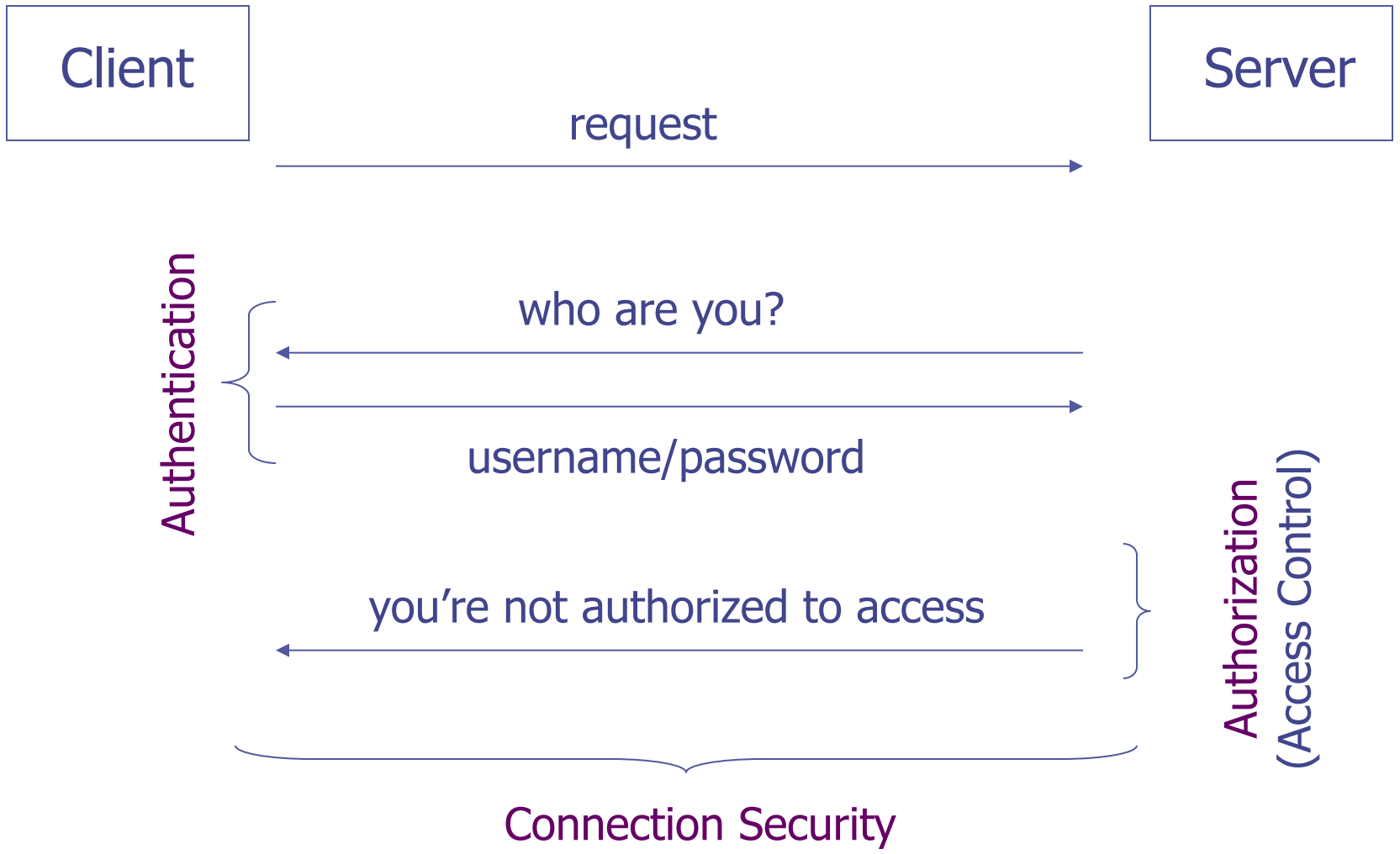
CS5220 Advanced Topics in Web Programming

