

Kryptografia postkwantowa oparta na funkcjach skrótu (hash-based cryptography)

Kryptografia

Magdalena Ślusarczyk, Michał Bełzak



Wydział Informatyki, Elektroniki i Telekomunikacji
Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie

20.01.2022r.

Spis treści

1	Wprowadzenie	2
1.1	Komputery kwantowe jako zagrożenie wobec współczesnej kryptografii	2
1.2	Wstęp do kryptografii post-quantowej	2
1.3	Wykorzystanie funkcji skrótu w kryptografii post-quantowej	3
2	Systemy podpisów jednorazowych	4
2.1	System Lamporta	4
2.2	System Winternitza	6
3	System podpisów cyfrowych z wykorzystaniem drzew Merkle'a	8
3.1	Generowanie par kluczy	8
3.2	Ścieżki uwierzytelniające	9
3.3	Generowanie podpisu	11
3.4	Weryfikowanie podpisu	11
4	Podsumowanie	12
	Bibliografia	13

1 Wprowadzenie

Na możliwość budowy komputerów wykorzystujących prawa fizyki kwantowej zwrócono uwagę już w latach 80-tych. Urządzenia te wykorzystują zadady mechaniki kwantowej w oparciu o algorytmy, które mają za zadanie odpowiednie zaplanowanie ewolucji układu kwantowego. Algorytmy kwantowe wykorzystują do rozwiązywania konkretnych problemów obliczeniowych podstawową kwantową jednostkę informacji - kubit. Od klasycznych bitów wykorzystywanych w tradycyjnych komputerach kubit odróżnia to, że mogą znajdować się w całym spektrum stanów pośrednich, nie tylko 0 oraz 1. Oznacza to, że jest układem, który może przechowywać i przenosić znacznie większą ilość informacji niż bit, a więc i jego wydajność jest większa.

1.1 Komputery kwantowe jako zagrożenie wobec współczesnej kryptografii

Większą uwagę koncept komputerów kwantowych przyciągnął w 1994 roku, gdy Peter Shor opracował algorytm kwantowy pozwalający na rozkład bardzo dużych liczb na iloczyny liczb pierwszych o złożoności czasowej znacznie mniejszej niż jakiegokolwiek algorytm klasyczny [1]. Algorytm ten może zostać z powodzeniem wykorzystany do ataków przeciwko powszechnie stosowanym obecnie kryptosystemom opierających swoje bezpieczeństwo na problemie faktoryzacji dużych liczb całkowitych, takim jak RSA.

Innym przykładem kryptosystemu, który może być złamany przez komputery kwantowe jest protokół Buchmanna-Williamsa, opierający się na problemie znalezienia rozwiązań równania Pella [2]. Problem polega na znalezieniu takiej pary liczb całkowitych (x, y) , że mając daną liczbę całkowitą d para ta spełnia równanie $x^2 - dy^2 = 1$. Najlepszy klasyczny algorytm rozwiązujący równanie Pella jest wykładniczo wolniejszy od najlepszego znanego algorytmu faktoryzacji, więc wydawać by się mogło, że kryptosystemy oparte na tym problemie są bezpieczniejsze. Zaprezentowano jednak algorytm kwantowy Hallgrene'a który jest w stanie rozwiązać równania Pella'a w czasie wielomianowym. Podobnie jest z kryptografią opartą na problemie logarytmu dyskretnego (Protokół Diffiego-Hellmana) i kryptografią krzywych eliptycznych.

1.2 Wstęp do kryptografii post-quantowej

W obliczu zagrożeń, jakie niesie ze sobą potencjalne zbudowanie komputera kwantowego konieczne jest ponowne przeanalizowanie kryptosystemów, które są powszechnie stosowane. W ten sposób narodził się koncept kryptografii postkwantowej - kryptografii, która oprze się nawet atakom z wykorzystaniem komputera kwantowego [3].

Jak się okazuje, jest kilka systemów kryptograficznych, które są uważane za zdolne do zapewnienia bezpieczeństwa nawet jeśli zaatakuje się je komputerem kwantowym. Kryptografię postkwantową można więc podzielić na:

- opartą na funkcjach skrótu, o której będzie mowa w tej pracy;
- opartą o kody korekcyjne, jak np.: system klucza publicznego McEliece'a;
- opartą na kratkach np.: NTRU;
- opartą o isogenię np.: SIKE.

