# Supplementary Information

## Competition of individual domain folding with inter-domain interaction in WW domain engineered repeat proteins

Kapil Dave,† Andrei G. Gasic,† Margaret S. Cheung* and M. Gruebele*

† These authors contributed equally to the work

*Corresponding author emails: mgruebel@illinois.edu; mscheung@uh.edu

**SI Contents:**



### 1. Sequences used in the experiments and computational model

**Table S1.** Sequences of FiP35 WW domain oligomer constructs used in this work. Linkers in **bold**. The FiP35 sequence is derived from the NCBI reference sequence NP_006212.1.

| **Fip35 (Monomer)** |
|---|
| HMKLPPGWEKRMSRDGRVYYFNHITNASQFERPSG |
| **DFiP35 (Dimer)** |
| HMKLPPGWEKRMSRDGRVYYFNHITNASQFERPSG**GGSGGSGGSG**KLPPGWEKRM SRDGRVYYFNHITNASQFERPSG |
| **TFiP35 (Trimer)** |
| HMKLPPGWEKRMSRDGRVYYFNHITNASQFERPSG**GGSGGSGGSG**KLPPGWEKRM SRDGRVYYFNHITNASQFERPSG**GGSGGSGGSG**KLPPGWEKRMSRDGRVYYFNHIT NASQFERPSG |
| **QFiP35 (tetramer)** |
| HMKLPPGWEKRMSRDGRVYYFNHITNASQFERPSG**GGSGGSGGSG**KLPPGWEKRM SRDGRVYYFNHITNASQFERPSG**GGSGGSGGSG**KLPPGWEKRMSRDGRVYYFNHIT NASQFERPSG**GGSGGSGGSG**KLPPGWEKRMSRDGRVYYFNHITNASQFERPSG |

### 2. Additional information on the global fitting model

**Explanation of parameters in the model**

The model parameters mentioned in Methods and in Table 1 in the main text are described below. The model free energy is given by

$$\Delta G = \#N\{g_{UN}(T - T_m) + m_{UN}[GuHCl]\} + \quad \text{(S1)}$$

$$\#M\left\{g_{UM}(T - T_m) + m_{UM}[GuHCl] + g_{UM}^{(0)}\right\} + \#U \cdot 0 \quad \text{(S2)}$$

- Subscripts are: N (folded form); M (misfolded form); U (unfolded form)
- #N, #M and #U equals the number of N, M, and U domains present
- $T_m$ is the melting temperature of native domains in absence of denaturant or intermediates
- $g_{UN}$ is the thermal free energy derivative (folded domain relative to unfolded domain)
- $m_{UN}$ is the corresponding denaturant free energy derivative
- $g_{UM}$ is the thermal free energy derivative (misfolded domain relative to unfolded domain)
- $m_{UM}$ is the corresponding denaturant free energy derivative
- $g^{(0)}_{\mathrm{UM}}$ is the free energy of M relative to U at $T$=$T_m$ and no denaturant

Units are provided in Table 1 of the main text.
The observed fluorescence signals were fitted by assuming a linear baseline for each domain, and summing over the number of each type (N, M or U) of domain. For example, for the unfolded domain baseline $S_U$ is given by

$$S_U = b_U + a_U(T - T_m) \qquad \text{(S3)}$$

$b_U$ = unfolded intercept (average contribution per monomer to the overall intercept)
$a_U$ = unfolded slope (average contribution per monomer to the overall slope)

Similarly the native intercept, baseline and slope are represented with a subscript "*N*." We assumed that the parameters for misfolded domains were the average of unfolded and native domains as this was sufficient to globally fit all the data essentially within measurement uncertainty and greatly reduced the model's flexibility. Finally, the model assumed constant activation barriers and a constant prefactor (ignoring temperature-dependence of the solvent viscosity):
$G^{\dagger}_{NU}$ = kinetic barrier going from folded to unfolded, in kJ/mole
$G^{\dagger}_{MU}$= kinetic barrier going from intermediate to unfolded, in kJ/mole
In the kinetic model the starting point at $t$=0 for the experimental data and simulated data was matched, i.e. we assumed that there was no unresolved an ultra-fast kinetic phase or 'dead time.'

**Matlab code for the free energy of tethered oligomers**
Note: The variables g31= $g_{UN}$, g32= $g_{UM}$, go32= $g^{(0)}_{UM}$, gg31= $m_{UN,}$ gg3= $m_{UM}$, gnn=gmm=0, mu= $a_U$, mm= $a_M$, mf= $a_N$ in the notation at the top of this page.

```
Function[S,dG,Keq,Si]=ThermoFit(Mer,TotalStates,GHCL,TRange,Tm,g31,g32,go32,gg31,gg3
2,gnn,gmm,bu,mu,bf,mf,bm,mm)
%
NumStates = size(TotalStates,1);      % number of all possible species
%%
for p = 1:numel(TRange)35
   T= TRange(p);
   % Signals for individual N, M and U
   Su = bu+mu*(T-Tm)+ 5*(Mer==1)- 4*(Mer==4);     %unfolded baseline%%%%%%%
additional 5nm for monomer
   Sf = bf+mf*(T-Tm)+0.5*(Mer==1)- 4*(Mer==4);    %folded baseline %%%%%%%%%%
additional 1nm for monomer
   Sm = bm+mm*(T-Tm);  %misfolded baseline
   % Thermodynamic delta G for transitions
   % Here each of species is a separate state with associated G
   % U/UU/UU/UUUU is the ground/ref state with G ==0
   deltaG  = zeros(1,NumStates); % Initialize
```

```
   for i =1:NumStates % This loop will calculate signal & k_eq for each species coming from ground species
     howmanyN = numel(find(TotalStates(i,:)==1)); % how many N are there in order to make signal
     howmanyM = numel(find(TotalStates(i,:)==2)); % how many M are there in order to make signal
     howmanyU = numel(find(TotalStates(i,:)==3)); % how many U are there in order to make signal
     Si(p,i) = (howmanyN*Sf+howmanyM*Sm+howmanyU*Su)/Mer; % generate signal for all the species
     %
     speciesstr = sprintf('%u',TotalStates(i,:)); % change species into a string
     xn = numel(findstr(speciesstr, '11')); % find the pair MM in the species
     xm = numel(findstr(speciesstr, '22')); % find the pair MM in the species
     xu = numel(findstr(speciesstr, '33')); % find the pair MM in the species
     dG(p,i) = howmanyN*(g31*(T-Tm)+ gg31*GHCL + xn*gnn)+ howmanyM*(g32*(T-Tm)+ go32+ gg32*GHCL +xm*gmm) + howmanyU*0;
     Keq(p,i) =exp(-dG(p,i)/8.31/(T+273.15)); % equilibrium rate for all the species i
   end
   S(p,1) = Si(p,:)*Keq(p,:)'/sum(Keq(p,:),2); % generating signal for thermodynamics
end
end
```

