In [1]:
```python
import numpy as np
from RiskAnalysis import RiskAnalysis
import matplotlib.pyplot as plt
```

Analysis One

In [2]:
```python
# there is a random seed embedded in each analysis
# result is replicable
```

In [3]:
```python
analysis_one = RiskAnalysis(T_total=5070, T=650, T_out=39, inputRets='returns.txt')
analysis_one.start_simulation()
```

```
Standardizing the returns
Evaluating empirical covariance estimator
0.0% done
8.902% done
17.8% done
26.71% done
35.61% done
44.51% done
53.41% done
62.31% done
71.22% done
80.12% done
89.02% done
97.92% done
100.00% done
Evaluating eigenvalues clipping estimator
0.0% done
8.902% done
17.8% done
26.71% done
35.61% done
44.51% done
53.41% done
62.31% done
71.22% done
80.12% done
89.02% done
97.92% done
100.00% done
Evaluating optimal shrinkage estimator
0.0% done
8.902% done
17.8% done
26.71% done
35.61% done
44.51% done
53.41% done
62.31% done
71.22% done
80.12% done
89.02% done
97.92% done
100.00% done
Evaluating exponential weighting estimator
0.0% done
8.902% done
17.8% done
26.71% done
35.61% done
44.51% done
53.41% done
62.31% done
71.22% done
80.12% done
89.02% done
97.92% done
100.00% done
Finish the simulation for all covariance estimators
```

Analysis One Mean

In [4]:
```
print(np.mean(np.sqrt(analysis_one.var_emp_mvp) * np.sqrt(78*252)))
print(np.mean(np.sqrt(analysis_one.var_ev_clip_mvp) * np.sqrt(78*252)))
print(np.mean(np.sqrt(analysis_one.var_os_mvp) * np.sqrt(78*252)))
print(np.mean(np.sqrt(analysis_one.var_expo_mvp) * np.sqrt(78*252)))
print('\n')
print(np.mean(np.sqrt(analysis_one.var_emp_omni) * np.sqrt(78*252)))
print(np.mean(np.sqrt(analysis_one.var_ev_clip_omni) * np.sqrt(78*252)))
print(np.mean(np.sqrt(analysis_one.var_os_omni) * np.sqrt(78*252)))
print(np.mean(np.sqrt(analysis_one.var_expo_omni) * np.sqrt(78*252)))
print('\n')
print(np.mean(np.sqrt(analysis_one.var_emp_rand) * np.sqrt(78*252)))
print(np.mean(np.sqrt(analysis_one.var_ev_clip_rand) * np.sqrt(78*252)))
print(np.mean(np.sqrt(analysis_one.var_os_rand) * np.sqrt(78*252)))
print(np.mean(np.sqrt(analysis_one.var_expo_rand) * np.sqrt(78*252)))
```

```
0.0036959173408567607
0.22315623345390193
0.0033105602898594335
0.0037018958620569143


0.0007914462005613687
0.02069837389577778
0.0007872171214162059
0.0007920128450298429


0.00020445050330605027
0.006806342947466875
0.00016071593401288377
0.0002113284880324797
```