Jak dotąd nikt nie znalazł sposobu, jak wykorzystać w efektywny sposób komputery kwantowe do złamania tych rodzajów kryptosystemów. Za zagrożenie możnaby uważać algorytm przeszukiwania zaproponowany przez Grovera [4], który z dużym

prawdopodobieństwem znajduje wyróżniony element w zbiorze N -elementowym, ale oferuje on jedynie kwadratowe przyspieszenie przeszukiwania względem algorytmów klasycznych - złożoność klasycznych algorytmów przeszukiwania jest rzędu $O(N)$, natomiast algorytmu Grovera - $O(\sqrt{N})$. Algorytmowi temu można więc oprzeć się, stosując np.: dłuższe klucze kryptograficzne.

Prawdopodobne jest, że kiedyś zostaną wymyślone lepsze algorytmy, które sprawią, że wyżej wymienione systemy przestaną być odporne na ataki z użyciem komputera kwantowego. Kryptoanalitycy stale pracują nad atakami przeciw każdemu rodzajowi kryptosystemów. Systemy brane pod uwagę w kryptografii post-quantowej odznaczają się tym, że przez cały czas trwania badań nad ich złamaniem, nadal nie udało się tego osiągnąć. Daje to poczucie, że istnieje jakieś wyjście dla kryptografii, nawet w sytuacji zbudowania komputera kwantowego.

1.3 Wykorzystanie funkcji skrótu w kryptografii post-quantowej

Funkcją skrótu [5] nazywamy funkcję, która dowolnie dużej wiadomości przyporządkowuje nieodwracalny skrót, wartość o stałym rozmiarze. Pozwalają na weryfikację oraz ochronę przed naruszeniem integralności danych.

Doskonała funkcja skrótu jest funkcją różnowartościową, jednakże w rzeczywistości jest to niemożliwe do osiągnięcia, ponieważ nieskończony zbiór wartości, jakie można przekazać do funkcji skrótu nie może zostać zamknięty w zbiorze możliwych skrótów o określonej długości. Za funkcje skrótu uznawane za bezpieczne do wykorzystania w kryptografii zalicza się funkcje spełniające następujące warunki:

1. Jednokierunkowość - brak możliwości odwrócenia funkcji na podstawie wartości skrótu. Zmiana pojedynczego bitu wiadomości początkowej powinna powodować zmianę średnio połowy bitów skrótu.
2. Słaba odporność na kolizje - znając wcześniej $h(m_1)$ brak możliwości odnalezienia takiej wiadomości m_2 różnej od m_1 , że $h(m_1) = h(m_2)$.
3. Silna odporność na kolizje - brak możliwości znalezienia takiej pary wiadomości m_1 oraz m_2 różniących się od siebie i spełniających warunek $h(m_1) = h(m_2)$, przy czym funkcja musi dodatkowo być słabo odporna na kolizje.

Przykładami funkcji skrótu są funkcje takie jak MD5 (obecnie nie jest bezpieczna - w 2004 roku zaprezentowano sposób na generowanie kolizji) czy też rodzina funkcji SHA zaprojektowanych przez NSA.

Kryptografia oparta na funkcjach skrótu obejmuje ogół algorytmów wykorzystujących je w swoim działaniu. Ich bezpieczeństwo zależy od bezkolizyjności funkcji skrótu w nich wykorzystanych. Nie stworzono jeszcze algorytmu kwantowego, który byłby w stanie w efektywny sposób radzić sobie z generowaniem kolizji w funkcjach uważanych obecnie za bezpieczne. Sprawia to, że algorytmy na nich oparte są dobrymi kandydatami do powszechnego wykorzystywania w post-quantowej erze.

W naszej pracy przedstawimy kilka konceptów, które mogą znaleźć szersze zastosowanie w przyszłości.

2 Systemy podpisów jednorazowych

Podpisy cyfrowe są kluczową technologią, zapewniającą bezpieczeństwo systemów informatycznych. Są powszechnie stosowane w protokołach identyfikujących i uwierzytelniających, więc ważne jest, aby były one bezpieczne pod każdym względem. Obecnie stosowanymi algorytmami podpisów cyfrowych są np.: RSA, DSA, czy ECDSA. Nie są one odporne na ataki z wykorzystaniem komputerów kwantowych, ponieważ ich bezpieczeństwo polega na problemie faktoryzacji liczb całkowitych.

Bezpieczeństwo systemów podpisów cyfrowych opartych na funkcjach skrótu zależy od tego, czy użyta funkcja haszująca jest odporna na kolizje: nie powinno być możliwe wygenerowanie dwóch dokumentów z takim samym podpisem cyfrowym. Takie systemy są brane pod uwagę w kryptografii postkwantowej, gdyż ich bezpieczeństwo nie jest uzależnione od problemów matematycznych.

