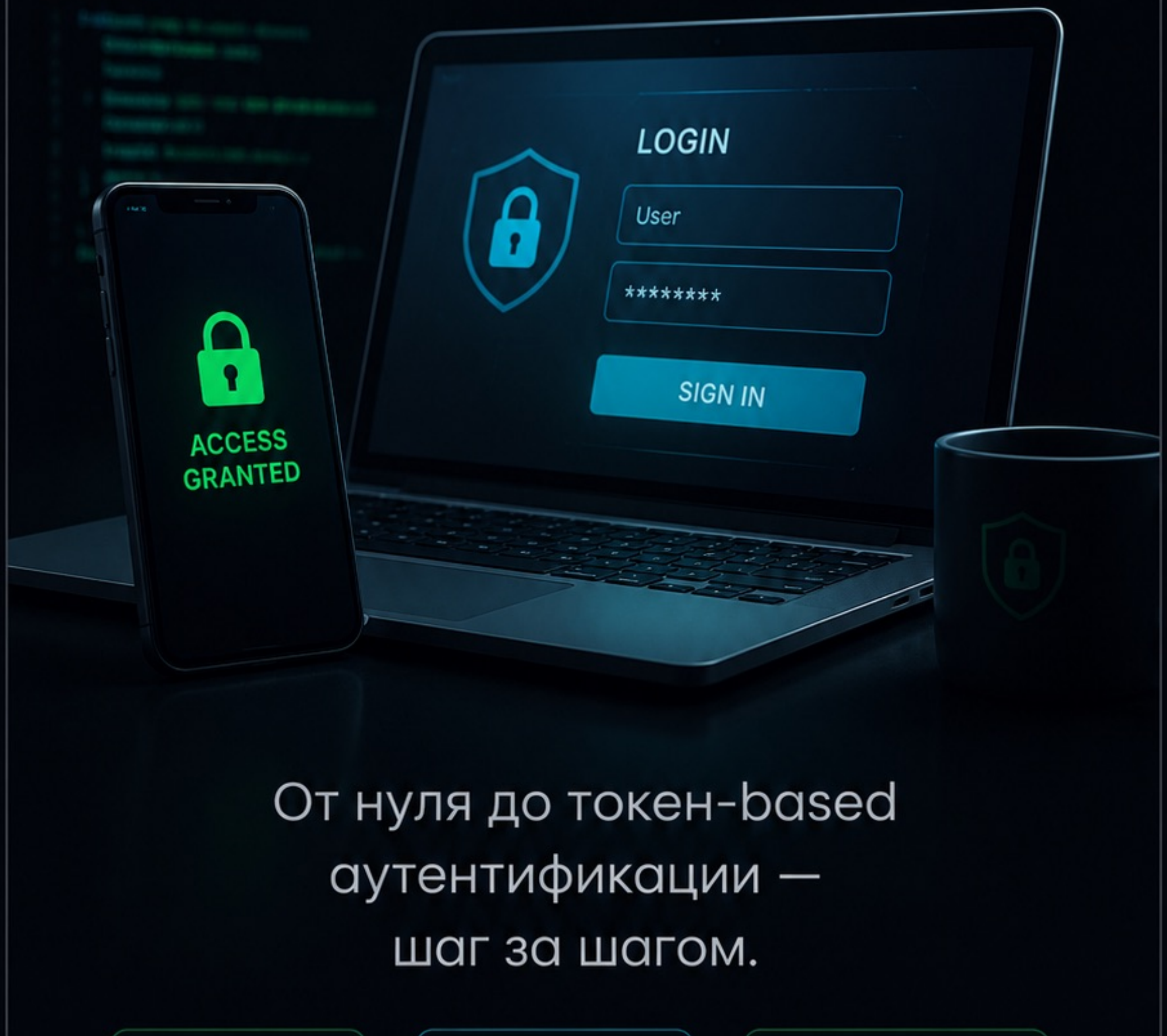
JWT

Объяснено



От нуля до токен-based аутентификации — шаг за шагом.

Header · Payload · Signature

Что такое JWT?



JWT (JSON Web Token) — это компактный, самодостаточный способ передачи проверенной информации между двумя сторонами в виде подписанной строки.



1. Подтверждение целостности

Похоже на запечатанный браслет на мероприятии. Кто угодно может прочитать, что на нём написано, но никто не сможет подделать настоящий без секрета.



2. Самодостаточный (self-contained)

Токен сам хранит данные пользователя. Серверу не нужно каждый раз запрашивать информацию, чтобы понять, кто ты.



3. Компактный (compact)

Это короткая строка текста, которая легко передаётся в HTTP header, URL или cookie.



Как эта проблема решается



До появления токенов серверу нужно было запоминать сессию каждого залогиненного пользователя. Это не масштабируется.



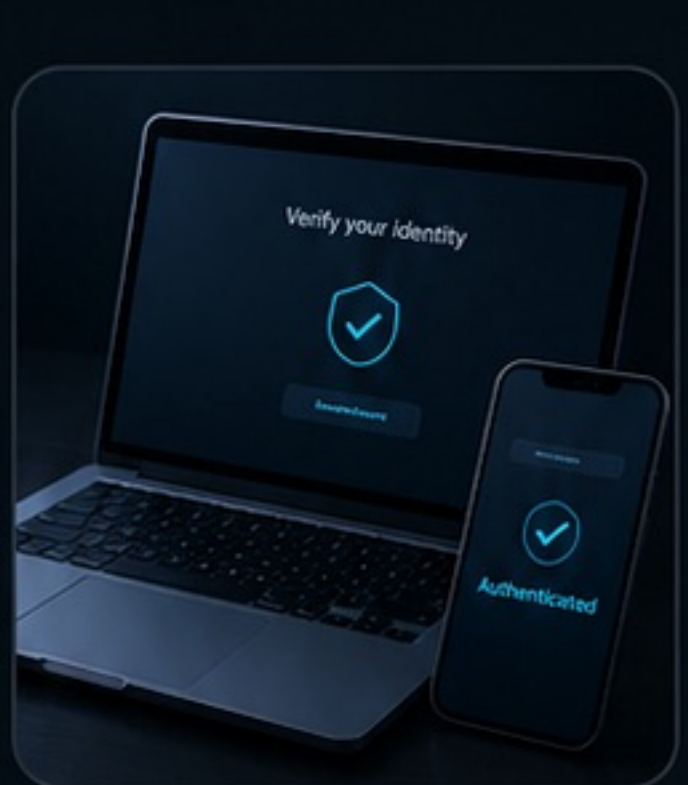
1. Сервер всё запоминает

Каждый login хранится на сервере. Чем больше пользователей — тем больше памяти используется и тем больше запросов на каждый lookup.



