DNA-nanotube-induced alignment of membrane proteins for NMR structure determination

Shawn M. Douglas*†‡, James J. Chou*§, and William M. Shih*‡§

Departments of *Biological Chemistry and Molecular Pharmacology and ‡Genetics, Harvard Medical School, Boston, MA 02115; and §Department of Cancer Biology, Dana–Farber Cancer Institute, Boston, MA 02115

Communicated by Tom A. Rapoport, Harvard Medical School, Boston, MA, January 31, 2007 (received for review December 15, 2006)

Membrane proteins are encoded by 20–35% of genes but represent <1% of known protein structures to date. Thus, improved methods for membrane-protein structure determination are of critical importance. Residual dipolar couplings (RDCs), commonly measured for biological macromolecules weakly aligned by liquid-crystalline media, are important global angular restraints for NMR structure determination. For α-helical membrane proteins >15 kDa in size, Nuclear-Overhauser effect-derived distance restraints are difficult to obtain, and RDCs could serve as the main reliable source of NMR structural information. In many of these cases, RDCs would enable full structure determination that otherwise would be impossible. However, none of the existing liquid-crystalline media used to align water-soluble proteins are compatible with the detergents required to solubilize membrane proteins. We report the design and construction of a detergent-resistant liquid crystal of 0.8-μm-long DNA-nanotubes that can be used to induce weak alignment of membrane proteins. The nanotubes are heterodimers of 0.4-μm-long six-helix bundles each self-assembled from a 7.3-kb scaffold strand and >170 short oligonucleotide staple strands. We show that the DNA-nanotube liquid crystal enables the accurate measurement of backbone NH and Cα dipolar couplings of 15 kDa detergent-reconstituted α-helix transmembrane domains of the T cell receptor. The measured RDCs validate the high-resolution structure of this transmembrane dimer. We anticipate that this medium will extend the advantages of weak alignment to NMR structure determination of a broad range of detergent-solubilized membrane proteins.

The general applicability of solution NMR spectroscopy to structural characterization of intact α-helical membrane proteins has been demonstrated by a number of recent studies. Examples include the structure determination of the 15-kDa Mistic protein (1) and the 30-kDa pentameric phospholamban (2) as well as the complete assignment of backbone resonances and secondary structures of the 44-kDa trimeric diacylglycerol kinase (3) and the 68-kDa tetrameric KcsA potassium channel (4). Despite such progress, full-scale structure determination of α-helical membrane proteins remains challenging and rare. Because of the large fraction of methyl-bearing residues in membrane proteins and of the added molecular weight of detergent micelles, the low chemical-shift dispersion of α-helical proteins is obscured by resonance overlap and line broadening, making assignment of side-chain methyl resonances extremely difficult. Without side-chain chemical shifts, it is impossible to obtain a sufficient number of long-range Nuclear-Overhauser effect-derived distance restraints for folding secondary segments into the correct tertiary structure.

Dipolar couplings between pairs of nuclei measured from weakly aligned proteins in solution provide global orientation restraints important for molecular structure determination and validation by NMR (5–8). Development of alignment media for accurate residual dipolar coupling (RDC) measurements from α-helical membrane proteins would enable significantly the capability of solution NMR in structure determination of these important class of targets. The most effective method for weak alignment involves mixing the protein of interest with large particles that form stable liquid crystals at low concentration (∼1.5–5% wt/vol) (8). Tjandra and Bax (9) first demonstrated this method using 1,2-dimyristoyl-sn-glycero-3-phosphocholine/1,2-distearoyl-sn-glycero-3-phosphocholine (DMPC/DHPC)-bicelle liquid crystals. Subsequently, a number of different liquid crystals, including filamentous phage particles (10, 11), ternary mixtures of cetylpyridinium Cl/Br, hexanol, and sodium Cl/Br (12, 13), binary mixtures of polyethylene glycol and hexanol (14), and cellulose crystallites (15), have been found to be suitable for aligning water-soluble proteins. However, none could be applied to membrane proteins because of incompatibility with the zwitterionic or anionic detergents typically used to solubilize membrane proteins for structural study. Ma and Opella (16) introduced lanthanide ions to an “EF-hand” attached to an HIV-1 membrane protein embedded in lipid micelles that enabled measurement of RDCs, but this method requires modification of proteins that may cause problems for expression and reconstitution of membrane proteins. The only other method currently available for weak alignment of membrane proteins involves the use of strained (radially or axially compressed) polycyclamides gels (17–19). But dissolving protein–micelle complexes to high concentration in gels is notoriously difficult because of the inhomogeneous pore size of randomly cross-linked gel matrices (20), thus the measured RDCs are of limited accuracy.

Liquid crystals of Pf1 phage, which are rod-like particles that are 7 nm in diameter and 2 μm in length (21), are often the first material tested for alignment of soluble proteins. The structural rigidity and negative-charge surface density (22) of Pf1 phage particles allow them to form a stable and useful liquid crystal at low concentrations (10); however, Pf1 is not compatible with detergents. At 20 mg·ml−1, Pf1 is not liquid-crystalline in modest concentrations of detergent, e.g., ~50 mM DPC (J.J.C., unpublished results). Structural DNA nanotechnology can be used for the design and construction of objects of various shapes (23, 24) that, by virtue of being built from DNA, generally are resistant to detergents. In this article, we describe the design and characterization of a DNA structure mimicking the shape and size of
a filamentous phage particle. We used a liquid crystal of these DNA structures to align the H9256
/H9256 transmembrane domain of the T cell receptor, thereby enabling the accurate measurement of RDCs that are in excellent agreement with the previously determined structure. Our results clearly demonstrate the feasibility of a DNA-nanotechnology-based approach for resolving a major bottleneck in membrane-protein NMR spectroscopy.

