

Parity-check Matrix H



Good Codes

- Efficient decoding
- k/n is close to 1
- Many errors can be corrected

BCH, RS, GRS, RM, Goppa, etc.

Parity-check Matrix H

Leakage of decoding
information

- k/n is close to 1
 - Many errors can be corrected
- BCH, RS, GRS, RM, Goppa, etc.

Parity-check Matrix H

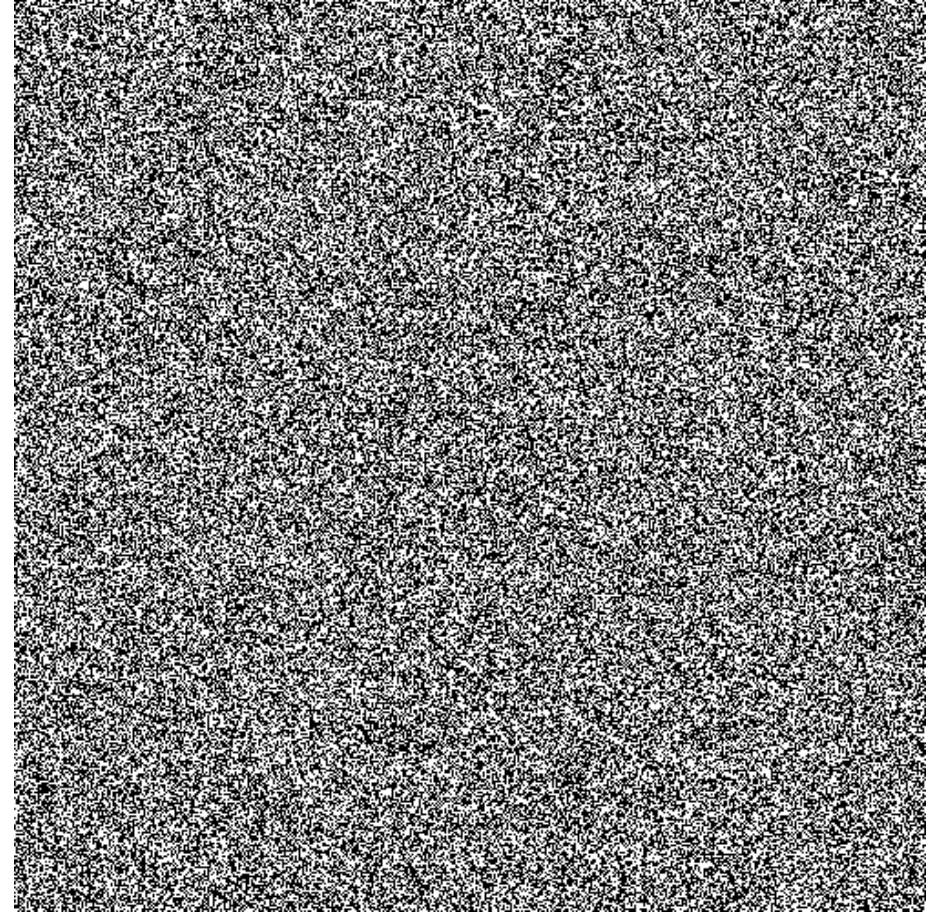
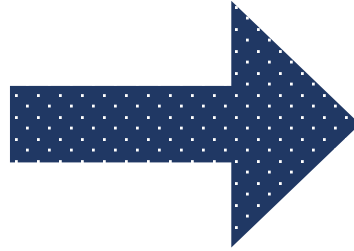
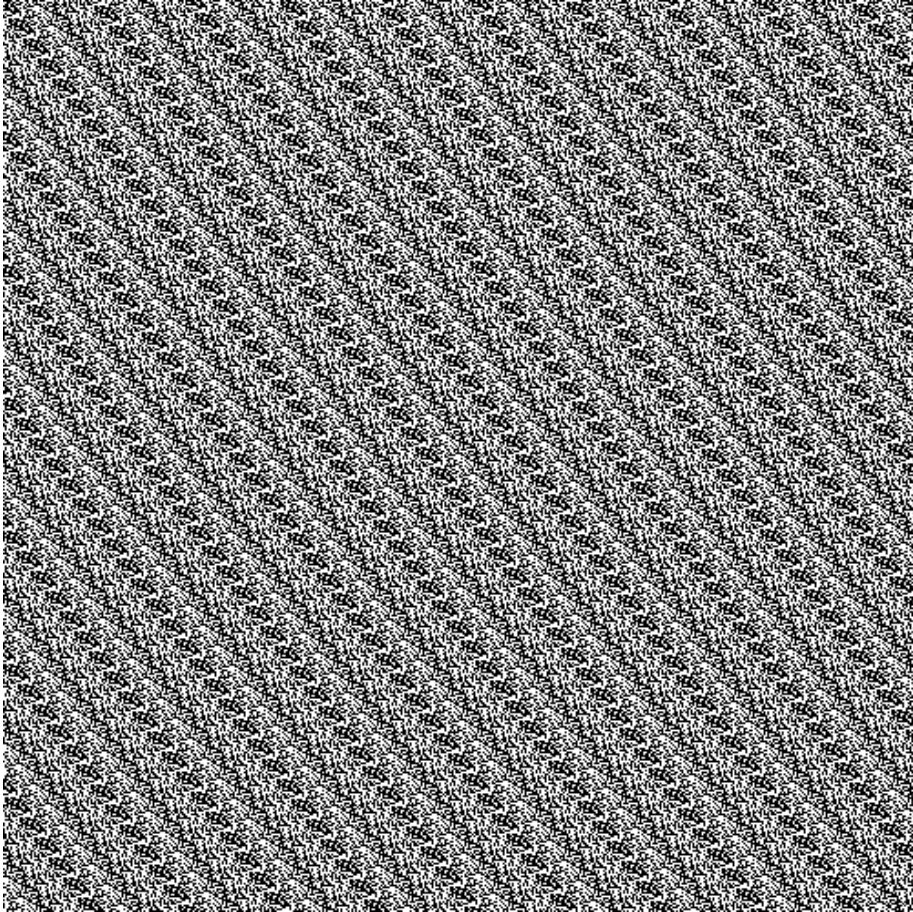
Leakage of decoding
information



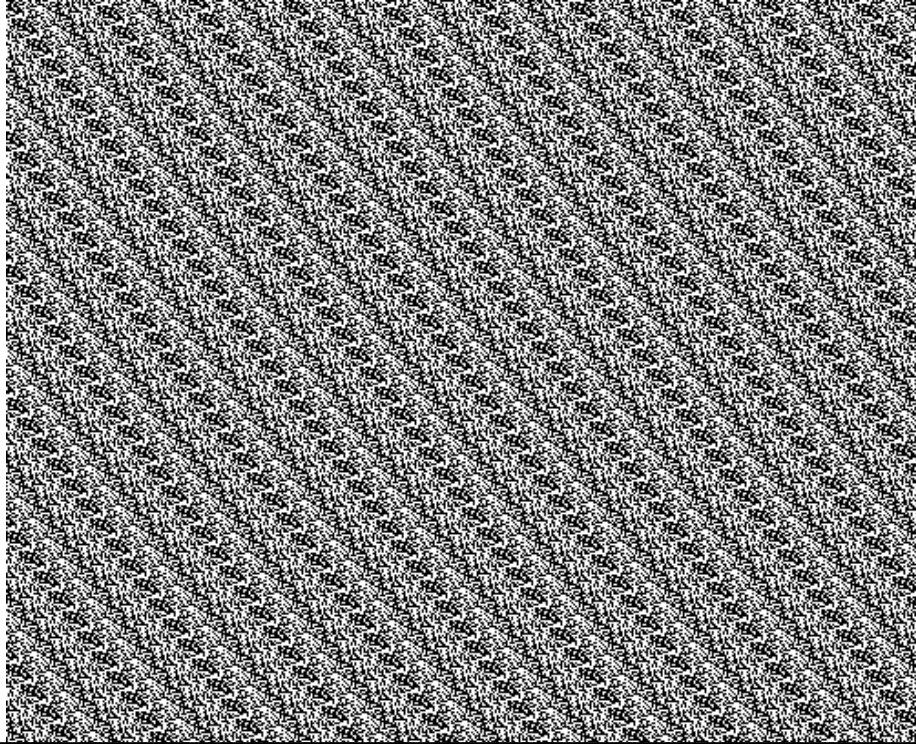
• k/n is close to 1

Easy SDP

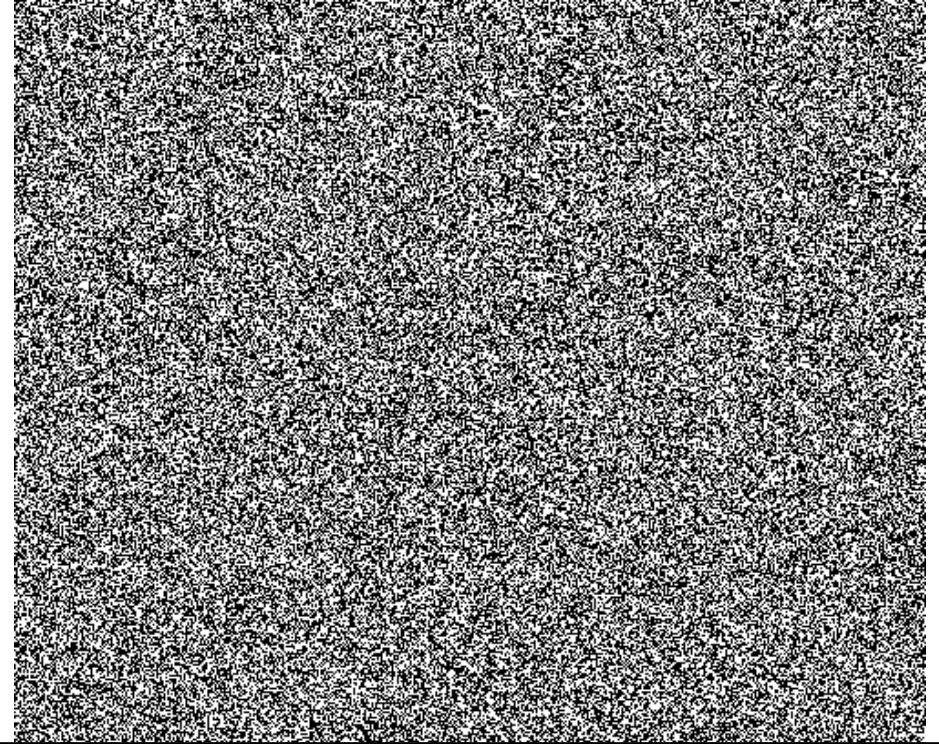
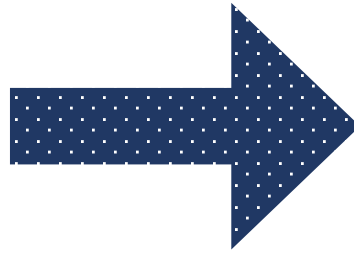
Scrambled Parity-check Matrix \hat{H}



Scrambled Parity-check Matrix \hat{H}

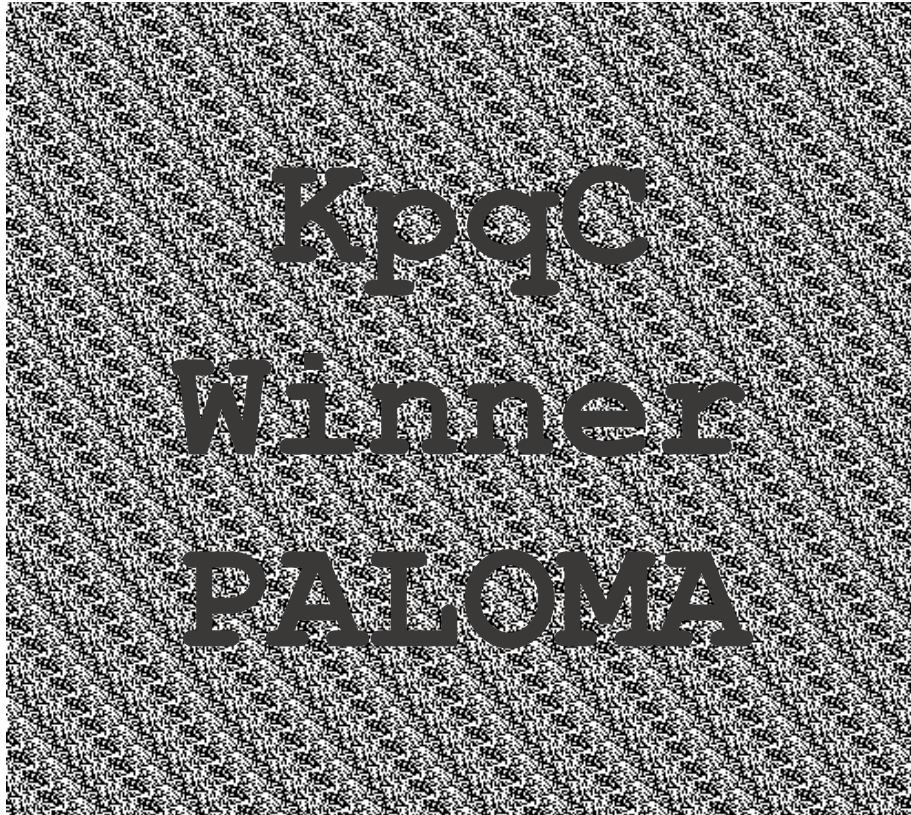


Easy SDP

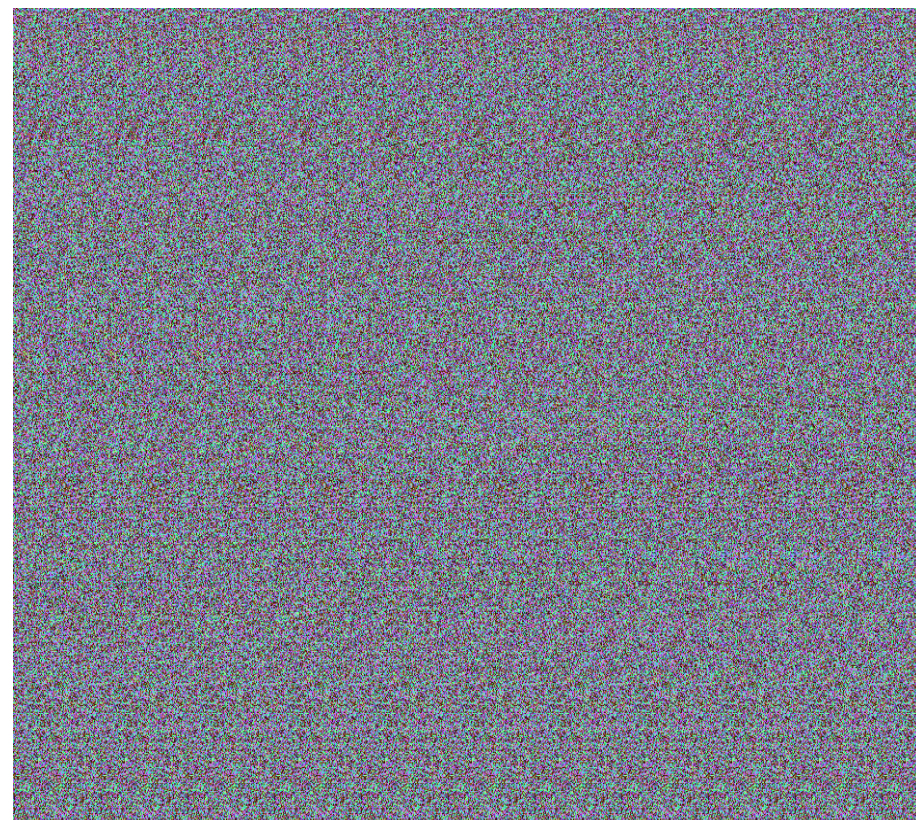
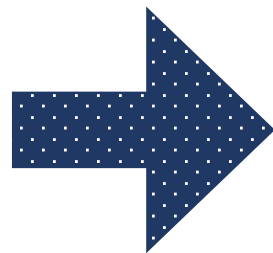
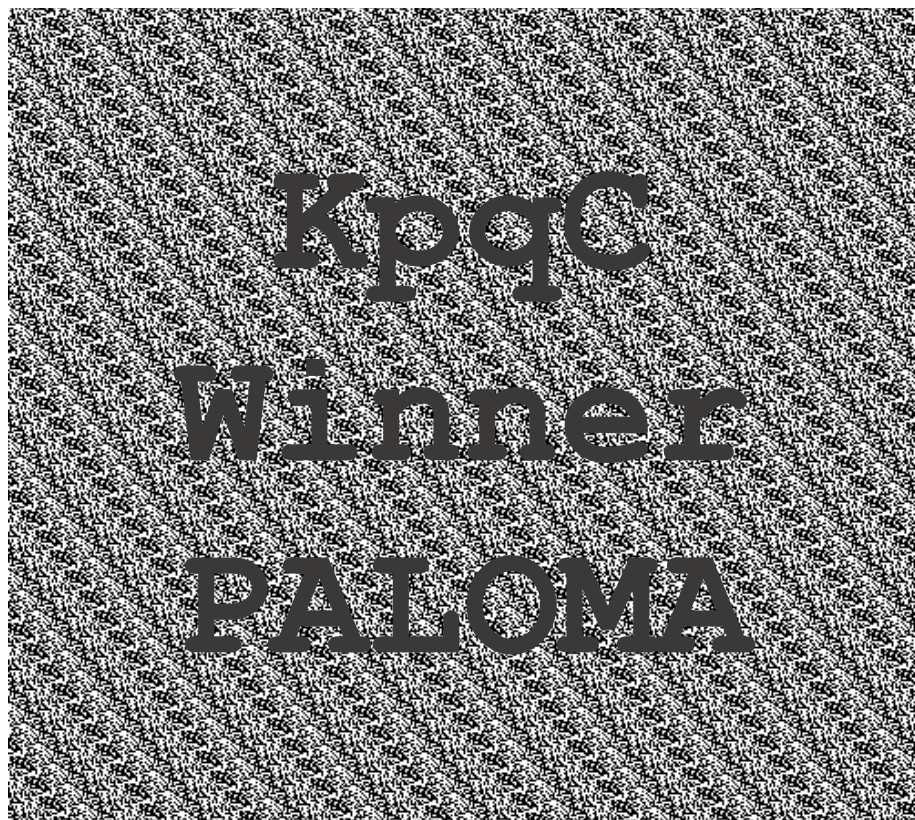


Difficult SDP

Scrambled Parity-check Matrix \hat{H}



Scrambled Parity-check Matrix \hat{H}



Trapdoor based on SDP

C

Efficient
Decoding Alg.
exists

Trapdoor based on SDP

C

Efficient
Decoding Alg.

Easy SDP

H

(parity-check mat.)

Trapdoor based on SDP

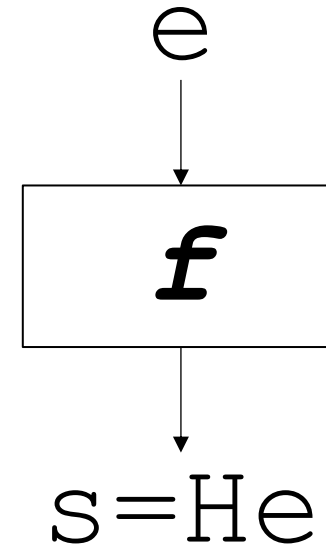
C

Efficient
Decoding Alg.

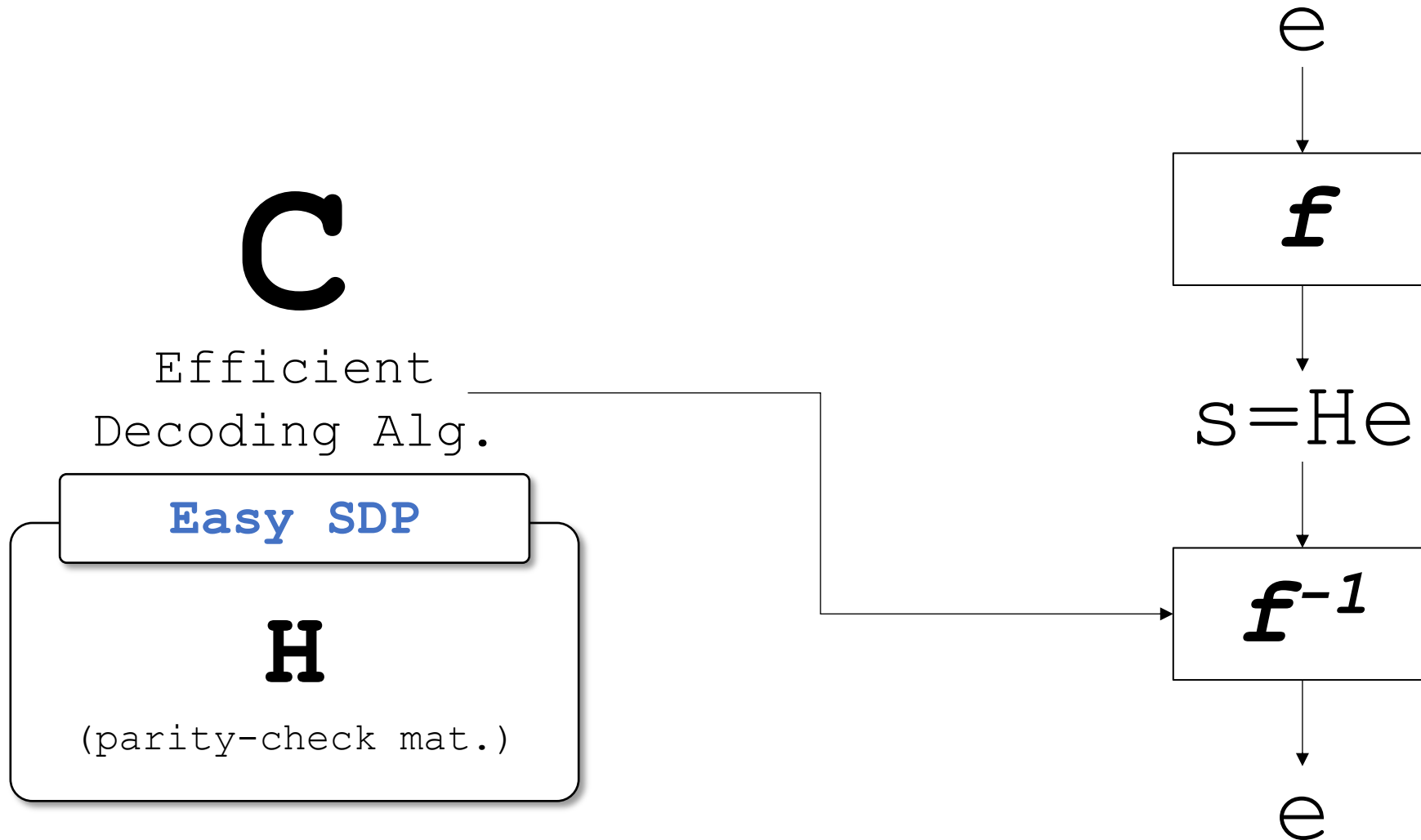
Easy SDP

H

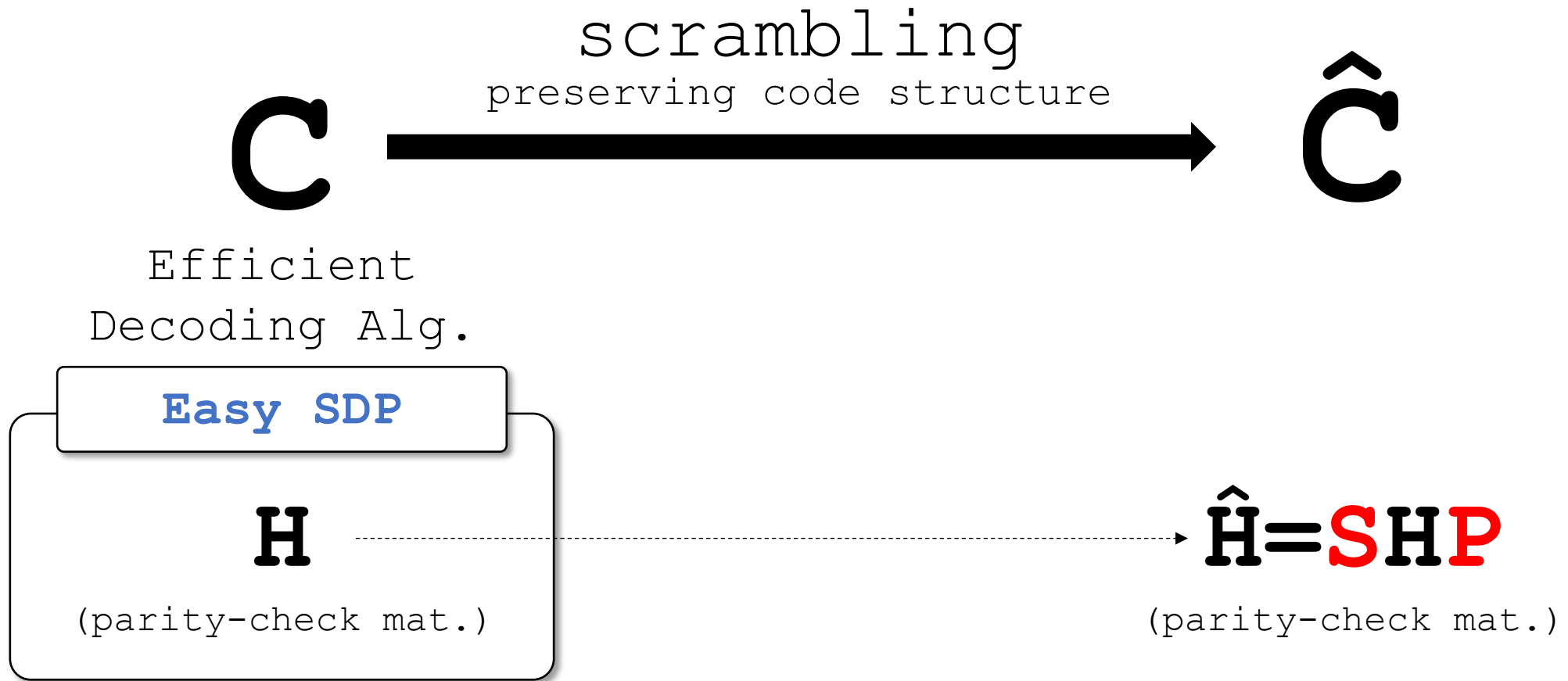
(parity-check mat.)