2.1 System Lamporta

W 1979 roku Leslie Lamport zaproponował system jednorazowych podpisów cyfrowych, polegający na skonstruowaniu podpisu cyfrowego na podstawie funkcji jednokierunkowej [6]. System ten pozwala na podpisanie jednej wiadomości jednym podpisem.

System Lamporta wymaga bezpiecznej funkcji haszującej oraz generatora liczb losowych. Przykładową bezpieczną funkcją haszującą w poniższym przykładzie będzie SHA256. Załóżmy, że osoba A chce wysłać wiadomość do osoby B.

- A musi wygenerować parę kluczy: (pk, sk) - klucz publiczny oraz klucz prywatny, tajny.
- Kluczem prywatnym będzie 256 par losowych liczb r_x , każda o długości 256 bitów:

$$sk = \begin{bmatrix} r_1^0 & r_2^0 & \cdots & r_{255}^0 & r_{256}^0 \\ r_1^1 & r_2^1 & \cdots & r_{255}^1 & r_{256}^1 \end{bmatrix} \quad (1)$$

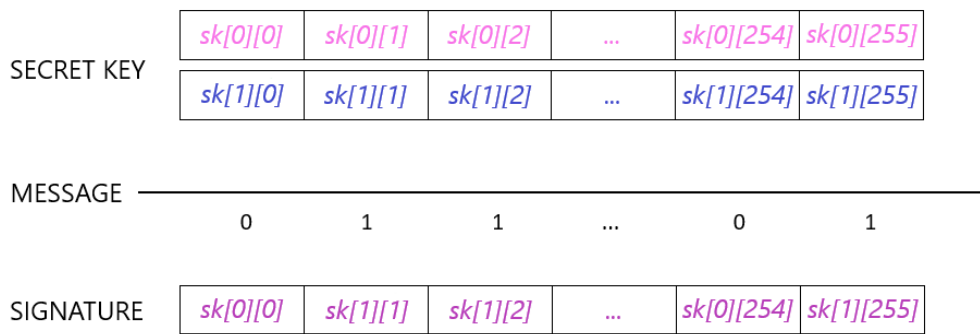
Klucz prywatny posiada tylko A.

- Klucz publiczny powstaje poprzez użycie na każdej z liczb w kluczu prywatnym funkcji haszującej $h(r)$:

$$pk = \begin{bmatrix} h(r_1^0) & h(r_2^0) & \cdots & h(r_{254}^0) & h(r_{255}^0) \\ h(r_1^1) & h(r_2^1) & \cdots & h(r_{254}^1) & h(r_{255}^1) \end{bmatrix} \quad (2)$$

Ten klucz publiczny zostaje przesłany do osoby B, aby mogła ona zweryfikować, czy wiadomość faktycznie pochodzi od A.

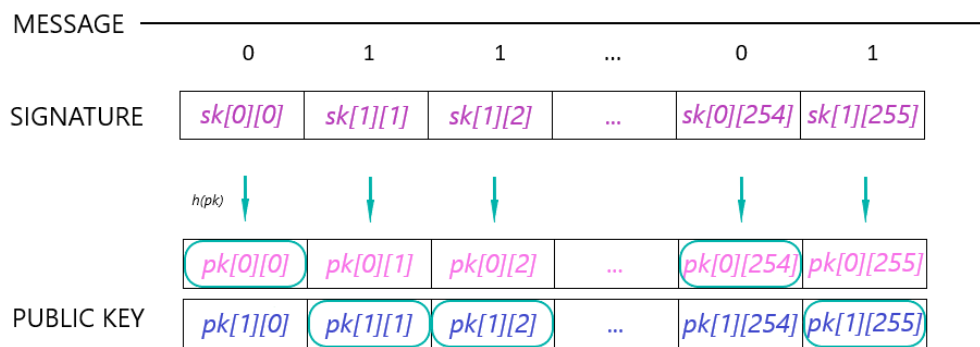
- Aby podpisać wiadomość m , A używa funkcji haszującej $h(m)$, aby otrzymać jej skrót o określonej długości (w tym przykładzie 256 bitów). Następnie, dla każdego bitu skrótu wiadomości, A wybiera odpowiadającą mu wartość z pary w jej kluczu prywatnym - jeśli bit jest równy 0, to wybiera pierwszą liczbę z pary, jeśli jest równy 1 - drugą:



Rysunek 1: Schemat generowania podpisu w systemie Lamporta.

Tak wygenerowany podpis jest rozsyłany razem z wiadomością.

