Identyfikacja błędnych podpisów w kolekcjach

Józef Pieprzyk
Jarosław Pastuszak

Weryfikacja poprawności podpisów cyfrowych jest obliczeniowo kosztowna. Aby ją zwiększyć, zamiast pojedynczych podpisów weryfikacji poddaje się kolekcje. Jeżeli wszystkie podpisy w kolekcji są poprawne, to cała kolekcja jest akceptowana. Pojawienie się błędnych podpisów w kolekcji powoduje, że weryfikacja jest błędna. Nie można odrzucić całej kolekcji, zachodzi więc konieczność identyfikacji błędnych podpisów w kolekcji. W artykule zdefiniowano metody identyfikacji błędnych podpisów. W szczególności określono weryfikacje typu „dziel i rządź", w których wejściowe kolekcje są dzielone na podkolekcje tak długo, aż końcowe błędne kolekcje zawierają pojedyncze podpisy. Opisano również weryfikator Hamminga identyfikujący jeden błędny podpis w kolekcji oraz uogólniono ten weryfikator do postaci dwupoziomowego weryfikatora, umożliwiającego identyfikacje dwóch błędnych podpisów. Podano też definicję ogólnego weryfikatora zdolnego do identyfikacji dowolnej liczby błędnych podpisów w kolekcjach.

weryfikacja podpisów cyfrowych, weryfikacja podpisów cyfrowych w kolekcjach, identyfikacja błędnych podpisów, kody identyfikujące błędne podpisy

Wprowadzenie

Podpis cyfrowy jest podstawowym narzędziem kryptograficznym, wykorzystywanym w procesie uwierzytelniania wiadomości. W odróżnieniu od tradycyjnych podpisów, podpisy cyfrowe jednocześnie identyfikują osobę podpisującą (a ściślej – jej prywatny klucz, służący do podpisywania wiadomości) oraz potwierdzają zawartość dokumentu, zatem podpis cyfrowy jest dla każdego podpisywanego dokumentu inny. Na podpis cyfrowy składają się więc dwa algorytmy: algorytm podpisu oraz algorytm weryfikacji.

Podpis cyfrowy jest generowany jednorazowo, jednak – z uwagi na upublicznienie klucza służącego do weryfikacji – weryfikacja takiego podpisu może odbywać się wielokrotnie. Rosnące znaczenie podpisu cyfrowego w elektronicznych systemach płatniczych (*e-commerce*) powoduje, iż rośnie potrzeba przyspieszenia procesu weryfikacji podpisów. Odpowiedzią na opisane zapotrzebowanie jest tzw. *batch verification* (w dalszej części artykułu tłumaczone jako **weryfikacja wsadowa**), czyli algorytmy weryfikowania dużych kolekcji podpisów cyfrowych „wyprodukowanych" za pomocą tego samego klucza prywatnego. Idea weryfikacji wsadowej została opisana w wielu publikacjach [1, 2, 4–6, 8].

Weryfikacja wsadowa kolekcji podpisów jest algorytmem probabilistycznym, w którym prawdopodobieństwo pomyłki (czyli zaakceptowania kolekcji zawierającej błędne podpisy) może zmniejszać się kosztem zwiększenia czasochłonności przetwarzania (weryfikacji) kolekcji.

Bellare, Garay oraz Rabin w [1] wskazali, iż istnieją trzy ogólne algorytmy (nazywane w dalszej części artykułu testami ogólnymi), wykorzystywane przy weryfikacji wsadowej. Wydajność tych algorytmów zależy w głównej mierze od rozmiarów weryfikowanej kolekcji. W przypadku weryfikacji pojedynczych podpisów ewentualny atakujący jest zmuszony złamać zastosowany schemat podpisów.

W sytuacji gdy do weryfikacji kolekcji podpisów stosuje się algorytm weryfikacji wsadowej, ewentualny atakujący może dodatkowo wykorzystać słabości tkwiące w samym teście weryfikującym. Gdy kolekcja podpisów pomyślnie przechodzi test weryfikujący, wszystkie podpisy są traktowane jako poprawne. W przeciwnym przypadku kolekcja jest odrzucana, niezależnie od liczby błędnych podpisów w kolekcji. W tej sytuacji weryfikujący musi zidentyfikować w odrzuconej kolekcji wszystkie błędne podpisy.

Identyfikacja błędnych podpisów w kolekcji przypomina znany problem 12 monet.

Dane jest 12 monet, 11 z nich ma równą wagę, a jedna z nich – inną, przy czym nie jest wiadomo, czy ta moneta jest lżejsza, czy też cięższa od pozostałych. Do dyspozycji jest także waga szalkowa, za pomocą której trzeba znaleźć uszkodzoną monetę. Należy znaleźć sekwencję testów (czyli ważeń), które doprowadzą do znalezienia uszkodzonej monety łącznie z określeniem, na czym defekt polega (moneta jest cięższa, czy może lżejsza od pozostałych) przy założeniu, że można użyć wagi trzykrotnie.

