SUPPLEMENTARY INFORMATION
doi:10.1038/nature11875

# Method for Encoding and Decoding Arbitrary Computer Files in DNA Fragments

## 1 Encoding

**1.1:** An arbitrary computer file is represented as a string $S_\emptyset$ of bytes (often interpreted as a number between $\emptyset$ and $2^8 - 1$, i.e. a value in the set $\{\emptyset \ldots 255\}$)*.

**1.2:** $S_\emptyset$ is encoded using a given Huffman code, converting it to base-3. This generates the string $S_1$ of characters in $\{\emptyset, 1, 2\}$, each such character called a 'trit'.

**1.3:** Write $len()$ for the function that computes the length (in characters) of a string, and define $n = len(S_1)$. Represent $n$ in base-3 and prepend $\emptyset$s to generate a string $S_2$ of trits such that $len(S_2) = 2\emptyset$. Form the string concatenation $S_4 = S_1 \,.\, S_3 \,.\, S_2$, where $S_3$ is a string of at most 24 $\emptyset$s chosen so that $len(S_4)$ is an integer multiple of 25.

**1.4:** $S_4$ is converted to a DNA string $S_5$ of characters in $\{\mathrm{A}, \mathrm{C}, \mathrm{G}, \mathrm{T}\}$ with no repeated nucleotides (nt) using the scheme illustrated in Figure 1. (The first trit of $S_4$ is coded using the 'A' row of the table. For each subsequent trit, characters are taken from the row defined by the previous character conversion.)

| previous nt written | next trit to encode | | |
|---|---|---|---|
| | ∅ | 1 | 2 |
| A | C | G | T |
| C | G | T | A |
| G | T | A | C |
| T | A | C | G |

**Figure 1: Base-3 to DNA encoding ensuring no repeated nucleotides.** For each trit $t$ to be encoded, select the row labelled with the previous nt used and the column labelled $t$ and encode using the nt in the corresponding table cell.

**1.5:** Define $N = len(S_5)$, and let $ID$ be a 2-trit string identifying the original file and unique within a given experiment (permitting mixing of DNA from different

*$\emptyset$ is used throughout to represent the number zero, to avoid confusion with letters o and O.

files $S_{\emptyset}$ in one experiment). Split $S_5$ into overlapping segments of length 1ØØ nt, each offset from the previous by 25 nt. This means there will be $\frac{N}{25} - 3$ segments, conveniently indexed $i = \emptyset \ldots \frac{N}{25} - 4$; segment $i$ is denoted $F_i$ and contains (DNA) characters $25i \ldots 25i + 99$ of $S_5$.

Each segment $F_i$ is further processed as follows:

**1.6:** If $i$ is odd, reverse complement $F_i$.

**1.7:** Let $i3$ be the base-3 representation of $i$, appending enough leading Øs so that $len(i3) = 12$. Compute $P$ as the sum (mod 3) of the odd-positioned trits in $ID$ and $i3$, i.e. $ID_1 + i3_1 + i3_3 + i3_5 + i3_7 + i3_9 + i3_{11}$. ($P$ acts as a 'parity trit'—analogous to a parity bit—to check for errors in the encoded information about $ID$ and $i$.)

**1.8:** Form the indexing information string $IX = ID \,.\, i3 \,.\, P$ (comprising 2+12+1 = 15 trits). Append the DNA-encoded version of $IX$ to $F_i$ using the same strategy as at step (1.4) above, starting with the code table row defined by the last character of $F_i$, to give indexed segment $F_i'$.

**1.9:** Form $F_i''$ by prepending A or T and appending C or G to $F_i'$—choosing between A and T, and between C and G, randomly if possible but always such that there are no repeated nt. (This ensures that we can distinguish a DNA segment that has been reverse complemented during DNA sequencing from one that has not—the former will start with G|C and end with T|A; the latter will start A|T and end C|G.)

See Figure 2 for a schematic representation of the DNA-encoding of computer files.

**1.10:** The segments $F_i''$ are synthesized as actual DNA oligonucleotides and stored, and may be supplied for sequencing.

## 2 Example

**2.1:** As it is difficult to represent all possible bytes in this document, we use a simple example of a file comprising just 18 bytes that happen to be easily

