УДК 004.056.55

doi.org/10.15407/fmmit2023.36.101

Аналіз CPA-to-CCA перетворення ДСТУ 8961:2019 у моделі випадкового оракула

Сергій Кандій[1]

[1] співробітник АТ «Інститут Інформаційних технологій» вул. Коломенська, 15, 61166, Харків, e-mail: sergeykandy@gmail.com

Для побудови сучасних механізмів інкапсуляції ключів використовується модульний підхід. За основу береться деяка асиметрична схема і за допомогою певного CPA-to-CCA перетворення на її основі будується механізм інкапсуляції ключів. У багатьох сучасних механізмах інкапсуляції ключів використовуються доказово безпечні перетворення, проте для ДСТУ 8961:2019 наразі спостерігається суттєвий брак такого аналізу. Аналіз подібних перетворень зазвичай відбувається не у стандартній моделі, а у моделі випадкового оракула, оскільки доволі важко врахувати вплив геш функцій. У роботі наведено перші результати аналізу у моделі випадкового оракула CPA-to-CCA перетворення, що використовується в стандарті ДСТУ 8961:2019 для побудови механізма інкапсуляції ключів. Показано, що конструкція є безпечною за умови, якщо виконується ряд модельних припущень.

Ключові слова: ДСТУ 8961:2019; модель випадкового оракула; криптографія на решітках; доказова безпека; механізми інкапсуляції ключів

Вступ. Зазвичай, для практичних застосувань механізми інкапсуляції ключів (КЕМ) вважаються безпечними [1,2], якщо складність будь-якої атаки з адаптивно підібраними шифротекстами є занадто великою для практичної реалізації. Якщо КЕМ містить доказ того, що він є безпечним у зазначеному вище сенсі за умови складності певного невеликого набору теоретико-числових проблем при конкретних значеннях загальносистемних параметрів, то кажуть, що він є безпечним у стандартній моделі [3]. Нажаль, докази у стандартній моделі часто не можливо отримати для реальних КЕМ. Причиною цього є використання криптографічних геш функцій [3]. Часто відсутні інструменти для того, щоб врахувати вплив алгебраїчної структури та властивостей геш функції на безпеку перетворень. Тому, на практиці, поширеною модифікацією є модель випадкового оракула [4], у якій усі геш функції замінюються на випадкові оракули - ідеалізовані геш функції, що не мають внутрішньої структури.

1. Формальний опис ДСТУ 8961:2019

Введемо позначення. Нехай $R_q = Z_q[X]/(X^n - X - 1)$ – кільце поліномів над Z_q з твірним поліномом $X^n - X - 1$, а R_3 – множина поліномів кільця R_q, усі

коефіцієнти яких належать до множини $\{-1,0,1\}$. Позначимо як $R_3^{a,b}$ множину усіх поліномів у R_3, що мають кількість ненульових елементів у діапазоні $[a,b]$. Якщо $a=b$, то використовується скорочене позначення $R_3^{t,t}=R_3^t$. ДСТУ 8961:2019 використовує наступні геш функції:

$$\begin{aligned} BPGM &: \{0,1\}^{8*\max_{MsgLenBytes}+db} \times R_q \to R_q \\ MGF &: R_q \to \{0,1\}_3 \\ H &: R_q \to \{0,1\}^{\lambda} \\ KDF &: R_q \to \{0,1\}^{K_{bytes}} \end{aligned} \tag{1}$$

де λ – параметр безпеки, t – загальносистемний параметр, а $\max_{MsgLenBytes}$, db, K_{bytes} є константами, що визначаються на основі загальносистемних параметрів. Також використовується бієктивне відображення:

$$\begin{aligned} Pad &: \{0,1\}^{8*\max_{MsgLenBytes}} \times \{0,1\}^{db} \to R_3 \\ Pad^{-1} &: R_3 \to \{0,1\}^{8*\max_{MsgLenBytes}} \times \{0,1\}^{db} \end{aligned} \tag{2}$$

,яке кодує повідомлення (у вигляді строки бітів), довжину повідомлення та випадкову строку бітів у поліном з R_3.

