small
methods

Supporting Information

for *Small Methods,* DOI: 10.1002/smtd.202101335

DNA-Based Concatenated Encoding System for High-Reliability and High-Density Data Storage

Yubin Ren, Yi Zhang, Yawei Liu, Qinglin Wu, Juanjuan Su, Fan Wang,* Dong Chen,* Chunhai Fan, Kai Liu,* and Hongjie Zhang*

DNA-based Concatenated Encoding System for High-Reliability and High-Density Data Storage

Yubin Ren[1], Yi Zhang[2], Yawei Liu[2], Qinglin Wu[3], Juanjuan Su[4]*, Fan Wang[2]*, Dong Chen[3]*, Chunhai Fan[5], Kai Liu[1]*, Hongjie Zhang[1]

1. RaptorQ fountain code

RaptorQ is a fountain code that could generate the required encoding symbols on the fly by the encoder from the source symbols. The decoder could decode successfully by receiving any set of encoding symbols whose cardinality is only slightly larger than the number of source symbols [1]. It is a systematic code improved from Raptor code defined in the RFC5053, providing higher reliability and better coding efficiency, and supporting a larger source block size. The mathematical symbols adopted in the subsequent description of RaptorQ are all defined in the RFC6330. For convenient encoding and decoding, given source block consisting of K source symbols, $K' - K$ additional padding symbols are used to expand the source block, where K' is at least the minimum value of K. $C'[0],\ldots,C'[K-1]$ denote the K source symbols. $Tuple[K', X]$ denotes a tuple generator where X denotes either an Internal Symbol ID (ISI) value or an Encoding Symbol ID (ESI) value. $(d, a, b, d1, a1, b1)$ is a source tuple determined from $ISI\ X$ using the $Tuple[]$ generator. L denotes the number of intermediate symbols. C denotes an array of intermediate symbols, $C[0], C[1], C[2],\ldots, C[L-1]$. S denotes the number of Low Density Parity Check (LDPC) symbols for a single extended source block. H denotes the number of High Density Parity Check (HDPC) symbols for a single extended source block. N denotes the number of encoding symbols received by the decoder. G_LDPC1 and G_LDPC2 represents the $S \times B$ and $S \times P$ LDPC matrices. I_S and I_H represents the $S \times S$ and the $H \times H$ identity matrix respectively. G_ENC is the $K' \times L$ matrix composed of received encoding symbols, and G_HDPC is the $H \times (K' + S)$ HDPC matrix. $Enc[K', (C[0], C[1],\ldots, C[L-1]), (d, a, b, d1, a1, b1)]$ denotes an encoding symbol generator.

(1) RaptorQ encoding

1) An extended source block is constructed by adding zero or more padding symbols such that its total number of symbols, K', is one of the specified numbers satisfying the minimum value greater than K or equal to K.

2) K' source tuples $(d[0], a[0], b[0], d1[0], a1[0], b1[0]), \ldots, (d[K'-1], a[K'-1], b[K'-1], d1[K'-1], a1[K'-1], b1[K'-1])$ are generated by the $Tuple[]$ generator.

3) In the pre-coding process, the intermediate symbols array $C[0], \ldots, C[L-1]$ from the source symbols are generated by the intermediate symbol generator.

4) The repair symbol with $ISI\ X (X \geq K')$ is generated by applying the generator $Enc[K', (C[0], C[1], \ldots, C[L-1]), (d, a, b, d1, a1, b1)]$. The coding symbol set is composed of repair symbols and source symbols.

(2) RaptorQ decoding

It is assumed that the structural parameters of the source block, including the symbol size T, the number of symbols K in the source block, and the number of symbols K' in the extended source block, are known to the decoder. $N\ (N \geq K')$ denotes the number of received encoding symbols to be used for decoding. Each encoding symbol received by the decoder is a known linear combination of intermediate symbols. Therefore, each received encoding symbol is represented as a linear equation related to the intermediate symbol, which is used to construct the linear equations, together with the known linear pre-coding relationship amongst the intermediate symbols. The intermediate symbol is obtained by solving the linear equations, and then the source symbol is decoded.

1) The pre-coding matrix A' is constructed according to the received encoding symbol and the known linear pre-coding relationship amongst the intermediate symbols.

2) The Gaussian elimination is adopted to solve linear equations $A' \times C = D$, then to obtain the array of intermediate symbols C.

3) The repair symbol with $ISI\ X$ is generated by applying the encoding generator $Enc[K', (C[0], C[1], \dots, C[L-1]), (d, a, b, d1, a1, b1)]$ and the source symbols are decoded.

