Tecnologie e applicazioni web

Autenticazione

Filippo Bergamasco (filippo.bergamasco@unive.it)
http://www.dais.unive.it/~bergamasco/
DAIS - Università Ca'Foscari di Venezia
Anno accademico: 2017/2018

Autenticazione

Il meccanismo dei cookie permette al server web di memorizzare coppie chiave-valore su un client che effettua una richiesta

- E' possibile "riconoscere" un determinato client in transazioni successive ma non la sua identità reale

HTTP prevede, nel protocollo, meccanismi con cui un client può **autenticarsi** attraverso delle credenziali

Autenticazione

Con autenticazione intendiamo mostrare una qualche "prova" dell'identità fisica di un determinato client

L'autenticazione è solitamente basata su una qualche informazione segreta e condivisa tra client e server (ex. Username e password) che va scambiata in modo sicuro attraverso un determinato protocollo

Autenticazione in HTTP

Quando un client cerca di accedere ad una risorsa considerata dal server “riservata”, il server può rispondere con lo status code 401 (Login required).

Insieme allo status code, l’header WWW-authenticate informa il client sul tipo di informazioni richieste per l’autenticazione

User login

Il browser, in base al tipo di autenticazione:

- Richiede username/password all'utente
- genera un header contenente le credenziali di accesso
- Utilizza l'header per tutte le richieste successive alle risorse appartenenti ad un certo "authorization realm"

Meccanismo simile a quello dei cookie: le informazioni di autenticazione sono scambiate negli headers HTTP

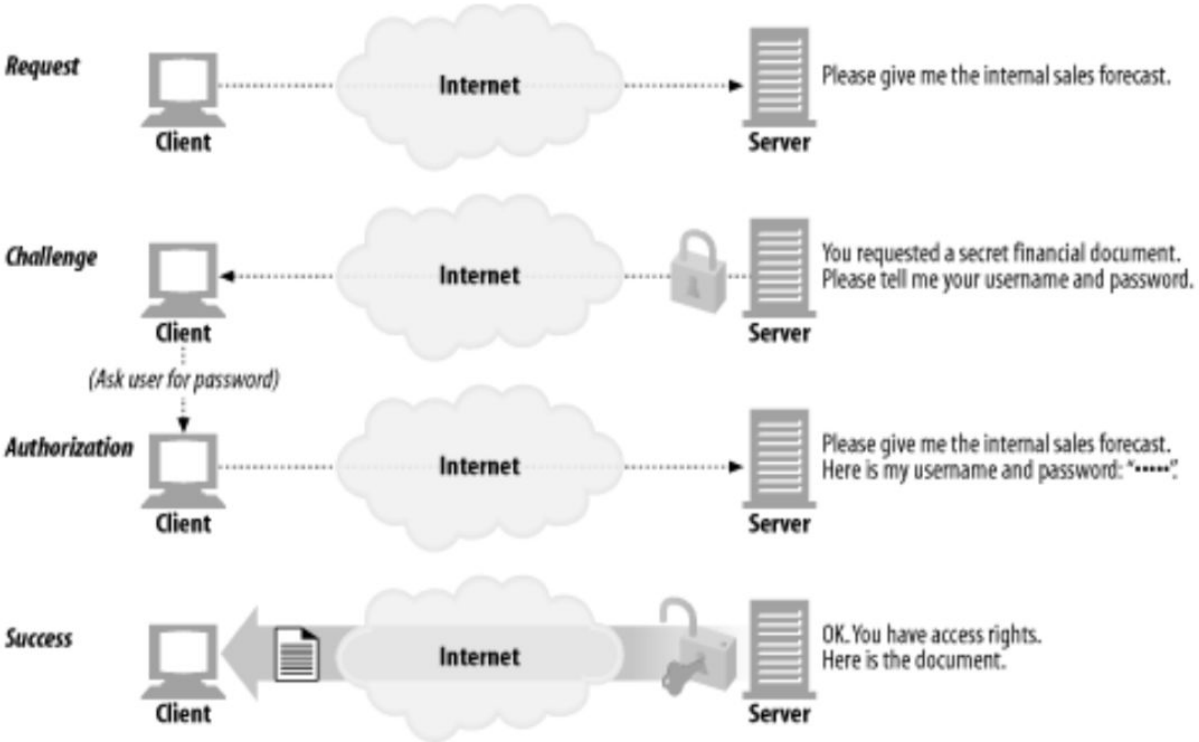
Tipologie di autenticazione

HTTP implementa due tipologie di di autenticazione:

1. Basic access authentication
2. Digest access authentication

Entrambi basati su framework di tipo challenge-response, ma differiscono sul modo in cui sono codificate e scambiate le credenziali.

