In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns


#subprocess.run(["curve_finder_opt.cpp"])


#subprocess.run(["curve_finder.cpp"])


df = pd.DataFrame(pd.read_csv("bdcurves.txt"))
df2 = pd.DataFrame(pd.read_csv("curves4.txt"))
```

In [2]:
```
df
```

Out[2]:

	p	g	время_генерации_ключа_мкс	размер_ключа	общий_ключ	общее_
0	2617	2612	98	11	1681	
1	2269	2267	96	11	1456	
2	2083	2079	101	11	1150	
3	2593	2586	58	12	2152	
4	2887	2885	62	10	698	
...	...	...	...	...	...	
195	2621	2619	107	7	96	
196	3169	3162	82	12	2434	
197	3343	3332	129	12	2738	
198	2521	2504	87	11	1417	
199	2677	2675	107	10	832	

200 rows × 6 columns

In [3]:
```
df2
```

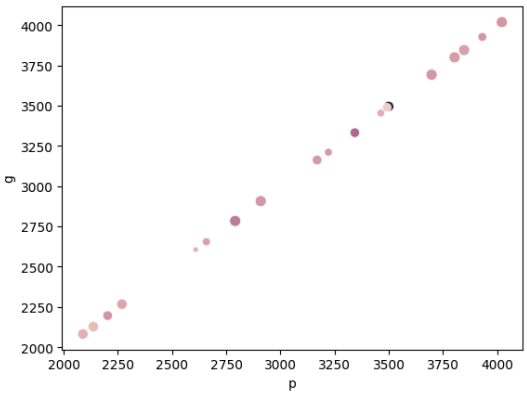
Out[3]:

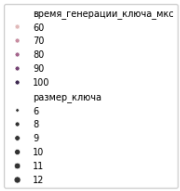
	a	b	p	время_генерации_общего_ключа_последовательным_умножени
0	494	2340	2617	
1	801	1691	2269	
2	751	1025	2083	
3	646	2138	2593	
4	2430	2668	2887	
...	...	...	...	
195	1901	231	2621	
196	1306	82	3169	
197	2301	194	3343	
198	481	1700	2521	
199	2385	979	2677	

200 rows × 10 columns

In [39]:
```
pl = df.loc[40:60]
#sns.scatterplot(data=pl, x=pl.p, y=pl.g, hue=pl["время_генерации_ключа_мкс"], s

sns.scatterplot(
    data=pl,
    x=pl["p"],
    y=pl["g"],
    hue=pl["время_генерации_ключа_мкс"],
    size=pl["размер_ключа"],
)

# перемещаем легенду вправо за пределы графика
plt.legend(
    bbox_to_anchor=(2, 1),
    loc='upper left',
    fontsize=7,          # размер текста легенды
    title_fontsize=7,    # размер заголовка
    markerscale=0.6      # масштаб точек в легенде
)
plt.tight_layout()
plt.show()
```

```
C:\Users\geezix\AppData\Local\Temp\ipykernel_12604\3124886779.py:20: UserWarning:
Tight layout not applied. The left and right margins cannot be made large enough
to accommodate all Axes decorations.
  plt.tight_layout()
```

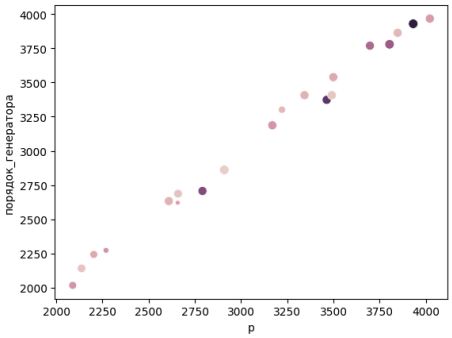




In [33]:
```
p = df2.loc[40:60]
#sns.scatterplot(data=p, x=p.p, y=p["порядок_генератора"], hue=p["время_генерации
```

```
sns.scatterplot(
    data=df,
    x=p["p"],
    y=p["порядок_генератора"],
    hue=p["время_генерации_общего_ключа_последовательным_умножением_мкс"],
    size=p["размер_ключа"],
)

