МЕТОДЫ И СРЕДСТВА АСИММЕТРИЧНЫХ КРИПТОГРАФИЧЕСКИХ ПРЕОБРАЗОВАНИЙ

УДК 004.056.55

ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ МОЖЛИВОСТІ ВИКОРИСТАННЯ ПАРАМЕТРІВ NTRUPRIME ДЛЯ НЕСИМЕТРИЧНОГО ШИФРУВАННІ ЗГІДНО З СТАНДАРТОМ ANSI X9.98 - 2010

І.Д. ГОРБЕНКО, О.Г. КАЧКО, Г.С. НАУМЕНКО

Сучасні атаки використовують спеціальний формат параметрів, які застосовуються в алгоритмі NTRU (ANSI X9.98). Сьогодні знайдені постквантові параметри, для яких ці атаки не можуть бути використані (NTRUPrime). В роботі досліджується можливість та доцільність використання нових параметрів в алгоритмі ANSI X9.98.

Ключові слова: ANSI X9.98-2010, NTRUPrime, NTRU parameters, швидкодія.

ВСТУП

У зв'язку з проблемами, пов'язаними з можливостями використання сучасних несиметричних алгоритмів в умовах наявності квантових комп'ютерів, найбільший інтерес становлять криптографічні алгоритми, криптостійкість яких базується на решітках [1]. Саме до цього класу відносяться NTRU подібні алгоритми.

Для стандарту несиметричного шифрування ANSI X9.98 - 2010[2], який базується на NTRU алгоритмі, визначені різні атаки, пов'язані з параметрами, а саме з модулями, які використовуються. В роботі [3] запропоновано нові параметри, які запобігають цим атакам, а саме, інший поліном й інші модулі для використання (NTRU Prime, подальшому NTRUPrime). Ми дослідили можливість використання цих параметрів для шифрування з боку найбільш важливої характеристики, яка відрізняє методи NTRU від решти несиметричних алгоритмів, а саме швидкісні характеристики. Для цього використовується фактично стандарт ANSI X9.98 – 2010, але з математикою NTRUPrime. Це наступна робота з циклу робіт, присвячених цьому класу алгоритмів [4], [5]. Подальший аналіз з боку криптостійкості, в тому числі, в умовах використання квантових комп'ютерів буде проведено в подальших роботах циклу.

1. ВІДМІННОСТІ КЛАСИЧНОГО NTRU-МЕТОДУ ВІД NTRUPRIME) ТА ЇХ АНАЛІЗ З БОКУ ОБЧИСЛЮВАЛЬНОЇ СКЛАДНОСТІ

1. Замість кільця $(Z/qZ)[X]\left(X^N-1\right)$ використовується поле $(Z/qZ)[X]\left(X^N-X-1\right)$. Як N запропоновано значення 739.

2. Замість значення параметру q = 2048, який використовується для обчислення модуля коефіцієнтів поліному, застосовується значення 9829. Значення параметрів N і q визначили назву методу, яке дали йому автори [3], а саме Streamlined NTRU Prime 9829739.

3. Кількість 1 та -1 в особистому ключі (значення d_f), та в поліномі для маскування (Blinding Polinomial) (значення d_r) визначається залежно від N та потрібної криптостійкості, але для усіх параметрів $d_f = d_r$. Для NTRUPrime $d_f = d_r = 202$.

4. Повний набір параметрів для класичного NTRU забезпечують криптостійкість від 112 до 256. Обраний набір параметрів (N=739, q = 2048), як стверджують автори [3], забезпечує криптостійкість не менше, ніж 128.

Використання кільця в класичному методі в ході обчислення добутку (найважчої операції при шифруванні - розшифруванні) просто загортали поліном, тобто індекс коефіцієнта брався за модулем N, використання поля для NTRUPrime потребує обчислення добутку за модулем $X^N - X - 1$.

Значення q визначає модуль, за яким виконується обчислення усіх коефіцієнтів поліному. Обчислення за модулем q = 2048 не потребує використовувати операцію ділення, для q = 9829 операція ділення потрібна. Крім того, значення q визначає довжину відкритого ключа. Відкритий ключ – це поліном порядку N, граничні значення коефіцієнтів якого визначаються значенням q. Для q = 2048 довжина відкритого ключа 11N бітів, для q = 9829 вона складає 14N. Для N = 739 отримаємо 10346 бітів або 1294 байта. Автори [3] пропонують для завдання відкритого ключа в упакованому форматі використовувати комбіновану систему числення (2 – 10), тоді відкритий ключ потребує 1232 байти. Економія менше, ніж 5% адресного простору для завдання відкритого ключа призведе до витрат часу на розгортання цього ключа, тим більше, що це відкритий ключ стороннього користувача, і таке

перетворення необхідно виконувати для кожного нового користувача.