Challenge-response



< Challenge

< Response

Basic authentication

Descritto originariamente in HTTP/1.0, implementato in tutti i browser oggi esistenti

- Il server può rifiutare una transazione, invitando (challenging) il client ad inviare la coppia username:password
- In caso di risposta corretta la risorsa viene inviata nella transazione seguente
- Se la risposta non è corretta, la challenge è ripetuta

Basic authentication

1. Il client invia una **request** al server (GET, POST, HEAD, etc.) per accedere ad una risorsa
2. Se la risorsa richiede credenziali di accesso, il server inserisce il seguente header nella **response**:

   `WWW-Authenticate: Basic realm="<nome-realm>"`
3. Il client recupera la coppia username/password dall'utente e la codifica in una stringa:

   <crd> = `base64( <username>:<password> )`

Basic authentication

4. Il client, ad ogni request successiva, invia il seguente header:

   Authorization: Basic <crd>

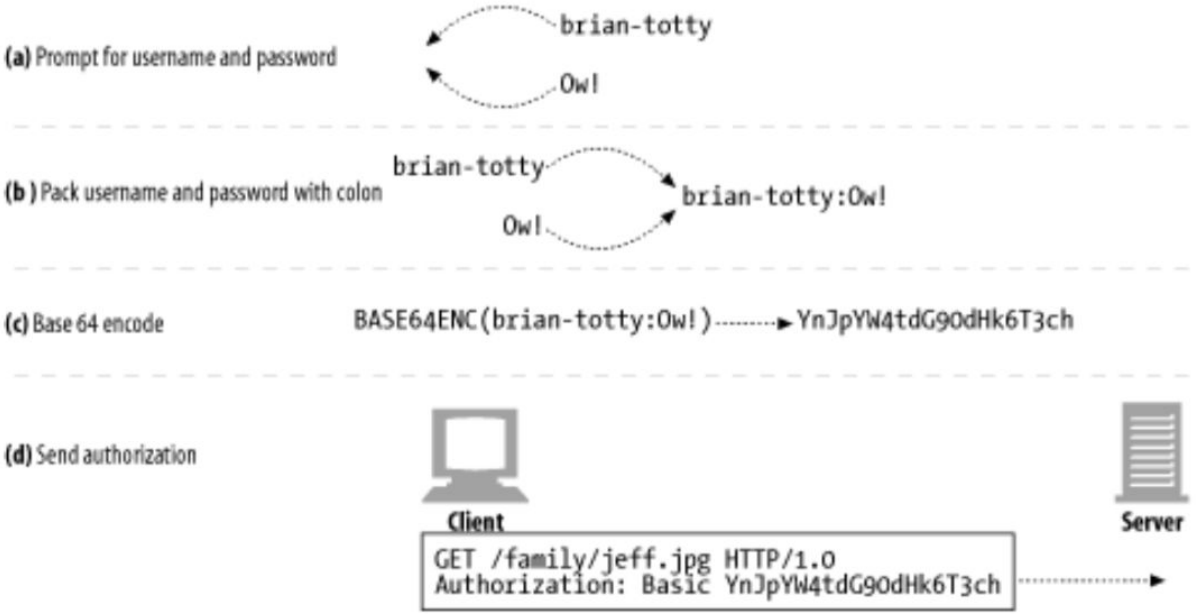
Il server decodifica <crd> in base64 per ottenere la stringa <username>:<password> e verifica le credenziali. Se sono validi, la risorsa è restituita come nel caso senza autenticazione

Codifica Base-64

La codifica Base-64 è una funzione **invertibile**, utilizzata per codificare un qualsiasi stream di bytes in una stringa contenente soltanto caratteri alfanumerici.

