Data Storage Based on Combinatorial Synthesis of DNA Shortmers

Inbal Preuss[1], Zohar Yakhini[1,2] Leon Anavy[1,2]

[1]School of Computer Science, Herzliya Interdisciplinary Center, Herzliya, 4610101, Israel.
[2]Computer Science Department, Technion, Haifa, 3200003, Israel.

June 16, 2021

Abstract

Storage needs represent a significant burden on the economy and the environment. Some of this can potentially be offset by improved density molecular storage. The potential of using DNA for storing data is immense. DNA can be harnessed as a high density, durable archiving medium for compressing and storing the exponentially growing quantities of digital data that mankind generates. Several studies have demonstrated the potential of DNA-based data storage systems. These include exploration of different encoding and error correction schemes and the use of different technologies for DNA synthesis and sequencing. Recently, the use of composite DNA letters has been demonstrated to leverage the inherent redundancy in DNA based storage systems to achieve higher logical density, offering a more cost-effective approach. However, the suggested composite DNA approach is still limited due to its sensitivity to the stochastic nature of the process. Combinatorial assembly methods were also suggested to encode information on DNA in high density, while avoding the challenges of the stochastic system. These are based on enzynatic assembly processes for producing the synthetic DNA.

In this paper, we propose a novel method to encode information into DNA molecules using combinatorial encoding and shortmer DNA synthesis, in compatibility with current chemical DNA synthesis technologies. Our approach is based on a set of easily distinguishable DNA shortmers serving as building blocks and allowing for near-zero error rates. We construct an extended combinatorial alphabet in which every letter is a subset of the set of building blocks. We suggest different combinatorial encoding schemes and explore their theoretical properties and practical implications in terms of error probabilities and required sequencing depth. To demonstrate the feasibility of our approach, we implemented an end-to-end computer simulation of a DNA-based storage system, using our suggested combinatorial encodings. We use simulations to assess the performance of the system and the effect of different parameters.

Our simulations suggest that our combinatorial approach can potentially achieve up to 6.5-fold increase in the logical density over standard DNA based storage systems, with near zero reconstruction error.

Implementing our approach at scale to perform actual synthesis, requires minimal alterations to current technologies. Our work thus suggests that the combination of combinatorial encoding with standard DNA chemical synthesis technologies can potentially improve current solutions, achieving scalable, efficient and cost-effective DNA-based storage.

1 Introduction

DNA is a promising candidate to serve as storage media for long-term data archiving due to its high information density, long-term stability, and robustness. In recent years, several studies have demonstrated the use of synthetic DNA for storing digital information on a megabyte scale, exceeding the physical density of current magnetic-tape based systems by roughly six orders of magnitude [1] [2].

Efforts in the field of DNA-based storage are mainly focused on using standard DNA synthesis and sequencing technologies, applying various encoding schemes to reduce error rate and ensure reliability [1] [2] [3] [4] [5] [6] [7]. Yet, despite the enormous benefits potentially associated with capacity, robustness, and size, existing DNA-based storage technologies create information redundancy. This is due to the nature of DNA synthesis and sequencing methodologies, which process multiple molecules that represent the same DNA sequence in parallel. Recent studies suggested exploiting this redundancy to increase the system's logical density, by extending the standard DNA alphabet using composite letters and thereby encoding more than 2 bits per letter [8] [9] [10].

A composite DNA letter uses all four DNA bases (A, C, G, and T), combined or mixed in a specified predetermined ratio $\sigma = (\sigma_A, \sigma_C, \sigma_G, \sigma_T)$. A resolution parameter $k = \sigma_A + \sigma_C + \sigma_G + \sigma_T$ is defined, to control the alphabet size. The full composite alphabet of resolution k, denoted Φ_k, is the set of all $\sigma = (\sigma_A, \sigma_C, \sigma_G, \sigma_T)$, so that $\Sigma_{i\in\{A,C,G,T\}}\sigma_i = k$. In order to "write" a composite letter onto a specific predefined position in a DNA sequence, the multiple molecules representing the custom encoded sequence must be synthesized, while preserving the desired ratio between the different letters across all the molecules. For this to be done, current synthesis technologies are utilized, which produce multiple copies, and thereby allow for the implementation of the mixtures. Using the extended composite alphabet increases the logical density of the DNA based storage, breaking the theoretical limit of two bits per synthesis cycle as has been previsouly demonstrated [9] [10].

A recent study shows the power of using combinatorial assembly to encode information in DNA. Using propriety high throughput machinery, based on enzymatic assembly, the authors encoded information in a megabit per second write speed. As a proof of principle, they wrote a 25KB message into DNA, reading it using commercially available nanopore sequencing technology [11] [12].

In this paper, we present a novel approach to encode information in DNA using combinatorial encoding and shortmer DNA synthesis. The method described herein leverages the advantages of combinatorial encoding schemes, while relying on existing DNA chemical synthesis methods with some modifications. Using shortmer DNA synthesis also minimizes the effect of synthesis and sequencing errors. We formally define shortmer-based combinatorial encoding schemes, explore different designs, and analyze their performance. One such design is a variation of combinatorial encoding, wherein the size of the subset is fixed as part of the design, allowing for higher confidence in the reading process. We use computer-based simulations to demonstrate an end-to-end DNA data storage system based on combinatorial shortmer encodings, and study its performance. Finally, we discuss the potential of combinatorial encoding schemes and the future work required to enable these schemes in DNA-based data storage systems.

2 Results

2.1 Shortmer combinatorial encoding for DNA storage

To get an improved capacity in DNA-based storage systems, we suggest a novel method that not only effectively extends the DNA alphabet, it also ensures a near-zero error rate. This encoding scheme is based on the following principles:

1. **A set Ω of DNA k-mers** that will serve as building blocks for the encoding scheme. Elements in Ω are designed to be sufficiently different from each other to minimize mix-up error probability. Hence, the set is designed to satisfy $d(X_i, X_j) \geq d$, so that $P(read\ X_i | write\ X_j) \leq \alpha, \forall X_i, X_j \in \Omega, i \neq j$, where $d(x, y)$ is the Hamming distance between x and y. Note that $N = |\Omega| \leq 4^k$. Table 4 demonstrates several examples of such sparse sets.
2. **DNA synthesis using the k-mers in Ω**, which can be non-standard reagents. This is compatible with current DNA synthesis methods. This synthesis mixtures of k-mers from Ω in each position, i.e., similar to the synthesis of composite DNA letters [9].
3. **A large combinatorial alphabet Σ** in which every letter is defined by a subset S of the k-mers in Ω. A letter $\sigma \in \Sigma$ (representing a subset S) is an N-dimensional binary vector, and the indices $\sigma_i = 1, 1 \leq i \leq N$ represent the k-mers included in the subset S. For example, $\sigma = (0,1,0,1,1,1)$ means that $S = \{X_2, X_4, X_5, X_6\}$ (and $|\Omega| = N = 6$). To write a combinatorial letter σ in a specific position, we synthesize a mixture of the four k-mers included in the subset S. To infer a combinatorial letter σ, a set of reads needs to be analyzed to determine which k-mers are observed (See Sections 2.2 and 2.3, for more details).
4. **DNA barcodes** that allow the grouping of reads representing the same sequence, for the inference of the combinatorial sequence. Our barcode length will be predefined as *bc*, representing optional barcodes in $\{A, C, G, T\}^{bc}$.

The extended combinatorial alphabets allow for higher logical density of the DNA-based storage system, while minimizing error rates. Figure 1 depicts a complete workflow of DNA-based storage with combinatorial shortmer encoding. This includes the following steps:

(i) **Combinatorial message encoding.** A binary message is encoded using a large k-mer combinatorial alphabet (e.g., trimer-based alphabet of size $|\Sigma| = 4096$ letters, with $N = |\Omega| = 16$). The complete message is broken into sequences of set length, each sequence is then marked with a standard DNA barcode and translated using the table presented in the Encode Legend (See Section 2.2, for details about the Binomial Encoding).

(ii) **Error correction.** 2D Systematic Reed-Solomon (RS) encoding is used for error correction. First, the barcode is encoded using RS(6,8) over $GF(2^4)$ and the payload is encoded using RS(120,134) over $GF(2^{12})$. Next, the columns of each block of 42 sequences are encoded using RS(42,48) over $GF(2^{12})$ [13] [14].

(iii) **DNA synthesis.** DNA molecules pertaining to the designed sequences are synthesized using combinatorial k-mer DNA synthesis (See Figure 2). In each position of the payload, each DNA molecule will contain one of the trimers included in the designed combinatorial letter σ.

(iv) **DNA sequencing.** DNA molecules are sequenced using any DNA sequencing technology.