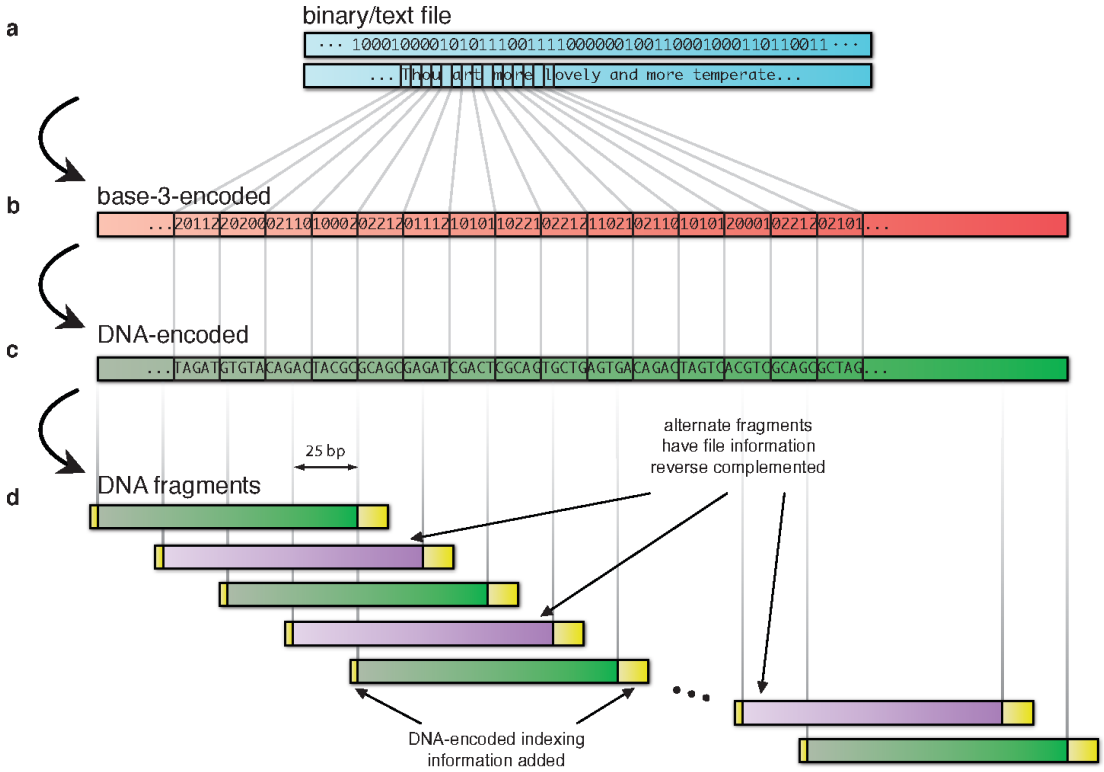


**Figure 2: Schematic of DNA-fragment-encoding of computer files.** The computer file (in any format, e.g. binary or text) shown in blue in (a) (step 1.1 above) is encoded in base-3 (red in b; 1.2–1.3) and then as DNA with no repeated nt (green in c; 1.4). This representation of the complete file is then split into overlapping segments (1.5), alternate segments reverse complemented (mauve; 1.6), and segment indexing and direction-determining information added (yellow; see d; 1.7–1.9).

represented via ASCII codes (for clarity, we show spaces as ␣):

$$S_{\emptyset} = \texttt{Birney␣and␣Goldman}$$

**2.2:** Using the Huffman code defined in the example file `View_huff3.cd`, we convert the bytes of $S_{\emptyset}$ (shown above/below $S_1$ for illustrative purposes) into base-3[†]:

[†]Throughout what follows, spaces are not part of base-3 or DNA strings but are included to assist in 'reading' how strings have been derived in each step.

| | B | i | r | n | e | y | ␣ | a | n |
|---|---|---|---|---|---|---|---|---|---|
| $S_1$ = | 2Ø1ØØ | 2Ø21Ø | 1Ø1Ø1 | ØØØ21 | 2ØØØ1 | 222111 | Ø2212 | Ø1112 | ØØØ21 |
| | 221ØØ | Ø2212 | 222212 | Ø211Ø | Ø21Ø1 | 221ØØ | 11Ø21 | Ø1112 | ØØØ21 |
| | d | ␣ | G | o | l | d | m | a | n |

**2.3:** $n = len(S_1) = 92$, which is 1Ø1Ø2 in base-3. So:

$$S_2 = \text{ØØØØØØØØØØØØØØØ}\ \ \text{1Ø1Ø2 (length 2Ø)}$$

$$S_3 = \text{ØØØØØØØØØØØØØ (length 13)}$$

$$\begin{aligned} S_4 &= S_1\ .\ S_3\ .\ S_2 \\ &= \text{2Ø1ØØ2Ø21Ø1Ø1Ø1ØØØ212ØØØ1222111Ø2212Ø1112ØØØ21} \\ &\quad \text{221ØØØ221222221Ø211ØØ21Ø1221ØØ11Ø21Ø1112ØØØ21} \\ &\quad \text{ØØØØØØØØØØØØØ ØØØØØØØØØØØØØØØ1Ø1Ø2} \\ &\quad \text{(length } 92 + 13 + 2Ø = 125 = 5 \times 25\text{)} \end{aligned}$$

**2.4:** Using the DNA coding strategy and table shown at step (1.4) above, convert $S_4$ to DNA:

$$\begin{aligned} S_5 &= \text{TAGTATATCGACTAGTACAGCGTAGCATCTCGCAGCGAGATACGCTGCTACGCAGCATGCTGT} \\ &\quad \text{GAGTATCGATGACGAGTGACTCTGTACAGTACGTACGTACGTACGTACGTACGACTAT} \end{aligned}$$

