CB3: Public Key Infrastructure

In a symmetric key system, each sender and receiver pair needs to share a secret key, thus resulting in a significant amount of shared keys. The great advantage of public key cryptography is that it makes it feasible for parties to communicate securely with each other, even if they have never encountered each other before and do not have access to an online service.

Alice sends a signed message to Bob. Bob can trust this message without having previously seen Alice's certificate if both of these statements are true:

- Alice signed the message, and the signature verifies using Alice's public key from her certificate
- Alice's certificate is signed by the private key, which corresponds to the public key from a CA certificate. Bob already knows the CA certificate and is able to verify Alice's certificate using the CA certificate's public key.

This may be extended. Bob does not need to know the CA certificate that issued Alice's certificate. This CA certificate, call it Certificate (CA1), could have been issued by another CA, call it CA2. If Bob knows Certificate (CA2), and receives both Certificate (Alice) and Certificate (CA1) in the signed message, he can still trust Alice's message by verifying that Alice signed the message, that her certificate was issued by CA1 and that CA1's certificate was issued by CA2. This can obviously be extended any number of times until the certificate chain reaches a root certificate. A root certificate is a certificate that was signed by its own private key. The root key is the key to trusting the entire PKI. The root public key has to be distributed securely so that recipients do not receive the wrong key and so trust the wrong certificates. The root private key also must be protected very carefully – anyone who had access to the private key would in principle be able to set up an entire CA hierarchy made of compromised CAs, which would be trusted by everyone who knew the public key. For this reason in real-world PKI deployments, the root key is used as infrequently as possible and is kept and used on a machine that cannot be accessed from an external network.

The CAMP SCMS design features a CA hierarchy, with:

- A root CA that issues certificates for other CAs but not for vehicles or other end-entities
- Optionally, intermediate CAs (ICAs), which obtain their certificates from other CAs above them and also issue certificates for other CAs rather than end-entities. The advantage of using intermediate CAs is that if an intermediate CA is compromised, it is less catastrophic than if the root CA is compromised, so this gives the system more flexibility to introduce new CAs without running the risks incurred by using the root CA key. It is possible to use intermediate CAs in a cascade, so an intermediate CA is either validated by the root CA or the intermediate CA above it.
- Enrollment authorities that issue enrollment certificates (long-term certificate signing requests) for the end-entities. These enrollment certificates are used only to communicate with the SCMS, not with other vehicles or end-entities. Note: the lifetime of the certificate is currently assumed to be the lifetime of a car (e.g., 30 years). However, this still needs discussion as it influences the size of the internal blacklist and is hence a cost issue. Note: the certificate lifetime and the lifetime of the actual CA do not have to be equal.
- Pseudonym CAs that issue certificates for the applications on the cars

The CAMP SCMS also distinguishes between the CA, which actually signs the certificate and the RA, which approves certificate requests. This results in a system diagram that appears complicated at first glance. In fact, this aspect of the CAMP approach is fully in line with standard PKIs used elsewhere in government and industry. This complexity of the abstract architecture allows for flexibility and robustness in introducing new CAs, retiring old ones, and allowing different organizations to take responsibility for authorizing activities that they properly have jurisdiction over. The initial deployment may not require all the boxes on the diagram to be filled immediately; however, it is important for the initial system to support migration to the full CAMP SCMS architecture, even if this migration happens slowly.

Certificates

A certificate links its holder's public key to a statement about the holder, such as an identity or a list of permissions. The statement is trusted because it is attested to by a CA. A receiver checks that the statement is true about a particular signed message by first using the public key of the CA to verify the certificate and subsequently the sender's public key to verify the signature on the message. If the receiver trusts the CA, and the signature on the message verifies, then the receiver knows that the public key owner signed the message and therefore the statement (identity, permissions, etc.) can be trusted as true about the message sender.

The standard way of creating and trusting a certificate is:

- The certificate contains the public key
- The CA signs the certificate
- The receiver verifies the CA signature on the certificate and the public key holder's signature on the message

This requires two verifications on the receiver's side and further requires (with recommended cryptographic algorithm choices) 64 bytes on each certificate to contain the CA signature.

