УДК 621.391:519.2:519.7

А.П. БУБИР, І.Д. ГОРБЕНКО, д-р техн. наук

ОЦІНКА СТІЙКОСТІ НАПРАВЛЕНОГО ШИФРУ NTRU ДО АТАКИ, ЗАСНОВАНОЇ НА ЗМІНИХ ЗНАЧЕННЯХ ЧАСУ РОЗШИФРУВАННЯ

Вступ

Нині суттєву актуальність має проблема підвищення швидкодії направленого шифрування. Але вона повинна бути вирішеною за умови забезпечення експоненційної складності криптоаналізу проти атаки типу «повне розкриття». Для її вирішення широке розповсюдження та використання отримали криптографічні механізми й протоколи направленого шифрування, що ґрунтуються на перетвореннях в кільцях, полях та групах точок еліптичних кривих. Як показує аналіз, усі вони є обчислювально складними, але тільки перетворення в групі точок ЕК забезпечує експоненційну складність атаки «повне розкриття».

Перспективним напрямом вирішення протиріччя, яке міститься в необхідності зменшення складності (збільшення швидкодії) прямого та зворотного криптоперетворень направленого шифрування, є застосування алгоритму направленого шифрування в фактор-кільцях, що отримав назву NTRU [1 – 3]. Підтвердження цьому є прийняття стандарту ANSI X9.98 [1] та значне його впровадження для направленого шифрування. Основний позитивний ефект, який досягається при використанні таких перетворень – це підвищення швидкодії на два-три десяткові порядки. Але, на наш погляд, стійкість даного класу асиметричних крипто перетворень вивчена недостатньо і маються відповідні сумніви відносно неї. Тому важливими як теоретичними так і практичними є завдання доведення криптографічної стійкості крипто перетворень в фактор – кільці. Серед атак, що вимагають відповідного аналізу відносно їх вразливості, є атаки, що засновані на аналізі часу розшифрування певного обсягу зашифрованої інформації [1, 3].

Метою статі є оцінка можливостей та визначення умов здійснення атаки, що заснована на визначенні часу розшифрування, в тому числі для різних значень загальних параметрів, а також обґрунтування та розробка рекомендацій відносно протидії такій атаці.

Загальна інформація про атаку

В NTRU [1] в процесі зашифрування та розшифрування використовуються дві геш-функції. Позначимо їх як G та H. На практиці зазвичай використовується SHA-1 або SHA-256 (в залежності від рівня стійкості). Ця атака заснована на тому факті, що кількість викликів SHA у якості складової геш-функції G залежить від вхідних даних. Таким чином, вимірюючи час розшифрування, криптоаналітик може отримати інформацію про дані, що були передані на вхід G, що, в свою чергу, розкриває інформацію про особистий ключ f.

Введемо наступні позначення:

$B_N = \{\textit{бінарні поліноми}\}$;

$B_N(d) = \{\textit{бінарні поліноми з d одиничними коефіцієнтами}\}$

На практиці при шифруванні за алгоритмом NTRUEncrypt вихідне повідомлення перетворюється у поліном спеціальним чином [1]. Процес шифрування із урахуванням такого перетворення виглядає так:

1) $M \in B_N$ – вихідне повідомлення після процедури доповнення (padding);

2) $r = G(M) \in B_N(d_r)$ – сеансовий ключ (зазвичай до повідомлення додають ще так звану «сіль» – рядок випадкових бітів, але для даної атаки обрана така схема генерації r);

3) $m' = M \oplus H(r * h \bmod q)$ – замасковане відображення M у якості елемента фактор – кільця.

Далі обчислюється шифротекст за вже відомим співвідношенням:

$$e = (r * h + m') \bmod q. \quad (1)$$

Алгоритм розшифрування в першу чергу відновлює відображення повідомлення *m'* та вихідне повідомлення *M* та використовує їх для відновлення *r* та перевірки, що (*m', e*) – дійсна пара для алгоритму NTRUEncrypt.

1) $m' = ((f * e \bmod q) \bmod p) * f_p^{-1} \bmod p$ – відновлення образу-кандидата *m'*;

2) $M = m' \oplus H(e - m' \bmod q)$ – розмаскування *m'* та отримання *M*;

3) $r = G(M)$ – відновлення сеансового ключа *r*, що використовувався при шифруванні.

Тепер можна перевірити, чи дорівнює *е значенню* виразу $r * h + m' \bmod q$. Якщо це не так, вважається, що виникла помилка розшифрування, і розшифроване повідомлення відкидається.