Results and Discussion
To achieve a Pf1-like DNA structure, a six-helix bundle DNA-nanotube architecture (25) was adopted. This design resembles a parallel array of six double helices for which every set of three adjacent helices frames a dihedral angle of 120° (Fig. 1
/25
/26
/27 a and d). Adjacent double helices are held together by Holliday-junction crossovers that occur every 42 bp (Fig. 1 f). Toward building nanotubes of 0.8-μm uniform length, an assembly strategy was conceived to adapt the scaffolded DNA origami method (26) for the self-assembly of 0.4-μm monomers of two types (“front” and “rear”). For each monomer, a 7,308-base, M13-derived single-stranded circle of DNA is used as a “scaffold,” and 168 single strands of DNA of length 42 bases, programmed with complementarity to three separate 14-base regions of the scaffold, are used as “staples” (Fig. 1 f). The staples self-assemble with the scaffold into the shape of six parallel double helices curled into a tube.

A monomer can be conceptualized as a series of 28 pseudorepeat segments, each consisting of six parallel double helices that are 42 bp long, flanked by jagged overhangs on either end of the object (Fig. 1 b). Each segment can be conceptualized as a series of three subsegments, for which every double helix is 14 bp long (Fig. 1 f). Six of the twelve strands of a subsegment are provided by the scaffold strand, three are provided by one staple strand, and three by another staple strand. Adjacent subsegments are related by 120-degree screw pseudosymmetry. The scaffold generally does not cross over between helices, except for four times in the middle of each monomer to produce a “seam,” and three times on each monomer end (Fig. 1 c). The inclusion of a seam in the design allows for the linkage of monomers in a head-to-tail fashion instead of in a head-to-head fashion, as is evident from consideration of the polarity of the scaffold strand within each double helix [see supporting information (SI) Fig. 4 e]. Three extra staple strands block the head of the front monomer, and four extra staple strands block the tail of the rear monomer (SI Fig. 4 a and d). To facilitate heterodimerization, three extra staple strands with unpaired bases decorate the tail of the front monomer, and three extra staple strands with unpaired bases decorate the head of the rear monomer (SI Fig. 4 b and c).

A computer program (see SI Appendix 1) was written to generate staple strand sequences (see SI Appendix 2) given the sequence of the scaffold. The use of two cyclic permutations of the scaffold sequence as input to the program generates independent sets of staple-strand sequences for folding two different monomer nanotubes. Therefore, copies of the same scaffold molecule can be used to generate two chemically distinct species. Front and rear monomers were folded in separate chambers by heat denaturation, followed by cooling for renaturation (see SI Methods). The majority of DNA objects migrate as a single band in agarose-gel electrophoresis (Fig. 2 a). This population presumably represents well formed nanotube monomers, whereas slower migrating species apparent on the gel presumably represent misfolded or multimerized structures. This mixture was analyzed by

Fig. 1. DNA-nanotube design. (a) 3D cartoon view. The front and rear monomers are folded separately and then combined to form heterodimers. (b) Segment diagram. Each monomer consists of 28 segments of 42-bp length as well as a head and tail segment on each end. (c) Scaffold-only schematic view. Each monomer consists of a modified M13-bacteriophage single-stranded DNA genome of 7,308 bases in length serving as a “scaffold” and 168 DNA strands of 42-bp length acting as “staples” that constrict the scaffold to the target structure. (d) Cross-sectional view. (e) Scaffold-plus-staples schematic view of the heterodimer junction. There are two 42-bp staple strands (orange) per 14-bp subsegment. (f) Scaffold-plus-staples schematic view of a typical 42-bp segment. The crossover pattern of six staple strands repeats for every 42-bp segment along the length of the nanotube.
erodimers at a concentration of 28 mg/ml for 57 segments that are 42 bp long. The measured length agrees well with the predicted length of 814 nm nanotubes (Fig. 2a). In this segment, all six staples migrate as a single band, although some misfolded objects are evident, as are a small population of monomeric objects. Agarose-gel electrophoresis of heterodimers assembled from the two monomers indicates that the majority of DNA nanotubes were tested for weak alignment of the transmembrane (TM) domain (residue 7–39) of the β2 domain of the T cell receptor complex reconstituted in mixed DPC/SDS detergent micelles (SI Fig. 6). The measured 1H–15N and 1H–13C RDCs agree very well with the known NMR structure of the β2–TM domain (27), with a correlation coefficient of the singular value decomposition (SVD) fit, RSVD, of 0.98, or a free quality factor, Qfree, of 16% (Fig. 3a). The magnitude of the alignment tensor, Dzz, is 9.9 Hz (normalized to DNH), which is ideal for RDC measurement and structure calculation. In addition, the axis of C2 rotational symmetry of β2 is parallel to the largest principal axis, Azz, of the alignment tensor (Fig. 3b). This result is expected from the rotational averaging of the dimeric complex around its C2 axis in the aligned medium.

The DNA-nanotube liquid crystal also was used to enable measurement of RDCs for a truncated version of BM2 (residues 26–109), which includes a 20-residue membrane anchor and a soluble coiled-coil tetramerization domain, reconstituted in 1- (1,2-dihexanoyl-sn-glycero-3-phosphoethanolamine)-N-methyl-2-(1-glycerol) (LMPG) detergent micelles (SI Fig. 7). BM2 is a pH-gated, proton-selective channel from influenza B virus. It is better known as the functional homolog of the well characterized M2 proton channel from influenza A virus. Like M2, BM2 plays a central role in equilibrating pH between compartments and is required in the life cycle of the virus (28, 29). The structure of BM2 is not known.