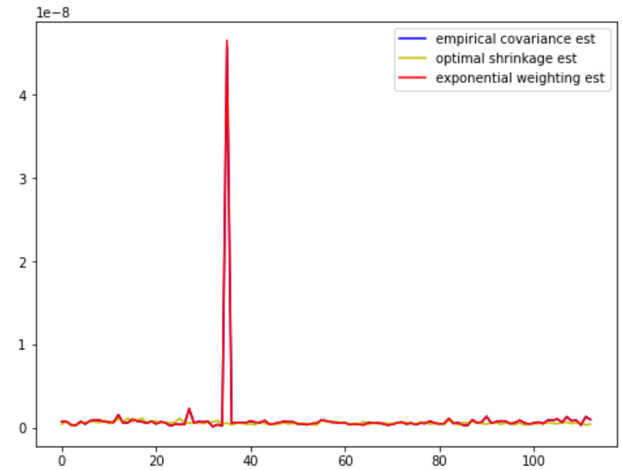
Analysis One Standard Deviation

In [5]:
```
print(np.std(np.sqrt(analysis_one.var_emp_mvp) * np.sqrt(78*252)))
print(np.std(np.sqrt(analysis_one.var_ev_clip_mvp) * np.sqrt(78*252)))
print(np.std(np.sqrt(analysis_one.var_os_mvp) * np.sqrt(78*252)))
print(np.std(np.sqrt(analysis_one.var_expo_mvp) * np.sqrt(78*252)))
print('\n')
print(np.std(np.sqrt(analysis_one.var_emp_omni) * np.sqrt(78*252)))
print(np.std(np.sqrt(analysis_one.var_ev_clip_omni) * np.sqrt(78*252)))
print(np.std(np.sqrt(analysis_one.var_os_omni) * np.sqrt(78*252)))
print(np.std(np.sqrt(analysis_one.var_expo_omni) * np.sqrt(78*252)))
print('\n')
print(np.std(np.sqrt(analysis_one.var_emp_rand) * np.sqrt(78*252)))
print(np.std(np.sqrt(analysis_one.var_ev_clip_rand) * np.sqrt(78*252)))
print(np.std(np.sqrt(analysis_one.var_os_rand) * np.sqrt(78*252)))
print(np.std(np.sqrt(analysis_one.var_expo_rand) * np.sqrt(78*252)))
```

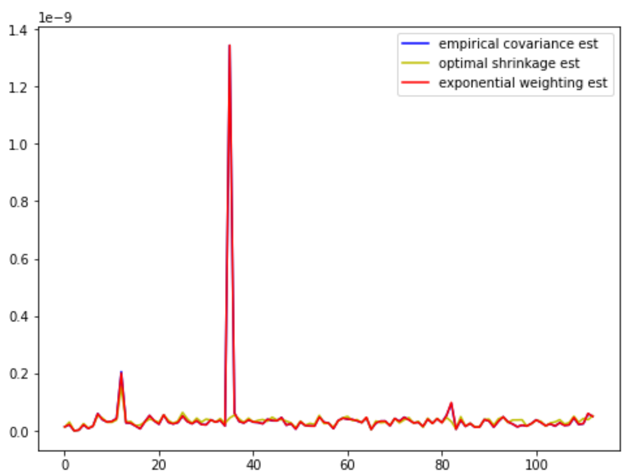
```
0.002588890468006907
0.03430614611689465
0.00048105136300690055
0.002609110471511013


0.00046907819079584467
0.013001479008498282
0.0001946849016764267
0.00046891151399521756


0.0001552535950850262
0.002052298587074055
3.069028007999694e-05
0.00015883119509302608
```

Plotting Analysis One Results

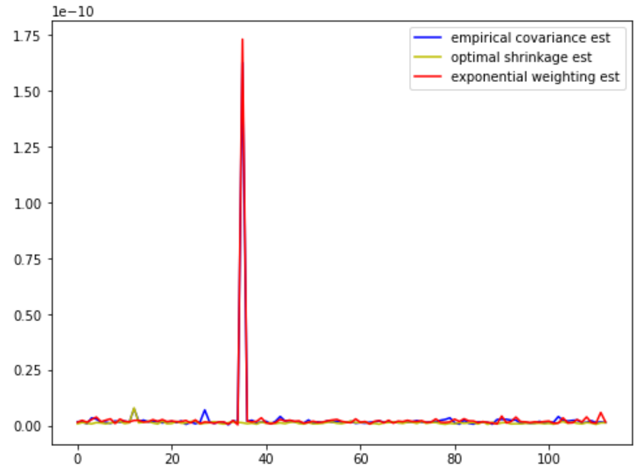
In [6]:
```
plt.figure(figsize=(8, 6))
plt.plot(analysis_one.var_emp_mvp, label = 'empirical covariance est', color = 'b')
# plt.plot(analysis_one.var_ev_clip_mvp, label = 'eigenvalue clipping cov', color ='g')
plt.plot(analysis_one.var_os_mvp, label = 'optimal shrinkage est', color ='y')
plt.plot(analysis_one.var_expo_mvp, label = 'exponential weighting est', color = 'r')
plt.legend()
plt.show()
```



