1 Encrypt-then-MAC transformations

Let $\mathtt{PKE}(\mathtt{KeyGen}, \mathtt{E}, \mathtt{D})$ be a probabilistic public-key encryption scheme defined over message space $\mathcal{M}_{\mathtt{PKE}}$, ciphertext space $\mathcal{C}_{\mathtt{PKE}}$, and coin space $\mathcal{R}_{\mathtt{PKE}}$. Where the encryption routine is deterministic, we simply set the coin space to contain a single element $\mathcal{R} = \{r\}$. Let $\mathtt{MAC}(\mathtt{Sign}, \mathtt{Verify})$ be a message authentication code defined over key space $\mathcal{K}_{\mathtt{MAC}}$. The message space of the $\mathtt{MAC}$ should contain the ciphertext space of the $\mathtt{PKE}$: $\mathcal{C}_{\mathtt{PKE}} \subseteq \mathcal{M}_{\mathtt{MAC}}$. Let $G : \mathcal{M}_{\mathtt{PKE}} \to \mathcal{R}_{\mathtt{PKE}}$ and $H : \mathcal{M}_{\mathtt{PKE}} \to \mathcal{K}_{\mathtt{MAC}}$ be hash functions.

The "encrypt-then-MAC" transformation $\mathtt{PKE}_{\mathtt{EtM}}(\mathtt{KeyGen}, \mathtt{E}_{\mathtt{EtM}}, \mathtt{D}_{\mathtt{EtM}}) = T_{\mathtt{EtM}}(\mathtt{PKE}, \mathtt{MAC}, H)$ outputs a public-key encryption scheme where the key generation routine is identical to the input $\mathtt{PKE}$'s key generation routine. The de-randomized "encrypt-then-MAC" transformation $\mathtt{PKE}^{\not\$}_{\mathtt{EtM}}(\mathtt{KeyGen}, \mathtt{E}^{\not\$}_{\mathtt{EtM}}, \mathtt{D}^{\not\$}_{\mathtt{EtM}}) = T^{\not\$}_{\mathtt{EtM}}(\mathtt{PKE}, \mathtt{MAC}, G, H)$ similarly outputs a public-key encryption scheme. In both transformations, the key generation routine remains unchanged. The modified encryption and decryption routines are described in figure 1 and 2.

```
Algorithm 1 $\mathtt{E}_{\mathtt{EtM}}(\mathtt{pk}, m)$
1: $r \xleftarrow{\$} \mathcal{R}_{\mathtt{PKE}}$          ▷ If $\mathtt{E}$ is randomized, then $\mathtt{E}_{\mathtt{EtM}}$ is randomized
2: $k_{\mathtt{MAC}} \leftarrow H(m)$
3: $c \leftarrow \mathtt{E}(\mathtt{pk}, m; r)$
4: $t \leftarrow \mathtt{Sign}(k_{\mathtt{MAC}}, c)$
5: return $(c, t)$
```

```
Algorithm 2 $\mathtt{D}_{\mathtt{EtM}}(\mathtt{sk}, (c, t))$
1: $\hat{m} \leftarrow \mathtt{D}(\mathtt{sk}, c)$
2: $\hat{k}_{\mathtt{MAC}} \leftarrow G(\hat{m})$
3: if $\mathtt{Verify}(\hat{k}_{\mathtt{MAC}}, c, t) \neq 1$ then
4:     return $\perp$
5: end if
6: return $\hat{m}$
```

Figure 1: "encrypt-then-MAC" transformation

```
Algorithm 3 $\mathtt{E}^{\not\$}_{\mathtt{EtM}}(\mathtt{pk}, m)$
1: $k_{\mathtt{MAC}} \leftarrow H(m)$
2: $r \leftarrow G(m)$
3: $c \leftarrow \mathtt{E}(\mathtt{pk}, m; r)$
4: $t \leftarrow \mathtt{Sign}(k_{\mathtt{MAC}}, c)$
5: return $(c, t)$
```

```
Algorithm 4 $\mathtt{D}^{\not\$}_{\mathtt{EtM}}(\mathtt{sk}, (c, t))$
1: $\hat{m} \leftarrow \mathtt{D}(\mathtt{sk}, c)$
2: $\hat{k}_{\mathtt{MAC}} \leftarrow G(\hat{m})$
3: if $\mathtt{Verify}(\hat{k}_{\mathtt{MAC}}, c, t) \neq 1$ then
4:     return $\perp$
5: end if
6: return $\hat{m}$
```