(v) **Combinatorial sequence reconstruction**. The sequenced reads are grouped by the barcode sequence, and the combinatorial sequences are recovered (See Sections 2.3 and 3.1).

(vi) **Error correction decoding.** The recovered sequences are decoded to correct errors. The barcode, payload, and each block of sequences (48, in our example) are decoded separately.

(vii) **Binary message decoding.** The resulting combinatorial sequence is decoded back to a binary message.

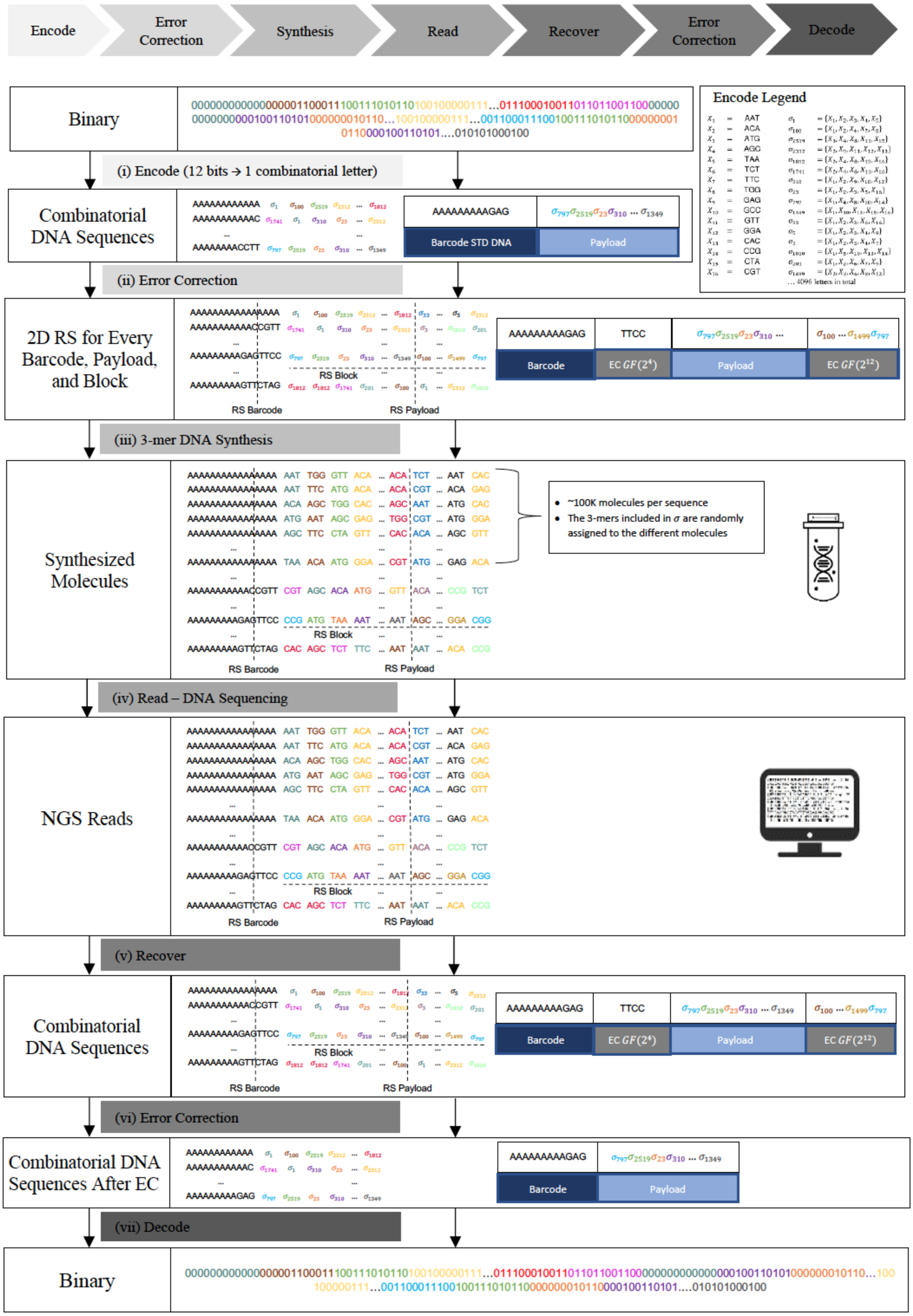

Encode
Error Correction
Synthesis
Read
Recover
Error Correction
Decode
Binary
(i) Encode (12 bits → 1 combinatorial letter)
Combinatorial DNA Sequences
Barcode STD DNA
Payload
Encode Legend
4096 letters in total
(ii) Error Correction
2D RS for Every Barcode, Payload, and Block
RS Block
RS Barcode
RS Payload
Barcode
EC $GF(2^4)$
EC $GF(2^{12})$
(iii) 3-mer DNA Synthesis
Synthesized Molecules
~100K molecules per sequence
The 3-mers included in σ are randomly assigned to the different molecules
(iv) Read – DNA Sequencing
NGS Reads
(v) Recover
Combinatorial DNA Sequences
(vi) Error Correction
Combinatorial DNA Sequences After EC
(vii) Decode
Binary

Figure 1. Workflow of a large-scale combinatorial trimer DNA-based data storage system. *The workflow consists of the following steps: (i) Binary to combinatorial letter encoding. (ii) 2D RS error correction encoding. (iii) Combinatorial shortmer DNA synthesis. (iv) DNA sequencing. (v) Inference of combinatorial letters. (vi) Decoding of the 2D RS error correction code. (vii) Combinatorial letter to binary decoding.*

The combinatorial shortmer encoding scheme is potentially based on using the standard phosphoramidite chemistry synthesis technology, with some alterations [15] [16] [17] [18] [19] [20]. Figure 2 demonstrates the required alterations:

1. **The use of DNA shortmers as building blocks.** The DNA print head will be fed from more than four cartridges, each containing a different building block (from Ω).
2. **Synthesis of mixed building blocks in a single cycle.** Each cycle in the synthesis process will be divided into two steps. First, all the desired building blocks will be added to a designated mixing chamber, and only then will the mixture of all desired building blocks be introduced (e.g, by injection) to the elongating molecules.

Specifically, Figure 2 exemplifies the synthesis of a message encoded using binomial encoding, with $N = 16, K = 3, bc = 2$. Note the $N + 4 = 20$ (4 for the standard nucleotides) cartridges on the print head and the designated mixing chamber. The synthesized (combinatorial) DNA sequence is AT$\sigma_{39}\sigma_{270}$, with AT being the barcode and $\sigma_{39}\sigma_{270}$ the payload $\sigma_{39} = \{AAT, AGC, CGT\}, \sigma_{270} = \{CAC, ATG, CTA\}$. As Figure 2 shows, first, the standard DNA letter barcode with length $= 2$ (namely AT here) is synthesized (steps 1-5). Next, a single combinatorial letter is synthesized (steps 6-10). This includes K steps of adding trimers to the mixing chamber (steps 6-8), mixing (step 9) and, finally, phosphoramidite elongation (step 10). Then, a second combinatorial letter is synthesized (steps 11-15). At the end of the process, we can cleave the synthesized molecules from the surface (step 16).

