Table of Contents

Materials and methods

1. Construction of DNA Template on the Streptavidin Magnetic Beads

The overall scheme for constructing DNA template immobilized on streptavidin magnetic beads is shown above. (i) Resuspend the magnetic beads in the vial (or vortex for 20 seconds), transfer 100 μl of streptavidin magnetic beads into a 1.5 ml tube. Place the tube into a magnetic stand to collect the beads against the side of the tube. Remove and discard the supernatant. (ii) Add 1 mL Buffer I (10 mM Tris-HCl, pH 7.5, 1 mM EDTA, 1 M NaCl, 0.01%-0.1% Tween-20) to the beads, invert the tube several times or vortex gently for 15 seconds to mix. Remove and discard the supernatant from magnetic separation. Repeat this step for 2 times. (iii) Add 500 μl of biotinylated DNA template diluted with Buffer I, makes the beads at a final concentration of 2 mg/ml. Rotate the tube for 30 minutes at room temperature or 2 hours at 4ºC. (iv) Separate the biotinylated DNA template coated beads with a magnetic stand. (v) Add 1 mL Buffer I to the beads, invert the tube several times or vortex gently for 15 seconds to mix. Remove and discard the supernatant from magnetic separation. Repeat this step for 2 times. (vi) Binding is now complete. Resuspend the beads in a buffer at a desired concentration with a low salt concentration, suitable for downstream applications. Use the beads immediately, or store at 4ºC for late use.

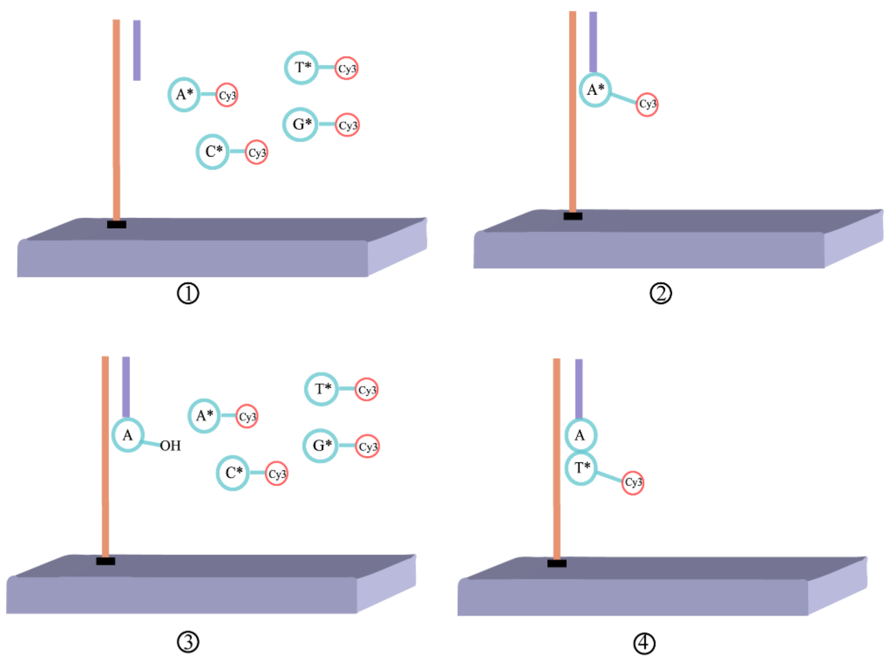
2. Construction of a Chip with Immobilized DNA Template

Pre-cleaned slides were soaked in a 95% acetone/dH2O solution containing 3% 3-aminopropyltrimethoxysilane (Sigma) for 30 min and then washed three times with acetone and three times with dH_2O, and finally dried by nitrogen and baked at 110℃ for 30 min. Then the carboxy-modified slides were obtained by immersing these amino-modified slides in 5mg/ml Polyacrylic acid (aladdin) for 20min and then washed with dH_2O, and finally dried by nitrogen [1].