**Matlab code for the kinetic representation**

```
function[Chi,Time,Conc,ConcEq,TransMatrix]=KinFit(Mer,TotalStates,GHCL,T,T_fin,...
Tm,g31,g32,go32,gg31,gg32,gnn,gmm,Gk13k,Gk23k,tspan,ExpData)
NumStates = size(TotalStates,1);      % number of all possible species
% Calculate the barriers and kinetic parameters
W = 20; % prefactor [1/us]
%% Solve ODE at T = temp
TransMatrix = zeros([NumStates, NumStates]);
% TransMatrix(i,j) is rate of reaction of species i going to species j
for i = 1:NumStates
  howmany(i,1) = numel(find(TotalStates(i,:)==1)); % how many N are there in order to make signal
  howmany(i,2) = numel(find(TotalStates(i,:)==2)); % how many M are there in order to make signal
  howmany(i,3) = numel(find(TotalStates(i,:)==3)); % how many U are there in order to make signal
  for j = 1:NumStates
  speciesstr1 = sprintf('%u', TotalStates(i,:)); % change reactant species into a string
  speciesstr2 = sprintf('%u',TotalStates(j,:)); % change product species into a string
  xn = numel(strfind(speciesstr2, '11'))- numel(strfind(speciesstr1, '11')); % find effective change in pairs
  xm = numel(strfind(speciesstr2, '22'))- numel(strfind(speciesstr1, '22'));
  xu = numel(strfind(speciesstr2, '33'))- numel(strfind(speciesstr1, '33'));
  %% Thermodynamic delta G for transitions
  % G31 is defined outside the for loop as it is NOT dependent on x
  G31     = g31*(T-Tm)+ gg31*GHCL + xn*gnn;
```

```
  G32     = g32*(T-Tm)+ go32+ gg32*GHCL + xm*gmm; % uses x
   %%% Remaining kinetics from here
   %Important parameter to play with
   Gk13=(Gk13k-0.5*G31);
   Gk31=(Gk13k+0.5*G31);
   Gk23=(Gk23k-0.5*G32);
   Gk32=(Gk23k+0.5*G32);
   %
   kmatrix=zeros([3,3]); % kmatrix initiation
   kmatrix(1,3)=W*exp(-Gk13/(8.31*(T+273.15))); % units would be microsec inverse
   kmatrix(3,1)=W*exp(-Gk31/(8.31*(T+273.15)));
   kmatrix(2,3)=W*exp(-Gk23/(8.31*(T+273.15)));
   kmatrix(3,2)=W*exp(-Gk32/(8.31*(T+273.15)));
   % when monomer
   if(Mer==1)
     if(i==j)
      TransMatrix(i, j) = 0;
     else
       TransMatrix(i, j) = kmatrix(TotalStates(i), TotalStates(j));% filling up transmatrix from the
kmatrix which is created in kinetic_nMer script
     end
   end
   % when polyMer more than monoMer system
   if(Mer>1)
     transformInd=[];flipMer=[];beforeSwitch=[];afterSwitch=[];
     subtract1 = TotalStates(i,:)- TotalStates(j,:);        % substraction of rows in order to
determine if only one of the N,M,U is switching
     subtract2 = fliplr(TotalStates(i,:))- TotalStates(j,:); % flipping the sequence 123-322 makes it
seems like 2 places are changed but if we flip 123 to 321-322 only one place is changed and
hence it should be allowed
     if(nnz(subtract1)==1)                          % if only subtraction lead to one non-zero entry
then do the below loop
         transformInd = find(subtract1~=0);              % what is the position/index where the
switch is happening
         beforeSwitch = TotalStates(i,transformInd);       % what was it (N=1,M=2,U=3) that
switched
         afterSwitch = TotalStates(j,transformInd);        % what was it (N=1,M=2,U=3) that it
switched to
         TransMatrix(i, j) =  kmatrix(beforeSwitch, afterSwitch);   % picking the rates from kinetic
Mer kmatrix and filling in trans matrix
     elseif(nnz(subtract2)==1)                          % for the flipping case doing the same
thing
         transformInd = find(subtract2~=0);
         flipMer = fliplr(TotalStates(i,:));
         beforeSwitch = flipMer(1,transformInd);
         afterSwitch = TotalStates(j,transformInd);
         TransMatrix(i, j) =  kmatrix(beforeSwitch, afterSwitch);
     else
       TransMatrix(i, j) = 0;
```

```
      end
    end
   end
  end
  ratematrix = TransMatrix'; % Transpose of the transmatrix should give us ratematrix for make
differential equation
    for i = 1:NumStates
      for j = 1:NumStates
        if (i==j)
          ratematrix(i,j) = - sum(TransMatrix(i, :)); % making the ratematrix from Transition matrix
        end
      end
    end
    conc0   = zeros([NumStates, 1]); % initial conc initialization for all the states to be zero
    conc0(1)= 40e-6;  % initial concentration of nn nnn nnnn
    options = odeset('RelTol',1e-8,'AbsTol',1e-14,'Stats','off',...
   'NormControl','on','NonNegative',numel(conc0),'Refine',1,...
   'MStateDependence','weak','MassSingular','maybe','BDF','off');
    [TEq,ConcEq] =
ode15s(@(t,conc)myODE(t,conc,ratematrix),linspace(0,1e4,1e2),conc0,options);
    %% Calulation of kinetic rates
    TransMatrix = zeros([NumStates, NumStates]); ratematrix = [];
    % TransMatrix(i,j) is rate of reaction of species i going to species j
    for i = 1:NumStates
      for j = 1:NumStates
        speciesstr1 = sprintf('%u', TotalStates(i,:)); % change reactant species into a string
        speciesstr2 = sprintf('%u',TotalStates(j,:)); % change product species into a string
        xn = numel(strfind(speciesstr2, '11'))- numel(strfind(speciesstr1, '11')); % find effective
change in pairs
        xm = numel(strfind(speciesstr2, '22'))- numel(strfind(speciesstr1, '22'));
        xu = numel(strfind(speciesstr2, '33'))- numel(strfind(speciesstr1, '33'));
        %% Thermodynamic delta G for transitions
        % G31 is defined outside the for loop as it is NOT dependent on x
        G31     = g31*(T-Tm)+ gg31*GHCL + xn*gnn;
        G32     = g32*(T-Tm)+ go32+ gg32*GHCL + xm*gmm; % uses x
        %% Remaining kinetics from here
        %Important parameter to play with
        Gk13=(Gk13k-0.5*G31);
        Gk31=(Gk13k+0.5*G31);
        Gk23=(Gk23k-0.5*G32);
        Gk32=(Gk23k+0.5*G32);
        %
        kmatrix=zeros([3,3]); % kmatrix initiation
        kmatrix(1,3)=W*exp(-Gk13/(8.31*(T_fin+273.15))); % units would be microsec inverse
        kmatrix(3,1)=W*exp(-Gk31/(8.31*(T_fin+273.15)));
        kmatrix(2,3)=W*exp(-Gk23/(8.31*(T_fin+273.15)));
        kmatrix(3,2)=W*exp(-Gk32/(8.31*(T_fin+273.15)));
        % when monoMer
        if(Mer==1)
```

```
        if(i==j)
         TransMatrix(i, j) = 0;
        else
          TransMatrix(i, j) = kmatrix(TotalStates(i), TotalStates(j));% filling up transmatrix from
the kmatrix which is created in kinetic_nMer script
        end
       end
       % when polyMer more than monoMer system
       if(Mer>1)
         transformInd=[];flipMer=[];beforeSwitch=[];afterSwitch=[];
         subtract1 = TotalStates(i,:)- TotalStates(j,:);        % substraction of rows in order to
determine if only one of the N,M,U is switching
         subtract2 = fliplr(TotalStates(i,:))- TotalStates(j,:); % flipping the sequence 123-322 makes
it seems like 2 places are changed but if we flip 123 to 321-322 only one place is changed and
hence it should be allowed
         if(nnz(subtract1)==1)                          % if only subtraction lead to one non-zero
entry then do the below loop
             transformInd = find(subtract1~=0);              % what is the position/index where the
switch is happening
             beforeSwitch = TotalStates(i,transformInd);       % what was it (N=1,M=2,U=3) that
switched
             afterSwitch = TotalStates(j,transformInd);        % what was it (N=1,M=2,U=3) that it
switched to
             TransMatrix(i, j) =  kmatrix(beforeSwitch, afterSwitch);   % picking the rates from
kinetic Mer kmatrix and filling in trans matrix
         elseif(nnz(subtract2)==1)                          % for the flipping case doing the same
thing
             transformInd = find(subtract2~=0);
             flipMer = fliplr(TotalStates(i,:));
             beforeSwitch = flipMer(1,transformInd);
             afterSwitch = TotalStates(j,transformInd);
             TransMatrix(i, j) =  kmatrix(beforeSwitch, afterSwitch);
         else
           TransMatrix(i, j) = 0;
         end
       end

     end
   end
   ratematrix = TransMatrix'; % Transpose of the transmatrix should give us ratematrix for make
differential equation
   for i = 1:NumStates
     for j = 1:NumStates
       if (i==j)
         ratematrix(i,j) = - sum(TransMatrix(i, :)); % making the ratematrix from Transition matrix
       end
     end
   end
   options = odeset('RelTol',1e-8,'AbsTol',1e-14,'Stats','off',...
```

```
'NormControl','on','NonNegative',numel(conc0),'Refine',1,...
'MStateDependence','weak','MassSingular','maybe','BDF','off');
[Time, Conc] = ode15s(@(t,conc)myODE(t,conc, ratematrix), tspan, ConcEq(end,:)', options);
%% Formulating the X(but not sure at this point)
concN=zeros(size(Conc(:,1)));
concM=zeros(size(Conc(:,1)));
concU=zeros(size(Conc(:,1)));
for i = 1:NumStates
concN = concN + Conc(:,i)*howmany(i,1)/Mer;
concM = concM + Conc(:,i)*howmany(i,2)/Mer;
concU = concU + Conc(:,i)*howmany(i,3)/Mer;
end
sumconc = concN+concM+concU;
concN=concN./sumconc;
concM=concM./sumconc;
concU=concU./sumconc;
end
```

