# How It Works

The objective of Let's Encrypt and the ACME protocol is to make it possible to set up an HTTPS server and have it automatically obtain a browser-trusted certificate, without any human intervention. This is accomplished by running a certificate management agent on the web server.

To understand how the technology works, let's walk through the process of setting up `https://example.com/` with a certificate management agent that supports Let's Encrypt.

There are two steps to this process. First, the agent proves to the CA that the web server controls a domain. Then, the agent can request, renew, and revoke certificates for that domain.

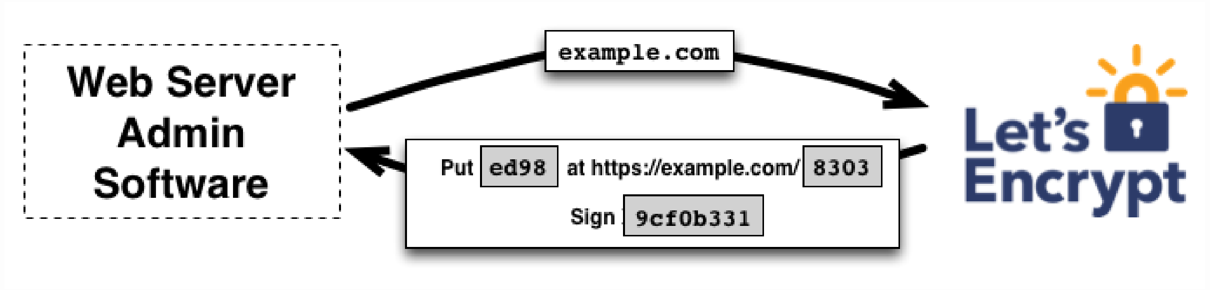
## Domain Validation

Let's Encrypt identifies the server administrator by public key. The first time the agent software interacts with Let's Encrypt, it generates a new key pair and proves to the Let's Encrypt CA that the server controls one or more domains. This is similar to the traditional CA process of creating an account and adding domains to that account.

To kick off the process, the agent asks the Let's Encrypt CA what it needs to do in order to prove that it controls `example.com`. The Let's Encrypt CA will look at the domain name being requested and issue one or more sets of challenges. These are different ways that the agent can prove control of the domain. For example, the CA might give the agent a choice of either:

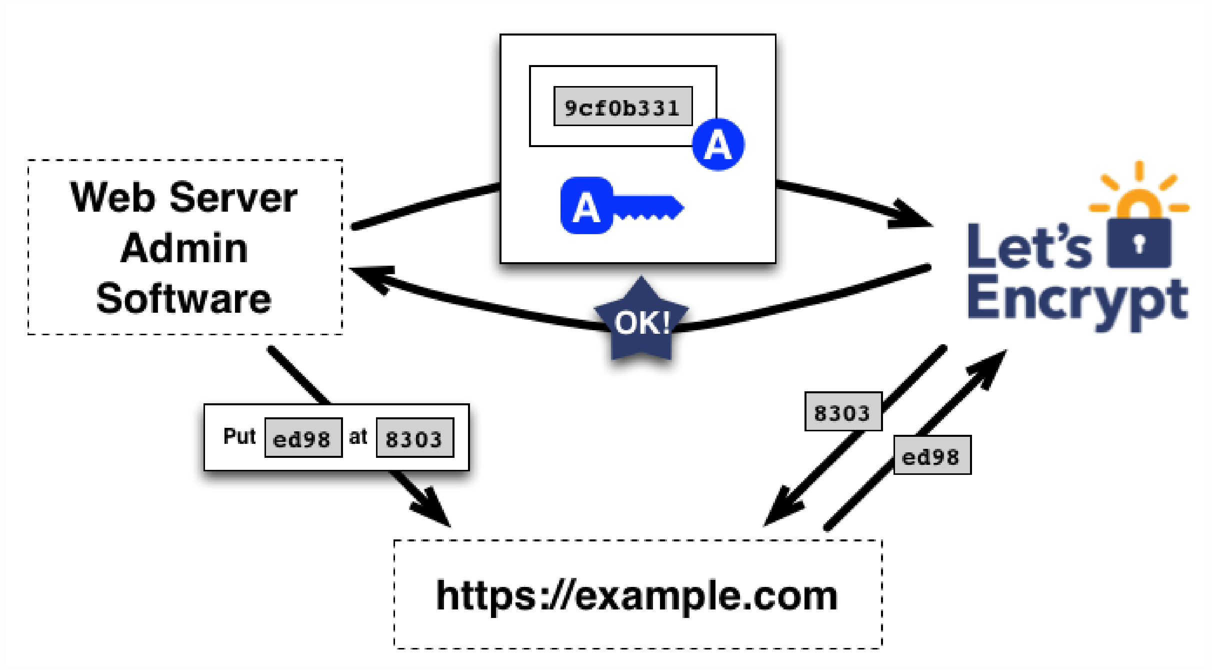
- Provisioning a DNS record under `example.com`, or
- Provisioning an HTTP resource under a well-known URI on `https://example.com/`

