Randomized encrypt-then-MAC

Ganyu (Bruce) Xu (g66xu)

June 21, 2024

1 Introduction

In the modular FO transform[1], the authors constructed IND-CCA KEM using two steps. The first step is adding ciphertext integrity to a one-way secure encryption scheme (the `T` transformation). The second step is constructing the IND-CCA KEM out of the OW-PCVA secure encryption scheme (the `U` transformation). The `T` transformation uses two techniques for achieving ciphertext integrity, which the authors called "*de-randomization*" and "*re-encryption*".

Our current proposal is to replace re-encryption with a message authentication code (MAC), but we maintained the deterministic encryption routine. For clarity we will call this "*de-randomized encrypt-then-MAC*". The algorithm is specified in figure 1.

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

$(r, k_{\mathtt{MAC}}) \leftarrow G(m)$
$c \leftarrow \mathtt{E}(\mathtt{pk}, m; r)$
$t \leftarrow \mathtt{MAC}(k_{\mathtt{MAC}}, c)$
return (c, t)

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

$\hat{m} \leftarrow \mathtt{D}(\mathtt{sk}, c)$
$(\hat{r}, \hat{k}_{\mathtt{MAC}}) \leftarrow G(\hat{m})$
if $\mathtt{MAC.Verify}(\hat{k}_{\mathtt{MAC}}, c, t) \neq 1$ **then**
    return $\perp$
end if
return $\hat{m}$

Figure 1: Derandomized encrypt-then-MAC

For the past week I have been thinking about removing the de-randomization, as well. This means that we no longer derive the pseudorandom coin from the message. If the input `PKE` has a randomized encryption routine, then the transformed encrypt-then-mac scheme will also be randomized. We call this "*randomized encrypt-then-MAC*". The algorithm is specified in figure 2

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

$k_{\mathtt{MAC}} \leftarrow G(m)$
$c \leftarrow \mathtt{E}(\mathtt{pk}, m)$
$t \leftarrow \mathtt{MAC}(k_{\mathtt{MAC}}, c)$
return (c, t)

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

$\hat{m} \leftarrow \mathtt{D}(\mathtt{sk}, c)$
$\hat{k}_{\mathtt{MAC}} \leftarrow G(\hat{m})$
if $\mathtt{MAC.Verify}(\hat{k}_{\mathtt{MAC}}, c, t) \neq 1$ **then**
    return $\perp$
end if
return $\hat{m}$

Figure 2: Randomized encrypt-then-MAC

2 Security of MAC

Intuitively, if the unauthenticated ciphertext is tempered with, then the tag should change unpredictably. Since the MAC key is derived from the plaintext message, and the plaintext message is hidden behind the OW-CPA security of the input scheme, it is safe to assume that the adversary does not have information on the MAC key. If the adversary does not have the MAC key, then it will not be able to produce the correct tag. Therefore, the adversary cannot produce valid authenticated ciphertexts without knowing the corresponding plaintext in the first place.

However, formalizing this idea into a security proof is surprisingly difficult. The most popular methods of formal security proof is

References

[1] 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.