Certificate Chains

PKI depends on Certificate chains, so that a signature applied by a Server to a TLS1.3 handshake exchange can be verified by a client. Each certificate owner has a secret signing key. The certificate contains the associated public key, and has appended a signature from the owner of the next certificate in the chain. The client hunts along this chain of certificates, checking signatures, until they find a recognised root certificate. Observe that the "issuer" of a certificate should be the "subject" of the next certificate in the chain, as we head towards the root. For the root certificate subject and issuer are identical – its self signed.

The server owns the "leaf certificate", and the simplest PKI-conformant configuration would be for this certificate to be signed by a root Certificate Authority. So a simple and short chain with two certificates would suffice, the leaf certificate and the CA certificate.

There is one problem with this – the exposure of the CA secret signing key. It has to be involved in every Server's certificate, and they are maybe billions of them. The security of PKI hinges on the CA signing key remaining a secret.

The mitigation is to introduce an Intermediate Certificate, so the chain length increases to three. The CA then is just required to sign a relatively small number of Intermediate certificates, and the compromise of an Intermediate secret key only damages a limited branch of the tree, rather than the whole tree.

There is no compelling reason for chains longer than three.

The root CA certificate

Next issue that arises is whether or not to include the root CA in the chain. The client needs to know that the root CA is genuine, but simply including it in the chain does not achieve that. The common solution is for a browser to have access to a recognised database of root CAs – the Trust Store. Then it is sufficient that the issuer of the intermediate certificate should be the subject of a root CA, and that the signature applied to the intermediate certificate is verified by the public key of that root CA.

Now since the root CA is available from the Trust Store, there seems to be no good reason to also include it in the chain. So we can shorten the chain back down to a length of two again.

From our list of 20 random websites, only two include the CA certificate in the chain (one example is www.bbc.co.uk). There are however apparently use cases where it should be (see OCSP stapling).

However I think I could make a case for including it. If the CA root cert were included in the chain, then we would only be required to maintain a database of the hashes of root CAs, and chain integrity could be confirmed by comparing the hash of the root CA as received against that which is stored. This would dramatically decrease the size of the Trust Store, which would permit a much smaller footprint for the client. The downside would be somewhat longer chains to be communicated and buffered.

Misconfigured chains

Sloppiness is the enemy of security. Sloppy programming has led to innumerable failures and hacks in the past. Therefore you would think it appropriate to take a zero tolerance approach to badly configured certificate chains. However the actual response of browsers when presented with a bad chain is "let me see if I can fix that for you". The browser bends over backwards to accommodate the broken chain rather than rejecting it out of hand. Because in practice no-one will use an overly fussy browser that won't load some web pages. Yet again the quality of the User eXperience trumps security.

Let us look briefly at two examples iscc2021.unipi.gr and www.academia.edu. Ironically the first advertises a security conference, and the latter is for academic researchers. Both issue horribly misconfigured certificate chains.

The iscc2021.unipi.gr chain consists of a server certificate and an intermediate certificate which appear not to be connected. The issuer of the server cert is not the subject of the intermediate certificate, and the server cert has a 4096-bit RSA signature, while the public key of the intermediate certificate is a mismatched 2048-bit RSA public key. Clearly its the wrong intermediate certificate. This is confirmed by visiting https://whatsmychaincert.com/, which confirms that the chain is misconfigured and by some magic finds the correct intermediate certificate. And clearly browsers do the same, which probably involves doing a time-wasting search for the correct cert on every connection.

The www.academia.edu chain is a complete mess. First the server certificate is mistakenly included twice, and secondly the chain is of total length four (five including the duplicate). That we might have to look for duplicate certificates and process them never occurred to me. This would be quite overwhelming for a small client. However browsers do not complain, and the page loads.

Zero Tolerance

Now since we are trying to implement a secure TLS1.3, and since security matters to us more than it does to others (we are not planning to implement a standard browser), I suggest that it is entirely appropriate to take a zero-tolerance approach to malformed cert chains. The message to servers should be "Fix your certificate chain, or we will not deal with you". Such sloppiness should not be tolerated in any application with pretensions to genuine security.

Our current approach is to just process the first two certificates in the chain, and look for the issuer of the intermediate certificate in the Trust Store. Misconfigured chains are rejected, and we will make no effort to fix them – that is the server's responsibility.

At the risk of cutting ourselves off from the mainstream we might consider insisting that the complete chain be sent, so that we can drastically reduce the client footprint as described above. Or we may make it an option (see below).

What are the options?

Certificate chain verification is an important component of a TLS1.3 implementation. It is neatly encapsulated – the input is a certificate chain and the output is a boolean – good or bad, plus the servers public key and identity should the chain be good. This can be outsourced and considered as a separate, pluggable component.

So a range of pluggable cert chain checkers can be considered..

1. A full PKI compatible checker, which verifies a chain against the standard database of CA root certs. We can restrict to a well-formed two cert chain plus root CA from a large database while maintaining 95% compatibility with existing Web servers.

2. At the other extreme a bespoke compatibility checker consisting of a single self-signed certificate. This would work outside of a PKI context, but might be a nice fit for a small IoT deployment.

3. A range of intermediate configurations might be appropriate. For example like 1. but using a smaller pared-back root certificate store.