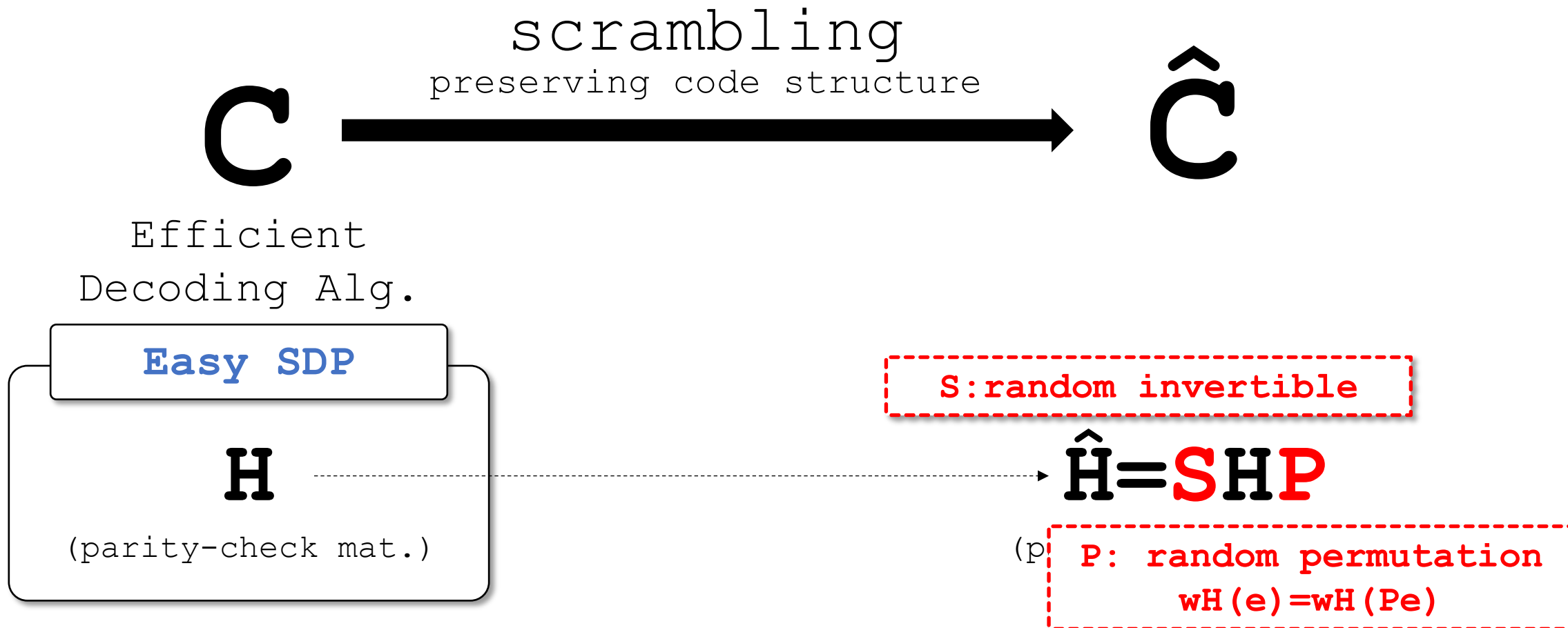
Trapdoor based on SDP



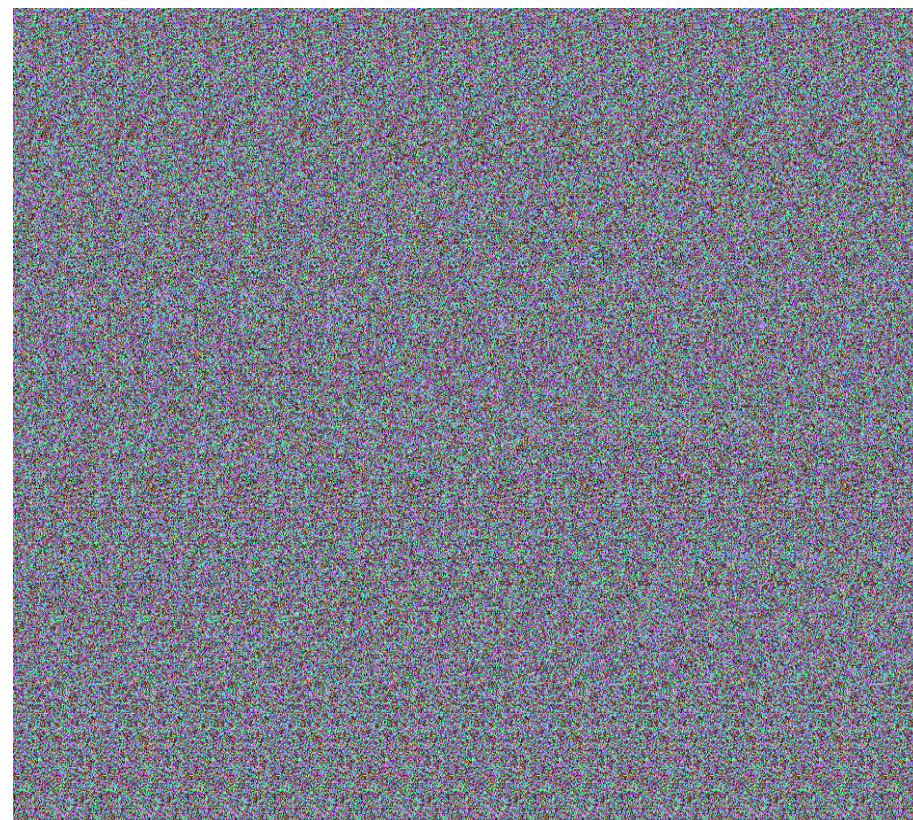
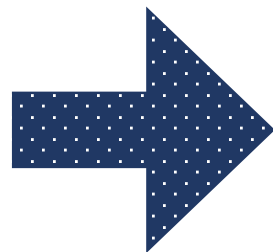
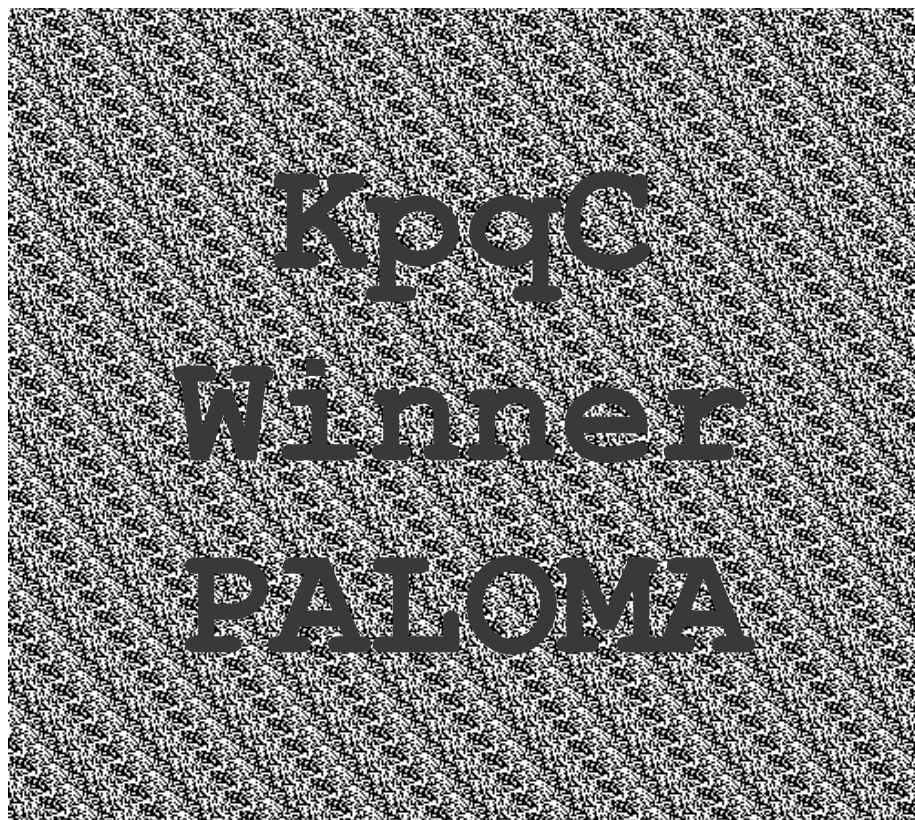
Trapdoor based on SDP



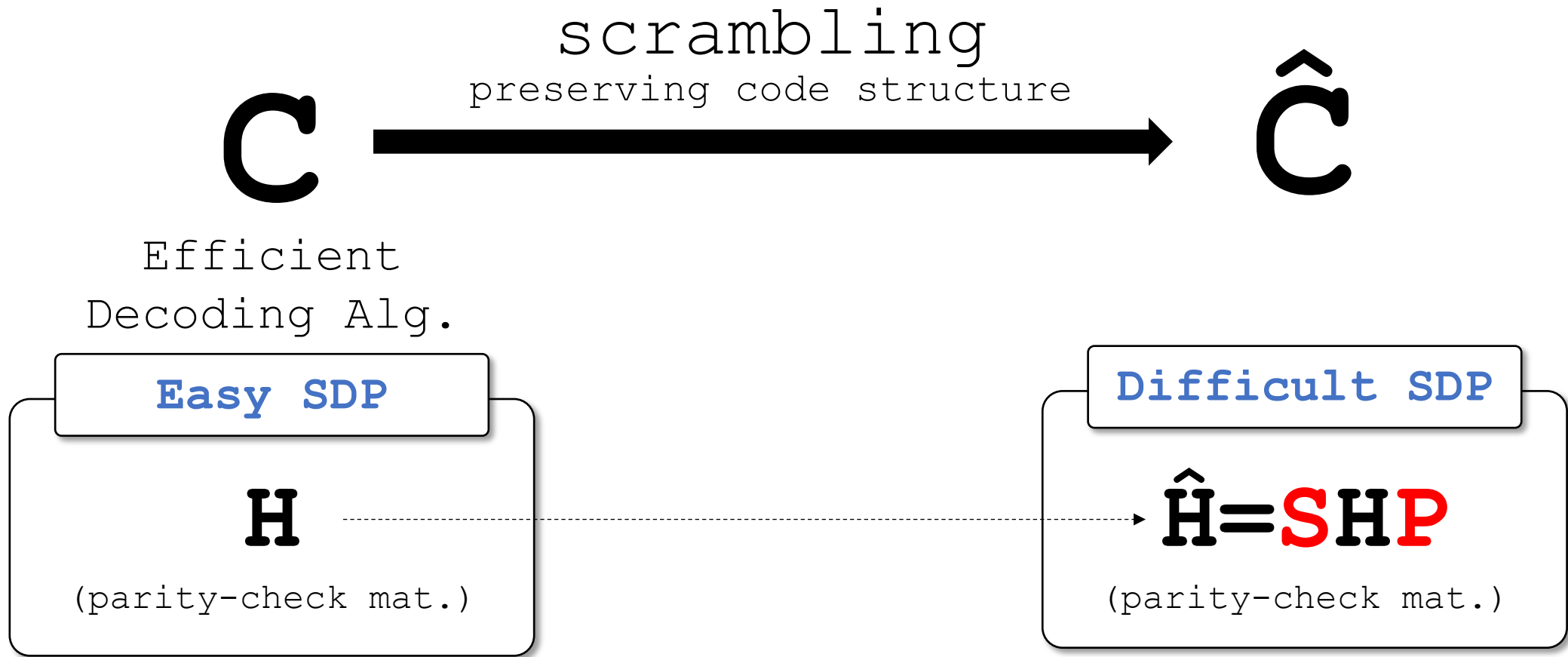
Trapdoor based on SDP



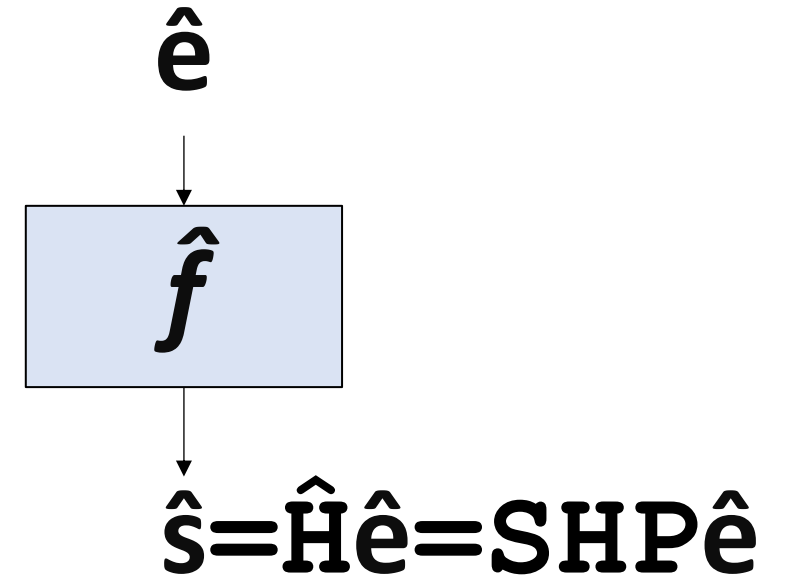
Scrambled Parity-check Matrix \hat{H}



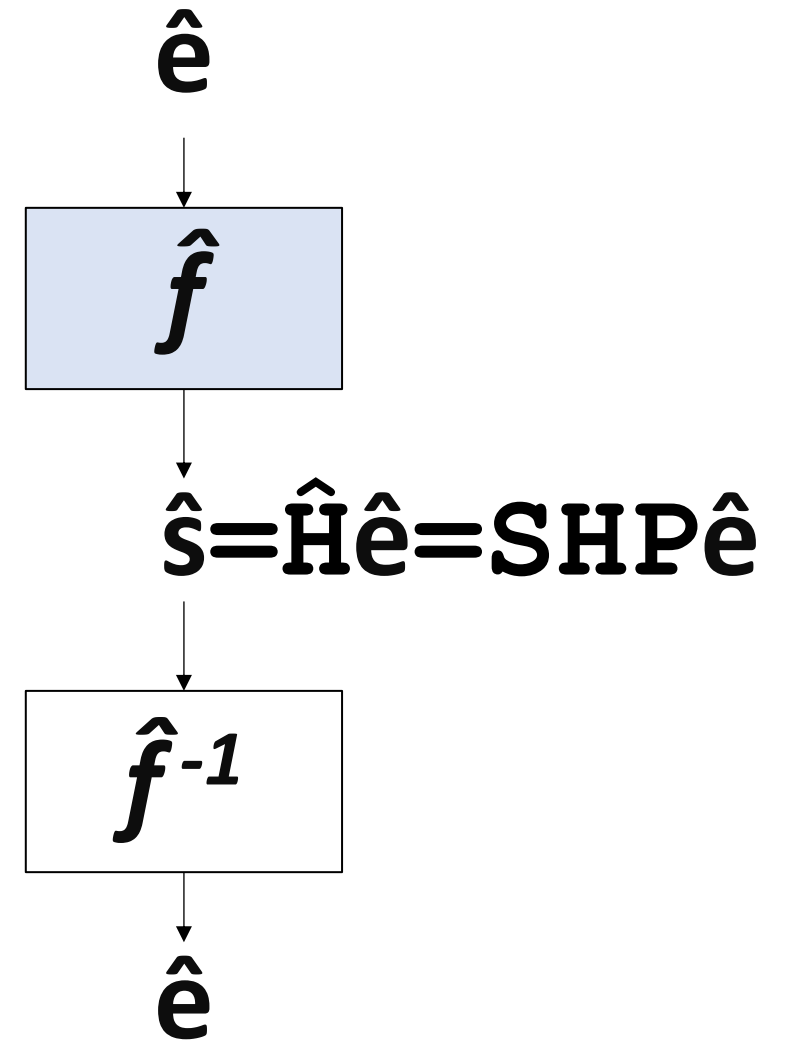
Trapdoor based on SDP



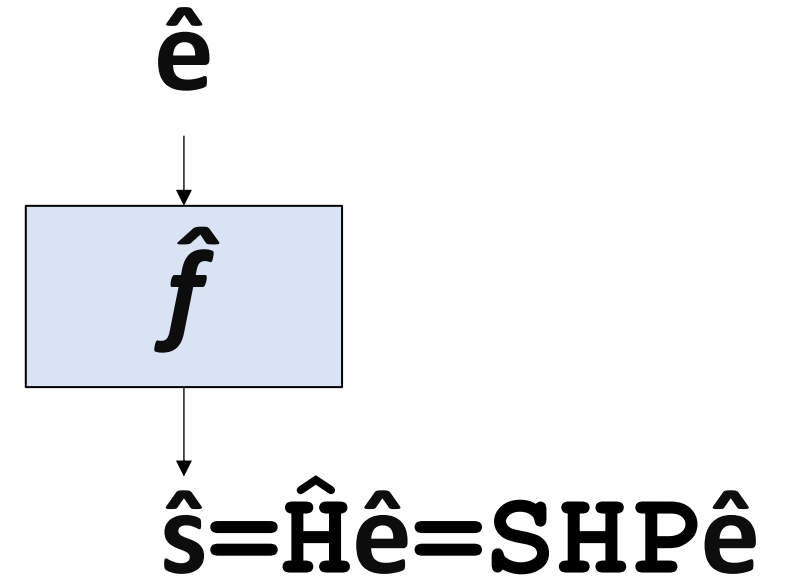
Trapdoor based on SDP



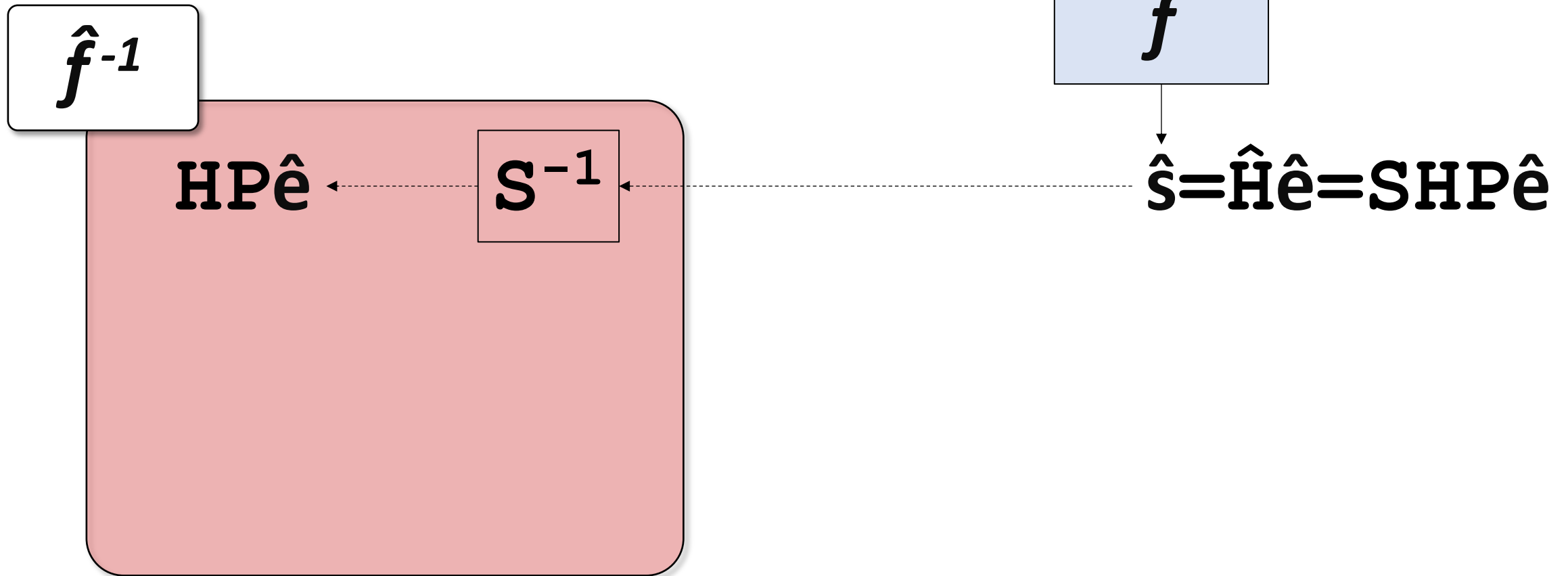
Trapdoor based on SDP



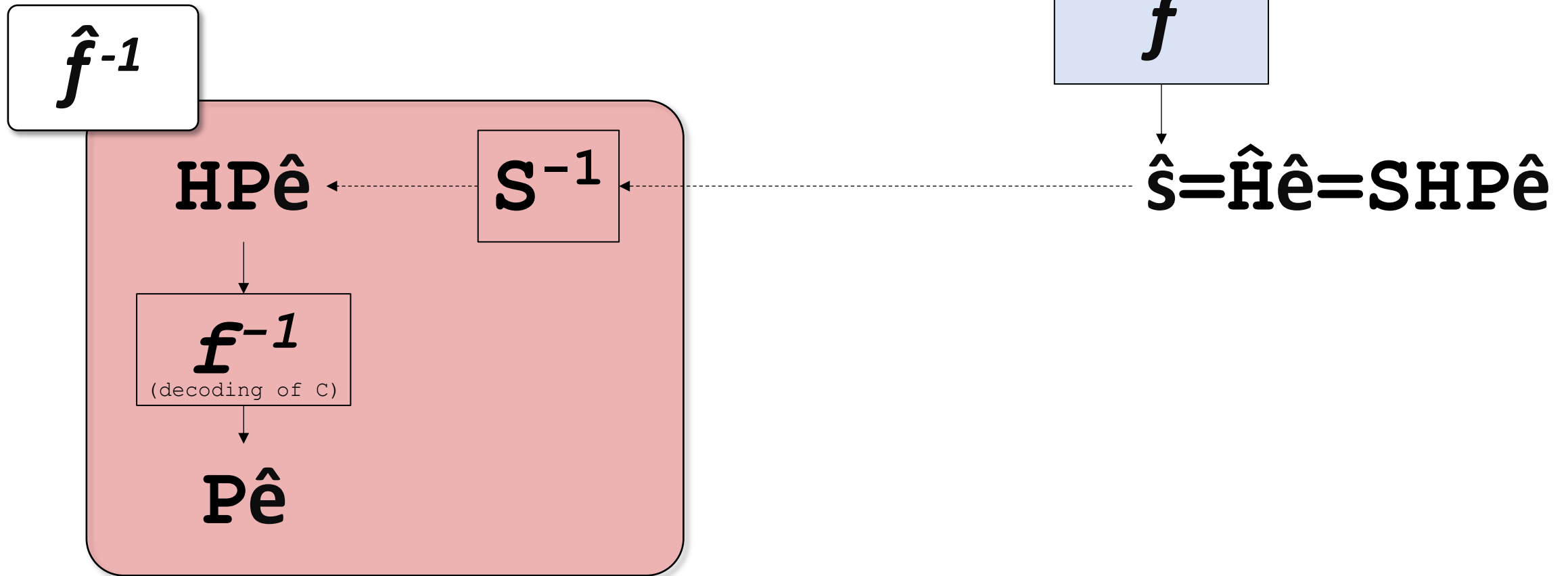
Trapdoor based on SDP



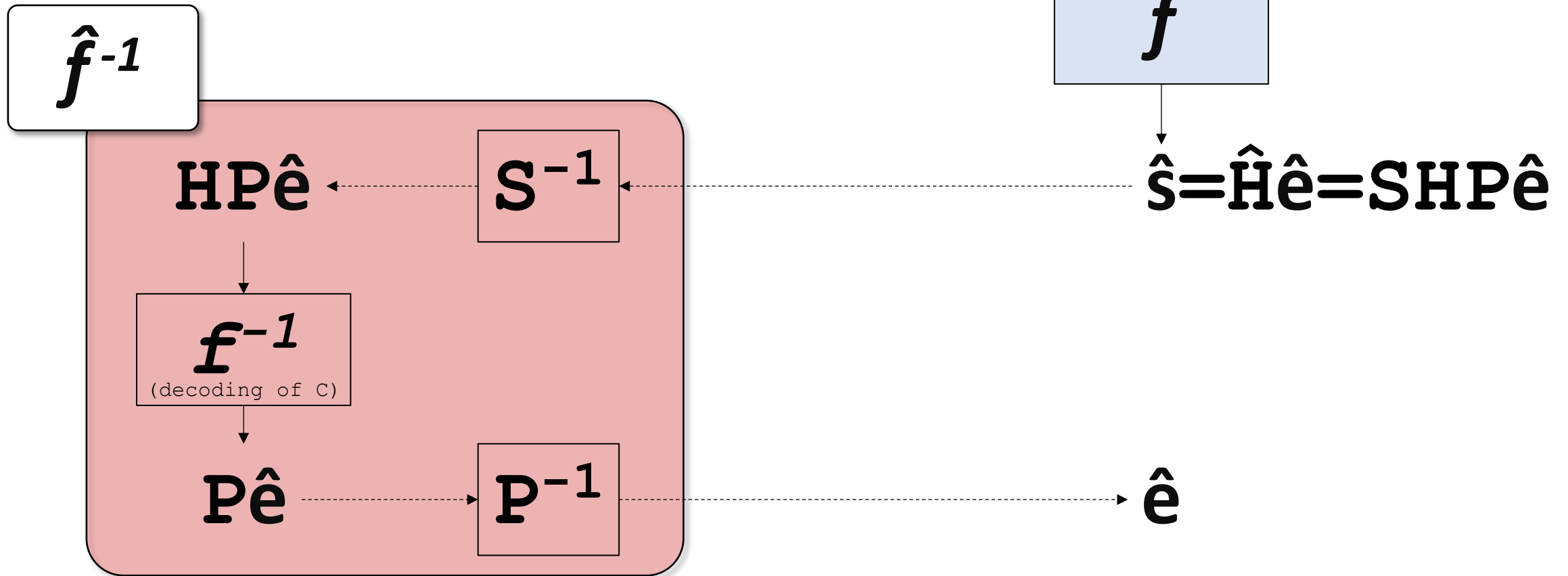
Trapdoor based on SDP



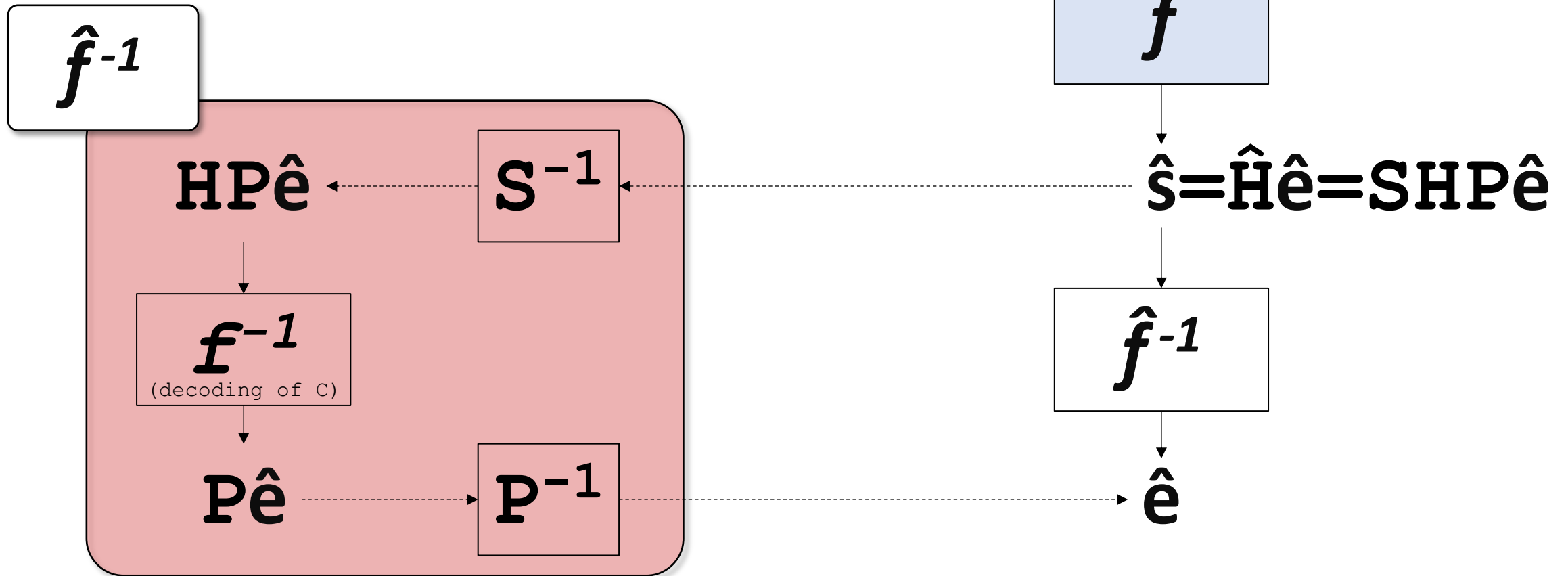
Trapdoor based on SDP



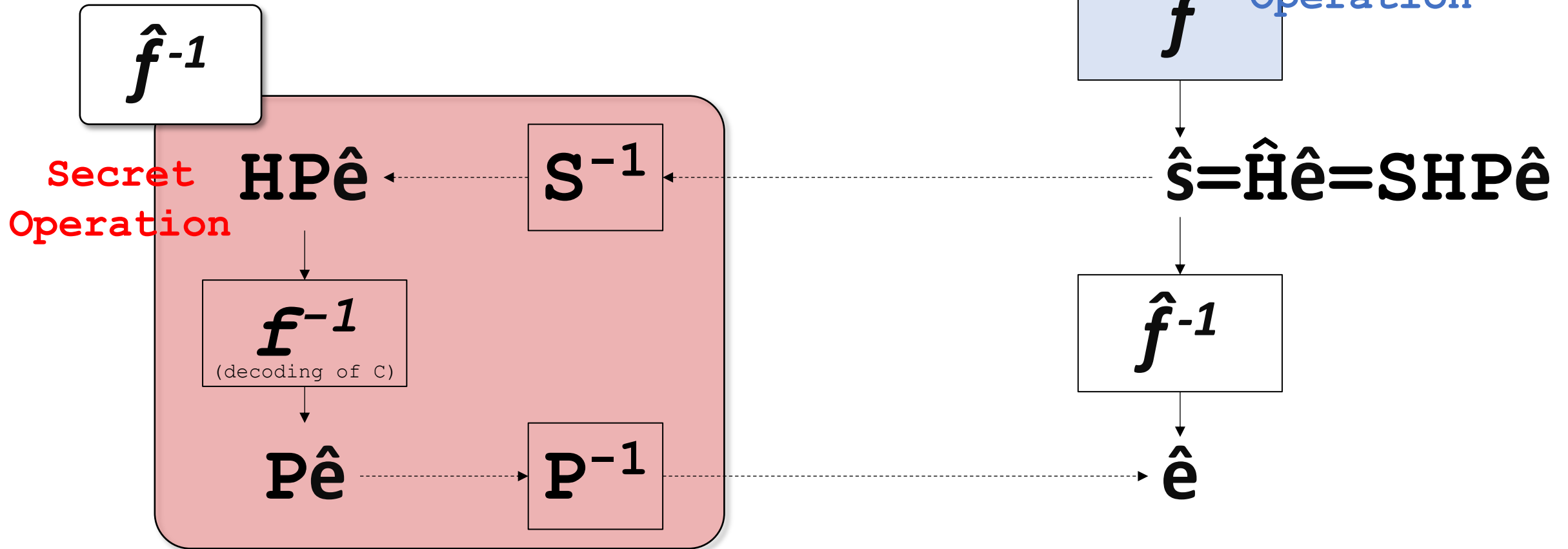
Trapdoor based on SDP



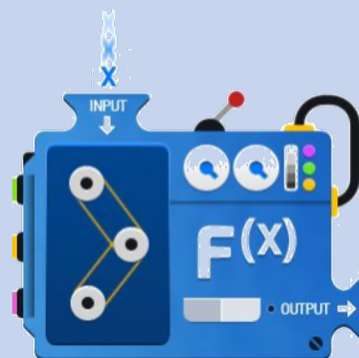
Trapdoor based on SDP



Trapdoor based on SDP



f

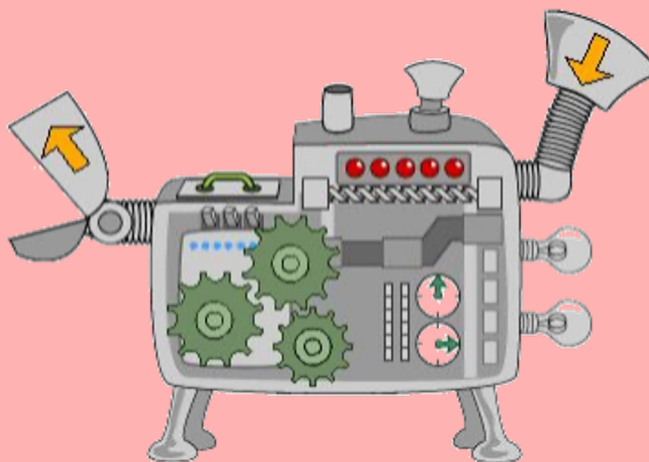


parity-check matrix H of C

error vector e
($wH(e) = t$)