Amino-modified DNA template solutions containing 50 mM MES (2-(N-morpholino)-ethanesulfonic-acid), pH 5.1 and 20 mg EDC (1-ethyl-3(3-dimethylaminopropyl)-carbodii-mide) were prepared and made the concentration of

DNA template is 2 uM. The solutions were printed onto the carboxy-modified slides by a contact-printing microarrayer. After sprinting, the slides were kept in a chamber with 80% relative humidity at room temperature for 4 h. Then the slides were washed once with 2×SSC,1% SDS for 5 min, then once with 0.2×SSC, 0.1% SDS for 5 min, and finally rinsed with dH_2O and air dried.

3. Continuous polymerase extension by 3'-O-modified fluorescent CRTs on a Chip



Scheme S1. Continuous polymerase extension by 3'-O-modified fluorescent CRTs on a Chip

As shown in the Scheme S1, after the hybridization of primer and DNA templates immobilized with a chip, 8 ul of a solution consisting of dATP-N_3-Cy5, dTTP-N_3-Cy5, dCTP-N_3-Cy5, and dGTP-N_3-Cy5, 2 unit of 9°N DNA polymerase, and 20 mM $MnCl_2$ was spotted on the DNA chip. The extension reaction was conducted at 60℃ for 20min. After washing with 2×SSC,1% SDS for 5 min, and 0.2×SSC, 0.1% SDS for 5 min, the chip was rinsed with dH_2O, and then scanned with a fluorescence scanner. To perform the deprotection, the DNA chip was immersed with 100 mM TCEP (pH 8.0) and incubated at 65°C for 15 min. After washing the surface with dH_2O, the chip was scanned again. After that the next extension–signal detection–deprotection cycle was initiated [2,3].

4. Decoding algorithm

For a single sequencing run with dual-mononucleotide addition, a set of two-digit strings (N^1M^1, N^2M^2, N^3M^3, ..., N^kM^k) is obtained sequentially. Assuming that conjugated mixes XY* and WZ* are alternately introduced to react with template in each sequencing cycle, and a two-digit string N^iM^i is obtained in i cycle. The decoding algorithm, converting the two-digit strings into base-encoding, is as follow:

(1) if $N^i > 0$, $M^i = 0$, $i = 1,2,...,k - 1$, there are $N^i - 1$ base(s) X and one base Y.

(2) if $N^i > 0$, $M^i = 0$, i = k, there are $N^i - 1$ base(s) X and an encoding (XY).

(3) if $N^i \geq 0$, $M^i > 0$, $i = 1,2,...,k$, there are N^i base(s) X, $M^i - 1$ base(s) W and an encoding (WZ).

(4) if $N^i \geq 0$, $M^i > 0$, $N^{i+1} = 0$, $M^{i+1} > 0$, $i = 1,2,...,k - 1$, there are N^i base(s) X, $M^i - 1$ base(s) W and one base Z.