Importante: Essendo invertibile, la funzione non è pensata per rendere il trasferimento delle credenziali “sicuro” ma solo offuscato

Codifica Base-64

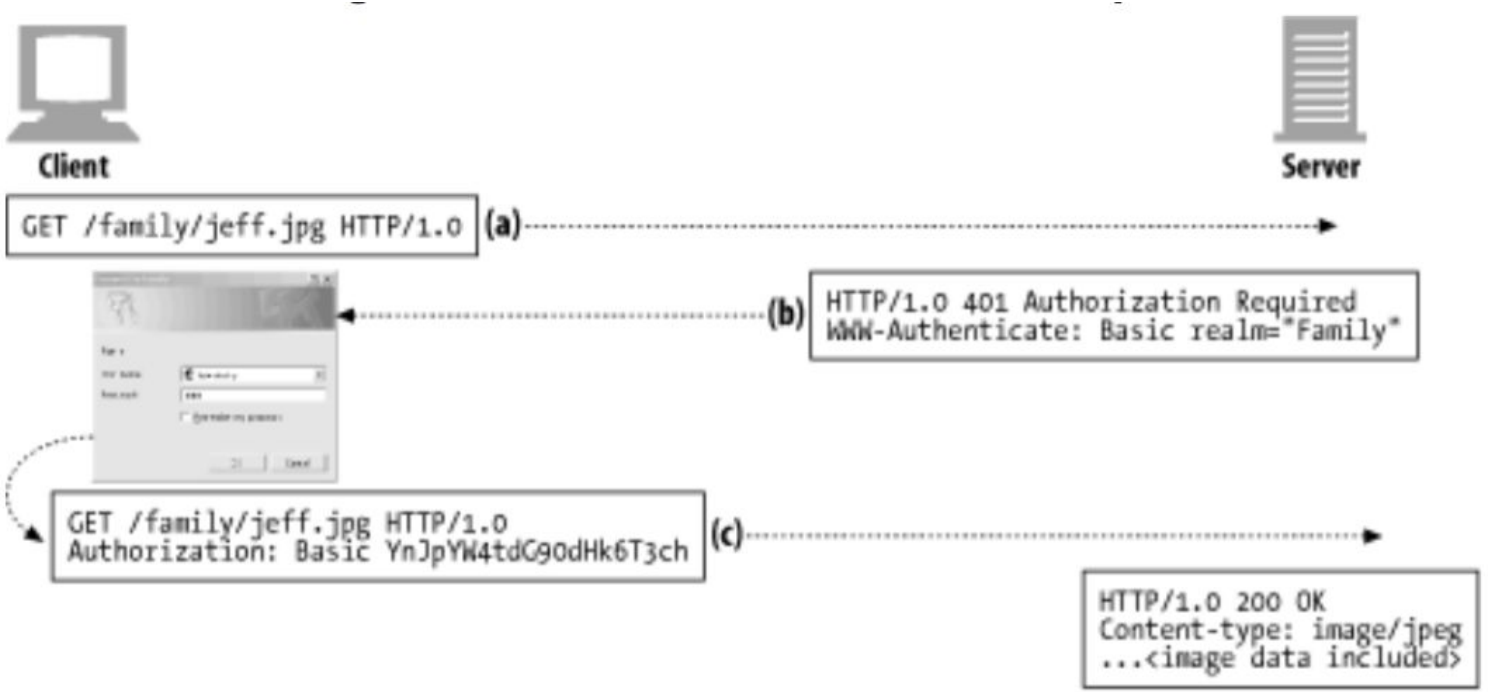


Codifica Base-64

Perchè la stringa <username>:<password> non è inviata in chiaro?

1. Caratteri non plain-ascii della password (ex. Lettere accentate) possono essere inserite nell'header senza problemi
2. La stringa è offuscata per impedire la lettura delle credenziali semplicemente osservando la stringa (necessario calcolare la funzione inversa)

Basic authentication



BA: problemi

1. La coppia user/pass anche se non è inviata in chiaro è **facilmente decodificabile**. Il livello di sicurezza è di fatto lo stesso di inviare la coppia senza alcuna codifica.