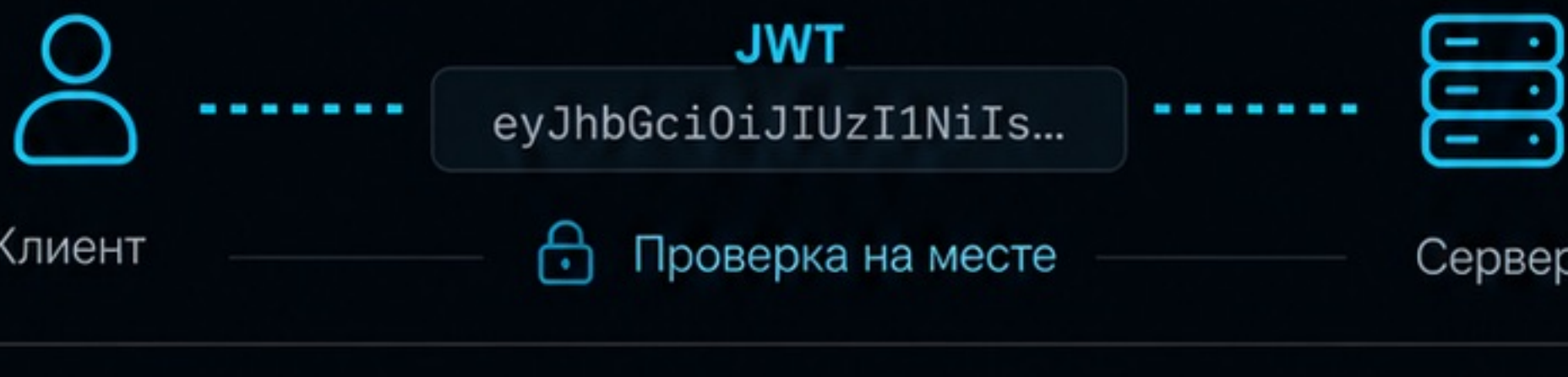
2. Не работает с множеством серверов

Если за load balancer несколько серверов, один сервер не знает о сессиях, созданных на другом. Пользователь может «потеряться».

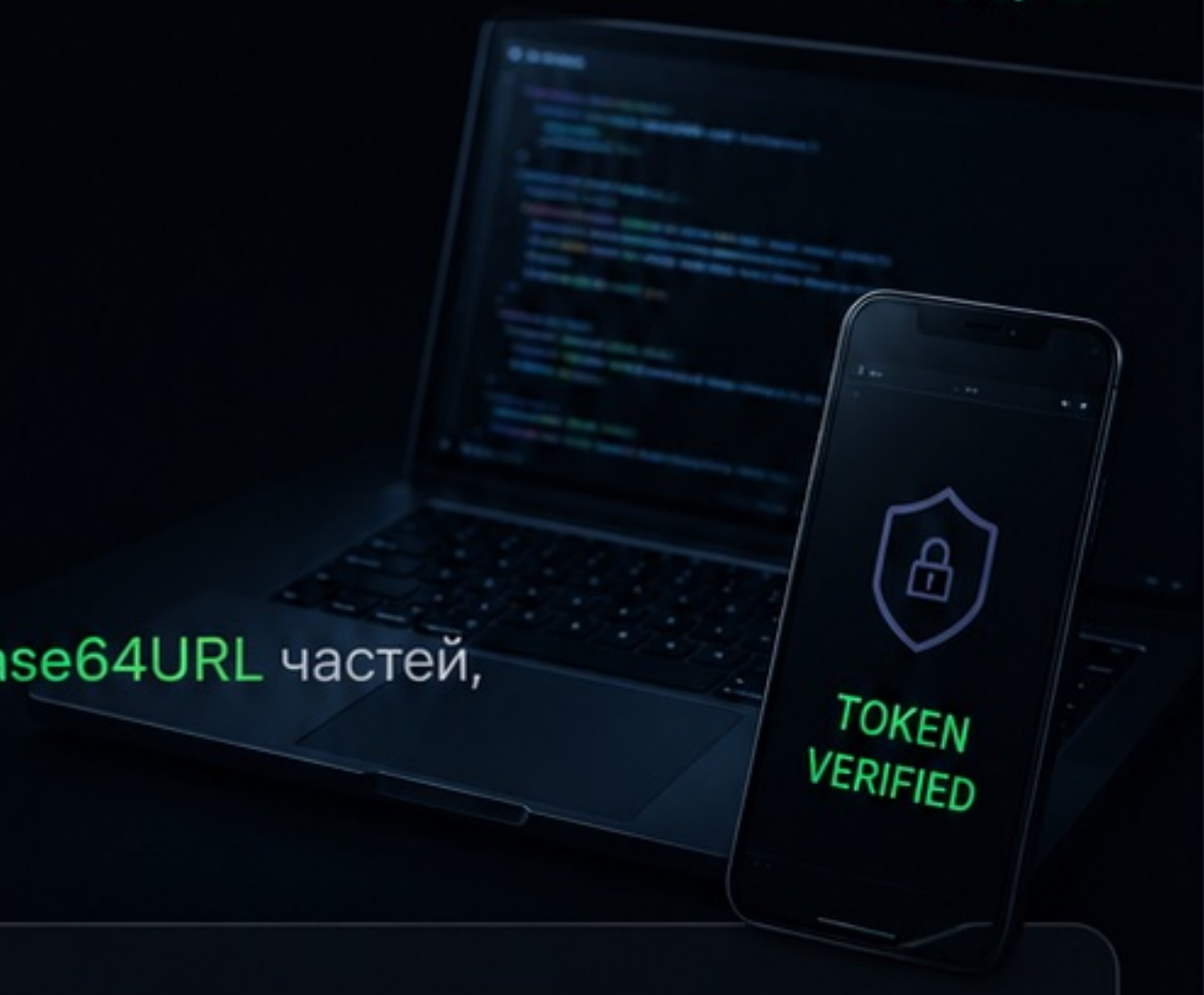


3. JWT меняет модель

Клиент сам хранит доказательство своей личности. Любой сервер может проверить его на месте, без обращения к другим серверам и без общей памяти. Это называется stateless auth.



