Część 8

Kryptologia (kryptografia) „post-kwantowa”
(Post-quantum cryptography,
quantum resistant cryptography)
Pierścienie i ciała wielomianów

„…however, that there is no justification for the leap from "quantum computers destroy RSA and DSA and ECDSA" to "quantum computers destroy cryptography." There are many important classes of cryptographic systems beyond RSA and DSA and ECDSA…"

Daniel J. Bernstein – „Introduction to post-quantum cryptography"

„Rodziny" algorytmów kryptograficznych, dla których „póki co" nie sformułowano kwantowych algorytmów kryptoanalitycznych „silniejszych" niż algorytm Grovera (Scott Aaronson w pracy „Quantum Computing and Hidden Variables" wskazuje możliwość poprawy skuteczności przeszukiwania do klasy $O(N^{1/3})$), to min.:

- Kryptografia symetryczna (o odpowiednio dużych rozmiarach kluczy);
- Hash-based cryptography (funkcje jednokierunkowe bez „trap-door");
- Lattice-based cryptography (wykorzystanie problemu poszukiwania najbliższego wektora w kracie, np. systemy NTRU);
- Code-based cryptography (systemy oparte na kodach korygujących błędy, np.: schemat podpisu McEliece'a z kodem Goppy);
- Multivariate quadratic equations cryptography (równania kwadratowe wielu zmiennych, np. „oil & vinegar" HFE^{v} system);
- Supersingular elliptic curve isogeny cryptography (wykorzystanie izogenii supersingularnych/super-osobliwych krzywych eliptycznych).

HASH-BASED CRYPTOGRAPHY

- Post-kwantowe algorytmy
- Tylko bezpieczne funkcje skrótu
- Bezpieczeństwo dobrze zrozumiane
- Szybkie

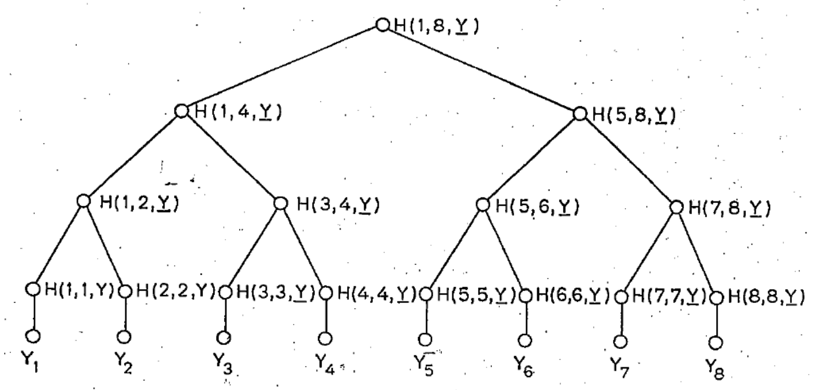
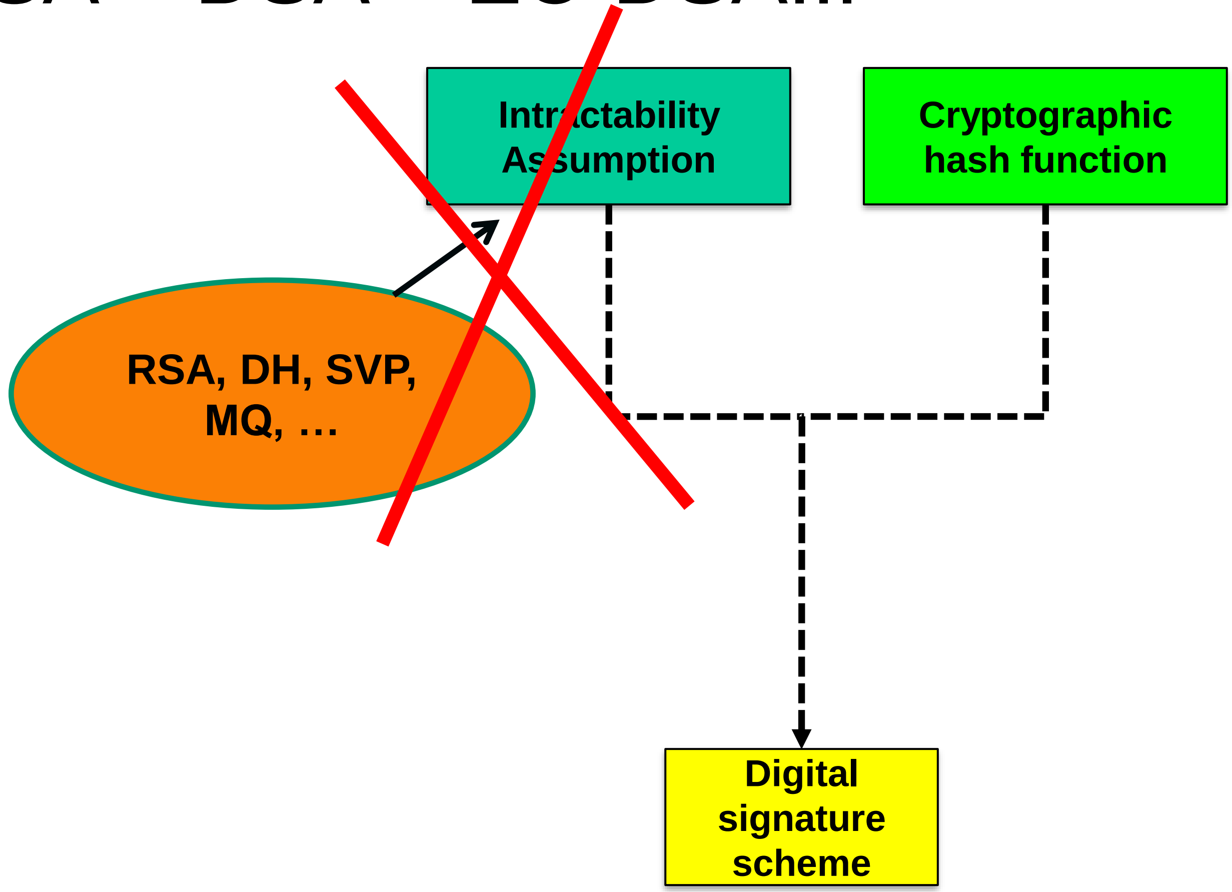


FIG 1
AN AUTHENTICATION TREE WITH N = 8.

PAGE 41B

HASH-BASED CRYPTOGRAPHY

RSA – DSA – EC-DSA...



HASH-BASED CRYPTOGRAPHY

Jednokierunkowość:

$$H_n \coloneqq \left\{ h_k\colon \{0,1\}^{m(n)} \to \{0,1\}^n \right\}$$

$$h_k \xleftarrow{\$} H_n$$
$$x \xleftarrow{\$} \{0,1\}^{m(n)}$$
$$y_c \leftarrow h_k(x)$$

Success if $h_k(x^*) = y_c$

y_c, k

$\downarrow$

$\mathcal{A}$ 

$\downarrow$

x^*

HASH-BASED CRYPTOGRAPHY

Odporność na kolizje

$$H_n \coloneqq \left\{ h_k \colon \{0,1\}^{m(n)} \rightarrow \{0,1\}^n \right\}$$

$$h_k \xleftarrow{\$} H_n$$

Success if

$h_k(x_1^*) = h_k(x_2^*)$ and
$x_1^* \neq x_2^*$

k

$\mathcal{A}$

(x_1^*, x_2^*)

HASH-BASED CRYPTOGRAPHY

Second-preimage resistance

$$H_n \coloneqq \left\{h_k\colon \{0,1\}^{m(n)} \rightarrow \{0,1\}^n\right\}$$

$$h_k \xleftarrow{\$} H_n$$

$$x_c \xleftarrow{\$} \{0,1\}^{m(n)}$$

Success if

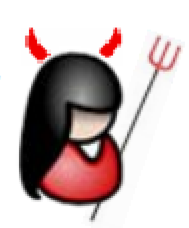
$h_k(x_c) = h_k(x^*)$ and

$x_c \neq x^*$

x_c, k

$\downarrow$

$\mathcal{A}$



$\downarrow$

x^*

HASH-BASED CRYPTOGRAPHY

Punktem wyjściowym dla konstrukcji systemów kryptograficznych opartych na funkcjach skrótu jest w zasadzie koncepcja podpisu jednorazowego Lamporta i Diffie'go (1979) przedstawiona poniżej.