Попередній аналіз основних параметрів показує, що обчислювальна складність для NTRUPrime очікується більше, ніж для класичного NTRU. В роботі виконується саме цей аналіз.

2. ОБЧИСЛЕННЯ ДОДАТКОВИХ ПАРАМЕТРІВ ДЛЯ NTRUPRIME

У зв'язку з необхідністю реалізації класичного NTRU для нових параметрів необхідно обчислити додаткові параметри, які використовуються для реалізації класичного NTRU.

Ці параметри, та формули для їх обчислення наведені в табл.1

Таблиця 1

Параметри та формули для обчислення

Позначення	Формула для обчислення	NTRU Prime
d_g	$d_g = N/3$	246
bLen	bLen = S	192
maxMsgLen Bytes	$\frac{\frac{3(N-1)}{2} - bLen}{8} - 1$	113
Llen	$\lceil \log_{256} \max MsgLenByte$	1
dm0	dm0 = d_f	202
HashLen	Залежить від криптостійкості	SHA 256
C	$\lceil \log_2 N \rceil$	11
minCallsR	$\frac{4d_r c}{HashLen}$	35
minCallsMask	$\left\lceil \frac{16N}{5 * HashLen} \right\rceil + 1$	11

Автори [3] декларують для обраних параметрів криптостійкість більше, ніж 128, тому значення криптостійкості обране 192, а відповідна геш-функція SHA256.

3 БАЗОВІ ОПЕРАЦІЇ

Як базові операції використовуються операції перетворення між байтовими рядками та поліномами, операції множення поліномів, та обчислення інверсії. Алгоритми для перетворень визначені в [2].

3.1 Операції множення поліномів

c = a01_1 * b; для шифрування;

c = (3 * a01_1 + 1) * b - для розшифрування та генерації відкритого ключа,

де a01_1 – поліном в $\frac{\left(\frac{Z}{3Z}\right)[X]}{\left(X^N - X - 1\right)}$;

b, c – поліноми в $\frac{\left(\frac{Z}{qZ}\right)[X]}{\left(X^N - X - 1\right)}$.

Для забезпечення найбільшої ефективності для кожного з типів операції множення використовується свій алгоритм.

В роботі [3] для підвищення швидкодії множення поліномів пропонується використання методів множення Карацуби, Тоома та їх комбінації. Ми перевірили ефективність цих методів для поліномів з N = =739 і не досягли підвищення швидкодії. Це пов'язано з великою кількістю нульових елементів у масивах (не менше, ніж N/3), а також фактичною відсутністю операцій множення, замість них використовується операція додавання.

Наш варіант алгоритмів для множення поліномів наведено на рис. 1, 2.

Алгоритм 1. Множення поліномів для шифрування
Вхід. *N, q, a* – поліном у полі $\frac{\left(\frac{Z}{3Z}\right)[X]}{\left(X^N - X - 1\right)}$,
b – поліном у полі $\frac{\left(\frac{Z}{qZ}\right)[X]}{\left(X^N - X - 1\right)}$
Вихід. c – поліном у полі $\frac{\left(\frac{Z}{qZ}\right)[X]}{\left(X^N - X - 1\right)}$
1. Set c := 0
2. Set i := 0
3. While i ≠N
3.1. if ($a_i \neq 0$)
3.1.1. Set pc := Address(c_i)
3.1.2 if ($a_i = 1$)
3.1.2.1 Set j := 0;
3.1.2.2.While j ≠N
3.1.2.2.1 Set pc_j:=pc_j+b_j
3.1.2.2.2 Set j:=j+1
3.1.3 else
3.1.3.1 Set j := 0;
3.1.3.2.While j ≠N
3.1.3.2.1 Set pc_j:=pc_j+b_j
3.1.3.2.2 Set j:=j+1
4.Set c:=c mod q
5 Set c:=c mod $(X^N - X - 1)$
6. Return c