# перемещаем легенду вправо за пределы графика
plt.legend(
    bbox_to_anchor=(2, 1),
    loc='upper left',
    fontsize=7,          # размер текста легенды
    title_fontsize=7,    # размер заголовка
    markerscale=0.6      # масштаб точек в легенде
)
plt.tight_layout()
plt.show()
```

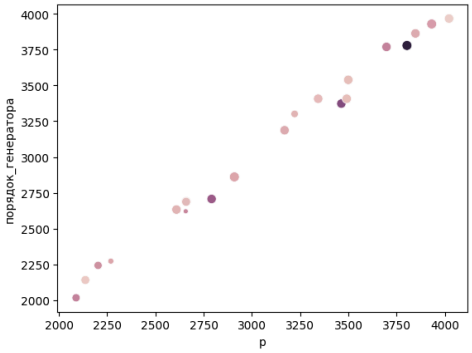
```
C:\Users\geezix\AppData\Local\Temp\ipykernel_12604\2478027468.py:23: UserWarning:
Tight layout not applied. The left and right margins cannot be made large enough
to accommodate all Axes decorations.
  plt.tight_layout()
```

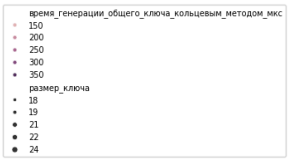


In [32]:
```
p = df2.loc[40:60]

#sns.scatterplot(data=p, x=p.p, y=p["порядок_генератора"], hue=p["время_генерации

sns.scatterplot(
    data=df,
    x=p["p"],
```

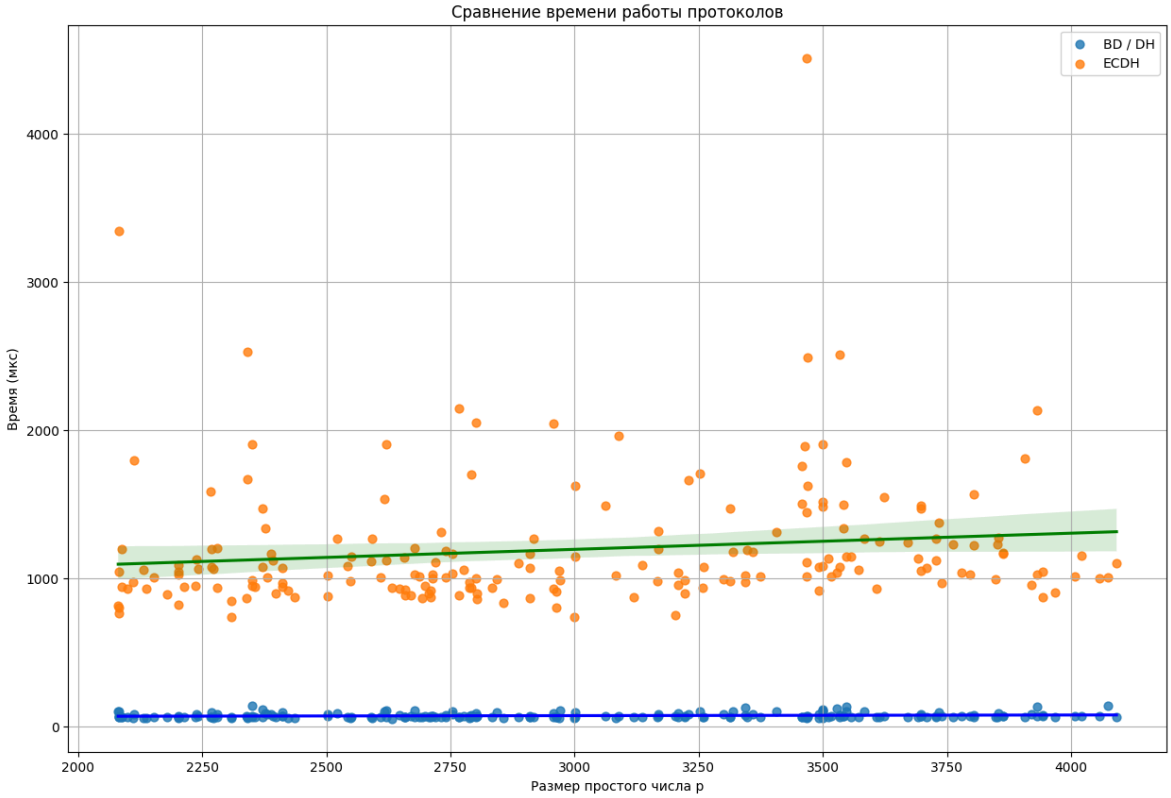
```
    y=p["порядок_генератора"],
    hue=p["время_генерации_общего_ключа_кольцевым_методом_мкс"],
    size=p["размер_ключа"],
)