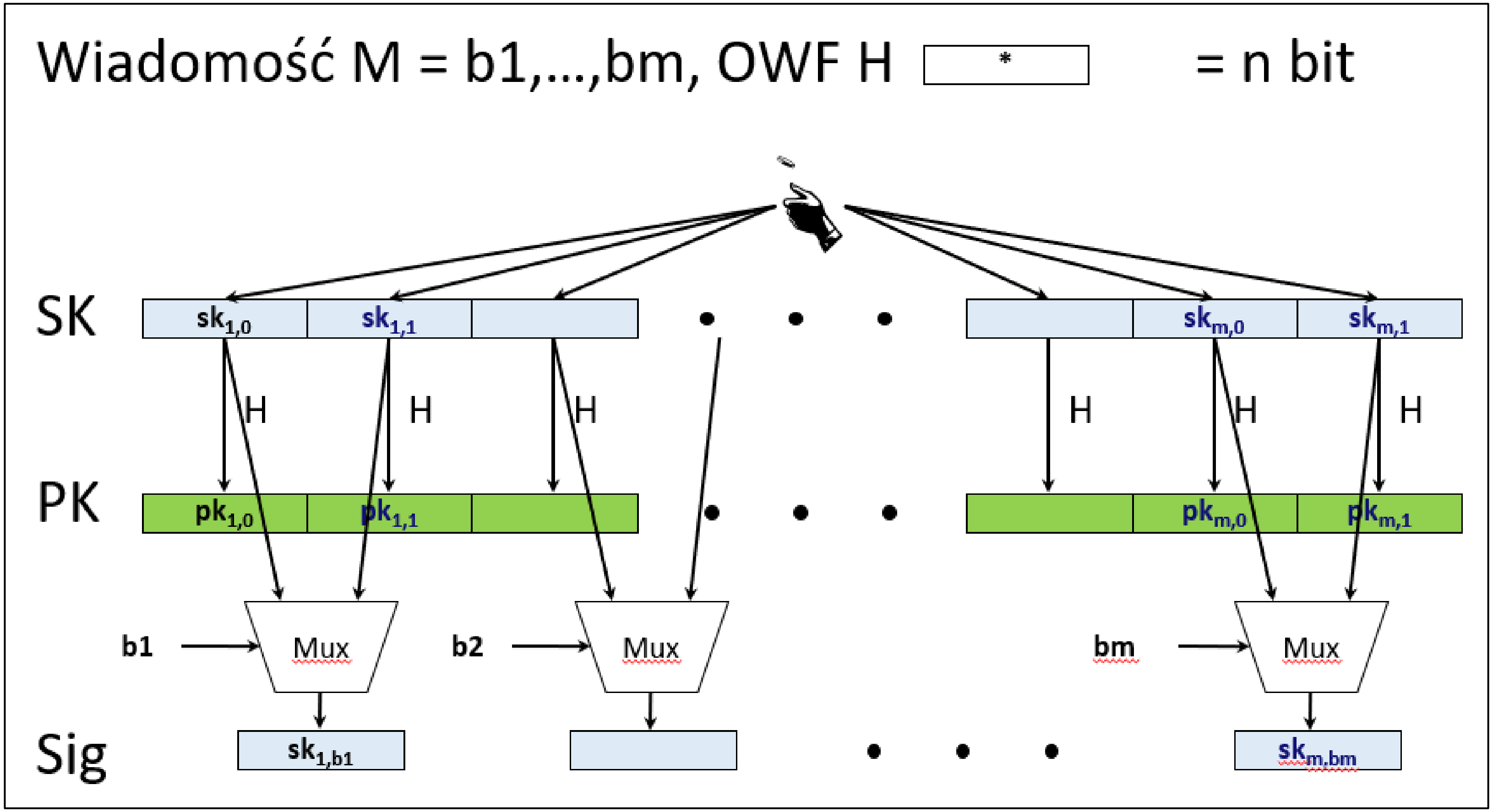
Generowanie pary kluczy (dla podpisania wiadomości k-bitowej)

Niech funkcją jednokierunkową będzie k-bitowa funkcja skrótu h(.). Podpisujący dysponując „przyzwoitym" RNG generuje k par losowych ciągów k-bitowych.

Kluczem prywatnym jest zestaw $2k^2$-bitowy, przy czym każdy z k-bitowych ciągów losowych indeksuje się pozycją i wartością binarną (tzn. pozycji i-tej odpowiadają ciągi losowe $y_{i,0}$ i $y_{i,1}$).

Kluczem publicznym będzie także zestaw $2k^2$-bitowy, a jego elementami będą indeksowane pozycją i wartością binarną ciągi $z_{i,j} = h(y_{i,j})$, gdzie $j \in \{0,1\}$, $1 \leq i \leq k$.

HASH-BASED CRYPTOGRAPHY



HASH-BASED CRYPTOGRAPHY (cd.)

Podpisywanie

Niech podpisywana wiadomość będzie ciągiem k-bitowym, którego kolejne bity to m_i ($1 \leq i \leq k$).

Podpisem będzie k-elementowy ciąg k-bitowych wartości s_i, przy czym wybór jednego z dwóch ciągów odpowiadającej części klucza prywatnego związanego z i-tym bitem podpisywanej wiadomości zależy od wartości i-tego bitu podpisywanej wiadomości:

jeżeli $m_i = 0$, to $s_i = y_{i,0}$;

jeżeli $m_i = 1$, to $s_i = y_{i,1}$.

UWAGA: dla 256-bitowej funkcji skrótu (np. 256-bitowe SHA2 lub SHA3) podpis 256-bitowej wiadomości to 256x256 = 65536 bitów, zaś klucz publiczny niezbędny do jego weryfikacji to 131072 bity !!!

HASH-BASED CRYPTOGRAPHY (cd.)

Weryfikacja podpisu

Ponieważ schemat dotyczy podpisu z załącznikiem, więc weryfikator zna podpisywaną wiadomość (ciąg m_i), podpis (ciąg s_i) oraz klucz publiczny (ciągi $z_{i,j} = h(y_{i,j})$, gdzie $j \in \{0,1\}$, $1 \leq i \leq k$).

Dla każdego bitu wiadomości m_i weryfikator sprawdza:

jeżeli $m_i = 0$, czy $h(s_i) = z_{i,0}$;

jeżeli $m_i = 1$, czy $h(s_i) = z_{i,1}$.

Podpis jest zweryfikowany pomyślnie wtedy, gdy dla każdego bitu podpisanej wiadomości powyższe równości są spełnione.