Рис.1 Множення поліномів для шифрування

Алгоритм 2. Множення поліномів для розшифрування та генерації відкритого ключа
Вхід. *N, q, f* – поліном, *f = 3a+1*, *a* поліном у полі $\frac{\left(\frac{Z}{3Z}\right)[X]}{\left(X^N - X - 1\right)}$, *b* – поліном у полі $\frac{\left(\frac{Z}{qZ}\right)[X]}{\left(X^N - X - 1\right)}$
Вихід. *c* – поліном у полі $\frac{\left(\frac{Z}{qZ}\right)[X]}{\left(X^N - X - 1\right)}$
1.Set p3 := 3b
2. Set p_3 := -3b
3.Set c:=f_0b
4.Set i := 1
5.While i ≠N
5.1 if ($a_i \neq 0$)
5.1.1 Set pc := Address(c_i)
5.1.2 if ($a_i = 3$) p = p3 else p = p_3
5.1.3. Set j := 0;
5.1.4 While (j ≠N)
5.1.4.1. Set pc_k:=pc_k+p_j
5.1.4.2. Set j:=j+1
6. Set c:=c mod q
7. Set c:=c mod $(X^N - X - 1)$
8. Return c

Рис.2 Множення поліномів для розшифрування та генерації відкритого ключа

Останнє значення використовується як початкове значення c.

3.2 Інверсія поліномів

Операція інверсії використовується для обчислення відкритого ключа, який визначається за формулою: $h = f^{-1}gp$, де F – поліном з коефіцієнтами -1, 0, 1, кількість 1, -1 однакова і дорівнює d_f, $f = 3F + 1$, g – поліном з коефіцієнтами -1, 0, 1, кількість -1 дорівнює $d_g = \frac{N}{3}$ кількість 1 дорівнює $d_g + 1$, p = 3, а також для перевірки наявності інверсії відносно p = 3.

Як визначено в роботі [6] серед розглянутих алгоритмів обчислення інверсії для класичного NTRU найбільш ефективним є алгоритм almost inverse, але, на жаль, цей алгоритм не можна використовувати для обчислення інверсії відносно поліному $X^P - X - 1$ для q, яке не є ступенем 2. В цьому разі необхідно використовувати розширений алгоритм Евкліда, який оптимізовано за рахунок заміни копіювань поліномів обміном їх адрес.

Алгоритм обчислення інверсії наведено на рис. 3.

Алгоритм 3. Обчислення f^{-1} такого, що $f^{-1} * f = f * f^{-1} = 1$ в полі $\frac{\left(\frac{Z}{qZ}\right)[X]}{\left(X^N - X - 1\right)}$
Вхід. *N, q, f* – поліном, *f = 3F+1, F* – поліном у полі $\frac{\left(\frac{Z}{3Z}\right)[X]}{\left(X^N - X - 1\right)}$
Вихід. f^{-1} або Error
1. Set c:=0, y := X^N
2. Set n3 :=f, n4 := y
3. Set n5 := 1, n6 := 0;
4. Set px := Address (n4), py := Address (n3), pa2 := Address (n6), pa1 := Address (n5)
5. Set RetValue := 1
6. Set n2 := *px / *py; *px := *px – *py * n2;
7.if (px == 0)
7.1 if (c is odd)
7.1.1 RetValue :=– RetValue
7.2. Set gcd := *py;
7.3. goto step 16;
8. Set c:=c+1
9.Set RetValue:= *pa1* n2 + *pa2;
10 if (deg (RetValue) >= deg (y))
10.1 Set RetValue:= RetValue – y;
11. swap (pa1, pa2);
12. Set *pa1 = RetValue;
13. swap (px, py)
14. goto Step 8
15. if (deg (gcd) ≠ 0)
Return Error
16. Set gcd_1 = gcd^{-1} mod q
17. Set RetValue = gcd_1 * RetValue
18. Return RetValue

Рис. 3. Алгоритм інверсії

Для ділення поліномів використовується алгоритм 11[2]. Для множення поліному на число виконується множення кожного коефіцієнта з урахуванням модуля q. Операція обрання значення за адресою (разова адресація) позначена символом *. Операція обміну адресами позначена swap. Алгоритм повертає помилку Error в разі відсутності зворотного елементу,

тобто відповідне Діафантове рівняння не має цілих розв'язань.

