Awesome question! You're digging into the heart of how **secure communication** works with public/private keys — exactly what happens *both ways* between browser and server. Let me explain step-by-step how the **server’s response is encrypted and decrypted** so the browser can read it securely.

**How Does the Browser Decrypt Server Responses?**

**The quick answer:**

The **server encrypts its response in a way that the browser can decrypt**, and this happens using a shared **session key** established during the connection setup — *not* by the server encrypting with the browser’s public key directly.

**Let’s break it down carefully:**

**1. Public/Private Keys: Used to Establish a Secure Channel**

When you open an HTTPS connection:

* The browser **uses the server’s public key** (from the SSL certificate) to encrypt a **random secret** called a **session key** (a symmetric key).
* The server uses its **private key** to decrypt this session key.

**2. Session Key (Symmetric Key) Is Now Shared**

* Both browser and server now have the **same session key**.
* This key is symmetric, meaning:
  + The **same key** is used to encrypt *and* decrypt data.
* Symmetric encryption is **much faster** than asymmetric (public/private key) encryption.

**3. Encrypted Communication Using Session Key**

* Browser encrypts requests with the session key and sends to server.
* Server decrypts requests with the session key.
* Server encrypts responses with the same session key and sends to browser.
* Browser decrypts responses with the session key.

**4. Why Not Use Public/Private Keys to Encrypt Everything?**

* Public/private (asymmetric) encryption is **slow and CPU-intensive**.
* It's only used to **safely exchange the session key**.
* After the session key is shared, all data uses **fast symmetric encryption**.

**Summary Flow**

| **Step** | **What Happens** |
| --- | --- |
| Browser gets server’s public key | From SSL certificate during handshake |
| Browser creates session key | Random symmetric key |
| Browser encrypts session key | Using server’s public key |
| Server decrypts session key | Using its private key |
| Both use session key | To encrypt and decrypt all further messages |

**Analogy: Public/Private Keys Are for Handshakes, Symmetric Keys Are for Talking**

* Imagine public/private keys are like exchanging a **secret password** securely.
* Once both sides know the password (session key), they use it to **talk privately**.
* You wouldn’t want to encrypt every word with a super complicated lock — just the initial password exchange.

If you want, I can:

* Explain the **TLS handshake** in more detail
* Show the difference between **asymmetric and symmetric encryption** with examples
* Walk you through how this looks in network tools (like Wireshark)

Just say the word!