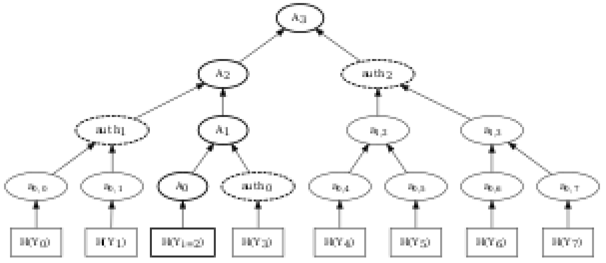
UWAGA: przedstawiony schemat nie uwzględnia wprowadzonego przez Diffie'go dodatkowego k-bitowego ciągu losowego, ale nie zmienia to logiki schematu, oraz faktu, że do podpisania kolejnej wiadomości trzeba ponownie wygenerować nową parę kluczy.

HASH-BASED CRYPTOGRAPHY (cd.)

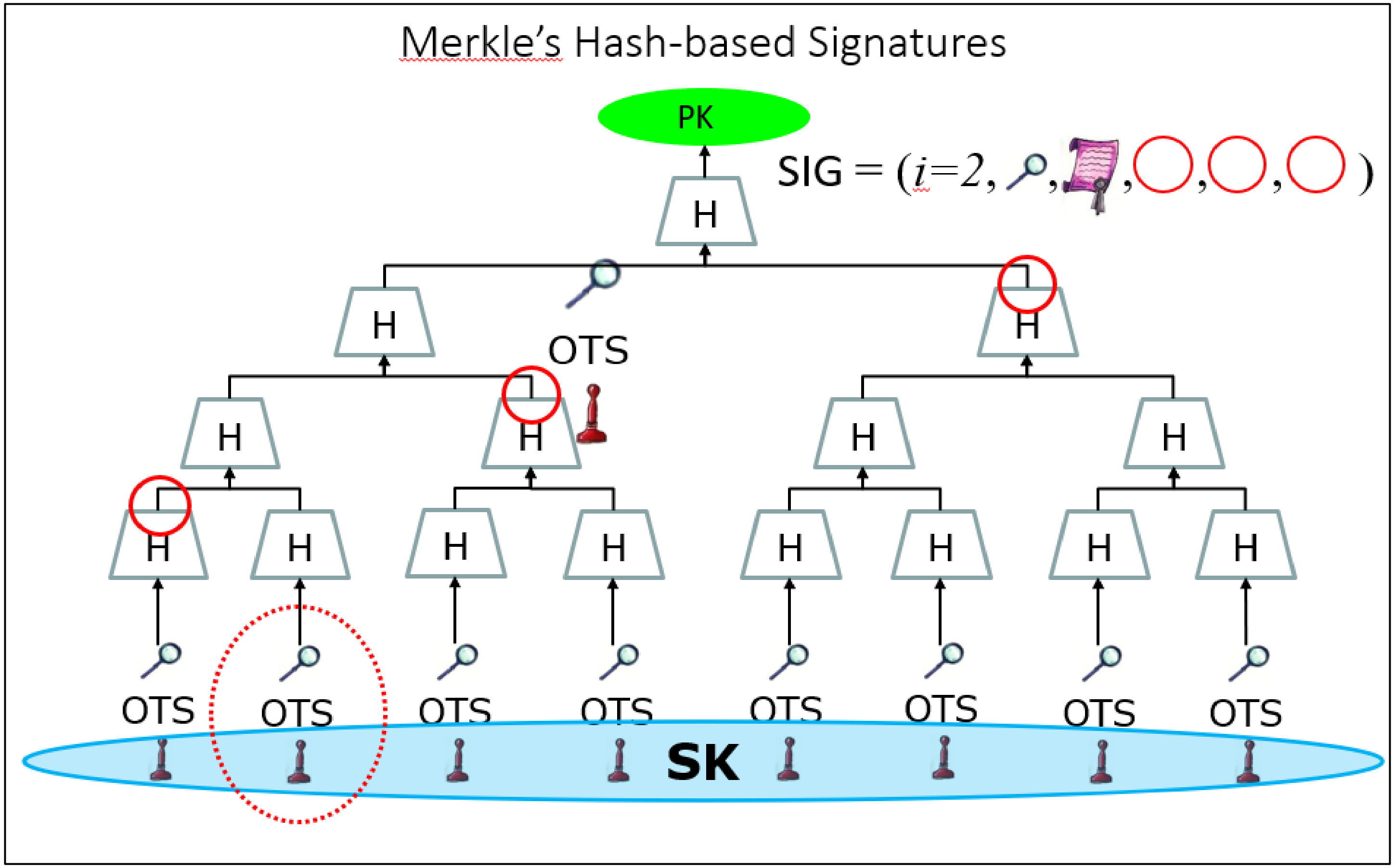
A jeśli chcemy podpisać więcej niż jedną wiadomość ?

Rozwiązaniem jest (jak zwykle w ostatnich czasach) „chaining”.
Podpisujący mógłby umieszczać w podpisanej wiadomości nowo wygenerowany klucz publiczny, który będzie wykorzystany do weryfikacji następnej podpisanej wiadomości.
Mogłby? Przecież sam klucz publiczny jest większy od podpisywanej wiadomości!!!

Ale przecież jest jeszcze drzewo Merkle’a i inne podobne pomysły!!!
Zamiast kolejnych kluczy publicznych wystarczy tylko informacja o skrótach ulokowanych w odpowiednich węzłach.
I tak np. w schemacie XMSS (eXtended Merkle Signature Scheme) klucz publiczny się nie zmienia i jest wartością umieszczoną w korzeniu odpowiedniego drzewa Merkle’a.



HASH-BASED CRYPTOGRAPHY (cd.)



