

Отправитель шифрует сообщение открытым ключом получателя, затем ставит свою подпись с помощью своего закрытого ключа. Получатель с помощью

своего закрытого ключа расшифровывает сообщение и проверяет хэш-сумму, которая должна сопоставляться с открытым ключом отправителя.

Хэш получается путём скармливания закрытого ключа отправителя хэш-функции. Естественно, круче когда коллизий мало.

Отвечу по порядку:

1. \*\*Хэширование сообщения и подпись\*\* — это \*\*разные операции\*\*. Хэш-функция просто берёт данные и возвращает фиксированную длину уникального значения (хэш), но её задача — создать однонаправленное представление данных без шифрования.

2. \*\*Процесс цифровой подписи\*\*: При создании цифровой подписи данные хэшируются, а затем хэш \*\*шифруется закрытым ключом\*\* отправителя. При проверке получатель берёт открытый ключ отправителя и расшифровывает подпись, чтобы получить оригинальный хэш, и сравнивает его с хэшом расшифрованного сообщения.

3. \*\*Шифрование данных и подпись\*\*: Это \*\*две разные криптографические задачи\*\*. Для конфиденциальности само сообщение шифруется, чтобы его можно было расшифровать только с помощью закрытого ключа получателя. Для проверки целостности и подлинности используется цифровая подпись, которая работает, по сути, с хэш-суммой сообщения и служит для проверки того, что данные не были изменены.

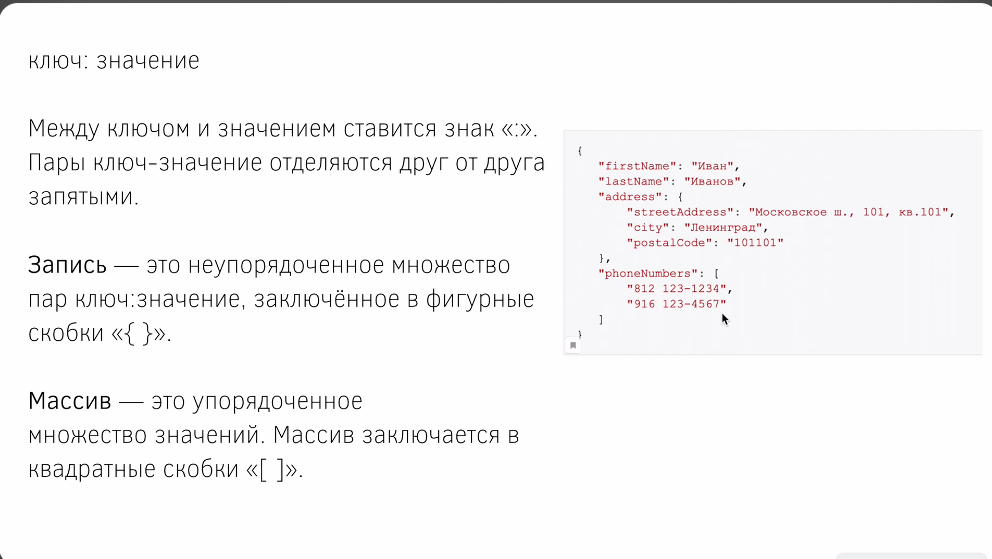
### Подробности о ключах и хэш-функциях

Хэш-функция \*\*не зависит от типа ключа\*\* (открытого или закрытого), потому что она не использует ключи для своей работы. Задача хэширования — просто преобразовать сообщение в короткую фиксированную последовательность, не требуя никаких ключей.

Когда мы говорим о подписи, закрытый ключ отправителя используется для \*\*зашифровки хэша сообщения\*\* и создания подписи. Открытый ключ отправителя, наоборот, используется для проверки — чтобы убедиться, что именно отправитель подписал сообщение.

\*\*Итак, если коротко\*\*: хэширование само по себе не использует ключи, а подпись требует закрытого ключа для создания и открытого ключа для проверки.



JSON: ключ - значение

API – архитектура взаимодействия разных веб частей.

REST API – конкретная API, достаточно лёгкая и понятная в использовании. Использует JSON и XML, чаще JSON, так как это легче.

Для работы с JSON возьмём serde.

Для шифрования ed25519\_dalek

Для хэширования либо std::collections::HashMap или SHA-256 from RustCrypto.