Схема асиметричного шифрування, що лежить в основі протоколу інкапсуляції ключів, наведена на рисунку 1. CPA-to-CCA перетворення ДСТУ 8961:2019 наведено на рисунку 2.

SkelyaPKE.Gen(1^{λ}):
1. $f \leftarrow_R R_3^{2t}$
2. $g \leftarrow_R R_3^{\left\lfloor \frac{2n}{3}+1 \right\rfloor}$
3. $if\ (3f+1)^{-1} = \perp$ goto 1
4. $h = (3f+1)^{-1} g \in R_q$
5. $return\ (pk = h, sk = f$

SkelyaPKE.Enc($msg, coins, h$):
1. $m = Pad(msg, coins)$
2. $r = BPGM(msg, coins, h)$
3. $R = rh$
4. $m' = m + MGF(R)$
5. $if\ m' \notin R_3^{2t,n-2t}$ return $\perp$
6. $c = R + m'$
7. $return(c)$

SkelyaPKE.Dec($c, (f,h)$):
1. $a = fc$
2. $m' = a \bmod 3$
3. $if\ m' \notin R_3^{2t,n-2t}$ return $\perp$
4. $R = c - m'$
5. $m = m' - MGF(R)$
6. $(msg, coins) = Pad^{-1}(m)$
7. $r' = BPGM(msg, coins, h)$
8. $R' = r'h$
9. $if\ R' = R\ return\ msg$
10. $return\ \perp$

Рис 1. Асиметрична схема шифрування у ДСТУ 8961:2019

SkelyaKEM.Gen(1^{λ}):
1. $return\ (pk, sk) = SkelyaPKE.Gen(1^{\lambda})$

SkelyaKEM.Encaps($pk = h$):
1. $x \leftarrow_R \{0,1\}^{MsgLen}$
2. $r = BPGM(x, h)$
3. $C_1 = SkelyaPKE.Enc(x, r, pk)$
4. $if\ C_1 = \perp$ goto 1
5. $C_2 = H(r)$
6. $K = KDF(r)$
7. $C = (C_1, C_2)$
8. 8. $return\ (C, K)$

SkelyaKEM.Decaps($C = (C_1, C_2), sk = (f,h)$):
1. $x = SkeylaPKE.Dec(C_1, sk)$
2. $if\ x = \perp$ return $\perp$
3. $r = BPGM(x, h)$
4. $C_2' = H(r)$
5. $C_1' = SkelyaPKE.Enc(x, r, h)$
6. $if\ C_1' = C_1$ && $C_2' = C_2$
7. $return\ K = KDF(r)$
8. $return\ \perp$

Рис 2. Механізм інкапсуляції ключів ДСТУ 8961:2019

Від асиметричної схеми додатково вимагається, щоб існувало достатньо мала константа γ, для якої виконується нерівність

$$|r : Enc(pk, r, m) = C| \leq \gamma(\lambda) = negl(\lambda) \quad (3)$$

2. Безпека CPA-to-CCA перетворення

IND-CCA2 безпеки SkelyaPKE у моделі випадкового оракула грунтується на стандартних техніках. Використовується наступна технічна лема [5]:

Лема 1. Позначимо як A, B, E деякі події в ймовірністному просторі. Якщо $\Pr[A|\neg E] = \Pr[B|\neg E]$, то має місце нерівність $|\Pr[A] - \Pr[B]| \leq \Pr[E]$.

