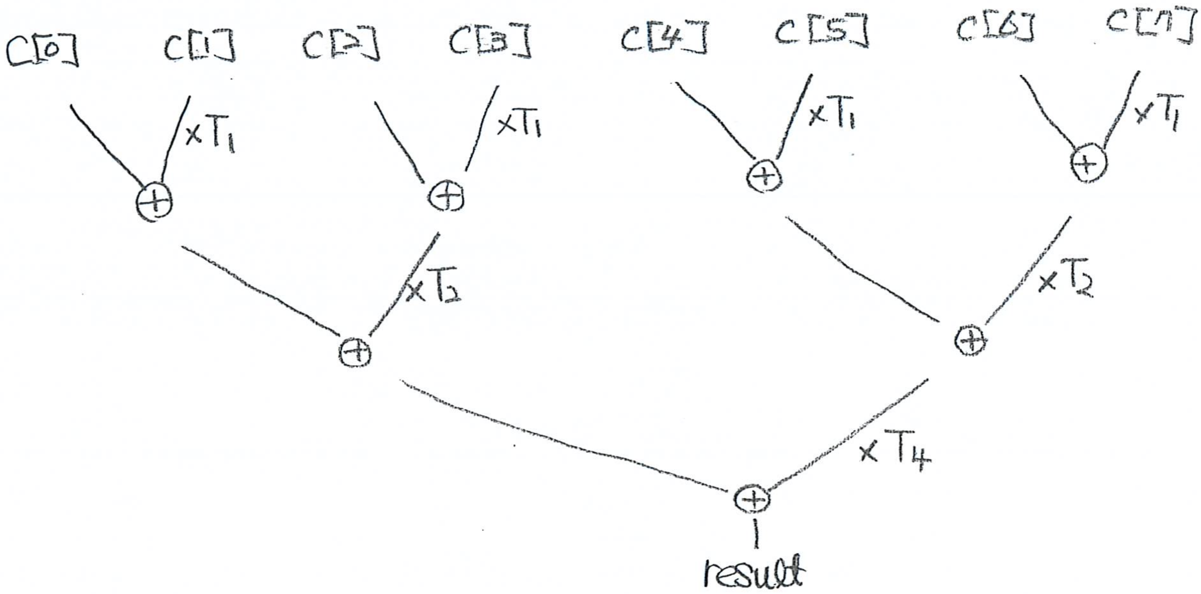
Conversion of polynomials

From $\displaystyle\sum_{i=0}^{N-1} c[i]\, T_i(x)$, [T_i : i^{th} Chebyshev polynomial]

To $\displaystyle\sum_{i=0}^{\frac{N}{2}-1} c[i]\, T_i(x) + T_{\frac{N}{2}}(x) \cdot \sum_{i=0}^{\frac{N}{2}-1} c[\tfrac{N}{2}+i] \cdot T_i(x)$

The conversion is recursively repeated to the binary-tree evaluation form, one of which is shown when N=8



```
void convert_poly_to_binarytreeform(double* c, int N){
    if(N==2) return;
    for(int i=0; i<N/2; i++){
        c[i] -= c[N/2+i];
        c[N/2+i] *= 2;
    }
    ----------------------------------------
    convert_poly_to_binarytreeform(c, N/2);
               "                  (c+N/2, N/2);
}
```

power of 2

$T_{\frac{N}{2}+i} = 2T_{\frac{N}{2}} \cdot T_i - T_i$

$\therefore c[\tfrac{N}{2}+i]\, T_{\frac{N}{2}+i} = T_{\frac{N}{2}} \cdot [c[\tfrac{N}{2}+i] \cdot 2 \cdot T_i] - T_i \cdot c[\tfrac{N}{2}+i]$

recursive application

- Chinese Remainder Theorem

Given $a[i] \in \mathbb{Z}_{q_i}$, $i = 0, \cdots, L-1$, there exists a unique $a \in \mathbb{Z}_Q$ $[Q = q_0 \times \cdots \times q_{L-1}]$ s.t. $\mathrm{mod}(a, q_i) = a[i], \forall i$.