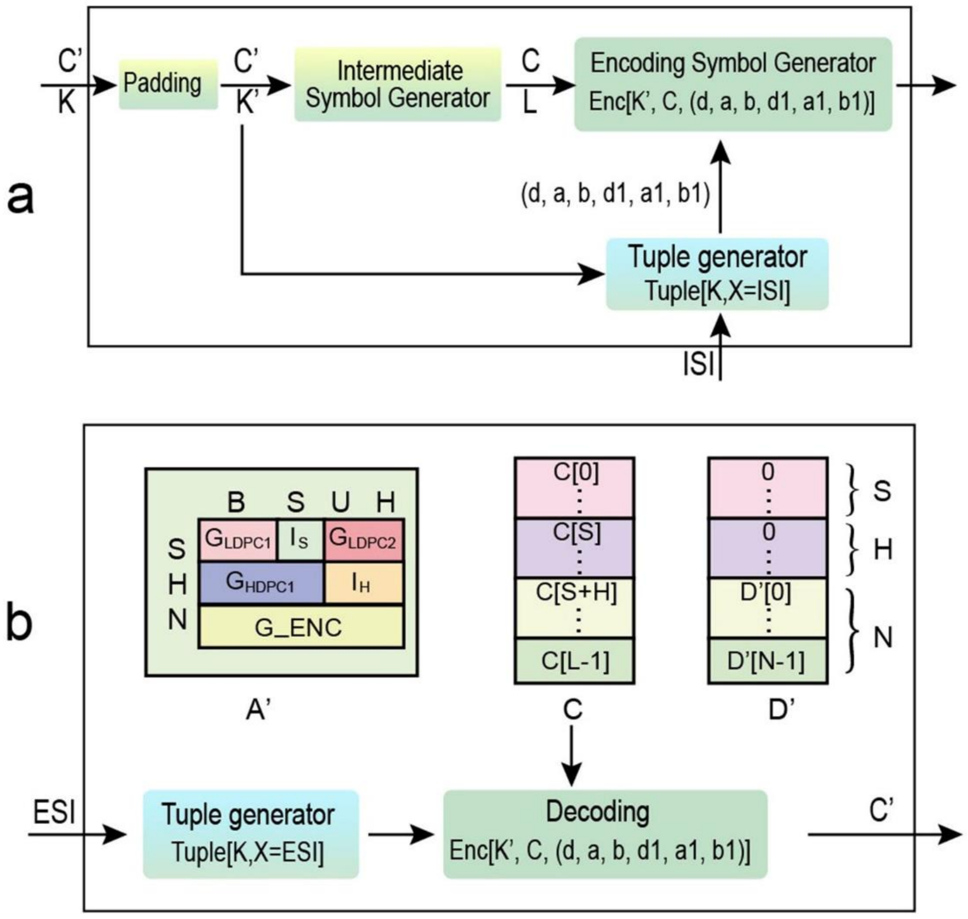


Figure S1 | *Schematic of the RaptorQ encoder and decoder.*

2. Arithmetic compression

Before starting to describe arithmetic code [2], we need to establish some mathematical symbols that map the source symbols or characters to numbers. For convenience, we adopt the mapping

$$X(a_i) = i \qquad a_i \in A \tag{1}$$

where $A = \left\{a_1 \ , a_2 \ , \ \cdots \ , a_m\right\}$ is a symbol set of discrete sources and X is a random variable. Suppose that the probability distribution function of the source is

$$P(X = i) = P(a_i) \tag{2}$$

and the cumulative density function can be defined as

$$F_X(i) = \sum_{k=1}^{i} P(X = k) \tag{3}$$

Define $T_X(a_i)$ as

$$T_X(a_i) = \sum_{k=1}^{i-1} P(X = k) + \frac{1}{2}P(X = i) \tag{4}$$

$$= F_X(i-1) + \frac{1}{2}P(X = i) \tag{5}$$

Note that for each sign a_i with nonzero probability, $F_X(i)$ is a discrete value corresponding to it one by one.

(1) arithmetic encoding

The process of arithmetic encoding is to gradually reduce the size of the subinterval where the label is located according to the cumulative probability distribution. The binary representation of any number in the subinterval, which is corresponded to the last element of the sequence, could act as the arithmetic code of the sequence.

1) Obtain the probability distribution of the source and the unit interval is divided into subintervals according to the cumulative probability distribution $\left[F_X(i-1), F_X(i)\right), i=1,\cdots,m$.

2) Associate the subinterval $\left[F_X(i-1), F_X(i)\right)$ with the symbol a_i and the first symbol in the sequence determines which subinterval of the source sequence label is located. Assuming that the first symbol is a_k, the interval of the label is $\left[F_X(k-1), F_X(k)\right)$.

3) The subinterval $\left[F_X(k-1), F_X(k)\right)$ is divided according to the same proportion as the original interval. The subinterval$\left[F_X(k-1)+F_X(j-1)\times\left(F_X(k)-F_X(k-1)\right), F_X(k-1)+F_X(j)\times\left(F_X(k)-F_X(k-1)\right)\right)$ obtained from the last dividing is associated with the symbol a_j. The second symbol is assumed as a_j, then the subinterval of the label of the source sequence becomes $\left[F_X(k-1)+F_X(j-1)\times\left(F_X(k)-F_X(k-1)\right), F_X(k-1)+F_X(j)\times\left(F_X(k)-F_X(k-1)\right)\right)$.

4) According to the above method, each follow-up symbol of the source sequence is processed sequentially, and the interval where the label locates is divided and associated with the subintervals according to the same proportion, until the end of the last symbol in the source sequence.

5) The binary representation corresponding to the value $T_X(a_i)$ of the subinterval where the last symbol of the sequence locates is the arithmetic codeword of the source sequence.

In the implementation of the specific algorithm, the subinterval containing the tag value of a specific source sequence will not intersect any subinterval containing other sequence tag values. Only the upper and lower limits of the subinterval containing the tag value of a specific sequence need to be calculated, and then any value of the subinterval can be selected as the arithmetic codeword of the source sequence. The upper and lower bounds of the interval are represented $u^{(n)}$ and $l^{(n)}$ respectively, where n is the sequence length. In general, for any source sequence $\mathrm{x} = (x_1 x_2 \cdots x_n)$, it can be proved that

$$l^{(n)} = l^{(n-1)} + \left(u^{(n-1)} - l^{(n-1)}\right) F_X(x_n - 1) \tag{6}$$

$$u^{(n)} = l^{(n-1)} + \left(u^{(n-1)} - l^{(n-1)}\right) F_X(x_n) \tag{7}$$

If the midpoint of the subinterval is used as the source label, then

$$T_X(x) = \frac{u^{(n)} + l^{(n)}}{2} \tag{8}$$

Therefore, the cumulative probability distribution of the source is the only needed information in the process of arithmetic coding. With the growth of the source sequence length, the interval becomes narrower and narrower, and the required precision is higher and higher, which cannot be expressed by finite precision. Therefore, interval scaling is used to solve this problem in practical application.

(2) arithmetic decoding

Arithmetic decoding is to obtain all intervals containing tags in the encoding process by simulating the encoder until the source sequence completes decoding. The specific process is as follows:

1) Initialize : $l^{(0)} = 0, u^{(0)} = 1$.

