



Adapting Cryptographic Tools to a Post-Quantum Context

Or how to improve HQC

Cyrius Nugier

cyrius.nugier@enac.fr

ENAC/SINA/TELECOM/ReSCo

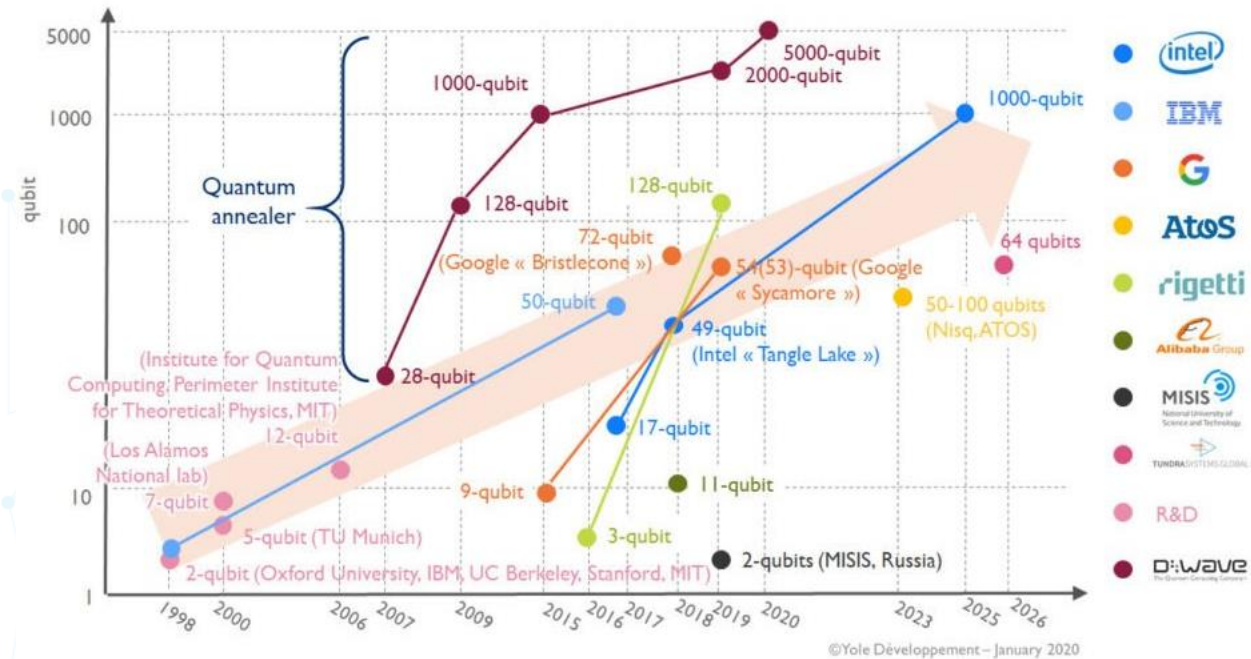


Institut Cybersécurité Occitanie

Why post-quantum cryptography ?

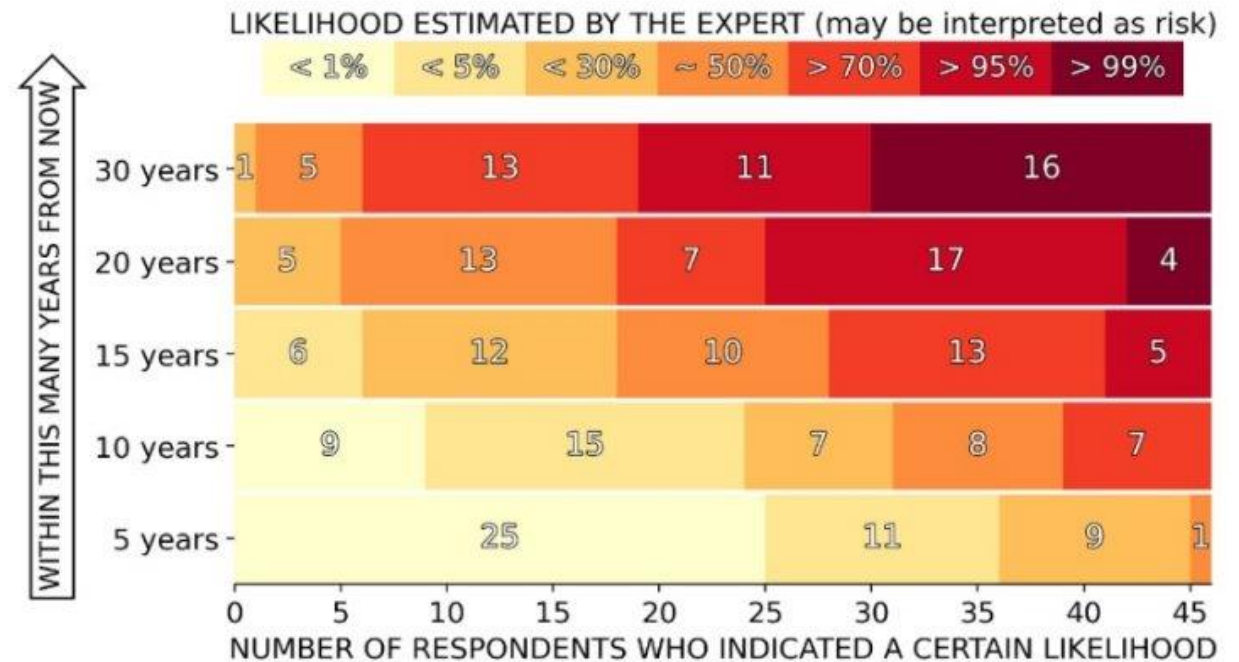
Physical qubit roadmap for quantum computer

(Source: Quantum Technologies 2020 report, Yole Développement)



EXPERTS' ESTIMATES OF THE LIKELIHOOD OF A QUANTUM COMPUTER ABLE TO BREAK RSA-2048 IN 24 HOURS

The experts were asked to indicate their estimate for the likelihood of a quantum computer that is cryptographically relevant—in the specific sense of being able to break RSA-2048 quickly—for various time frames, from a short term of 5 years all the way to 30 years.



When post-quantum cryptography ?

2016 : Call to standardization (69 submissions)

4 Rounds of selections (mathematical and performance criteria)

2025 : Two standardizations: Kyber and HQC

2035 : Classical cryptography disallowed by NIST and ANSSI

Biting the Bullet

Crystals-Kyber	Lattice: MLWE
KINDI	Lattice: MLWE
Saber	Lattice: MLWR
FrodoKEM	Lattice: LWE
Lotus	Lattice: LWE
Lizard	Lattice: LWE/RLWE
Emblem/Remblem	Lattice: LWE/RLWE
KCL	Lattice: LWE/RLWE/LWR
Round 2	Lattice: LWR/RLWR
Hila5	Lattice: RLWE
Ding's key exchange	Lattice: RLWE
Lima	Lattice: RLWE
Lila	Lattice: RLWE
NewHope	Lattice: RLWE
Three Bears	Lattice: IMLWE
Mersenne-756839	Lattice: ILWE
Titanium	Lattice: MP-LWE
Ramstake	Lattice: LWE like
Odd Manhattan	Lattice: Generic
NTRU Encrypt	Lattice: NTRU
NTRU-HRSS-KEM	Lattice: NTRU
NTRUprime	Lattice: NTRU

Crystals-Kyber	Lattice: MLWE
Saber	Lattice: MLWR
FrodoKEM	Lattice: LWE
Round 5	Lattice: LWR/RLWR
LAC	Lattice: RLWE
NewHope	Lattice: RLWE
Three Bears	Lattice: IMLWE
NTRU	Lattice: NTRU
NTRUprime	Lattice: NTRU

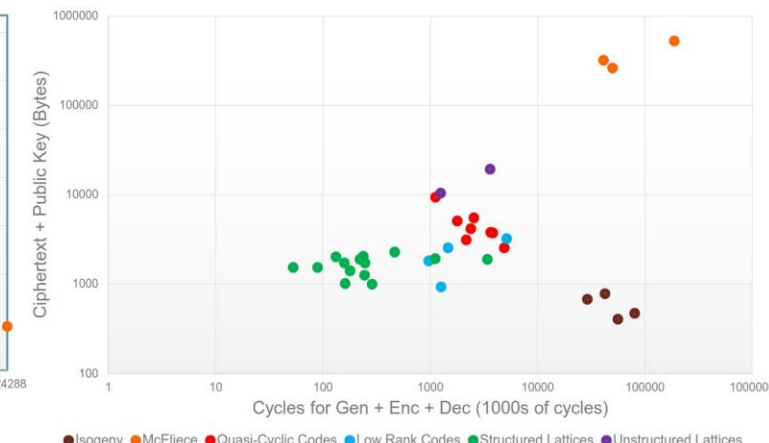
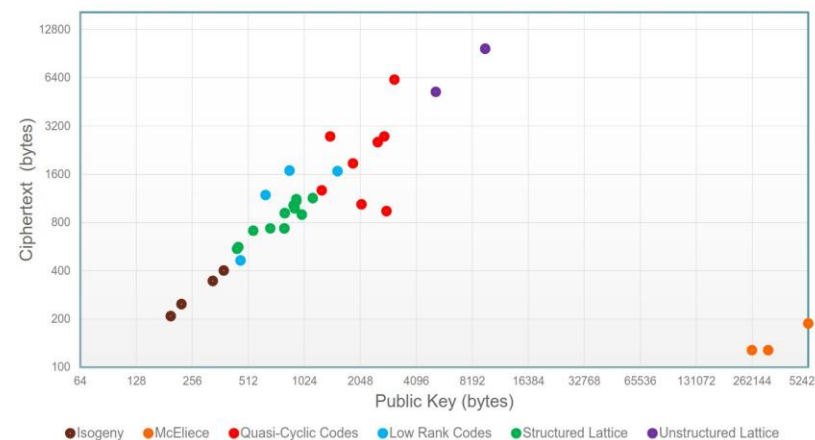
Big Quake	Codes	Goppa
Classic McEliece	Codes	Goppa
NTS-KEM	Codes	Goppa
BIKE	Codes	short Hamming
HQC	Codes	short Hamming
LEDAlkem	Codes	short Hamming
LEDAlke	Codes	short Hamming
QC-MDPC KEM	Codes	short Hamming
LAKE	Codes	low rank
LOCKER	Codes	low rank
Ouroboros-R	Codes	low rank
RQC	Codes	low rank

Classic McEliece	Codes	Goppa
NTS-KEM	Codes	Goppa
BIKE	Codes	short Hamming
HQC	Codes	short Hamming
LEDAlcrypt	Codes	short Hamming
Rollo	Codes	low rank
RQC	Codes	low rank

CRYSTALS-Dilithium	Lattice	Flat-Shamir
qTesla	Lattice	Flat-Shamir
Falcon	Lattice	Hash then sign
pqNTRUSign	Lattice	Hash then sign
Gravity-SPHINCS	Symm	Hash
SPHINCS+	Symm	Hash
Picnic	Symm	ZKP
GeMMS	MultVar	HFE
Gul	MultVar	HFE
HIMQ-3	MultVar	UOV
LUOV	MultVar	UOV
Rainbow	MultVar	UOV
MODSS	MultVar	Flat-Shamir

