Encryption and Encoding

In **Express.js** authorization, both **encryption** and **encoding** play roles in handling sensitive data like tokens and credentials, but they serve different purposes.

1. Encoding in Authorization

- **Purpose:** Converts data into a different format to ensure safe transmission without security guarantees.
- **Usage in Express.js:**
  - **Base64 Encoding:** Often used in Basic Authentication (`Authorization: Basic base64(username:password)`).
  - **URL Encoding:** Used when sending credentials or tokens in URLs (`encodeURIComponent()`).
- **Example in Express.js:**

```
js
CopyEdit
const encoded = Buffer.from("username:password").toString("base64");
console.log(encoded); // dXNlcm5hbWU6cGFzc3dvcmQ=
```

2. Encryption in Authorization

- **Purpose:** Converts data into an unreadable format using a key, ensuring confidentiality.
- **Usage in Express.js:**
  - **JWT (JSON Web Tokens):** Uses **HMAC (HS256) or RSA (RS256) encryption** for signing tokens.
  - **OAuth 2.0:** Uses encryption for securely storing and transmitting tokens.
  - **Password Hashing:** Uses `bcrypt` to encrypt passwords before storing them.

- **Example in Express.js (JWT):**

  ```
  js
  CopyEdit
  const jwt = require("jsonwebtoken");
  const secretKey = "yourSecretKey";
  const token = jwt.sign({ userId: 123 }, secretKey, { expiresIn: "1h" });
  console.log(token);
  ```

Key Differences

Feature	Encoding	Encryption
Purpose	Safe data transmission	Secure data storage & transmission
Reversible?	Yes	No (without a key)
Security	Not secure	Highly secure
Example Usage	Base64 encoding passwords	JWT tokens, password hashing

When to Use What?

- Use **encoding** for safe transmission but not for security.
- Use **encryption** for securing sensitive data like JWT tokens and passwords in Express.js.