Implicit Certificates

Implicit certificates are a different way of creating and trusting a certificate. With implicit certificates, the certificate requester and the CA cooperate to derive a final public key from the seed public key that the requester submits with the request. Instead of including a signature in the certificate, the CA includes a reconstruction value. A message recipient can combine the reconstruction value with the CA's public key and the rest of the contents of the certificate to recover the certificate holder's public key. This public key is only correct if the reconstruction value was created by the CA. Therefore, the CA' s approval of the holder's public key is implicit, which means the public key only works if the CA was involved in creating it. This is different to an explicit approval as in standard certificates, where the public key's validity is explicitly confirmed by the CA signature.

The information flow for implicit certificates is:

- Certificate Creation
    - The certificate requester creates a seed public key
    - The CA calculates a mathematical transformation using the CA private key, the contents of the certificate, and the seed public key, to create:
        - A new public key for the certificate requester, the certified public key
        - A transformation that the certificate requester can use on the seed private key

CAMP LLC Proprietary. The information contained in this document is considered an interim work product and is subject to revision without notice. The content is provided as is, only for informational purposes with no express or implied warranties that the information is accurate, up-to-date or complete. Any reliance on the content is solely at the user's own risk. Contact the VSC5 Consortium if you have questions regarding the content.

        - A reconstruction value
    - The CA sends the certificate contents, the reconstruction value and the private key transformation back to the certificate requester
    - The certificate requester applies the private key transformation to the seed private key to obtain the certified private key
    - The certificate requester checks that the certified private key corresponds to the certified public key
- Certificate Use
    - The certificate holder (who was the certificate requester in the previous step) signs a message with the certified private key and attaches the certificate (contents + reconstruction value)
    - The receiver uses the certificate contents, reconstruction value and CA public key to recover the certified public key
    - The receiver verifies the signature on the message with the certified public key

Implicit certificates have the following advantages over standard (explicit) certificates:

- An explicit certificate contains a public key (which is an elliptic curve point) and a signature, while an implicit certificate contains only a reconstruction value (which is an elliptic curve point). An implicit certificate is therefore smaller by the size of the signature, which in this case is 64 bytes. (The private key transformation adds 32 bytes to the certificate response compared to a response for an explicit certificate, but this is less than the signature size and is only included in the certificate response, not in signed messages). It is important to note that more details on this topic can be found in Standards for Efficient Cryptography Group, “SEC 4: Elliptic Curve Qu-Vanstone Implicit Certificate Scheme (ECQV)”, Working Draft Version 0.97, March 2011, available from http://www.secg.org.
- The public key recovery operation and the signature verification can be combined into a single operation that takes approximately the same amount of time as required for a single verification. This is an advantage over explicit certificates, which require two verifications when assuming that the chain of trust ends at the authority issuing the certificate. However, this advantage applies only if a receiver verifies very occasionally. If the receiver verifies multiple messages signed by the same certificate, it is more efficient overall to recover the public key once and cache it. In this case, implicit and explicit certificate verification takes about the same time. A significant population of devices that verify only occasional messages and verify in software is anticipated and for these devices the performance advantages of implicit certificates are very important.

Implicit certificates are covered by patents owned by Certicom Corp. of Mississauga, Ontario, which is currently a wholly-owned subsidiary of BlackBerry Ltd. At the time of this document, there has been an agreement reached between Certicom and the Institute of Electrical and Electronics Engineers (IEEE) concerning the use of the associated patents. OEM lawyers should review this agreement carefully to determine whether it is acceptable and understand what alternatives might exist.

Detailed Comparison of Explicit and Implicit Certificate Calculations

There are two cases to consider: verifying the certificate chain and message signature and the case where only the message signature is being verified.

Explicit Certificates

Let us first focus on the case of verifying the certificate chain and message signature. In this case, one needs to verify the message signature and the signature on each of the certificates. Verifying requires to perform a “double multiply and add,” i.e., calculating aX + bY, where X and Y are elliptic curve points and a and b are integers. Let us denote the cost for one double multiply and add by V. The cost for full certificate chain verification is V * n, where n is the length of the chain.