5. Algorithm to correct errors of original four-color codes

we develop a correction algorithm, combining dynamic programming with genetic algorithm, to determine the global optimal decoded sequence of four-color codes.

```
import numpy as np
import random
from tqdm import tqdm
import matplotlib.pyplot as plt
import time
from dynamic import dynamic, read_gene


class GA:
    def __init__(self, epoch=100, N=40):
        self.epoch = epoch
        self.N = N
        self.mu = 0.2
        self.mu_r = 0.2

        # condition for early termination
        self.ep_th = 15
        self.un_mt = 4

        self._init_seed(11)
    def load_data(self, path):
        self.gene_T, self.gene_X, self.mark = read_gene(path)
        self.len_gene = self.gene_T.shape[0]

    def _init_seed(self, seed=0):
        random.seed(seed)
        np.random.seed(seed)

    def pop_init(self):
        population = [np.random.randint(0, 2, self.len_gene) for _ in
range(self.N)]

        return population

    def fit(self, population):
        fitness = []
        for pop in tqdm(population):
            gene = self.gene_X.copy()

            #turn uncertain to determine
            gene[gene == 'X'] =
self.gene_T[np.eye(2)[pop].astype(np.bool)]
            L = dynamic(gene, self.mark)
            fitness.append(L)

        return np.array(fitness)

    def selection(self, pop, fitness):
        # fitness = 1 / (fitness + 1e-9)
        prob = fitness / fitness.sum()
        cumsum = np.cumsum(prob)

        darts = [random.random() for i in range(self.N)]
        darts.sort()

        # roulette wheel selection
        fitin, newin, sele_pop = 0, 0, []
        while newin < self.N:
            if darts[newin] < cumsum[fitin]:
                sele_pop.append(pop[fitin])
                newin = newin + 1
            else:
                fitin = fitin + 1
        random.shuffle(sele_pop)

        # matrix
        father = sele_pop[::2]
        mother = sele_pop[1::2]
        return father, mother

    def crossover(self, father, mother):
        son = []
        for i in range(len(father)):
            ind = np.random.randint(0, 2,
self.len_gene).astype(np.bool)
            son1 = father[i].copy()
            son2 = mother[i].copy()

            son1[ind] = mother[i][ind]
            son2[ind] = father[i][ind]

            son.extend([son1, son2])

        return son

    def mutation(self, _son):
        son = []
        for sn in _son:
            if random.random() < self.mu:
                ind = np.random.choice(a=range(self.len_gene),
size=int(self.mu_r*self.len_gene), replace=False)
                sn[ind] = 1-sn[ind]
            son.append(sn)
        return son

    def update(self, old_pop, son, old_fitness):
        son_fitness = self.fit(son)


        new_pop = np.array(old_pop + son)
        new_fitness = np.concatenate([old_fitness, son_fitness])

        # removal of duplicate individuals
        new_pop, ind = np.unique(new_pop, return_index=True, axis=0)
        new_fitness = new_fitness[ind]
        pop_fitness = list(zip(new_fitness, new_pop))
        pop_fitness = sorted(pop_fitness, key=lambda x: x[0],
reverse=True)
        fitness, pop = zip(*pop_fitness)
        return list(pop[:self.N]), np.array(fitness[:self.N])

    def __call__(self, path):
        self.load_data(path)

        if self.len_gene > 0:
            pop = self.pop_init()
            fitness = self.fit(pop)

            log_fitness_mean, log_mape_max = [fitness.mean()],
[fitness.max()]
            plt.ion()
            plt.figure()
            for i in range(self.epoch):
                father, mother = self.selection(pop, fitness)
                son = self.crossover(father, mother)
                son = self.mutation(son)
                pop, fitness = self.update(pop, son, fitness)
                log_fitness_mean.append(fitness.mean())
                log_mape_max.append(fitness.max())


                print('epoch: {} best_fit: {}'.format(i+1,
log_mape_max[-1]))

                # real-time display of iteration curve
                plt.clf()
                plt.plot(log_fitness_mean, label="fitness_mean",
color='g', linewidth=3)
                plt.plot(log_mape_max, label='fitness_max', color='r',
linewidth=3)
                plt.xlabel('epoch')
                plt.ylabel('fitness')
                plt.legend()
                plt.draw()
                plt.pause(0.01)

                # terminate the iteration early
                if len(log_mape_max) >= self.ep_th and abs(self.mark
- min(log_mape_max[-self.ep_th:])) <= self.un_mt:
```

```
                    if len(np.unique(log_mape_max[-self.ep_th:])) ==
1:
                        break

            best_select = pop[0]
            gene = self.gene_X.copy()
            gene[gene == 'X'] =
self.gene_T[np.eye(2)[best_select].astype(np.bool)]
        else:
            best_select = []
            gene = self.gene_X

        id, L = dynamic(gene, self.mark, is_decode=True)
        np.savez('id', e1=id[2], e2=id[3], select=best_select)
        print('unmatched: {}   match length: {}'.format(id[2:], L))


if __name__ == "__main__":
    ga = GA()
    ga('data/aabb.txt') # randomly selected data irrelevant to the
experiment
```

