POLITECNICO DI TORINO

Corso di Laurea in Ingegneria Informatica

Tesi di Laurea

Gestione automatica di certificati X.509

Relatore
prof. Antonio Lioy

Candidato
Roberta Sgroi

Dicembre 2019

Ringraziamenti

Indice

Elenco delle figure 8

Elenco delle tabelle 9

1 Introduzione 10

2 Let's Encrypt 12

2.1 Richiesta di un certificato standard . . . 12

2.2 Richiesta di un certificato utilizzando Let's Encrypt . . . 13

2.2.1 Generalità . . . 14

2.2.2 Richiedere un certificato . . . 15

2.2.3 Limiti . . . 18

3 Il Protocollo ACME 19

3.1 Panoramica sul Protocollo . . . 19

3.2 Json Web Signature . . . 20

3.2.1 Creazione di un oggetto JWS . . . 21

3.3 Configurazioni per le richieste HTTPS . . . 21

3.4 Validazione degli account . . . 23

3.4.1 Validazione HTTP . . . 24

3.4.2 Validazione DNS . . . 24

3.5 Considerazioni sulla sicurezza . . . 24

4 Analisi dei processi di comunicazione tra *client* e *server* 27

4.1 Oggetti utilizzati . . . 28

4.1.1 Oggetto account . . . 28

4.1.2 Oggetto ordine . . . 29

4.1.3 Oggetto autorizzazione . . . 29

4.1.4 Oggetto sfida . . . 30

4.2 Directory . . . 31

4.2.1 Specifiche del protocollo . . . 31

4.2.2 Server Apache . . . 32

4.3 Nonce . . . 32

4.3.1 Specifiche del protocollo . . . 32
4.3.2 Server Apache . . . 33
4.4 Gestione di un account . . . 33
4.4.1 Specifiche del protocollo - Creazione di un account . . . 33
4.4.2 Server Apache - Creazione di un account . . . 35
4.4.3 Specifiche del protocollo - Associazione con un account esterno . . . 37
4.4.4 Specifiche del protocollo - Aggiornamento delle chiavi . . . 38
4.4.5 Specifiche del protocollo - Disattivazione dell'account . . . 39
4.4.6 Server Apache - Disattivazione dell'account . . . 40
4.5 Richiesta di un certificato . . . 42
4.5.1 Specifiche del protocollo - Richiesta di un certificato . . . 42
4.5.2 Server Apache - Richiesta di un certificato . . . 44
4.5.3 Specifiche del protocollo - Pre-autorizzazione . . . 47
4.5.4 Specifiche del protocollo - Download di un certificato . . . 47
4.5.5 Server Apache - Download di un certificato . . . 48
4.6 Autorizzazione di un identificatore . . . 50
4.6.1 Specifiche del protocollo . . . 50
4.6.2 Server Apache . . . 52
4.7 Revoca di un certificato . . . 56
4.7.1 Specifiche del protocollo . . . 56
4.7.2 Server Apache . . . 57

5 Adattamento implementativo del software 61
5.1 Certbot . . . 61
5.2 Analisi dell'implementazione . . . 61

6 Risultati 66
6.1 Directory . . . 67
6.2 Nonce . . . 67
6.3 Gestione di un account . . . 68
6.3.1 Creazione di un account . . . 68
6.3.2 Disattivazione di un account . . . 70
6.4 Richiesta di un certificato . . . 73
6.4.1 Richiesta di un certificato . . . 73
6.4.2 Autorizzazione dell'identificatore . . . 74
6.4.3 Download del certificato . . . 80
6.5 Rinnovo di un certificato . . . 82
6.6 Revoca di un certificato . . . 90
6.7 Risultati . . . 92

7 Conclusioni e sviluppi futuri 93

Appendices 95

A SSL e Certificati 96

A.1 I Certificati 96

A.1.1 Crittografia e Certificati a Chiave Pubblica 96

A.1.2 Certification Authority 97

A.1.3 Certificati X.509 98

A.2 Il Protocollo SSL 100

A.2.1 Autenticazione in SSL 101

A.2.2 Riservatezza dei Messaggi 104

A.2.3 Autenticazione e Integrità dei Messaggi 106

A.2.4 Settaggio del Canale Sicuro 107

A.2.5 Vantaggi e Svantaggi 108

Bibliografia 109

Elenco delle figure

2.1 Let's Encrypt certificate's chain . . . 14
2.2 Certificate request . . . 15
2.3 Certificate request validation . . . 16
2.4 Certificate issuance . . . 17
2.5 Certificate revocation . . . 17

3.1 ACME Resources and Relationships . . . 20
3.2 ACME Communication Channels . . . 25

4.1 Response from GET directory. . . . 31
4.2 Response from HEAD nonce. . . . 33
4.3 POST-Request for new account creation. . . . 34
4.4 POST-Response for new account creation. . . . 35
4.5 POST-Request for new account creation with external account binding. . . . 38
4.6 POST-Request to update certificate associated keys. . . . 39
4.7 POST-Request to deactivate an account . . . 40
4.8 POST-Request for certificate issuance . . . 42
4.9 POST-Response for certificate issuance . . . 43
4.10 POST-Request to finalize the certificate order . . . 43
4.11 POST-Response to finalize the certificate order . . . 44
4.12 POST-Request for a new account authorization . . . 47
4.13 POST-Request/Response for certificate download . . . 48
4.14 POST-Request for identifier authorization . . . 51
4.15 POST-Request to notify the server for challenges checks . . . 52
4.16 POST-Request to deactivate authorization of identifiers . . . 52
4.17 POST-Request for certificate revocation - signed using the account key . . . 56
4.18 POST-Request for certificate revocation - signed using the private key . . . 57
4.19 POST-Response for certificate revocation . . . 57

A.1 Certificate Revocation List . . . 98
A.2 Certificato X.509 . . . 99
A.3 SSL Handshake . . . 101
A.4 Sistema a Sfida Simmetrico . . . 102
A.5 Sistema a Sfida Asimmetrico . . . 103
A.6 SSL Handshake . . . 107

Elenco delle tabelle

4.1 Client-Server sequence of requests. 28
4.2 Response fields from GET directory. 31

A.1 Algoritmi di criptaggio simmetrico 104
A.2 Algoritmi crittografici di Hash 106

Capitolo 1

Introduzione

La percentuale di utilizzatori del *World Wide Web* è cresciuta esponenzialmente nel corso degli anni. Conseguentemente è aumentata a dismisura anche la quantità di informazioni sensibili che vengono trasmesse tramite internet. È stato dunque indispensabile pensare ad un meccanismo che riducesse l'esposizione del contenuto delle comunicazioni che avvengono attraverso le reti esposte su internet, così da fornire protezioni per tutte le informazioni scambiate durante il processo stesso.

A partire da questo obiettivo, sono nati i protocolli crittografici per il *Presentation Layer* del modello *ISO/OSI*[1]. Il primo dei quali è stato *SSL*[2] seguito poi da *TLS*[3].

L'utilizzo di questi protocolli, consente di creare comunicazioni sicure tra un *endpoint* e un altro, mediante istaurazione di un canale in cui le informazioni vengono trasmesse in maniera crittografata. Il ruolo svolto dalle *certification authority* e i relativi certificati emessi, rappresenta un punto focale nell'utilizzare un meccanismo di crittografia attendibile e allo stesso tempo consente alle parti coinvolte, una lettura comprensibile delle informazioni scambiate. Maggiori dettagli sui certificati e su tutti gli step eseguiti per istaurare una connessione sicura sono descritti nell'appendice A.

Il processo di emissione di un certificato e del suo successivo mantenimento, non è semplice ed economico. Lo standard, ormai affermato, ha sia un costo dal punto di vista economico che dal punto di vista dell'effort richiesto. È infatti indispensabile che un utente abbia le abilità tecniche necessarie ad effettuare la richiesta di emissione e che successivamente si occupi della gestione del certificato (rinnovi, eventuali revoche).

Una nuova emergente *certification authority*, *Let's Encrypt*, ha messo a punto, in collaborazione con l'*ISRG*[4], un innovativo meccanismo di richiesta e gestione dei certificati, al fine di semplificare la vita del richiedente. Con lo stesso obiettivo di fornire sicurezza e privacy delle informazioni sensibili, *Let's Encrypt* fornisce un servizio totalmente gratuito e sopratutto automatizzato. Si usufruisce di quest'ultima importante funzionalità grazie all'implementazione di un software open source, *Certbot*, che consente le comunicazioni con la *certification authority*. In questo processo entrambe le parti devono rispettare gli standard definiti dal protocollo *ACME*[5], il quale è stato creato ad hoc per poter consentire le comunicazioni automatizzate tra la *certification*

[1] *International Organization for Standardization/Open System Interconnecion* - è uno standard che definisce le varie funzioni del processo di comunicazione nell'ambito delle telecomunicazioni. Il modello è suddivisibile in due livelli, i tre più bassi relativi ai mezzi fisici necessari per istaurare una comunicazione (in ordine: Fisico, Collegamento e Rete), e i quattro più alti relativi a funzionalità che l'*host* deve esporre per consentire una corretta comunicazione (in ordine: Trasporto, Sessione, Presentazione e Applicazione).

[2] *Secure Sockets Layer.*

[3] *Transport Layer Security.*

[4] *Internet Security Research Group.*

[5] *Automatic Certicate Management Environment.*

authority ed il richiedente . Recentemente il protocollo è stato standardizzato dall'*IETF*[6] come *RFC-8555*[7].

Tra le funzionalità base del software vi è la generazione delle chiavi da associare ad un certificato. Potenzialmente potrebbe risultare vincolante per un utente che è già in possesso di una coppia di chiavi. Lo scopo di questo lavoro di tesi è stato quello di verificare l'eventuale possibilità di convalidare certificati utilizzando chiavi generate esternamente, ad esempio utilizzando un *HSM*[8].

Nel raggiungere l'obiettivo finale, si è partiti da una prima fase di analisi del protocollo e del software. Successivamente si è passati ad una prima fase di test della versione attualmente implementata e disponibile per l'installazione. Lo scopo ultimo di questa fase è stato quello di appurare l'effettiva corretta interazione tra le parti coinvolte. Si è dunque poi passati all'implementazione delle modifiche necessarie per far funzionare il software utilizzando chiavi generate esternamente. Completate le modifiche al software, si è passati alla fase di validazione delle nuove funzionalità aggiunte a *Certbot*. Tale fase ha previsto una serie di test che hanno appurato l'effettiva possibilità di integrare la nuova funzionalità proposta.

Questo elaborato è strutturato come segue:

- ***Capitolo 2*** - presenta informazioni generali sulla nuova *certification authority Let's Encrypt* e l'analisi dei servizi forniti;
- ***Capitolo 3*** - descrive l'analisi del protocollo *ACME* che definisce gli standard di comunicazione tra *Let's Encrypt* e *Certbot*;
- ***Capitolo 4*** - contiene l'analisi dei risultati relativamente ai test effettuati sul processo di comunicazione;
- ***Capitolo 5*** - fornisce i dettagli relativi all'implementazione effettuata;
- ***Capitolo 6*** - riporta l'analisi dei risultati sulla comunicazione tra *Let's Encrypt* e *Certbot* in seguito alle modifiche implementative;
- ***Capitolo 7*** - espone le conclusione ed i possibili sviluppi futuri.

[6] *Internet Engineering Task Force.*

[7] *Request for Comments.*

[8] *Hardware Security Module.*

Capitolo 2

Let's Encrypt

Un sito web viene considerato affidabile quando permette la navigazione utilizzando il protocollo *SSL*. Come largamente discusso nel capitolo precedente, questo è possibile solo se sul server è installato un certificato rilasciato da una *certification authority* riconosciuta.

Una *certification authority* si occupa dunque di verificare che il richiedente di un certificato sia l'effettivo possessore o controlli il dominio per il quale si sta facendo richiesta. Di fatto quello che viene effettuato dalla *certification authority* è quella che si chiama *Domain-Validation*. Questa validazione esclude il controllo sulla *real-world identity*, contemplata invece in quelle che sono validazioni di organizzazioni (*Organization-Validation*[1]) e validazioni estese (*Extended Validation*[2]).

2.1 Richiesta di un certificato standard

Prima di introdurre quelli che sono i vantaggi apportati da *Let's enctypt*, effettuiamo una panoramica su quello che è il processo standard per la richiesta e il mantenimento di un certificato. I passi base e le informazioni necessarie ad una *certification authority* per l'emissione di un certificato, sono stati già stati descritti nel capitolo precedente. Vediamo adesso in che modo viene coinvolto il richiedente:

- Generazione di una richiesta PKCS#10 *Certificate Signing Request (CSR)*;
- La *CSR* viene inserita manualmente nell'apposita richiesta effettuata alla *certification authority* tramite sito web;
- L'utente deve provare di possedere il controllo dei domini per i quali viene effettuata richiesta, rispondendo alla sfida fornita dalla *certification authority* in uno dei seguenti metodi:
  - L'utente posta la risposta alla sfida in uno specifico path sul *web server* per il quale effettua richiesta di convalida;
  - L'utente posta la risposta alla sfida nel *DNS* record corrispondente al dominio per il quale effettua richiesta di convalida;
  - L'utente inserisce la risposta alla sfida sul sito web della *certification authority*, questo si verifica nel caso in cui quest'ultima ha inviato la sfida all'indirizzo di posta elettronica del richiedente e che dovrebbe corrispondere a quello dell'amministratore del dominio.

[1] L'OV viene utilizzata da aziende, organizzazioni per la convalida di un dominio aziendale. Di fatto è più sicura per le aziende che gestiscono business online.

[2] L'EV verifica anche l'entità legale del richiedente.

- All'emissione del certificato, l'utente deve scaricarlo e installarlo sul *web server* identificato dal dominio.

Ad eccezione della creazione della *CSR* e della generazione stessa del certificato, questo modo di procedere richiede un grande coinvolgimento dell'utente che effettua una richiesta di certificato. Le istruzioni sono spesso di difficile comprensione e la loro esecuzione può durare a lungo (test di usabilità hanno indicato che possono volerci dall'una alle tre ore per avere un certificato correttamente installato). Si può facilmente intuire che il rilascio di un certificato e la sua successiva gestione, necessaria per mantenerne la validità, comporta dei costi, più o meno elevati, per il proprietario del *server* che vuole usufruire di questo servizio fornito dalle *certification authority*.

2.2 Richiesta di un certificato utilizzando Let's Encrypt

Let's Encrypt, che da questo momento in poi verrà identificata dall'acronimo *LE*, è una nuova *certification authority* che, grazie al supporto fornito dall'*ISRG* [3], garantisce esattamente gli stessi servizi, ma non richiede nessun costo di fornitura e manutenzione. Inoltre, l'elemento innovativo introdotto da *LE* riguarda l'automatizzazione del processo di richiesta, emissione, rinnovo ed eventuale revoca di un certificato X.509 e questo si traduce nella completa eliminazione delle interazioni da parte dell'utente. A supporto dell'intero processo vi sono il protocollo *ACME* [4] e il software *Certbot* che si occupa di tutte le interazioni tra il *web server* e la *certification authority*.

LE è una *certification authority* ancora recente e poco conosciuta e ciò la rende poco affidabile per la maggior parte dei browser. In attesa di essere riconosciuta ufficialmente[5], è stato ovviato il problema della scarsa attendibilità attraverso il contributo di una *certification authority* già accreditata. Il certificato di root di *LE* viene mantenuto offline e vengono eletti degli intermediari per l'emissione effettiva dei certificati. L'affidabilità degli intermediari è accertata dal fatto che esistono due certificati che li rappresentano, uno viene firmato da *ISRG* e l'altro da *IdenTrust* [6]. Osservando la figura 2.1 è possibile analizzare la catena di *certification authority* utilizzata in *LE*. Partendo dal basso, si può notare che i certificati vengono emessi dagli intermediari eletti da *LE*. Ognuno di essi, risalendo la catena, si riferisce a *IdenTrust* e *ISGR*, è possibile distinguere i due diversi certificati di root semplicemente controllando il campo *Issuer*.

[3] Internet Security Research Group.

[4] Automatic Certificate Management Environment.

[5] Come annunciato in questo link [8], il riconoscimento ufficiale dovrebbe avvenire a Luglio 2020.

[6] IdenTrust è una *certification authority* riconosciuta dalla maggior parte dei browser, essa utilizza il proprio *DST Root CA X3* per firmare i certificati emessi.

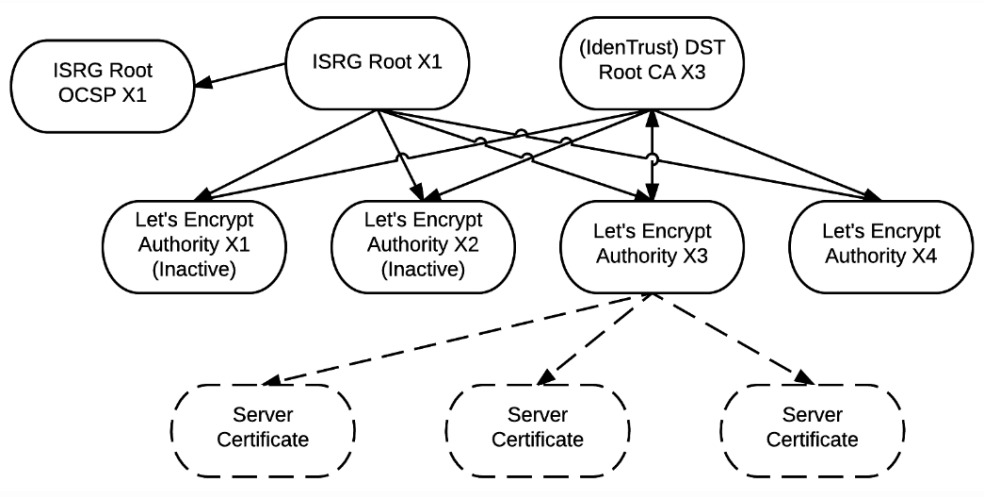

Figura 2.1. Let's Encrypt certificate's chain

2.2.1 Generalità

L'idea di fornire un servizio nuovo e diverso dalla norma nasce nel 2012 e vede la collaborazione tra quattro importanti istituzioni: *Eletronic Frontier Foundation* [7], *Mozilla Foundation* [8], *Università del Michigan* e infine l'*ISRG* che supporta e segue gli sviluppi del progetto *LE*.

Il primo annuncio al pubblico avviene nel 2014, ma solo agli inizi del 2015 il protocollo *ACME* viene consegnato all'*IETF* [9] al fine di essere standardizzato. Nello stesso anno viene firmata la collaborazione tra l'*ISRG* e la *Linux Foundation*. La data di rilascio del servizio al pubblico viene fissata per la fine del 2015. Prima di arrivare a questo importante step, l'*ISRG* richiede a quattro grandi aziende (Apple, Google, Mozilla e Microsoft) l'aggiornamento dei loro *trusted root store* così da rendere i certificati rilasciati da *LE* riconosciuti dai più importanti browser. Poco dopo tutti i certificati emessi dagli intermediari vengono controfirmati da *IdenTrust* confermandone l'affidabilità. A dicembre del 2015 *LE* entra nel mercato dei certificati con una versione beta del servizio e nei primi mesi del 2016 ha già rilasciato quasi due milioni di certificati per il doppio dei siti web. Sullo stesso sito ufficiale di *LE* è presente un grafico che analizza la rapida crescita [7].

Il servizio è a tutti gli effetti funzionante e performante, ma ancora in via di miglioramento. Del protocollo *ACME*, utilizzato da *LE* e di cui si discuterà nel capitolo successivo, è stato mantenuto per molto tempo un *Internet Draft* strutturato esattamente come un documento *RFC*[10]. Dalle ultime versioni dell'*Internet Draft*, esso risultava già pienamente conforme con gli attuali *Internet Best Current Practices BCP* 78[11] e *BCP* 79[12]. Da Marzo 2019 il protocollo è stato standardizzato e ufficializzato da *IETF*, diventando RFC-8555.

[7] Organizzazione no profit specializzata nella tutela dei diritti in ambito digitale.

[8] Organizzazione no profit che si occupa della gestione del progetto open source *Mozilla*.

[9] Internet Engineering Task Force

[10] Request for Comments.

[11] Rights Contributors Provide to the IETF Trust - Documento contenente le specifiche IETF sui diritti dei contributi a IETF, ovvero le politiche di tutela per collaboratori in pubblicazioni di Internet-Draft o RFC o in qualsiasi attività riguardante IETF.

[12] Intellectual Property Rights in IETF Technology - Documento contenente le specifiche IETF sui diritti di proprietà intellettuale (come i diritti di brevetto) per tutti i gruppi di lavoro che elaborano tecnologie all'interno di IETF.

2.2.2 Richiedere un certificato

Il processo di richiesta di un certificato e il suo successivo mantenimento prevede diverse interazioni tra il software installato sul *server* e la *certification authority*. Queste interazioni avvengono tutte utilizzando il protocollo *HTTPS*, quindi mediante canale sicuro, e possono essere suddivise in due fasi principali, la prima riguarda la validazione del dominio e la seconda comprende l'emissione e il mantenimento. In aggiunta a queste due fasi, una tantum, vi è anche una fase di registrazione presso la *certification authority* al fine di creare un account che verrà utilizzato per identificare l'utente in tutte le fasi successive.

Creazione di un account

Questa fase è presente solo una volta nel flusso di richiesta o revoca di un certificato. Il software genera una coppia di chiavi (pubblica e privata) utilizzando uno degli algoritmi supportati dalla *certification authority*. Esse vengono associate all'account registrato presso *LE* e verranno utilizzate per identificare in maniera univoca il richiedente. La chiave privata verrà poi usata per firmare tutte le richieste inviate a *LE*, mentre la chiave pubblica corrispondente verrà utilizzata dalla *certification authority* per validare le richieste ricevute. All'account creato è possibile associare uno o più domini, tutti quelli per cui viene effettuata una richiesta di emissione di un certificato.

Validazione del dominio

Un *client* che richiede l'emissione di un certificato, deve dare prova alla *certification authority* di controllo sul dominio, o domini, per cui richiede il certificato. Il primo step è quello di fare richiesta di certificato per uno specifico dominio. La *certification authority* richiede a questo punto di risolvere una sfida asimmetrica e di salvare il risultato in uno specifico *path* all'interno del dominio per il quale è stata fatta richiesta di emissione del certificato. Un esempio del flusso in figura 2.2 dove un *web server* richiede un certificato per un dominio e *LE* risponde con la sfida.

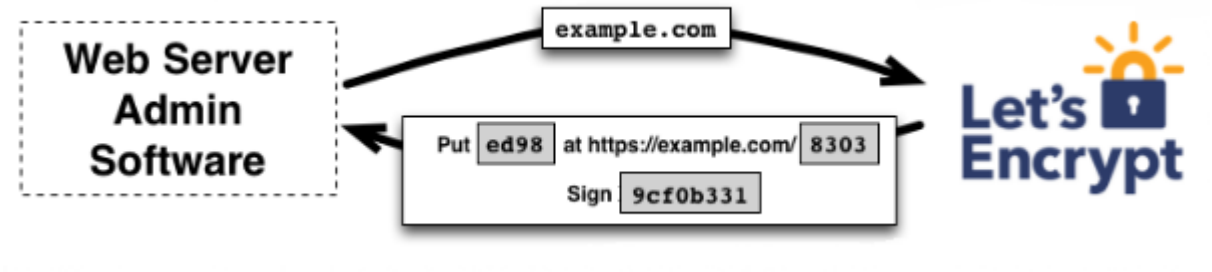

Figura 2.2. Certificate request

Quando il software risolve la sfida, quindi la risposta è disponibile nel percorso specificato (step 1 in figura 2.3), deve notificare la *certification authority* per far partire il processo di validazione (step 2 in figura 2.3). Quest'ultima scarica il file presente sul dominio del richiedente e verifica che il risultato sia quello atteso (step 3 e 4 in figura 2.3), se tutto va a buon fine si ha la certezza del controllo sul dominio e si può emettere il certificato (step 5 in figura 2.3).

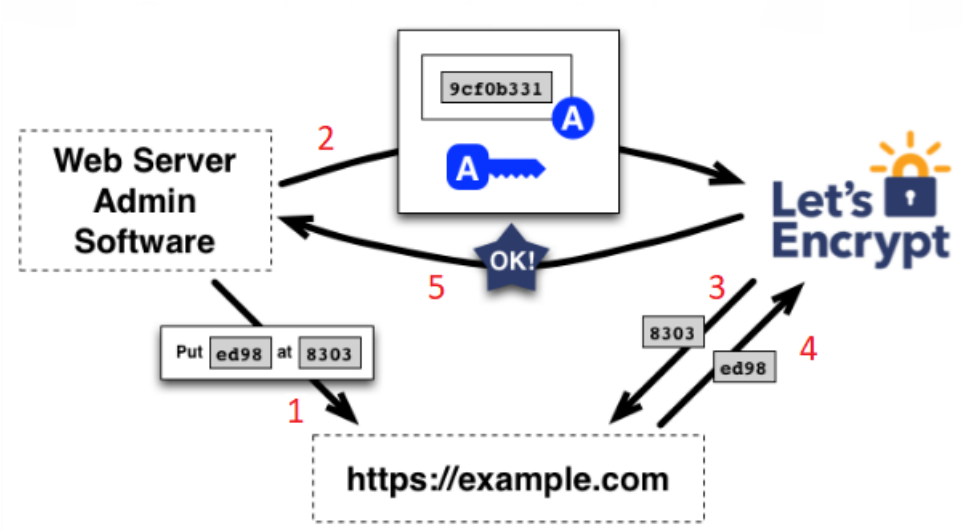

Figura 2.3. Certificate request validation

LE permette esclusivamente la validazione del dominio, non è stata ancora implementata la possibilità di effettuare validazioni estese e di organizzazioni, ovvero sono escluse le categorie per cui è richiesta la validazione della reale identità del richiedente.

Emissione e revoca di un certificato

A questo punto il software è abilitato a mandare qualsiasi richiesta, quindi di emissione, rinnovo o revoca di un certificato per il dominio validato. Queste operazioni avvengono mediante l'utilizzo di *PKCS#10 Certificate Signing Request* che vengono firmate con la chiave privata associata all'account registrato presso la *certification authority*. L'elaborazione della richiesta è subordinata ad opportune verifiche, da parte della *certification authority*, in merito al match con la chiave pubblica e alla correttezza del formato. In caso di esito positivo, la *certification authority* notifica il software che è abilitato a scaricare il certificato che dovrà poi installare sul *server*. Questo meccanismo si può notare in figura 2.4, nella parte superiore dell'immagine è presente la richiesta di emissione del certificato. Il software genera una nuova coppia di chiavi (pubblica e privata), che la *certification authority* dovrà associare al certificato emesso, e firmerà la richiesta con la chiave pubblica associata all'account registrato. La *certification authority* risponderà con il certificato emesso e quindi firmato con la propria chiave pubblica.

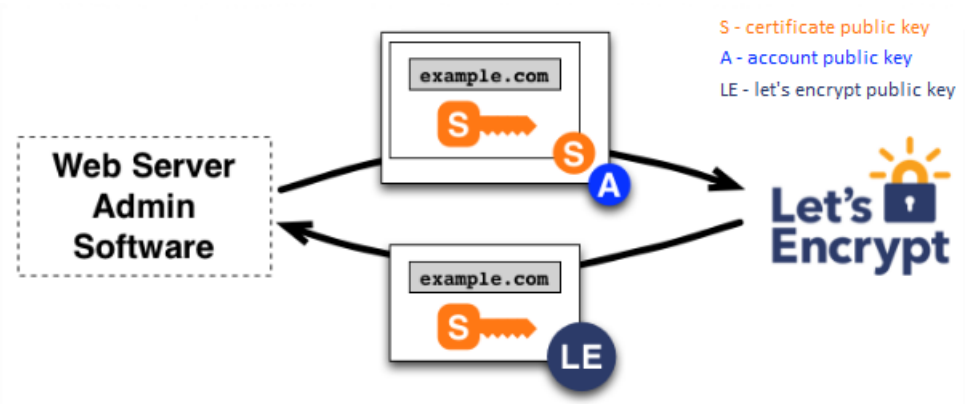

Figura 2.4. Certificate issuance

Il processo previsto per la richiesta di rinnovo e di revoca di un certificato è il medesimo. In entrambi i casi viene creata una *Certificate Signing Request* e si attende che la la *certification authority* faccia gli opportuni controlli di validità. Nel caso di richiesta di revoca di un certificato, la *certification authority* dovrà inoltre procedere con l'aggiornamento della *CRL* [13] e/o dell'*OCSP* [14], così da rendere effettiva la non validità del certificato relativo.

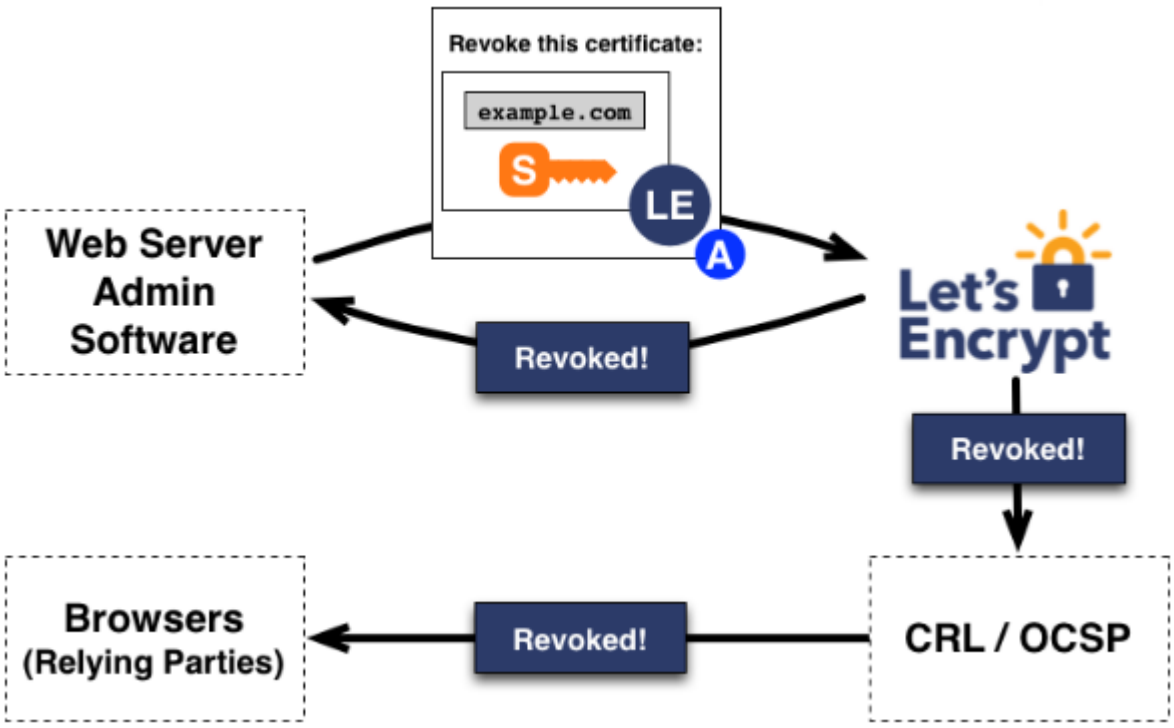

Figura 2.5. Certificate revocation

[13] Certificate Revocation List.

[14] Online Certificate Status Protocol.

2.2.3 Limiti

Per assicurare un corretto utilizzo ed efficiente al maggior numero di utenti possibile, *LE* impone dei limiti sulla frequenza delle richieste effettuate.

- creazione di un account: *LE* permette di creare un massimo di dieci account dallo stesso indirizzo IP in 3 ore;
- autorizzazioni pendenti per account: ad ogni account viene data la possibilità di avere al massimo 300 autorizzazioni pendenti, ma è un limite che difficilmente viene raggiunto. Questo problema può essere facilmente fronteggiato facendo un tentativo di validazione per quegli oggetti che risultano in stato pendente. Non importa quale sia il risultato ritornato dalla *certification authority*, la richiesta di autorizzazione non sarà più in stato pendente.
- emissione di un certificato: è il limite più importante che viene imposto. Il numero di certificati che possono essere richiesti per ogni dominio registrato è settato a cinquanta a settimana. Ad ogni dominio possono però essere associati un numero massimo di cento sotto domini per cui lo stesso certificato verrà ritenuto valido;
- validazioni fallite: ogni ora, ogni dominio di ogni account ha la possibilità di fallire 5 richieste di validazione;
- rinnovo di un certificato: le richieste di rinnovo di un certificato vengono gestite in maniera separata da quelle di emissione così da evitare un blocco qualora il numero massimo di richieste settimanali sia stato raggiunto. Il rinnovo di un certificato equivale all'emissione di un nuovo certificato relativo però allo stesso insieme di *hostname*, questo verrà considerato un duplicato del precedente. In questo caso il limite massimo di richieste accettabili è di cinque a settimana.

Tutti questi limiti non vengono resettati alla revoca di un qualsiasi certificato associato ad un dominio poiché la risorsa è stata comunque consumata.

Capitolo 3

Il Protocollo ACME

Come già specificato in precedenza, il processo che permette la richiesta e la gestione di un certificato è interamente automatizzato grazie all'utilizzo di un software (*Certbot*) e del protocollo *ACME*[1]. Esso è stato progettato dall'*ISRG* proprio per fornire un servizio ad hoc alla nascente *certification authority*. Di questo protocollo, che è di tipo *challenge-response*, sono state implementate due versioni, una utilizzata dalla *certification authority* e l'altra utilizzata dal software installato sul *server*. Lo scopo ultimo dell'utilizzare questo protocollo e i due software implementati appositamente per *LE* è quello di limitare al minimo le interazioni dell'utente.

Lo scopo principale di questo protocollo riguarda la validazione dei domini al fine di emettere un valido certificato, ma il team coinvolto nell'evoluzione di questo protocollo, sta lavorando anche su altre funzionalità. Esempi sono il rilascio di certificati associati ad indirizzi IP, certificati STIR[2] per associazione a numeri telefonici, e ancora la possibilità di fare *Extended Validation*. Non meno importante, il team si sta occupando di aggiungere funzionalità per permettere l'integrazione parziale con account non registrati utilizzando il protocollo *ACME*.

3.1 Panoramica sul Protocollo

Il protocollo *ACME* è strutturato come un'applicazione *Rest*, utilizza il protocollo *HTTPS* e richiede che tutti i messaggi scambiati tra il software installato sul *web server* (che chiameremo *client*) e la *certification authority* (che chiameremo *server*), siano oggetti *JWS* [3] inviati in HTTPS. La scelta di accoppiare il protocollo HTTPS agli oggetti *JWS* deriva dalla possibilità di aumentare la sicurezza applicata ai messaggi inviati dal *client*. Il protocollo HTTPS fornisce autenticazione e riservatezza per tutte le comunicazioni che provengono dal *server*, mentre l'utilizzo dell'oggetto *JWS* permette di autenticare il payload delle richieste provenienti dal *client*, offrendo inoltre protezione da attacchi replay e integrità degli url delle richieste.

Ogni richiesta effettuata dal *client* corrisponde ad una chiamata *Rest*, il cui contenuto varierà a seconda della risorsa alla quale si vuole accedere. Il processo di richiesta ed emissione di un certificato consta di specifici step ad ognuno dei quali corrisponde una o più chiamate al *server*. In figura 3.1 un riassunto delle relazioni tra le varie risorse offerte dal *server* e che verranno analizzati nel capitolo successivo.

[1] Automatic Certificate Management Environment.

[2] Secure Telephone Identity Revisited

[3] JSON Web Signature

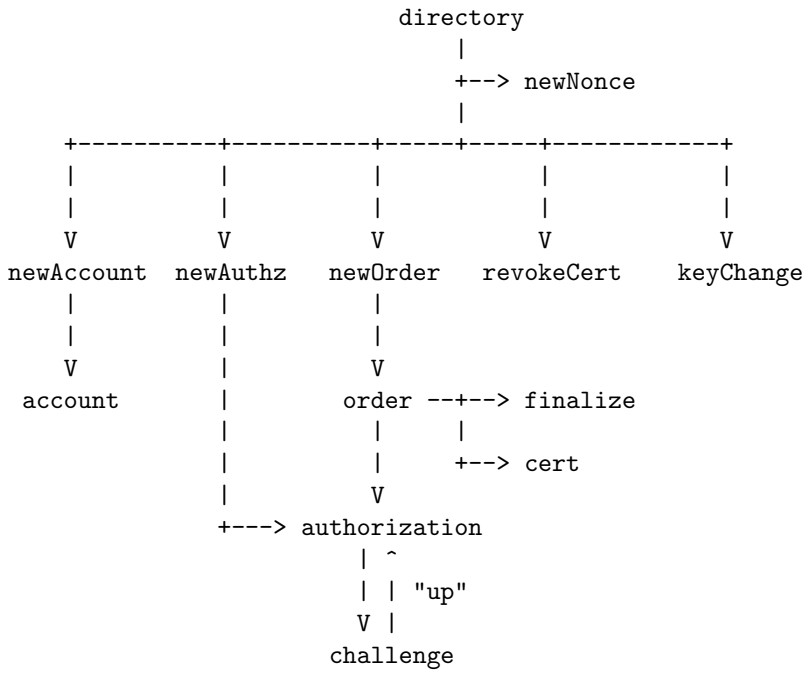

Figura 3.1. ACME Resources and Relationships

3.2 Json Web Signature

Come definito nell'RFC-7515, un oggetto *JWS*[4] rappresenta un contenuto, basato sulla struttura dati di un *JSON*[5], messo in sicurezza da una firma digitale o da un MAC (message authentication code) al fine di fornire integrità.

L'oggetto *JWS* consta di tre componenti

- *JOSE* [6] *Header* - composto dalla parte protetta e dalla parte non protetta dell'*header* dell'oggetto. L'*header* è una sorta di dizionario, quindi una lista di coppie "chiave-valore", nel quale una chiave deve essere univoca. La differenza tra la parte protetta e quella non protetta, come si evince dall'aggettivo stesso, è che la prima parte contiene informazioni sensibili e deve quindi essere protetta da firma e/o operazioni di MAC, mentre per la seconda parte sono sufficienti operazioni di serializzazione. Va notato che un *header* malformato comporterà una cattiva deserializzazione e il conseguente rifiuto della richiesta.
- *JWS Payload* - contiene l'effettivo corpo della richiesta che, come gli altri componenti dell'oggetto, verrà codificato utilizzando la codifica *Base64*;
- *JWS Signature* - contiene la firma e/o MAC applicato al contenuto e all'*header*.

Sono due i metodi di serializzazione possibili per questa tipologia di oggetti, entrambi condividono la stessa modalità di codifica. La codifica *Base64* per URL e file name, di cui è possibile apprendere i dettagli nell'RFC-4648, permette il parsing di caratteri speciali (quali i simboli "~"

[4] Json Web Signature.

[5] JavaScript Object Notation.

[6] *JSON Object Signing and Encryption*

e "." e una gestione differente del carattere "=" di padding. Utilizza dunque un set di caratteri sicuro per la codifica di messaggi.

Una volta codificato l'oggeto *JWS*, viene serializzato. Esistono due metodi definiti nell'RFC-7515

- *Compact Serialization* - rappresentazione compatta e sicura, il cui valore è dato dalla concatenazione del risultato logico della codifica *Base64* applicata ai tre componenti dell'oggetto *JWS*;
- *Json Serialization* - esistono due versioni di questo metodo di serializzazione, uno che permette di applicare una o più firme e/o MAC e una che permette di applicarne una ed una sola. Quest'ultima, denominata *Flattened*, è quella effettivamente utilizzata all'interno del protocollo *ACME*.

3.2.1 Creazione di un oggetto JWS

Nella creazione di un oggetto *JWS* vanno seguiti passi specifici il cui scopo è quello di creare un oggetto ben formato, esso garantirà che all'atto della deserializzazione esso verrà riconosciuto come valido e possa dunque essere preso in considerazione per l'elaborazione della richiesta effettuata.

- Creazione del *payload*;
- Codifica del *payload* utilizzando la codifica *Base64URL*;
- Creazione dell'*header*;
- Codifica dell'*header* utilizzando la codifica *Base64URL*;
- Calcolo della *JWS Signature* utilizzando l'algoritmo specificato nel campo "alg" dell' *header*;
- Codifica della *JWS Signature* utilizzando la codifica *Base64URL*;
- Serializzazione dell'oggetto utilizzando il metodo prescelto.

In fase di deserializzazione, il ricevente dovrà considerare valido il messaggio se corretto alla fine dei seguenti passi

- Deserializzazione dell'oggetto utilizzando lo stesso metodo utilizzato in fase di serializzazione;
- Verifica che l'oggetto sia quello effettivamente atteso, quindi composto da due *header* (uno protetto, l'altro non protetto), un *payload* e una *signature*;
- Verifica che i campi presenti nell'*header* siano comprensibili e validi;
- Decodifica della firma;
- Validazione della firma in base all'algoritmo specificato nell'*header*
- Decodifica del contenuto;

3.3 Configurazioni per le richieste HTTPS

Affinchè *client* e *server* comunichino senza problemi, e ciò implica anche rendere valide le richieste effettuate e le risposte ricevute, vanno definite delle regole standard nella creazione delle richieste HTTPS.

Un *server* dovrebbe seguire le raccomandazioni presenti nell'RFC-7525 per tutto il setup riguardante l'implementazione del TLS [7]. Un *server* che implementa il TLS 1.3 potrebbe permette ai *client* di effettuare richieste *0-RTT*[8], ovvero di inviare dati già nella prima fase dell'*handshake*. Questo risulta essere sufficientemente sicuro dal momento che il protocollo *ACME* è protetto per costruzione da attacchi di tipo *replay*. Inoltre i *server* che vogliono essere generalmente raggiungibili e accessibili anche da *browser-based client*, dovrebbero utilizzare il *Cross-Origin Resource Sharing*, quindi non bloccare informazioni ricevute e/o inviate su diverse origini.

Un *client* deve passare tra i campi dell'*header* un campo denominato *User-Agent*, nel quale viene specificata la versione del protocollo HTTP e in aggiunta il nome e la versione del protocollo *ACME*. Maggiori dettagli sono specificati nell'RFC-7231. Un altro *header* consigliato vivamente ai *client* è *Accept-Language* (RFC-7231) per favorire la localizzazione dei vari messaggi restituiti dal *server*. Entrambi questi *header* non sono gestiti dal *server*. Il primo viene utilizzato per scopi statistici dagli sviluppatori stessi e per entrambi non esiste una gestione di default da parte del *client*. Si dovrebbe procede con una modifica al codice dello stesso *client* per abilitarli e gestirli correttamente.