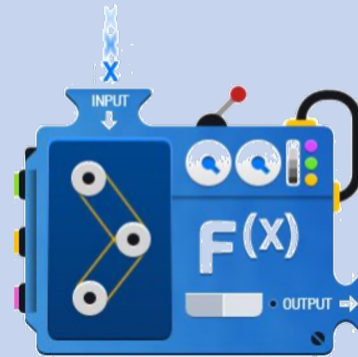
syndrome $s = He$

f^{-1}



decoding of C

\hat{f} (onewayness)



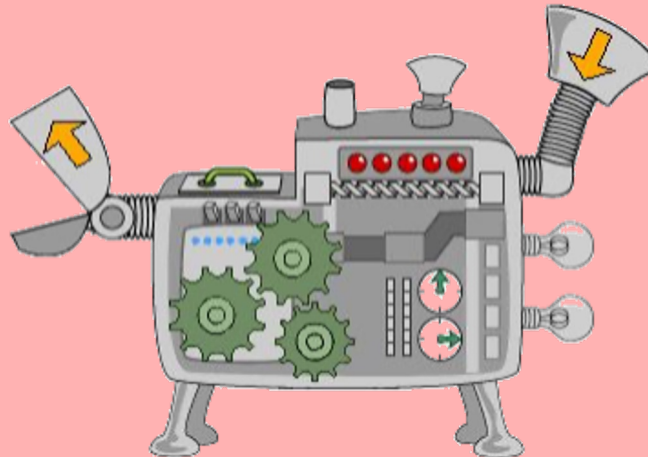
parity-check matrix \hat{H} of \hat{C}

error vector \hat{e}
($w_H(\hat{e}) = t$)

syndrome $\hat{s} = \hat{H}\hat{e}$

\hat{f}^{-1}

P^{-1}



decoding of C

S^{-1}

$\hat{f}_{(onewayness)}$

public operation



parity-check matrix \hat{H} of \hat{C}

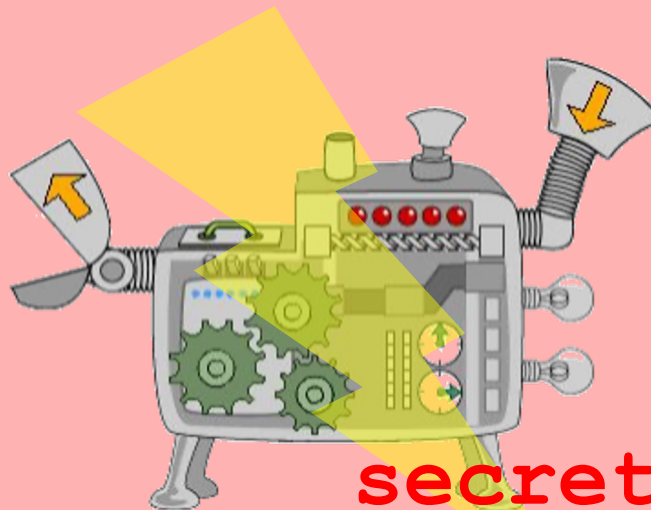
error vector \hat{e}
($wH(\hat{e}) = t$)

syndrome $\hat{s} = \hat{H}\hat{e}$

\hat{f}^{-1}

P^{-1}

S^{-1}



secret operation

decoding of C

Encrypt

public operation



parity-check matrix \hat{H} of \hat{C}

Plaintext

error vector \hat{e}
($wH(\hat{e}) = t$)

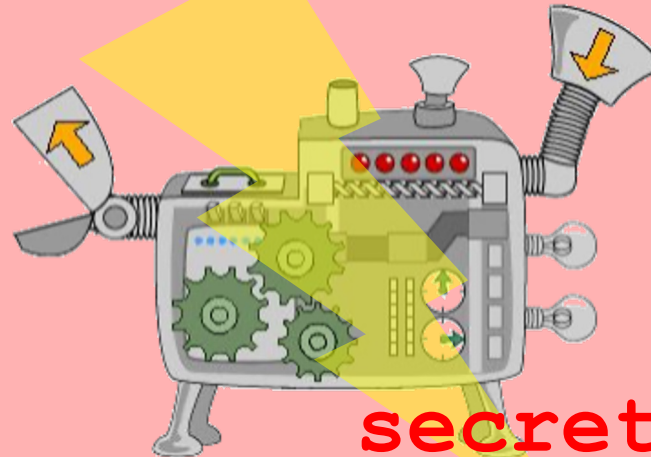
Ciphertext

syndrome s

P^{-1}

S^{-1}

Decrypt



secret operation

decoding of C

Encrypt

public operation



parity-check matrix \hat{H} of \hat{C}

Plaintext

error vector \hat{e}
($wH(\hat{e}) = t$)

Ciphertext

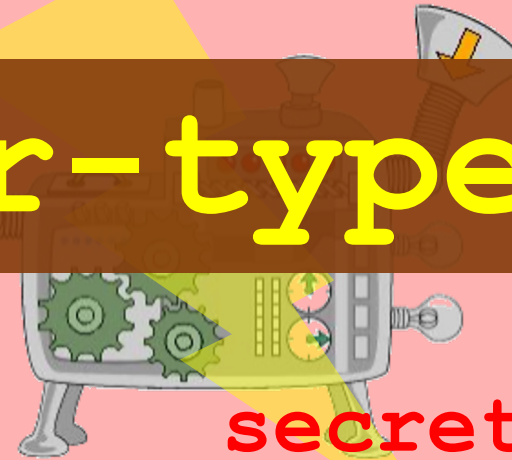
syndrome s

Niederreiter-type trapdoor

Decrypt

secret operation

decoding of C



Goppa Codes

Valery Denisovich Goppa (1970). "A New Class of Linear Error Correcting Codes". Problemy Peredachi Informatsii.

Binary Goppa code – definition

1. **Support set** $L = \{\alpha_0, \dots, \alpha_{n-1}\} \subset \mathbb{F}_2^m$ such that $\alpha_i \neq \alpha_j$ for $i \neq j$
2. **Goppa polynomial** $g(X) = \sum_j g_j X^j$ in $\mathbb{F}_2^m[X]$ such that
 - (1) separable, (2) $\deg(g) = t$ and (3) $g(\alpha_j) \neq 0$ for all $\alpha_j \in L$
(all zeros are distinct)

$$C_{L,g} := \{ (c_0, \dots, c_{n-1}) \in \mathbb{F}_2^n : \sum_j c_j (X - \alpha_j)^{-1} \equiv 0 \pmod{g(X)} \}$$

dimension $k \geq n - mt$

minimum Hamming distance $d(C_{L,g}) \geq 2t + 1$

Binary Goppa code – parity-check matrix

$$H = A \times B \times C$$

$$\begin{pmatrix} g_1 & g_2 & \cdots & g_t \\ g_2 & g_3 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ g_t & 0 & \cdots & 0 \end{pmatrix} \begin{pmatrix} \alpha_0^0 & \alpha_1^0 & \cdots & \alpha_{n-1}^0 \\ \vdots & \vdots & \cdots & \vdots \\ \alpha_0^{t-2} & \alpha_1^{t-2} & \ddots & \alpha_{n-1}^{t-2} \\ \alpha_0^{t-1} & \alpha_1^{t-1} & \cdots & \alpha_{n-1}^{t-1} \end{pmatrix} \begin{pmatrix} g(\alpha_0)^{-1} & 0 & \cdots & 0 \\ 0 & g(\alpha_1)^{-1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & g(\alpha_{n-1})^{-1} \end{pmatrix}$$

invertible

Since A is invertible, BC is also a parity-check matrix of C . **Classic McEliece** uses $H=BC$ for **Berlekamp-Massey** decoding, but **PALOMA** uses $H=ABC$ for **Patterson** decoding.

Goppa-SDP-based schemes

- 1978. **McEliece** cryptosystem (PKE)
 - Binary Goppa code
- 1986. **Niederreiter** cryptosystem (PKE)
 - RS code (broken) → Binary Goppa code (secure)
- 2017. **Classic McEliece** (KEM, NIST PQC 4th round)
 - Binary Goppa code

Contents

1. Preliminaries (ECC / SDP / code-based trapdoor / binary Goppa code)
- 2. The design rationale of PALOMA**
3. Specification (trapdoor / KEM / parameter sets)
4. Evaluation of security (trapdoor / KEM)
5. Performance (data size / speed)
6. Comparison between PALOMA and Classic McEliece

The design rationale of PALOMA

PALOMA priorities security over efficiency

1. To be a secure scheme conservatively, we choose

- SDP (NP-hard),
- binary Goppa code (no critical attack so far), and
- Fujisaki-Okamoto structure KEM (IND-CCA2-secure in ROM).

2. To reduce a key size, we choose

- Niederreiter-type trapdoor using a systematic parity-check matrix $\hat{H} = SHP = [I|M]$ (best option when a Goppa code is used), and
- Random components derived from a 256-bit seed.

3. To be a constant-time scheme, we choose

- separable (not irreducible) Goppa code (extended Patterson decoding), and
- deterministic Fisher-Yates-based shuffling with a 256-bit seed.

Contents

1. Preliminaries (ECC / SDP / code-based trapdoor / binary Goppa code)
2. The design rationale of PALOMA
- 3. Specification (trapdoor / KEM / parameter sets)**
4. Evaluation of security (trapdoor / KEM)
5. Performance (data size / speed)
6. Comparison between PALOMA and Classic McEliece

Binary Separable Goppa Codes

and

Extended Patterson Decoding

Binary Separable Goppa Codes

L and $g(x)$



[0, 1, ..., 8191]



3703, 6511, 7140, 2709, 1594, 8153, 2014, 8046, 1209, 5813, 6837, 5347, 782, 2302, 2445, 3516, 6530, 5138, 319, 6332, 2523, 7756, 6659, 922, 1817, 6807, 5369, 6410, 6649, 2190, 5012, 5864, 7067, 2352, 7477, 5113, 2167, 2543, 6100, 713, 3559, 482, 2992, 7401, 4479, 4001, 1708, 1, 4759, 6543, 4009, 3217, 4919, 1667, 2163, 337, 5211, 5477, 4161, 1596, 7979, 471, 4456, 2726, 2552, 4027, 7920, 7823, 6889, 3309, 4995, 7087, 5907, 322, 261, 1658, 99, 4129, 973, 4870, 7371, 277, 3902, 907, 4296, 7129, 657, 3980, 4691, 6938, 2673, 6463, 6810, 6643, 6114, 2425, 7943, 5265, 7662, 4197, 6126, 7776, 5413, 4918, 4207, 4390, 1621, 7314, 6914, 3524, 3323, 5205, 5692, 1816, 2996, 4023, 3177, 3191, 5721, 753, 825, 4386, 7078, 4609, 22, 2987, 3803, 5844, 6419, 2421, 3494, 2988, 6560, 3785, 2246, 2455, 1713, 2573, 8016, 8070, 631, 6206, 4373, 4148, 6701, 6401, 490, 3809, 2210, 7882, 224, 1476, 6953, 4248, 6852, 4876, 5099, 6651, 5011, 7921, 6957, 3096, 6699, 7855, 1006, 3509, 5534, 7939, 2077, 3659, 1308, 1378, ...

[0, 1, ..., 8191]



Deterministic

Fisher-Yates Shuffling

3703, 6511, 7140, 2709, 1594, 8153, 2014, 8046, 1209, 5813, 6837, 5347, 782, 2302, 2445, 3516, 6530, 5138, 319, 6332, 2523,
659, 922, 817, 6807, 5369, 6410, 84, 2190, 5112, 5864, 7067, 2352, 787, 1113, 2167, 2163, 6707, 71, 3558, 482,
2, 7101, 349, 4011, 1289, 4758, 5543, 4004, 3211, 494, 416, 2163, 337, 321, 347, 4161, 1596, 7979, 471, 456, 2126,
52, 227, 700, 781, 381, 299, 4995, 387, 5637, 22, 71, 158, 99, 412, 0, 3, 4130, 741, 277, 1092, 617, 4156, 112,
657, 3980, 4691, 6938, 2673, 6463, 6810, 6643, 6114, 2425, 7943, 5265, 7662, 4197, 6126, 7776, 5413, 4918, 4207, 4390, 1621,
7314, 6914, 3524, 3323, 5205, 5692, 1816, 2996, 4023, 3177, 3191, 5721, 753, 825, 4386, 7078, 4609, 22, 2987, 3803, 5844, 6419,
2421, 3494, 2988, 6560, 3785, 2246, 2455, 1713, 2573, 8016, 8070, 631, 6206, 4373, 4148, 6701, 6401, 490, 3809, 2210, 7882,
224, 1476, 6953, 4248, 6852, 4876, 5099, 6651, 5011, 7921, 6957, 3096, 6699, 7855, 1006, 3509, 5534, 7939, 2077, 3659, 1308,
1378, ...

PALOMA Shuffle (a.k.a. deterministic Fisher-Yates Shuffle)

Algorithm 6 Shuffle: Array Shuffling with a 256-bit Seed

Input: An array $A = [A_0, A_1, \dots, A_{l-1}]$ and a 256-bit seed $r \in \{0, 1\}^{256}$

Output: A shuffled array A

```
1: procedure Shuffle( $A, r$ )
2:   for  $j = 0$  to 15 do
3:      $\hat{r}_j \leftarrow \text{BitToInt}(r_{[16j:16(j+1)]})$  ▷  $\text{BitToInt}(r_0 \| \dots \| r_{15}) := \sum_{j=0}^{15} r_{15-j} 2^j < 2^{16}$ 
4:   end for
5:    $w \leftarrow 0$ 
6:   for  $i \leftarrow |A| - 1$  downto 1 do ▷  $|A| = l$ , the number of elements of  $A$ 
7:      $j \leftarrow \hat{r}_w \bmod i + 1$ 
8:      $A_i, A_j \leftarrow A_j, A_i$  ▷ Swapping of  $A_i$  and  $A_j$ 
9:      $w \leftarrow w + 1 \bmod 16$ 
10:  end for
11:  return  $A$ 
12: end procedure
```

$$r \stackrel{\$}{\leftarrow} \{0, 1\}^{256}$$

$$r \stackrel{\$}{\leftarrow} \{0, 1\}^{256}$$

$$[\alpha_0, \alpha_1, \dots, \alpha_{2^{13}-1}] \leftarrow \mathbf{Shuffle}(\mathbb{F}_2^{13}, r)$$

$$r \stackrel{\$}{\leftarrow} \{0, 1\}^{256}$$

$$[\alpha_0, \alpha_1, \dots, \alpha_{2^{13}-1}] \leftarrow \mathbf{Shuffle}(\mathbb{F}_2^{13}, r)$$

$$\mathbf{L} \leftarrow [\alpha_0, \alpha_1, \dots, \alpha_{n-1}]$$

$$g(x) \leftarrow (x - \alpha_n)(x - \alpha_{n+1}) \dots (x - \alpha_{n+t-1})$$

$$r \stackrel{\$}{\leftarrow} \{0, 1\}^{256}$$

$$[\alpha_0, \alpha_1, \dots, \alpha_{2^{13}-1}] \leftarrow \mathbf{Shuffle}(\mathbb{F}_2^{13}, r)$$

$$\mathbf{L} \leftarrow [\alpha_0, \alpha_1, \dots, \alpha_{n-1}]$$

$$g(x) \leftarrow (x - \alpha_n)(x - \alpha_{n+1}) \dots (x - \alpha_{n+t-1})$$

Separable, but not irreducible

Decodings for Binary Goppa Code



Decodings for Binary Goppa Code



Berlekamp-Massey

Patterson

Extended Patterson decoding for a binary separable Goppa code

Patterson decoding

" $g(X)$ is irreducible"

If $g(X)$ is not irreducible,



$$\begin{array}{l} \text{syndrome poly.} \\ \sigma(X) s(X) \equiv \sigma'(X) \pmod{g(X)} \\ \text{error locator poly.} \\ \downarrow \\ b(X)^2 (1 + Xs(X)) \equiv a(X)^2 s(X) \pmod{g(X)} \\ \text{where } \sigma(X) = a(X)^2 + b(X)^2 X \\ \downarrow \\ b(X) (s^{-1}(X) + X)^{1/2} \equiv a(X) \pmod{g(X)} \\ \downarrow \\ \text{Extended Euclid algorithm} \\ \downarrow \\ \sigma(X) = a(X)^2 + b(X)^2 X \end{array}$$

Extended Patterson decoding for a binary separable Goppa code

Extended Patterson decoding

$$b_1(X) u(X)^{1/2} \equiv a_2(X) \pmod{g_{12}(X)}$$

$$\begin{aligned} s^*(X) &= 1 + Xs(X), \\ g_1(X) &= \gcd(g(X), s(X)), \\ g_2(X) &= \gcd(g(X), s^*(X)), \\ a_2(X) &= a(X) / g_2(X), \\ b_1(X) &= b(X) / g_1(X), \\ g_{12}(X) &= g(X) / g_1(X) g_2(X), \\ s_1(X) &= s(X) / g_1(X), \\ s_2^*(X) &= s^*(X) / g_2(X), \\ u(X) &= g_1(X) s_2^*(X) (g_2(X) s_1(X))^{-1}. \end{aligned}$$

$$\begin{array}{l} \text{syndrome poly.} \\ \sigma(X) s(X) \equiv \sigma'(X) \pmod{g(X)} \\ \text{error locator poly.} \end{array}$$

$$\begin{aligned} b(X)^2 (1 + Xs(X)) &\equiv a(X)^2 s(X) \pmod{g(X)} \\ \text{where } \sigma(X) &= a(X)^2 + b(X)^2 X \end{aligned}$$

~~$$b(X) (s^{-1}(X) + X)^{1/2} \equiv a(X) \pmod{g(X)}$$~~

Extended Euclid algorithm

$$\sigma(X) = a(X)^2 + b(X)^2 X$$

A dynamic illustration of two basketball players in red jerseys and shorts, competing for a ball. The player on the left is seen from the back, reaching out with his right arm. The player on the right is facing him, also reaching out. The background is white with scattered black dots and small circles, suggesting motion or a crowd. The text "Binary Separable Goppa Codes" is written in a large, blue-outlined font across the top, "and" is in a smaller font in the middle, and "Extended Patterson Decoding" is in a large, blue-outlined font across the bottom.

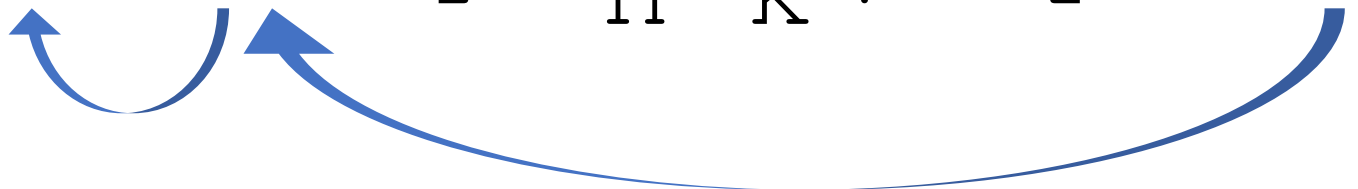
Binary Separable Goppa Codes and Extended Patterson Decoding

Random generation of Binary separable Goppa code

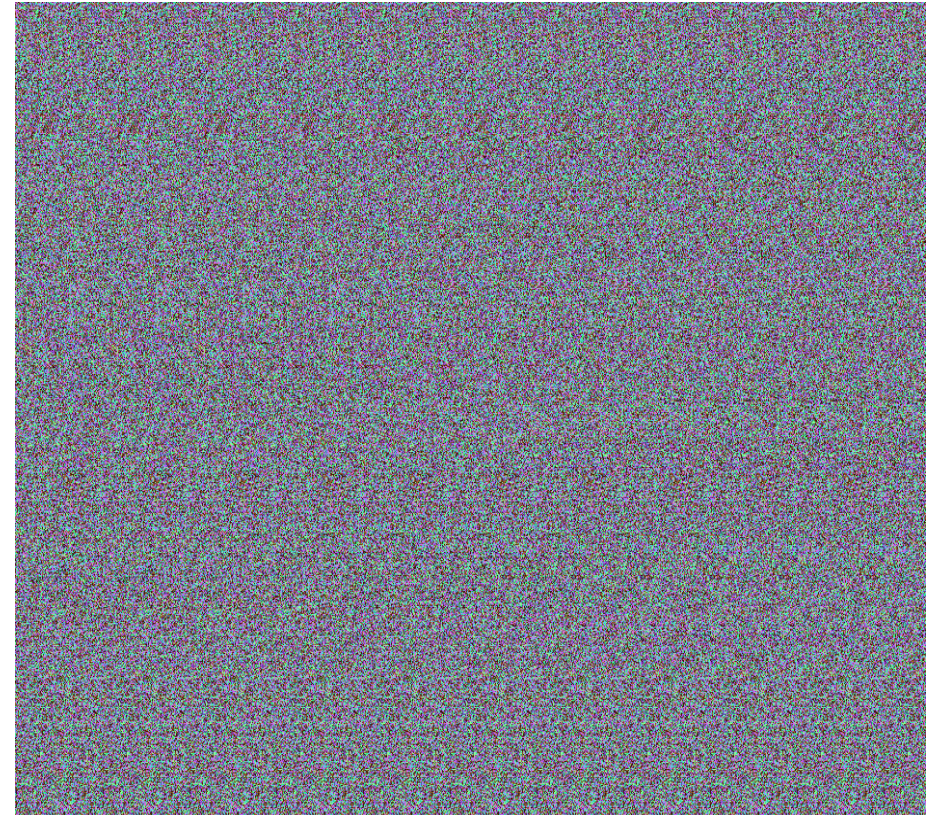
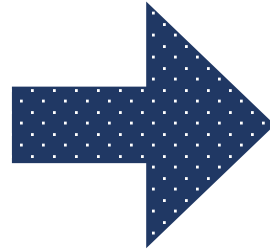
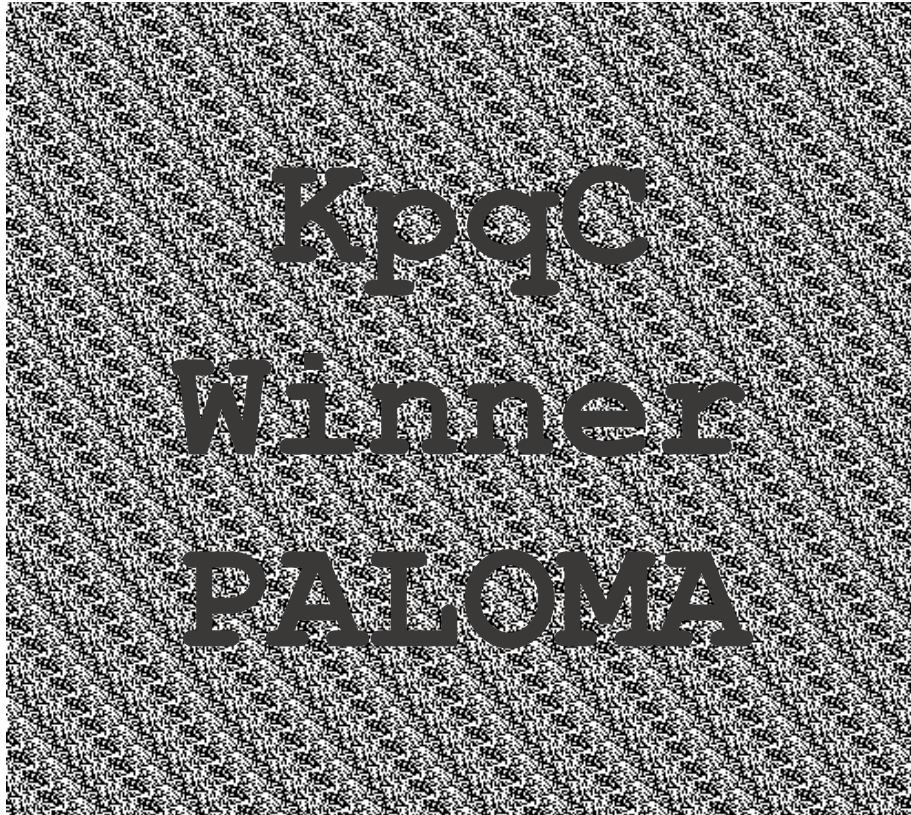
1. Choose a 256-bit string r uniformly
2. Shuffle \mathbb{F}_2^{13} with the r deterministic Fisher-Yates
 - $[\alpha_0, \alpha_1, \dots, \alpha_{2^{13}-1}] \leftarrow \mathbf{Shuffle}(\mathbb{F}_2^{13}, r)$
3. Set a support set L and a Goppa polynomial $g(X)$:
 - $L \leftarrow [\alpha_0, \alpha_1, \dots, \alpha_{n-1}]$ Separable, but not irreducible
 - $g(X) \leftarrow (X - \alpha_n)(X - \alpha_{n+1}) \dots (X - \alpha_{n+t-1})$ → constant-time operation
4. Compute the parity-check matrix **$H=ABC$** from L and $g(X)$
Extended Patterson Decoding

