Mermory-efficient Fujisaki-Okamoto transformation

Ganyu (Bruce) Xu (g66xu)

May 23, 2024

1 OW-PCVA transformation

Let (K, E, D) be a public-key encryption scheme defined over $(\mathcal{K}_{\text{PKE}}, \mathcal{M}_{\text{PKE}}, \mathcal{C})$. Let (S, V) be a message authentication code defined over $(\mathcal{K}_{\text{MAC}}, \mathcal{M}_{\text{MAC}}, \mathcal{T})$. Let $G : \mathcal{M}_{\text{PKE}} \to \mathcal{K}_{\text{MAC}}$ be a hash function. The transformation outputs a public-key encryption scheme (K, E_1, D_1). The key generation routine and key space of the transformed scheme are identical to those of the input scheme. The encryption and decryption routines of the transformed scheme are as follows:

Algorithm 1: E_1

```
  Input: $\text{pk}, m \in \mathcal{M}_{\text{PKE}}$
1 $k \leftarrow G(m)$;
2 $\sigma \leftarrow E(\text{pk}, m)$;
3 $t \leftarrow S(k, \sigma)$;
4 return $c = (\sigma, t)$;
```

Algorithm 2: D_1

```
  Input: $\text{sk}, c = (\sigma \in \mathcal{C}, t \in \mathcal{T})$
1 $\hat{m} \leftarrow D(\text{sk}, \sigma)$;
2 $\hat{k} \leftarrow G(\hat{m})$;
3 if $V(\hat{k}, \sigma, t) \neq 1$ then
4     return $\perp$;
5 end
6 return $\hat{m}$;
```

If the input PKE is deterministic, OW-CPA secure, and δ-correct, and the input MAC is existentially unforgeable, then the "encrypt-then-MAC" PKE is OW-PCVA secure. More specifically:

Theorem 1.1. *For every OW-PCVA adversary against the transformed scheme (E_1, D_1) who makes q_G hash queries, q_P plaintext checking queries, and q_V ciphertext validation queries, and who has advantage $\epsilon_{\textit{OW-PCVA}}$, there exists an OW-CPA adversary against the underlying PKE with advantage $\epsilon_{\textit{OW-CPA}}$, a correctness adversary against the underlying PKE with advantage δ, and a existential forgery adversary against the MAC with advantage $\epsilon_{\textit{MAC}}$ such that:*

$$\epsilon_{\textit{OW-PCVA}} \leq (q_P + q_G) \cdot \delta + q_V \cdot \epsilon_{\textit{MAC}} + 2 \cdot \epsilon_{\textit{OW-CPA}}$$

Proof. To prove theorem 1.1, we use a sequence of game.

Game 0 is the OW-PCVA game, as described in the section above.

$$\epsilon_{\text{OW-PCVA}} = \epsilon_0$$

In Game 1, the PCO is replaced with an alternative implementation PCO_1:

Algorithm 3: PCO

```
  Input: $(m, c = (\sigma, t))$
1 $\hat{m} \leftarrow D(\text{sk}, \sigma)$;
2 $\hat{k} \leftarrow G(\hat{m})$;
3 return $[\![\hat{m} = m]\!]$ and $[\![V(\hat{k}, \sigma, t)]\!]$
```

Algorithm 4: PCO_1

```
  Input: $(m, c = (\sigma, t))$
1 $k \leftarrow G(m)$;
2 $\hat{\sigma} \leftarrow E(\text{pk}, m)$;
3 return $[\![\sigma = \hat{\sigma}]\!]$ and $[\![V(k, \sigma, t)]\!]$
```

For any single plaintext checking query, the two oracles will disagree if and only if correctness of (K, E, D) is broken, so such probability can be bounded by δ. From the OW-PCVA adversary's perspective, the two games behave differently if and only if the two oracles disagree on at least one of the plaintext checking queries, which is at most $q_P \cdot \delta$:

$$\epsilon_0 - \epsilon_1 \leq q_P \cdot \delta$$

In Game 2, the CVO is replaced with an alternative implementation CVO_1, the latter of which checks the hash oracle's tape $\mathcal{L}^G$

Algorithm 5: CVO

```
Input: $c = (\sigma, t)$
1 $\hat{m} \leftarrow D(\text{sk}, \sigma)$;
2 $\hat{k} \leftarrow G(\hat{m})$;
3 return $[\![V(\hat{k}, \sigma, t) = 1]\!]$
```

Algorithm 6: CVO_1

```
Input: $c = (\sigma, t)$
1 if $\exists(\tilde{m}, \tilde{k}) \in \mathcal{L}^G$ s.t. $E(\text{pk}, \tilde{m}) = c$ and
    $V(\tilde{k}, \sigma, t) = 1$ then
2     return 1;
3 end
4 return 0;
```