Results and Discussion

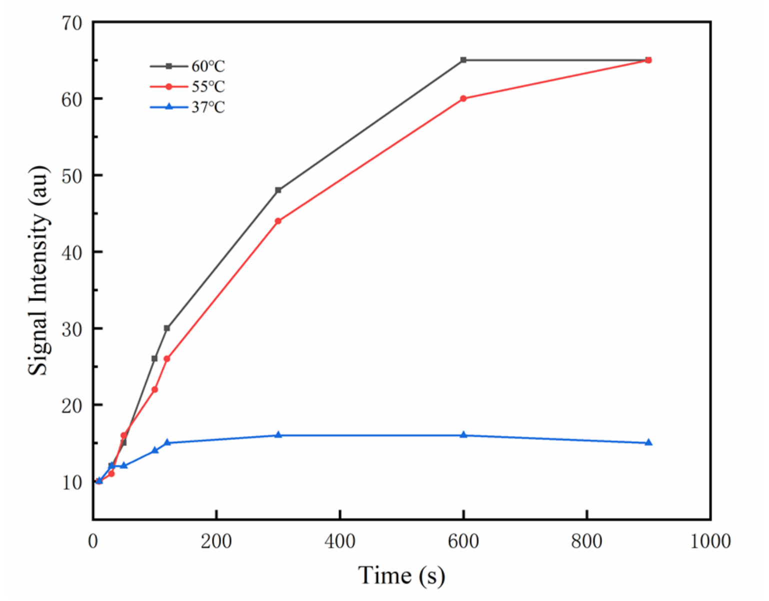


Fig. S1. The enzymatic activity of DNA polymerase at different temperature. The sequencing reaction rate is temperature dependent, with optimal enzyme activity at 60 ℃for 10 min.

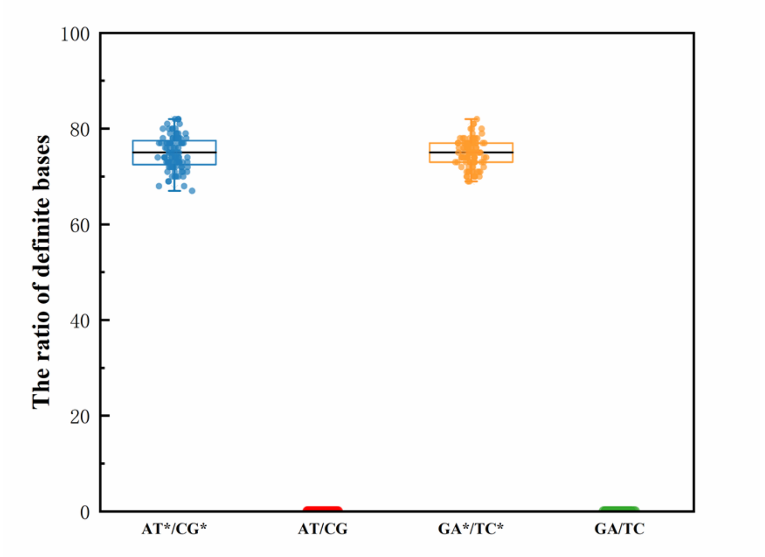
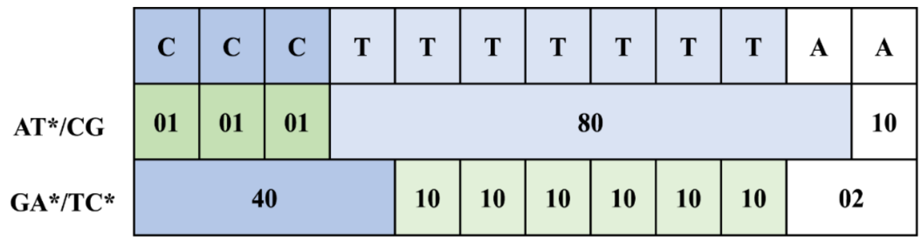


Fig. S2. The ratio of definite bases in a single sequencing cycle obtained by our correctable decoding sequencing and the current dual-mononucleotide sequencing. We randomly generated 100 different DNA template sequences with length 100 bp and calculated the ratio of determined bases in a single sequencing cycle by these two kinds of sequencing approach. Different from the current dual-mononucleotide sequencing that cannot obtain explicit bases, about 75% of the calls of a single sequencing cycle are unambiguous in the correctable decoding sequencing, with decoded substantially less effort.