Podstawową różnicą między identyfikacją błędnych podpisów w kolekcjach o zanieczyszczeniu większym niż jeden błędny podpis a problemem 12 monet jest fakt, iż test wskazuje jedynie, że dana kolekcja jest zanieczyszczona (ma przynajmniej jeden błędny podpis), nie określa natomiast, w jaki sposób jest zanieczyszczona (czyli np. nie podaje liczby błędnych podpisów w kolekcji).

W dalszej części artykułu omówiono podstawowe metody identyfikacji błędnych podpisów w kolekcji.

Podstawowe typy weryfikatorów

Istnieją dwie homomorficzne operacje powszechnie używane przy podpisywaniu wiadomości: potęgowanie modularne przy ustalonej podstawie oraz potęgowanie RSA (gdzie wykładnik jest ustalony).

Zakłada się, iż potęgowanie modulo jest zdefiniowane dla pewnej grupy cyklicznej rzędu q, gdzie g jest generatorem grupy (podpis ElGamala jest schematem tego typu).

Dana jest pewna kolekcja wiadomości oraz ich podpisów:

$$x = \big((m_1, s_1), \ldots, (m_n, s_n)\big) .$$

Podpisy mogą zostać zweryfikowane po kolei według reguły:

$$g^{m_i} \stackrel{?}{=} s_i \quad \text{dla} \quad i = 1, \ldots, n .$$

Weryfikacja według tej reguły „kosztuje” n nodularnych operacji potęgowania.

W celu zmniejszenia liczby kosztownych operacji potęgowania oraz przyspieszenia procesu weryfikacji można użyć następującej formuły:

$$V_g(x) \equiv \left(g^{\sum_{i=1}^{n} m_i} \stackrel{?}{=} \prod_{i=1}^{n} s_i \right) .$$

Weryfikacja tym razem „kosztuje” jedno modularne potęgowanie oraz $n-1$ modularnych dodawań i mnożeń.

Warto rozważyć teraz potęgowanie RSA, gdzie moduł N jest iloczynem dwóch liczb pierwszych p oraz q. Tajnym kluczem podpisującego jest wtedy d, a publicznym – e. Seryjna weryfikacja kolekcji podpisów x ma postać:

$$s_i^e \stackrel{?}{=} m_i \quad \text{dla} \quad i = 1, \ldots, n$$

i kosztuje n operacji potęgowania.

Proces może być przyspieszony według formuły:

$$V_e(x) \equiv \left(\prod_{i=1}^{n} m_i \stackrel{?}{=} \left(\prod_{i=1}^{n} s_i \right)^e \right).$$

Weryfikacja kosztuje jedno potęgowanie oraz $2(n-1)$ modularnych mnożeń.

Sformułowanie testu ogólnego oraz weryfikatora naiwnego

Ogólnie weryfikator wsadowy jest to pewien probabilistyczny algorytm B, który dla kolekcji podpisów x oraz parametru bezpieczeństwa l:

- zwraca **„0”** zawsze, gdy wejściowa kolekcja jest poprawna (czyli wszystkie podpisy w kolekcji są prawidłowe);
- zwraca **„1”** z prawdopodobieństwem $1 - 2^{-l}$ w przypadku, gdy kolekcja wejściowa zawiera przynajmniej jeden błędny podpis.

Można to sformalizować, definiując **test ogólny**.

Definicja 1

Dana jest wejściowa kolekcja $x = \big((m_1, s_1), \ldots, (m_n, s_n)\big)$, l – parametr bezpieczeństwa. Test ogólny (*Generic Test* – GT) przebiega następująco.

1. Test zwraca „0”, gdy wejściowa kolekcja jest w całości poprawna.
2. Test zwraca „1”, gdy wejściowa kolekcja zawiera przynajmniej jeden błędny podpis, w takim przypadku test popełnia błąd z prawdopodobieństwem 2^{-l}.

Uniwersalny test zdefiniowano w pracy [1]. Można go użyć dla dowolnego homomorficznego schematu podpisów cyfrowych. W [1] test ten nazwano **testem podzbiorów losowych**.

Definicja 2

Dana jest wejściowa kolekcja $x = \big((m_1, s_1), \ldots, (m_n, s_n)\big)$ oraz parametr bezpieczeństwa l. Test uniwersalny (*Universal Test* – UT) trwa l rund. Podczas każdej rundy postępuje się w ten sam sposób.

1. Wybiera się losowy zbiór $T = \{t_1, \ldots, t_n\}$, gdzie każde t_i jest wybierane niezależnie i z takim samym prawdopodobieństwem ze zbioru $\{0, 1\}$.
2. Tworzy się podzbiór $x_T = \{(m_i, s_i) \,|\, t_i = 1\}$.

3. Uruchamia się test $V(x_T)$ ($V_g(x_T)$ lub $V_e(x_T)$; jeśli test akceptuje kolekcję, jest uruchamiana kolejna runda, w przeciwnym przypadku wejściowa kolekcja jest odrzucana.

Test uniwersalny służy do poprawnej weryfikacji w okolicznościach, kiedy do kolekcji mogą być wprowadzone podpisy manipulowane przez inteligentnego nieprzyjaciela, który na przykład wprowadza pary (m', s'), $(m'^{\{-1\}}, s'^{\{-1\}})$ fałszywych podpisów.

