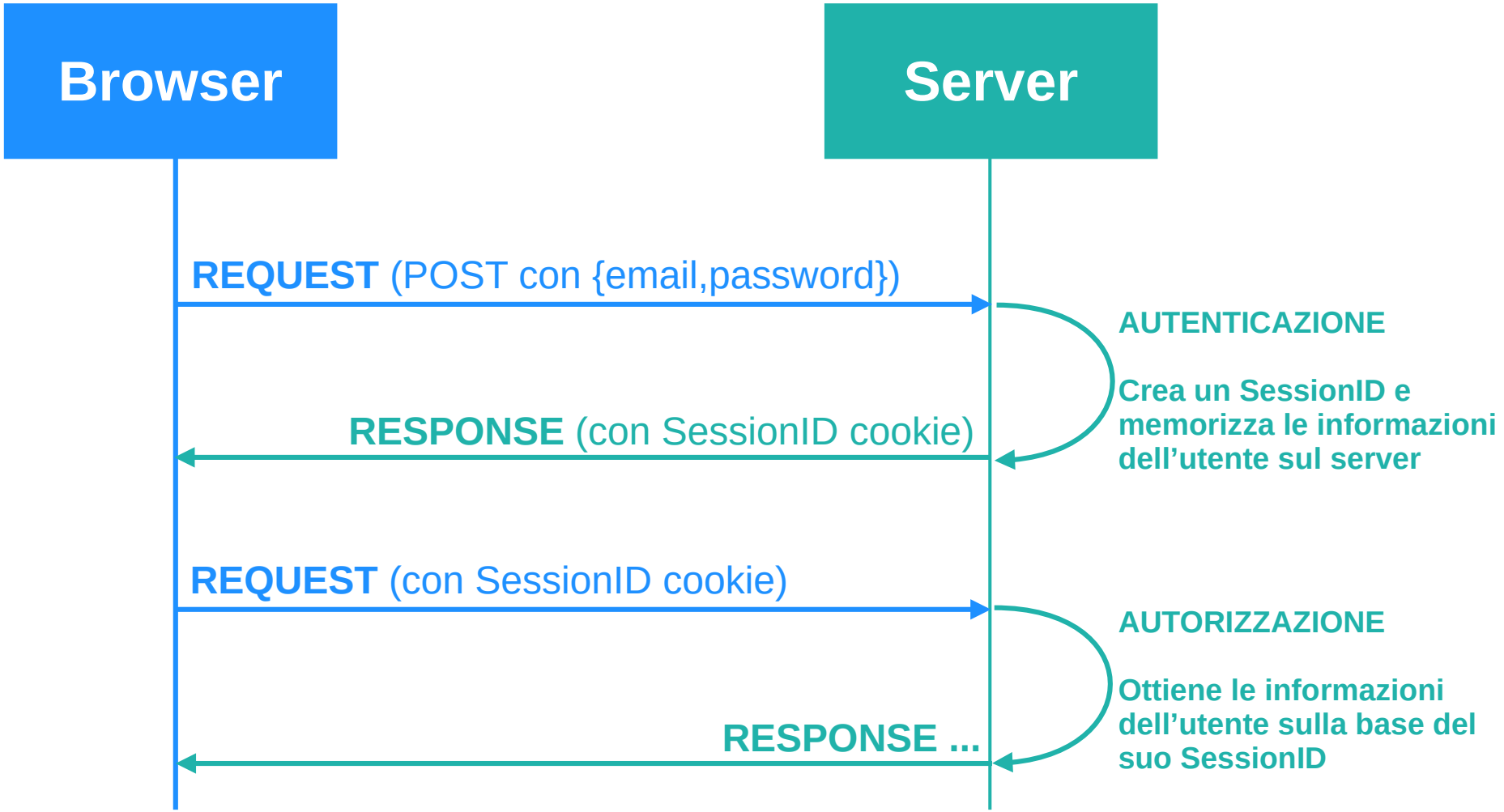
Università di Genova