4 АЛГОРИТМИ ШИФРУВАННЯ ТА РОЗШИФРУВАННЯ

Визначені в [2] відповідно до Algorithm 23 та Algorithm 24.

Далі розглянуто основні етапи цих алгоритмів та їхню оптимізацію.

4.1 Алгоритм шифрування

Вхід. Рядок байтів для шифрування m з довжиною l, та відкритий ключ h.

Вихід. Шифротекст – рядок байтів або Error, якщо довжина l перевищує параметр *maxMsgLenBytes.*

Алгоритм шифрування складається з таких кроків:

Крок 1. Формування рядка M форматом:

b || *octL* || *m* || *p0*,

де *b* – випадковий рядок довжиною *bLen*;

octL, m – довжина повідомлення (*octL*) та саме повідомлення (*m*). Для завдання довжини повідомлення використовується параметр *Llen;*

p0 – кількість нульових байтів, які доповнюють повідомлення m до максимальної довжини. Обчислюється за формулою *maxMsgLenBytes* + 1 – *l.*

Для оптимізації цього кроку заповнення рядка *p0* нулями не обов'язково, достатньо додати один 0.

Крок 2. Перетворення отриманого рядка M у поліном *MTrin* з коефіцієнтами {-1, 0, 1}. Для оптимізації цього кроку замість перетворення трибітних послідовностей перетворюємо шестибітні, в результаті отримуємо 4 байти, які відповідають 4 коефіцієнтам *MTrin*. Для заміни шестибітних даних використовується масив констант розміром 64 елемента, який має вигляд:

```
0x00000000,          //{0, 0, 0, 0},
0x01000000,          //{0, 0, 0, 1},
0xFF000000,          //{0, 0, 0, -1},
…
```

Якщо в результаті обробки рядка M отримуємо не усі коефіцієнти, решта коефіцієнтів заповнюється нулями.

Крок 3. Формування рядка *sData* з форматом:

OID || *m* || *b* || *hTrunc,*

де *OID* – ідентифікатор, береться з параметрів;

m – повідомлення для шифрування;

b – випадковий рядок довжиною bLen;

hTrunc – частина упакованого відкритого ключа довжиною bLen.

Для оптимізації цього кроку для рядків *sData* та *M* виділяється загальна пам'ять, що дозволяє не копіювати повідомлення для формування рядка *sData*. Відкритий ключ записується в контейнер при його встановленні не тільки у форматі поліному, а і в упакованому форматі, що дозволяє не виконувати його перетворення в бітовий рядок у ході формування рядка *sData.*

Крок 4. Формування полінома *r* для осліплення (blinding polynomial). Алгоритм формування (Algorithm 18 [2]) використовує IGF алгоритм для формування послідовності, яка далі застосовується для визначення номерів (індексів) коефіцієнтів поліному, які приймають значення 1, а потім -1. В [2] визначено два алгоритми формування IGF: IGF-2 (Алгоритм 20) та IGF-RBG (Алгоритм 21). Згідно з алгоритмом 20 спочатку обчислюється геш для рядка *sData*, а потім *minCallsR* викликається функція обчислення геша для отриманого геша (постійна частина) та номера виклику функції обчислення геша. В результаті щоразу отримуємо рядок, довжина якого не перевищує розмір блоку, для обчислення геша. Для оптимізації цього алгоритму один раз виконуються постійні операції в ході обчислення геша для блоку і *minCallsR* разів операції, які залежать від номера.

Крок 5. Найчастіше затратна операція множення R = r * h. Тут використовується Алгоритм 1 функції множення.

Крок 6. Обчислення *R mod 4* та їх упакування (4 коефіцієнта на байт), отримаємо oR4.

Крок 7. MGF перетворення для рядка oR4 з урахуванням параметра *minCallsMask* і отримання поліному *mask.* Формується бітова послідовність аналогічно кроку 4. Але як вхідна послідовність використовується oR4, а як кількість викликів – *minCallsMask*. Отриманий байтів масив використовується для формування поліному. З кожного байта, значення якого не перевищує 3^5, формується 5 коефіцієнтів поліному. Для оптимізації обчислення коефіцієнтів використовується масив констант довжиною 273 рядка.