HASH-BASED CRYPTOGRAPHY - Demo

Konsola debugowania programu Microsoft Visual Studio

```
Skrót z wiadomości:     02548620E4854FE9B54D2BD786A69AFB86EC5C2201E02F7E1BF0D95E976F8CD0
Liczba bitów skrótu:    256
Podpis:
8FBDA6790F8C3A1C49CEE6536D7EDB2BD715F2D6D16DA4BDCAFF4C76A9770175
5520315337B04FBF6E5F329EBFC250C05353BB18E0E153DD6C2F4452428550DC
9DE30D6A86C59027DB349969D3E5EAF90CF71BD2D08FD7937597AD18EB702790
AC16E4A95E8D3424395B127358E1F01DA315DFF29483408FA0E25B280DA3E5CB
1F5336C79939AECBA5F7EE8D2F11EE1EFA369972125111878A73EAB8B4D5420A
215FEC09FCE7E04BDEFC2819F9F728D8F67228CEFEF02CE388CDBBCDB8A0759F
4A06564C178EEBC8A5507A1E57C49DADAF6A049909A9AAD5A8056675C4726000
284E6EDB89CA9ECDDF3613440D733BB1D78739F6B5BF474A21C944B2A8947383
93E3EB2CA07E8D3DAEEF6C39810C6701211FA2283F256C6F6E05CC18C351B0C0
0ECBF73F9E6732B1E8008B4F9E7B107718FF01ED6EAAF360A4AF53AC3BEC1B4E
D676C9BDF92EA877EA5F6BEAE6056FDFFE6A2A8FC3CA4448C16FC57A0828EF3F
5E7F3FD1E21DF0A6DF007D61A272E0B282677D1544160143D9D1942015F61B17
BF7BF22A7B87E1BCDF94C281E7F87BE098F90A45796F2EB7C72F94CCC9CE8AEE
FAB5360F94904AF8DBE7EB82560A7B66136D242CA97FF766CD6C46F104AF26D5
AD9BF4E27586F8B419230BE8236F00E15106586BD3A84785FA2FCDBA7AC5D4EA
0A2764E912DF61C93136AFA20B73211E887081E3C64EFA80365D51E4410E66D9
CA7A21441AE500DFEEC7285CD9C44585178B980E64DC1BCBEDE9A4A6135F43B7
0DD086A791719FDDE99A4B50298CD7DCC0E827F5B09C0B2973FF334E86675AC0
8B4F06660CB1F3145DAB295B81DADDD0D721FC46C523EFBB199D38012B3874CF
0B2CD699C5A7EED3D503F73B01D2CF3783EC723A8B872BEFD4806C19EE77462A
8E1EDF76FDE0F5B9EB560B7994249305DC82829A5B7A01A4E7DCE49B33F219C5
FB0E358BB5F1C4DE9E4598F493159D713F99118CF29944145EAEE48F69668569
8B42F3AFE9CC96561FEE58B631FB8F3B7ADB32D04CBA14C65296E0172B7EA5D3
5C4C5253306985D5A72715A9B66FCFCEF164BF94D5C5626AB3103E06351CB530
DD84F691ED601A2B9F89E11294A8C1FCCB8266E49104685D7BF1E720FA21C816
F8BFC0E47A8E4ADA5D074364ED1C9783547002FAFBF18A0C96F529C62AB0C5F4
1AF07D6F7A86E5804710D0A19DF3A05AF36ED93954F801F984CA86B4ED6B330A
7B743D091FD5C82338FD939F704F53146F42D311F30EFD4820DF4A04E6FEA397
717D947D933B53A03029B927C8ADDD00C45041BA6A516F426667714A5C9A24B4
68445D629DF07F00B614187D09C075A281B33A29531174B207CE8F97090D0611
C7C69A76FD41DAD76D94A91F4A9B30A9F2B278F62E3EA04F92847DEB0FCEC3AE
4663273EF3ED15C515677DE9E037C796B019AD507A264F5C3B0641963A6FCA6B
C598F49837F381EBC5BA0F3184176236C2275DAAF7C33FDF4D5E6D90299ED203
```

PRZESTRZENIE WEKTOROWE (LINIOWE)

Przestrzeń wektorową V nad ciałem F tworzy grupa abelowa (przemienna) $(V, +)$ wraz z operacją mnożenia $\circ : F \times V \to V$, spełniająca następujące aksjomaty:

$$\forall\, a, b \in F \wedge \forall\, v, w \in V$$

$a(v + w) = av + aw$ (uproszczony zapis $av = a \circ v$)

$(a + b)v = av + bv$

$(ab)v = a(bv)$

$1 \circ v = v$

Elementy przestrzeni V są nazywane *wektorami*, zaś elementy ciała F - *skalarami*. Odpowiednio: operacja $+$ jest nazywana *dodawaniem wektorów*, zaś operacja $\circ$ - *mnożeniem skalarnym*.

Podprzestrzenią U przestrzeni V jest addytywna podgrupa U przestrzeni V domknięta ze względu na mnożenie skalarne, tzn.:

$$\forall\, a \in F\ \forall\, v \in U : av \in U$$

Podprzestrzeń przestrzeni wektorowej <u>jest także</u> przestrzenią wektorową.

PRZESTRZENIE WEKTOROWE (cd.)

Jeżeli F jest dowolnym ciałem, to n-krotny produkt kartezjański

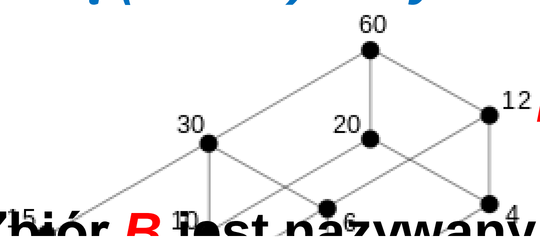
$$V = F \times F \times \ldots \times F = F^{\,n}$$

jest przestrzenią wektorową nad ciałem F, zaś wymiar przestrzeni V:

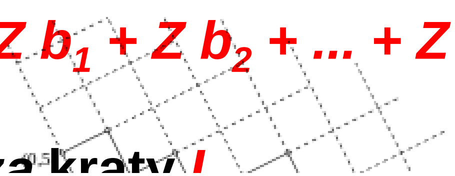
$$\dim V = n.$$

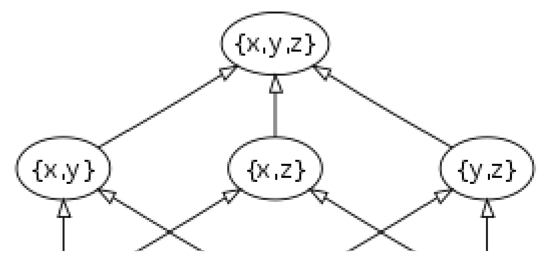
Niech $B = \{ b_1 , b_2 , \ldots, b_m \}$ będzie zbiorem liniowo niezależnych wektorów w przestrzeni wektorowej $F^{\,n}$ oraz $m \le n$.

Zbiór L wszystkich kombinacji liniowych wektorów zbioru B o współczynnikach będących liczbami całkowitymi jest nazywany *kratą (lattice) o wymiarze m*, tzn.:



$$L = Z\,b_1 + Z\,b_2 + \ldots + Z\,b_m$$





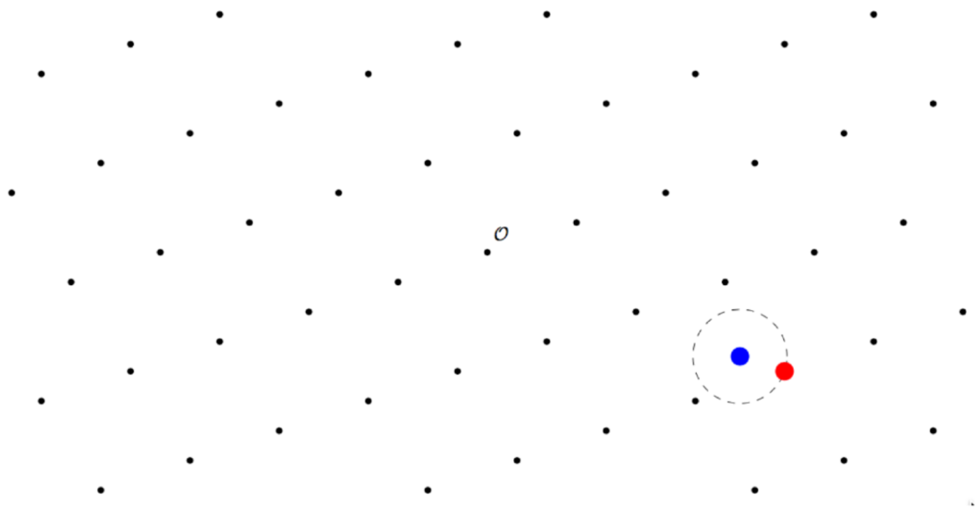
Zbiór B jest nazywany *bazą kraty L*

<u>Problem poszukiwania najbliższego wektora w kracie</u>

Dla danego wektora ***v*** należącego (lub nie) do danej kraty ***L*** (o dużej wartości ***dim L***) wyznaczyć taki wektor ***w*** należący do kraty ***L***, który jest "najbliższy" wektora ***v***.

PRZESTRZENIE WEKTOROWE (cd.)

Closest vector problem



Given some basis for the lattice and a target point in the space, find the closest lattice point.