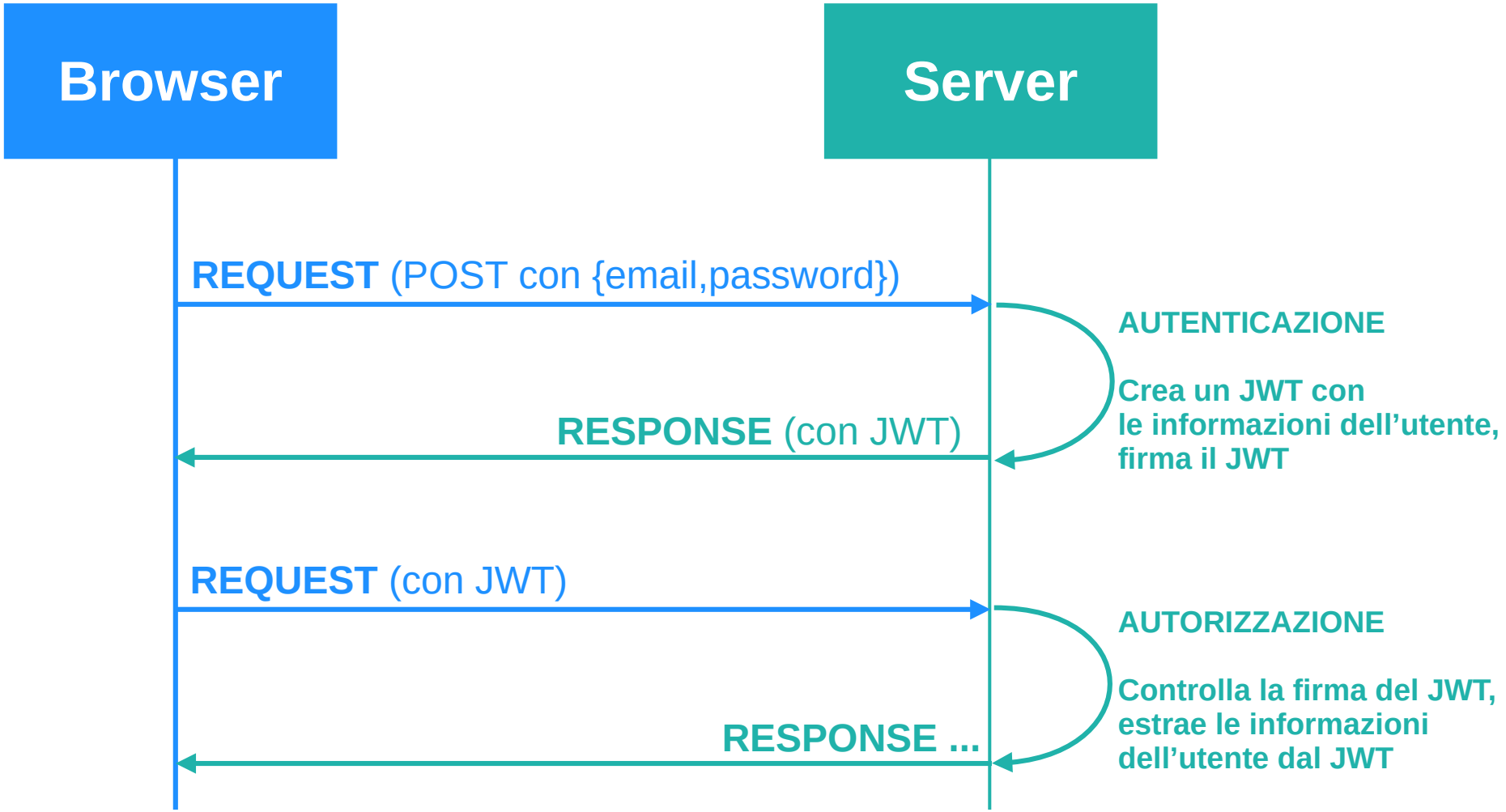
PHP (4)