This alignment medium should be broadly useful for providing global structural restraints in solution NMR studies of mem-
brane proteins. Because a large number of helical membrane proteins of great biomedical interest are between 20 and 30 kDa, well below the current size limitation of solution NMR spectroscopy, new experimental systems for obtaining NMR structural information in the presence of detergents are of fundamental importance. DNA nanotechnology, which affords versatile molecular design and subnanometer-scale precision, has been pursued as a route toward building host lattices to position guest macromolecules for crystallographic structural studies (25, 30–32). The present study uses solution NMR instead of crystallographic methods to validate the potential of DNA nanotechnology for imposing order on target macromolecules to acquire atomic-resolution structural information.

Methods

DNA-nanotubes were designed and generated as described in SI Methods and SI Appendices 1 and 2.


Negative Stain Electron Microscopy. After the dimerization step, DNA-nanotube dimers were diluted to 1 nM concentration and prepared for imaging by negative stain with 0.7% uranyl formate (Pfaltz & Bauer, Waterbury, CT) as described (33). Gilder fine bar grids (Ted Pella, Redding, OR), 400 mesh, 3.05 mm o.d., were used. Imaging was performed on a Tecnai G2 Spirit BioTWIN (FEI, Hillsboro, OR).

NMR Sample Preparation. The uniform 15N-, 13C-labeled TM domain (residues 7–39) of the ɛ chain of the T cell receptor complex was expressed and purified as described (27). The TM dimer was reconstituted in mixed detergent micelles consisting of a 5:1 molar ratio of DPC/SDS. The aligned sample was made by manual mixing of 250 μl of a 0.5 mM protein solution with 250 μl of 28 mg/ml DNA-nanotube liquid crystal and subsequently concentrated back to ≈250 μl in a YM-50 Centricon (Millipore, Billerica, MA). The final sample, which consisted of ≈28 mg/ml nanotubes, 0.5 mM protein (monomer concentration), 150 mM DPC, 30 mM SDS, 20 mM sodium phosphate (pH 7.0), and 5% D2O, was loaded to a microcell (Shigemi, Allison Park, PA) for NMR measurement.

NMR Measurements. All NMR experiments were performed on spectrometers (Bruker, Billerica, MA) equipped with cryogenic TCI probes at 300° C. The RDCs were obtained by subtracting J of the unaligned sample from J + D of the aligned sample. The sign of dipolar couplings follows the convention that [1] D NH + [2] D NH < 90 Hz when [2] D NH is positive. The [1] H, 15N RDCs were obtained from [1] J NH/2 and [2] J NH/2, which were measured at 600 MHz (1H frequency) by interleaving a regular gradient-enhanced HSQC and a gradient-selected TROSY, both acquired with 80 ms of 15N evolution. The [1] H–[13] C, [1] C–[13] C RDCs (J C–H) were measured at 500 MHz (1H frequency) by using a 2D CACONH quantitative J C–H experiment with interleaved spectra recorded at J C–H modulation times of 1.83, 3.63, and 7.12 ms. This experiment was modified from the 3D CBCACONH quantitative J C–H experiment (34) used primarily for measuring protein side-chain [1] H–[13] C RDCs. The CACONH was optimized for measuring the backbone [1] Hα–[13] Cα RDCs only. The frequency-labeled dimensions in this experiment are [1] HN (direct) and [2] 15N (indirect). Because of the limited sensitivity of the CACONH experiment, [1] J C–H couplings were extracted from those resonances in the reference spectra ([1] J C–H modulation time of 3.63 ms) with S/N ratio >15. Moreover, overlapped resonances were not used. Because the ɛ–ɛ TM domain is a homodimer obeying 2-fold rotational symmetry, the same RDCs are assigned to both subunits.

We thank Payal Pallavi, Eva Volf, and Xingping Su for assistance with preparation of phage DNA; Matthew Call for preparation of the T cell receptor complex, weakly chain (residues 7–39) of the T cell receptor complex, weakly aligned in 28 mg/ml DNA-nanotube. (a) Correlation between observed backbone RDCs (normalized to D NH) and RDCs predicted for the known NMR structure of ɛ–ɛ TM domain (PDB ID code 2HAC) by using an alignment tensor obtained from the SVD of the ɛ–ɛ TM domain represents the axis of Cα rotational symmetry.

Fig. 3. Analysis of RDCs measured for detergent-reconstituted transmembrane domain of the ɛ chain (residues 7–39) of the T cell receptor complex, weakly aligned in 28 mg/ml DNA-nanotube. (a) Correlation between observed backbone RDCs (normalized to D NH) and RDCs predicted for the known NMR structure of ɛ–ɛ TM domain (PDB ID code 2HAC) by using an alignment tensor obtained from the SVD of the ɛ–ɛ TM domain. The correlation coefficient R2 SVD is 0.98, and Q SVD is 16%. (b) Principal axes of the alignment tensor relative to 2HAC: Azz = 19.8 Hz; Ayy, 15.2 Hz; Axx, 4.6 Hz; D NH = 9.9 Hz; and rhombicity R, 0.357. The solid line in the ribbon structure of the ɛ–ɛ TM domain represents the axis of Cα rotational symmetry.
SI Appendix 1