There are two scenarios in which the two oracles can disagree. First, there is a matching hash query $(\tilde{m}, \tilde{k}) \in \mathcal{L}^G$, but $\tilde{m}$ triggers a decryption failure, so the same σ decrypts to $\hat{m} \neq \tilde{m}$. For a single hash query, the probability of decryption error is at most δ, so across q_G hash queries, the probability of at least one decryption error is at most $q_G \cdot \delta$. Second, (σ, t) is a valid message-tag pair that will pass CVO, but there is no matching hash query. Under the random oracle model, from the perspective of the adversary $k \leftarrow G(m)$ is a truly random and unknown MAC key, which means that (σ, t) is a forgery. The probability of a single ciphertet validation query being a valid forgery is bounded by the advantage of a MAC adversary, so across all q_V ciphertext validation queries, the probability of having at least one valid forgery is at most $q_V \cdot \epsilon_{\text{MAC}}$. Finally, the probability of the two implementations disagree is at most the sum of the probabilities of the two scenarios:

$$\epsilon_1 - \epsilon_2 \leq q_G \cdot \delta + q_V \cdot \epsilon_{\text{MAC}}$$

In game 3, the challenge encryption routine is modified. Instead of computing a pseudorandom MAC key $k^* \leftarrow G(m^*)$, a truly random MAC key is sampled uniformly from the message space $k^* \xleftarrow{\$} \mathcal{K}_{\text{MAC}}$.

Algorithm 7: OW-PCVA game 2

```
1 $(\text{pk}, \text{sk}) \leftarrow \text{KeyGen}(1^\lambda)$;
2 $m^* \xleftarrow{\$} \mathcal{M}_{\text{PKE}}$;
3 $k^* \leftarrow G(m^*)$;
4 $\sigma^* \leftarrow E(\text{pk}, m^*)$;
5 $t^* \leftarrow S(k^*, \sigma^*)$;
6 $c^* = (\sigma^*, t^*)$;
7 $\hat{m} \leftarrow \mathcal{A}_{\text{OW-PCVA}}^{\mathcal{O}^G, \text{PCO}_1, \text{CVO}_1}(1^\lambda, \text{pk}, c^*)$;
8 return $[\![\hat{m} = m^*]\!]$
```

Algorithm 8: OW-PCVA game 3

```
1 $(\text{pk}, \text{sk}) \leftarrow \text{KeyGen}(1^\lambda)$;
2 $m^* \xleftarrow{\$} \mathcal{M}_{\text{PKE}}$;
3 $k^* \xleftarrow{\$} \mathcal{K}_{\text{MAC}}$;
4 $\sigma^* \leftarrow E(\text{pk}, m^*)$;
5 $t^* \leftarrow S(k^*, \sigma^*)$;
6 $c^* = (\sigma^*, t^*)$;
7 $\hat{m} \leftarrow \mathcal{A}_{\text{OW-PCVA}}^{\mathcal{O}^G, \text{PCO}_1, \text{CVO}_1}(1^\lambda, \text{pk}, c^*)$;
8 return $[\![\hat{m} = m^*]\!]$
```

Under the random oracle model, game 3 and game 2 are indistinguishable from the adversary's perspective unless it makes a hash query on the value of m^*. Denote the probability that the adversary makes a hash query on m^* by $P[\text{QUERY}^*]$, then:

$$\epsilon_2 - \epsilon_3 \leq P[\text{QUERY}^*]$$

we can bound $P[\text{QUERY}^*]$ by constructing an OW-CPA adversary that simulates game 3 for a OW-PCVA adversary as a sub-routine. After the OW-PCVA adversary halts, the OW-CPA adversary checks through the tape of the hash oracle. Since the encryption routine is deterministic, if m^* exists in the hash oracle tape, then it can be identified for sure. In other words, if m^* has been queried by the OW-PCVA sub-routine, then the OW-CPA adversary can be guaranteed to win its game

$$P[\text{QUERY}^*] = \epsilon_{\text{OW-CPA}}$$

Finally, game 3 can be simulated by another OW-CPA adversary. First, all three oracles $\text{PCO}_1, \text{CVO}_1$ and G can be simulated using the same public key. When the OW-CPA challenger produces the the challenge ciphertext $\sigma^* \in \mathcal{C}$, the OW-CPA adversary samples a random MAC key $k^* \leftarrow \mathcal{K}_{\text{MAC}}$ and computes the tag $t^* \leftarrow S(k^*, \sigma^*)$ before passing $c^* = (\sigma^*, t^*)$ to the OW-PCVA adversary as its challenge ciphertext. When the OW-PCVA adversary halts, the OW-CPA adversary passes OW-PCVA adversary's output as its own output. It's easy to see that the OW-CPA adversary wins if and only if the OW-PCVA adversary wins:

$$\epsilon_3 = \epsilon_{\text{OW-CPA}}$$

Putting all the inequalities above together gives us the desired inequality. □

1.1 Deterministic OW-CPA transformation

In the OW-PCVA transformation, we required the input encryption scheme to be deterministic. In this section, we present a transformation that takes a probabilistic PKE and output a deterministic PKE that provides identical level of OW-CPA security.

