Mermory-efficient Fujisaki-Okamoto transformation

Ganyu (Bruce) Xu (g66xu)

April 25, 2024

1 OW-PCVA transformation

Let (KeyGen, E, D) define a probabilistic public-key encryption scheme, then T is a transformation that takes this scheme and outputs a second public-key encryption scheme. The transformation makes use of two additional components:

1. A $\text{MAC} : \mathcal{K}^{\text{MAC}} \times \{0,1\}^* \to \mathcal{T}$
2. A hash function $G : \mathcal{M} \to \text{Coin} \times \mathcal{K}^{\text{MAC}}$

Algorithm 1: OW-PCVA encryption E^T

Input: pk, $m \in \mathcal{M}$
$(r, k) \leftarrow G(m)$;
$c \leftarrow E(\text{pk}, m, r)$;
$t \leftarrow \text{MAC}(k, c)$ `// "encrypt-then-mac"`;
return (c, t);

Algorithm 2: OW-PCVA decryption D^T

Input: $\text{sk}, c \in \mathcal{C}, t \in \mathcal{T}$
$\hat{m} \leftarrow D(\text{sk}, c)$;
$(\hat{r}, \hat{k}) \leftarrow G(\hat{m})$;
if $\text{MAC}(\hat{k}, c) \neq t$ **then**
  return $\perp$;
end
return $\hat{m}$;

Instead of using re-encryption to check for the integrity of the ciphertext, a MAC tag is used. At the cost of adding a fixed number of bytes to the ciphertext, the decryption routine becomes significantly faster and memory-efficient. The memory trade-off is especially important since in Kyber, re-encryption needs to compute the entire public key $A \in R_q^{k \times k}$ from the 32-byte seed.

We claim that this transformation is still OW-PCVA. The proof will be largely similar to that of the original OW-PCVA transformation proof, but with some notable differences:

- When queried on some authenticated ciphertext $(\tilde{c}, \tilde{t})$, the secret-key-less CVO_1 searches the hash oracle G for (m, r, k) such that $\text{MAC}(k, \tilde{c}) = \tilde{t}$. The diverging event between using CVO and using CVO_1 is "adversary produces valid authenticated ciphertext without consulting the hash oracle", which is equivalent to selective forgery (or existential forgery, I am not sure which): $\epsilon_a - \epsilon_b \leq q_V \cdot \epsilon_{\text{MAC}}$

Theorem 1.1. *Let* (KeyGen, E, D) *be a public-key encryption scheme that is* δ*-correct and that has* γ*-spread. For every OW-PCVA adversary against the transformed encryption scheme* (E^T, D^T) *that makes* q_G *hash queries and* q_V *ciphertext validation queries and that has advantage* $\epsilon_{OW\text{-}PCVA}$*, there exists an IND-CPA adversary against* (KeyGen, E, D) *with advantage* $\epsilon_{IND\text{-}CPA}$ *and an EF-CMA adversary against the MAC with advantage* ϵ_{MAC} *such that:*

$$\epsilon_{OW\text{-}PCVA} \leq q_G \cdot \delta + q_V \cdot \epsilon_{MAC} + \frac{2q+1}{|\mathcal{M}|} + \epsilon_{IND\text{-}CPA}$$

The proof will be discussed in section 3.

2 Proof techniques

Similar to the 2017 paper, we will use a sequence of games to incrementally replace PCO, CVO, and other aspects of the standard OW-PCVA game with some simulation. Then we will show that the total loss of security is negligible.

2.1 Replacing plaintext-checking oracle

The plaintext-checking oracle PCO takes as input a plaintext-ciphertext pair $(m, (c, t))$ and return "reject" if and only if the the ciphertext is a valid encryption of the plaintext and vice versa under the context of some fixed keypair.

The simulated plaintext-checking oracle PCO_1 removes the requirement for the secret key by removing the step that checks whether the queried ciphertext (c, t) decrypts back to the queried plaintext m.

From the OW-PCVA adversary's perspective, the two oracles are indistinguishable, except for when there is a decryption error $D(\text{sk}, E(\text{pk}, m, r)) \neq m$ for the input plaintext m and the corresponding coin r. The MAC is not involved in this argument. The loss of security is still $q_G \cdot \delta$

Algorithm 3: Vanilla PCO

Input: $m \in \mathcal{M}, c \in \mathcal{C}, t \in \mathcal{T}$

if $D^T(sk, (c, t)) \neq m$ **then**
  return $\perp$;
  `// Decryption did not match`
end
if $E^T(pk, m) \neq (c, t)$ **then**
  return $\perp$;
  `// Encryption did not match`
end

Algorithm 4: Simulated PCO_1

Input: $m \in \mathcal{M}, c \in \mathcal{C}, t \in \mathcal{T}$

if $E^T(pk, m) \neq (c, t)$ **then**
  return $\perp$;
  `// Encryption did not match`
end

2.2 Replacing ciphertext-validation oracle

This is a similar trick to what was used in the 2017 paper, as well. Where the ciphertext is honestly generated, both oracles will return accept. Where the vanilla CVO rejects the ciphertext (aka ciphertext is malformed or the tag doesn't match), the simulated CVO will also reject. Therefore, the diverging event is when the vanilla CVO accepts but the simulated CVO rejects. Since the vanilla CVO accepts the query, the tag t must be valid for the queried ciphertext c; on the other hand, the simulated CVO's rejection means there is no matching query in the hash oracle. Under the random oracle assumption, t is a valid tag for some data c under some key that the adversary does not know. In other words, (c, t) is some kind of forgery.