# перемещаем легенду вправо за пределы графика
plt.legend(
    bbox_to_anchor=(2, 1),
    loc='upper left',
    fontsize=7,          # размер текста легенды
    title_fontsize=7,    # размер заголовка
    markerscale=0.6      # масштаб точек в легенде
)
plt.tight_layout()
plt.show()
```

```
C:\Users\geezix\AppData\Local\Temp\ipykernel_12604\1669518235.py:21: UserWarning:
Tight layout not applied. The left and right margins cannot be made large enough
to accommodate all Axes decorations.
  plt.tight_layout()
```

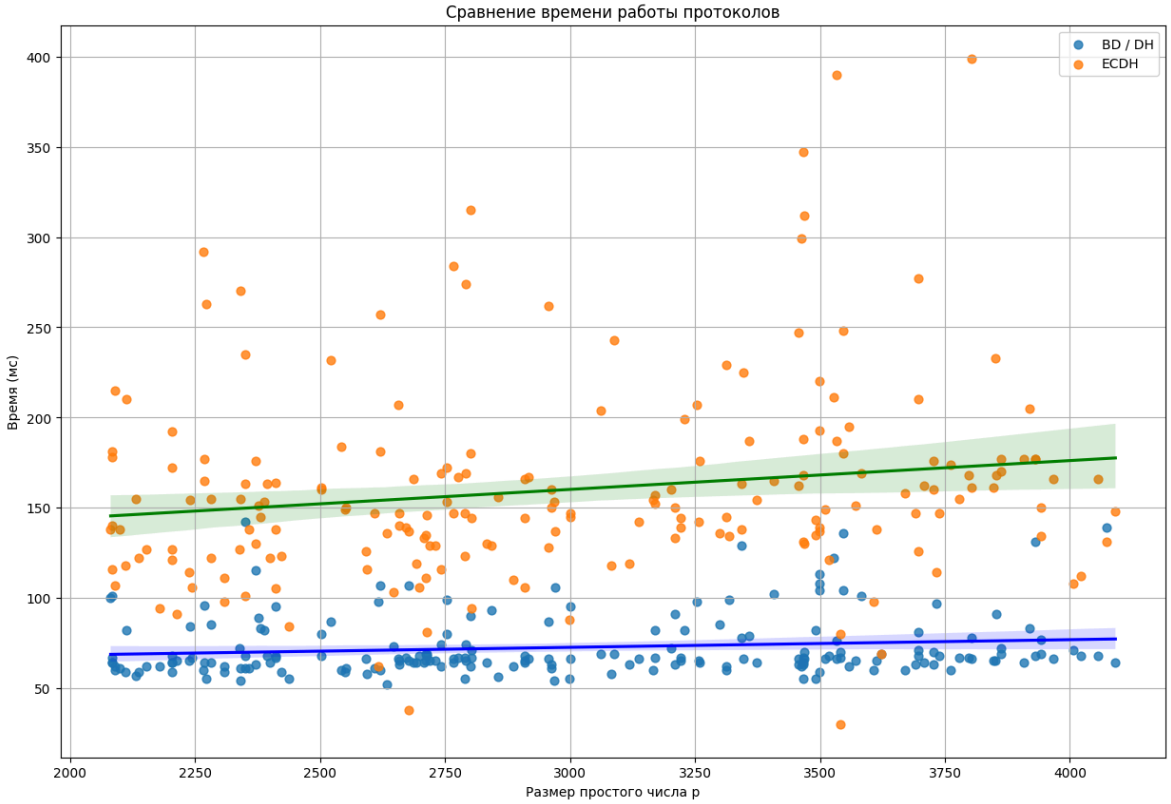




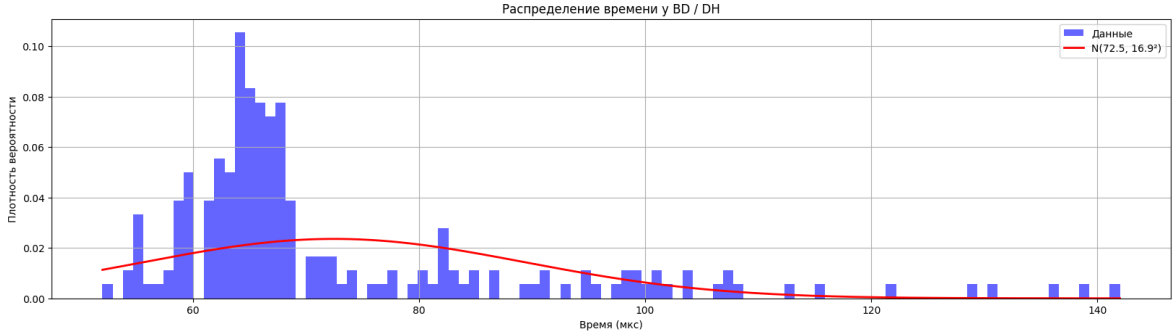
In [47]:
```
# Преобразуем названия столбцов в EC таблице, чтобы убрать лишние пробелы