Once the full chain is verified, the following information is cached:

[Cert ID, public key, “successfully verified”]

This means that any time a message signed by that certificate is received, only one verification step needs to be performed: a lookup of the certificate establishes that it already has been verified. The cached public key is used to verify the message. The computational cost of this reads V.

Summarizing, the total cost for verifying a certificate chain using explicit certificates reads V * n for the first verify and V for the subsequent ones.

Implicit Certificates

Verifying a message signed with an implicit certificate can be done in two steps: extracting the public key from the certificate and verifying the message. To extract the public key from a certificate, the public key from the issuer’s certificate is required. The public key extraction operation is also a double-multiply-and-add. Thus, verifying an implicit certificate chain can be done using V * (n + 1) operations: V for extracting the public key, and V * n for verifying the certificate chain. At the end of the operation,

[Cert ID, public key, “successfully verified”]

is cached. Subsequent messages signed by that certificate can be verified at a cost of V.

Summarizing, the total cost for verifying reads V * (n + 1) for the first verify, and V for the subsequent ones. This is slightly higher than for the explicit certificates case, but it should be observed that the same hardware as for the explicit case can be used. Recall that implicit certificates have an advantage in terms of size (64 byte in the considered case).

Finally, there is a way to improve the computational performance. Consider the case of a signed message with a certificate chain of length 2, i.e.,

[message, end-entity (implicit) certificate, known trusted (explicit) CA cert].

One can combine public key extraction and verification into a single operation, a "triple" multiply and add operation with cost approximately 1.16 * V. So the first verification comes at a cost of approximately 1.16 * V instead of 2 * V. However, combining operations in this way does not output the public key, so all subsequent operations (e.g., verifying subsequent messages signed with the same certificate) also come at a cost of 1.16 * V.

Hardware Support

There are two types of double-multiply-and-add that may be supported by hardware:

CAMP LLC Proprietary. The information contained in this document is considered an interim work product and is subject to revision without notice. The content is provided as is, only for informational purposes with no express or implied warranties that the information is accurate, up-to-date or complete. Any reliance on the content is solely at the user’s own risk. Contact the VSC5 Consortium if you have questions regarding the content.

- Generic double-multiply-and-add, aX + bY
- Double-multiply-and-add where one point is the base, aX + bG. This second type is easier to accelerate because G is known, so various values can be pre-computed.

Verifying a signature only requires the second type of operation. Implicit certificate key extraction needs the first type. More precisely, it needs a subset of the first type, aX + Y. As a consequence, an accelerator for signature verification can only be applied partially for key extraction: it would be used to calculate aX, and Y would have to be added in software.

Adding Y in software would slow things down, but only marginally as a single point add takes less than 1/50 the time for a full multiply. This would add less than one msec to total latency on a 400 MHz processor. However, it is a slowdown compared to explicit certificates.

In conclusion, hardware that supports signature verification may support implicit certificate key extraction with no performance cost (if generic double multiply-and-add is supported), or it may require additional software processing to support implicit certificate key extraction. The software processing is non-zero time, but given that key extraction happens only when a certificate is first seen, if software processing is needed, its impact is very low.

Conclusions

In the following, certificate chains of reasonable length are assumed. Assuming one verifies signatures only occasionally (verify-on-demand), implicit certificates allow for an improvement in terms of size and computational effort, as there is no need to extract the public key from the implicit certificate. If every message is verified, it makes sense to extract the public key from the implicit certificate. In this case, implicit certificates allow only for improvements in terms of size which comes at the cost of one additional double-multiply-and-add operation at the first verify. As extraction of the public key needs to be performed on the first verify only, the first type of double-multiply-and-add does not necessarily have to be implemented in hardware.

CAMP LLC Proprietary. The information contained in this document is considered an interim work product and is subject to revision without notice. The content is provided as is, only for informational purposes with no express or implied warranties that the information is accurate, up-to-date or complete. Any reliance on the content is solely at the user's own risk. Contact the VSC5 Consortium if you have questions regarding the content.