2) For each k , find $t^* = \left(T_X(x) - l^{(k-1)}\right)/\left(u^{(k-1)} - l^{(k-1)}\right)$.

3) Find the value of x_k satisfying $F_X(x_k - 1) \ll t^* < F_X(x_k)$.

4) Update $u^{(k)}$and l^k.

5) Continue until the entire sequence is decoded.

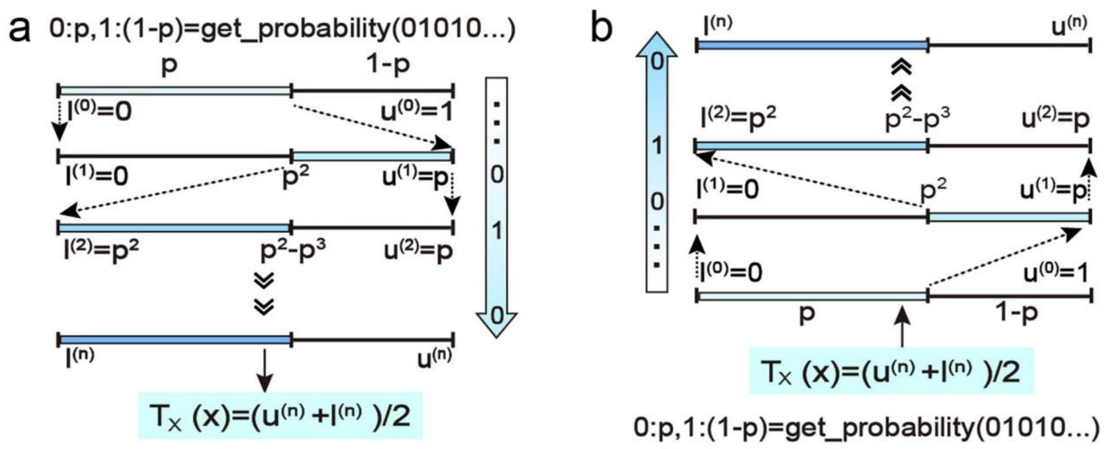


Figure S2 | *Schematic of the arithmetic encoding and decoding of the binary sequence.*

3. Improved Base64 encoding

The basic principle of improved Base64 encoding refers to the previous article [3].

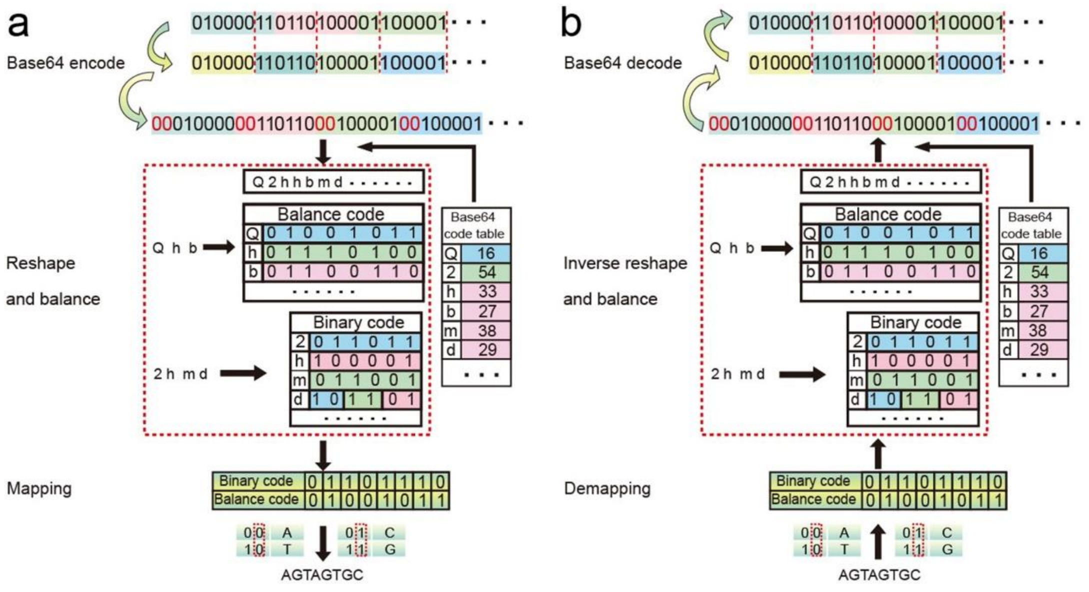


Figure S3 | *Schematic of the improved 4-nucleotide Base64 encoding and decoding of DNA sequence.*

4. LZW algorithm

(1) LZW encoding

The key point of the LZW algorithm is that the symbol patterns that have already appeared in the source sequence are mapped to shorter coding symbols to achieve data compression [4]. The encoder reads the source symbols one by one from the source sequence and attempts to encode the source symbols or the units composed of multiple source symbols into codewords. For convenience, the variable P represents the symbol unit that has been read but not yet encoded, and the variable C represents the new fetched source symbol. The 4-nucleotide LZW encoding process is described as follows：

1）In the initial state, there are only symbols in the set of all source symbols in the dictionary, i.e. {0: A, 1: T, 2: C, 3: G}. At this moment, variables P and C are null.

2）Read the new source symbol and assign it to C, and concatenate it with the variable P

to form the symbol sequence $P+C$.

3) Look up $P+C$ in the dictionary, if

- $P+C$ in the dictionary , $P=P+C$.

- $P+C$ not in the dictionary, encode and output the source symbol unit of variable P.

  At the same time, expand the $P+C$ item in the dictionary and update $P=C$.

4) Return to step 2 and repeat execution until all characters in the source sequence are encoded. If the variable C has no assignable source symbol at the end of the sequence, output the codeword corresponding to the variable P, and end the coding process.