Теорема 1. Нехай $PPKE = (Gen, Enc, Dec)$ - деяка ймовірнісна схема асиметричного шифрування, а SkelyaKEM - механізм інкапсуляції ключей, що побудований за допомогою застосування перетворення на Рис 2 до PPKE. Якщо існує алгоритм A, що може перемогти у IND-CCA2 грі SkelyaKEM за поліноміальний час з ймовірністю ε та робить $q_{BPGM}, q_H, q_{KDF}, q_{Decaps}$ запитів до випадкових оракулів BPGM ,H,KDF та оракула дешифрування, то існує алгоритм B, що може інвертувати PPKE з ймовірністю ε'.

$$\varepsilon' \geq \varepsilon - \frac{q_{Decaps}}{|R_3^{2t,N-2t}|} - \frac{\gamma q_{Decaps}}{2^{\lambda}} \quad (4)$$

Доказ.

Алгоритм B приймає у якості аргумента шифротекст PPKE- C_1^*, відкритий ключ pk та повертає x. Сутність алгоритму B полягає у симуляції IND-CCA2 гри для алгоритму A та працює наступним чином:

Крок 1: Алгоритм B генерує випадкові бітові строки C_2^*, K^*.

Крок 2: Алгоритм B передає pk до A.

Крок 3: Алгоритм B чекає доки A запросить завдання, на запит B посилає пару $\left(C^* = (C_1^*, C_2^*), K^*\right)$.

Крок 4: Алгоритм B чекає доки A поверне біт σ'.

Крок 5: Алгоритм B перевіряє чи існує $(x, r) \in BPGM_{LIST}$, для якого виконується $Enc(x, r, pk) = C_1^*$. Якщо так, то обчислює та повертає x.

Крок 6: Алгоритм B перевіряє чи існує $(r, K) \in KDF_{LIST}$ або $(r, hash) \in H_{LIST}$, для якого виконується $x = Pad^{-1}(C_1^* - rh - MGF(rh)); Enc(x, r, pk) = C_1^*$. Якщо так, то повертає x.

Крок 7: Алгоритм B повертає випадкове x.

Оракул для $BPGM(x)$ працює наступним чином:

Крок 1: Якщо $(x, r) \in BPGM_{LIST}$, то повернути r.

Крок 2: Якщо $Decaps(C_1^*, sk) = x$, то повернути $BPGM(x)$.

Крок 3: Інакше обрати випадковие r, додати до $BPGM_{LIST}$ (x,r) та повернути r.

Оракул для $H(r)$ працює наступним чином:

Крок 1: Якщо $(r,hash) \in H_{LIST}$, то повернути $hash$.

Крок 2: Якщо $BPGM(Decaps(C_1^*, sk), pk) = r$, то повернути $H(r)$.

Крок 3: Інакше обрати випадковие $hash$, додати до H_{LIST} $(r,hash)$ та повернути $hash$.

Оракул для $KDF(r)$ працює наступним чином:

Крок 1: Якщо $(r,K) \in KDF_{LIST}$, то повернути K.

Крок 2: Якщо $BPGM(Decaps(C_1^*, sk), pk) = r$, то повернути $KDF(r)$.

Крок 3: Інакше обрати випадковие K, додати до KDF_{LIST} (r,K) та повернути K.

Оракул декапсуляції $Decaps((C_1, C_2))$ працює наступним чином:

Крок 1: Якщо $C_1 = C_1^*$, то повернути $\perp$.

Крок 2: Для кожного значення $(x,r) \in BPGM_{LIST}$ перевірити, чи виконується $Enc(x,r,pk)$ та $H(r) = C_2$. Якщо такої пари не знайдено, то повернути $\perp$.

Крок 3: Обчислити $K = KDF(r)$ з використанням оракула для KDF.

Крок 4: Повернути K.

Позначимо як W_{real}^{win} та W_{oracle}^{win} події, що відповідають перемозі A у IND-CCA2 грі з реальними геш функціями та з оракулами відповідно. Розглянемо як зміниться ймовірність перемоги, якщо замінити справжні геш функції на випадкові оракули. Для алгоритму A різниця не буде помітна, якщо: A робив запит на розшифрування завдання до того, як отримав завдання або A робив запит на декапсуляцію валідного шифротексту C_1, C_2, але A не робив відповідних запитів до H або BPGM.