PALOMA - Scrambling code ($H \rightarrow \hat{H} = \mathbf{SHP}$)

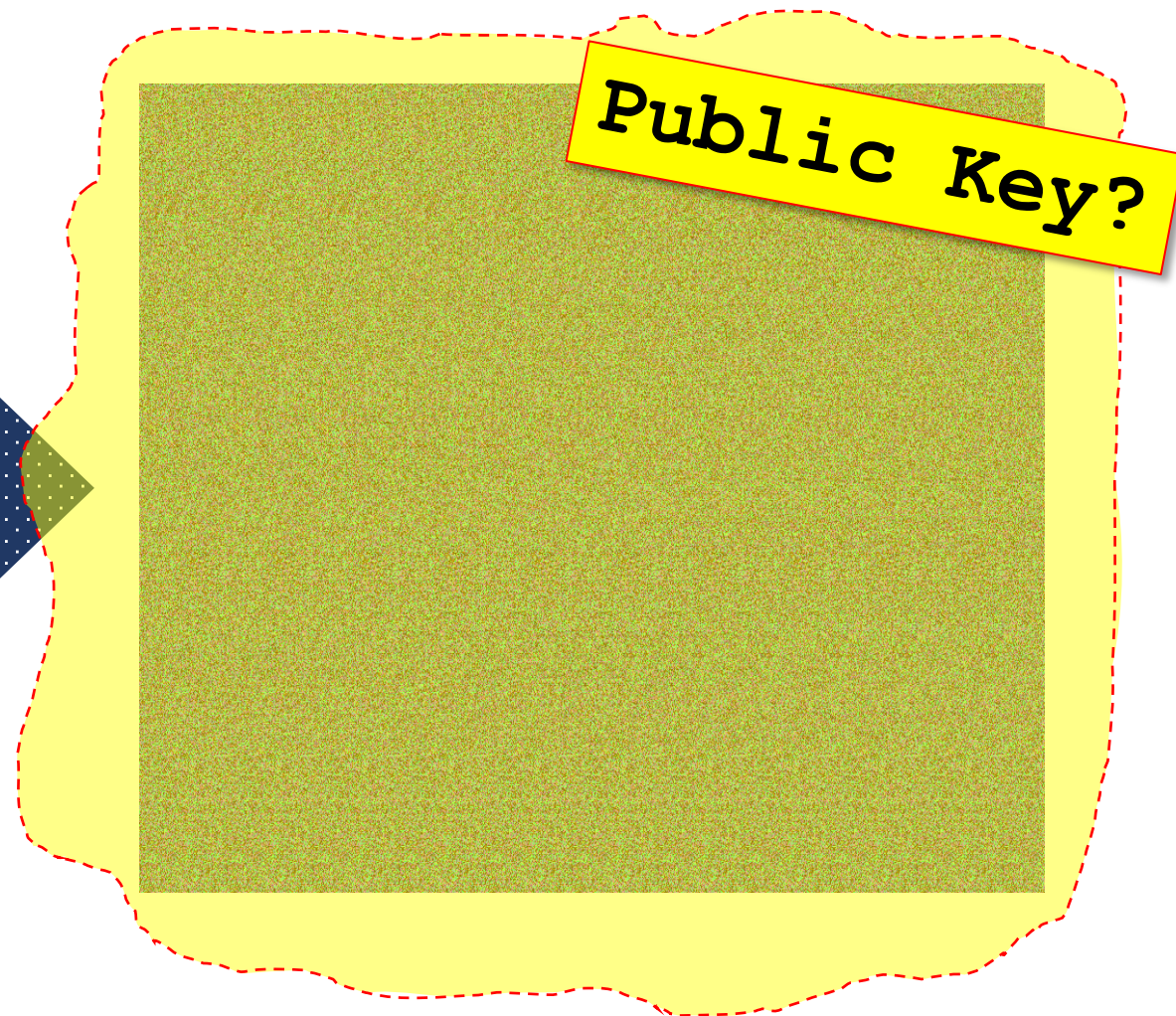
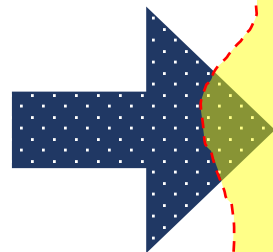
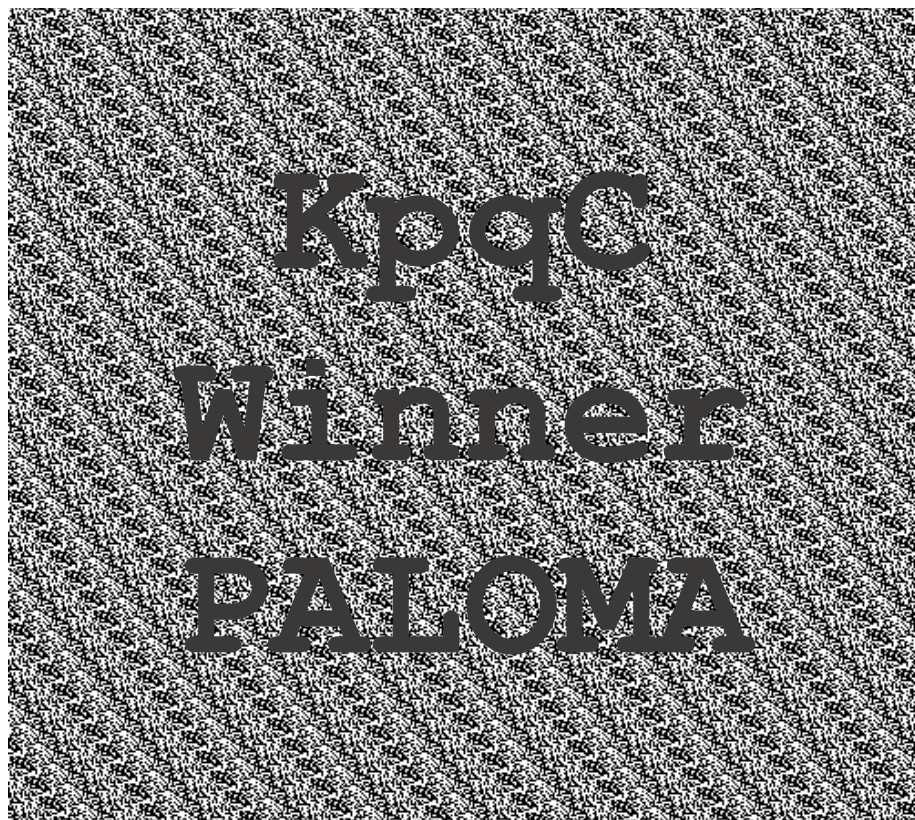
1. Choose a 256-bit string r uniformly
2. Generate a permutation matrix P with the r
3. $\hat{H} \leftarrow \mathbf{RREF}(\mathbf{HP})$ (Reduced Row Echelon Form)
4. If $\hat{H}_{[0:n-k]} \neq I_{n-k}$, then go back to Step 1. (prob. > 0.28)

$$\hat{H} = \mathbf{SHP} = [I_{n-k} \mid M]$$


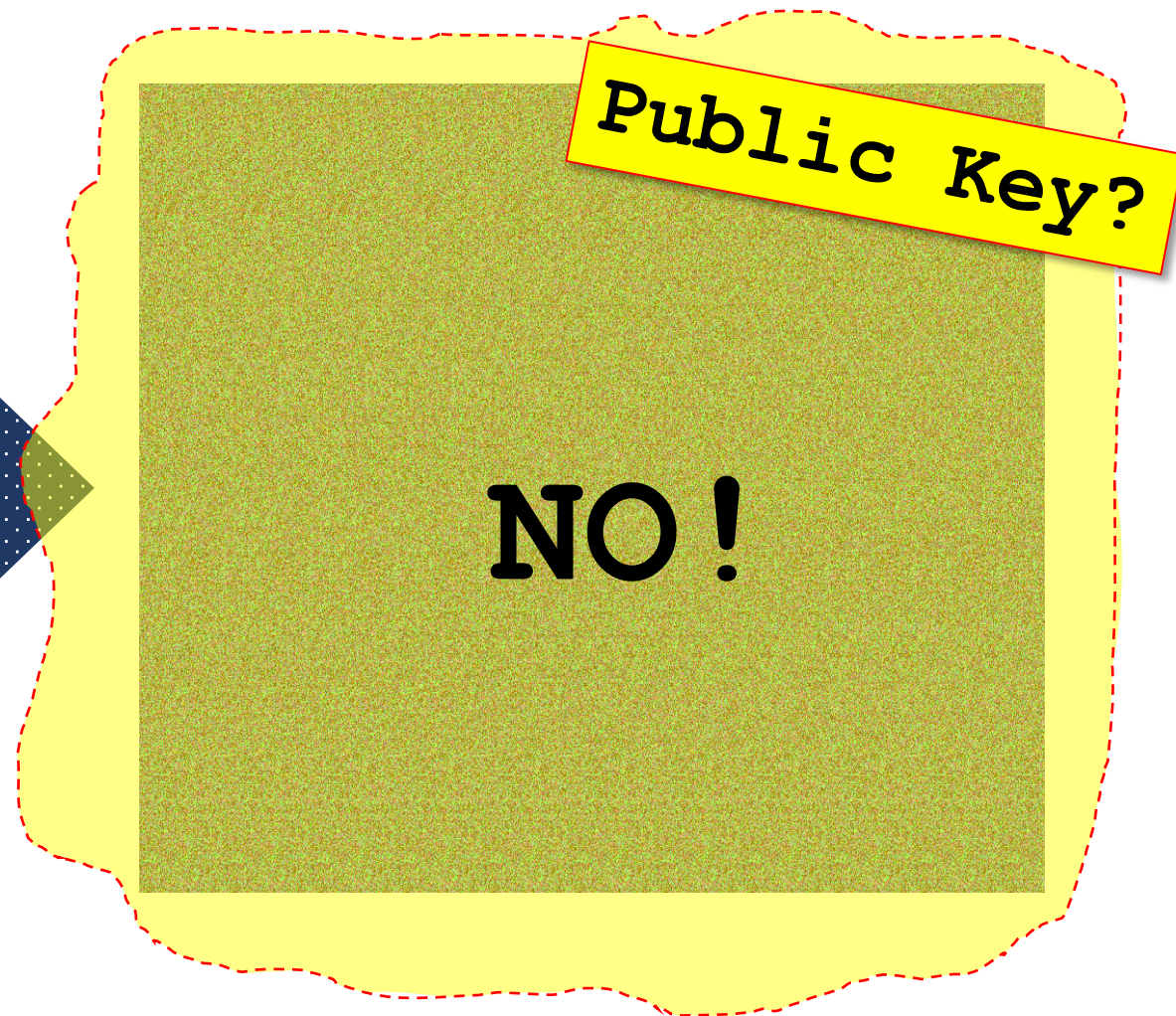
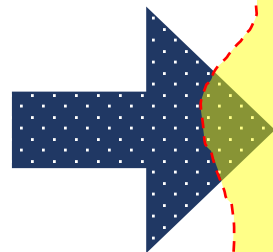
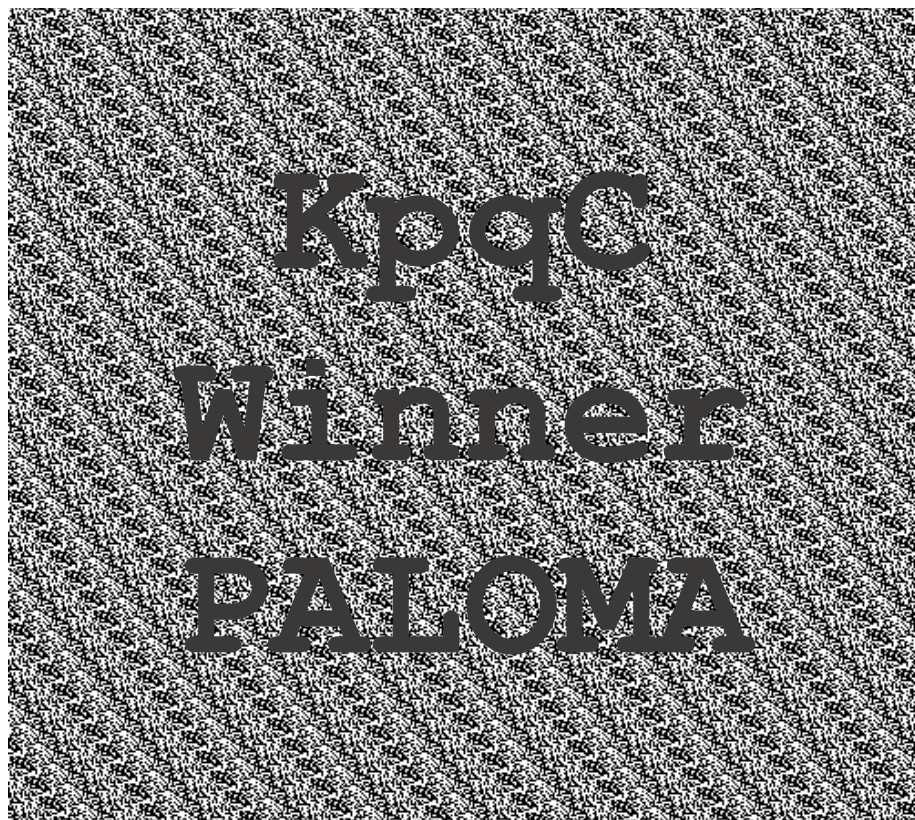
Scrambled Parity-check Matrix \hat{H}



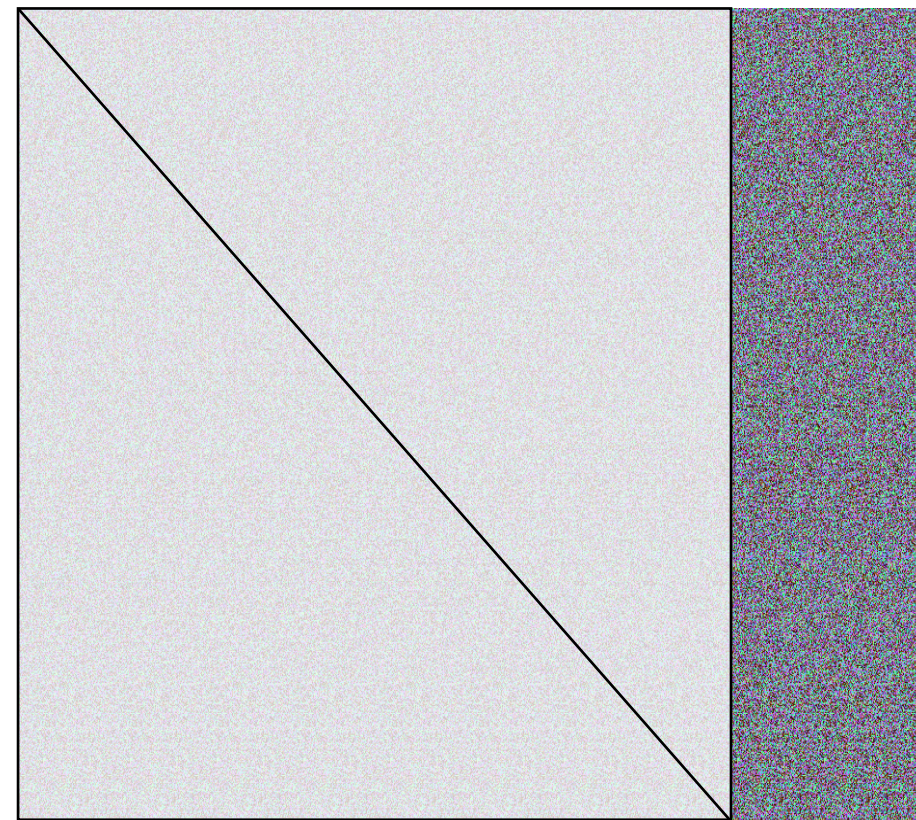
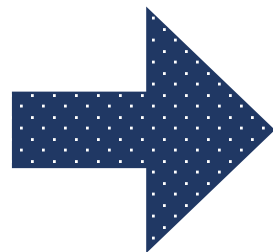
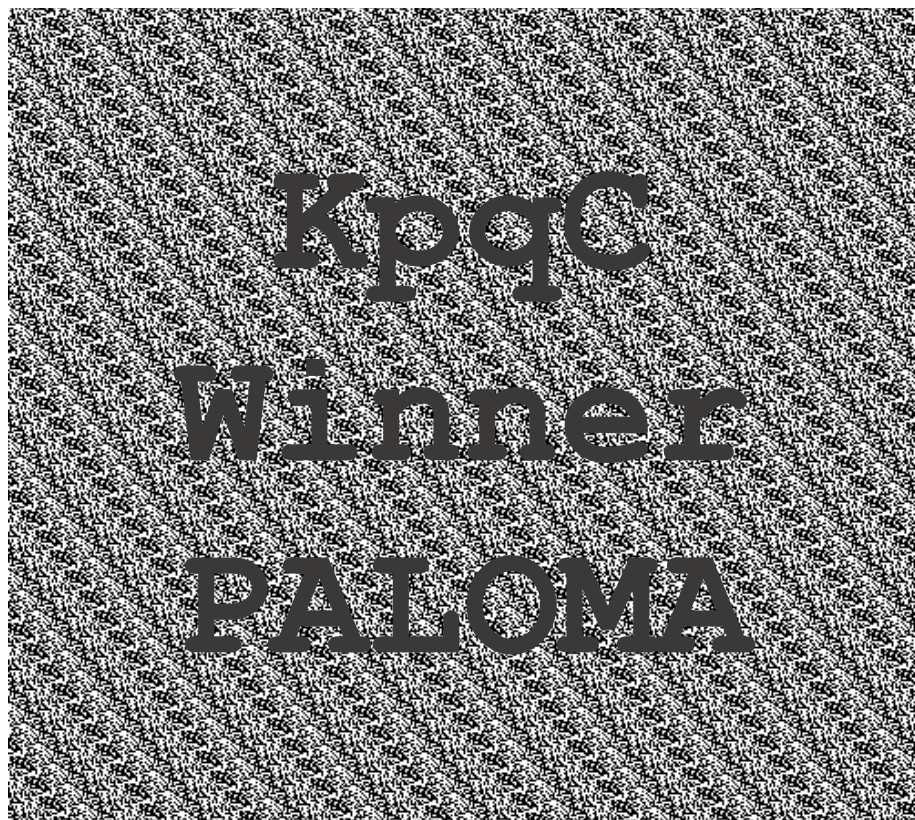
Scrambled Parity-check Matrix \hat{H}



Scrambled Parity-check Matrix \hat{H}



Scrambled Parity-check Matrix \hat{H}



RREF

1

```
[0 1 0 1 1 1 1 0 0 1 1 0 1 1 1 1 0 0 0 1]
[1 1 0 1 1 1 1 0 1 0 1 1 1 1 0 1 0 1 1 0]
[0 1 1 1 0 1 0 0 1 1 1 1 1 0 1 0 0 1 1 1]
[0 0 1 1 1 0 1 1 0 0 0 0 0 0 1 0 0 1 0 0]
[0 0 0 1 0 0 1 0 1 0 0 0 0 0 0 1 0 1 0 0 1]
[1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 1 1 0 1 1]
[1 1 0 0 1 1 1 0 0 1 0 1 1 1 0 0 1 1 1 0]
[0 0 1 1 1 1 1 0 0 1 1 0 1 1 1 0 1 0 1 0]
[0 1 0 1 1 1 1 1 0 1 0 1 0 0 0 1 0 0 1 1]
[1 1 0 1 1 0 0 0 0 1 1 0 1 0 1 1 1 1 1 1]
```

2

[1	1	0	1	1	1	1	0	1	0	1	1	1	1	0	1	0	1	1	0]
[0	1	0	1	1	1	1	0	0	1	1	0	1	1	1	1	0	0	0	1]
[0	1	1	1	0	1	0	0	1	1	1	1	1	0	1	0	0	1	1	1]
[0	0	1	1	1	0	1	1	0	0	0	0	0	1	0	0	1	0	1	0]
[0	0	0	1	0	0	1	0	1	0	0	0	0	0	1	0	1	0	0	1]
[0	0	1	0	0	0	0	1	0	0	0	0	0	1	1	0	1	1	0	1]
[0	0	0	1	0	0	0	0	1	1	1	0	0	0	0	1	1	0	0	0]
[0	0	1	1	1	1	1	0	0	1	1	0	1	1	1	0	1	0	1	0]
[0	1	0	1	1	1	1	1	0	1	0	1	0	0	0	1	0	0	1	1]
[0	0	0	0	0	1	1	0	1	1	0	1	0	1	1	0	1	0	0	1]

3

[1	0	0	0	0	0	0	1	1	0	1	0	0	1	0	0	1	1	1]
[0	1	0	1	1	1	1	0	0	1	1	0	1	1	1	1	0	0	0]
[0	0	1	0	1	0	1	0	1	0	0	1	0	1	0	1	0	1	0]
[0	0	1	1	1	0	1	1	0	0	0	0	0	1	0	0	1	0	0]
[0	0	0	1	0	0	1	0	1	0	0	0	0	0	1	0	1	0	0]
[0	0	1	0	0	0	0	1	0	0	0	0	0	1	1	0	1	1	0]
[0	0	0	1	0	0	0	0	1	1	1	0	0	0	0	1	1	0	0]
[0	0	1	1	1	1	1	0	0	1	1	0	1	1	1	0	1	0	0]
[0	0	0	0	0	0	0	1	0	0	1	1	1	1	1	0	0	0	0]
[0	0	0	0	0	1	1	0	1	1	0	1	0	1	1	0	1	0	0]

4

[1	0	0	0	0	0	0	0	1	1	0	1	0	0	1	0	0	1	1	1]
[0	1	0	1	1	1	1	0	0	1	1	0	1	1	1	1	0	0	0	1]
[0	0	1	0	1	0	1	0	1	0	0	1	0	1	0	1	0	1	1	0]
[0	0	0	1	0	0	0	1	1	0	0	1	0	0	0	1	1	1	0	0]
[0	0	0	1	0	0	1	0	1	0	0	0	0	0	1	0	1	0	0	1]
[0	0	0	0	1	0	1	1	1	0	0	1	0	0	1	1	1	0	1	1]
[0	0	0	1	0	0	0	0	1	1	1	0	0	0	0	1	1	0	0	0]
[0	0	0	1	0	1	0	0	1	1	1	1	1	0	1	1	1	1	0	0]
[0	0	0	0	0	0	0	1	0	0	1	1	1	1	1	0	0	0	1	0]
[0	0	0	0	0	1	1	0	1	1	0	1	0	1	1	0	1	0	0	1]

5

[1	0	0	0	0	0	0	1	1	0	1	0	0	1	0	0	1	1	1]
[0	1	0	0	1	1	1	1	1	1	1	1	1	1	0	1	1	0	1]
[0	0	1	0	1	0	1	0	1	0	0	1	0	1	0	1	0	1	0]
[0	0	0	1	0	0	0	1	1	0	0	1	0	0	0	1	1	1	0]
[0	0	0	0	0	0	1	1	0	0	0	1	0	0	1	1	0	1	0]
[0	0	0	0	1	0	1	1	1	0	0	1	0	0	1	1	1	0	1]
[0	0	0	0	0	0	0	1	0	1	1	1	0	0	0	0	0	1	0]
[0	0	0	0	0	1	0	1	0	1	1	0	1	0	1	0	0	0	0]
[0	0	0	0	0	0	0	1	0	0	1	1	1	1	1	0	0	0	1]
[0	0	0	0	0	1	1	0	1	1	0	1	0	1	1	0	1	0	0]

6

[1	0	0	0	0	0	0	0	1	1	0	1	0	0	1	0	0	1	1	1]
[0	1	0	0	0	1	0	0	0	1	1	0	1	1	0	1	0	1	1	0]
[0	0	1	0	0	0	0	1	0	0	0	0	0	1	1	0	1	1	0	1]
[0	0	0	1	0	0	0	1	1	0	0	1	0	0	0	1	1	1	0	0]
[0	0	0	0	1	0	1	1	1	0	0	1	0	0	1	1	1	0	1	1]
[0	0	0	0	0	0	1	1	0	0	0	1	0	0	1	1	0	1	0	1]
[0	0	0	0	0	0	0	1	0	1	1	1	0	0	0	0	0	1	0	0]
[0	0	0	0	0	1	0	1	0	1	1	0	1	0	1	0	0	0	0	0]
[0	0	0	0	0	0	0	1	0	0	1	1	1	1	1	0	0	0	1	0]
[0	0	0	0	0	1	1	0	1	1	0	1	0	1	1	0	1	0	0	1]

7

[1	0	0	0	0	0	0	1	1	0	1	0	0	1	0	0	1	1	1]
[0	1	0	0	0	0	1	0	0	0	0	0	1	1	1	0	1	1	0]
[0	0	1	0	0	0	1	0	0	0	0	0	1	1	0	1	1	0	1]
[0	0	0	1	0	0	1	1	0	0	1	0	0	0	1	1	1	0	0]
[0	0	0	0	1	0	1	1	1	0	0	1	0	0	1	1	1	0	1]
[0	0	0	0	0	1	0	1	1	1	0	1	0	1	0	0	0	0	0]
[0	0	0	0	0	0	1	0	1	1	1	0	0	0	0	0	1	0	0]
[0	0	0	0	0	1	1	0	0	0	1	0	0	1	1	0	1	0	1]
[0	0	0	0	0	0	1	0	0	1	1	1	1	1	0	0	0	1	0]
[0	0	0	0	0	1	1	1	0	1	1	1	1	0	0	1	0	0	1]

8

[1	0	0	0	0	0	0	1	1	0	1	0	0	1	0	0	1	1	1]
[0	1	0	0	0	0	1	0	0	0	0	0	1	1	1	0	1	1	0]
[0	0	1	0	0	0	1	0	0	0	0	0	1	1	0	1	1	0	1]
[0	0	0	1	0	0	1	1	0	0	1	0	0	0	1	1	1	0	0]
[0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	1	1	1	0]
[0	0	0	0	0	1	0	1	0	1	1	0	1	0	1	0	0	0	0]
[0	0	0	0	0	0	1	0	0	0	1	0	0	1	1	0	1	0	1]
[0	0	0	0	0	0	1	0	1	1	1	0	0	0	0	0	1	0	0]
[0	0	0	0	0	0	1	0	0	1	1	1	1	1	1	0	0	0	1]
[0	0	0	0	0	0	0	1	0	1	0	1	1	1	1	1	1	1	0]

9

[1	0	0	0	0	0	0	0	1	1	0	1	0	0	1	0	0	1	1	1]
[0	1	0	0	0	0	0	0	0	1	1	1	0	1	1	1	0	0	1	0]
[0	0	1	0	0	0	0	0	0	1	1	1	0	1	1	0	1	0	0	1]
[0	0	0	1	0	0	0	0	1	1	1	0	0	0	0	1	1	0	0	0]
[0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	1	1	1	0]
[0	0	0	0	0	1	0	0	0	0	0	1	1	0	1	0	0	1	0	0]
[0	0	0	0	0	0	1	0	0	1	1	0	0	0	1	1	0	0	0	1]
[0	0	0	0	0	0	0	1	0	1	1	1	0	0	0	0	0	1	0	0]
[0	0	0	0	0	0	0	0	1	0	0	1	1	1	0	0	1	1	0]	
[0	0	0	0	0	0	0	1	0	1	0	1	1	1	1	1	1	1	0	0]

