The Need for Being Explicit When Communicating

Anonymous submission to CFail 2021

May 13, 2021

Transition to Quantum-Secure Cryptography Algorithms. With the advent of the quantum computing era comes the requirement to upgrade public-key cryptographic systems to be secure against quantum attackers, i.e., to use post-quantum (PQ) algorithms. Whilst there are significant research and standardization activities taking place, for example NIST's efforts to select PQ key encapsulation mechanisms (KEMs), public-key encryption (PKE) and digital signature schemes (DSS) to be used in the new SP 800-56 and FIPS 186-4 standards, exploration into how these algorithms are compatible with different real world applications is still behind. For instance, many applications require cryptographic algorithms with more advanced functionality than KEMs, PKEs, and DSSs offer. Many of these advanced algorithms have yet to be realized in the PQ setting and evaluated in practice.

One such advanced construction is that of *implicit certificates*. As explained in due course, implicit certificates aim to reduce the large overheads associated with certificate management. To date, implicit certificates have been instantiated over elliptic curves, but no PQ variants have yet been proposed. Within the five main PQ families, lattices look to be the most promising in terms of offering more advanced constructions, as other sophisticated primitives such as identity-based or homomorphic encryption schemes have already been constructed from them.

In this research, we tried–and failed–to construct implicit certificates from the lattice-based DSS Falcon [6] and Dilithium [5] which are finalists in NIST's on-going PQ standardization. Our presentation will explain why our constructions fail and what lessons we learned, hoping to inspire more research on this topic.

Explicit vs Implicit Certificates. For both explicit and implicit certificates, the root cert Cert_C includes the CA's public key pk_C and the CA's signature s_{Cert_C}, i.e., it is self-signed. Every explicit sub-cert Cert_U of a user U consists of the user's public key pk_U, and a signature of U's public key generated by the CA $s_{\text{Cert}_U} \leftarrow \mathsf{Sign}(sk_C, \text{Cert}_U)$. Simplified, if a verifier V wants to verify a signature s_m generated by U on a message m, the verifier first downloads Cert_C and Cert_U. V then computes $\mathsf{Verify}(\text{Cert}_U, s_{\text{Cert}_U}, pk_C)$. If Verify returns "accept", the verifier can then trustingly use pk_U to verify the signature s_m.
In contrast, an implicit sub-cert Cert_U of user U consists only of a so-called reconstruction value R_U but *not* of the user's public key or the CA's signature. In order to verify s_m on m, the verifier V first reconstructs the user's public key pk_U using R_U and pk_C. Therefore, using implicit certs saves space as long as the size of a signature and the size of the user's public key is larger than the size of the reconstruction value, i.e. $|s_{\text{Cert}_U}| + |pk_U| > |R_U|$.

The Use of Implicit Certificates in Vehicle Communication. Implicit certificates are heavily used in the IEEE 1609.2 and 1609.2.1 standards [2, 3] describing secure vehicle-to-vehicle (V2V) communication. More concretely, they use the standardized Elliptic Curve Qu-Vanstone (ECQV) implicit certificate scheme [1]. ECQV seems, as far as we know, the only efficient construction for implicit certificate schemes. We depict the ECQV scheme in fig. 1. The idea of the construction is as follows: the user chooses a kind of ephemeral key and sends it to the CA. The CA chooses a randomness that is added to the user's ephemeral key, becoming the reconstruction value. The randomness, however, is also used as the randomness when signing the certificate. The final user's secret key is the secret key of the ephemeral key added to the signature on the user's certificate. This is, in its core, enabled because essentially all elements in this scheme are integers or points on the elliptic curve and hence, can be easily multiplied or added.

