Модифицировать модуль шифрования данных клиента

Доработать задачу с задания #1

1. Реализовать в компоненте **Guardian** механизм подписи отправляемых данных. Для подписи данных необходимо использовать класс **Sign** (встроенный класс модуля crypto).
2. Реализовать в компоненте **AccountManager** проверку подписи. Для проверки подписи необходимо использовать класс **Verify** (встроенный класс модуля crypto).

Пример преобразования для первого объекта.

```
// Было
{
  name: 'Pitter Black',
  email: 'pblack@email.com',
  password: 'pblack_123'
}

//  Стало
{
    meta: {
        source: 'ui',
        signature:
"7c8ca3c622df2b317bdeefdb1073a047c3f0adb46f23403bc7edb6d50c8dd34691bbfca4a
e46daf70375d30a2b0d49857f295dc00d0b92185ef845e8780400b0544a2b2fbb2f7a541d0
dad041acf621199506541fd42c45c45ddfd5101a27f71ee58467c7cc95db395bcd5693d450
db27a0d4261c8eaacde2525aa9702029b2c"
    },
    payload: {
        name: 'Pitter Black',
        email: '70626c61636b40656d61696c2e636f6d',
        password: '70626c61636b5f313233'
    }
}
```

Обратите внимание!

Для генерации сертификатов, последовательно выполните следующие команды:

```
openssl genrsa -out server-key.pem 1024
```

```
openssl req -new -key server-key.pem -out server-csr.pem
```

```
openssl x509 -req -in server-csr.pem -signkey server-key.pem -out server-cert.pem
```

1. Для подписи данных скопируйте содержимое файла **server-key.pem** в переменную и используйте эту переменную для создания подписи.
2. Для проверки скопируйте содержимое файла **server-cert.pem** в переменную и используйте эту переменную для проверки подписи.