Randomized encrypt-then-MAC

Ganyu (Bruce) Xu (g66xu)

June 24, 2024

1 Encrypt-then-MAC transformation

In this section, we introduce two "encrypt-then-MAC" ($\texttt{EtM}$ for short) transformations. For a given input public-key encryption scheme $\texttt{PKE} = (\texttt{KeyGen}, \texttt{E}, \texttt{D})$, the de-randomized $\texttt{EtM}$ transformation returns a a public-key encryption scheme $\texttt{PKE}^{\$}_{\texttt{EtM}} = (\texttt{KeyGen}, \texttt{E}^{\$}_{\texttt{EtM}}, \texttt{D}^{\$}_{\texttt{EtM}})$ such that $\texttt{E}^{\$}_{\texttt{EtM}}$ is deterministic. On the other hand, the randomized $\texttt{EtM}$ transformation returns a public-key encryption scheme $\texttt{PKE}_{\texttt{EtM}} = (\texttt{KeyGen}, \texttt{E}_{\texttt{EtM}}, \texttt{D}_{\texttt{EtM}})$ whose encryption routine preserves the randomization of the input encryption routine.

The encryption routine of both transformations uses the plaintext $m \in \mathcal{M}_{\texttt{PKE}}$ to derive a MAC key $k_{\texttt{MAC}}$, then after using the input encryption routine $\texttt{E}$ to produce an unauthenticated ciphertext $c \in \mathcal{C}_{\texttt{PKE}}$, the MAC key is used to compute a tag t on the unauthenticated ciphertext. The tuple (c, t) is the authenticated ciphertext that the transformed encryption routine outputs. The decryption routine of both transformations first decrypt the unauthenticated ciphertext to recover the plaintext, then uses the decryption to derive a MAC key, which is used to verify the tag.

Because the tag is computed on the unauthenticated ciphertext, if the unauthenticated ciphertext is tempered with, then the tag must be recomputed. Because of the one-way security of the underlying encryption scheme, it is safe to assume that no adversary can use the unauthenticated ciphertext to recover the message, which means that producing valid tags for new message constitutes a forgery attack. Therefore, mounting any meaningful chosen ciphertext attacks is at least as hard as producing a forgery against the underlying $\texttt{MAC}$ scheme.

The transformed routines $(\texttt{E}_{\texttt{EtM}}, \texttt{D}_{\texttt{EtM}})$ and $(\texttt{E}^{\$}_{\texttt{EtM}}, \texttt{D}^{\$}_{\texttt{EtM}})$ are described in figure 1 and 2 respectively.

Algorithm 1 $\texttt{E}_{\texttt{EtM}}(\texttt{pk}, m)$

```
1: $k_{\texttt{MAC}} \leftarrow G(m)$
2: $c \leftarrow \texttt{E}(\texttt{pk}, m)$    ▷ If $\texttt{E}$ is randomized, then $\texttt{E}_{\texttt{EtM}}$ is randomized
3: $t \leftarrow \texttt{MAC}(k_{\texttt{MAC}}, c)$
4: return $(c, t)$
```

Algorithm 2 $\texttt{D}_{\texttt{EtM}}(\texttt{sk}, (c, t))$

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

Figure 1: $\texttt{EtM}$ transformation.

Algorithm 3 $\mathtt{E}^{\$}_{\mathtt{EtM}}(\mathtt{pk}, m)$

```
1: $k_{\mathtt{MAC}} \leftarrow G(m)$
2: $r \leftarrow H(m)$
3: $c \leftarrow \mathtt{E}(\mathtt{pk}, m; r)$
4: $t \leftarrow \mathtt{MAC}(k_{\mathtt{MAC}}, c)$
5: return $(c, t)$
```

Algorithm 4 $\mathtt{D}^{\$}_{\mathtt{EtM}}(\mathtt{sk}, (c, t))$

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

Figure 2: Derandomized encrypt-then-MAC