In [7]:
```
plt.figure(figsize=(8, 6))
plt.plot(analysis_one.var_emp_omni, label = 'empirical covariance est', color = 'b')
# plt.plot(analysis_one.var_ev_clip_omni, label = 'eigenvalue clipping cov', color ='g')
plt.plot(analysis_one.var_os_omni, label = 'optimal shrinkage est', color ='y')
plt.plot(analysis_one.var_expo_omni, label = 'exponential weighting est', color = 'r')
plt.legend()
plt.show()
```



In [8]:
```
plt.figure(figsize=(8, 6))
plt.plot(analysis_one.var_emp_rand, label = 'empirical covariance est', color = 'b')
# plt.plot(analysis_one.var_ev_clip_rand, label = 'eigenvalue clipping cov', color ='g')
plt.plot(analysis_one.var_os_rand, label = 'optimal shrinkage est', color ='y')
plt.plot(analysis_one.var_expo_rand, label = 'exponential weighting est', color = 'r')
plt.legend()
plt.show()
```



In [9]:
```
# The exponential weighting estimator and empirical covariance estimators have highly similar results
# But it is very computationally expensive
```

Analysis Two

In [10]:
```
analysis_two = RiskAnalysis(T_total=5070, T=1010, T_out=60, inputRets='returns.txt')
analysis_two.start_simulation()
```

```
Standardizing the returns
Evaluating empirical covariance estimator
0.0% done
15.0% done
30.0% done
45.0% done
60.0% done
75.0% done
90.0% done
100.00% done
Evaluating eigenvalues clipping estimator
0.0% done
15.0% done
30.0% done
45.0% done
60.0% done
75.0% done
90.0% done
100.00% done
Evaluating optimal shrinkage estimator
0.0% done
15.0% done
30.0% done
45.0% done
60.0% done
75.0% done
90.0% done
100.00% done
Evaluating exponential weighting estimator
0.0% done
15.0% done
30.0% done
45.0% done
60.0% done
75.0% done
90.0% done
100.00% done
Finish the simulation for all covariance estimators
```

Analysis Two Mean

In [11]:
```
print(np.mean(np.sqrt(analysis_two.var_emp_mvp) * np.sqrt(78*252)))
print(np.mean(np.sqrt(analysis_two.var_ev_clip_mvp) * np.sqrt(78*252)))
print(np.mean(np.sqrt(analysis_two.var_os_mvp) * np.sqrt(78*252)))
print(np.mean(np.sqrt(analysis_two.var_expo_mvp) * np.sqrt(78*252)))
print('\n')
print(np.mean(np.sqrt(analysis_two.var_emp_omni) * np.sqrt(78*252)))
print(np.mean(np.sqrt(analysis_two.var_ev_clip_omni) * np.sqrt(78*252)))
print(np.mean(np.sqrt(analysis_two.var_os_omni) * np.sqrt(78*252)))
print(np.mean(np.sqrt(analysis_two.var_expo_omni) * np.sqrt(78*252)))
print('\n')
print(np.mean(np.sqrt(analysis_two.var_emp_rand) * np.sqrt(78*252)))
print(np.mean(np.sqrt(analysis_two.var_ev_clip_rand) * np.sqrt(78*252)))
print(np.mean(np.sqrt(analysis_two.var_os_rand) * np.sqrt(78*252)))
print(np.mean(np.sqrt(analysis_two.var_expo_rand) * np.sqrt(78*252)))
```

```
0.002388119651288764
0.22738690834903222
0.0027308370526825335
0.002414430876515071


0.000470803541027947
0.01692720880929969
0.0004907772015586883
0.00047194085671877666


0.00013692906850056133
0.007108141156086025
0.0001382612872663056
0.00013277525857687865
```