Python Scripts Used to Generate DNA Staple Strand Sequences. The following Python script was used to generate front monomer core oligos and head caps:

```python
#!/usr/bin/env python
import sys
import string

###
# This program generates a six-helix bundle with bilateral pseudosymmetry
# 1. Create intermediate strands
# 2. Generate tokens
# 3. Generate oligos
###

# This function returns the complementary sequence
complement = string.maketrans('ACGTacgt','TGCAtgca')
def comp(s):
    return s.translate(complement)[::-1]

# This function returns the sequence with all unnecessary characters stripped out
def stripped_seq(seq):
    return ''.join([c for c in seq if c in string.letters])

# read in clonal strand sequence
input_file = file('p7308.txt', 'r')
CSS = stripped_seq(input_file.read())
CSS = CSS[-467:] + CSS[:-467]
input_file.close()

# Generate intermediate strands
num_LH_repeats = 13
num_RH_repeats = 14
start_marker = 0
fragment_ra = []
RH_overhang_length_ra = [26, 26, 40, 40, 2, 2]
LH_overhang_length_ra = [16, 16, 2, 2, 40, 40]
RC_length_ra = [20, -2, -2, 26, 26, 21]
LC_length_ra = [22, 44, 44, 16, 16, 21]
for RH_fragment_num in range(6):
    end_marker = start_marker + RC_length_ra[RH_fragment_num]
    end_marker += num_RH_repeats*42
    end_marker += RH_overhang_length_ra[RH_fragment_num]
    fragment_ra.append(CSS[start_marker:end_marker])
    start_marker = end_marker
for LH_fragment_num in range(6):
    end_marker = start_marker + LC_length_ra[5 - LH_fragment_num]
    end_marker += num_LH_repeats*42
    end_marker += LH_overhang_length_ra[LH_fragment_num]
    fragment_ra.append(CSS[start_marker:end_marker])
    start_marker = end_marker
I_strand_seq_ra = []
for I_strand_num in range(6):
    if I_strand_num%2 == 0:
        upstream_fragment = strand_seq[LH_overhang_length_ra[I_strand_num]:]
        downstream_fragment = strand_seq[:LH_overhang_length_ra[I_strand_num]]
    else:
        upstream_fragment = strand_seq[-LH_overhang_length_ra[I_strand_num]:]
        downstream_fragment = strand_seq[:LH_overhang_length_ra[I_strand_num]]
    I_strand_seq_ra.append(upstream_fragment + downstream_fragment)

# Print intermediate strands
# output_file = file('WS_strands_6hb_v5.txt', 'w')
# for strand in I_strand_seq_ra:
#    output_file.write(strand + '
')
# output_file.close()

# Generate token array
token_ra = []
um_strand_tokens = len(I_strand_seq_ra[0])/7
for I_strand_num in range(6):
```

# Output the token array
```python
```
```
start_marker = 0
sub_token_ra = []
for token_num in range(num_strand_tokens):
    end_marker = start_marker + 7
    sub_token_ra.append(comp(I_strand_seq_ra[I_strand_num][start_marker:end_marker]))
    start_marker = end_marker
token_ra.append(sub_token_ra)

# Generate oligos
ITN_ra = [4, 4, 0, 2, 2, 0]  # initial token number

oligo_ra = []
um_strand_oligos = num_strand_tokens/6
for I_strand_num in range(6):
    SO = int(((I_strand_num%2 - 0.5)*(-2))
    TN = ITN_ra[I_strand_num]

    sub_oligo_ra = []
    for oligo_num in range(num_strand_oligos):
        oligo_seq = token_ra[(I_strand_num + SO*2)%6][TN + 1]
        oligo_seq += token_ra[(I_strand_num + SO*2)%6][TN]
        oligo_seq += token_ra[(I_strand_num + SO)%6][num_strand_tokens - TN - 1]
        oligo_seq += token_ra[(I_strand_num + SO)%6][num_strand_tokens - TN - 2]
        oligo_seq += token_ra[I_strand_num][TN + 1]
        oligo_seq += token_ra[I_strand_num][TN]
    sub_oligo_ra.append(oligo_seq)
    TN += 6
oligo_ra.append(sub_oligo_ra)

# Sort oligos by putting connecting oligos at the end of the list
sorted_oligo_ra = []
connector_oligo_ra = []
for I_strand_num in range(6):
    if I_strand_num%2 == 0:
        sorted_oligo_ra += oligo_ra[I_strand_num][:-1]
        connector_oligo_ra += oligo_ra[I_strand_num][-1:]
    else:
        connector_oligo_ra += oligo_ra[I_strand_num][1:]
        sorted_oligo_ra += connector_oligo_ra

# Generate head-cap oligos
head_cap_oligo_ra = []
head_cap_oligo_ra.append(comp(I_strand_seq_ra[1][:16]) + comp(I_strand_seq_ra[0][-16:]))
head_cap_oligo_ra.append(comp(I_strand_seq_ra[4][-40:]))
head_cap_oligo_ra.append(comp(I_strand_seq_ra[5][0:40]))

# Generate tail-cap oligos
tail_cap_oligo_ra = []
tail_cap_oligo_ra.append(comp(I_strand_seq_ra[0][-42:-16]))
tail_cap_oligo_ra.append(comp(I_strand_seq_ra[1][16:42]))
tail_cap_oligo_ra.append(comp(I_strand_seq_ra[2][-42:-2]))
tail_cap_oligo_ra.append(comp(I_strand_seq_ra[3][2:42]))

#sorted_oligo_ra += head_cap_oligo_ra + tail_cap_oligo_ra
sorted_oligo_ra += head_cap_oligo_ra
output_file = file('front_monomer_oligos.txt', 'w')
num_oligos = 0
for oligo in sorted_oligo_ra:
    output_file.write(oligo + 'n')
    num_oligos += 1
output_file.close()
###

# This function returns the complementary sequence
```python
complement = string.maketrans('ACGTacgt','TGCAtgca')
def comp(s):
    return s.translate(complement)[::-1]
```

# This function returns the sequence with all unnecessary characters stripped out
```python
def stripped_seq(raw_sequence):
    return ''.join([c for c in raw_sequence if c in string.letters])
```

# read in clonal strand sequence
```python
input_file = file('p7308.txt', 'r')
CSS = stripped_seq(input_file.read())
```

CSS0 = CSS[-467:] + CSS[:-467]
CSS1 = CSS[-467 + 100:] + CSS[:-467 + 100]
CSS = ''
```