	C	C	C	T	T	T	T	T	T	T	A	A
AT*/CG	01	01	01	80								10
GA*/TC*	40				10	10	10	10	10	10	02	

Fig. S3. Sequencing of homopolymer with two different sequencing runs. For any homopolymer regions, they can be extended exclusively in at least one of the two cycles. So, there is always a sequencing cycle that the homopolymer is encoded in such a way that only one base information can be measured per reaction cycle, so every homopolymer can be extend exclusively in at least one of the two cycles**.**

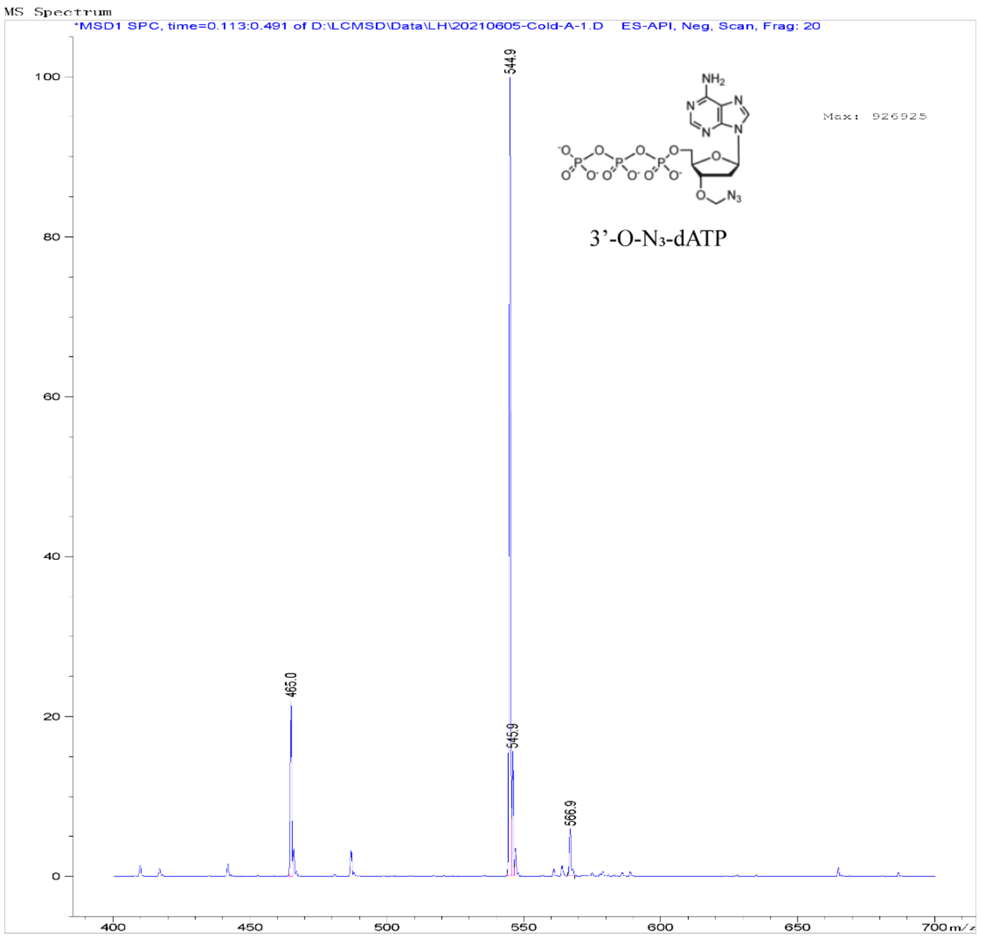


Fig. S4. LC/MSD of 3‘-O-N_3-dATP

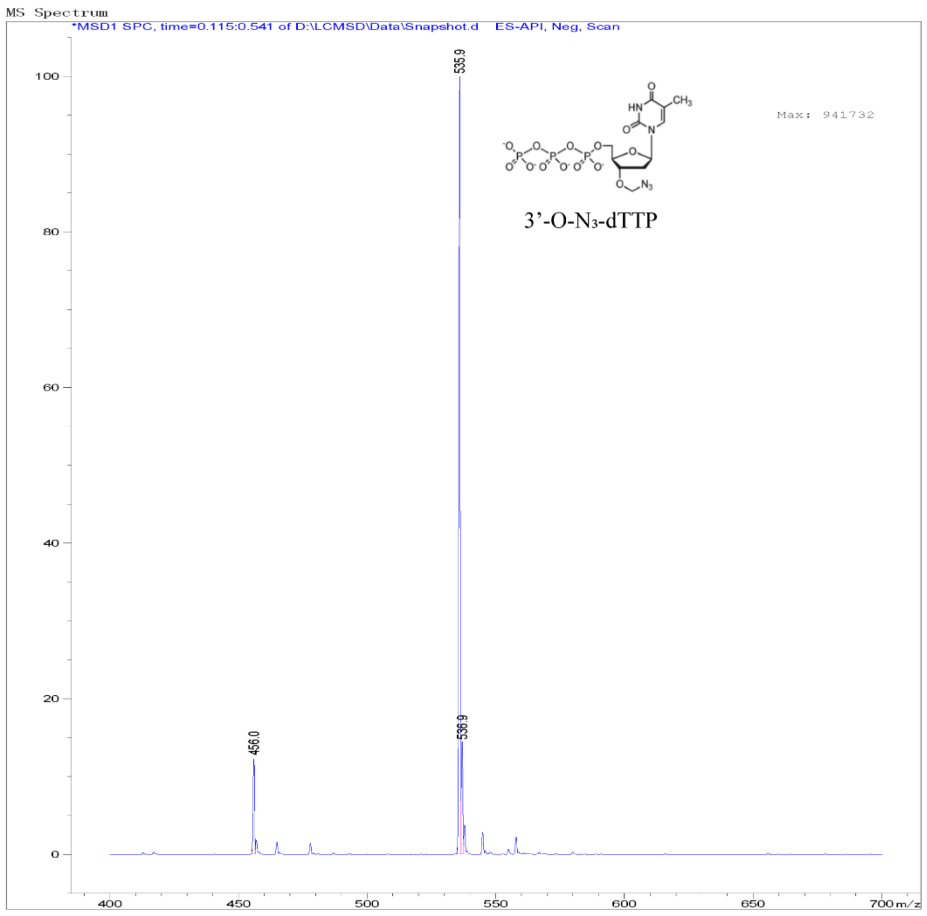


Fig. S5. LC/MSD of 3‘-O-N_3-dTTP

