

Post-Quantum Cryptography #1

Prof. Claude Crépeau
McGill University

Why post-Quantum ?

Why post-Quantum ?



Why post- Quantum ?



Why post-Quantum?

- RSA



Why post-Quantum?

- RSA

- ElGamal



Why post-Quantum?

- RSA
- ElGamal
- Goldwasser-Micali



Why post-Quantum ?

- RSA
- ElGamal
- Goldwasser-Micali
- Blum-Goldwasser



Why post-Quantum ?



- RSA
- ElGamal
- Goldwasser-Micali
- Blum-Goldwasser
- DSS

Why post-Quantum ?



- RSA
- ElGamal
- Goldwasser-Micali
- Blum-Goldwasser
- DSS
- Paillier

Post-Quantum Cryptography

- Finite Fields based cryptography
 - Codes
 - Multi-variate Polynomials
- Integers based cryptography
 - Approximate Integer GCD
 - Lattices

Finite Fields based cryptography

PART 1

Finite Fields



- $(F, +, \times)$
- $+$: 0, $-a$, commutativity, associativity
- $\times$: 1, a^{-1} , commutativity, associativity
- distributivity of $\times$ over $+$.
- $\mathbb{F}_p = (\{0, 1, 2, \dots, p-1\}, + \bmod p, \times \bmod p)$ for primes p
- $\mathbb{F}_q = (\{0, 1, 2, \dots, p-1\}^m, +^m \bmod p, \times \bmod P \bmod p)$ for prime powers $q=p^m$, where P is an irreducible polynomial $(\bmod p)$ of degree $m-1$



1.5 Prime numbers

If we want a random prime (Maple `rand`, `isprime`) of a given size, we use the following theorem to estimate the number of integers we must try before finding a prime. Let $\pi(n) = \#\{a : 0 < a \leq n \text{ and } a \text{ is prime}\}$.

Theorem 1.9* $\lim_{n \rightarrow \infty} \frac{\pi(n) \log n}{n} = 1$

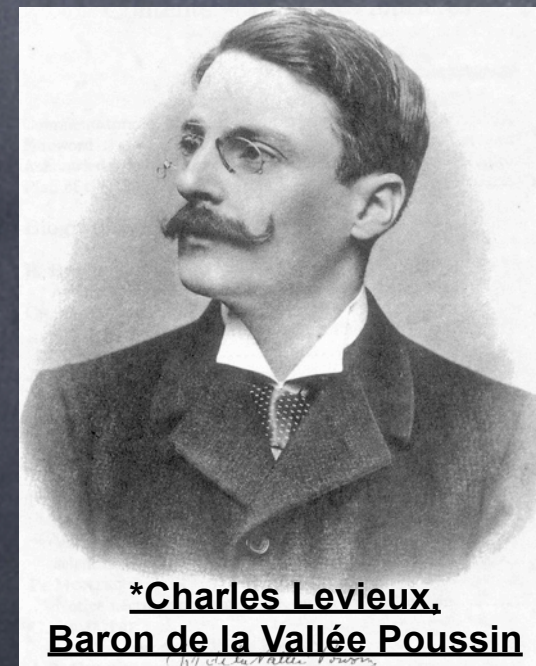
To decide whether a number n is prime or not we rely on Miller-Rabin's probabilistic algorithm. This algorithm introduces the notion of “pseudo-primality” base a . Miller defined this test as an extension of Fermat's test. If the Extended Riemann Hypothesis is true then it is sufficient to use the test with small values of a to decide whether a number n is prime or composite. However the ERH is not proven and we use the test in a probabilistic fashion as suggested by Rabin.



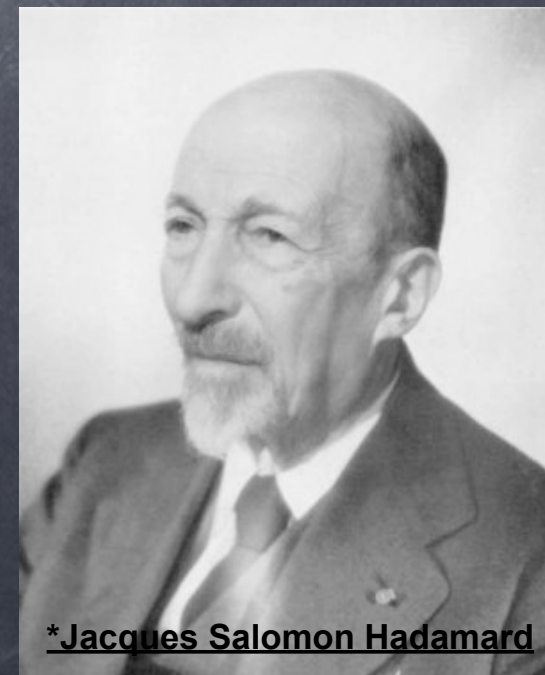
Gary L. Miller



Michael O. Rabin



*Charles Leveux,
Baron de la Vallée Poussin



*Jacques Salomon Hadamard



Pierre de Fermat



1.4.2 Fermat-Euler

Theorem 1.3 (Fermat) *Let p be a prime number and a be an integer not a multiple of p , then*

$$a^{p-1} \equiv 1 \pmod{p}.$$

Theorem 1.4 (Euler) *Let p be a prime number and a be an integer, then*

$$a^{(p-1)/2} \equiv \left(\frac{a}{p}\right) \pmod{p}.$$

Theorem 1.5 (Euler) *Let n be an integer and a another integer such that $\gcd(a, n) = 1$, then*

$$a^{\phi(n)} \equiv 1 \pmod{n}.$$

Algorithm 1.5 ($Pseudo(a, n)$)

1: IF $\gcd(a, n) \neq 1$ THEN RETURN “composite”,
2: Let t be an odd number and s a positive integer such that $n - 1 = t2^s$
3: $x \leftarrow a^t \bmod n$, $y \leftarrow n - 1$,
4: FOR $i \leftarrow 0$ TO s
5: IF $x = 1$ AND $y = n - 1$ THEN RETURN “pseudo”
6: $y \leftarrow x$, $x \leftarrow x^2 \bmod n$,
7: ENDFOR
8: RETURN “composite”.



It is easy to show that if n is prime, then $Pseudo(a, n)$ returns “pseudo” for all a , $0 < a < n$. Rabin showed that if n is composite, then $pseudo(a, n)$ returns “composite” for at least $3n/4$ of the values of a , $0 < a < n$.

Theorem 1.10

$$\#\{a : Pseudo(a, n) = \text{“pseudo”}\} \begin{cases} = \phi(n) & = n - 1 & \text{if } n \text{ is prime} \\ \leq \phi(n)/4 & \leq (n - 1)/4 & \text{if } n \text{ is composite.} \end{cases}$$

2.2.1 Irreducible Polynomials

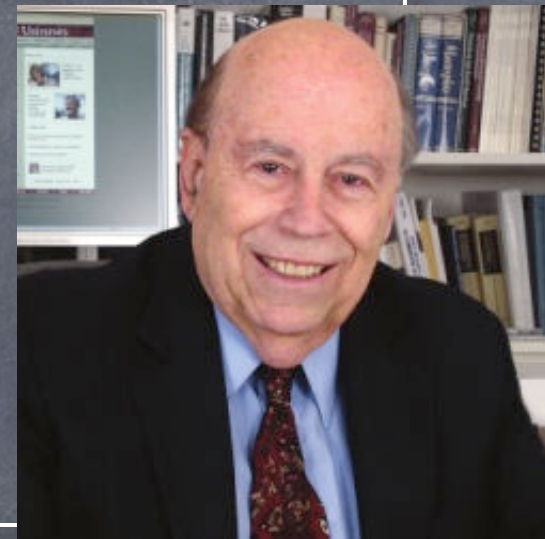
A polynomial $g(x)$ is *irreducible* (Maple `irreduc`) if it is not the product of two polynomials $h(x), k(x)$ of lower degrees. We use the following theorem to find irreducible polynomials.

Theorem 2.6 *Let $l_1, l_2, \dots, l_k$ be the prime factors of n and $m_i = n/l_i$ for $1 \leq i \leq k$. A polynomial $g(x)$ of degree n is irreducible over $\mathcal{F}_p$ iff*

- $g(x) \mid x^{p^n} - x$
- $\gcd(g(x), x^{p^{m_i}} - x) = 1$ for $1 \leq i \leq k$

Algorithm 2.3 (Rabin $Irr(p, n)$)

- 1: let $l_1, l_2, \dots, l_k$ be the prime factors of n and $m_i = n/l_i$ for $1 \leq i \leq k$,
- 2: **REPEAT**
- 3: pick a random polynomial $h(x)$ of degree $n - 1$ over $\mathcal{F}_p$, and set $g(x) \leftarrow x^n + h(x)$,
- 4: **UNTIL** $x^{p^n} \bmod g(x) = x$ and $\gcd(g(x), x^{p^{m_i}} \bmod g(x) - x) = 1$ for $1 \leq i \leq k$,
- 5: **RETURN** g .



We use the following theorem to estimate the number of polynomials we have to try on average before finding one that is irreducible.

Theorem 2.7 Let $m(n)$ be the number of irreducible polynomials $g(x)$ of degree n of the form $g(x) = x^n + h(x)$ where $h(x)$ is of degree $n - 1$. We have

$$\frac{p^n}{2n} \leq \frac{p^n - p^{n/2} \log n}{n} \leq m(n) \leq \frac{p^n}{n}.$$

2.1.1 Primitive Elements

In all finite fields $\mathcal{F}_q$ (and some groups in general) there exists a *primitive element*, that is an element g of the field such that $g^1, g^2, \dots, g^{q-1}$ enumerate all of the $q - 1$ non-zero elements of the field. We use the following theorem to find a primitive element over $\mathcal{F}_q$.

Theorem 2.1 *Let $l_1, l_2, \dots, l_k$ be the prime factors of $q-1$ and $m_i = (q-1)/l_i$ for $1 \leq i \leq k$. An element g is primitive over $\mathcal{F}_q$ if and only if*

- $g^{q-1} = 1$
- $g^{m_i} \neq 1$ for $1 \leq i \leq k$

Algorithm 2.1 (*Primitive*(q))

1: *Let $l_1, l_2, \dots, l_k$ be the prime factors of $q-1$ and $m_i = \frac{q-1}{l_i}$ for $1 \leq i \leq k$,*

2: REPEAT

3: *pick a random non-zero element g of $\mathcal{F}_q$,*

4: UNTIL $g^{m_i} \neq 1$ *for $1 \leq i \leq k$,*

5: RETURN g .

(Maple `primroot`, `G[PrimitiveElement]`)

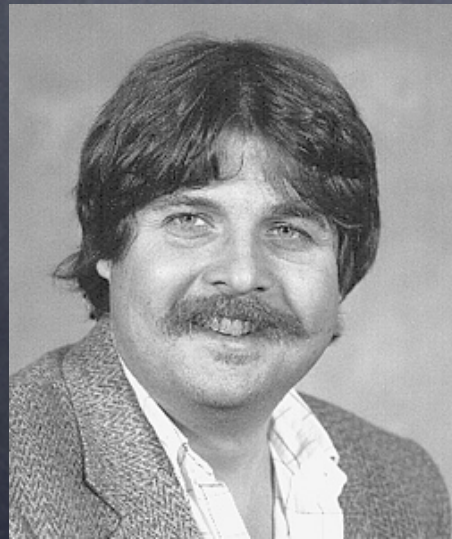
We use the following theorems to estimate the number of field elements we must try in order to find a random primitive element.

Theorem 2.2 $\#\{g : g \text{ is a primitive element of } \mathcal{F}_q\} = \phi(q-1).$

Theorem 2.3 $\liminf_{n \rightarrow \infty} \frac{\phi(n) \log \log n}{n} = e^{-\gamma} \approx 0.5614594836$

Example: 2 is a primitive element of $\mathcal{F}_5$ since $\{2, 2^2, 2^3, 2^4\} = \{2, 4, 3, 1\}.$

Factoring $q - 1$... The only efficient way we know to finding a primitive element in fields $\mathcal{F}_q$ is when the factorization of $q - 1$ is known. In general, it may be difficult to factor $q - 1$. However, if we are after a large field with a random number of elements, Eric Bach has devised an efficient probabilistic algorithm to generate random integers of a given size with known factorization. Recently, Adam Kalai has invented a somewhat slower algorithm that is much simpler. Suppose we randomly select r with its factorization using Bach's or Kalai's algorithm. We may check whether $r + 1$ is a prime or a prime power. In this case a finite field of $r + 1$ elements is obtained and a primitive element may be computed.



Eric Bach



Adam Kalai

<http://homes.cerias.purdue.edu/~ssw/cun/index.html>

Algorithm 2.2 (Kalai *randfact*(n))



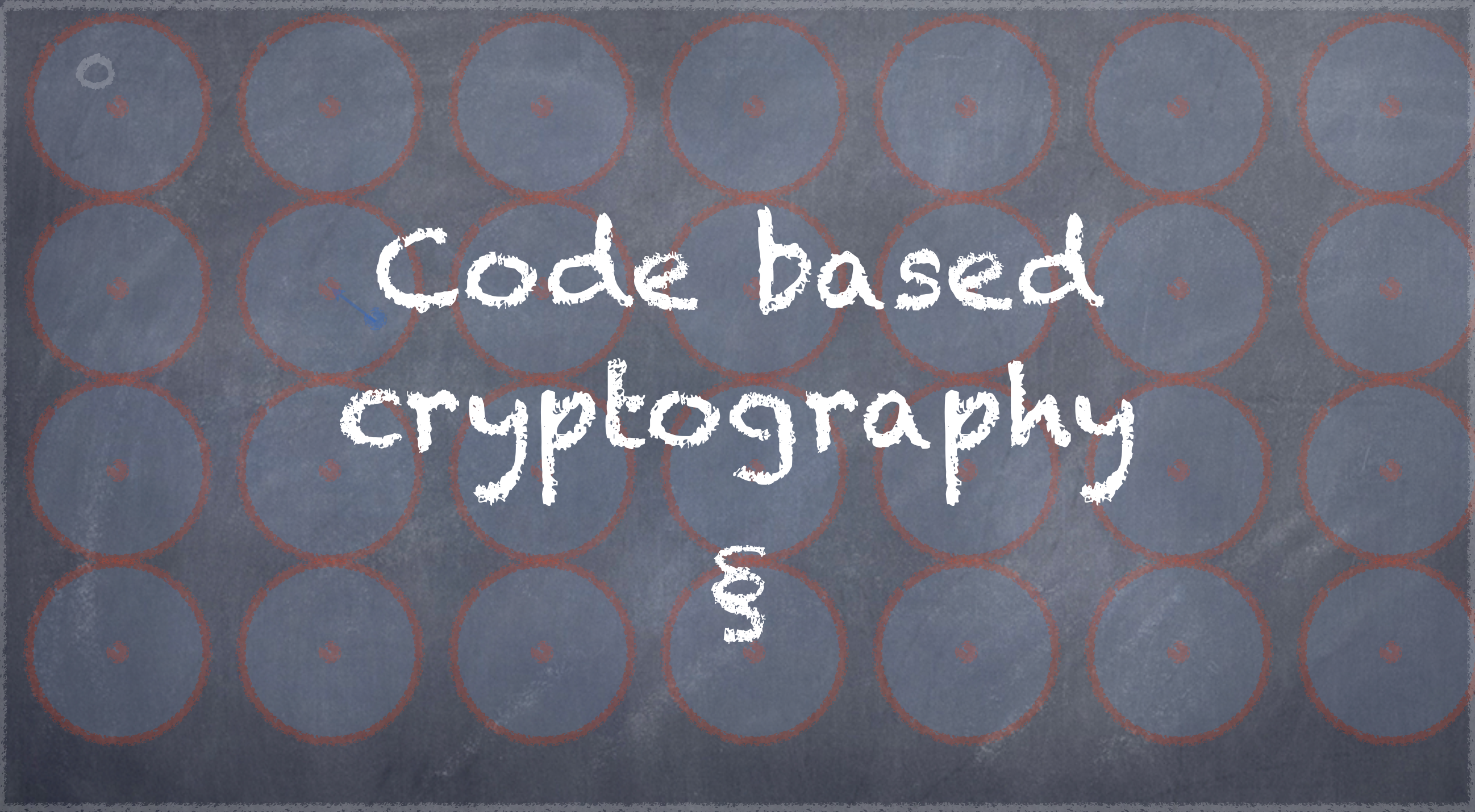
- 1: Generate a sequence $n = s_0 \geq s_1 \geq s_2 \geq \dots \geq s_\ell = 1$ by picking $s_{i+1} \in_R \{1, 2, \dots, s_i\}$, until reaching $s_\ell = 1$.
- 2: Let r be the product of the prime s_i 's, $1 \leq i \leq \ell$.
- 3: **IF** $r \leq n$ **THEN** with probability r/n **RETURN** $(r, \{\text{prime } s_i \text{'s}\})$.
- 4: Otherwise, **RESTART**.