# Generate intermediate strands
```python
num_LH_repeats = 13
num_RH_repeats = 14
start Marker = 0
fragment ra0 = []
fragment_ra1 = []
RH_overhang_length_ra = [26, 26, 40, 40, 2, 2]
LH_overhang_length_ra = [16, 16, 2, 2, 40, 40]
RC_length_ra = [20, -2, -2, 26, 26, 21]
LC_length_ra = [22, 44, 44, 16, 16, 21]
```

```python
for RH_fragment_num in range(6):
    end_marker = start_marker + RC_length_ra[RH_fragment_num]
    end_marker += num_RH_repeats*42
    end_marker += RH_overhang_length_ra[RH_fragment_num]
    fragment Ra0.append(CSS0[start_marker:end_marker])
    fragment Ra1.append(CSS1[start_marker:end_marker])
    start Marker = end Marker
```

```python
for LH_fragment_num in range(6):
    end_marker = start_marker + LC_length_ra[5 - LH_fragment_num]
    end_marker += num_LH_repeats*42
    end_marker += LH_overhang_length_ra[5 - LH_fragment_num]
    fragment Ra0.append(CSS0[start_marker:end Marker])
    fragment Ra1.append(CSS1[start_marker:end Marker])
    start Marker = end Marker
```

```python
strand Ra = []
```

```python
for strand_num in range(6):
    if strand_num%2 == 0:
        strand0 = fragment Ra0[11 - strand_num] + fragment Ra0[strand_num]
        strand1 = fragment Ra1[11 - strand_num] + fragment Ra1[strand_num]
        strand Ra.append(strand0[-42 - RH_overhang_length_ra[strand_num]:] + strand1[:42 + LH_overhang_length_ra[strand_num]])
    else:
        strand0 = fragment Ra0[strand_num] + fragment Ra0[11 - strand_num]
        strand1 = fragment Ra1[11 - strand_num] + fragment Ra1[strand_num]
        strand Ra.append(strand1[-42 - LH_overhang_length Ra[strand_num]:] + strand0[:42 + RH_overhang_length Ra[strand_num]])
```

```python
for strand in strand Ra:
    sys.stdout.write(strand + 'n')
```

```python
token Ra = []
```

```python
for strand_num in range(6):
    sub_token Ra = []
    num_tokens = 9
    for token_num in range(num_tokens):
        if strand_num%2 == 0:
            sub_token Ra.append(comp(strand Ra[strand_num][token_num*14:token_num*14 + 14]))
        else:
            sub_token Ra.insert(0, comp(strand Ra[strand_num][token_num*14:token_num*14 + 14]))
    token Ra.append(sub_token Ra)
```

```python
oligo Ra = []
```

```python
for zone_num in range(num zones):
    oligo = token Ra[zone_num*2] + token Ra[zone_num*2 + 0]
```
The following Python script was used to generate front monomer tail connector oligos and rear head connector oligos:

```python
#!/usr/bin/env python
import sys
import string

# This function returns the complementary sequence
complement = string.maketrans('ACGTACG', 'TGCATGCA')
def comp(s):
    return s.translate(complement)[::-1]

# This function returns the sequence with all unnecessary characters stripped out
def stripped_seq(raw_sequence):
    return ''.join([c for c in raw_sequence if c in string.letters])

# read in clonal strand sequence
input_file = file('clonal_strand.txt', 'r')
CSS = stripped_seq(input_file.read())
CSS0 = CSS[-467:] + CSS[:-467]
CSS1 = CSS[-467 + 100:] + CSS[:-467 + 100]
CSS = ''
input_file.close()

# Generate intermediate strands
num_LH_repeats = 13
num_RH_repeats = 14
start_marker = 0
fragment_ra0 = []
fragment_ra1 = []
RH_overhang_length_ra = [26, 26, 40, 40, 2, 2]
LH_overhang_length_ra = [16, 16, 2, 2, 40, 40]
RC_length_ra = [20, -2, -2, 26, 26, 21]
LC_length_ra = [22, 44, 44, 16, 16, 21]

for RH_fragment_num in range(6):
    end_marker = start_marker + RC_length_ra[RH_fragment_num]
    end_marker += num_RH_repeats*42
    end_marker += RH_overhang_length_ra[RH_fragment_num]
```

fragment_ra0.append(CSS0[start_marker:end_marker])
fragment_ra1.append(CSS1[start_marker:end_marker])
start_marker = end_marker

for LH_fragment_num in range(6):
    end_marker = start_marker + LC_length_ra[5 - LH_fragment_num]
    end_marker += num_LH_repeats*42
    end_marker += LH_overhang_length_ra[5 - LH_fragment_num]
    fragment_ra0.append(CSS0[start_marker:end_marker])
    fragment_ra1.append(CSS1[start_marker:end_marker])
    start_marker = end_marker

strand_ra = []
for strand_num in range(6):
    if strand_num%2 == 0:
        strand0 = fragment_ra0[11 - strand_num] + fragment_ra0[ strand_num]
        strand_ra.append(strand0[-42 - LH_overhang_length_ra[ strand_num]:] +
                         strand1[:42 + LH_overhang_length_ra[ strand_num]])
    else:
        strand0 = fragment_ra0[ strand_num] + fragment_ra0[11 - strand_num]
        strand_ra.append(strand1[-42 - LH_overhang_length_ra[ strand_num]:] +
                         strand0[:42 + LH_overhang_length_ra[ strand_num]])

for strand in strand_ra:
    sys.stdout.write(strand + '\n')