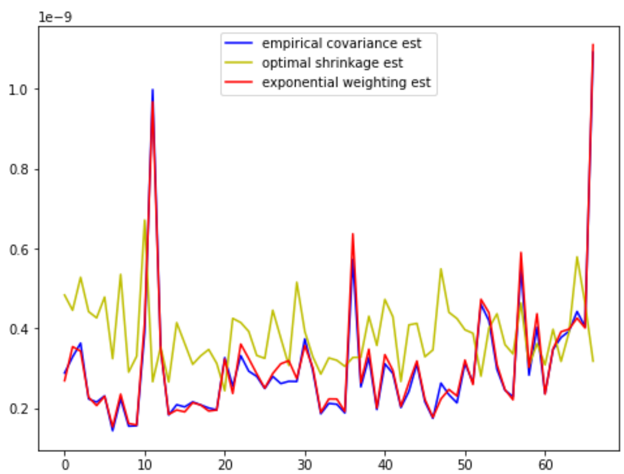
Analysis Two Standard Deviation

In [12]:
```python
print(np.std(np.sqrt(analysis_two.var_emp_mvp) * np.sqrt(78*252)))
print(np.std(np.sqrt(analysis_two.var_ev_clip_mvp) * np.sqrt(78*252)))
print(np.std(np.sqrt(analysis_two.var_os_mvp) * np.sqrt(78*252)))
print(np.std(np.sqrt(analysis_two.var_expo_mvp) * np.sqrt(78*252)))
print('\n')
print(np.std(np.sqrt(analysis_two.var_emp_omni) * np.sqrt(78*252)))
print(np.std(np.sqrt(analysis_two.var_ev_clip_omni) * np.sqrt(78*252)))
print(np.std(np.sqrt(analysis_two.var_os_omni) * np.sqrt(78*252)))
print(np.std(np.sqrt(analysis_two.var_expo_omni) * np.sqrt(78*252)))
print('\n')
print(np.std(np.sqrt(analysis_two.var_emp_rand) * np.sqrt(78*252)))
print(np.std(np.sqrt(analysis_two.var_ev_clip_rand) * np.sqrt(78*252)))
print(np.std(np.sqrt(analysis_two.var_os_rand) * np.sqrt(78*252)))
print(np.std(np.sqrt(analysis_two.var_expo_rand) * np.sqrt(78*252)))
```

```
0.0005150000393128671
0.031210128817816297
0.0002867206872216914
0.0005249878714983875


0.00012166625701168793
0.010455181747910432
0.00012092038202681932
0.00012041431742658429


3.627246244173789e-05
0.002242505690903154
1.7442931450695433e-05
2.652361862982941e-05
```