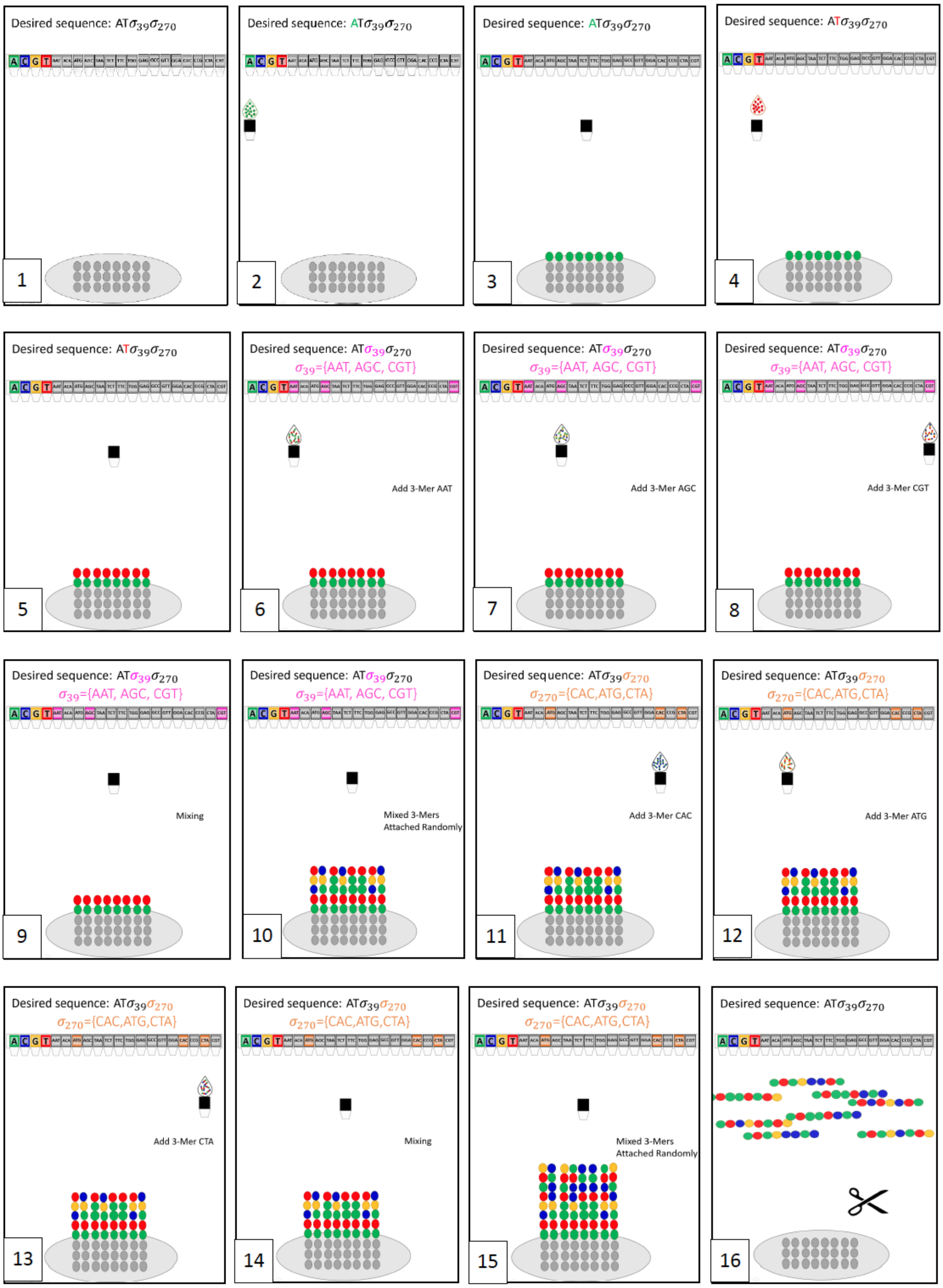

Desired sequence: AT$\sigma_{39}\sigma_{270}$
σ_{39}={AAT, AGC, CGT}
σ_{270}={CAC,ATG,CTA}
A C G T AAT ACA ATG AGC TAA TCT TTC TGG GAG GCC GTT GGA CAC CCG CTA CGT
Add 3-Mer AAT
Add 3-Mer AGC
Add 3-Mer CGT
Mixing
Mixed 3-Mers Attached Randomly
Add 3-Mer CAC
Add 3-Mer ATG
Add 3-Mer CTA
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16

Figure 2. Combinatorial shortmer synthesis. *Synthesis starts with a linker connected to a solid surface (1). For barcode synthesis, standard DNA bases are injected, and the synthesized molecules elongate (2-5). For the synthesis of combinatorial trimers (6-10), each cycle consists of the following steps: each participating trimer is injected into the mixing chamber (6,7,8), then all the desired trimers are mixed together (9), and finally introduced, by injection, to the elongating molecules (10). The process repeats for the next combinatorial letter (11-15), and finally the resulting molecules are cleaved and collected (16).*

2.2 Binary and binomial combinatorial alphabets

The main parameter that defines a combinatorial encoding scheme is the alphabet Σ. More specifically, the set of valid subsets of Ω that can be used as letters. Here we define two general approaches for the construction of Σ. Namely, the *binomial encoding* and the *binary encoding*. We start with the former.

In the *binomial encoding* scheme, only subsets of Ω of size exactly K represent valid letters in Σ, so that every letter $\sigma \in \Sigma$ consists of exactly K k-mers. This yields an effective output alphabet of size $|\Sigma| = \binom{N}{K}$ letters. Therefore, all the letters in the alphabet have the same Hamming weight - $w(\sigma) = K, \forall \sigma \in \Sigma$. This way, an r-bit binary message will require $\frac{r}{\log_2(|\Sigma|)}$ synthesis cycles (and a DNA molecule with length $\frac{kr}{\log_2(|\Sigma|)}$). Conversely, every single letter in the output alphabet encodes $\log_2\binom{N}{K} = \lfloor \log_2(|\Sigma|) \rfloor$ bits. Note that this calculation ignores the overhead caused by the error correction redundancy.

For example, consider a binary message encoded using a binomial encoding alphabet Σ, derived from using trimer building blocks, a set Ω of size $N = 10$ and $K = 5$ distinct trimers for each letter in Σ. In this case the alphabet size is $|\Sigma| = \binom{10}{5} = 252$. To fit blocks of bits, a subset of 128 letters is used. The basic block capacity encodes 7 bits in one single letter. A message of length $r = 10KB$ can be synthsis with $\frac{10KB}{\log_2 128} < 12{,}000$ synthsis cycles. If we use trimers and 120 synthesis cycles to yield oligos of length $120 * 3 = 360nt$, we will need 100 such oligos.

Table 1 demonstrates the binomial alphabet (also used in Figure 1) with $N = |\Omega| = 16$ and $\mathrm{K} = |\mathrm{S}| = 5$. The effective alphabet has $|\Sigma| = \binom{16}{5} = 4{,}368$ letters. Since $4096 \leq 4368$, each synthesized letter encodes 12 bits.

trimers, *Ω*	**Alphabet *Σ***
X_1 = AAT	$\sigma_1 = \{X_1, X_2, X_3, X_4, X_5\}$
X_2 = ACA	$\sigma_{100} = \{X_1, X_2, X_4, X_7, X_8\}$
X_3 = ATG	$\sigma_{2519} = \{X_3, X_4, X_8, X_{11}, X_{15}\}$
X_4 = AGC	$\sigma_{2312} = \{X_2, X_9, X_{11}, X_{12}, X_{13}\}$
X_5 = TAA	$\sigma_{1812} = \{X_2, X_4, X_8, X_{13}, X_{16}\}$
X_6 = TCT	$\sigma_{1741} = \{X_2, X_4, X_6, X_{11}, X_{16}\}$
X_7 = TTC	$\sigma_{310} = \{X_1, X_2, X_9, X_{10}, X_{12}\}$
X_8 = TGG	$\sigma_{23} = \{X_1, X_2, X_3, X_5, X_{16}\}$
X_9 = GAG	$\sigma_{797} = \{X_1, X_4, X_8, X_{10}, X_{14}\}$
X_{10} = GCC	$\sigma_{1349} = \{X_1, X_{10}, X_{13}, X_{15}, X_{16}\}$
X_{11} = GTT	$\sigma_{33} = \{X_1, X_2, X_3, X_6, X_{16}\}$
X_{12} = GGA	$\sigma_5 = \{X_1, X_2, X_3, X_4, X_9\}$
X_{13} = CAC	$\sigma_3 = \{X_1, X_2, X_3, X_4, X_7\}$
X_{14} = CCG	$\sigma_{1010} = \{X_1, X_5, X_{10}, X_{13}, X_{14}\}$
X_{15} = CTA	$\sigma_{201} = \{X_1, X_2, X_6, X_7, X_9\}$
X_{16} = CGT	$\sigma_{1499} = \{X_2, X_3, X_6, X_8, X_{12}\}$
	… 4096 letters in total

Table 1. Example of binomial shortmer alphabet. $\boldsymbol{N = |\Omega| = 16, K = 5.}$
The Hamming distance of this $\boldsymbol{\Omega}$ *is* $\boldsymbol{d = 2.}$ $\boldsymbol{|\Sigma| = 4096 \leq 4368 = \binom{16}{5}}$

In the *binary encoding* scheme, all possible nonempty subsets of Ω represent valid letters in the alphabet. This yields an effective alphabet of size $|\Sigma| = 2^N - 1$ letters. This way, an r-bit binary message will require $\frac{r}{\log_2(|\Sigma|)}$ synthesis cycles (yielding a DNA molecule of length $\frac{kr}{\log_2(|\Sigma|)}$). Every single letter in the output alphabet encodes $N - 1 = \lfloor \log_2(|\Sigma|) \rfloor$ bits. Note that this calculation ignores the overhead caused by the error correction redundancy.

For example, consider a binary message encoded using a binary encoding alphabet Σ, derived from using a trimer building blocks, a set Ω of size $N = 10$. In this case, the alphabet size is $|\Sigma| = 2^{10} - 1 = 1023$. The basic block capacity encodes 9 bits in every letter.