token_ra = []
for strand_num in range(6):
    sub_token_ra = []
    num_tokens = 9
    for token_num in range(num_tokens):
        if strand_num%2 == 0:
            sub_token_ra.append(comp(strand_ra[strand_num][token_num*14:token_num*14 + 14]))
        else:
            sub_token_ra.insert(0, comp(strand_ra[strand_num][token_num*14:token_num*14 + 14]))
    token_ra.append(sub_token_ra)

oligo_ra = []
for zone_num in range(num_zones):
    oligo = token_ra[(zone_num*2 - 1)%6][zone_num]
    oligo += token_ra[(zone_num*2 + 0)%6][zone_num]
    oligo += token_ra[(zone_num*2 + 1)%6][zone_num]
    oligo_ra.append(oligo)
    oligo = token_ra[(zone_num*2 + 4)%6][zone_num]
    oligo += token_ra[(zone_num*2 + 3)%6][zone_num]
    oligo += token_ra[(zone_num*2 + 2)%6][zone_num]
    oligo_ra.append(oligo)

output_file = file('front_rear_connector_oligos.txt', 'w')
for oligo in oligo_ra[6:-6]:
    output_file.write(oligo + '\n')
output_file.close()

input_file = file('front_monomer_oligos.txt', 'r')
lines = input_file.readlines()
input_file.close()
v5_oligo_ra = [stripped_seq(line) for line in lines]

input_file = file('rear_monomer_oligos.txt', 'r')
lines = input_file.readlines()
input_file.close()
v6_oligo_ra = [stripped_seq(line) for line in lines]

for oligo_num in range(18):
    if v5_oligo_ra.count(oligo_ra[oligo_num]) == 1:
        sys.stdout.write('Oligo ' + str(oligo_num) + ' present in v5 oligo array.' + '\n')
    elif v6_oligo_ra.count(oligo_ra[oligo_num]) == 1:
        sys.stdout.write('Oligo ' + str(oligo_num) + ' present in v6 oligo array.' + '\n')
    else:
        sys.stdout.write(oligo_ra[oligo_num] + '\n')
SI Appendix 2

DNA Sequences

1) Computer generated random 59 base sequence, inserted into M13mp18 at insert position 6258

```
TATACGGGTACTAGCCATGCGTATACGGTCGCTAGCGGACTTGCCTCGCTATCAAAGGT
```

2) Deoxyribonucleotide sequences that were used to construct M13mp18 insert fragment

```
TCGAGCTCGGTACCCGGGGATCCTTATACG
CGCATGGGCTAGTACCCGTATAAGGATCCCC
GGTACTAGCCATGCGTATACGGTCGCTAGC
TAGCGAGGCAAGTCCGCTAGCGACCGTATA
GGACTTGCCTCGCTATCAAAGGTCTAGAGT
CATGCTGCAAGGTCGACTCTAGACCTTTGA
```

3) Recombinant M13 filamentous bacteriophage genome sequence. This sequence serves as the input to all Python scripts in the form of a text file named "p7308.txt".

```
AATGCTACTACTATTAGTAGAATTGATGCCACCTTTTCAGCTCGCGCCCCAAATGAAAAT
ATAGCTAAACAGGGTTATTGAGCCATTTTCCGAAAATGTAATCTAATTGGTCAGGTAACTCT
CTGTCGACAGAATTTGATCGCTACAGCATTATATTCAAGCAATTAAAATCTCTAAGCCA
TTTCGCGAGAAATCACTCTTTATCAAAAGGAAAGCAATTTAAAGGTACTCTTTTAATCT
TTCAGGCTGTTTTGCTTCCGCTTGGTGTTGGTTGAAGGCCTGGATTAAAAAGCGATATTT
ACTCTTTTCTCTCTCTATACTTTGGTGATGCAATCGCTTCTCTCTCTCTATAATAGT
CAGGCTGAAAGCATCTTTGTGATTGAACTCTTCCTCGTTTGCTGACTATAATAGT
CATGCTGCAAGGTCGACTCTTTTAATCTCTTAATTTGATCGCTACAGCATTATATTCAAG
```

4) Deoxyribonucleotide sequences for six-helix bundle DNA nanotube, front monomer head cap staples

AGGATCCCCCGGTACC GGCTAGTACCCGTATA
ATATTTTAGTTAATTTTCATCTTCTGACCTAAATTTAAATGG
TTTGAATACCGACCGTG TGATAAAATAAGCGTTAAATA

5) Deoxyribonucleotide sequences for six-helix bundle DNA nanotube, front monomer core staples