Qualsiasi richiesta che abbia un *body* non vuoto viene considerata valida solo se il *payload* è correttamente protetto da firma. La prima azione effettuata alla ricezione di una richiesta è dunque verificarne la correttezza, solo in caso di esito positivo essa verrà successivamente processata. Una richiesta viene considerata valida se l'oggetto *JWS* rispetta anche i seguenti criteri:

- L'oggetto è stato serializzato utilizzando la *Json Flattened Serialization* e questo per definizione della serializzazione *flattened* implica che l'oggetto non può essere protetto da più di una firma;
- L'*unencoded payload option* e l'*unprotected header* non possono essere utilizzati;
- i seguenti campi devono essere contenuti nel *protected header*
  - ALG (Algorithm) - questo contenuto deve riferirsi chiaramente all'algoritmo di cifratura utilizzato e correttamente registrato (es: MAC/HMAC). Se il *server* riceve una richiesta con un algoritmo non supportato, risponderà con un errore (*BadSignatureAlgorithm*) e una lista di algoritmi per lui validi. Uno degli algoritmi che il *server* deve implementare è *ES256*[9], e dovrebbe inoltre permettere l'utilizzo dell'algoritmo *EdDSA*[10] supportando la variante Ed25519.
  - NONCE - utilizzato per la protezione da attacchi di tipo *Replay*. Il *server* mantiene una lista di NONCE generati e ne crea uno nuovo ad ogni risposta poiché deve inserirlo nel campo *HTTP Replay-Nonce* dell'*header*. Il *client* utilizza ogni nuovo NONCE per le successive richieste. Il *server* effettua delle verifiche su questo campo dell'*header* della richiesta e rifiuta la richiesta con *BadNonce* qualora non dovesse riuscire a trovare una corrispondenza con quanto inviato in precedenza;
  - URL - specifica l'url della risorsa che si vuole accedere. Tutte le risorse messe a disposizione dal *server*, sono accessibili mediante specifico URL. Criptare lo stesso nell'*header* della richiesta ne garantisce l'integrità. Questo è necessario poiché le richieste, nel tragitto dal *client* al *server*, possono passare attravero livelli di *request routing*, i quali possono cambiare il contenuto della parte non firmata della richiesta. Codificando l'URL nell'*header* della richiesta, esso non potrà essere alterato e il *server* potrà effettuare un controllo di integrità ed eventualmente rifiutare la richiesta come non autorizzata;
  - JWK (JSON Web Key) o KID (Key Id) - questi due campi sono mutualmente esclusivi, la loro presenza contemporaneamente comporta un rifiuto della richiesta. Il primo viene

[7] Transport Layer Security

[8] Zero Round Trip Time

[9] È un algoritmo di cifratura a chiave asimmetrica che sfrutta un algoritmo di firma digitale a curva ellittica (ECDSA) con P-256 e funzione di hash SHA-256.

[10] È un algoritmo di firma digitare a curva di Edwards.

utilizzato per le richieste di creazione di nuovi account e per la revoca dei certificati. Esso conterrà la chiave pubblica corrispondente alla chiave privata utilizzata per firmare la richiesta. Mentre il secondo viene utilizzato in tutte le altre richieste e contiene l'url dell'account di riferimento. Se il *server* non supporta la chiave pubblica specificata nel campo JWK, risponderà con un errore (*BadPublicKey*) e deve specificare il motivo per cui l'algoritmo è valido, ma non va bene contestualmente alla chiave generata (es: "alg" è "RS256"[11] ma il modulo "n" è troppo piccolo).

Dal momento che tutte le richieste effettuate verso il *server* utilizzano un oggetto *JWS*, ci sono due aspetti importanti da tenere in considerazione. Come prima cosa, il *content-type* nell'*header* detta richiesta deve essere uguale a *application/jose+json*. Questo è dovuto al fatto che si sta utilizzando un oggetto il cui *header* è di tipo *JOSE*, come specificato nella sezione 3.2. Il secondo aspetto, molto più importante, è che non è possibile effettuare richieste *GET* al *server*, ad eccezione di quelle fatte per l'accesso alle funzionalità e quelle per la generazione di un nuovo *NONCE*. Questo deriva dal fatto che il *server* si aspetta di validare un oggetto *JWS* che nel caso di richieste *GET* non sarebbe presente. Conseguentemente, lo stardard richiede che tutte le *GET* siano in realtà effettuate utilizzando richieste *POST*, settando il contenuto del *body* a stringa vuota. In tal modo, si mantiene la struttura dell'oggetto *JWS* atteso dal *server* e la richiesta non verrà rigettata come malformata.

3.4 Validazione degli account

Come si evince dalla figura 3.1, la prima comunicazione che avviene tra il *client* e il *server* riguarda la creazione di account che verrà utilizzato per effettuare tutte le richieste di associazione della coppia di chiavi ad un certificato. La *certification authority* deve accertarsi che l'account registrato identifichi effettivamente il possessore del dominio per il quale si richiede l'emissione di un certificato e per farlo fornisce al *client* un insieme di sfide alle quali quest'ultimo deve rispondere. Sfide diverse permettono al *server* di verificare aspetti diversi del controllo posseduto dal richiedente e tutte sono formate come segue:

- Contenuto della sfida;
- Contenuto della risposta;
- Modalità di utilizzo del contenuto della sfida da parte del *server* e la risposta per verificare l'identità del richiedente.

Tutte fanno utilizzo di quella che viene definitiva come *key authorization string*, ovvero una stringa che è il risultato della concatenazione tra il token della sfida e la chiave, separati da un ".".

Affinchè il risultato del controllo sia positivo, il *client* deve inoltre effettuare dei settaggi di rete tali per cui il *server* possa riuscire ad accedere alla risposta alla sfida. È abbastanza comune che una risposta ad una sfida non venga considerata valida solo perchè il *server* non riesce ad accedervi correttamente. Per prevenire rifiuti ingiustificati viene impostato sul *server* un numero di volte e un time slot per cadenzare nuovi tentativi. In questo frangente la richiesta rimane nello stato *processing*, il *client* non è dunque in grado di fare niente in questa fase. Dal momento che i tentativi sono stati esauriti, lo stato diventa *invalid* e il *client* può provare ad effettuare nuovamente le richieste, anche in questo caso per un numero limitato di volte.

[11] È un algoritmo di cifratura a chiave asimmetrica che sfrutta RSA e la funzione di hash SHA-256.

3.4.1 Validazione HTTP

Attraverso questo tipo di validazione, il richiedente prova di avere un controllo sul dominio, per il quale sta effettuando la richiesta, grazie al fatto che riesce a piazzare risorse in uno specifico path all'interno del dominio. Il path viene specificato dal *server* e anche il suo contenuto, tutti seguono però uno standard, ovvero tutti hanno un prefisso del tipo */.wellknown/acme-challenge/* e sono seguiti dal token relativo alla sfida. A sua discrezione il *server* decide anche quale *host* presente tra i record A e AAA del DNS utilizzare. Questo perchè un dominio può essere risolto con più indirizzi *IPv4* o *IPv6*.

Dal momento in cui il *client* ha piazzato la sfida, risponde con un oggetto vuoto così da avvisare il *server* che la sfida può essere validata. A questo punto il *server* accederà alla risposta alla sfida e verificherà se il contenuto è quello atteso o meno.

3.4.2 Validazione DNS

In questo caso il *client* prova il controllo sul dominio grazie all'inserimento di un record DNS relativo al dominio da validare, dopo averne temporaneamente modificato il nome. Come prima cosa costruisce la chiave e ne calcola il digest utilizzando *SHA-256*. A questo punto modifica il dominio aggiungendo *_acme-challenge* davanti al dominio attuale e aggiunge il record TXT il cui contenuto sarà il digest appena calcolato. Ipotizzando che il dominio da validare sia "`www.example.org`", il *client* fornirà un record DNS composto da dominio, TTL[12], la tipologia di record[13] e infine il digest calcolato. Di seguito un esempio.

```
_acme-challenge.www.example.org. 300 IN TXT "gfj9Xq...Rg85nM"
```

Anche in questo caso il *client* deve notificare il *server* e ciò avviene rispondendo alla richiesta precedente con un oggetto vuoto. Il *server* può verificare l'effettiva esistenza del record DNS e quindi concedere o meno il certificato.

Una volta che la richiesta è stata ritenuta validata, il *client* dovrebbe eliminare la risposta alla sfida.

3.5 Considerazioni sulla sicurezza

Lo scopo del protocollo *ACME*, analizzato in questo lavoro di tesi, è quello di creare un'associazione tra un dominio e un certificato, garantendone l'integrità e l'autenticità.

Per evitare che la chiave associata ad un account registrato venga utilizzata in maniera impropria da parti non autorizzate, *ACME* utilizza due canali di comunicazione diversi, uno tramite il quale il *server* richiede validazioni aggiuntive sul controllo del dominio e l'altro per lo scambio delle richieste successive, come mostrato in figura 3.2.

[12] Time To Live - indicazione temporale sulla validità del record.

[13] Le possibili opzioni sono TXT (record di testo versatile e dinamico) o SRV (utilizzato per collegare due domini utilizzando una specifica porta).

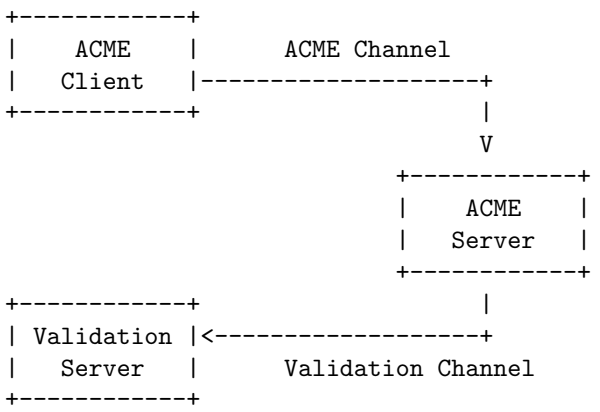

Figura 3.2. ACME Communication Channels

Il canale di validazione in figura 3.2 utilizza il protocollo di comunicazione *HTTP*, mentre il canale *ACME* utilizza *HTTPS*. Entrambi i canali sono dunque esposti a rischi, ma data la natura intrinseca dei protocolli utilizzati, essi saranno di differenti tipologie. Si deve dunque sempre tenere in considerazione che il canale *ACME* è esposto ad attacchi a livello di rete, ma anche ad attacchi di tipo MitM [14], e inoltre, non meno importante, potrebbe essere soggetto a problemi causati da un uso improprio del protocollo stesso. Fornire una protezione da attacchi ti tipo MitM, potrebbe creare un ambiente facilmente sfruttabile da CDN [15] e middlebox con funzione di TLS-MitM. Ad esempio, un tale utente malintenzionato potrebbe inviare una falsa risposta del tipo *badSignatureAlgorithm* al fine di far utilizzare al *client* un algoritmo di firma di qualità inferiore rispetto a quello supportato dal *server*. Un MitM presente su tutte le connessioni (come nel caso di una CDN) può causare condizioni di DoS [16] in vari modi.

L'integrità del processo di autorizzazione dipende dalle sfide utilizzate per la validazione dell'account. È indispensabile assicurarsi che la sfida venga completata dall'identificatore corretto. Si potrebbe infatti verificare un attacco di tipo MitM in cui la coppia di chiavi del legittimo richiedente venga sostituita con un'altra scelta dall'attaccante. Grazie ai processi di validazione utilizzati da *LE*, gli effetti di questo attacco sarebbero irrisori in quanto la validazione avverrebbe con la chiave dell'attaccante e non con quella dell'utente originale. L'esito della richiesta sfocerebbe nella creazione di un nuovo account e l'unico svantaggio per l'utente originale sarebbe quello di non riceve risposta dalla *certification authority*. Un altro rischio per la validazione degli account potrebbe essere l'utilizzo di *host provider* esterni, in questo caso non ci sarebbe nessun controllo sul fatto che terzi possano manomettere le informazioni. Esistono però contromisure che il *server* può mettere in atto per mitigare gli effetti di queste intromissioni. Il *server* può utilizzare una validazione *DNSSEC*, aggiungere randomicità agli ID delle richieste o utilizzare solo il TCP, e interrogare il DNS da diversi punti della rete. Quest'ultima opzione rende inoltre più difficile sfruttare le vulnerabità dei protocolli di routing. Questo tipo di attacco è spesso localizzato o legato alla posizione dell'attaccante. Se il *server* interroga il DNS e instaura connessioni HTTP da più punti della rete, rende più difficile minare la validazione *ACME* in quanto anche l'attacco dovrebbe arrivare da più punti della rete. Alla luce di queste considerazioni, il processo di convalida utilizzato dal protocollo *ACME* offre un livello di protezione tale da rendere impossibile un attacco che permetta di autorizzare una chiave scelta dell'attaccante.

È necessario fornire protezione anche da possibili attacchi di tipo DoS, in quanto l'utilizzo del protocollo *ACME* richiede che il *server* esegua operazioni dispendiose a livello applicativo. Ne sono esempio le operazioni di convalida che richiedono al *server* di effettuare connessioni verso

[14] Man in the Middle.

[15] Content Distribution Networks

[16] Denial of Service.

altri *server* che potrebbero essere sotto il controllo di un attaccante, o le emissioni dei certificati che possono richiedere al *server* l'interazione con hardware crittografici. Tutti questi attacchi possono essere mitigati applicando delle limitazioni

- porre un limite alle richieste HTTP;
- porre un limite alla frequenza con cui un account registrato può richiedere dei certificati.

Per impedire agli attaccanti di aggirare questi limiti semplicemente creando nuovi account, i *server* dovrebbero limitare la frequenza con la quale è possibile effettuare nuove registrazioni.

Un altro possibile attacco che potrebbe verificarsi è il *SSRF* [17]. Un utente malintenzionato potrebbe far in modo che un *server* vittima invii delle richieste ad un determinato *URL* scelto dall'attaccante. La tipologia di attacco appena descritta potrebbe verificarsi durante il processo di convalida della sfida, ovvero quando il *server* deve effettuare una *GET* all'*URL* del dominio per il quale si effettua richiesta di certificato. Gli effetti di questo attacco solo limitati dal fatto che l'attaccante può controllare solo l'*URL*, ma non il dominio vero e proprio. Tuttavia, se l'attaccante per prima cosa imposta il reindirizzamento su un dominio che è sotto il suo controllo, ciò farà sì che il *server* interroghi un *URL* arbitrario. Pertanto per limitare ulteriormente il rischio *SSRF*, i *server* dovrebbero garantire che le *GET* di convalida possano essere inviate solo ai *server* sulla Internet pubblica e non, ad esempio, all'interno della rete interna del richiedente.

[17] Server-Side Request Forgery.

Capitolo 4

Analisi dei processi di comunicazione tra *client* e *server*

Come già descritto nel capitolo precedente, un *server ACME* mette a disposizione una serie di risorse alle quali è possibile accedere mediante apposito link. Di seguito una lista delle risorse che verranno analizzate in questo capitolo

- *Creazione di un account* - permette la gestione delle informazioni riguardanti un account. Questa risorsa viene acceduta una volta sola, l'account creato sarà utilizzato per tutte le richieste successive effettuate alla *certification authority*;
- *Richiesta di un certificato* - tramite l'account creato utilizzando la risorsa precedente, un *client* può effettuare richiesta di certificato per un determinato dominio;
- *Autorizzazione dell'identificatore* - in seguito alla richiesta effettuata utilizzando la risorsa precedente, il *client* deve dare prova di poter agire per conto dell'identificatore per il quale si richiede l'emissione e la gestione di un certificato. Per identificatore si intenderà la coppia di informazioni riguardi il dominio, ovvero la tipologia (per esempio "dns") e il valore (il nome del dominio);
- *Emissione di un certificato* - in caso di controlli andati a buon fine ed emissione del certificato, attraverso questa risorsa è possibile effettuare il download per la successiva installazione;
- *Revoca di un certificato* - questa risorsa è acceduta per la richiesta di revoca di un certificato, insieme alla richiesta di registrazione di un account, è l'unica risorsa che non richiede l'account per l'accesso, questa richiesta dovrà essere firmata utilizzando la chiave privata associata al certificato.

Come mostrato in figura 3.1, le risorse sono suddivise gerarchicamente in *directory* e ogni risorsa è identificata da un *URL* che ne permette l'accesso diretto. A partire dalla risorsa principale, tutte le altre sono collegate tra di loro mediante link relazionali. Le risorse che sono sempre a disposizione del *client* sono:

- *directory* - per l'accesso a tutte le funzionalità offerte dal *server*. Viene inoltre riportato il *link* corrispondente in tutte le varie risorse così da avere un riferimento alla *root*;
- *new nonce* - per la generazione di un nuovo *nonce* da inserire nell'*header* delle richieste.

Il link *up* mostrato in figura 3.1 viene utilizzato dalle risorse riguardanti le sfide per creare un link tra la risorsa da autorizzare e la sfida a cui appartiene. È inoltre messo a disposizione del *client* per conoscere la catena di *certification authority* e risalire al convalidatore originale dei vari certificati.

In tabella 4.1 sono raffigurate le interazioni tra *client* e *server*, dalla fase di registrazione a quella di download di un certificato.

Action	Request	Response
Get directory	GET directory	200
Get nonce	HEAD newNonce	200
Create account	POST newAccount	201 → account
Submit order	POST newOrder	201 → order
Fetch challenges	POST-as-GET order's authorization urls	200
Respond to challenges	POST authorization challenge urls	200
Poll for status	POST-as-GET order	200
Finalize order	POST order's finalize url	200
Poll for status	POST-as-GET order	200
Download certificate	POST-as-GET order's certificate url	200

Tabella 4.1. Client-Server sequence of requests.

Nei paragrafi successivi verranno analizzati i processi di comunicazione tra un *client* e un *server* utilizzando quelle che sono le specifiche definite nel draft del protocollo *ACME* rilasciato il 28 Febbraio 2019. Lo scambio di messaggi e le valutazioni saranno estratte sulla base dei risultati ottenuti applicando le tecnologie ai *server Apache* e *Nginx*. Entrambi i *server* sono stati installati su macchine virtuali *Linux Ubuntu Server LTS 16.04*. Al fine di valutare i risultati dei test in maniera trasparente, effettuando richieste per domini diversi a partire da indirizzi IP pubblici diversi, si è scelto di utilizzare le macchine virtuali esposte dal servizio *Azure* di Microsoft. I risultati che verranno mostrati nelle sezioni successive, saranno estratti dai file di log che il software *Certbot* mantiene durante le fasi di comunicazione con la *certification authority*. In caso di uguaglianza nei risultati, verranno riportati solo gli oggetti ricavati dal log relativo al *server Apache*, in presenza di differenze, verranno mostrati entrambe le versioni.

4.1 Oggetti utilizzati

Prima di procedere con l'analisi delle singole chiamate, definiamo quelle che sono le proprietà degli oggetti utilizzati. Alcune di queste saranno obbligatorie al fine di rendere le richieste valide, altre saranno semplicemente opzionali.

4.1.1 Oggetto account

Utilizzato per le interazioni riguardanti la creazione e la successiva gestione degli account. Di seguito una lista di proprietà appartenenti a questo oggetto

- *status* - una stringa obbligatoria indicante lo stato attuale dell'account. I possibili valori sono i seguenti
  - *valid* - valore assegnato per *default* al momento della creazione poiché non ci sono verifiche particolari da effettuare in questa fase;
  - *deactivated* - utilizzato nella richiesta del *client* che intende disattivare un account;
  - *revoked* - utilizzato dal *server* per indicare la non validità dell'account.
- *orders* - una stringa obbligatoria contenente un *URL* tramite il quale si può accedere agli ordini relativi all'account. Effettuando una chiamata *POST-as-GET* al link indicato, si ottiene un lista delle richieste dalle quali vengono escluse le stesse ritenute invalide;
- *contact* - una lista di stringhe opzionali che dovrebbe contenere informazioni utili al *server* per poter contattare il richiedente;
- *termsOfServiceAgreed* - un booleano opzionale. Se presente e settato a *true* in fase di registrazione, l'utente accetta i termini del servizio;
- *externalAccountBinding* - un oggetto JWS opzionale che contiene informazioni riguardo account esterni ad *ACME* e che l'utente può associare direttamente in fase di registrazione.

4.1.2 Oggetto ordine

Utilizzato sia per effettuare richiesta di emissione di un nuovo certificato, e ciò comporta che debbano essere fornite più informazioni, sia per tracciare lo stato di un ordine già richiesto.

- *status* - una stringa obbligatoria con i seguenti possibili valori:
  - *pending* - impostato inizialmente all'atto della creazione dell'account;
  - *ready* - quando tutte le autorizzazioni presenti nella richiesta sono in stato *valid*, la richiesta di ordine passa in questo stato in quanto pronta ad essere processata;
  - *processing* - utilizzando nella fase di analisi e verifica della richiesta da parte della *certification authority*;
  - *valid* - settato quando tutte le verifiche da parte della *certification authority* vanno a buon fine e si può procedere con l'emissione del certificato;
  - *invalid* - impostato se una delle verifiche non va a buon fine.
- *identifiers* - contiene obbligatoriamente una lista di coppie di valori che permettono alla *certification authority* di verificare l'identità del richiedente. Ogni oggetto è composto dal tipo di identificatore e dall'identificatore stesso. Una volta valorizzato questa proprietà, essa non potrà pièssere modificata;
- *authorizations* - una lista di stringe obbligatorie contenti l'*URL* tramite il quale si possono riprendere, utilizzando sempre chiamate del tipo *POST-as-GET*,le autorizzazioni necessarie per le richieste pendenti, scadute e già completate. Una volta valorizzato questa proprietà, essa non potrà pièssere modificata;
- *finalize* - stringa obbligatoria con l'*URL* riguardante la *CSR*[1] per poter finalizzare l'ordine non appena tutte le richieste sono state autorizzate. Il risultato sarà la corretta valorizzazione del campo *certificate* con l'*URL* per procedere al download dello stesso;
- *expires* - campo di tipo data, obbligatorio solo per richieste in stato *pending* o *valid*. Utilizzato dalla *certification authority* per definire un tempo limite oltre il quale la richiesta non può più essere considerata valida;
- *notBefore* - stringa opzionale contenente il valore da settare nella proprietà relativa all'interno del certificato;
- *notAfter* - stringa opzionale contenente il valore da settare nella proprietà relativa all'interno del certificato;
- *error* - valorizzato solo in caso di problemi durante l'elaborazione della richiesta con la relativa stringa d'errore;
- *certificate* - stringa opzionale con *URL* per l'accesso al certificato emesso.

4.1.3 Oggetto autorizzazione

Viene utilizzato dal *client* per fornire tutte le informazioni necessarie al *server* per poter effettuare tutte le validazioni al fine di autorizzare gli identificatori. L'unica tipologia ammessa per questi ultimi è il *DNS* codificato nella forma in cui comparirà poi nel certificato.

Di seguito una lista delle proprietà appartenenti a questo oggetto

- *identifiers* - un oggetto obbligatorio composto dalla coppia di valori *type* - contenente la tipologia di identificatore da autorizzare - e *value* - contenente l'identificativo vero e proprio;

[1] Certificate Signing Request.

- *status* - anche in questo caso di tratta di una stringa obbligatoria con i seguenti possibili valori
  - *pending* - stato di default con il quale viene creato l'oggetto;
  - *valid* - il passaggio a questo stato avviene se una delle sfida contenute nell'oggetto passa nello stato *valid*;
  - *invalid* - se in fase di autorizzazione ci fossero problemi o il *client* non dovesse riuscire a completare una sfida correttamente, lo stato di questo oggetto passerebbe a invalido;
  - *deactivated* - viene utilizzato in caso di esplicita richiesta dal parte del *client* in seguito a richiesta valida;
  - *expired* - settato in seguito a scadenza di autorizzazione. Come per lo stato precedente, la richiesta deve essere stata validata almeno una volta;
  - *revoked* - viene settata dal *server* in presenza di eventuali motivazioni di revoca e solo se la richiesta è stata precedentemente considerata valida.
- *challenges* - una lista di oggetti obbligatori, tramite i quali il *client* può provare la possessione dell'identificativo ed essere successivamente autorizzato. Conterrà le sfide per le richieste pendenti, quelle fallite e quelle portate a termine;
- *expires* - stringa opzionale contenente un *timestamp* oltre il quale la richiesta viene considerata non valida;
- *wildcard* - valore booleano che specifica se il *DNS* è di tipo *wildcard*[2].

4.1.4 Oggetto sfida

Per questi oggetti non esiste uno standard definito, poiché il suo contenuto dipende dal metodo di validazione che viene scelto. È comprensibile che venga utilizzato per poter fornire al *server* informazioni necessarie al fine di validare e autorizzare l'identificatore. Un campo ricorrente anche in questo caso è lo stato (*status*) il quale può avere i seguenti possibili valori

- *pending* - stato di default settato all'atto della creazione;
- *processing* - la richiesta passa a questo stato non appena il *client* risponde ad una sfida e ci rimane finchè il *server* non ha terminato con le validazioni dovute;
- *valid* - settato in maniera mutualmente esclusiva con lo stato successivo, solo in caso di validazione andata a buon fine;
- *invalid* - utilizzato in caso di validazione fallita.

[2]Un record del tipo *Wildcard DNS* permette l'accesso ad una porzione più ampia e contigua di un dominio. L'identificativo utilizzato è '*' e viene preposto al nome del dominio (e.g. *.example.com).

4.2 Directory

L'*URL* relativo a questa risorsa non presenta nessun vincolo di accesso ed è l'unico che il *client* deve conoscere per poter procedere con tutti gli accessi successivi. La risposta che verrà restituita dal *server* sarà un oggetto Json i cui campi conterranno informazioni utili per l'accesso alle risorse.

Nella tabella 4.2 una lista dei campi contenuti nel Json, di questi *newAuthz* verrà omesso qualora il *server* non supportasse la pre-autorizzazione che verrà analizzata nei paragrafi successivi.

Field	URL in Value
newNonce	New nonce
newAccount	New account
newOrder	New order
newAuthz	New authorization
revokeCert	Revoke certificate
keyChange	Key change

Tabella 4.2. Response fields from GET directory.

4.2.1 Specifiche del protocollo

La risposta ottenuta da un *client* in seguito ad una richiesta *GET*, del tipo `https://example.com/acme/directory`, per l'accesso al contenuto della *directory*, è mostrata in figura 4.1

```
HTTP/1.1 200 OK
Content-Type: application/json
{
     "newNonce": "https://example.com/acme/new-nonce",
     "newAccount": "https://example.com/acme/new-account",
     "newOrder": "https://example.com/acme/new-order",
     "newAuthz": "https://example.com/acme/new-authz",
     "revokeCert": "https://example.com/acme/revoke-cert",
     "keyChange": "https://example.com/acme/key-change",
     "meta": {
                "termsOfService": "https://example.com/acme/terms/2017-5-30",
                "website": "https://example.com/",
                "caaIdentities": ["exmple.com"],
                "externalAccountRequired": false
     }
}
```

Figura 4.1. Response from GET directory.

Il contenuto del campo *meta* è opzionale e lo stesso vale per i vari campi presenti al suo interno, viene utilizzato solo in caso sia necessario indicare informazioni aggiuntive su quelli che sono i servizi offerti dal *server*. Di seguito il dettaglio delle informazioni presenti in ognuno dei campi opzionali

- *termsOfService* - una stringa il cui contenuto sarà un *url* per l'accesso alle informazioni sugli attuali termini del servizio;
- *website* - una stringa contenente un *url* relativo ad un sito web che fornirà maggiori informazioni sul *server*;

- *caaIdentities* - un array di stringhe con tutti gli *hostname* relativi al *server* per la validazione dei record *CAA*[3].
- *externalAccountRequired* - un booleano che determina la possibilità per il *client* di associare un account esterno o meno.

4.2.2 Server Apache

Il risultato della *GET* effettuata a link https://acme-v02.api.letsencrypt.org/directory è il seguente:


```
{
  "keyChange": "https://acme-v02.api.letsencrypt.org/acme/key-change",
  "meta": {
    "caaIdentities": [
      "letsencrypt.org"
    ],
    "termsOfService":
        "https://letsencrypt.org/documents/LE-SA-v1.2-November-15-2017.pdf",
    "website": "https://letsencrypt.org"
  },
  "newAccount": "https://acme-v02.api.letsencrypt.org/acme/new-acct",
  "newNonce": "https://acme-v02.api.letsencrypt.org/acme/new-nonce",
  "newOrder": "https://acme-v02.api.letsencrypt.org/acme/new-order",
  "revokeCert": "https://acme-v02.api.letsencrypt.org/acme/revoke-cert",
  "zWln_9wivuY":
      "https://community.letsencrypt.org/t/adding-random-entries-to-the
    -directory/33417"
}
```


È subito possibile notare che rispetto allo standard atteso, nella risposta a questa *GET* sono assenti due campi - *newAuthz* e *externalAccountRequired*. Questi ultimi sono relativi alla possibilità di effettuare l'associazione con un account esterno. La loro assenza implica, che il *server* non espone questa funzionalià di cui i dettagli nella sezione 4.4.3.

4.3 Nonce

Per il meccanismo di protezione da attacchi *replay* implementato dal protocollo, ogni richiesta *POST* inviata dal *client* al *server*, deve contenere all'interno dell'*header*, un *nonce* valido. Solitamente questo valore viene preso dall'*header* della risposta alla precedente chiamata, ma a volte potrebbe essere necessario richiedere esplicitamente un nuovo valore da utilizzare per le chiamate.

4.3.1 Specifiche del protocollo

Come si può notare nella figura 4.2, è importante che il *server* specifichi un parametro per bloccare il *caching* della risposta[4], così da evitare che lo stesso *nonce* venga restituito più volte e si possa incorrere in errori del tipo *badNonce*. Questo risultato è ottenuto settando *no-store* nel campo *Cache-Control* dell'*header*.

[3] Certification Authority Authorization.

[4] Solitamente utilizzato nelle chiamate effettuate da un *client* o un *proxy* al fine di ridurre in numero delle stesse qualora si richiedesse l'accesso ad informazioni già ottenute.

```
HEAD /acme/new-nonce HTTP/1.1
Host: example.com

HTTP/1.1 200 OK
Replay-Nonce: oFvnlFP1hIhRlYS2jTaXbA
Cache-Control: no-store
Link: <https://example.com/acme/directory>;rel="index
```

Figura 4.2. Response from HEAD nonce.

4.3.2 Server Apache

Il risultato della richiesta *HEAD* effettuata a link https://acme-v02.api.letsencrypt.org/acme/new-nonce è il seguente:

```
HTTP 200
Server: nginx
Link: <https://acme-v02.api.letsencrypt.org/directory>;rel="index"
Replay-Nonce: 3ukb-WlNQQRZ8rv3PTcnjhyB_XKvUHR_KIm9u8sWEmU
X-Frame-Options: DENY
Strict-Transport-Security: max-age=604800
Content-Length: 0
Expires: Sun, 21 Jul 2019 17:34:28 GMT
Cache-Control: max-age=0, no-cache, no-store
Pragma: no-cache
Date: Sun, 21 Jul 2019 17:34:28 GMT
Connection: keep-alive
```

Il risultato atteso in questo caso viene perfettamente rispettato.

4.4 Gestione di un account

Per l'accesso a questa e alle risorse successive, il *client* deve effettuare una chiamata *POST* al *server*, utilizzando lo standard specificato nei capitoli precedenti.

Il *server* dovrà effettuare delle verifiche sulla validità dei campi specificati nella richiesta e anche del loro contenuto. In caso di anomalie dovrà considerare le richieste come mal formate e restituire un errore appropriato.

Lo stesso *URL* utilizzato per la registrazione di un nuovo account presso la *certification authority*, viene utilizzato per la completa gestione dello stesso.

Uno degli utilizzi è il semplice accesso alle informazioni presenti. Qualora dovesse essere effettuata una richiesta di registrazione per un nuovo account e la richiesta sia firmata con una chiave già registrata, il *server* non effettua una nuova registrazione, ma restituisce un oggetto di tipo *account* e l'*URL*. Lo stesso risultato è ottenuto effettuando la stessa chiamata ed esplicitando il campo *onlyReturnExisting* nel *payload* della richiesta.

L'aggiornamento delle informazioni avviene allo stesso modo, in caso di successo, il *server* restituirà 200.

4.4.1 Specifiche del protocollo - Creazione di un account

Nell'effettuare richiesta di creazione di un account, il *client* deve fare una chiamata *POST* richiedendo l'accesso ad una specifica risorsa (e.g. https://example.com/acme/new-accountHTTP/

1.1). Il corpo della chiamata deve essere un oggetto *JWS* con il *payload* costruito in maniera appropriata, come specificato in precedenza.

```
POST /acme/new-account HTTP/1.1
Host: example.com
Content-Type: application/jose+json
{
        "protected": base64url({
                "alg": "ES256",
                "jwk": {...},
                "nonce": "6S8IqOGY7eL2lsGoTZYifg",
                "url": "https://example.com/acme/new-account"
        }),
        "payload": base64url({
                "termsOfServiceAgreed": true,
                "contact": [
                        "mailto:cert-admin@example.org",
                        "mailto:admin@example.org"
                ]
        }),
        "signature": "RZPOnYoPs1PhjszF...-nh6X1qtOFPB519I"
}
```

Figura 4.3. POST-Request for new account creation.

Nel campo *contact*, come definito negli oggetti elencati, viene inserito un *URL* che il *server* può utilizzare per accedere alle informazioni di contatto dell'utente. Rientra nei compiti del *server* verificare che il contenuto sia supportato in base alle proprie implementazioni. Nello specifico, come definito nel documento [5], dovrebbe essere utilizzato lo schema *mailTo*[5] e qualora il *server* dovesse riscontrare contenuto non supportato, dovrebbe rigettare la richiesta come non valida. In questo caso il tipo di errore verrà valorizzato con *unsupportedContact*. Qualora invece la richiesta dovesse essere rifiutata per contenuto non valido, il tipo d'errore restituito sarà *invalidContact*.

La richiesta di creazione di un account può essere rifiutata per molteplici motivi, alcuni già elencati, un altro motivo riguarda i termini di accettazione del servizio. Se per il *server* è obbligatorio che il *client* accetti le condizioni e nell'oggetto di richiesta non è presente il campo *termsOfServiceAgreed* con valore a *true*, allora il *server* rifiuterà la richiesta di creazione di un nuovo account. È possibile che nel tempo i termini del contratto si modifichino, in occasioni di questo tipo, in caso di nuove richieste da parte del *client*, il *server* restituirà un errore che indichi l'impossibilità di accesso alla risorsa fino a che non vengano intraprese operazioni da parte dell'utente (e.g. *userActionRequired*). A questo punto il *client* deve accedere alla specifica risorsa che restituisce i termini del contratto aggiornati ai quali l'utente può dare il consenso. Anche questo processo può essere automatizzato, ma è altamente sconsigliato.

Qualora la richiesta dovesse invece essere processata e completata, il *server* salverà la chiave pubblica, associandola all'account, e manderà al *client* una risposta come quella della figura 4.4. Tale chiave verrà inoltre utilizzata per la validazione delle richieste successive.

[5]Definisce lo standard per l'accesso a risorse utilizzando mail, nella versione meno complessa, un *URI mailTo* contiene un indirizzo email.

```
HTTP/1.1 201 Created
Content-Type: application/json
Replay-Nonce: D8s4D2mLs8Vn-goWuPQeKA
Link: <https://example.com/acme/directory>;rel="index"
Location: https://example.com/acme/acct/evOfKhNU60wg
{
        "status": "valid",
        "contact": [
                "mailto:cert-admin@example.org",
                "mailto:admin@example.org"
        ],
        "orders": "https://example.com/acme/acct/evOfKhNU60wg/orders"
}
```

Figura 4.4. POST-Response for new account creation.

4.4.2 Server Apache - Creazione di un account

Durante la fase di richiesta di un account, il software *Certbot* richiede all'utente il consenso al trattamento dei dati e un indirizzo email utilizzabile dalla *certification authority* in caso voglia esplicitamente contattare il *client*. Queste informazioni formano il contenuto del payload che verrà codificato in *base64*.

Viene dunque effettuata una *POST* all'indirizzo https://acme-v02.api.letsencrypt.org/acme/new-acct e il *JWS* utilizzato è il seguente:

```
/*Richiesta codificata*/
{
  "protected":"eyJub25jZSI6ICIzdWtiLVdsTlFRUlo4cnYzUFRjbmpoeUJfWEt2VUhSX0tJbTl1
    OHNXRW1VIiwgInVybCI6ICJodHRwczovL2FjbWUtdjAyLmFwaS5sZXRzZW5jcnlwdC5vcmcvYWN
    tZS9uZXctYWNjdCIsICJqd2siOiB7ImUiOiAiQVFBQiIsICJrdHkiOiAiUlNBIiwgIm4iOiAicU
    MtSlV6S3ROWHJ1cWpSVTdkaXFHQlhXaE9OSUxibUVrNlpRUEFuSnVxbzY3Q2wyaUo1SnZ4RGotW
    TZBcERqOGNQRV9wdzJPY1lNaXROOW05N1k1d0JJZEJ0NHh3WTRDR3RZLVU4cXN2V0JGaXVsd2tL
    NTRTR24tU2pXcUk3QlE3bVBucUE1OC1QbEQyUjA5bmVfZmZZbjk4ZFlqQ2F4alZrYmViNEx6Vk5
    Od3RXWHBKS21LZWxxYWhsZFIxLUdjWXdnRVUzTVZIdlBCczhvaTJFVzhidE5EUy1tZmg4c3k3Sl
    hnQ1hnaFVBWV85SWZwRlpjUmttOFZJdE84TzZUb3NLRlEwWkIxZWpvTUNvckliV2c3d2Q4TkRQb
    klqaS1QR3ExWHRLS1A3VmM5ZnFIcOF3WUFzT1FoRmRfUFM3S19zWkJ1ZG1wczI4TXo1NOxzYTZm
    TVJ3InOsICJhbGciOiAiUlMyNTYifQ",
  "payload":"ewogICJOZXJtc09mU2VydmljZUFncmVlZCI6IHRydWUsIAogICJyZXNvdXJjZSI6IC
    JuZXctcmVnIiwgCiAgImNvbnRhY3QiOiBbCiAgICAibWFpbHRvOnJvYmVydGFfc2dyb2lAaG90b
    WFpbC5jb2OiCiAgXQp9",
  "signature":"D9jTfkI-ZuKn6ZUhHODfNVgrJuDmNnraTiZKE01u_ioVQrksHzL9XI4b_Eq6RRaZ
    fGPycXhr1Cw02TczfoXTzn6TDUKeEG3sXdNg9acWNsyLiHhshy39degFNAVyvmbuWYsm8sPd-MN
    kPPH-S7uPz4-ISezSvKmt5GkxKUc7QULnObAi2I5cxRZS-noRu93e4ZUW65n7MQKVtqsdQq8MvF
    ggaKcYNfZMjkp0qZTh2xE-oNCPsnYhoS_OOVrmjQYbPCYV2FGmkZd6NsHluXQBsyvCAuGQvZIXC
    gNolUJu-qVwNhm2so9JTL2HvpdvbVRLqTCisUDhT6_4gfKQv53uag"
}

/*Richiesta decodificata*/
{
  "protected":{
     "nonce":"3ukb-WlNQQRZ8rv3PTcnjhyB_XKvUHR_KIm9u8sWEmU",
     "url":"https://acme-v02.api.letsencrypt.org/acme/new-acct",
     "jwk":{
```

```
        "e":"AQAB",
        "kty":"RSA",
        "n":"qC-JUzKtNXruqjRU7diqGBXWhONILbmEk6ZQPAnJuqo67Cl2iJ5JvxDj-Y6ApDj8
          cPE_pw2OcYMitt9m97Y5wBIdBt4xwY4CGtY-U8qsvWBFiulwkK54SGn-SjWqI7BQ7mP
          nqA58-PlD2R09ne_ffYn98dYjCaxjVkbeb4LzVNNwtWXpJKmKelqahldR1-GcYwgEU3
          MVHvPBs8oi2EW8btNDS-mfh8sy7JXgCXghUAY_9IfpFZcRkm8VItO8O6TosKFQ0ZB1e
          joMCorIbWg7wd8NDPnIji-PGq1XtKKP7Vc9fqHsAwYAsOQhFd_PS7K_sZBudmps28Mz
          57Lsa6fMRw"
      },
      "alg":"RS256"
  },
  "payload":{
      "termsOfServiceAgreed": true,
      "resource": "new-reg",
      "contact": [
        "mailto:roberta_sgroi@hotmail.com"
      ]
  },
  "signature":"D9jTfkI-ZuKn6ZUhHODfNVgrJuDmNnraTiZKE01u_ioVQrksHzL9XI4b_Eq6RRaZ
    fGPycXhr1Cw02TczfoXTzn6TDUKeEG3sXdNg9acWNsyLiHhshy39degFNAVyvmbuWYsm8sPd-MN
    kPPH-S7uPz4-ISezSvKmt5GkxKUc7QULnObAi2I5cxRZS-noRu93e4ZUW65n7MQKVtqsdQq8MvF
    ggaKcYNfZMjkp0qZTh2xE-oNCPsnYhoS_OOVrmjQYbPCYV2FGmkZd6NsHluXQBsyvCAuGQvZIXC
    gNolUJu-qVwNhm2so9JTL2HvpdvbVRLqTCisUDhT6_4gfKQv53uag"
}
```


La risposta del *server* è stata dunque la seguente:


```
HTTP 201
Server: nginx
Content-Type: application/json
Content-Length: 586
Boulder-Requester: 61709049
Link: <https://acme-v02.api.letsencrypt.org/directory>;rel="index",
    <https://letsencrypt.org/documents/LE-SA-v1.2-November-15-2017.pdf>;
    rel="terms-of-service"
Location: https://acme-v02.api.letsencrypt.org/acme/acct/61709049
Replay-Nonce: mNNLKO9TRMACSqF6aFYTW-cV1N1WaiDPl656OkyjjUY
X-Frame-Options: DENY
Strict-Transport-Security: max-age=604800
Expires: Sun, 21 Jul 2019 17:34:28 GMT
Cache-Control: max-age=0, no-cache, no-store
Pragma: no-cache
Date: Sun, 21 Jul 2019 17:34:28 GMT
Connection: keep-alive
{
  "id": 61709049,
  "key": {
    "kty": "RSA",
    "n":"qC-JUzKtNXruqjRU7diqGBXWhONILbmEk6ZQPAnJuqo67Cl2iJ5JvxDj-Y6ApDj8cPE_pw2
        OcYMitt9m97Y5wBIdBt4xwY4CGtY-U8qsvWBFiulwkK54SGn-SjWqI7BQ7mPnqA58-PlD2R0
        9ne_ffYn98dYjCaxjVkbeb4LzVNNwtWXpJKmKelqahldR1-GcYwgEU3MVHvPBs8oi2EW8btN
        DS-mfh8sy7JXgCXghUAY_9IfpFZcRkm8VItO8O6TosKFQ0ZB1ejoMCorIbWg7wd8NDPnIji-
        PGq1XtKKP7Vc9fqHsAwYAsOQhFd_PS7K_sZBudmps28Mz57Lsa6fMRw",
    "e": "AQAB"
  },
  "contact": [
    "mailto:roberta_sgroi@hotmail.com"
```

```
  ],
  "initialIp": "104.211.30.207",
  "createdAt": "2019-07-21T17:34:28.14821161Z",
  "status": "valid"
}
```


Se la risposta ricevuta dal *server* è come quella mostrata in **??**, il *client* si occupa di generare la coppia di chiavi, di salvare il certificato nel seguente path "*/etc/letsencrypt/keys/0000_key-certbot.pem*" e procede con la creazione della *CSR*[6].