2.3 Reconstruction probabilities for binomial encoding

With binomial encoding, it is possible to collect reads and stop after observing all the K distinct k-mers at every position. This simplifies the analysis of the reconstruction probability and reduces error rates, making it the preferred combinatorial encoding system.

Since every letter $\sigma \in \Sigma$ consists exactly of the K participating k-mers, the required number of reads for observing at least one read of each k-mer follows the coupon collector distribution [21]. The number of reads required to achieve this goal can be described as a random variable $R = \sum_{i=1}^{K} R_i$ where $R_1 = 1$ and $R_i \sim Geom\left(\frac{K-i+1}{K}\right), i = 2, \ldots, K$. The expected number of required reads is then:

$$E[R] = \sum_{i=1}^{K} E[R_i] = K \sum_{i=1}^{K} \frac{1}{i} \cong Klog(K)$$

The expected number of reads required for observing all the participating k-mers remains reasonable for the relevant values of K.

Using the independence of R_i we can derive that $Var(R) = \sum_{i=1}^{K} Var(R_i) < \frac{\pi^2}{6} K^2$. By Chebyshev inequality, we get an upper bound (a loose bound) on the probability of requiring more than $E[R] + cK$ reads to observe at least one read of each k-mer:

$$P(|R - E[R]| \geq cK) \leq \frac{\pi^2}{6c^2}$$

$$P(|R - Har(\mathrm{K})| \geq cK) \leq \frac{\pi^2}{6c^2}$$

When we examine an entire read of length l, assuming independence and not taking error correction into account, we get the following relationship between c and any desired confidence level $1 - \delta$:

$$P(|R(l) - Har(\mathrm{K})| \geq cK) \leq 1 - \left(1 - \frac{\pi^2}{6c^2}\right)^{l} < \delta$$

$$P(\mathrm{R}(l) < Har(\mathrm{K}) + cK) \geq \left(1 - \frac{\pi^2}{6c^2}\right)^{l} \geq 1 - \delta$$

Table 2 presents several examples for the loose upper bound, derived as above, on the number of reads required to $1 - \delta$ ensure reconstruction of a binomial message with $K = 5$, sequence of length l. As demonstrated in the simulations, these numbers are definitely not tight (Section 2.4).

		Error probability (δ)			
Sequence length (l)		10^{-2}	10^{-3}	10^{-4}	10^{-5}
	100	*642*	*2030*	*6415*	*20282*
	150	*786*	*2486*	*7857*	*24839*

Table 2. Upper bounds on the required number of reads to reconstruct a binomial sequence

Clearly, for reasonable values of K (i.e., $K \leq 10$), a standard coverage of 100 reads per oligos yields low probabilities for missing one of the included k-mers. Note that with an online sequencing technology (i.e., nanopore sequencing) we can simply keep sequencing until K distinct k-mers have been confidently observed. The above bounds will then provide an estimate on the sequencing cost.

To take into account the probability of observing a k-mer that is not included in the designed set of K k-mers, we can require at least $m > 1$ reads of each of the K k-mers to be observed. In this case, the derivation of the number of required reads is not as trivial, but is expected to be approximated by $E[R] \cong K(\log(K) + m \log\log(K))$ [21]. Again, we obtain reasonable numbers for relevant values of K and m, and can use the selective nanopore approach to guide the process, avoiding reconstruction issues.

This analysis is based on neglecting mix-up errors (i.e., there are no incorrect k-mer readings). This assumption is based on the near-zero mix-up probability that is attained by the construction of Ω with a minimal Hamming distance, see Table 4.

2.4 Simulation of an end-to-end combinatorial shortmer storage system

To demonstrate our suggested encoding approach, we created an in-silico end-to-end system based on combinatorial shortmer encoding, simulated combinatorial DNA synthesis, simulated DNA sequencing, and message decoding (See Section 3). We simulated systems with different binomial alphabets and error probabilities, and measured the resulting reconstruction and decoding rates. Figure 3 depicts a schematic representation of our simulation workflow and indicates how the error rates are calculated.

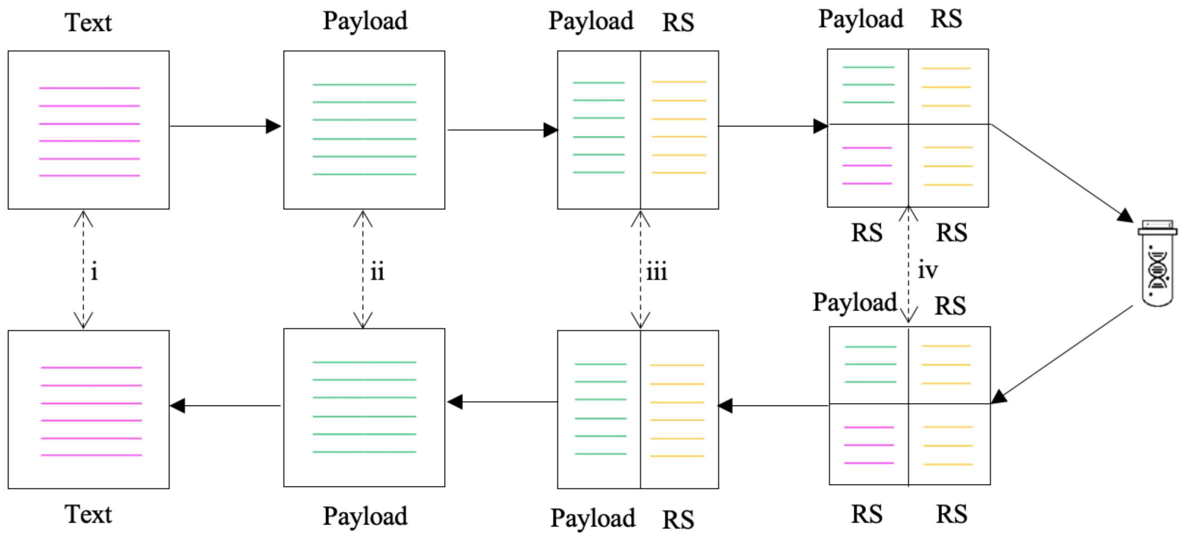


Figure 3. A schematic view of the simulation workflow. *A 10KB text message (1) is translated into a combinatorial message (2). Barcode and payload RS error correction is added to each sequence (3). Every block is encoded using RS error correction (4). DNA synthesis and sequencing is simulated. Combinatorial sequences are reconstructed from the sequencing reads (5). RS decoding is performed on each block (6) and each sequence (7). Finally, the combinatorial message is translated back to a text message (8). Errors are calculated on the text files (i), the "clean" payloads (ii), and the payloads with RS (iii, iv).*

The results of the simulation runs are summarized in Figure 4 and Figure 5. Each run included 30 repeats with random input texts of 10KB each. The results presented are for an alphabet of size $|\Sigma| = 4{,}096$ ($N = |\Omega| = 16$, $K = 5$), barcode length of 12nt with 4nt of RS, payload of length 120 trimers with payload RS 14 trimers. The crosswise RS consists of 6 trimers for every block of 42 payloads, in any given position. From each barcode, 1000 copies were synthesized (simulated). Errors are then simulated into the resulting sequences, to represent synthesis and sequencing errors, as expected in actual usage [22].

The following is a summary of the conclusions from the simulation study:

- As expected, higher synthesis and sequencing error probabilities produce a lower reconstruction rate.
- Smaller samples of 10 and 20 reads per barcode did not allow for full reconstruction, even with zero error rate. Also, increasing the sampling rate results in better reconstruction. This demonstrates the crucial effect of random sampling on the overall performance of the system.
- Substitution errors are easier to detect and correct than deletion and insertion errors. This is because substitution errors affect the nucleotide level rather than the trimer level. The minimal Hamming distance $d = 2$ of the trimer set Ω allows for the correction of single-base substitutions.
- 2D RS error correction significantly improved reconstruction rates.

(ii) Rate of full combinatorial message reconstruction. Before RS decoding.

(iii) Rate of full combinatorial message reconstruction. After payload RS decoding.

(iv) Rate of full combinatorial message reconstruction. After 2D RS decoding.

(i) Rate of full text reconstruction. After 2D RS decoding.