Основою цієї атаки є спосіб, у який *G* використовує SHA для отримання *r* з *M*. Зауважимо, що на етапі розшифрування *e* та *m'* повністю визначають *M*, і, таким чином, час на обрахунок $r = G(M)$. Також вимагається, щоб сеансовий ключ *r* був бінарним поліномом, що містить точно d_r одиниць. Процес обчислення *r* з *M,* описаний у [2], може використовувати різну кількість ітерацій алгоритму SHA для різних значень *M*.

Те ж саме твердження справедливе і для процедури розшифрування: необхідна кількість викликів геш-функції SHA для відновлення ключа сеансу із пари образ повідомлення/криптограма може різнитись в залежності від значень пари (*m', e*).

Часовий слід шифротексту

Кожен виклик геш-функції потребує деякого часу, тож зловмисник може визначити, скільки саме викликів було здійснено при спробі розшифрувати криптограму *e* (можливо, фіктивну).

На практиці це буде деяке число *K* таке, що кількість викликів геш-функції для обчислення *r* з (*m', e*) звичайно дорівнюватиме або *K*,або *K*+1. Для кожної пари (*m', e*) незалежно від того, є вона дійсною парою чи ні, позначимо вихід процедури розшифрування як *r*(*m', e*),

$$r(m', e) = G(m' + H(e - m' \bmod q)) \bmod 2, \quad (2)$$

та введемо бінарну функцію $\beta(m', e) \in \{0, 1\}$, що приймає свої значення за наступним правилом:

$$\beta(m', e) = \begin{cases} 0, \text{ якщо потрібно} \le K \text{ викликів для обчислення } r(m', e), \\ 1, \text{ якщо потрібно} > K \text{ викликів для обчислення } r(m', e) \end{cases} \quad (3)$$

Відмітимо, що для відомої пари (m', e) обчислення $r(m', e)$ та, відповідно $\beta(m', e)$ не потребує ніяких секретних знань.

Тому для даної пари (m', e) далі будемо здійснювати обернення (ротації) виду $(X^i m', X^i e)$ *для* $i = 0, 1, \ldots$. Визначимо також часовий слід пари (m', e) як бінарний вектор

$$T(m', e) = (\beta(m', e), \beta(Xm', Xe), \beta(X^2 m', X^2 e), \ldots, \beta(X^{N-1} m', X^{N-1} e)) \in \{0, 1\}^N. \quad (4)$$

Часовий слід показує, скільки викликів потрібно для кожного обернення пари (m', e).

Нехай *P* – це ймовірність того, що випадково обрана (m', e) вимагає не більше, ніж *K* викликів, а 1 – *P* – ймовірність того, що вимагається не менше, ніж *K* + 1 викликів геш-функції. Якщо ні *P,* ні 1 – *P* не є малими значеннями, ймовірність того, що дві пари (m'_1, e_1) та (m'_2, e_2) матимуть однакові часові сліди, є доволі малою. Знайдемо вираз для цієї ймовірності.

Ймовірність того, що співпадуть перші координати двох випадкових часових слідів, дорівнює:

$$prob\,(both\,0) + prob\,(both\,1) = P^2 + (1-P)^2 = 1 - 2P + 2P^2. \quad (5)$$

Тоді ймовірність того, що два часові сліди співпадають (тобто співпадають всі їхні координати), обчислюється так:

$$prob\,(T_1 = T_2) = (1 - 2P + 2P^2)^N \quad (6)$$

Таким чином, для будь-яких заданих $e \in \varepsilon$ та $m' \in M$ ймовірність того, що існує деяке інше відображення повідомлення $m'' \in M$ із $T(e, m'') = T(e, m')$ приблизно дорівнює

$$\#M \cdot (1 - 2P + 2P^2)^N. \quad (7)$$

Опис атаки

Розглянемо один із можливих варіантів здійснення атаки. Нехай спочатку зловмисник обирає набір (можливо, підроблених) ε криптограм (де ε – сукупність поліномів за модулем q). Він також обирає набір відображень повідомлень M (сукупність бінарних поліномів) такий, що містить у собі більшість поліномів з набору

$$\{((f * e \bmod q) \bmod 2) * (f_p^{-1} \bmod 2) : e \in \varepsilon\}. \quad (8)$$

Зауважимо, що саме цей набір відображень повідомлень абонент повинен отримати після розшифрування шифротекстів із набору ε. Тобто вважається, що ймовірність

$$p_{M,\varepsilon} = prob(((f * e \bmod q) \bmod 2) * (f_p^{-1} \bmod 2) \in M)$$

є не дуже низькою.