4.4.3 Specifiche del protocollo - Associazione con un account esterno

Il *server* potrebbe richiedere, in fase di registrazione di un nuovo account, il *binding* con un account esterno, non registrato e non gestito dal protocollo *ACME*. Questa richiesta viene esplicitata utilizzando il campo *externalAccountRegistered* e assegnandogli il valore *true*. La presenza di questa richiesta e l'assenza delle relative informazioni nel *JWS* di registrazione, causerà un rigetto da parte del *server* per *externalAccountRequired*.

L'associazione di un account esterno richiede interazioni extra tra il *client* e il *server*, quest'ultimo deve fornire al *client* un *MAC* - codificato utilizzando *base64* - e un identifico chiave - una stringa *ASCII* - utilizzando però meccanismi che stanno all'esterno del protocollo *ACME*.

La *POST* effettuata dal *client* in questo caso avrà dunque valorizzato il campo *externalAccountBinding* come si può notare in figura 4.5.

[6]Certificate Signing Request.

```
POST /acme/new-account HTTP/1.1
Host: example.com
Content-Type: application/jose+json
{
    "protected": base64url({
            "alg": "ES256",
            "jwk": /* account key */,
            "nonce": "K60BWPrMQG9SDxBDS_xtSw",
            "url": "https://example.com/acme/new-account"
    }),
    "payload": base64url({
            "contact": [
                    "mailto:cert-admin@example.org",
                    "mailto:admin@example.org"
            ],
            "termsOfServiceAgreed": true,
            "externalAccountBinding": {
                    "protected": base64url({
                            "alg": "HS256",
                            "kid": /* key identifier from CA */,
                            "url": "https://example.com/acme/new-account"
                    }),
                    "payload": base64url(/* same as in "jwk" above */),
                    "signature": /* MAC using MAC key from CA */
            }
    }),
    "signature": "5TWiqIYQfIDfALQv...x9C2mg8JGPxl5bI4"
}
```

Figura 4.5. POST-Request for new account creation with external account binding.

Nel *payload* della richiesta, il *client* dovrà inserire la chiave fornita dal *server*, mentre il *MAC* relativo verrà utilizzato per la corretta generazione del campo *signature*. Anche in questo caso l'*header* conterrà informazioni utili per decodificare e validare in maniera corretta il contenuto della richiesta:

- *alg* - indicazione sull'algoritmo di *MAC*;
- *kid* - l'identificativo chiave fornito dal *server*;
- *nonce* - in questo caso non deve essere valorizzato;
- *url* - valorizzato con lo stesso *URL* dell'oggetto esterno.

Come specificato nella sezione 4.2.2, il *server* non espone questa funzionalità, non è stato dunque possibile effettuare i test del caso.

4.4.4 Specifiche del protocollo - Aggiornamento delle chiavi

È indispensabile che venga data al *client* la possibilità di modificare la coppia di chiavi associate ad un certificato, soprattutto in situazioni in cui le chiavi potrebbero essere state compromesse.

Il *payload* di questa richiesta sarà composto da un altro oggetto *JWS* che verrà protetto utilizzando la nuova chiave, mentre la vecchia proteggerà la richiesta. Come specificato nel capitolo precedente, ogni oggetto *JWS* non può avere firme multiple, per questo motivo si utilizzano oggetti innestati, per garantire che ognuno di essi sia firmato una ed una sola volta.

```
POST /acme/key-change HTTP/1.1
Host: example.com
Content-Type: application/jose+json
{
    "protected": base64url({
            "alg": "ES256",
            "kid": "https://example.com/acme/acct/evOfKhNU60wg",
            "nonce": "S9XaOcxP5McpnTcWPIhYuB",
            "url": "https://example.com/acme/key-change"
    }),
    "payload": base64url({
            "protected": base64url({
                    "alg": "ES256",
                    "jwk": /* new key */,
                    "url": "https://example.com/acme/key-change"
            }),
            "payload": base64url({
                    "account": "https://example.com/acme/acct/evOfKhNU60wg",
                    "oldKey": /* old key */
            }),
            "signature": "Xe8B94RD30Azj2ea...8BmZIRtcSKPSd8gU"
    }),
    "signature": "5TWiqIYQfIDfALQv...x9C2mg8JGPxl5bI4"
}
```

Figura 4.6. POST-Request to update certificate associated keys.

Quando il *server* riceve una richiesta di questo tipo, oltre a validazioni sintattiche sulla forma dell'oggetto in input, dovrà anche controllare che la nuova chiave pubblica non venga usata in associazione ad un altro account. In questo caso il *server* deve restituire un errore che identifichi il conflitto e un *URL* con l'account di riferimento.

In seguito a ricerca approfondita, entrando anche in contatto con gli sviluppatori stessi, si è evinto che questa funzionalità non viene in realtà esposta. Comunemente gli utenti disattivano l'account ed eventualmente ne creano uno nuovo.

4.4.5 Specifiche del protocollo - Disattivazione dell'account

La disattivazione di account avviene mediante esplicita richiesta di aggiornamento dell'account, valorizzando il payload con l'apposito stato *deactivated*. Se questa richiesta va a buon fine, tutte le successive verranno rifiutate e il *server* restituirà un errore del tipo *unauthorized*.

```
POST /acme/acct/evOfKhNU60wg HTTP/1.1
Host: example.com
Content-Type: application/jose+json
{
    "protected": base64url({
            "alg": "ES256",
            "kid": "https://example.com/acme/acct/evOfKhNU60wg",
            "nonce": "ntuJWWSic4WVNSqeUmshgg",
            "url": "https://example.com/acme/acct/evOfKhNU60wg"
    }),
    "payload": base64url({
            "status": "deactivated"
    }),
    "signature": "earzVLd3m5M4xJzR...bVTqn7R08AKOVf3Y"
}
```

Figura 4.7. POST-Request to deactivate an account

Non è attualmente prevista una funzionalità che permetta di riattivare l'account, inoltre alla disattivazione, il *server* non revocherà nessun certificato rilasciato per l'account al fine di evitare mal funzionamenti.

4.4.6 Server Apache - Disattivazione dell'account

La richiesta di disattivazione di un account viene effettuata tramite *POST* al link https://acme-v01.api.letsencrypt.org/acme/reg/66187204 con il seguente input:

```
/*Richiesta codificata*/
{
  "protected":
     "eyJub25jZSI6ICIwMTAyaUJ0QmwzdFEzSE56TVl5ODBKczhHM19IanNrTTRrMVZ
    HUHdUaUpQdVZWVSIsICJhbGciOiAiUlMyNTYiLCAiandrIjogeyJlIjogIkFRQUIiLCAia3R5Ij
    ogIlJTQSIsICJuIjogIm1BU1lzOFNsaktLUzlJSkx5Q0tfVTBBUEM1SDlVY1BlODN6eTBBbnR6U
    FlCdWZrWEt4c0dEd3ZLdTh5aXE5RmE5SGFESldKWkVQelY2WGpVS18tTlFrcVdKTzR1RVl3SVJK
    V3hvSWdBWFhJYUhELVZpV2l0ZjRCVUpfUmRhWGQ2MVNpLTNWb19xemU0cHZHbm5jQjk1TnIwSll
    RbS1hd3lmVC1oWV9lYWUyODVCZGRfc3dBNk5LMUZPYnIxNFJXdG10WEo2NDJScWNSanB2Ry1ZYm
    9UNENoOVdHMHNTRXJvTDBSamRfSnNBTEk0dnE0TlFYVVk1SlVoa3Y0NU9zSFZCMjlReEVLZU1RQ
    1dTc0hGdWhBaGMwd2pOa1gxaVdXUktWdWVYUC1IdHB1WmFLWU5iUXotRFluQVRrSHZfcmFXWHdv
    LVFVRUo3b0tRQzkOU3YzdFFTdyJ9fQ",
  "payload":
     "ewogICJzdGF0dXMiOiAiZGVhY3RpdmF0ZWQiLCAKICAicmVzb3VyY2UiOiAicmVnI
    gp9",
  "signature":
     "ThH11T_WnEkYUGNpPv82x-SVgtFVR2c19XKRhp7PvdVjh8rcDJpKH_9ogvjciOQ
    WpNdOZmmpd_BGf1RpkdpITkdxyfmXNEWSt2fddUKRbr_WshzKwa32nlMUOgbi3HG-OOyz1A3vPM
    V1t1yR1qZVFts2eZYN9le0roOZUSm4SSaOn6kVQFf3Ga-FfUIYx1NYvMORrGYL8bVKjBlut_lhx
    GkCU1sbG-t-hEhIH-pkDS99kvxrZ1lAKu9FR44pO-KEC-vc9Ztor84EIBDZSW369EZ5eNHAccdC
    Kla4B97rQmhSYrSboIXzQmHvbXytGFEaxM-XbZn8tumdymzP6Oci5A"
}

/*Richiesta decodificata*/
{
  "protected": {
    "nonce":"0102iBtBl3tQ3HNzMYy80Js8G3_HjskM4k1VGPwTiJPuVVU",
```

```
    "alg":"RS256",
    "jwk":{
        "e":"AQAB",
        "kty":"RSA",
        "n":"mASYs8SljKKS9IJLyCK_UOAPC5H9UcPe83zy0AntzPYBufkXKxsGDwvKu8yiq9Fa9H
            aDJWJZEPzV6XjUK_-NQkqWJO4uEYwIRJWxoIgAXXIaHD-ViWitf4BUJ_RdaXd61Si-3
            Vo_qze4pvGnncB95Nr0JYQm-awyfT-hY_eae285Bdd_swA6NK1FObr14RWtmtXJ642R
            qcRjpvG-YboT4Ch9WGOsSEroLORjd_JsALI4vq4NQXUY5JUhkv45OsHVB29QxEKeMQC
            WSsHFuhAhc0wjNkX1iWWRKVueXP-HtpuZaKYNbQz-DYnATkHv_raWXwo-QUEJ7oKQC9
            4Sv3tQSw"
    }
  },
  "payload":{
      "status": "deactivated",
      "resource": "reg"
  },
  "signature":
      "ThH11T_WnEkYUGNpPv82x-SVgtFVR2c19XKRhp7PvdVjh8rcDJpKH_9ogvjciOQ
    WpNdOZmmpd_BGf1RpkdpITkdxyfmXNEWSt2fddUKRbr_WshzKwa32nlMUOgbi3HG-OOyz1A3vPM
    V1t1yR1qZVFts2eZYN9le0roOZUSm4SSaOn6kVQFf3Ga-FfUIYx1NYvMORrGYL8bVKjBlut_lhx
    GkCU1sbG-t-hEhIH-pkDS99kvxrZ1lAKu9FR44pO-KEC-vc9Ztor84EIBDZSW369EZ5eNHAccdC
    Kla4B97rQmhSYrSboIXzQmHvbXytGFEaxM-XbZn8tumdymzP6Oci5A"
}
```


La riposta del ottenuta dal *server* è stata la seguente:


```
HTTP 200
Content-Length: 666
Strict-Transport-Security: max-age=604800
Boulder-Requester: 66187204
Expires: Mon, 16 Sep 2019 22:17:45 GMT
Server: nginx
Connection: keep-alive
Pragma: no-cache
Cache-Control: max-age=0, no-cache, no-store
Date: Mon, 16 Sep 2019 22:17:45 GMT
X-Frame-Options: DENY
Content-Type: application/json
Replay-Nonce: 0002enxBBzx3e9v2azfkzkrkM7cyLPRwHlPn4oJHY7ITaBU

{
  "id": 66187204,
  "key": {
    "kty": "RSA",
    "n":
        "0DYz3lTMt05zN7CwhdWdC8ES7E8kMFqlaVz_VpqCXmihnWcjwZovA6i8xD-mJyuLiR_Gf
        IhdmR6s0J2zKL5Z-MzakcVmtNaSvFmgMe1ncIcLVzqC13bm4EEhzYcFjwZE5iDVj96_poox
        IENPSXCJoB-J3huXe7fKzJYLvdjQtdsFvTWl5qILVh5DKpPt_EJu6qbuhASDPv6xldrSf8y
        pTPfHooZEj6o7L0lJsZW-oU31Duh1f9BcQU1uMeDGlbK4bLqBygfbpp237-PA0XiCc_tans
        NRAwaPNrHgOPR8RmYEJQAvth9iZ7lDta27mIOci1n-qnbMou1mumZewxpy7w",
    "e": "AQAB"
  },
  "contact": [
    "mailto:roberta_sgroi@hotmail.com"
  ],
  "agreement":
      "https://letsencrypt.org/documents/LE-SA-v1.2-November-15-2017.pdf",
  "initialIp": "137.135.90.21",
```

```
  "createdAt": "2019-09-16T21:47:29Z",
  "status": "deactivated"
}
```

4.5 Richiesta di un certificato

La richiesta di emissione di un certificato avviene mediante accesso alla risorsa relativa da parte del *client*, ovvero viene effettuata una *POST* all'*URL* specifico (e.g. https://example.com/acme/new-orderHTTP/1.1).

4.5.1 Specifiche del protocollo - Richiesta di un certificato

Affinchè il *server* possa considerare valida la richiesta, il *client* deve creare l'oggetto utilizzato nella chiamata come definito nel paragrafo 4.1.2.

```
POST /acme/new-order HTTP/1.1
Host: example.com
Content-Type: application/jose+json
{
    "protected": base64url({
            "alg": "ES256",
            "kid": "https://example.com/acme/acct/evOfKhNU60wg",
            "nonce": "5XJ1L3lEkMG7tR6pA00clA",
            "url": "https://example.com/acme/new-order"
    }),
    "payload": base64url({
            "identifiers": [
                    { "type": "dns", "value": "www.example.org" },
                    { "type": "dns", "value": "example.org" }
            ],
            "notBefore": "2016-01-01T00:04:00+04:00",
            "notAfter": "2016-01-08T00:04:00+04:00"
    }),
    "signature": "H6ZXtGjTZyUnPeKn...wEA4TklBdh3e454g"
}
```

Figura 4.8. POST-Request for certificate issuance

Il *server* valuterà la correttezza della richiesta e risponderà appropriatamente, indicando in caso di non correttezza della richiesta, quali potrebbero essere le modifiche da apportare per poterla ritenere valida. In caso di riscontro positivo, il *server* risponderà con uno *Status Code* uguale a *201 - Created* e creerà la richiesta in stato *pending*.

```
HTTP/1.1 201 Created
Replay-Nonce: MYAuvOpaoIiywTezizk5vw
Link: <https://example.com/acme/directory>;rel="index"
Location: https://example.com/acme/order/TOlocE8rfgo
{
    "status": "pending",
    "expires": "2016-01-05T14:09:07.99Z",
    "notBefore": "2016-01-01T00:00:00Z",
    "notAfter": "2016-01-08T00:00:00Z",
    "identifiers": [
        { "type": "dns", "value": "www.example.org" },
        { "type": "dns", "value": "example.org" }
    ],
    "authorizations": [
        "https://example.com/acme/authz/PAniVnsZcis",
        "https://example.com/acme/authz/r4HqLzrSrpI"
    ],
    "finalize": "https://example.com/acme/order/TOlocE8rfgo/finalize"
}
```

Figura 4.9. POST-Response for certificate issuance

Il *client*, al ricevimento di questa risposta, dovrà procedere con gli step successivi prima della scadenza presente nel campo *expires*, altrimenti lo stato della richiesta verrà aggiornato a *invalid*. Per poter notificare il *server* del fatto che tutte le richieste sono in uno stato *valid*, il *client* deve prima finalizzare le autorizzazioni il cui stato è ancora pending e deve poi effettuare una richiesta *POST* alla risorsa che si occupa della finalizzazione delle richieste, mettendo nel corpo della richiesta una *CSR*[7] con esattamente lo stesso insieme di informazioni passati nella chiamata precedente al *server*. In figura 4.10 un esempio della suddetta chiamata.

```
POST /acme/order/TOlocE8rfgo/finalize HTTP/1.1
Host: example.com
Content-Type: application/jose+json
{
    "protected": base64url({
            "alg": "ES256",
            "kid": "https://example.com/acme/acct/evOfKhNU60wg",
            "nonce": "MSF2j2nawWHPxxkE3ZJtKQ",
            "url": "https://example.com/acme/order/TOlocE8rfgo/finalize"
    }),
    "payload": base64url({
            "csr": "MIIBPTCBxAIBADBFMQ...FS6aKdZeGsysoCo4H9P",
    }),
    "signature": "uOrUfIIk5RyQ...nw62Ay1cl6AB"
}
```

Figura 4.10. POST-Request to finalize the certificate order

Qualora la richiesta sia correttamente formata e il suo contenuto sia valido, il *server* si occuperà di modificare lo stato della richiesta in *valid* e restituirà al *client* un *200 - OK*.

[7] Certificate Signing Request.

```
HTTP/1.1 200 OK
Replay-Nonce: CGf81JWBsq8QyIgPCi9Q9X
Link: <https://example.com/acme/directory>;rel="index"
Location: https://example.com/acme/order/TOlocE8rfgo
{
    "status": "valid",
    "expires": "2016-01-20T14:09:07.99Z",
    "notBefore": "2016-01-01T00:00:00Z",
    "notAfter": "2016-01-08T00:00:00Z",
    "identifiers": [
            { "type": "dns", "value": "www.example.org" },
            { "type": "dns", "value": "example.org" }
    ],
    "authorizations": [
            "https://example.com/acme/authz/PAniVnsZcis",
            "https://example.com/acme/authz/r4HqLzrSrpI"
    ],
    "finalize": "https://example.com/acme/order/TOlocE8rfgo/finalize",
    "certificate": "https://example.com/acme/cert/mAt3xBGaobw"
}
```

Figura 4.11. POST-Response to finalize the certificate order

4.5.2 Server Apache - Richiesta di un certificato

La richiesta di emissione di un certificato viene effettuata tramite *POST* al link https://acme-v02.api.letsencrypt.org/acme/new-order con il seguente contenuto:

```
/*Richiesta codificata*/
{
  "protected":"eyJub25jZSI6ICJtTk5MSO85VFJNQUNTcUY2YUZZVFctY1YxTjFXYWlEUGw2NTZP
    a3lqalVZIiwgInVybCI6ICJodHRwczovL2FjbWUtdjAyLmFwaS5sZXRzZW5jcnlwdC5vcmcvYWN
    tZS9uZXctb3JkZXIiLCAia2lkIjogImh0dHBzOi8vYWNtZS12MDIuYXBpLmxldHNlbmNyeXB0Lm
    9yZy9hY21lL2FjY3QvNjE3MDkwNDkiLCAiYWxnIjogIlJTMjU2In0",
  "payload":"ewogICJpZGVudGlmaWVycyI6IFsKICAgIHsKICAgICAgInR5cGUiOiAiZG5zIiwgCi
    AgICAgICJ2YWx1ZSI6ICJ1YnVudHVhcGFjaGUuZWFzdHVzLmNsb3VkYXBwLmF6dXJlLmNvbSIKI
    CAgIH0KICBdCn0",
  "signature":"aRVEeBuV6UsPJ1T0yPBe6ptS4uVKiXRm7cNIzByC8OETdcgzDjrSEGhz_e5B28R1
    o3xz_PUn5b_EyL8w3u5qvJ9c6o2BS1yMe7Am7rxD7k9SDgkJmao6_xa7XUKfLbdYZN-ZolZrHIu
    jJLkAZ1gj6DNOAA7-KRdO6fWpI_QmmxlmMrOkFgjIvLm_Gf-aDCcpm3udvz5nE-jpaCQ8Xr4Lz1
    nXtb0U64malVlrHG4iscHX4sKvGkC9-xb_QBzY2KCPADnWOplyVEgI8C9lsv-X5Y9rBuQHlQyCz
    YvdV_NzF21DGyAwgjAZ1y6JeVbrMUGjyNqlNBVOckVoaMEpc7kSAw"
}

/*Richiesta decodificata*/
{
  "protected":{
    "nonce":"mNNLKO9TRMACSqF6aFYTW-cV1N1WaiDPl656OkyjjUY",
    "url":"https://acme-v02.api.letsencrypt.org/acme/new-order",
    "kid":"https://acme-v02.api.letsencrypt.org/acme/acct/61709049",
    "alg":"RS256"
  },
  "payload":{
    "identifiers": [
```

```
      {
        "type": "dns",
        "value": "ubuntuapache.eastus.cloudapp.azure.com"
      }]
  },
  "signature":"aRVEeBuV6UsPJ1T0yPBe6ptS4uVKiXRm7cNIzByC8OETdcgzDjrSEGhz_e5B28R1
    o3xz_PUn5b_EyL8w3u5qvJ9c6o2BS1yMe7Am7rxD7k9SDgkJmao6_xa7XUKfLbdYZN-ZolZrHIu
    jJLkAZ1gj6DNOAA7-KRdO6fWpI_QmmxlmMrOkFgjIvLm_Gf-aDCcpm3udvz5nE-jpaCQ8Xr4Lz1
    nXtb0U64malVlrHG4iscHX4sKvGkC9-xb_QBzY2KCPADnWOplyVEgI8C9lsv-X5Y9rBuQHlQyCz
    YvdV_NzF21DGyAwgjAZ1y6JeVbrMUGjyNqlNBVOckVoaMEpc7kSAw"
}
```


Rispetto a quanto mostrato in figura 4.8, si può facilmente notare che non è stato richiesto il settaggio esplicito del valore dei campi *notBefore* e *notAfter*, così facendo sarà il *server* a settare il valore di queste due proprietà utilizzando quelli che sono i valori standard.

La risposta ricevuto dal *server* è stata la seguente:


```
{
  "status": "pending",
  "expires": "2019-07-28T17:35:32.945684184Z",
  "identifiers": [
    {
      "type":"dns",
      "value":"ubuntuapache.eastus.cloudapp.azure.com"
    }
  ],
  "authorizations": [
    "https://acme-v02.api.letsencrypt.org/acme/authz/8621zDGUjKxqTsdMa2PPfVgDyY
       FRPPu3MpaOjKr99js"
  ],
  "finalize":"https://acme-v02.api.letsencrypt.org/acme/finalize/61709049/757557065"
}
```


A questo punto il *client* deve rispondere alla sfide per poter portare avanti la richiesta di emissione e procedere successivamente con la finalizzazione. Una volta risolte le sfide, il *client* crea una *CSR*[8] per procedere con la finalizzazione ed effettua una *POST* all'url https://acme-v02.api.letsencrypt.org/acme/finalize/61709049/757557065 con il seguente contenuto:


```
/*Richiesta codificata*/
{
  "protected":"eyJub25jZSI6ICJQVEFfRXFzMnkwYV91WDJsUkszWFlQZTRXcWlmc1QwTzlCZDhJ
    cDh5a2JRIiwgInVybCI6ICJodHRwczovL2FjbWUtdjAyLmFwaS5sZXRzZW5jcnlwdC5vcmcvYWN
    tZS9maW5hbGl6ZS82MTcwOTA0OS83NTc1NTcwNjUiLCAia2lkIjogImh0dHBzOi8vYWNtZS12MD
    IuYXBpLmxldHNlbmNyeXB0Lm9yZy9hY21lL2FjY3QvNjE3MDkwNDkiLCAiYWxnIjogIlJTMjU2I
    n0",
  "payload":"ewogICJyZXNvdXJjZSI6ICJuZXctY2VydCIsIAogICJjc3IiOiAiTUlJQ2lUQ0NBWE
    VDQVFJd0FEQ0NBU0l3RFFZSktvWklodmNOQVFFQkJRQURnZ0VQQURDQ0FRb0NnZ0VCQVBGejRHc
    GpCQk1OLXFuU0VCb2J3bktOTkZQV19iN2c3czdqZVQyXzFvQzBuZnhSNm1rd0hoUlBweHBkMFJx
    cHhlUTRWQnhucVFOZ000LVp1bVNlWjc4YkFwXzRJaUlPWmhJemJ5QnVDTGxTekxfZExGd3RlMjZ
    2N0xQRXI2UkxVZlF4b2JPcjVyT1pEQ3htVUpPLS1QZ1Z2NXpEY185SFRwWWgtTTdLU3c3NS1pd1
    A2ZGhQVGpTZnF6MVJXYmhUTlJIanZaaXd6Vkhxb3hPS0NIUEZUQ0x0VWhGQ1Z4UWYyaDFpWUc1e
    W94c01uTDNzeVBtNnpfa2tsWU41cXJGRkZfUGdIZG9ELWY2MndTazFRdGtyUkFCalQ5YXgyNXhI
    THMweWRQcmhzMk5KOUY1RW9qVldsSWVucFY0Ri1ha2JPSUYtYmhXVHBJWDJHYjdiaXBIYVhqMEN
    Bd0VBQWFCRU1FSUdDU3FHU0liM0RRRUpEakUxTURNd01RWURWUjBSQkNvd0tJSW1kV0oxYm5SMV
    lYQmhZMmhsTG1WaGMzUjFjeTVqYkc5MVpHRndjQzVoZW5WeVpTNWpiMjB3RFFZSktvWklodmNOQ
```


[8] Certificate Signing Request.

```
    VFFTEJRQURnZ0VCQU5XcHh5N1hHTXUxUkJHeHM2VUJmVHB3RGNJTFlCeVdIaHp5V0lPOUVsMO9R
    TTZhZ1F3QTB1bnVic2VhQ3ZfeFdWbGozTnI5Z1poUjRTSGtJSTBIOU9TTzUxQXlCbTJ2bUQ0R1R
    OM3BWcEJmd3ZETmxPUUJ6VXVxWHNUcUxoQVlJTDdLM19nUkNZQjYxTEJjdFcydjl1QzRQcWlDMV
    BzTTJfckpGbGoyczlrcTBvNnlfZzVHeDVhZkZPOHI0SnJKNTlzaFgxenFmeGpULTFWSzB3MDM0V
    DMydXZadEowOEhZeERVMEhrenlMZm5CNEJKYkl2SWVmMXhGa21NLWVIS3E2NUJLSG1hT3c3R3Nn
    N2dzZkxWdmF5MEs1VW4yUk5HVkpCRnFIS1JLN0FhbXhqUFVORUdBX1Y1WlR0TzJfS0hEOHFuRno
    xZ2JNY1BxWjJIX25Mb1NqWSIKfQ",
  "signature":"mnKmlRGrFRCBPXdx6lQMU7iPp2-fNc3WDQfMj4CJwZXqU5uZ86mNaiTF93OXOf9j
    tobe3AfHsWQagXkP8ubX98qYx6D6A9oi9IYqQ3wUZtVJRnz2cNqiYL5u18WL8KVgPunPv4Xcn36
    yzq-m3KpltvIhRcOheg5EKhJ_yzHL9lvAgF6_tXzhdsJZoak6XtQsvQcugmRiHR0aIvDpO7yzB9
    aCwZXZ2PlawVvPikc2GTXxDl1LsPwAIcngEBV1rVYJdMQ7yceX9kBTEB8rcsS8nqkPrhWR4i5Cm
    nOrWCCifoXQZ-z9o9UmFeweVrOzVEpELNSLq8R2mOSKjK0W6aWQpA"
}

/*Richiesta decodificata*/
{
  "protected":{
    "nonce":"PTA_Eqs2y0a_uX2lRK3XYPe4WqifsTOO9Bd8Ip8ykbQ",
    "url":"https://acme-v02.api.letsencrypt.org/acme/finalize/61709049/757557065",
    "kid":"https://acme-v02.api.letsencrypt.org/acme/acct/61709049",
    "alg":"RS256"
  },
  "payload":{
    "resource": "new-cert",
    "csr":"MIICiTCCAXECAQIwADCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAPFz4Gpj
        BBMN-qnSEBobwnKNNFPW_b7g7s7jeT2_1oC0nfxR6mkwHhRPpxpd0RqpxeQ4VBxnqQNgM4-
        ZumSeZ78bAp_4IiIOZhIzbyBuCLlSzL_dLFwte26v7LPEr6RLUfQxobOr5rOZDCxmUJO--P
        gVv5zDc_9HTpYh-M7KSw75-iwP6dhPTjSfqz1RWbhTNRHjvZiwzVHqoxOKCHPFTCLtUhFCV
        xQf2h1iYG5yoxsMnL3syPm6z_kklYN5qrFFF_PgHdoD-f62wSk1QtkrRABjT9ax25xHLs0y
        dPrhs2NJ9F5EojVWlIenpV4F-akbOIF-bhWTpIX2Gb7bipHaXj0CAwEAAaBEMEIGCSqGSIb
        3DQEJDjE1MDMwMQYDVR0RBCowKIImdWJ1bnR1YXBhY2hlLmVhc3R1cy5jbG91ZGFwcC5hen
        VyZS5jb20wDQYJKoZIhvcNAQELBQADggEBANWpxy7XGMu1RBGxs6UBfTpwDcILYByWHhzyW
        IO9El3OQM6agQwA0unubseaCv_xWVlj3Nr9gZhR4SHkII0H9OSO51AyBm2vmD4GTN3pVpBf
        wvDNlOQBzUuqXsTqLhAYIL7K3_gRCYB61LBctW2v9uC4PqiC1PsM2_rJFlj2s9kq0o6y_g5
        Gx5afFO8r4JrJ59shX1zqfxjT-1VK0w034T32uvZtJ08HYxDU0HkzyLfnB4BJbIvIef1xFk
        mM-eHKq65BKHmaOw7Gsg7gsfLVvayOK5Un2RNGVJBFqHKRK7AamxjPUNEGA_V5ZTtO2_KHD
        8qnFz1gbMcPqZ2H_nLoSjY"
  },
  "signature":"mnKmlRGrFRCBPXdx6lQMU7iPp2-fNc3WDQfMj4CJwZXqU5uZ86mNaiTF93OXOf9j
    tobe3AfHsWQagXkP8ubX98qYx6D6A9oi9IYqQ3wUZtVJRnz2cNqiYL5u18WL8KVgPunPv4Xcn36
    yzq-m3KpltvIhRcOheg5EKhJ_yzHL9lvAgF6_tXzhdsJZoak6XtQsvQcugmRiHR0aIvDpO7yzB9
    aCwZXZ2PlawVvPikc2GTXxDl1LsPwAIcngEBV1rVYJdMQ7yceX9kBTEB8rcsS8nqkPrhWR4i5Cm
    nOrWCCifoXQZ-z9o9UmFeweVrOzVEpELNSLq8R2mOSKjK0W6aWQpA"
}
```


La risposta del *server* specifica al *client* la validità della richiesta:


```
{
  "status": "valid",
  "expires": "2019-07-28T17:35:32Z",
  "identifiers": [
    {
      "type": "dns",
      "value": "ubuntuapache.eastus.cloudapp.azure.com"
    }
  ],
  "authorizations": [
```

```
    "https://acme-v02.api.letsencrypt.org/acme/authz/8621zDGUjKxqTsdMa2PPfVgDyY
        FRPPu3MpaOjKr99js"
  ],
  "finalize":
      "https://acme-v02.api.letsencrypt.org/acme/finalize/61709049/757557065",
  "certificate":
      "https://acme-v02.api.letsencrypt.org/acme/cert/04a953433f1fd1
    6b6484e339d379c9b4a6e1"
}
```

4.5.3 Specifiche del protocollo - Pre-autorizzazione

L'oggetto autorizzazione viene creato automaticamente in seguito alla corretta validazione della richiesta di emissione di un nuovo certificato. Dal momento che il protocollo *ACME* consente di effettuare l'associazione di account esterni allo stesso sistema, viene inoltre fornita la possibilià di generare, sotto esplicita richiesta, un oggetto autorizzazione che verrà ritenuto valido per tutte le richieste provenienti da esso.

Il *client* effettua una richiesta *POST* alla risorsa *new authorization* di cui un esempio in figura 4.12

```
POST /acme/new-authz HTTP/1.1
Host: example.com
Content-Type: application/jose+json
{
    "protected": base64url({
            "alg": "ES256",
            "kid": "https://example.com/acme/acct/evOfKhNU60wg",
            "nonce": "uQpSjlRb4vQVCjVYAyyUWg",
            "url": "https://example.com/acme/new-authz"
    }),
    "payload": base64url({
            "identifier": {
                    "type": "dns",
                    "value": "example.org"
            }
    }),
    "signature": "nuSDISbWG8mMgE7H...QyVUL68yzf3Zawps"
}
```

Figura 4.12. POST-Request for a new account authorization

Prima di rilasciare una pre-autorizzazione, il *server* si accerterà che l'identificatore si trovi in una posizione tale per cui sarà quasi certa l'emissione del certificato. Se tutti i controlli dovessero andare a buon fine il *server* tiene traccia di questa nuova autorizzazione, restituisce il relativo *URL* nel *location header* della risposta e il *client* potrà procedere con i passi successivi per completare il processo di autorizzazione (vedi 4.6).

Come specificato nella sezione 4.2.2, il *server* non espone questa funzionalità, non è stato dunque possibile effettuare i test del caso.

4.5.4 Specifiche del protocollo - Download di un certificato

Il *client*, al fine di scaricare il certificato, dovrà effettuare una richiesta *POST-as-GET* all'*URL* del certificato appena emesso. Il *server* permettà il download del certificato e fornirà esplicite informazioni che possano permettere al *client* di risalire la *chain* delle *certification authority*.

```
POST /acme/cert/mAt3xBGaobw HTTP/1.1
Host: example.com
Content-Type: application/jose+json
Accept: application/pem-certificate-chain
{
    "protected": base64url({
            "alg": "ES256",
            "kid": "https://example.com/acme/acct/evOfKhNU60wg",
            "nonce": "uQpSjlRb4vQVCjVYAyyUWg",
            "url": "https://example.com/acme/cert/mAt3xBGaobw"
    }),
    "payload": "",
    "signature": "nuSDISbWG8mMgE7H...QyVUL68yzf3Zawps"
}

HTTP/1.1 200 OK
Content-Type: application/pem-certificate-chain
Link: <https://example.com/acme/directory>;rel="index"
-----BEGIN CERTIFICATE-----
[End-entity certificate contents]
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
[Issuer certificate contents]
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
[Other certificate contents]
-----END CERTIFICATE-----
```

Figura 4.13. POST-Request/Response for certificate download

Il formato del certificato che viene scaricato di default è *PEM*[9], ma viene fornita la possibilità di specificare nell'*header* della richiesta il formato che preferisce.

4.5.5 Server Apache - Download di un certificato

Prima di scaricare il certificato, il *client* effettua una nuova chiamata alla risorsa *order* al fine di verificare lo stato della richiesta e ottenendo una risposta da parte del *server* identica a quella ricevuto in seguito a finalizzazione della richiesta. Considerando lo stato valido della richiesta, viene dunque effettuata una richiesta *POST* all'url https://acme-v02.api.letsencrypt.org/acme/cert/04a953433f1fd16b6484e339d379c9b4a6e1 con il seguente contenuto:

```
/*Richiesta codificata*/
{
  "protected":"eyJub25jZSI6ICI1ajBZTzJNWGZpa1F2Y0RMaDJrNEpaVjJKTkRhVEFwVmU0MWE1
    c1p2cXN3IiwgInVybCI6ICJodHRwczovL2FjbWUtdjAyLmFwaS5sZXRzZW5jcnlwdC5vcmcvYWN
    tZS9jZXJ0LzA0YTk1MzQzM2YxZmQxNmI2NDg0ZTMzOWQzNzljOWI0YTZlMSIsICJraWQiOiAiaH
    R0cHM6Ly9hY211LXYwMi5hcGkubGV0c2VuY3J5cHQub3JnL2FjbWUvYWNjdC82MTcwOTA0OSIsI
    CJhbGciOiAiUlMyNTYifQ",
  "payload": "",
  "signature":"TnfbbnGHCFvqqi5ht4aDDjXBt6-sz_I89RS3y2prnSPfYdW7mt4VNb36hi6AqHHI
    NBYa8MHz0SXSZutI3fe9hWVNSxo1Q-Go4VikjnS-xBLjHPQBRAC7Wr1OETWSDzSqTOQwuVqzTCp
```

[9]Privacy Enhanced Mail.

```
    oKztoimBDdC8gZfWgwNQiDDWNKAVT7MFrg1TYisneTtF1m62WjRqOqkmIJRY6AmOwrFLw8Pg7CH
    8-Rs0Haffye8oBD2zJ26OTXkTGt2sVTPK1nDcizwcoen9cnm32gWE01a8i5CZHmB3ShX9N8A7aS
    srdG6cSVjX5jwIJfuyvMC1SPqLUqM3nBjW1qX3V1ix0_NVP3RKC2w"
}

/*Richiesta decodificata*
{
  "protected":{
    "nonce":"PTA_Eqs2y0a_uX2lRK3XYPe4WqifsTOO9Bd8Ip8ykbQ",
    "url":"https://acme-v02.api.letsencrypt.org/acme/finalize/61709049/757557065",
    "kid":"https://acme-v02.api.letsencrypt.org/acme/acct/61709049",
    "alg":"RS256"
  },
  "payload": "",
  "signature":"TnfbbnGHCFvqqi5ht4aDDjXBt6-sz_I89RS3y2prnSPfYdW7mt4VNb36hi6AqHHI
    NBYa8MHz0SXSZutI3fe9hWVNSxo1Q-Go4VikjnS-xBLjHPQBRAC7Wr1OETWSDzSqTOQwuVqzTCp
    oKztoimBDdC8gZfWgwNQiDDWNKAVT7MFrg1TYisneTtF1m62WjRqOqkmIJRY6AmOwrFLw8Pg7CH
    8-Rs0Haffye8oBD2zJ26OTXkTGt2sVTPK1nDcizwcoen9cnm32gWE01a8i5CZHmB3ShX9N8A7aS
    srdG6cSVjX5jwIJfuyvMC1SPqLUqM3nBjW1qX3V1ix0_NVP3RKC2w"
}
```

La risposta ricevuta dal *server* è il certificato vero e proprio che il *client* salverà nel file system - nel seguente path "/etc/letsencrypt/live/ubuntuapache.eastus.cloudapp.azure.com/cert.pem" -, insieme alla chiave privata, e installerà correttamente:

```
-----BEGIN CERTIFICATE-----
MIIFhDCCBGygAwIBAgISBKlTQz8f0WtkhOM503nJtKbhMA0GCSqGSIb3DQEBCwUA
MEoxCzAJBgNVBAYTAlVTMRYwFAYDVQQKEw1MZXQncyBFbmNyeXB0MSMwIQYDVQQD
ExpMZXQncyBFbmNyeXB0IEF1dGhvcml0eSBYMzAeFw0xOTA3MjExNjM1MzhaFw0x
OTEwMTkxNjM1MzhaMDExLzAtBgNVBAMTJnVidW50dWFwYWNoZS5lYXN0dXMuY2xv
dWRhcHAuYXp1cmUuY29tMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA
8XPgamMEEw36qdIQGhvCco00U9b9vuDuzuN5Pb/WgLSd/FHqaTAeFE+nGl3RGqnF
5DhUHGepA2Azj5m6ZJ5nvxsCn/giIg5mEjNvIG4IuVLMv90sXC17bq/ss8SvpEtR
9DGhs6vms5kMLGZQk774+BW/nMNz/0dOliH4zspLDvn6LA/p2E9ONJ+rPVFZuFM1
EeO9mLDNUeqjE4oIc8VMIu1SEUJXFB/aHWJgbnKjGwycvezI+brP+SSVg3mqsUUX
8+Ad2gP5/rbBKTVC2StEAGNP1rHbnEcuzTJ0+uGzY0n0XkSiNVaUh6elXgX5qRs4
gX5uFZOkhfYZvtuKkdpePQIDAQABo4ICezCCAncwDgYDVR0PAQH/BAQDAgWgMB0G
A1UdJQQWMBQGCCsGAQUFBwMBBggrBgEFBQcDAjAMBgNVHRMBAf8EAjAAMB0GA1Ud
DgQWBBTG/kHMEKC1EVwpfzM+n318CorpMDAfBgNVHSMEGDAWgBSoSmpjBH3duubR
ObemRWXv86jsoTBvBggrBgEFBQcBAQRjMGEwLgYIKwYBBQUHMAGGImh0dHA6Ly9v
Y3NwLmludC14My5sZXRzZW5jcnlwdC5vcmcwLwYIKwYBBQUHMAKGI2h0dHA6Ly9j
ZXJ0LmludC14My5sZXRzZW5jcnlwdC5vcmcvMDEGA1UdEQQqMCiCJnVidW50dWFw
YWNoZS5lYXN0dXMuY2xvdWRhcHAuYXp1cmUuY29tMEwGA1UdIARFMEMwCAYGZ4EM
AQIBMDcGCysGAQQBgt8TAQEBMCgwJgYIKwYBBQUHAgEWGmh0dHA6Ly9jcHMubGV0
c2VuY3J5cHQub3JnMIIBBAYKKwYBBAHWeQIEAgSB9QSB8gDwAHcAdH7agzGtMxCR
IZzOJU9CcMK//V5CIAjGNzV55hB7zFYAAAFsFZpuZAAABAMASDBGAiEA/i9ihA59
nshyEK1QVnSrchfbbm14j/BmVeyImmBT334CIQDTDKWWjbCwj+5JWeghAk0cK3KG
cBKQrSiZuLLNog6vQgB1AGPy283oO8wszwtyhCdXazOkjWF3j711pjixx2hUS9iN
AAABbBWablEAAAQDAEYwRAIfe29cCWXznswaOLO0BBNAS/qqZSJ5TEA+DTa1kR2M
KQIhAPLLLzgUjYHmvGctpidux2fEPGHYI6xiEDOPP5tK0+GgMA0GCSqGSIb3DQEB
CwUAA4IBAQBbxXEQsgpEEBQHSadXHz51Yc0pxHQGMh6aI7CPXfDhJceEolC9X7PG
YmCMMTp32e5Qn50rO8xTWNFU5zJHnu3MxbU6E4YcpI9KtB8X0fKANK+rqlwTMuII
KJCImEh4bsD3Ey9Gr707s+JovHkxre27WYV0GvTlEQrxG/SyqG2YYwmBCyDljvkU
yzXCrbdmKzQVNxsxq7yGab9njEyMIwyu9MCgQelG4CE1Z4MbMbK9KfRy1KB5725n
KmmqWFXVTERGhvTJqHfoVABpT0+6UpY9RVlaJtEJqMEmeVRyNUpD+Sl1d7O+SMqq
ba07jGRIasfMo5em0aBkEJ+k06JXb4D6
-----END CERTIFICATE-----
```

```
-----BEGIN CERTIFICATE-----
MIIEkjCCA3qgAwIBAgIQCgFBQgAAAVOFc2oLheynCDANBgkqhkiG9w0BAQsFADA/
MSQwIgYDVQQKExtEaWdpdGFsIFNpZ25hdHVyZSBUcnVzdCBDby4xFzAVBgNVBAMT
DkRTVCBSb290IENBIFgzMB4XDTE2MDMxNzE2NDA0NloXDTIxMDMxNzE2NDA0Nlow
SjELMAkGA1UEBhMCVVMxFjAUBgNVBAoTDUxldCdzIEVuY3J5cHQxIzAhBgNVBAMT
GkxldCdzIEVuY3J5cHQgQXV0aG9yaXR5IFgzMIIBIjANBgkqhkiG9w0BAQEFAAOC
AQ8AMIIBCgKCAQEAnNMM8FrlLke3cl03g7NoYzDq1zUmGSXhvb418XCSL7e4S0EF
q6meNQhY7LEqxGiHC6PjdeTm86dicbp5gWAf15Gan/PQeGdxyGkOlZHP/uaZ6WA8
SMx+yk13EiSdRxta67nsHjcAHJyse6cF6s5K671B5TaYucv9bTyWaN8jKkKQDIZ0
Z8h/pZq4UmEUEz9l6YKHy9v6Dlb2honzhT+Xhq+w3Brvaw2VFn3EK6BlspkENnWA
a6xK8xuQSXgvopZPKiAlKQTGdMDQMc2PMTiVFrqoM7hD8bEfwzB/onkxEz0tNvjj
/PIzark5McWvxI0NHWQWM6r6hCm21AvA2H3DkwIDAQABo4IBfTCCAXkwEgYDVR0T
AQH/BAgwBgEB/wIBADAOBgNVHQ8BAf8EBAMCAYYwfwYIKwYBBQUHAQEEczBxMDIG
CCsGAQUFBzABhiZodHRwOi8vaXNyZy50cnVzdGlkLm9jc3AuaWRlbnRydXN0LmNv
bTA7BggrBgEFBQcwAoYvaHR0cDovL2FwcHMuaWRlbnRydXN0LmNvbS9yb290cy9k
c3Ryb290Y2F4My5wN2MwHwYDVR0jBBgwFoAUxKexpHsscfrb4UuQdf/EFWCFiRAw
VAYDVR0gBE0wSzAIBgZngQwBAgEwPwYLKwYBBAGC3xMBAQEwMDAuBggrBgEFBQcC
ARYiaHR0cDovL2Nwcy5yb290LXgxLmxldHNlbmNyeXB0Lm9yZzA8BgNVHR8ENTAz
MDGgL6AthitodHRwOi8vY3JsLmlkZW50cnVzdC5jb20vRFNUUk9PVENBWDNDUkwu
Y3JsMB0GA1UdDgQWBBSoSmpjBH3duubRObemRWXv86jsoTANBgkqhkiG9w0BAQsF
AAOCAQEA3TPXEfNjWDjdGBX7CVW+dla5cEilaUcne8IkCJLxWh9KEik3JHRRHGJo
uM2VcGfl96S8TihRzZvoroed6ti6WqEBmtzw3Wodatg+VyOeph4EYpr/1wXKtx8/
wApIvJSwtmVi4MFU5aMqrSDE6ea73Mj2tcMyo5jMd6jmeWUHK8so/joWUoHOUgwu
X4Po1QYz+3dszkDqMp4fklxBwXRsW10KXzPMTZ+sOPAveyxindmjkW8lGy+QsRlG
PfZ+G6Z6h7mjem0Y+iWlkYcV4PIWL1iwBi8saCbGS5jN2p8M+X+Q7UNKEkROb3N6
KOqkqm57TH2H3eDJAkSnh6/DNFu0Qg==
-----END CERTIFICATE-----
```


4.6 Autorizzazione di un identificatore

Questa funzionalità è indispensabile per il *server* al fine di accertarsi che il richiedente sia il possessore della coppia di chiavi e che abbia il controllo sull'identificatore.