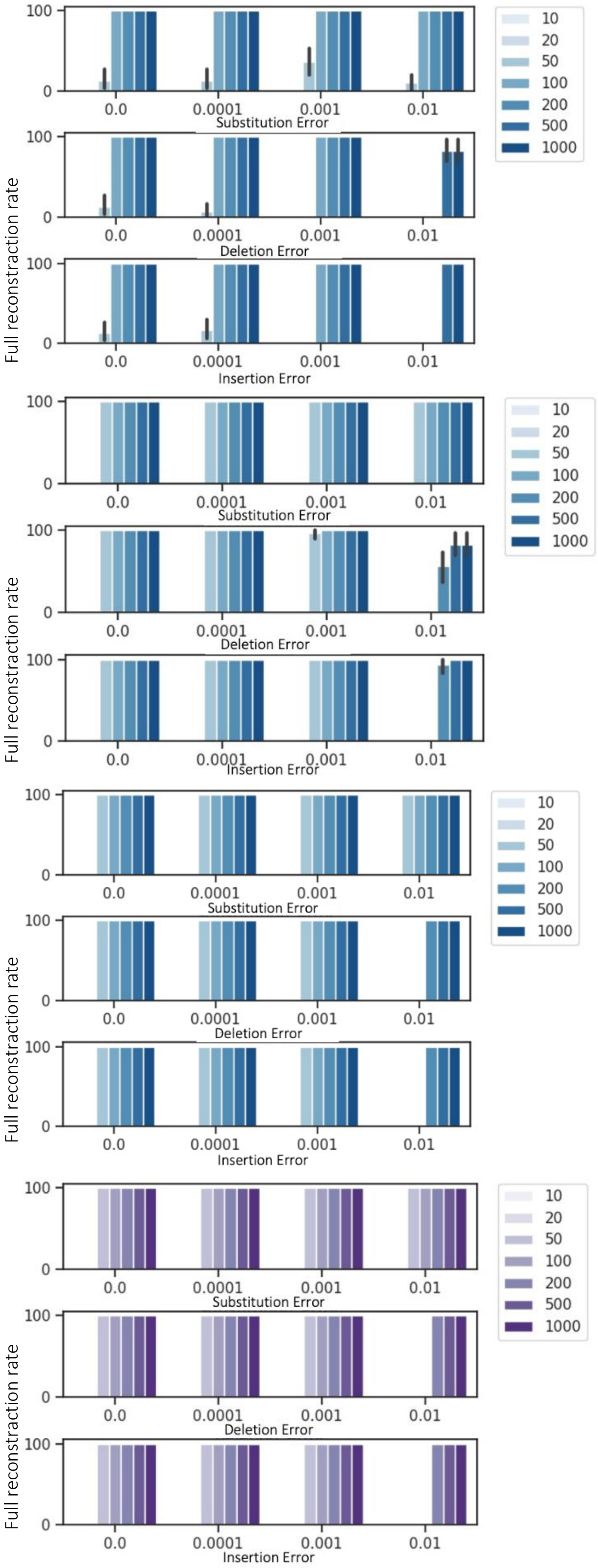
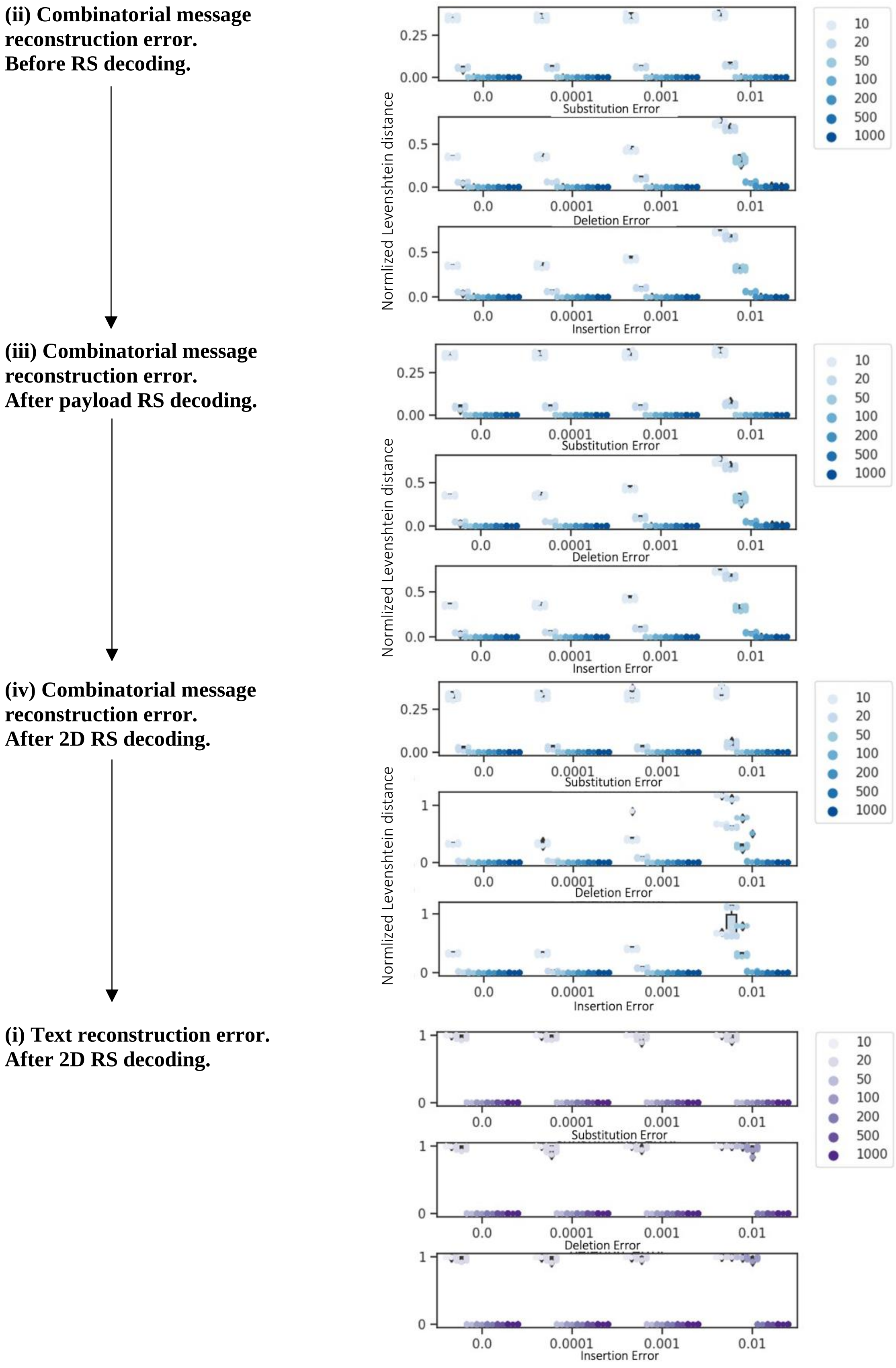


Figure 4. Simulation of the entire binomial process, full reconstruction rate. Full reconstruction rate, calculated on simulated data with different error types (substitution, insertion, and deletion) and error rates (0, 0.0001, 0.001, 0.01, respectively), and different sampling depths (10 – 1000 reads on average per barcode). Different stages shown: (ii) Before RS decoding. (iii) After payload RS decoding. (iv) After 2D RS decoding. (i) Text After 2D Rs Decoding.



(ii) Combinatorial message reconstruction error. Before RS decoding.

(iii) Combinatorial message reconstruction error. After payload RS decoding.

(iv) Combinatorial message reconstruction error. After 2D RS decoding.

(i) Text reconstruction error. After 2D RS decoding.

Figure 5. Simulation of the entire binomial process, normalized Levenshtein distance.
Similar to Figure 4. Levenshtein distance normalized by the length of the original message.

3 Methods

3.1 Implementation of a combinatorial k-mer storage system

In our first run of simulations encoding/storing/decoding, we will address some of the process parameters. Table 3 presents a message encoding example.
The data will be encoded using molecules of length 418nt. Each molecule will contain 16nt barcode bases (out of which 4nt are dedicated for RS error correction), 402nt payload bases representing 402nt/3=134 trimers (out of which 14 trimers are dedicated for RS error correction). To protect from sequence dropouts, every 48 molecules are treated as a single block for RS error correction, to protect from sequence dropout).

1. Encoding

1.1. **Data padding.** To fit the binary data onto the molecule, it must be divided by the molecule size and the block size. If the division results in a gap, the data is padded with zeros to close this gap.

1.2. **2D-error correction using Reed-Solomon decoding.** Reed Solomon (RS) is used a total of three times. It is applied lengthwise on each sequence twice, error correcting each barcode sequence and then each payload sequence. It is also used crosswise on all the sequences in one block size. (See Section 3.2).

2. Synthesis and sequencing

2.1. **Simulating the synthesis process.** The synthesis of each combinatorial sequence was simulated separately. For a fixed sequence we first draw, from $X \sim N(\mu = \text{predtermined}, \sigma^2 = 100)$, the number of molecules that will represent it. Let this number be x. All k-mers that occur within a single position (cycle) are then generated. To do this, x numbers of the subset are selected, representing the relevant σ. The size of this subset is K, and its members will most likely be represented many times. This random composition is achieved by drawing a total of x independent times, according to $Y \sim U(1, K)$. 1s in σ are indexed at $1, \dots, K$, and the appropriate k-mers are "synthesized" in accordance with the drawn index.