**3. AWSEM additional information and parameter input file**

The total Hamiltonian consists of a backbone term $\mathcal{H}_{BB}$, a potential of mean force $\mathcal{H}_{PMF}$, and a fragment memory term $\mathcal{H}_{FM}$:

$$\mathcal{H}_{AWSEM} = \mathcal{H}_{BB} + \mathcal{H}_{PMF} + \mathcal{H}_{FM}. \quad \text{(S4)}$$

$\mathcal{H}_{BB}$ constrains the backbone chain to physically realistic heteropolymer conformations. using connectivity of beads, chain bonding, chiral, Ramachandran, and excluded volume terms:

$$\mathcal{H}_{BB} = \mathcal{H}_{con} + \mathcal{H}_{chain} + \mathcal{H}_{\chi} + \mathcal{H}_{rama} + \mathcal{H}_{excl} \quad \text{(S5)}$$

The potential of mean force $\mathcal{H}_{PMF}$ depends on the identities of the interacting residues and contains direct contacts, water-mediated contacts, burial, β-strand hydrogen bonding, parallel-antiparallel cooperative hydrogen bonding, and helix hydrogen bonding terms:

$$\mathcal{H}_{PMF} = \mathcal{H}_{direct} + \mathcal{H}_{water} + \mathcal{H}_{burial} + \mathcal{H}_{\beta} + \mathcal{H}_{P-AP} + \mathcal{H}_{helix} \quad \text{(S6)}$$

The form of the P-AP potential is

$$\mathcal{H}_{P-AP} = -\lambda_{P-AP}\Big[\gamma_{APH}\sum_{I=1}^{N-13}\sum_{J=I+13}^{\min(I+16,N)} \nu_{I,J}\nu_{I+4,J-4} + \gamma_{AP}\sum_{I=1}^{N-17}\sum_{J=I+17}^{N} \nu_{I,J}\nu_{I+4,J-4} + \gamma_{P}\sum_{I=1}^{N-13}\sum_{J=I+9}^{N-4} \nu_{J,I}\nu_{I+4,J+4}\Big] \quad \text{(S7a)}$$

where

$$\nu_{I,J} = \frac{1}{2}\Big[1 + \tanh\Big(\eta\Big(r_0 - r_{C\alpha_I,C\alpha_J}\Big)\Big)\Big]. \quad \text{(S7b)}$$

In the smooth switching term $\nu_{I,J}$, the screening length $\eta$ = 7.0 Å, and the cutoff $r_0$ = 8.0 Å. Here, $I$ and $J$ are the residue indices (or $C_\alpha$ atom indices) The first summation term in $\mathcal{H}_{P-AP}$ aligns continuously connected chain segments into β-hairpins with weight $\gamma_{APH}$ = 1.0, and second and

third terms induce antiparallel and parallel alignments of two chain segments, respectively, with weights $\gamma_{AP} = \gamma_P = 0.4$. The scaling factor $\lambda_{P-AP}$ was set to 0.5 kcal/mole in all simulations.

To further correct the secondary structure bias, we used the Protein Secondary Structure Prediction server JPRED, which provides information to adjust AWSEM's Ramachandran term $\mathcal{H}_{Rama}$ and β-strand hydrogen bonding term $\mathcal{H}_{\beta}$. The default Ramachandran potential is not sequence-dependent (with the exception of proline), but by predicting the secondary structure using JPRED, we bias the potential towards Ramachandran angles found in β-sheets or α-helices. As for the hydrogen bonding term $\mathcal{H}_{\beta}$, the secondary structure information from JPRED is used to derive the propensity for residue pairs to hydrogen bond. See SI section 4 for secondary structure prediction.