3 части JWT



Каждый JWT состоит из трёх Base64URL частей, разделённых точками.

пример JWT (encoded token)

`eyJhbGciOiJI..` `eyJzdWIiOiIx.` `SflkXwRJSMe`

Header · Payload · Signature

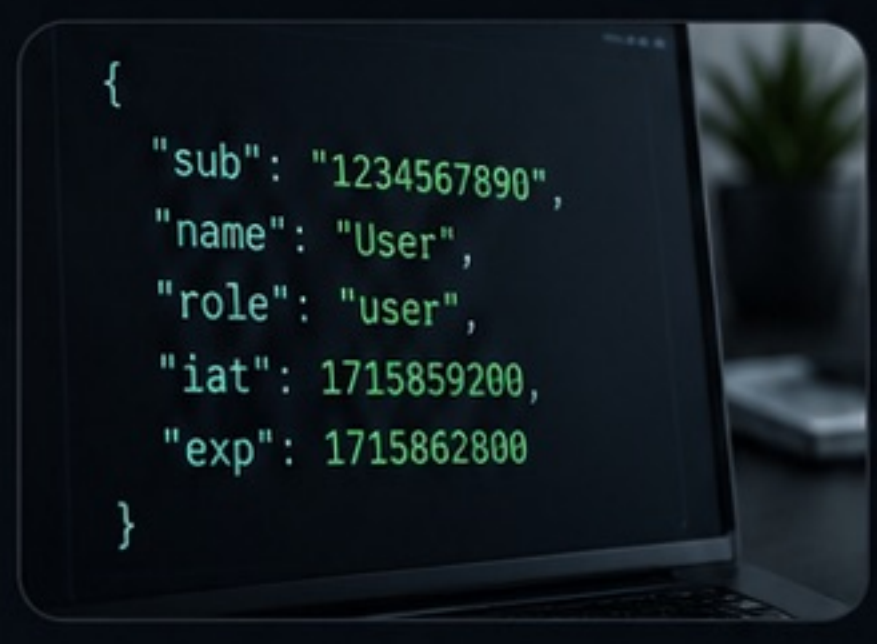
01. Header

Содержит тип токена (JWT) и алгоритм подписи (signing algorithm).

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

02. Payload

Содержит claims — данные о пользователе и дополнительную информацию.



03. Signature

Криптографическая подпись, которая защищает токен от изменений (tamper-proof).



Header

Метаданные, которые описывают сам token.



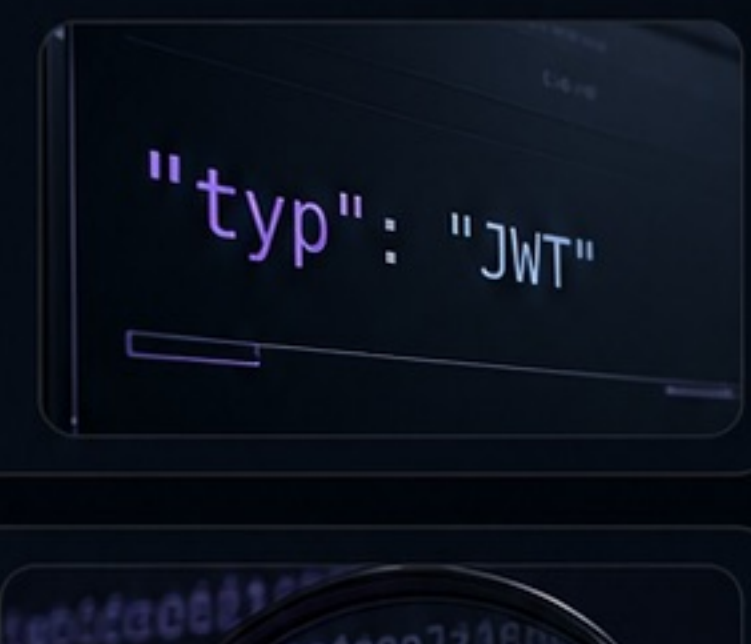
decoded header

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

alg Алгоритм подписи, который используется для защиты token, например HS256 или RS256.



typ Тип token. Для JWT это обычно просто "JWT".



Это не секрет

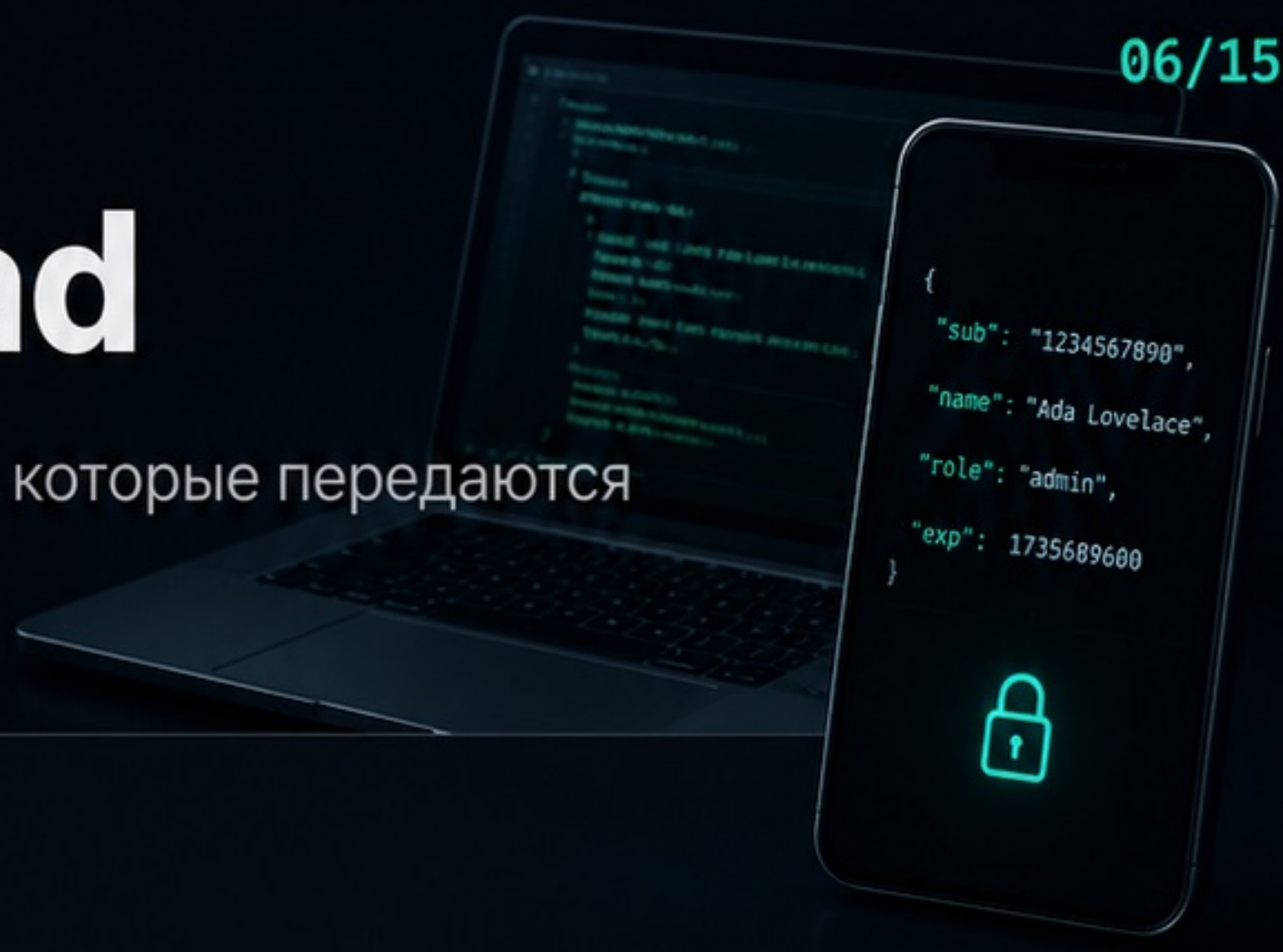
Header закодирован только в Base64URL, поэтому его может прочитать любой.

