# Some implementation considerations

Ganyu (Bruce) Xu (g66xu)

June 18, 2024

## 1 One-time MAC

The challenge encryption $c^*$ in the `OW-PCVA` game is computed using a uniformly randomly sampled plaintext message $m^* \xleftarrow{\$} \mathcal{M}_{\texttt{PKE}}$, and a `MAC` key $k_{\texttt{MAC}}$ derived from said plaintext $m^*$. Assuming that $m^*$ is not already known (otherwise the `OW-CPA` security of the underlying `PKE` is already broken), then $k^*_{\texttt{MAC}}$ is also not known, so the `OW-PCVA` adversary has no way to evaluate the `MAC` under $k^*_{\texttt{MAC}}$ offline.

Furthermore, I propose that the `OW-PCVA` adversary also has no way of issuing signing queries for $k^*_{\texttt{MAC}}$. The adversary can run `E` or `Encap` offline, but the tags generated from `E` or `Encap` will be associated with a chosen message or a random message, which has $\frac{1}{|\mathcal{M}_{\texttt{PKE}}|}$ probability of being $m^*$. In other words, for the challenge `MAC` key, the adversary will only see one valid message-tag pair, which is the challenge message $(c^*, t^*)$.

## 2 Choice of MAC

When thinking about choosing a MAC there are two parameters that we need to care about: the key size and the tag size.

Whether working with Round 3 Kyber [1] or ML-KEM [3], the message is always 256 bits in size, and since the MAC key is derived from the message, the MAC key should also always be 256 bits. A 256-bit MAC key is great since both GCM and Poly1305 accepts 256-bit MAC keys.

Reasoning about the security implication of the tag size is more nuanced.

### 2.1 Generic collision resistance

We can model a MAC has a keyed hash function, which means that for a fixed MAC key, we can argue about the seurity of a MAC using the standard security notions for hash function: pre-image resistance, 2nd pre-image resistance, and collision resistance.

However, there is an important distinction between the security of a MAC as a keyed hash function and the security of a hash function. With MAC as a keyed hash function, assuming that the adversary doesn't already have a way to evaluating this function offline (otherwise forgery is trivial), **it has to query the signing oracle to evaluate the hash function**. Given an n-bit tag, a generic birthday attack takes $2^{\frac{n}{2}}$ evaluations to find a collision, and the BHT algorithm [2] takes $2^{\frac{n}{3}}$ evaluations to find a collision, but neither attack is relevant, because "evaluation" means that the message-tag pair is queried from the signing oracle and **does not count as forgery**.

This means that we only need to care about preimage resistance for our MAC (as a keyed hash function),

### 2.2 Offline versus online attacks

There should be a distinction between attacks that involve interaction with another party (online) versus attacks that can be done without such interaction (offline). When we measure the security of a scheme by the number of computational steps needed to carry out an attack, we specifically mean an offline attack. **An online attack with substantial number of interactions can be trivially thwarted** because the other party can implement rate limiting with exponential backoff, so after only a few (think fewer than 10 password

tries) invalid interactions, the other party will effectively block the adversary from further interactions, thus stopping the attack in the track.

**Adversary obtains** $(c, t)$, tampers with $c$ to obtain $c'$ which still correspond with the same plaintext $m$, which means that $t'$ should be created under the same key $(c, t)$ is created.

# References


[1] Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals-kyber algorithm specifications and supporting documentation. *NIST PQC Round*, 2(4):1–43, 2019.

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

[3] NIST Module-Lattice-Based Key-Encapsulation. Mechanism standard. *NIST Post-Quantum Cryptography Standardization Process; NIST: Gaithersburg, MD, USA*, 2023.