Gli step presenti in questa fase, sono quelli standard di un sistema a sfide. Il *client* dovrà accedere all'*URL* per l'accesso alle sfide messe a disposizione dal *server*, rispondere ad esse utilizzando la tipologia prescelta, e dovrà infine notificare il *server* in modo da avvertirlo che i controlli di validità possono essere effettuati.

Il compito del *server*, una volta ricevuta la notifica, sarà quello di aggiornare il documento di autorizzazione con la risposta alla sfida ricevuta. Una volta terminata una delle convalide, l'autorizzazione passerà in stato *valid* o *invalid*, corrispondente al risultato dei controlli effettuati per considerare l'account autorizzato per l'identificatore. Se lo stato finale sarà *valid*, il *server* dovrà includere un campo *expires*. Le sfide considerate valide dal *server* possono anche essere rimosse una volta validate, quelle invece considerate non valide non potranno essere cancellate. Qualora la convalida non dovesse andare a buon fine, il *client* dovrebbe annullare qualsiasi operazione intrapresa per rispondere alla sfida.

4.6.1 Specifiche del protocollo

Le informazioni necessarie al *client* per poter rispondere alle sfide, e che porteranno il *server* a fare le verifiche richieste per l'autorizzazione di un identificatore, sono contenute nella risposta ottenuta dal *server* in seguito ad un richiesta di un nuovo certificato o di pre-autorizzazione, se supportata dal *server*. In figura 4.14 un esempio dello scenario classico.

```
POST /acme/authz/PAniVnsZcis HTTP/1.1
Host: example.com
Content-Type: application/jose+json
{
    "protected": base64url({
            "alg": "ES256",
            "kid": "https://example.com/acme/acct/evOfKhNU60wg",
            "nonce": "uQpSjlRb4vQVCjVYAyyUWg",
            "url": "https://example.com/acme/authz/PAniVnsZcis"
    }),
    "payload": "",
    "signature": "nuSDISbWG8mMgE7H...QyVUL68yzf3Zawps"
}

HTTP/1.1 200 OK
Content-Type: application/json
Link: <https://example.com/acme/directory>;rel="index"
{
    "status": "pending",
    "expires": "2016-01-02T14:09:30Z",
    "identifier": {
            "type": "dns",
            "value": "www.example.org"
    },
    "challenges": [
            {
                    "type": "http-01",
                    "url": "https://example.com/acme/chall/prV_B7yEyA4",
                    "token": "DGyRejmCefe7v4NfDGDKfA"
            },
            {
                    "type": "dns-01",
                    "url": "https://example.com/acme/chall/Rg5dV14Gh1Q",
                    "token": "DGyRejmCefe7v4NfDGDKfA"
            }
    ]
}
```

Figura 4.14. POST-Request for identifier authorization

A partire da questa risposta, il *client* preparerà la risposta alle sfide e notificherà il *server*. La *POST* di notifica al *server* dovrà avere il corpo dell'oggetto *JWS* vuoto.

```
POST /acme/chall/prV_B7yEyA4 HTTP/1.1
Host: example.com
Content-Type: application/jose+json
    {
    "protected": base64url({
            "alg": "ES256",
            "kid": "https://example.com/acme/acct/evOfKhNU60wg",
            "nonce": "Q_s3MWoqT05TrdkM2MTDcw",
            "url": "https://example.com/acme/chall/prV_B7yEyA4"
    }),
    "payload": base64url({}),
    "signature": "9cbg5JO1Gf5YLjjz...SpkUfcdPai9uVYYQ"
}
```

Figura 4.15. POST-Request to notify the server for challenges checks

In seguito ad autorizzazione andata a buon fine, il *client* può richiedere in qualsiasi momento la sua disattivazione. Una volta ricevuta esplicita richiesta, il *server* dovrà aggiornare il contenuto del documento di autorizzazione.

```
POST /acme/authz/PAniVnsZcis HTTP/1.1
Host: example.com
Content-Type: application/jose+json
{
    "protected": base64url({
            "alg": "ES256",
            "kid": "https://example.com/acme/acct/evOfKhNU60wg",
            "nonce": "xWCM9lGbIyCgue8di6ueWQ",
            "url": "https://example.com/acme/authz/PAniVnsZcis"
    }),
    "payload": base64url({
            "status": "deactivated"
    }),
    "signature": "srX9Ji7Le9bjszhu...WTFdtujObzMtZcx4"
}
```

Figura 4.16. POST-Request to deactivate authorization of identifiers

4.6.2 Server Apache

Questo step viene eseguito dopo la richiesta di un certificato e prima delle finalizzazione che porterà al download del certificato. Il *client* effettuata una *POST* all'url https://acme-v02.api.letsencrypt.org/acme/authz/8621zDGUjKxqTsdMa2PPfVgDyYFRPPu3MpaOjKr99js utilizzando il seguente oggetto, così da poter accedere alle sfide che vanno completate per poter finalizzare la richiesta di emissione:

```
/*Richiesta codificata*/
{
  "protected":"eyJub25jZSI6ICJVeTkwQ0huaTY4d0x5S21ZeUltQU5fbklXd1JPX2dTMjBlUm5Q
    TlpKOWcwIiwgInVybCI6ICJodHRwczovL2FjbWUtdjAyLmFwaS5sZXRzZW5jcnlwdC5vcmcvYWN
    tZS9hdXRoei84NjIxekRHVWpLeHFUc2RNYTJQUGZWZOR5WUZSUFB1MO1wYU9qS3I5OWpzIiwgIm
```

```
    tpZCI6ICJodHRwczovL2FjbWUtdjAyLmFwaS5sZXRzZW5jcnlwdC5vcmcvYWNtZS9hY2N0LzYxN
    zA5MDQ5IiwgImFsZyI6ICJSUzI1NiJ9",
  "payload": "",
  "signature":"UHOfKDNRVSyvP3Tq_Wv8f6bchMGbW_dA9eIORjukxTBUACVZdMuxALJPH3Xxe74F
    aKaoUkedf-tbQku1pGXZFkUuEHRJv9dooFWNE17mm5vN9DXmc2JhkJ2VsiEiBNo-bCapIFnMHvK
    htY6gkhB989Z41dnVGgpCvxlMPy_WDCl63dznevrDEmjL2L4xV0zb2PnZ-DA_wZemH6eUAoEYln
    Pa98nEvSu3mzPqY-M9h0zUTZgU6JZrmsLKRDWDr6gVFRQITavm12WEISwgVKpvaeVLxeXEyWJ0-
    daDEOwOCzlnNgvCvjwwGTsDa0fRMIJRoRyO1Is90-cPVd6mn4MnQg"
}

/*Richiesta decodificata*/
{
  "protected":{
    "nonce":"Uy90CHni68wLyKiYyImAN_nIWwRO_gS20eRnPNZJ9g0",
    "url":"https://acme-v02.api.letsencrypt.org/acme/authz/8621zDGUjKxqTsdMa2PP
        fVgDyYFRPPu3MpaOjKr99js",
    "kid":"https://acme-v02.api.letsencrypt.org/acme/acct/61709049",
    "alg":"RS256"
  },
  "payload": "",
  "signature":"UHOfKDNRVSyvP3Tq_Wv8f6bchMGbW_dA9eIORjukxTBUACVZdMuxALJPH3Xxe74F
    aKaoUkedf-tbQku1pGXZFkUuEHRJv9dooFWNE17mm5vN9DXmc2JhkJ2VsiEiBNo-bCapIFnMHvK
    htY6gkhB989Z41dnVGgpCvxlMPy_WDCl63dznevrDEmjL2L4xV0zb2PnZ-DA_wZemH6eUAoEYln
    Pa98nEvSu3mzPqY-M9h0zUTZgU6JZrmsLKRDWDr6gVFRQITavm12WEISwgVKpvaeVLxeXEyWJ0-
    daDEOwOCzlnNgvCvjwwGTsDa0fRMIJRoRyO1Is90-cPVd6mn4MnQg"
}
```


La risposta del *server* fornisce le informazioni sulle sfide come atteso:


```
{
  "identifier": {
    "type": "dns",
    "value": "ubuntuapache.eastus.cloudapp.azure.com"
  },
  "status": "pending",
  "expires": "2019-07-28T17:35:32Z",
  "challenges": [
    {
      "type": "dns-01",
      "status": "pending",
      "url":
          "https://acme-v02.api.letsencrypt.org/acme/challenge/8621zDGUjKxqT
        sdMa2PPfVgDyYFRPPu3MpaOjKr99js/18558014733",
      "token": "L-JiAxGiHcP968c6_IDjmADfdIfPWP7kSrT-FOqzIlw"
    },
    {
      "type": "http-01",
      "status": "pending",
      "url":
          "https://acme-v02.api.letsencrypt.org/acme/challenge/8621zDGUjKxqT
        sdMa2PPfVgDyYFRPPu3MpaOjKr99js/18558014734",
      "token": "-IbjwLr4y4yk6XKDFODB-FETe5_Tlb2tosR4IYRm7WM"
    },
    {
      "type": "tls-alpn-01",
      "status": "pending",
```

```
      "url":
          "https://acme-v02.api.letsencrypt.org/acme/challenge/8621zDGUjKxqT
        sdMa2PPfVgDyYFRPPu3MpaOjKr99js/18558014735",
      "token": "bKaLyKwaELnV7twgBreK7-1Htpmcn8sbWn4iDHy4ATI"
    }
  ]
}
```

Si evince dal log che il *client* ha deciso di rispondere alla sfida *http-01 challenge for ubuntuapache.eastus.cloudapp.azure.com* aggiungendo al file di configurazione del *server* le seguenti linee:

```
RewriteRule ^/\.well-known/acme-challenge/([A-Za-z0-9-_=]+)$
    /var/lib/letsencrypt/http_challenges/$1 [END]

<Directory /var/lib/letsencrypt/http_challenges>
    Require all granted
</Directory>
<Location /.well-known/acme-challenge>
    Require all granted
</Location>
```

Viene dunque inviata una notifica al *server* per consentirgli di effettuare la validazione alla sfida, effettuando una *POST* al link https://acme-v02.api.letsencrypt.org/acme/challenge/8621zDGUjKxqTsdMa2PPfVgDyYFRPPu3MpaOjKr99js/18558014734 con il seguente contenuto:

```
/*Richiesta codificata*/
{
  "protected":"eyJub25jZSI6ICJHWkZvNHFWOVFoTXI0bWZaQ0dIN0NYbzdyQ3lSYXdiblAzZGk5
    R3U1VE5VIiwgInVybCI6ICJodHRwczovL2FjbWUtdjAyLmFwaS5sZXRzZW5jcnlwdC5vcmcvYWN
    tZS9jaGFsbGVuZ2UvODYyMXpER1VqS3hxVHNkTWEyUFBmVmdEeVlGUlBQdTNNcGFPaktyOTlqcy
    8xODU1ODAxNDczNCIsICJraWQiOiAiaHR0cHM6Ly9hY211LXYwMi5hcGkubGV0c2VuY3J5cHQub
    3JnL2FjbWUvYWNjdC82MTcwOTA0OSIsICJhbGciOiAiUlMyNTYifQ",
  "payload":
      "ewogICJ0eXBlIjogImh0dHAtMDEiLCAKICAicmVzb3VyY2UiOiAiY2hhbGxlbmdlI
    gp9",
  "signature":"IB3K15lqwqYYDFikj5REI5IgGnD365VTp5wdY_5GvuSzxWDbt3hmtwe56VlhZQzJ
    5mMDIUyt9W3D67NKJ-yJK9C1SSWPGcn_b_dsnJGq0g16eiN0Sei9R2HaH-VuSeB1CPLuoRZm_Dc
    zwSHqVoYXRT-a7olItPSYaNLrMpcQdKxUaOjvM7nuJEm38OKL-5p14uE92_cGpUeqHB-fHkjsXW
    P9em_Q7tWqcf1I4gbPPc1y4TEO8nGbr8820WjVYxpkRY14foozQOXwl7QPSh_UHc-IuVhFkEUTc
    2Fvhp6Zks1pz9EsDnmJDbJ2giE2zSAVQeduJwdJEsDVqrnPHg9ZYA"
}

/*Richiesta decodificata*/
{
  "protected":{
    "nonce":"GZFo4qV9QhMr4mfZCGH7CXo7rCyRawbnP3di9Gu5TNU",
    "url":"https://acme-v02.api.letsencrypt.org/acme/challenge/8621zDGUjKxqTsdM
        a2PPfVgDyYFRPPu3MpaOjKr99js/18558014734",
    "kid":"https://acme-v02.api.letsencrypt.org/acme/acct/61709049",
    "alg":"RS256"
  },
  "payload":{
      "type": "http-01",
      "resource": "challenge"
  },
  "signature":"IB3K15lqwqYYDFikj5REI5IgGnD365VTp5wdY_5GvuSzxWDbt3hmtwe56VlhZQzJ
```

```
    5mMDIUyt9W3D67NKJ-yJK9C1SSWPGcn_b_dsnJGq0g16eiN0Sei9R2HaH-VuSeB1CPLuoRZm_Dc
    zwSHqVoYXRT-a7olItPSYaNLrMpcQdKxUaOjvM7nuJEm38OKL-5p14uE92_cGpUeqHB-fHkjsXW
    P9em_Q7tWqcf1I4gbPPc1y4TEO8nGbr8820WjVYxpkRY14foozQOXwl7QPSh_UHc-IuVhFkEUTc
    2Fvhp6Zks1pz9EsDnmJDbJ2giE2zSAVQeduJwdJEsDVqrnPHg9ZYA"
}
```


Il *client* effettua dunque una nuova richiesta di accesso alle sfide, ottenendo un risultato diverso dal *server*:


```
{
  "identifier": {
    "type": "dns",
    "value": "ubuntuapache.eastus.cloudapp.azure.com"
  },
  "status": "valid",
  "expires": "2019-08-20T17:35:36Z",
  "challenges": [
    {
      "type": "dns-01",
      "status": "pending",
      "url":
          "https://acme-v02.api.letsencrypt.org/acme/challenge/8621zDGUjKxqT
        sdMa2PPfVgDyYFRPPu3MpaOjKr99js/18558014733",
      "token": "L-JiAxGiHcP968c6_IDjmADfdIfPWP7kSrT-FOqzIlw"
    },
    {
      "type": "http-01",
      "status": "valid",
      "url":
          "https://acme-v02.api.letsencrypt.org/acme/challenge/8621zDGUjKxqT
        sdMa2PPfVgDyYFRPPu3MpaOjKr99js/18558014734",
      "token": "-IbjwLr4y4yk6XKDFODB-FETe5_Tlb2tosR4IYRm7WM",
      "validationRecord": [
        {
          "url": "http://ubuntuapache.eastus.cloudapp.azure.com/.well-known/
            acme-challenge/-IbjwLr4y4yk6XKDFODB-FETe5_Tlb2tosR4IYRm7WM",
          "hostname": "ubuntuapache.eastus.cloudapp.azure.com",
          "port": "80",
          "addressesResolved": [
            "104.211.30.207"
          ],
          "addressUsed": "104.211.30.207"
        }
      ]
    },
    {
      "type": "tls-alpn-01",
      "status": "pending",
      "url":
          "https://acme-v02.api.letsencrypt.org/acme/challenge/8621zDGUjKxqT
        sdMa2PPfVgDyYFRPPu3MpaOjKr99js/18558014735",
      "token": "bKaLyKwaELnV7twgBreK7-1Htpmcn8sbWn4iDHy4ATI"
    }
  ]
}
```


Una volta ricevuta una risposta positiva, il *client* può procedere con la finalizzazione della richiesta.

4.7 Revoca di un certificato

Il corpo della chiamata effettuata conterrà il vero e proprio certificato, nella versione codificata in *base64* del formato *DER*[10] e le motivazioni di richiesta di revoca. Queste ultime non sono obbligatorie, ma potrebbero risultare utili come informazione aggiuntiva per l'aggiornamento di *OCSP*[11] e della *CRL*[12].

4.7.1 Specifiche del protocollo

Diversamente dalle altre richieste, questa può essere firmata utilizzando sia la chiave privata relativa al certificato, sia la chiave relativa all'account.

```
POST /acme/revoke-cert HTTP/1.1
Host: example.com
Content-Type: application/jose+json
{
    "protected": base64url({
            "alg": "ES256",
            "kid": "https://example.com/acme/acct/evOfKhNU60wg",
            "nonce": "JHb54aT_KTXBWQOzGYkt9A",
            "url": "https://example.com/acme/revoke-cert"
    }),
    "payload": base64url({
            "certificate": "MIIEDTCCAvegAwIBAgIRAP8...",
            "reason": 4
    }),
    "signature": "Q1bURgJoEslbD1c5...3pYdSMLio57mQNN4"
}
```

Figura 4.17. POST-Request for certificate revocation - signed using the account key

[10] Distinguished Encoding Rules.

[11] Online Certificate Status Protocol.

[12] Certificate Revocation List.

```
POST /acme/revoke-cert HTTP/1.1
Host: example.com
Content-Type: application/jose+json
{
    "protected": base64url({
            "alg": "RS256",
            "jwk": /* certificate's public key */,
            "nonce": "JHb54aT_KTXBWQOzGYkt9A",
            "url": "https://example.com/acme/revoke-cert"
    }),
    "payload": base64url({
            "certificate": "MIIEDTCCAvegAwIBAgIRAP8...",
            "reason": 1
    }),
    "signature": "Q1bURgJoEslbD1c5...3pYdSMLio57mQNN4"
}
```

Figura 4.18. POST-Request for certificate revocation - signed using the private key

Oltre ad effettuare controlli sulla validità delle chiavi utilizzate per la firma della richiesta, il *server* dovrà accertarsi che l'account che richiede la revoca del certificato, sia lo stesso che ne ha richiesto l'emissione o che abbia tutte le autorizzazioni per tutti gli identificatori del caso.

```
HTTP/1.1 200 OK
Replay-Nonce: IXVHDyxIRGcTE0VSblhPzw
Content-Length: 0
Link: <https://example.com/acme/directory>;rel="index"

--- or ---

HTTP/1.1 403 Forbidden
Replay-Nonce: lXfyFzi6238tfPQRwgfmPU
Content-Type: application/problem+json
Content-Language: en
Link: <https://example.com/acme/directory>;rel="index"
{
    "type": "urn:ietf:params:acme:error:unauthorized",
    "detail": "No authorization provided for name example.org"
}
```

Figura 4.19. POST-Response for certificate revocation

4.7.2 Server Apache

Come esplicitato nel protocollo, la richiesta di revoca di un certificato può essere firmata utilizzando la chiave privata associata al certificato o quella relativa all'account. I test sono stati eseguiti utilizzando la chiave privata relativa al certificato. Di seguito l'output ottenuto effettuando richiesta *POST* all'*URL* https://acme-v01.api.letsencrypt.org/acme/revoke-cert:

```
/*Richiesta codificata*/
{
  "protected":
     "eyJub25jZSI6ICIwMTAyWDlnRkJ6aUh6VlVCQkp4ekZiNGdEak5ITkk4a3VTUlB
    tYkZKMzB6cDM3RSIsICJhbGciOiAiUlMyNTYiLCAiandrIjogeyJlIjogIkFRQUIiLCAia3R5Ij
    ogIlJTQSIsICJuIjogIjQwQVYxMOZ1ZWk5bWUyeVAxbTZpMzZlY2plWEdCNFNwMWt0X1ZOODJ0M
    FBIUmpIY2l5ampyNmtIcml0U3lZbFBjNHp6ZEdvbUFrbWJMWVBzaUI3RlhNVHRXcHo5dGpmb09D
    MnVyNO1HZWs3Uk9wMmFacWdiWVlDRHpYdlhZcklrMThEQ2hxZng4RmlzckVFcGFyZmIzdk8yZU5
    pTOo2cHJwVzdYNGJSWlI4Q3JQNFBFQTJtemhsVlJ6UOpQNjFpMlFFSms3RktwR3QwUnVOQV9jd1
    VHWHZRd3Bsd1FTbGg4dndWLXlTb19uSlZlWkdmY2hWVldwdVAtRDlQbUlLdUhOTGloUlp3QzhYM
    3VaUlZLaFRyRORzT2MwbHlRUFl3cDFUUHZta0lqQWZvWU9PbnNFbnZOaHBUb2Z4Y1dNRElFWWJI
    N1lKQk5NN1RZUmtsNEFEaENWUSJ9fQ",
  "payload":
     "ewogICJyZWFzb24iOiAwLCAKICAicmVzb3VyY2UiOiAicmV2b2tlLWNlcnQiLCAKI
    CAiY2VydGlmaWNhdGUiOiAiTUlJRmd6QONCR3VnQXdJQkFnSVNCREFLcnVaZl9pVi14QW1oQUtF
    R19Kc0lNQTBHQ1NxR1NJYjNEUUVCQ3dVQU1Fb3hDekFKQmdOVkJBWVRBbFZUTVJZd0ZBWURWUVF
    LRXcxTVpYUW5jeUJGYm1OeWVYQjBNU013SVFZRFZRUURFeHBNWlhRbmN5QkZibU55ZVhCMElFRj
    FkR2h2Y21sMGVTQllNekFlRncweE9UQTVNVFV3TnpRM01EVmFGdzB4T1RFeU1UUXdOelEzTURWY
    U1ERXhMekF0QmdOVkJBTVRKblZpZFc1MGRXRndZV05vWlM1bFlYTjBkWE11WTJ4dmRXUmhjSEF1
    WVhwMWNtVXVZMjlOTUlJQklqQU5CZ2txaGtpRzl3MEJBUUVGQUFPQ0FRQEFNSUlCQ2dLQ0FRRUE
    OMEFWMTNGdWVpOW1lMnlQMWO2aTM2ZWNqZVhHQjRTcDFrdF9WTjgydDBQSFJqSGNpeWpqcjZrSH
    JpdFN5WWxQYzR6emRHb21Ba21iTFlQc2lCNOZYTVROV3B6OXRqZm9PQzJ1cjdNR2VrN1JPcDJhW
    nFnYllZQOR6WHZYWXJJazE4RENocWZ4OEZpc3JFRXBhcmZiM3ZPMmVOaU9KNnBycFc3WDRiUlpS
    OENyUDRQRUEybXpobFZSelNKUDYxaTJRRUprNOZLcEdOMFJ1TkFfY3dVR1h2UXdwbHdRU2xoOHZ
    3Vi15U29fbkpWZVpHZmNoVlZXcHVQLUQ5UG1JS3VITkxpaFJadOM4WDN1WlJWS2hUckdEc09jMG
    x5UVBZd3AxVFB2bWtJakFmb1lPT25zRW52TmhwVG9meGNXTURJRVliSDdZSkJOTTdUWVJrbDRBR
    GhDVlFJREFRQUJvNElDZWpDQOFuWXdEZ1lEVlIwUEFRSF9CQVFEQWdXZO1CMEdBMVVkSlFRVO1C
    UUdDQ3NHQVFVRkJ3TUJCZ2dyQmdFRkJRYORBakFNQmdOVkhSTUJBZjhFQWpBQU1CMEdBMVVkRGd
    RVOJCUnNCdGl5akpfS2NQUjM3OGJwUWNQNXRjTVRPekFmQmdOVkhTTUVHREFXZOJTb1NtcGpCSD
    NkdXViUk9iZW1SV1h2ODZqc29UQnZCZ2dyQmdFRkJRY0JBUVJqTUdFd0xnWUlLd1lCQlFVSE1BR
    OdJbWgwZEhBNkx5OXZZMO53TG1sdWRDMTRNeTVzWlhSelpXNWpjbmx3ZEM1dmNtY3dMd1lJS3dZ
    QkJRVUhNQUtHSTJoMGRIQTZMeTlqWlhKMExtbHVkQzEOTXk1c1pYUnpaVzVqY25sd2RDNXZjbWN
    2TURFROExVWRFUVFxTUNpQOpuVmlkVzUwZFdGd1lXTm9aUzVsWVhOMGRYTXVZMnh2ZFdSaGNIQX
    VZWHAxY21VdVkyOXRNRXdHQTFVZElBUkZNRU13QOFZR1oORU1BUUlCTURjR0N5c0dBUVFCZ3Q4V
    EFRRUJNQ2d3SmdZSUt3WUJCUVVIQWdFV0dtaDBkSEE2THk5amNITXViR1YwYzJWdVkzSjVjSFF1
    YjNKbk1JSUJBd1lLS3dZQkJBSFdlUUlFQWdTQjlBUOI4UUR2QUhZQWRIN2FnekdOTXhDUklaek9
    KVTlDY01LX19WNUNJQWpHTnpWNTVoQjd6RllBQUFGdE5CcW95Z0FBQkFNQVJ6QkZBaUFEVU5mXy
    1sMF81QTh6UjZqMGN1dlF0YlZIR1ctMnBoSVlSUHcwb1pUeUFRSWhBTXNRaER6VE1yNjM1Q2luT
    HdHVk1Fc29TQ2ppWHBEaWdiQkhSTm1saOY1MkFIVUFZXOximVnN3pDelBDM0tFSjFkck02UO5Z
    WGVQdlhXbU9MSEhhRlJMMkkwQUFBRnROQnFvd1FBQUJBTUFSakJFQWlBMEc5T1gxODRTbkNqWkM
    1Mlo5aFE5bkJjQnl6UOd5eEVmTF9nT2pHWjZOQUlnVk9USmxSUWF5SktMdWN0QThFQTN6ckhYYU
    1EYzVvbVYyTOQwcGtuQTNGRXdEUVlKS29aSWh2YO5BUUVMQlFBRGdnRUJBSkE1bTFBaTNzV3VLc
    mFzUmxwODBmWVVsMlZGSGO1dlVFTlUzZ2pqVC1mUXM5ZTBqc2VOYl9udElKRTROV2tDQTZKMUdm
    ZGFXVWdHZOhkWDBoT1ZWMmJ4MTNDSFI5cl8xNXFtS2ZfZHFONlNyc1VwUWIzbW1nRkZZSnBhNk4
    telBkejR4bnpJRkQ1ME5XaHd6Tm9ab1ptRORpbVIyaDNSX2puTHJGbk0zSFFOaHZUQURXdnhsSF
    FuUk4xc2VLcHdtcjFRZ3phTlNZangzYzItcHdId1o5aVJtZHAOLWhwM2QyOGtwb2hXVl9RT3BQd
    0tDNkJ6Y1lsWDI3TUZWbEZBaTdkdHZGcGVmSTk1ck9rQjBJYjlja1ZSc2R5Rk9KamlXOHV1STAw
    UWt3QUhUYTd2ekNHMDA1RDhONHp5ZEF1ZGxOVzJkamU2UTF6OXg4OUlncm5EQUZFIgp9",
  "signature":
     "cquWXa7Nnw1lR19Amk_539cr6Cbo5gQ1VDN7SNPralczA6fk-0ECWlatmhFleSe
    zLdZMTJ_p5Ke83iOht-CsOP3AE4Mkm_UCECIBfsTALz-e2t77lyJMkZtm3oa0A2XdxzkgrRk3LS
    8WqxRiQlomLL9bCyT4jJ6MRJUrMqyQsxL3DIDCiHwJ78NB8fAGpfnKQpx8o4u0Lkd9JdXXCQRTI
    -0q0F4nTOk7leUdZGPZQSTCjcAzxxTWk4nsG8A9eQKDJgPeOGcVZhzNkcOD4DRNtN1CqnmzY-II
    9AoT1cRDzaVNoqBXIOEqBVOFmmog-lkdPixZ7qCbQb1lw9FOfmiAXQ"
}
```

```
/*Richiesta decodificata*/
{
  "protected": {
    "nonce":"0102X9gFBziHzVUBBJxzFb4gDjNHNI8kuSRPmbFJ30zp37E",
    "alg":"RS256",
    "jwk":{
        "e":"AQAB",
        "kty":"RSA",
        "n":"40AV13Fuei9me2yP1m6i36ecjeXGB4Sp1kt_VN82t0PHRjHciyjjr6kHritSyYlPc4
            zzdGomAkmbLYPsiB7FXMTtWpz9tjfoOC2ur7MGek7ROp2aZqgbYYCDzXvXYrIk18DCh
            qfx8FisrEEparfb3vO2eNiOJ6prpW7X4bRZR8CrP4PEA2mzhlVRzSJP61i2QEJk7FKp
            Gt0RuNA_cwUGXvQwplwQSlh8vwV-ySo_nJVeZGfchVVWpuP-D9PmIKuHNLihRZwC8X3
            uZRVKhTrGDsOc0lyQPYwp1TPvmkIjAfoYOOnsEnvNhpTofxcWMDIEYbH7YJBNM7TYRk
            l4ADhCVQ"
    }
  },
  "payload": {
    "reason": 0,
    "resource": "revoke-cert",
    "certificate":
        "MIIFgzCCBGugAwIBAgISBDAKruZf_iV-xAmhAKEG_JsIMA0GCSqGSIb3DQE
        BCwUAMEoxCzAJBgNVBAYTAlVTMRYwFAYDVQQKEw1MZXQncyBFbmNyeXB0MSMwIQYDVQQDEx
        pMZXQncyBFbmNyeXB0IEF1dGhvcml0eSBYMzAeFw0xOTA5MTUwNzQ3MDVaFw0xOTEyMTQwN
        zQ3MDVaMDExLzAtBgNVBAMTJnVidW50dWFwYWNoZS5lYXN0dXMuY2xvdWRhcHAuYXp1cmUu
        Y29tMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA40AV13Fuei9me2yP1m6i36e
        cjeXGB4Sp1kt_VN82t0PHRjHciyjjr6kHritSyYlPc4zzdGomAkmbLYPsiB7FXMTtWpz9tj
        foOC2ur7MGek7ROp2aZqgbYYCDzXvXYrIk18DChqfx8FisrEEparfb3vO2eNiOJ6prpW7X4
        bRZR8CrP4PEA2mzhlVRzSJP61i2QEJk7FKpGt0RuNA_cwUGXvQwplwQSlh8vwV-ySo_nJVe
        ZGfchVVWpuP-D9PmIKuHNLihRZwC8X3uZRVKhTrGDsOc0lyQPYwp1TPvmkIjAfoYOOnsEnv
        NhpTofxcWMDIEYbH7YJBNM7TYRkl4ADhCVQIDAQABo4ICejCCAnYwDgYDVR0PAQH_BAQDAg
        WgMB0GA1UdJQQWMBQGCCsGAQUFBwMBBggrBgEFBQcDAjAMBgNVHRMBAf8EAjAAMB0GA1UdD
        gQWBBRsBtiyjJ_KcPR378bpQcP5tcMTOzAfBgNVHSMEGDAWgBSoSmpjBH3duubRObemRWXv
        86jsoTBvBggrBgEFBQcBAQRjMGEwLgYIKwYBBQUHMAGGImh0dHA6Ly9vY3NwLmludC14My5
        sZXRzZW5jcnlwdC5vcmcwLwYIKwYBBQUHMAKGI2h0dHA6Ly9jZXJ0LmludC14My5sZXRzZW
        5jcnlwdC5vcmcvMDEGA1UdEQQqMCiCJnVidW50dWFwYWNoZS5lYXN0dXMuY2xvdWRhcHAuY
        Xp1cmUuY29tMEwGA1UdIARFMEMwCAYGZ4EMAQIBMDcGCysGAQQBgt8TAQEBMCgwJgYIKwYB
        BQUHAgEWGmh0dHA6Ly9jcHMubGV0c2VuY3J5cHQub3JnMIIBAwYKKwYBBAHWeQIEAgSB9AS
        B8QDvAHYAdH7agzGtMxCRIZzOJU9CcMK__V5CIAjGNzV55hB7zFYAAAFtNBqoygAABAMARz
        BFAiADUNf_-l0_5A8zR6j0cuvQtbVHGW-2phIYRPw0oZTyAQIhAMsQhDzTMr635CinLwGVM
        EsoSCjiXpDigbBHRNmlkF52AHUAY_Lbzeg7zCzPC3KEJ1drM6SNYXePvXWmOLHHaFRL2I0A
        AAFtNBqowQAABAMARjBEAiA0G9OX184SnCjZC52Z9hQ9nBcByzSGyxEfL_gOjGZ6NAIgVOT
        JlRQayJKLuctA8EA3zrHXaMDc5omV2OD0pknA3FEwDQYJKoZIhvcNAQELBQADggEBAJA5m1
        Ai3sWuKrasRlp80fYUl2VFHm5vUENU3gjjT-fQs9e0jsetb_ntIJE4tWkCA6J1GfdaWUgGg
        HdX0hOVV2bx13CHR9r_15qmKf_dqt6SrsUpQb3mmgFFYJpa6N-zPdz4xnzIFD50NWhwzNoZ
        oZmGDimR2h3R_jnLrFnM3HQNhvTADWvxlHQnRN1seKpwmr1QgzaNSYjx3c2-pwHwZ9iRmdp
        4-hp3d28kpohWV_QOpPwKC6BzcYlX27MFVlFAi7dtvFpefI95rOkB0Ib9ckVRsdyFOJjiW8
        uuI00QkwAHTa7vzCG005D8t4zydAudlNW2dje6Q1z9x89IgrnDAFE"
  },
  "signature":
      "cquWXa7Nnw1lR19Amk_539cr6Cbo5gQ1VDN7SNPralczA6fk-0ECWlatmhFleSe
    zLdZMTJ_p5Ke83i0ht-CsOP3AE4Mkm_UCECIBfsTALz-e2t77lyJMkZtm3oa0A2XdxzkgrRk3LS
    8WqxRiQlomLL9bCyT4jJ6MRJUrMqyQsxL3DIDCiHwJ78NB8fAGpfnKQpx8o4u0Lkd9JdXXCQRTI
    -0q0F4nTOk7leUdZGPZQSTCjcAzxxTWk4nsG8A9eQKDJgPeOGcVZhzNkcOD4DRNtN1CqnmzY-II
    9AoT1cRDzaVNoqBXI0EqBVOFmmog-lkdPixZ7qCbQb1lw9FOfmiAXQ"
}
```


In seguito a risposta affermativa da parte del *server*, è stata effettuata una nuova prova di revoca del medesimo certificato ottenendo la seguente risposta da parte del *server*:

```
HTTP 409
Content-Length: 100
Expires: Mon, 16 Sep 2019 22:04:10 GMT
Server: nginx
Connection: close
Pragma: no-cache
Cache-Control: max-age=0, no-cache, no-store
Date: Mon, 16 Sep 2019 22:04:10 GMT
Content-Type: application/problem+json
Replay-Nonce: 0102N0490ShSCGBnXgAgtoziQDYW_6aJcF88Rn95sIw9fc4

{
  "type": "urn:acme:error:malformed",
  "detail": "Certificate already revoked",
  "status": 409
}
```

Capitolo 5

Adattamento implementativo del software

Nei capitoli precedenti è stato analizzato in maniera approfondita il protocollo *ACME*, che definisce le regole per la comunicazione con *Let's encrypt*, e sono stati dettagliati i risultati dei test effettuati utilizzando il software *Certbot*, che permette la comunicazione con *certification authority* favorendo l'automazione del processo di richiesta e successiva gestione dei certificati.

In questo capitolo si discuterà delle modifiche apportate al codice del software *Certbot*, in modo che quest'ultimo possa essere in grado di gestire i certificati in maniera più dinamica.

5.1 Certbot

Certbot è un software open source offerto da *Eletronic Frontier Foundation*. Esso permette, in qualità di *client*, di comunicare in maniera semplice con *Let's Encrypt* al fine di ottenere un certificato, per procedere poi con la sua installazione sul *web server* per il quale viene effettuata richiesta di emissione. La sua implementazione rispetta le regole definite nel protocollo *ACME* per quanto riguarda la comunicazione con la *certification authority*, consentendo quindi una gestione automatica dei certificati. Il software è compatibile solo con sistemi operativi *Unix-based* e richiede che sul *web server* sia installata una versione di python pari a 2.7, o 3.4 e superiori. Le uniche interazioni richieste da parte di un amministratore di sistema sono relative alla sua prima installazione e avvio. La prima avviene tramite l'utilizzo del comando *sudo apt-get install certbot*, mentre la prima esecuzione avviene utilizzando il comando *certbot –apache* o *certbot –nginx* in base al *web server* di riferimento per il quale si richiede l'emissione di un certificato. Durante la fase di avvio del software, l'interazione da parte dell'utente è ancora richiesta al fine di fornire un indirizzo email valido e per poter accettare i termini di utilizzo dei servizi esposti dalla *certification authority*. Una volta registrato l'account, richiesto e installato il certificato correttamente, il software si occupa automaticamente del suo mantenimento. Non viene dunque più richiesta alcuna interazione da parte dell'utente, ad eccezione di casi particolari in cui l'utente stesso non voglia revocare il certificato o disattivare l'account.

5.2 Analisi dell'implementazione

La versione *1.0.0* del software *certbot* da cui si è partiti e che attualmente si può installare sui *server*, consente esclusivamente la gestione di certificati associati a chiavi generate internamente dal software stesso. Lo scopo della tesi è stato quello di modificare il codice esistente in modo che si possano utilizzare anche chiavi generate esternamente.

La nuova versione del software permette di continuare a lavorare sfruttando la vecchia metodologia, quindi utilizzando chiavi generate dal software, ma consente anche di utilizzare chiavi generate esternamente. La modifica è applicabile sia relativamente alla chiave da associare all'account, sia per quella da associare al certificato.

La discriminazione tra i vari flussi possibili, avviene sulla base degli input ricevuti dal software e per poter gestire questi nuovi casi, sono stati introdotti nuovi parametri di input possibili che permettono di parsificare correttamente il nome del file ricevuto

- *account=absolute_path_file_name.pem* - se presente, garantisce che il nuovo account registrato presso la *certification authority* verrà associato alla chiave passata in input;
- *certificate=absolute_path_file_name.pem* - se presente, garantisce che il nuovo certificato validato dalla *certification authority* verrà associato alla chiave passata in input.

Ovviamente se uno e/o entrambi i parametri di input non saranno presenti, il software continuerà a generare tutte le chiavi necessarie internamente.

Durante l'analisi del codice del software, si è notato che la chiave generata e successivamente letta, viene mantenuta in variabili locali fino a quando il flusso non termina in maniera corretta con un riscontro positivo da parte della *certification authority*. Solo dopo aver registrato l'account correttamente o dopo aver validato correttamente un certificato, le chiavi coinvolte, vengono salvate permanentemente nel file system del server. Questo ci ha permesso di mantenere il più e possibile le *naming convention* utilizzate dal software nonostante l'utente possa aver scelto un nome qualsiasi per la chiave passata in input. La chiave viene salvata temporaneamente in una cartella di sistema, tra quelle relative a *Let's Encrypt*. Il contenuto viene letto ed utilizzato per tutta la durata della richiesta, una volta ottenuto un riscontro positivo da parte della *certification authority*, la stessa chiave viene spostata in una cartella di archivio e il suo contenuto viene salvato utilizzando la sintassi di default già presente nel software. Si è scelto di mantenere un archivio delle chiavi originariamente passate in input al software con il semplice scopo di evitare il riutilizzo della stessa chiave più volte.