Theorem 2.4 The probability of producing r at step 2 is M_n/r , where $M_n = \prod_{p \leq n} (1 - 1/p)$.

Thus by outputting r with probability r/n in step 3, each possible value is generated with equal probability $\frac{M_n}{r} \frac{r}{n} = \frac{M_n}{n}$. The overall probability that some small enough r is produced and chosen in step 3 is $\sum_{1 \leq r \leq n} \frac{M_n}{n} = M_n$.

Theorem 2.5 $\lim_{n \rightarrow \infty} M_n \log n = e^{-\gamma} \approx 0.5614594836$





code based cryptography

§

basic Coding theory

basic Coding theory

- an (N, m, d) code C is a set of cardinality $\#C = m$ of *codewords* of length N .

basic Coding theory

- an (N, m, d) code C is a set of cardinality $\#C = m$ of *codewords* of length N .
- The minimal distance d of such a code is the least value such that there exist $c, c' \in C$ with $d_H(c, c') = d$.

basic Coding theory

$\mathbb{F}_q^N$



basic Coding theory

$\mathbb{F}_q^N$

• No efficient encoding



basic Coding theory

$\mathbb{F}_q^N$

• No efficient encoding

• No efficient decoding



basic Coding theory

$\mathbb{F}_q^N$

- No efficient encoding

- No efficient decoding

- No efficient error detection



basic Coding theory

$\mathbb{F}_q^N$

- No efficient encoding

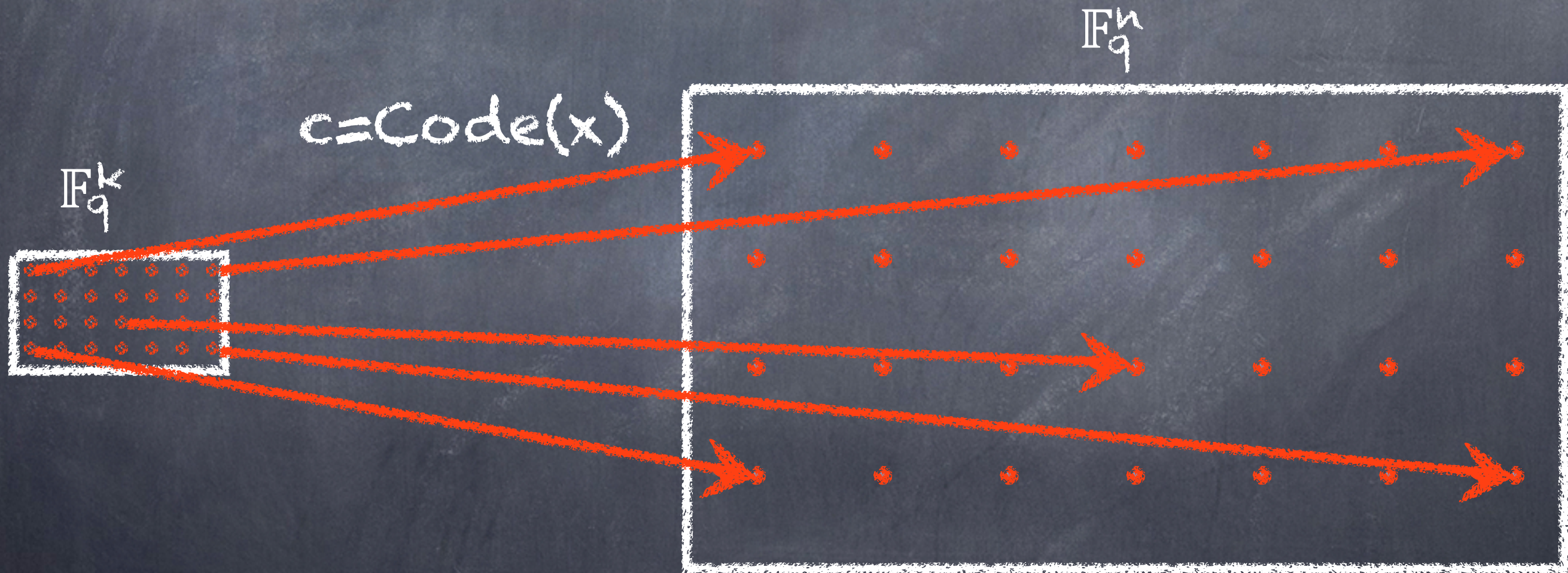
- No efficient decoding

- No efficient error detection

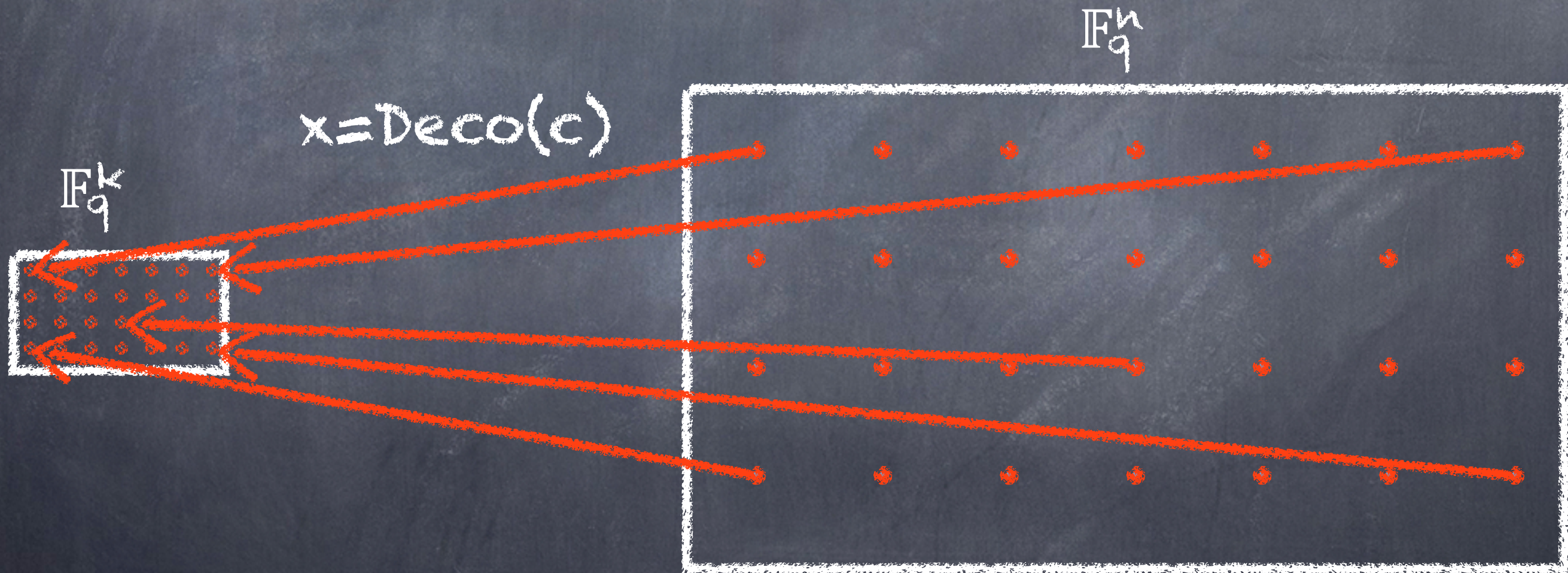
- No efficient error correction



coding vs decoding

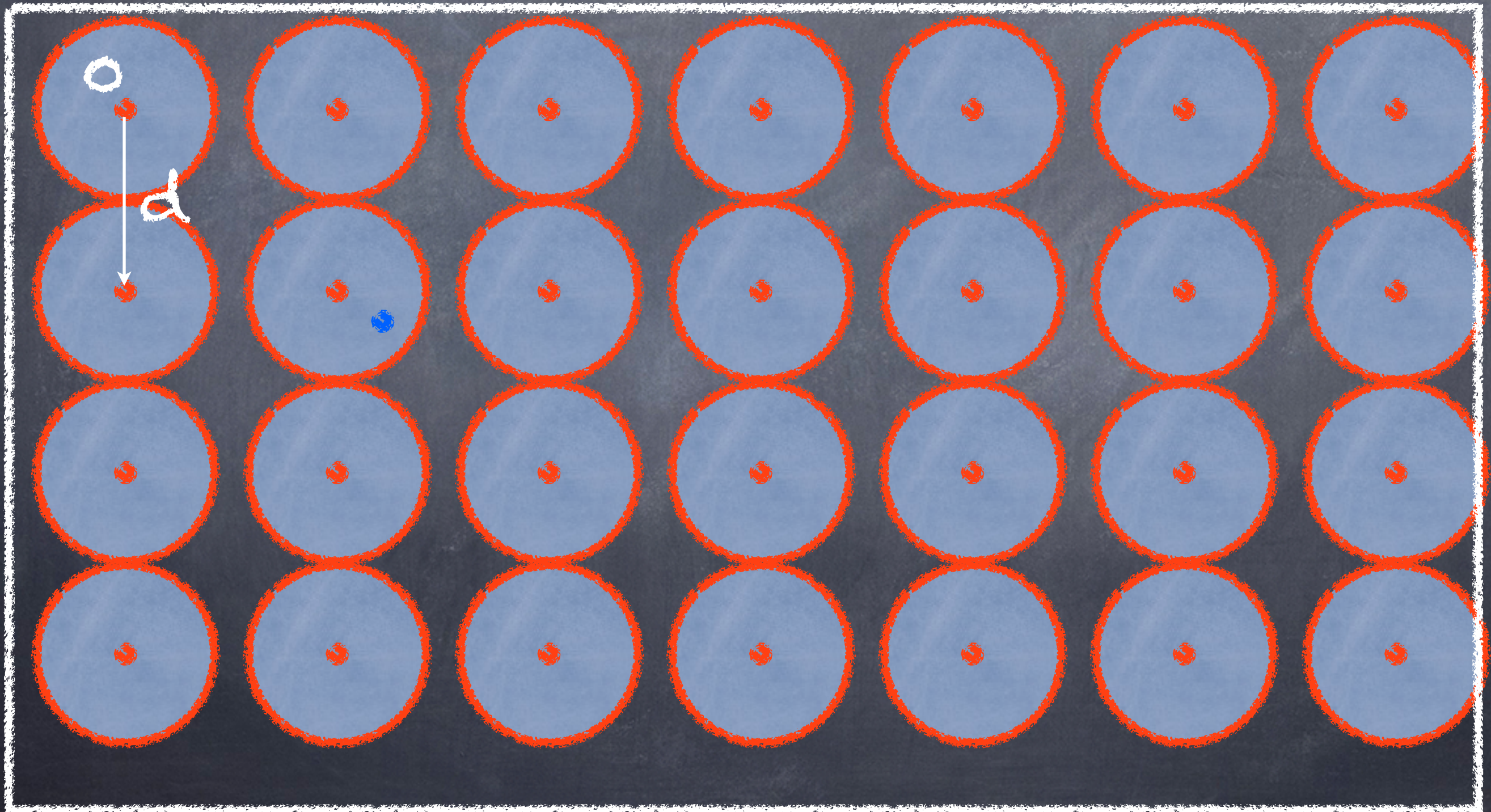


coding vs decoding



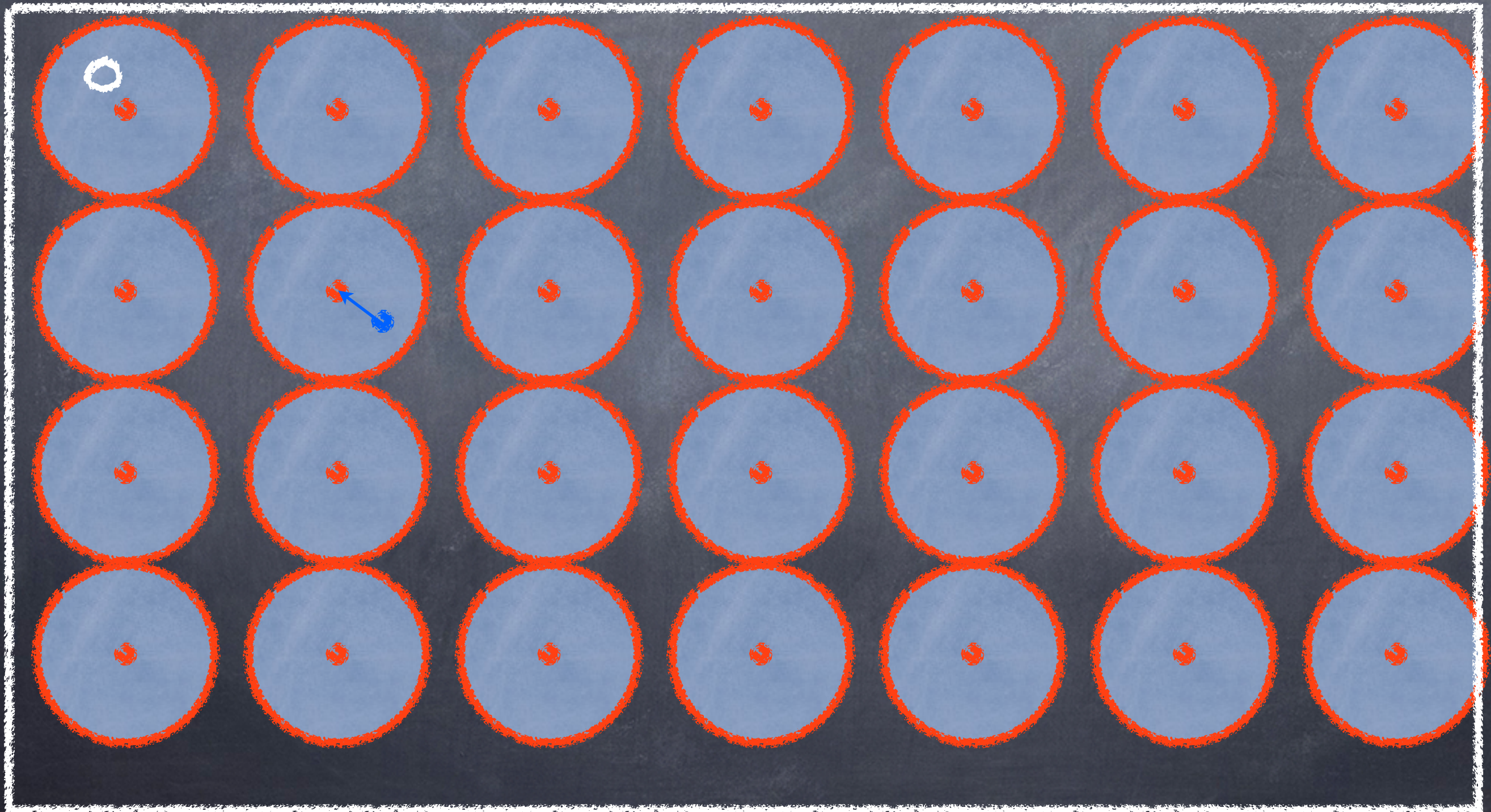
error detection vs error correction

$\mathbb{F}_q^n$



error detection vs error correction

$\mathbb{F}_q^n$



basic Linear Coding theory

basic Linear Coding theory

- an $[n, k, d]$ linear code C is a linear subspace of dimension k of *codewords* of length n .