Он не содержит конфиденциальных данных — только инструкции для проверки token.



Payload

Реальные данные, которые передаются в виде **claims**.



decoded payload

```
{
  "sub": "1234567890",
  "name": "Ada Lovelace",
  "role": "admin",
  "exp": 1735689600
}
```

Claims — это утверждения

Каждый claim — это пара ключ-значение о пользователе или токене.

Здесь хранится реальная информация о личности.

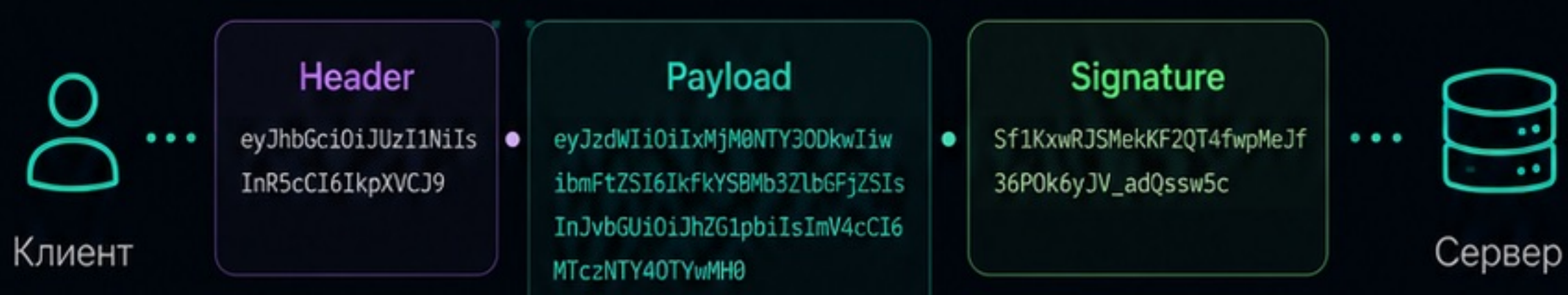


Доступно для всех

Anyone может прочитать payload.

Никогда не храните пароли или секреты в payload.





Типы Claims

Claims делятся на три группы.



1. Registered

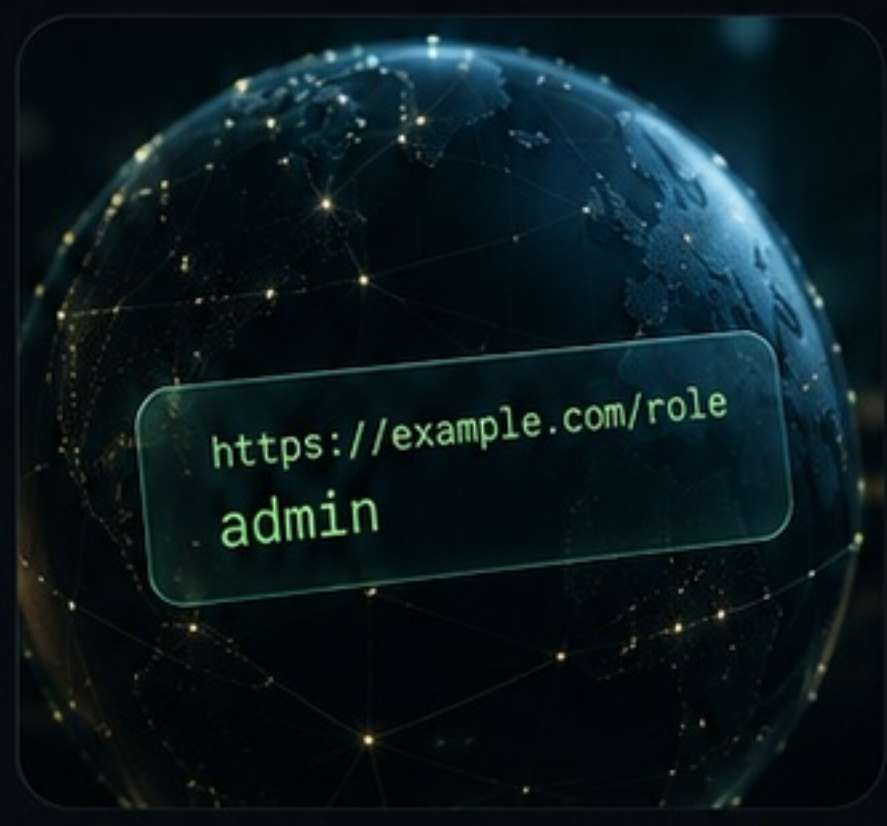
Стандартные ключи, такие как `iss`, `sub`, `exp`.