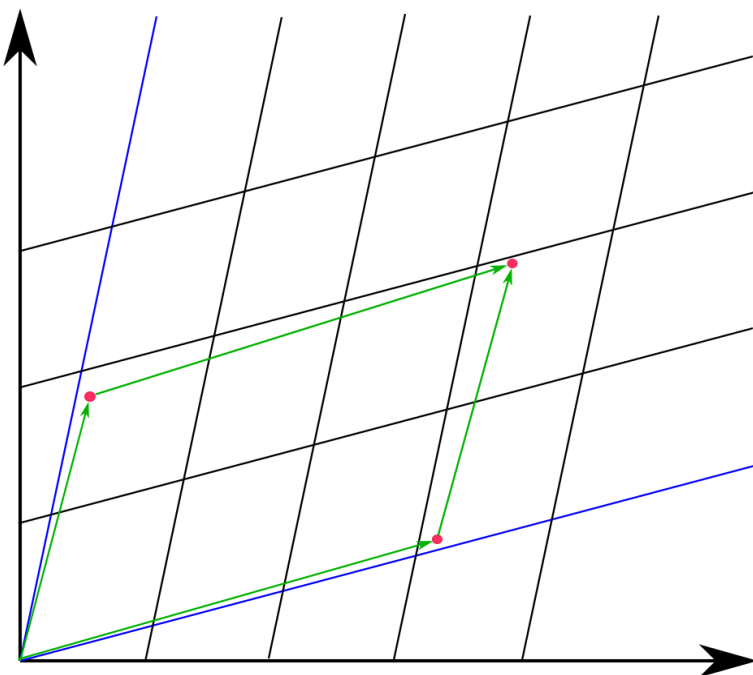
CRYSTALS-Dilithium	Lattice	Flat-Shamir
qTesla	Lattice	Flat-Shamir
Falcon	Lattice	Hash then sign
SPHINCS+	Symm	Hash
Picnic	Symm	ZKP
GeMMS	MultVar	HFE
LUOV	MultVar	UOV
Rainbow	MultVar	UOV
MODSS	MultVar	Flat-Shamir

Key Establishment Scheme	Parameters	Transition
Finite Field DH and MQV [SP80056A]	112 bits of security strength	Deprecated after 2030
	≥ 128 bits of security strength	Disallowed after 2035
Elliptic Curve DH and MQC [SP80056A]	112 bits of security strength	Deprecated after 2030
	≥ 128 bits of security strength	Disallowed after 2035
RSA [SP80056B]	112 bits of security strength	Deprecated after 2030
	≥ 128 bits of security strength	Disallowed after 2035



What is post-quantum cryptography ?

Lattices (Kyber)



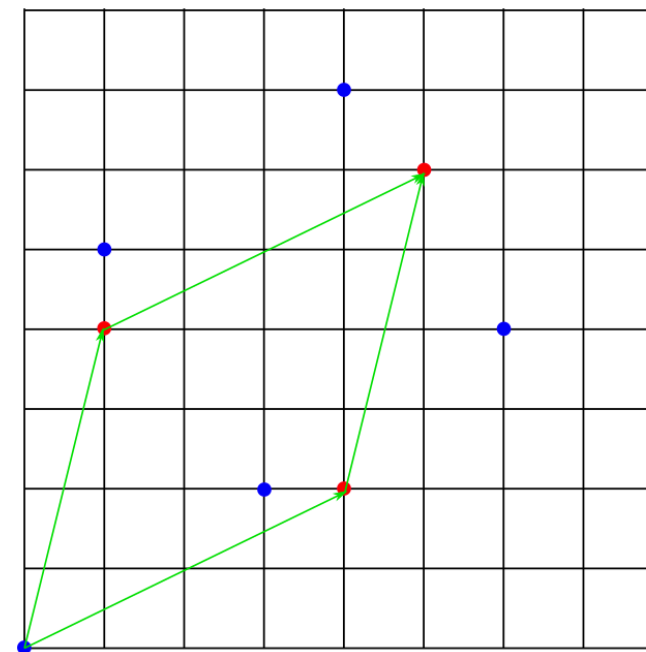
Closest vector problem:

Given a matrix $\mathbf{A}$

and a vector $\mathbf{b} = \mathbf{x} + \mathbf{A}\mathbf{y}$

Find $\mathbf{y}$ such that $\mathbf{x}$ is of the small norm

Codes (HQC)



Syndrome decoding problem:

Given a generator matrix $\mathbf{A}$

and a noisy codeword $\mathbf{b} = \mathbf{x} + \mathbf{A}\mathbf{y}$, with $\mathbf{x}$ of low weight

Find $\mathbf{y}$ such that $\mathbf{x}$ of small weight

The same, but different, but the same

Kyber:

KeyGen:

$$\mathbf{A} \leftarrow \mathbf{R}^{k \times k}$$

$$\text{SK} = (\mathbf{e}, \mathbf{s}) \leftarrow \mathbf{R}^k \text{ from a binomial distribution}$$

$$\text{PK} = (\mathbf{A}, \mathbf{t} = \mathbf{A}\mathbf{s} + \mathbf{e})$$

Encrypt:

$$\mathbf{r}, \mathbf{e}_1, \mathbf{e}_2 \leftarrow \mathbf{R}^k \text{ from a binomial distribution}$$

$$\text{CT} = (\mathbf{u}, \mathbf{v}) = (\mathbf{A}\mathbf{r} + \mathbf{e}_1, \text{Decompress}(m) + \mathbf{t}\mathbf{r} + \mathbf{e}_2)$$

Decrypt:

$$m' = \text{Compress}(\mathbf{v} - \mathbf{u}\mathbf{s})$$

$$= \text{Compress}(D(m) + \mathbf{A}\mathbf{s}\mathbf{r} + \mathbf{e}\mathbf{r} + \mathbf{e}_2 + \mathbf{A}\mathbf{s}\mathbf{r} + \mathbf{s}\mathbf{e}_1)$$

$$= \text{Compress}(D(m) + \mathbf{e}\mathbf{r} + \mathbf{e}_2 + \mathbf{s}\mathbf{e}_1)$$

$$= m \text{ with overwhelming probability}$$

HQC:

KeyGen:

$$\mathbf{h} \leftarrow \mathbf{R}$$

$$\text{SK} = (\mathbf{x}, \mathbf{y}) \leftarrow \mathbf{R}_w^2$$

$$\text{PK} = (\mathbf{h}, \mathbf{s} = \mathbf{h} \cdot \mathbf{y} + \mathbf{x})$$

Encrypt:

$$\mathbf{r}_1, \mathbf{r}_2, \mathbf{e} \leftarrow \mathbf{R}_w^2$$

$$\text{CT} = (\mathbf{u}, \mathbf{v}) = (\mathbf{h} \cdot \mathbf{r}_2 + \mathbf{r}_1, mG + \mathbf{s} \cdot \mathbf{r}_2 + \mathbf{e})$$

Decrypt:

$$m' = \text{Decode}(\mathbf{v} - \mathbf{u} \cdot \mathbf{y})$$

$$= \text{Decode}(mG + \mathbf{y} \cdot \mathbf{h} \cdot \mathbf{r}_2 + \mathbf{x} \cdot \mathbf{r}_2 + \mathbf{e} + \mathbf{y} \cdot \mathbf{h} \cdot \mathbf{r}_2 + \mathbf{y} \cdot \mathbf{r}_1)$$

$$= \text{Decode}(mG + \mathbf{x} \cdot \mathbf{r}_2 + \mathbf{e} + \mathbf{y} \cdot \mathbf{r}_1)$$

$$= m \text{ with overwhelming probability}$$

HQC has worse performance than Kyber, but has a less intensive history of optimization



Contents

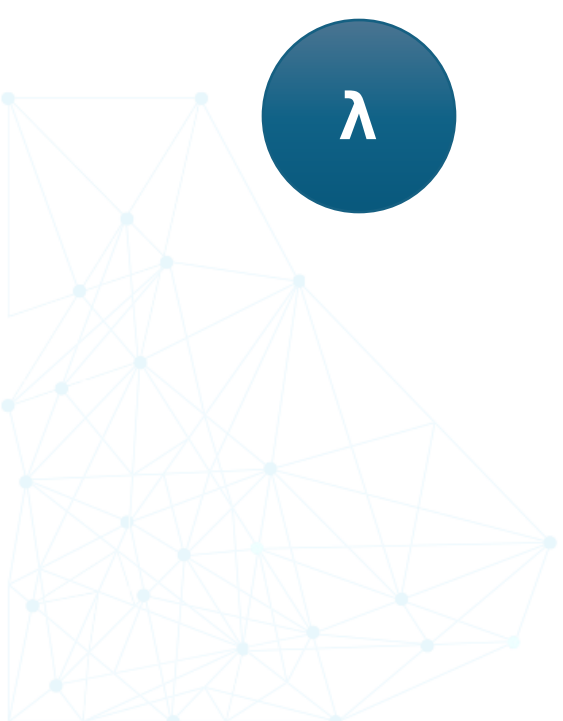
1. **Why is HQC hard to improve ?**
2. **How do we propose to bypass these limitations to improve HQC ?**
3. **Why improving HQC can sometimes be dangerous ?**

N : the golden parameter

PK size	$O(N)$	3024 bytes
SK size	$O(N)$	3064 bytes
CT size	$O(2 \cdot N)$	6017 bytes
KeyGen	mult	$2^{17.60}$ cycles
Encrypt	$2 \cdot \text{mult} + \text{encode}$	$2^{18.32}$ cycles
Decrypt	$3 \cdot \text{mult} + \text{encode} + \text{decode}$	$2^{19.24}$ cycles
mult	$O(N^{1.6})$	
encode	$O(N)$ (~2% of mult)	
decode	$O(N)$ (~10% of mult)	