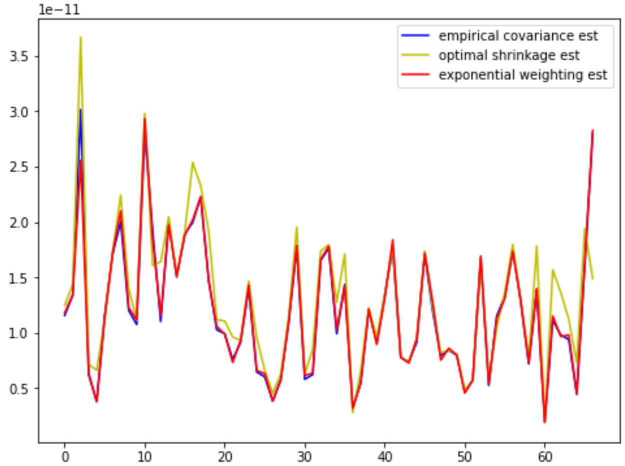
Plotting Analysis Two Results

In [13]:
```python
plt.figure(figsize=(8, 6))
plt.plot(analysis_two.var_emp_mvp, label = 'empirical covariance est', color = 'b')
# plt.plot(analysis_two.var_ev_clip_mvp, label = 'eigenvalue clipping cov', color ='g')
plt.plot(analysis_two.var_os_mvp, label = 'optimal shrinkage est', color ='y')
plt.plot(analysis_two.var_expo_mvp, label = 'exponential weighting est', color = 'r')
plt.legend()
plt.show()
```



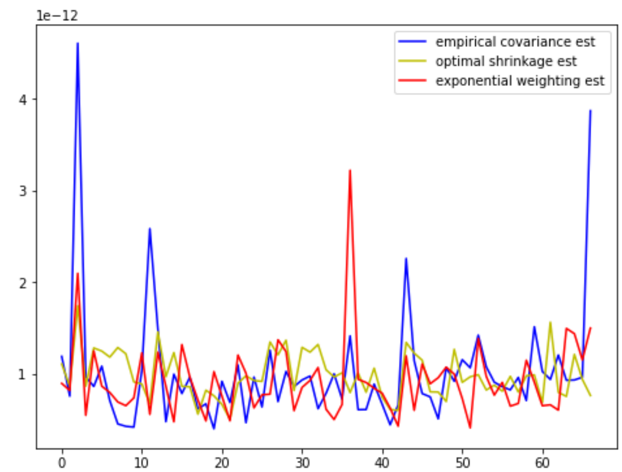
In [14]:

```
plt.figure(figsize=(8, 6))
plt.plot(analysis_two.var_emp_omni, label = 'empirical covariance est', color = 'b')
# plt.plot(analysis_two.var_ev_clip_omni, label = 'eigenvalue clipping cov', color ='g')
plt.plot(analysis_two.var_os_omni, label = 'optimal shrinkage est', color ='y')
plt.plot(analysis_two.var_expo_omni, label = 'exponential weighting est', color = 'r')
plt.legend()
plt.show()
```



In [15]:

```
plt.figure(figsize=(8, 6))
plt.plot(analysis_two.var_emp_rand, label = 'empirical covariance est', color = 'b')
# plt.plot(analysis_two.var_ev_clip_rand, label = 'eigenvalue clipping cov', color ='g')
plt.plot(analysis_two.var_os_rand, label = 'optimal shrinkage est', color ='y')
plt.plot(analysis_two.var_expo_rand, label = 'exponential weighting est', color = 'r')
plt.legend()
plt.show()
```



In [16]:

```
# empirical and exponential shrinkage looks different in this case because they have different uniform predictor
```

In [17]:

```
# The exponential weighting estimator and empirical covariance estimators have highly similar results
# But it is very computationally expensive
```

Analysis Three

In [18]:
```
analysis_three = RiskAnalysis(T_total=1258, T=600, T_out=36, inputRets='daily_returns.txt')
print(analysis_three.raw_returns.shape)
analysis_three.random_select_n_stocks(n=300)
print(analysis_three.raw_returns.shape)
analysis_three.start_simulation()
```

```
(470, 1258)
(300, 1258)
Standardizing the returns
Evaluating empirical covariance estimator
0.0% done
57.88% done
100.00% done
Evaluating eigenvalues clipping estimator
0.0% done
57.88% done
100.00% done
Evaluating optimal shrinkage estimator
0.0% done
57.88% done
100.00% done
Evaluating exponential weighting estimator
0.0% done
57.88% done
100.00% done
Finish the simulation for all covariance estimators
```