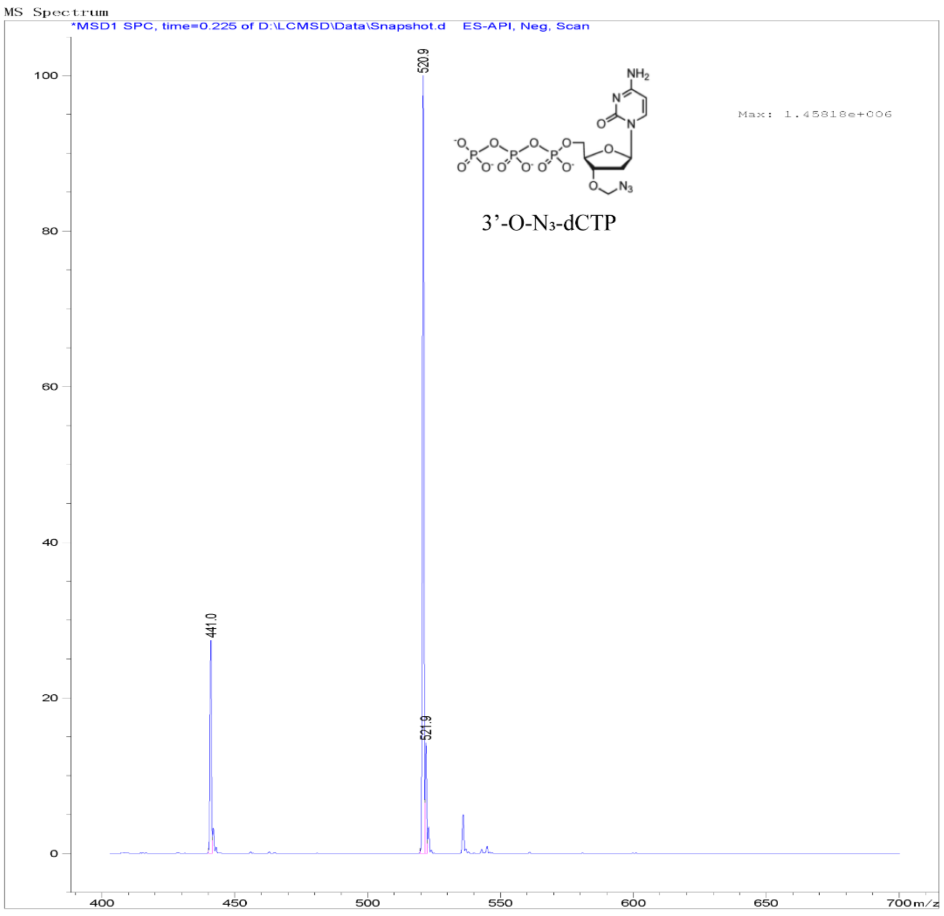


Fig. S6. LC/MSD of 3'-O-N_3-dCTP

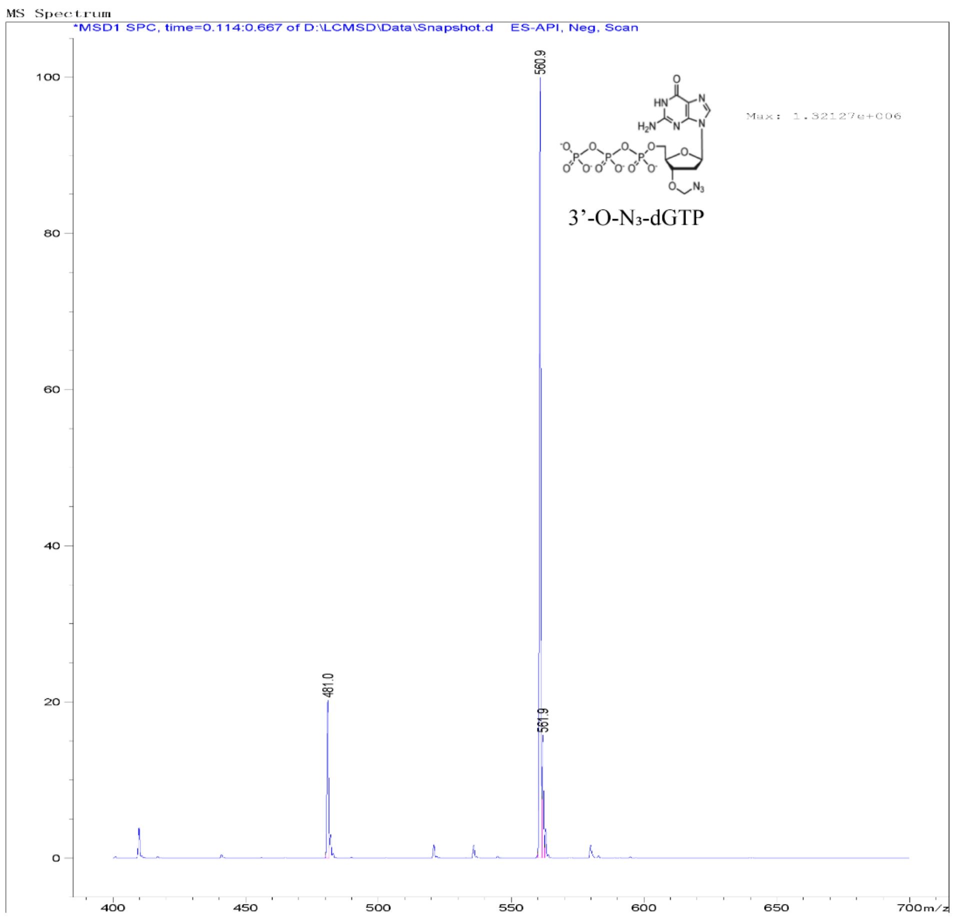


Fig. S7. LC/MSD of 3‘-O-N_3-Dgtp

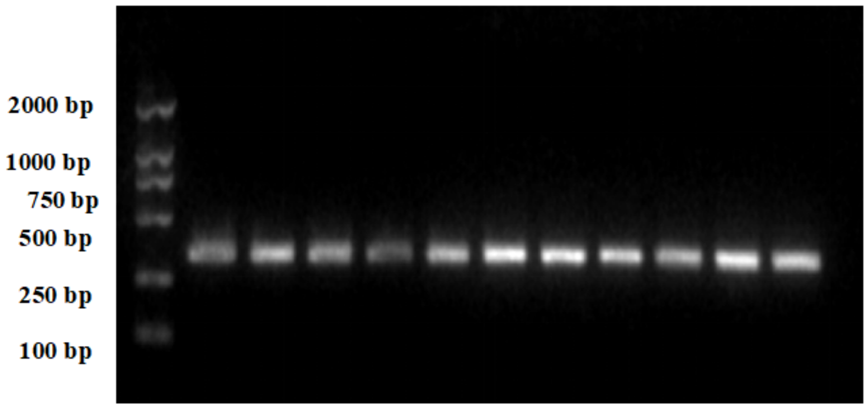


Fig. S8. The gel electrophoresis of PCR amplified product (m.5301A>G [340 bp])

Table S1. Primer and templates used in used in this study

Template	Sequence (5’-3’)
T1	TAAAATGGTCCTCATCGAGACTAGGTGACTAG CCAGTACATCCGATGCCAGTCTG
T2	GATGCGTTAACTCCCCCCCCT CCAGTACATCCGATGCCAGT CTG
T3	GATGCGTTAACTCCCCCCCTA CCAGTACATCCGATGCCAGT CTG
T4	GATGCGTTAACTCCCCCCTAA CCAGTACATCCGATGCCAGT CTG
T5	GATGCGTTAACTCCCCCTAAA CCAGTACATCCGATGCCAGT CTG