**Parameter input file**

```
[Chain]
10.0 10.0 30.0
2.45798 2.50665 2.44973

[Chi]
20.0 -0.83

[Epsilon]
1.0

[Rama]
2.0
5
 1.3149  15.398 0.15   1.74 0.65 -2.138
1.32016 49.0521 0.25  1.265 0.45  0.318
 1.0264 49.0954 0.65 -1.041 0.25  -0.78
    2.0   419.0  1.0  0.995  1.0  0.820
    2.0  15.398  1.0   2.25  1.0  -2.16

[Rama_P]
3
 0.0    0.0 1.0   0.0  1.0   0.0
2.17 105.52 1.0 1.153 0.15  -2.4
2.15 109.09 1.0  0.95 0.15 0.218
 0.0    0.0 1.0   0.0  2.0   0.0
 0.0    0.0 1.0   0.0  2.0   0.0

[SSWeight]
0 0 0 1 1 0
0 0 0 0 0 0

[ABC]
0.483 0.703 -0.186
0.444 0.235 0.321
0.841 0.893 -0.734
```

```
#Chain, Chi, Rama, Rama_P, ABC, are parameters for the backbone potential.
#SSWeight turns on the JPRED Secondary structure bias for Rama, and
#for Dssp_Hdrgn.

#Dssp_Hdrgn is H_beta
#first line for all the terms below are scaling factors

[Dssp_Hdrgn]
1.0
0.0  0.0
1.37  0.0  3.49 1.30 1.32 1.22   0.0
1.36  0.0  3.50 1.30 1.32 1.22   3.47  0.33 1.01
1.17  0.0  3.52 1.30 1.32 1.22   3.62  0.33 1.01
0.76   0.68
2.06   2.98
7.0
1.0    0.5
12.0

#P_AP is H_P_AP and scaling factor is changed from 1.0 to 0.5

[P_AP]
0.5
1.5
1.0 0.4 0.4
8.0
7.0
5 8
4

#Water is H_water

[Water]
1.0
5.0 7.0
2.6
13
2
4.5 6.5 1
6.5 9.5 1

#Burial is H_burial

[Burial]
1.0
4.0
0.0 3.0
3.0 6.0
6.0 9.0

#Helix is H_Helix and is turned off indicated by "-"

[Helix]-
1.5
2.0 -1.0
7.0 7.0
3.0
4
15.0
```

```
4.5 6.5
0.77 0.68 0.07 0.15 0.23 0.33 0.27 0.0 0.06 0.23 0.62 0.65 0.50 0.41 -3.0 0.35
0.11 0.45 0.17 0.14
0 -3.0
0.76   0.68
2.06   2.98

#[Fragment_Memory_Table]
#scaling_factor (varied from 0.1 to 0.3)
#mem_file
#gamma_file
#rmin rmax dr
#frag_table_well_width (changed from 1.0 to 0.2)
#fm_energy_debug_flag
#fm_sigma_exp

[Fragment_Memory_Table]
0.2
fragsLAMW.mem.single
seq.gama
0 50 0.01
0.2
0
0.15
```

**4. Secondary structure prediction from JPRED for monomer, dimer, trimer, and tetramer. "E" stands for β-strand.**

**Monomer:**

```
HMKLPPGWEKRMSRDGRVYYFNHITNASQFERPSG
--------EEEE-----EEEEE----EEEE-----
```

**Dimer:**

```
HMKLPPGWEKRMSRDGRVYYFNHITNASQFERPSGGGSGGSGGSGGSKLPPGWEKRMSRDGRVYYFNHITNASQFERPS
--------EEEE-----EEEEE----EEEE--------------------EEEE-----EEEEE----EEEE----
G
-
```

**Trimer:**

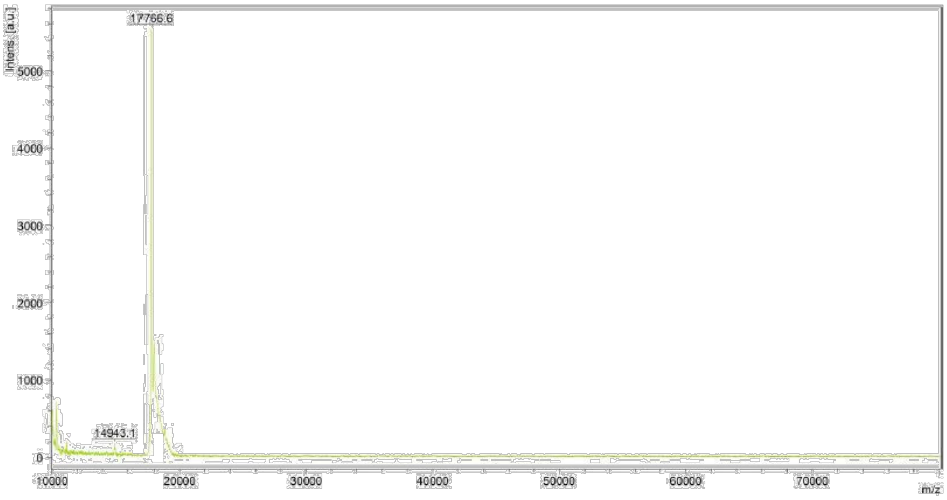
```
HMKLPPGWEKRMSRDGRVYYFNHITNASQFERPSGGGSGGSGGSGGSKLPPGWEKRMSRDGRVYYFNHITNASQFERPS
--------EEEE-----EEEEE----EEEE--------------------EEEE-----EEEEE----EEEE----
GGGSGGSGGSGGSKLPPGWEKRMSRDGRVYYFNHITNASQFERPSG
-----------------EEEE-----EEEEE----EEEE-----
```

**Tetramer:**

```
HMKLPPGWEKRMSRDGRVYYFNHITNASQFERPSGGGSGGSGGSGGSKLPPGWEKRMSRDGRVYYFNHITNASQFERPS
--------EEEE-----EEEEE----EEEE--------------------EEEE-----EEEEE----EEEE----
GGGSGGSGGSGGSKLPPGWEKRMSRDGRVYYFNHITNASQFERPSGGGSGGSGGSGGSKLPPGWEKRMSRDGRVYYFNHIT
-----------------EEEE-----EEEEE----EEEE--------------------EEEE-----EEEEE---
NASQFERPSG
-EEEE-----
```

## 5. Mass spectrometry of QFiP variants

**A**




**B**

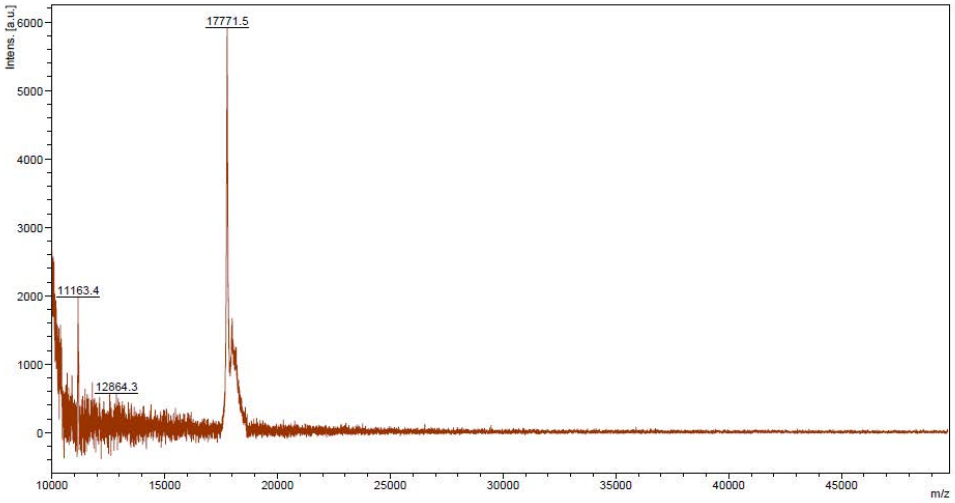



**Fig. S1:** Mass spectrometry results of Qfip35 protein purified using **A)** GST and **B)** His tag both showing a peak at m/z = 17.766 K Da and 17.771 K Da respectively.