Let (K, E, D) be a PKE defined over $(\mathcal{K}, \mathcal{M}, \mathcal{C})$. We assume that the encryption routine is probabilistic and its randomness is derived from some seed $r \in \text{COIN}$. The encryption routine accepts as an argument a coin value that will be used to pseudorandomly derive all randomness in the routine. Let H be a hash function $H : \mathcal{M} \rightarrow \text{COIN}$.

The transformed PKE is defined over the same spaces and shares the key generation and decryption routine with the input PKE, The only difference lies with its encryption routine:

Algorithm 9: $E^{\cancel{\$}}$

```
Input: $(\text{pk}, m)$
$r \leftarrow H(m)$;
$\sigma \leftarrow E(\text{pk}, m; r)$;
return $\sigma$
```

The transformed PKE is deterministic, and offers comparable level of OW-CPA security as the input PKE. More specifically:

Theorem 1.2. *For every OW-CPA adversary against the deterministic PKE* $(K, E^{\cancel{\$}}), D$ *that makes* q_H *hash queries to* H *and has advantage* $\epsilon_{\cancel{\$}}$*, there exists an OW-CPA adversary against the probabilistic PKE* (K, E, D) *with advantage* $\epsilon_{\$}$ *such that:*

$$\epsilon_{\cancel{\$}} \leq (1 + q_H) \cdot \epsilon_{\$}$$

Proof. We prove using a sequence of game. Denote the OW-CPA adversary against the deterministic PKE by $\mathcal{A}_{\cancel{\$}}$ and its advantage by $\epsilon_{\cancel{\$}}$; denote the OW-CPA adversary against the probabilistic PKE by $\mathcal{A}_{\$}$ and its advantage by $\epsilon_{\$}$

Game 0 is the standard OW-CPA game: $\epsilon_0 = \epsilon_{\cancel{\$}}$

In game 1, the challenge encryption routine is modified. Instead of using a pseudorandom coin, a truly random coin is used.

Algorithm 10: Game 0

```
$(\text{pk}, \text{sk}) \xleftarrow{\$} \text{KeyGen}(1^\lambda)$;
$m^* \xleftarrow{\$} \mathcal{M}$;
$r^* \leftarrow H(m^*)$;
$c^* \leftarrow E(\text{pk}, m^*; r^*)$;
$\hat{m} \xleftarrow{\$} \mathcal{A}_{\text{OW-CPA}}(1^\lambda, \text{pk}, c^*)$;
return $[\![\hat{m} = m^*]\!]$
```

Algorithm 11: Game 1

```
$(\text{pk}, \text{sk}) \xleftarrow{\$} \text{KeyGen}(1^\lambda)$;
$m^* \xleftarrow{\$} \mathcal{M}$;
$r^* \xleftarrow{\$} \text{COIN}$;
$c^* \leftarrow E(\text{pk}, m^*; r^*)$;
$\hat{m} \xleftarrow{\$} \mathcal{A}_{\text{OW-CPA}}(1^\lambda, \text{pk}, c^*)$;
return $[\![\hat{m} = m^*]\!]$
```

Under the random oracle assumption, game 0 and game 1 are indistinguishable from the perspective of $\mathcal{A}_{\not\$}$ except for when $\mathcal{A}_{\not\$}$ makes a hash query $\tilde{m} = m^*$. Denote the event that $\mathcal{A}_{\not\$}$ makes such a query by QUERY*, then by the difference lemma:

$$\epsilon_0 - \epsilon_1 \leq P[\text{QUERY}^*]$$

We can bound $P[\text{QUERY}^*]$ by constructing a OW-CPA adversary against the probabilistic scheme that uses $\mathcal{A}_{\not\$}$ as a sub-routine and simulates game 1 for $\mathcal{A}_{\not\$}$. When $\mathcal{A}_\$$ receives the probabilistic challenge ciphertext c^*, it directly passes it to the sub-routine $\mathcal{A}_{\not\$}$. After the sub-routine halts, $\mathcal{A}_\$$ samples from the hash oracle tape a random message as its output. If the sub-routine $\mathcal{A}_{\not\$}$ indeed makes the correct query m^*, then the probability that a randomly chosen query value being the correct value is $\frac{1}{q_H}$:

$$\epsilon_\$ = P[\text{QUERY}^*] \cdot \frac{1}{q_H}$$

Game 1 can be simulated by an OW-CPA adversary against the probabilistic PKE:

$$\epsilon_1 = \epsilon_\$$$

Putting the three equations above together gives us the desired inequality. □