Vediamo adesso più nel dettaglio come agisce il nuovo flusso. All'interno del file *certbot/main.py* è stata implementata la logica che permette il *parsing* corretto dei nuovi input. La prima modifica, mostrata nella porzione di codice seguente, ha riguardato l'estrazione dei nuovi input

```
for var in cli_args:
    if 'account=' in var:
        account_name = var['account='.__len__():None]
        logger.debug("Account key file name: %s", account_name)
    elif 'certificate=' in var:
        certificate_name = var['certificate='.__len__():None]
        logger.debug("Certificate key file name: %s", certificate_name)
    else:
        final_args.append(var)
```

Per poter tenere traccia dei nuovi input e mantenere il corretto comportamento del software, sono state aggiunte quattro nuove proprietà. Queste ultime sono state inserite nella classe che contiene tutte le informazioni di configurazione necessarie alla corretta esecuzione dei flussi. Nello specifico la classe si chiama *NamespaceConfig* ed è presente all'interno del file *certbot/configuration.py*. Due proprietà sono di tipo *booleano* e vengono utilizzate per verificare la presenza o meno di una chiave custom. Esse sono rispettivamente *account_custom* e *certificate_custom* e servono a verificare quale delle due opzioni è stata utilizzata. Le altre due proprietà sono stringhe che contengono eventualmente il path assoluto da cui leggere le chiavi. Nel listato seguente viene evidenziato il costruttore della classe dove vengono assegnati i valori di default alle varie proprietà.

```
...
def __init__(self, namespace):
    object.__setattr__(self, 'namespace', namespace)

    ...

    self.account_custom = False
    self.certificate_custom = False
    self.account_file_name = None
    self.certificate_file_name = None

    ...
...
```

Una volta valorizzate tutte le proprietà standard già presenti nel software, prima di poter procedere con l'esecuzione della funzionalità richiesta, vengono verificati i nuovi input. Al fine di estrarre le informazioni necessarie e valorizzare le proprietà dovute, è stato creato un metodo, *check_customization*, nel file *certbot/main.py*. Di seguito i dettagli

```
""" Metodo utilizzato per il controllo degli input """

def check_customization(account_name, certificate_name, config, cli_args):
    try:
        if account_name is not None and config.accounts_dir is not None:
            config.namespace.account_file_name =
                copy_and_rename_key('account', account_name,
                config.accounts_dir, cli_args)
            config.namespace.account_custom = True
        if certificate_name is not None and config.key_dir is not None:
            config.namespace.cert_file_name =
                copy_and_rename_key('certificate', certificate_name,
                config.key_dir, cli_args)
            config.namespace.certificate_custom = True
    except Exception as e:
        return e

    return config

""" Metodo che effettua la copia dei file """

def copy_and_rename_key(key_type, source_path, dest_path, args):
    import shutil

    file_name = source_path.rsplit('/', 1)[1]
    file_exists = True
    dest_full_path = dest_path + '/' + file_name

    if key_type == 'account':
        existing_file_path = dest_path + '/' + PARSED_KEY_DIR_NAME
    elif key_type == 'certificate':
        existing_file_path = dest_path + '/' + PARSED_CERT_DIR_NAME

    """ Creating archive folder for key/certificate if it doesn't exist"""
    if not os.path.exists(existing_file_path):
        try:
            from certbot.compat import filesystem
            filesystem.makedirs(existing_file_path, 0o777)
```

```
        except OSError:
            print("Creation of the directory %s failed" % existing_file_path)
            raise

    """ Set correct path to check into archive folder """
    existing_file_path = existing_file_path + '/' + file_name
    if not os.path.exists(dest_full_path) and not
        os.path.exists(existing_file_path):
        shutil.copy(source_path, dest_path)
        file_exists = False
    elif os.path.exists(dest_full_path) and not
        os.path.exists(existing_file_path):
        file_exists = False

    if file_exists and not args.__contains__('unregister'):
        raise Exception("File " + file_name + " already exists in one of the
            following path: \n- " + existing_file_path
                        + "\n- " + dest_path + "\nYou cannot use it .
                            Please check if you typed it correctly")

    return dest_full_path
```

Come si può notare nella porzione di codice precedente, oltre a valorizzare tutte le proprietà necessarie, il software si occupa di creare eventuali path mancanti e di copiare i file relativi alle chiavi in cartelle valide per il software. Nello specifico in caso di chiave relativa ad un account, il file *.pem* verrà copiato nella cartella */etc/letsencrypt/accounts/acme-v02.api.letsencrypt.org/directory/*, mentre se il file *.pem* è relativo ad un certificato, esso verrà copiato nella cartella */etc/letsencrypt/-keys/*. Prima di effettuare una copia, viene fatta una verifica sulla presenza o meno del file nella cartella.

Una volta terminato il setup relativo all'utilizzo delle chiavi custom, il flusso prosegue seguendo lo standard definito dal protocollo. La modifica fondamentale applicata in maniera estesa ogni qualvolta si procedeva con la generazione di una chiave, è mostrata nel listato seguente, e permette di evitare la generazione di una nuova chiave se passata in input al software.

```
""" Controllo relativo all'account """

if not config.namespace.account_custom:
    rsa_key = generate_private_key(
        public_exponent=65537,
        key_size=config.rsa_key_size,
        backend=default_backend())
elif config.namespace.account_file_name is not None:
    account_name = config.namespace.account_file_name

    with open(account_name, "rb") as key_file:
        rsa_key = serialization.load_pem_private_key(key_file.read(),
                            password=None, backend=default_backend())

""" Controllo relativo al certificato """

if self.config.namespace.certificate_custom
        and self.config.namespace.certificate_file_name is not None:
    with open(self.config.namespace.certificate_file_name, "rb") as key_file:
        key = serialization.load_pem_private_key(key_file.read(),
            password=None,
                backend=default_backend())
```

```
    csr = crypto_util.init_save_csr(key, domains, self.config.csr_dir)
else:
    key = key or crypto_util.init_save_key(self.config.rsa_key_size,
                                           self.config.key_dir)
    csr = crypto_util.init_save_csr(key, domains, self.config.csr_dir)
```

Grazie a questa implementazione sufficientemente semplice, si è riusciti a gestire il software mantenendo il funzionamento di default e integrando la possibilità di input esterni.

Riassumiamo brevemente gli step relativi al nuovo flusso:

1. verifica dell'esistenza del file di input nelle cartelle temporanea e di archiviazione di *Let's Encrypt*. A seconda della presenza o meno del file di input, il codice gestisce i casi nel seguente modo:
   - se il file è presente nella cartella di archiviazione, viene restituito un errore all'utente notificando che il file è già stato utilizzato;
   - se il file è presente solo nella cartella temporanea, viene eseguito lo step 2;
   - se il file non è presente in nessuna cartella, viene effettuata una copia del file nella cartella temporanea e si prosegue con lo step 2.
2. valorizzazione di alcune proprietà della variabile di configurazione che viene utilizzata in tutti i contesti. Questo permette di controllare durante i vari flussi (creazione di un account, richiesta di certificato, rinnovo di un certificato, ...) di verificare se l'utente ha richiesto l'utilizzo di una chiave generata esternamente ed eventualmente procedere con l'accesso alla risorsa;
3. esecuzione del flusso standard descritto nel capitolo 4 saltando la generazione delle chiavi dove presente e utilizzando le chiavi passate in input. Questo termina con la corretta creazione di un nuovo account presso la *certification authority* e/o con il download di un certificato;
4. salvataggio delle chiavi utilizzate per procedere con l'operazione dello step precedente. Nel caso specifico di chiavi esterne, esse verranno salvate nuovamente per non intaccare i flussi di mantenimento e la convenzioni sui nomi;

Capitolo 6

Risultati

In seguito alle modifiche implementative apportate al codice e descritte nel capitolo 5, si è passati ad una fase di test. Lo scopo ultimo è stato quello di creare un account a partire da una chiave generata esternamente, richiedere e ottenere un certificato associato ad una chiave generata esternamente, e nel mantenendo l'automatismo garantito dalla versione standard del software *Certbot*.

I test sono effettuati su due server Linux differenti, installati su macchine virtuali Azure. In uno è stato installato il *web server Apache* e sull'altro il *web server Nginx*. I risultati ottenuti sui due *server* sono i medesimi, riportiamo dunque solo gli esiti relativi al *web server Apache* e validi comunque per il *web server Nginx*.

Prima di procedere con l'analisi delle chiamate effettuate, si specifica che sono stati creati due file con estensione *pem*[1] utilizzando *OpenSSL*[2]. Nello specifico è stato eseguito il seguente comando, una volta per la generazione della chiave da associare all'account, registrato presso la *certification authority* e una volta per la generazione della chiave da associare al certificato

```
openssl genrsa -out name_key.pem 2048
```

Saranno riportate nelle sezioni successive, mantenendo l'ordine di chiamata per quanto possibile, tutte le richieste presenti nel log generato dal software e analizzate già nel capitolo 4 in riferimento ad un'implementazione diversa.

Di seguito una lista dei comandi utilizzati per avviare il software durante le diverse richieste:

- Creazione di un account con relativa richiesta di un certificato:

```
python3 main.py --apache
    account=/home/roberta_sgroi/OpenSSL/test_acc.pem
    certificate=/home/roberta_sgroi/OpenSSL/test_cert.pem
```

- Richiesta emissione di un certificato:

```
python3 main.py --apache
    certificate=/home/roberta_sgroi/OpenSSL/test_cert.pem
```

- Rinnovo di un certificato:

```
python3 main.py renew
```

[1] *Privacy Enhanced Mail.*

[2] Fornisce librerie crittografiche e funzionalità di interfacciamento per TLS, il tutto *open source*.

- Revoca di un certificato:

```
python3 main.py revoke --cert-path
    /etc/letsencrypt/live/apacheserver.eastus.cloudapp.azure.com/cert.pem
```

- Disattivazione di un account:

```
python3 main.py unregister
```

6.1 Directory

All'avvio del software, dopo l'opportuno parsing dei due nuovi input precedentemente non gestiti, si è subito riscontrata una richiesta di accesso alle risorse esposte dalla *certification authority*. Di seguito la risposta del *server* alla richiesta *GET* effettuata all'url https://acme-v02.api.letsencrypt.org/directory

```
HTTP 200
Server: nginx
Date: Sat, 07 Dec 2019 20:44:51 GMT
Content-Type: application/json
Content-Length: 658
Connection: keep-alive
Cache-Control: public, max-age=0, no-cache
X-Frame-Options: DENY
Strict-Transport-Security: max-age=604800

{
  "1AsP1XMgaJQ":
      "https://community.letsencrypt.org/t/adding-random-entries-to-the-directory/33417",
  "keyChange": "https://acme-v02.api.letsencrypt.org/acme/key-change",
  "meta": {
    "caaIdentities": [
      "letsencrypt.org"
    ],
    "termsOfService":
        "https://letsencrypt.org/documents/LE-SA-v1.2-November-15-2017.pdf",
    "website": "https://letsencrypt.org"
  },
  "newAccount": "https://acme-v02.api.letsencrypt.org/acme/new-acct",
  "newNonce": "https://acme-v02.api.letsencrypt.org/acme/new-nonce",
  "newOrder": "https://acme-v02.api.letsencrypt.org/acme/new-order",
  "revokeCert": "https://acme-v02.api.letsencrypt.org/acme/revoke-cert"
}
```

La risposta ottenuta è perfettamente in linea con quanto atteso, sopratutto perchè nessuna modifica riguardante chiavi generate esternamente riguarda questa parte di richieste.

6.2 Nonce

La chiamata successiva a quella di accesso alle risorse, è stata una richiesta *HEAD* all'url https://acme-v02.api.letsencrypt.org/acme/new-nonce. Anche relativamente a questa richiesta non sono state apportate modifiche, si vede rispettato in piena regola quanto atteso, ovvero un nuovo *nonce* da utilizzare per la richiesta successiva postata alla *certification authority*. Di seguito il risultato della richiesta

```
HTTP 200
Server: nginx
Date: Sat, 07 Dec 2019 20:44:54 GMT
Connection: keep-alive
Cache-Control: public, max-age=0, no-cache
Link: <https://acme-v02.api.letsencrypt.org/directory>;rel="index"
Replay-Nonce: 0002N_eHzTI7MJ2wLPB-lWtDdkMI88P6PZsYm-1cy-Q2xds
X-Frame-Options: DENY
Strict-Transport-Security: max-age=604800
```

6.3 Gestione di un account

6.3.1 Creazione di un account

Questa sezione entra nel vivo delle modifiche effettuate al codice. La chiave utilizzata per firmare la richiesta di creazione di un account non è più generata internamente, ma deriva dalla corretta gestione del file con estensione *pem* passato in input.

La richiesta *POST* effettuata all'url https://acme-v02.api.letsencrypt.org/acme/new-acct è la prima della serie ad contenere l'oggetto *JWS* ampiamente citato nei capitoli precedenti. Di seguito i dettagli

```
/*Richiesta codificata*/
{
  "protected":
      "eyJ1cmwiOiAiaHR0cHM6Ly9hY21lLXYwMi5hcGkubGV0c2VuY3J5cHQub3JnL2F
    jbWUvbmV3LWFjY3QiLCAiandrIjogeyJuIjogIi1JOW9vSHhzeUJRXzZDRXdYS0llVGVYMVJ5U0
    tkOUwtbGM0Q1VMdEJuZFZvd3VIZ1BoSENQbF8wMTRtNDd1V2JHdm9aUk1lajZLenI2RE53M29ta
    TJtZjgzVmNxazVMMzVVRDNVR0lRM0ZzVERaS1BUdWdJOEZmanlQQWhCQkM1M2NBajBXTzRzRnNl
    azJNbTNWMS1IRzRHX3laa3UwLXFqemlwMW52UGdPbzdfM0pWc3JMRl9USWpIWHY3bFdlc3k5MlN
    Ya2NXMXBKdnpZTlVtcHZTZGlzWEJTN2VObWgyMnBFWW5CUExnckV0MXRXVnBPY2pXQURscEU4ck
    xNbHFkRkNfUGdoMGllS1dGU2NrRVBONW9RSmRTRTJfQVRQdVNqV2xWSjE5NO1Gc1dMaWQ2QVdMT
    FRHbUl5aOhnN25mZVYwTGh6aDRuU2o0c2VjWkNpNFNOdyIsICJlIjogIkFRQUIiLCAia3R5Ijog
    IlJTQSJ9LCAiYWxnIjogIlJTMjU2IiwgIm5vbmNlIjogIjAwMDJOX2VIelRJNO1KMndMUEItbFd
    ORGRrTUk4OFA2UFpzWWOtMWN5LVEyeGRzIn0",
  "signature":
      "oY37L9ysMF22qFYyoMOvCa4CyCLd3XVRCMqnmXMjBGqIxXoASfk9Vl-aWKgNr8T
    nLmJE-K8Td46tpWYegZAO23OOqUU3BOUrfpptUbDO6e_SGLUMptFOhV2gO8qd1GyAOwYRAdnDMS
    aiyUmH3pZiq2n-mD9ZXPedmF8Jwpt4lUVbMK67-8VfK8wUhSou2sMWCg4sFThTx60Ujqwee2esg
    ByPaFJumMXQa6CF9NIk-20rWobGFfq7UyEzB539BCbzY0aJPOzc3W9v_zodUymT5ZUJeAKklz4t
    loS6nmDs1Sm88PSxDi_y0LgO9cAsp2VH_Q94yAjOiJE8LLNXzvhWtw",
  "payload":
      "ewogICJyZXNvdXJjZSI6ICJuZXctcmVnIiwKICAiY29udGFjdCI6IFsKICAgICJtY
    WlsdG86cm9iZXJ0YV9zZ3JvaUBob3RtYWlsLmNvbSIKICBdLAogICJ0ZXJtc09mU2VydmljZUFn
    cmVlZCI6IHRydWUKfQ"
}

/* Richiesta decodificata */
{
  "protected": {
    "url":"https://acme-v02.api.letsencrypt.org/acme/new-acct",
    "jwk":{
        "n":"-I9ooHxsyBQ_6CEwXKIeTeX1RySKd9L-lc4CULtBndVowuHgPhHCPl_014m47uWbGv
            oZRMej6Kzr6DNw3omi2mf83Vcqk5L35UD3UGIQ3FsTDZKPTugI8FfjyPAhBBC53cAj0
            WO4sFsek2Mm3V1-HG4G_yZku0-qjzip1nvPgOo7_3JVsrLF_TIjHXv7lWesy92SXkcW
```

```
            1pJvzYNUmpvSdisXBS7eNmh22pEYnBPLgrEt1tWVpOcjWADlpE8rLMlqdFC_Pgh0ieK
            WFSckEPN5oQJdSE2_ATPuSjWlVJ197MFsWLid6AWLLTGmIykHg7nfeV0Lhzh4nSj4se
            cZCi4Stw",
          "e":"AQAB",
          "kty":"RSA"
      },
      "alg":"RS256",
      "nonce":"0002N_eHzTI7MJ2wLPB-lWtDdkMI88P6PZsYm-1cy-Q2xds"
    },
    "signature":
        "oY37L9ysMF22qFYyoMOvCa4CyCLd3XVRCMqnmXMjBGqIxXoASfk9Vl-aWKgNr8T
      nLmJE-K8Td46tpWYegZAO23OOqUU3BOUrfpptUbD06e_SGLUMptFOhV2g08qd1GyAOwYRAdnDMS
      aiyUmH3pZiq2n-mD9ZXPedmF8Jwpt4lUVbMK67-8VfK8wUhSou2sMWCg4sFThTx60Ujqwee2esg
      ByPaFJumMXQa6CF9NIk-20rWobGFfq7UyEzB539BCbzY0aJP0zc3W9v_zodUymT5ZUJeAKklz4t
      loS6nmDs1Sm88PSxDi_y0LgO9cAsp2VH_Q94yAjOiJE8LLNXzvhWtw",
    "payload": {
      "resource": "new-reg",
      "contact": [
      "mailto:roberta_sgroi@hotmail.com"
      ],
      "termsOfServiceAgreed": true
    }
  }
```


Il *server* dopo opportuni controlli sull'esistenza o meno di un account associato alla stessa chiave, ha risposto in maniera attesa. Di seguito i dettagli


```
  HTTP 201
  Server: nginx
  Date: Sat, 07 Dec 2019 20:44:54 GMT
  Content-Type: application/json
  Content-Length: 568
  Connection: keep-alive
  Boulder-Requester: 73332542
  Cache-Control: public, max-age=0, no-cache
  Link: <https://acme-v02.api.letsencrypt.org/directory>;rel="index",
      <https://letsencrypt.org/documents/LE-SA-v1.2-November-15-2017.pdf>;
      rel="terms-of-service"
  Location: https://acme-v02.api.letsencrypt.org/acme/acct/73332542
  Replay-Nonce: 0002s7dZStffRcdePlMohS1P4igdTonABe-ZmB4pLETfbNw
  X-Frame-Options: DENY
  Strict-Transport-Security: max-age=604800

  {
    "key": {
      "kty": "RSA",
      "n":
          "-I9ooHxsyBQ_6CEwXKIeTeX1RySKd9L-lc4CULtBndVowuHgPhHCPl_014m47uWbGvoZR
          Mej6Kzr6DNw3omi2mf83Vcqk5L35UD3UGIQ3FsTDZKPTugI8FfjyPAhBBC53cAj0WO4sFse
          k2Mm3V1-HG4G_yZku0-qjzip1nvPgOo7_3JVsrLF_TIjHXv7lWesy92SXkcW1pJvzYNUmpv
          SdisXBS7eNmh22pEYnBPLgrEt1tWVpOcjWADlpE8rLMlqdFC_Pgh0ieKWFSckEPN5oQJdSE
          2_ATPuSjWlVJ197MFsWLid6AWLLTGmIykHg7nfeV0Lhzh4nSj4secZCi4Stw",
      "e": "AQAB"
    },
    "contact": [
      "mailto:roberta_sgroi@hotmail.com"
    ],
```

```
  "initialIp": "40.112.60.168",
  "createdAt": "2019-12-07T20:44:54.124293588Z",
  "status": "valid"
}
```


L'account è stato creato con successo. Confrontando il contenuto del file *.pem* passato in input e la chiave effettiva salvata nella cartella di sistema, file anch'esso con estensione *pem*, si nota il medesimo contenuto.

6.3.2 Disattivazione di un account

Questa funzionalità è stata testata lanciando il software e passando tra i parametri di input il parametro chiave *unregister*. Le risorse esposte dalla *certification authority* per poter portare a termine la rimozione di un certificato, sono le medesime utilizzate per verificare la validità di un account e per richiederne uno nuovo.

La prima chiamata effettuata alla *certification authority* serve a verificare l'effettiva esistenza e validità dell'account. L'url utilizzato è https://acme-v02.api.letsencrypt.org/acme/new-acct, a cambiare è il corpo della chiamata. Non abbiamo più richiesta di accesso alla risorsa, ma viene esclusivamente richiesta verifica.


```
/* Richiesta Codificata */
{
  "protected":
      "eyJ1cmwiOiAiaHR0cHM6Ly9hY21lLXYwMi5hcGkubGV0c2VuY3J5cHQub3JnL2F
    jbWUvbmV3LWFjY3QiLCAiandrIjogeyJrdHkiOiAiUlNBIiwgIm4iOiAiLUk5b29IeHN5QlFfNk
    NFd1hLSWVUZVgxUnlTS2Q5TC1sYzRDVUx0Qm5kVm93dUhnUGhIQ1BsXzAxNG00N3VXYkd2b1pST
    WVqNkt6cjZETnczb21pMm1mODNWY3FrNUwzNVVEM1VHSVEzRnNURFpLUFR1Z0k4RmZqeVBBaEJC
    QzUzY0FqMFdPNHNGc2VrMk1tM1YxLUhHNEdfeVprdTAtcWp6aXAxbnZQZO9vN18zSlZzckxGX1R
    JakhYdjdsV2VzeTkyU1hrY1cxcEp2ellOVW1wdlNkaXNYQlM3ZU5taDIycEVZbkJQTGdyRXQxdF
    dWcE9jaldBRGxwRThyTE1scWRGQ19QZ2gwaWVLV0ZTY2tFUE41b1FKZFNFMl9BVFB1U2pXbFZKM
    Tk3TUZzV0xpZDZBV0xMVEdtSXlrSGc3bmZlVjBMaHpoNG5TajRzZWNaQ2k0U3R3IiwgImUiOiAi
    QVFBQiJ9LCAibm9uY2UiOiAiMDAwMk5CTzZySkNTOEhPM2RaWllKcGxhaDVJMUhkN1dSZkFSekF
    IMkR3VDh5VFUiLCAiYWxnIjogIlJTMjU2In0",
  "signature":
      "inAPyo4LRjZx-14-ma5hyC5Jqj3IuV8Rn1oL4qRqGAuV3Grcz3-skYsfUTVyRNT
    10_ghGYPPoZKhh0_m4srMWVlOJhP0waw33V-rjLzebvEAFWh3grtgs6XNaRNsR5QQzb9OTwxac9
    HjexHTdXgvnWUtAOz8w-KxKxoYPggUH4VDnn1b7PAwQNQJ5V8hAdnlBFjV3iJlTirwHdWTshZa9
    JJhCCcfwblJjgH49s2akr1KrYHD5No5xe7rZyjpERZHdlhbV4ACObgeMhjS7kS7AgnJ8Rpvy_EU
    hAj0dR3GayaNqXIhL4wvWkfmp3bG6saiHQnvUKXjvl_Aa1OhbXlcuw",
  "payload": "ewogICJvbmx5UmV0dXJuRXhpc3RpbmciOiB0cnVlCn0"
}

/* Richiesta Decodificata */
{
  "protected":{
    "url":"https://acme-v02.api.letsencrypt.org/acme/new-acct",
    "jwk":{
        "kty":"RSA",
        "n":"-I9ooHxsyBQ_6CEwXKIeTeX1RySKd9L-lc4CULtBndVowuHgPhHCPl_014m47uWbGv
            oZRMej6Kzr6DNw3omi2mf83Vcqk5L35UD3UGIQ3FsTDZKPTugI8FfjyPAhBBC53cAj0
            WO4sFsek2Mm3V1-HG4G_yZku0-qjzip1nvPgOo7_3JVsrLF_TIjHXv7lWesy92SXkcW
            1pJvzYNUmpvSdisXBS7eNmh22pEYnBPLgrEt1tWVpOcjWADlpE8rLMlqdFC_Pgh0ieK
            WFSckEPN5oQJdSE2_ATPuSjWlVJ197MFsWLid6AWLLTGmIykHg7nfeVOLhzh4nSj4se
            cZCi4Stw",
        "e":"AQAB"
```

```
    },
    "nonce":"0002NBO6rJCS8HO3dZZYJplah5I1Hd7WRfARzAH2DwT8yTU",
    "alg":"RS256"
  },
  "payload": {
    "onlyReturnExisting": true
  },
  "signature":
      "inAPyo4LRjZx-14-ma5hyC5Jqj3IuV8Rn1oL4qRqGAuV3Grcz3-skYsfUTVyRNT
    10_ghGYPPoZKhh0_m4srMWVlOJhP0waw33V-rjLzebvEAFWh3grtgs6XNaRNsR5QQzb9OTwxac9
    HjexHTdXgvnWUtA0z8w-KxKxoYPggUH4VDnn1b7PAwQNQJ5V8hAdnlBFjV3iJlTirwHdWTshZa9
    JJhCCcfwblJjgH49s2akr1KrYHD5No5xe7rZyjpERZHdlhbV4ACObgeMhjS7kS7AgnJ8Rpvy_EU
    hAj0dR3GayaNqXIhL4wvWkfmp3bG6saiHQnvUKXjvl_Aa1OhbXlcuw"
}
```

Di seguito la risposta da parte del *server* con le informazioni richieste

```
HTTP 200
Server: nginx
Date: Sat, 07 Dec 2019 21:15:59 GMT
Content-Type: application/json
Content-Length: 558
Connection: keep-alive
Cache-Control: public, max-age=0, no-cache
Link: <https://acme-v02.api.letsencrypt.org/directory>;rel="index"
Location: https://acme-v02.api.letsencrypt.org/acme/acct/73332542
Replay-Nonce: 00029qgIvbH2wijdSncrKRocjONgy58HqfgDwFaslieRQUk
X-Frame-Options: DENY
Strict-Transport-Security: max-age=604800

{
  "key": {
    "kty": "RSA",
    "n":
        "-I9ooHxsyBQ_6CEwXKIeTeX1RySKd9L-lc4CULtBndVowuHgPhHCPl_014m47uWbGvoZR
        Mej6Kzr6DNw3omi2mf83Vcqk5L35UD3UGIQ3FsTDZKPTugI8FfjyPAhBBC53cAj0WO4sFse
        k2Mm3V1-HG4G_yZku0-qjzip1nvPgOo7_3JVsrLF_TIjHXv7lWesy92SXkcW1pJvzYNUmpv
        SdisXBS7eNmh22pEYnBPLgrEt1tWVpOcjWADlpE8rLMlqdFC_Pgh0ieKWFSckEPN5oQJdSE
        2_ATPuSjWlVJ197MFsWLid6AWLLTGmIykHg7nfeV0Lhzh4nSj4secZCi4Stw",
    "e": "AQAB"
  },
  "contact": [
    "mailto:roberta_sgroi@hotmail.com"
  ],
  "initialIp": "40.112.60.168",
  "createdAt": "2019-12-07T20:44:54Z",
  "status": "valid"
}
```

La chiamata successiva è all'url https://acme-v02.api.letsencrypt.org/acme/acct/73332542. Il corpo della chiamata conterrà la richiesta di disattivazione dell'account.

```
/* Richiesta Codificata */
{
  "protected":
      "eyJ1cmwiOiAiaHR0cHM6Ly9hY21lLXYwMi5hcGkubGV0c2VuY3J5cHQub3JnL2F
```

```
    jbWUvYWNjdC83MzMzMjU0MiIsICJraWQiOiAiaHR0cHM6Ly9hY21lLXYwMi5hcGkubGV0c2VuY3
    J5cHQub3JnL2FjbWUvYWNjdC83MzMzMjU0MiIsICJub25jZSI6ICIwMDAyOXFnSXZiSDJ3aWpkU
    25jcktSb2NqT05neTU4SHFmZ0R3RmFzbGllUlFVayIsICJhbGciOiAiUlMyNTYifQ",
  "signature":
      "GCRbpjp9afby9VKwWHHRr1IGazimwUzVPkBZ_fAHVVVFC5CZqMkHCvMT22difJ3
    5cVxt71CxWczOof9xoDLOPhjbs9w8mTsXnZX3-AxMftChhOmXncBFL3W0Xh2QqPm5uCIR7o8FTw
    bCOZEzuTlm_GDY5N5LjL6aak65VRKtAJCLx5wSuj9206p9WyXMw3keLBYuFAdyUwKaCrc0JpIij
    JkO0r4eezoSgPKC_MdaGKBrLF3l0gWy16s_HgFvl6IPA3gnrNHLuZPl6yzkrFfoLteFDOlvDHSw
    gnZdyvK5OOPZg_jwC8CdawEt7cZJKNK9U_I9t3mh5s_AcToGvbOJGw",
  "payload":
      "ewogICJzdGF0dXMiOiAiZGVhY3RpdmF0ZWQiLAogICJyZXNvdXJjZSI6ICJyZWciCn0"
}

/* Richiesta Decodificata */
{
  "protected": {
    "url":"https://acme-v02.api.letsencrypt.org/acme/acct/73332542",
    "kid":"https://acme-v02.api.letsencrypt.org/acme/acct/73332542",
    "nonce":"00029qgIvbH2wijdSncrKRocjONgy58HqfgDwFaslieRQUk",
    "alg":"RS256"
  },
  "payload":{
    "resource": "reg",
    "status": "deactivated"
  },
  "signature":
      "GCRbpjp9afby9VKwWHHRr1IGazimwUzVPkBZ_fAHVVVFC5CZqMkHCvMT22difJ3
    5cVxt71CxWczOof9xoDLOPhjbs9w8mTsXnZX3-AxMftChhOmXncBFL3W0Xh2QqPm5uCIR7o8FTw
    bCOZEzuTlm_GDY5N5LjL6aak65VRKtAJCLx5wSuj9206p9WyXMw3keLBYuFAdyUwKaCrc0JpIij
    JkO0r4eezoSgPKC_MdaGKBrLF3l0gWy16s_HgFvl6IPA3gnrNHLuZPl6yzkrFfoLteFDOlvDHSw
    gnZdyvK5OOPZg_jwC8CdawEt7cZJKNK9U_I9t3mh5s_AcToGvbOJGw"
}
```


La risposta affermativa del *server* è stata la seguente


```
HTTP 200
Server: nginx
Date: Sat, 07 Dec 2019 21:15:59 GMT
Content-Type: application/json
Content-Length: 564
Connection: keep-alive
Boulder-Requester: 73332542
Cache-Control: public, max-age=0, no-cache
Link: <https://acme-v02.api.letsencrypt.org/directory>;rel="index",
    <https://letsencrypt.org/documents/LE-SA-v1.2-November-15-2017.pdf>;
    rel="terms-of-service"
Replay-Nonce: 0001kT6j-goJ09W0vruGW0MScx-NYi9N-1XGeGcMK4tGPGA
X-Frame-Options: DENY
Strict-Transport-Security: max-age=604800

{
  "key": {
    "kty": "RSA",
    "n":
        "-I9ooHxsyBQ_6CEwXKIeTeX1RySKd9L-lc4CULtBndVowuHgPhHCPl_014m47uWbGvoZR
       Mej6Kzr6DNw3omi2mf83Vcqk5L35UD3UGIQ3FsTDZKPTugI8FfjyPAhBBC53cAj0WO4sFse
       k2Mm3V1-HG4G_yZku0-qjzip1nvPgOo7_3JVsrLF_TIjHXv7lWesy92SXkcW1pJvzYNUmpv
```

```
        SdisXBS7eNmh22pEYnBPLgrEt1tWVpOcjWADlpE8rLMlqdFC_Pgh0ieKWFSckEPN5oQJdSE
        2_ATPuSjWlVJ197MFsWLid6AWLLTGmIykHg7nfeV0Lhzh4nSj4secZCi4Stw",
      "e": "AQAB"
    },
    "contact": [
      "mailto:roberta_sgroi@hotmail.com"
    ],
    "initialIp": "40.112.60.168",
    "createdAt": "2019-12-07T20:44:54Z",
    "status": "deactivated"
  }
```


In seguito a disattivazione dell'account, è stato fatto un test di richiesta di registrazione utilizzando la stessa chiave, ma *certification authority* impedito la positiva conclusione dell'operazione, specificando che esiste già un account associato alla stessa chiave, ma esso è disattivato.

6.4 Richiesta di un certificato

6.4.1 Richiesta di un certificato

Una volta creato correttamente l'account, è subito stata effettuata richiesta di emissione di un nuovo certificato. L'url https://acme-v02.api.letsencrypt.org/acme/new-order è stato chiamato in *POST* con il seguente contenuto


```
/* Richiesta codificata */
{
  "protected":
      "eyJ1cmwiOiAiaHR0cHM6Ly9hY21lLXYwMi5hcGkubGV0c2VuY3J5cHQub3JnL2F
    jbWUvbmV3LW9yZGVyIiwgImFsZyI6ICJSUzI1NiIsICJraWQiOiAiaHR0cHM6Ly9hY21lLXYwMi
    5hcGkubGV0c2VuY3J5cHQub3JnL2FjbWUvYWNjdC83MzMzMjU0MiIsICJub25jZSI6ICIwMDAyc
    zdkWlN0ZmZSY2RlUGxNb2hTMVA0aWdkVG9uQUJlLVptQjRwTEVUZmJOdyJ9",
  "signature":
      "vNLQ7GsaLJ0oHecU4AvtxxR1XMvYbk267ZD4oh3bAAlW8dYmtLrYO1e-4_2ziDn
    wTWPDj1sq1Z2ShVHlW5OgJb8hsQVrJTXwNGcmqAJyX89sttdDU9388vnYrb2HN9K_Iv14vDLxjw
    WKt_Dl5H-jG9Li55xcve5RURYCg1gSEo4Wf96cIPI_6Jb2Ec0a1aTqYzutBaed1wqNsobSsM6Kc
    UeFxg-CCEB1H0yyd9sV8L7XL-S9FJf250NRw23HSjmbcMA0_m5A-hr9JM1FB6bOvg9_2d3e2LaO
    Rq4GxaxpZnB-FCvyA_NOIx7PAOtnZbeu3lSO3M7aI8Z1ipq7WG52Pw",
  "payload":
      "ewogICJpZGVudGlmaWVycyI6IFsKICAgIHsKICAgICAgInR5cGUiOiAiZG5zIiwKI
    CAgICAgInZhbHVlIjogImFwYWNoZXNlcnZlci5lYXN0dXMuY2xvdWRhcHAuYXp1cmUuY29tIgog
    ICAgfQogIF0KfQ"
}
```



```
/* Richiesta decodificata */
{
  "protected":{
    "url":"https://acme-v02.api.letsencrypt.org/acme/new-order",
    "alg":"RS256",
    "kid":"https://acme-v02.api.letsencrypt.org/acme/acct/73332542",
    "nonce":"0002s7dZStffRcdePlMohS1P4igdTonABe-ZmB4pLETfbNw"
  },
  "signature":
      "vNLQ7GsaLJ0oHecU4AvtxxR1XMvYbk267ZD4oh3bAAlW8dYmtLrYO1e-4_2ziDn
    wTWPDj1sq1Z2ShVHlW5OgJb8hsQVrJTXwNGcmqAJyX89sttdDU9388vnYrb2HN9K_Iv14vDLxjw
```

```
    WKt_Dl5H-jG9Li55xcve5RURYCg1gSEo4Wf96cIPI_6Jb2Ec0a1aTqYzutBaed1wqNsobSsM6Kc
    UeFxg-CCEB1H0yyd9sV8L7XL-S9FJf250NRw23HSjmbcMA0_m5A-hr9JM1FB6bOvg9_2d3e2LaO
    Rq4GxaxpZnB-FCvyA_NOIx7PAOtnZbeu3lSO3M7aI8Z1ipq7WG52Pw"
  "payload": {
    "identifiers": [
    {
      "type": "dns",
      "value": "apacheserver.eastus.cloudapp.azure.com"
    }]
  }
}
```


Il *server* ha dunque risposto come segue, fornendo il link per poter procedere con l'autorizzazione


```
HTTP 201
Server: nginx
Date: Sat, 07 Dec 2019 20:45:10 GMT
Content-Type: application/json
Content-Length: 368
Connection: keep-alive
Boulder-Requester: 73332542
Cache-Control: public, max-age=0, no-cache
Link: <https://acme-v02.api.letsencrypt.org/directory>;rel="index"
Location: https://acme-v02.api.letsencrypt.org/acme/order/73332542/1697427747
Replay-Nonce: 0002jDxq56ZMJGHTp1NO2D4oaLUeVubc81MkJXaM4Dy5zUU
X-Frame-Options: DENY
Strict-Transport-Security: max-age=604800

{
  "status": "pending",
  "expires": "2019-12-14T20:45:10.436814187Z",
  "identifiers": [
    {
      "type": "dns",
      "value": "apacheserver.eastus.cloudapp.azure.com"
    }
  ],
  "authorizations": [
    "https://acme-v02.api.letsencrypt.org/acme/authz-v3/1604805081"
  ],
  "finalize":
      "https://acme-v02.api.letsencrypt.org/acme/finalize/73332542/1697427747"
}
```


Lo stesso flusso viene seguito anche nel caso di richiesta di un nuovo certificato, utilizzando un account esistente. La richiesta di creazione di un account restituirà solo le informazioni relative all'account già registrato e il software procederà con la richiesta di emissione di un nuovo certificato.

