Classic McEliece: conservative code-based cryptography: modifications for round 4

23 October 2022

The 4th-round submission includes one tweak to the CCA transform. The previous CCA transform included plaintext confirmation and implicit rejection as dual defenses. The tweaked CCA transform includes implicit rejection without plaintext confirmation. This tweak was already mentioned as a possibility in the Classic McEliece submission.

The purpose of this tweak is to proactively eliminate any concerns regarding U.S. patent 9912479. The threat posed by the patent was already very low, for at least two reasons:

- The patent does not literally apply to any version of Classic McEliece. For example, one of the requirements in the patent is to obtain a ciphertext from "the error vector and the plaintext value"; Classic McEliece has only one of these two objects. The decryption mechanism in the patent is even farther away from what Classic McEliece does.
- All of the overlap between the patent and Classic McEliece is already in the prior art, such as https://eprint.iacr.org/2015/610.

However, users may be concerned about the risk of a court (1) declaring Classic McEliece's encapsulation "equivalent" to the patented encryption mechanism and (2) mishandling the prior art. Eliminating plaintext confirmation creates a much more clear separation between Classic McEliece and the patent, and returns Classic McEliece to the traditional Niederreiter form of encryption.

Here are five further axes for evaluating the effect of removing plaintext confirmation:

- The specification and software are slightly simpler.
- Ciphertexts are 32 bytes smaller. This is not noticeable for most post-quantum KEMs, but it is noticeable for Classic McEliece since Classic McEliece ciphertexts are very short to begin with.
- For proving $O(\epsilon)$-tight ROM IND-CCA2 security and $O(\sqrt{\epsilon})$-tight QROM IND-CCA2 security (with small O constants) assuming OW-CPA security, implicit rejection is sufficient, and removing plaintext confirmation has no effect. These proofs do not eliminate the risk of an IND-CCA2 attack being faster than any QROM IND-CCA2 attack, and it is conceivable that removing plaintext confirmation increases this risk, but Classic McEliece has always addressed this risk by selecting a well-studied, high-security, "unstructured" hash function, namely SHAKE256.
- Removing plaintext confirmation opens up an efficient attack in a beyond-IND-CCA2 security model where a single bit is flipped in a stored private key (for example, from a

naturally occurring DRAM fault). See https://eprint.iacr.org/2022/1125 for an analogous attack against NTRU-HRSS, which removed plaintext confirmation in 2019. On the other hand, implementors can convert IND-CCA2 security into security in this model by encoding stored private keys using an error-correcting code; see below.

- Plaintext confirmation can require extra work to protect against side-channel attacks (and combined side-channel/fault attacks), but can also help protect the rest of the decapsulation process against attack. The overall cost of protection, with or without plaintext confirmation, is a complicated function of the set of possible attacks, which in turn depend on the usage scenario.

Most proposals for post-quantum IND-CCA2 KEMs do not include plaintext confirmation. The usual analysis treats implicit-rejection proofs as enough justification to skip plaintext confirmation, even when the proofs have much larger tightness losses than the proofs applicable to Classic McEliece.

Regarding error correction for private keys: Commonly available SECDED ECC DRAM hardware stores 64 logical bits as 72 physical bits using a distance-4 extended Hamming code. ECC here means "error-correcting code"; SECDED means "single error correction, double error detection". Implementors can also use the new `libsecded` software library from https://pqsrc.cr.yp.to/downloads.html.

Note that it would be possible to integrate SECDED into the specified private-key format. Classic McEliece instead follows the principle that any generic transformation aiming at a goal beyond IND-CCA2 is out of scope for a KEM specification. Factoring the transformation out of KEM specifications simplifies the cryptographic ecosystem, making design and review easier, because the transformation is modularized instead of being handled redundantly by each cryptosystem. Each component is simpler, without any change in the composition provided to the end user.

This systems-engineering principle covers a point from Shoup already quoted in the submission, namely that KEMs should avoid hashing "labels" such as identities since "it is easier to implement labels in the data encapsulation mechanism". The same principle also implies that KEMs should avoid specifying SECDED.