#График сравния времени на генерацию ключей, все точки ECDH и BD соединены прямы
# График 1: сравнение времени (Diffie-Hellman vs ECDH)
plt.figure(figsize=(15,10))
sns.regplot(x='p', y='время_генерации_ключа_мкс', data=df, label='BD / DH', scat
sns.regplot(x='p', y='время_генерации_общего_ключа_последовательным_умножением_м
#plt.plot(df['p'], df['время_генерации_ключа_мкс'], 'o-', label='BD / DH (ms)')
#plt.plot(df2['p'], df2['время_генерации_ключа_мкс'], 'o-', label='ECDH (ms)')
plt.xlabel('Размер простого числа p')
plt.ylabel('Время (мкс)')
plt.title('Сравнение времени работы протоколов')
plt.legend()
plt.grid(True)
plt.show()
```



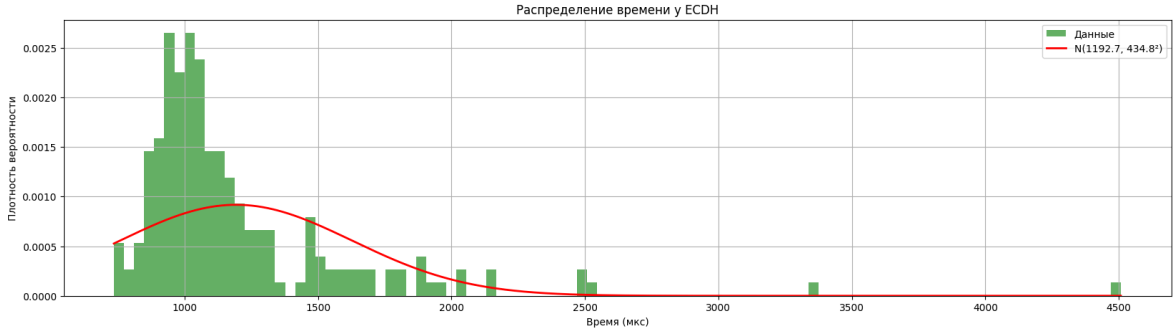
In [48]:
```
# Преобразуем названия столбцов в EC таблице, чтобы убрать лишние пробелы