PIERŚCIENIE WIELOMIANÓW (1/7)

Jeżeli R jest pierścieniem przemiennym, to wówczas wielomianem zmiennej x nad pierścieniem R jest wyrażenie postaci:

$$f(x) = a_n x^n + \ldots + a_2 x^2 + a_1 x + a_0 \ ,$$

gdzie $n \in Z$ i $n \geq 0$, oraz $\forall\, i : a_i \in R$.

Stopień wielomianu deg f(x) jest równy najwyższej potędze zmiennej x o niezerowym współczynniku a_i.

Wielomianem zerowym jest wielomian o wszystkich współczynnikach zerowych (a jego stopień definiuje się jako $-\infty$).

Pierścieniem wielomianów R[x] jest pierścień utworzony ze zbioru wszystkich wielomianów zmiennej x o współczynnikach z pierścienia przemiennego R. Operacjami binarnymi pierścienia wielomianów są dodawanie i mnożenie wielomianów.

PIERŚCIENIE WIELOMIANÓW (2/7)

Przykład

$Z_2 = \{0, 1\}$ jest pierścieniem przemiennym. Zbiór wszystkich wielomianów o współczynnikach należących do Z_2 stanowi pierścień wielomianów $Z_2[x]$.

Niech: $f(x) = x^3 + x + 1$ i $g(x) = x^2 + x$

Wtedy:

$f(x) + g(x) = x^3 + x^2 + 1$ (bo $x + x = 0 \cdot x$!!!)

$f(x) \cdot g(x) = x^5 + x^4 + x^3 + x$ (bo $x^2 + x^2 = 0 \cdot x^2$!!!)

Niech $F[x]$ będzie pierścieniem wielomianów nad ciałem F.

Wielomian $f(x) \in F[x]$ jest nazywany wielomianem nierozkładalnym w pierścieniu $F[x]$ wtedy, gdy nie można go przedstawić w postaci iloczynu dwóch wielomianów należących do tego pierścienia o stopniu co najmniej równym 1.

PIERŚCIENIE WIELOMIANÓW (3/7)

Dzielenie wielomianów

Niech $g(x), h(x) \in F[x]$, a ponadto $h(x) \neq 0$. Wtedy wynikiem dzielenia $g(x)$ przez $h(x)$ są wielomiany $q(x), r(x) \in F[x]$ takie, że:

$$g(x) = q(x)h(x) + r(x) \quad \text{i} \quad \deg r(x) < \deg h(x),$$

a ponadto wielomiany te są wyznaczone jednoznacznie (i nazywają się odpowiednio: **ilorazem** i **resztą**).

Niekiedy przyjmuje się oznaczenia:

$$q(x) = g(x) \text{ div } h(x) \quad \text{oraz} \quad r(x) = g(x) \text{ mod } h(x).$$

Przykład

Niech $g(x), h(x) \in Z_2[x]$: $g(x) = x^3 + x + 1$, $h(x) = x^2 + x$.

Wtedy: $g(x) \text{ div } h(x) = x + 1$ oraz $g(x) \text{ mod } h(x) = 1$

(bo $(x + 1)(x^2 + x) + 1 = (x^3 + x^2 + x^2 + x) + 1 = x^3 + x + 1$).

PIERŚCIENIE WIELOMIANÓW (4/7)

Niech $g(x), h(x) \in F[x]$.

Wielomian $h(x)$ dzieli wielomian $g(x)$ wtedy, gdy $g(x) \bmod h(x) = 0$.

Oznacza się to: $h(x) \mid g(x)$.

Niech $f(x) \in F[x]$.

Podobnie jak dla liczb całkowitych, wprowadza się pojęcie **kongruencji (przystawania) wielomianów** w pierścieniu $F[x]$ w oparciu o dzielenie przez wielomian $f(x)$.

Jeżeli $g(x), h(x) \in F[x]$, to $g(x)$ jest **kongruentny** do $h(x)$ **modulo** $f(x)$ wtedy, gdy $f(x)$ dzieli $g(x) - h(x)$, czyli $f(x) \mid (g(x) - h(x))$ (znak $-$ oznacza dodanie elementu odwrotnego względem dodawania w pierścieniu).

Oznaczane jest to: $g(x) \equiv h(x) \pmod{f(x)}$.

PIERŚCIENIE WIELOMIANÓW (5/7)

Niech f(x) będzie pewnym ustalonym wielomianem z pierścienia F[x]. Podobnie jak w przypadku kongruencji liczb całkowitych, wprowadza się pojęcie klasy równoważności wielomianów g(x) ∈ F[x], czyli wszystkich wielomianów z tego pierścienia kongruentnych do g(x).

Reprezentantem klasy równoważności jest jednoznacznie wyznaczony wielomian r(x) = g(x) mod f(x) taki, że deg r(x) < deg f(x).

Symbol F[x] / f(x) oznacza zbiór wszystkich wielomianów (rozłącznych klas równoważności) w pierścieniu F[x] o stopniu n < deg f(x), przy czym klasy te wyznaczone są przez kongruencje, których modułem jest f(x), a ponadto operacje dodawania i mnożenia wielomianów też są wykonywane modulo f(x).

F[x] / f(x) jest pierścieniem przemiennym.

f(x) jest wielomianem nierozkładalnym w pierścieniu F[x] ⇒ F[x] / f(x) jest ciałem.

PIERŚCIENIE WIELOMIANÓW (6/7)

Rozważmy pierścień wielomianów Z_p [x], przy czym p jest liczbą pierwszą Jeżeli f(x) jest nierozkładalny, to można zdefiniować ciało skończone wielomianów: Z_p [x] / f(x).
Liczba wszystkich takich wielomianów wynosi p^m, gdzie m = deg f(x).
Ciało to zawiera także ciało Z_p (wszystkie wielomiany stopnia 0).

W ciele Z_p [x] / f(x) w analogiczny sposób, jak dla ciała Z_p, definiuje się pojęcia elementu odwrotnego, generatora, dodawania, mnożenia i dzielenia modulo f(x), największego wspólnego dzielnika g(x) i h(x), określa jednoznaczną formę faktoryzacji (przy czym odpowiednikami czynników pierwszych są wielomiany nierozkładalne), itp.

Jeżeli współczynnik przy najwyższej potędze wielomianu jest równy 1, to taki wielomian nazywa się wielomianem unormowanym (monic).
Dla każdego $m \geq 1$ istnieje nierozkładalny wielomian unormowany $f(x) \in Z_p[x]$ stopnia m.
Wniosek: Każde ciało skończone posiada swoją reprezentację wielomianową.

PIERŚCIENIE WIELOMIANÓW (7/7)

$x^2 + x = x(x + 1)$
$x^2 + 1 = (x+1)(x+1)$
$x^2 = x \bullet x$

Przykład

Wielomian $f(x) = x^2 + x + 1$ jest unormowanym wielomianem nierozkładalnym w $Z_2 [x]$.
$Z_2 [x] / f(x)$ jest ciałem skończonym F_2^2, zawierającym 4 wielomiany:

$$x + 1, x, 1, 0$$

Dodawanie				
+	x+1	x	1	0
x+1	0	1	x	x+1
x	1	0	x+1	x
1	x	x+1	0	1
0	x+1	x	1	0

Mnożenie				
*	x+1	x	1	0
x+1	x	1	x+1	0
x	1	x+1	x	0
1	x+1	x	1	0
0	0	0	0	0

Grupę multiplikatywną $F^*{}_2^2$ tworzą wielomiany $x + 1$, x i 1, zaś wielomianami do nich odwrotnymi są:

$(x + 1)^{-1} = x$ $\quad$ $x^{-1} = x + 1$ $\quad$ $1^{-1} = 1$

PIERŚCIENIE WIELOMIANÓW OBCIĘTYCH (OBCINANYCH)

W 1996 roku Jeffrey Hoffstein, Jill Pipher i Joseph H.Silverman przedstawili asymetryczny system kryptograficzny NTRU, wykorzystujący tzw. wielomiany obcięte (truncated polynomials).