10

[1	0	0	0	0	0	0	0	0	1	1	1	1	1	0	1	1	0	1	1]
[0	1	0	0	0	0	0	0	0	1	1	1	0	1	1	1	0	0	1	0]
[0	0	1	0	0	0	0	0	0	1	1	1	0	1	1	0	1	0	0	1]
[0	0	0	1	0	0	0	0	0	1	0	0	1	1	1	0	0	1	0	0]
[0	0	0	0	1	0	0	0	0	0	1	0	1	1	1	1	0	0	1	0]
[0	0	0	0	0	1	0	0	0	0	0	1	1	0	1	0	0	1	0	0]
[0	0	0	0	0	0	1	0	0	1	1	0	0	0	1	1	0	0	0	1]
[0	0	0	0	0	0	0	1	0	1	1	1	0	0	0	0	0	1	0	0]
[0	0	0	0	0	0	0	0	1	0	1	0	1	1	1	1	1	1	0	0]
[0	0	0	0	0	0	0	0	0	1	0	0	1	1	1	0	0	1	1	0]

11

[1	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1	1	1	0	1]
[0	1	0	0	0	0	0	0	0	0	1	1	1	0	0	1	0	1	0	0]
[0	0	1	0	0	0	0	0	0	0	1	1	1	0	0	0	1	1	1	1]
[0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0]
[0	0	0	0	1	0	0	0	0	0	1	0	1	1	1	1	0	0	1	0]
[0	0	0	0	0	1	0	0	0	0	0	1	1	0	1	0	0	1	0	0]
[0	0	0	0	0	0	1	0	0	0	1	0	1	1	0	1	0	1	1	1]
[0	0	0	0	0	0	0	1	0	0	1	1	1	1	1	0	0	0	1	0]
[0	0	0	0	0	0	0	0	1	0	1	0	1	1	1	1	1	1	0	0]
[0	0	0	0	0	0	0	0	1	0	0	1	1	1	0	0	1	1	0]

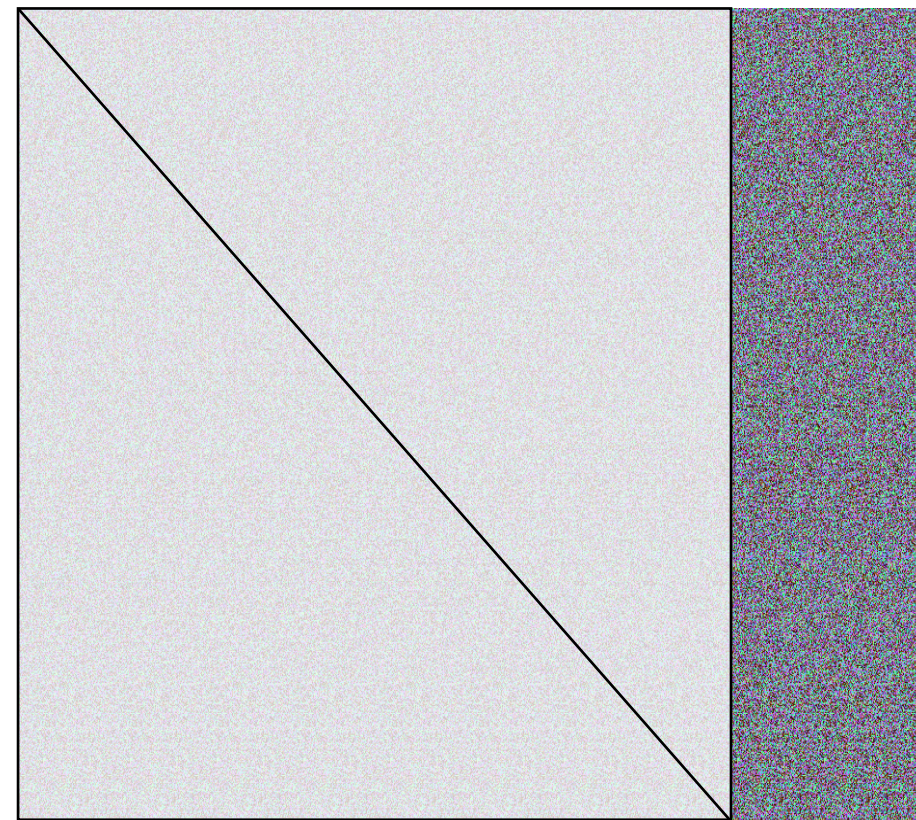
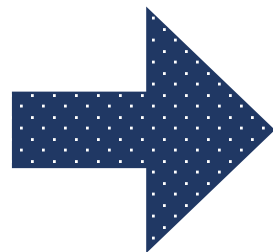
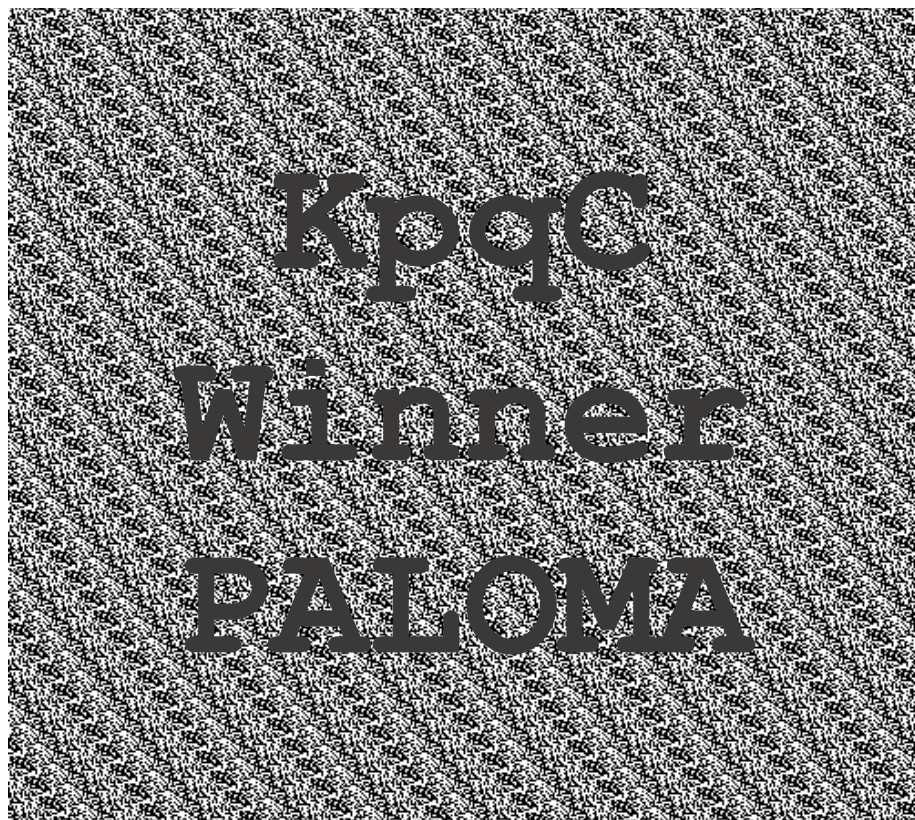
11

[1 0 0 0 0 0 0 0 0 0	1 1 0 0 1 1 1 1 0 1]
[0 1 0 0 0 0 0 0 0 0	1 1 1 0 0 1 0 1 0 0]
[0 0 1 0 0 0 0 0 0 0	1 1 1 0 0 0 1 1 1 1]
[0 0 0 1 0 0 0 0 0 0	0 0 0 0 0 0 0 0 1 0]
[0 0 0 0 1 0 0 0 0 0	1 0 1 1 1 1 0 0 1 0]
[0 0 0 0 0 1 0 0 0 0	0 1 1 0 1 0 0 1 0 0]
[0 0 0 0 0 0 1 0 0 0	1 0 1 1 0 1 0 1 1 1]
[0 0 0 0 0 0 0 1 0 0	1 1 1 1 1 0 0 0 1 0]
[0 0 0 0 0 0 0 0 1 0	1 0 1 1 1 1 1 1 0 0]
[0 0 0 0 0 0 0 0 0 1	0 0 1 1 1 0 0 1 1 0]

11

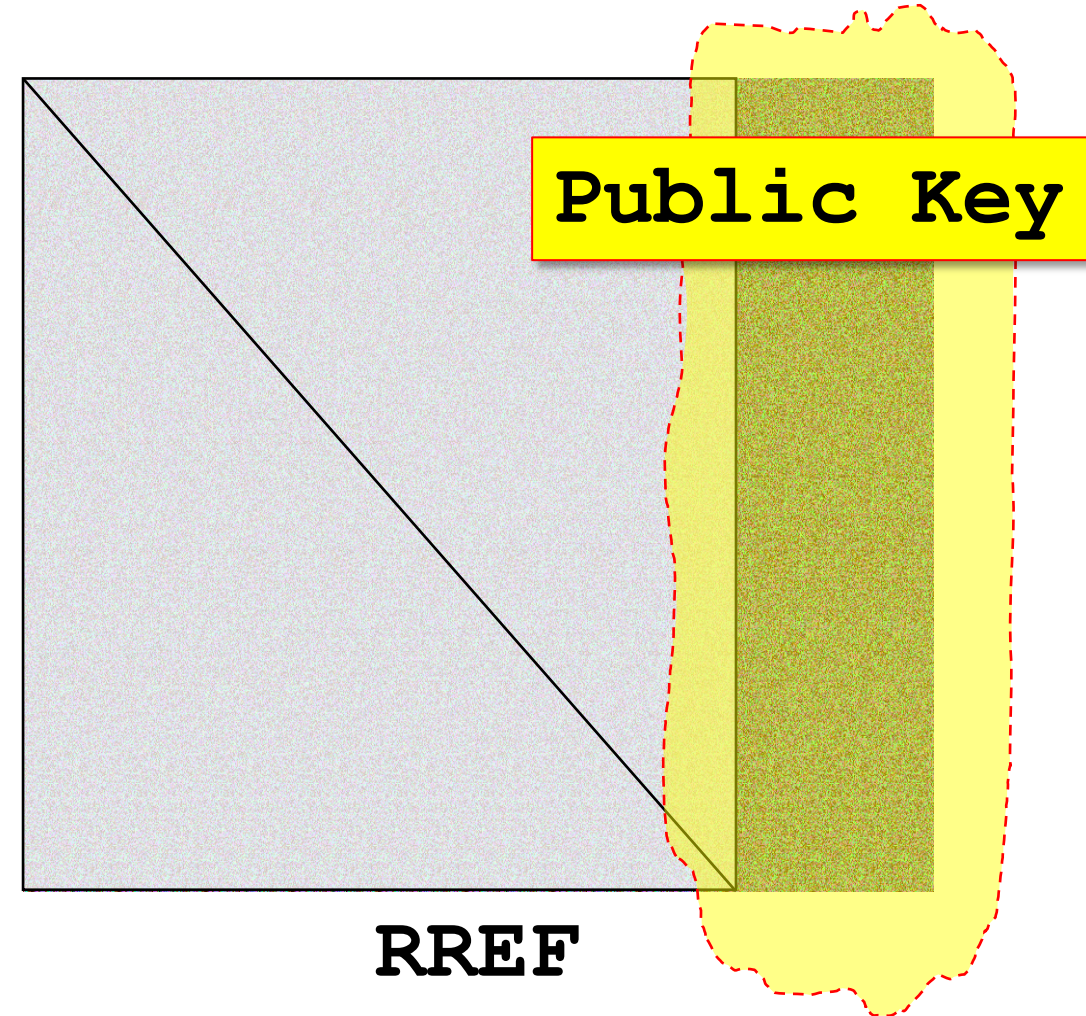
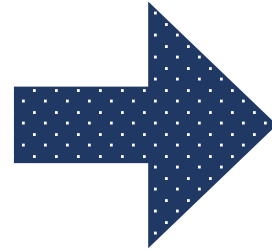
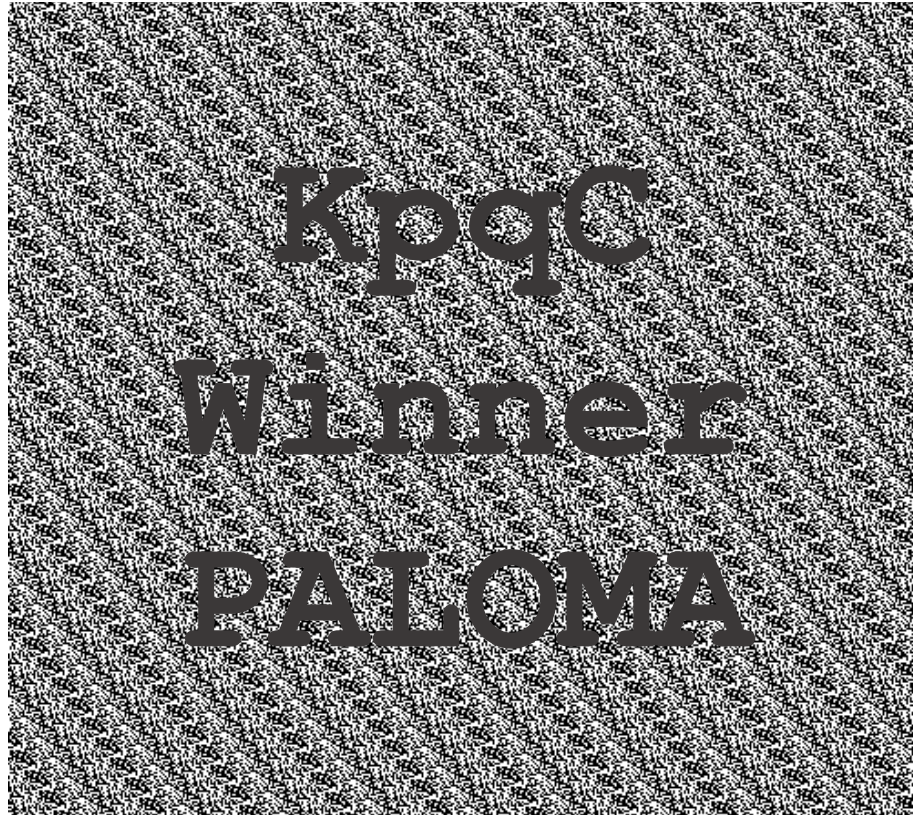
1	1	0	0	1	1	1	1	0	1]
1	1	1	0	0	1	0	1	0	0]
1	1	1	0	0	0	1	1	1	1]
0	0	0	0	0	0	0	0	1	0]
1	0	1	1	1	1	0	0	1	0]
0	1	1	0	1	0	0	1	0	0]
1	0	1	1	0	1	0	1	1	1]
1	1	1	1	1	0	0	0	1	0]
1	0	1	1	1	1	1	1	0	0]
0	0	1	1	1	0	0	1	1	0]

Scrambled Parity-check Matrix \hat{H}

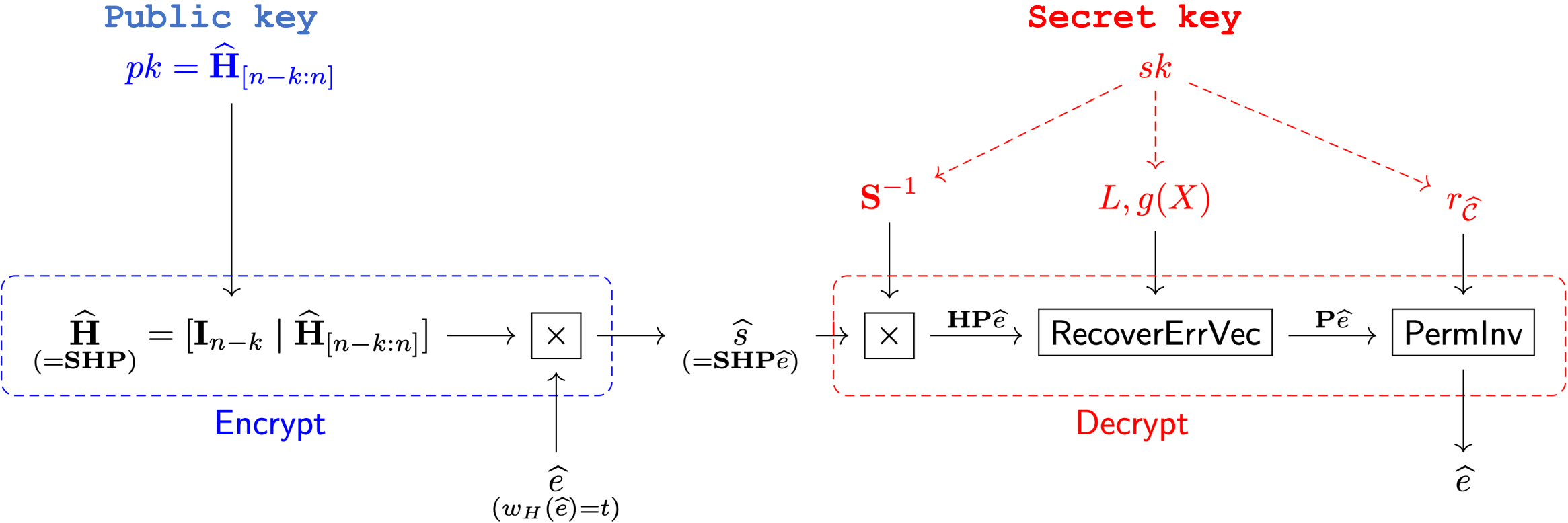


RREF

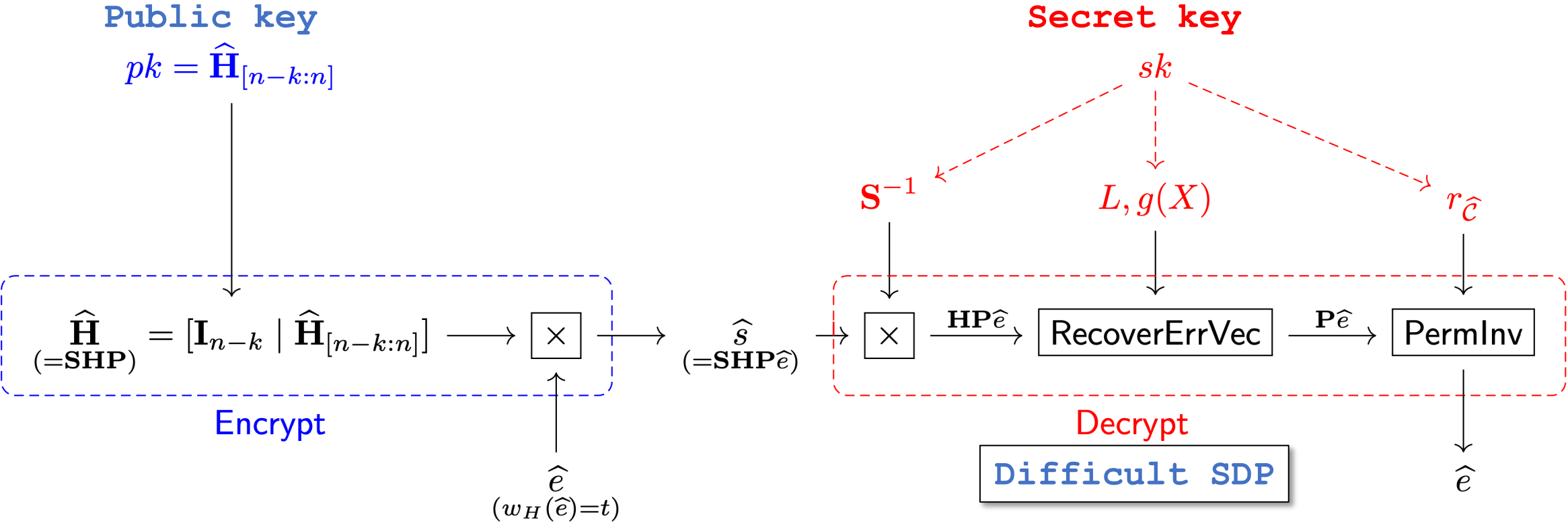
Scrambled Parity-check Matrix \hat{H}



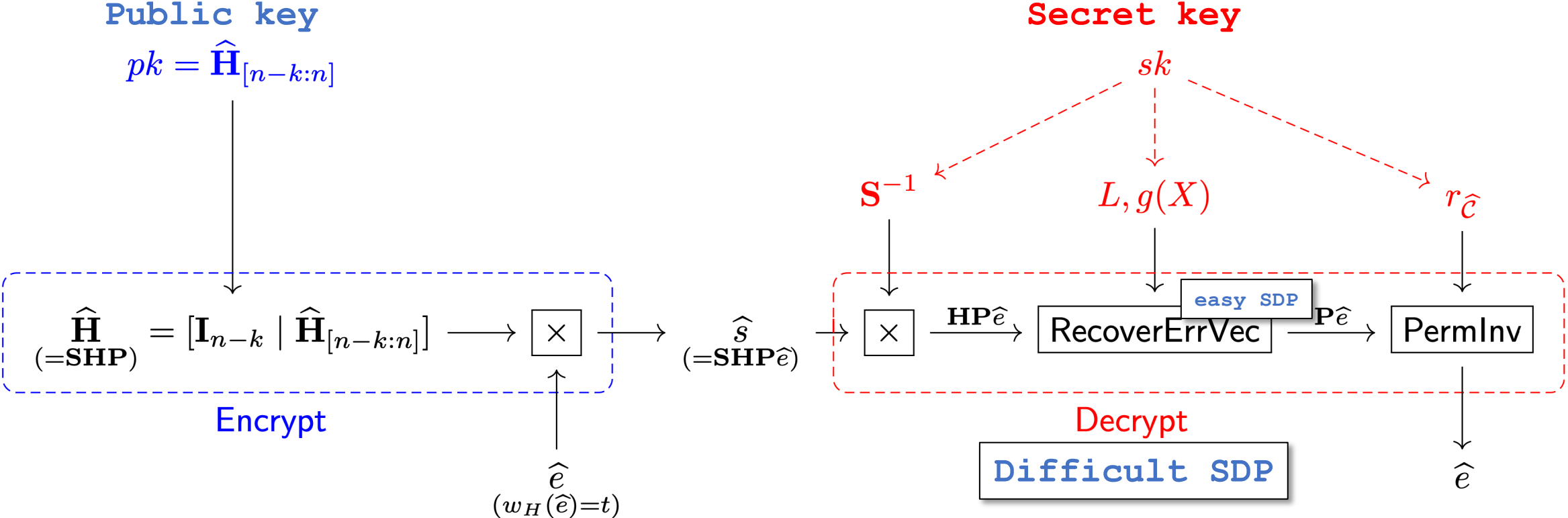
PALOMA trapdoor - Encrypt/Decrypt



PALOMA trapdoor - Encrypt/Decrypt



PALOMA trapdoor - Encrypt/Decrypt



A close-up photograph of golden rice stalks, likely during harvest. The rice grains are ripe and yellow, with some green leaves still visible. The background is a soft, out-of-focus field of similar rice plants under a bright sky. A semi-transparent yellow rectangular box is positioned in the lower-left area of the image, containing the text "Good Trapdoor" in a bold, black, sans-serif font.

Good Trapdoor



Good Trapdoor

2024.02.28. @ KpqC Winter Camp



**Ready to build
Good KEM**

218 / 266

Fujisaki-Okamoto Transformation

- Eiichiro Fujisaki & Tatsuaki Okamoto, "Secure Integration of Asymmetric and Symmetric Encryption Schemes", CRYPTO 1999

$$\mathcal{E}_{pk}^{\text{hy}}(m; \sigma) = \mathcal{E}_{pk}^{\text{asy}}(\sigma; H(\sigma, c)) \parallel \mathcal{E}_{G(\sigma)}^{\text{sy}}(m)$$

Let Π^{asy} be a γ -spread $(t^{\text{asy}}, \epsilon^{\text{asy}})$ -OWE secure asymmetric encryption scheme. Let Π^{sy} be a $(t^{\text{sy}}, \epsilon^{\text{sy}})$ -OT secure symmetric encryption scheme. $T^{\text{asy}}(k)$ denotes the worst case of the running time of $\mathcal{E}_{pk}^{\text{asy}}$ for every pk generated by $\mathcal{K}^{\text{asy}}(1^k)$. Let Π^{hy} be the hybrid encryption scheme obtained by our conversion. We then have the following theorem.

Theorem 6.1. Π^{hy} is $(t^{\text{hy}}, q_{\text{hash}}, q_{\text{dec}}, \epsilon^{\text{hy}})$ -IND-CCA secure in the random oracle model where

$$t^{\text{hy}} = \min(t^{\text{asy}}, t^{\text{sy}}) - (q_{\text{dec}} + 1)T^{\text{asy}}(k) - q_{\text{hash}}O(k) \quad \text{and}$$

$$\epsilon^{\text{hy}} = 2q_{\text{hash}}\epsilon^{\text{asy}} + \epsilon^{\text{sy}} + 2q_{\text{dec}}2^{-\gamma}.$$

Fujisaki-Okamoto Transformation

- Eiichiro Fujisaki & Tatsuaki Okamoto, "Secure Integration of Asymmetric and Symmetric Encryption Schemes", CRYPTO 1999

$$\mathcal{E}_{pk}^{\text{hy}}(m; \sigma) = \mathcal{E}_{pk}^{\text{asy}}(\sigma; H(\sigma, c)) \parallel \mathcal{E}_{G(\sigma)}^{\text{sy}}(m)$$

**IND-CCA2-secure Encryption
in ROM**

Let Π^{asy} be a γ -spread $(t^{\text{asy}}, \epsilon^{\text{asy}})$ -OWE secure asymmetric encryption scheme. Let Π^{sy} be a $(t^{\text{sy}}, \epsilon^{\text{sy}})$ -OT secure symmetric encryption scheme. $T^{\text{asy}}(k)$ denotes the worst case of the running time of $\mathcal{E}_{pk}^{\text{asy}}$ for every pk generated by $\mathcal{K}^{\text{asy}}(1^k)$. Let \mathcal{H} be the hybrid encryption scheme obtained by our construction. We then have the following theorem.

Theorem 6.1 \mathcal{H} is $(t^{\text{hy}}, q_{\text{hash}}, q_{\text{dec}}, \epsilon^{\text{hy}})$ -IND-CCA secure in the random oracle model where

$$t^{\text{hy}} = \min(t^{\text{asy}}, t^{\text{sy}}) - (q_{\text{dec}} + 1)T^{\text{asy}}(k) - q_{\text{hash}}O(k) \quad \text{and}$$

$$\epsilon^{\text{hy}} = 2q_{\text{hash}}\epsilon^{\text{asy}} + \epsilon^{\text{sy}} + 2q_{\text{dec}}2^{-\gamma}.$$



<https://www.moshir>



<https://www.moshimoshi-nippon.jp/wp/wp-content/uploads/>



<https://www.moshimoshi-nippon.jp/wp/wp-content/uploads/2016/12/piko02.png>

Fujisaki-Okamoto Transformation

- Eiichiro Fujisaki & Tatsuaki Okamoto, "Secure Integration of Asymmetric and Symmetric Encryption Schemes", CRYPTO 1999

$$\mathcal{E}_{pk}^{\text{hy}}(m; \sigma) = \mathcal{E}_{pk}^{\text{asy}}(\sigma; H(\sigma, c)) \parallel \mathcal{E}_{G(\sigma)}^{\text{sy}}(m)$$

Random Oracle



Random Function

$$t^{\text{hy}} = \min(t^{\text{asy}}, t^{\text{sy}}) - (q_{\text{dec}} + 1)T^{\text{asy}}(k) - q_{\text{hash}}O(k) \quad \text{and}$$

$$\epsilon^{\text{hy}} = 2q_{\text{hash}}\epsilon^{\text{asy}} + \epsilon^{\text{sy}} + 2q_{\text{dec}}2^{-\gamma}.$$

Fujisaki-Okamoto Transformation

- Eiichiro Fujisaki & Tatsuaki Okamoto, "Secure Integration of Asymmetric and Symmetric Encryption Schemes", CRYPTO 1999

$$\mathcal{E}_{pk}^{\text{hy}}(m; \sigma) = \mathcal{E}_{pk}^{\text{asy}}(\sigma; H(\sigma, c)) \parallel \mathcal{E}_{G(\sigma)}^{\text{sy}}(m)$$

Let Π^{asy} be a γ -spread $(t^{\text{asy}}, \epsilon^{\text{asy}})$ -OWE secure asymmetric encryption scheme. Let Π^{sy} be a $(t^{\text{sy}}, \epsilon^{\text{sy}})$ -OT secure symmetric encryption scheme. $T^{\text{asy}}(k)$ denotes the worst case of the running time of $\mathcal{E}_{pk}^{\text{asy}}$ for every pk generated by $\mathcal{K}^{\text{asy}}(1^k)$. Let Π^{hy} be the hybrid encryption scheme obtained by our conversion. We then have the following theorem.