- Aby zweryfikować podpis s , B musi obliczyć skrót z wiadomości $h(m)$, a następnie, dla każdego bitu b w skrócie wiadomości $h(m)$ obliczyć $h(s_i^b)$ i sprawdzić, czy otrzymana wartość jest zgodna z wartością odpowiedniego elementu w kluczu publicznym:



Rysunek 2: Schemat weryfikowania podpisu w systemie Lamporta.

Jeśli wszystkie elementy będą się zgadzać z kluczem publicznym, to B może być pewny, że wiadomość pochodzi od A.

Podpisy utworzone w ten sposób opierają swoje bezpieczeństwo na tym, jak odporna na kolizje jest funkcja haszująca. Jeśli funkcja jest odporna na kolizje, to osoba C, chcąc podrobić podpis, stoi przed niemożliwym do wykonania zadaniem znalezienia takich wiadomości dla każdego elementu klucza publicznego (w powyższym przykładzie 512).

System ten może posłużyć jedynie do podpisów jednorazowych, ponieważ już przy pierwszej wiadomości poznajemy połowę elementów klucza prywatnego - są one przekazywane w podpisie. Gdyby podpisać więcej wiadomości przy pomocy tego samego klucza, dość szybko poznalibyśmy wszystkie jego wartości.

2.2 System Winternitza

O ile system Lamporta w wydajny sposób generuje klucze oraz podpis, tak ich rozmiary są dosyć duże - dla funkcji haszującej produkującej 256-bitowe skróty, każdy z kluczy składa się z 256 par takich skrótów, co daje aż 131 072 bity, a podpis ma długość 65 536 bitów.

Relatywnie mniejsze rozmiary kluczy oraz podpisu używa system zaproponowany przez Roberta Winternitza w 1979 roku [7]. W przypadku tego systemu, przy użyciu SHA256 jako funkcji haszującej, generowane są klucze oraz podpis o długości 32x256 bitów każdy. Generowanie podpisu odbywa się w następujący sposób:

- Osoba A, chcąc wysłać podpisaną wiadomość do osoby B, najpierw musi wygenerować parę kluczy: klucz prywatny i publiczny.
- Kluczem prywatnym są 32 256-bitowe, losowe liczby:

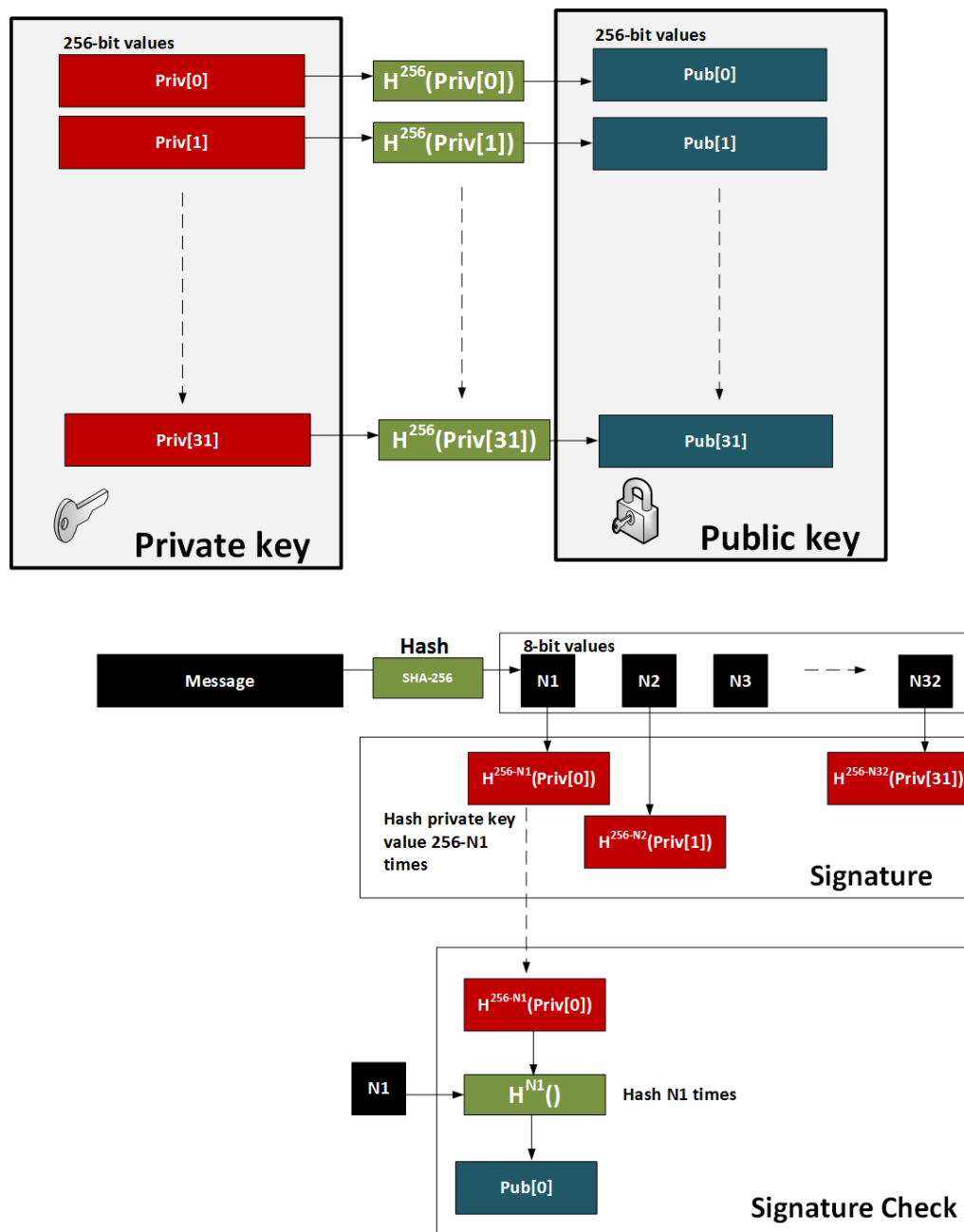
$$priv = [r_0 \ r_1 \ \dots \ r_{30} \ r_{31}] \quad (3)$$