## 6. Additional kinetics and thermodynamic melt traces and global fits

### a. Kinetics traces for MFiP35, TFiP35 and QFiP35 (model with 3 states per domain)

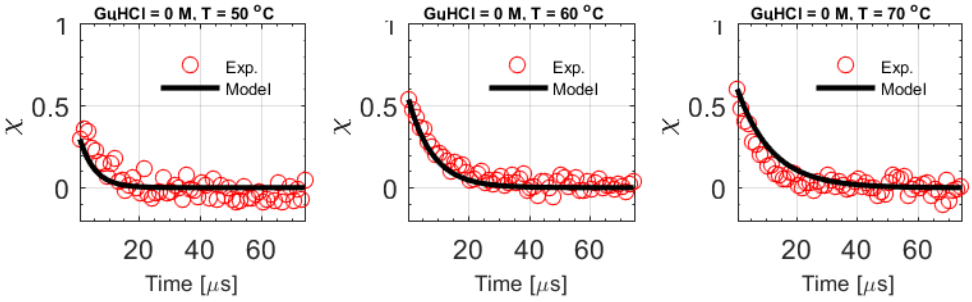



**Fig. S2.** Kinetic traces for the monomer, with global three-states-per-domain fit.

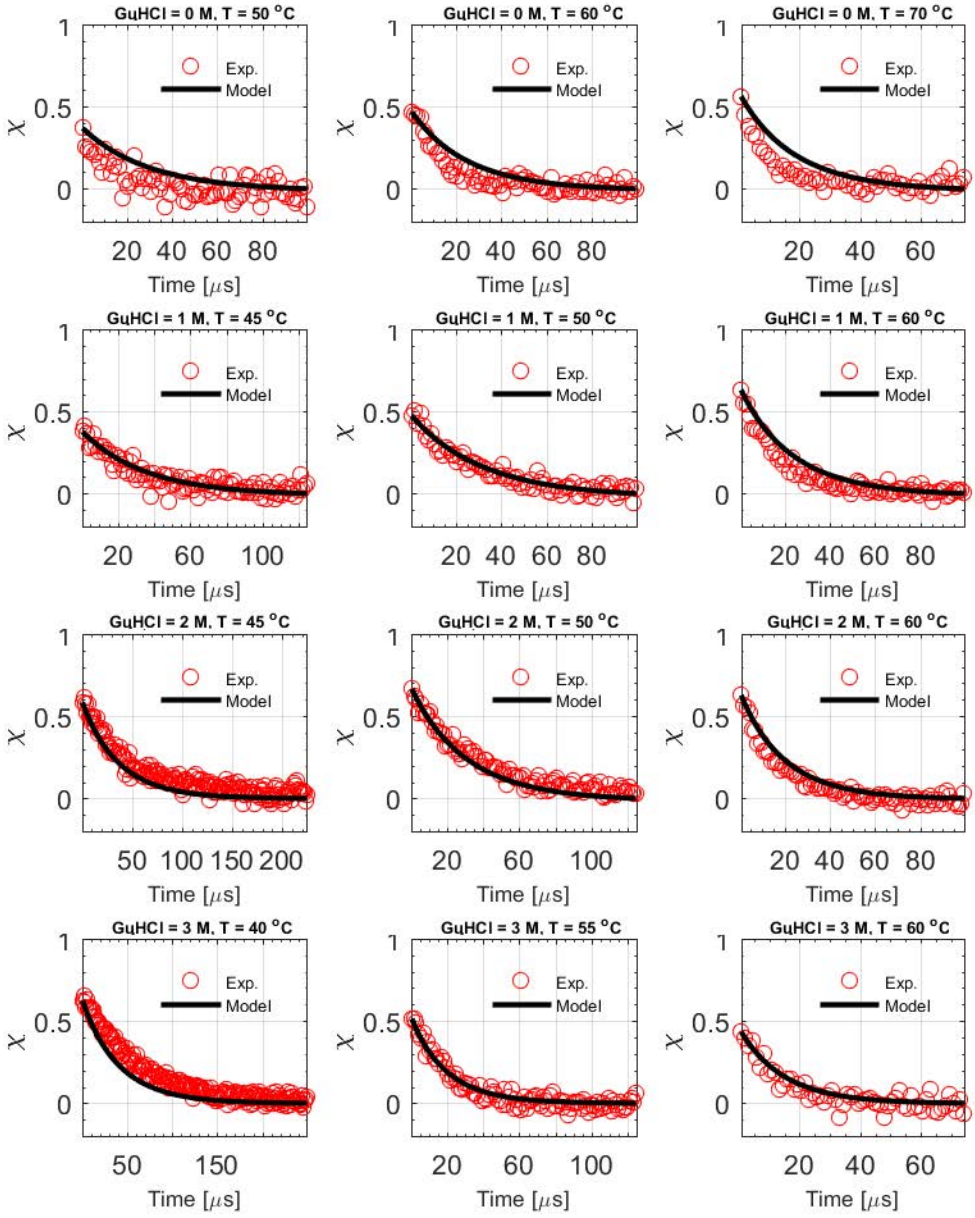



**Fig. S3.** Kinetic traces for the trimer, with global three-states-per-domain fit.

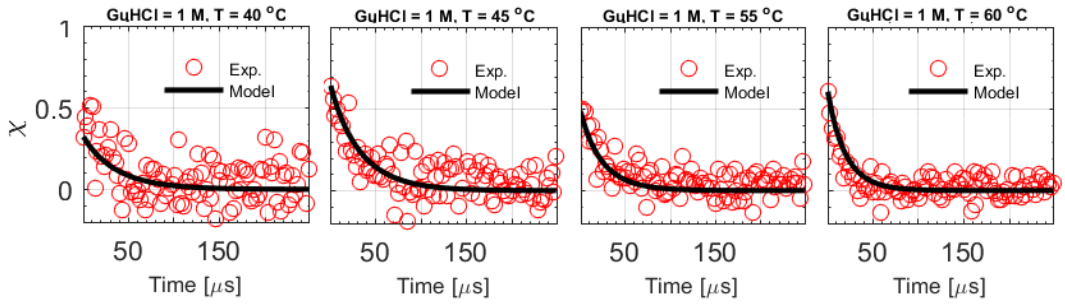



**Fig. S4.** Kinetic traces for the tetramer (His tag), with global three-states-per-domain fit.

### b. Complete thermodynamic melt and kinetics traces (model with two states per domain)

As discussed in the main text, the two-state fit is generally inferior to the three-state fit. This is easily evident by comparing how
(a) the fitted curves in Fig. 3 (main text) go through the data, *vs.* how they do in Fig. 6d below;
(b) the fitted curves in Fig. 5 (main text) and Figs. 6abs go through the data, *vs*. how they do in Figures S6e-h. Note in particular the rightmost panel in S6e, the last two rows and third column in S6f and S6g.