Secure REST API

Chengyu Sun
California State University, Los Angeles

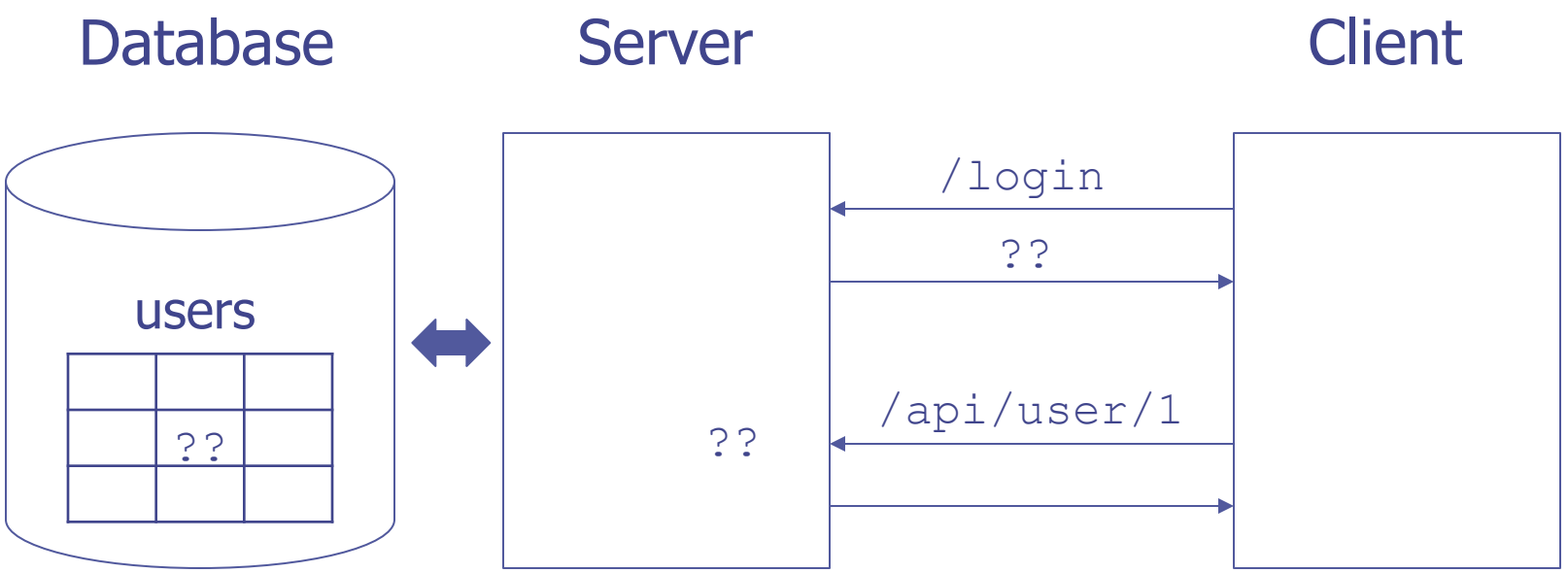
Web Application Security



HTTP Secure (HTTPS)

- HTTP over SSL/TLS
- Getting free SSL/TLS certificates from Let's Encrypt - https://letsencrypt.org/

Basic Security Implementation



- How to store passwords properly??
- What happens if authentication is successful??
- How does the server authenticate/authorize subsequent requests??

Storing Passwords ...

- Hash stored passwords
  - Why?? How?? *Encryption* vs *Hashing*??
- How to check against hashed passwords??
- Common attack on hashed passwords
  - Brute force and some variations
  - Dictionary

… Storing Passwords

- Common defenses
  - Long and random passwords
  - Make cryptographic hash functions slower
  - Salt