2.2. **Mixing.** Once all of the molecules are synthesized, they are mixed to mimic real molecules in a container.

2.3. **Error simulation.** To replicate a real synthesis and sequencing process, several error scenarios were simulated. These include the three error types Deletion, Insertion, and Substitution of a letter in the sequence, each predefined by an error percentage. A Bernoulli trial is then performed on every sequence and letter position, where P is the predefined error, inserting the errors in each position. To replicate the Substitution error, we implemented the error per nucleotide, and for the Insertion and Deletions errors we implemented the error on each full k-mer. This method is closest to the expected error scenarios in combinatorial DNA synthesis and sequencing.

2.4. **Reading and sampling.** Several different samples were drawn, to analyze their impact on the accuracy of the data retrieved.

3. Decoding

3.1. **Sequence retrieval**. To retrieve the original sequence, first each sequence barcode undergoes RS error correction. Next, each sequence payload is reviewed individually, and undergoes RS, too. For sequences in the same block, RS is also done, crosswise on the block.

3.2. **Grouping by barcode and determining σ in each position**. Once barcode retrieval is complete, sequences are grouped by the same barcode. In each of the groups, all the sequences are reviewed at the same exact position, where we extract the K most common k-mers to determine the σ in that position. In the process of determining the K most common k-mers, we may encounter invalid k-mer (not in Ω). Should an invalid k-mer be encountered in the payload sequence, the following steps are taken:

- **If the length of the sequence is equal to the predetermined length.** The sequence is reviewed, and if an invalid k-mer is encountered, which is not part of our alphabet, an X_{dummy} is inserted instead, followed by skipping 3nts.
- **If the length of the sequence is smaller than the predetermined length.** When Δ<SL*, it indicates that there is a deletion in the sequence. We pad it with a dummy nucleotide R that restores it to the predetermined length, and then review the sequence.
- **If the length of the sequence is greater than the predetermined length.** When Δ>SL*, it indicates that there is an insertion in the sequence. We cut the sequence to restore it to the predetermined length, and then review the sequence.

  $* \Delta - Current\ sequence\ length, SL - Predetermined\ sequence\ length$

3.3. **Missing barcode.** After the reading process is complete, if a barcode is found missing, the missing barcode and a dummy sequence is added to enable Reed Solomon to retrieve the data correctly.

4. Validation

4.1. **Levenshtein distance.** After recovering the data, the magnitude of the error in the information recovery process is reviewed, by assessing the Levenshtein distance between the input I and output O [23] [24]. The normalized Levenshtein distance is shown in our graphs in Figure 4 and Figure 5.

3.2 Data Padding and Error Correction Using 2D Reed Solomon codes

The binary message is first padded, so that its total length is divisible by r (r is the number of bits per letter). Next it is encoded into a combinatorial message. The combinatorial message is broken down into sequences of length l, and another padding is done to complete a block of $B-1$ full sequences. The padding information is included in the final single combinatorial sequence to complete a block of B sequences.

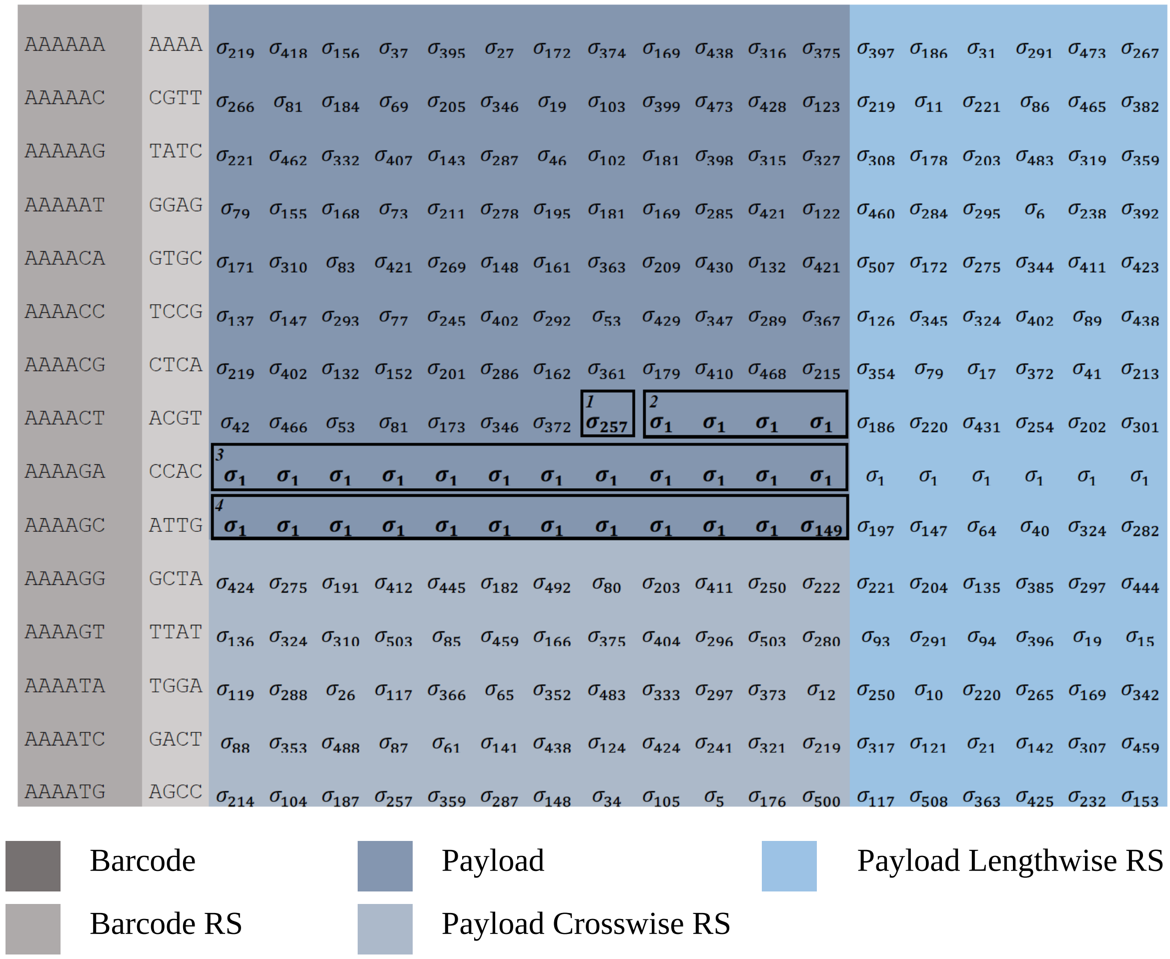
For the barcode sequence, a systematic (6, 8) RS code over $GF(2^4)$ was used to transform the unique 12nt barcode to a 16nt sequence. The 120 combinatorial letter payload sequence was encoded using a (120, 134) RS code over $GF(2^9)$, $GF(2^{12}), GF(2^{13})$ for the binomial alphabets $\binom{16}{3}, \binom{16}{5}, \binom{16}{7}$, respectively. This resulted in sequences of length 134 combinatorial letters + 16nt barcodes. Using trimers, the (theoretical) resulting oligos have overall length of 418nt.

To protect against sequence dropouts, we used RS error correction on the columns of the matrix (See Figure 6). In each block of 42 sequences, we apply a (42,48) RS code over $GF(2^9)$, $GF(2^{12}), GF(2^{13})$ for the binomial alphabets $\binom{16}{3}, \binom{16}{5}, \binom{16}{7}$, respectively. This is applied in each column separately.

Figure 6 demonstrates the encoding of ~0.1 KB using the following parameters:

- A (3,5) RS code over $GF(4^2)$ for the barcodes.
- A (12,18) RS code over $GF(2^9)$ for the $\binom{16}{3}$ binomial alphabet payload sequence.
- A 10-sequence block encoded, column wise, using a (10,15) RS code over $GF(2^9)$.

The 824 bits are first padded to be $828 = 92 * 9$. The 92 combinatorial letter message is split into 7 sequences of 12 letters and an additional sequence of 8 letters. Finally, a complete block of 12 sequences (total of $12 * 12 = 144$ letters) is created by padding with one additional sequence of 12 letters and including the padding information as the last sequence.