Theorem 6.1. Π^{hy} is $(t^{\text{hy}}, q_{\text{hash}}, q_{\text{dec}}, \epsilon^{\text{hy}})$ -IND-CCA secure in the random oracle model where

$$t^{\text{hy}} = \min(t^{\text{asy}}, t^{\text{sy}}) - (q_{\text{dec}} + 1)T^{\text{asy}}(k) - q_{\text{hash}}O(k) \quad \text{and}$$

$$\epsilon^{\text{hy}} = 2q_{\text{hash}}\epsilon^{\text{asy}} + \epsilon^{\text{sy}} + 2q_{\text{dec}}2^{-\gamma}.$$

Fujisaki-Okamoto Transformation

- Eiichiro Fujisaki & Tatsuaki Okamoto, "Secure Integration of Asymmetric and Symmetric Encryption Schemes", CRYPTO 1999

$$\mathcal{E}_{pk}^{\text{hy}}(m; \sigma) = \mathcal{E}_{pk}^{\text{asy}}(\sigma; H(\sigma, c)) \parallel G(\sigma)$$

Let Π^{asy} be a γ -spread $(t^{\text{asy}}, \epsilon^{\text{asy}})$ -OWE secure asymmetric encryption scheme. Let Π^{sy} be a $(t^{\text{sy}}, \epsilon^{\text{sy}})$ -OT secure symmetric encryption scheme. $T^{\text{asy}}(k)$ denotes the worst case of the running time of $\mathcal{E}_{pk}^{\text{asy}}$ for every pk generated by $\mathcal{K}^{\text{asy}}(1^k)$. Let Π^{hy} be the hybrid encryption scheme obtained by our conversion. We then have the following theorem.

Theorem 6.1. Π^{hy} is $(t^{\text{hy}}, q_{\text{hash}}, q_{\text{dec}}, \epsilon^{\text{hy}})$ -IND-CCA secure in the random oracle model where

$$t^{\text{hy}} = \min(t^{\text{asy}}, t^{\text{sy}}) - (q_{\text{dec}} + 1)T^{\text{asy}}(k) - q_{\text{hash}}O(k) \quad \text{and}$$

$$\epsilon^{\text{hy}} = 2q_{\text{hash}}\epsilon^{\text{asy}} + \epsilon^{\text{sy}} + 2q_{\text{dec}}2^{-\gamma}.$$

Fujisaki-Okamoto Transformation

- Eiichiro Fujisaki & Tatsuaki Okamoto, "Secure Integration of Asymmetric and Symmetric Encryption Schemes", CRYPTO 1999

$$\mathcal{E}_{pk}^{\text{hy}}(m; \sigma) = \mathcal{E}_{pk}^{\text{asy}}(\sigma; H(\sigma, c)) \parallel G(\sigma)$$

**IND-CCA2-secure KEM
in ROM**

Let Π^{asy} be a γ -spread $(t^{\text{asy}}, \epsilon^{\text{asy}})$ -OWE secure asymmetric encryption scheme. Let Π^{sy} be a $(t^{\text{sy}}, \epsilon^{\text{sy}})$ -OT secure symmetric encryption scheme. $T^{\text{asy}}(k)$ denotes the worst case running time of $\mathcal{E}_{pk}^{\text{asy}}$ for every $k \in \mathcal{K}^{\text{asy}}(1^k)$. Let Π^{hy} be the hybrid encryption scheme obtained by our conversion. We then have the following theorem.

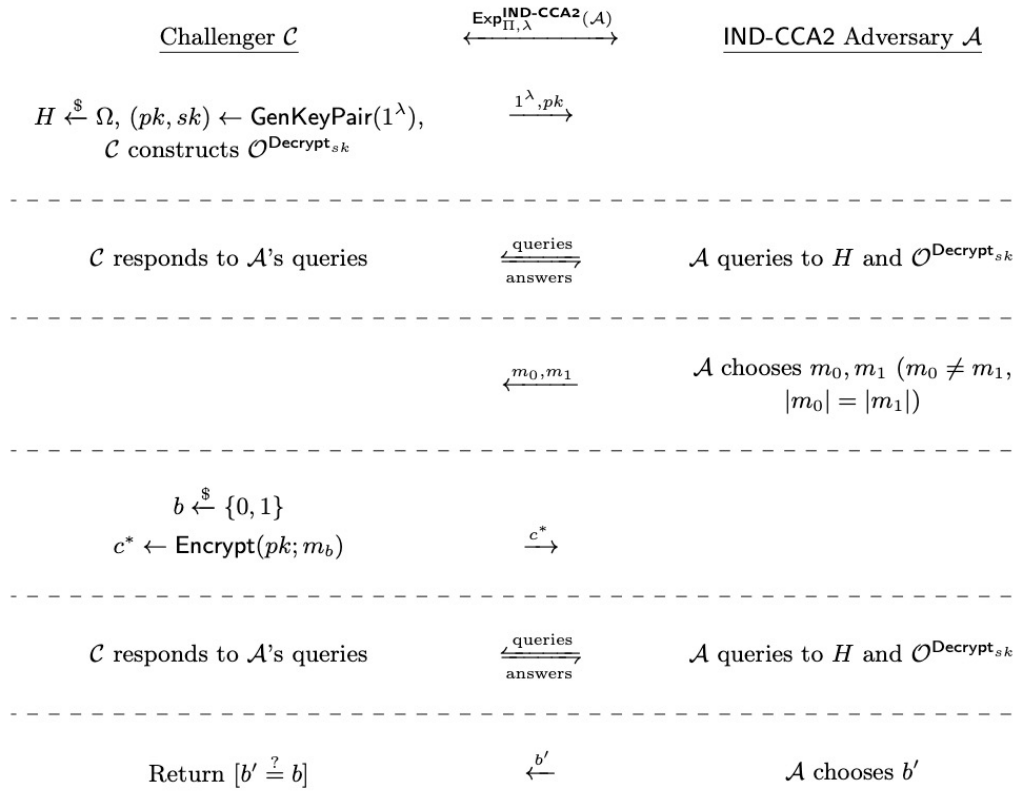
Theorem 6.1 Π^{hy} is $(t^{\text{hy}}, q_{\text{hash}}, q_{\text{dec}}, \epsilon^{\text{hy}})$ -IND-CCA secure in the random oracle model where

$$t^{\text{hy}} = \min(t^{\text{asy}}, t^{\text{sy}}) - (q_{\text{dec}} + 1)T^{\text{asy}}(k) - q_{\text{hash}}O(k) \quad \text{and}$$

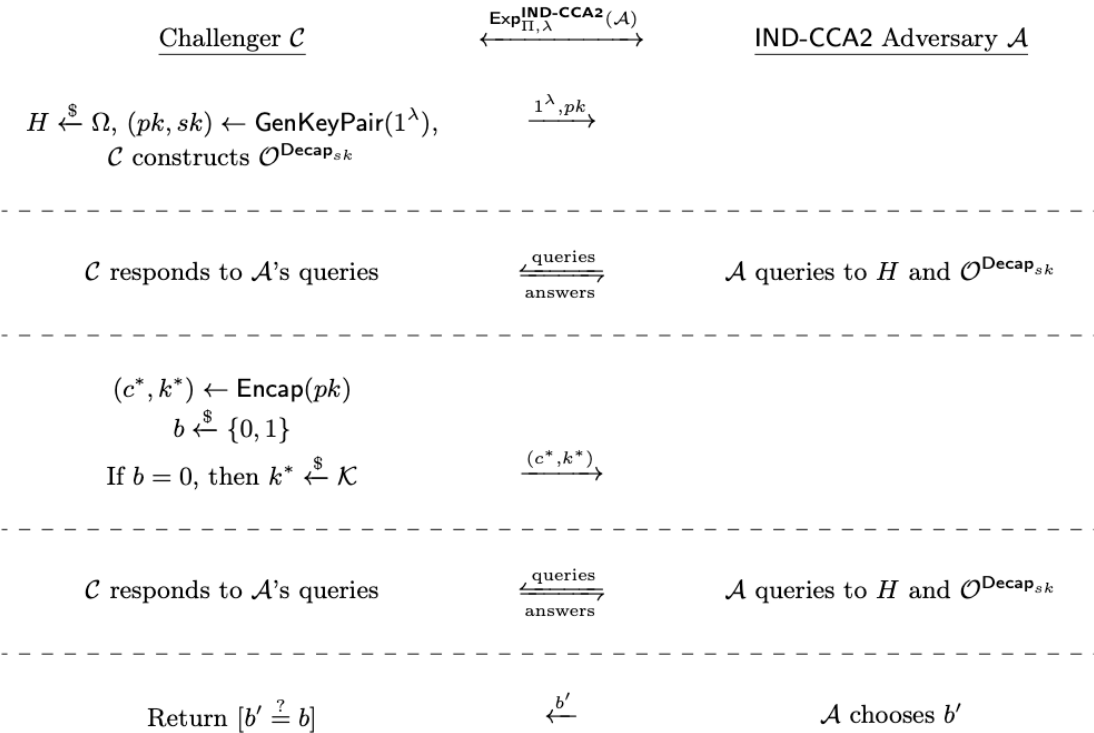
$$\epsilon^{\text{hy}} = 2q_{\text{hash}}\epsilon^{\text{asy}} + \epsilon^{\text{sy}} + 2q_{\text{dec}}2^{-\gamma}.$$

IND-CCA2 Security Exp. in ROM

IND-CCA2 Security Experiment for Encryption: $\text{Exp}_{\Pi, \lambda}^{\text{IND-CCA2}}(\mathcal{A})$



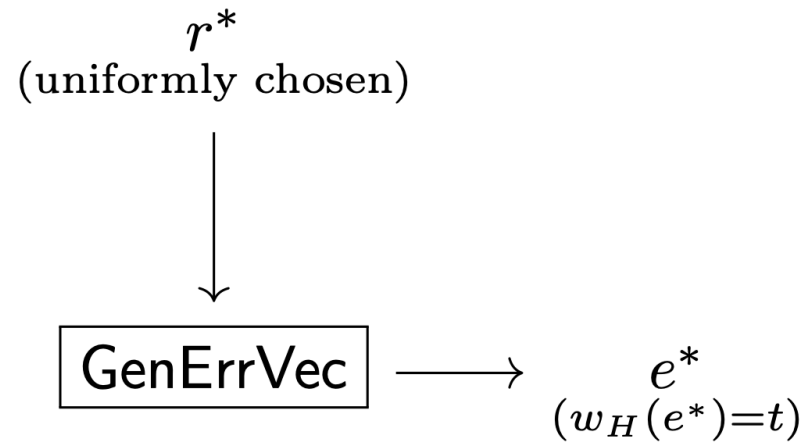
IND-CCA2 Security Experiment for KEM: $\text{Exp}_{\Pi, \lambda}^{\text{IND-CCA2}}(\mathcal{A})$



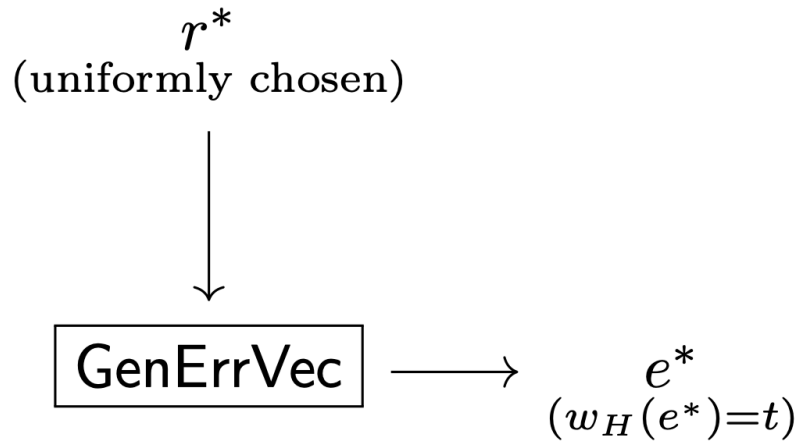
KEM structure – Encapsulation

r^*
(uniformly chosen)

KEM structure – Encapsulation



KEM structure – Encapsulation



Algorithm 9 GenErrVec: Generating a t -Hamming weight Error Vector with a 256-bit Seed

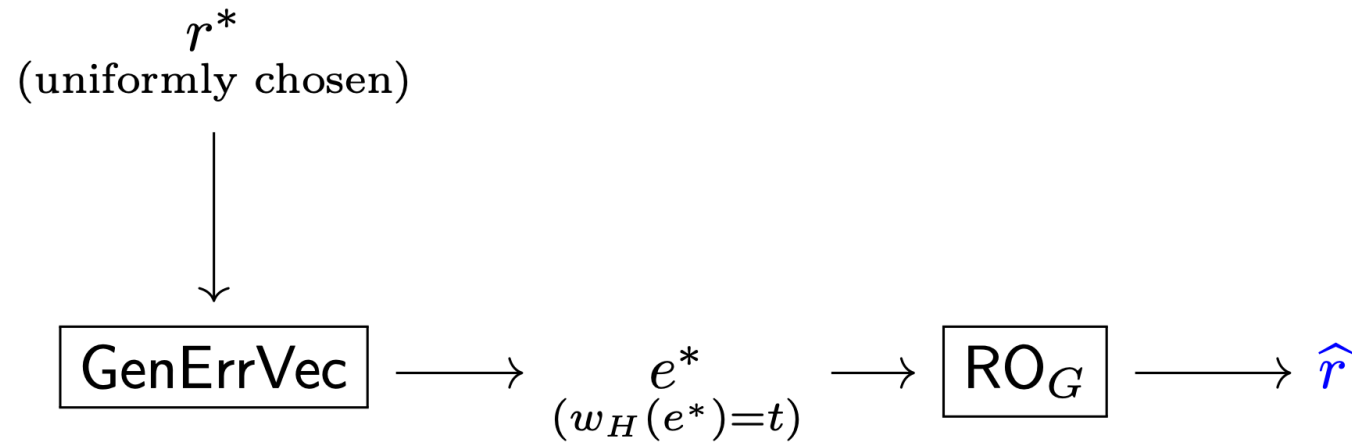
Input: A vector length n , a Hamming weight t , and a 256-bit seed $r \in \{0, 1\}^{256}$

Output: An error vector $e = (e_0, e_1, \dots, e_{n-1}) \in \mathbb{F}_2^n$ with $w_H(e) = t$

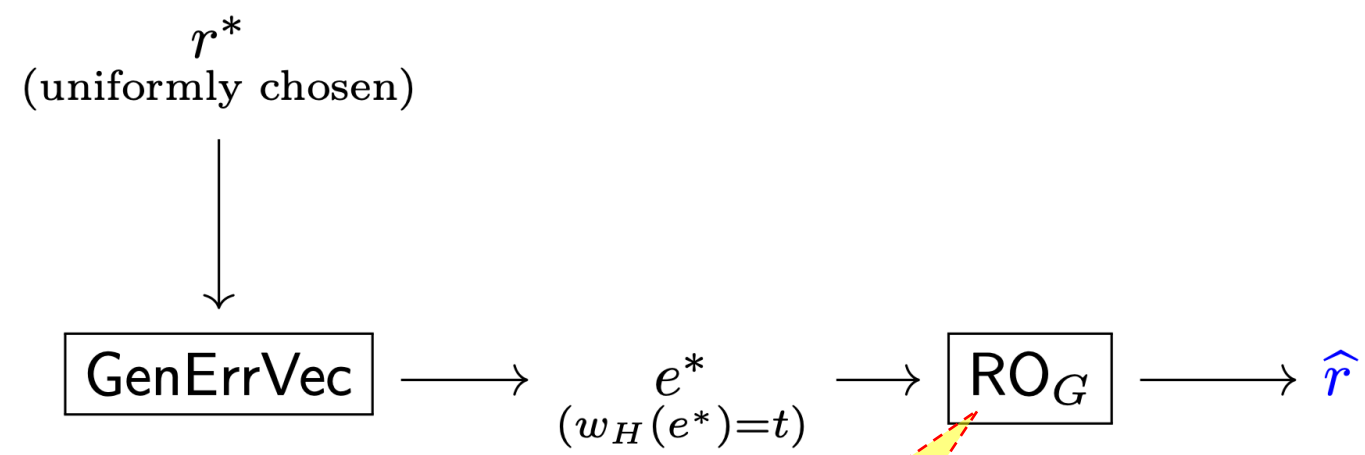
```
1: procedure GenErrVec( $n, t, r$ )  
2:    $[l_0, l_1, \dots, l_{n-1}] \leftarrow \text{Shuffle}([0, 1, \dots, n-1], r)$   
3:    $e = (e_0, e_1, \dots, e_{n-1}) \leftarrow (0, 0, \dots, 0)$   
4:   for  $j = 0$  to  $t - 1$  do  
5:      $e_{l_j} \leftarrow 1$   
6:   end for  
7:   return  $e$   
8: end procedure
```

▷ $\text{supp}(e) = \{l_0, l_1, \dots, l_{t-1}\}$

KEM structure – Encapsulation



KEM structure – Encapsulation



LSH512

Algorithm 10 RO_G, RO_H : Random Oracles

Input: A bit string $x \in \{0, 1\}^*$

Output: A 256-bit string $r \in \{0, 1\}^{256}$

1: **procedure** $RO_G(x)$

2: **return** $LSH("PALOMAGG" || x)_{[0:256]}$

3: **end procedure**

1: **procedure** $RO_H(x)$

2: **return** $LSH("PALOMAHH" || x)_{[0:256]}$

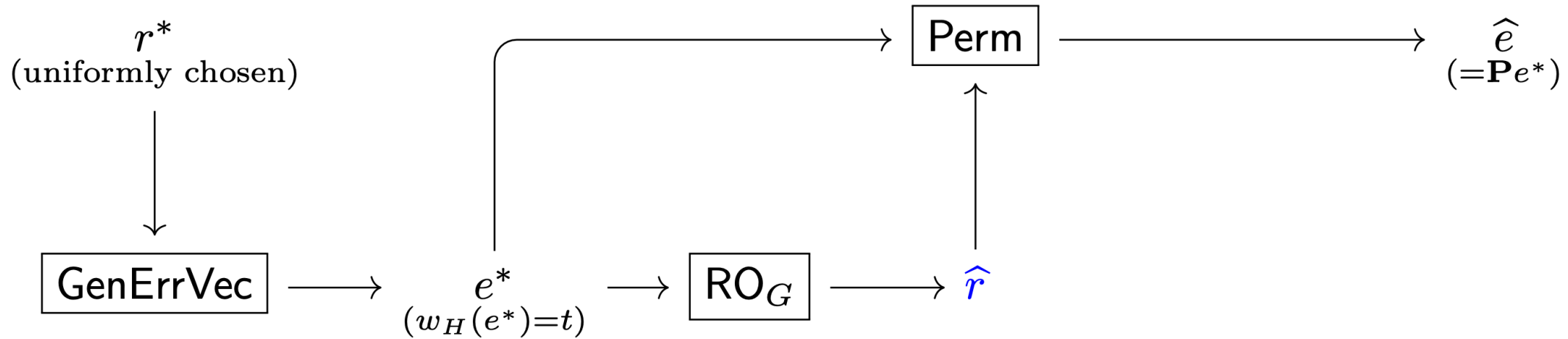
3: **end procedure**

2024.02.28. @ KpqC Winter Camp

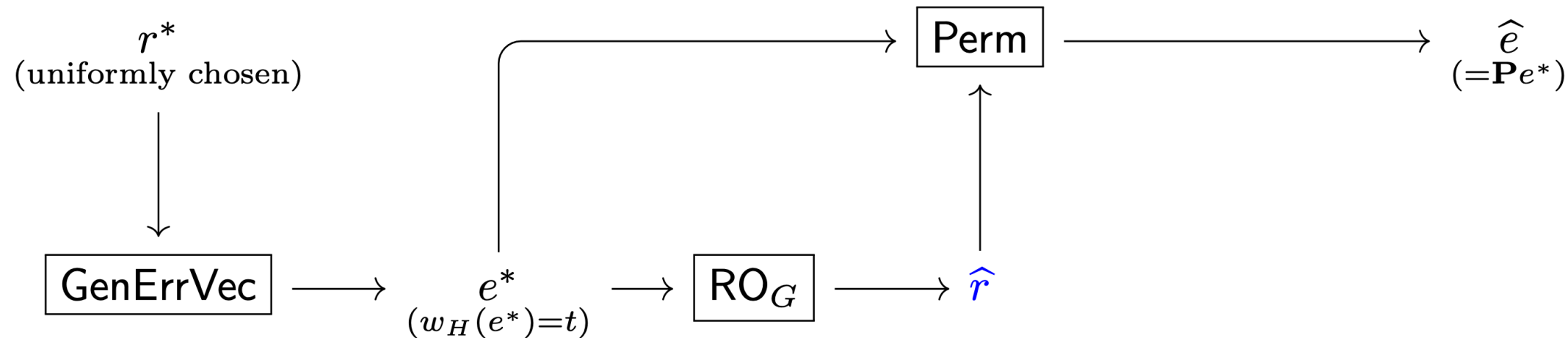
PALOMA: Binary Separable Goppa-based KEM

233 / 266

KEM structure – Encapsulation

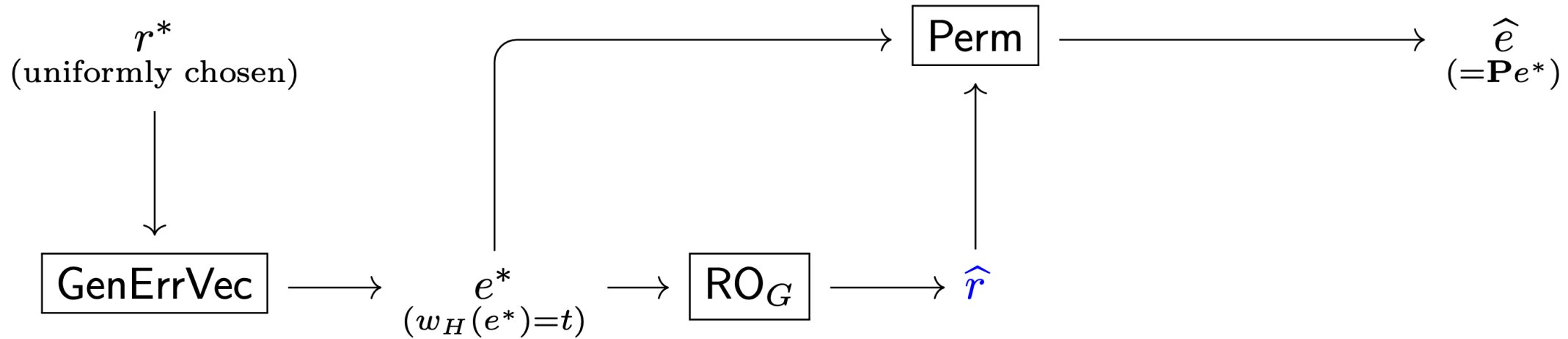


KEM structure – Encapsulation

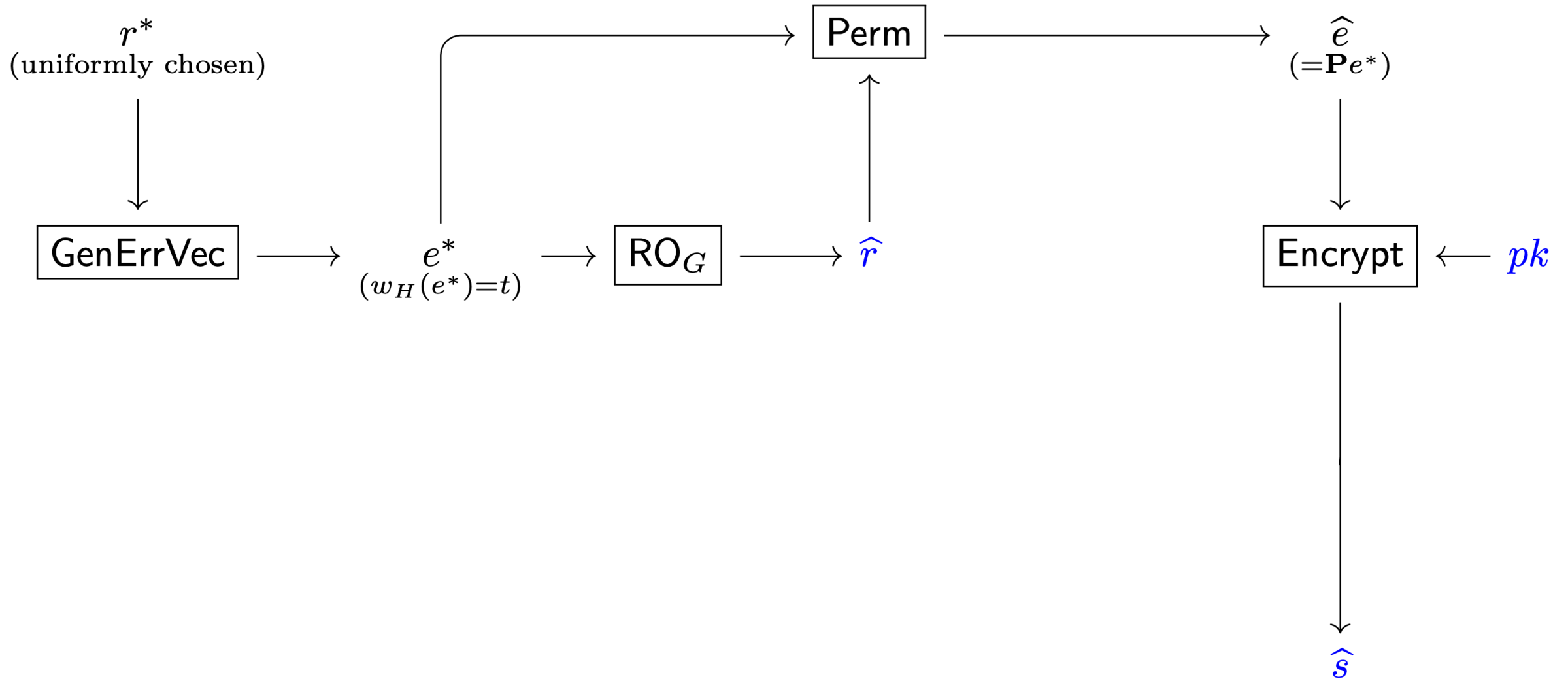


Algorithm 8 Perm and Permlnv: Vector Permutation	Algorithm 7 GenPermMat: Generating a n -bit Permutation with a 256-bit Seed
Input: A vector $v \in \mathbb{F}_2^n$ and $r \in \{0, 1\}^{256}$ Output: A vector $v \in \mathbb{F}_2^n$	Input: A permutation size n and a 256-bit seed $r \in \{0, 1\}^{256}$ Output: An $n \times n$ permutation matrix $\mathbf{P}, \mathbf{P}^{-1} \in \mathbb{F}_2^{n \times n}$
<pre> 1: procedure Perm(v, r) 2: $\mathbf{P}, \mathbf{P}^{-1} \leftarrow \text{GenPermMat}(n, r)$ 3: $v \leftarrow \mathbf{P}v$ 4: return v 5: end procedure </pre>	<pre> 1: procedure GenPermMat(n, r) 2: $[l_0, l_1, \dots, l_{n-1}] \leftarrow \text{Shuffle}([0, 1, \dots, n-1], r)$ 3: $\mathbf{P} \leftarrow [u_{l_0} \mid u_{l_1} \mid \dots \mid u_{l_{n-1}}] \in \mathbb{F}_2^{n \times n}$ 4: return $\mathbf{P}, \mathbf{P}^{-1}$ 5: end procedure </pre>