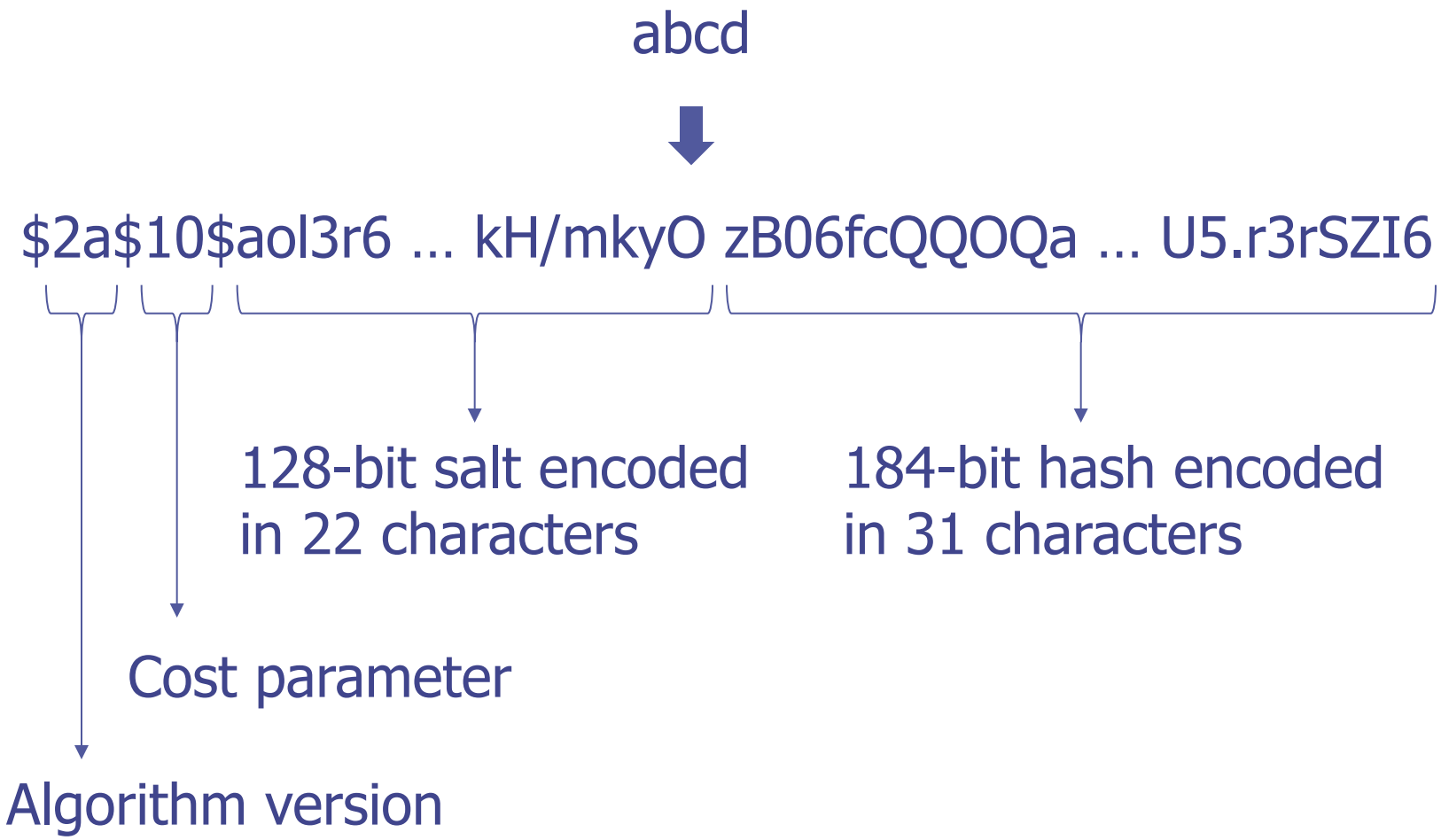
Cryptographic Hash Function…

- String of arbitrary length → `n` bits *digest*
- Properties
  1. Given a hash value, it's virtually impossible to find a message that hashes to this value
  2. Given a message, it's virtually impossible to find another message that hashes to the same value
  3. It's virtually impossible to find two messages that hash to the same value
- A.K.A.
  - *One-way hashing*, *message digest*, *digital fingerprint*

…Cryptographic Hash Function

- Common usage
  - *Store passwords*, software checksum …
- Popular algorithms
  - Good for password hashing: `Argon2`, `PBKDF2`, `scrypt`, `bcrypt`
  - Not good for password hashing: `MD5` (broken), `SHA` and variants