#График сравния времени на генерацию ключей, все точки ECDH и BD соединены прямы
# График 1: сравнение времени (Diffie-Hellman vs ECDH)
plt.figure(figsize=(15,10))
sns.regplot(x='p', y='время_генерации_ключа_мкс', data=df, label='BD / DH', scat
sns.regplot(x='p', y='время_генерации_общего_ключа_кольцевым_методом_мкс', data=
#plt.plot(df['p'], df['время_генерации_ключа_мкс'], 'o-', label='BD / DH (ms)')
#plt.plot(df2['p'], df2['время_генерации_ключа_мкс'], 'o-', label='ECDH (ms)')
plt.xlabel('Размер простого числа p')
plt.ylabel('Время (мс)')
plt.title('Сравнение времени работы протоколов')
plt.legend()
plt.grid(True)
plt.show()
```



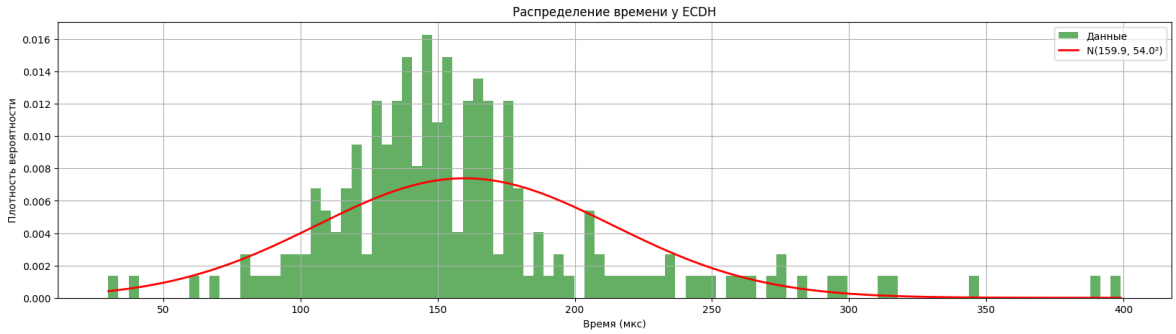
In [56]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm

data = df['время_генерации_ключа_мкс']

# параметры нормального распределения
mu, sigma = np.mean(data), np.std(data)

plt.figure(figsize=(20, 5))
# гистограмма (с нормировкой, чтобы совпала по высоте с кривой)
plt.hist(data, bins=100, color='blue', alpha=0.6, density=True, label='Данные')

# теоретическая кривая нормального распределения
x = np.linspace(min(data), max(data), 1000)
plt.plot(x, norm.pdf(x, mu, sigma), 'r-', lw=2, label=f'N({mu:.1f}, {sigma:.1f}²

plt.xlabel('Время (мкс)')
plt.ylabel('Плотность вероятности')
plt.title('Распределение времени у BD / DH')
plt.legend()
plt.grid(True)
plt.show()
```



In [55]:
```
data2 = df2['время_генерации_общего_ключа_последовательным_умножением_мкс']
mu2, sigma2 = np.mean(data2), np.std(data2)

plt.figure(figsize=(20, 5))
plt.hist(data2, bins=100, color='green', alpha=0.6, density=True, label='Данные'
x2 = np.linspace(min(data2), max(data2), 1000)
plt.plot(x2, norm.pdf(x2, mu2, sigma2), 'r-', lw=2, label=f'N({mu2:.1f}, {sigma2
plt.xlabel('Время (мкс)')
plt.ylabel('Плотность вероятности')
plt.title('Распределение времени у ECDH')
plt.legend()
plt.grid(True)
plt.show()
```