Перед початком активної фази атаки зловмисник створює таблицю, яка міститиме часові сліди кожної пари з $M \times \varepsilon$. Інакше кажучи, він створює список бінарних векторів $\{T(m', e) : m' \in M, e \in \varepsilon\}$. Ці перед обчислення, необхідні для проведення атаки, потребують $O(\#M \cdot \#\varepsilon)$ часу та пам'яті.

Щоб розпочати атаку, зловмисник випадково обирає деяке $e \in \varepsilon$, відправляє його адресату (який володіє особистим ключем) та відслідковує, який час затрачує адресат на розшифрування. Використання схеми доповнення NAEP або NTRU-FORST гарантує, що підробні шифро тексти (такі, що були отримані за рахунок використання іншого відкритого ключа) будуть відхилені. Але навіть в цьому випадку криптоаналітику байдуже, тому що він все одно має на меті отримати час, затрачений на розшифрування (а пройшло воно успішно чи повідомлення було відкинуте, для нього немає значення).

Інформація про час розшифрування дозволяє йому визначити, скільки викликів гешфункції необхідно для обчислення ключа сеансу r із криптограми e та відображення повідомлення $m'(e) = ((f * e \bmod q) \bmod 2) * (f^{-1} \bmod 2)$, і зловмисник знаходить значення $\beta(m'(e), e)$. Звичайно, він не знає значення $m'(e)$.

Подібним чином крипто аналітик відсилає кожен з поліномів $e, Xe, X^2e, \ldots, X^{N-1}e$ адресату і отримує значення $\beta(m'(X^i e), X^i e)$ *для* $i = 0, 1, N-1$. Помітимо, що

$$m'(X^i e) = ((f * X^i e \bmod q) \bmod 2) * (f^{-1} \bmod 2) = X^i ((f * e \bmod q) \bmod 2) * (f^{-1} \bmod 2) = X^i m'(e). \quad (9)$$

Отже, аналітик, маючи всі N значень функції $\beta(m'(X^i e), X^i e)$, дізнається часовий слід $T(m'(e), e)$ для пари $(m'(e), e)$.

Зловмисник тепер виконує пошук у таблиці обчислень та зі значною ймовірністю знаходить невелику кількість пар $(m'(e), e)$, що мають саме такий часовий слід. Тобто він тепер знає поліном e та має невеликий перелік варіантів відображення повідомлення m', які задовольняють рівнянню

$$m' * f = (f * e \bmod q) \bmod 2. \qquad (10)$$

Рівняння (10) містить значну кількість інформації про особистий ключ f, хоча можливість успішного використання такої інформації залежить від спеціальної форми e. Наприклад, якщо елементи множини ε мають малу кількість ненульових коефіцієнтів, то рівняння (10) може надати інформацію стосовно відстаней між ненульовими коефіцієнтами в f.

Оцінка захищеності NTRU від даної атаки

Можливість реалізації даної атаки на практиці у першу чергу залежить від ймовірності того, що різні входи вимагатимуть різну, а не однакову, кількість викликів геш-функції SHA при обчисленні r на передостанньому кроці розшифрування.

До даної атаки потенційно вразливі всі існуючі реалізації алгоритму НШ NTRU. Опишемо детальніше, як саме використовується геш – функція сімейства SHA при генерації сеансового ключа r згідно із [2]. Зауважимо, що саме ця версія із деякими змінами та модифікаціями була покладена в основу стандартів IEEE P 1363.1 та ANSI X9.98 (основні відмінності – використання двійкових поліномів замість тернарних).

Алгоритм атаки.

1. Фіксується таке значення c, яке задовольняє умові $2^c > N$. Це значення задане в [2] для кожного із наборів параметрів. Також обчислюються змінні $b = \lceil c/8 \rceil$ та $n = \lfloor 2^c / N \rfloor$. b – це значення параметру c у байтах, nN – найменше кратне N, що є меншим за 2^c.
2. Визивається зазначена у наборі параметрів версія SHA, її вихід розбивається на сегменти довжиною b байт кожен. У кожному сегменті враховуються тільки c молодших бітів (інші $8b - c$ бітів вважаються рівними нулю), які перетворюються на цілі числа (кодування – little endian) $i_1, i_2, ..., i_t$. Тут t – таке ціле число, що tb дорівнює довжині виходу відповідної версії SHA.
3. За наступним алгоритмом згенерувати r:
   (1) jList = {}
   (2) Викликати геш – функцію SHA для отримання $i_1, i_2, ..., i_t$
   (3) Loop $\alpha = 1, 2, \ldots, t$
   (4) If $i_\alpha < n$ *ma* $(i_\alpha \bmod N) \notin$ jList