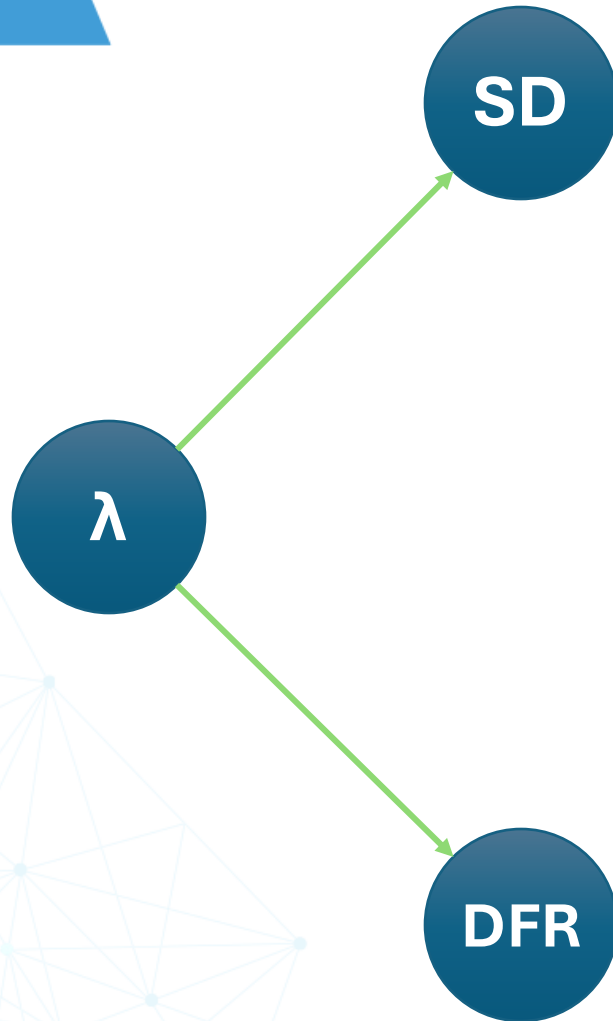


A web of parameters interactions

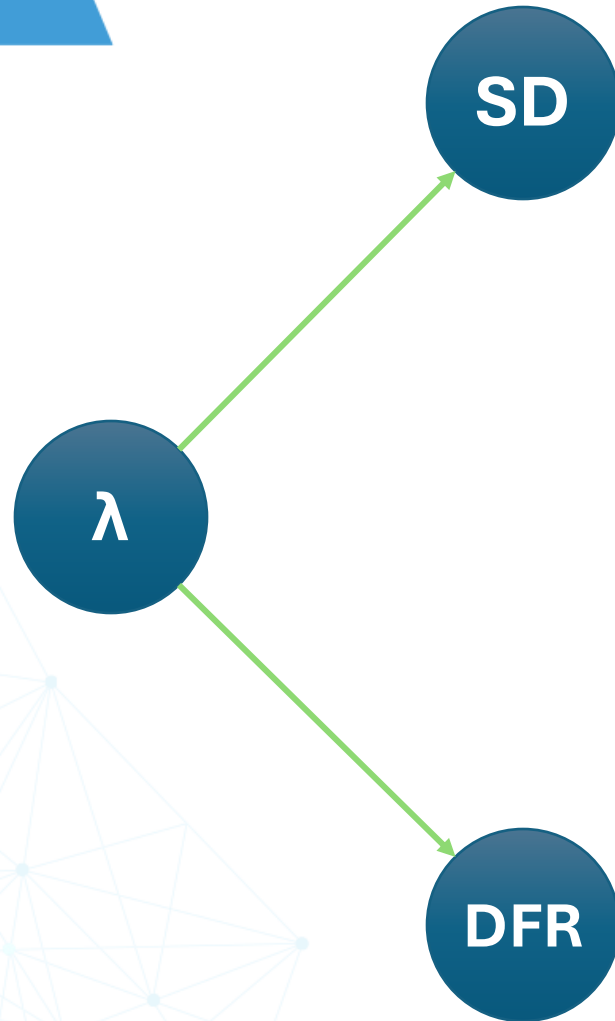
A network diagram consisting of numerous light blue nodes connected by thin lines, forming a complex web-like structure. It is positioned in the bottom left corner of the slide.