(2) LZW decoding

The LZW decoding process is to simulate the encoder by restoring the encoding dictionary continuously to achieve decoding. For ease of explanation, the variable P_w represents the codeword has just been decoded, and the variable C_w represents the codeword currently reading. The $\mathrm{decode}(P_w)$ and $\mathrm{decode}(C_w)$ are used to represent the source symbol obtained by decoding the codeword or a unit of source symbols. The decoding process is described as follows:

1) In the initial state, there are only symbols in the set of all source symbols in the dictionary, i.e. {0: A, 1: T, 2: C, 3: G}. At this moment, variables P_w and C_w are null.

2) Read the first codeword in the codeword sequence and assign it to the variable C_w, then the corresponding source symbol is decoded and output.

3) Assign the variable C_w to the variable P_w, then read the next codeword and assign it to

the variable C_w.

4) Look up C_w in the dictionary, if

- C_w in the dictionary：

  a. Decode C_w, that is, output $\mathrm{decode}(C_w)$.

  b. Let $P = \mathrm{decode}(P_w)$, C is equal to the first character of $\mathrm{decode}(C_w)$.

  c. Expand $P + C$ items in the dictionary.

- C_w not in the dictionary：

  a. Let $P = \mathrm{decode}(P_w)$, C is equal to the first character of $\mathrm{decode}(C_w)$.

  b. Expand the $P + C$ term corresponding to C_w in the dictionary.

  c. Output $P + C$.

5) Return to step 3 and repeat execution until all codewords are decoded.

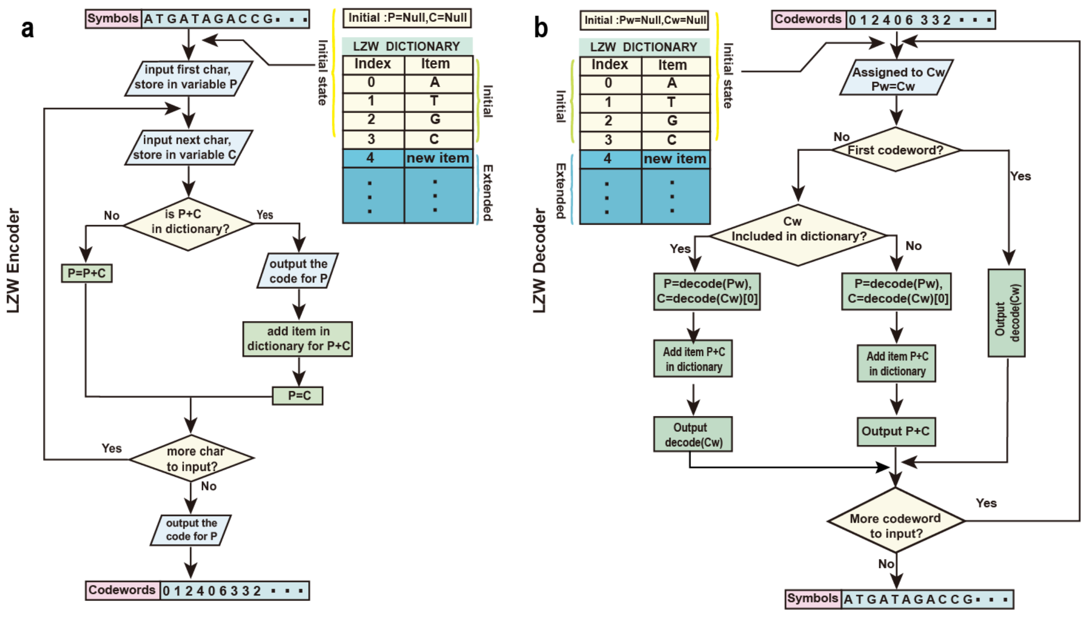


Figure S4 | *Schematic of the LZW encoding and decoding of 4-nucleotide DNA coding.*

5. Index designing

The indexing algorithm generates an index sequence space automatically according to the number of source blocks and symbols required for data volume. The index generator is designed based on the formula $N \leq d^m - q$, where N, d, m, q denote the number of source blocks or source symbols, the number of nucleotide types, the index length, and the number of indexing sequences containing homopolymer respectively. All the possible source block indexing sequence set and source symbol indexing sequence set are generated according to the number of source blocks and source symbols respectively. The index sequences containing homopolymer are removed by regular pattern matching and remaining the rational index sequences. The source block index as the primary index and the source symbol index as the secondary index are combined into the final composite index.