   then додати $i_\alpha \bmod N$ в jList
   (5) If jList вже містить d_r елементів, вийти.
   (6) Кінець циклу α.
   (7) Перейти на крок (2) для отримання більшої кількості значень i.

З опису алгоритму стає зрозуміло, чому кількість викликів геш-функції може варіюватися в залежності від вхідних значень. Якщо вважати числа $i_1, i_2, ..., i_t$ випадковою послідовністю в діапазоні $0 \le i < 2^c$, то можна обчислити ймовірність $P_{C,N,n}(L, d)$ того, що набір із L випадково обраних чисел $i \in [0, C)$ містить d елементів, які одночасно задовольняють вимо-

гам пункту (4) кроку 3 вищенаведеного алгоритму, за наступним рекурсивним співвідношенням:

$$P_{C,N,n}(L,d) = P_{C,N,n}(L-1,d-1)\left(\frac{n(N-\alpha+1)}{C}\right) + P_{C,N,n}(L-1,d)\left(1-\frac{n(N-d)}{C}\right), \quad (11)$$

$$P_{C,N,n}(L,d) = 0 \text{ } якщо \text{ } L < d, \quad (12)$$

$$P_{C,N,n}(L,0) = \left(1-\frac{nN}{C}\right)^{L}. \quad (13)$$

Ймовірність того, що буде достатньо не більше, ніж *s* викликів SHA (тобто що *st* випадково обраних цілих чисел в діапазоні $[0, 2^C)$ включають як мінімум d_r значень з діапазону $[0, n)$, які відрізняються за модулем *N*) описується виразом

$$prob(s \text{ } викликів SHA \text{ } достатньо) = \sum_{d_r \le d \le st} P_{2^C,N,n}(st, d). \quad (14)$$

Відмітимо, що формули цих ймовірностей є універсальними та можуть бути застосовані до всіх сучасних реалізацій NTRUEncrypt. Нами була виконана практична реалізація обчислень ймовірності того, що знадобиться не більше, ніж *s* викликів геш-функції згідно із формулами (11) – (14) для кожного із наборів параметрів згідно із стандартом ANSI X9.98. Результати обчислень представлені в таблиці. Також згідно із співвідношенням (6) були обраховані ймовірності того, що дві різні пари (m'_1, e_1) та (m'_2, e_2) матимуть однакові часові сліди (див. стовбці $P_{\text{nonunique}}$ таблиці).

Рівень стійкості, біт	*N*	d_r	Довжина виходу SHA, біти	*c*	*minCallsR*	*s* : P(достатньо *s* викликів SHA)			*s* : P($T_1 = T_2$)		
112	401	113	160	11	32	13:0,1808	14:0,8339	15:0,995	$13{:}2^{-203}$	$14{:}2^{-187}$	$15{:}2^{-5,73}$
	541	49	160	12	15	5:0,0153	6:0,9559	7:0,9999	$5{:}2^{-23,8}$	$6{:}2^{-68,7}$	7:0,999
	659	38	160	11	11	-	4:0,5449	5:0,9999	-	$4{:}2^{-651}$	5:0,988
128	449	134	160	9	31	18:0,4789	20:0,9868	21:0,9994	$18{:}2^{-448}$	$20{:}2^{-17,14}$	21:0,5808
	613	55	160	11	16	6:0,1281	7:0,9649	8:0,9999	$6{:}2^{-223,7}$	$7{:}2^{-62,1}$	8:0,9649
	761	42	160	12	13	-	5:0,9523	6:0,9999	-	$5{:}2^{-104,64}$	6:0,9968
192	677	157	256	11	27	11:0,2626	12:0,9868	13:0,9999	$11{:}2^{-478}$	$12{:}2^{-25,69}$	13:0,9967
	887	81	256	10	13	-	6:0,3886	7:0,9977	-	$6{:}2^{-825}$	$7{:}2^{-5,86}$
	1087	63	256	13	13	-	4:0,0109	5:0,9994	-	$4{:}2^{-34,3}$	$5{:}2^{-2}$
256	1087	120	256	13	25	8:0,0194	9:0,9518	10:0,9999	$8{:}2^{-60,8}$	$9{:}2^{-150,98}$	10:0,9918
	1171	106	256	12	20	-	8:0,4442	9:0,9947	-	8:0	$9{:}2^{-17,89}$
	1499	79	256	13	17	5:0,00105	6:0,9845	7:0,9999	$5{:}2^{-4,56}$	$6{:}2^{-66,82}$	7:0,9999

