Урок 7. Хэши, пароли и цифровая подпись

Криптография · ~35 минут

Последний урок — про то, как крипто защищает не только тайну, но и **честность**. Как сайт хранит твой пароль, не зная его? Как отличить настоящий документ от подделки? Ответ на всё это — хэш-функции и цифровая подпись.

🎯 Что ты узнаешь

- Что такое хэш-функция и почему она «односторонняя».
- Зачем сайты хранят хэши паролей, а не сами пароли.
- Что такое «соль» и зачем она нужна.
- Как работает цифровая подпись.

📖 Разбираемся в теме

Хэш-функция

Хэш-функция берёт данные любого размера (слово, файл, целую книгу) и превращает их в короткую строку фиксированной длины — **хэш** (или «отпечаток»). Например, знаменитая функция SHA-256 всегда выдаёт ровно 256 бит, что бы ей ни подали.

Хорошая криптографическая хэш-функция обладает свойствами:

- **Односторонность.** По хэшу практически невозможно восстановить исходные данные. Легко посчитать хэш от «пароль123», но по хэшу пароль не вернуть.
- **Лавинный эффект.** Измени во входе хоть один символ — хэш поменяется целиком, до неузнаваемости.
- **Стойкость к коллизиям.** Крайне трудно найти два разных входа с одинаковым хэшем.

📌 **Запомни:** хэш — это «отпечаток пальца» данных. Одинаковые данные → одинаковый хэш всегда. Разные данные → почти наверняка разные хэши.

💡 Хэш — это **не** шифрование! Шифровку можно расшифровать обратно ключом. А хэш расшифровать нельзя в принципе — информация «сжата» безвозвратно.

Зачем хранят хэши паролей

Представь, что сайт хранит пароли как есть, в открытом виде. Если базу украдут — все пароли утекут разом. Катастрофа.

Поэтому серьёзные сайты хранят **не пароль, а его хэш**. Когда ты входишь:

1. Ты вводишь пароль.
2. Сайт считает от него хэш.
3. Сравнивает с сохранённым хэшем. Совпало — пускает.

Сам пароль сайт **не хранит и не знает**. Даже если базу украдут, там будут только хэши, из которых пароль не восстановить (односторонность!).

Соль

Есть проблема: у многих людей пароли одинаковые («123456»), и хэш от них тоже одинаковый. Злоумышленник может заранее посчитать хэши миллионов популярных паролей (это называют «радужной таблицей») и мгновенно узнавать их по хэшу.

Решение — **соль**: к паролю перед хэшированием добавляют случайную строку, свою для каждого пользователя.

- Пароль «123456» + соль «x9Fk2» → один хэш.
- Тот же «123456» + соль «pQ7wz» → совсем другой хэш.

Соль хранится рядом с хэшем (она не секрет). Но теперь одинаковые пароли дают **разные** хэши, и заранее заготовленные таблицы бесполезны — под каждую соль пришлось бы считать таблицу заново.

⚠️ Соль **не заменяет** сложный пароль. Она защищает от массовых таблиц, но короткий пароль всё равно можно подобрать перебором. Длинный пароль + соль — вот надёжно.

Цифровая подпись

Хэш решает ещё одну задачу — **подтвердить авторство и целостность**. Вспомни RSA из урока 6: там открытым ключом шифруют, а закрытым расшифровывают. Подпись работает **наоборот**.

Чтобы подписать документ, автор:

1. Считает **хэш** документа (короткий отпечаток).
2. «Шифрует» этот хэш своим **закрытым** ключом — получается **подпись**.
3. Прикладывает подпись к документу.

Любой проверяющий:

1. Считает хэш документа сам.
2. «Расшифровывает» подпись **открытым** ключом автора — получает хэш, который заложил автор.
3. Сравнивает два хэша. Совпали — значит, подпись подлинная **и** документ не изменён.

📌 **Запомни:** подписать может только владелец закрытого ключа, а проверить — кто угодно с помощью открытого. Это доказывает авторство. А поскольку подпись привязана к хэшу документа, любое изменение текста ломает проверку — это доказывает целостность.

🤔 **А знаешь ли ты?** Когда ты обновляешь приложение на телефоне, система проверяет цифровую подпись разработчика. Если бы кто-то подменил файл вирусом, хэш изменился бы, подпись не сошлась — и обновление было бы отклонено. Так подпись защищает миллиарды устройств.

Как всё складывается

Посмотри, как три идеи курса работают вместе в защищённом соединении:

- **Диффи–Хеллман / RSA** — договориться о секрете и зашифровать данные (тайна).
- **Хэши** — проверить, что данные не подменили (целостность).
- **Подпись** — убедиться, что собеседник тот, за кого себя выдаёт (подлинность).

Тайна, целостность, подлинность — три кита, на которых стоит безопасность интернета.

✍️ Разбор примера

Задача: Аня прислала Боре договор и цифровую подпись к нему. Как Боря убедится, что договор настоящий и никто не изменил ни буквы?

По шагам:

1. Боря считает **хэш** полученного текста договора.
2. Берёт **открытый ключ Ани** и «расшифровывает» им подпись — получает хэш, который заложила Аня.
3. **Сравнивает** оба хэша.
    - Совпали → договор подлинный и целый: подпись мог создать только тот, у кого закрытый ключ Ани, а хэш подтверждает, что текст не тронут.
    - Не совпали → либо текст изменили, либо подпись подделали. Боря такому договору не доверяет.

📝 Задачи

1. Чем хэш-функция принципиально отличается от шифрования? Назови главное отличие.
2. Перечисли три свойства хорошей криптографической хэш-функции.
3. Почему сайту безопаснее хранить хэш пароля, а не сам пароль?

4. Что такое соль и от какой именно атаки она защищает?
5. У двух пользователей одинаковый пароль «qwerty». Почему в базе с солью их записи будут выглядеть по-разному?
6. Какой ключ используют, чтобы **создать** подпись, а какой — чтобы её **проверить**?
7. Аня подписала договор. Боря поменял в тексте одну цифру суммы и переслал дальше. Пройдёт ли теперь проверка подписи? Почему?
8. *Со звёздочкой.* Объясни, почему в цифровой подписи подписывают **хэш** документа, а не сам документ целиком. (Подумай про длину и удобство.)