AAAAAA	AAAA	σ_{219}	σ_{418}	σ_{156}	σ_{37}	σ_{395}	σ_{27}	σ_{172}	σ_{374}	σ_{169}	σ_{438}	σ_{316}	σ_{375}	σ_{397}	σ_{186}	σ_{31}	σ_{291}	σ_{473}	σ_{267}
AAAAAC	CGTT	σ_{266}	σ_{81}	σ_{184}	σ_{69}	σ_{205}	σ_{346}	σ_{19}	σ_{103}	σ_{399}	σ_{473}	σ_{428}	σ_{123}	σ_{219}	σ_{11}	σ_{221}	σ_{86}	σ_{465}	σ_{382}
AAAAAG	TATC	σ_{221}	σ_{462}	σ_{332}	σ_{407}	σ_{143}	σ_{287}	σ_{46}	σ_{102}	σ_{181}	σ_{398}	σ_{315}	σ_{327}	σ_{308}	σ_{178}	σ_{203}	σ_{483}	σ_{319}	σ_{359}
AAAAAT	GGAG	σ_{79}	σ_{155}	σ_{168}	σ_{73}	σ_{211}	σ_{278}	σ_{195}	σ_{181}	σ_{169}	σ_{285}	σ_{421}	σ_{122}	σ_{460}	σ_{284}	σ_{295}	σ_{6}	σ_{238}	σ_{392}
AAAACA	GTGC	σ_{171}	σ_{310}	σ_{83}	σ_{421}	σ_{269}	σ_{148}	σ_{161}	σ_{363}	σ_{209}	σ_{430}	σ_{132}	σ_{421}	σ_{507}	σ_{172}	σ_{275}	σ_{344}	σ_{411}	σ_{423}
AAAACC	TCCG	σ_{137}	σ_{147}	σ_{293}	σ_{77}	σ_{245}	σ_{402}	σ_{292}	σ_{53}	σ_{429}	σ_{347}	σ_{289}	σ_{367}	σ_{126}	σ_{345}	σ_{324}	σ_{402}	σ_{89}	σ_{438}
AAAACG	CTCA	σ_{219}	σ_{402}	σ_{132}	σ_{152}	σ_{201}	σ_{286}	σ_{162}	σ_{361}	σ_{179}	σ_{410}	σ_{468}	σ_{215}	σ_{354}	σ_{79}	σ_{17}	σ_{372}	σ_{41}	σ_{213}
AAAACT	ACGT	σ_{42}	σ_{466}	σ_{53}	σ_{81}	σ_{173}	σ_{346}	σ_{372}	1 $\boldsymbol{\sigma_{257}}$	2 $\boldsymbol{\sigma_{1}}$	$\boldsymbol{\sigma_{1}}$	$\boldsymbol{\sigma_{1}}$	$\boldsymbol{\sigma_{1}}$	σ_{186}	σ_{220}	σ_{431}	σ_{254}	σ_{202}	σ_{301}
AAAAGA	CCAC	3 $\boldsymbol{\sigma_{1}}$	$\boldsymbol{\sigma_{1}}$	$\boldsymbol{\sigma_{1}}$	$\boldsymbol{\sigma_{1}}$	$\boldsymbol{\sigma_{1}}$	$\boldsymbol{\sigma_{1}}$	$\boldsymbol{\sigma_{1}}$	$\boldsymbol{\sigma_{1}}$	$\boldsymbol{\sigma_{1}}$	$\boldsymbol{\sigma_{1}}$	$\boldsymbol{\sigma_{1}}$	$\boldsymbol{\sigma_{1}}$	σ_{1}	σ_{1}	σ_{1}	σ_{1}	σ_{1}	σ_{1}
AAAAGC	ATTG	4 $\boldsymbol{\sigma_{1}}$	$\boldsymbol{\sigma_{1}}$	$\boldsymbol{\sigma_{1}}$	$\boldsymbol{\sigma_{1}}$	$\boldsymbol{\sigma_{1}}$	$\boldsymbol{\sigma_{1}}$	$\boldsymbol{\sigma_{1}}$	$\boldsymbol{\sigma_{1}}$	$\boldsymbol{\sigma_{1}}$	$\boldsymbol{\sigma_{1}}$	$\boldsymbol{\sigma_{1}}$	$\boldsymbol{\sigma_{149}}$	σ_{197}	σ_{147}	σ_{64}	σ_{40}	σ_{324}	σ_{282}
AAAAGG	GCTA	σ_{424}	σ_{275}	σ_{191}	σ_{412}	σ_{445}	σ_{182}	σ_{492}	σ_{80}	σ_{203}	σ_{411}	σ_{250}	σ_{222}	σ_{221}	σ_{204}	σ_{135}	σ_{385}	σ_{297}	σ_{444}
AAAAGT	TTAT	σ_{136}	σ_{324}	σ_{310}	σ_{503}	σ_{85}	σ_{459}	σ_{166}	σ_{375}	σ_{404}	σ_{296}	σ_{503}	σ_{280}	σ_{93}	σ_{291}	σ_{94}	σ_{396}	σ_{19}	σ_{15}
AAAATA	TGGA	σ_{119}	σ_{288}	σ_{26}	σ_{117}	σ_{366}	σ_{65}	σ_{352}	σ_{483}	σ_{333}	σ_{297}	σ_{373}	σ_{12}	σ_{250}	σ_{10}	σ_{220}	σ_{265}	σ_{169}	σ_{342}
AAAATC	GACT	σ_{88}	σ_{353}	σ_{488}	σ_{87}	σ_{61}	σ_{141}	σ_{438}	σ_{124}	σ_{424}	σ_{241}	σ_{321}	σ_{219}	σ_{317}	σ_{121}	σ_{21}	σ_{142}	σ_{307}	σ_{459}
AAAATG	AGCC	σ_{214}	σ_{104}	σ_{187}	σ_{257}	σ_{359}	σ_{287}	σ_{148}	σ_{34}	σ_{105}	σ_{5}	σ_{176}	σ_{500}	σ_{117}	σ_{508}	σ_{363}	σ_{425}	σ_{232}	σ_{153}

Figure 6. Example of message coding including padding and Reed-Solomon error correction. *Encoding of a ~0.1KB message to a 4096 letter binomial alphabet* ($\boldsymbol{N}=\mathbf{16}, \boldsymbol{K}=\mathbf{3}$)*. (i) First, bit padding is added, included here in the letter* $\overset{1}{\boxed{\sigma_{257}}}$*. (ii) Next, block padding is added, included here in* $\overset{2}{\boxed{\sigma_{1}}}$ *and* $\overset{3}{\boxed{\sigma_{1}}}$ *(iii) Padding information is included in the last sequence of all blocks. The last sequence holds the number of padding binary bits. In this example,* $\overset{4}{\boxed{\sigma_{149}}}$ *represents 148 bits of padding, composed of* $\mathbf{4}+(\mathbf{4}*\mathbf{9})+(\mathbf{12}*\mathbf{9})\ \boldsymbol{bits}$ *, 4 bits from* $\overset{1}{\boxed{\sigma_{257}}}$*, 4 letters from* $\overset{2}{\boxed{\sigma_{1}}}$ *and 12 letters from* $\overset{3}{\boxed{\sigma_{1}}}$*.*

4 Discussion

4.1 Information capacities for selected encodings

Table 3 demonstrates the encoding of a 1GB input file, with standard encoding and two combinatorial encoding schemes, binomial and binary, using six different alphabets. In the binomial encoding scheme, three different trimer alphabets of sizes $\binom{16}{3}, \binom{16}{5}, \binom{16}{7}$ were used. In the binary encoding scheme, three different trimer sets of sizes 10, 16, 20 were used. All calculations are based on error correction parameters similar to those previously described [9] [10] [2] [14].

With these alphabets, up to 9.5-fold and 6.5-fold increase in information capacity is achieved per synthesis cycle and per DNA base respectively, compared with standard DNA based storage. We note that the 6.5 increase obtained using the binary coding also guarantees near zero reconstruction error.

Type	$\boldsymbol{N}$	$\boldsymbol{K}$	$\binom{N}{K}$	$\boldsymbol{2^N - 1}$	Bits per Letter	Alphabet Size	Bits per Sequence	Number of Sequences	Reed Solomon (RS)	Bits per Synthesis Cycle, Payload Only	Bits per Synthesis Cycle	Fold Increase
Standard					2	4	240	33,333,334	38,095,248	1.57	1.40	1.0
Binomial	16	3	560		9	512	1,080	7,407,408	8,465,616	7.05	6.30	4.5
Binomial	16	5	4,368		12	4,096	1,440	5,555,556	6,349,248	9.40	8.40	6.0
Binomial	16	7	11,440		13	8,192	1,560	5,128,206	5,860,848	10.19	9.10	6.5
Binary	10			1,023	9	512	1,080	7,407,408	8,465,616	7.05	6.30	4.5
Binary	16			65,535	15	32,768	1,800	4,444,445	5,079,408	11.75	10.50	7.5
Binary	20			1,048,575	19	524,288	2,280	3,508,772	4,010,064	14.89	13.30	9.5

Table 3. Logical densities for selected encoding schemes. *The numbers represent encoding a 1 GB binary message using oligos with 14nt barcodes +2nt RS (standard DNA), and 120 payload letters with 14 extra RS for the payload (combinatorial with N and K as indicated)*

We briefly touched on analytically evaluating reconstruction errors when using the schemes proposed herein. Further research is required to obtain better estimates of the theoretical reconstruction rates or Levenshtein distances.