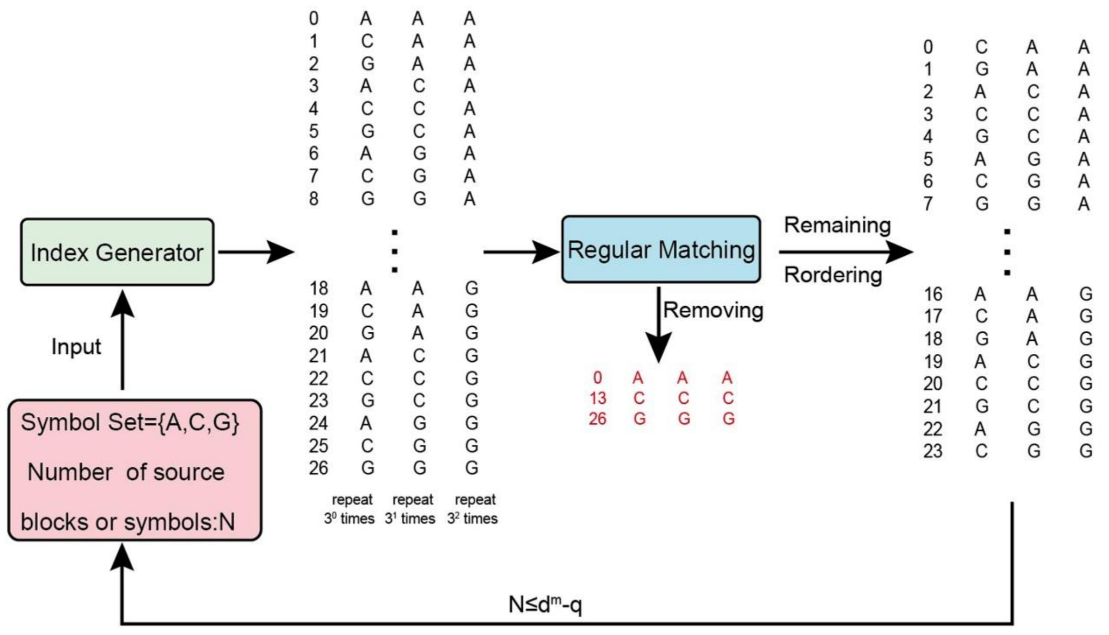


Figure S5 | *Taking 3-base symbols composing index characters as an example to show the principle of 6-digit composite index generating.*

6. Reed-Solomon (RS) code

RS code is a special case of primitive Bose–Chaudhuri–Hocquenghem (BCH) codes possessing the strong ability to correct random errors and burst errors **[5]**. The(n,k) RS code includes k information symbols, and each symbol includes m bits. The code length of the RS correcting t error-symbols is $n = 2^m - 1$, contains $n - k = 2t$ supervision symbols and the minimum code distance is $d = 2t + 1$. For an RS code with the length of $2^m - 1$, each symbol can be regarded as an element in the finite field $GF(2^m)$, and the generator polynomial is $g(x) = (x + \alpha)(x + \alpha^2) \cdots (x + \alpha^{d-1})$, where α^i is an element in the finite field of $GF(2^m)$. The following is only a procedural description of the encoding and decoding of RS error correction codes.

(1) RS encoding

1) calculate the primitive element α in the finite filed $GF(2^m)$ and find the minimum polynomials $m_1(x)$, $m_2(x), \cdots, m_{2t}(x)$ corresponding to 2t continuous n-th roots α , α^2 , ⋯ , α^{2t} respectively;

2) calculate the least common multiple of a polynomial $g(x) = LCM\left[m_1(x)\ ,\ m_2(x), \cdots, m_{2t}(x)\right]$;

3) calculate code polynomial $\mathrm{c}(x) = m(x)g(x)$.

(2) RS Algebraic decoding

Notation: received codeword $r(x) \in F_q[x]/(x^n - 1)$; $\hat{e}(x)$ is error pattern;$\hat{c}(x) \in F_q[x]/(x^n - 1)$ is the estimated value of decoded codeword , where $q = 2^m - 1$.

1) receive codeword $r(x) \in F_q[x]/(x^n - 1)$.

2) calculate syndromes $S_j = r(\alpha^j) = \sum_{i=0}^{n-1} r_i \alpha^{i_j}$, for $1 \le j \le 2t$.

3) if $S_j = 0$, for $1 \le j \le 2t$

then $\hat{e}(x) = 0$;

else

calculate syndrome polynomial $S(x) = \sum_{j=1}^{2t} S_j x^{j-1}$;

use Euclid's algorithm to calculate the corresponding error locator and error evaluator polynomial $\delta(x)$, $\omega(x)$, where $deg(\sigma(x)) \le t, \deg(\omega(x)) \le t - 1$, $\omega(x) \equiv S(x)\sigma(x) mod x^{2t}$;

```
    determine zeros $X_j^{-1} = \alpha^{-i_j}$ of $\delta(x)$, for $1 \le j \le w, w = deg(\sigma(x))$;

    calculate error values $Y_j = -\frac{\omega(X_j^{-1})}{\sigma\prime(X_j^{-1})}$ , for $1 \le j \le w$ ;

    calculate error positions $i_j$ from $X_j = \alpha^{i_j}$, for $1 \le j \le w$;

    calculate error polynomial $\hat{e}(x) = \sum_{j=1}^{w} Y_j\, x^{i_j}$;

    end

4)  calculate code polynomial $\hat{c}(x) = r(x) - \hat{e}(x)$.
```

7. The algorithm parameter setting and coding results of the RALR system

Table S1 | *The algorithm parameter setting and coding results of the RALR system without symbol-level compression.*

RALR Without Symbol-level Compression												
Coding Type	**File Properties**				**DNA strand properties**					**Error correction redundancy**		**Coding Efficiency (Bits/NT)**
	Name	**Type**	**Size (byte)**	**Block Size**	**Chain Number**	**Chain Length (NT)**	**Index Base Number**	**C Content**	**G Content**	**Raptor Q**	**RS**	
4-Base	**1.txt**	**Text**	**682**	**684**	**18*1**	**195~223**	**5**	**24.7%**	**25.2%**	**0**	**0**	**1.44**
	–	**–**	**–**	**–**	**19*1**	**204~232**	**–**	**25.4%**	**24.7%**	**38*1**	**2*19*1**	**1.30**
	2.jpg	**Picture**	**39048**	**1568**	**49*25**	**44~236**	**6**	**25.6%**	**24.5%**	**0**	**0**	**1.34**
	–	**–**	**–**	**–**	**50*25**	**52~244**	**–**	**25.9%**	**24.4%**	**32*25**	**2*50*25**	**1.26**
	3.wmv	**Video**	**2573476**	**1152**	**32*2234**	**49~295**	**9**	**25.6%**	**24.2%**	**0**	**0**	**1.34**
	–	**–**	**–**	**–**	**33*2234**	**60~304**	**–**	**25.3%**	**24.1%**	**36*2234**	**2*33*2234**	**1.24**
6-Base	**1.txt**	**Text**	**682**	**684**	**18*1**	**162~180**	**4**	**17.0%**	**15.9%**	**0**	**0**	**1.79**
	–	**–**	**–**	**–**	**19*1**	**170~188**	**–**	**16.7%**	**16.3%**	**38*1**	**2*19*1**	**1.62**