Wielomiany obcięte określone są nad pierścieniem R i są postaci:

$$f(x) = a_{N-1}x^{N-1} + \ldots + a_2x^2 + a_1x + a_0,$$

gdzie $N \geq 1$ jest ustalone oraz $\forall i{:}\ a_i \in R$.

Wielomiany te tworzą pierścień R_N, w którym operacja dodawania wielomianów zdefiniowana jest tak, jak w pierścieniu wielomianów $R[x]$:

$$f(x) = a_{N-1}x^{N-1} + \ldots + a_2x^2 + a_1x + a_0$$
$$g(x) = b_{N-1}x^{N-1} + \ldots + b_2x^2 + b_1x + b_0$$

$$h(x) = f(x) + g(x) = (a_{N-1} + b_{N-1})x^{N-1} + \ldots + (a_2 + b_2)x^2 + (a_1 + b_1)x + (a_0 + b_0).$$

Operacja mnożenia wielomianów nie kończy się po obliczeniu współczynników wielomianu stopnia $2(N-1)$, lecz po wykonaniu mnożenia składniki wielomianu postaci $a_{N+k}\,x^{N+k}$, gdzie $0 \leq k \leq N-2$, są zastępowane przez $a_{N+k}\,x^{k}$, a następnie współczynniki przy składnikach o tych samych potęgach zmiennej x są sumowane.

Przykład

$R = Z \quad N = 3 \quad f(x) = 3x^2 - x + 2 \quad g(x) = - x^2 + 2x + 1$

$f(x) + g(x) = (3-1)\,x^2 + (-1+2)\,x + (2+1) = 2x^2 + x + 3$

$$f(x)g(x) = (3x^2 - x + 2)(- x^2 + 2x + 1) = -3x^4 + 7x^3 - x^2 + 3x + 2 =$$
$$= \underline{-3x + 7} - x^2 + 3x + 2 = - x^2 + 9$$

Pierścień wielomianów obciętych R_N jest w tym przypadku izomorficzny do pierścienia $Z[x]/(x^N-1)$, utworzonego z wielomianów będących resztą z dzielenia wielomianów przez wielomian (x^N-1).

W systemach NTRU wykorzystuje się pierścienie wielomianów obciętych $Z_q[x]/(x^N-1)$.

Przykład

$R = Z_5 \quad N = 3 \quad f(x) = 3x^2 + x + 2 \qquad g(x) = x^2 + 4x + 4$

$f(x) + g(x) = ((3+1) \bmod 5)\, x^2 + ((1+4) \bmod 5)\, x + ((2+4) \bmod 5) =$
$= (4x^2 + 1) \pmod 5$

$f(x)g(x) = (3x^2 + x + 2)(\ x^2 + 4x + 4) \pmod 5 =$
$= (3x^4 + 13x^3 + 14x^2 + 12x + 8) \pmod 5 =$
$= (\underline{3x + 13} + 14x^2 + 12x + 8) \pmod 5 =$
$= (14x^2 + 15x + 21) \pmod 5 = (4x^2 + 1) \pmod 5$

W pierścieniu $Z_q[x]/(x^N-1)$ wielomian $g(x) = f^{-1}(x)$ odwrotny do $f(x)$ określa zależność:

$$f(x)g(x) \equiv 1 \pmod q$$

SZYFROWANIE ASYMETRYCZNE NTRU (PRZYKŁAD)

Parametrami, które muszą określić użytkownicy systemu, są:

- N - wykorzystywane wielomiany będą stopnia $N-1$;
- $q \in Z$ - tzw. duży moduł;
- $p \in Z$ - tzw. mały moduł (zazwyczaj $p = 3$);
- d_f - prywatny wielomian $f(x)$ będzie miał d_f współczynników równych $+1$, oraz $(d_f - 1)$ współczynników równych -1;
- d_g - prywatny wielomian $g(x)$ będzie miał d_g współczynników równych $+1$, oraz d_g współczynników równych -1;
- d_r - losowy wielomian szyfrujący $r(x)$ będzie miał d_r współczynników równych $+1$, oraz d_r współczynników równych -1.

Generowanie pary kluczy

Podmiot A:

- **wybiera losowe dwa wielomiany *f(x)* i *g(x)* należące do R_N i spełniające warunki:**

  - d_f współczynników wielomianu *f(x)* ma wartość *+1*, $(d_f - 1)$ współczynników ma wartość *-1*, zaś pozostałe współczynniki mają wartość *0*;

  - d_g współczynników wielomianu *g(x)* ma wartość *+1*, d_g współczynników ma wartość *-1*, zaś pozostałe współczynniki mają wartość *0*;

UWAGA: Ujawnienie któregokolwiek z tych wielomianów umożliwia atak na system.

Generowanie pary kluczy (cd.)

- **oblicza wielomiany odwrotne do $f(x)$: $F_q(x)$ i $F_p(x)$ należące do R_N i spełniające warunki:**

$$f(x)F_q(x) \equiv 1 \pmod q$$
$$f(x)F_p(x) \equiv 1 \pmod p$$

(jeżeli nie istnieje którykolwiek z wielomianów odwrotnych, wówczas podmiot A musi wybrać inny losowy wielomian $f(x)$).

- **oblicza wielomian $h(x)$:**

$$h(x) = pF_q(x)g(x) \bmod q.$$

klucz prywatny **- para wielomianów $(f(x),F_p(x))$**
klucz publiczny **- wielomian $h(x)$ oraz N, p, q, d_r**

Szyfrowanie

Podmiot B:

- **przekształca wiadomość jawną do postaci wielomianu:**

$$m(x) \in Z_p[x]/(x^N-1);$$

  (zaleca się, by współczynniki wielomianu $m(x)$ należały do przedziału $[-p/2; p/2]$);

- **wybiera losowy wielomian $r(x)$ należący do R_N i spełniający warunek:**
  d_r współczynników wielomianu $r(x)$ ma wartość $+1$,
  d_r współczynników ma wartość -1, zaś pozostałe współczynniki mają wartość 0;

- **wykorzystując klucz publiczny podmiotu A, czyli wielomian $h(x)$, oblicza wielomian będący kryptogramem $c(x)$:**

$$c(x) = (r(x)h(x) + m(x)) \bmod q.$$

Deszyfrowanie

Podmiot A:

- **korzystając ze swego wielomianu prywatnego *f(x)* oblicza wielomian:**

$$a(x) = f(x)c(x) \bmod q$$

(*bardzo ważne* jest, by współczynniki wielomianu *a(x)* należały do przedziału [-q/2; q/2]);

- **oblicza (przez redukcję współczynników wielomianu *a(x) mod p*) wielomian:**

$$b(x) = a(x) \bmod p$$

- **odtwarza wiadomość jawną wykorzystując drugi z wielomianów tworzących jego klucz prywatny:**

$$m'(x) = F_p(x)b(x) \bmod p.$$

Uzasadnienie poprawności deszyfrowania

Pierwsza faza obliczeń:

$$a(x) = f(x)c(x) \bmod q = f(x)[r(x)h(x) + m(x)] \bmod q =$$
$$= f(x)[r(x)pF_q(x)g(x) + m(x)] \bmod q =$$
$$= [pr(x)g(x) + f(x)m(x)] \bmod q,$$

gdyż: $f(x)F_q(x) \equiv 1 \pmod q$.

