Some implementation considerations

Ganyu (Bruce) Xu (g66xu)

June, 2024

1 June 12, 2024 considerations

Since ML-KEM as defined in FIPS 203 uses the SHA3 family of hash functions and XOFs, we will continue using SHA3 instead of introducing new families of hash functions so the code size can be reduced (and so the codebase can be more manageable).

Using HMAC with SHA3 does not make much sense from an efficiency perspective, especially given that KMAC exists and is approved by NIST[2]. We can use the SHA2 family in `HMAC-SHA256` for 128-bit classical security and `HMAC-SHA384` for 256-bit classical security to modify the *90s-variant* of Kyber, but I doubt how meaningful this modification can be, given that NIST gave up on it anyways.

NIST SP.800.185[2] specified two approved variations of KMAC. `KMAC-128` uses `Shake128` as the underlying XOF, and `KMAC-256` uses `Shake256` as the underlying XOF. `Shake128` and `Shake256` respectively claim 128-bit and 256-bit (classical) collision resistance, though **citation is needed**. I also don't know about **quantum attacks on Shake**. If time permits I will look into security under quantum attacks.

So here are the updated parameter recommendations

Security level	`MAC`	Key size	Tag size	Preimage resistance	Collision resistance
`Kyber512`	`KMAC-128`	256	256	256 (c), 128 (q)	128 (c), 85.3 (q)
`Kyber768`	`KMAC-256`	256	384	384 (c), 192 (q)	192 (c), 128 (q)
`Kyber1024`	`KMAC-256`	256	512	512 (c), 256(q)	256 (c), 170.6 (q)

Table 1: MAC recommendations

2 June 11, 2024 consdierations

Here are some considerations when implementing Kyber using "encrypt-then-mac"

1. Strong candidates for `MAC` are HMAC and KMAC. From an efficiency standpoint KMAC is preferable because $\texttt{HMAC}(k, m) = H(k \| H(k \| m))$ makes two calls to the hash function, but `KMAC` makes only one call to `Shake`. From a security and usability standpoint, HMAC is more widely used than KMAC. We can simply try both and report the results.

2. Since the `MAC` key is derived from the message, and in Kyber, the message $m \in \mathcal{M} = \{0,1\}^{256}$ is exactly 256-bit in length, we should choose the `MAC` key to also be exactly 256-bit in length. Any longer key is unnecessary, since it then because easier to simply search for the message.

3. There is argument about how HMAC is not affected by weakness in collision resistance in the underlying hash function[3] (the idea is that an adversary can only obtain a tag online, which is much slower than computing hash offline). In addition, collision resistance is less affected by quantum computing than preimage resistance is. The best classical attack on collision resistance is the birthday attack, with the expected number of steps being $2^{\frac{n}{2}}$ for an n-bit hash; the best quantum attack[1] is expected to take $2^{\frac{n}{3}}$ steps for an n-bit hash.

	key size	tag size	pre-image resistance	collision resistance
classical, 128-bit security	256	256	256	128
classical, 256-bit security	256	384	384	192
quantum, optimistic	256	256	128	≈ 86
quantum, moderate	256	384	192	128
quantum, conservative	256	512	256	≈ 171

Table 2: Parameter sizes and estimated security levels, all in bits

With the three points above, I came up with this table for the recommended parameters. For classical 128-bit security and 256-bit security, I chose a 256-bit and 384-bit hash for the tag because TLS 1.2 pairs SHA-256 with AES-128 and SHA-384 with AES-256. For quantum computing, we can choose more conservative parameters (such as using 384-bit hash for 128-bit security and 512-bit hash for 256-bit security).

References

[1] Gilles Brassard, Peter Hoyer, and Alain Tapp. Quantum algorithm for the collision problem. *arXiv preprint quant-ph/9705002*, 1997.

[2] John Kelsey, Shu-jen Chang, and Ray Perlner. Sha-3 derived functions: cshake, kmac, tuplehash, and parallelhash. *NIST special publication*, 800:185, 2016.

[3] Hugo Krawczyk, Mihir Bellare, and Ran Canetti. Hmac: Keyed-hashing for message authentication. Technical report, 1997.