Along with the challenges, the Let's Encrypt CA also provides a nonce that the agent must sign with its private key pair to prove that it controls the key pair.


Web Server Admin Software
example.com
Put ed98 at https://example.com/ 8303
Sign 9cf0b331
Let's Encrypt

The agent software completes one of the provided sets of challenges. Let's say it is able to accomplish the second task above: it creates a file on a specified path on the `https://example.com` site. The agent also signs the provided nonce with its private key. Once the agent has completed these steps, it notifies the CA that it's ready to complete validation.

Then, it's the CA's job to check that the challenges have been satisfied. The CA verifies the signature on the nonce, and it attempts to download the file from the web server and make sure it has the expected content.



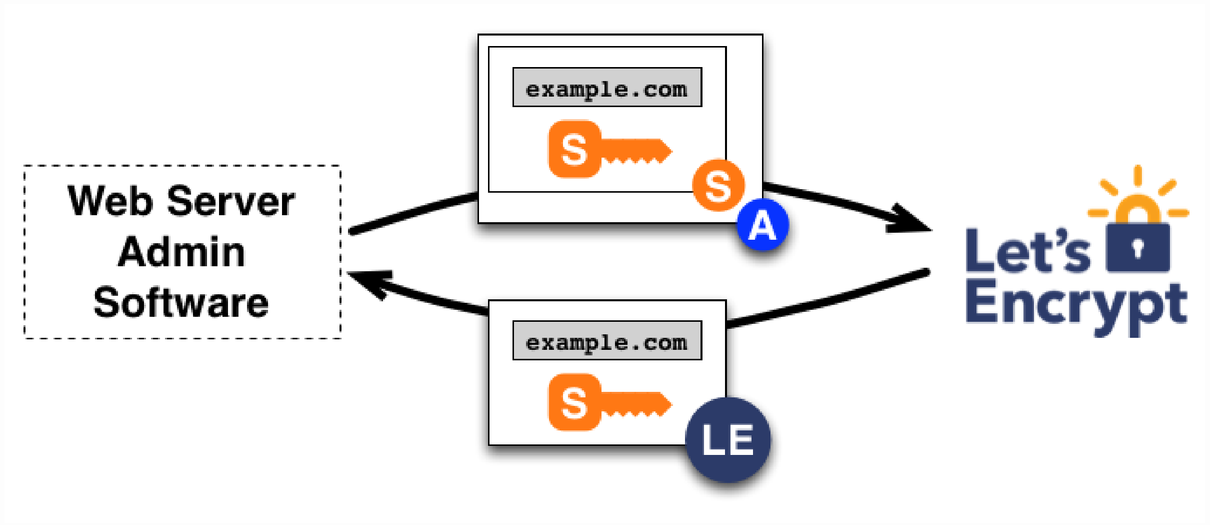
If the signature over the nonce is valid, and the challenges check out, then the agent identified by the public key is authorized to do certificate management for `example.com`. We call the key pair the agent used an “authorized key pair” for `example.com`.