AGTAATAAAAGGGACTGTTTTCTGTGCTTTTGATAGCGAG
AAATGGATTAATTTAACATACGAGCGGCAGGCAGTGCCAA
AACGCTCATGGAATAATGAGTGAAGCTATGGGGAACCAGG
AATATCCAGAACAACCCGCTTTCCAGTCCCGCCAGCTGGGA
ACTTGCCCTGAGTATGAAATCGGCAACAGATTTGGGAAGG
ATTACCGTTGTAGCGCAGGGTGGTTGCCGGAAACCAGG
ATCAGTGAGGCCACCTGATTGCCCTTCAAGATCGCACTC
AGACAGGAACGGACGGTAACGCTGGTGACCTGAGG
ATCAGAGCGGGACGGTGGTTCCGAATGGGATAGGTCAC
GGTTGCTTTGACTGAAAATGCGTCCGGCGATTCTC
ACACCCGCGCGCTAAAGAGCTCCACTATTTGTAGCCAGCTTT
AGGGCGCTGGCAAGCGAAAAACCGTCTACCAATAGGAACGCC
GTGGCGAGAAAGGATCACCCCAATCAAGAAAAATTCGCAAT
GGGAGCCCCCGATTCTAATCGGAACCCCCCTTTGATAAATGCCGAAG
AAGAARCTGCTCTAGCGAGAAAACATTATTACAGGGGCTATCAGGTCAT
CGTAACAAAGCTGCCTCGTTACCACAGACATTAAGCAGGAG
GAGTAATCTTGAACATTTCGAAAAAGGAAGCAAATCACCATCAA
CGGTGTACAGACCATTATGACTGGATAGTGAGTTAAGAGATT
TAAGGGAAACGACATTTCAATTTGAAATTCAACCCTATTCCGCAA
CTCCATGTTACTTACGAGAATGACCATATTTCGTCGGAAG
TTGTATCTGCCTATATTATAGTCAGAAGAGCTAATCAGGGTG
CCCAGGATTATACAGGAAGGGCGAAAGCAAGAAATGAAAGCAA
CGAAAGGGCAAAAATTCAAAGCGGAACAAAAATAGTAGATGATGCAT
GGGTAAATACGTAATTAGAGATGACCTTTCATTTGGGGCGC
TTGAGGACTAAGATTGTCGATGGCAGTACATATTCCGCGA
AAAGACAGCATCGGTAGCTCAACATGTTTGATTCCCAATTCT
TTACCAGCGCCAAATTAGTTTGACCATTAGAGCTTAAATTGC
AATAAAGTTATTTTTGTTATAGCTATATTATATTGCAGTCTCTTT
ATAAAGGGGTGAAGCAGCAATTCCTACTGACAGGGAAGCAAC
TCCTTATTACGAGCTACAGGGAAGCTTCAATATCGC
CAATAATAACGGAGC CCTAAGGCATAAACAGCGGGATTGCA
CAGATAGCGCGACATGACCCCTTGTAATCAATCATCAAAATCAGG
AGCAATAGCTATCTCAAGGATAAAAATTTCAATGCAGTTTA
AATTGAGTTAAGCCATGCCCAGTAATGCGTCAATACCTGCG
AGAGGTTAATTGGAGAGGCGGAGACAGTTTTTGGCCAGAGGG
CGCATTAGACGGGAGTTCTAGCTGATAAGCAGTAAAAACCA
GTAATAGTAAAATGGGCAGATTAGGCTGGTAGGAACCCATGTA
AAATAGCGAGAGGCAGAACCGGATATTCCCTCAGACGTACACCAGGC
ATAAAAAGAACATATATATCCAGTGAGCTGAACTCCTCCTCA
GGTGCCGTAAAGCATAGAGTGTGACGCTGGGCTTTGCCAATCTA
CACTACGTGAAACAGGGAAGGAAAGCGATGATGGCAATTCAT
CCAAAGCTAAGGGGTGATACCGTGACCGCAAAGGAAGGC
TTCCAGTTTGGAGTTACTATGCGCCTACAAAGTTTGAAGTAA
CTTATAATCAAAAAGCAAGCTATAACGTGGAACACTGATTA
CGAAAATCCTGTGTTCAAACAGGAGCCGATACATTGGATT
GAGAGTTCAGCAACCGCCAGAATCTGGAAGGACTAAACA
GAGACGGGCAACAGCGAGTAAAAGAGTCAACAGTTGGAAGGA
GTGTTCGTATTGGCAATACTTCTTTGAAAAATATCAAACCT
GCCAGCTGCATAAAAGAACTCAAATGCAAGCAATGAAA
CGTTGCAGCCACTGTATTACGCCAGGCCATGATTAACAC
AAGCCTGGGTGCTACACTACATTGTGAAACCGAAGACACG
TCACAAATTCCACACCATTGGAATACGCGCGATG
AATCATGGTCAACAGGCTTTGCGCAATAACATGAGAGC
GCAAAGTCGCTAGCATCATATAATTACAGCAAAAAGGAACCGG
GCTTGCATGCTGCTACATCTTTCTTGTACATATATATGT
GTTTTCGTACCGCTTTGGAATGTCGAGGACTAC
AGGGGGATGTGCTGTTTTAGGCAGAGCAAGACGCTGAGAAA
GCGATCGTGCGGGAGTACCGACAAAAGTTTTAGATTGCAATCTC
AAAGCGCCATTCCGAATAACACATGGTTGATGAAATAACCTTG
CAGCCAGCTTTTCCGAACAATAGAATGTTACCTTTTTAAT
6) Deoxyribonucleotide sequences for six-helix bundle DNA nanotube, front monomer
tail connector staples
7) Deoxyribonucleotide sequences for six-helix bundle DNA nanotube, rear monomer head connector staples

CGCTGGAAGTTTCAATGCAAATCCAATCCGGCTTAGGTTGGG
GGCTTGCAGGGACGACCTTTTTAACCTCGCAAGACAAAGAAC
GAGTTAAAGGCCGCGGCCAGTGCCAAGACGCACGCTTTGTAAAA

8) Deoxyribonucleotide sequences for six-helix bundle DNA nanotube, rear monomer core staples