A zatem:

$$a(x) = [pr(x)g(x) + f(x)m(x)] \bmod q.$$

Współczynniki wielomianów $r(x)$, $g(x)$, $f(x)$ i $m(x)$ są niewielkie (jeżeli $p = 3$, to wszystkie należą do zbioru $\{-1, 0, +1\}$).

Zatem także współczynniki iloczynów wielomianów $pr(x)g(x)$ i $f(x)m(x)$ będą niewielkie (przynajmniej w porównaniu z q).

Uzasadnienie poprawności deszyfrowania (cd.)

Współczynniki wielomianu *a(x)* nadal będą należały do przedziału [$-q/2; q/2$], a zatem redukcja współczynników *mod q* jest operacją *praktycznie bez znaczenia*.

Redukcja współczynników wielomianu *a(x)* prowadzi do zależności:

$$b(x) = a(x) \bmod p = p\,r(x)g(x) \bmod p + f(x)m(x) \bmod p = $$
$$= f(x)m(x) \bmod p \ (!!!)$$

Stąd:

$$m'(x) = F_p(x)b(x) \bmod p = F_p(x)f(x)m(x) \bmod p = m(x) \bmod p,$$

gdyż: $f(x)F_p(x) \equiv 1 \pmod p$.

Przykład

Ustalono parametry systemu:

$$N = 11 \quad q = 32 \quad p = 3 \quad d_f = 4 \quad d_g = d_r = 3$$

Generowanie klucza

Podmiot A wybrał wielomiany:

$$f(x) = - x^{10} + x^9 + x^6 - x^4 + x^2 + x - 1$$
$$g(x) = - x^{10} - x^8 + x^5 + x^3 + x^2 - 1$$

Odpowiednie wielomiany odwrotne:

$F_p(x)= f(x)^{-1} \bmod p = 2x^9 + x^8 + 2x^7 + x^5 + 2x^4 +2x^3 + 2x + 1,$

$F_q(x)= f(x)^{-1} \bmod q =$

$= 30x^{10} + 18x^9 + 20x^8 + 22x^7 + 16x^6 + 15x^5 + 4x^4 +16x^3 + 6x^2 + 9x + 5.$

$F_p(x)$ i $f(x)$ **tworzą klucz prywatny podmiotu A.**

Obliczenie klucza publicznego:

$h(x) = pF_q(x)g(x) \bmod q =$

$= 16x^{10}+ 19x^9+ 12x^8+ 19x^7+ 15x^6+ 24x^5+ 12x^4+ 20x^3+ 22x^2+ 25x + 8.$

Szyfrowanie

Niech wiadomość jawna (wyrażona w formie wielomianu):

$$m(x) = x^{10} + x^9 - x^8 - x^4 + x^3 - 1.$$

Podmiot B wybiera losowy wielomian szyfrujący:

$$r(x) = - x^7 - x^5 + x^4 + x^3 + x^2 - 1.$$

Kryptogram:

$$c(x) = (r(x)h(x) + m(x)) \bmod q =$$
$$= (19x^{10} + 6x^9 + 25x^8 + 7x^7 + 30x^6 + 16x^5 + 14x^4 + 24x^3 + 26x^2 + 11x + 14) \bmod 32.$$

Deszyfrowanie

Podmiot A oblicza:

$a(x) = f(x)c(x) \bmod q =$
$= (-7x^{10} - 3x^9 + 5x^8 + 7x^7 + 6x^6 + 7x^5 + 10x^4 - 11x^3 - 10x^2 - 7x + 3) \bmod 32.$

> **UWAGA:** zredukowane współczynniki są wybierane z przedziału $[-15, 16]$, nie zaś z przedziału $[0, 31]$.

Obliczenie wielomianu $b(x)$:

$$b(x) = a(x) \bmod 3 = (-x^{10} - x^8 + x^7 + x^5 + x^4 + x^3 - x^2 - x) \bmod 3.$$

Odtworzenie wiadomości jawnej:

$$m'(x) = F_p(x)b(x) \bmod 3 = x^{10} + x^9 - x^8 - x^4 + x^3 - 1 = m(x).$$

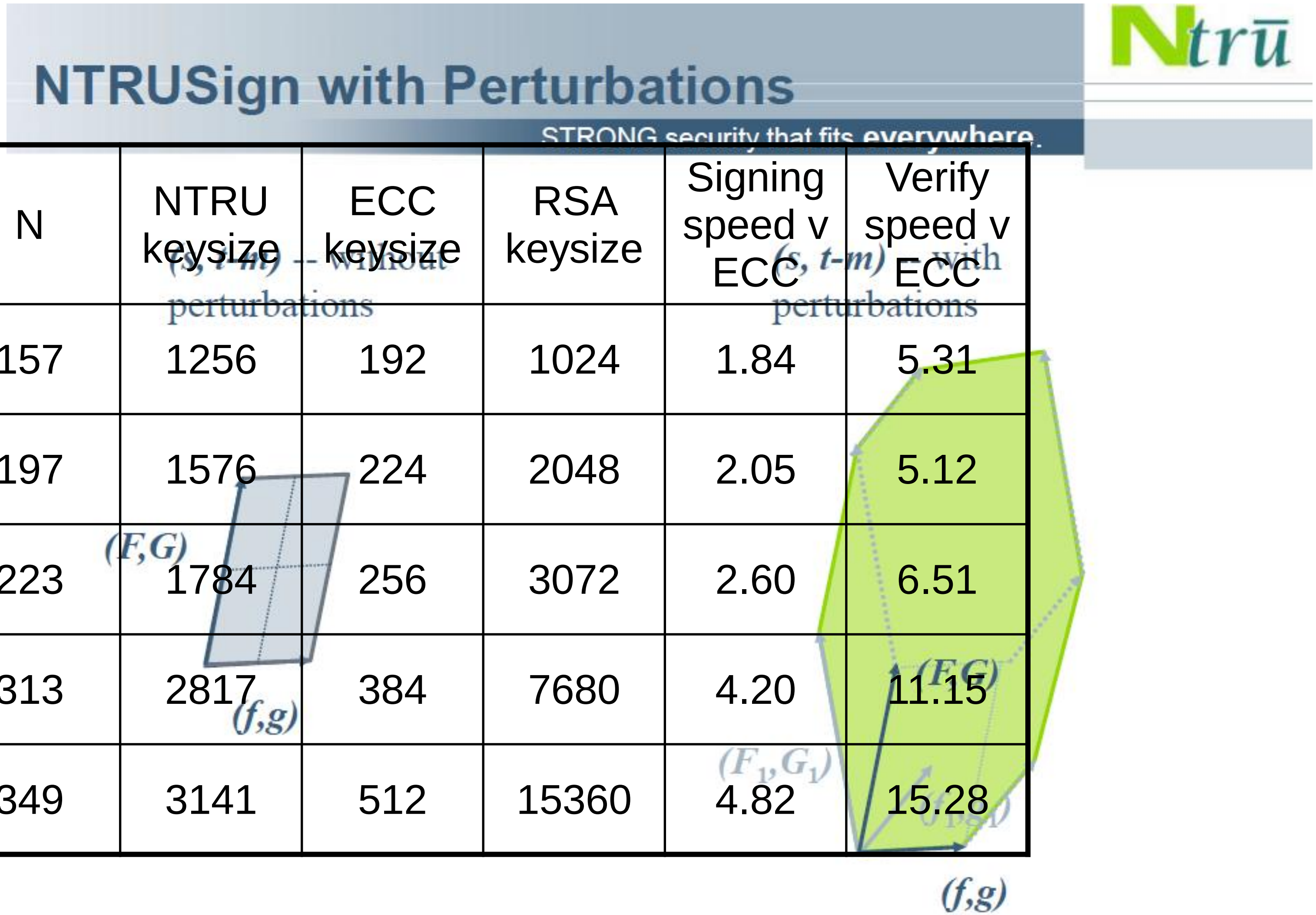
Zalecane wartości parametrów systemu NTRU