Figure 2: de-randomized "encrypt-then-MAC" transformation

Intuition the design goal of this transformation is to achieve plaintext awareness (PA). According to [BDPR98], PA implies resistance to adaptive chosen ciphertext attacks. For an adversary to produce a valid authenticated ciphertext (c, t), it must be able to produce a valid tag t for the unauthenticated ciphertext c under the correct MAC key $k_{\mathtt{MAC}} = H(\mathtt{D}(\mathtt{sk}, c))$, which means that the adversary either knows $k_{\mathtt{MAC}}$ or performs a forgery against the MAC. Since $k_{\mathtt{MAC}} \leftarrow H(m)$ is derived from the plaintext m, under the random oracle model, I argue that knowing $k_{\mathtt{MAC}}$ implies knowing m. In other words, if the adversary cannot perform forgery against the MAC nor break the one-way security of the encryption routine, then it cannot produce valid ciphertext for which it does not not the plaintext, thus rendering the advantage of a decryption oracle (or any ciphertext-processing oracles, such PCO or CVO) negligible.

Let A be an one-way adversary against the transformed scheme with access to a decryption oracle. I want to show that under the random oracle model, **we can construct a second adversary B that uses the public key $\mathtt{pk}$ and the tapes of the hash oracles $\mathcal{O}^G, \mathcal{O}^H$ to simulate a decryption oracle for** A such that A can only distinguish the true decryption oracle from the simulated decryption oracle with negligible advantage. Note that in [HHK17], the first transformation asks for OW-PCVA security, but since the true PCO and CVO are both implemented using the decryption routine, it is easy to build simulated PCO and CVO using the simulated decryption oracle.

Unfortunately, constructing a simulated decryption oracle is annoyingly elusive. Here are the things I have tried so far.

1.1 Identify plaintext using the coin and the key

With derandomized encrypt-then-MAC, if an authenticated ciphertext (c, t) is honestly generated, then there should be a pair of matching hash queries $(\tilde{m}, \tilde{r}) \in \mathcal{O}^G$ and $(\tilde{m}, \tilde{k}) \in \mathcal{O}^H$ such that $\texttt{E}(\texttt{pk}, \tilde{m}; r) = c$ and $\texttt{Verify}(\tilde{k}, c, t) = 1$. See the algorithm below:

Algorithm 5 $\mathcal{O}^{\texttt{D}}_{\texttt{sim}}(c, t)$ attempt 1

```
1: if $\exists(\tilde{m}, \tilde{r}) \in \mathcal{O}^G, (\tilde{m}, \tilde{k}) \in \mathcal{O}^H : \texttt{E}(\texttt{pk}, \tilde{m}; \tilde{r}) = c \land \texttt{Verify}(\tilde{k}, c, t) = 1$ then
2:     return $\tilde{m}$
3: end if
4: return $\perp$
```

Figure 3: Simulate decryption oracle using coin and MAC key

This simulated decryption oracle can be efficiently distinguished from the true decryption oracle if the ciphertext can be predictably perturbed while still decrpyting to the same plaintext. This ciphertext perturbation is possible in most LWE schemes: one can add a small amount of noise to the unauthenticated ciphertext, and the resulting "new ciphertext" will still decrypt to the same plaintext. To distinguish simulation from true decryption oracle:

1. Generate an honest plaintext-ciphertext pair $(m, (c, t))$. In this process, store the coin $r \leftarrow G(m)$ and MAC key $k_{\texttt{MAC}} \leftarrow H(m)$
2. Perturb the unauthenticated ciphertext. Denote the result by c'.
3. Compute $t' = \texttt{Sign}(k_{\texttt{MAC}}, c')$
4. Submit (c', t') as a decryption query. The true decryption oracle will return m (which is correct), but the simulation will reject this ciphertext (which is too strict).