$$a = \sum_{i=0}^{L-1} a[i] \left(\frac{Q}{q_i}\right) \mathrm{invmod}\left(\frac{Q}{q_i}, q_i\right) \quad (\mathrm{mod}\ Q)$$

- ModUp operation, $a \in \mathbb{Z}_Q \Rightarrow \tilde{a} \in \mathbb{Z}$

- $\tilde{a}$ is not unique. $\tilde{a} = a + Q \cdot I$, $I \in \mathbb{Z}$
- Want to find $\tilde{a}$ with I as least as possible.
- Typically,

$$\tilde{a} = \underbrace{\sum_{i=0}^{L-1} \underbrace{a[i] \cdot \mathrm{invmod}\left(\frac{Q}{q_i}, q_i\right)}_{\text{in } [-\frac{q_i}{2}, \frac{q_i}{2})} \left(\frac{Q}{q_i}\right)}_{\text{in } [-\frac{Q}{2}, \frac{Q}{2})}$$

Then $\tilde{a} = a + QI$, $|I| < \frac{L}{2}$.

- Keyswitching

$pt \xrightarrow{S_{fr}} ct = [ct_0, ct_1] \in R_Q$

$P \cdot S_{fr} + e \xrightarrow{S_{to}} swk \in R_{QP}$

Given ct and swk

$[Q \cong P]$

$\underbrace{e * I}_{\text{small}} + pt \cong$

$ct_1 \in R_Q \xrightarrow{\text{modUp}} ct_1 + QI \in R_{PQ}$ ($\tilde{ct}_1$)

$P S_{fr} * ct_1$
$+ \cancel{PQ S_{fr} * I}$
$e * ct_1 + e * QI$

$\xrightarrow{S_{to}}$ $\tilde{ct}_1 * swk \in R_{QP}$

$\Big|$ RS

$\|$

$S_{fr} * ct_1 \quad e * I \lfloor \frac{Q}{P} \rceil$
$+ \frac{e * ct_1}{P} \cong 0$

$\xleftarrow{S_{to}}$ $RS(\tilde{ct}_1 * swk) \in R_Q$

pt
$\|$
$ct_0 + S * ct_1$
$+ e * I \lfloor \frac{Q}{P} \rceil$

$\xleftarrow{S_{to}}$

$ks(ct)$
$\|$
$\boxed{[ct_0, 0] + RS(\tilde{ct}_1 * swk) \in R_Q}$

```
template <int N, int L, int K>
void modUp(const uint64_t q[L],
           const uint64_t p[K],
      const uint64_t a[L][N],
           uint64_t ã[L+K][N]) {
```

$a \in \mathbb{Z}_Q,\quad Q = q_0 \times \cdots \times q_{L-1}$

$P = p_0 \times \cdots \times p_{K-1}$

$\tilde{a} \in \mathbb{Z}_{QP}$

$\llcorner\ \sum_{i=0}^{L-1} \underbrace{a[i]\ \mathrm{inv}\left(\frac{Q}{q_i}, q_i\right)}_{\text{in } \mathbb{Z}_{q_i}} \cdot \frac{Q}{q_i}$

$a \in \mathbb{Z}_Q[x]/_{x^N+1} \Rightarrow \tilde{a} \in \mathbb{Z}_{QP}[x]/_{x^N+1}$

```
uint64_t Qmod[L][K];
for(int j=0; j<L; j++)
for(int k=0; k<K; k++){
   Qmod[j][k] = 1;
   for(int i=0; i<L; i++)
      if(i!=j) Qmod[j][k] = mul_mod(Qmod[j][k],
                          q[i]% p[k], p[k]);
}
```