basic Linear Coding theory

- an $[n, k, d]$ linear code C is a linear subspace of dimension k of *codewords* of length n .
- The minimal distance $d > 0$ of such a code is the least value such that there exists $c \in C \setminus \{0\}$ with $w_H(c) = d$.

basic Linear Coding theory

$\mathbb{F}_q^N$



basic Linear Coding theory

basic Linear Coding theory

- an $[n, k, d]$ linear code C is defined via either

basic Linear Coding theory

- an $[n, k, d]$ linear code C is defined via either
- an $k \times n$ generating matrix G s.t.
 $C = \text{Span}(G)$

basic Linear Coding theory

- an $[n, k, d]$ linear code C is defined via either
- an $k \times n$ generating matrix G s.t.
 $C = \text{Span}(G)$
- an $n \times (n-k)$ parity check matrix H s.t.
 $C = \text{Ker}(H)$.

basic Linear Coding theory

- an $[n, k, d]$ linear code C is defined via either
- an $k \times n$ generating matrix G s.t.
$$C = \{ xG \mid x \in \mathbb{F}_q^k \}$$
- an $n \times (n-k)$ parity check matrix H s.t.
$$C = \{ c \in \mathbb{F}_q^n \mid Hc = 0 \}.$$

basic Linear Coding theory

basic Linear Coding theory

- Let C be an $[n, k, d]$ linear code defined via an $k \times n$ generating matrix G or an $n \times (n-k)$ parity check matrix H and let w be $1 \leq w \leq n$

basic Linear Coding theory

- Let C be an $[n, k, d]$ linear code defined via an $k \times n$ generating matrix G or an $n \times (n-k)$ parity check matrix H and let w be $1 \leq w \leq n$
- Deciding existence of a codeword $c \in C$ of weight $\leq w$ is NP-complete

basic Linear Coding theory

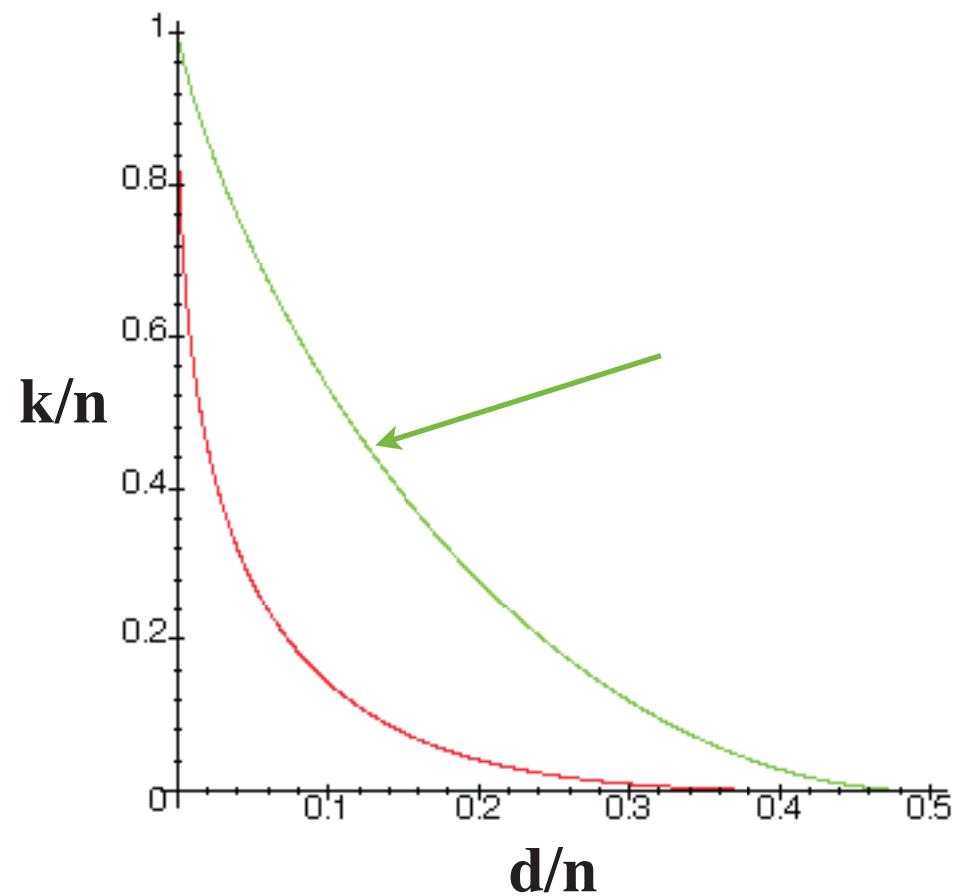
basic Linear Coding theory

- Let C be an $[n, k, d]$ linear code defined via an $k \times n$ generating matrix G or an $n \times (n-k)$ parity check matrix H and let w be $1 \leq w \leq n$

basic Linear Coding theory

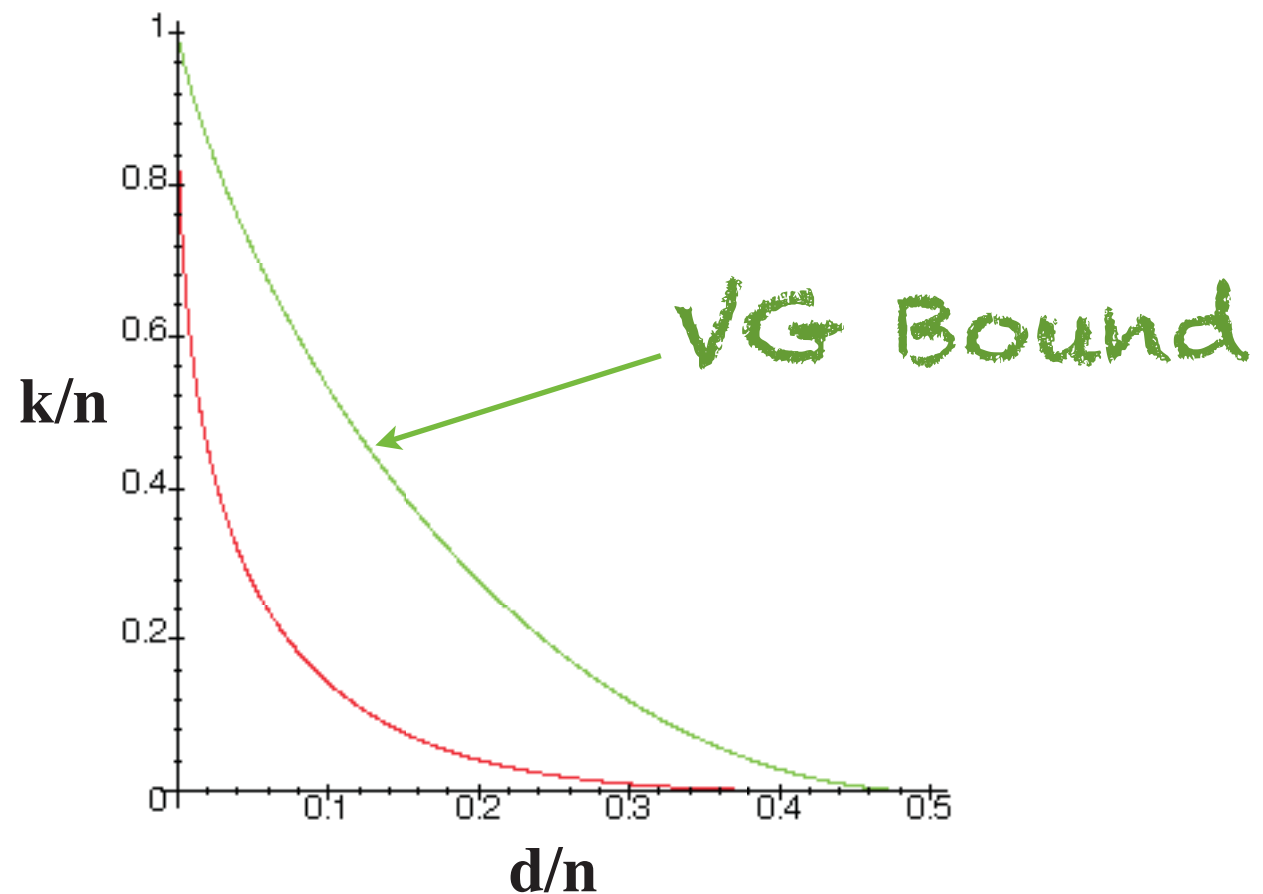
- Let C be an $[n, k, d]$ linear code defined via an $k \times n$ generating matrix G or an $n \times (n-k)$ parity check matrix H and let w be $1 \leq w \leq n$
- Determination of a codeword $c \in C$ of weight w is NP-hard

basic Linear Coding theory



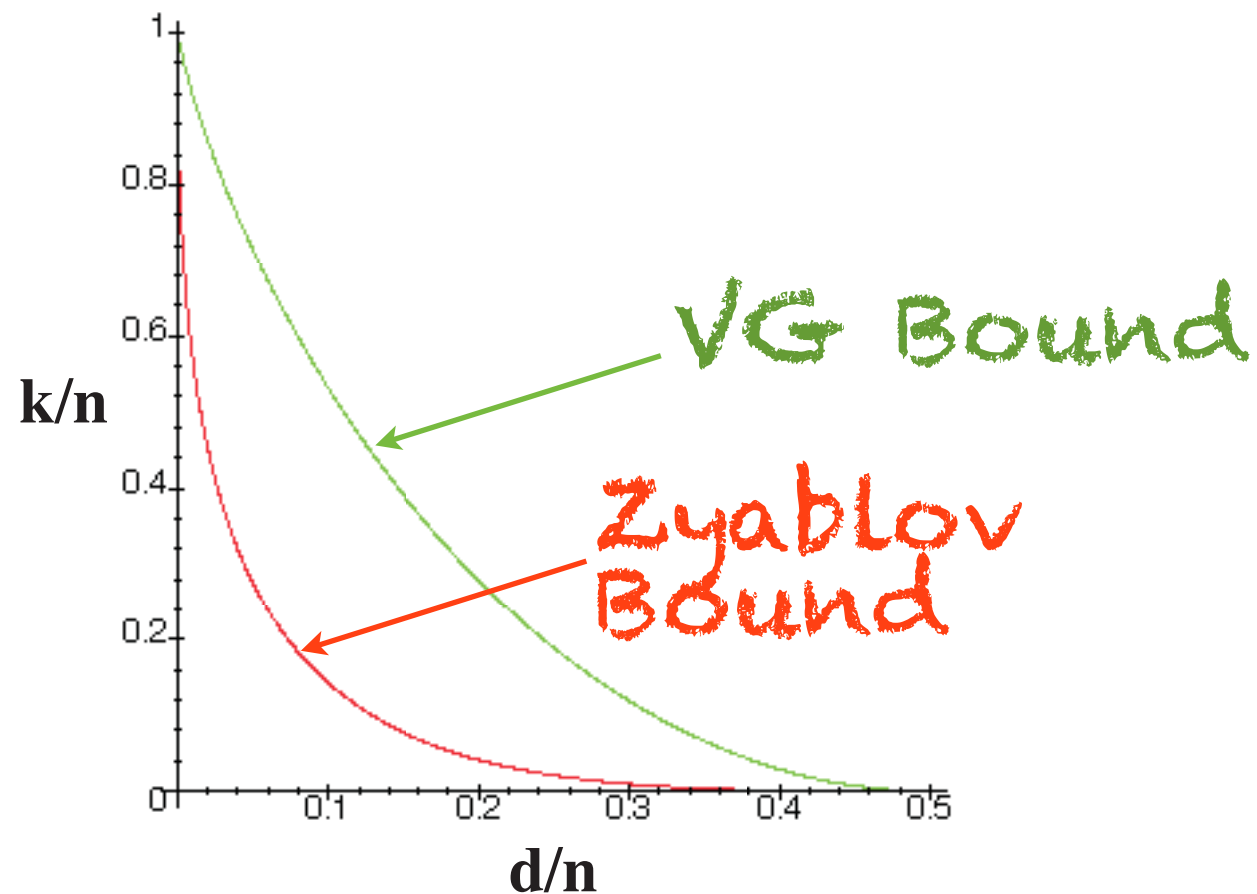
Varshamov-Gilbert Bound

basic Linear Coding theory



Varshamov-Gilbert Bound

basic Linear Coding theory



Varshamov-Gilbert Bound

basic Linear Coding theory

basic Linear Coding theory

- the DUAL to an $[n, k, d]$ linear code C is named $C^\perp$ & is the $[n, n-k, d^\perp]$ linear code defined by