CCATTGCAACAGGATTGGTATAGCGAGGCTGCAAGGCGATTAA
TATCGGCCCTTGCTGCTAGTACCCGTATAGGCCTTCCGCTAT
GATTAGTAATAACAGTAATCATGGTCATGCCATTCAGGCTGC
TCTGTCCATCACGCGCTCACAATTCCACCGGCACCGCTTCTG
GAGAAGTGTTTTTATAAAGCCTGGGCTGAGCATTACGTCTGAGCTG
CGATTTAAGGGATTTGCGTTGCGCTACGGCGCATCGTAAC
TGCTTTCTCGCTAGTCAGGTCCAGCTGATTCGCGGATGGACCG
ACAGGGCGCGGTACTCGGTTTGCAGATTGTGAGCGAGTAAC
GCTGCGCGTACCCAGCTGAGACGGCAACTCGCGTCTGGCCTT
GAAAGGAGCGGGCGGAGAGAGTTGCAGCTCAGCTCATTTTTT
GGGAAAGCCGGCGAGGCAAAATCCCTGCTACGTAAAAATTTTGT
AAATCGGAACCCCTACCTTTTAAATCAACCAAAAAACAGGAAG
CCCAGCTACAATTTAATATGATATTTCAATCAGGACGTTGGGA
GAGGTTTGAAGCCGAGGGGTAGCTATTTGAATTACCTTATGC
TCTAAGAAGCGGAGATTTGCTAGAGCTGGATGTTGCTTGAG
CCCAATAGCAAGCATAATCGTAAAAACTAACAGAGTCCACTA
AAGCAAGCGGTTTTTAAATCAGAAAAAGCCAAGATAGCCCGAG
CCAAGAACGGGTATTATTTAAATTGTAATTGATGTTGTTCCC
ATGTAAGAAACAAATAAATTTTTGGTTAAAAAGCGGTCCACGCT
GTCCTGAACAGGACATCAAATAATTAGCTGATTTGCCCTTT
TTTAGCTTAATGCAGCTCAATCACTAAATAGCTGATTGCCCTT
TTCAGCTAATGCAGCTCATCAACATTTAAAGCCAGGGCGGTTGG
AGGTAAAGTAAAATTCTCCGTGGGAACAAAAATGAATCGCCCAA
CATTTCGAGCCGAGCTTTGGTGTAATGGGCTGCCCCCTTTCCAG
ATGCACCATATTTAATTGGAGGGACGACCCCTAAATGAGTGGAC
ACCAGTATAAAGCGCTCCAGCCAGCTTTCAACACATACGAGCC
AGAAAAAGCCTGTTGGCAAAGCCATTCAGCCTTTTCTGTG
ATAGGGCTTAATGGGACATCGGGTACAGAGATCCCCTCGGTA
CTGACCTAATTTAAAAAGGGGATGTCAGAATCCGCTAGCGA
GCGAGAAAACCTTTGCCCCCCGTGCTGATCCGCTGCAG
AGGTCTGAGAGACTCTCTACATCGTCTGAAAATACCTACATTT
TAAGACGCTGAGAAAATGCATCCAGCACGAGATAATTACCGCCAG
ATTTTCCCTTGAAGATAGAACCCTTCTAGAAAGACTCAAAC
TGATGCAATAACCTTAGACAATATTTTGGAGCAATACTTTCTTT
AATTACCTTTTTATGATAGCCCTAAAAACCGAGTAAAGAG
CATCAAGAAAAACCAAACCAGCAGAGATATACGCCAGAATCCT
ATTCTTCTTCAATACCGGCTGACACCGCTAAAACAGGAGGC
TTGCTTTGAAATACCCAAATCTAAAGCATGACACGTATAACG
AGTACCTTTTCATCTCAATCAAATCTCTTAAATGCGGCGCCT
GCGTACATTCTCAGGAATTGGAGGTAGGTAGTGAGGTCGAC
ACATTCAACTAATGCATAAGGGAACCGATCGAGAGGGTTGAT
ATTACAGGTGAAAAAAACGGTGTACAGACGGATTAGCGGGGTT
AGAAAAATCTACGTAAGAGTAATCTTGAAATGAAAGTATTAG
GATTTTAAGAACTGAACGTAACAGCAGCCATGGCTTTTGA
ATAGGGTTGAGTGTGTGTGTTTGGGTCGAGGAAACGTATATAATT
GAAATCGGCAAATATAGGGAGCCTCCGAAACTTTTACAACAAAT
GGTTTGCCCACGCAACGTGCGAGAAAGAGCCGCATCTGCTGAATAGAT
CACCCTGGCCCTCTAGGGCGCTGCAATCTCAAAATATCT
TTTCTTTTCCCACCTAGACACCGGGGCGTACGTGGAAATAC
CGCGCGGGGAGAGGATGGTTGCTTTGACCACCTTGCTGAAC
TCGGGAAACCTGTCGAATCAGAGCGGGATGCACGCTGAGAG
TAACTCATATATTGACAGGAACCGGAAACAGAGGTGAGG
GGAAGCATAAAATGTGATCAGTGAGCCGCATGCACTAAAA
TGAAATTGTTATCCAAATTTAACCCTGGTATAATGGCTATTAGTC
CCGAGCTCGAATTCTCACTTGCTGAGTGACCTGAAAGCGTA
CCGTATACGCTGGTAATATCAGAACTAATAAAAGGGACA
GTCGACTCTAGACCAAAACGCTCATGGAATGGATTATTTACA
GTTGGTAAAGCCATCAAATATATTTTTATTTTATCAAATCAT
TACGCGCTTGCGGAGTGGTGTGAAATACCGCATGCTTAGAT
GCAAACGTGTTGGAAAAAGAAACACCGGTGCCTTAATAATT
GTGCCGGAAACCCAGTAGTATCATATGCTGAAATCATATATG
CAGGAAGATCGCACAACCTGACAGAAACATGTTAATATCAGG
CGTGCATCTGCGACAAGCAAGCCTTCAGTAACATATTCAGTTTG
TAATGGGATAGGGTGCTCATAAAAGAATATCGCGAGGGCAATT
AACCAGTCGATTTCTGTCAGACGACGAACGATTCGCCCTGA
9) Deoxyribonucleotide sequences for six-helix bundle DNA nanotube, rear monomer tail cap staples
CTTTTGATAAGAGGTCATTTTTTCG
GGATTAGAGGTACCTTTAATTGCTC
TGAATTTCTTAAACAGCTTGATACCGATAGTTGCGCCGAC
GAGCCTTTAATTGTATCGGTTTATCAGCTTGCTTTGAGG