$Qmod[j][k] = \mathrm{mod}\left(\frac{Q}{q[j]}, p[k]\right)$

```
uint64_t invQmod[L];
for(int j=0; j<L; j++){
   invQmod[j] = 1;
   for(int i=0; i<L; i++)
      if(i!=j) invQmod[j] = mul_mod(invQmod[j],
                      inv_mod(q[i], q[j]), q[j]);
}
```

$invQmod[j] = \mathrm{inv}\left(\frac{Q}{q_j}, q_j\right)$

```
uint64_t QmodP[K];
for(int k=0; k<K; k++){ QmodP[k]=1;
   for(int j=0; j<L; j++)
      QmodP[k] = mul_mod(QmodP[k], q[j]%p[k], p[k]);
}
```

$QmodP[k] = \mathrm{mod}(Q, p[k])$

```
for(int i=0; i<N; i++){
   uint64_t b[L]; int count=0;
   for(int j=0; j<L; j++){
      b[j] = mul_mod(a[j][i], invQmod[j], q[j]);
      if(2*b[j] >= q[j]) count++;
   }
```

$a[i] \in \mathbb{Z}_Q \Rightarrow \tilde{a}[i] \in \mathbb{Z}_{QP}$

$\tilde{a} = \sum_{i=0}^{L-1} \underbrace{\boxed{a[i] \cdot \mathrm{inv}\left(\frac{Q}{q_i}, q_i\right)}}_{=\ b[i] \in [0, q_i]} \frac{Q}{q_i} = \sum_{i=0}^{L-1} b[i] \cdot \frac{Q}{q_i} - \mathrm{count} \cdot Q$

$b[i] - 1 q_i \in \left[-\frac{q_i}{2}, \frac{q_i}{2}\right]$

```
for(int k=0; k<L; k++) ã[k][i] = a[k][i];
```

$\mathrm{mod}(\tilde{a}, Q) = \mathrm{mod}(a, Q)$

```
for(int k=0; k<K; k++){
    ã[L+k][i]=0;
    for(int j=0; j<L; j++)
        ã[L+k][i] += mul_mod(b[j] % p[k],
                        Qmod[j][k], p[k]);


    if(count>0)
        ã[L+k][i] += mul_mod(QmodP[k], p[k]-count, p[k]);
    ã[L+k][i]= ã[L+k][i] % p[k];
}
```

$\mathrm{mod}(\tilde{a}, p_k)$

$= \sum_{j=0}^{L-1} b[j]\left(\frac{Q}{q_j}\right) - \mathrm{count}.\ Q \quad \text{in } \mathbb{Z}_{p_k}$

}

}

◦ Gadget decomposition

$$Q = \underbrace{q_0 q_1}_{D_0}\ \underbrace{q_2 q_3}_{D_1}\ \underbrace{q_4 q_5}_{D_2} \qquad (\text{e.g. } dnum = 3)$$

$$P = P_0 P_1$$

$$\vec{g} \in \mathbb{Z}_{QP}^{dnum}, \quad \vec{g} = [D_1 D_2\, \text{inv\_mod}(D_1 D_2, D_0),\ D_0 D_2\, \text{inv\_mod}(D_0 D_2, D_1),\ D_0 D_1\, \text{inv\_mod}(D_0 D_1, D_2)]$$

$$\vec{g^{-1}} : \mathbb{Z}_Q \to \mathbb{Z}_{QP}^{dnum}, \quad \vec{g^{-1}(a)} = [\,\text{modUp}_{D_0 \to QP}([a]_{D_0}),\ \text{modUp}_{D_1 \to QP}([a]_{D_1}),\ \text{modUp}_{D_2 \to QP}([a]_{D_2})\,]$$

$$\begin{aligned} \vec{g} \circ \vec{g^{-1}(a)} = {} & g^{-1}(a)[0] \cdot D_1 D_2\, \text{inv\_mod}(D_1 D_2, D_0) \\ & + g^{-1}(a)[1]\ D_0 D_2\, \text{inv\_mod}(D_0 D_2, D_1) \\ & + g^{-1}(a)[2]\ D_0 D_1 \cdot \text{inv\_mod}(D_0 D_1, D_2) = a \pmod{Q} \end{aligned}$$