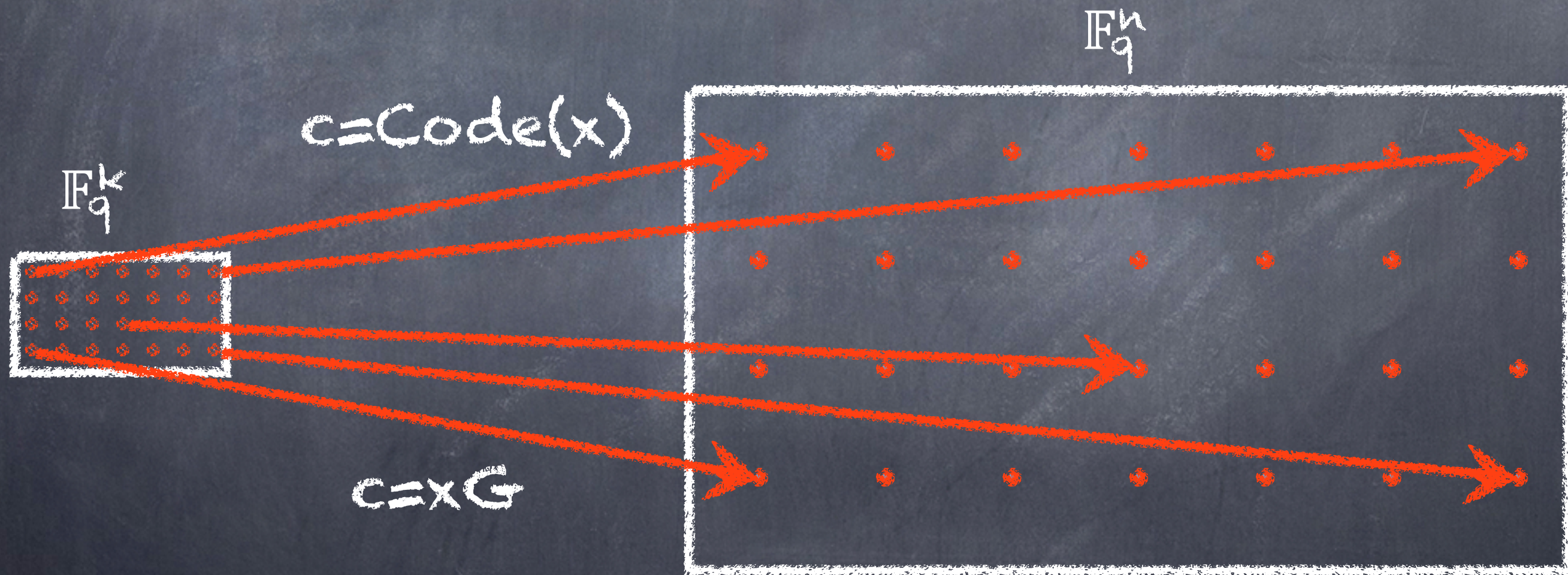
basic Linear Coding theory

- the DUAL to an $[n, k, d]$ linear code C is named $C^\perp$ & is the $[n, n-k, d^\perp]$ linear code defined by
- an $n \times (n-k)$ generating matrix H s.t.
 $C^\perp = \text{Span}(H)$ with $C = \text{Ker}(H)$,

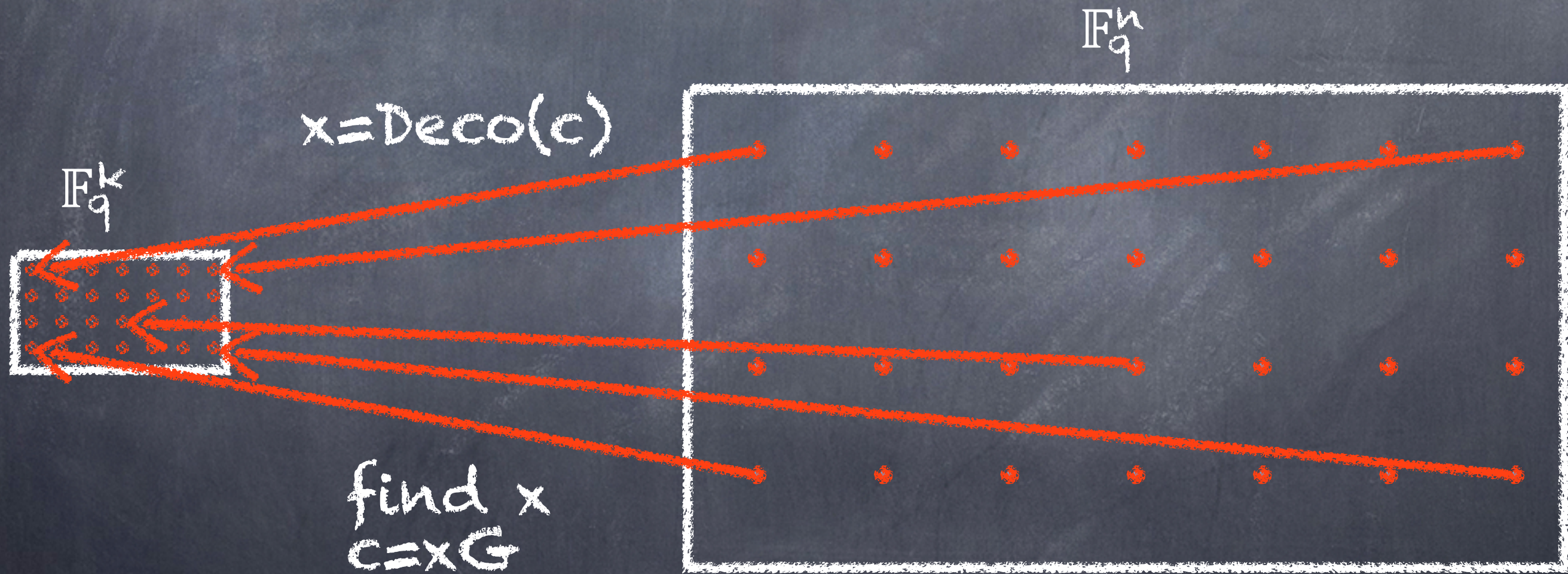
basic Linear Coding theory

- the DUAL to an $[n, k, d]$ linear code C is named $C^\perp$ & is the $[n, n-k, d^\perp]$ linear code defined by
- an $n \times (n-k)$ generating matrix H s.t.
 $C^\perp = \text{Span}(H)$ with $C = \text{Ker}(H)$,
- an $n \times k$ parity check matrix G s.t.
 $C^\perp = \text{Ker}(G)$ with $C = \text{Span}(G)$.

