JWT Secure Authentication in MERN & Microservices Beginner Friendly Summary

What is JWT?

- JWT (JSON Web Token) = A small token carrying user info securely.
- Used for authentication in MERN & Microservices.
- Structure: Header.Payload.Signature

Header: Type of token & algorithm (e.g., RS256)

Payload: User data (e.g., ID, email, role)

Signature: Digital stamp to prevent tampering

JWT Flow in Microservices

1. User logs in at Auth Service.
2. JWT is created and signed using private key.
3. JWT is sent to frontend (via HttpOnly cookie).
4. Client sends JWT with each request to any microservice.
5. Microservices verify JWT using public key.
6. If valid, access is granted.

Why It's Called Secure Authentication

- Uses digital signatures cant be faked or altered.
- No sessions stateless & scalable.
- Tokens expire limits risk if stolen.
- Works great with HttpOnly Cookies (protected from JS).

Types of JWT

Access Token:

- Short-lived (e.g., 15 mins)

- Used to access protected routes/resources

Refresh Token:

- Long-lived (e.g., 7 days)

- Used to get a new access token when old one expires

- Stored securely (e.g., HttpOnly cookie or DB)

Local Storage vs HttpOnly Cookies

Local Storage:

- Accessible by JS

- Not sent with every request automatically

- Vulnerable to XSS

HttpOnly Cookie:

- Not accessible by JS

- Automatically sent with every request

- Secure from XSS

Public & Private Keys in RS256

Private Key:

- Signs the JWT (secret)

- Used by Auth service only

Public Key:

- Verifies JWT authenticity

- Shared with all services

Generate Keys Using OpenSSL

```
openssl genrsa -out private.key 2048
openssl rsa -in private.key -pubout -out public.key
```

Sign JWT with RS256 (Node.js)

```
const jwt = require('jsonwebtoken');
const fs = require('fs');
const privateKey = fs.readFileSync('./private.key');
const payload = { userId: '123', email: 'user@example.com' };
const token = jwt.sign(payload, privateKey, { algorithm: 'RS256', expiresIn: '15m' });
```

Verify JWT (using Public Key)

```
const publicKey = fs.readFileSync('./public.key');
jwt.verify(token, publicKey, { algorithms: ['RS256'] }, (err, decoded) => {
  if (err) return console.log("Invalid token");
  console.log("Valid token payload:", decoded);
});
```

Store JWT in HttpOnly Cookie (Express)

```
res.cookie('access_token', token, {
  httpOnly: true,
  secure: true,
  sameSite: 'Strict',
  maxAge: 15 * 60 * 1000
});
```

Real-World Analogy

JWT: Sealed envelope with user info inside

Header: Stamp info on the envelope

Payload: Info inside (user ID, email)

Signature: Wax seal proves its original

Private Key: Your secret pen for sealing

Public Key: Others use it to verify your seal

HttpOnly Cookie: Locked drawer only browser/server can open

Local Storage: Open shelf anyone (even JS) can grab from