	2.jpg	**Picture**	**39048**	**1568**	**49*25**	**33~169**	**5**	**16.9%**	**16.5%**	**0**	**0**	**1.71**
	—	**—**	**—**	**—**	**50*25**	**41~177**	**—**	**17.2%**	**17.0%**	**32*25**	**2*50*25**	**1.59**
	3.wmv	**Video**	**2573476**	**1152**	**32*2234**	**36~200**	**8**	**16.7%**	**16.1%**	**0**	**0**	**1.74**
	—	**—**	**—**	**—**	**33*2234**	**44~108**	**—**	**17.1%**	**16.5%**	**36*2234**	**2*33*2234**	**1.61**
8-Base	**1.txt**	**Text**	**682**	**684**	**18*1**	**140~158**	**4**	**12.6%**	**13.4%**	**0**	**0**	**2.06**
	—	**—**	**—**	**—**	**19*1**	**147~166**	**—**	**12.4%**	**13.3%**	**38*1**	**2*19*1**	**1.85**
	2.jpg	**Picture**	**39048**	**1568**	**49*25**	**30~142**	**4**	**12.6%**	**13.3%**	**0**	**0**	**1.97**
	—	**—**	**—**	**—**	**50*25**	**38~150**	**—**	**12.5%**	**13.2%**	**32*25**	**2*50*25**	**1.83**
	3.wmv	**Video**	**2573476**	**1152**	**32*2234**	**34~176**	**6**	**13.0%**	**13.4%**	**0**	**0**	**2.01**
	—	**—**	**—**	**—**	**33*2234**	**40~182**	**—**	**12.9%**	**13.3%**	**36*2234**	**2*33*2234**	**1.86**

Table S2 | *The algorithm parameter setting and coding results of the RABR system without symbol-level compression.*

RABR Without Symbol-level Compression												
Coding Type	**File Properties**				**DNA strand properties**					**Error correction redundancy**		**Coding Efficiency (Bits/NT)**
	Name	**Type**	**Size(byte)**	**Block Size**	**Chain Number**	**Length**	**Index Base Number**	**C Content**	**G Content**	**RaptorQ**	**RS**	
4-Base	**1.txt**	**Text**	**682**	**684**	**18*1**	**185**	**5**	**25.6%**	**25.4%**	**0**	**0**	**1.64**
	—	**—**	**—**	**—**	**19*1**	**193**	**—**	**25.4%**	**25.6%**	**38*1**	**2*19*1**	**1.49**
	2.jpg	**Picture**	**39048**	**1568**	**49*25**	**159**	**7**	**24.7%**	**25.8%**	**0**	**0**	**1.60**
	—	**—**	**—**	**—**	**50*25**	**167**	**—**	**24.7%**	**25.8%**	**32*25**	**2**50*25**	**1.50**
	3.wmv	**Video**	**2573476**	**1152**	**32*2234**	**177**	**9**	**25.4%**	**24.7%**	**0**	**0**	**1.63**
	—	**—**	**—**	**—**	**33*2234**	**185**	**—**	**25.3%**	**24.8%**	**36*2234**	**2*33*2234**	**1.51**
6-Base	**1.txt**	**Text**	**682**	**684**	**18*1**	**135**	**5**	**16.8%**	**16.8%**	**0**	**0**	**2.25**
	—	**—**	**—**	**—**	**19*1**	**143**	**—**	**17.4%**	**17.2%**	**38*1**	**2*19*1**	**2.01**
	2.jpg	**Picture**	**39048**	**1568**	**49*25**	**118**	**6**	**16.6%**	**16.7%**	**0**	**0**	**2.16**
	—	**—**	**—**	**—**	**50*25**	**126**	**—**	**17.2%**	**17.2%**	**32*25**	**2*50*25**	**1.98**

	3.wmv	Video	2573476	1152	32*2234	130	8	16.2%	17.3%	0	0	2.22
	—	—	—	—	33*2234	138	—	17.5%	17.4%	36*2234	2*33*2234	2.02
8-Base	1.txt	Text	682	684	18*1	117	4	14.6%	10.4%	0	0	2.59
	—	—	—	—	19*1	123	—	14.6%	10.6%	38*1	2*19*1	2.33
	2.jpg	Picture	39048	1568	49*25	101	5	11.9%	13.0%	0	0	2.52
	—	—	—	—	50*25	107	—	12.4%	12.9%	32*25	2*50*25	2.34
	3.wmv	Video	2573476	1152	32*2234	111	6	13.4%	12.3%	0	0	2.59
	—	—	—	—	33*2234	117	—	12.6%	12.2%	36*2234	2*33*2234	2.39

Table S3 | The algorithm parameter setting and coding results of the RALR system with symbol-level compression.

RALR With Symbol-level Compression												
Coding Type	File Properties				DNA strand properties					Error correction redundancy		Coding Efficiency (Bits/NT)
	Name	Type	Size(byte)	Block Size	Chain Number	Length	Index Base Number	C Content	G Content	RaptorQ	RS	
4-Base	1.txt	Text	682	684	18*1	105~137	5	25.6%	24.1%	0	0	2.50
	—	—	—	—	19*1	116~156	—	24.3%	24.5%	38*1	2*19*1	2.17
	2.jpg	Picture	39048	1568	49*25	14~144	6	24.3%	26.0%	0	0	1.97
	—	—	—	—	50*25	24~152	—	24.2%	25.8%	32*25	2*50*25	1.81
	3.wmv	Video	2573476	1152	32*2234	17~175	9	24.7%	24.4%	0	0	1.95
	—	—	—	—	33*2234	18~184	—	24.4%	24.2%	36*2234	2*33*2234	1.77
6-Base	1.txt	Text	682	684	18*1	84~104	4	16.8%	17.1%	0	0	3.11
	—	—	—	—	19*1	92~124	—	16.9%	17.7%	38*1	2*19*1	2.70
	2.jpg	Picture	39048	1568	49*25	13~111	5	17.0%	16.0%	0	0	2.60
	—	—	—	—	50*25	21~119	—	16.0%	16.5%	32*25	2*50*25	2.36
	3.wmv	Video	2573476	1152	32*2234	16~136	8	17.3%	16.4%	0	0	2.52
	—	—	—	—	33*2234	24~144	—	16.7%	16.9%	36*2234	2*33*2234	2.29