coding vs decoding

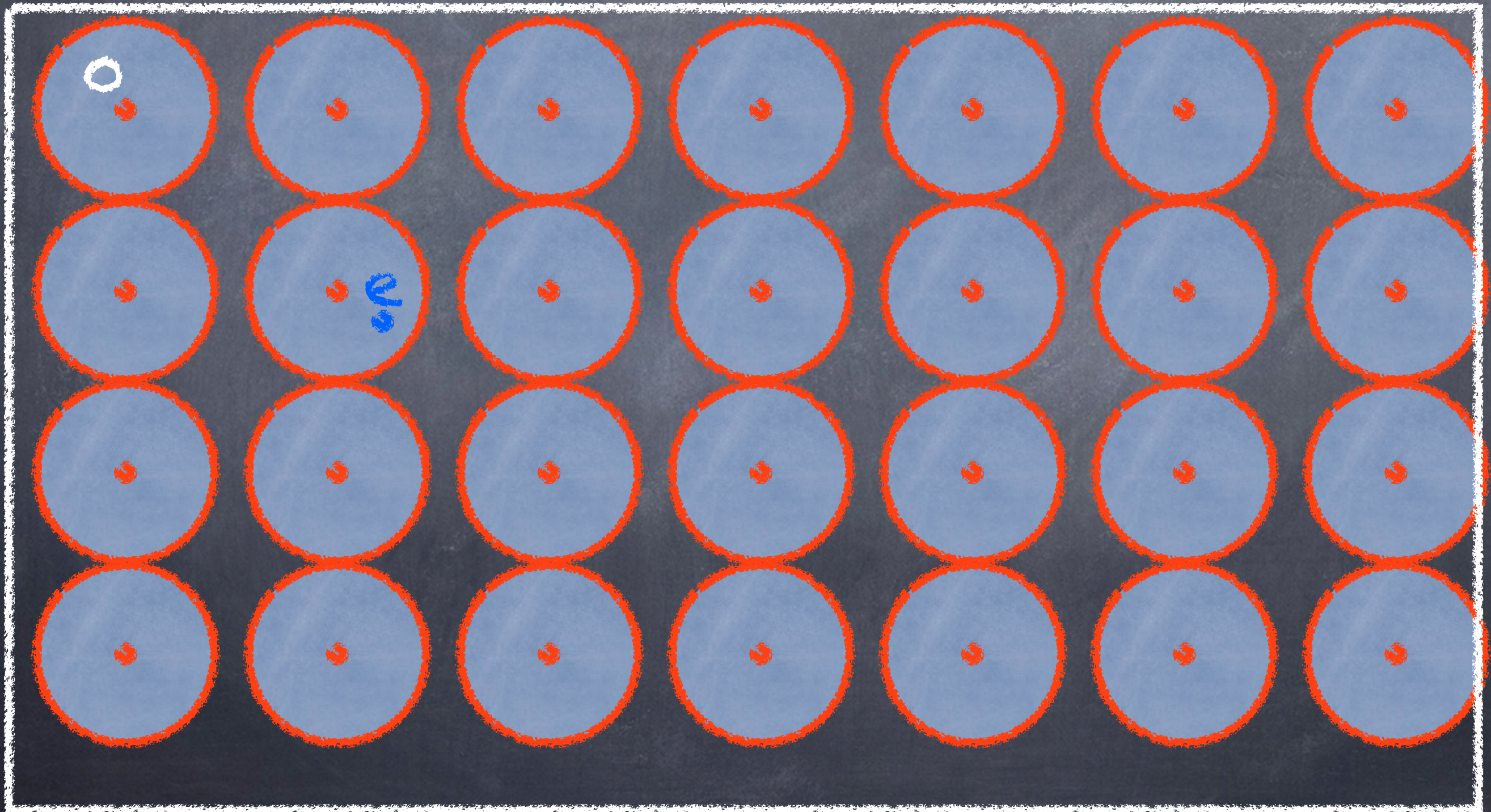


coding vs decoding



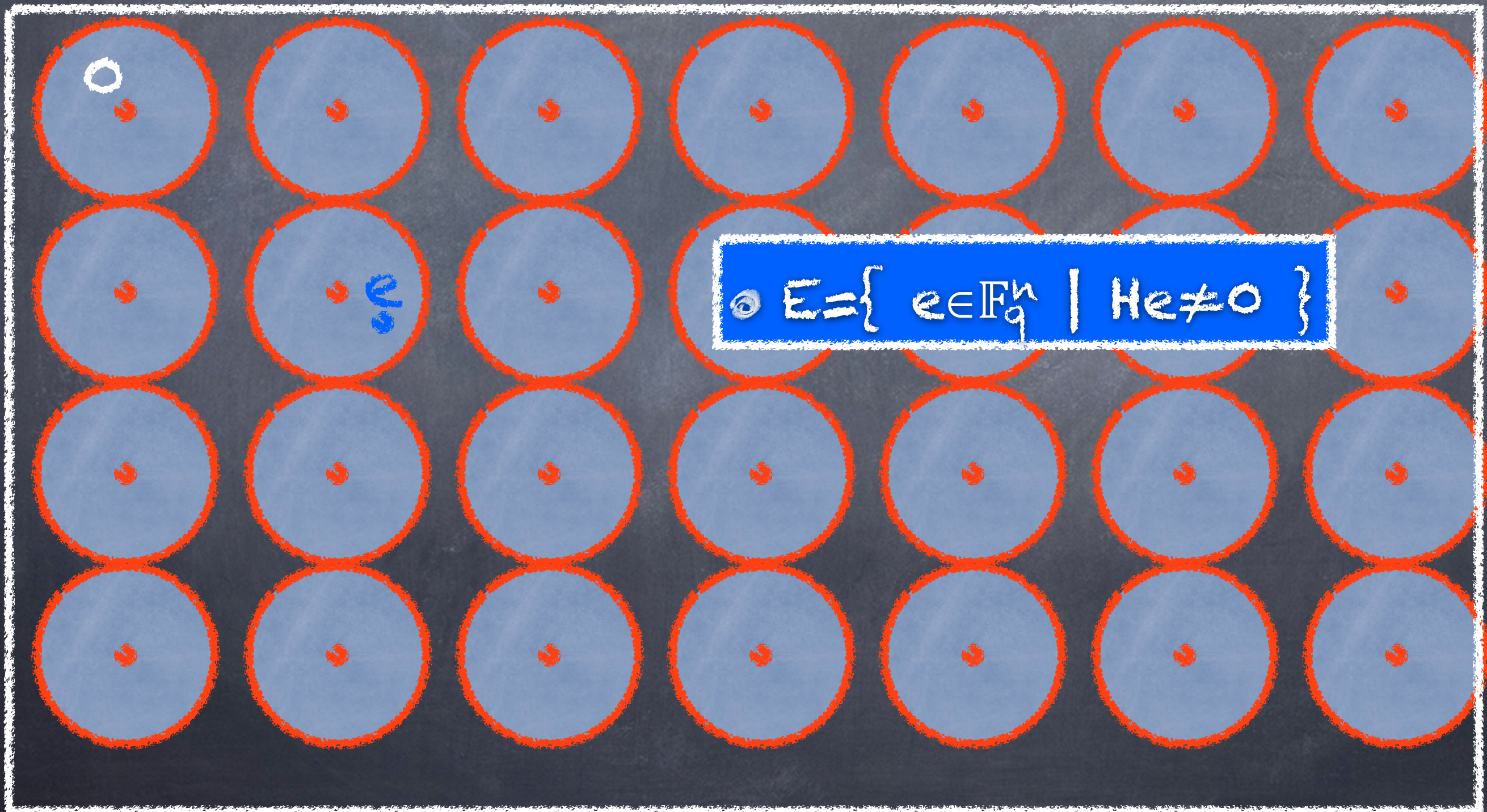
error detection vs error correction

$\mathbb{F}_q^n$



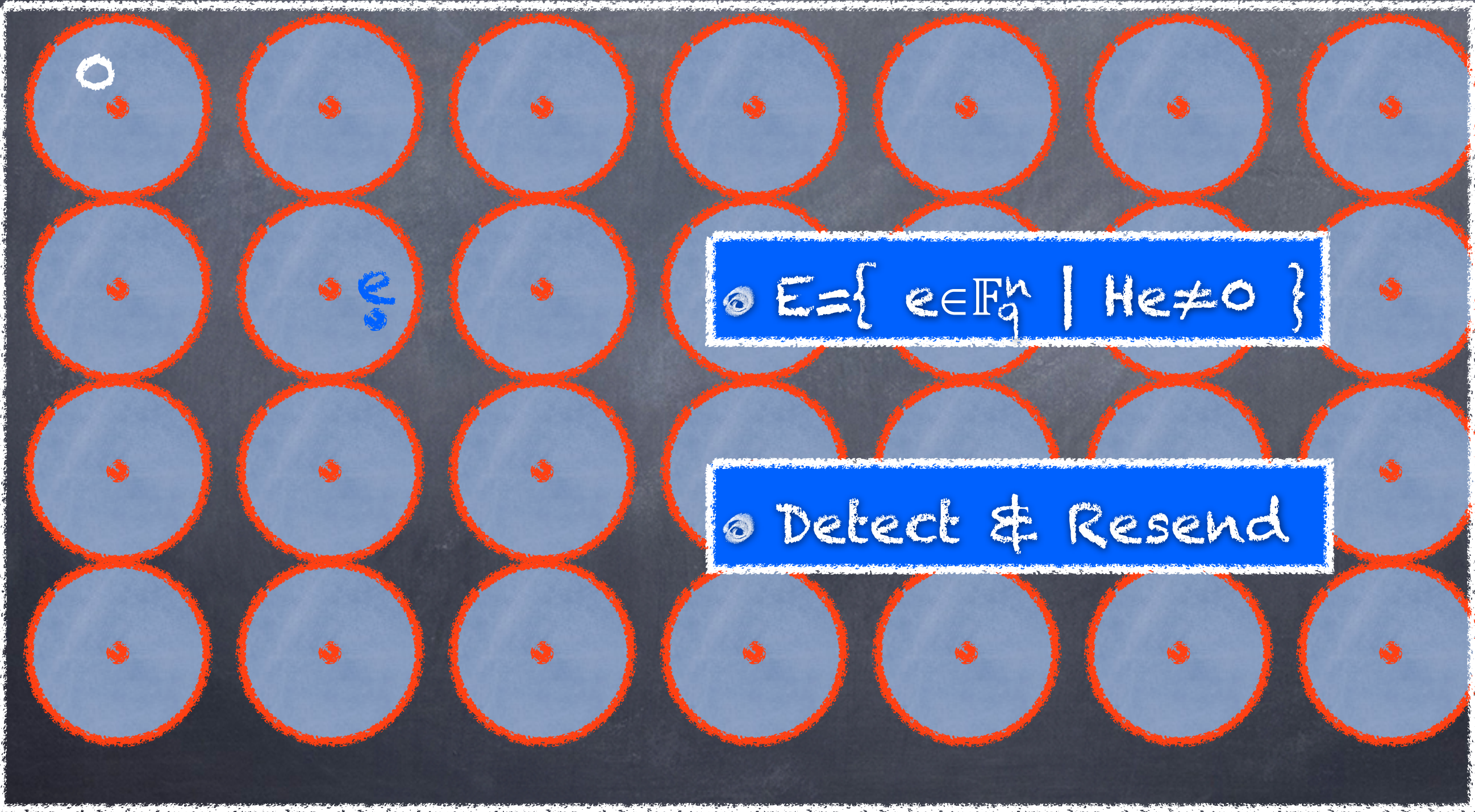
error detection vs error correction

$\mathbb{F}_q^n$



error detection vs error correction

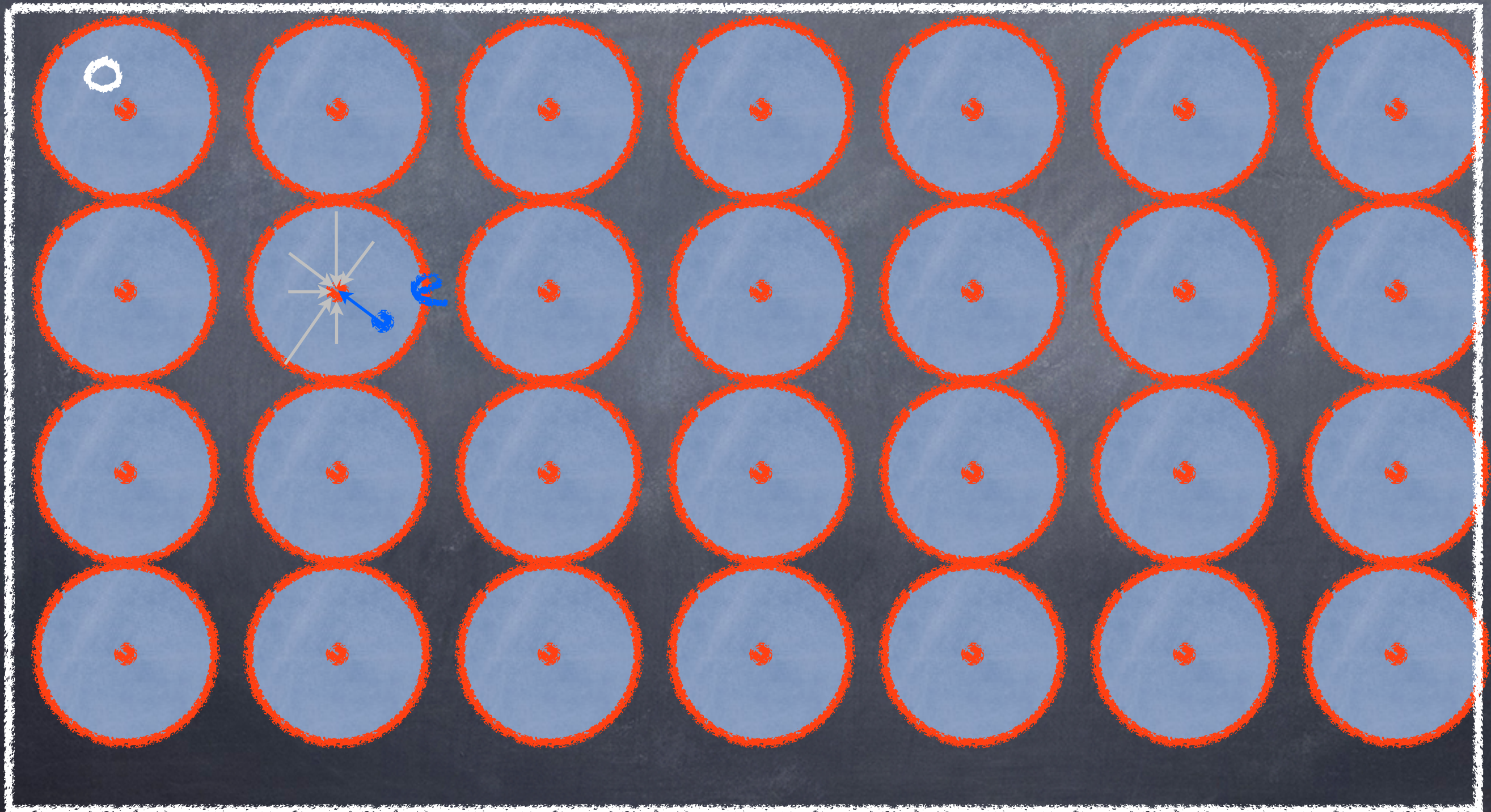
$\mathbb{F}_q^n$


$$E = \{ e \in \mathbb{F}_q^n \mid He \neq 0 \}$$

Detect & Resend

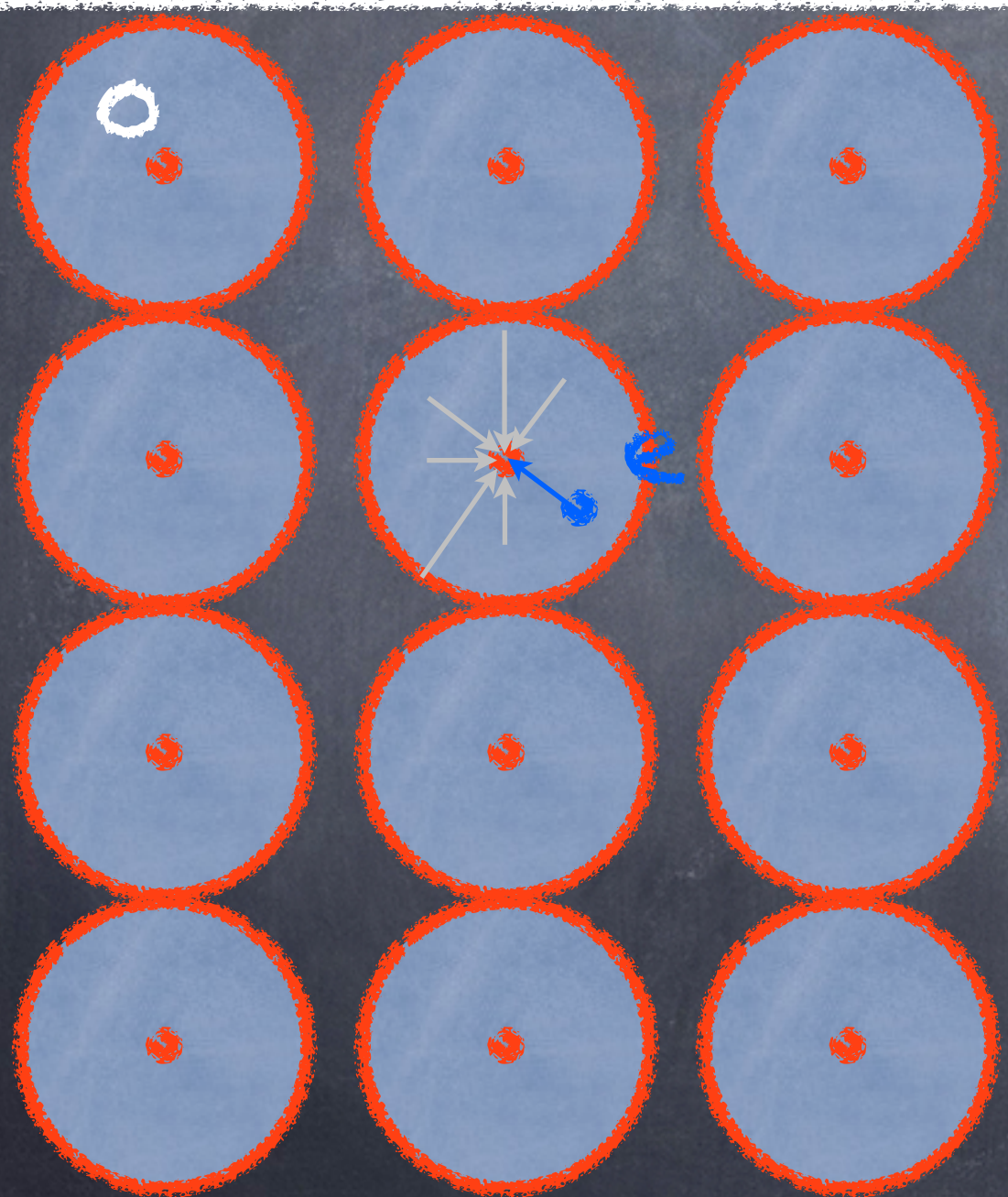
error detection vs error correction

$\mathbb{F}_q^n$



error detection vs error correction

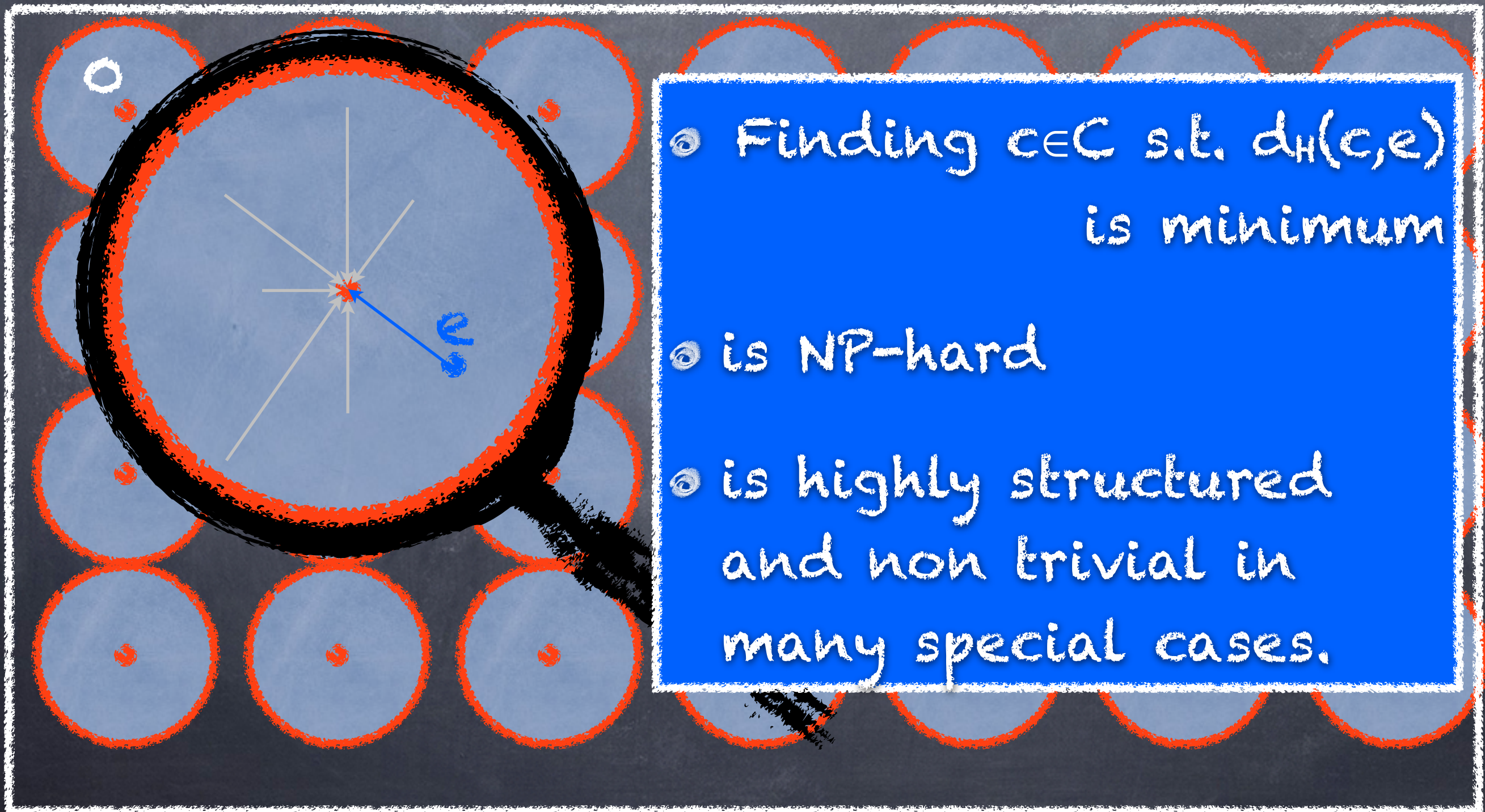
$\mathbb{F}_q^n$



- Finding $c \in C$ s.t. $d_H(c, e)$ is minimum
- is NP-hard
- is highly structured and non trivial in many special cases.