$$
\begin{array}{lcl}
\textbf{User } U & & \textbf{CA } C \\
 & & sk_C = d_C \\
 & & pk_C = Q_C = d_C \cdot G \\
\hline
k_U \xleftarrow{\$} [1, ..., n-1] & & \\
K_U \leftarrow k_U \cdot G & \xrightarrow{K_U, U} & k_C \xleftarrow{\$} [1, ..., n-1] \\
 & & R_U \leftarrow K_U + k_C \cdot G \\
 & & \mathrm{Cert}_U \leftarrow \mathrm{Encode}(R_U, U) \\
 & & e \leftarrow H(\mathrm{Cert}_U) \\
 & \xleftarrow{r, \mathrm{Cert}_U} & r \leftarrow ek_C + d_C \mod n \\
e \leftarrow H(\mathrm{Cert}_U) & & \\
sk_U = d_U \leftarrow ek_U + r \mod n & & \\
pk_U = Q_U \leftarrow eR_U + Q_C & &
\end{array}
$$

$$
\begin{array}{lcl}
\textbf{User } U & & \textbf{CA } C \\
 & & g_C, F_C \leftarrow \mathsf{KeyGen}_{\mathrm{Falcon}} \\
 & & h_C = g_C \cdot F_C^{-1} \\
 & & v_C^{-1} \leftarrow_\sigma \mathcal{R}_q^\ell,\ f_C'^{-1} \leftarrow v_C^{-1} \cdot h_C \\
 & & sk_C = (g_C, F_C) \\
 & & pk_C = (h_C, f_C'^{-1}) \\
\hline
\rho \xleftarrow{\$} \{0,1\}^{256} & & \\
A \in \mathcal{R}_q^{k \times \ell} \leftarrow \mathrm{Expand}(\rho) & & \\
(s_1, s_2, e_U) \xleftarrow{\$} \mathcal{S}_\eta^\ell \times \mathcal{S}_\eta^k \times \mathcal{S}_\eta^\ell & & \\
t \leftarrow As_1 + s_2 \mod q & \xrightarrow{\rho, t, U} & r \xleftarrow{\$} \{0,1\}^{128} \\
 & & \mathrm{Cert}_U \leftarrow \mathrm{Encode}(\rho, t, r, U) \\
 & & c \leftarrow F(r || \mathrm{Cert}_U) \\
 & & (z, Z) \leftarrow F_1((c, 0, ..., 0), g_C \cdot f_C', f_C) \\
(\rho, t, r, U) \leftarrow \mathrm{Cert}_U & \xleftarrow{Z, \mathrm{Cert}_U} & \text{s.t. } z + Z \cdot v_C^{-1} = (c, 0, ..., 0) \cdot v_C^{-1} \\
c \leftarrow F(r || \mathrm{Cert}_U) & & \\
v_C^{-1} \leftarrow h_C^{-1} \cdot f_C'^{-1} & & \\
sk_U = (s_1 + Z \cdot v_C^{-1} + z, e_U) & & \\
pk_U = t + A \cdot c \cdot h_C^{-1} \cdot f_C'^{-1} & &
\end{array}
$$

Figure 1: ECQV implicit cert scheme [1] with G being the group generator (left) in comparison with our failed construction from (simplified) Falcon and Dilithium (right), where $\mathcal{S}_\eta$ is the subset of $\mathcal{R}_q$ with polynomial coefficients $\leq \eta$

Constructing Implicit Certificates. When constructing secure implicit certificate schemes, the following construction principles must be followed intuitively[1]:

- The reconstruction value should be smaller than the CA's signature plus the user's public key.
- Only the user should be able to compute their private key.
- Every party should be able to compute the user's public key from the reconstruction value and the CA's public key.
- The CA's secret mustn't be revealed to any other party; in particular, not to the user.
- The user mustn't be able to create their own certificates; this more or less implies that the CA's signature must be part of the user's private key.

Implicit Certificates from Lattices. It is important to note that in both Falcon and Dilithium, the secret key essentially consists of two or more polynomials with small coefficients and the public key of one or more polynomials with coefficients modulo q. Moreover, the signature consists of a bit string and a polynomial with rather small coefficients, but not as small as the secret polynomial's coefficients as explained below. Furthermore, the key and signature generation of Falcon are more complex than in elliptic curve crypto, involving polynomial arithmetic and sophisticated sampling procedures. In our presentation we will briefly explain the design principles of Falcon and Dilithium. The main recurring difficulties with constructing implicit certificates from Falcon and Dilithium are

- conflicting requirements; some operations require variables to be large or small depending on how they are used, while others require values to be "small enough but not so small that the secret is leaked".
- Dilithium's and Falcon's security are based the learning with errors (LWE) or NTRU problem, respectively. This brings additional constraints in how we can engineer the keys – it is not as simple as adding or subtracting points on an elliptic curve.
- in implicit certificates, both user and CA are required to share secret information between them, but without leaking their entire secret to each other.