Приклади констант:

```
{ 0, 0, 0, 0, 0 },
{ 1, 0, 0, 0, 0 },
{ -1, 0, 0, 0, 0 },
…
```

Крок 8. Обчислення шифротексту:

r1 = (r + Mtrin) %3

if (r1.count (1) < df or r1.count (-1) < df or r1.count (0) < df) goto 1

rh := R + r1

Для оптимізації цього кроку для одного коефіцієнта виконуються усі необхідні операції, і ведеться підрахунок значень коефіцієнтів

Крок 9. Перетворення поліному *rh* в рядок байтів *em.*

4.2 Алгоритм розшифрування

Вхід. Шифротекст (рядок байтів, *em*) та його довжина (*len*)

Вихід. Відкритий текст (рядок байтів, *m*) та його довжина (*l*) або Error

Складається з наступних кроків.

Крок 1. Перетворення байтового рядка *em* в поліном e.

Крок 2. Set *cm':=f*e.* Для множення використовується алгоритм 2.

Крок 3. Set d := cm' mod 3

Крок 4. if (d.count (-1) < df or d.count (-1) < df of d.count (0) < df)

Return Error;

Крок 5. Set *coR4 := ConvertToBytes((e - cm') mod 4)*

Крок 6. Генерація поліному для маскування. MGF перетворення для рядка coR4 з урахуванням параметра *minCallsMask* і отримання поліному *mask* (див. крок 7 алгоритму шифрування) та генерація *cMtrin*.

Крок 7. Перетворення *cMtrin* у бітовий рядок.

Для оптимізації використовується індекса таблиця:0000	0001	00FF	0100	0101	01FF	FF00	FF01
0	3	6	1	4	7	2	5

Верхній рядок таблиці задано в шістнадцятковій системі.

Якщо серед вхідних елементів є елемент 0xFFFF, то return Error.

Крок 8. Визначення відкритого тексту та його довжини.

Під час виконання цього пункту необхідно перевірити довжину відкритого повідомлення (вона не може перевищувати максимальну можливу, та наявність необхідної кількості нульових елементі в кінці повідомлення, якщо його довжина менше, ніж *maxMsgLenBytes*).

5 ЕКСПЕРИМЕНТАЛЬНІ РЕЗУЛЬТАТИ

Усі експерименти виконувались на комп'ютері Intel (R) Core (TM) i5 -4400 CPU @3.10 GHz, Windows 7, 64 bit.

За допомогою профілювання визначені функції, які потребують найбільшого часу в ході шифрування та розшифрування. Це функції множення поліномів.

Визначався час виконання функцій множення для поліномів з коефіцієнтами (-1, 0, 1) для шифрування і коефіцієнтами (-3, 0, 3) за виключенням першого коефіцієнта для дешифрування, а також функцій шифрування та розшифрування.

Для порівняння наведені аналогічні результати для класичного NTRU та для NTRUPrime. З усіх параметрів для класичного NTRU обирались параметри, які задовольняють такі критерії:

— рівень криптостійкості S = 192;

— значення N найближче до 739;

— співвідношення кількості ненульових елементів до N найближче до 404/739. ≈ 0.55. Найближчі параметри, які задовольнять усі вимоги, N = 677, $d_f = 157$.

В табл. 2 наведені результати обчислювального експерименту.

Для оптимізації кроки 3 – 5 виконуються як один крок для кожного коефіцієнта поліному.

Для операцій множення час задано в мілісекундах. Для операцій шифрування та розшифрування задана швидкість у кілобітах за секунду.

Таблиця 2

Результати обчислювального експерименту

Алгоритм	Множення (ms)	Шифрування		Розшифрування	
		ms	kbit/s	ms	kbit/s
Класичний NTRU	0.04	0.066	12202	0.038	20859
NTRUPrime	0.06	0.098	9224.3	0.078	18200

ВИСНОВКИ

З появою квантових комп'ютерів, одним з найперспективніших асиметричних криптографічних методів, який задовольняє вимоги стійкості і швидкодії, стала NTRU криптосистема, яка базується на криптостійкості решіток [2].

Класичний NTRU [2] за час своєї експлуатації з 2010 року має недоліки, які можуть бути усунені завдяки використанню нових параметрів [3].

У роботі показано можливість застосування параметрів з [3] для алгоритму [2] та виконано реалізацію запропонованої комбінованої схеми.

Отримані результати наведені в табл.. 2.