error detection vs error correction

$\mathbb{F}_q^n$



error detection vs error correction

$\mathbb{F}_q^n$

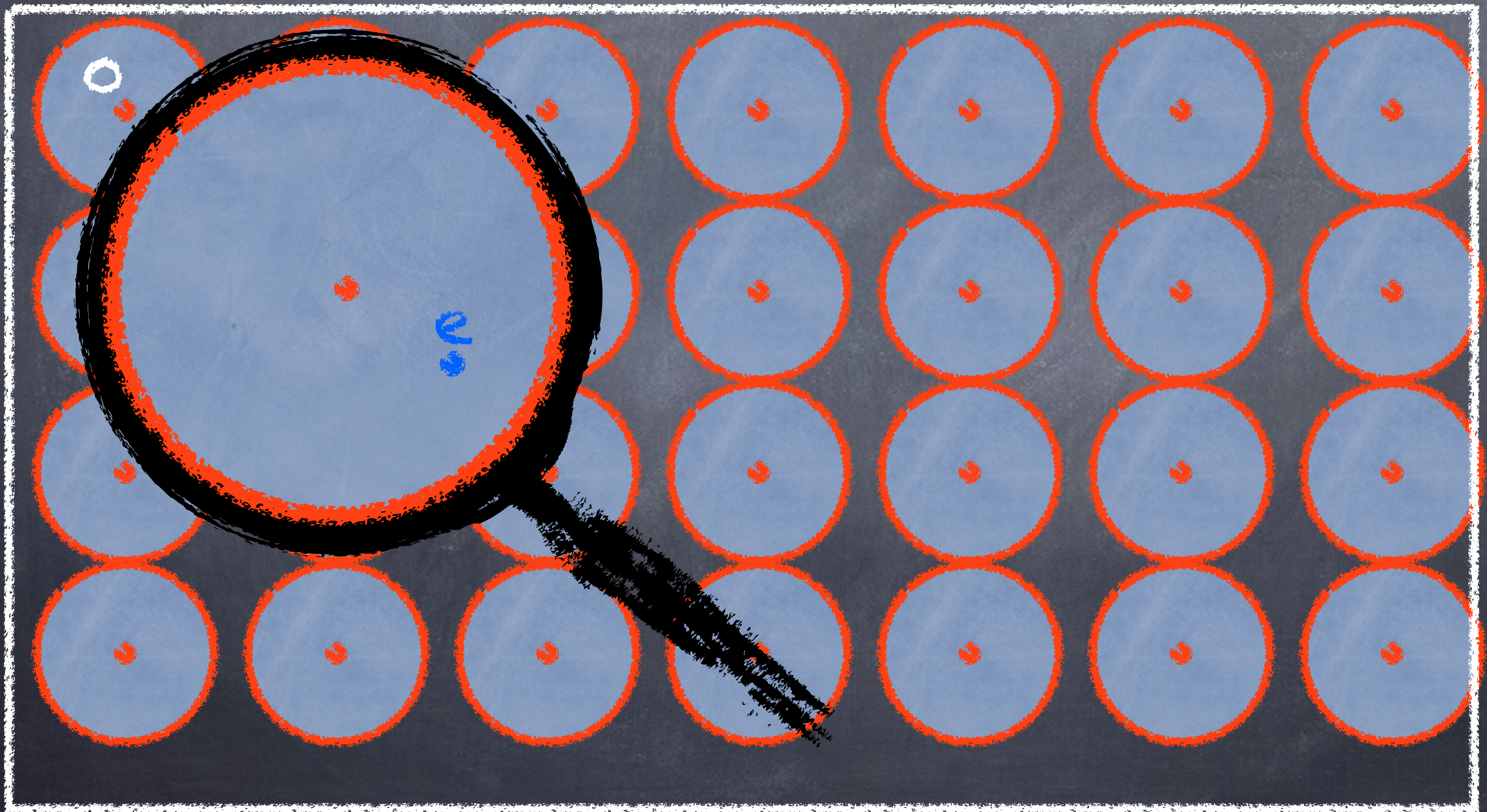
- Finding w s.t. $\text{wt}(w)$ is minimum & $Hw = He$
- is NP-hard
- is highly structured and non trivial in many special cases.

error detection vs error correction

- (inefficient correction procedure)
- To correct a word w upto $d-1/2$ errors applied to an $[n,k,d]$ codeword
- try all error patterns e upto $d-1/2$ errors
- check whether $w-e$ is in the code.