◦ Keyswitching with Gadget decomposition

$$\text{Given } S_{fr}, S_{to} \in \mathbb{Z}[x]/_{x^N+1} \Rightarrow \underbrace{P\vec{g}}_{\text{constant}} \cdot S_{fr} \xrightarrow{\text{encoding}} \vec{swk} = \text{Enc}(P\vec{g} \cdot S_{fr}, S_{to})$$

$$\text{Given } ct = [ct_0, ct_1] \in R_Q^2, \quad \vec{g^{-1}(ct_1)} \in R_{QP}^{DNUM}$$

$$\begin{array}{ccc} P\vec{g} \cdot S_{fr} + \vec{e} & \overset{S_{to}}{\text{———}} & \vec{swk} \in (R_{QP}^2)^{DNUM} \\ | & & | \\ P\vec{g} \cdot S_{fr} * \vec{g^{-1}(ct_1)} + \vec{e} * \vec{g^{-1}(ct_1)} & \overset{S_{to}}{\text{———}} & \vec{g^{-1}(ct_1)} * \vec{swk} \in (R_{QP}^2)^{DNUM} \end{array}$$

```
template<int N, int L, int DNUM>
void swkgen( const int Sfr[N],
             const int Sto[N],
             const uint64_t q[L],
             const uint64_t p[L/DNUM],
                 uint64_t swk̂[DNUM][2][L+(L/DNUM)][N]) {
  const int K=L/DNUM;
  uint64_t g[DNUM][L+K];
  gadget_g<L, DNUM>(q, p, g);
```

$\vec{g} \in \mathbb{Z}_{PQ}^{DNUM}$

```
  for(int n=0; n<DNUM; n++){
      uint64_t pt[L+K][N];
      for(int j=0; j<L; j++) {
```

$pt = P \cdot S_{fr}\, g[n]$

mod$(P, q[j])$

```
          uint64_t P = 1;
          for(int k=0; k<K ; k++)
              P = mul_mod(P, p[k]%q[j], q[j]);
```

$pt = P \cdot g[n] \cdot S_{fr}$

```
          uint64_t Pg = mul_mod(P, g[n][j], q[j]);
          for(int i=0; i<N; i++)
              pt[j][i] = mul_mod((q[j]+sfr[i])%q[j], Pg, q[j]);
      }
```

mod$(pt, P) = 0$

```
      for(int j=0; j<K; j++)
      for(int i=0; i<N; i++)
          pt[j+L][i]=0;
```

$\vec{\hat{swk}} = \text{enc}_{S_{to}}(P S_{fr} \vec{g})$

```
      uint64_t qp[L+K];
      for(int j=0; j<L; j++) qp[j]=q[j];
      for(int j=0; j<K ;j++) qp[L+j]= p[j];

      enc<N, L+K>(pt, Sto, qp, swk̂[n]);
  }
}
```

```
template<int N, int L, int DNUM>
void RS ( const uint64t  g[L],  const uint64t  p[L/DNUM],
          const uint64t  ŝwk[DNUM][2][L+(L/DNUM)][N],
          const uint64_t ĉt       [2][L][N],
                uint64_t ôut      [2][L][N]){
```

$a = ct_1$

```
uint64t  a[L][N];
for(int i=0; i<L; i++)
for(int j=0; j<N; j++)   a[i][j]= ĉt[1][i][j];
intt<N,L>(g,a);
```