Короткие имена, которые понимают все библиотеки.



2. Public

Пользовательские ключи, которые ты задаёшь сам. Рекомендуется использовать `namespace`, например с префиксом `URL`, чтобы избежать конфликтов.

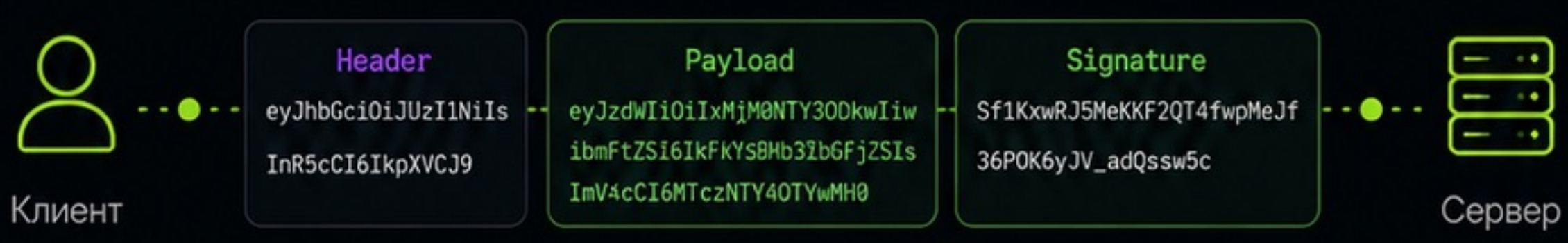


3. Private

Приватные ключи. Соглашение только между сторонами, которые используют `token`.

Не стандартные, основаны на договорённости.





Signature



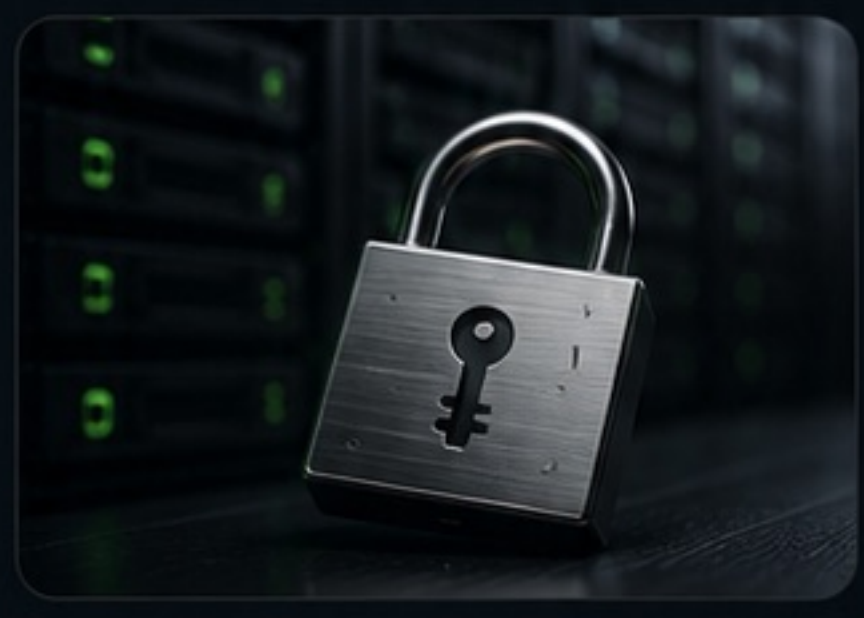
Защита от изменений. Подтверждает, что header и payload не были изменены.

как это устроено

```
signature =
   HMACSHA256(
      base64url(header) + "." + base64url(payload),
      secret
)
```

Создано из секрета

Сервер хэширует закодированные header и payload вместе с секретным ключом. Воспроизвести это может только тот, у кого есть этот secret.



Один символ ломает всё

Изменение даже одного байта в token или в signature делает их несовместимыми. Сервер отклоняет такую попытку сразу.

Никаких скрытых изменений

Signature гарантирует подлинность данных. Если token был изменён — это будет обнаружено сразу.





Закодирован, но **НЕ** зашифрован

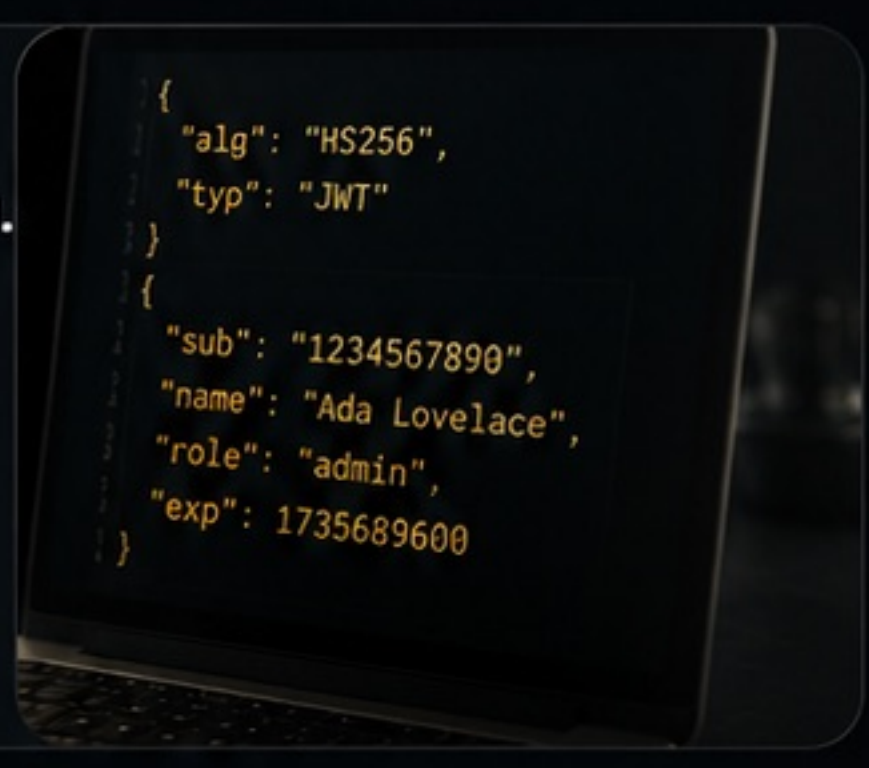




Base64URL — это **не шифрование**.
Любой, у кого есть **token**,
может декодировать и прочитать **payload**.