Analysis Three Mean

In [19]:
```
print(np.mean(np.sqrt(analysis_three.var_emp_mvp) * np.sqrt(252)))
print(np.mean(np.sqrt(analysis_three.var_ev_clip_mvp) * np.sqrt(252)))
print(np.mean(np.sqrt(analysis_three.var_os_mvp) * np.sqrt(252)))
print(np.mean(np.sqrt(analysis_three.var_expo_mvp) * np.sqrt(252)))
print('\n')
print(np.mean(np.sqrt(analysis_three.var_emp_omni) * np.sqrt(252)))
print(np.mean(np.sqrt(analysis_three.var_ev_clip_omni) * np.sqrt(252)))
print(np.mean(np.sqrt(analysis_three.var_os_omni) * np.sqrt(252)))
print(np.mean(np.sqrt(analysis_three.var_expo_omni) * np.sqrt(252)))
print('\n')
print(np.mean(np.sqrt(analysis_three.var_emp_rand) * np.sqrt(252)))
print(np.mean(np.sqrt(analysis_three.var_ev_clip_rand) * np.sqrt(252)))
print(np.mean(np.sqrt(analysis_three.var_os_rand) * np.sqrt(252)))
print(np.mean(np.sqrt(analysis_three.var_expo_rand) * np.sqrt(252)))
```

```
6.985435752702731
6.2815687562723
5.46641986334149
7.209895662332741


0.44982799921463124
0.43429619011158815
0.417104645203165
0.44989624034731573


0.08319468588657102
0.07613460192273566
0.06625858571570284
0.09124968387048099
```

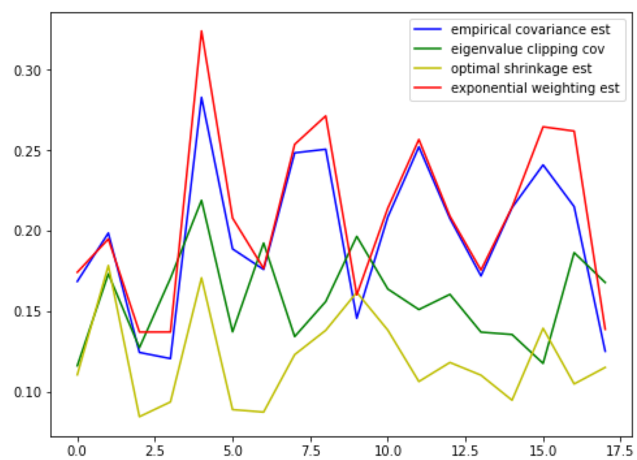
Analysis Three Standard Deviation

In [20]:
```
print(np.std(np.sqrt(analysis_three.var_emp_mvp) * np.sqrt(252)))
print(np.std(np.sqrt(analysis_three.var_ev_clip_mvp) * np.sqrt(252)))
print(np.std(np.sqrt(analysis_three.var_os_mvp) * np.sqrt(252)))
print(np.std(np.sqrt(analysis_three.var_expo_mvp) * np.sqrt(252)))
print('\n')
print(np.std(np.sqrt(analysis_three.var_emp_omni) * np.sqrt(252)))
print(np.std(np.sqrt(analysis_three.var_ev_clip_omni) * np.sqrt(252)))
print(np.std(np.sqrt(analysis_three.var_os_omni) * np.sqrt(252)))
print(np.std(np.sqrt(analysis_three.var_expo_omni) * np.sqrt(252)))
print('\n')
print(np.std(np.sqrt(analysis_three.var_emp_rand) * np.sqrt(252)))
print(np.std(np.sqrt(analysis_three.var_ev_clip_rand) * np.sqrt(252)))
print(np.std(np.sqrt(analysis_three.var_os_rand) * np.sqrt(252)))
print(np.std(np.sqrt(analysis_three.var_expo_rand) * np.sqrt(252)))
```