$\vec{g}^{-1}(a)$

```
const int K= L/DNUM;
uint64t ginva[DNUM][L+K][N];
gadget_ginv<N,L,DNUM>(g,p,a, ginva);
```

gp=[g,p]

```
uint64_t gp[L+K];  for(int j=0; j<L; j++) gp[j]= g[j];
                   for(int j=0; j<K; j++) gp[L+j]= p[j];
```

```
uint64t temp[2][L+K][N];  uint64t temp_rs[2][L][N];
for( int d=0; d<DNOM; d++){
```

$temp = \vec{g}^{-1}(a)[d] * swk[d]$

```
ntt<N,L+K>(gp, ginva[d]);
for(int i=0; i<L+K; i++)
for(int j=0; j<N; j++){
    temp[0][i][j]= mul_mod(ginva[d][i][j], ŝwk[d][0][i][j], gp[i]);
    temp[1][i][j]= mul_mod(ginva[d][i][j], ŝwk[d][1][i][j], gp[i]);
}
```

$out = \sum_{d=0}^{DNUM-1} temp(d)$

```
RS_hat<N,L+K,L>(gp, temp, temp_rs);
for(int i=0; i<L; i++)
for(int j=0; j<N; j++){
    if(d==0){ ôut[0][i][j]=temp_rs[0][i][j];
              ôut[1][i][j]=temp_rs[1][i][j]; }
    else    { ôut[0][i][j]= (ôut[0][i][j] + temp_rs[0][i][j])%q[i];
              ôut[1][i][j]= (ôut[1][i][j] + temp_rs[1][i][j])%q[i];
}
}
```

$out = out + [c_0, 0]$

```
for(int i=0; i<L; i++)
for(int j=0; j<N; j++)  ôut[0][i][j]= (ôut[0][i][j] + ĉt[0][i][j]) % q[i];
}
```

```
template <int L, int DNUM>
void gadget_g (const uint64_t q[L],
               const uint64_t P[L/DNUM],
                     uint64_t g[DNUM][L+(L/DNUM)]) {

  const int K = L/DNUM;   assert(L%DNUM == 0);

  for (int d=0; d<DNUM; d++)
  for (int i=0; i<L+K ; i++) {
      if( d*K <= i  &&  i < (d+1)*K)  g[d][i] = 1;
      else  g[d][i] = 0;
  }
}
```

$g = [g_0, g_1, \cdots, g_{DNUM-1}] \in \mathbb{Z}_{QP}^{DNUM}$

$g_i \equiv 1 \pmod{D_i}$

$g_i \equiv 0 \pmod{D_j \text{ s.t } i \neq j}$

```
template <int N, int L, int DNUM>
void gadget_ginv ( const uint64_t q[L],
                   const uint64_t p[L/DNUM],
                   const uint64_t a[L][N],
                         uint64_t ginva[DNUM][L+(L/DNUM)][N]) {
    const int K = L/DNUM;
    uint64_t QP[L+K];
    for (int i=0; i<L; i++) QP[i] = q[i];
    for (int i=0; i<K; i++) QP[L+i] = p[i];

    for (int d=0; d<DNUM; d++){
        for(int i=0; i<K; i++){
            QP[i] = QP[d*K+i];
            QP[d*K+i] = q[i];
        }
        uint64_t a_Dd[K][N];
        for (int i=0; i<K; i++)
        for (int j=0; j<N; j++)
            a_Dd[i][j] = a[d*K+i][j];
        modUp<N,K,L>(QP, QP+K, a_Dd, ginva[d]);
                     ↑     ↑
                     Dd    QP/Dd

        for (int i=0; i<K; i++)
        for (int j=0; j<N; j++){
            uint64_t temp = ginva[d][i][j];
            ginva[d][i][j] = ginva[d][d*K+i][j];
            ginva[d][d*K+i][j] = temp;
        }
        for (int i=0; i<K; i++){
            uint64_t temp = QP[i];
            QP[i] = QP[d*K+i];
            QP[d*K+i] = temp;
        }
    }
}
```