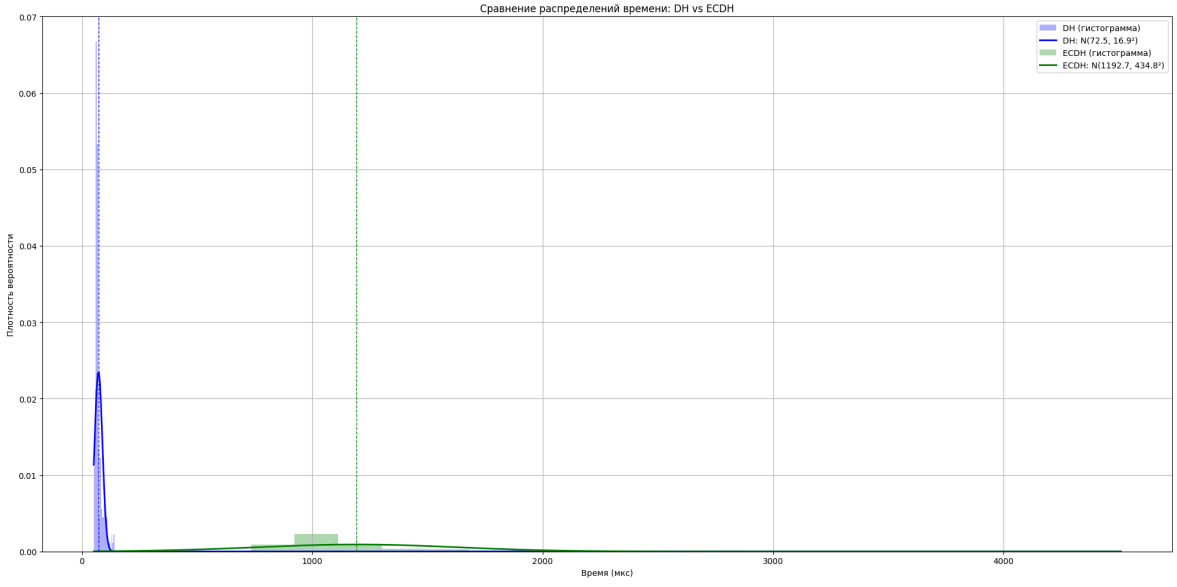


In [61]:
```
data2 = df2['время_генерации_общего_ключа_кольцевым_методом_мкс']
mu2, sigma2 = np.mean(data2), np.std(data2)

plt.figure(figsize=(20, 5))
plt.hist(data2, bins=100, color='green', alpha=0.6, density=True, label='Данные'
x2 = np.linspace(min(data2), max(data2), 1000)
plt.plot(x2, norm.pdf(x2, mu2, sigma2), 'r-', lw=2, label=f'N({mu2:.1f}, {sigma2
plt.xlabel('Время (мкс)')
plt.ylabel('Плотность вероятности')
plt.title('Распределение времени у ECDH')
plt.legend()
plt.grid(True)
plt.show()
```



In [62]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm

# Данные
data_dh = df['время_генерации_ключа_мкс']
data_ecdh = df2['время_генерации_общего_ключа_последовательным_умножением_мкс']
```

```
# Параметры нормального распределения
mu_dh, sigma_dh = np.mean(data_dh), np.std(data_dh)
mu_ecdh, sigma_ecdh = np.mean(data_ecdh), np.std(data_ecdh)

# Общий диапазон по оси X
xmin = min(data_dh.min(), data_ecdh.min())
xmax = max(data_dh.max(), data_ecdh.max())
x = np.linspace(xmin, xmax, 1000)

# Построение
plt.figure(figsize=(20, 10))

# DH
plt.hist(data_dh, bins=20, color='blue', alpha=0.3, density=True, label='DH (гис
plt.plot(x, norm.pdf(x, mu_dh, sigma_dh), 'b-', lw=2, label=f'DH: N({mu_dh:.1f},

# ECDH
plt.hist(data_ecdh, bins=20, color='green', alpha=0.3, density=True, label='ECDH
plt.plot(x, norm.pdf(x, mu_ecdh, sigma_ecdh), 'g-', lw=2, label=f'ECDH: N({mu_ec

# Средние значения
plt.axvline(mu_dh, color='blue', linestyle='--', lw=1)
plt.axvline(mu_ecdh, color='green', linestyle='--', lw=1)

plt.xlabel('Время (мкс)')
plt.ylabel('Плотность вероятности')
plt.title('Сравнение распределений времени: DH vs ECDH')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```



In [63]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm

# Данные
data_dh = df['время_генерации_ключа_мкс']
data_ecdh = df2['время_генерации_общего_ключа_кольцевым_методом_мкс']

# Параметры нормального распределения
mu_dh, sigma_dh = np.mean(data_dh), np.std(data_dh)
```

```
mu_ecdh, sigma_ecdh = np.mean(data_ecdh), np.std(data_ecdh)

# Общий диапазон по оси X
xmin = min(data_dh.min(), data_ecdh.min())
xmax = max(data_dh.max(), data_ecdh.max())
x = np.linspace(xmin, xmax, 1000)

# Построение
plt.figure(figsize=(20, 10))

# DH
plt.hist(data_dh, bins=20, color='blue', alpha=0.3, density=True, label='DH (гис
plt.plot(x, norm.pdf(x, mu_dh, sigma_dh), 'b-', lw=2, label=f'DH: N({mu_dh:.1f},

# ECDH
plt.hist(data_ecdh, bins=20, color='green', alpha=0.3, density=True, label='ECDH
plt.plot(x, norm.pdf(x, mu_ecdh, sigma_ecdh), 'g-', lw=2, label=f'ECDH: N({mu_ec

# Средние значения
plt.axvline(mu_dh, color='blue', linestyle='--', lw=1)
plt.axvline(mu_ecdh, color='green', linestyle='--', lw=1)

plt.xlabel('Время (мкс)')
plt.ylabel('Плотность вероятности')
plt.title('Сравнение распределений времени: DH vs ECDH')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```