Marina Ribaudo, marina.ribaudo@unige.it

Autenticazione e sessione



Autenticazione e JWT



JSON Web Token

- JWT **non** è usato per la fase di autenticazione (servono sempre username e password), ma per la **fase di autorizzazione**

- Il server **non memorizza informazioni dell'utente** e, per ogni interazione
  - **verifica** che il JWT non sia stato modificato
  - **deserializza** il JWT
  - **verifica** se l'utente è autorizzato ad accedere alla risorsa

JSON Web Token

Internet Engineering Task Force (IETF)
Request for Comments: 7519
Category: Standards Track
ISSN: 2070-1721

M. Jones
Microsoft
J. Bradley
Ping Identity
N. Sakimura
NRI
May 2015

JSON Web Token (JWT)

Abstract

JSON Web Token (JWT) is a compact, URL-safe means of representing claims to be transferred between two parties. The claims in a JWT are encoded as a JSON object that is used as the payload of a JSON Web Signature (JWS) structure or as the plaintext of a JSON Web Encryption (JWE) structure, enabling the claims to be digitally signed or integrity protected with a Message Authentication Code (MAC) and/or encrypted.

- Un JWT è formato da tre diverse parti (1) **Header**, (2) **Payload** e (3) **Signature**

JSON Web Token

- Il **Payload** JWT contiene dei **claim** che sono i “building block” del token
- Un **claim** è una **informazione** che descrive *qualcosa* sull'utente o sul token stesso
- Un token JWT può essere visto come una tessera universitaria che, per ogni studente, specifica
  - Nome, Cognome, Matricola, ...
- Un token JWT specifica informazioni simili nei suoi claim

JSON Web Token

- **Registered claims**: predefiniti, forniscono **metadati** utili per il token (https://datatracker.ietf.org/doc/html/rfc7519#section-4.1.1)
  - Chi ha generato il token, Scadenza del token,...
- **Public claims**: sono definiti dagli sviluppatori e devono evitare collisioni nei nomi (https://datatracker.ietf.org/doc/html/rfc7519#section-4.2)
  - Ruolo (es. Studente), Scuola (es. Scuola di Scienze MFN o Scuola Politecnica), Matricola,...
- **Private claims**: permettono la comunicazione tra il client (browser) e il server, per esempio fornendo informazioni sull'utente (https://datatracker.ietf.org/doc/html/rfc7519#section-4.3)
  - Identificatore interno (ID nel database)

JSON Web Token

- L'**Header** JWT contiene dei **claim** che sono usati per definire la sicurezza del token
  - Ad esempio, forniscono informazioni sull'algorimo crittografico usato per firmare il token

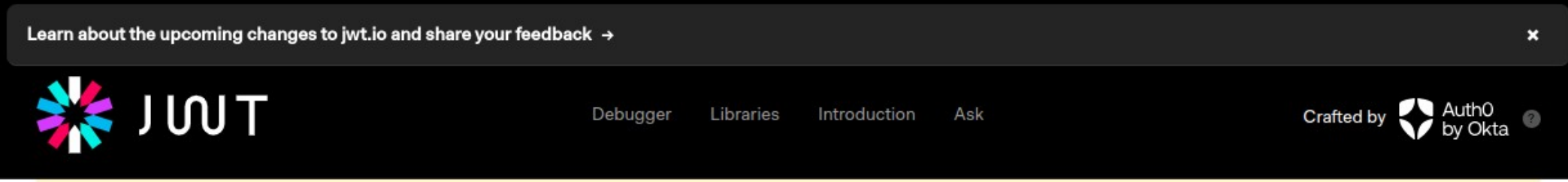
Tipo	Caratteristiche di Sicurezza	Caso d'Uso	Esempi di Algoritmi
JWT Non Sicuro	Nessuna (nessuna integrità o confidenzialità).	Test o applicazioni non sensibili.	`"alg": "none"`
JWT Firmato (JWS)	Garantisce integrità e autenticità.	Autenticazione, autorizzazione API.	`HS256`, `RS256`, `ES256`
JWT Cifrato (JWE)	Garantisce confidenzialità.	Trasmissione sicura di dati sensibili.	`RSA-OAEP`, `A256GCM`