Bezpieczeństwo	N	q	p	d_f	d_g	d_r
Umiarkowane	167	128	3	61	20	18
Standardowe	251	128	3	bd.	bd.	bd.
Bardzo wysokie	503	256	3	216	72	55

Podkreśla się następujące „przewagi” systemów NTRU w stosunku do systemów opartych na pierścieniach liczb całkowitych Z_n (np. RSA), czy systemach opartych na krzywych eliptycznych (ECC):

- **porównywalne bezpieczeństwo przy 100-krotnie szybszych obliczeniach;**
- **krótkie, łatwo generowane klucze;**
- **małe wymagania zasobów pamięci i mocy obliczeniowej;**
- **większa elastyczność przy wyborze parametrów stosownie do potrzeb użytkownika;**
- **nieznane kwantowe algorytmy kryptoanalityczne.**

Source: Jeff Hoffstein, Nick Howgrave-Graham, Jill Pipher, Joe Silverman, William Whyte „NTRUSign: Digital Signatures in the NTRU Lattice”, 2003



k	N	NTRU keysize	ECC keysize	RSA keysize	Signing speed v ECC	Verify speed v ECC
80	157	1256	192	1024	1.84	5.31
112	197	1576	224	2048	2.05	5.12
128	223	1784	256	3072	2.60	6.51
192	313	2817	384	7680	4.20	11.15
256	349	3141	512	15360	4.82	15.28

IEEE

1363.1™

IEEE Standard Specification for Public Key Cryptographic Techniques Based on Hard Problems over Lattices

IEEE Computer Society

Sponsored by the
Microprocessor Standards Committee

IEEE
3 Park Avenue
New York, NY 10016-5997, USA

10 March 2009

IEEE S

ANSI X9.98-2010

Lattice-Based Polynomial Public Key Establishment Algorithm for the Financial Services Industry

Specifies the cryptographic functions for establishing symmetric keys using a lattice-based polynomial public key encryption algorithm and the associated parameters for key generation (see Note 1). The mechanism supported is key transport, where one party selects keying material and conveys it to the other party with cryptographic protection. The keying material may consist of one or more individual keys used to provide other cryptographic services outside the scope of this Standard, e.g. data confidentiality, data integrity, or symmetric-key-based key establishment. The standard also specifies key pair generators and corresponding key pair validation methods supporting the key transport schemes

Table A.15—Strengths of recommended parameter sets in this standard vs best current attacks

Parameter set	Recommended security level	N	q	d_f	Known hybrid strength	c	Basic lattice strength
ees401ep1	112	401	2048	113	154.88	2.02	139.5
ees541ep1	112	541	2048	49	141.766	1.77	189.4
ees659ep1	112	659	2048	38	137.861	1.74	231.5
ees449ep1	128	449	2048	134	179.899	2.17	156.6
ees613ep1	128	613	2048	55	162.385	1.88	215.1
ees761ep1	128	761	2048	42	157.191	1.85	267.8
ees677ep1	192	677	2048	157	269.93	2.50	239.0
ees887ep1	192	887	2048	81	245.126	2.27	312.7
ees1087ep1	192	1087	2048	63	236.586	2.24	384.0
ees1087ep2	256	1087	2048	120	334.85	2.64	459.2
ees1171ep1	256	1171	2048	106	327.881	2.60	494.8
ees1499ep1	256	1499	2048	79	312.949	2.57	530.8

NTRUEncrypt zaadoptowany do norm IEEE 1363.1 i ANSI X9.98.

CO NAS CZEKA W PRZYPADKU USŁUG ZAUFANIA (PKI) ?

hash-based

lattice-based

multivariate-based

Signature algorithm	Security level classic/quantum	Signature size (bytes)	Public key size (bytes)	Private key size (bytes)	Implementation
XMSS (SHA2-256_W16_H10)	256/128	2500	64	132	Botan
$XMSS^{MT}$ (SHA2-256_W16_H20_D2)	256/128	4964	64	132	https://huelsing.wordpress.com/code/
SPHINCS-256	<256/<128	41,000	1056	1088	Bouncy Castle
BLISS-B-IV	159/96	832	896	384	strongSwan
REBLISS-I	128/128	809	870	166	-
Rainbow(256,31,21,22)	101/101	74	122,600	87,700	Bouncy Castle
Gui-127	120/120(60)	21	142,576	5350	-
Isogeny-based	192/128	122,880	336	48	https://github.com/yhyoo93/isogenysignature
RSA-3072	128/Broken	384	384	1728	Widespread
ECDSA (P-256)	128/Broken	64	64	96	Widespread

Porównanie szacowanych poziomów bezpieczeństwa, rozmiarów podpisów i kluczy oraz dostępnych implementacji „post-kwantowych" i klasycznych algorytmów podpisów cyfrowych.

Źródło: Mikael Sjöberg – „Post-quantum algorithms for digital signing in Public Key Infrastructures"

Logowanie do Zimbra Web Clien | goppa code - Bing | nist postquantum - Bing | NIST Post-Quantum Cryptograph

https://en.wikipedia.org/wiki/NIST_Post-Quantum_Cryptography_Competition#Finalists

On July 22, 2020, NIST announced seven finalists ("first track"), as well as eight alternate algorithms ("second track"). The first track contains the algorithms which appear to have the most promise, and will be considered for standardization at the end of the third round. Algorithms in the second track could still become part of the standard, after the third round ends.[51] NIST expects some of the alternate candidates to be considered in a fourth round.

Finalists [edit]

Type	PKE/KEM	Signature
Lattice[a]	• CRYSTALS-KYBER • NTRU • SABER	• CRYSTALS-DILITHIUM • FALCON
Code-based	• Classic McEliece	
Multivariate		• Rainbow

On July 5, 2022, NIST announced the first group of winners from its six-year competition.[58][59]

Type	PKE/KEM	Signature
Lattice	• CRYSTALS-Kyber	• CRYSTALS-Dilithium • FALCON
Hash-based		• SPHINCS+

Alternate candidates [edit]

Type	PKE/KEM	Signature
Lattice	• FrodoKEM • NTRU Prime	
Code-based	• BIKE • HQC	
Hash-based		• SPHINCS+
Multivariate		• GeMSS
Supersingular Elliptic Curve Isogeny	• SIKE	
Zero-knowledge proofs		• Picnic

Intellectual property concerns [edit]

After NIST's announcement regarding the finalists and the alternate candidates, various intellectual property concerns were voiced, notably surrounding lattice-based schemes such as Kyber and NewHope. NIST holds signed statements from submitting groups clearing any legal claims, but there is still a concern that third parties could raise claims. NIST claims that they will take such considerations into account while picking the winning algorithms.[52]

PL 09:54 2021-02-01

Koniec części 8