Класичний метод має кращі швидкісні характеристики, ніж новий практично вдвічі, що очікувано. Дійсно, використання модулів, «незручних» з боку обчислювальної складності, призводить до такого результату. Але з урахуванням того, що NTRU алгоритм за швидкісними характеристиками обганяє існуючі несиметричні методи шифрування в десятки та сотні разів, таке уповільнення не є суттєвим. Отримані швидкісні характеристики є попередніми, в подальшому буде досліджене більш повне використання SIMD операцій та графічних процесорів для покращення цих характеристик.

У подальших роботах буде також розглянуто детально криптостійкість запропонованого методу та можливість використання інших параметрів з метою збільшення криптостійкості.

Література

[1] Report on Post-Quantum Cryptography. http://nvlpubs.nist.gov/nistpubs/ir/2016/NIST.IR.8105.pdf

[2] American National Standard for Financial Services ANSI X9.98 – 2010. Lattice-Based Polynomial Public Key Establishment Algorithm for the Financial Services Industry

[3] *Daniel J. Bernstein1,2, Chitchanok Chuengsatiansup , Tanja Lange1, and Christine van Vredendaal1*. NTRU Prime , https://ntruprime.cr.yp.to/ntruprime-20160511.pdf (visited 22.12.2016)

[4] https://github.com/NTRUOpenSourceProject/ntru-crypto (visited 22.12.2016)

[5] *I. Gorbenko, O. Kachko, K. Pogrebnyak*. Features OF parameterS calculation for NTRU algorithm. Прикладная радиоэлектроника, 2015. – Том 14, № 3. – С. 272-277.

[6] *Качко О.Г., Погребняк К.А., Макутоніна Л.В.* Аналіз, оцінки та пропозиції відносно методу генерації системних параметрів у NTRU-подібних асиметричних системах. Радіотехніка. – 2016. – Т.186. – С. 103 – 110.

[7] *Качко Е.Г., Балагура Д.С., Погребняк К.А., Горбенко Ю.І.* Исследование методов вычисления инверсии в алгоритме NTRU. Прикладная радиоэлектроника. – 2013. – Т.12, № 2. – С. 254 – 257.

Горбенко Іван Дмитрович, доктор технічних наук, професор, Харківський національний університет ім. В.Н.Каразіна, професор кафедри безпеки інформаційних систем і технологій.

Качко Олена Григорівна, кандидат технічних наук, професор кафедри ПІ ХНУРЕ. Наукові інтереси: криптографія, криптоаналіз, паралельні обчислення.

Науменко Гліб Сергійович, студент групи ПІ-13-1 факультету КН ХНУРЕ. Наукові інтереси: розподілені системи, криптографія.

УДК 004.056.55

Экспериментальное исследование возможности использования параметров NTRUPrime для несимметричного шифрования в соответствии со стандартом ANSI X9.98 - 2010 / И. Д. Горбенко, Е. Г. Качко, Г.С.Науменко. // Прикладная радиоэлектроника: науч.-техн. журнал. – 2016. – Том 15, № 3. – С. 135 – 140.

Современные атаки используют специальный формат параметров, которые применяются в алгоритме NTRU (ANSI X9.98). Сегодня найдены постквантовые параметры, для которых эти атаки не могут бать выполнены (NTRUPrime). В работе исследуется возможность и целесообразность использование нових параметров в алгоритме ANSI X9.98-2010.

Ключевые слова: ANSI X9.98, NTRUPrime, NTRU parameters, быстродействие.

Табл.: 02. Ил.: 03. Библиогр.: 07 назв.

UDC 004.056.55

Experimental study of the possibility of using NTRUPrime parameters for asymmetric encryption in accordance with ANSI X9.98 – 2010 standard / I.D. Gorbenko, O.G. Kachko G.S. Naumenko // Applied Radio Electronics: Sci. Journ. – 2016. – Vol. 15, № 3. – P. 135 – 140.

Modern attacks use a special format of the parameters which are used in NTRU algorithm (ANSI X9.98 - 2010). Postquantum parameters have been found for which the attacks can not be performed (NTRUPrime). The paper studies the possibility and advisability of using these parameters for ANSI X9.98-2010.

Keywords: ANSI X9.98-2010, NTRUPrime, NTRU parameters, speed.

Tab.: 02. Fig.: 03. Ref.: 07 items.