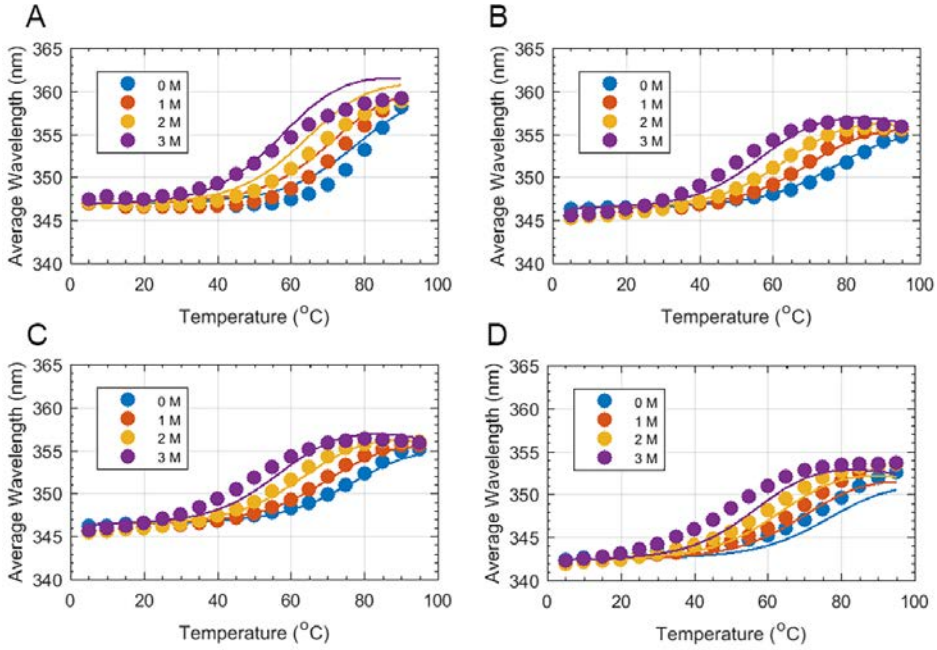



**Fig. S5.** Two-states-per-domain global fitting (A-D) of the thermal melts at different GuHCl concentrations and comparison at 2 M GuHCl (E). (A) MFiP35; (B) DFiP35; (C) TFiP(35); (D) QFiP35 expressed with a His tag. The curves are from the global multi-domain model fit of all thermodynamic and kinetic data of all *n*-mers, assuming two states per domain.