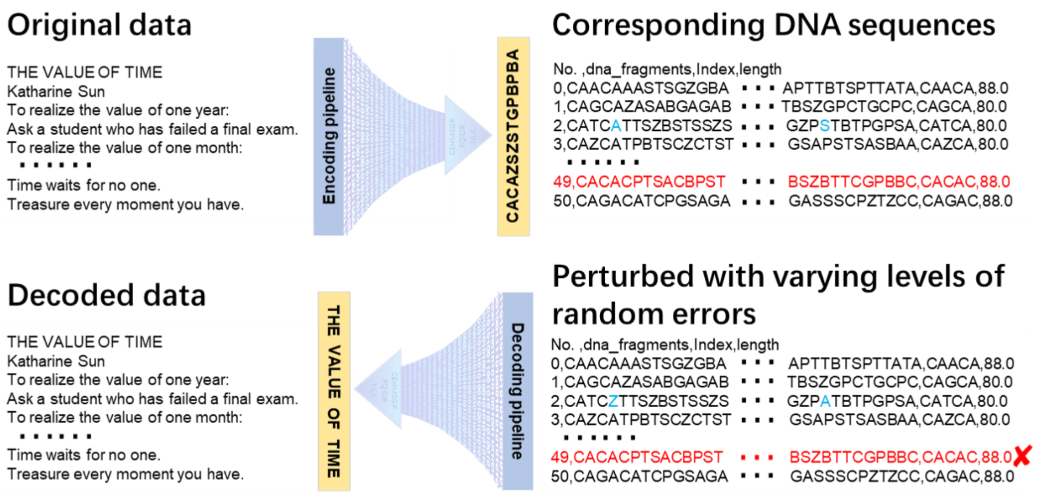
8-Base	1.txt	Text	682	684	18*1	80~98	4	13.1%	12.3%	0	0	3.41
	—	—	—	—	19*1	86~115	—	12.8%	12.0%	38*1	2*19*1	2.97
	2.jpg	Picture	39048	1568	49*25	18~98	4	12.4%	13.6%	0	0	2.93
	—	—	—	—	50*25	16~102	—	12.3%	13.4%	32*25	2*50*25	2.66
	3.wmv	Video	2573476	1152	32*2234	12~116	6	12.1%	12.7%	0	0	2.82
	—	—	—	—	33*2234	19~123	—	12.1%	12.5%	36*2234	2*33*2234	2.57

Table S4 | *The algorithm parameter setting and coding results of RABR with symbol-level compression.*

RABR With Symbol-level Compression												
Coding Type	File Properties				DNA strand properties					Error correction redundancy		Coding Efficiency (Bits/NT)
	Name	Type	Size(byte)	Block Size	Chain Number	Length	Index Base Number	C Content	G Content	RaptorQ	RS	
4-Base	1.txt	Text	682	684	18*1	101~117	5	24.3%	24.3%	0	0	2.97
	—	—	—	—	19*1	109~137	—	24.3%	24.2%	38*1	2*19*1	2.58
	2.jpg	Picture	39048	1568	49*25	23~103	7	24.0%	24.4%	0	0	2.49
	—	—	—	—	50*25	31~111	—	24.1%	24.4%	32*25	2*50*25	2.26
	3.wmv	Video	2573476	1152	32*2234	25~133	9	25.4%	24.6%	0	0	2.40
	—	—	—	—	33*2234	33~141	—	25.3%	24.6%	36*2234	2*33*2234	2.18
6-Base	1.txt	Text	682	684	18*1	75~85	5	16.3%	17.2%	0	0	4.01
	—	—	—	—	19*1	83~103	—	17.2%	17.9%	38*1	2*19*1	3.40
	2.jpg	Picture	39048	1568	49*25	16~78	6	16.6%	16.1%	0	0	3.37
	—	—	—	—	50*25	24~84	—	17.0%	17.3%	32*25	2*50*25	2.99
	3.wmv	Video	2573476	1152	32*2234	18~98	8	17.2%	16.6%	0	0	3.29
	—	—	—	—	33*2234	26~106	—	16.5%	16.9%	36*2234	2*33*2234	2.92
8-Base	1.txt	Text	682	684	18*1	65~73	4	13.2%	11.3%	0	0	4.63
	—	—	—	—	19*1	71~89	—	12.9%	11.1%	38*1	2*19*1	3.97

	2.jpg	Picture	39048	1568	49*25	14~66	5	12.3%	11.2%	0	0	3,88
	—	—	—	—	50*25	20~72	—	12.9%	11.1%	32*25	2*50*25	3.49
	3.wmv	Video	2573476	1152	32*2234	15~85	6	11.8%	13.1%	0	0	3.86
	—	—	—	—	33*2234	21~91	—	12.3%	12.9%	36*2234	2*33*2234	3.46

8. The test of the decoding steps using perturbed DNA sequences generated by introducing varying levels of random errors



Original data

THE VALUE OF TIME
Katharine Sun
To realize the value of one year:
Ask a student who has failed a final exam.
To realize the value of one month:
······
Time waits for no one.
Treasure every moment you have.