Finally, it is important to note that none of the methods described here has been tested in the lab for actual synthesis. This is mostly due to hardware limitations. Modifications to the existing phosphoramidite DNA synthesis machinery needs to include a mixing chamber [25], such as depicted in Figure 2. They should also include access to multiple feeding containers, holding the $N = |\Omega|$ reagents necessary for the desired alphabet design.

5 Bibliography

[1] Church, G.M., Gao, Y. and Kosuri, S., 2012. Next-generation digital information storage in DNA. *Science*, *337*(6102), pp.1628-1628.

[2] Goldman, N., Bertone, P., Chen, S., Dessimoz, C., LeProust, E.M., Sipos, B. and Birney, E., 2013. Towards practical, high-capacity, low-maintenance information storage in synthesized DNA. *Nature*, *494*(7435), pp.77-80.

[3] Bornholt, J., Lopez, R., Carmean, D.M., Ceze, L., Seelig, G. and Strauss, K., 2017. Toward a DNA-based archival storage system. *Ieee Micro*, *37*(3), pp.98-104.

[4] Yazdi, S.H.T., Yuan, Y., Ma, J., Zhao, H. and Milenkovic, O., 2015. A rewritable, random-access DNA-based storage system. *Scientific reports*, *5*(1), pp.1-10.

[5] Erlich, Y. and Zielinski, D., 2017. DNA Fountain enables a robust and efficient storage architecture. *Science*, *355*(6328), pp.950-954.

[6] Organick, L., Ang, S.D., Chen, Y.J., Lopez, R., Yekhanin, S., Makarychev, K., Racz, M.Z., Kamath, G., Gopalan, P., Nguyen, B. and Takahashi, C.N., 2018. Random access in large-scale DNA data storage. *Nature biotechnology*, *36*(3), pp.242-248.

[7] Gabrys, R., Kiah, H.M. and Milenkovic, O., 2017. Asymmetric Lee distance codes for DNA-based storage. *IEEE Transactions on Information Theory*, *63*(8), pp.4982-4995.

[8] Choi, Y., Ryu, T., Lee, A.C., Choi, H., Lee, H., Park, J., Song, S.H., Kim, S., Kim, H., Park, W. and Kwon, S., 2019. High information capacity DNA-based data storage with augmented encoding characters using degenerate bases. *Scientific reports*, *9*(1), pp.1-7.

[9] Anavy, L., Vaknin, I., Atar, O., Amit, R. and Yakhini, Z., 2019. Data storage in DNA with fewer synthesis cycles using composite DNA letters. *Nature biotechnology*, *37*(10), pp.1229-1236.

[10] Anavy, L., Yakhini, Z., and Amit, R., 2021. Molecular data storage systems and methods. *United States of America Patent* US20210141568A1.

[11] Roquet, N., Bhatia, S.P., Flickinger, S.A., Mihm, S., Norsworthy, M.W., Leake, D. and Park, H., 2021. DNA-based data storage via combinatorial assembly. *bioRxiv*. Available: https://www.biorxiv.org/content/10.1101/2021.04.20.440194v1.

[12] Roquet, N., Park, H., and Bhatia, S.P., 2017. Nucleic acid-based data storage. *United States Patent Application Patent* 20180137418.

[13] Blawat, M., Gaedke, K., Huetter, I., Chen, X.M., Turczyk, B., Inverso, S., Pruitt, B.W. and Church, G.M., 2016. Forward error correction for DNA data storage. *Procedia Computer Science*, *80*, pp.1011-1022.

[14] Reed, I.S. and Solomon, G., 1960. Polynomial codes over certain finite fields. *Journal of the society for industrial and applied mathematics*, *8*(2), pp.300-304.

[15] LeProust, E.M., Peck, B.J., Spirin, K., McCuen, H.B., Moore, B., Namsaraev, E. and Caruthers, M.H., 2010. Synthesis of high-quality libraries of long (150mer) oligonucleotides by a novel depurination controlled process. *Nucleic acids research*, *38*(8), pp.2522-2540.

[16] Barrett, M.T., Scheffer, A., Ben-Dor, A., Sampas, N., Lipson, D., Kincaid, R., Tsang, P., Curry, B., Baird, K., Meltzer, P.S. and Yakhini, Z., 2004. Comparative genomic hybridization using oligonucleotide microarrays and total genomic DNA. *Proceedings of the National Academy of Sciences*, *101*(51), pp.17765-17770.

[17] Eleuteri, A., Capaldi, D.C., Cole, D.L. and Ravikumar, V.T., 1999. Oligodeoxyribonucleotide Phosphorothioates: Substantial Reduction of (N-1)-mer Content Through the Use of Trimeric Phosphoramidite Synthons. *Nucleosides, Nucleotides & Nucleic Acids*, *18*(3), pp.475-483.

[18] Yagodkin, A., Azhayev, A., Roivainen, J., Antopolsky, M., Kayushin, A., Korosteleva, M., Miroshnikov, A., Randolph, J. and Mackie, H., 2007. Improved synthesis of trinucleotide phosphoramidites and generation of randomized oligonucleotide libraries. *Nucleosides, Nucleotides, and Nucleic Acids*, *26*(5), pp.473-497.

[19] Randolph, J., Yagodkin, A., Azhayev, A. and Mackie, H., 2008. Codon-based Mutagenesis. In *Nucleic Acids Symposium Series* (Vol. 52, p. 479).

[20] Glen Research. DNA & RNA Nucleosides, Analogs, and Supports," [Online]. Available: https://www.glenresearch.com. [Accessed 10 July 2021].

[21] Ferrante, M. and Saltalamacchia, M., 2014. The coupon collector's problem. *Materials matemàtics*, pp.0001-35.

[22] Sabary, O., Orlev, Y., Shafir, R., Anavy, L., Yaakobi, E. and Yakhini, Z., 2021. SOLQC: Synthetic oligo library quality control tool. *Bioinformatics*, *37*(5), pp.720-722.

[23] Levenshtein, V., 1965. Binary codes capable of correcting spurious insertions and deletion of ones. *Problems of information Transmission*, *1*(1), pp.8-17.

[24] Levenshtein, V.I., 1966, February. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady* (Vol. 10, No. 8, pp. 707-710).

[25] Strickland, E., 2015. DNA manufacturing enters the age of mass production. *IEEE Spectrum*, *53*(1), pp.55-56.

6 Supplement

Table 4 is an example of k-mer sets. To the left, two sets of trimers that have a minimal Hamming distance of 2, with $|\Omega_1| = 16 \; and \; |\Omega_2| = 12$. To the right, a set of 54 6-mers that have a minimal Hamming distance of 4.

trimers		**6-mers**		
$\boldsymbol{\Omega_1}$	$\boldsymbol{\Omega_2}$	$\boldsymbol{\Omega_3}$		
AAT	ACG	TTGACG	CAGTCA	GCATTA
ACA	AAA	AAAAAA	CATGAC	GCCGGC
ATG	AGC	AACCCC	CCAACC	GCTAAT
AGC	ATT	AAGGGG	CCCCAA	GGAAGG
TAA	CAC	AATTTT	CCGGTT	GGCCTT
TCT	CCA	ACACGT	CCTTGG	GGGGAA
TTC	GAG	ACCATG	CGATAT	GGTTCC
TGG	GCC	ACGTAC	CGCGCG	GTACAC
GAG	GGA	ACTGCA	CGGCGC	GTGTGT
GCC	TAT	AGAGTC	CGTATA	GTTGTG
GTT	TTA	AGCTGA	CTAGGA	TAATGC
GGA		AGTCAG	CTCTTC	TACGTA
CAC		ATCGAT	CTTCCT	TAGCAT
CCG		ATGCTA	GAAGCT	TCAGAG
CTA		ATTAGC	GACTAG	TCCTCT
CGT		CAACTG	GAGATC	TCTCTC
		CACAGT	GATCGA	TGACCA
		TTTTAA	TGTGGT	TGCAAC

Table 4. Example of k-mer sets, Ω. For $\boldsymbol{\Omega_1}$ and $\boldsymbol{\Omega_2}$ the minimum Hamming distance is 2. *For $\boldsymbol{\Omega_3}$ the minimum hamming distance is 4.*