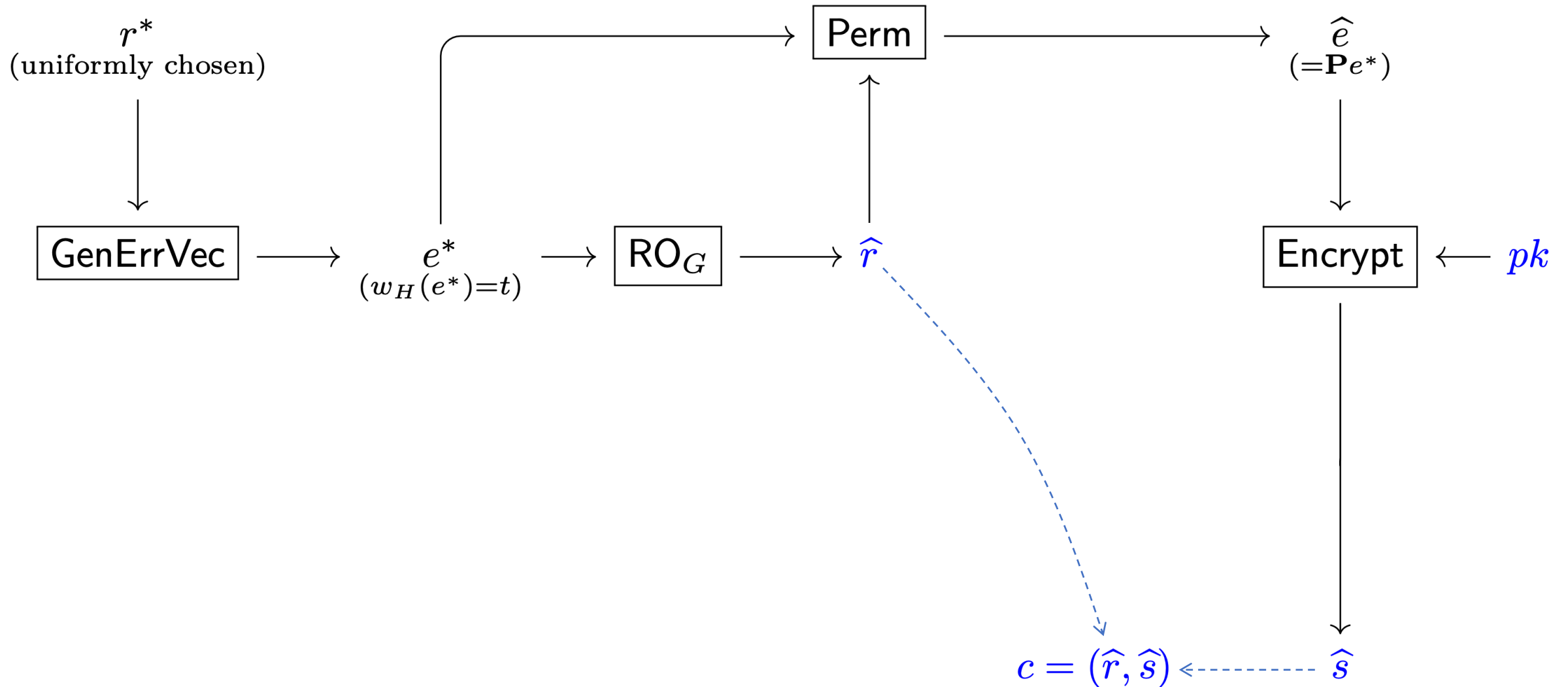
KEM structure – Encapsulation



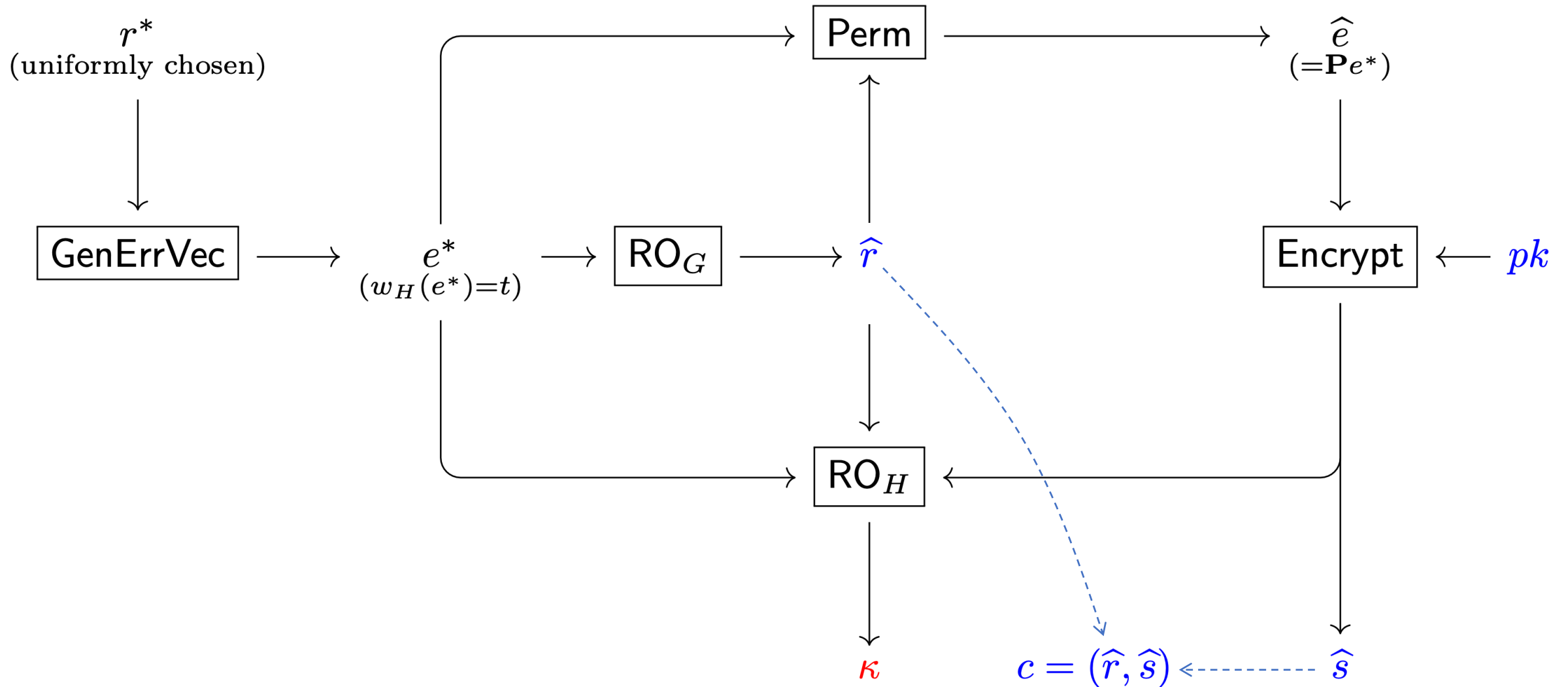
KEM structure – Encapsulation



KEM structure – Encapsulation



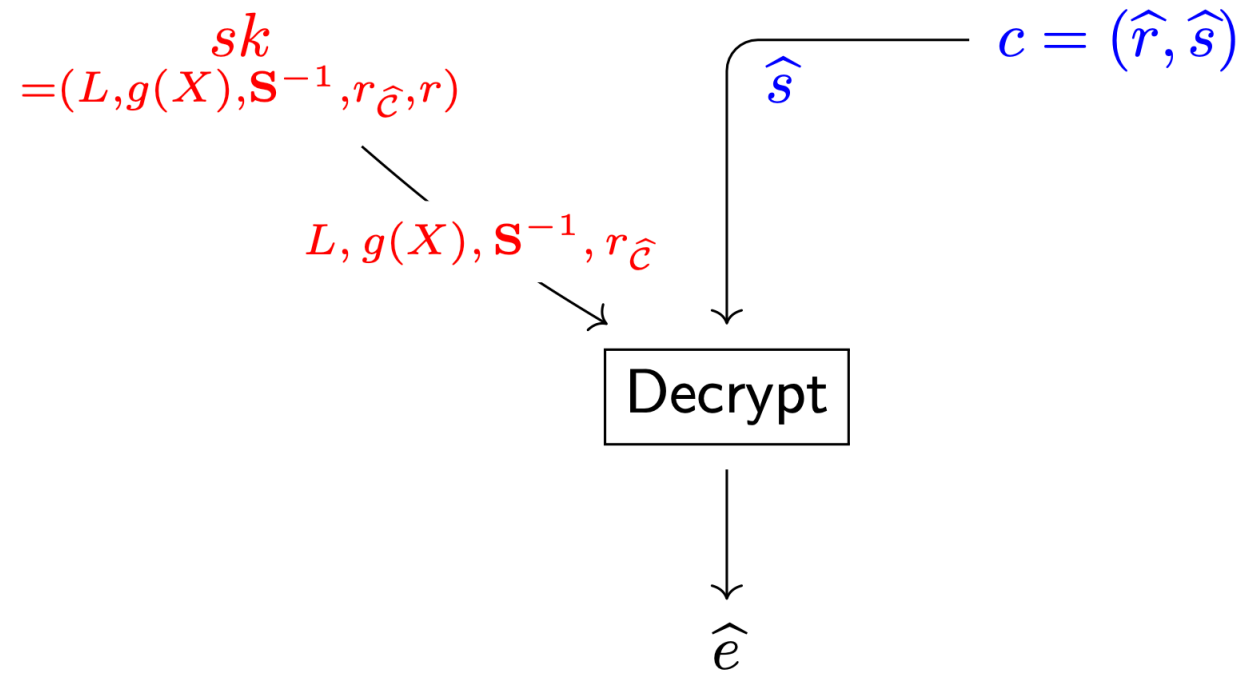
KEM structure – Encapsulation



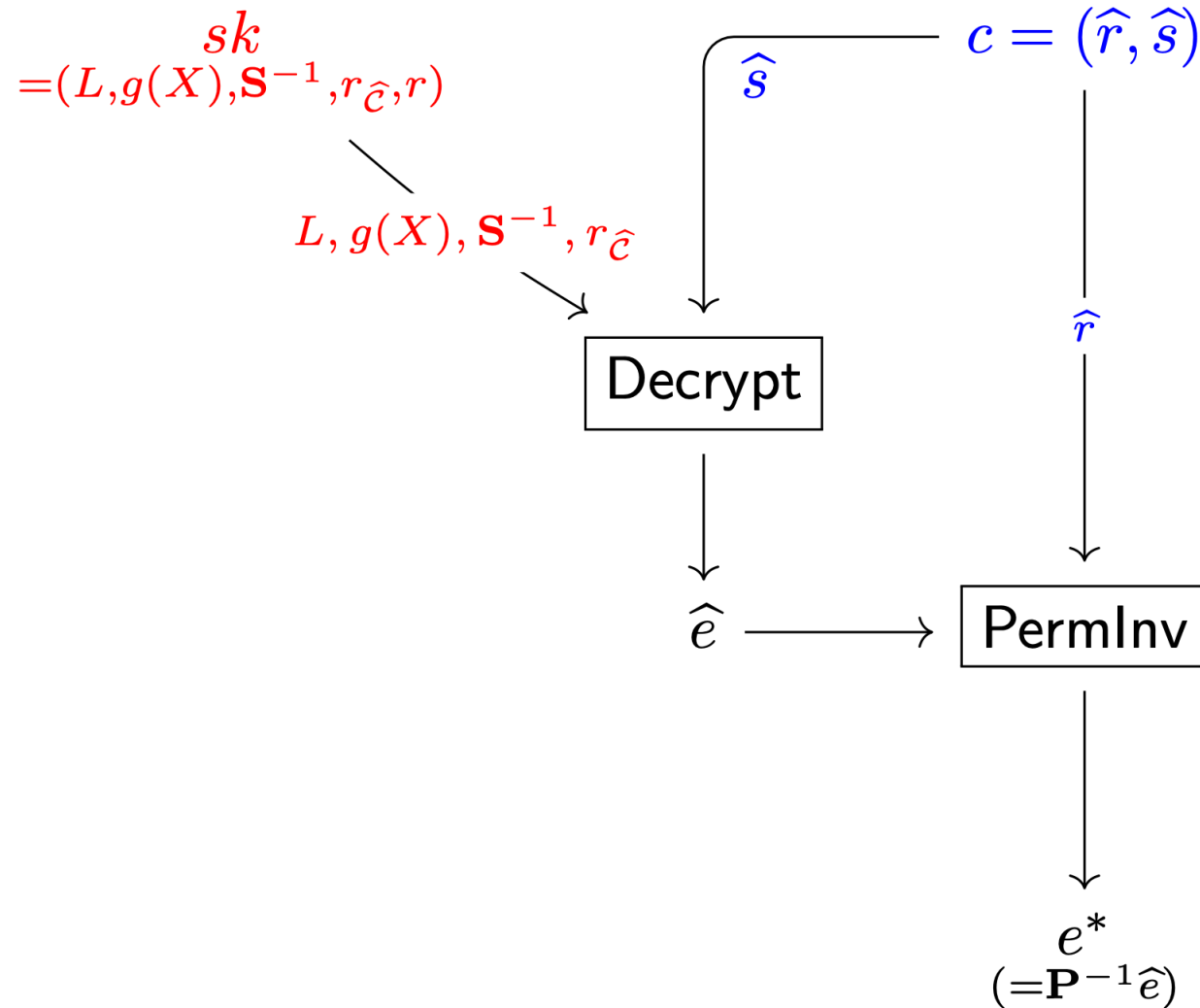
KEM structure – Decapsulation

$$c = (\hat{r}, \hat{s})$$

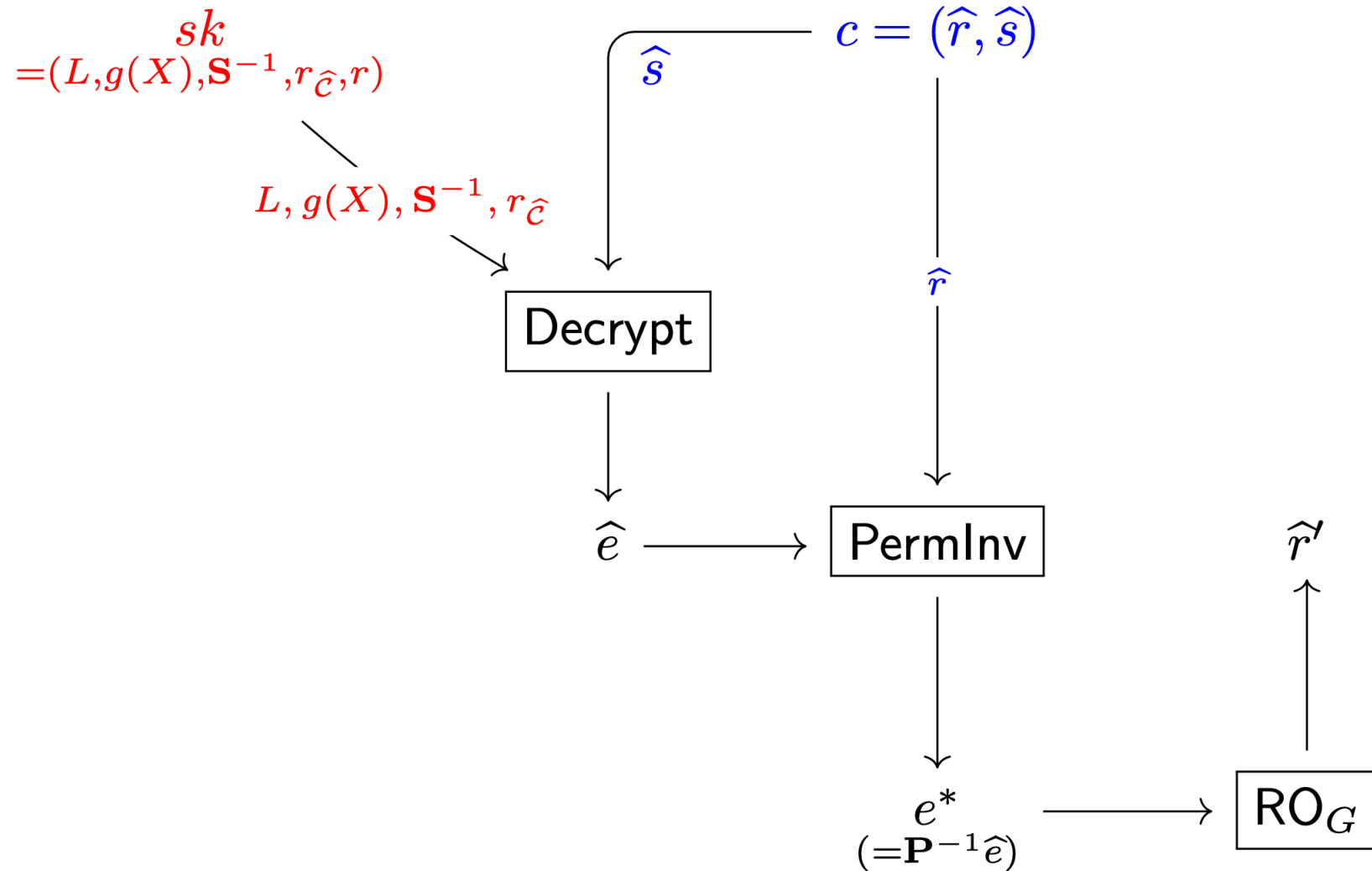
KEM structure – Decapsulation



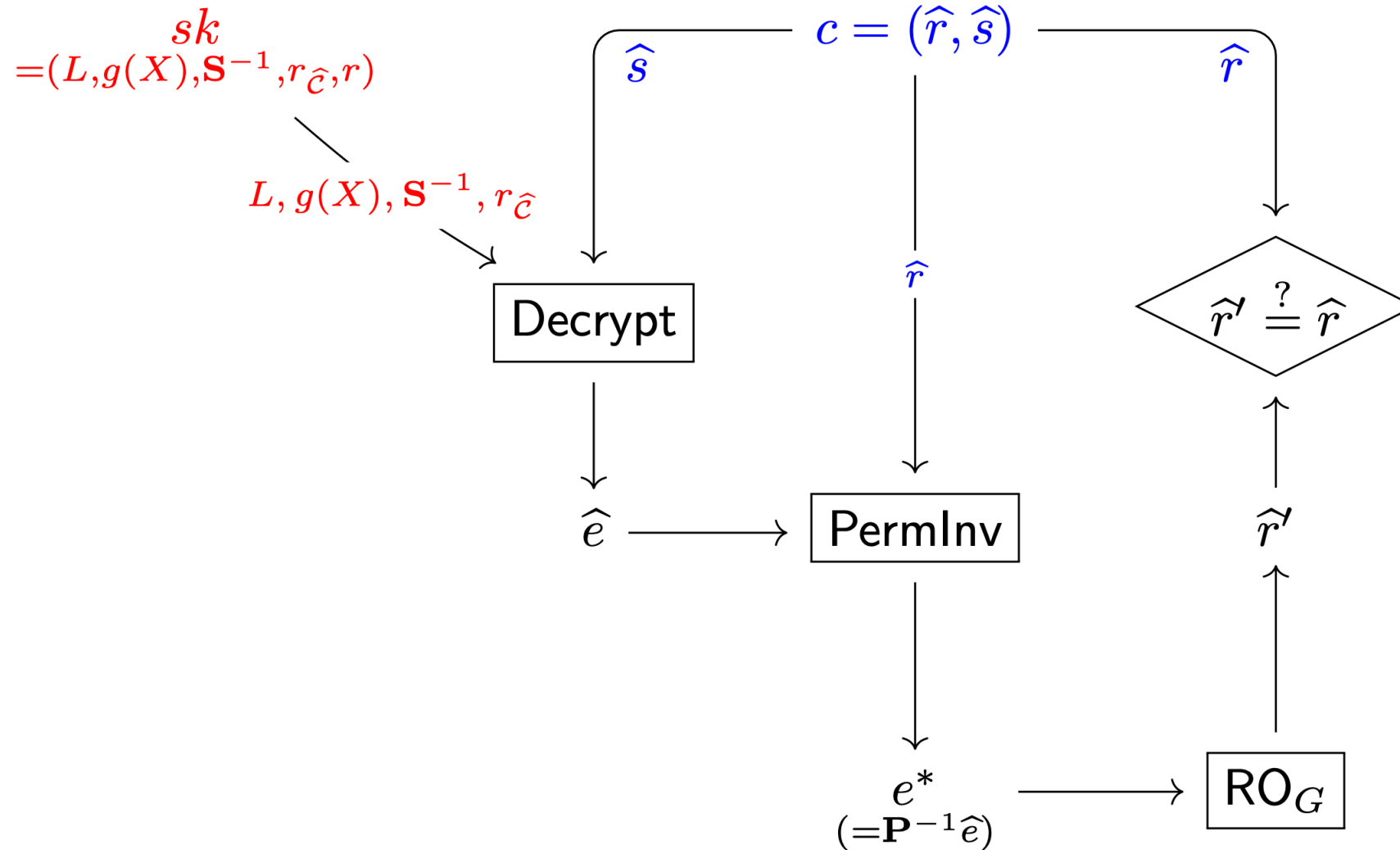
KEM structure – Decapsulation



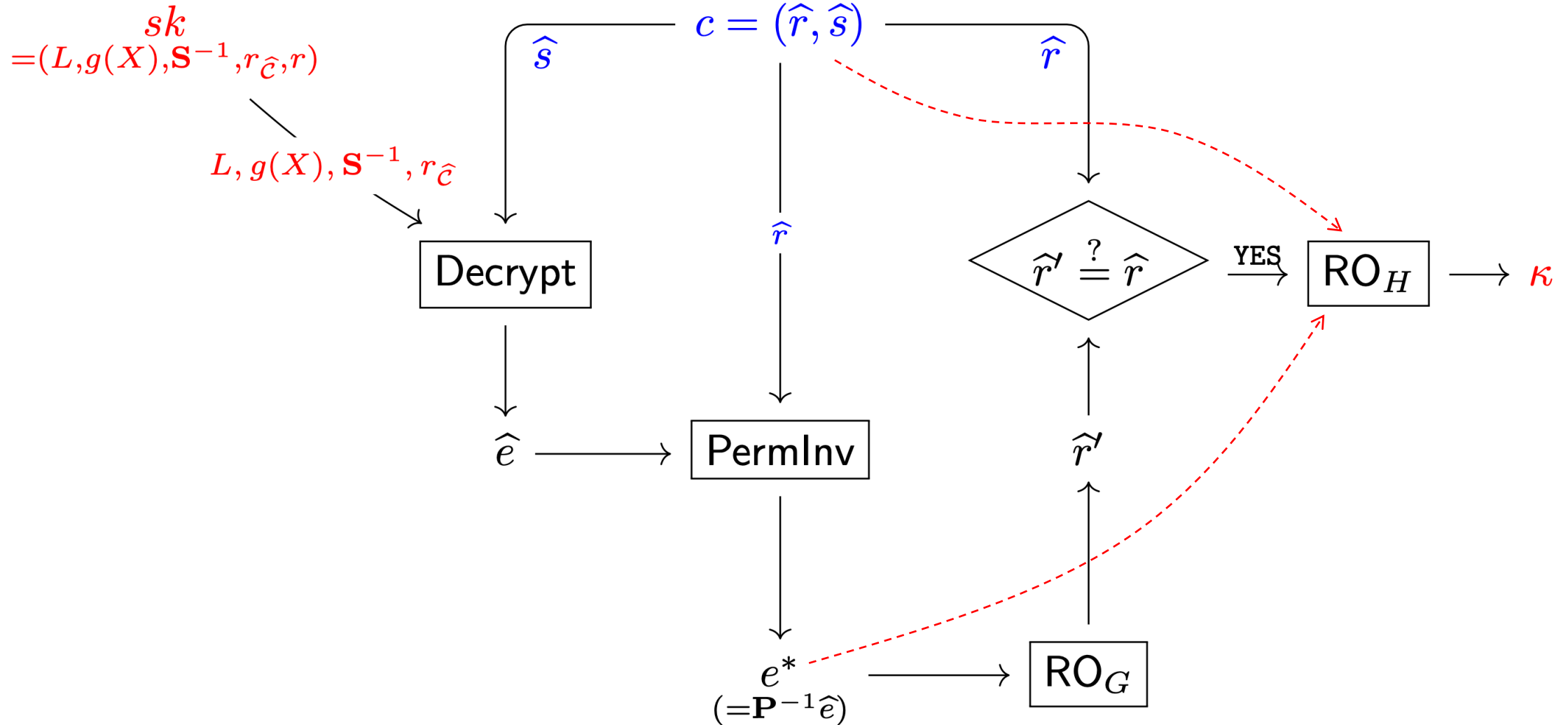
KEM structure – Decapsulation



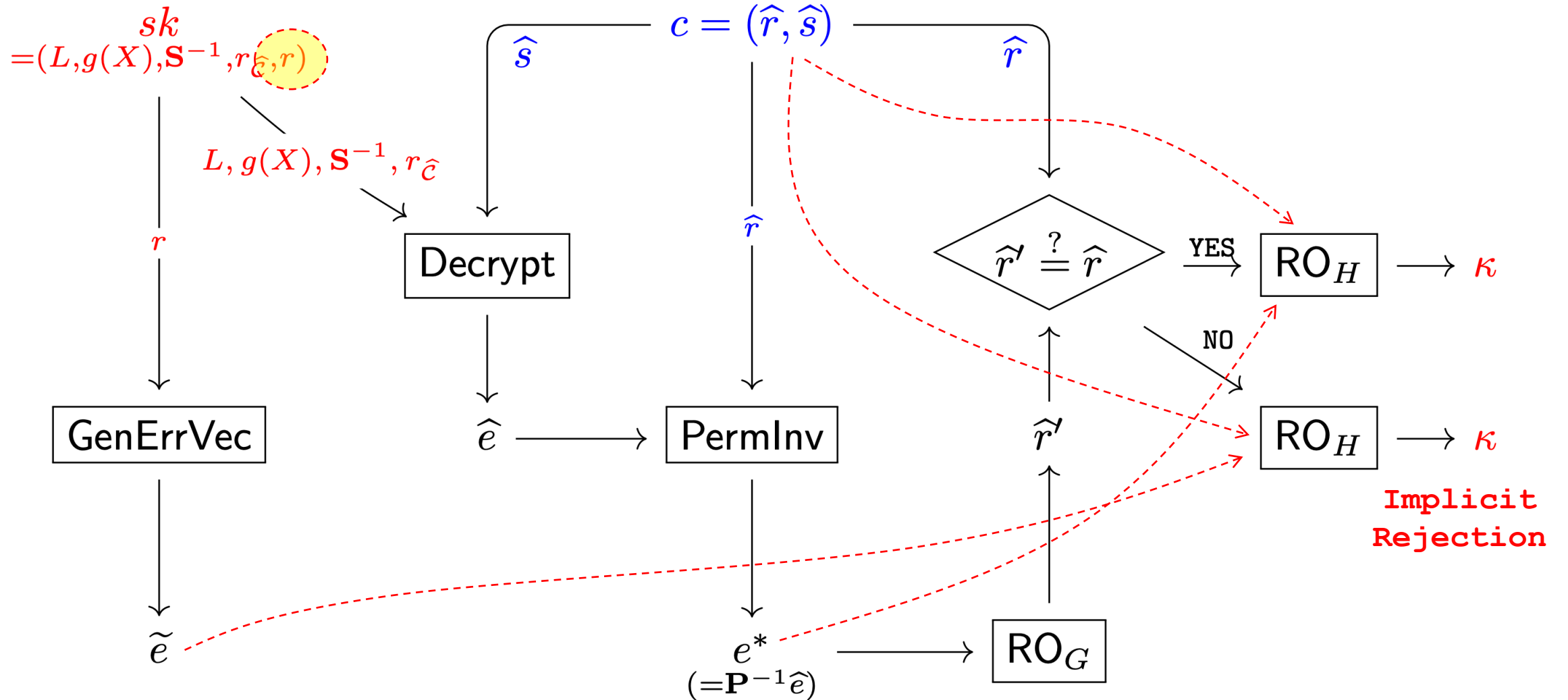
KEM structure – Decapsulation



KEM structure – Decapsulation



KEM structure – Decapsulation



Parameter sets

Criteria for Parameter Set Selection

- 1. For implementation,
(1) $m=13 \rightarrow n+t \leq 2^{13}$, (2) $n \equiv k \equiv t \equiv 0 \pmod{64}$
- 2. For key size, $k/n > 0.7$
- 3. For security, enough security margin

Parameter set	n	t	$k(= n - mt)$	m
PALOMA-128	3904	64	3072	13
PALOMA-192	5568	128	3904	13
PALOMA-256	6592	128	4928	13

Contents

1. Preliminaries (ECC / SDP / code-based trapdoor / binary Goppa code)
2. The design rationale of PALOMA
3. Specification (trapdoor / KEM / parameter sets)
- 4. Evaluation of security (trapdoor / KEM)**
5. Performance (data size / speed)
6. Comparison between PALOMA and Classic McEliece

Security – Onewayness trapdoor

	BJMM- ISD	Improved Birthday-type	Birthday-type	Exhaustive Search
PALOMA-128	$2^{166.21} (l = 67, p = 14)$	$2^{225.78}$	$2^{244.11}$	$2^{476.52}$
PALOMA-192	$2^{267.77} (l = 105, p = 22)$	$2^{399.67}$	$2^{448.91}$	$2^{885.11}$
PALOMA-256	$2^{289.66} (l = 126, p = 26)$	$2^{415.59}$	$2^{464.66}$	$2^{916.62}$
mceliece348864	$2^{161.97} (l = 66, p = 14)$	$2^{220.26}$	$2^{238.75}$	$2^{465.91}$
mceliece460896	$2^{215.59} (l = 86, p = 18)$	$2^{311.80}$	$2^{345.58}$	$2^{678.88}$
mceliece6688128	$2^{291.56} (l = 126, p = 26)$	$2^{416.95}$	$2^{466.01}$	$2^{919.32}$
mceliece6960119	$2^{289.92} (l = 136, p = 28)$	$2^{402.41}$	$2^{443.58}$	$2^{874.57}$
mceliece8192128	$2^{318.34} (l = 157, p = 32)$	$2^{436.05}$	$2^{484.90}$	$2^{957.10}$

Security – Onewayness trapdoor

ePrint 2023/940

CryptAttackTester: high-assurance attack analysis

Daniel J. Bernstein^{1,2} and Tung Chou³

¹ University of Illinois at Chicago, USA

² Ruhr University Bochum, Germany

³ Academia Sinica, Taiwan

Abstract. Quantitative analyses of the costs of cryptographic attack algorithms play a central role in comparing cryptosystems, guiding the search for improved attacks, and deciding which cryptosystems to standardize. Unfortunately, these analyses often turn out to be wrong. Sometimes errors are not caught until years later.