Se un terzo soggetto può osservare il canale di comunicazione, può rubare e usare le credenziali di accesso. **Soluzione:** utilizzare BA solo con HTTPS

BA: problemi

2. Anche se l'autenticazione è usata per applicazioni non-critiche, gli utenti potrebbero riciclare stesse coppie username/password utilizzate anche su altri siti sensibili

Problema di social behaviour degli utenti.

BA: problemi

3. Anche se l'autenticazione avviene correttamente e l'attaccante non è interessato alle credenziali, non vi è alcun legame tra il contenuto della risorsa e le credenziali.

Un soggetto che si interpone tra la comunicazione può cambiare l'intero contenuto delle risorse lasciando inalterati gli header di autenticazione
Soluzione: utilizzare BA solo con HTTPS

BA: problemi

4. Sensibile allo spoofing da parte di server contraffatti. In altre parole, il client non può verificare l'identità del server, che può quindi spacciarsi per il server originale

Soluzione: utilizzare certificati digitali per autenticare il server prima di procedere con basic access authentication (usare HTTPS)

BA: utilizzi

Se non accoppiato con HTTPS, BA è utile soltanto per fornire un semplice livello di autenticazione **dove la privacy dei contenuti è desiderata ma non assolutamente necessaria**.

Blocca di fatto soltanto utenti curiosi ma non determinati ad accedere alla risorsa

Digest Authentication

Il problema principale di basic authentication è che la coppia username:password è inviata in chiaro nel canale.

Utilizzare tecniche di crittografia (simmetrica/asimmetrica) per creare un canale sicuro complicano però il protocollo per instaurare un canale sicuro

Digest Authentication

Il server non ha bisogno di ricevere dal client la sua password (segreto condiviso) per poterlo autenticare, ma soltanto un digest (riassunto) della password per verificare se corrisponde con quella presente in suo possesso