The argument on how the security of the MAC relates to this diverging event is still a bit fuzzy, but here are two possible ways I can think of:

- For each CVO query, the adversary is trying to forge tag for that specific ciphertext. This means there is a selective forgery attack, and over all q_V validation queries, the probability of having at least one selective forgery attack that works is at most $q_V \epsilon_{\text{SF}}$
- Across all CVO query, the adversary wants to forge tag for some ciphertext, meaning there is a existential forgery attack, and the probability is at most ϵ_{EF}

Algorithm 5: Vanilla CVO

Input: $(c \in \mathcal{C}, t \in \mathcal{T})$

if $D^T(sk, (c, t)) = \perp$ **then**
  return $\perp$;
end

Algorithm 6: Simulated CVO_1

Input: $(c \in \mathcal{C}, t \in \mathcal{T})$

if $\exists (m, r, k) \in \mathcal{O}^G$ *such that* $\text{MAC}(k, c) = t$ **then**
  return *Accept*;
end
return $\perp$

2.3 Random until queried

We then replace the pseudorandom coin and the MAC key with truly random coins and truly random keys in the challenge encryption. These two challenge encryption routines are identical, unless the adversary queries G with the challenge plaintext m^*, but the probability of making such query can be bounded.

Algorithm 7: Challenge encryption

$m^* \xleftarrow{\$} \mathcal{M}$;
$(c^*, t^*) \leftarrow E^T(\text{pk}, m^*)$;
return (c^*, t^*)

Algorithm 8: Simulated challenge encryption

$m^* \xleftarrow{\$} \mathcal{M}$;
$r^* \xleftarrow{\$} \text{Coin}, k^* \xleftarrow{\$} \mathcal{K}_{\text{MAC}}$;
$c^* \leftarrow E(\text{pk}, m^*, r^*)$;
$t^* \leftarrow \text{MAC}(k^*, c^*)$ **return** (c^*, t^*)

Let the OW-PCVA adversary's advantage using the vanilla challenge encryption be ϵ_0 and its advantage using the simulated challenge encryption be ϵ_1. Denote the event that the adversary queries the hash oracle G with the challenge plaintext m^* by QUERY*, then:

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

We can build an IND-CPA adversary against the underlying encryption scheme to bound $P[\text{QUERY}^*]$: if the OW-PCVA adversary makes the magic query, then the IND-CPA adversary can find it in the hash oracle's tape and win the IND-CPA game; if the OW-PCVA adversary does not make the magic query, then the IND-CPA adversary outputs a blind guess:

$$P[\text{QUERY}^*] \leq \epsilon_{\text{IND-CPA}}$$

3 Complete proof of 1.1

Proof. We will prove using a sequence of games. Game 0 is the standard OW-PCVA game.

Game 1 is identical to game 0, except PCO is replaced with PCO_1, the loss of security is described in section 2.1:

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

Game 2 is identical to game 1, except CVO is replaced with CVO_1, the loss of security is described in section 2.2. Here I put a placeholder ϵ_{MAC} to indicate that this quantity is tied to the security of the MAC, but I don't have a solid answer yet

$$\epsilon_1 - \epsilon_2 \leq \epsilon_{\text{MAC}}$$

Game 3 is identical to game 2, except the challenge encryption is replaced with the simulated challenge encryption

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

From section 2.3 we know that $P[\text{QUERY}^*] \leq \epsilon_{\text{IND-CPA}}$ for some IND-CPA adversary against the underlying PKE.

Game 3 can be entirely simulated by an OW-CPA adversary against the underlying PKE:

- The keypairs are identical between the two PKE's
- The OW-CPA adversary can simulate PCO, CVO, and hash oracle G
- The OW-CPA adversary receives challenge ciphertext c^* which is computed from a truly random coin; it can then random a truly random MAC key and compute the tag t^*

- The OW-CPA adversary passes (c^*, t^*) to the OW-PCVA adversary and returns whatever the OW-PCVA adversary returns

The OW-CPA adversary wins if and only if the OW-PCVA adversary wins: $\epsilon_3 = \epsilon_{\text{OW-CPA}}$.
Putting everything together we have:

$$\epsilon_0 \leq q_G \cdot \delta + \epsilon_{\text{MAC}} + \epsilon_{\text{IND-CPA}} + \epsilon_{\text{OW-CPA}}$$

□

4 Open questions

Can we get rid of the coin and just let the encryption scheme be probabilistic? Do we need to?