Зловмисник зацікавлений в якомога меншому значенні ймовірності спів падіння часових слідів P($T_1 = T_2$). Тоді з дуже високою ймовірністю (а саме 1 – P($T_1 = T_2$)) отриманий слід є

унікальним, тобто була віднайдена дійсна пара $(m'(e_i), e_i)$. При цьому необхідність перевірки правильності e_i відпадає.

Аналізуючи вираз (6), можна помітити, що найменшого значення ймовірність $P(T_1 = T_2)$ досягає тільки тоді, коли ймовірність того, що буде достатньо не більше, ніж *s* викликів функції хешування, буде дорівнювати 0,5.

Висновки

1. Отримані експериментальні результати показали, що найкращі шанси на відновлення особистого ключа криптоаналітик мав би у випадку використання наборів параметрів ees659ep1 (рівень стійкості 112 бітів), ees677ep1, ees887ep1 (192 біти) та ees1171ep1 (256 бітів), але лише за умови, коли *s* дорівнює 4, 11, 6 та 8 відповідно.

2. Для запобігання атаці необхідно досягти постійного часу розшифрування за рахунок встановлення такої кількості викликів функції хешування *s* (в термінах стандарту ANSI X9.98 – параметр *minCallsR*), якої гарантовано (з ймовірністю, практично рівною одиниці) було достатньо для переважної більшості вхідних даних. Наприклад, для набору параметрів ees659ep1 при *minCallsR* = 4 атака завершиться успіхом з дуже високою ймовірністю, а вже при *minCallsR* = 5 вона стає практично нездійсненною (тому що ймовірність колізії часових слідів наближається до одиниці).

3. Розробники ANSI X9.98 вважають, що практично постійний час розшифрування може бути отриманий шляхом вибору такого параметру *oLenMin*, при якому ймовірність того, що знадобиться більше, ніж *oLenMin* октетів виходу, менша, ніж 2^{-k}, де *k* – рівень стійкості, *oLenMin = minCallsR * hLen, hLen* – довжина в октетах виходу геш-функції. Параметр *minCallsR* повинен бути таким найменшим цілим числом, що ймовірність того, що знадобиться більш, ніж *oLenMin* октетів, буде менша за 2^{-k}.

4. Розробники навмисно завищили значення параметру *minCallsR* для всіх наборів параметрів більше, ніж удвічі, щоб гарантовано унеможливити практичну реалізацію timing-атаки на поточну версію NTRUEncrypt та створити деякий запас стійкості на майбутнє. Для наведеного вище прикладу (набір параметрів ees659ep1) при теоретично достатньому для забезпечення стійкості значенні *minCallsR* = 5, 6 стандарт рекомендує використовувати при виробленні сеансового ключа *r* 11 ітерацій гешування за алгоритмом SHA-1.

Отже, реалізація алгоритму НШ NTRU із використанням наборів параметрів, описаних у стандартах ANSI X9.98, IEEE P 1363.1 є стійкою до даної атаки, тому вона може бути рекомендованою до застосування..

Список літератури: 1. *American* National Standard for Financial Services ANSI X9.98-2010 Lattice Based Polynomial Public Key Establishment Algorithm for the Financial Services Industry. – 2010. 3. *Jens Hermans, Frederik Vercauteren, Bart Preneel.* Speed records for NTRU. ежим доступа: www/ URL: https://www.securityinnovation.com/uploads/ntru_speed_benchmark_research.pdf. – 2010. 2. *Горбенко І.Д., Горбенко Ю.І.* Прикладна криптологія. – Харків : ХНУРЕ, Форт, 2012. – 868 с. 3. Consortium for Efficient Embedded Security, Efficient Embedded Security Standard (EESS) #1 version 3, 2005. – Режим доступу: www/ URL: http://grouper.ieee.org/groups/1363/lattPK/submissions/EESS1v2.pdf.

Харківський національний
університет радіоелектроніки

Надійшла до редколегії 11.01.2014