Si utilizzano **funzioni di hashing** per calcolare questi digest

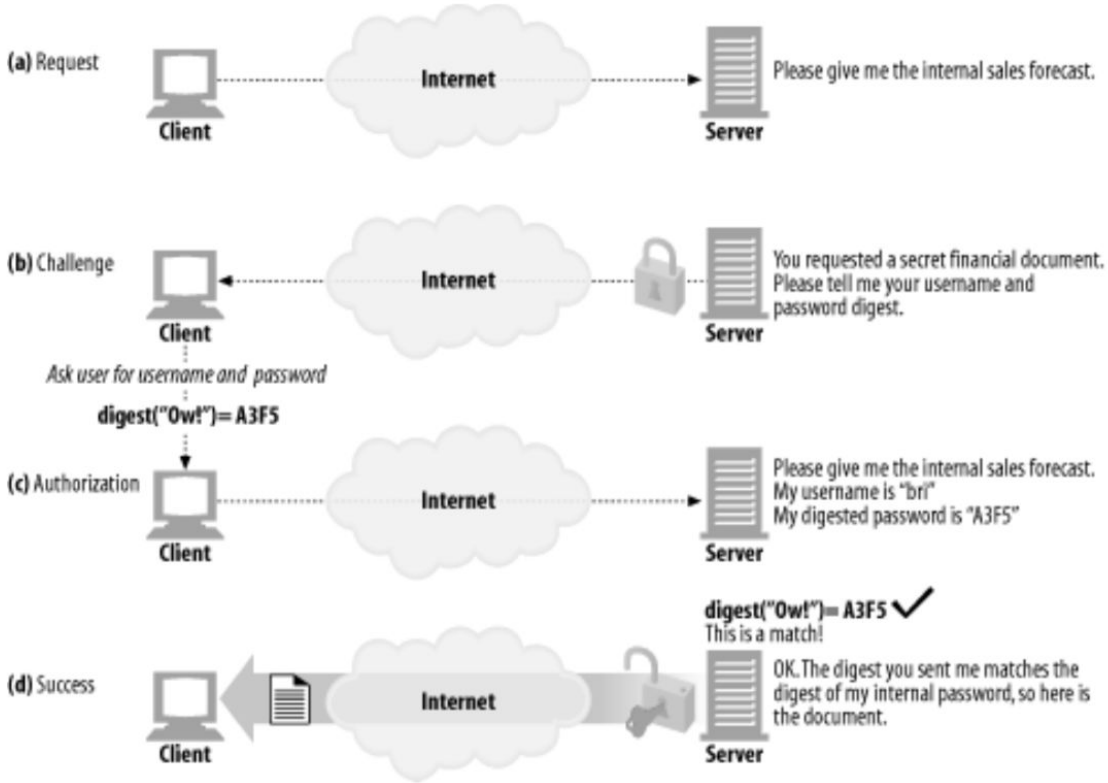
Funzioni di hash

Una funzione di hash permette di convertire un messaggio (stringa) di lunghezza arbitraria in una sequenza di lunghezza fissa (ex. 128 bit)

Caratteristiche:

- Messaggi uguali generano stessi hash
- La funzione è one-way. Da un hash è praticamente impossibile “indovinare” il messaggio originale

Autenticazione con digest



Autenticazione con digest

Utilizzando una funzione di hash è possibile evitare di inviare le password in chiaro nel canale.

Problema: per definizione la stessa password genererà sempre lo stesso hash (digest).

Se un soggetto in ascolto del canale legge il digest, può usarlo per autenticarsi anche senza conoscere la password: **Replay attack**

Nonce

Per evitare replay-attack, il server invia al client insieme alla richiesta di autenticazione un token speciale chiamato nonce.

Il client crea il digest della coppia (<password>,<nonce>) di modo da generare digests ogni volta diversi ma utilizzabili soltanto una volta

Digest Authentication

Il protocollo (semplificato) che permette l'autenticazione del client attraverso Digest Authentication è il seguente:

1. Come nel caso di BA, il server inserisce nella propria response l'header WWW-Authenticate inviando: realm di autenticazione, una lista di algoritmi di hashing supportati e il nonce