Corresponding DNA sequences

```
No. ,dna_fragments,Index,length
0,CAACAAASTSGZGBA ··· APTTBTSPTTATA,CAACA,88.0
1,CAGCAZASABGAGAB ··· TBSZGPCTGCPC,CAGCA,80.0
2,CATCATTSZBSTSSZS ··· GZPSTBTPGPSA,CATCA,80.0
3,CAZCATPBTSCZCTST ··· GSAPSTSASBAA,CAZCA,80.0
······
49,CACACPTSACBPST ··· BSZBTTCGPBBC,CACAC,88.0
50,CAGACATCPGSAGA ··· GASSSCPZTZCC,CAGAC,88.0
```

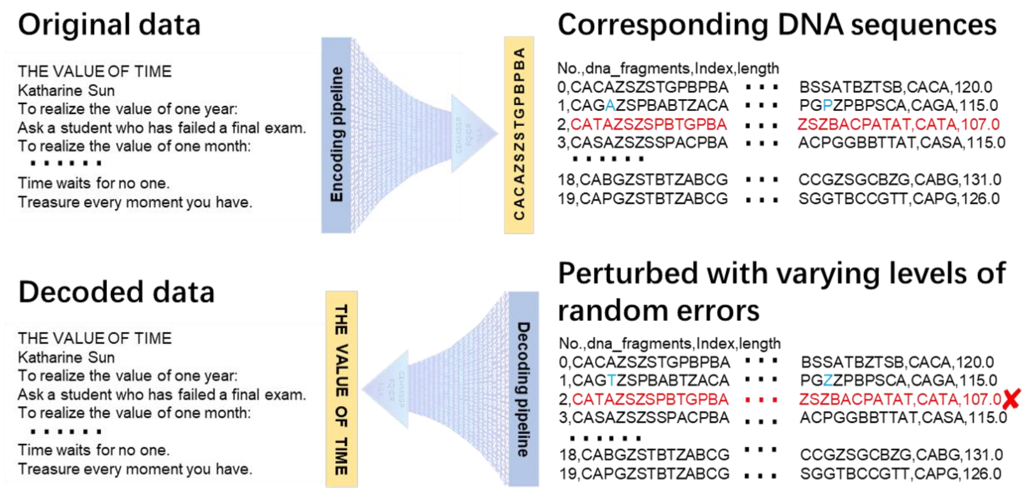
Decoded data

THE VALUE OF TIME
Katharine Sun
To realize the value of one year:
Ask a student who has failed a final exam.
To realize the value of one month:
······
Time waits for no one.
Treasure every moment you have.

Perturbed with varying levels of random errors

```
No. ,dna_fragments,Index,length
0,CAACAAASTSGZGBA ··· APTTBTSPTTATA,CAACA,88.0
1,CAGCAZASABGAGAB ··· TBSZGPCTGCPC,CAGCA,80.0
2,CATCZTTSZBSTSSZS ··· GZPATBTPGPSA,CATCA,80.0
3,CAZCATPBTSCZCTST ··· GSAPSTSASBAA,CAZCA,80.0
······
49,CACACPTSACBPST ··· BSZBTTCGPBBC,CACAC,88.0 ✗
50,CAGACATCPGSAGA ··· GASSSCPZTZCC,CAGAC,88.0
```

Figure S6 | *Successful decoding of correct information from perturbed DNA sequences with DNA sequence loss and substitution error by the RABR system.*



```
THE VALUE OF TIME
Katharine Sun
To realize the value of one year:
Ask a student who has failed a final exam.
To realize the value of one month:
  ......
Time waits for no one.
Treasure every moment you have.
```

```
No.,dna_fragments,Index,length
0,CACAZSZSTGPBPBA  ...  BSSATBZTSB,CACA,120.0
1,CAGAZSPBABTZACA  ...  PGPZPBPSCA,CAGA,115.0
2,CATAZSZSPBTGPBA  ...  ZSZBACPATAT,CATA,107.0
3,CASAZSZSSPACPBA  ...  ACPGGBBTTAT,CASA,115.0
  ......
18,CABGZSTBTZABCG  ...  CCGZSGCBZG,CABG,131.0
19,CAPGZSTBTZABCG  ...  SGGTBCCGTT,CAPG,126.0
```

```
No.,dna_fragments,Index,length
0,CACAZSZSTGPBPBA  ...  BSSATBZTSB,CACA,120.0
1,CAGTZSPBABTZACA  ...  PGZZPBPSCA,CAGA,115.0
2,CATAZSZSPBTGPBA  ...  ZSZBACPATAT,CATA,107.0✗
3,CASAZSZSSPACPBA  ...  ACPGGBBTTAT,CASA,115.0
  ......
18,CABGZSTBTZABCG  ...  CCGZSGCBZG,CABG,131.0
19,CAPGZSTBTZABCG  ...  SGGTBCCGTT,CAPG,126.0
```

```
THE VALUE OF TIME
Katharine Sun
To realize the value of one year:
Ask a student who has failed a final exam.
To realize the value of one month:
  ......
Time waits for no one.
Treasure every moment you have.
```

Figure S7 | *Successful decoding of correct information from perturbed DNA sequences with DNA sequence loss and substitution error by the RALR system.*

References

[1] M. Luby, A. Shokrollahi, M. Watson, T. Stockhammer, L. Minder, RaptorQ Forward Error Correction Scheme for Object Delivery, **2011**.

[2] K. Sayood, Arithmetic Coding, An *Introduction to Data Compression (Fourth Edition)*, Sayood, K., Ed. Morgan Kaufmann: Boston, **2012**; pp 91-133.

[3] Y. Zhang, L. Kong, F. Wang, B. Li, C. Ma, D. Chen, K. Liu, C. Fan, H. Zhang, *Nano Today* **2020**, *33*, 100871. DOI: 10.1016/j.nantod.2020.100871.

[4] T.A. Welch, Welch T. A Technique for High-Performance Data Compression, *Computer* **1984**, *17*, 8-19.

[5] A. Neubauer, J. Freudenberger, V. Kuehn, *Coding Theory: Algorithms, Architectures, and Applications*. **2007**, doi: 10.1002/9780470519837.