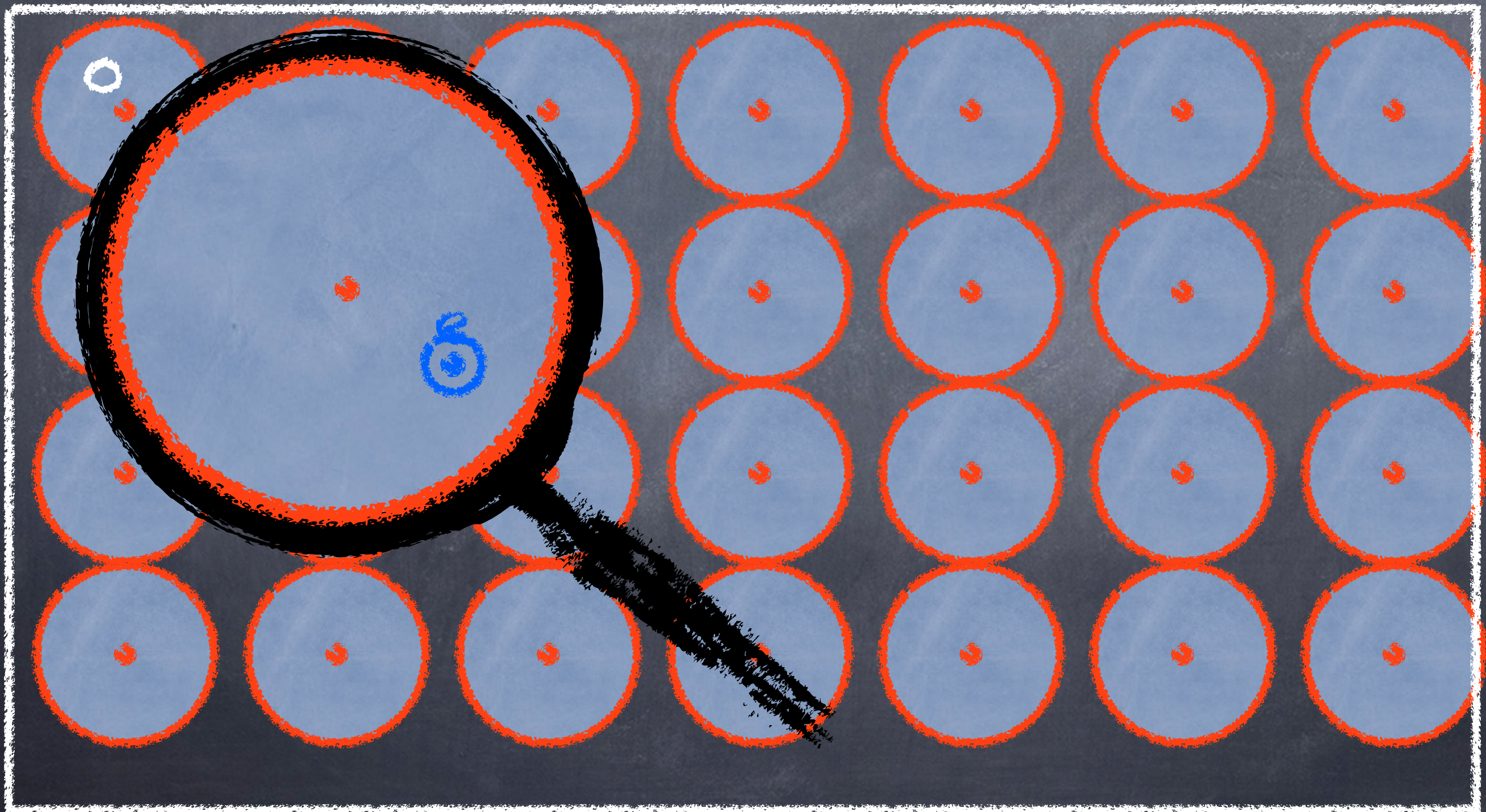
error detection vs error correction

$\mathbb{F}_q^n$



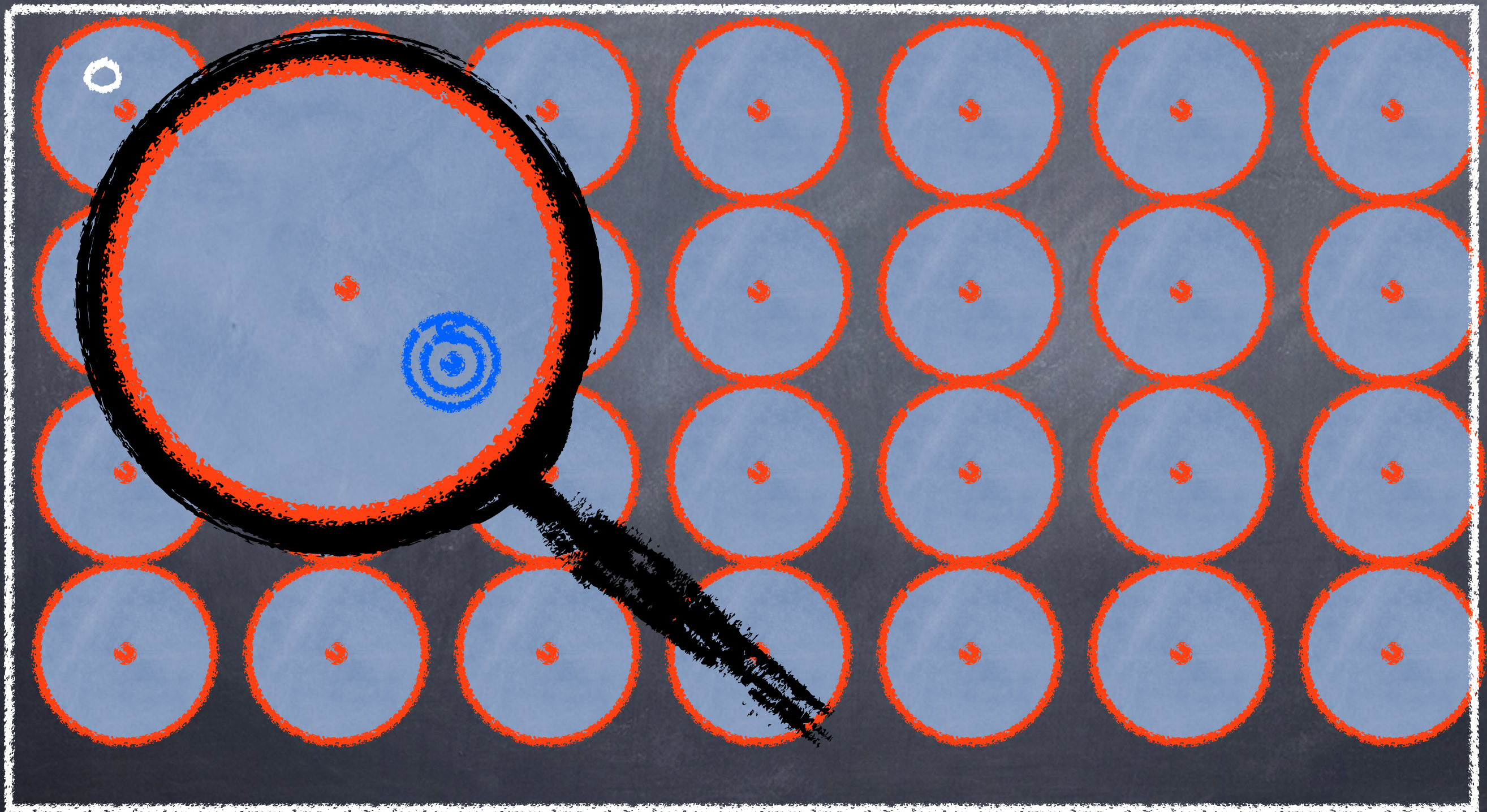
error detection vs error correction

$\mathbb{F}_q^n$



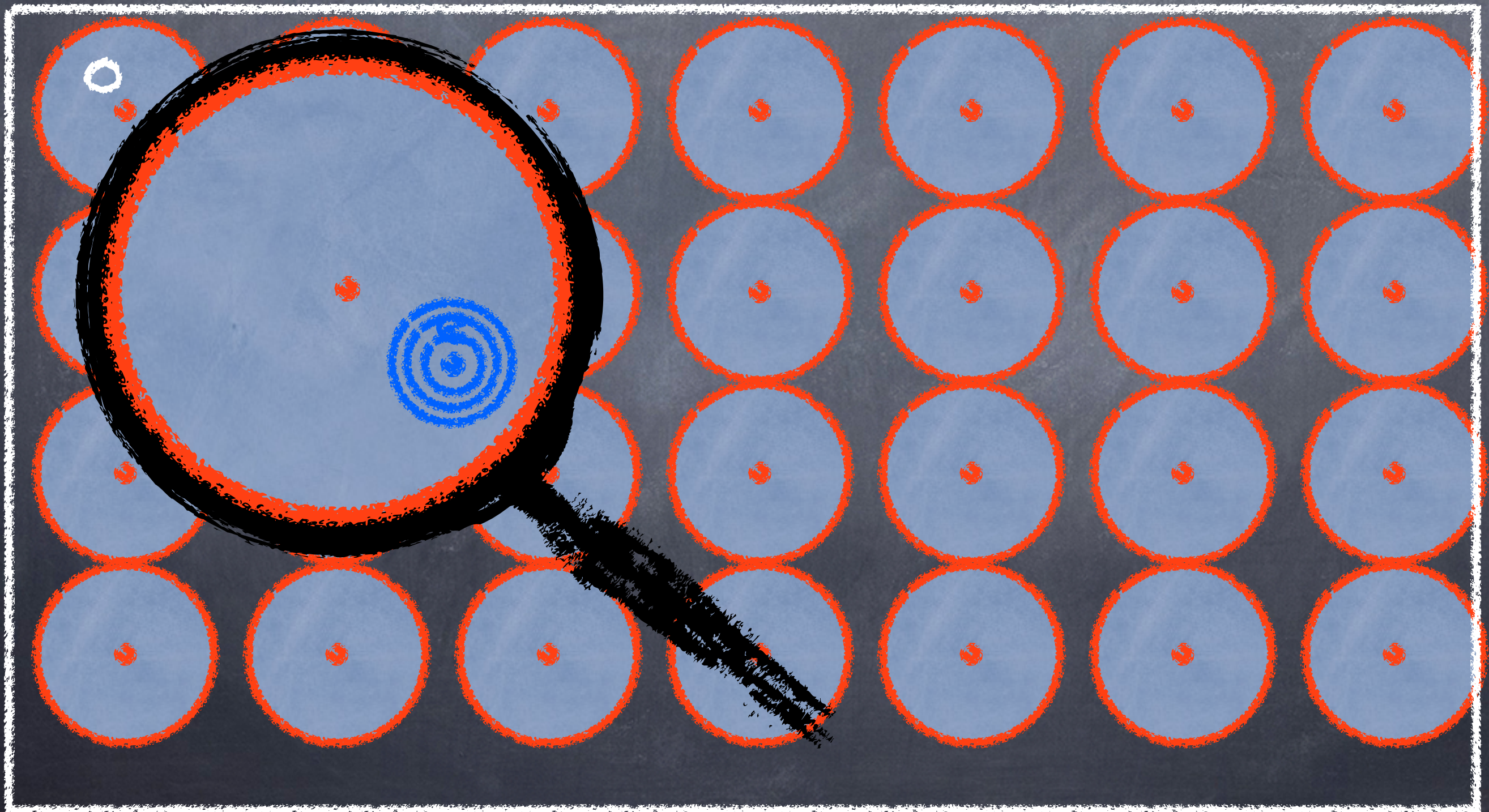
error detection vs error correction

$\mathbb{F}_q^n$



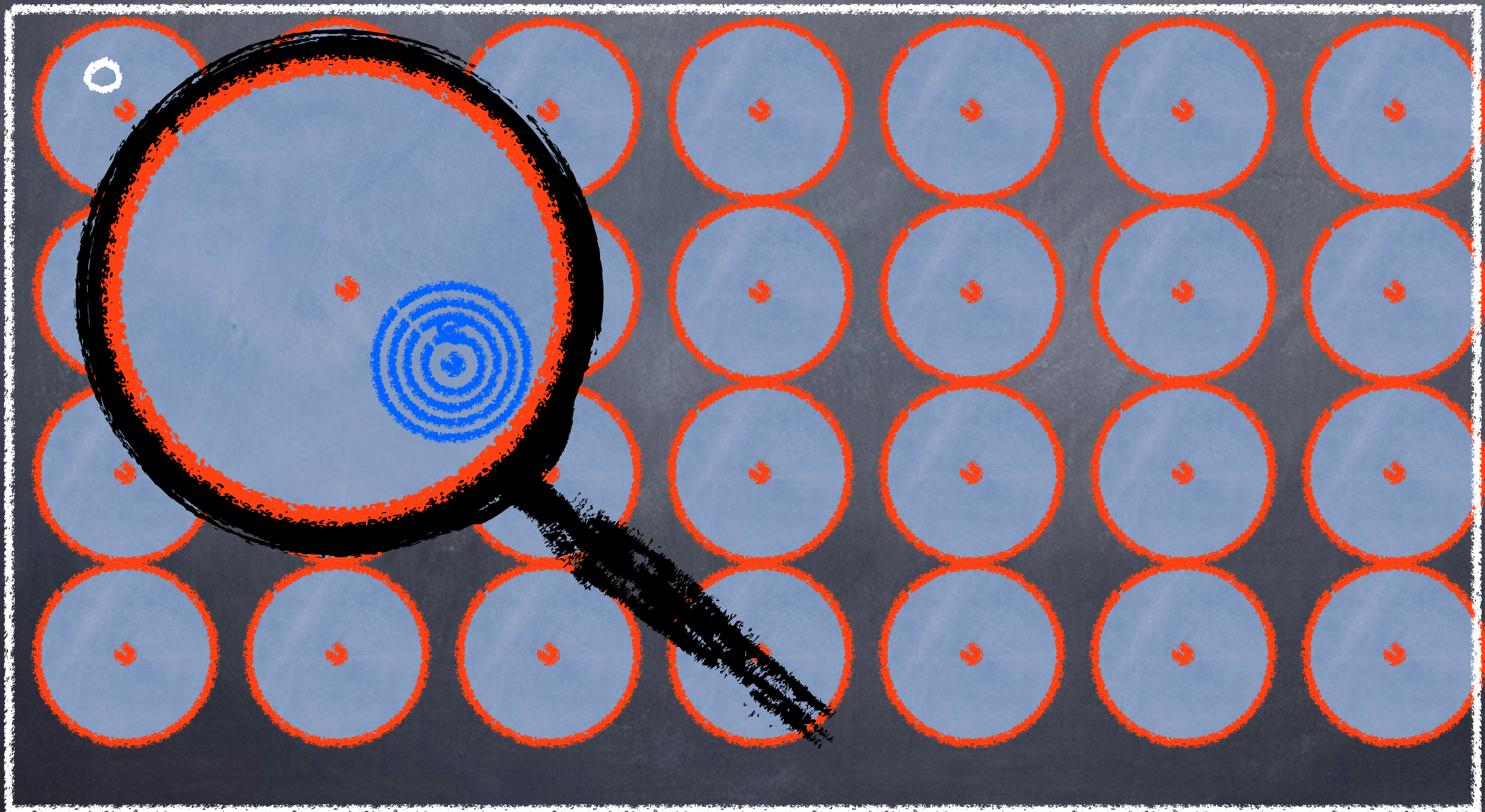
error detection vs error correction

$\mathbb{F}_q^n$



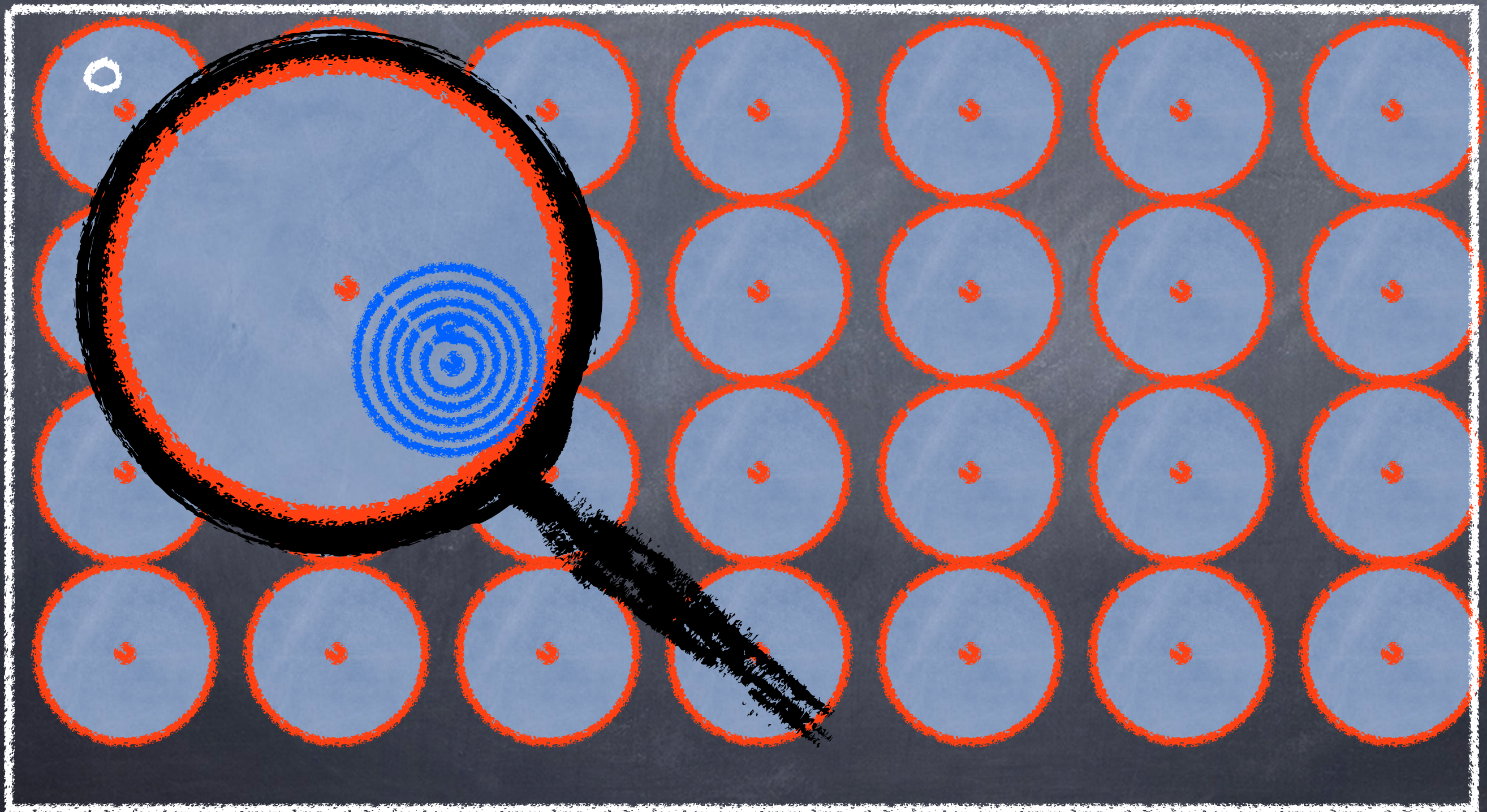
error detection vs error correction

$\mathbb{F}_q^n$



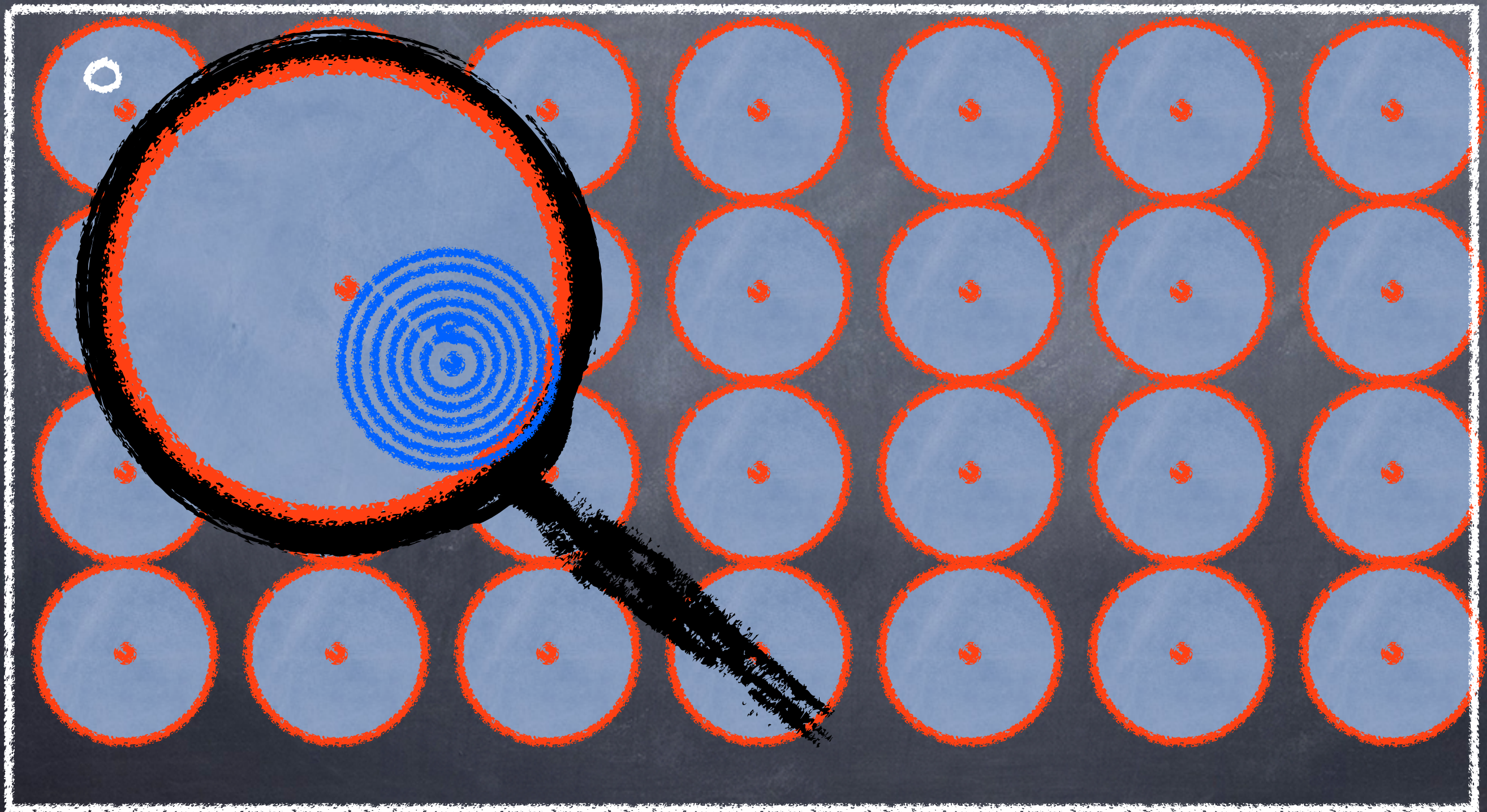
error detection vs error correction

$\mathbb{F}_q^n$



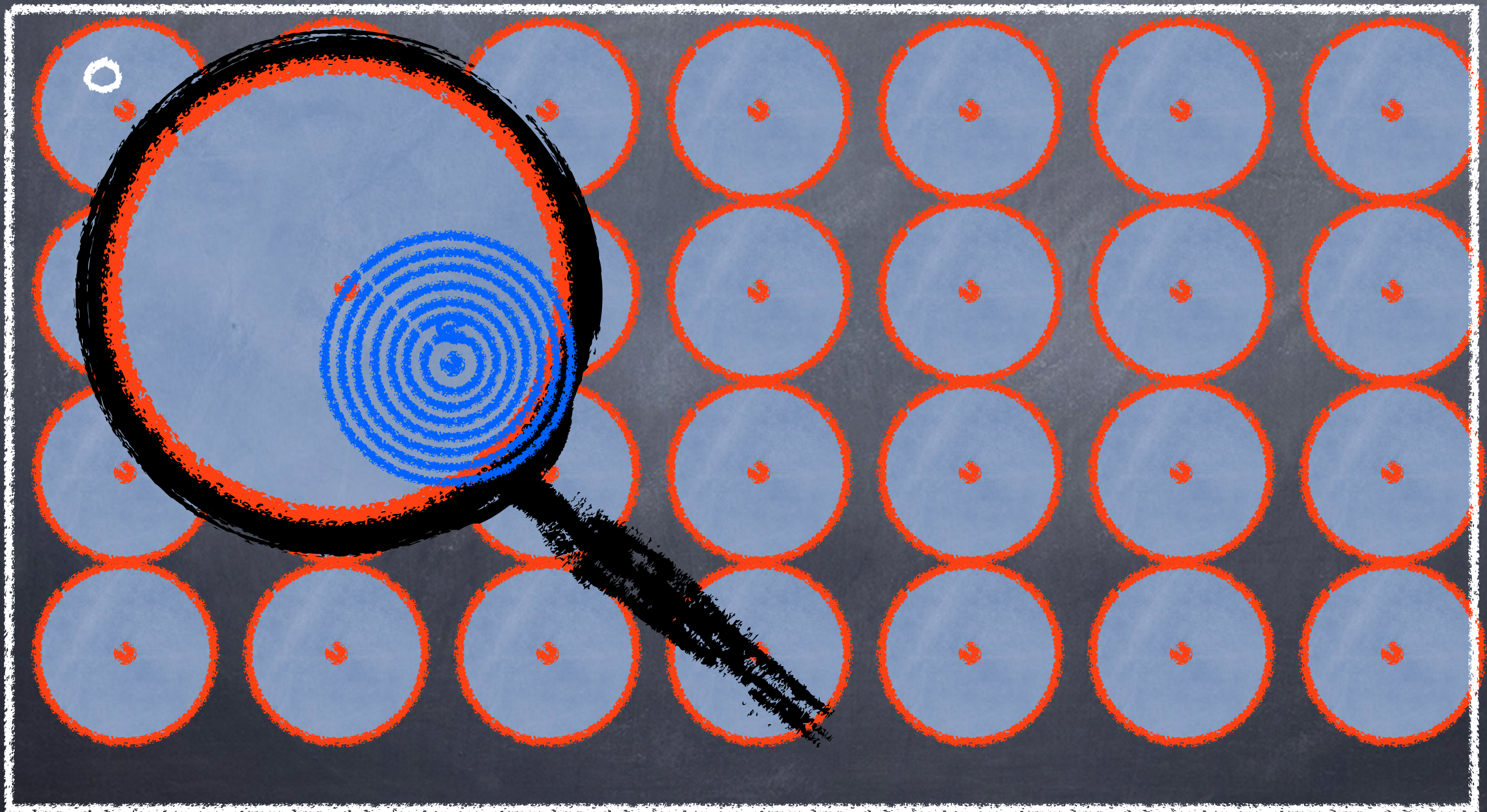
error detection vs error correction

$\mathbb{F}_q^n$



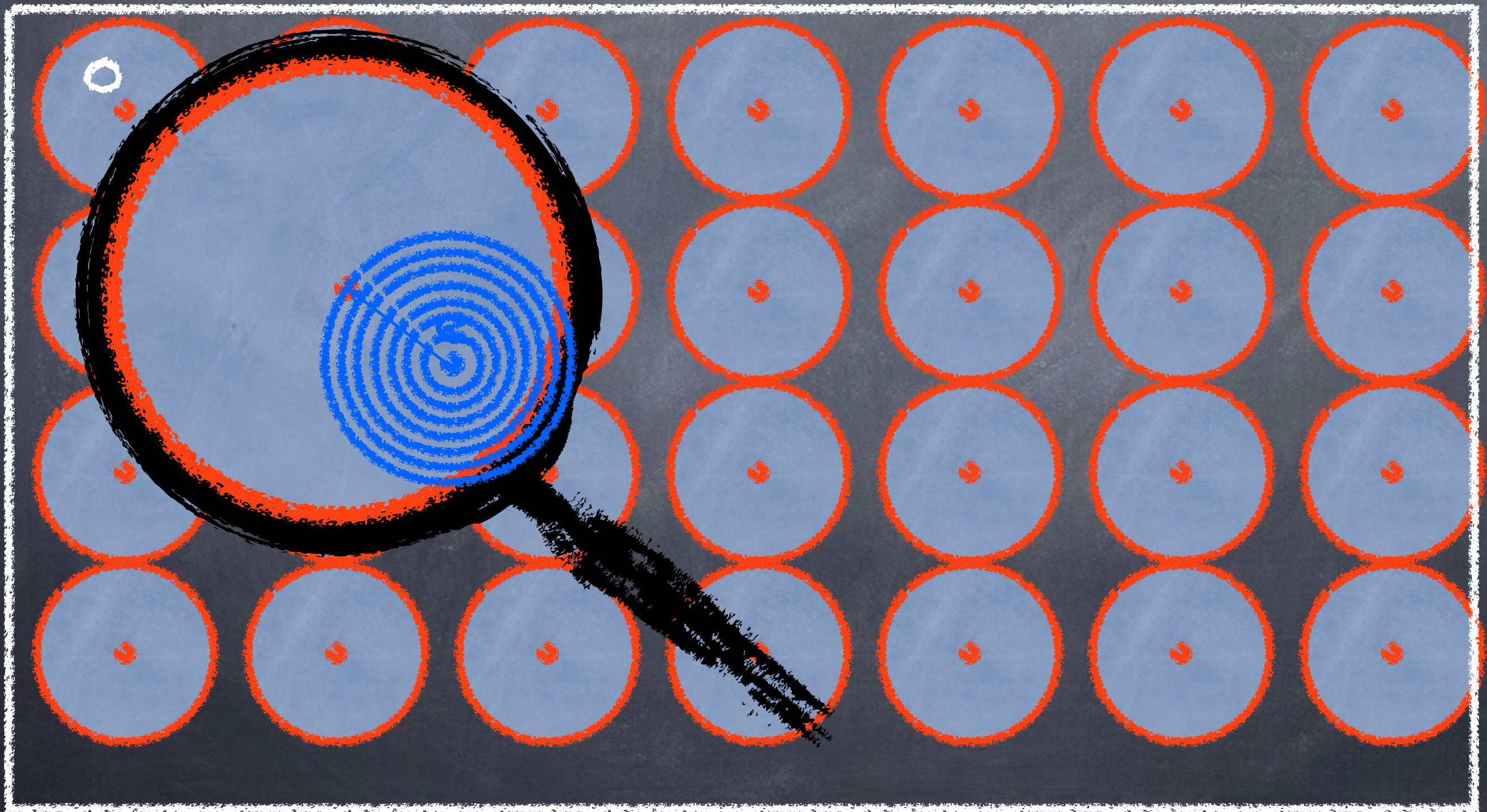
error detection vs error correction

$\mathbb{F}_q^n$



error detection vs error correction

$\mathbb{F}_q^n$



$[2^r-1, 2^r-r-1, 3]$ Hamming codes

- Hamming code: $[7, 4, 3]$

$$H = \begin{array}{c|ccc|ccc} & / & 0 & 0 & 0 & 1 & 1 & 1 & 1 & \backslash & / & \% & \backslash \\ H = & | & 0 & 1 & 1 & 0 & 0 & 1 & 1 & | & | & \% & | \\ & & \backslash & 1 & 0 & 1 & 0 & 1 & 0 & 1 & / & | & \% & | \\ & & & & & & & & & & & \backslash & \% & / \end{array}$$

$$C = \{ c \mid Hc = 0 \}$$

[7,4,3]

Hamming code

$$H = \begin{array}{c|ccc} & / & 0001111 & \backslash & / \% \backslash \\ \hline & | & 0110011 & || & \% | \\ & \backslash & 1010101 & / & | \% | \\ & & & & \backslash \% / \end{array}$$

$$H(c+e_i) = 0+He_i = He_i = i \text{ such that}$$

$$e_i = (\begin{array}{ccccccc} 0 & 0 & \dots & 1 & \dots & 0 & \end{array})$$

1 2 3 ... i ... 7

Correcting [7,4,3] Hamming code

$$\begin{array}{r}
 \bullet \quad \begin{array}{c} / \ 0001111 \ \backslash \ / \% \backslash \\
 H = \left| \begin{array}{ccc} 0110011 & || \% | \\
 \backslash \ 1010101 & / | \% | \\
 & \backslash \% / \end{array} \right.
 \end{array}
 \end{array}$$

$H(w) = i$ then $c = w - e_i$ is the nearest
codeword...

Reed-Solomon Codes

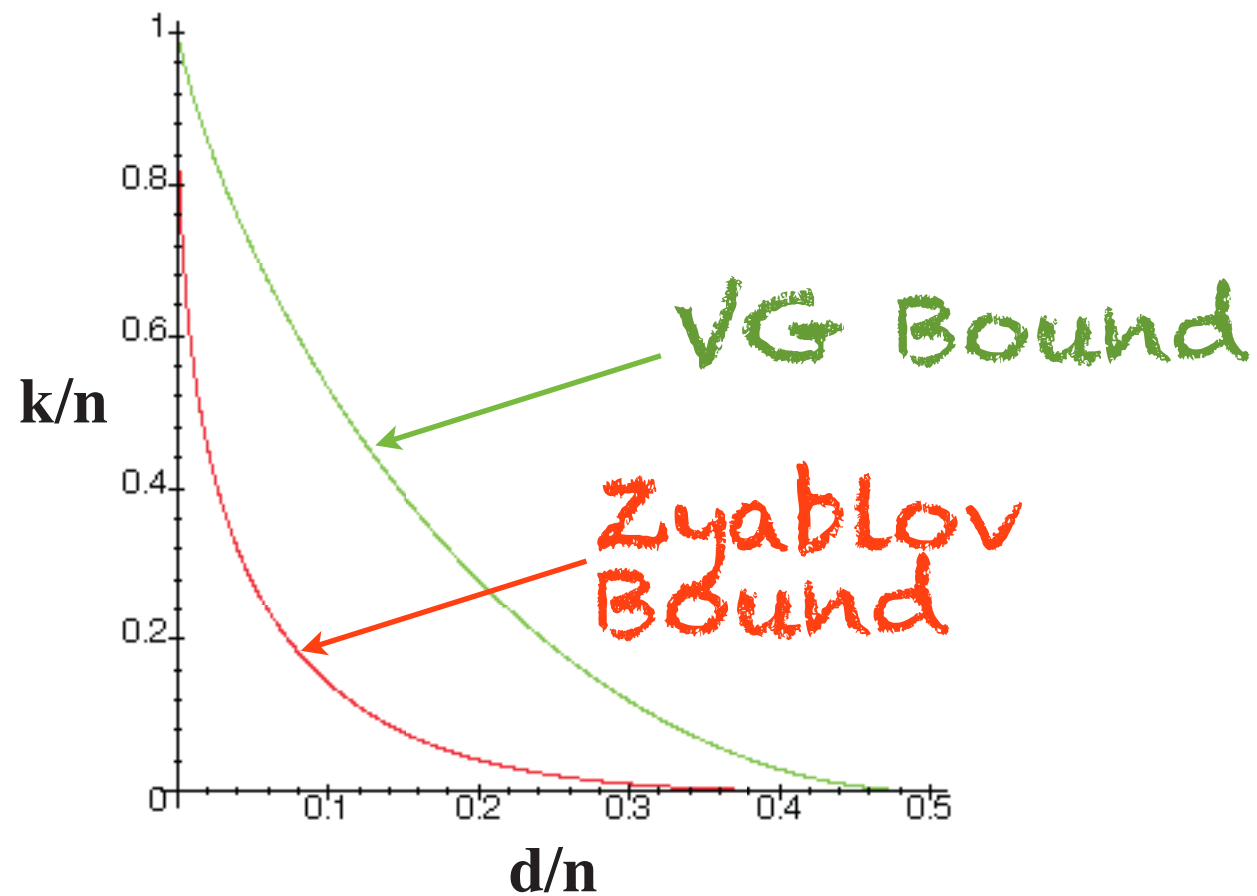
Reed-Solomon Codes

- an $[n, k, d]$ Reed-Solomon code C over a field $\mathbb{F}_n$ is the set of n^k codewords of the form $(p(0), p(\alpha^1), \dots, p(\alpha^{n-1}))$ where p is any polynomial of degree less than k and $0, \alpha^1, \dots, \alpha^{n-1}$ is a list of the elements of $\mathbb{F}_n$.

Reed-Solomon Codes