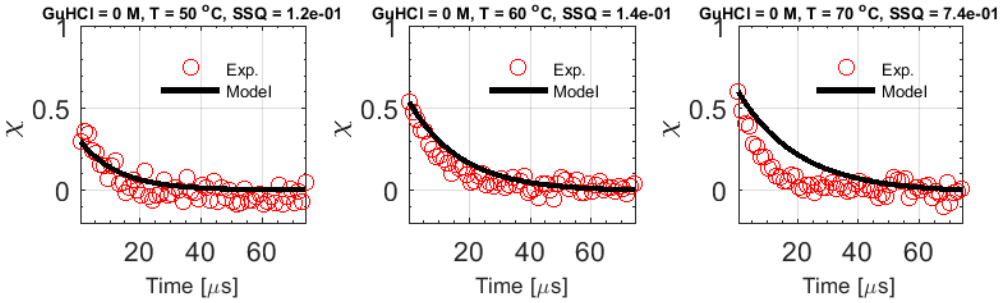



**Fig. S6.** Kinetic traces for the monomer, with global two-states-per-domain fit (black curve).

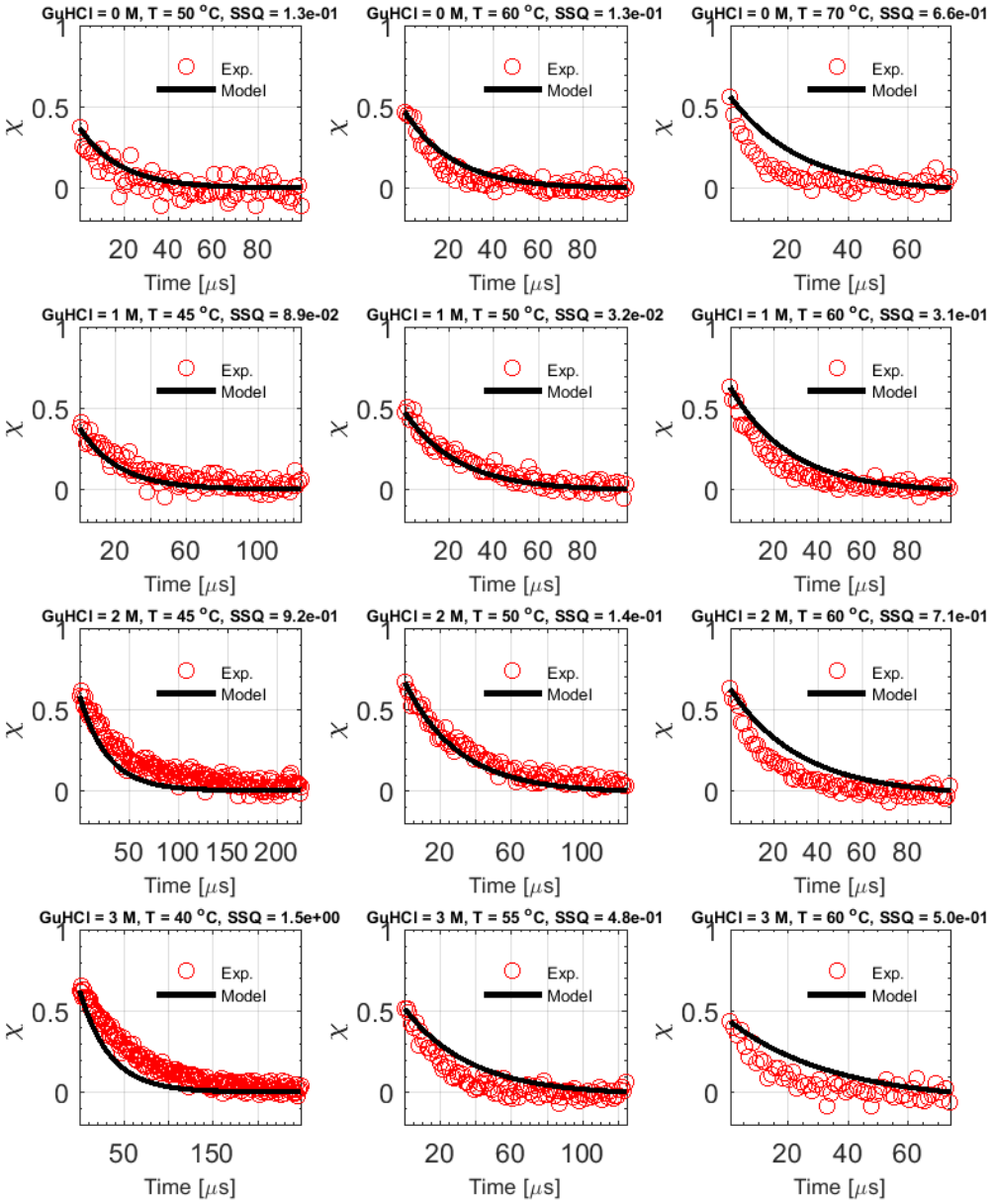



**Fig. S7.** Kinetic traces for the dimer, with global two-states-per-domain fit (black curve).

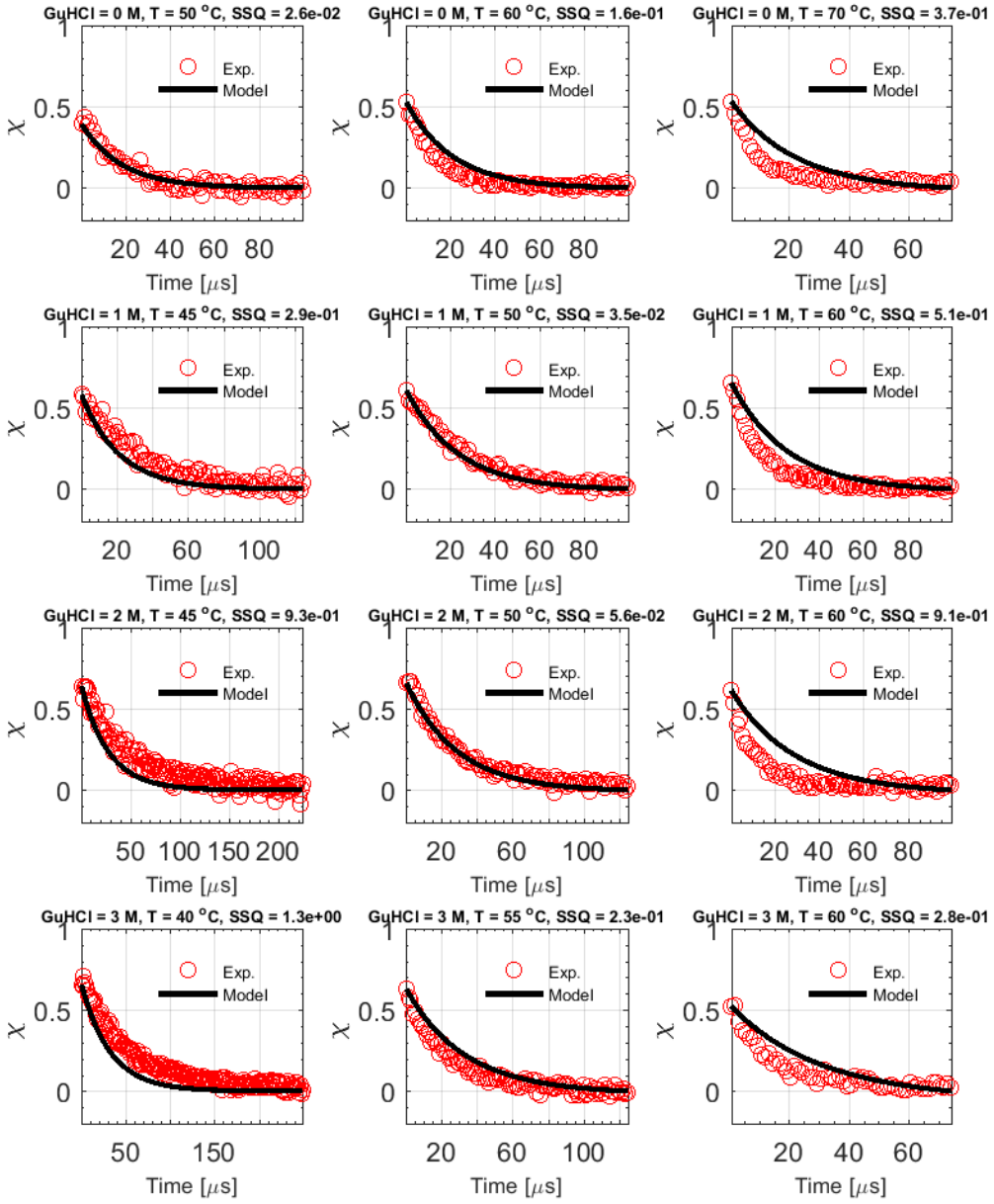



**Fig. S8.** Kinetic traces for the trimer, with global two-states-per-domain fit (black curve).

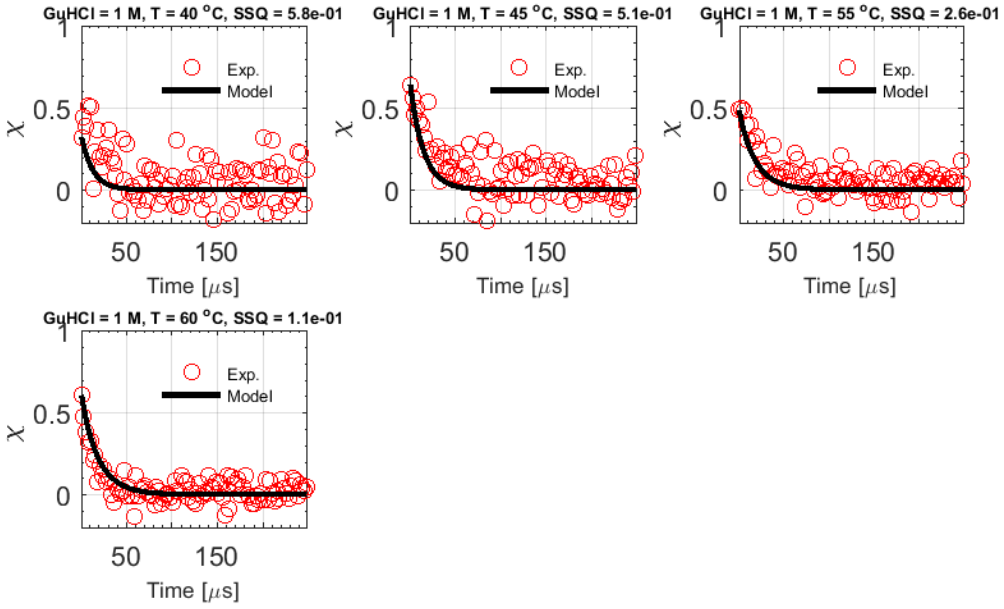



**Fig. S9.** Kinetic traces for the tetramer (*His tag), with global two-states-per-domain fit (black curve).

## 7. Complete set of domain state model populations

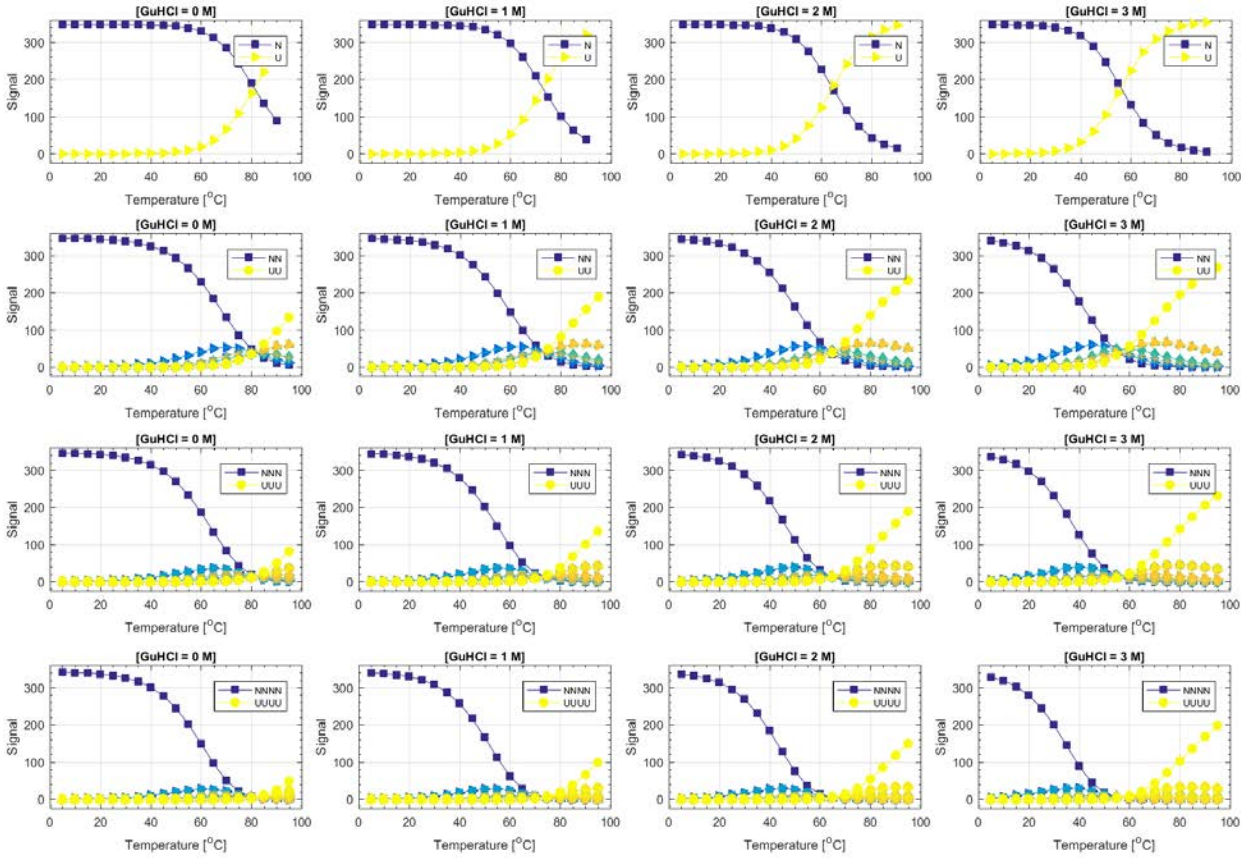



**Fig. S10.** Complete set of traces analogous to Fig. 6 in the main text, derived from fitting 47 thermal melts and kinetic relaxation traces.

## 8. Fraction of native state achieved by WW monomer in models I, II and III

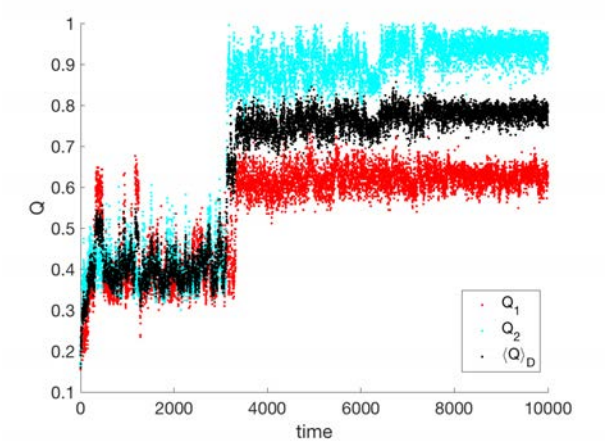