Header и **payload** полностью читаемы.

Вставьте token в любой decoder —
и вы увидите всё содержимое.



Signature защищает целостность,
а не приватность.

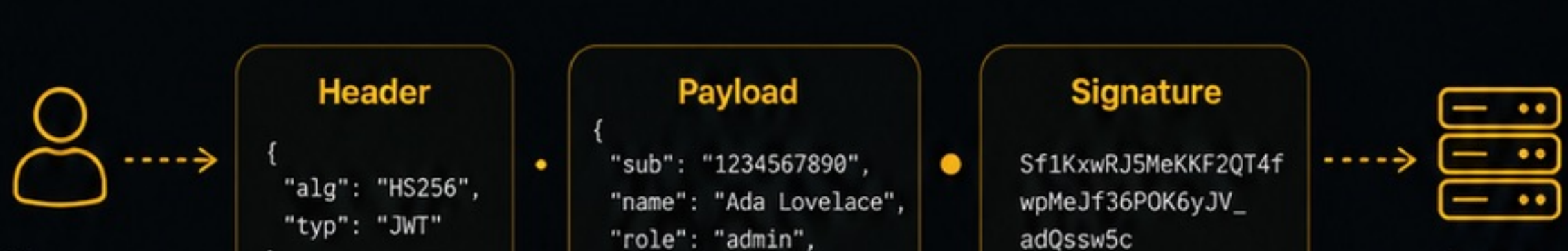
Она доказывает, что данные
не были изменены, но не скрывает
содержимое.



Правило: не храните чувствительные данные в payload.

Считайте каждый claim публичным.



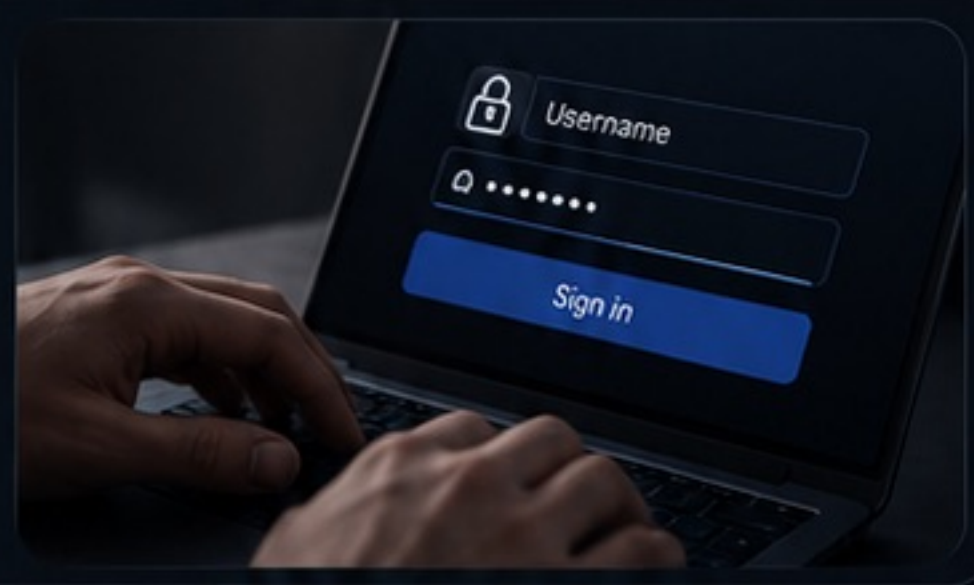


Как работает Authentication Flow



1 Login

Пользователь отправляет свои credentials на сервер.



2 Issue

Сервер проверяет данные, подписывает новый JWT и возвращает его.



3 Store

Клиент сохраняет token в памяти, cookie или хранилище.