λ

A web of parameters interactions



A web of parameters interactions



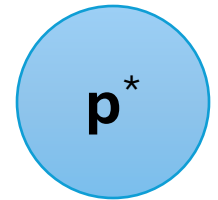
Weight of low-weight polynomials



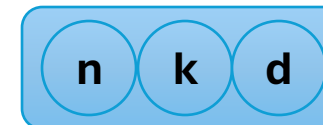
Size of polynomials



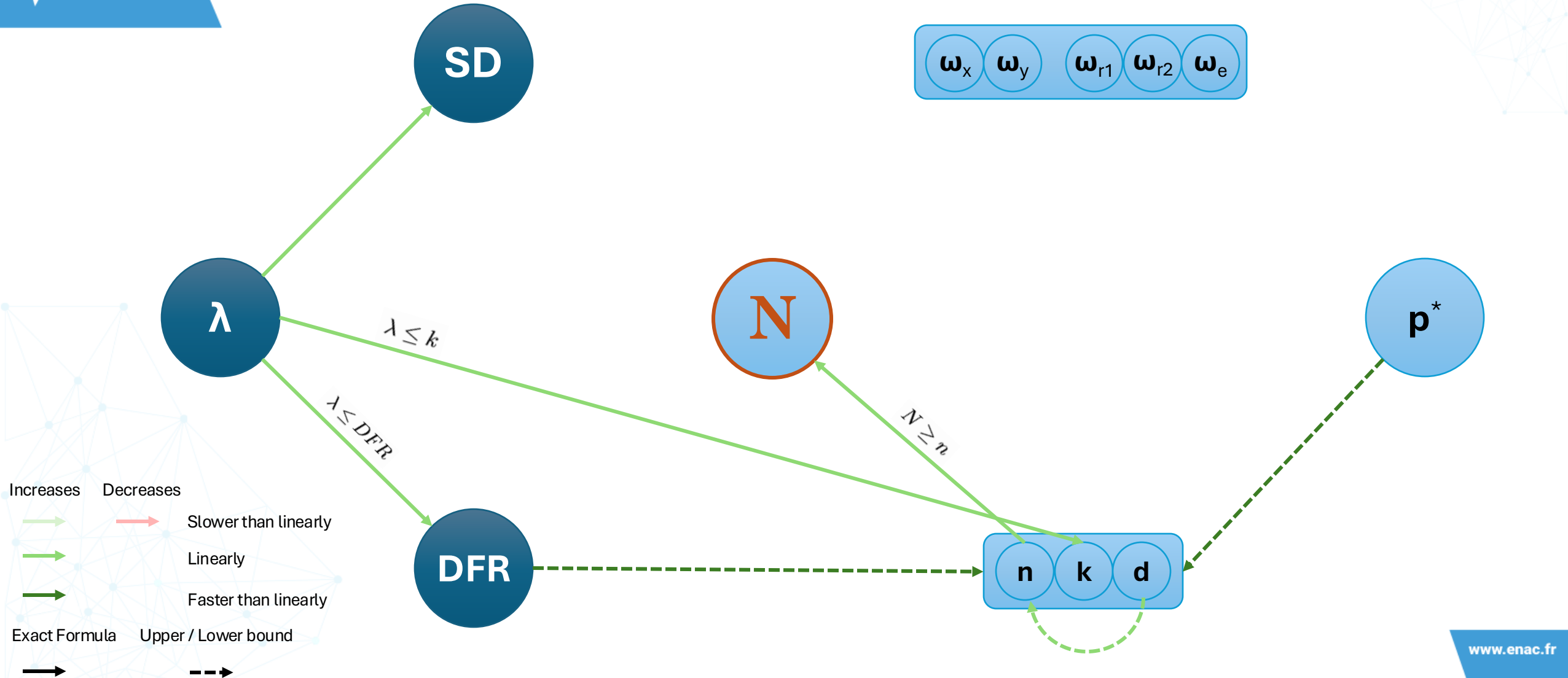
Remaining error rate
before decoding



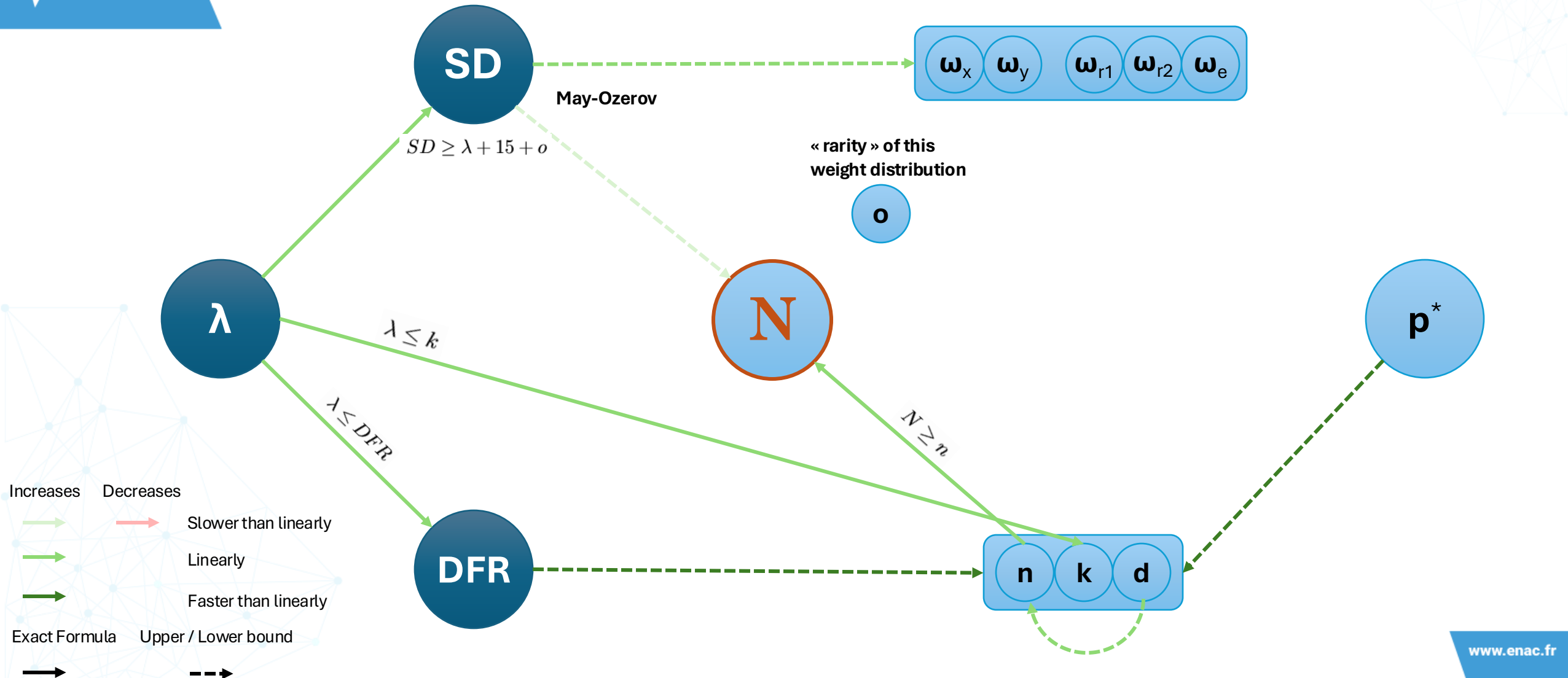
Error Correcting Code

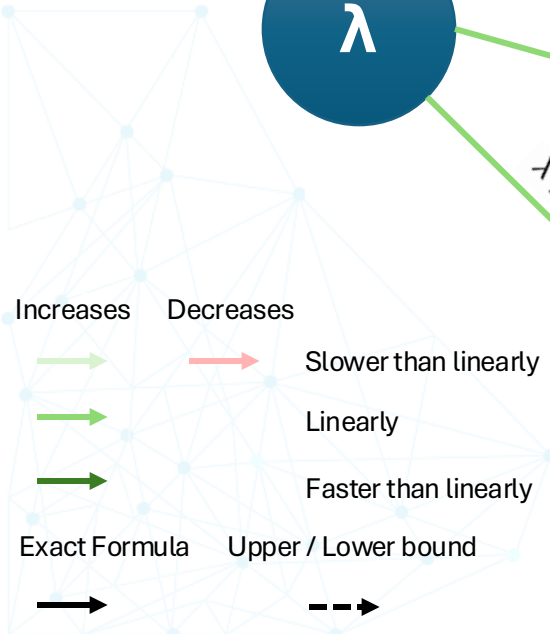


A web of parameters interactions

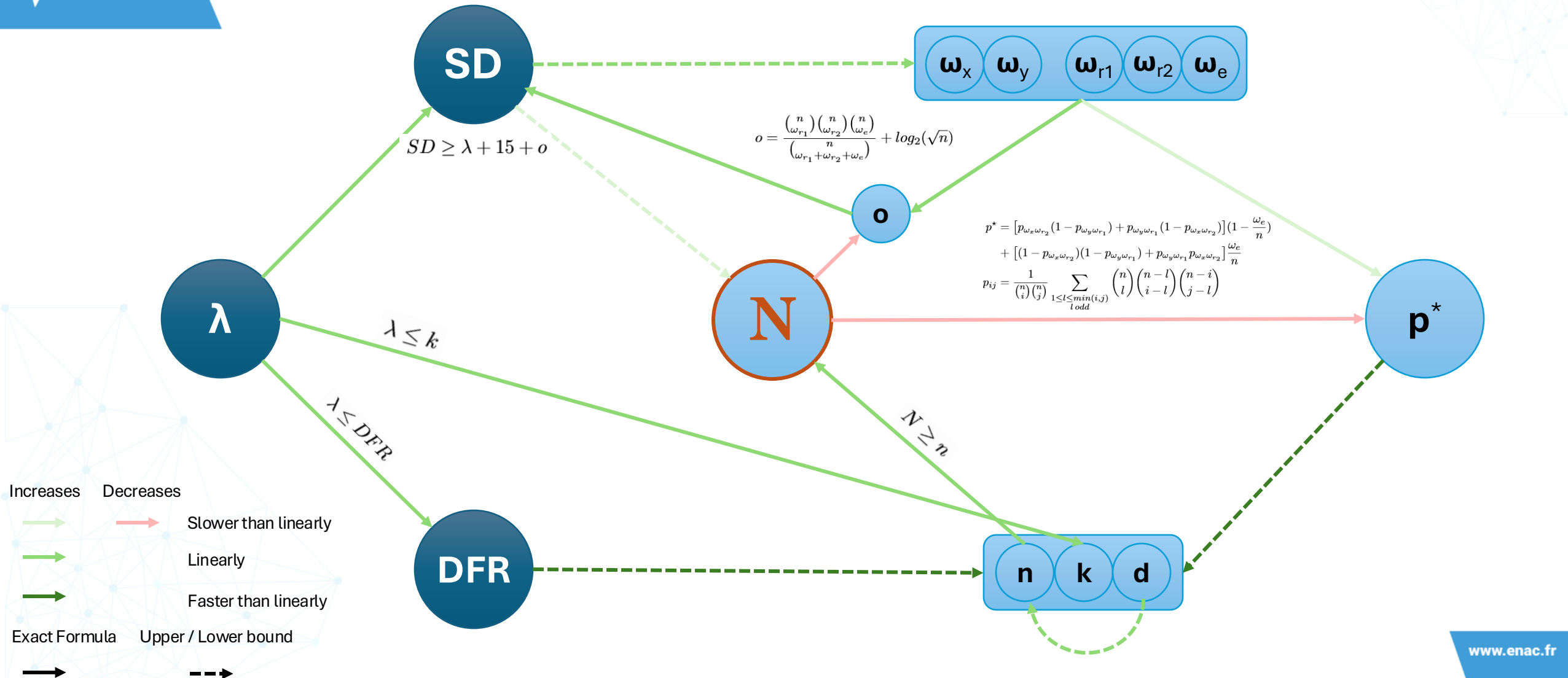


A web of parameters interactions

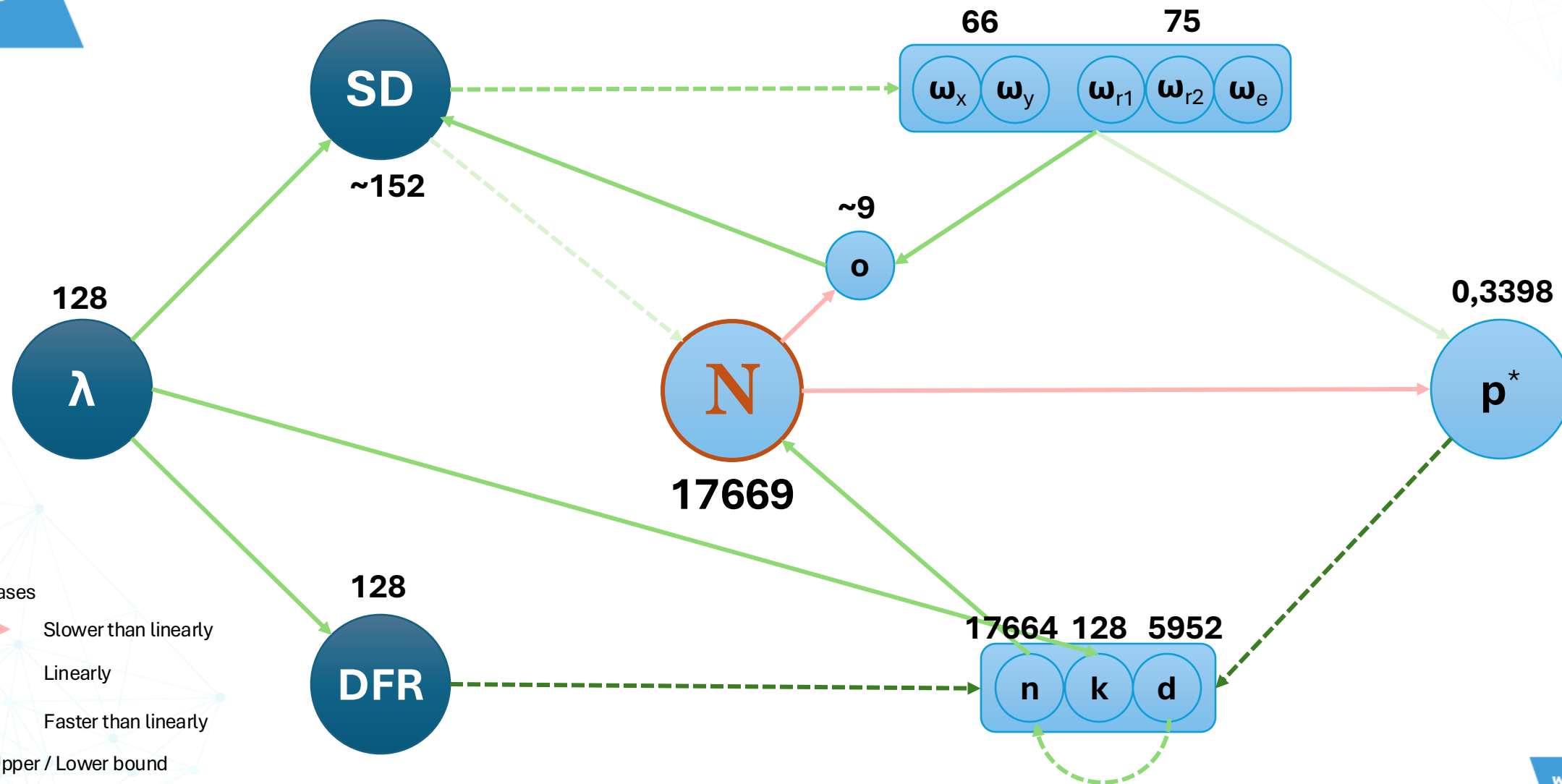




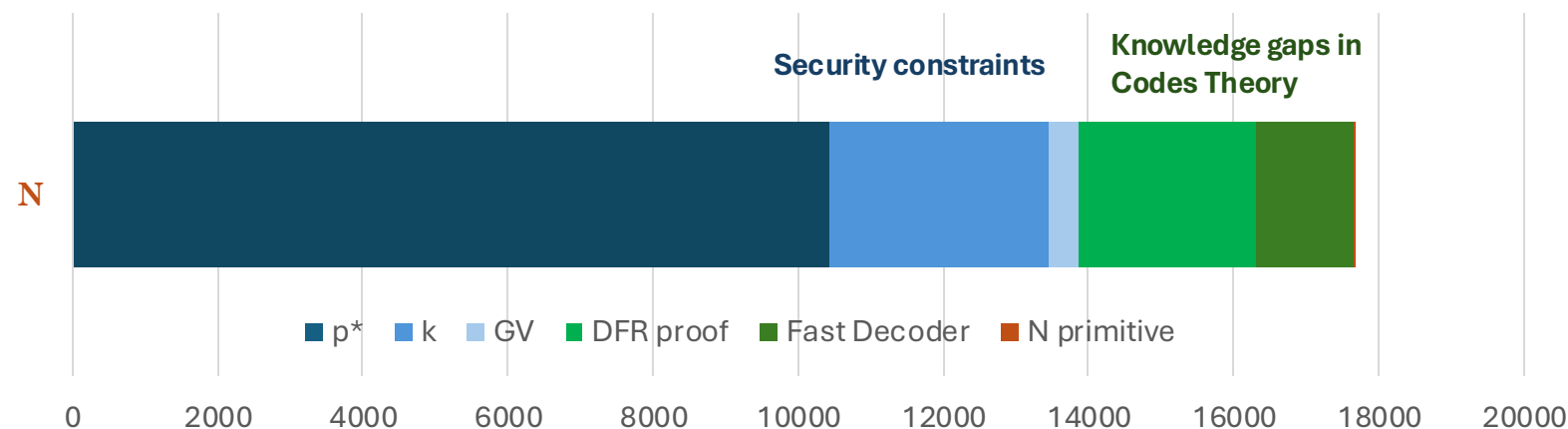
A web of parameters interactions



A web of parameters interactions

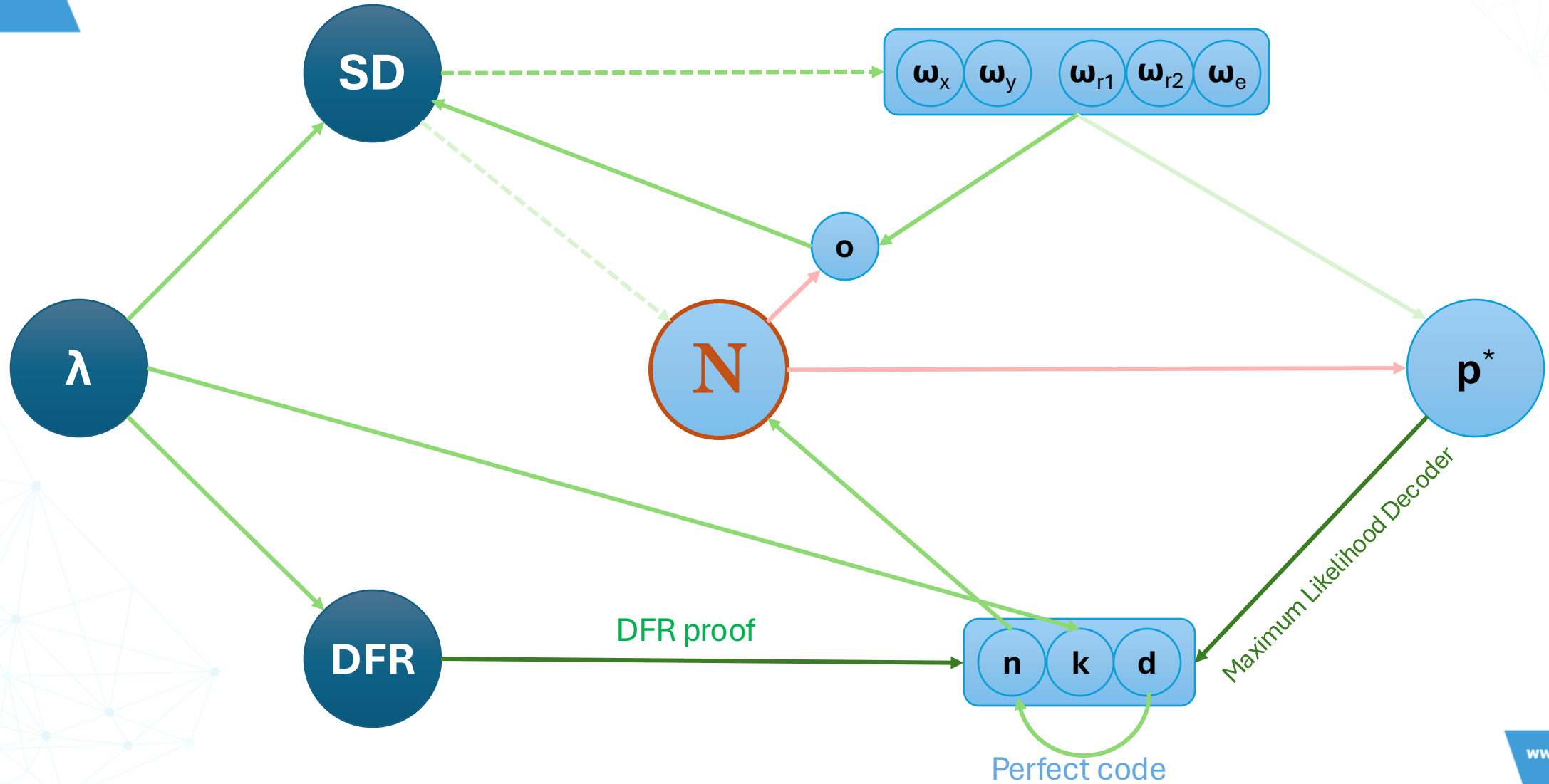


Why is **N** this big ?