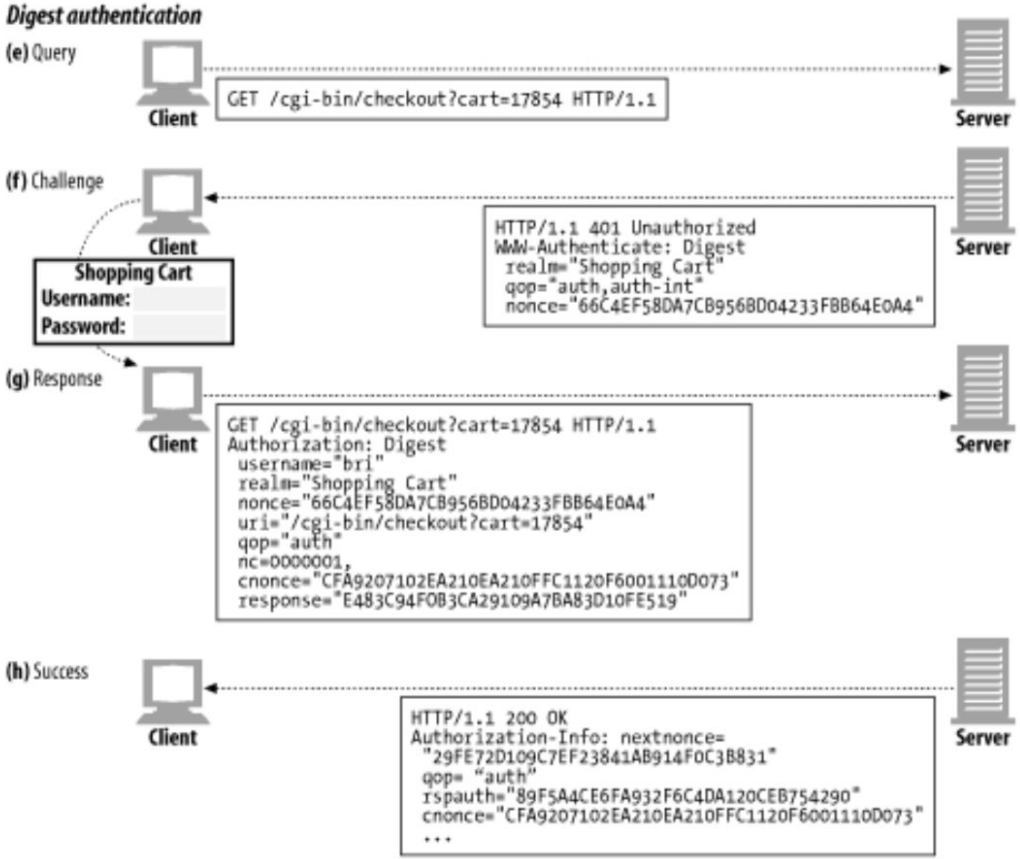
Digest Authentication

2. Se l'algoritmo di hashing è supportato, il client crea il digest della tripla (user, nonce, password)
3. Il client invia nella request successiva, attraverso l'header Authorization, il proprio username e il digest calcolato. Se il client lo desidera, genera a sua volta un nonce da inviare al server
4. Il server genera il digest associato al client e lo confronta con quello ricevuto

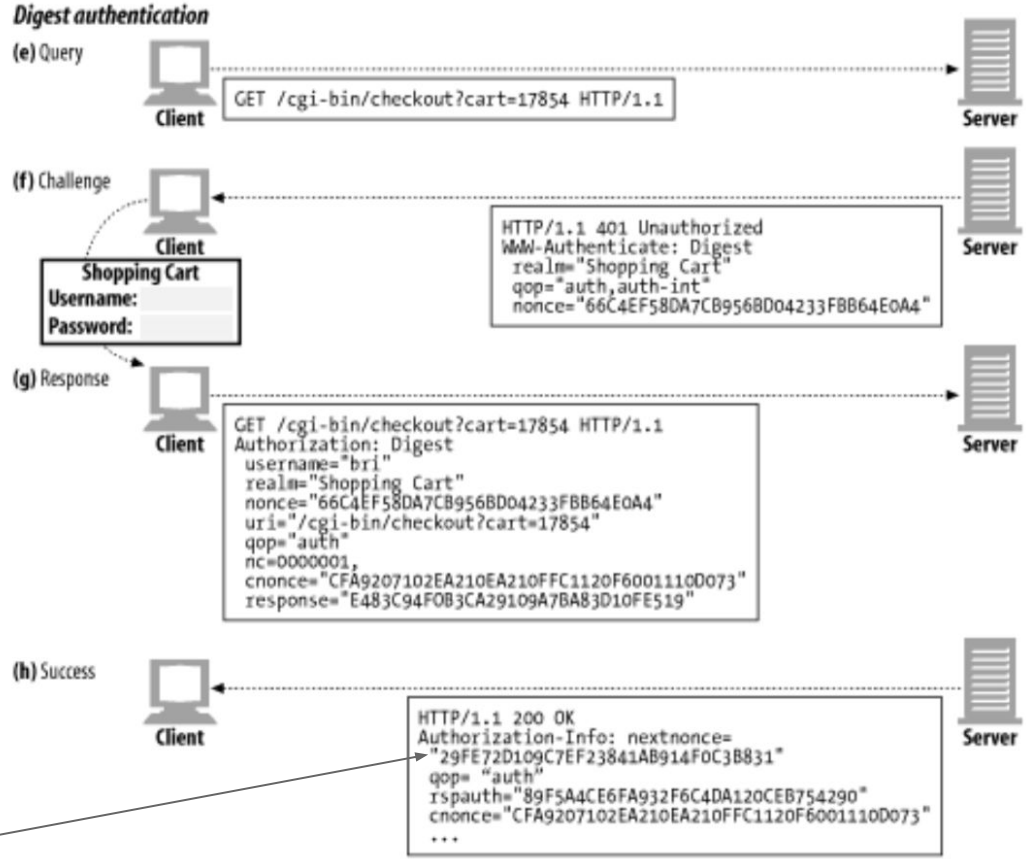
Digest Authentication

5. Se i due digest sono uguali, il client è autenticato. Altrimenti, la richiesta viene rifiutata e viene generata una nuova challenge
6. Se il client aveva a sua volta inviato un nonce, viene generato il digest per il client e ritornato nell'header Authorization-Info