- Klucz publiczny tworzony jest poprzez 256-krotne użycie funkcji haszującej $h(r_i)$ na każdym z 32 elementów klucza prywatnego:

$$pub = [h^{256}(r_0) \ h^{256}(r_1) \ \dots \ h^{256}(r_{30}) \ h^{256}(r_{31})] \quad (4)$$

Ten klucz publiczny zostaje przesłany do osoby B, aby mogła ona zweryfikować, czy wiadomość faktycznie pochodzi od A.

- Aby wygenerować podpis do wiadomości m , A oblicza $h(m)$, a następnie dzieli otrzymany skrót na 32 ośmiobitowe części. Każdą z 32 części klucza prywatnego A haszuje 256 - N razy, gdzie N jest wartością zawartą w odpowiadającym fragmencie $h(m)$. Przykładowo: jeśli mamy 8 bitów o postaci 10001101, co po zamianie na system dziesiętny daje liczbę 141, to $h(priv_1)$ wykona się 256 - 141 = 115 razy. Następnie wiadomość jest przesyłana do B wraz z wygenerowanym podpisem.
- Aby zweryfikować podpis, B oblicza $h(m)$, również dzieli otrzymany skrót na 8-bitowe kawałki, a następnie haszuje odpowiedni element podpisu N razy - w ten sposób otrzymuje klucz prywatny po 256 rundach funkcji haszującej. Na koniec B sprawdza, czy otrzymane wartości zgadzają się z wartościami w kluczu publicznym.



Rysunek 3: Schemat przedstawiający działanie systemu podpisów jednorazowych Winternitza.
 Źródło: <https://asecuritysite.com/encryption/wint>

3 System podpisów cyfrowych z wykorzystaniem drzew Merkle'a

Zaprezentowane systemy Lamporta oraz Winternitza można w praktyce wykorzystać tylko do podpisów jednorazowych, w miarę podpisywania kolejnych wiadomości zostaje ujawnione coraz więcej bitów klucza prywatnego, co w efekcie szybko prowadzi do jego kompromitacji. Rozwiązaniem problemu podpisywania wielu wiadomości są stosowane od dawna drzewa skrótów, znane też jako drzewa Merkle'a od nazwiska Ralpha Merkle'a, który opatentował je w 1979 [8]. W założeniu struktura ta pozwala na wygenerowanie jednego klucza publicznego, którym można zweryfikować wiele wiadomości zaszyfrowanych jednorazowymi kluczami prywatnymi.

3.1 Generowanie par kluczy

Niech osoba A będzie podpisującym wiadomość, a B weryfikującym. Chcąc wygenerować parę kluczy, A zaczyna od wybrania liczby H reprezentującej wysokość drzewa takiej, że:

$$H \in \mathbb{N}, H \geq 2 \quad (5)$$

Wygenerowana para kluczy dla drzewa o wysokości H pozwoli na podpisanie 2^H wiadomości, każda z nich będzie mogła zostać zweryfikowana tym samym kluczem publicznym.