## Certificate Issuance and Revocation

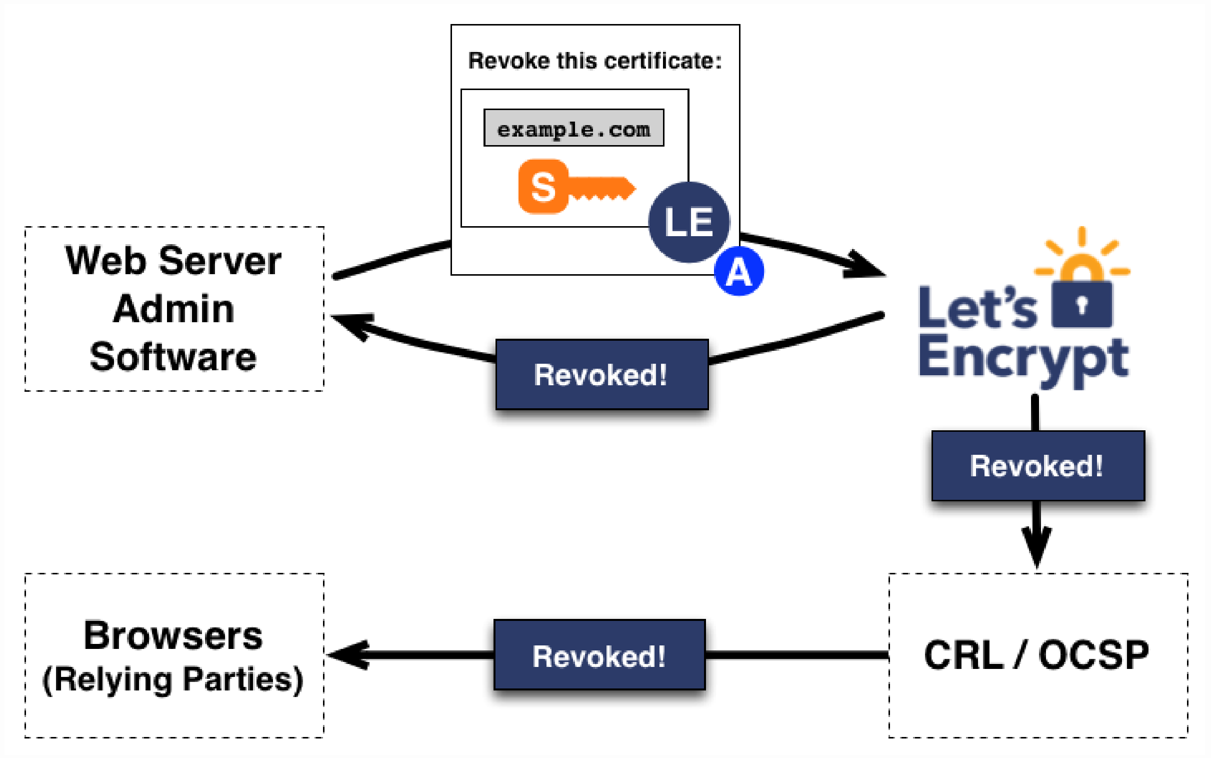
Once the agent has an authorized key pair, requesting, renewing, and revoking certificates is simple—just send certificate management messages and sign them with the authorized key pair.

To obtain a certificate for the domain, the agent constructs a PKCS#10 Certificate Signing Request that asks the Let’s Encrypt CA to issue a certificate for `example.com` with a specified public key. As usual, the CSR includes a signature by the private key corresponding to the public key in the CSR. The agent also signs the whole CSR with the authorized key for `example.com` so that the Let’s Encrypt CA knows it’s authorized.

When the Let’s Encrypt CA receives the request, it verifies both signatures. If everything looks good, it issues a certificate for `example.com` with the public key from the CSR and returns it to the agent.



Revocation works in a similar manner. The agent signs a revocation request with the key pair authorized for `example.com`, and the Let's Encrypt CA verifies that the request is authorized. If so, it publishes revocation information into the normal revocation channels (i.e. OCSP), so that relying parties such as browsers can know that they shouldn't accept the revoked certificate.



Support a more secure and privacy-respecting Web.

Donate

letsencrypt
letsencrypt

Let's Encrypt is a free, automated, and open certificate authority brought to you by the non-profit Internet Security Research Group (ISRG).

View our privacy policy.
View our trademark policy.

1 Letterman Drive, Suite D4700, San Francisco, CA 94129