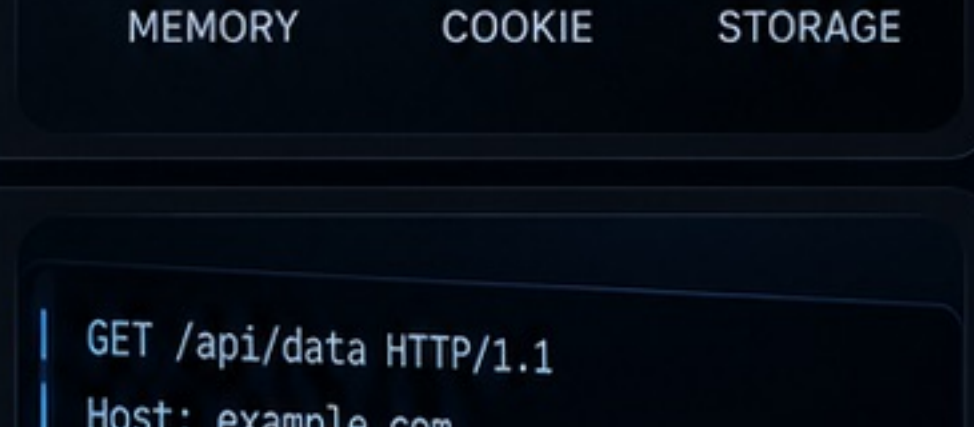
4 Send

При каждом запросе клиент отправляет: Authorization: Bearer <token>.

```
GET /api/data HTTP/1.1
Host: example.com
Authorization: Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

5 Verify

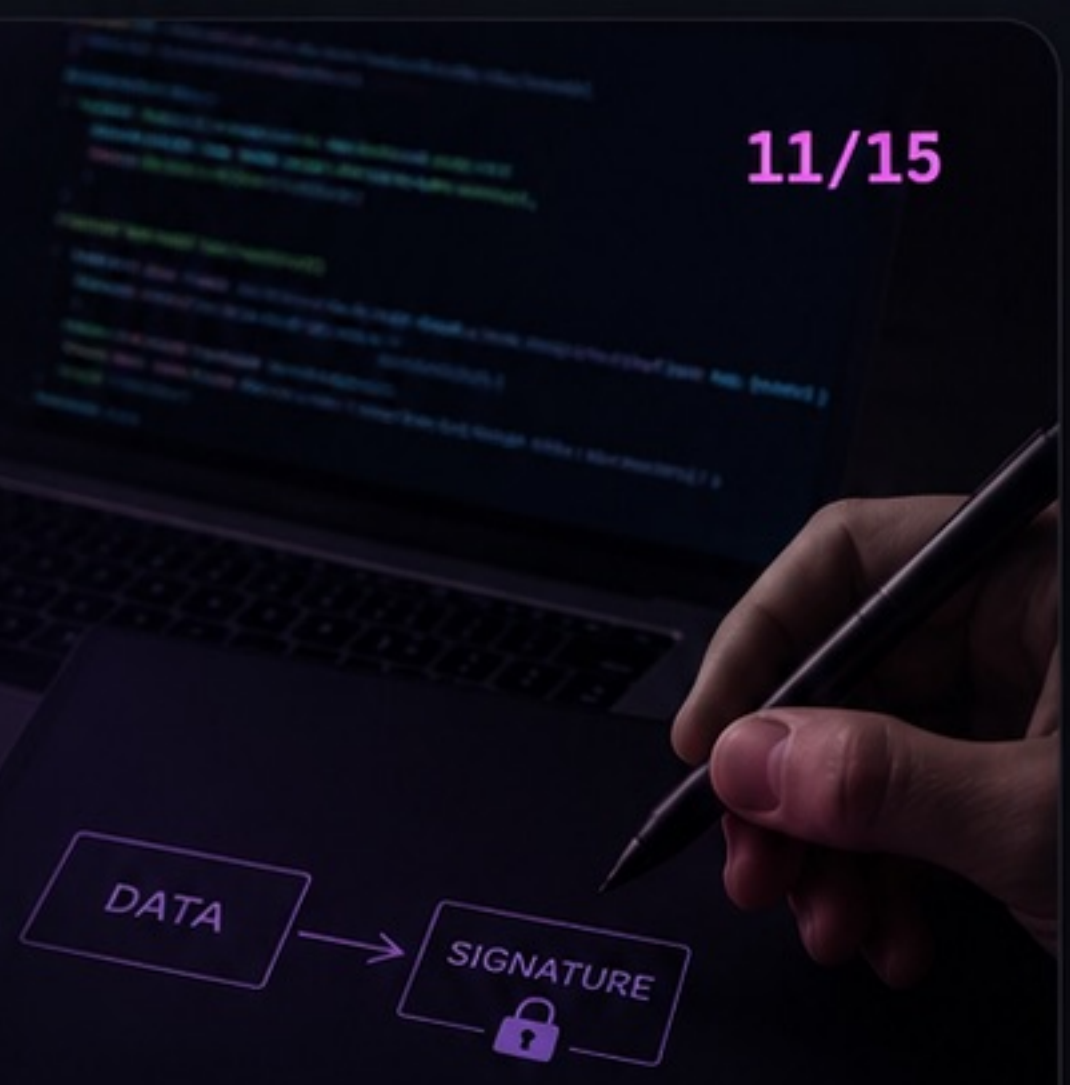
Сервер проверяет подпись и expiry, затем доверяет claims. Без поиска в базе данных.



Клиент → 1 Login → 2 Issue → 3 Store → 4 Send → 5 Verify → Сервер

Алгоритмы подписи

Как создаётся и проверяется signature.



HS256

SYMMETRIC

Один общий secret используется и для подписи, и для проверки.

Просто и быстро.
Лучше всего, когда одна доверенная сторона выдаёт token и проверяет его.



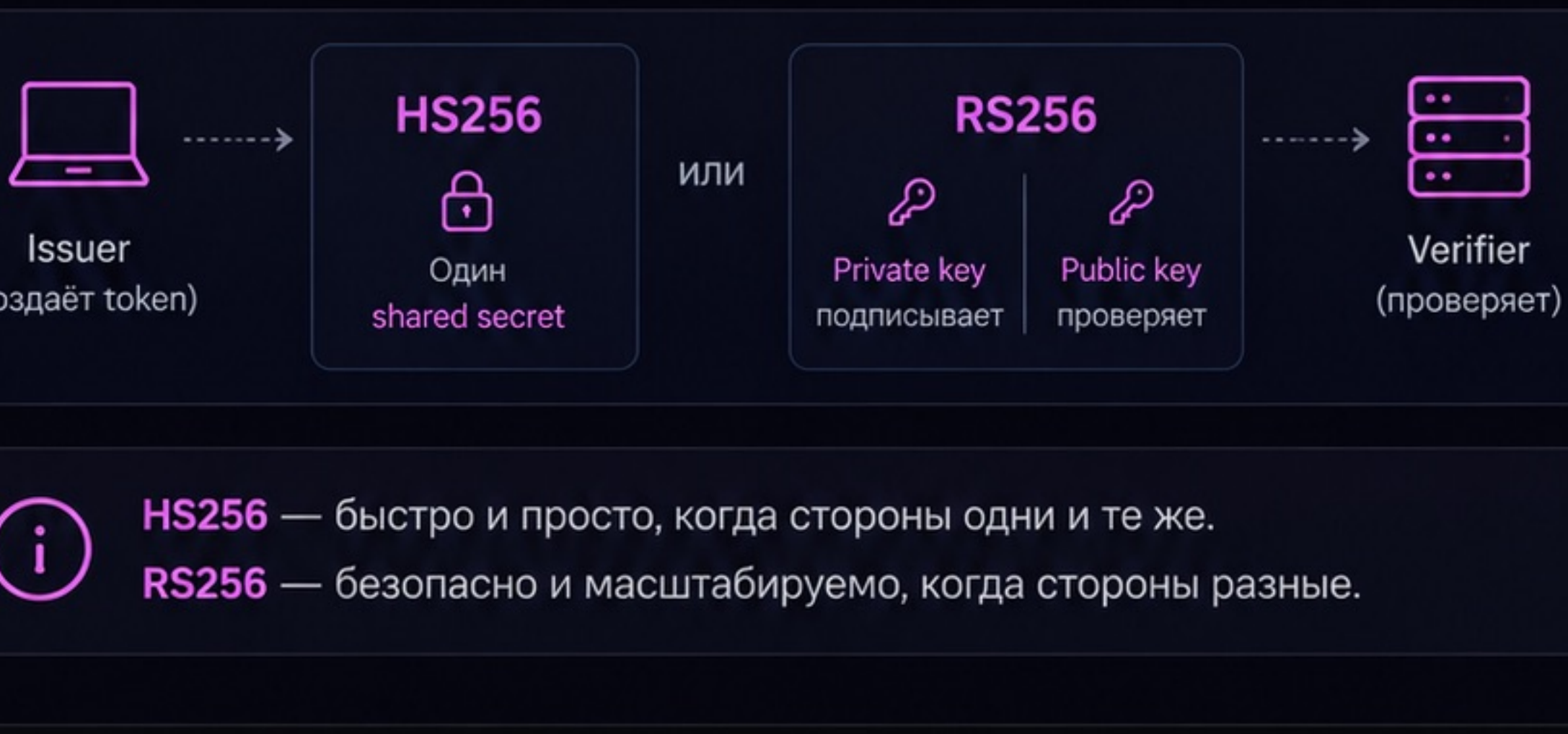
RS256

ASYMMETRIC

Приватный key подписывает, соответствующий public key проверяет.

Проверяющей стороне не нужен secret. Лучше всего для разных сервисов и открытых систем.





HS256 — быстро и просто, когда стороны одни и те же.
RS256 — безопасно и масштабируемо, когда стороны разные.

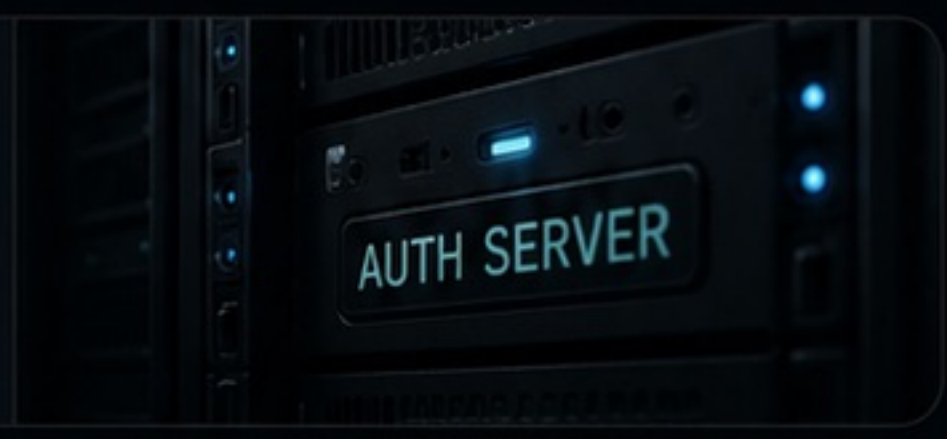
Claims Cheat Sheet

Краткий справочник по claims в JWT.

`iss`

Issuer

Кто создал и подписал token.



`sub`

Subject

Кто является субъектом token, обычно это user id.



`aud`

Audience

Для кого предназначен token, то есть целевая аудитория.



`exp`

Expiry

Время истечения срока действия. После этого token становится недействительным.



`iat`

Issued At

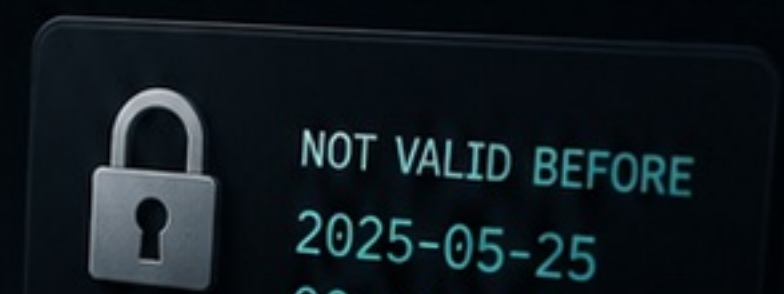