Digest Authentication

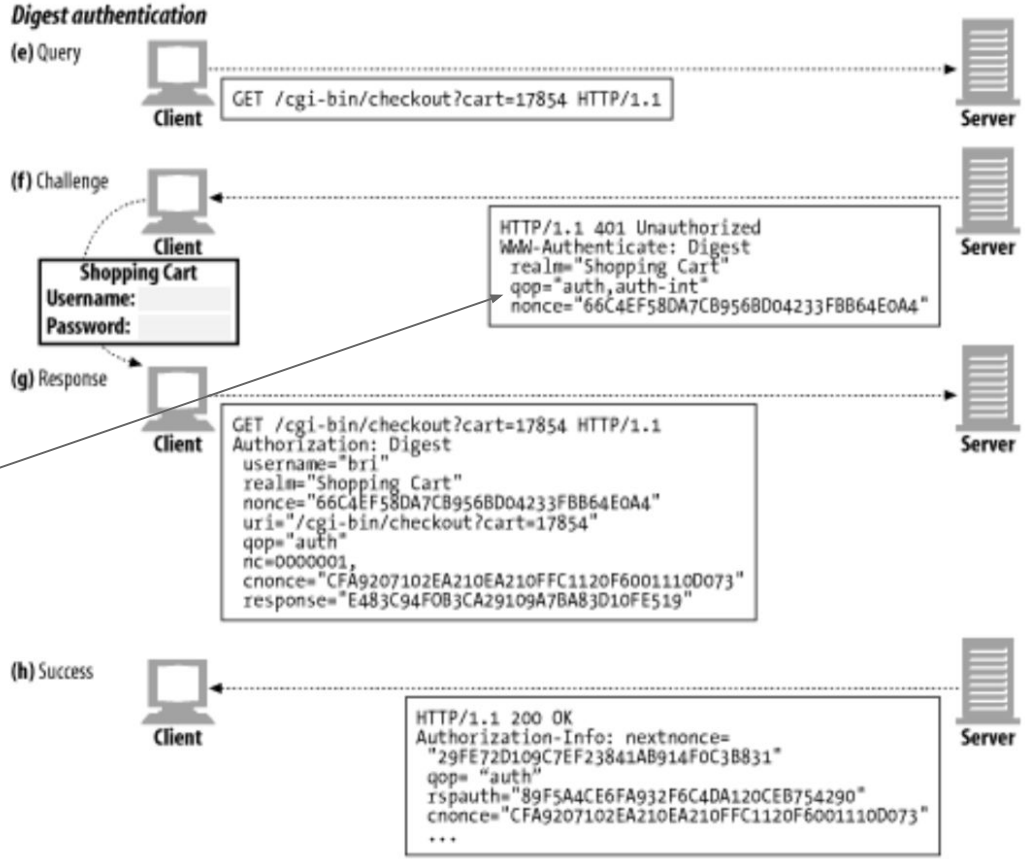


Digest Authentication



nextnonce permette al client di inviare una nuova richiesta con già l'Authorization header formato senza passare per una nuova challenge da parte del server

Digest Authentication



Qop (quality of protection enhancements) permette al client e al server di negoziare il tipo di sicurezza da implementare (ex. Integrity protection anche dei message body)

DA: Vantaggi

Digest authentication risolve gran parte dei problemi di BA:

- La password non è inviata in chiaro
- Replay attack non possibili (a patto che il nonce cambi ogni volta)
- Il client può verificare anche che il server sia lo stesso che ha richiesto la challenge (client nonce) per evitare attacchi da server malevoli

DA: Problemi

- Vari livelli di qualità di sicurezza (per retro-compatibilità con le prime versioni) possono portare problemi di sicurezza
- Non è possibile verificare l'identità del server ma soltanto che il server sia lo stesso soggetto che ha generato la challenge (man-in-the-middle attack possibile)
- Algoritmo MD5 considerato non più sicuro