Sudipta Halder (2021202011)

Use CPA security and secure MAC to design a provably CCA-secure encryption scheme

Let's first go through the construction mechanism of CCA secure encryption scheme using CPA security and MAC. I have used OFB mode in CPA security and CBC for generating MAC.

CONSTRUCTION 4.17 CCA-secure encryption.

Define a CCA-secure encryption scheme as follows:

- $\mathsf{Gen}'(1^n)$: upon input 1^n, choose $k_1, k_2 \leftarrow \{0,1\}^n$
- $\mathsf{Enc}'_k(m)$: upon input key (k_1, k_2) and plaintext message m, compute $c = \mathsf{Enc}_{k_1}(m)$ and $t = \mathsf{Mac}_{k_2}(c)$ and output the pair (c, t)
- $\mathsf{Dec}'_k(c, t)$: upon input key (k_1, k_2) and ciphertext (c, t), first verify that $\mathsf{Vrfy}_{k_2}(c, t) = 1$. If yes, then output $\mathsf{Dec}_{k_1}(c)$; if no, then output $\perp$.

[1]

So, basically, we need to choose two keys k1 and k2 of length any length n in binary. Then we have to encrypt the data using the k1 and the encryption scheme will be CPA (OFB mode).

After that, we have to create a MAC tag of the encrypted text using CBC-MAC and key k2.

We would then send the cipher-text and MAC tag to the receiver side along with necessary things (data for verification, keys k1, k2, initial vector for CPA etc.) for verification of data, ciphertext and MAC.

Receiver side will first take the ciphertext and will regenerate the MAC-tag using the key k2. If the regenerated MAC-tag does not match with the received tag then any one of cipher-text or MAC is tampered. If the regenerated MAC matches with the received MAC TAG he would procced further. He would decrypt the ciphertext with the key k1 and verify whether the decrypted data is same with the original data or not. If it does not match then data was tampered. If it matches then everything is perfect.

Now, let's look at the proof of the above construction why it is a CCA-secure encryption scheme.

THEOREM 4.18 *Assume that* $\Pi_E = (\mathsf{Gen}_E, \mathsf{Enc}, \mathsf{Dec})$ *is a CPA-secure encryption scheme and that* $\Pi_M = (\mathsf{Gen}_M, \mathsf{Mac}, \mathsf{Vrfy})$ *is a secure message authentication code with unique tags. Then, Construction 4.17 is a CCA-secure encryption scheme.*

PROOF The idea behind the proof of this theorem is as follows. Since $(\mathsf{Gen}_M, \mathsf{Mac}, \mathsf{Vrfy})$ is a secure message authentication code, we can assume that all queries to the decryption oracle are *invalid*, unless the queried ciphertext was previously obtained by the adversary from its encryption oracle. Therefore, the security of the scheme $\Pi' = (\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$ in Construction 4.17 is reduced to the CPA-security of $(\mathsf{Gen}_E, \mathsf{Enc}, \mathsf{Dec})$ (because the decryption oracle is effectively useless). In more detail, we first prove that except with negligible probability, the only valid queries made by the adversary to the decryption oracle are ciphertexts that were previously obtained from the encryption oracle. Then, given this claim, we prove that if the CCA-secure scheme is not secure, then neither is the underlying CPA-scheme $(\mathsf{Gen}_E, \mathsf{Enc}, \mathsf{Dec})$. This is due to the fact that an adversary for the CPA-secure scheme can actually simulate a decryption oracle for the CCA adversary. This simulation works by returning $\perp$ if the received ciphertext was never queried before, and returning the appropriate message if the ciphertext was generated by querying the encryption oracle. The validity of this simulation follows from the above claim. We now proceed to the formal proof.