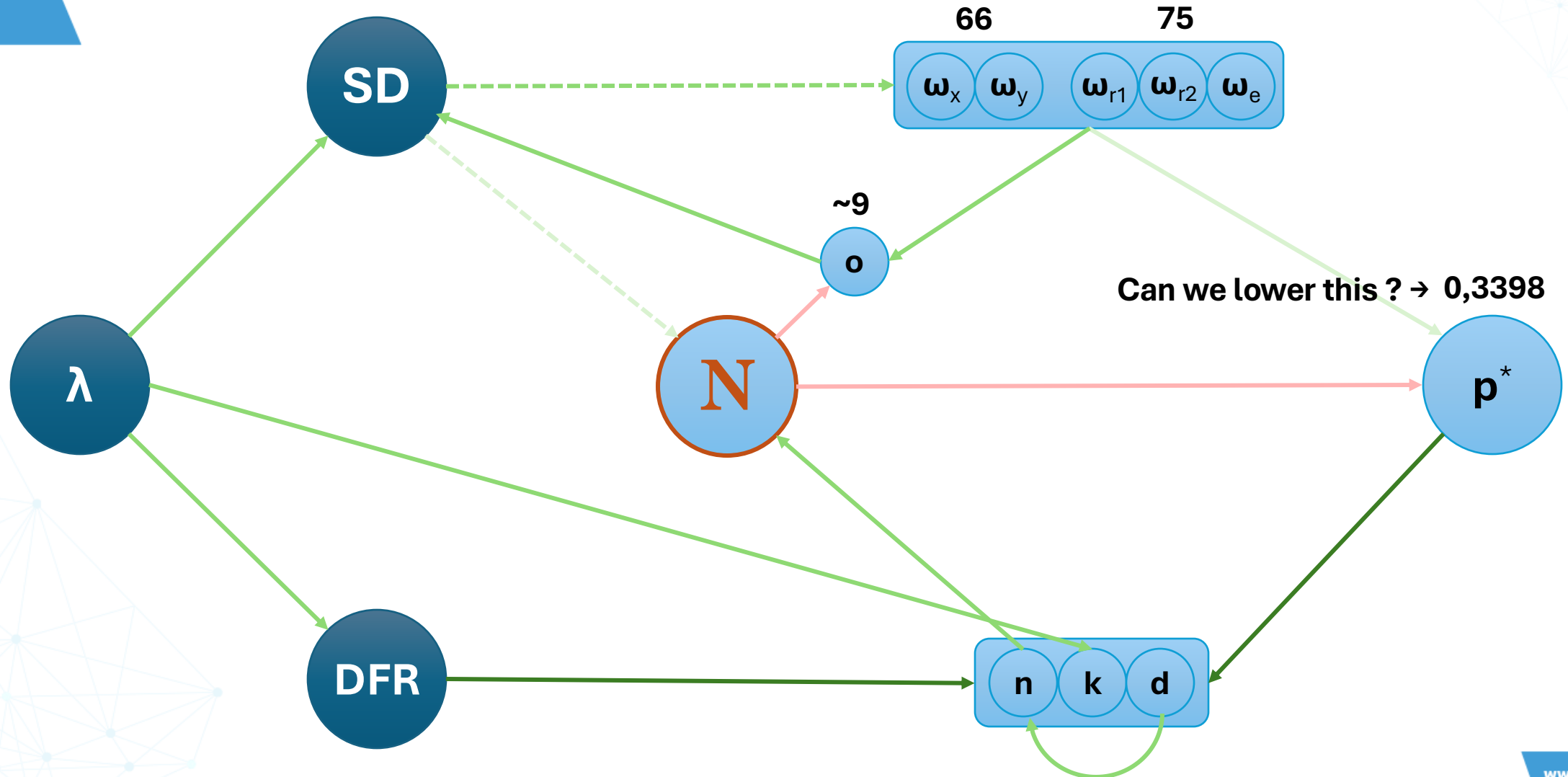


1. For this noise level, even for a **1 bit** message, no smaller code can have a correct DFR
2. Existence bound : there is no code for a **k bit** message below this size
3. GV bound : **codes** are proved to **exist** above this size
4. DFR proof : a tighter **bound**, or another code family can be found
5. Decoder : could be **slower** to correct more errors

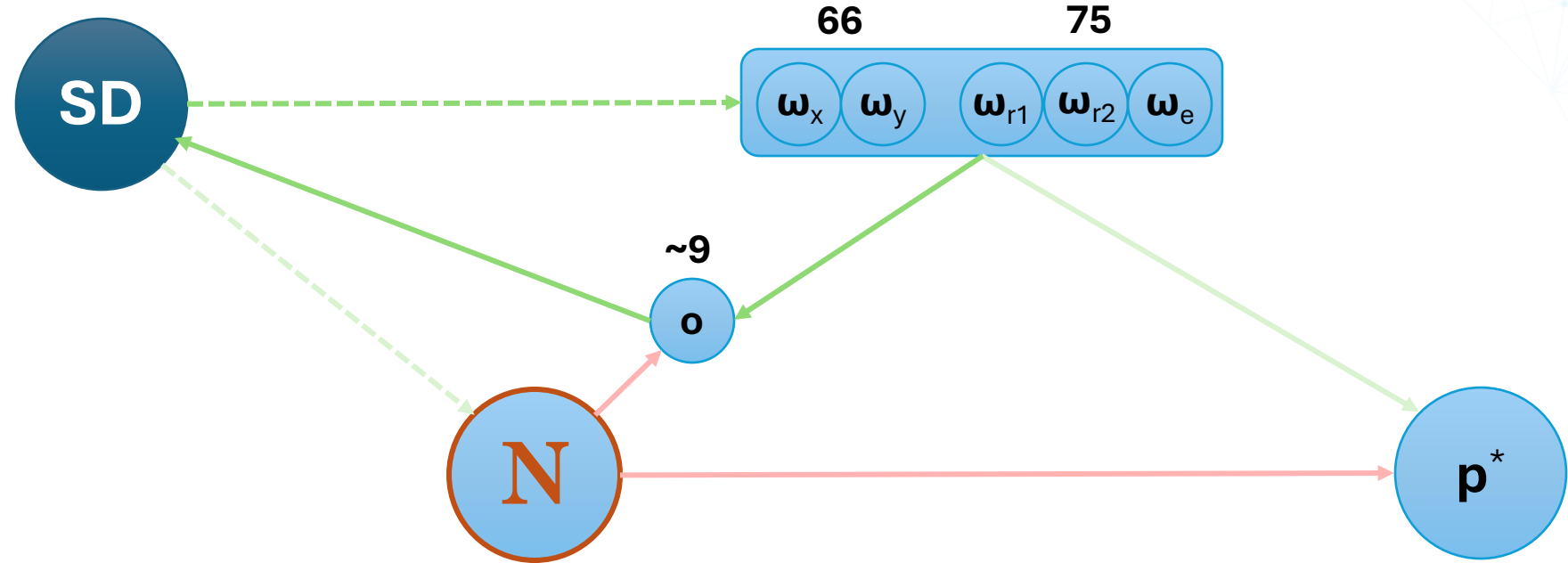
A focus on the code



« Make your requirements less dumb »



Reducing residual errors



$$p^* = [p_{\omega_x \omega_{r2}} (1 - p_{\omega_y \omega_{r1}}) + p_{\omega_y \omega_{r1}} (1 - p_{\omega_x \omega_{r2}})] (1 - \frac{\omega_e}{n}) + [(1 - p_{\omega_x \omega_{r2}}) (1 - p_{\omega_y \omega_{r1}}) + p_{\omega_y \omega_{r1}} p_{\omega_x \omega_{r2}}] \frac{\omega_e}{n}$$

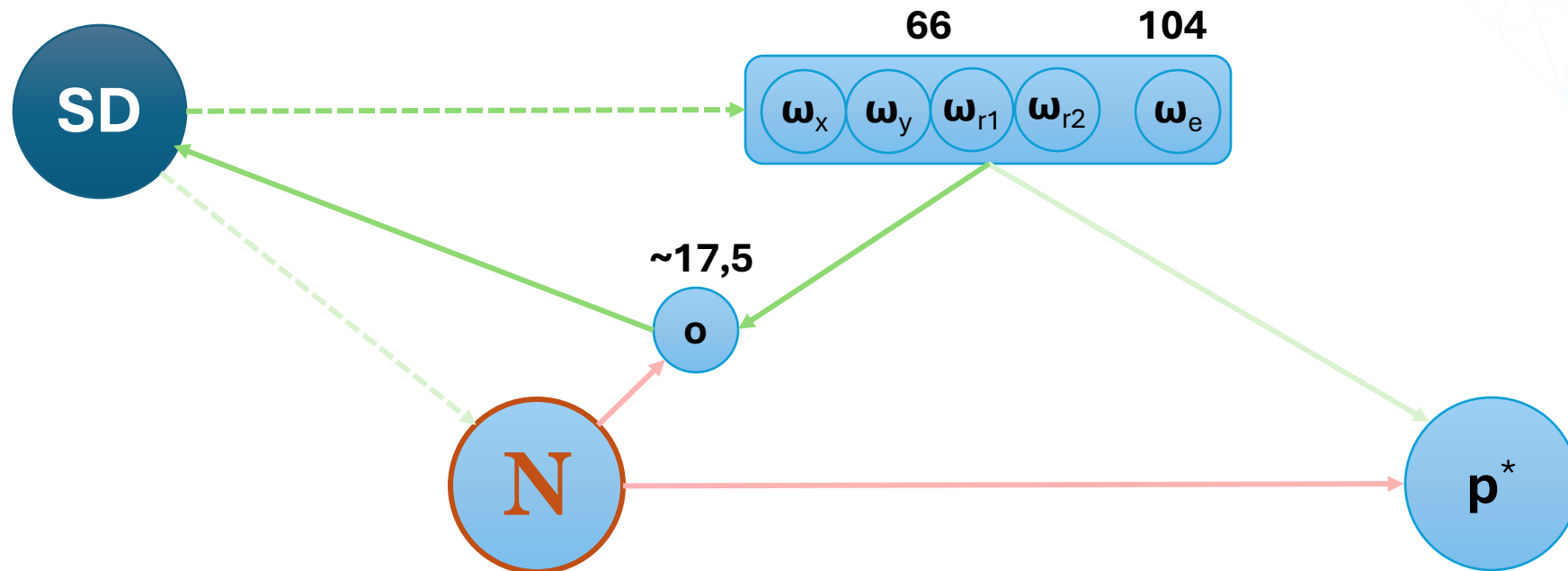
$$p_{ij} = \frac{1}{\binom{n}{i} \binom{n}{j}} \sum_{\substack{1 \leq l \leq \min(i,j) \\ l \text{ odd}}} \binom{n}{l} \binom{n-l}{i-l} \binom{n-l}{j-l}$$

$\omega_x \omega_y \omega_{r1} \omega_{r2}$ all have the same **quadratic** impact

ω_e has a smaller, **linear** impact

Noise can be moved in ω_e , but the distribution becomes rarer (**o** increases)
 → the noise is not moved **1:1**, but **p*** is still **reduced**

Reducing residual errors



$$p^* = [p_{\omega_x \omega_{r_2}} (1 - p_{\omega_y \omega_{r_1}}) + p_{\omega_y \omega_{r_1}} (1 - p_{\omega_x \omega_{r_2}})] (1 - \frac{\omega_e}{n}) + [(1 - p_{\omega_x \omega_{r_2}}) (1 - p_{\omega_y \omega_{r_1}}) + p_{\omega_y \omega_{r_1}} p_{\omega_x \omega_{r_2}}] \frac{\omega_e}{n}$$

$$p_{ij} = \frac{1}{\binom{n}{i} \binom{n}{j}} \sum_{\substack{1 \leq l \leq \min(i,j) \\ l \text{ odd}}} \binom{n}{l} \binom{n-l}{i-l} \binom{n-l}{j-l}$$

$\omega_x \omega_y \omega_{r1} \omega_{r2}$ all have the same **quadratic** impact

ω_e has a smaller, **linear** impact

Noise can be moved in ω_e , but the distribution becomes rarer (**o** increases)
 → the noise is not moved **1:1**, but **p*** is still **reduced**

With this change we can **reduce N** by **7%**

« Make your requirements less dumb » - again

Do we really want to reduce N ?