JSON Web Token

- I claim sono utili per molte ragioni, tra cui
  - permettono di decidere la durata di validità del token
  - il server non deve interrogare il database tutte le volte per avere i dettagli dell'utente, che sono già nel token
  - si possono scrivere dati aggiuntivi fino a un massimo di 4 KB per garantire compatibilità con la maggior parte dei protocolli

JSON Web Token

- Casi d’uso
  - Single Sign-On tra domini multipli della stessa organizzazione
  - Access control nelle API
  - Comunicazione sicura tra microservizi


Learn about the upcoming changes to jwt.io and share your feedback →
JWT
Debugger
Libraries
Introduction
Ask
Crafted by
Auth0 by Okta
Algorithm
HS256
Encoded
PASTE A TOKEN HERE
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV_adQssw5c
Decoded
EDIT THE PAYLOAD AND SECRET
HEADER: ALGORITHM & TOKEN TYPE
{
"alg": "HS256",
"typ": "JWT"
}
PAYLOAD: DATA
{
"sub": "1234567890",
"name": "John Doe",
"iat": 1516239022
}
VERIFY SIGNATURE
HMACSHA256(
base64UrlEncode(header) + "." +
base64UrlEncode(payload),
your-256-bit-secret
) secret base64 encoded

https://jwt.io/

Implementazione in PHP

- Esistono varie librerie, per esempio https://github.com/firebase/php-jwt
  - Permette **encoding/decoding** di **JWT**
  - Non si occupa dell'invio del token JWT, che si può gestire con PHP/JavaScript mediante l'uso di opportuni header HTTP

Header per JWT

- Nel caso di applicazioni web tradizionali, lato server si usa l'header **Set-Cookie:** … , e il browser ritorna l'header **Cookie:** jwt = <jwt token value>

```
setcookie("jwt", $jwt, [
"expires" => time() + 3600,
"path" => "/",
"secure" => true,          // Solo HTTPS
"httponly" => true,        // Disabilita accesso via JavaScript
"samesite" => "Strict",    // Disabilita CSRF
]);
```

Header per JWT

- Il browser può anche inviare l'header, **Authorization: Bearer** <jwt token value>
  - **Bearer** specifica che il client presenta il token come prova della propria identità e autorizzazione senza ulteriori credenziali nella
  - Il token JWT deve essere **salvato nel browser** via JavaScript (nella Local Storage, nella Session Storage, in una variabile JavaScript)
  - Sempre via JavaScript, va costruito e inviato l'header **Authorization:** (solitamente usando l'API Fetch)

Header per JWT

Caratteristica	`Authorization: Bearer`	`Cookie`
Manuale/Automatico	Richiede implementazione manuale da parte dello sviluppatore.	Gestito automaticamente dal browser.
Richieste Cross-Origin	Deve includere le credenziali manualmente (es. `fetch` con CORS).	I cookie sono limitati per impostazione predefinita (politiche `SameSite` e CORS).
Sicurezza	Il token memorizzato in local/session storage è vulnerabile agli attacchi XSS.	I cookie possono essere resi `HttpOnly` per prevenire attacchi XSS, ma sono vulnerabili agli attacchi CSRF.
Caso d'Uso Previsto	Applicazioni basate su API (es. SPA) dove il token è gestito esplicitamente dal client.	Applicazioni web tradizionali in cui i cookie sono lo standard per la gestione delle sessioni.