**2.5:** $N = len(S_5) = 125$, and we choose (e.g.) $ID = 12$. $S_5$ will be split into overlapping segments $F_i$ of length 1ØØ nt for $i \in \{Ø \ldots \frac{125}{25} - 4\}$, i.e. $i \in \{Ø, 1\}$. With overlapping parts underlined for illustration, $F_Ø$ and $F_1$ are:

$$\begin{aligned} F_Ø &= \text{TAGTATATCGACTAGTACAGCGTAG } \underline{\text{CATCTCGCAGCGAGATACGCTGCTA}} \\ &\quad \underline{\text{CGCAGCATGCTGTGAGTATCGATGA}}\ \underline{\text{CGAGTGACTCTGTACAGTACGTACG}} \end{aligned}$$

$F_1$ = CATCTCGCAGCGAGATACGCTGCTA CGCAGCATGCTGTGAGTATCGATGA

CGAGTGACTCTGTACAGTACGTACG TACGTACGTACGTACGTACGACTAT

**2.6:** Only $i = 1$ is odd, so $F_1$ is reverse complemented:

$F_1$ = ATAGTCGTACGTACGTACGTACGTACGTACGTACTGTACAGAGTCACTCG

TCATCGATACTCACAGCATGCTGCGTAGCAGCGTATCTCGCTGCGAGATG

**2.7:** For $i = \emptyset$, $i3 = \emptyset\emptyset\emptyset\emptyset\emptyset\emptyset\emptyset\emptyset\emptyset\emptyset\emptyset\emptyset$ (length 12) and the sum (mod 3) of the odd-positioned trits of $ID$ and $i3$ is $P = 1 + \emptyset + \emptyset + \emptyset + \emptyset + \emptyset + \emptyset \pmod 3 = 1$. For $i = 1$, $i3 = \emptyset\emptyset\emptyset\emptyset\emptyset\emptyset\emptyset\emptyset\emptyset\emptyset\emptyset 1$ and $P = 1 + \emptyset + \emptyset + \emptyset + \emptyset + \emptyset + \emptyset \pmod 3 = 1$.

**2.8:** For $i = \emptyset$, $IX = ID\ .\ i3\ .\ P = 12\ \emptyset\emptyset\emptyset\emptyset\emptyset\emptyset\emptyset\emptyset\emptyset\emptyset\emptyset\emptyset\ 1$; for $i = 1$, $IX = 12\ \emptyset\emptyset\emptyset\emptyset\emptyset\emptyset\emptyset\emptyset\emptyset\emptyset\emptyset 1\ 1$. So:

$F'_\emptyset$ = TAGTATATCGACTAGTACAGCGTAGCATCTCGCAGCGAGATACGCTGCTA

CGCAGCATGCTGTGAGTATCGATGACGAGTGACTCTGTACAGTACGTACG

AT ACGTACGTACGT C (length $1\emptyset\emptyset + 15 = 115$)

$F'_1$ = ATAGTCGTACGTACGTACGTACGTACGTACGTACTGTACAGAGTCACTCG

TCATCGATACTCACAGCATGCTGCGTAGCAGCGTATCTCGCTGCGAGATG

AT ACGTACGTACGA G (length 115)

**2.9:** Prepend A|T and append C|G (note that in this example we have only one random choice, at the end of $F''_\emptyset$):

$F''_\emptyset$ = A TAGTATATCGACTAGTACAGCGTAGCATCTCGCAGCGAGATACGCTGCTA

CGCAGCATGCTGTGAGTATCGATGACGAGTGACTCTGTACAGTACGTACG

ATACGTACGTACGTC G (length $1 + 115 + 1 = 117$)

$F''_1$ = T ATAGTCGTACGTACGTACGTACGTACGTACGTACTGTACAGAGTCACTCG

TCATCGATACTCACAGCATGCTGCGTAGCAGCGTATCTCGCTGCGAGATG

ATACGTACGTACGAG C (length 117)

# 3 Decoding

Decoding is simply the reverse of encoding, starting with sequenced DNA fragments $F_i''$ of length 117 nt. Reverse complementation during the DNA sequencing procedure (e.g. during PCR reactions) can be identified for subsequent reversal by observing whether fragments start with A|T and end with C|G, or start G|C and end T|A. With these two 'orientation' nt removed, the remaining 115 nt of each segment can be split into the first 1ØØ 'message' nt and the remaining 15 'indexing' nt. The indexing nt can be decoded to determine the file identifier *ID* and the position index *i3* and hence $i$, and errors may be detected by testing the parity trit $P$. Position indexing information permits the reconstruction of the DNA-encoded file, which can then be converted to base-3 using the reverse of the encoding table in step (1.4) above and then to the original bytes using the given Huffman code. Precise details of the decoding procedure are left as an exercise for the reader.

Errors introduced during DNA synthesis, storage or sequencing could lead to various artefacts, particularly nt insertion, deletion or substitution. Recovery of information from fragments with such errors may be possible (details left as exercise)—we have not found this to be necessary due to the large numbers of perfectly-sequenced fragments available via high-throughput sequencing.