Позначимо як W_{ind} подію, що зазначені вище випадки не стануться, отже, використовуючи Лемму 1, маємо:

$$\begin{aligned} &\Pr[W_{real}^{win} \mid \neg W_{ind}] = \Pr[W_{oracle}^{win} \mid \neg W_{ind}] \Rightarrow \\ &\quad \mid \Pr[W_{real}^{win}] - \Pr[W_{oracle}^{win}] \mid \leq \Pr[W_{ind}] \end{aligned} \tag{5}$$

Так як шифротекст був обраний випадково, то ймовірність першої події обмежена $q_{Decaps} / \mid R_3^{2t,N-2t} \mid$. Ймовірність того, що $H(BPGM(Decaps(C, sk), pk))$ обмежена $\gamma q_{Decaps} / 2^{\lambda}$. Отже, повна ймовірність буде

$$\frac{q_{Decaps}}{\mid R_3^{2t,N-2t} \mid} + \frac{\gamma q_{Decaps}}{2^{\lambda}} \tag{6}$$

Оскільки перевага у грі з оракулом є нижньою оцінкою ймовірності того, що серед запитів до геш функцій буде інформація, якої достатньо для дешифрування шифротексту C_1^* , то маємо:

$$\varepsilon' \geq \varepsilon - \frac{q_{Decaps}}{|R_3^{2t,N-2t}|} - \frac{\gamma q_{Decaps}}{2^{\lambda}} \tag{7}$$

Що і треба було довести.

Висновки. У роботі було доведена в моделі випадкового оракула IND-CCA2 безпека CPA-to-CCA перетворення ДСТУ 8961:2019. Важливо розуміти, що доказ у моделі випадкового оракула не означає, що взагалі атак не має, а лише доводить захист від широкого класу атак, що підпадають під модельні припущення. Наведений в цій роботі доказ може в подальшому бути вдосконалений з використанням більш сильних технік та з меншою кількістю модельних припущень. Досліджуючи властивості загальних параметрів та доповнюючи існуючі докази можливо створити більш комплексні моделі безпеки, що враховують більшу кількість загроз і мають менше ризиків.

Література

[1] CRYSTALS – Kyber: a CCA-secure module-lattice-based KEM / Leo Ducas., and other. // URL: https://eprint.iacr.org/2017/634.pdf
[2] FrodoKEM. Learning With Errors Key Encapsulation Algorithm Specifications. And Supporting Documentation. / Leo Ducas., and other. // URL: https://frodokem.org/files/FrodoKEM-specification-20210604.pdf
[3] Introduction to Modern Cryptography: Principles and Protocols / Katz, Jonathan; Lindell, Yehuda // Chapman & Hall/CRC Cryptography and Network Security Series, CRC Press, 2014.
[4] Canetti R., Goldreich O., Halevi S. The random oracle methodology, revisited. In: 30th symposium on theory of computing. STOC, 1998. P. 209–218.
[5] A. Dent. A Designer's Guide to KEMs. // URL: http://www.cogentcryptography.com/papers/designer.pdf

Analysis of DSTU 8961:2019 CPA-to-CCA transformation in random oracle model

Kandii Serhii

A modular approach is usually used to build modern key encapsulation mechanisms. Some asymmetric scheme is taken as a basis, and with the help of a certain CPA-to-CCA transformation, a key encapsulation mechanism is built on its basis. Many modern key encapsulation mechanisms use provably secure transformations, but for DSTU 8961:2019 there is currently a significant lack of such an analysis. The analysis of such transformations usually takes place not in the standard model, but in the random oracle model, since it is quite difficult to take into account the influence of hash functions. The paper presents the first results of the analysis in the model of the random oracle CPA-to-CCA conversion, which is used in the DSTU 8961:2019 standard to build the key encapsulation mechanism. It is shown that the construction is safe provided that a number of model assumptions are fulfilled.

Отримано 05.04.23