Następnie A generuje 2^H par jednorazowych kluczy (w oryginalnej specyfikacji klucze wg. systemu Lamport'a) takich, że:

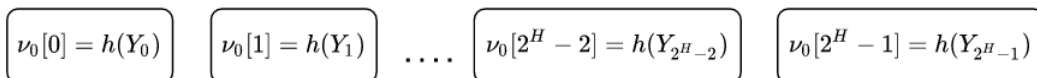
$$\begin{aligned} pairs = [(X_0, Y_0), (X_1, Y_1) \dots (X_j, Y_j)] \\ 0 \leq j < 2^H \end{aligned} \quad (6)$$

gdzie: X_j – klucz prywatny,
 Y_j – klucz publiczny.

Niech $h()$ będzie kryptograficzną funkcją haszującą. Przy jej wykorzystaniu A tworzy liście drzewa binarnego według wzoru:

$$\begin{aligned} \nu_0[j] = h(Y_j) \\ 0 \leq j < 2^H \end{aligned} \quad (7)$$

gdzie: ν – węzeł drzewa binarnego,
 H – wysokość drzewa.


$$\boxed{\nu_0[0] = h(Y_0)} \quad \boxed{\nu_0[1] = h(Y_1)} \quad \dots \quad \boxed{\nu_0[2^H - 2] = h(Y_{2^H - 2})} \quad \boxed{\nu_0[2^H - 1] = h(Y_{2^H - 1})}$$

Rysunek 4: Liście drzewa binarnego powstałe z kluczy publicznych

Kolejne poziomy drzewa binarnego powstają według zasady: każdy rodzic jest wynikiem haszowania konkatenacji swoich dzieci (lewe z prawym). Otrzymany korzeń drzewa jest kluczem publicznym, przy pomocy którego można zweryfikować każdą wiadomość podpisaną jednym z kluczy prywatnych $X_0, X_1 \dots X_{2^H-1}$. W ogólności dowolny węzeł w drzewie można opisać wzorem:

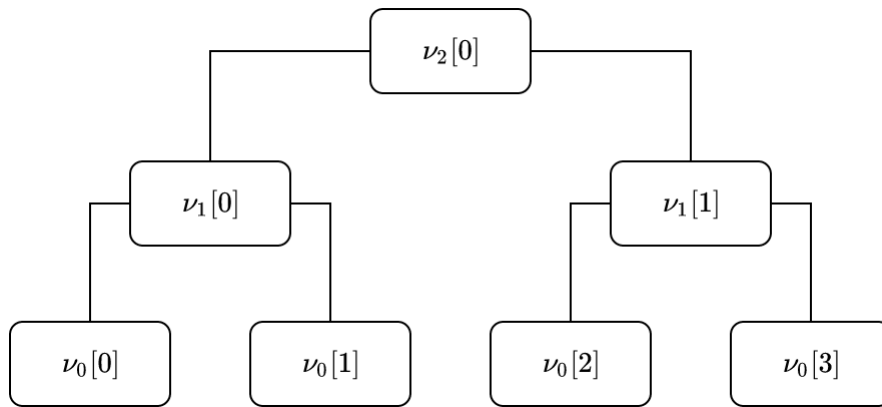
$$\nu_i[j] = h(\nu_{i-1}[2j] \parallel \nu_{i-1}[2j+1]) \quad (8)$$

$$1 \leq i \leq H$$

$$0 \leq j \leq 2^{H-i}$$

gdzie: i – wysokość węzła,

\parallel – konkatenacja.



Rysunek 5: Struktura drzewa Merkla o wysokości $H=2$

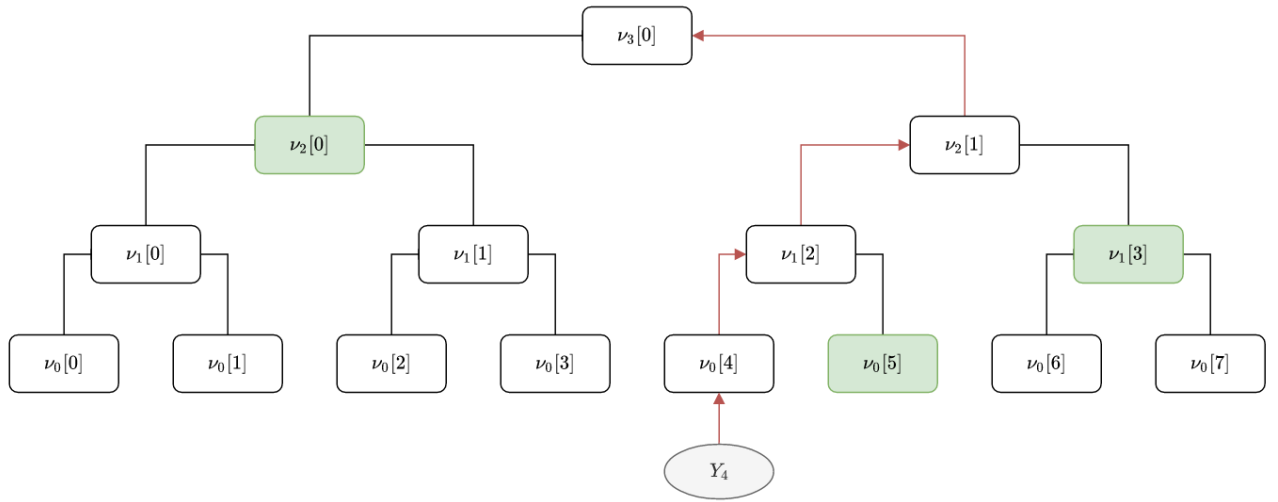
Ostatecznie A otrzymuje parę klucz prywatny:klucz publiczny w formie:

$$[X_0, X_1 \dots X_{2^H-1}] : \nu_H[0] \quad (9)$$

Widoczne jest zatem, że wygenerowanie pary kluczy wg. schematu Merkla wymaga wyznaczenia 2^H par kluczy jednorazowych Lamporta oraz wyznaczenia wartości funkcji haszującej $2^{H+1} - 1$ razy.

3.2 Ścieżki uwierzytelniające

Przed przejściem do generowania i weryfikowania podpisów w systemie drzew Merkla kluczowe jest zapoznanie się z koncepcją ścieżki uwierzytelniającej. Ideą ścieżki jest możliwość zweryfikowania obecności dowolnego węzła w drzewie bez znajomości całej jego struktury. Ponieważ węzły-rodzice tworzone są jako skróty sum swoich dzieci, to jest możliwe wyznaczenie puli węzłów $A_s = (a_0, \dots, a_{H-1})$ takiej, że licząc kolejne skróty ich sum odbiorca jest w stanie uzyskać identyczny skrót końcowy ze skrótem całego drzewa (będącym znanym kluczem publicznym). Jeżeli zastosowana funkcja skrótu jest co najmniej słabo odporna na kolizje to zgodność tych skrótów implikuje obecność weryfikowanego klucza w drzewie.



Rysunek 6: Ścieżka uwierzytelniająca dla elementu Y_4 w drzewie o $H=3$

Rozpatrzmy przykład na Rys. 6. W celu potwierdzenia autentyczności Y_4 , A musi wybrać węzły takie, że obliczanie skrótów ich kolejnych sum doprowadzi do wyznaczenia $\nu_3[0]$. Łatwo zauważyć, że na każdy poziom drzewa konieczny będzie dokładnie jeden taki **węzeł**. Bezpośrednio z tego wynika również wniosek, że dla drzewa o wysokości H ścieżka uwierzytelniająca będzie miała długość H . W ogólności wyznaczanie ścieżki można sprowadzić do rekurencji:

1. Dodaj węzeł bliźniaka do listy
2. Przejdź do rodzica
3. Czy jestem w korzeniu?
 - tak - zakończ i zwróć listę
 - nie - kontynuuj

Osoba A chcąc przesłać Y_4 i umożliwić jego weryfikację, poza samym Y_4 musi udostępnić także klucz publiczny całego drzewa $\nu_H[0]$ (czyli jego korzeń), ścieżkę uwierzytelniającą A_s oraz indeks n weryfikowanego elementu.

Indeks n jest konieczny w procesie rekonstruowania ścieżki przez B do określenia kierunku konkatencji (która nie jest przemienne, zatem $A||B \neq B||A$). W ogólności rekonstrukcję ścieżki można przedstawić równaniem:

$$p_i = \begin{cases} h(a_{i-1} || p_{i-1}) & \text{jeżeli } n/2^{i-1} \equiv 1 \mod 2 \\ h(p_{i-1} || a_{i-1}) & \text{jeżeli } n/2^{i-1} \equiv 0 \mod 2 \end{cases} \quad (10)$$

dla $i = 1, \dots, H$

gdzie: p_i – rekonstruowana ścieżka,
 a_i – otrzymana ścieżka,
 i – wysokość węzła,
 $||$ – konkatencja.