6.4.2 Autorizzazione dell'identificatore

Come si può notare dalla risposta ottenuta dal server nello step precedente di richiesta di emissione di un certificato, adesso bisogna procedere con l'autorizzazione dell'identificatore. Viene dunque preparato il seguente *JWS* che comporrà la *POST* effettuata all'url https://acme-v02.api.letsencrypt.org/acme/authz-v3/1604805081

```
/* Richiesta codificata */
{
  "protected":
      "eyJ1cmwiOiAiaHR0cHM6Ly9hY21lLXYwMi5hcGkubGV0c2VuY3J5cHQub3JnL2F
    jbWUvYXV0aHotdjMvMTYwNDgwNTA4MSIsICJhbGciOiAiUlMyNTYiLCAia2lkIjogImh0dHBzOi
    8vYWNtZS12MDIuYXBpLmxldHNlbmNyeXB0Lm9yZy9hY21lL2FjY3QvNzMzMzI1NDIiLCAibm9uY
    2UiOiAiMDAwMmpEeHE1NlpNSkdIVHAxTk8yRDRvYUxVZVZ1YmM4MU1rSlhhTTREeTV6VVUifQ",
  "signature":
      "Ps6GRRASHGsflfQdH3KJpt0CVmFdNBoYpwr6nzKB0iHgelhgWyk3ijBVEyRx1b4
    Vz1Dtpx9qsWnZ83QY-YlxEc2xbL9D4IA6148vPJi5aekmkv3CmKuJR6boW8rBAOMDjxlYdQFqbQ
    OUwFE6BnjTeMqkcEZFM-EzkH9hpqUk9EBjnEBU1zI1Cijb1jVlts1LgmbTwtmPbSyAjCQDTB4tx
    E-TYW2-zJngwT0Miq-YLbU2rDqRp1FLfpXR1XqXaTYXgiwmBdyARmHBhRRbap6Q_e83ckYXbAIt
    rnriNcZ5aPWfQ8TIRP9NnWPnfQ2jgtH8MpDe4W5bTCM9zmxWiCO1lA",
  "payload": ""
}

/*Richiesta decodificata*/
{
  "protected": {
    "url":"https://acme-v02.api.letsencrypt.org/acme/authz-v3/1604805081",
    "alg":"RS256",
    "kid":"https://acme-v02.api.letsencrypt.org/acme/acct/73332542",
    "nonce":"0002jDxq56ZMJGHTp1NO2D4oaLUeVubc81MkJXaM4Dy5zUU"
  },
  "signature":
      "Ps6GRRASHGsflfQdH3KJpt0CVmFdNBoYpwr6nzKB0iHgelhgWyk3ijBVEyRx1b4
    Vz1Dtpx9qsWnZ83QY-YlxEc2xbL9D4IA6148vPJi5aekmkv3CmKuJR6boW8rBAOMDjxlYdQFqbQ
    OUwFE6BnjTeMqkcEZFM-EzkH9hpqUk9EBjnEBU1zI1Cijb1jVlts1LgmbTwtmPbSyAjCQDTB4tx
    E-TYW2-zJngwT0Miq-YLbU2rDqRp1FLfpXR1XqXaTYXgiwmBdyARmHBhRRbap6Q_e83ckYXbAIt
    rnriNcZ5aPWfQ8TIRP9NnWPnfQ2jgtH8MpDe4W5bTCM9zmxWiCO1lA",
  "payload": ""
}
```

La risposta del *server* a questa richiesta, come atteso, contiene la lista di informazioni per l'accesso a tutte le possibili sfide.

```
HTTP 200
Server: nginx
Date: Sat, 07 Dec 2019 20:45:10 GMT
Content-Type: application/json
Content-Length: 816
Connection: keep-alive
Boulder-Requester: 73332542
Cache-Control: public, max-age=0, no-cache
Link: <https://acme-v02.api.letsencrypt.org/directory>;rel="index"
Replay-Nonce: 0001IsU_L-H2TeyqWvfIkbOL7qCAkiqMFfRk7ZH_H5dt4_M
X-Frame-Options: DENY
Strict-Transport-Security: max-age=604800

{
  "identifier": {
    "type": "dns",
    "value": "apacheserver.eastus.cloudapp.azure.com"
  },
  "status": "pending",
```

```
  "expires": "2019-12-14T20:45:10Z",
  "challenges": [
    {
      "type": "http-01",
      "status": "pending",
      "url":
          "https://acme-v02.api.letsencrypt.org/acme/chall-v3/1604805081/N2Ke_Q",
      "token": "sAsBbG-2puvxT0CrybAAZkttM_K0-yCxpg9vVWQ5snw"
    },
    {
      "type": "dns-01",
      "status": "pending",
      "url":
          "https://acme-v02.api.letsencrypt.org/acme/chall-v3/1604805081/_MxAVw",
      "token": "sAsBbG-2puvxT0CrybAAZkttM_K0-yCxpg9vVWQ5snw"
    },
    {
      "type": "tls-alpn-01",
      "status": "pending",
      "url":
          "https://acme-v02.api.letsencrypt.org/acme/chall-v3/1604805081/rJuH3Q",
      "token": "sAsBbG-2puvxT0CrybAAZkttM_K0-yCxpg9vVWQ5snw"
    }
  ]
}
```


Dopo aver risposto alla sfida, viene effettuata correttamente una *POST* all'url https://acme-v02.api.letsencrypt.org/acme/chall-v3/1604805081/N2Ke_Q, con il seguente contenuto al fine di notificare il *server* di poter procedere con le verifiche finali che consentiranno il download del certificato


```
/* Richiesta Codificata */
{
  "protected":
      "eyJ1cmwiOiAiaHR0cHM6Ly9hY21lLXYwMi5hcGkubGV0c2VuY3J5cHQub3JnL2F
    jbWUvY2hhbGwtdjMvMTYwNDgwNTA4MS9OMktlX1EiLCAiYWxnIjogIlJTMjU2IiwgImtpZCI6IC
    JodHRwczovL2FjbWUtdjAyLmFwaS5sZXRzZW5jcnlwdC5vcmcvYWNtZS9hY2N0LzczMzMyNTQyI
    iwgIm5vbmNlIjogIjAwMDFJc1VfTC1IMlRleXFXdmZJa2JPTDdxQ0FraXFNRmZSazdaSF9INWR0
    NF9NIn0",
  "signature":
      "Z7QcTxAIuzekP5o7eZfNqYvSCyrRFWNdqPLwzqjZ_O00qXPg63qUvENhPbA-YSb
    VJAEwJiE90tLyW7m06zGhlYatiq1GASxuPFdS1g48j92XCwvirM4cZipjfGHtOeeFqKMQ56PXQp
    5DSJi6NKuLNmo1reyWDK5gVP75XGSfJeoB3QiHmJbFugiGzlPvb_ac0pQQfXJwLkoTFl_rBB1j1
    LWC9YnmJZ5Q-91d0cH33i5IdZGE40_7SnPNfRFjdQrXU7sNWeO_Kcp0YX8zUl-A8QiyMRwYXvyl
    Up54Xe-FlKERshFgmcM4ggHHGY4WRbx0k9G6Ur5ZD9jZaiRPnGTyeg",
  "payload":
      "ewogICJyZXNvdXJjZSI6ICJjaGFsbGVuZ2UiLAogICJ0eXBlIjogImh0dHAtMDEiCn0"
}

/* Richiesta decodificata */
{
  "protected": {
    "url":"https://acme-v02.api.letsencrypt.org/acme/chall-v3/1604805081/N2Ke_Q",
    "alg":"RS256",
    "kid":"https://acme-v02.api.letsencrypt.org/acme/acct/73332542",
    "nonce":"0001IsU_L-H2TeyqWvfIkbOL7qCAkiqMFfRk7ZH_H5dt4_M"
  },
```

```
  "signature":
      "Z7QcTxAIuzekP5o7eZfNqYvSCyrRFWNdqPLwzqjZ_O00qXPg63qUvENhPbA-YSb
    VJAEwJiE90tLyW7m06zGhlYatiq1GASxuPFdS1g48j92XCwvirM4cZipjfGHtOeeFqKMQ56PXQp
    5DSJi6NKuLNmo1reyWDK5gVP75XGSfJeoB3QiHmJbFugiGzlPvb_ac0pQQfXJwLkoTFl_rBB1j1
    LWC9YnmJZ5Q-91d0cH33i5IdZGE40_7SnPNfRFjdQrXU7sNWeO_Kcp0YX8zUl-A8QiyMRwYXvyl
    Up54Xe-FlKERshFgmcM4ggHHGY4WRbx0k9G6Ur5ZD9jZaiRPnGTyeg",
  "payload":{
    "resource": "challenge",
    "type": "http-01"
  }
}
```

Come atteso, il *server* notifica il richiedente che la richiesta è in stato *pending* in attesa di appropriate verifiche

```
HTTP 200
Server: nginx
Date: Sat, 07 Dec 2019 20:45:14 GMT
Content-Type: application/json
Content-Length: 185
Connection: keep-alive
Boulder-Requester: 73332542
Cache-Control: public, max-age=0, no-cache
Link: <https://acme-v02.api.letsencrypt.org/directory>;rel="index",
    <https://acme-v02.api.letsencrypt.org/acme/authz-v3/1604805081>;rel="up"
Location: https://acme-v02.api.letsencrypt.org/acme/chall-v3/1604805081/N2Ke_Q
Replay-Nonce: 0002_8CedLWiC2Y_5bXztnuXmw1cQQWhvcg4tvp-Fb2jgd0
X-Frame-Options: DENY
Strict-Transport-Security: max-age=604800

{
  "type": "http-01",
  "status": "pending",
  "url":
      "https://acme-v02.api.letsencrypt.org/acme/chall-v3/1604805081/N2Ke_Q",
  "token": "sAsBbG-2puvxT0CrybAAZkttM_K0-yCxpg9vVWQ5snw"
}
```

Anche nel caso di software modificato, viene nuovamente fatta richiesta di accesso alla sfida. La risposta ottenuta in questa caso è però differente. Mentre nel primo caso la richiesta risulta ancora in stato *pending*, adesso si nota può notare come lo stato risulta essere valido. Inoltre vengono restituiti tutti i dettagli relativi al dominio per il quale è stata effettuata richiesta di emissione, incluse le informazioni sulle sfide risolte. Di seguito i dettagli della risposta

```
HTTP 200
Server: nginx
Date: Sat, 07 Dec 2019 20:45:15 GMT
Content-Type: application/json
Content-Length: 1209
Connection: keep-alive
Boulder-Requester: 73332542
Cache-Control: public, max-age=0, no-cache
Link: <https://acme-v02.api.letsencrypt.org/directory>;rel="index"
Replay-Nonce: 0001OBQ4OG0wJRU7d2oH8HHLYmN5TgJKXkJqNgdwSzTzDUY
X-Frame-Options: DENY
Strict-Transport-Security: max-age=604800
```

```
{
  "identifier": {
    "type": "dns",
    "value": "apacheserver.eastus.cloudapp.azure.com"
  },
  "status": "valid",
  "expires": "2020-01-13T20:45:10Z",
  "challenges": [
    {
      "type": "http-01",
      "status": "valid",
      "url":
          "https://acme-v02.api.letsencrypt.org/acme/chall-v3/1604805081/N2Ke_Q",
      "token": "sAsBbG-2puvxTOCrybAAZkttM_KO-yCxpg9vVWQ5snw",
      "validationRecord": [
        {
          "url": "http://apacheserver.eastus.cloudapp.azure.com/.well-known/
            acme-challenge/sAsBbG-2puvxTOCrybAAZkttM_KO-yCxpg9vVWQ5snw",
          "hostname": "apacheserver.eastus.cloudapp.azure.com",
          "port": "80",
          "addressesResolved": [
            "40.112.60.168"
          ],
          "addressUsed": "40.112.60.168"
        }
      ]
    },
    {
      "type": "dns-01",
      "status": "pending",
      "url":
          "https://acme-v02.api.letsencrypt.org/acme/chall-v3/1604805081/_MxAVw",
      "token": "sAsBbG-2puvxTOCrybAAZkttM_KO-yCxpg9vVWQ5snw"
    },
    {
      "type": "tls-alpn-01",
      "status": "pending",
      "url":
          "https://acme-v02.api.letsencrypt.org/acme/chall-v3/1604805081/rJuH3Q",
      "token": "sAsBbG-2puvxTOCrybAAZkttM_KO-yCxpg9vVWQ5snw"
    }
  ]
}
```


La sfida è stata validata correttamente e il viene dunque creata la *CSR*[3] utile a finalizzare il processo di richiesta. La richiesta viene effettuata al link https://acme-v02.api.letsencrypt.org/acme/finalize/73332542/1697427747 ed avrà il seguente contenuto


```
/* Richiesta codificata */
{
  "protected":
      "eyJ1cmwiOiAiaHR0cHM6Ly9hY21lLXYwMi5hcGkubGV0c2VuY3J5cHQub3JnL2F
    jbWUvZmluYWxpemUvNzMzMzI1NDIvMTY5NzQyNzc0NyIsICJhbGciOiAiUlMyNTYiLCAia2lkIj
```


[3] *Certificate Signing Request*

```
    ogImh0dHBzOi8vYWNtZS12MDIuYXBpLmxldHNlbmNyeXB0Lm9yZy9hY21lL2FjY3QvNzMzMzI1N
    DIiLCAibm9uY2UiOiAiMDAwMTBCUTRPRzB3SlJVN2Qyb0g4SEhMWW1ONVRnSktYa0pxTmdkd1N6
    VHpEVVkifQ",
  "signature":
      "DFd66iyNI46bSyLLCyrd31w6n5EbZ3XGfHxGBLwGoG2ATi8mgb98hznV9Jt_8uK
    XjMgVoDcesFTtYOTurJKCf1io2OZi3WfdXsgsP68ZoJKjHo7xRO4rZ1HsbJscceSs_68yhens6v
    K6eiK9_kNhQvOpv6P89POi7uDeJGrDUP59lByfTchQoD_DCZuOwGX_wbZVIkxL7J9dG0fBm0sIr
    BJsAdRQVl95ySE_svXef79X90HGUNTLbiSLufbKiFRfYWQAODosW_7WLixKvfWS4H2RsBiJu03g
    W2WprX6p-GRn2CkIZrceUIIzANSz48lqple8PuS-iNX2Q25HTdvKTw",
  "payload":
      "ewogICJyZXNvdXJjZSI6ICJuZXctY2VydCIsCiAgImNzciI6ICJNSUlDaVRDQ0FYR
    UNBUUl3QURDQ0FTSXdEUVlKS29aSWh2Y05BUUVCQlFBRGdnRVBBRENDQVFvQ2dnRUJBTHAyaDNy
    aGIxS2VSVWtwaDJ3MGVxTXItVWlMSGowVTJvTkxwbzVGYkdlMXhWcUpiMjVFTzhkTFhuMjBoSDl
    DbnNsVTFhQVFlWHk2bTJ4V0FuRUlvc3h4ZOUxcmh2TWZySDk0ZzE5UkNxTktRLW9UX2VQZzNQRl
    cwMzBhVE1iSkhqOXBlNlVYUTM1bHB1Qk9tZVYwTmFjRG44STNQVTl4clVFOXZBZ2QtdnlGRWtmW
    mFUY19NaE5qVUR4eWFTOVBCUFBBMlhCNzNBdnBrY0hHMjlBRUJGOUdwanphU25mVHNCV3k1elFm
    XzhsMzFCVTNmWW1FS1hhVFlsclk3VFAtLUo0MEhhdDJ3OUJrSU9xSl9GTkQweS14R2tqN0ZjSzh
    rSlZEbEFDN3lmdWpJcVFqWE83dzJ5SzlBcVRLeUY5Wm9uSUVLenEyQjNPS2NrUjFqbXVsQ21NQ0
    F3RUFBYUJFTUVJRONTcUdTSWIzRFFFSkRqRTFNRE13TVFZRFZSMFJCQ293S0lJbVlYQmhZMmhsY
    zJWeWRtVnlMbVZoYzNSMWN5NWpiRzkxWkdGd2NDNWhlblZ5WlM1amIyMHdEUVlKS29aSWh2Y05B
    UUVMQlFBRGdnRUJBQ29xb1NSREpJSllJR2tZVXJmTzgxZFdXMmp0QVN2UXNnelMtZHFhNEt3S2x
    LMm04azR1VnlfOFB5ZWlUSHFrWFdINTQ3ajUyTHdVQ0V0dFVGNUpLLTlQRF8wbF8zNURTeXVNQk
    dOWF9VTTZaVmJaNW9zNjhFdzFLUVg5ZGNHT2xMd3l5U1pQVXh5UEEzSXJYUOFuTzVGWFdsblZKY
    1VCQlVzQWtnUzlRbmdwVGktUGlwelZNcTlIcUNER0pTYlMtRGhlcE80UnJNSTRPOGdoRUlRUWJ5
    alY2WlZPTjEOTkRIZC1PYmFIRXUxYWxtb2paclBNYnJfYmI5TGxWR2cwdHhIOTZWNUpTYjlvdHV
    oTjRDN2IwSOs2a2czSWNWSlhpbkFzdzJpNERyemg5UFJHcXhXSHFiNDBkUzRRbTNyM3RuaVJqNE
    9OeTdrX0g4R2lGWV9zY3hJIgp9"
}

/* Richiesta decodificata */
{
  "protected":{
    "url":"https://acme-v02.api.letsencrypt.org/acme/finalize/73332542/1697427747",
    "alg":"RS256",
    "kid":"https://acme-v02.api.letsencrypt.org/acme/acct/73332542",
    "nonce":"OOO1OBQ4OG0wJRU7d2oH8HHLYmN5TgJKXkJqNgdwSzTzDUY"
  },
  "signature":
      "DFd66iyNI46bSyLLCyrd31w6n5EbZ3XGfHxGBLwGoG2ATi8mgb98hznV9Jt_8uK
    XjMgVoDcesFTtYOTurJKCf1io2OZi3WfdXsgsP68ZoJKjHo7xRO4rZ1HsbJscceSs_68yhens6v
    K6eiK9_kNhQvOpv6P89POi7uDeJGrDUP59lByfTchQoD_DCZuOwGX_wbZVIkxL7J9dG0fBm0sIr
    BJsAdRQVl95ySE_svXef79X90HGUNTLbiSLufbKiFRfYWQAODosW_7WLixKvfWS4H2RsBiJu03g
    W2WprX6p-GRn2CkIZrceUIIzANSz48lqple8PuS-iNX2Q25HTdvKTw",
  "payload": {
    "resource": "new-cert",
    "csr":
        "MIICiTCCAXECAQIwADCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBALp2h3r
       hb1KeRUkph2w0eqMr-UiLHj0U2oNLpo5FbGe1xVqJb25EO8dLXn20hH9CnslU1aAQeXy6m2
       xWAnEIosxxgE1rhvMfrH94g19RCqNKQ-oT_ePg3PFW030aTMbJHj9pe6UXQ35lpuBOmeV0N
       acDn8I3PU9xrUE9vAgd-vyFEkfZaTc_MhNjUDxyaS9PBPPA2XB73AvpkcHG29AEBF9Gpjza
       SnfTsBWy5zQf_8l31BU3fYmEKXaTYlrY7TP--J4OHat2w9BkIOqJ_FND0y-xGkj7FcK8kJV
       DlAC7yfujIqQjXO7w2yK9AqTKyF9ZonIEKzq2B3OKckR1jmulCmMCAwEAAaBEMEIGCSqGSI
       b3DQEJDjE1MDMwMQYDVR0RBCowKIImYXBhY2hlc2VydmVyLmVhc3R1cy5jbG91ZGFwcC5he
       nVyZS5jb20wDQYJKoZIhvcNAQELBQADggEBACoqoSRDJIJYIGkYUrfO81dWW2jtASvQsgzS
       -dqa4KwKlK2m8k4uVy_8PyeiTHqkXWH547j52LwUCEttUF5JK-9PD_0l_35DSyuMBGNX_UM
       6ZVbZ5os68Ew1KQX9dcGOlLwyySZPUxyPA3IrXSAnO5FXWlnVJcUBBUsAkgS9QngpTi-Pip
       zVMq9HqCDGJSbS-DhepO4RrMI4O8ghEIQQbyjV6ZVON14NDHd-ObaHEu1almojZrPMbr_bb
```

```
        9LlVGg0txH96V5JSb9otuhN4C7b0KK6kg3IcVJXinAsw2i4Drzh9PRGqxWHqb40dS4Qm3r3
        tniRj4ONy7k_H8GiFY_scxI"
  }
}
```

La risposta da parte del *server*, anche in questo caso in maniera del tutto attesa, contiene il link per il download del certificato appena emesso

```
HTTP 200
Server: nginx
Date: Sat, 07 Dec 2019 20:45:15 GMT
Content-Type: application/json
Content-Length: 460
Connection: keep-alive
Boulder-Requester: 73332542
Cache-Control: public, max-age=0, no-cache
Link: <https://acme-v02.api.letsencrypt.org/directory>;rel="index"
Location: https://acme-v02.api.letsencrypt.org/acme/order/73332542/1697427747
Replay-Nonce: 0002TB9u45djYApjpevmomamxObpNCRd8aeYpE9XRwo1kuA
X-Frame-Options: DENY
Strict-Transport-Security: max-age=604800

{
  "status": "valid",
  "expires": "2019-12-14T20:45:10Z",
  "identifiers": [
    {
      "type": "dns",
      "value": "apacheserver.eastus.cloudapp.azure.com"
    }
  ],
  "authorizations": [
    "https://acme-v02.api.letsencrypt.org/acme/authz-v3/1604805081"
  ],
  "finalize": "https://acme-v02.api.letsencrypt.org/acme/finalize/73332542/
       1697427747",
  "certificate": "https://acme-v02.api.letsencrypt.org/acme/cert/
       0338904c36b135307aa91c34de1d5156e420"
}
```

6.4.3 Download del certificato

Dopo aver superato tutti gli step di validazione, dopo la finalizzazione della richiesta, avendo adesso ottenuto il link per il download del certificato, la chiamata successiva è proprio la seguente richiesta *POST* effettuata all'url https://acme-v02.api.letsencrypt.org/acme/cert/0338904c36b135307aa91c34de1d5156e420, che consentirà il download e salvataggio del certificato.

```
/* Richiesta Codificata */
{
  "protected":
      "eyJ1cmwiOiAiaHR0cHM6Ly9hY21lLXYwMi5hcGkubGV0c2VuY3J5cHQub3JnL2F
    jbWUvY2VydC8wMzM4OTA0YzM2YjEzNTMwN2FhOTFjMzRkZTFkNTE1NmU0MjAiLCAiYWxnIjogIl
    JTMjU2IiwgImtpZCI6ICJodHRwczovL2FjbWUtdjAyLmFwaS5sZXRzZW5jcnlwdC5vcmcvYWNtZ
```

```
    S9hY2N0LzczMzMyNTQyIiwgIm5vbmNlIjogIjAwMDFLaF9GcjRzN3pWaGpobkpXWXNSdmNGc0dY
    cjlXWmhUbHVhMFpRb1RnZFprIn0",
  "signature":
      "rvix-F_p-ZOPZMRff0-lsAq4nZryc2pGyXxtcicpU7-wwFepOJR4DISa7KU3zFU
    yCqBy96Mfw0MyS7e9OKD1sx_iVDNUyu4_gAX44xoW_Rkzv9UzvCk0tdnDZJNkYD-AZpavu_1ETY
    0pZVmy3O0JppE3fUrLFPJU24KQTSVLiN7p9ZR2TXfzRgFuOjT5OwA1_s4kcHEkgxOKcgJiCDqy5
    SK9eimX8Dxf6X89O2SGWsOc5i7NctJhoR_zc2mHItflucz_dlNquhNUDws64U9KztDwk_aIsioU
    7HeDcYvYOWHrQ-nFOZ1VoWU_Cqra6AjteoJzOG7F-ilz0pSl0VnJ6A",
  "payload": ""
}

/* Richiesta Decodificata */
{
  "protected": {
    "url":"https://acme-v02.api.letsencrypt.org/acme/cert/
              0338904c36b135307aa91c34de1d5156e420",
    "alg":"RS256",
    "kid":"https://acme-v02.api.letsencrypt.org/acme/acct/73332542",
    "nonce":"0001Kh_Fr4s7zVhjhnJWYsRvcFsGXr9WZhTlua0ZQoTgdZk"
  },
  "payload": "",
  "signature":
      "rvix-F_p-ZOPZMRff0-lsAq4nZryc2pGyXxtcicpU7-wwFepOJR4DISa7KU3zFU
    yCqBy96Mfw0MyS7e9OKD1sx_iVDNUyu4_gAX44xoW_Rkzv9UzvCk0tdnDZJNkYD-AZpavu_1ETY
    0pZVmy3O0JppE3fUrLFPJU24KQTSVLiN7p9ZR2TXfzRgFuOjT5OwA1_s4kcHEkgxOKcgJiCDqy5
    SK9eimX8Dxf6X89O2SGWsOc5i7NctJhoR_zc2mHItflucz_dlNquhNUDws64U9KztDwk_aIsioU
    7HeDcYvYOWHrQ-nFOZ1VoWU_Cqra6AjteoJzOG7F-ilz0pSl0VnJ6A"
}
```

La risposta del *server* è stata infatti un *200 OK* e il contenuto della risposta è il certificato come si può vedere di seguito

```
HTTP 200
Server: nginx
Date: Sat, 07 Dec 2019 20:45:16 GMT
Content-Type: application/pem-certificate-chain
Content-Length: 3620
Connection: keep-alive
Cache-Control: public, max-age=0, no-cache
Link: <https://acme-v02.api.letsencrypt.org/directory>;rel="index"
Replay-Nonce: 0002lLeGSCP7fpKZORwy7nsnwcGWEYeRyaXqQKNO8JCOlFo
X-Frame-Options: DENY
Strict-Transport-Security: max-age=604800

-----BEGIN CERTIFICATE-----
MIIFhDCCBGygAwIBAgISAziQTDaxNTB6qRw03h1RVuQgMA0GCSqGSIb3DQEBCwUA
MEoxCzAJBgNVBAYTAlVTMRYwFAYDVQQKEw1MZXQncyBFbmNyeXB0MSMwIQYDVQQD
ExpMZXQncyBFbmNyeXB0IEF1dGhvcml0eSBYMzAeFw0xOTEyMDcxOTQ1MTVaFw0y
MDAzMDYxOTQ1MTVaMDExLzAtBgNVBAMTJmFwYWNoZXNlcnZlci5lYXN0dXMuY2xv
dWRhcHAuYXp1cmUuY29tMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA
unaHeuFvUp5FSSmHbDR6oyv5SIsePRTag0umjkVsZ7XFWolvbkQ7x0tefbSEf0Ke
yVTVoBB5fLqbbFYCcQiizHGATWuG8x+sf3iDX1EKo0pD6hP94+Dc8VbTfRpMxske
P2l7pRdDfmWm4E6Z5XQ1pwOfwjc9T3GtQT28CB36/IUSR9lpNz8yE2NQPHJpL08E
88DZcHvcC+mRwcbb0AQEX0amPNpKd9OwFbLnNB//yXfUFTd9iYQpdpNiWtjtM/74
njQdq3bD0GQg6on8U0PTL7EaSPsVwryQlUOUALvJ+6MipCNc7vDbIr0CpMrIX1mi
cgQrOrYHc4pyRHWOa6UKYwIDAQABo4ICezCCAncwDgYDVR0PAQH/BAQDAgWgMB0G
A1UdJQQWMBQGCCsGAQUFBwMBBggrBgEFBQcDAjAMBgNVHRMBAf8EAjAAMB0GA1Ud
```

```
DgQWBBTcXLzcOPbn7Q+3vJQHuaVCmVgQWDAfBgNVHSMEGDAWgBSoSmpjBH3duubR
ObemRWXv86jsoTBvBggrBgEFBQcBAQRjMGEwLgYIKwYBBQUHMAGGImh0dHA6Ly9v
Y3NwLmludC14My5sZXRzZW5jcnlwdC5vcmcwLwYIKwYBBQUHMAKGI2h0dHA6Ly9j
ZXJ0LmludC14My5sZXRzZW5jcnlwdC5vcmcvMDEGA1UdEQQqMCiCJmFwYWNoZXNl
cnZlci5lYXN0dXMuY2xvdWRhcHAuYXp1cmUuY29tMEwGA1UdIARFMEMwCAYGZ4EM
AQIBMDcGCysGAQQBgt8TAQEBMCgwJgYIKwYBBQUHAgEWGmh0dHA6Ly9jcHMubGV0
c2VuY3J5cHQub3JnMIIBBAYKKwYBBAHWeQIEAgSB9QSB8gDwAHUA8JWkWfIA0YJA
EC0vk4iOrUv+HUfjmeHQNKawqKqOsnMAAAFu4hv91AAABAMARjBEAiBk+/RlYQVG
8baMhopqsHUPQpjihvIwrIKtOlNDVRLYrgIgWUoTaqdPKIBXGEdjAQcNaT67HNtw
4Bnsje3DWz/uiMMAdwCyHgXMi6LNiiBOh2b5K7mKJSBna9r6cOeySVMt74uQXgAA
AW7iG/3FAAAEAwBIMEYCIQDBzH7iukErFhYegQG5qmN96ya7hfFCdNbXUO0kNMM8
UQIhAPHOtF2jseL5SAbfetcdXqsePwjH8Crj/dwtkC6ISAVjMA0GCSqGSIb3DQEB
CwUAA4IBAQCNXHjVNlHTofdBr8zRooTZfPAnqCSwDplLXa/lfjjAcucjr1y+R/8/
8jAxkp1jNDQBoEBP9OZ/IotbYHI/8yq31D5fDbidK3Q6aygCU/E9mccWGqRsCVJl
5FkJW7dp0fKN1Eg1RMea4p1GfjPzrefyZL3nS2DiUGyksR3X/FtSPViopUIudUTl
mU1sfOwvmhhp3AgfKT82/gwa+eLdUQtmtBlmKU87oRE3hABm+ZwP6sJo12ao8BxS
SCggvk76LFE4XhDDK+ThRAwgQlykiVxRSBtBvzH5DP3iLic0dwDApxWp489fHih7
XP3qUWu+PXE/tduVQX/k6nBdiq3Bt6bD
-----END CERTIFICATE-----

-----BEGIN CERTIFICATE-----
MIIEkjCCA3qgAwIBAgIQCgFBQgAAAVOFc2oLheynCDANBgkqhkiG9w0BAQsFADA/
MSQwIgYDVQQKExtEaWdpdGFsIFNpZ25hdHVyZSBUcnVzdCBDby4xFzAVBgNVBAMT
DkRTVCBSb290IENBIFgzMB4XDTE2MDMxNzE2NDA0NloXDTIxMDMxNzE2NDA0Nlow
SjELMAkGA1UEBhMCVVMxFjAUBgNVBAoTDUxldCdzIEVuY3J5cHQxIzAhBgNVBAMT
GkxldCdzIEVuY3J5cHQgQXV0aG9yaXR5IFgzMIIBIjANBgkqhkiG9w0BAQEFAAOC
AQ8AMIIBCgKCAQEAnNMM8FrlLke3cl03g7NoYzDq1zUmGSXhvb418XCSL7e4S0EF
q6meNQhY7LEqxGiHC6PjdeTm86dicbp5gWAf15Gan/PQeGdxyGkOlZHP/uaZ6WA8
SMx+yk13EiSdRxta67nsHjcAHJyse6cF6s5K671B5TaYucv9bTyWaN8jKkKQDIZ0
Z8h/pZq4UmEUEz9l6YKHy9v6Dlb2honzhT+Xhq+w3Brvaw2VFn3EK6BlspkENnWA
a6xK8xuQSXgvopZPKiAlKQTGdMDQMc2PMTiVFrqoM7hD8bEfwzB/onkxEz0tNvjj
/PIzark5McWvxI0NHWQWM6r6hCm21AvA2H3DkwIDAQABo4IBfTCCAXkwEgYDVR0T
AQH/BAgwBgEB/wIBADAOBgNVHQ8BAf8EBAMCAYYwfwYIKwYBBQUHAQEEczBxMDIG
CCsGAQUFBzABhiZodHRwOi8vaXNyZy50cnVzdGlkLm9jc3AuaWRlbnRydXN0LmNv
bTA7BggrBgEFBQcwAoYvaHR0cDovL2FwcHMuaWRlbnRydXN0LmNvbS9yb290cy9k
c3Ryb290Y2F4My5wN2MwHwYDVR0jBBgwFoAUxKexpHsscfrb4UuQdf/EFWCFiRAw
VAYDVR0gBE0wSzAIBgZngQwBAgEwPwYLKwYBBAGC3xMBAQEwMDAuBggrBgEFBQcC
ARYiaHR0cDovL2Nwcy5yb290LXgxLmxldHNlbmNyeXB0Lm9yZzA8BgNVHR8ENTAz
MDGgL6AthitodHRwOi8vY3JsLmlkZW50cnVzdC5jb20vRFNUUk9PVENBWDNDUkwu
Y3JsMB0GA1UdDgQWBBSoSmpjBH3duubRObemRWXv86jsoTANBgkqhkiG9w0BAQsF
AAOCAQEA3TPXEfNjWDjdGBX7CVW+dla5cEilaUcne8IkCJLxWh9KEik3JHRRHGJo
uM2VcGfl96S8TihRzZvoroed6ti6WqEBmtzw3Wodatg+VyOeph4EYpr/1wXKtx8/
wApIvJSwtmVi4MFU5aMqrSDE6ea73Mj2tcMyo5jMd6jmeWUHK8so/joWUoHOUgwu
X4Po1QYz+3dszkDqMp4fklxBwXRsW10KXzPMTZ+sOPAveyxindmjkW8lGy+QsRlG
PfZ+G6Z6h7mjem0Y+iWlkYcV4PIWL1iwBi8saCbGS5jN2p8M+X+Q7UNKEkROb3N6
KOqkqm57TH2H3eDJAkSnh6/DNFu0Qg==
-----END CERTIFICATE-----
```


A questo punto il certificato viene salvato nella cartella di sistema contenente i certificati validi ed attivi, ovvero */etc/letsencrypt/live/apacheserver.eastus.cloudapp.azure.com*.

6.5 Rinnovo di un certificato

Il comando utilizzato per il rinnovo, verifica tutti i certificati attivi presenti nel sul *server* e procede con il rinnovo di quelli in scadenza, tralasciando quelli ancora in corso di validità. Per poter testare il flusso di rinnovo, è stata forzata la proceduta utilizzando il seguente comando

```
python3 main.py renew --force-renewal
    certificate=/home/roberta_sgroi/OpenSSL/test_cert_2.pem
```

Si è subito riscontrato il medesimo flusso presente nel caso di prima di richiesta di certificato, con qualche piccola differenza. Il flusso comincia con una richiesta di nuovo ordine, questo non risulterà in stato *pending* ma *ready*. Di seguito i dettagli della chiamata effettuata all'url https://acme-v02.api.letsencrypt.org/acme/new-order


```
/* Richiesta codificata */
{
  "payload":
      "ewogICJpZGVudGlmaWVycyI6IFsKICAgIHsKICAgICAgInR5cGUiOiAiZG5zIiwKI
    CAgICAgInZhbHVlIjogImFwYWNoZXNlcnlci5lYXN0dXMuY2xvdWRhcHAuYXp1cmUuY29tIgogI
    CAgfQogIF0KfQ",
  "signature":
      "B5AhsMco4Tj3JENQmIZCM4tHA6w0J4f2WhLwLLVlNge_a2gKuTpd2Yciw55x44G
    WN92OejD1lupwY1yrrNb5QMhTmYMYYI__E8z3-W17oe_DcP4uUbnwioSL2s0PwrMi-7RXQ-shS8
    gNGR-fGKLbYaPdN-S_hTnuwyWax5-N5P1ylADC6M3lcnn_3C6AqDx1oKfAo-OGlbI_3xfem3oFj
    PXNz-r4LVHuv5RnW4jJ4tG-jgX6EmEJr-xdq7pn7iIjTFPhWta3AD2PnEfRaEpTXpFDYNv7cIHz
    w2ixs7I3A5bbMscVSD06lKCuWIyMS29IbMSXSUdQlh6GMO6zcyGUIw",
  "protected":
      "eyJraWQiOiAiaHR0cHM6Ly9hY21lLXYwMi5hcGkubGV0c2VuY3J5cHQub3JnL2F
    jbWUvYWNjdC83MzMzMjU0MiIsICJhbGciOiAiUlMyNTYiLCAidXJsIjogImh0dHBzOi8vYWNtZS
    12MDIuYXBpLmxldHNlbmNyeXB0Lm9yZy9hY21lL25ldy1vcmRlciIsICJub25jZSI6ICIwMTAxS
    UV1RkZxanVrU29IVEdpUUxTWV92WjdwUElGTmdZcDdXNXNROXBvNm1BRSJ9"
}

/* Richiesta decodificata */
{
  "payload":{
    "identifiers": [
    {
    "type": "dns",
    "value": "apacheserver.eastus.cloudapp.azure.com"
    }]
  },
  "signature":"B5AhsMco4Tj3JENQmIZCM4tHA6w0J4f2WhLwLLVlNge_a2gKuTpd2Yciw55x44G
    WN92OejD1lupwY1yrrNb5QMhTmYMYYI__E8z3-W17oe_DcP4uUbnwioSL2s0PwrMi-7RXQ-shS8
    gNGR-fGKLbYaPdN-S_hTnuwyWax5-N5P1ylADC6M3lcnn_3C6AqDx1oKfAo-OGlbI_3xfem3oFj
    PXNz-r4LVHuv5RnW4jJ4tG-jgX6EmEJr-xdq7pn7iIjTFPhWta3AD2PnEfRaEpTXpFDYNv7cIHz
    w2ixs7I3A5bbMscVSD06lKCuWIyMS29IbMSXSUdQlh6GMO6zcyGUIw",
  "protected":{
    "kid":"https://acme-v02.api.letsencrypt.org/acme/acct/73332542",
    "alg":"RS256",
    "url":"https://acme-v02.api.letsencrypt.org/acme/new-order",
    "nonce":"0101IEuFFqjukSoHTGiQLSY_vZ7pPIFNgYp7W5sQ9po6mAE"
  }
}
```


Nella riposta del *server*, mostrata di seguito, si può notare la differenza sullo stato della richiesta.

```
HTTP 201
Server: nginx
Date: Sat, 07 Dec 2019 21:06:36 GMT
Content-Type: application/json
```

```
Content-Length: 366
Connection: keep-alive
Boulder-Requester: 73332542
Cache-Control: public, max-age=0, no-cache
Link: <https://acme-v02.api.letsencrypt.org/directory>;rel="index"
Location: https://acme-v02.api.letsencrypt.org/acme/order/73332542/1697546454
Replay-Nonce: 0102hh1oxz7kahFi-RB3B_2dxDnFjqi-3pIRvPOWXkE4zqQ
X-Frame-Options: DENY
Strict-Transport-Security: max-age=604800

{
  "status": "ready",
  "expires": "2019-12-14T21:06:36.251698232Z",
  "identifiers": [
    {
      "type": "dns",
      "value": "apacheserver.eastus.cloudapp.azure.com"
    }
  ],
  "authorizations": [
    "https://acme-v02.api.letsencrypt.org/acme/authz-v3/1604805081"
  ],
  "finalize":
      "https://acme-v02.api.letsencrypt.org/acme/finalize/73332542/1697546454"
}
```

Lo step successivo, come per la richiesta di nuova emissione, è una chiamata all'url https://acme-v02.api.letsencrypt.org/acme/authz-v3/1604805081 per procedere con l'autorizzazione dell'identificatore. Di seguito i dettagli.

```
/* Richiesta codificata */
{
  "payload": "",
  "signature":
      "HZSWt2JrqbpXx-YtnkIfW7K4kp3KDjQbzE9FC2ykvI2SRFkL8BQEwBOorCi3OOj
    WzxF4nTJcfcWSo6iWqxoon5fF7oYP2Ygc_FvKYCFsSVT_16u2_DCzzvGUCPACSZZcpfM0Fovnz-
    -tcE3GYhmddJZPALBKdvVev7UmwH9voRYrooENIOygW3pbfEoS0t9h7gkDUY1Vy_BPkhJ6EO_u2
    i6PfQEB0XP_9OYSfIT6TDb3aMvxC2Gbc7EWU-pzLis_xcjNkWWTzYOdwQ9kRqOaw8bcce8c9jh1
    NxGyJb7ouXBI7I3Yos0Uah3Y2Fov33Lt3mTFP_E6X7gf29jcPn5XfA",
  "protected":
      "eyJraWQiOiAiaHR0cHM6Ly9hY21lLXYwMi5hcGkubGV0c2VuY3J5cHQub3JnL2F
    jbWUvYWNjdC83MzMzMjU0MiIsICJhbGciOiAiUlMyNTYiLCAidXJsIjogImh0dHBzOi8vYWNtZS
    12MDIuYXBpLmxldHNlbmNyeXB0Lm9yZy9hY21lL2F1dGh6LXYzLzE2MDQ4MDUwODEiLCAibm9uY
    2UiOiAiMDEwMmhoMW94ejdrYWhGaS1SQjNCXzJkeERuRmpxaS0zcElSdlBPV1hrRTR6cVEifQ"
}

/* Richiesta decodificata */
{
  "payload": "",
  "signature":
      "HZSWt2JrqbpXx-YtnkIfW7K4kp3KDjQbzE9FC2ykvI2SRFkL8BQEwBOorCi3OOj
    WzxF4nTJcfcWSo6iWqxoon5fF7oYP2Ygc_FvKYCFsSVT_16u2_DCzzvGUCPACSZZcpfM0Fovnz-
    -tcE3GYhmddJZPALBKdvVev7UmwH9voRYrooENIOygW3pbfEoS0t9h7gkDUY1Vy_BPkhJ6EO_u2
    i6PfQEB0XP_9OYSfIT6TDb3aMvxC2Gbc7EWU-pzLis_xcjNkWWTzYOdwQ9kRqOaw8bcce8c9jh1
    NxGyJb7ouXBI7I3Yos0Uah3Y2Fov33Lt3mTFP_E6X7gf29jcPn5XfA",
  "protected":{
    "kid":"https://acme-v02.api.letsencrypt.org/acme/acct/73332542",
```

```
    "alg":"RS256",
    "url":"https://acme-v02.api.letsencrypt.org/acme/authz-v3/1604805081",
    "nonce":"0102hh1oxz7kahFi-RB3B_2dxDnFjqi-3pIRvPOWXkE4zqQ"
  }
}
```


Anche in questa risposta da parte del *server* si nota una piccola differenza. Lo stato della sfida risulta essere *valid* e non *pending*.


```
HTTP 200
Server: nginx
Date: Sat, 07 Dec 2019 21:06:36 GMT
Content-Type: application/json
Content-Length: 1209
Connection: keep-alive
Boulder-Requester: 73332542
Cache-Control: public, max-age=0, no-cache
Link: <https://acme-v02.api.letsencrypt.org/directory>;rel="index"
Replay-Nonce: 0102O335GBRaF3xBLbr7cYooIal1s5OrawB8G6G-DWdEzzI
X-Frame-Options: DENY
Strict-Transport-Security: max-age=604800

{
  "identifier": {
    "type": "dns",
    "value": "apacheserver.eastus.cloudapp.azure.com"
  },
  "status": "valid",
  "expires": "2020-01-13T20:45:10Z",
  "challenges": [
    {
      "type": "http-01",
      "status": "valid",
      "url":
          "https://acme-v02.api.letsencrypt.org/acme/chall-v3/1604805081/N2Ke_Q",
      "token": "sAsBbG-2puvxT0CrybAAZkttM_K0-yCxpg9vVWQ5snw",
      "validationRecord": [
        {
          "url": "http://apacheserver.eastus.cloudapp.azure.com/.well-known/
                    acme-challenge/sAsBbG-2puvxT0CrybAAZkttM_K0-yCxpg9vVWQ5snw",
          "hostname": "apacheserver.eastus.cloudapp.azure.com",
          "port": "80",
          "addressesResolved": [
            "40.112.60.168"
          ],
          "addressUsed": "40.112.60.168"
        }
      ]
    },
    {
      "type": "dns-01",
      "status": "pending",
      "url":
          "https://acme-v02.api.letsencrypt.org/acme/chall-v3/1604805081/_MxAVw",
      "token": "sAsBbG-2puvxT0CrybAAZkttM_K0-yCxpg9vVWQ5snw"
    },
    {
```

```
      "type": "tls-alpn-01",
      "status": "pending",
      "url":
          "https://acme-v02.api.letsencrypt.org/acme/chall-v3/1604805081/rJuH3Q",
      "token": "sAsBbG-2puvxTOCrybAAZkttM_KO-yCxpg9vVWQ5snw"
    }
  ]
}
```


Il fatto che il *server* consideri valida la sfida effettuata in fase di emissione del certificato, permette di saltare uno step in fase di richiesta di rinnovo. Il *client* in questo caso non è infatti tenuto a rispondere a nessuna sfida e si può passare alla chiamata successiva che contiene la *CSR*. Di seguito i dettagli della chiamata effettuata all'url https://acme-v02.api.letsencrypt.org/acme/finalize/73332542/1697546454


```
/* Richiesta codificata */
{
  "payload":
      "ewogICJyZXNvdXJjZSI6ICJuZXctY2VydCIsCiAgImNzciI6ICJNSUlDaVRDQ0FYR
    UNBUUl3QURDQ0FTSXdEUVlKS29aSWh2Y05BUUVCQlFBRGdnRVBBRENDQVFvQ2dnRUJBTXYxTDdI
    YUZueW9WRFFnS3lQa3pKRG9sN1NVb1VxLUdTTOgxMnhISjVIRE15bXplNkROeFpEOTU5a1kOUVN
    RWW5aUDJ6VO5PMG1jTGxkbjFHM1pScmlpd3JNQnJrRmJWLTNNZmFEQkhlQThZZVkzUnFrSWNoVj
    Nsd2NjbWpUMzBwWkVCNlk3R1ZjSERSamtsUFVraTRZWERzc1owMzUtQTlOb21LYjkyRXB6MGlhX
    19oUVpqMWg3a2FOenlaUmJaWWZNMV9oekhFT2ozakRmejBYSmJpSTREMOdsQnpraVRXbS1EaVZs
    NjJnZOVZTWlnY1lOTktmUGNzZjItcXVqYlRXVi1hRWpEWEJ4UjlBaFVLMThLalRubnJrMU5pZTl
    CdV9pXzA4bFhXNFQ3WHVFRlRDcTU5YURpUVUxZmdxZUc2OTNpNlpVYnpySW40RlFETjh5aXY4Q0
    F3RUFBYUJFTUVJRONTcUdTSWIzRFFFSkRqRTFNRE13TVFZRFZSMFJCQ293SOlJbVlYQmhZMmhsY
    zJWeWRtVnlMbVZoYzNSMWN5NWpiRzkxWkdGd2NDNWhlblZ5WlM1amIyMHdEUVlKS29aSWh2Y05B
    UUVMQlFBRGdnRUJBRmVsTmVwNXl4RUVlVVJkMGdpZC1YcVR6YVJpUOxHVEVXaFFkRGh5aTB5eks
    wak9NUndTdElwdXVBLTFvVU5BTzZfcFp0NXFZcHpQYm1odkwtcW1CSkRHVXgxRDY4aXB4YTcwUk
    w3VVJxXzNmX1VhSTNBQThjckNJOEdHRXJQQng5TnF1Rmc0QThETzB1aHJMY0stWmNjaGdKN255N
    WJpTHNzZOUyNm9TTndWdW1FTC1BblN5TVR3bHRmbU42SzhJc3dQcTBmb2tHU2dzSFJKRklGRngt
    WSOzR244RlY5cFYySGFPMnF1ekVtaOsyQmdqRVNqRlVQcUhCNFpRUOM5WXVRVTh1VURlVGZ2dDl
    ObnBBcl83WF9icUpvNGoyMkMzbjVmcUNtX2s3Zl1ZKNlcxM2xRdEdrQXRoMkJ6TTNkdW9LR2Z6a3
    Exa25lSTdNMlBOUGxfWUwwIgp9",
  "signature":
      "pwvrhnbXBy2VWjFmWAgSUYvhoEbLEbj-pRuCRzloAKnZk7CP7nD6nFSkrZudyrn
    A2IVqLkR4NmdrU3JqGFkVOneH1pRGSNGZBrCTxldmcLjqfFwwPvMlPj8QG3wYThqgDtSQmk7q-Y
    70Grq1xGlGzNDC_88b2eGv7wUcUQqUcA3ZEiJ0_fUoUuh-8zQ85Os-O4jmx2aC7Aozsqi530bZz
    0yERgBaYDejK-72qdDWHEO5ENSzeZsBZ2ifBfKc0sZogOlDlorJsL_xma3ETXVwjTMXTOcuxTJR
    Guv_cPYQsjH-HyZ1SOAmOfOCpkr9GUjnFIJQ3trAMnvTmTqivcwkSA",
  "protected":
      "eyJraWQiOiAiaHR0cHM6Ly9hY21lLXYwMi5hcGkubGV0c2VuY3J5cHQub3JnL2F
    j7bWUvYWNjdC83MzMzMjU0MiIsICJhbGciOiAiUlMyNTYiLCAidXJsIjogImh0dHBzOi8vYWNtZ
    S12MDIuYXBpLmxldHNlbmNyeXB0Lm9yZy9hY21lL2ZpbmFsaXplLzczMzMyNTQyLzE2OTc1NDY0
    NTQiLCAibm9uY2UiOiAiMDEwMk8zMzVHQlJhRjN4QkxicjdjWW9vSWFsMXM1T3Jhd0I4RzZHLUR
    XZEV6ekkifQ"
}

/* Richiesta decodificata */
{
  "payload": {
    "resource": "new-cert",
    "csr":
        "MIICiTCCAXECAQIwADCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAMv1L7H
        aFnyoVDQgKyPkzJDol7SUoUq-GSOH12xHJ5HDMymze6DtxZD959kY4QSQYnZP2zWNOOmcLl
        dn1G3ZRriiwrMBrkFbV-3MfaDBHeA8YeY3RqkIchV3lwccmjT30pZEB6Y7GVcHDRjklPUki
```

```
        4YXDssZ035-A9tomKb92Epz0ia__hQZj1h7katzyZRbZYfM1_hzHEOj3jDfz0XJbiI4D3Gl
        BzkiTWm-DiVl62ggEYMigcYNNKfPcsf2-qujbTWV-aEjDXBxR9AhUK18KjTnnrk1Nie9Bu_
        i_08lXW4T7XuEFTCq59aDiQU1fgqeG693i6ZUbzrIn4FQDN8yiv8CAwEAAaBEMEIGCSqGSI
        b3DQEJDjE1MDMwMQYDVR0RBCowKIImYXBhY2hlc2VydmVyLmVhc3R1cy5jbG91ZGFwcC5he
        nVyZS5jb20wDQYJKoZIhvcNAQELBQADggEBAFelNep5yxEEeURd0gid-XqTzaRiSLGTEWhQ
        dDhyi0yzK0jOMRwStIpuuA-1oUNAO6_pZt5qYpzPbmhvL-qmBJDGUx1D68ipxa70RL7URq_
        3f_UaI3AA8crCI8GGErPBx9NquFg4A8DO0uhrLcK-ZcchgJ7ny5biLssgE26oSNwVumEL-A
        nSyMTwltfmN6K8IswPq0fokGSgsHRJFIFFx-Y-3Gn8FV9pV2HaO2quzEmkK2BgjESjFUPqH
        B4ZQSC9YuQU8uUDeTfvt9tnpAr_7X_bqJo4j22C3n5fqCm_k7fVJ6W13lQtGkAth2BzM3du
        oKGfzkq1kneI7M2PtPl_YL0"
  },
  "signature":
      "pwvrhnbXBy2VWjFmWAgSUYvhoEbLEbj-pRuCRzloAKnZk7CP7nD6nFSkrZudyrn
    A2IVqLkR4NmdrU3JqGFkV0neH1pRGSNGZBrCTxldmcLjqfFwwPvMlPj8QG3wYThqgDtSQmk7q-Y
    70Grq1xGlGzNDC_88b2eGv7wUcUQqUcA3ZEiJ0_fUoUuh-8zQ85Os-O4jmx2aC7Aozsqi530bZz
    0yERgBaYDejK-72qdDWHEO5ENSzeZsBZ2ifBfKc0sZogOlDlorJsL_xma3ETXVwjTMXTOcuxTJR
    Guv_cPYQsjH-HyZ1S0Am0f0Cpkr9GUjnFIJQ3trAMnvTmTqivcwkSA",
  "protected": {
    "kid":"https://acme-v02.api.letsencrypt.org/acme/acct/73332542",
    "alg":"RS256",
    "url":"https://acme-v02.api.letsencrypt.org/acme/finalize/73332542/1697546454",
    "nonce":"0102O335GBRaF3xBLbr7cYooIal1s5OrawB8G6G-DWdEzzI"
  }
}
```