This paper introduces CryptAttackTester (CAT), a software framework for high-assurance quantification of attack effectiveness. CAT enforces complete definitions of attack algorithms all the way down through the model of computation, enforces complete definitions of probability predictions and cost predictions all the way down through the cost metric, and systematically tests the predictions on small-scale inputs.

Birthday-type	Exhaustive Search
$2^{244.11}$	$2^{476.52}$
$2^{448.91}$	$2^{885.11}$
$2^{464.66}$	$2^{916.62}$
$2^{238.75}$	$2^{465.91}$
$2^{345.58}$	$2^{678.88}$
$2^{466.01}$	$2^{919.32}$
$2^{443.58}$	$2^{874.57}$
$2^{484.90}$	$2^{957.10}$

Security – Onewayness trapdoor

	BJMM-ISD	Improved Birthday-type	Birthday-type	Exhaustive Search
PALOMA-128	$2^{166.21}$ ($l = 67, p = 14$)	$2^{225.78}$	$2^{244.11}$	$2^{476.52}$
PALOMA-192	$2^{267.77}$ ($l = 104, p = 14$)			
PALOMA-256	$2^{289.66}$ ($l = 136, p = 14$)			
mceliece348864	$2^{161.97}$ ($l = 67, p = 14$)			
mceliece460896	$2^{215.59}$ ($l = 104, p = 14$)			
mceliece6688128	$2^{291.56}$ ($l = 136, p = 14$)			
mceliece6960119	$2^{289.92}$ ($l = 136, p = 14$)			
mceliece8192128	$2^{318.34}$ ($l = 136, p = 14$)			

F.12. 2022 PALOMA. [78, Section 5.1] uses the “number of bit operations of ISD” to assess the security of PALOMA, a McEliece-based KEM proposed in [78]. No definition of “bit operations” is specified in [78].

[78, page 39, Table 5.2] lists “BJMM-ISD” as $2^{166.21}$ bit operations for $(n, k, t) = (3904, 3072, 64)$, $2^{267.77}$ bit operations for $(n, k, t) = (5568, 3904, 128)$, and $2^{289.66}$ bit operations for $(n, k, t) = (6592, 4928, 128)$.

Starting from the CAT package, we changed the `isdpredict1.py` script to list these choices of (n, k, t) , changed the definition of the `uniformmatrix` problem to allow $k = n - 13t$ rather than $k = n - 12t$ for the $n = 3904$ case, and then ran `isdpredict1.py` followed by `isdpredict2.py` as in Section 10.2. CAT found attack parameters for which the predicted costs of `isd2` are $2^{153.74}$, $2^{229.63}$, and $2^{255.45}$ respectively.

It is easy to see that [78] considers fewer attack speedups than this paper does. A complete reconciliation of the numbers would be much more time-consuming: one would have to trace through the bit-operation counts in [78] and the calculations of success probabilities, comparing to the details in CAT.

Security – IND-CCA2-secure KEM

PALOMA is designed by the FO-like transformation $\text{KEM}^\mathcal{A}$:

$$\text{KEM}^\mathcal{A} = \mathcal{U}^\mathcal{A}[\mathcal{T}[\text{PKE}, G], H]$$

$$\text{PKE}_0 \rightarrow \text{PKE}_1 = \mathcal{T}[\text{PKE}, G] \rightarrow \text{KEM}^\mathcal{A} = \mathcal{U}^\mathcal{A}[\text{PKE}_1, H]$$

OW-CPA-secure
(assumption)

OW-PCA-secure in ROM

IND-CCA2-secure in ROM

Security – IND-CCA2-secure KEM

Algorithm 20 PALOMA: PKE_0

Input: A public key $pk \in \mathcal{PK}$, a random coin $\hat{r} \in \{0, 1\}^{256}$, $e^* \in \mathcal{E}_t^n$
Output: A ciphertext $(\hat{r}, \hat{s}) \in \{0, 1\}^{256} \times \{0, 1\}^{13t}$

1: **procedure** $\text{Encrypt}_0(pk; \hat{r}; e^*)$
2: $\hat{e} \leftarrow \text{Perm}(e^*, \hat{r})$
3: $\hat{s} \leftarrow \text{Encrypt}(pk; \hat{e})$
4: **return** (\hat{r}, \hat{s})
5: **end procedure**

Input: A secret key $sk \in \mathcal{SK}$, a ciphertext $(\hat{r}, \hat{s}) \in \{0, 1\}^{256} \times \{0, 1\}^{13t}$
Output: $e^* \in \mathcal{E}_t^n$

1: **procedure** $\text{Decrypt}_0(sk; (\hat{r}, \hat{s}))$
2: $\hat{e} \leftarrow \text{Decrypt}(sk; \hat{s})$
3: $e^* \leftarrow \text{PermInv}(\hat{e}, \hat{r})$
4: **return** e^*
5: **end procedure**

Algorithm 21 PALOMA: PKE_1

Input: A public key $pk \in \mathcal{PK}$, $e^* \in \mathcal{E}_t^n$
Output: A ciphertext $(\hat{r}, \hat{s}) \in \{0, 1\}^{256} \times \{0, 1\}^{13t}$

1: **procedure** $\text{Encrypt}_1(pk; e^*)$
2: $\hat{r} \leftarrow \text{RO}_G(e^*)$
3: $(\hat{r}, \hat{s}) \leftarrow \text{Encrypt}_0(pk; \hat{r}; e^*)$
4: **return** (\hat{r}, \hat{s})
5: **end procedure**

Input: A secret key $sk \in \mathcal{SK}$, a ciphertext $(\hat{r}, \hat{s}) \in \{0, 1\}^{256} \times \{0, 1\}^{13t}$
Output: $e^* \in \mathcal{E}_t^n$ or \perp

1: **procedure** $\text{Decrypt}_1(sk; (\hat{r}, \hat{s}))$
2: $e^* \leftarrow \text{Decrypt}_0(sk; \hat{s})$
3: **if** $w_H(e^*) \neq t$ **then**
4: **return** \perp
5: **end if**
6: $\hat{r}' \leftarrow \text{RO}_G(e^*)$
7: **if** $\hat{r}' \neq \hat{r}$ **then**
8: **return** \perp
9: **end if**
10: **return** e^*
11: **end procedure**

Algorithm 23 PALOMA: KEM^\neq

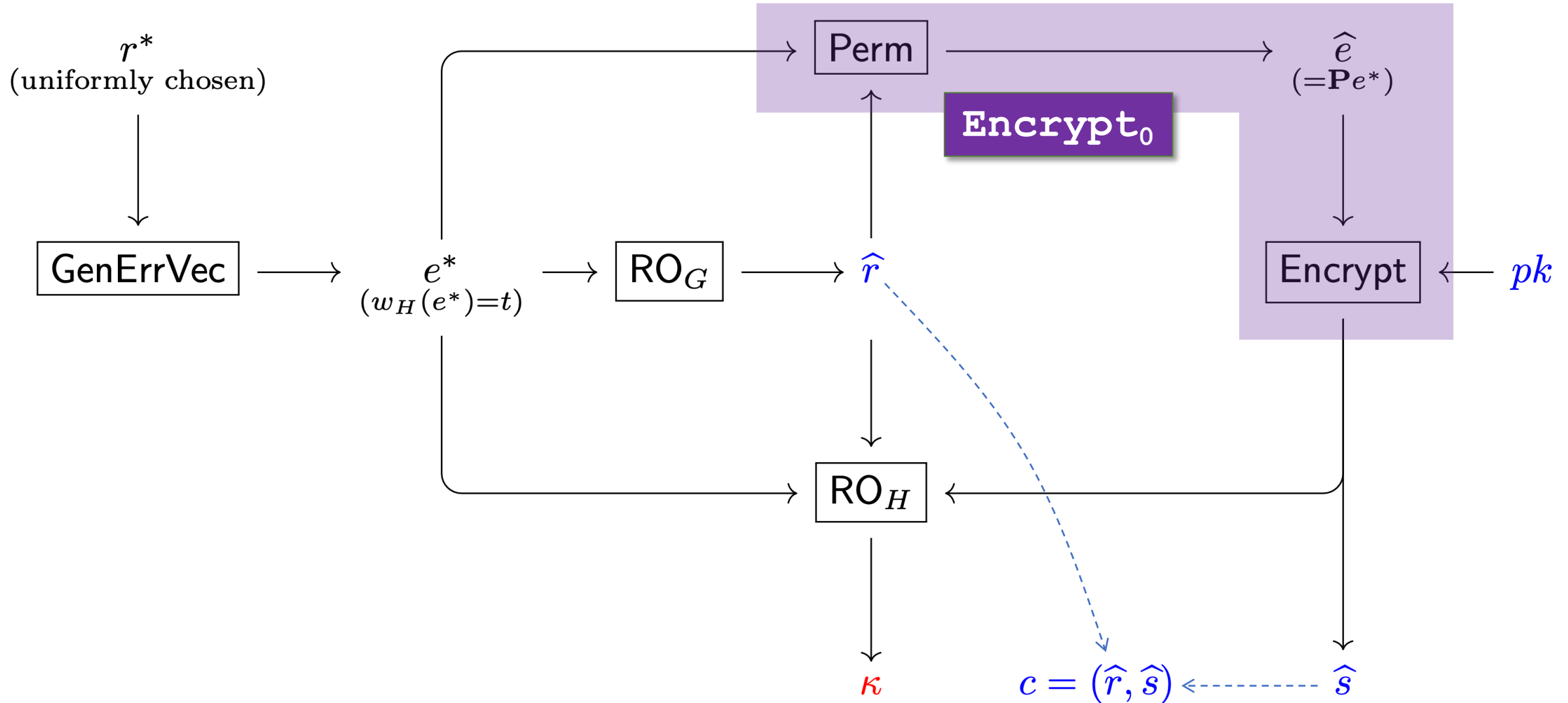
Input: A public key $pk \in \mathcal{PK}$
Output: A ciphertext $(\hat{r}, \hat{s}) \in \{0, 1\}^{256} \times \{0, 1\}^{13t}$ and a key $\kappa \in \{0, 1\}^{256}$

1: **procedure** $\text{Encap}(pk)$
2: $r^* \xleftarrow{\$} \{0, 1\}^{256}$
3: $e^* \leftarrow \text{GenErrVec}(n, t, r^*)$
4: $(\hat{r}, \hat{s}) \leftarrow \text{Encrypt}_1(pk; e^*)$
5: $\kappa \leftarrow \text{RO}_H(e^* || \hat{r} || \hat{s})$
6: **return** (\hat{r}, \hat{s}) and κ
7: **end procedure**

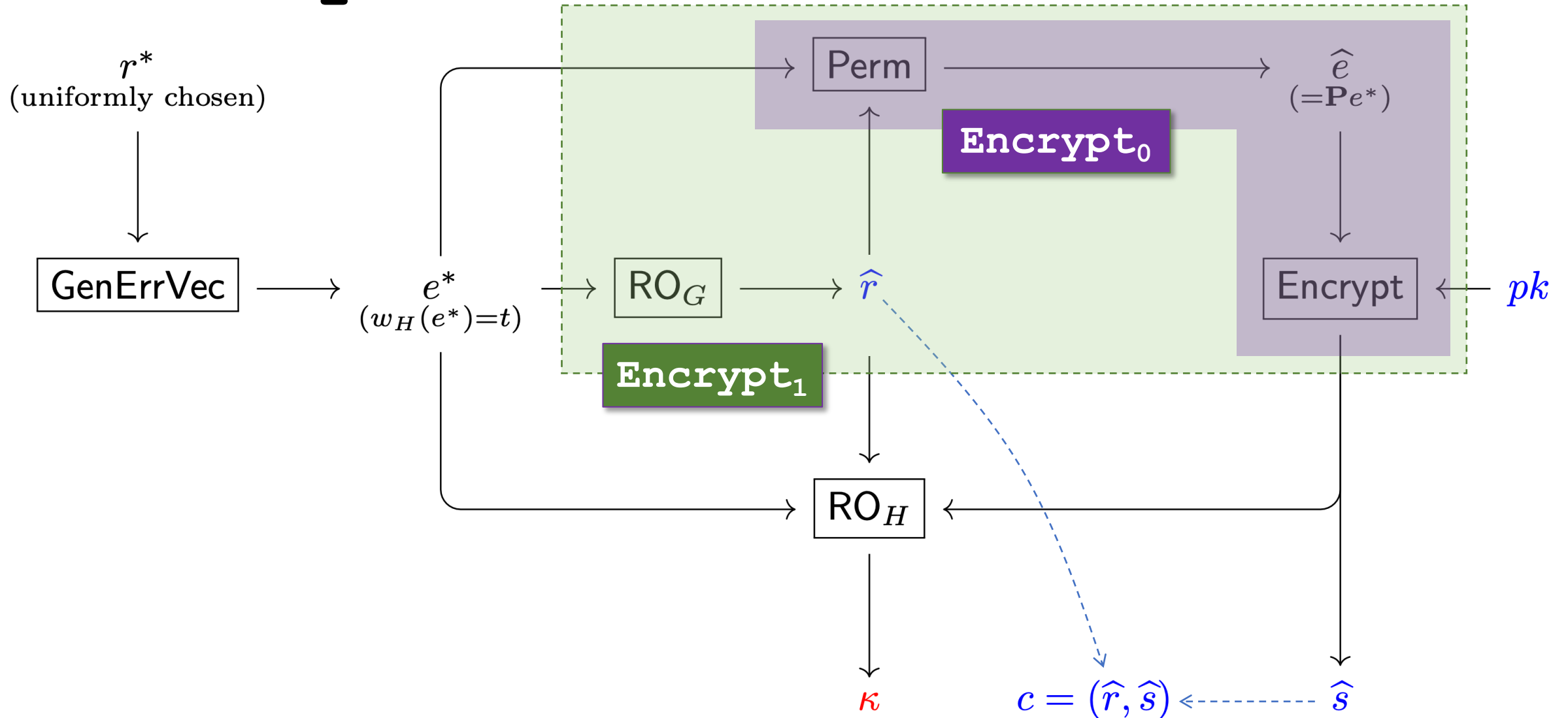
Input: A secret key $sk \in \mathcal{SK}$, a ciphertext $(\hat{r}, \hat{s}) \in \{0, 1\}^{256} \times \{0, 1\}^{13t}$
Output: $e^* \in \mathcal{E}_t^n$ or \perp

1: **procedure** $\text{Decap}(sk; (\hat{r}, \hat{s}))$
2: $e^* \leftarrow \text{Decrypt}_1(sk; (\hat{r}, \hat{s}))$
3: $\tilde{e} \leftarrow \text{GenErrVec}(n, t, r)$ // $r \leftarrow sk$
4: **if** $e^* = \perp$ **then**
5: $\kappa \leftarrow \text{RO}_H(\tilde{e} || \hat{r} || \hat{s})$
6: **else**
7: $\kappa \leftarrow \text{RO}_H(e^* || \hat{r} || \hat{s})$
8: **end if**
9: **return** κ
10: **end procedure**

Security – IND-CCA2-secure KEM



Security – IND-CCA2-secure KEM



Contents

1. Preliminaries (ECC / SDP / code-based trapdoor / binary Goppa code)
2. The design rationale of PALOMA
3. Specification (trapdoor / KEM / parameter sets)
4. Evaluation of security (trapdoor / KEM)
- 5. Performance (data size / speed)**
6. Comparison between PALOMA and Classic McEliece

Performance – Size (in bytes)

		PALOMA-128	PALOMA-192	PALOMA-256
Public key pk	$\hat{\mathbf{H}}_{[n-k:n]} \in \mathbb{F}_2^{(n-k) \times k}$	319,488	812,032	1,025,024
Secret key sk	$L \in \mathbb{F}_{2^{13}}^n$	7,808	11,136	13,184
	$g(X) \in \mathbb{F}_{2^{13}}[X]$	128	256	256
	$\mathbf{S}^{-1} \in \mathbb{F}_2^{(n-k) \times (n-k)}$	86,528	346,112	346,112
	$r_{\hat{c}} \in \{0, 1\}^{256}$	32	32	32
	$r \in \{0, 1\}^{256}$	32	32	32
	Total	94,528	357,568	359,616
Ciphertext c	$\hat{r} \in \{0, 1\}^{256}$	32	32	32
	$\hat{s} \in \mathbb{F}_2^{(n-k)}$	104	208	208
	Total	136	240	240
Key κ	$\kappa \in \{0, 1\}^{256}$	32	32	32

Performance – Size (in bytes)

		PALOMA-128	PALOMA-192	PALOMA-256
Public key pk	$\hat{\mathbf{H}}_{[n-k:n]} \in \mathbb{F}_2^{(n-k) \times k}$	319,488	812,032	1,025,024
Secret key sk	$L \in \mathbb{F}_{2^{13}}^n$	7,808	11,136	13,184
	$g(X) \in \mathbb{F}_{2^{13}}[X]$	128	256	256
	$\mathbf{S}^{-1} \in \mathbb{F}_2^{(n-k) \times (n-k)}$	86,528	346,112	346,112
	$r_{\hat{c}} \in \{0, 1\}^{256}$	32	32	32
	$r \in \{0, 1\}^{256}$	32	32	32
	Total	94,528	357,568	359,616
Ciphertext c	$\hat{r} \in \{0, 1\}^{256}$	32	32	32
	$\hat{s} \in \mathbb{F}_2^{(n-k)}$	104	208	208
	Total	136	240	240
Key κ	$\kappa \in \{0, 1\}^{256}$	32	32	32

96
PALOMA can decrease the secret key to 96-byte

Performance – Size (in bytes)

Security	Algorithm	Public key	Secret key	Ciphertext	Key
128	hqc-128	2,249	40	4,481	64
	BIKE	1,541	281	1,573	32
	mceliece348864	261,120	6,492	96	32
	PALOMA-128	319,488	94,528	136	32
192	hqc-192	4,522	40	9,026	64
	BIKE	3,083	419	3,115	32
	mceliece460896	524,160	13,608	156	32
	PALOMA-192	812,032	357,568	240	32
256	hqc-256	7,245	40	14,469	64
	BIKE	5,122	580	5,154	32
	mceliece6688128	1,044,992	13,932	208	32
	PALOMA-256	1,025,024	359,616	240	32

Performace – Speed (in ms (cycles))

	PALOMA-128		PALOMA-192		PALOMA-256	
	Plat. 1	Plat. 2	Plat. 1	Plat. 2	Plat. 1	Plat. 2
GenKeyPair	39.58 (947,517)	27.81 (1,021,606)	167.70 (4,025,479)	119.77 (2,939,786)	207.22 (5,001,633)	150.03 (3,665,877)
Encrypt	0.003 (56)	0.003 (42)	0.005 (89)	0.004 (74)	0.006 (109)	0.005 (88)
Decrypt	2.57 (61,778)	1.71 (42,278)	13.03 (310,759)	8.92 (214,352)	13.70 (328,884)	9.42 (225,706)
Encap	0.052 (1,257)	0.042 (1,014)	0.061 (1,444)	0.051 (1,279)	0.067 (1,611)	0.060 (1,368)
Decap	2.56 (62,322)	1.75 (40,848)	13.00 (312,372)	8.97 (212,087)	13.67 (329,616)	9.43 (225,231)

Plat 1. MacBookPro macOS, Sonoma 14.1.2, Apple M1 Max, 32GB RAM, Apple clang version 15.0.0
Plat 2. MacBookPro macOS, Sonoma 14.2.1, Apple M3, 8GB RAM, Apple clang version 14.0.3
* AES256-CTR-DRBG is applied (OpenSSL)

Performace – Speed (in ms (cycles))

		PALOMA-128		PALOMA-192		PALOMA-256	
		M1	Intel	M1	Intel	M1	Intel
GENKEYPAIR	GENRANDGOPPACODE	15	26	74	144	93	168
	GETSCRAMBLEDCODE	42	61	179	263	211	281
	total	64	89	261	423	323	469
ENCRYPT		0.002	0.003	0.003	0.004	0.003	0.005
DECRYPT	CONSTRUCTKEYEQN	8	12	53	92	53	92
	SOLVEKEYEQN	0.2	0.4	2	3	2	3
	FINDERREVEC	1	2	3	4	4	5
	total	10	14	59	100	59	101
ENCAP		0.03	0.05	0.04	0.07	0.04	0.08
DECAP		9	15	59	101	60	101

1R

2R

	PALOMA-128		PALOMA-192		PALOMA-256	
	Plat. 1	Plat. 2	Plat. 1	Plat. 2	Plat. 1	Plat. 2
GenKeyPair	39.58 (947,517)	27.81 (1,021,606)	167.70 (4,025,479)	119.77 (2,939,786)	207.22 (5,001,633)	150.03 (3,665,877)
Encrypt	0.003 (56)	0.003 (42)	0.005 (89)	0.004 (74)	0.006 (109)	0.005 (88)
Decrypt	2.57 (61,778)	1.71 (42,278)	13.03 (310,759)	8.92 (214,352)	13.70 (328,884)	9.42 (225,706)
Encap	0.052 (1,257)	0.042 (1,014)	0.061 (1,444)	0.051 (1,279)	0.067 (1,611)	0.060 (1,368)
Decap	2.56 (62,322)	1.75 (40,848)	13.00 (312,372)	8.97 (212,087)	13.67 (329,616)	9.43 (225,231)

Plat 1. MacBookPro macOS, Sonoma 14.1.2, Apple M1 Max, 32GB RAM, Apple clang version 15.0.0
Plat 2. MacBookPro macOS, Sonoma 14.2.1, Apple M3, 8GB RAM, Apple clang version 14.0.3
* AES256-CTR-DRBG is applied (OpenSSL)

Performace – Speed (in ms (cycles))

		GenKeyPair	Encap	Decap
128-bit	PALOMA-128	27.81 (1,021,606)	0.042 (1,014)	1.75 (40,848)
	mceliece348864f	28.24 (651,804)	0.038 (940)	15.03 (353,312)
192-bit	PALOMA-192	119.77 (2,939,786)	0.051 (1,279)	8.97 (212,087)
	mceliece460896f	79.67 (1,922,618)	0.073 (1,817)	37.16 (898,408)
256-bit	PALOMA-256	150.03 (3,665,877)	0.060 (1,368)	9.43 (225,231)
	mceliece6688128f	139.61 (3,331,050)	0.124 (2,920)	71.94 (1,726,978)

Plat 2. MacBookPro macOS, Sonoma 14.2.1, Apple M3, 8GB RAM, Apple clang version 14.0.3

- AES256-CTR-DRBG is applied (OpenSSL)
- Classic McEliece developer's code is applied

Performace – Speed (in ms (cycles))

		GenKeyPair	Encap	Decap
128-bit	PALOMA-128	27.81 (1,021,606)	0.042 (1,014)	1.75 (40,848)
	mceliece348864f	28.24 (651,804)	0.038 (940)	15.03 (353,312)
192-bit	PALOMA-192	119.77 (2,939,786)	0.051 (1,279)	8.97 (212,087)
	mceliece460896f	79.67 (1,922,618)	0.073 (1,817)	37.16 (898,408)
256-bit	PALOMA-256	150.03 (3,665,877)	0.060 (1,368)	9.43 (225,231)
	mceliece6688128f	139.61 (3,331,050)	0.124 (2,920)	71.94 (1,726,978)

<https://classic.mceliece.org/impl.html>

		quartile	median	quartile
keypair	mceliece348864	35,039,714	56,705,880	67,615,011
	mceliece348864f	35,970,884	35,976,620	35,981,416
	mceliece460896	116,209,838	153,266,214	264,539,700
	mceliece460896f	117,267,744	117,297,677	117,331,130
	mceliece6688128	265,554,240	443,746,986	532,990,499
	mceliece6688128f	274,329,761	274,384,229	274,430,338
enc	mceliece348864	34,951	36,457	38,980
	mceliece460896	69,674	76,086	88,956
	mceliece6688128	165,296	171,442	185,077
dec	mceliece348864	127,036	127,140	127,256
	mceliece460896	262,919	263,046	263,225
	mceliece6688128	305,910	306,212	306,925

Contents

1. Preliminaries (ECC / SDP / code-based trapdoor / binary Goppa code)
2. The design rationale of PALOMA
3. Specification (trapdoor / KEM / parameter sets)
4. Evaluation of security (trapdoor / KEM)
5. Performance (data size / speed)
- 6. Comparison between PALOMA and Classic McEliece**

PALOMA vs. Classic McEliece

	PALOMA	Classic McEliece
Structure	Fujisaki-Okamoto-structure KEM (implicit rejection)	SXY-structure KEM (implicit rejection)
Problem	SDP	SDP
Trapdoor type	Niederreiter	Niederreiter
Field \mathbb{F}_{q^m}	$\mathbb{F}_{2^{13}}$	$\mathbb{F}_{2^{12}}, \mathbb{F}_{2^{13}}$
Linear code \mathcal{C}	Binary separable Goppa code	Binary irreducible Goppa code
Goppa polynomial $g(X)$	Separable (not irreducible)	Irreducible
Time for generating $g(X)$	Constant	Non-constant
Parity-check matrix \mathbf{H} of \mathcal{C}	\mathbf{ABC}	\mathbf{BC}
Parity-check matrix $\hat{\mathbf{H}}$ of $\hat{\mathcal{C}}$	Systematic	Systematic
Decoding algorithm	Extended Patterson	Berlekamp-Massey
Probability of decryption failure (correctness)	0	0



The end

You can download
the 2R proposal of PALOMA
on the KpqC website.

Questions or comments?