Proces weryfikacji przez osobę B został zaznaczony kolorem czerwonym:

1. Wyznaczenie skrótu Y_4
2. Dla każdego elementu ścieżki A_s :
 - a) Konkatenacja wynikowego skrótu z elementem ścieżki zgodnie z wzorem 10
 - b) Wyznaczenie skrótu konkatenacji

Jeżeli uzyskany wynik jest identyczny z kluczem publicznym, B może mieć pewność, że Y_4 jest częścią drzewa reprezentowanego kluczem publicznym $\nu_H[0]$.

3.3 Generowanie podpisu

W celu wygenerowania podpisu A musi podjąć następujące kroki:

1. Wyznaczenie skrótu d podpisywanej wiadomości M
2. Wygenerowanie podpisu jednorazowego systemem Lamporta σ_{OTS} z wykorzystaniem n -tego klucza jednorazowego X_n , takiego, że $n \in 0, 1, \dots, 2^H - 1$
3. Wyznaczenie ścieżki uwierzytelniającej A_n w postaci listy węzłów a koniecznych do weryfikacji autentyczności klucza Y_n .

Ostatecznie n -tym podpisem jest struktura:

$$\sigma_n = (n, \sigma_{OTS}, Y_n, A_n) \quad (11)$$

gdzie: σ_n – podpis wiadomości,
 n – numer wiadomości.

Ważna uwaga: A musi zapewnić sobie możliwość kontrolowania które klucze jednorazowe w drzewie zostały już wykorzystane. Dobrym pomysłem jest sekwencyjne wykorzystywanie kluczy i zapisywanie numeru ostatniej wiadomości. Po wykorzystaniu całej puli kluczy A musi wygenerować nowe drzewo.

3.4 Weryfikowanie podpisu

Przypomnijmy - odbiorca B otrzymuje:

- Wiadomość M
- Podpis jednorazowy systemem Lamporta σ_{OTS}
- Klucz publiczny w systemie Lamporta Y_n
- Numer klucza Y_n w drzewie Merkla n
- Ścieżkę weryfikacyjną A_n
- Klucz publiczny Merkla $\nu_H[0]$

Proces weryfikacji podpisu wiadomości przez B jest dwuetapowy:

1. Weryfikacja podpisu jednorazowego przy wykorzystaniu klucza Y_n zgodnie z 2.1
2. Weryfikacja autentyczności klucza Y_n przy wykorzystaniu ścieżki uwierzytelniającej w drzewie Merkla według 3.2

Weryfikacja jest uznana za udaną wtedy i tylko wtedy kiedy otrzymany skrót p_h jest równy kluczowi publicznemu.

4 Podsumowanie

W pracy zaprezentowane zostały obecne metody kryptografii postkwantowej opartej na funkcjach skrótów, scharakteryzowano i opisano ich implementację wraz z przykładami zawartymi w dołączonych plikach. Na podstawie przeprowadzonych obserwacji mieliśmy szansę przekonać się o zaskakującej sile funkcji skrótów w obliczu kryptoanalizy z wykorzystaniem komputera kwantowego, która od lat opiewana jest jako "pogromca" kryptografii jaką znamy dziś. O ile jest w tych słowach wiele racji, o tyle kryptografia jak każda inna dziedzina rozwija się bardzo dynamicznie. Już teraz można zauważyć, że metody kryptograficzne dostosowują się do nadchodzącej rewolucji kwantowej a kryptografowie nie zamierzają na tym poprzestać. Kryptografia przechodzi nieustanny rozwój, dlatego warto śledzić jego przebieg i szukać nowych metod, które mogą otworzyć nowe możliwości dla kryptografii post-quantowej.

Bibliografia

- [1] *Algorytm faktoryzacji Shora*, URL: https://pl.wikipedia.org/wiki/Algorytm_faktoryzacji_Shora, (dostęp: 16.01.2022)
- [2] *Równanie Pella*, URL: https://pl.wikipedia.org/wiki/R%C3%B3wnanie_Pella, (dostęp: 16.01.2022)
- [3] Erik Dahmen Daniel J. Bernstein Johannes Buchmann: *Post-Quantum Cryptography*, Springer, 2009
- [4] *Grover's algorithm*, URL: https://en.wikipedia.org/wiki/Grover%27s_algorithm, (dostęp: 18.01.2022)
- [5] *Funkcja skrótu*, URL: https://pl.wikipedia.org/wiki/Funkcja_skr%C3%B3tu, (dostęp: 18.01.2022)
- [6] Leslie Lamport: *Constructing Digital Signatures from a One Way Function*, tech. rep., SRI International, 1979
- [7] Ralph Merkle: *A Certified Digital Signature*, tech. rep., Xerox PARC, 1979
- [8] *Merkle Tree*, URL: https://en.wikipedia.org/wiki/Merkle_tree, (dostęp: 16.01.2022)