Let $\mathcal{A}$ be any probabilistic polynomial-time CCA adversary attacking Construction 4.17. Define $\text{VALID-QUERY}_{\mathcal{A},\Pi'}(n)$ to be the event that in the experiment $\mathsf{PrivK}^{\mathsf{cca}}_{\mathcal{A},\Pi'}(n)$, the adversary $\mathcal{A}$ generates a query (c,t) to the decryption oracle that was not obtained from the encryption oracle and does not result in an oracle reply $\perp$. We claim that $\Pr[\text{VALID-QUERY}_{\mathcal{A},\Pi'}(n)]$ is at most negligible. Intuitively, this is due to the fact that if the oracle does not reply $\perp$, then t is a valid MAC tag for c. Thus, if (c,t) was not obtained by querying the encryption oracle, this means that $\mathcal{A}$ must have forged a MAC. Formally, we prove that if the probability that VALID-QUERY occurs is non-negligible, then we can construct an adversary $\mathcal{A}_{\text{mac}}$ that breaks the MAC as follows. Let $q(\cdot)$ be a polynomial that upper-bounds the running-time of $\mathcal{A}$ (and thus the number of oracle queries it makes). Then, adversary $\mathcal{A}_{\text{mac}}$, interacting in $\mathsf{Mac\text{-}forge}_{\mathcal{A}_{\text{mac}},\Pi_M}(n)$, chooses a random key k_1 for Enc and a random value

$i \leftarrow \{1, \ldots, q(n)\}$, and invokes the CCA-adversary $\mathcal{A}$. Adversary $\mathcal{A}_{\text{mac}}$ then simulates the encryption and decryption oracles for $\mathcal{A}$. The way it does this is to use k_1 and its MAC-generating oracle to simulate the encryption oracle for $\mathcal{A}$. Regarding the decryption oracle, all but the i^{th} query will be assumed to be invalid, and $\mathcal{A}_{\text{mac}}$ will "hope" that the i^{th} query is valid. In this case, $\mathcal{A}_{\text{mac}}$ will hope to have obtained a forged tag. More specifically, when $\mathcal{A}$ queries the encryption oracle with m, adversary $\mathcal{A}_{\text{mac}}$ computes $c = \mathsf{Enc}_{k_1}(m)$ and requests a tag t for c. Adversary $\mathcal{A}_{\text{mac}}$ then returns the pair (c, t) to $\mathcal{A}$ as its oracle reply. In contrast, in every decryption oracle query (c, t) from $\mathcal{A}$ *apart from the* i^{th} *one*, adversary $\mathcal{A}_{\text{mac}}$ first checks if (c, t) was ever generated from an encryption query. If yes, $\mathcal{A}_{\text{mac}}$ returns the plaintext m that was queried by $\mathcal{A}$ when (c, t) was generated. If not, $\mathcal{A}_{\text{mac}}$ returns $\perp$. In contrast, for the i^{th} decryption oracle query (c, t), adversary $\mathcal{A}_{\text{mac}}$ outputs (c, t) as its MAC forgery and halts. (We remark that the generation of the challenge ciphertext from the pair (m_0, m_1) is also carried out by $\mathcal{A}_{\text{mac}}$ for $\mathcal{A}$ as in the CCA experiment.)

Clearly $\mathcal{A}_{\text{mac}}$ runs in probabilistic polynomial-time. We now analyze the probability that $\mathcal{A}_{\text{mac}}$ generates a good forgery, and so succeeds in Mac-forge. By our contradicting assumption, with non-negligible probability, adversary $\mathcal{A}$ generates a query (c, t) to the decryption oracle that was not obtained from the encryption oracle, and does not return $\perp$. We remark that since $(\mathsf{Gen}_M, \mathsf{Mac}, \mathsf{Vrfy})$ has unique tags, it follows that the query c was never asked by $\mathcal{A}_{\text{mac}}$ to its MAC-tag oracle (because (c, t) was not obtained from an encryption query and there is only a single possible t that is a valid MAC tag for c). Therefore, such a pair (c, t) is a "good forgery" for $\mathcal{A}_{\text{mac}}$. Now, if all the decryption oracle queries generated by $\mathcal{A}$ up until the i^{th} one were indeed invalid, then the simulation by $\mathcal{A}_{\text{mac}}$ for $\mathcal{A}$ up until the i^{th} query is perfect. Furthermore, the probability that the i^{th} query is the *first* valid one generated by $\mathcal{A}$ is at least $1/q(n)$ because $\mathcal{A}$ makes at most $q(n)$ oracle queries, and one of these is the first valid one. Therefore, the probability that $\mathcal{A}_{\text{mac}}$ succeeds in Mac-forge is at least $1/q(n)$ times the probability that the VALID-QUERY event occurs. Since $\mathcal{A}_{\text{mac}}$ can succeed in Mac-forge with at most negligible probability, it follows that VALID-QUERY occurs with at most negligible probability. That is, we have that for some negligible function negl,