Decode takes a size n input

u has to be of size N , but v only of size n .

Right now the two are very close,
but they do not have to be...

As **increasing** N reduces errors,
smaller codes can be found,
therefore reducing ciphertext size ($N + n$).

HQC:

KeyGen:

$$h \leftarrow R$$

$$SK = (x, y) \leftarrow R_w^2$$

$$PK = (h, s = h \cdot y + x)$$

Encrypt:

$$r_1, r_2, e \leftarrow R_w^2$$

$$CT = (u, v) = (h \cdot r_2 + r_1, mG + s \cdot r_2 + e)$$

Decrypt:

$$m' = \text{Decode}(v - u \cdot y)$$

$$= \text{Decode}(mG + y \cdot h \cdot r_2 + x \cdot r_2 + e + y \cdot h \cdot r_2 + y \cdot r_1)$$

$$= \text{Decode}(mG + x \cdot r_2 + e + y \cdot r_1)$$

$$= m \text{ with overwhelming probability}$$

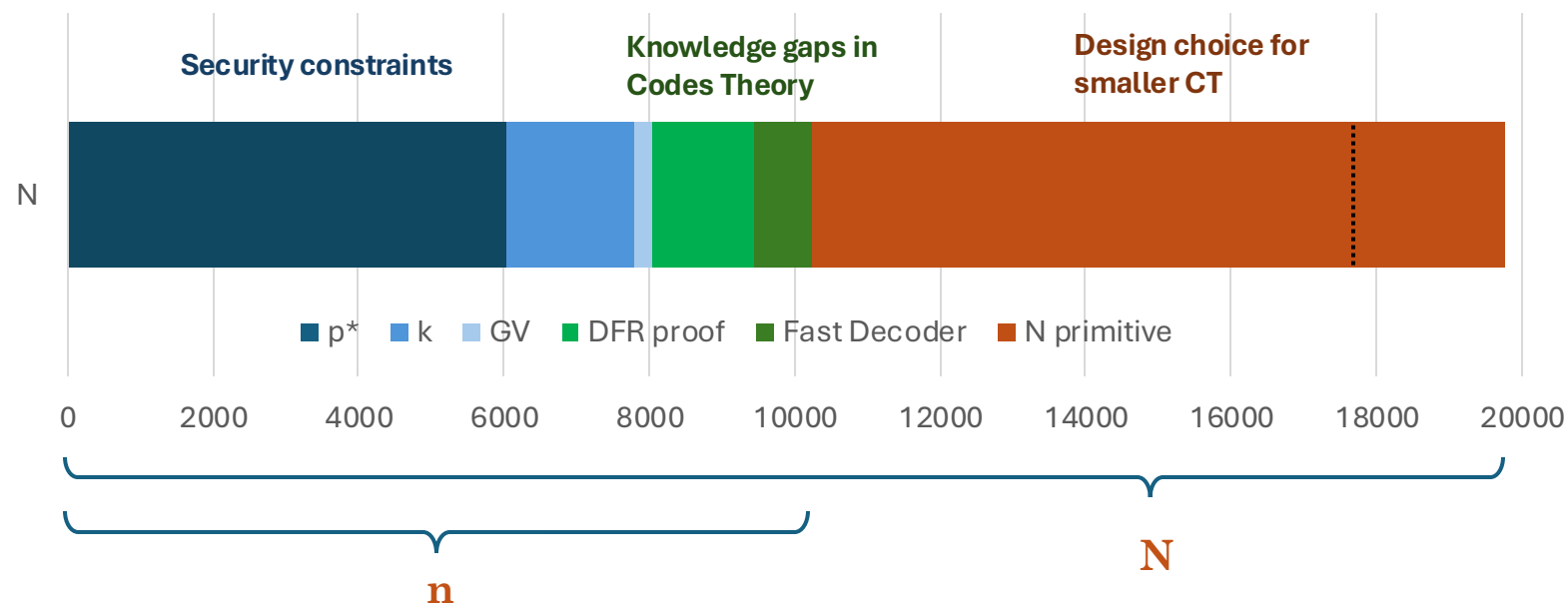
N and n, the important parameters

PK size	$O(N)$	+ 11%
SK size	$O(N)$	+ 11%
CT size	$O(N+n)$	- 15%
KeyGen	multNN	+ 18%
Encrypt	multNN + multNn + encode	> - 8%
Decrypt	multNN + 2·multNn + encode + decode	> - 18%
multNN	$O(N^{1.6})$	+18%
multNn	$O((N \cdot n)^?)$	> - 35%*
encode	$O(n)$	~ - 42%
decode	$O(n)$	~ - 25%

This trade-off is better
for reused keys

*Polynomial multiplication is already
very optimized (Toom cook + Karatsuba)
Speed-up will likely be less than schoolbook

Why are **N** and **n** this big ?



Noise placement has been optimized

Minimal CT has been found, but CT/SK trade-offs are possible

Improvements on Code Theory are now less impactful

Improvements on implementation are crucial, but may lead to weaknesses...

Attacks on HQC decryption

Decode:

```
// Retrieve x, y, pk from secret key
hqc_secret_key_from_string(x, y, sigma, pk, sk);

// Compute v - u.y
vect_resize(tmp1, PARAM_N, v, PARAM_N1N2);
vect_mul(tmp2, y, u);
vect_add(tmp2, tmp1, tmp2, VEC_N_SIZE 64);

// Compute m by decoding v - u.y
code_decode(m, tmp2);
```

```
void code_decode(uint64_t *m, const uint64_t *em) {
    uint64_t tmp[VEC_N1_SIZE 64] = {0};

    reed_muller_decode(tmp, em);
    reed_solomon_decode(m, tmp);
}
```

```
void reed_muller_decode(uint64_t *msg, const uint64_t *cdw) {
    uint8_t *message_array = (uint8_t *) msg;
    codeword *codeArray = (codeword *) cdw;
    expandedCodeword expanded;
    for (size_t i = 0; i < VEC_N1_SIZE_BYTES; i++) {
        // collect the codewords
        expand_and_sum(&expanded, &codeArray[i * MULTIPLICITY]);
        // apply hadamard transform
        expandedCodeword transform;
        hadamard(&expanded, &transform);
        // fix the first entry to get the half Hadamard transform
        transform[0] -= 64 * MULTIPLICITY;
        // finish the decoding
        message_array[i] = find_peaks(&transform);
    }
}
```

```
void expand_and_sum(expandedCodeword *dest, codeword src[]) {
    // start with the first copy
    for (int32_t part = 0; part < 4; part++) {
        for (int32_t bit = 0; bit < 32; bit++) {
            (*dest)[part * 32 + bit] = src[0].u32[part] >> bit & 1;
        }
    }
    // sum the rest of the copies
    for (int32_t copy = 1; copy < MULTIPLICITY; copy++) {
        for (int32_t part = 0; part < 4; part++) {
            for (int32_t bit = 0; bit < 32; bit++) {
                (*dest)[part * 32 + bit] += src[copy].u32[part] >> bit & 1;
            }
        }
    }
}
```

The input of `expand_and_sum` is $v - u \cdot y$

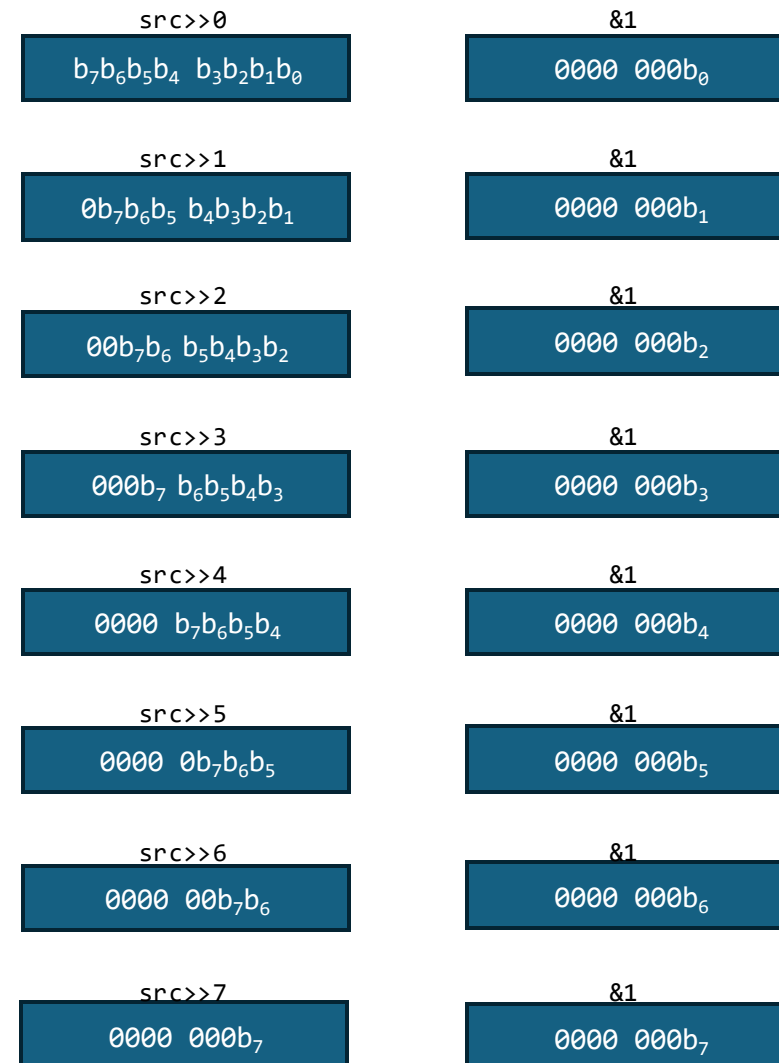
v and u are known, as parts of the ciphertext