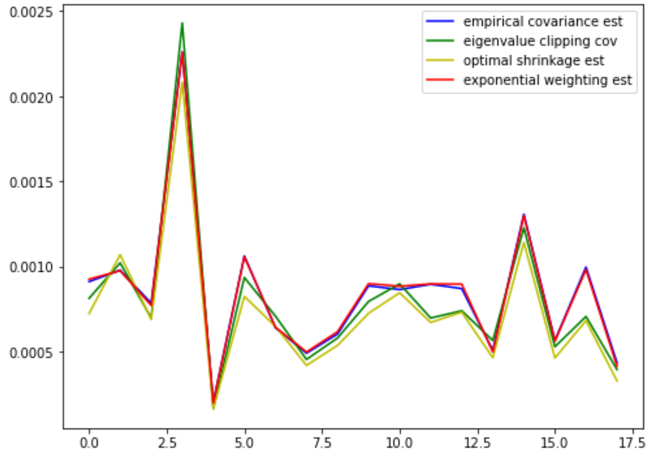
```
0.8586689658786676
0.5539733694578173
0.6247831160590127
0.8982811081707777


0.10875841136187818
0.11377269807685378
0.10717716373002388
0.10940208935595037


0.015384354867080745
0.012555988529566828
0.008010303650348766
0.018890357502658446
```

Plotting Analysis Three Results

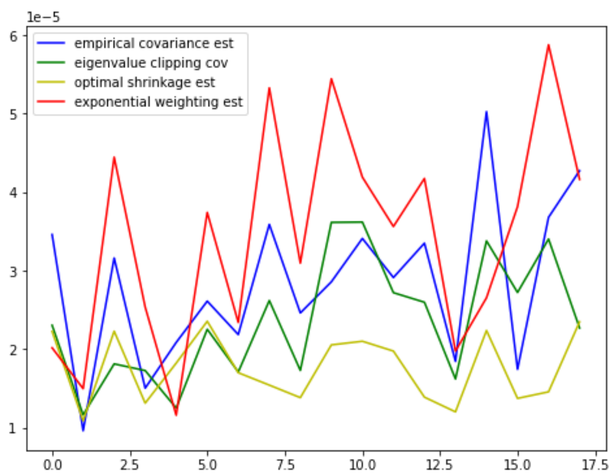
In [21]:
```
plt.figure(figsize=(8, 6))
plt.plot(analysis_three.var_emp_mvp, label = 'empirical covariance est', color = 'b')
plt.plot(analysis_three.var_ev_clip_mvp, label = 'eigenvalue clipping cov', color ='g')
plt.plot(analysis_three.var_os_mvp, label = 'optimal shrinkage est', color ='y')
plt.plot(analysis_three.var_expo_mvp, label = 'exponential weighting est', color = 'r')
plt.legend()
plt.show()
```



In [22]:
```
plt.figure(figsize=(8, 6))
plt.plot(analysis_three.var_emp_omni, label = 'empirical covariance est', color = 'b')
plt.plot(analysis_three.var_ev_clip_omni, label = 'eigenvalue clipping cov', color ='g')
plt.plot(analysis_three.var_os_omni, label = 'optimal shrinkage est', color ='y')
plt.plot(analysis_three.var_expo_omni, label = 'exponential weighting est', color = 'r')
plt.legend()
plt.show()
```



In [23]:
```
plt.figure(figsize=(8, 6))
plt.plot(analysis_three.var_emp_rand, label = 'empirical covariance est', color = 'b')
plt.plot(analysis_three.var_ev_clip_rand, label = 'eigenvalue clipping cov', color ='g')
plt.plot(analysis_three.var_os_rand, label = 'optimal shrinkage est', color ='y')
plt.plot(analysis_three.var_expo_rand, label = 'exponential weighting est', color = 'r')
plt.legend()
plt.show()
```



In [24]:
```
# empirical and exponential shrinkage looks different in this case because they have different uniform predictor
```

In [25]:
```
# The exponential weighting estimator and empirical covariance estimators have highly similar results
# But it is very computationally expensive
```

In [ ]:
```
```