Il *server*, come atteso, ha risposto con uno stato valido della richiesta e il link per procedere con il download del nuovo certificato. Di seguito i dettagli.


```
HTTP 200
Server: nginx
Date: Sat, 07 Dec 2019 21:06:36 GMT
Content-Type: application/json
Content-Length: 460
Connection: keep-alive
Boulder-Requester: 73332542
Cache-Control: public, max-age=0, no-cache
Link: <https://acme-v02.api.letsencrypt.org/directory>;rel="index"
Location: https://acme-v02.api.letsencrypt.org/acme/order/73332542/1697546454
Replay-Nonce: 01013HGI4bV-p-D5KlzeFc9CvYaSzftF1ye7vG97ZVXXMz4
X-Frame-Options: DENY
Strict-Transport-Security: max-age=604800

{
  "status": "valid",
  "expires": "2019-12-14T21:06:36Z",
  "identifiers": [
    {
      "type": "dns",
      "value": "apacheserver.eastus.cloudapp.azure.com"
    }
  ],
  "authorizations": [
    "https://acme-v02.api.letsencrypt.org/acme/authz-v3/1604805081"
  ],
  "finalize": "https://acme-v02.api.letsencrypt.org/acme/finalize/73332542/
       1697546454",
  "certificate": "https://acme-v02.api.letsencrypt.org/acme/cert/
```

```
        046388a23e335f27459f1f0123fb551de392"
}
```

A questo punto il *client* può procedere con il download e l'installazione del nuovo certificato, effettuando una richiesta *POST* all'url https://acme-v02.api.letsencrypt.org/acme/cert/046388a23e335f27459f1f0123fb551de392.

```
/* Richiesta codificata */
{
  "payload": "",
  "signature":
      "ogdTnJqdZK0AcrjP1DfEz2sjC04P7slcyjAaO6Nl2TUGB7H2IqttmaJVJqLdbEa
    awUyod5Y4gMXn7kCrJPAlKd4IIdzTq_i0BoaVqe-qXOHb-bSODubwzQKRL9kzKhm6Bv7ubxIujg
    NITsYvYrm_KRU0bvBdpvefiapajXGEFjsaBp3Lf-Kv3R0u0-DqYRSWXqA1HBL1H-GKPRd46kzuf
    gg16z2tlESTvVvXursdqoS5AIq8I8c1CjRDcHYnP8KR3mtghtsTXCCvrZh6Q1XgddoJ7sRHli94
    Ciy4C8xE2EV-K7gBBYH_8Tl7fdjv_urGNtQATcGRJOAf6eB5s4d8QA",
  "protected":
      "eyJraWQiOiAiaHR0cHM6Ly9hY21lLXYwMi5hcGkubGV0c2VuY3J5cHQub3JnL2F
    jbWUvYWNjdC83MzMzMjU0MiIsICJhbGciOiAiUlMyNTYiLCAidXJsIjogImh0dHBzOi8vYWNtZS
    12MDIuYXBpLmxldHNlbmNyeXB0Lm9yZy9hY21lL2NlcnQvMDQ2Mzg4YTIzZTMzNWYyNzQ1OWYxZ
    jAxMjNmYjU1MWRlMzkyIiwgIm5vbmNlIjogIjAxMDJNOG9IeU9pTFR5ZHlLcF9vcVY5U2JtY0dv
    dmlYQjJnNllmTkl2SjFIMFZnIn0"
}

/* Richiesta decodificata */
{
  "payload": "",
  "signature":
      "ogdTnJqdZK0AcrjP1DfEz2sjC04P7slcyjAaO6Nl2TUGB7H2IqttmaJVJqLdbEa
    awUyod5Y4gMXn7kCrJPAlKd4IIdzTq_i0BoaVqe-qXOHb-bSODubwzQKRL9kzKhm6Bv7ubxIujg
    NITsYvYrm_KRU0bvBdpvefiapajXGEFjsaBp3Lf-Kv3R0u0-DqYRSWXqA1HBL1H-GKPRd46kzuf
    gg16z2tlESTvVvXursdqoS5AIq8I8c1CjRDcHYnP8KR3mtghtsTXCCvrZh6Q1XgddoJ7sRHli94
    Ciy4C8xE2EV-K7gBBYH_8Tl7fdjv_urGNtQATcGRJOAf6eB5s4d8QA",
  "protected": {
    "kid":"https://acme-v02.api.letsencrypt.org/acme/acct/73332542",
    "alg":"RS256",
    "url":"https://acme-v02.api.letsencrypt.org/acme/cert/046388a23e335f27459f1f0123fb551de3
    "nonce":"0102M8oHyOiLTydyKp_oqV9SbmcGoviXB2g6YfNIvJ1H0Vg"
  }
}

/* Risposta del server */
HTTP 200
Server: nginx
Date: Sat, 07 Dec 2019 21:06:38 GMT
Content-Type: application/pem-certificate-chain
Content-Length: 3620
Connection: keep-alive
Cache-Control: public, max-age=0, no-cache
Link: <https://acme-v02.api.letsencrypt.org/directory>;rel="index"
Replay-Nonce: 01011avH_46DHMne6mNXkWwYzYZ0hnkLh7C7ajij19DV8QA
X-Frame-Options: DENY
Strict-Transport-Security: max-age=604800

-----BEGIN CERTIFICATE-----
MIIFgzCCBGugAwIBAgISBGOIoj4zXydFnx8BI/tVHeOSMA0GCSqGSIb3DQEBCwUA
MEoxCzAJBgNVBAYTAlVTMRYwFAYDVQQKEw1MZXQncyBFbmNyeXB0MSMwIQYDVQQD
```

ExpMZXQncyBFbmNyeXB0IEF1dGhvcml0eSBYMzAeFw0xOTEyMDcyMDA2MzZaFw0y
MDAzMDYyMDA2MzZaMDExLzAtBgNVBAMTJmFwYWNoZXNlcnZlci5lYXN0dXMuY2xv
dWRhcHAuYXp1cmUuY29tMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA
y/UvsdoWfKhUNCArI+TMkOiXtJShSr4ZI4fXbEcnkcMzKbN7oO3FkP3n2RjhBJBi
dk/bNY07SZwuV2fUbdlGuKLCswGuQVtX7cx9oMEd4Dxh5jdGqQhyFXeXBxyaNPfS
lkQHpjsZVwcNGOSU9SSLhhcOyxnTfn4D22iYpv3YSnPSJr/+FBmPWHuRq3PJlFtl
h8zX+HMcQ6PeMN/PRcluIjgPcaUHOSJNab4OJWXraCARgyKBxg00p89yx/b6q6Nt
NZX5oSMNcHFHOCFQrXwqNOeeuTU2J70G7+L/TyVdbhPte4QVMKrn1oOJBTV+Cp4b
r3eLplRvOsifgVAM3zKK/wIDAQABo4ICejCCAnYwDgYDVR0PAQH/BAQDAgWgMB0G
A1UdJQQWMBQGCCsGAQUFBwMBBggrBgEFBQcDAjAMBgNVHRMBAf8EAjAAMB0GA1Ud
DgQWBBRqDHxfLrrrvl3kxo3vf7vsEvANtTAfBgNVHSMEGDAWgBSoSmpjBH3duubR
ObemRWXv86jsoTBvBggrBgEFBQcBAQRjMGEwLgYIKwYBBQUHMAGGImh0dHA6Ly9v
Y3NwLmludC14My5sZXRzZW5jcnlwdC5vcmcwLwYIKwYBBQUHMAKGI2h0dHA6Ly9j
ZXJ0LmludC14My5sZXRzZW5jcnlwdC5vcmcvMDEGA1UdEQQqMCiCJmFwYWNoZXNl
cnZlci5lYXN0dXMuY2xvdWRhcHAuYXp1cmUuY29tMEwGA1UdIARFMEMwCAYGZ4EM
AQIBMDcGCysGAQQBgt8TAQEBMCgwJgYIKwYBBQUHAgEWGmh0dHA6Ly9jcHMubGV0
c2VuY3J5cHQub3JnMIIBAwYKKwYBBAHWeQIEAgSB9ASB8QDvAHYAsh4FzIuizYog
Todm+Su5iiUgZ2va+nDnsklTLe+LkF4AAAFu4i+JcgAABAMARzBFAiAckH4goMMP
8TNZRaMn/gU7gkmcY148jbDvZ3Yv/izEwwIhAKa6nehBb5xaVErTIeeGoC2WjwLc
sW1NA7dDzKUw7lszAHUAb1N2rDHwMRnYmQCkURX/dxUcEdkCwQApBo2yCJo32RMA
AAFu4i+KMgAABAMARjBEAiArzBGKovhHCDMdQeORFEokQmjQ/ZAZXj7YJKgc9cTy
CwIgR/9oM40ZIEjozQ818zJkJ3J29hXK/edVmrmBInCBX/cwDQYJKoZIhvcNAQEL
BQADggEBAIZ3XHXcI10aDlVOioWnYedZbz1kopcK1m+LWoDVcoB8FQNNyuU/VZvr
5rchDMyxhe9m7+6WvHFe4/f0UG/6ftYjeyebvL32NPxeYwdwi5u18XN0q1sBOQTy
lftRF+WiD4HZSt3EnoxjTfvQDcNHJBSlQ5HbftfYV5DON7KdPfakK9hknmhvtbNU
8di4ND9owWXDQALuAnMDgi6BsUsFlhr/fYoSly6d2SHfjtN8BySbAhh0C+idopeQ
gT6dVLqO2F9+6oVdegecxpakyGq4VMW6VRch+c2xlKkaQiVf1BqVuPJP7Pkr9Cmx
yO44qPg3dnE4Wtr6H6YP0WBQ8D2E6HQ=
-----END CERTIFICATE-----

-----BEGIN CERTIFICATE-----
MIIEkjCCA3qgAwIBAgIQCgFBQgAAAVOFc2oLheynCDANBgkqhkiG9w0BAQsFADA/
MSQwIgYDVQQKExtEaWdpdGFsIFNpZ25hdHVyZSBUcnVzdCBDby4xFzAVBgNVBAMT
DkRTVCBSb290IENBIFgzMB4XDTE2MDMxNzE2NDA0NloXDTIxMDMxNzE2NDA0Nlow
SjELMAkGA1UEBhMCVVMxFjAUBgNVBAoTDUxldCdzIEVuY3J5cHQxIzAhBgNVBAMT
GkxldCdzIEVuY3J5cHQgQXV0aG9yaXR5IFgzMIIBIjANBgkqhkiG9w0BAQEFAAOC
AQ8AMIIBCgKCAQEAnNMM8FrlLke3cl03g7NoYzDq1zUmGSXhvb418XCSL7e4S0EF
q6meNQhY7LEqxGiHC6PjdeTm86dicbp5gWAf15Gan/PQeGdxyGkOlZHP/uaZ6WA8
SMx+yk13EiSdRxta67nsHjcAHJyse6cF6s5K671B5TaYucv9bTyWaN8jKkKQDIZ0
Z8h/pZq4UmEUEz9l6YKHy9v6Dlb2honzhT+Xhq+w3Brvaw2VFn3EK6BlspkENnWA
a6xK8xuQSXgvopZPKiAlKQTGdMDQMc2PMTiVFrqoM7hD8bEfwzB/onkxEz0tNvjj
/PIzark5McWvxI0NHWQWM6r6hCm21AvA2H3DkwIDAQABo4IBfTCCAXkwEgYDVR0T
AQH/BAgwBgEB/wIBADAOBgNVHQ8BAf8EBAMCAYYwfwYIKwYBBQUHAQEEczBxMDIG
CCsGAQUFBzABhiZodHRwOi8vaXNyZy50cnVzdGlkLm9jc3AuaWRlbnRydXN0LmNv
bTA7BggrBgEFBQcwAoYvaHR0cDovL2FwcHMuaWRlbnRydXN0LmNvbS9yb290cy9k
c3Ryb290Y2F4My5wN2MwHwYDVR0jBBgwFoAUxKexpHsscfrb4UuQdf/EFWCFiRAw
VAYDVR0gBE0wSzAIBgZngQwBAgEwPwYLKwYBBAGC3xMBAQEwMDAuBggrBgEFBQcC
ARYiaHR0cDovL2Nwcy5yb290LXgxLmxldHNlbmNyeXB0Lm9yZzA8BgNVHR8ENTAz
MDGgL6AthitodHRwOi8vY3JsLmlkZW50cnVzdC5jb20vRFNUUk9PVENBWDNDUkwu
Y3JsMB0GA1UdDgQWBBSoSmpjBH3duubRObemRWXv86jsoTANBgkqhkiG9w0BAQsF
AAOCAQEA3TPXEfNjWDjdGBX7CVW+dla5cEilaUcne8IkCJLxWh9KEik3JHRRHGJo
uM2VcGfl96S8TihRzZvoroed6ti6WqEBmtzw3Wodatg+VyOeph4EYpr/1wXKtx8/
wApIvJSwtmVi4MFU5aMqrSDE6ea73Mj2tcMyo5jMd6jmeWUHK8so/joWUoHOUgwu
X4Po1QYz+3dszkDqMp4fklxBwXRsW10KXzPMTZ+sOPAveyxindmjkW8lGy+QsRlG
PfZ+G6Z6h7mjem0Y+iWlkYcV4PIWL1iwBi8saCbGS5jN2p8M+X+Q7UNKEkROb3N6
KOqkqm57TH2H3eDJAkSnh6/DNFu0Qg==
-----END CERTIFICATE-----

Tutto questo processo potrebbe essere ulteriormente automatizzato creando un *cron job*[4] che lanci il comando al posto dell'utente quando il certificato è in scadenza. In questa fase non è infatti necessaria nessuna interazione da parte dell'uomo poiché non vengono richieste informazioni aggiuntive rispetto a quelle già inserite in fase di prima registrazione.

6.6 Revoca di un certificato

Al fine di procedere con la revoca di un certificato, il software necessita che venga esplicitamente passato in input i seguenti parametri *revoke –cert-path certificate_absolute_file_path.pem*. Una volta controllato se il certificato è ancora esistente nel file system, viene preparato l'oggetto *JWS* che viene utilizzato come corpo della chiamata *POST* effettuata all'url https://acme-v02.api.letsencrypt.org/acme/revoke-cert. Di seguito i dettagli

```
/* Richiesta codificata*/
{
  "payload":
     "ewogICJjZXJ0aWZpY2F0ZSI6ICJNSUlGZ3pDQ0JHdWdBd0lCQWdJU0JHT0lvajR6W
    HlkRm54OEJJX3RWSGVPU01BMEdDU3FHU0liM0RRRUJDd1VBTUVveEN6QUpCZO5WQkFZVEFsVlRN
    Ull3RkFZRFZRUUtFdzFNWlhRbmN5QkZibU55ZVhCME1TTXdJUVlEVlFREV4cE1aWFFuY3lCRmJ
    tTnllWEIwSUVGMWRHaHZjbWwwZVNCWU16QWVGdzB4T1RFeU1EY3lNREEyTXpaYUZ3MHlNREF6TU
    RZeU1EQTJNelphTURFeEx6QXRCZO5WQkFNVEptRndZVO5vWlhOb GNuWmxjaTVsWVhOMGRYTXVZM
    nh2ZFdSaGNIQXVZWHAxY21VdVkyOXRNSUlCSWpBTkJna3Foa2lHOXcwQkFRUZBQU9DQVE4QU1J
    SUJDZ0tDQVFFQX1fVXZzZG9XZktoVU5DQXJJLVRNa09pWHRKU2hTcjRaSTRmWGJFY25rY016S2J
    ON29PM0ZrUDNuMlJqaEJKQmlka19iTlkwN1Nad3VWMmZVYmRsR3VLTENzd0d1UVZ0WDdjeDlvTU
    VkNER4aDVqZEdxUWh5RlhlWEJ4eWFOUGZTbGtRSHBqc1pWd2NORO9TVTlTU0xoaGNPeXhuVGZuN
    EQyMmlZcHYzWVNuUFNKcl8tRkJtUFdIdVJxM1BKbEZ0bGg4elgtSE1jUTZQZU1OX1BSY2x1SWpn
    UGNhVUhPU0pOYWI0T0pXWHJhQ0FSZ3lLQnhnMDBwODl5eF9iNnE2TnROWlg1b1NNTmNIRkgwQ0Z
    Rclh3cU5PZWV1VFUySjcwRzctTF9UeVZkYmhQdGU0UVZNS3JuMW9PSkJUVi1DcDRicjNlTHBsUn
    ZPc2lmZ1ZBTTN6S0tfd0lEQVFBQm80OSUNlakNDQW5ZdORnWURWUjBQQVFIX0JBUURBZ1dnTUIwR
    OExVWRKUVFXTUJRR0NDc0dBUVVGQndNQkJnZ3JCZ0VGQlFjREFqQU1CZ05WSFJNQkFmOEVBakFB
    TUIwR0ExVWREZ1FXQkJScURIeGZMcnJydmwza3hvM3ZmN3ZzRXZBTnRUQWZCZ05WSFNNRUdEQVd
    nQlNvU21wakJIM2R1dWJST2JlbVJXWHY4Nmpzb1RCdkJnZ3JCZ0VGQlFjQkFRUmpNR0V3TGdZSU
    t3WUJCUVVITUFHR0ltaDBkSEE2THk5dlkzTndMbWx1ZEMxNE15NXNaWFJ6Wlc1amNubHdkQzV2Y
    21jd0x3WUlLd1lCQlFVSE1BS0dJMmgwZEhBNkx5OWpaWEowTG1sdWRDMTRNeTVzWlhSelpXNWpj
    bmx3ZEM1dmNtY3ZNREVHQTFVZEVRUXFNQ2lDSm1Gd1lXTm9aWE5sY25abGNpNWxZWE4wZFhNdVk
    yeHZkV1JoY0hBdVlYcDFjbVV1WTI5dE1Fd0dBMVVkSUFSRk1FTXdDQVlHWjRFTUFRSUJNRGNHQ3
    lzR0FRUUJndDhUQVFFQk1DZ3dKZ1lJS3dZQkJRVUhBZ0VXR21oMGRIQTZMeTlqY0hNdWJHVjBjM
    lZ1WTNKNWNIUXViMOpuTUlJQkF3WUtLd1lCQkFIV2VRSUVBZ1NCOUFTQjhRRHZBSFlBc2g0RnpJ
    dWl6WW9nVG9kbS1TdTVpaVVnWjJ2YS1uRG5za2xUTGUtTGtGNEFBQUZ1NGktSmNnQUFCQU1BUnp
    CRkFpQWNrSDRnb01NUDhUTlpSYU1uX2dVN2drbWNZMTQ4amJEdlozWXZfaXpFd3dJaEFLYTZuZW
    hCYjV4YVZFclRJZWVHb0MyV2p3TGNzVzFOQTdkRHpLVXc3bHN6QUhVQWIxTjJyREh3TVJuWW1RQ
    2tVUlhfZHhVY0VkaON3UUFwQm8yeUNKbzMyUk1BQUFGdTRpLUtNZ0FBQkFNQVJqQkVBaUFyekJH
    S292aEhDRE1kUWVPUkZFb2tRbWpRX1pBWlhqN1lKS2djOWNUeUN3SWdSXzlvTTQwWklFam96UTg
    xOHpKaOozSjI5aFhLX2VkVm1ybUJJbkNCWF9jdORRWUpLb1pJaHZjTkFRRUxCUUFEZ2dFQkFJWj
    NYSFhjSTEwYURsVk9pb1duWWVkWmJ6MWtvcGNLMW0tTFdvRFZjbOI4RlFOTnl1VV9WWnZyNXJja
    ERNeXhoZTltNy02V3ZIRmU0X2YwVUdfNmZOWWpleWVidkwzMk5QeGVZd2R3aTV1MThYTjBxMXNC
    T1FUeWxmdFJGLVdpRDRIWlNOMOVub3hqVGZ2UURjTkhKQlNsUTVIYmZOZllWNURPTjdLZFBmYWt
    LOWhrbm1odnRiTlU4ZGk0TkQ5b3dXWERRQUx1QW5NRGdpNkJzVXNGbGhyX2ZZb1NseTZkMlNIZm
    pOTjhCeVNiQWhoMEMtaWRvcGVRZ1Q2ZFZMcU8yRjktNm9WZGVnZWN4cGFreUdxNFZNVzZWUmNoL
    WMyeGxLa2FRaVZmMUJxVnVQSlA3UGtyOUNteHlPNDRxUGczZG5FNFd0cjZINllQMFdCUThEMkU2
    SFEiLAogICJyZWFzb24iOiAwLAogICJyZXNvdXJjZSI6ICJyZXZva2UtY2VydCIKfQ",
```

[4]Si tratta di un'unità di esecuzione, il cui compito è specificato dall'utente sulla base delle esigenze e il cui tempo di avvio può essere impotato dinamicamente.

```
  "signature":
      "Cg7je4LO7vntVeILnGtu14Jtxzpmmn43jJoMTxFVzsGlRD8w431L8gJmk9gxfNO
    55TX5hEFoLhX9pimEt9yQGn4i7O-C3WMmpHLjPEROuduu9Ly822aK7HdQAfARV3wb_lTphBDqz7
    TbNwp1garbfWIXwsugFnRyi-luLouKZe3jcP4URnxVxo0aC0lbN9iLtXI1RSpXl31wyTDVR8gFS
    DgbcyKiL4GyYLlf1LEY8FHa9nvujBdW61ZbYPIH2wUt08ogGH5VJ5UKJm22lodqkWFyHEndM6gU
    RPSzJkTLFwFNhol4kv5JEeDfSroaXqy5d6CXcvNO_Tgzy0cVk-O7Yg",
  "protected":
      "eyJraWQiOiAiaHR0cHM6Ly9hY21lLXYwMi5hcGkubGV0c2VuY3J5cHQub3JnL2F
    jbWUvYWNjdC83MzMzMjU0MiIsICJub25jZSI6ICIwMDAxVndfb3pudjFzRVJ4TkVnWHJpYUNEbl
    kzYXREUXlteVh6aWNnOWFIQnY4TSIsICJ1cmwiOiAiaHR0cHM6Ly9hY21lLXYwMi5hcGkubGV0c
    2VuY3J5cHQub3JnL2FjbWUvcmV2b2tlLWNlcnQiLCAiYWxnIjogIlJTMjU2In0"
}

/* Richiesta decodificata */
{
  "signature":
      "Cg7je4LO7vntVeILnGtu14Jtxzpmmn43jJoMTxFVzsGlRD8w431L8gJmk9gxfNO
    55TX5hEFoLhX9pimEt9yQGn4i7O-C3WMmpHLjPEROuduu9Ly822aK7HdQAfARV3wb_lTphBDqz7
    TbNwp1garbfWIXwsugFnRyi-luLouKZe3jcP4URnxVxo0aC0lbN9iLtXI1RSpXl31wyTDVR8gFS
    DgbcyKiL4GyYLlf1LEY8FHa9nvujBdW61ZbYPIH2wUt08ogGH5VJ5UKJm22lodqkWFyHEndM6gU
    RPSzJkTLFwFNhol4kv5JEeDfSroaXqy5d6CXcvNO_Tgzy0cVk-O7Yg",
  "protected": {
    "kid":"https://acme-v02.api.letsencrypt.org/acme/acct/73332542",
    "nonce":"0001Vw_oznv1sERxNEgXriaCDnY3atDQymyXzicg9aHBv8M",
    "url":"https://acme-v02.api.letsencrypt.org/acme/revoke-cert",
    "alg":"RS256"
  },
  "payload": {
    "certificate":
        "MIIFgzCCBGugAwIBAgISBGOIoj4zXydFnx8BI_tVHeOSMA0GCSqGSIb3DQE
        BCwUAMEoxCzAJBgNVBAYTAlVTMRYwFAYDVQQKEw1MZXQncyBFbmNyeXB0MSMwIQYDVQQDEx
        pMZXQncyBFbmNyeXB0IEF1dGhvcml0eSBYMzAeFw0xOTEyMDcyMDA2MzZaFw0yMDAzMDYyM
        DA2MzZaMDExLzAtBgNVBAMTJmFwYWNoZXNlcnZlci5lYXN0dXMuY2xvdWRhcHAuYXp1cmUu
        Y29tMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAy_UvsdoWfKhUNCArI-TMkOi
        XtJShSr4ZI4fXbEcnkcMzKbN7oO3FkP3n2RjhBJBidk_bNY07SZwuV2fUbdlGuKLCswGuQV
        tX7cx9oMEd4Dxh5jdGqQhyFXeXBxyaNPfSlkQHpjsZVwcNGOSU9SSLhhcOyxnTfn4D22iYp
        v3YSnPSJr_-FBmPWHuRq3PJlFtlh8zX-HMcQ6PeMN_PRcluIjgPcaUHOSJNab4OJWXraCAR
        gyKBxg00p89yx_b6q6NtNZX5oSMNcHFH0CFQrXwqNOeeuTU2J70G7-L_TyVdbhPte4QVMKr
        n1oOJBTV-Cp4br3eLplRvOsifgVAM3zKK_wIDAQABo4ICejCCAnYwDgYDVR0PAQH_BAQDAg
        WgMB0GA1UdJQQWMBQGCCsGAQUFBwMBBggrBgEFBQcDAjAMBgNVHRMBAf8EAjAAMB0GA1UdD
        gQWBBRqDHxfLrrrvl3kxo3vf7vsEvANtTAfBgNVHSMEGDAWgBSoSmpjBH3duubRObemRWXv
        86jsoTBvBggrBgEFBQcBAQRjMGEwLgYIKwYBBQUHMAGGImh0dHA6Ly9vY3NwLmludC14My5
        sZXRzZW5jcnlwdC5vcmcwLwYIKwYBBQUHMAKGI2h0dHA6Ly9jZXJ0LmludC14My5sZXRzZW
        5jcnlwdC5vcmcvMDEGA1UdEQQqMCiCJmFwYWNoZXNlcnZlci5lYXN0dXMuY2xvdWRhcHAuY
        Xp1cmUuY29tMEwGA1UdIARFMEMwCAYGZ4EMAQIBMDcGCysGAQQBgt8TAQEBMCgwJgYIKwYB
        BQUHAgEWGmh0dHA6Ly9jcHMubGV0c2VuY3J5cHQub3JnMIIBAwYKKwYBBAHWeQIEAgSB9AS
        B8QDvAHYAsh4FzIuizYogTodm-Su5iiUgZ2va-nDnsklTLe-LkF4AAAFu4i-JcgAABAMARz
        BFAiAckH4goMMP8TNZRaMn_gU7gkmcY148jbDvZ3Yv_izEwwIhAKa6nehBb5xaVErTIeeGo
        C2WjwLcsW1NA7dDzKUw7lszAHUAb1N2rDHwMRnYmQCkURX_dxUcEdkCwQApBo2yCJo32RMA
        AAFu4i-KMgAABAMARjBEAiArzBGKovhHCDMdQeORFEokQmjQ_ZAZXj7YJKgc9cTyCwIgR_9
        oM40ZIEjozQ818zJkJ3J29hXK_edVmrmBInCBX_cwDQYJKoZIhvcNAQELBQADggEBAIZ3XH
        XcI10aDlVOioWnYedZbz1kopcK1m-LWoDVcoB8FQNNyuU_VZvr5rchDMyxhe9m7-6WvHFe4
        _f0UG_6ftYjeyebvL32NPxeYwdwi5u18XN0q1sBOQTylftRF-WiD4HZSt3EnoxjTfvQDcNH
        JBSlQ5HbftfYV5DON7KdPfakK9hknmhvtbNU8di4ND9owWXDQALuAnMDgi6BsUsFlhr_fYo
        Sly6d2SHfjtN8BySbAhh0C-idopeQgT6dVLqO2F9-6oVdegecxpakyGq4VMW6VRch-c2xlK
        kaQiVf1BqVuPJP7Pkr9CmxyO44qPg3dnE4Wtr6H6YP0WBQ8D2E6HQ",
    "reason": 0,
```

```
    "resource": "revoke-cert"
  }
}
```

Come previsto il *server*, dopo aver preso atto della revoca, ha restituito una conferma di avvenuta revoca.

```
HTTP 200
Server: nginx
Date: Sat, 07 Dec 2019 21:11:02 GMT
Content-Length: 0
Connection: keep-alive
Boulder-Requester: 73332542
Cache-Control: public, max-age=0, no-cache
Link: <https://acme-v02.api.letsencrypt.org/directory>;rel="index"
Replay-Nonce: 0002LNECneITL9cQh6K0gPflNq-oihCcoW19oLlz151DdDk
X-Frame-Options: DENY
Strict-Transport-Security: max-age=604800
```

A questo punto il software chiede all'utente se vuole che le chiavi, il certificato e ogni oggetto correlato vengano eliminati o meno, e procede sulla base del feedback.

In seguito a revoca, si è stata la possibilità di richiedere un certificato a partire dalla stessa chiave e la *certification authority* ha correttamente negato la richiesta. La motivazione è che un certificato associato alla stessa chiave è già stato emesso e successivamente revocato.

6.7 Risultati

Effettuando oun'analisi delle due fasi di test, osservando i comportamenti riscontrati, confrontando gli input e gli output ottenuti, si evidenzia un'estrema versatilità sia all'interno del software *Certbot*, sia da parte della *certification authority*. Si è riusciti con successo a procedere nella validazione di un certificato non generato in maniera ad hoc dal software. Si è riusciti, senza troppi problemi, a procedere con la fase di mantenimento del certificato. E non sono state riscontrate difficoltà nell'eseguire, forzatamente, procedure che esulano dal flusso standard di gestione automatica del certificato, quali revoca dello stesso e disattivazione dell'account.

Il riscontro positivo ottenuto conferma che è possibile lavorare con chiavi generate esternamente. Esse riescono ad integrarsi in maniera omogenea all'interno del flusso implementato, purché rispettino gli standard supportati dalla *certification authority*.

Capitolo 7

Conclusioni e sviluppi futuri

La percentuale di utenti con accesso ad internet cresce ogni giorno di più. Insieme alla percentuale di utilizzatori, cresce esponenzialmente la condivisione, a volte anche non volontaria, di informazioni personali e sensibili. Diventa dunque sempre più importante garantire un minimo di sicurezza nella comunicazione attraverso il *World Wide Web* e quindi la possibilità di istaurare un canale sicuro, che fornisca protezione da attacchi o intromissioni da parte di terzi.

Nel garantire la protezione dovuta, la maggior parte dei *server*, su cui è installato un sito *web* utilizzabile dall'utente medio, permette di comunicare utilizzando il protocollo *TLS*[1]. L'*owner* del dominio o del *server* ha dunque richiesto l'emissione di un certificato che ha poi provveduto ad installare sul *server*.

La procedura standard di richiesta, emissione e mantenimento, come ampiamente discusso, richiede un grande effort economico e tecnico da parte del richiedente. Una nuova *certification authority* ha dunque affermato di poter garantire un servizio gratuito e una gestione automatica di tutti i certificati emessi e ed installati. Il lavoro di tesi nasce dalla curiosità di esaminare questo nuovo servizio offerto e capire quanto esso possa essere flessibile e dinamico.

Al termine del percorso di studi, si evince che grazie alla nuova e sempre più fidata *certification authority Let's Encrypt*, la sicurezza delle informazioni e la gestione dei certificati può diventare sempre più semplice pur rimanendo del tutto gratuita.

Si è testato che il processo di generazione e gestione dei certificati funziona perfettamente sia in maniera automatizzata, sia se forzato. Il grande vantaggio apportato da questo nuovo servizio sta nel ridurre notevolmente la necessità di possedere competenze tecniche da parte dell'utente utilizzatore. L'utente deve solo avviare la richiesta e può dimenticarsi di avere un certificato da gestire.

Si è riscontrata una grande flessibilità e capacità di adattamento a input diversi. Si è infatti implementata e successivamente testata la possibilità di fornire all'utente il vantaggio di poter decidere in che modo generare la chiave da associare ad un certificato, ed ciò implicitamente consente di utilizzarne una già in proprio possesso. Una miglioria banale apportabile al software consiste in una gestione più specifica e precisa della connessione tra il file *.pem* passato in input e la chiave e/o il certificato risultante. Il software richiede il nome del certificato da revocare qualora se ne facesse richiesta, e attualmente bisogna essere a conoscenza del nome finale per poter procedere correttamente. Se il nome finale combaciasse con quello della chiave passata in input inizialmente, la richiesta di revoca di un certificato da parte dell'utente potrebbe essere più semplice. Oltre ad utilizzare chiavi generate esternamente da associare ad un certificato, la medesima opportunità è stata fornita relativamente alla chiave utilizzata per la registrazione dell'account. Attualmente non viene concessa la possibilità di aggiornamento di questa chiave. Gli utenti normalmente disattivano l'account e ne creano uno nuovo. Questo comporta l'iter di accettazione dei termini di accordo, inserimento di nuove informazioni e richiesta di un nuovo

[1] *Transport Layer Security.*

certificato, perdendo potenzialmente la validità di uno associato al precedente account. Fornire la possibilità di aggiornamento di questa chiave, mantenendo immutate tutte le informazioni e i servizi già collegati e funzionanti per l'account, risulterebbe in un grosso vantaggio.

Uno spunto interessante, su come far evolvere l'analisi cominciata con il presente lavoro di tesi, potrebbe essere quello di procedere con la generazione della chiave da associare al certificato mediante *Hardware Security Module*, associare questa prima chiave ad una seconda chiave, e utilizzare poi effettivamente quest'ultima per effettuare richiesta di emissione di un certificato. In questo modo la chiave originale verrebbe mantenuta offline e libera da ogni compromissione. Similmente, si potrebbe entrare nell'ambito del *Trusted computing*. Quest'ultimo prende sempre più piede ai nostri giorni. Consiste nell'inserimento di dispositivi hardware e software per rendere più sicuri computer e cellulari. In tutti verrebbe inserito un *Trusted Platform Module* che consta di una coppia di chiavi crittografiche asimmetriche, uniche per ogni chip. Quindi si potrebbe pensare di partire da questa coppia di chiavi per richiedere l'emissione di un nuovo certificato, se il modulo è applicato ad una macchina *server*.

Una costante fondamentale che va mantenuta nel tempo, riguarda la necessità di aggiornare la lista degli algoritmi di cifratura supportati e implementati. Man mano che quelli già presenti diventano obsoleti, si rendono maggiormente esposti a nuovi possibili attacchi.

Appendices

Appendice A

SSL e Certificati

SSL (Secure Sockets Layer) è un protocollo crittografico, originariamente creato dalla Netscape Communications [1], che viene utilizzato nell'ambito della sicurezza delle applicazioni di rete. Il fine ultimo di questo protocollo è quello di proteggere la comunicazione tra un *client* (web browser) e un *server*.

L'utilizzo di SSL è diventato indispensabile a causa della debolezza dei metodi di autenticazione odierni. Nel passato le informazioni erano centralizzate, i terminali avevano accesso a funzioni limitate e comunicavano coi *server* in *unicast* e attraverso linee dedicate. Era dunque sufficiente immettere username e password per avere accesso ai dati ed essere allo stesso tempo protetti.
Oggi le informazioni sono distribuite, le comunicazioni avvengono in *broadcast* e su linee condivise, ma soprattutto i terminali hanno una maggiore capacità computazionale. Nonostante i sistemi siano molto più complessi rispetto al passato, i metodi di autenticazione sono rimasti per lo più gli stessi. Da una parte quindi si mantiene semplice l'accesso alle informazioni, ma dall'altra si è maggiormente esposti a diversi tipi di attacchi; non è inoltre garantita la privacy durante lo scambio dei dati.

Nell'architettura ISO/OSI [2], SSL si colloca a livello di sessione, quindi appena sopra il livello di trasporto nel quale non sono previste funzionalità di sicurezza. Il protocollo SSL colma questa carenza, offrendo protezione da alcuni tipi di attacchi, garantendo la privacy dei dati sensibili e assicurando l'autenticità del *server* con il quale si sta comunicando. In particolare, per portare a termine quest'ultimo compito, è indispensabile l'utilizzo dei certificati, i quali permettono l'identificazione del *server* in maniera univoca all'interno della rete.

A.1 I Certificati

A.1.1 Crittografia e Certificati a Chiave Pubblica

Per poter introdurre e approfondire il concetto di certificato, è indispensabile definire la crittografia a chiave pubblica. Essa è una crittografia di tipo asimmetrico che utilizza una coppia di chiavi, una pubblica e una privata, generate allo stesso tempo e strettamente correlate. Se una di queste due chiavi viene utilizzata per criptare, l'altra servirà a decriptare, e qualora non ci fosse corrispondenza tra le due, si presenteranno errori in decriptazione o si potrà facilmente riconoscere un possibile attacco. Le chiavi pubbliche e private vengono utilizzate per effettuare:

[1] Netscape Communications è un'azienda americana che fornisce servizi informatici, diventata famosa per l'invenzione del primo browser grafico di successo.

[2] International Organization for Standardization (ISO) stabilì nel 1978 lo standard per reti di calcolatori Open Systems Interconnection (OSI) composto attualmente da 7 livelli, che dal basso verso l'alto sono: Fisico, Collegamento, Rete, Trasporto, Sessione, Presentazione, Applicazione.

- autenticazione → la chiave pubblica è utile a verificare che solo il titolare della corrispondente chiave privata abbia inviato un determinato messaggio;
- riservatezza senza condivisione del segreto → solo il titolare della chiave privata sarà in grado di decriptare un determinato messaggio, criptato con la relativa chiave pubblica.

Per poter usufruire di questo sistema, la chiave pubblica deve essere distribuita, mentre quella privata deve rimanere esclusivamente al proprietario senza essere resa nota, rendendo così questo meccanismo il più sicuro per la distribuzione di chiavi crittografiche.

Un certificato è una struttura dati utilizzata per associare univocamente una chiave pubblica al possessore della chiave privata corrispondente, la quale non è pubblicamente nota. Esso contiene dunque informazioni sulla chiave pubblica, sul suo possessore e inoltre include la firma digitale [3] dell'ente che ha emesso il certificato.

A.1.2 Certification Authority

Ogni certificato viene emesso da un ente autorizzato, chiamato *Certification Authority*, che è riconosciuto come affidabile sia dal possessore di un certificato, sia da terze parti che vengono in contatto con lo stesso certificato. Una *Certification Authority*, nella creazione, distribuzione e revoca di un certificato, segue determinate regole e procedure specificate dalla *Public Key Infrastructure*. Quest'ultima permette, attraverso una fase di registrazione, di effettuare un collegamento tra una chiave pubblica e la persona, o l'organizzazione, che richiede il certificato.

Quando un utente richiede l'emissione di un certificato, effettua una *Certificate Signing Request* utilizzando lo standard *PKCS#10*. La richiesta contiene il *Distiguished Name*, la chiave pubblica per il quale si richiede l'emissione del certificato e altri attributi. Quest'ultimo campo viene utilizzato per richiedere la registrazione o la revoca, per fornire informazioni che si vogliono inserire nel certificato, o contiene informazioni aggiuntive sul richiedente. Per garantire l'integrità della richiesta, essa viene poi firmata con la chiave privata del richiedente che corrisponde a quella pubblica inserita nella richiesta.
Non appena una *Certification Authority* riceve una CSR [4], effettua dei controlli sulle credenziali del richiedente, quest'ultimo deve quindi fornire alla *Registration Authority* un valido documento di riconoscimento. Se la richiesta viene ritenuta valida, è compito della *Validation Authority* procedere con i controlli che riguardano la possessione del dominio per il quale il richiedente desidera l'emissione del certificato. Se anche quest'ultima fase va a buon fine, un nuovo certificato, che identificherà, in maniera univoca, il dominio del richiedente, viene generato e ad esso viene associata la chiave pubblica.

