**Supplementary File 1A**

dsDNA-mediated oligomerization rates of FRET-labeled IFI16. Each experiment was performed at least three times and errors were calculated by using the standard deviations.

|  |  |  |
| --- | --- | --- |
| dsDNA size (bps) | 25 nM IFI16 (sec-1) | 50 nM IFI16 (sec-1) |
| 60 | 0.0004 ± 0.0002 | 0.0009 ± 0.0002 |
| 70 | 0.0008 ± 0.0002 | 0.0021 ± 0.0008 |
| 100 | 0.0052 ± 0.0008 | 0.012 ± 0.005 |
| 150 | 0.019 ± 0.004 | 0.037 ± 0.008 |

|  |  |  |
| --- | --- | --- |
| 200 | 0.021 ± 0.007 | 0.049 ± 0.011 |
| 300 | 0.029 ± 0.008 | 0.045 ± 0.014 |
| 600 | 0.021 ± 0.008 | 0.043 ± 0.012 |

**Supplementary File 1B**

Oligonucleotides used in this study are listed below.

|  |  |
| --- | --- |
| oligo 1 | 5’GGGCGGCGACCTGGACAGCAAGTTGGACAATCTCGTTCTATCACTAATTCACTAATGCAGGGAGGATTTCAGATATGGCA-3’ |
| oligo 2 | 5’-biotin-A(16)GAGTACTGTACGATCTAGCATCAATCACAGG  GTCAGGTTCGTTATTGTCCA-3’ |
| oligo 3 | 5’-AGGTCGCCGCCCA(12)-biotin-3’ |
| 601-sequence | 5’-biotin-ATCGAGAATCCCGGTGCCGAGGCCGCTCAATTGGTCGT AGACAGCTCTAGCACCGCTTAAACGCACGTACGCGCTGTCCCCCGCGTTTTAACCGCCAAGGGGATTACTCCCTAGTCTCCAGGCACGTGTCAGATATATACATCCGAT |

**Supplementary File 2**

Code for random walk simulation written in Python

import numpy as np

import random

import math

import matplotlib.pyplot as plt

# Diffusion coefficient d (um2/s)

d=np.double(0.026)

# Number of random walks n

n=np.double(10000)

# Simulation time t (s)

t=np.double(1000)

# Step size dt (s)

dt=t/(n-1)

tv=np.linspace(0,t,n)

# Segment length l (um)

def rnwlk(l):

x = np.zeros((2,n)) + np.random.uniform(0,l,(2,1))

s = math.sqrt(d\*dt)\*(np.random.randn(2,n-1))

# add vdt to steps s for introducing flow bias

# 0.01 µm distance == dimerization

x[:,1:] = x[:,1:] + np.cumsum(s,1)

try:

bmin = np.min(np.nonzero(np.mod(np.absolute(x[0]-x[1]),l) < 0.010))

tsearch = tv[bmin]

except:

tsearch=np.nan

return (x,tsearch)

searchtimes = np.zeros((10000,1))

for i in range(searchtimes.shape[0]):

(x,tsearch) = rnwlk(l)

searchtimes[i] = tsearch

searchtimesperc=np.percentile(searchtimes[np.isnan(searchtimes)==False],(2.5,50,97.5))

plt.hist(searchtimes[np.isnan(searchtimes)==False],bins=math.sqrt(n),normed=True, cumulative=True)

plt.title("2.5%: " + str(searchtimesperc[0])+" 50.0%: "+str(searchtimesperc[1])+" 97.5%: "+str(searchtimesperc[2]) )

plt.ylabel('Cumulative probability density')

plt.xlabel('Search time (s)')

plt.show()