They can be used to retrieve the secret key y

Attacks on HQC decryption

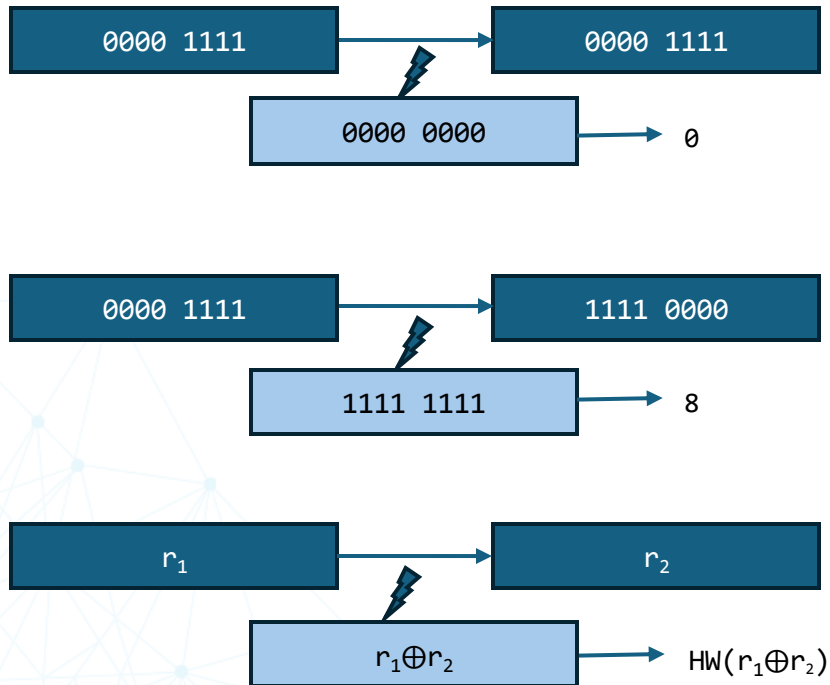
Expand_and_sum:

```
void expand_and_sum(expandedCodeword *dest, codeword src[]) {
    // start with the first copy
    for (int32_t part = 0; part < 4; part++) {
        for (int32_t bit = 0; bit < 32; bit++) {
            (*dest)[part * 32 + bit] = src[0].u32[part] >> bit & 1;
        }
    }
    // sum the rest of the copies
    for (int32_t copy = 1; copy < MULTIPLICITY; copy++) {
        for (int32_t part = 0; part < 4; part++) {
            for (int32_t bit = 0; bit < 32; bit++) {
                (*dest)[part * 32 + bit] += src[copy].u32[part] >> bit & 1;
            }
        }
    }
}
```



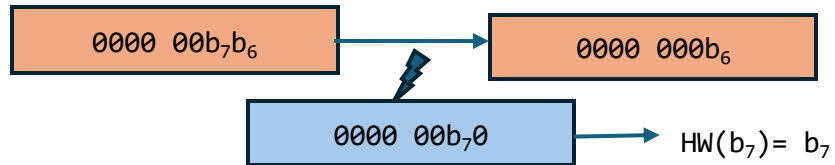
Side Channel Attacks – Instruction Set

Leakage in Hamming distance model



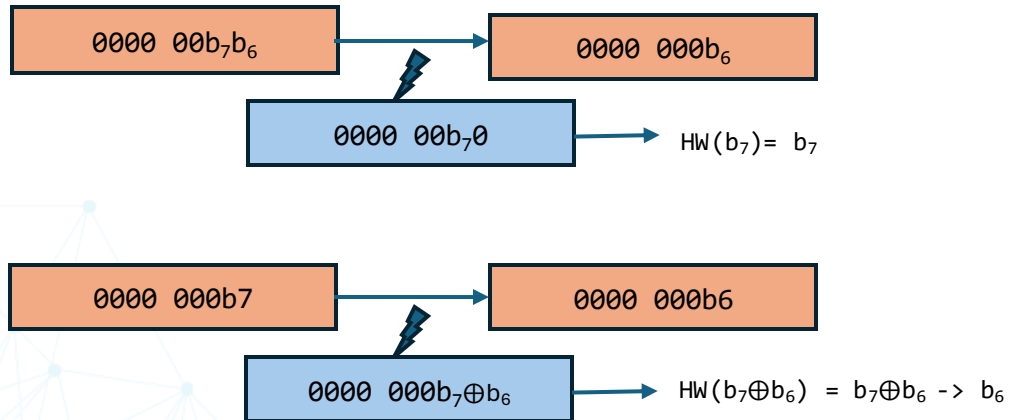
src>>0 b ₇ b ₆ b ₅ b ₄ b ₃ b ₂ b ₁ b ₀	&1 0000 000b ₀
src>>1 0b ₇ b ₆ b ₅ b ₄ b ₃ b ₂ b ₁	&1 0000 000b ₁
src>>2 00b ₇ b ₆ b ₅ b ₄ b ₃ b ₂	&1 0000 000b ₂
src>>3 000b ₇ b ₆ b ₅ b ₄ b ₃	&1 0000 000b ₃
src>>4 0000 b ₇ b ₆ b ₅ b ₄	&1 0000 000b ₄
src>>5 0000 0b ₇ b ₆ b ₅	&1 0000 000b ₅
src>>6 0000 00b ₇ b ₆	&1 0000 000b ₆
src>>7 0000 000b ₇	&1 0000 000b ₇

Side Channel Attacks – Instruction Set



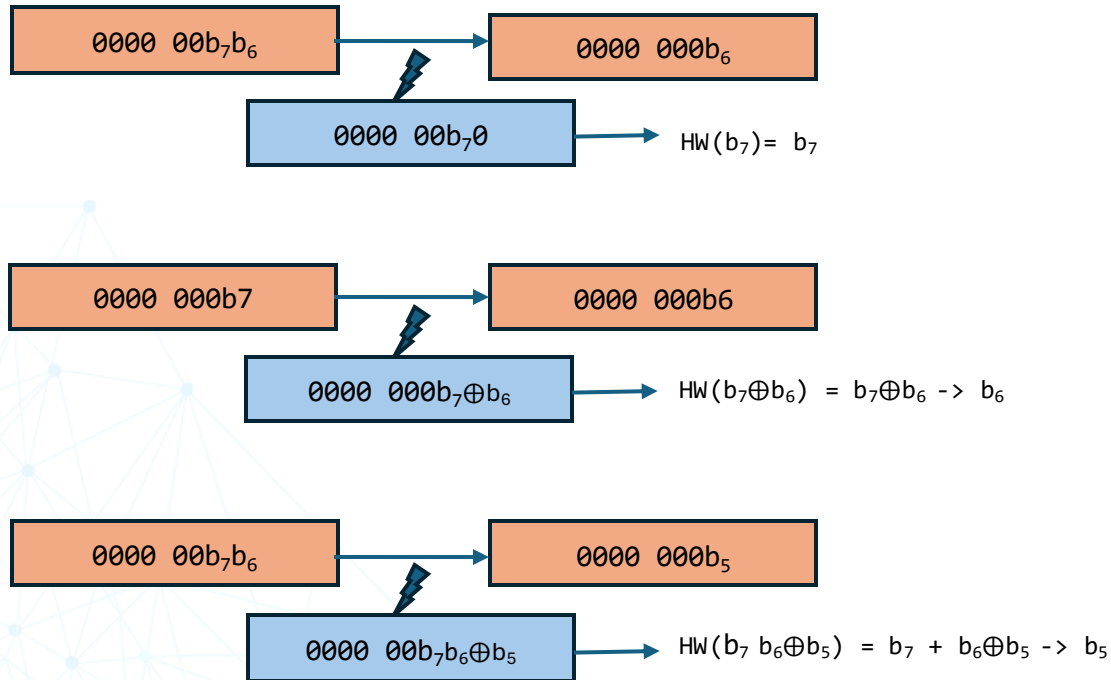
src>>0 b ₇ b ₆ b ₅ b ₄ b ₃ b ₂ b ₁ b ₀	&1 0000 000b ₀
src>>1 0b ₇ b ₆ b ₅ b ₄ b ₃ b ₂ b ₁	&1 0000 000b ₁
src>>2 00b ₇ b ₆ b ₅ b ₄ b ₃ b ₂	&1 0000 000b ₂
src>>3 000b ₇ b ₆ b ₅ b ₄ b ₃	&1 0000 000b ₃
src>>4 0000 b ₇ b ₆ b ₅ b ₄	&1 0000 000b ₄
src>>5 0000 0b ₇ b ₆ b ₅	&1 0000 000b ₅
src>>6 0000 00b ₇ b ₆	&1 0000 000b ₆
src>>7 0000 000b ₇	&1 0000 000b ₇

Side Channel Attacks – Instruction Set



$src \gg 0$	$b_7b_6b_5b_4\ b_3b_2b_1b_0$	$\&1$	$0000\ 000b_0$
$src \gg 1$	$0b_7b_6b_5\ b_4b_3b_2b_1$	$\&1$	$0000\ 000b_1$
$src \gg 2$	$00b_7b_6\ b_5b_4b_3b_2$	$\&1$	$0000\ 000b_2$
$src \gg 3$	$000b_7\ b_6b_5b_4b_3$	$\&1$	$0000\ 000b_3$
$src \gg 4$	$0000\ b_7b_6b_5b_4$	$\&1$	$0000\ 000b_4$
$src \gg 5$	$0000\ 0b_7b_6b_5$	$\&1$	$0000\ 000b_5$
$src \gg 6$	$0000\ 00b_7b_6$	$\&1$	$0000\ 000b_6$
$src \gg 7$	$0000\ 000b_7$	$\&1$	$0000\ 000b_7$

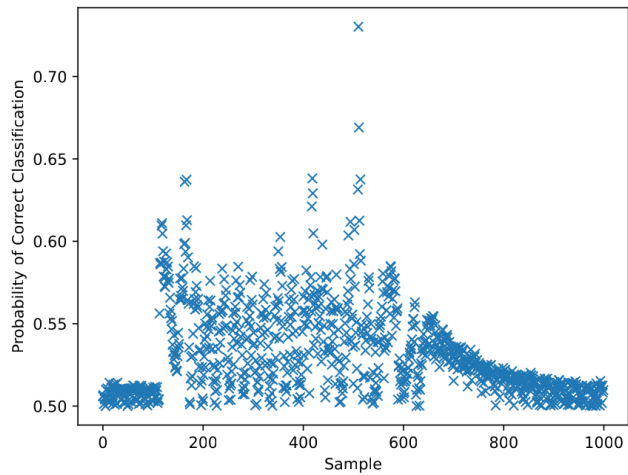
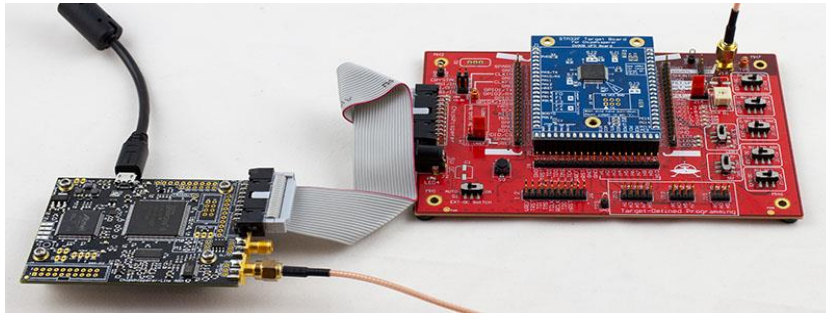
Side Channel Attacks – Instruction Set



Repeat to find all bits

$src \gg 0$ $b_7b_6b_5b_4\ b_3b_2b_1b_0$	$\&1$ $0000\ 000b_0$
$src \gg 1$ $0b_7b_6b_5\ b_4b_3b_2b_1$	$\&1$ $0000\ 000b_1$
$src \gg 2$ $00b_7b_6\ b_5b_4b_3b_2$	$\&1$ $0000\ 000b_2$
$src \gg 3$ $000b_7\ b_6b_5b_4b_3$	$\&1$ $0000\ 000b_3$
$src \gg 4$ $0000\ b_7b_6b_5b_4$	$\&1$ $0000\ 000b_4$
$src \gg 5$ $0000\ 0b_7b_6b_5$	$\&1$ $0000\ 000b_5$
$src \gg 6$ $0000\ 00b_7b_6$	$\&1$ $0000\ 000b_6$
$src \gg 7$ $0000\ 000b_7$	$\&1$ $0000\ 000b_7$