[1]To ensure security, in addition, an appropriate security model should be defined, and the construction's security should be reduced from the security or hardness of another algorithms or mathematical problem.

- how to include enough information in the reconstruction value to enable reconstruction of the user's key without it becoming too large or revealing the CA's secret key.

These difficulties lead to many different constructions based on Falcon or Dilithium that failed to achieve all construction principles for implicit certificates. In our presentation, we will present a selection to demonstrate the evolution of our ideas and the reasons why they haven't worked out.

Implicit Certificate Scheme from Falcon and Dilithium. Our most promising construction is based on both schemes, Dilithium and Falcon[2], and is depicted in fig. 1. As pointed out above, the main issue is that the signature size of Falcon (resp. Dilithium) is much larger than the allowed secret key size, and hence, the signature over Cert_U cannot be added to the secret key. Another issue arises from the construction of Falcon signatures: Originally, $(z, Z) \leftarrow F_1((c, 0, ..., 0), g_C, f_C)$ such that $z + Zh_C = (c, 0, ..., 0)$. This means that while (z, Z) consist of rather small polynomials, to reconstruct pk_U using pk_C, we need to somehow include h_C which likely leads to adding a larger element. To overcome these issues we i) combined Falcon and Dilithium because Falcon comes with smaller signature size and Dilithium allows for larger secret key size and ii) introduced another small element v_C, used throughout signature generation, to decrease the size of the element that is then added to the secret key. Unfortunately, to not reveal the CA's secret key and by construction of the ModNTRU problem, we cannot decrease the size of the element to be added to the secret key much further and it is still too large to be added to Dilithium parameters.

Summary. In this presentation, we explain common patterns and reasons why our ideas to construct implicit certs from lattices have failed. As we do not prove that it is impossible, our results do not exclude that implicit certificates can be constructed from lattices or even from Falcon and/or Dilithium. However, our results indicate that it is not straight-forwardly possible without a major breakthrough or more clever idea. This could have detrimental consequences for applications like V2V, as explicit certificates would have to be used instead, bringing large overheads. This is particularly undesirable in safety-critical scenarios. With this presentation we intend to stimulate further research on this topic and invite other researchers to consider more sophisticated constructions. For example, to construct a variant of Falcon's trapdoor sampling that would allow to decrease the signature size further.

References

[1] SEC 4: Elliptic Curve Qu-Vanstone Implicit Certificate Scheme (ECQV) (2013)

[2] IEEE Standard for Wireless Access in Vehicular Environments—Security Services for Applications and Management Messages (2016)

[3] IEEE Standard for Wireless Access in Vehicular Environments (WAVE)–Certificate Management Interfaces for End Entities (2020)

[4] Chuengsatiansup, C., Prest, T., Stehlé, D., Wallet, A., Xagawa, K.: Modfalcon: Compact signatures based on module-ntru lattices. In: Proceedings of the 15th ACM Asia Conference on Computer and Communications Security. pp. 853–866 (2020)

[5] Lyubashevsky, V., Ducas, L., Kiltz, E., Lepoint, T., Schwabe, P., Seiler, G., Stehle, D.: Crystals-dilithium. Tech. rep., National Institute of Standards and Technology (2020), available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions`

[6] Prest, T., Fouque, P.A., Hoffstein, J., Kirchner, P., Lyubashevsky, V., Pornin, T., Ricosset, T., Seiler, G., Whyte, W., Zhang, Z.: Falcon. Tech. rep., National Institute of Standards and Technology (2020), available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions`

[2]Strictly speaking, we use a generalized variant of Falcon, called ModFalcon [4]. In the original Falcon, $n = 1$, and ModFalcon generalizes this to cases where $n \geq 2$. This enables the combination of Falcon and Dilithium with $dn = \ell$, where d is ModFalcon's ring dimension and n corresponds to Dilithium's parameter ℓ. We follow Dilithium's notation in our pseudo code.