$$\Pr\left[\text{VALID-QUERY}_{\mathcal{A},\Pi'}(n)\right] < \mathsf{negl}(n).$$

Given that VALID-QUERY occurs with at most negligible probability, we now show that Construction 4.17 is CCA-secure. In this part of the proof, we reduce the security to the CPA-security of $(\mathsf{Gen}_E, \mathsf{Enc}, \mathsf{Dec})$. Specifically, let $\mathcal{A}$ be any probabilistic polynomial-time adversary for $\mathsf{PrivK}^{\mathsf{cca}}$. We use $\mathcal{A}$ to construct an adversary $\mathcal{A}_{\text{enc}}$ for the CPA experiment with $(\mathsf{Gen}_E, \mathsf{Enc}, \mathsf{Dec})$. Adversary $\mathcal{A}_{\text{enc}}$ chooses a key k_2 and invokes the adversary $\mathcal{A}$. Whenever $\mathcal{A}$ asks an encryption query m, adversary $\mathcal{A}_{\text{enc}}$ queries its encryption oracle with m and receives back some c. Then $\mathcal{A}_{\text{enc}}$ computes $t = \mathsf{Mac}_{k_2}(c)$ and hands $\mathcal{A}$ the pair (c, t). Whenever $\mathcal{A}$ asks for a decryption query (c, t), $\mathcal{A}_{\text{enc}}$ checks if

(c,t) was generated in a previous encryption query. If yes, $\mathcal{A}_{\text{enc}}$ hands $\mathcal{A}$ the value m that was queried when (c,t) was generated. If no, $\mathcal{A}_{\text{enc}}$ hands $\mathcal{A}$ the response $\perp$. When $\mathcal{A}$ outputs a pair (m_0, m_1), adversary $\mathcal{A}_{\text{enc}}$ outputs the same pair and receives back a challenge ciphertext c. As above, $\mathcal{A}_{\text{enc}}$ hands $\mathcal{A}$ the challenge ciphertext (c,t) where $t = \mathsf{Mac}_{k_2}(c)$. Notice that $\mathcal{A}_{\text{enc}}$ does not need a decryption oracle because it assumes that any new query is always invalid. Furthermore, $\mathcal{A}_{\text{enc}}$ runs in probabilistic polynomial-time because it just invokes $\mathcal{A}$ and adds MAC tags (that are efficiently computable because $\mathcal{A}_{\text{enc}}$ chose k_2). It is straightforward to see that the success of $\mathcal{A}_{\text{enc}}$ in $\mathsf{PrivK}^{\mathsf{cpa}}$ when VALID-QUERY does not occur *equals* the success of $\mathcal{A}$ in $\mathsf{PrivK}^{\mathsf{cca}}$ when VALID-QUERY does not occur. That is,

$$\begin{aligned}\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A}_{\text{enc}},\Pi_E}(n) = 1 \wedge \neg\text{VALID-QUERY}_{\mathcal{A},\Pi'}(n)] \\ = \Pr[\mathsf{PrivK}^{\mathsf{cca}}_{\mathcal{A},\Pi'}(n) = 1 \wedge \neg\text{VALID-QUERY}_{\mathcal{A},\Pi'}(n)]]\end{aligned}$$