Side Channel Attacks – Micro Architecture



(b) Probability of correct classification of bit_{10} for each sample

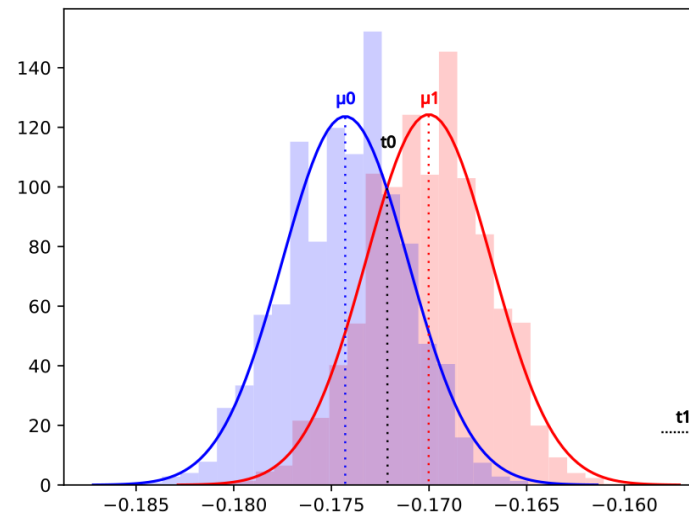
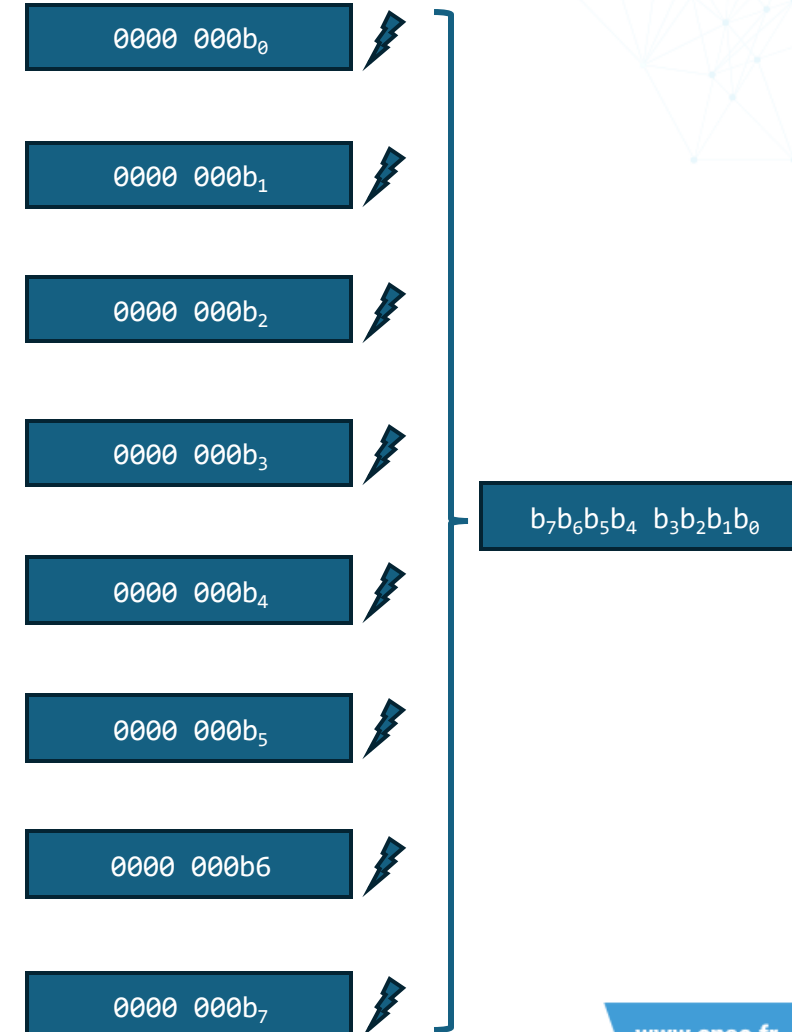


Fig. 7: Samples distributions for 0 and 1 on the POI of bit_0



Side Channel Attacks – Micro Architecture

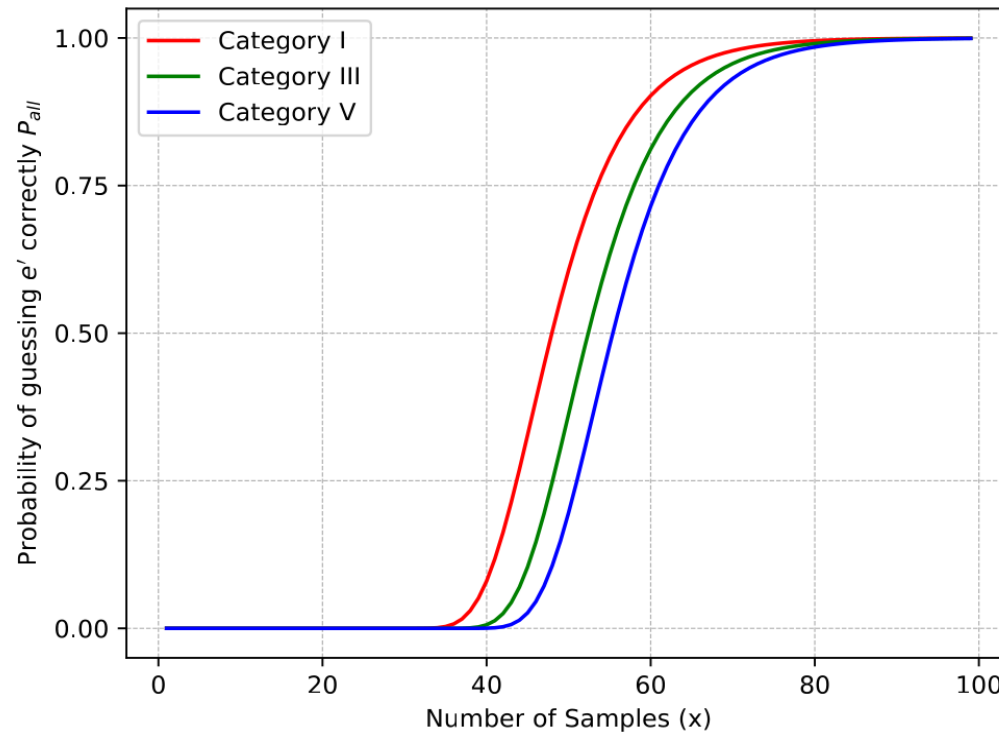
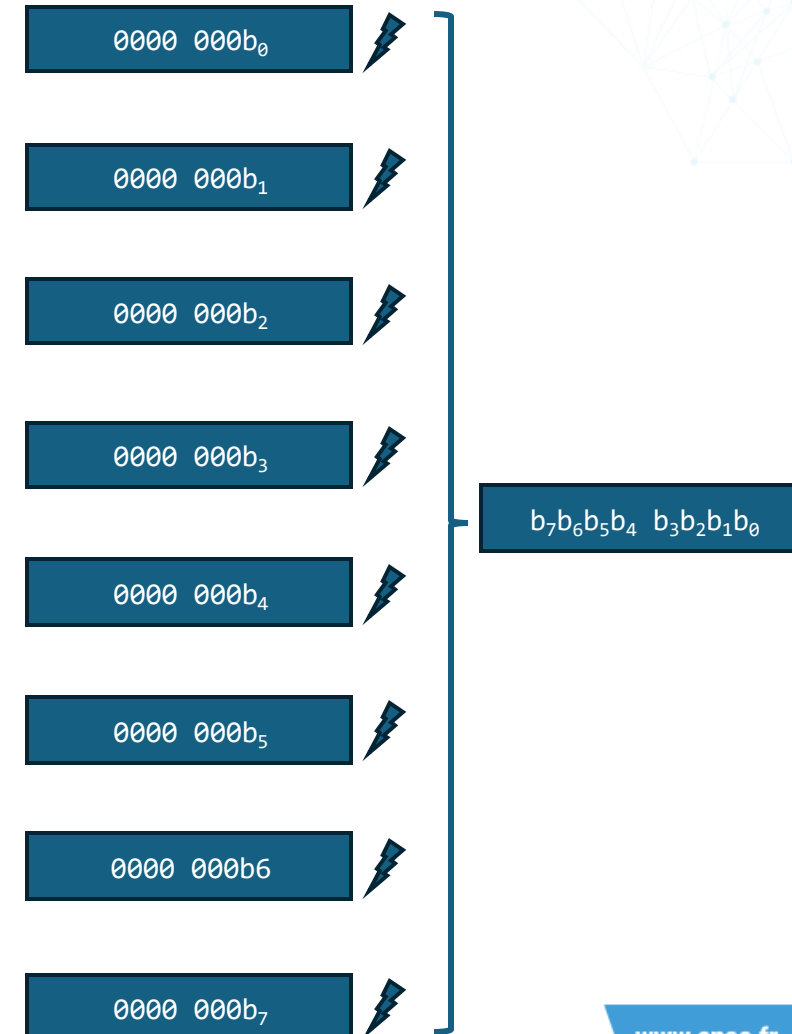


Fig. 9: Probabilities that $\mathbf{v} - \mathbf{u} \cdot \mathbf{y}$ is guessed correctly according to the number of traces and security category.





Attacks on HQC decryption

Some possible countermeasures

1. Change the decoder to a Minimum Distance Decoder to **avoid** `expand_and_sum` (5.3% slowdown)



Attacks on HQC decryption

Some possible countermeasures

1. Change the decoder to a Minimum Distance Decoder to **avoid** `expand_and_sum` (5.3% slowdown)
2. **Add noise** to the input (0.14% slowdown)

Decrypt:

$$m' = \text{Decode}(\mathbf{v} - \mathbf{u} \cdot \mathbf{y})$$

Replace by

$$\mathbf{e}'' \leftarrow R_{\omega'', \text{seed}(\mathbf{u}, \mathbf{v})}$$

$$m' = \text{Decode}(\mathbf{v} - \mathbf{u} \cdot \mathbf{y} + \mathbf{e}'')$$

Leakage still happens, but too many possible $\mathbf{e}''$, no way to bruteforce $\mathbf{y}$

Has an impact on DFR, but negligible

Attacks on HQC decryption

Some possible countermeasures

1. Change the decoder to a Minimum Distance Decoder to **avoid** `expand_and_sum` (5.3% slowdown)
2. Add **noise** to the input (0.14% slowdown)
3. **Mask the codeword** (1.14% slowdown)

Decrypt:

$$m' = \text{Decode}(\mathbf{v} - \mathbf{u} \cdot \mathbf{y})$$

Replace by

$$m'' \leftarrow \{0,1\}^k$$

$$m' = \text{Decode}(\mathbf{v} - \mathbf{u} \cdot \mathbf{y} + m''G) \oplus m''$$

Leakage still happens but computing $\mathbf{u}^{-1}(\mathbf{v} - (\mathbf{v} - \mathbf{u} \cdot \mathbf{y} + m''G))$ does not return $\mathbf{y}$

Conclusion

1. Why is HQC hard to improve ?

- Interactions between many variables which cause N to increase
- Based on state-of-the-art Error Correcting Codes

2. How do we propose to bypass these limitations to improve HQC ?

- Move some of the weight of r_1 and r_2 to e (at a cost)
- Still the residual noise and N are lower
- Also, a trade-off is to increase N to decrease CT, Encrypt and Decrypt

3. Why improving HQC can sometimes be dangerous ?

- The most optimized decryption leaks the key
- Countermeasures exist but requires slowdown

Adapting Cryptographic Tools to a Post-Quantum Context

Or how to improve HQC

Cyrius Nugier

cyrius.nugier@enac.fr

ENAC/SINA/TELECOM/ReSCo