Una *Certification Authority* ha anche il compito fondamentale di delegare ad altre entità la possibilità di emettere dei certificati, considerati ancora affidabili, eleggendoli di fatto ad essere delle *Certification Authority* ad essa subordinate. Si viene così a creare una gerarchia fidata di *Certification Authority*, continuando a mantenere l'affidabilità dei certificati emessi grazie al fatto che su ogni certificato è apportata la firma dell'ente emittente. In questo modo, in fase di verifica di affidabilità del certificato, è possibile risalire fino alla *Certification Authority* di *root*, la quale è autofirmata e considerata dunque affidabile per definizione.

Inoltre, è della *Certification Authority* il compito di gestire l'aggiornamento dei certificati (ad ognuno di essi è assegnata una data di scadenza) e la loro revoca, quest'ultima può dipendere dal volere del possessore o dall'emittente che ne riscontra un utilizzo improprio. Un *client* che vuole verificare l'autenticità di un certificato può farlo attraverso due meccanismi:

- *Certificate Revocation List* → ogni volta che un certificato viene revocato, la *Certification Authority* si premura di inserirlo in una lista contenente tutti i certificati che non devono

[3] Una firma digitale viene applicata ad un documento o un messaggio, al fine di garantire l'autenticazione del mittente, che non potrà negare di aver inviato il messaggio grazie alla proprietà di non ripudio, e inoltre assicura l'integrità del messaggio.

[4] Certificate Signing Request

più essere ritenuti validi. Queste liste vengono aggiornate periodicamente e firmate dalle *Certification Authority* che hanno emesso i certificati presenti nella lista, così da considerare affidabile anche quest'ultima. Nel processo di revoca di un certificato, e quindi del suo inserimento nella CRL, vi sono dei tempi di latenza. In una prima fase (che va dal momento in cui il possessore si accorge che la propria chiave è stata compromessa al momento in cui viene stabilita una data di revoca) il possessore del certificato è responsabile di tutto ciò che ne consegue; durante la seconda fase (che prevede l'inserimento del certificato revocato nella CRL e termina con la pubblicazione della sua nuova versione) è la *Certification Authority* a rispondere di eventuali implicazioni. Per verificare la validità di un certificato è dunque possibile controllare la lista emessa dalla *Certification Authority* che ha firmato il certificato, tenendo sempre conto della data in cui la lista è stata aggiornata; questo processo richiede del tempo a causa della grandezza del file contenente la lista.

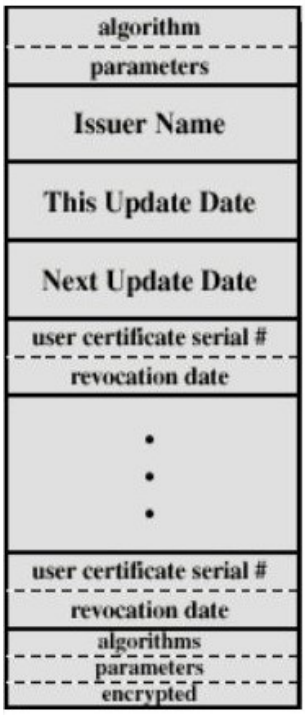

Figura A.1. Certificate Revocation List

- *Online Certificate Status Protocol* → piuttosto che accedere ad una lista veramente grande di certificati, è possibile chiedere informazioni solo sul certificato a cui si è interessati mediante l'utilizzo di questo protocollo. Un *client*, che vuole verificare la validità di un certificato, invia una richiesta OCSP alla *Certification Authority* che ha firmato il certificato. Quest'ultima delega a uno dei suoi *server* OCSP il compito di fare i controlli opportuni e di rispondere al *client*. La risposta viene firmata dunque dal *server* e non dalla *Certification Authority*, e può contenere una delle seguenti informazioni sul certificato: valido, sconosciuto o revocato, e in quest'ultimo caso verranno specificate anche data e motivazioni di revoca. Questo metodo è sicuramente più veloce, ma espone maggiormente il *client*, che richiede informazioni, ad un attacco di tipo *replay*. Un attaccante potrebbe infatti intercettare la risposta del *server* e consegnarla al *client* solo successivamente, ovvero quando lo stato del certificato è stato modificato.

A.1.3 Certificati X.509

X.509 è lo standard più utilizzato dai protocolli di rete, tra cui SSL, che definisce il formato di un certificato. La figura A.2 mostra la struttura di un certificato di tipo X.509 versione 3.

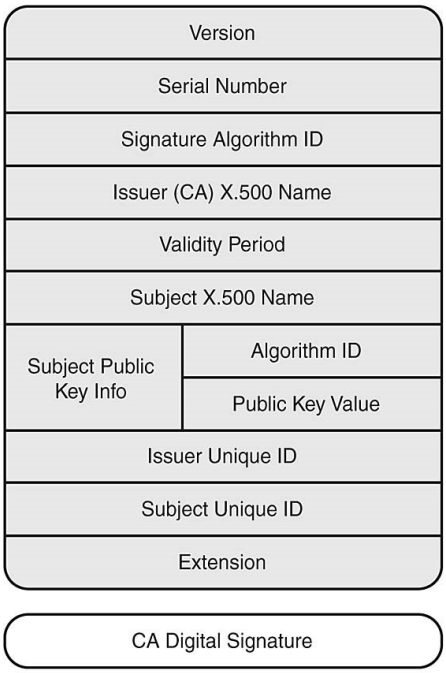

Figura A.2. Certificato X.509

- *version* → specifica con quale versione di X.509 è stato creato il certificato, determinando quali informazioni possono essere aggiunte al certificato. Nelle versioni precedenti alla 3, non erano incluse ad esempio le estensioni, mentre *subject* e *issuer* ID sono stati introdotti nella seconda versione;
- *serial number* → è un numero che identifica in maniera univoca il certificato, che risulta utile in fase di revoca, poiché è proprio il seriale che viene inserito nella *Certificate Revocation List*;
- *signature algorithm ID* → indica quale algoritmo ha utilizzato la *Certification Authority* per firmare il certificato;
- *issuer name* → contiene informazioni riguardanti la *Certification Authority* che ha emesso il certificato. Esso è espresso come *distinguished name* e contiene campi come la nazione (C), l'organizzazione (O), l'autorità organizzativa (OA), il nome comune (CN), l'email (MAIL), e molti altri;
- *validity period* → al fine di mantenere le proprietà di sicurezza, ad ogni certificato viene assegnato un periodo di validità limitato, ed esso dipende da diversi fattori (la lunghezza della chiave privata o quanto ha pagato il richiedente per un certificato). Questo periodo è espresso da una data di inizio validità e da una data di fine;
- *subject name* → esattamente come nel caso dell'emittente, questo campo contiene informazioni sull'entità che ha richiesto l'emissione del certificato, e viene ugualmente espresso attraverso un *distinguished name*;
- *subject public key information* → questo campo contiene la chiave pubblica vera e propria, specificando anche a quale sistema di crittografia a chiave pubblica essa appartiene;
- *issuer and subject unique ID* → questi due campi sono stati introdotti dalla versione 2 in poi, e hanno lo scopo di gestire l'unicità dei rispettivi nomi qualora essi siano stati utilizzati da diverse entità;
- *extension* → dalla versione 3 in poi è possibile specificare informazioni aggiuntive su un certificato, utilizzando il campo estensione. Un'estensione può essere definita come critica

o non critica: se in fase di verifica di un certificato si riscontra un'estensione sconosciuta ed essa è marcata come critica, come conseguenza si ha l'automatico rifiuto del certificato; viceversa se essa viene marcata come non critica e non viene riconosciuta, l'estensione viene semplicemente ignorata. Esistono inoltre due tipologie di estensioni: una pubblica, definita dallo standard e nota a tutti, e una privata, nota solo ad una cerchia ristretta.

Le estensioni pubbliche sono divise in quattro differenti classi:

- *key and policy information* → contiene informazioni sulle procedure che si possono seguire e specifica per quali applicazioni la *Certification Authority* garantisce l'utilizzo del certificato. Se marcata come critica, le applicazioni sono esclusivamente quelle definite, se marcata come non critica, il possessore può utilizzarla in altre modalità, ma in questo caso sarà l'unico responsabile. A seconda delle opzioni specificate in questa estensione, è possibile utilizzare il certificato: per firmare un'altra chiave pubblica, per la firma digitale, per il non ripudio, per criptare i dati, per la firma dei certificati e delle liste di revoca; è inoltre possibile utilizzare il certificato negli algoritmi di *key agreement*, come ad esempio il *Diffie Hellman*;
- *certificate subject and certificate issuer attributes* → fornisce informazioni aggiuntive sul possessore del certificato. È sempre considerata come estensione critica se il campo *subject name* è vuoto, e consente di esprimere attraverso nomi alternativi l'identità del possessore (ad esempio attraverso il suo indirizzo IP, un URL, un URI, ...);
- *certificate path constraints* → applica delle restrizioni sulle *Certification Authority* che fanno parte della catena e di cui non ci si fida. Una restrizione di tipo base specifica se un possessore può agire da *Certification Authority* e, in tal caso, il numero massimo di *Certification Authority* che essa stessa può nominare. Nelle restrizioni nominative sono presenti l'insieme di regole formali sulla nomenclatura che le *Certification Authority* devono utilizzare nella creazione di un certificato. Una restrizione di tipo applicativa specifica quali applicazioni per un certificato possono essere fornite da una *Certification Authority* emittente;
- *CRL distribution points* → specifica la locazione della lista di revoca da utilizzare per verificare la validità del certificato. Può trattarsi di una cartella, di una mail o di un indirizzo URL.

Come si vede dalla figura A.2, tutte le informazioni contenute nel certificato vengono firmate dalla *Certification Authority* che lo sta emettendo. A questo punto il richiedente può scaricare il certificato, installarlo sul proprio *server* e quindi essere in grado d'ora in poi di stabilire una connessione sicura con i suoi *client*, utilizzando SSL.

A.2 Il Protocollo SSL

Per poter comunicare utilizzando SSL, *client* e *server* devono concordare alcuni parametri di criptaggio dei dati. Vi è dunque una fase di negoziazione prima di poter procedere con l'effettiva comunicazione attraverso un canale SSL sicuro. Essa è composta da diversi step come mostrato dalla figura A.3.

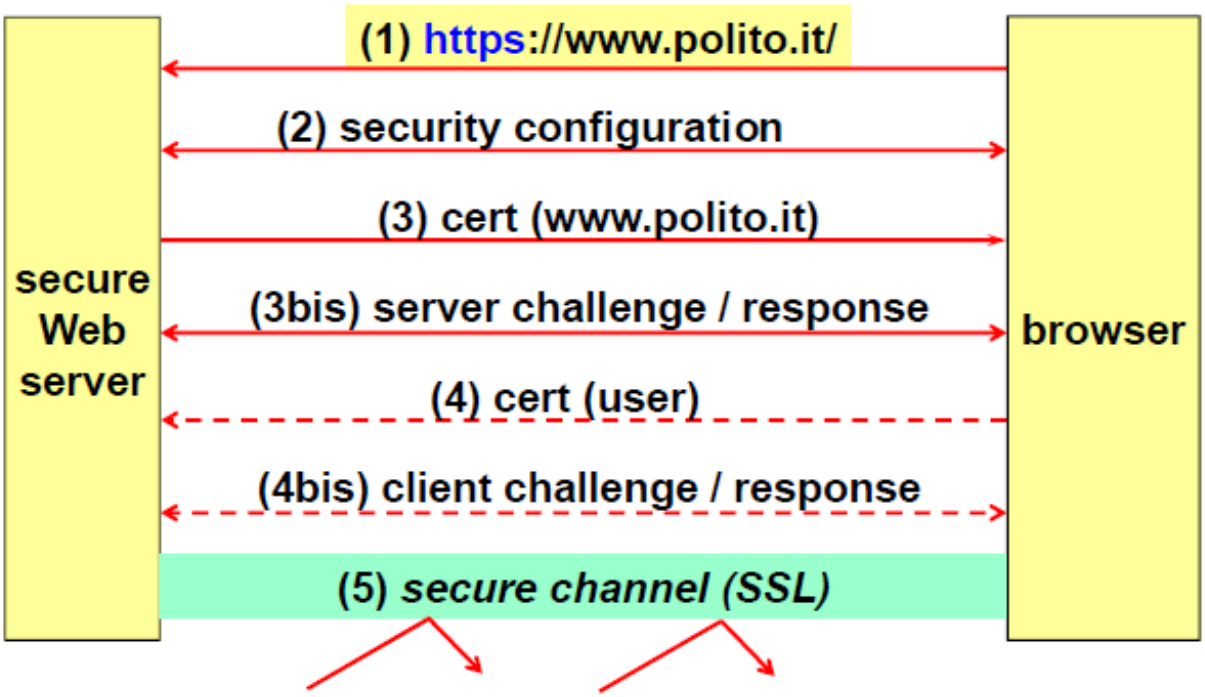

Figura A.3. SSL Handshake

Ogni volta che un *client* richiede l'accesso ad un *server*, bisogna procedere con la negoziazione dei parametri crittografici, anche nel caso in cui il *client* abbia già comunicato con il *server* in questione. Questa fase ripetuta nel tempo, pur mantenendo un alto livello di sicurezza, causa una diminuzione delle performance. Al fine di trovare un compromesso tra mantenere la sicurezza ed evitare il decadimento delle performance, SSL ha introdotto l'utilizzo di sessioni a cui è associato un identificativo.

Una sessione è una connessione logica tra *client* e *server*, che contiene i parametri negoziati la prima volta che i due *peer* hanno comunicato. Il *server* assegna un identificativo di sessione al relativo *client* e lo comunica ad esso in maniera criptata. L'ID di sessione viene ritenuto valido solo per un breve arco di tempo, così da continuare a garantire le proprietà di sicurezza. Utilizzando questa tecnica, un *client* in possesso di un ID ancora valido, può direttamente procedere con la comunicazione senza doversi preoccupare della ri-negoziazione dei parametri.
In particolare, le proprietà di sicurezza offerte da SSL sono le seguenti:

- autenticazione dei due *peer*, di cui è obbligatoria solo l'autenticazione del *server* e opzionale quella del *client*;
- riservatezza dei messaggi, ottenuta mediante criptaggio simmetrico dei dati;
- autenticazione e integrità dei messaggi, ottenuta grazie all'utilizzo di un *keyed digest*;
- protezione da attacchi di tipo *replay* o *filtering*.

A.2.1 Autenticazione in SSL

Durante la fase di *handshake* mostrata in figura A.3, oltre a concordare gli algoritmi da utilizzare in fase di comunicazione, avviene l'autenticazione del *server* e, se richiesto, anche quella del *client*.

L'autenticazione del *server* è obbligatoria al fine di proteggere il *client* da attacchi di tipo *shadow server* [5], assicurando dunque la reale identità del *server* con il quale si sta comunicando. Un *server* conferma la sua identità mediante l'invio della sua chiave pubblica, quindi attraverso un certificato X.509. Qualora un attaccante volesse spacciarsi per esso, il *client* ne verrebbe subito a conoscenza poiché un certificato identifica in maniera univoca il suo possessore. In aggiunta all'invio del certificato, in fase di autenticazione, viene anche utilizzato un sistema a sfida asimmetrico.

L'introduzione dei meccanismi a sfida deriva dalla necessità di fronteggiare i problemi riguardanti la debolezza dei sistemi di autenticazione, problemi che dipendono sia dall'utente (vengono ad esempio scelte password troppo deboli e dunque facilmente indovinabili), ma anche dal sistema (le password possono essere lette durante la trasmissione, o vengono salvate in chiaro sul *server* o viene utilizzato un *digest* non protetto e ci si espone ad attacchi di tipo *dizionario* [6]).
Sono dunque stati istituiti due sistemi a sfida, uno simmetrico e uno asimmetrico.

- **Sistema a sfida simmetrico**

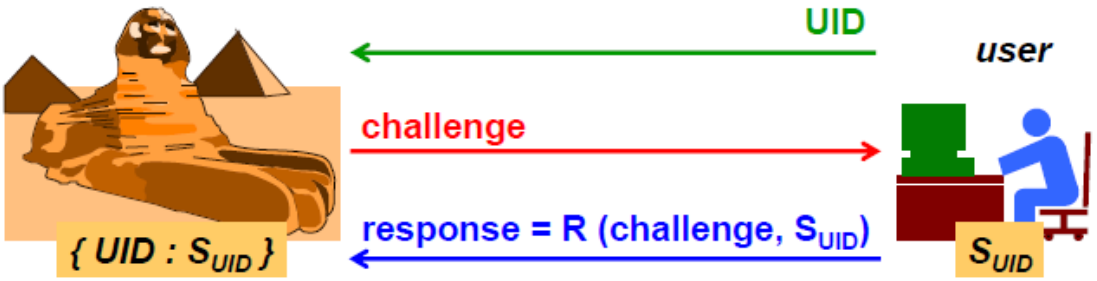

Figura A.4. Sistema a Sfida Simmetrico

Un utente che vuole identificarsi presso un *server*, di norma invia il proprio identificativo, successivamente il *server* risponde con una sfida, che consiste in un *random nonce* [7], e il *client* dovrà procedere con il calcolo della risposta alla sfida ricevuta. Essa è ottenuta mediante l'utilizzo di una funzione di hash che dipende da due parametri, la sfida e il segreto che si deve trasmettere al *server*. Ricevuta la risposta, il *server* ne ricalcola una propria (grazie al fatto che il segreto è ancora salvato in chiaro sul *server* stesso) al fine di confrontarla con quella inviatagli dal *client*.

Qualora si volesse procedere con l'autenticazione mutua, la chiave che viene utilizzata per calcolare il valore di hash o per criptare la sfida, è quella condivisa tra *client* e *server*. L'utente dopo aver risposto alla sfida inviatagli dal *server*, ne invia un'altra a sua volta. Al fine di migliorare le performance, è possibile ridurre il numero di messaggi scambiati nel processo di mutua autenticazione procedendo come segue:

1. l'utente invia il proprio identificativo insieme alla sfida per il *server*;
2. il *server* cripta la sfida inviatagli utilizzando la chiave condivisa e la invia all'utente insieme ad un'altra sfida;

[5] Consiste nel mostrarsi alla vittima al posto del *server* reale. Nel farlo è necessario l'*address spoofing*, il *packet sniffing*, che lo *shadow server* risponda più velocemente del *server* reale o che quest'ultimo sia sotto attacco *denial of service*.

[6] L'attaccante ha a disposizione un database contenente una grande varietà di possibili password utilizzabili dagli utenti. Se conosciuto l'algoritmo di hash, l'attaccante può completare il suo database inserendo accanto alla password il corrispondente valore di hash. A questo punto una volta intercettato un valore hash, deve limitarsi a cercare nel suo database per ottenere la password corrispondente.

[7] Un *nonce* è un numero calcolato in maniera random, casuale e che viene utilizzato una ed una sola volta. Questo permette di evitare attacchi di tipo *replay* e pre-computazionali.

3. l'utente cripta a sua volta la sfida ricevuta utilizzando la stessa chiave condivisa e la rimanda al *server*;

Sfruttando il meccanismo di mutua autenticazione, un attaccante che vuole fingere di essere un altro utente, è in grado di autenticarsi col *server* pur non conoscendo il segreto del vero utente, e questo si verifica perchè è possibile stabilire due connessioni diverse nelle quali viene utilizzata la stessa chiave condivisa. Per far ciò l'attaccante procede come segue:

1. l'attaccante invia al *server*, attraverso una prima connessione, l'identificativo di un altro utente insieme ad una sfida;
2. il *server* risponde con la sfida, criptata utilizzando la chiave pubblica, e con un'altra sfida;
3. l'attaccante a questo punto non conosce la chiave condivisa, apre dunque una seconda connessione sfruttando lo stesso identificativo utente e manda al *server* la sfida che ha appena ricevuto;
4. il *server* risponde sulla seconda connessione con la sfida criptata e con una terza sfida;
5. l'attaccante possiede adesso la risposta alla prima sfida, la invia sulla prima connessione e chiude la seconda.

Con questo primo meccanismo non vengono dunque risolti tutti i problemi, anzi se ne aggiungono di nuovi. Il *server* continua a mantenere i segreti in chiaro, il *client* deve essere in grado di calcolare un *digest*, quindi di utilizzare una funzione di *hash*, ed è soprattutto possibile rubare l'identitità di un utente. La soluzione a questi problemi è fornita dall'utilizzo del secondo protocollo a sfida, quello asimmetrico.

- **Sistema a sfida asimmetrico**

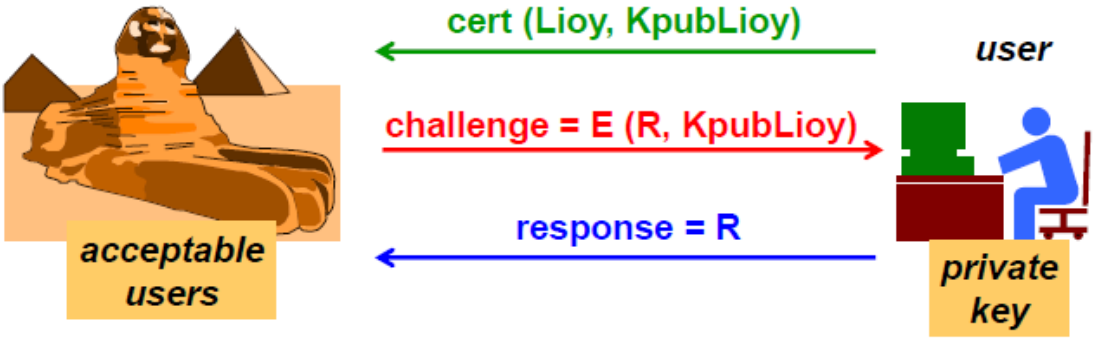

Figura A.5. Sistema a Sfida Asimmetrico

Il primo miglioramento apportato dal protocollo a sfida asimmetrico è il mantenimento di una lista di utenti accettati, evitando dunque il salvataggio dei segreti in chiaro e dando un minor carico di lavoro al *server*, il quale non deve mantenere un alto livello di sicurezza. Inoltre in questo sistema vengono utilizzati i certificati a chiave pubblica e privata, il che elimina sia il problema della condivisione delle chiavi, sia la possibilità per un attaccante di rubare l'identità ad un altro utente.

Quando un utente vuole autenticarsi manda al *server* il proprio certificato con la relativa chiave pubblica, quest'ultima viene utilizzata dal *server* per criptare un *random nonce*, cioè la sfida, e lo invia all'utente. Dal momento che la chiave privata associata ad un certificato non viene mai condivisa, solo il reale possessore di quest'ultima sarà in grado di decriptare la sfida e mandarla in chiaro al *server*.

Indubbiamente sono presenti degli svantaggi anche in questa tipologia di protocollo a sfida. Ad esempio bisogna essere certi che il certificato provenga da una *Certification Authority* abilitata al rilascio di quella tipologia di certificato, ma nonostante ciò lo si può comunque ritenere un protocollo estremamente forte.

A.2.2 Riservatezza dei Messaggi

Prima di essere inviati attraverso un canale di comunicazione, è buona norma che i dati vengano criptati, così da rendere le informazioni scambiate illegibili a terze parti. La riservatezza è dunque ottenuta attraverso l'utilizzo di un algoritmo di criptaggio simmetrico e di una chiave nota sia al *client* che al *server*; questo è necessario poiché se uno si occupa di criptare con una determinata chiave, l'altro dovrà decriptare con la medesima chiave. Nel caso specifico di SSL è il *client* a generare una chiave e a renderla nota al *server* mediante un algoritmo di crittografia a chiave pubblica. Durante la configurazione del canale sicuro, è indispensabile quindi concordare quale algoritmo, tra i vari permessi in SSL, si vuole utilizzare.
Gli algoritmi di criptaggio simmetrico, messi a disposizione da SSL, sono molteplici.

Nome	**Chiave**	**Blocco**
DES	56 bit	64 bit
3-DES	112 bit	64 bit
3-DES	168 bit	64 bit
IDEA	128 bit	64 bit
RC2	8-1024 bit	64 bit
RC4	variabile	stream
AES	128-256 bit	128 bit

Tabella A.1. Algoritmi di criptaggio simmetrico

Si distinguono due tipologie di algoritmi, una che lavora su blocchi di dati e un'altra che invece gestisce un flusso di dati.

Gli **algoritmi a blocchi** utilizzano appunto blocchi di dimensione fissa, il cui valore varia a seconda dell'algoritmo. Il primo passo, eseguito da tutti gli algoritmi di questa categoria, è quello di dividere i dati da criptare in N blocchi di uguale grandezza; raramente la dimensione dei dati da inviare è un multiplo della dimensione scelta dall'algoritmo. Quindi il secondo passo è quello di uniformarne la dimensione tramite l'utilizzo di diverse tecniche, sia per gestire il caso in cui la dimensione dei dati supera quella richiesta dall'algoritmo, sia per fronteggiare una dimensione inferiore a quella richiesta. In generale gli algoritmi a blocchi si differenziano per la lunghezza della chiave utilizzata, per la dimensione dei blocchi preferita e per le operazioni che effettuano in fase di criptaggio. Alcuni sono più performanti se implementati a livello hardware (poiché utilizzano funzioni che richiederebbero un lungo tempo computazionale se implementati a livello software), e questo è il caso di DES e 3-DES, altri funzionano bene se implementati a entrambi i livelli.

Un **algoritmo di tipo stream** lavora invece con flussi di dati, la cui dimensione può essere di un bit o di un byte. In questo caso, piuttosto che criptare un blocco di dati utilizzando una chiave, essa viene combinata ai bit che devono essere inviati, solitamente mediante operazioni di XOR. Essenzialmente i bit non vengono criptati in blocco, ma singolarmente, utilizzando il bit corrispondente della chiave stessa, che viene solitamente generata in maniera random o pseudo-random a partire da un seme. Questo stesso seme deve essere utilizzato a destinazione per generare la chiave da utilizzare in fase di decriptazione e ciò rende necessario che sorgente e destinazione concordino quale seme utilizzare per la generazione delle chiavi.
Questa tipologia di algoritmo è più veloce rispetto alla precedente, non richiede particolari implementazioni a livello hardware, ma è anche maggiormente esposta ad attacchi attivi. Se ad esempio un attaccante modifica un bit tra i dati criptati, questa modifica si ripercuoterà anche sul bit non criptato. Inoltre, non si dovrebbe utilizzare una stessa chiave per criptare due diversi messaggi, poiché un attaccante che intercetta ambedue i messaggi, riuscirebbe facilmente a decriptarli entrambi, e questa debolezza dipende dalle proprietà intrinseche della funzione XOR utilizzata.

Per poter criptare i dati con una determinata chiave, è necessario renderla nota al *peer* con il quale si sta comunicando; SSL utilizza in particolare degli algoritmi di crittografia a chiave pubblica asimmetrici per lo scambio delle chiavi. Anche questo algoritmo deve essere concordato in fase di settaggio del canale ed è possibile scegliere tra i seguenti:

- *RSA - Rivest Shamir Adleman* → nella creazione della chiave pubblica e di quella privata sono indispensabili tre valori: un numero pubblico (identificante la lunghezza della chiave), un esponente pubblico (utilizzato per la creazione della chiave pubblica) e un esponente privato (usato nella creazione della chiave privata). Tutti e tre i valori dipendono da due numeri primi, molto grandi, che devono essere mantenuti segreti e che devono essere cancellati una volta generate le chiavi. Nel calcolo dei valori è anche indispensabile l'utilizzo dell'operatore modulo [8]. I passi che vengono dunque eseguiti da questo algoritmo sono i seguenti:

  1. scelta dei due numeri primi P e Q;
  2. calcolo del numero pubblico a tutti: N = P x Q;
  3. calcolo di PHI = (P-1)(Q-1), indispensabile nella scelta e calcolo dei due esponenti;
  4. scelta dell'esponente pubblico E, il cui valore deve essere compreso tra 1 e PHI;
  5. calcolo dell'esponente privato $D = E^{-1} \bmod PHI$;

  La chiave pubblica dipende dunque da N ed E, mentre quella privata dipende da N e D; a questo punto, una volta che le chiavi sono state generate, è possibile eliminare P e Q. Questo algoritmo può anche essere utilizzato per criptare e decriptare dati, purchè la loro lunghezza sia minore del valore di modulo N. I valori del testo criptato (*ciphertext*) e del testo decriptato (*plaintext*) sono quindi calcolati nel seguente modo:

  – $C = P^E \bmod N$;
  – $P = C^D \bmod N$;

  Solitamente il valore di E è composto da un numero binario contenente solo due bit ad 1, e questo rende l'operazione di elevamento a potenza molto semplice e veloce. Un possibile attacco potrebbe essere quello di fornire una firma contenente molti bit con valore 1, provocando un grosso carico computazionale. Inoltre questo stesso esponente ha spesso valori piccoli, e se lo stesso valore di E viene utilizzato per cifrare messaggi diretti a persone diverse, si rischia che essi siano decriptabili anche da terzi. Una soluzione a questo problema è l'aggiunta di un *salt* [9] al messaggio prima di criptarlo. Un'altra debolezza di RSA è che la stessa chiave viene utilizzata sia per la firma che per il criptaggio, ciò permette di decriptare facilmente un messaggio se si riesce ad obbligare un utente a firmarlo con la medesima chiave. È dunque indispensabile utilizzare due chiavi diverse, una in fase di criptaggio e l'altra per la firma digitale.

- *DH - Diffie Hellman* → in questo caso, la chiave pubblica e la chiave privata vengono generate a partire da due interi, concordati dai due comunicanti, e sempre mediante l'utilizzo dell'operatore modulo. I passi per la creazione delle chiavi sono i seguenti:

  1. A e B scelgono due interi, p e g, dove g è compreso tra 1 e il valore di p;
  2. A sceglie un intero positivo x, e calcola $X = g^x \bmod p$;
  3. B sceglie un intero positivo y, e calcola $Y = g^y \bmod p$;
  4. A e B pubblicano i due valori appena calcolati, X e Y;
  5. A calcola la propria chiave: $K_A = Y^x \bmod p$;
  6. B calcola la propria chiave: $K_B = X^y \bmod p$;

  Grazie a questo algortimo, i due *peer* riusciranno a comunicare con la stessa chiave pur non avendola mai scambiata. Mediante semplici sostituzioni matematiche, è infatti banale verificare che

  $$K_A = K_B = g^{xy} \bmod p.$$

  Dal momento che i due numeri scelti arbitrariamente da A e B non vengono mai resi noti, un attaccante non potrà riuscire a trovare la chiave e quindi avere accesso al messaggio.

[8] L'operatore modulo (*mod*) ritorna il resto della divisione tra i due operandi

[9] È un valore lungo, generato in maniera random, e che non contiene caratteri di controllo o comuni.

A.2.3 Autenticazione e Integrità dei Messaggi

Un attaccante che intercetta un messaggio, anche se non riesce a comprenderlo perchè criptato, può comunque modificarne il contenuto in maniera inattesa. Per evitare che ciò accada, e garantire dunque l'integrità dei messaggi, viene utilizzato un *digest*. Un *digest* è un messaggio di lunghezza fissa che contiene il "riassunto" del messaggio da inviare, qualsiasi sia la sua lunghezza. Perchè questo sistema funzioni, un *digest* deve essere facile da calcolare, difficile da invertire e soprattutto deve identificare in maniera univoca il messaggio relativo. Il *digest* viene calcolato mediante l'utilizzo di algoritmi di hash, i quali permettono di trasformare i dati in una stringa binaria di lunghezza fissa. Il messaggio viene diviso in diversi blocchi di uguale dimensione e su ogni blocco viene applicata una funzione che tiene conto anche del blocco precedente (nel caso del primo blocco, viene utilizzato un *initialization vector* [10]). Questa concatenazione tra i blocchi del messaggio, fa si che il valore di hash finale, ovvero il *digest*, sia rappresentativo dell'intero messaggio.

Nome	Blocco	Digest	Note
MD2	8 bit	128 bit	
MD4	512 bit	128 bit	
MD5	512 bit	160 bit	
RIPEMD	512 bit	160 bit	
SHA-1	512 bit	160 bit	
SHA-224	512 bit	224 bit	SHA-2
SHA-256	512 bit	256 bit	SHA-2
SHA-384	1024 bit	384 bit	SHA-2
SHA-512	1024 bit	512 bit	SHA-2

Tabella A.2. Algoritmi crittografici di Hash

Tra tutte le tipologie indicate nella tabella A.2, i più utilizzati sono MD5 e SHA1, anche se quest'ultimo è stato recentemente esposto ad un attacco di tipo collisione [11].

Se si vuole anche procedere con l'autenticazione, e questo è il caso di SSL, è possibile utilizzare un *keyed digest*. In questo caso il *digest* viene calcolato tenendo conto sia del messaggio, che della chiave; così facendo solo il possessore della chiave può confrontare il *digest* appena ricevuto con quello calcolato da lui stesso.

Bisogna essere prudenti nella creazione del *keyed digest* poiché l'integrità potrebbe essere compromessa in uno dei seguenti modi:

1. se kd = H(K ‖ M) è possibile aggiungere altri dati alla fine del messaggio;
2. se kd = H(M ‖ K) è possibile anteporre un nuovo messaggio prima di M.

dove H è la funzione di hash, K è la chiave e M è il messaggio.
Al fine di evitare questo tipo di attacchi è dunque consigliabile usare uno degli standard definiti per la creazione del *keyed digest*. Lo standard utilizzato da SSL è *HMAC (Hash Message Autentication Code)*, quest'algoritmo richiede la creazione di un doppio *digest*, ovvero la funzione di hash viene applicata due volte su un blocco di dati di dimensione B (byte). Se la chiave ha una lunghezza maggiore di B, la chiave che verrà utilizzata sarà quella ottenuta applicando la funzione di hash alla stessa (K' = H(K)), viceversa se la chiave è minore della dimensione del blocco dei dati, sarà ottenuta aggiungendo tanti 0 fino al raggiungimento del valore di B. La funzione di hash utilizzata è la seguente:

$$\mathrm{HMAC(M)} = \mathrm{H(K' \oplus opad \parallel H(K' \oplus ipad \parallel data))}$$

[10] È un blocco di bit random (o pseudo-random)di lunghezza fissa, molto utilizzato negli algoritmi di criptaggio.

[11] Si ha una collisione quando uno stesso valore di hash viene generato per identificare due diversi messaggi.

dove M è il messaggio, ipad ha valore 0x36, ripetuto B volte, e opad ha valore 0x5C, anch'esso ripetuto B volte. Affinchè l'integrità venga garantita, la funzione di hash deve essere non invertibile e non deve generare collisioni. Dovendo applicare due volte la funzione di hash, questo algoritmo risulta inevitabilmente più lento rispetto agli altri, ma allo stesso tempo è più difficile da attaccare.

A.2.4 Settaggio del Canale Sicuro

Vediamo adesso nel dettaglio come avviene il settaggio di un canale sicuro in SSL.

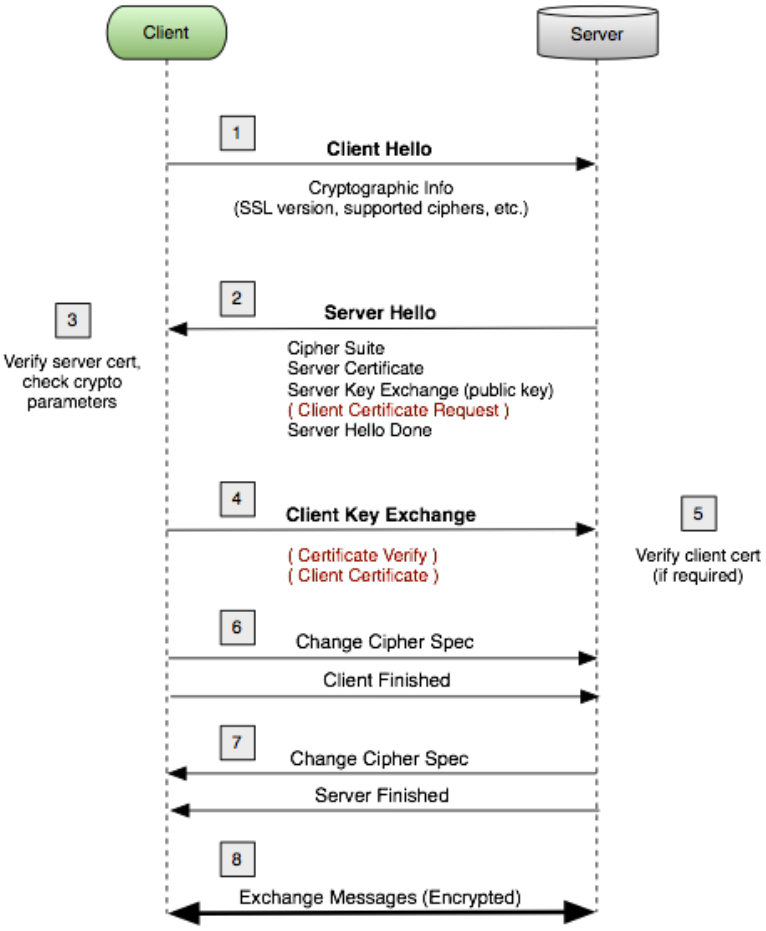

Figura A.6. SSL Handshake

Come mostrato dalla figura A.6, gli step necessari per stabilire un canale sicuro sono i seguenti:

1. Il *client* che vuole comunicare con un *server* attraverso un canale sicuro, invia un primo messaggio di "*Hello*" contenente le sue preferenze riguardo la crittografia. Nello specifico i campi sono i seguenti:
   - *key* → specifica il metodo per lo scambio delle chiavi (RSA, Diffie Hellman);
   - *cipher* → indica l'algoritmo di criptaggio dei dati (RC4, 3DES, AES);
   - *hash* → definisce la funzione di hash che si vuole utilizzare per l'autenticazione del messaggio al fine di garantirne l'integrità (HMAC-MD5, HMAC-SHA);
   - *version* → identifica la versione di SSL supportata dal *client*;
   - *random number* → è utilizzato per generare il *master secret*, è un valore condiviso tra *client* e *server* ed è indispensabile per la generazione delle chiavi utilizzate in fase di comunicazione.

2. Il *server* risponde al *client* con un altro messaggio di "*Hello*", stavolta contenente le specifiche che lui stesso ha scelto, e comunicando al *client* l'ID di sessione ad esso assegnato. È in questa fase che il *server* si autentica, inviando il suo certificato, e richiede quello del *client* se necessario;
3. Una volta ricevuto il certificato del *server*, il *client* si preoccupa di effettuare i controlli opportuni e in caso di esito positivo risponde con un altro messaggio contenente le chiavi simmetriche da lui generate. Se è stato richiesto, il *client* invia inoltre il proprio certificato;
4. Il *client* avverte dunque il *server* che è possibile passare alla comunicazione protetta, e lo fa inviando due messaggi: "*Change Cipher Spec*" e "*Finished*". Quest'ultimo è il primo messaggio ad essere protetto utilizzando le specifiche appena negoziate;
5. Il *server* a sua volta risponde con gli stessi due messaggi del *client*.
6. Il *client* e il *server* sono adesso in grado di comunicare in maniera sicura, utilizzando un canale criptato.

Quando vogliono terminare la comunicazione, *client* e *server* si scambiano un messaggio di notifica, avvertendosi reciprocamente che si sta chiudendo la connessione.
Se un *client* vuole creare una nuova connessione presso lo stesso *server*, gli basterà inviare l'ID di sessione ad esso associato. Se quest'ultimo è ancora valido, sarà possibile evitare la fase di negoziazione dei parametri e passare direttamente all'utilizzo di un canale sicuro, semplicemente scambiandosi i messaggi che riguardano lo switch tra comunicazione non protetta e comunicazione protetta.

A.2.5 Vantaggi e Svantaggi

Sono tanti gli attacchi a cui si è esposti nella comunicazione con un *server* e da cui SSL fornisce una protezione. Vediamo adesso nel dettaglio che caratteristiche hanno.

- *Replay e Filtering*
  In un attacco di tipo *replay*, se un attaccante riesce ad intercettare un pacchetto attraverso il *packet sniffing* [12] può ripeterne la trasmissione. In un attacco di tipo *filtering*, se l'attaccante intercetta un pacchetto ne impedisce l'arrivo a destinazione. SSL fornisce protezione da questo tipo di attacchi grazie al fatto che lavora sul protocollo TCP.
  Quest'ultimo, attraverso l'utilizzo di un *sequence number*, riesce a gestire autonomamente sia la cancellazione di un pacchetto, richiedendolo, che la ricezione di un duplicato, scartandolo;
- *Man in the Middle*
  Quando si comunica attraverso SSL, si passa dall'utilizzo del protocollo *HTTP* all'utilizzo di *HTTPS*, il quale protegge da attacchi di tipo *Man in the Middle*. Questo tipo di attacco consiste nell'alterare la comunicazione tra due *peer*, prendendo il controllo del canale di comunicazione e avendo dunque la capacità di manipolare il traffico in transito. Sostanzialmente i due *peer* comunicano, senza rendersene conto, con un terzo (l'attaccante), invece di comunicare tra di loro. SSL fornisce protezione da quest'attacco, poiché richiede l'autenticazione del *server* e garantisce l'integrità dei messaggi.

Si può notare facilmente che i vantaggi apportati dall'utilizzo di SSL sono veramente tanti, ma ciò non toglie che esso presenta anche degli svantaggi. A causa del fatto che tutti i dati trasmessi devono essere criptati, vengono utilizzate molte più risorse sul *server* rispetto ad una situazione in cui non bisogna criptare le informazioni, e quindi a risentirne sono sicuramente le performance. Questo problema è riscontrato maggiormente nei siti web che hanno un alto numero di visitatori, e potrebbe essere aggirato utilizzando dei componenti hardware più prestanti. Ciò introduce a sua volta il secondo problema, quello dei costi: per comunicare in maniera sicura attraverso SSL, bisogna settare un'infrastruttura adeguata e affidabile che, da una parte fornisca i certificati, dall'altra permetta la loro validazione.

[12] Un pacchetto trasmesso in rete viene intercettato da terzi prima che arrivi a destinazione.

Bibliografia

[1] M. Jones, J. Bradley, N. Sakimura, "Json Web Signature (JWS)", RFC-7515, May 2015

[2] S. Josefsson, "The Base16, Base32, and Base64 Data Encodings", RFC-4648, October 2006

[3] Y. Sheffer, R. Holz, P. Saint-Andre, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", RFC-7525, May 2015

[4] R. Fielding, J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC-7231, June 2014

[5] M. Duerst, L. Masinter, J. Zawinski, "The 'mailto' URI Scheme", RFC-6068, October 2010

[6] R. Barnes, J. Hoffman-Andrews, D. McCarney, J. Kasten, "Automatic Certificate Management Environment (ACME)", RFC-8555, March 2019

[7] Let's Encrypt - Leaving Beta, New Sponsors, https://letsencrypt.org/2016/04/12/leaving-beta-new-sponsors.html

[8] Let's Encrypt - Transitioning to ISGR's Root https://letsencrypt.org/2019/04/15/transitioning-to-isrg-root.html