Время создания token.



`nbf`

Not Before

Не раньше этого времени. Не принимается ранее указанного времени.



`jti`

JWT ID

Уникальный идентификатор token, который помогает предотвратить повторное использование (replay).



Эти claims помогают обеспечить безопасность, срок действия и контекст использования JWT.

Лучшие практики по Security



Всегда устанавливайте короткий exp.

Долгоживущие tokens опасны, если утекут.



Отклоняйте alg = none и не доверяйте claims вслепую.

Всегда проверяйте claims токена.



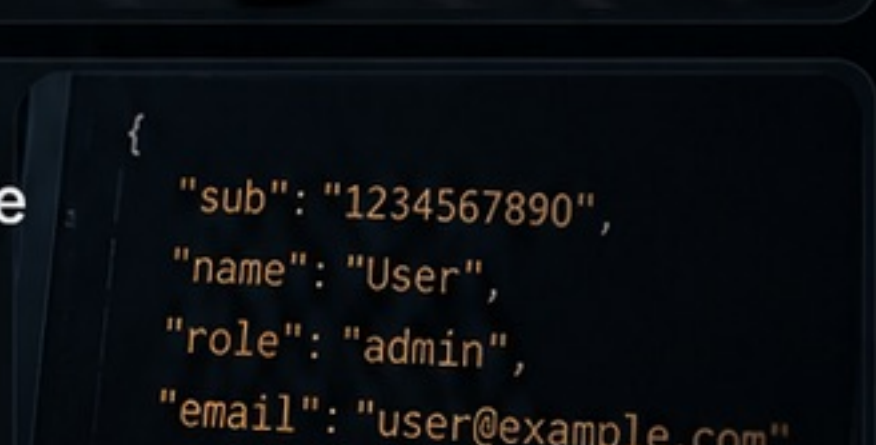
Используйте длинный random secret или сильные key.

Слабые secrets можно подобрать перебором.



Держите чувствительные данные вне payload.

Каждый claim должен быть читаемым.



Передавайте tokens только по HTTPS и храните их аккуратно на клиенте.

Используйте secure storage и защищённые каналы.



Хорошая безопасность — это баланс между удобством и защитой. Следуйте лучшим практикам.

JWT vs Sessions

Нет универсального решения.
Всё зависит от задачи.



	JWT	Sessions
Где хранится состояние	Client-side	Server-side
Масштабируемость между серверами	Easily	Нужен shared store
Отзыв доступа	Harder	Instant
Читается ли payload	Yes	No, только id
Лучше всего подходит для	APIs, microservices	Классические web-приложения

JWT — отлично для **stateless** систем и распределённых сервисов.

Sessions — проще управлять и отзывать, лучше для традиционных **web-приложений**.