- an $[n, k, d]$ Reed-Solomon code C over a field $\mathbb{F}_n$ is the set of n^k codewords of the form $(p(0), p(\alpha^1), \dots, p(\alpha^{n-1}))$ where p is any polynomial of degree less than k and $0, \alpha^1, \dots, \alpha^{n-1}$ is a list of the elements of $\mathbb{F}_n$.
- Yields Zyablov Bound

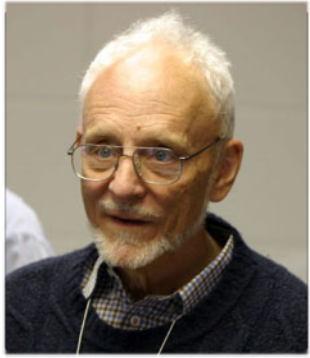
Reed-Solomon Codes



- Yields Zyablov Bound

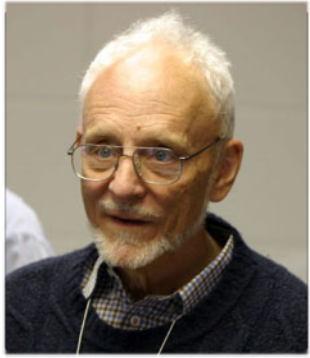
Reed-Solomon Codes

- all $[n, k, d]$ Reed-Solomon codes have $d = n - k + 1$ since any codeword with at least k values = zero must be the all-zero codeword.
(unique interpolation theorem)



Berlekamp-Welch error correction

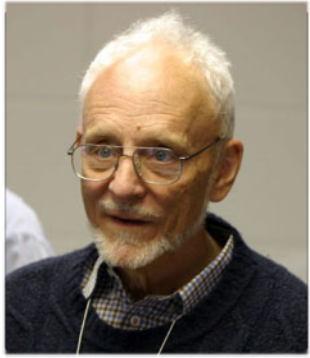




Berlekamp-Welch error correction

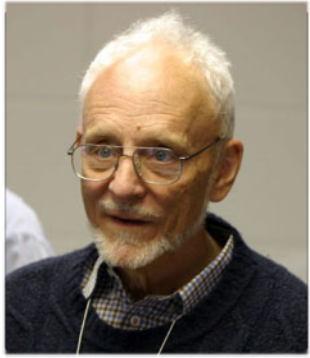


- (efficient correction procedure)



Berlekamp-Welch error correction

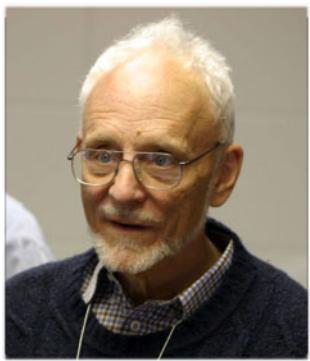
- (efficient correction procedure)
- To correct a word w upto $(n-k)/2$ errors applied to an $[n, k, n-k+1]$ Reed-Solomon codeword,



Berlekamp-Welch error correction

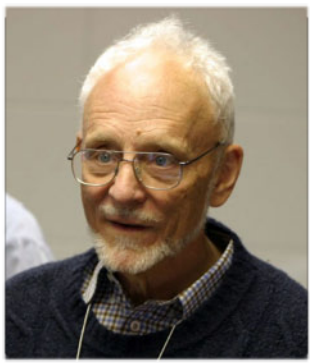


- (efficient correction procedure)
- To correct a word w upto $(n-k)/2$ errors applied to an $[n, k, n-k+1]$ Reed-Solomon codeword,
- find the positions in error,



Berlekamp-Welch error correction

- (efficient correction procedure)
- To correct a word w upto $(n-k)/2$ errors applied to an $[n, k, n-k+1]$ Reed-Solomon codeword,
- find the positions in error,
- interpolate p using correct positions



Berlekamp-Welch error correction

- (efficient correction procedure)
- To correct a word w upto $(n-k)/2$ errors applied to an $[n, k, n-k+1]$ Reed-Solomon codeword,
- find the positions in error,
- interpolate p using correct positions
- substitute each error by the correct value of $p(x)$.

Generalized RS

- an $[n, k, d]$ Generalized RS code C over a field $\mathbb{F}_n$ is a set of n^k codewords of the form $(z_1 p(a_0), z_2 p(a_1), \dots, z_n p(a_{n-1}))$, $z_i \neq 0$ where p is any polynomial of degree less than k and $a_0, a_1, \dots, a_{n-1}$ is a list of distinct elements of $\mathbb{F}_n$.

error correction

error correction

- in the following cryptosystems,

error correction

- in the following cryptosystems,
- the Neiderreiter system uses GRS codes...

error correction

- in the following cryptosystems,
- the Neiderreiter system uses GRS codes...
- the McEliece system uses Goppa codes, a special sub-field sub-code family of GRS codes...