implying that

$$\begin{aligned}&\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A}_{\text{enc}},\Pi_E}(n) = 1] \\ &\geq \Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A}_{\text{enc}},\Pi_E}(n) = 1 \wedge \neg\text{VALID-QUERY}_{\mathcal{A},\Pi'}(n)] \\ &= \Pr[\mathsf{PrivK}^{\mathsf{cca}}_{\mathcal{A},\Pi'}(n) = 1 \wedge \neg\text{VALID-QUERY}_{\mathcal{A},\Pi'}(n)]\end{aligned} \tag{4.5}$$

Assume now by contradiction that there exists a *non-negligible function* ε such that

$$\Pr[\mathsf{PrivK}^{\mathsf{cca}}_{\mathcal{A},\Pi'}(n) = 1] = \frac{1}{2} + \varepsilon(n).$$

By the fact that $\Pr[\text{VALID-QUERY}_{\mathcal{A},\Pi'}(n)]$ is negligible, we have that it is smaller than $\varepsilon(n)/2$. This in turn implies that

$$\Pr[\mathsf{PrivK}^{\mathsf{cca}}_{\mathcal{A},\Pi'}(n) = 1 \wedge \text{VALID-QUERY}_{\mathcal{A},\Pi'}(n)] < \varepsilon(n)/2$$

and so

$$\begin{aligned}\Pr[\mathsf{PrivK}^{\mathsf{cca}}_{\mathcal{A},\Pi'}(n) = 1] &= \Pr[\mathsf{PrivK}^{\mathsf{cca}}_{\mathcal{A},\Pi'}(n) = 1 \wedge \neg\text{VALID-QUERY}_{\mathcal{A},\Pi'}(n)] \\ &\quad + \Pr[\mathsf{PrivK}^{\mathsf{cca}}_{\mathcal{A},\Pi'}(n) = 1 \wedge \text{VALID-QUERY}_{\mathcal{A},\Pi'}(n)] \\ &< \Pr[\mathsf{PrivK}^{\mathsf{cca}}_{\mathcal{A},\Pi'}(n) = 1 \wedge \neg\text{VALID-QUERY}_{\mathcal{A},\Pi'}(n)] + \frac{\varepsilon(n)}{2}.\end{aligned}$$

Rearranging the above, and using the fact that $\mathcal{A}$ succeeds in $\mathsf{PrivK}^{\mathsf{cca}}$ with probability $1/2 + \varepsilon(n)$, we have that

$$\begin{aligned}\Pr[\mathsf{PrivK}^{\mathsf{cca}}_{\mathcal{A},\Pi'}(n) = 1 \wedge \neg\text{VALID-QUERY}_{\mathcal{A},\Pi'}(n)] &> \Pr[\mathsf{PrivK}^{\mathsf{cca}}_{\mathcal{A},\Pi'}(n) = 1] - \frac{\varepsilon(n)}{2} \\ &= \frac{1}{2} + \frac{\varepsilon(n)}{2}.\end{aligned}$$

Combining this with Equation (4.5), we have that

$$\Pr[\mathsf{PrivK}^{\mathsf{cpa}}_{\mathcal{A}_{\text{enc}},\Pi_E}(n) = 1] > \frac{1}{2} + \frac{\varepsilon(n)}{2}$$

implying that $\mathcal{A}_{\text{enc}}$ succeeds in $\mathsf{PrivK}^{\mathsf{cpa}}$ with non-negligible advantage over $1/2$. Since this contradicts the CPA-security of $\Pi_E = (\mathsf{Gen}_E, \mathsf{Enc}, \mathsf{Dec})$, we conclude that Construction 4.17 is CCA-secure. ■

[1]

References

[1] J. K. a. Y. Lindell, Introduction to Modern Cryptography.