A similar attempt to simulate decryption oracle can be made on the (possibly randomized) "encrypt-then-MAC" scheme, as well. We need to replace the hash oracle $\mathcal{O}^G$ with a "randomness oracle" $\mathcal{O}^{\mathcal{R}}$: when adversary A wants to sample $r \overset{\$}{\leftarrow} \mathcal{R}$, A needs to query this randomness oracle, which will return a random value at this query, and store the query output on a tape:

Algorithm 6 $\mathcal{O}^{\texttt{D}}_{\texttt{sim}}(c, t)$ attempt 1.1

```
1: if $\exists\tilde{r} \in \mathcal{O}^{\mathcal{R}}, (\tilde{m}, \tilde{k}) \in \mathcal{O}^H : \texttt{E}(\texttt{pk}, \tilde{m}; \tilde{r}) = c \land \texttt{Verify}(\tilde{k}, c, t) = 1$ then
2:     return $\tilde{m}$
3: end if
4: return $\perp$
```

Figure 4: Simulate decryption oracle in randomized `EtM`

1.2 Identify plaintext using only the key

My second attempt at recovering the plaintext from the hash oracles uses only the MAC key:

Algorithm 7 $\mathcal{O}^{\mathtt{D}}_{\mathtt{sim}}(c, t)$ attempt 2

```
1: if $\exists(\tilde{m}, \tilde{k}) \in \mathcal{O}^H : \mathtt{Verify}(\tilde{k}, c, t) = 1$ then
2:     return $\tilde{m}$
3: end if
4: return $\perp$
```

This allows the adversary to sign unauthenticated ciphertext produced from perturbing honestly generated unauthenticated ciphertext, but then creates the problem that the adversary can now sign unauthenticated ciphertext with MAC keys derived from arbitrary plaintext:

1. Sample a random plaintext and run the encryption routine. This creates an honest plaintext-ciphertext pair $(m, (c, t))$
2. Sample a second, uncorrelated plaintext m' and derive the corresponding MAC key $k'_{\mathtt{MAC}} = H(m')$
3. Sign c with $k'_{\mathtt{MAC}}$: $t' \leftarrow \mathtt{Sign}(k'_{\mathtt{MAC}}, c)$
4. Submit (c, t') for a decryption query. The true decryption oracle will reject this query, but the simulation will return m', which is just wrong.

1.3 Identify matching plaintext-ciphertext using only the public key

Let $\mathtt{match}(\mathtt{pk}, m, c)$ be an efficient algorithm that takes the public key $\mathtt{pk}$ and a plaintext-ciphertext pair (m, c) as input. $\mathtt{match}$ returns 1 if m is the decryption of c under the keypair $(\mathtt{pk}, \mathtt{sk})$ and 0 otherwise.

If such an algorithm exists, then it is easy to implement the simulated decryption oracle.

Algorithm 8 $\mathcal{O}^{\mathtt{D}}_{\mathtt{sim}}(c, t)$ if $\mathtt{match}$ exists

```
1: if $\exists(\tilde{m}, \tilde{k}) \in \mathcal{O}^H : \mathtt{match}(\tilde{m}, c) = 1 \land \mathtt{Verify}(\tilde{k}, c, t)$ then
2:     return $\tilde{m}$
3: end if
4: return $\perp$
```

On the other hand, if such an algorithm exists, then the encryption scheme is necessarily not semantically secure. Since Kyber begins with a semantically secure encryption scheme before transforming into a key encapsulation mechanism, such an algorithm must not exist, which means that we need to build the capabilities of "identifying matching plaintext-ciphertext pair" into the $\mathtt{EtM}$ transformation(s).

1.4 Questioning the premise

It is of course possible the reason why I cannot come up with a security reduction is that $\mathtt{EtM}$ transformation is insecure, but so far I also could not come up with any meaningful attack that breaks the one-way security of the transformed schemes.

Note that the two distinguishing algorithms in the previous sections only reveal flaws in the simulated decryption oracle. They do not lead to meaningful weakness (or meaningful advantages) of the $\mathtt{EtM}$ transformation under chosen ciphertext attacks

References

[BDPR98] Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. Relations among notions of security for public-key encryption schemes. In *Advances in Cryptology—CRYPTO'98: 18th*

Annual International Cryptology Conference Santa Barbara, California, USA August 23–27, 1998 Proceedings 18, pages 26–45. Springer, 1998.

[HHK17] Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. A modular analysis of the fujisaki-okamoto transformation. In *Theory of Cryptography Conference*, pages 341–371. Springer, 2017.