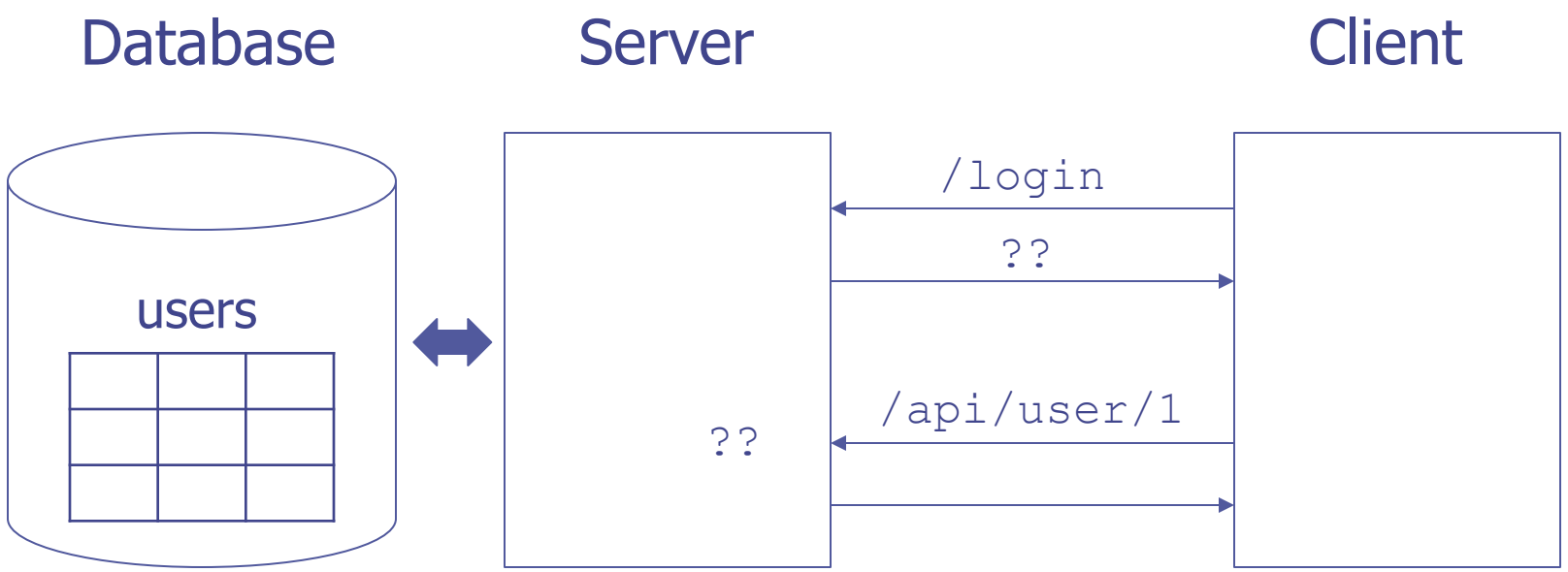
bcrypt hash Example



bcrypt Libraries

- jBcrypt (org.mindrot:jbcrypt) for Java
- bcrypt or bcryptjs for JavaScript

Session-Based Security



- What happens if authentication is successful??
- What does the server authenticate/authorize subsequent requests??
- *Why is it not suitable for RESTful web service??*

Security without Session

- What's wrong with sending username and password in every request?
- How about using an access key / token instead?
- Issues need to be considered: *access control*, *performance*, *expiration*, *revocation*

JSON Web Token (JWT)

- Header, payload, and signature
- Example at https://jwt.io/

JWT Header

- A JSON object
- Specifies the token type and the cryptographic hashing algorithm used for signature
- Base64URL encoded
  - Encoding != Encryption

JWT Payload

- A JSON object
- Usually contains information about user identity and authorization, a.k.a. *claims*
- There are some predefined claims, e.g. `sub` (subject), `exp` (expiration), `iat` (Issued at), but applications are free to do anything they want

JWT Signature

- The hash of header + payload + a secrete key (that only the server knows)

Using JWT – Login

- Receive username and password
- Load user information from database
- Verify username and password
- Success
  - Create a JWT
  - Send it back to client
- Failure??

Using JWT – Subsequent Request

- Client includes JWT in each request
  - Header, e.g. `Authorization: Bearer <token>`
  - Cookie
  - request parameter
- Server verifies the signature in JWT
- If signature is valid, server grants access based on JWT (*no need to access database again!*)
- Failure??

Implement Login Endpoint

- Use jsonwebtoken to create JWT

```
jwt = require('jsonwebtoken');
jwt.sign( <payload>, <secret> [, options] );
```

Usually a User object

Common claims such as `sub` and `exp` are in options

JWT Authorization Using Middleware

- Extract JWT token from Authorization header
- Verify the signature
- Attach the payload (i.e. the user object) to `req` and pass it onto the next middleware
- Return 401 if anything is wrong

Passport

- Passport is a popular Node.js authentication framework
- Different authentication schemes (called *strategies*) can be implemented on top of Passport

Configure Passport with JWT Strategy

```
const passport = require("passport");
const passportJWT = require("passport-jwt");

passport.use(
    new passportJWT.Strategy({
        secretOrKey: "abcd",
        jwtFromRequest: passportJWT.ExtractJwt.
        fromAuthHeaderAsBearerToken()
    },
    function(payload, done) {return done(null, payload);}
  )
);
```

About Passport JWT Strategy ...

```
new passportJWT.Strategy(options, callback)
```

- See more options in documentation
- See example for using multiple extractors
- If JWT signature if invalid, an error response will be sent back, or an error can be generated if Passport's `failWithError` option is true (which can then be handled by an Express error handling middleware)

… About Passport JWT Strategy

- If JWT signature is valid, `callback(payload, done)` is called that allows *additional* processing
- `payload`: decoded JWT payload
- `done(err, user)`: a function that sets either an error object or `req.user` to be used by subsequent Express middleware

Add Passport as Express Middleware

```
app.use( passport.initialize() );

app.use( '/api/', passport.authenticate('jwt', {
    session: false
}));
```