**Figure S11.** WW-domain dimer simulated annealing trajectory with respect to fraction of native contacts Q for domains 1, 2, and average of both domains using model II. Trajectories start at 650 K and are gradually cooled to 300 K.

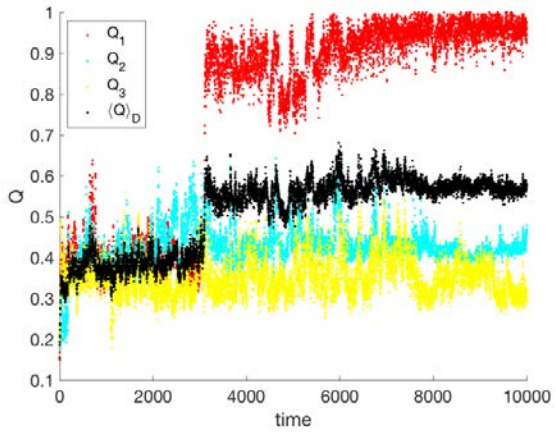



**Figure S12.** WW-domain trimer simulated annealing trajectory with respect to fraction of native contacts Q for domains 1, 2, 3, and average of all three domains using model II. Trajectories start at 650 K and are gradually cooled to 300 K.

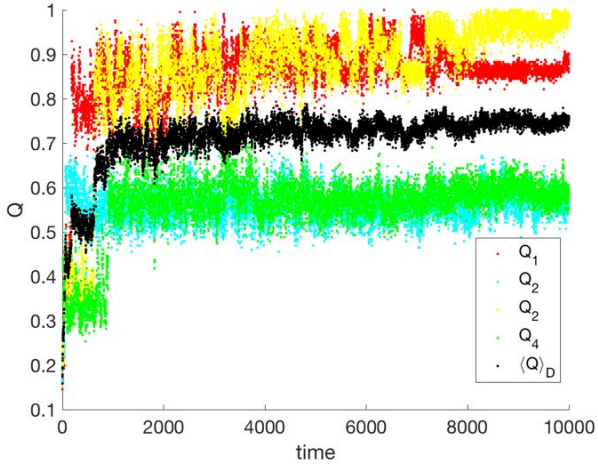



**Figure S13.** WW-domain trimer simulated annealing trajectory with respect to fraction of native contacts Q for domains 1, 2, 3, 4, and average of all four domains using model II. Trajectories start at 650 K and are gradually cooled to 300 K.