T6	GATGCGTTAACTCCCCTAAAA CCAGTACATCCGATGCCAGT CTG
T7	GATGCGTTAACTCCCTAAAAA CCAGTACATCCGATGCCAGT CTG
T8	GATGCGTTAACTCCTAAAAAA CCAGTACATCCGATGCCAGT CTG
T9	GATGCGTTAACTCTAAAAAAA CCAGTACATCCGATGCCAGT CTG
T10	TAAAATGGTCCTCATCGAGACTAGGTGACATTTAG CCAGTACATCCGATGCCAGTCTG
T11	NH_2-TAAAATAAGTTCTTTGGAAATTTCCACAT TTGAAGAAATTTTTCACAT CTCTA
SP1	Biotin-ACTGGCATCGGATGTACTGG
SP2	ATGTGAAAAATTTCTTTCAA
F-P[5301]	CAATTACCCACATAGGATGAA
R-P[5301]	Biotin-AGGCGTAGGTAGAAGTAGAGGTT
S-P	CAAATGGGCCATTATCGAAGAATTCACAAAAAAC

References

[1] X. Shi, C. Tang, D. Zhou, J. Sun, Z. J. E. Lu,PCR-product microarray based on polyacrylic acid-modified surface for SNP genotyping, Electrophoresis 30 (2010) 1286-1296.

[2] J. Ju, D. H. Kim, L. Bi, Q. Meng, X. Bai, Z. Li, X. Li, M. S. Marma, S. Shi, J. Wu, J. R. Edwards, A. Romu, N. J. Turro, Four-color DNA sequencing by synthesis using cleavable fluorescent nucleotide reversible terminators, P. Natl. Acad. Sci. USA. 103 (2006) 19635-19640.

[3] J. Guo, N. Xu, Z. Li, S. Zhang, J. Wu, D. H. Kim, M. Sano Marma, Q. Meng, H. Cao, X. Li, S. Shi, L. Yu, S. Kalachikov, J. J. Russo, N. J. Turro, J. Ju, Four-color DNA sequencing with 3'-O-modified nucleotide reversible terminators and chemically cleavable fluorescent dideoxynucleotides, P. Natl. Acad. Sci. USA. 105 (2008) 9145-9150.