Warto zwrócić uwagę, że **test uniwersalny** wykrywa manipulacje tylko wtedy, gdy w losowo wybranej podkolekcji znaleziono jeden z fałszywych podpisów (lub kilka fałszywych, „nie kompensujących się" podpisów).

Inne podejście do weryfikacji, zwane **testem małych wykładników**, jest proponowane przez autorów pracy [1]. W teście tym wiadomości są mnożone przez stosunkowo małą liczbę przypadkową (wybieraną niezależnie dla każdego podpisu), a podpisy są podnoszone do potęgi. Wykrycie pary fałszywych podpisów jest możliwe, jeżeli wylosowane liczby dla pary podpisów są różne.

Definicja 3

Dana jest wejściowa kolekcja $x = \big((m_1, s_1), \ldots, (m_n, s_n)\big)$ oraz parametr bezpieczeństwa l. Test małych wykładników (*Small Exponent Test* – SE) przeprowadza się następująco.

1. Wybiera się losową kolekcję małych liczb naturalnych $e = \{e_1, \ldots, e_n\}$, gdzie $e_i < 2^l$.
2. Wejściowa kolekcja jest konwertowana do postaci:

$$x_e = \big((m_1 e_1, s_1^{e_1}), \ldots, (m_n e_n, s_n^{e_n})\big).$$

3. Uruchamia się test $V_g(x_e)$; jeśli test akceptuje kolekcję, jest akceptowana kolekcja x, w przeciwnym przypadku wejściowa kolekcja jest odrzucana.

Problem identyfikacji złych podpisów – weryfikator naiwny

Jeśli którykolwiek z opisanych poprzednio testów zwróci „1" (czyli badana kolekcja zawiera przynajmniej jeden błędny podpis), pojawia się problem identyfikacji błędnych podpisów w kolekcji.

Najbardziej podstawową metodą jest sprawdzenie każdego podpisu osobno. Weryfikator taki nosi nazwę **weryfikatora naiwnego**.

Definicja 4. Weryfikator naiwny (*Naive Verifier* – NV)

Dana jest wejściowa kolekcja $x = \big((m_1, s_1), \ldots, (m_n, s_n)\big)$.

1. Uruchamia się test $\mathrm{GT}(x, n)$. Jeśli $\mathrm{GT}(x, n) = 0$, to kolekcja x jest akceptowana w całości, a weryfikator kończy działanie. Jeśli $\mathrm{GT}(x, n) = 1$, to od $i = 1$ do $i = n$ weryfikator uruchamia $\mathrm{GT}(x_i, 1)$; jeśli $\mathrm{GT}(x_i, 1) = 1$, zapamiętuje x_i, gdzie: $x_i = (m_i, s_i)$.
2. Test zwraca wszystkie zapamiętane podpisy w postaci listy $\mathrm{NV}(x)$.

Weryfikator „dziel i rządź"

Weryfikator „dziel i rządź" jest zdefiniowany jako funkcja rekurencyjna.

Definicja 5. Weryfikator „dziel i rządź” (*Divide-and-Conquer Verifier* – DCV)

Dana jest wejściowa kolekcja $x = \big((m_1, s_1), \ldots, (m_n, s_n)\big)$ z $n = 2^k$ podpisami. Weryfikator DCV_α jest zdefiniowany następująco.

1. Warunek stopu. Jeśli instancja wejściowa zawiera $n = 1$ podpis, wtedy jest uruchamiany test ogólny $\mathrm{GT}(x, 1)$. Jeśli $\mathrm{GT}(x, 1) = 0$, weryfikator zwraca „0”, w przeciwnym przypadku zwraca błędny podpis i kończy działanie.

2. Jeśli instancja wejściowa zawiera $n > 1$ podpisów, wtedy jest uruchamiany test ogólny $\mathrm{GT}(x, n)$. Jeśli $\mathrm{GT}(x, n) = 0$, weryfikator zwraca „0” i kończy działanie, w przeciwnym przypadku weryfikator przechodzi do kroku rekurencyjnego.

3. Krok rekurencyjny. Wejściowa kolekcja jest dzielona na α podkolekcji $(x_1, \ldots, x_a)$, zawierających n/α podpisów każda, przy czym podział jest wykonany losowo. Dla każdej z α podkolekcji jest uruchamiany weryfikator $\mathrm{DCV}_\alpha(x_1, n/\alpha), \ldots, \mathrm{DCV}_\alpha(x_\alpha, n/\alpha)$.

Wpływ stopnia zanieczyszczenia kolekcji na efektywność identyfikacji złych podpisów

Wydajność weryfikatora DCV jest mierzona liczbą testów ogólnych GT wykonywanych podczas procesu weryfikacji. Najgorszy przypadek (czyli maksymalizujący liczbę testów GT) występuje w sytuacji, gdy wejściowa kolekcja jest w całości błędna.

Maksymalna liczba testów dla kolekcji o liczności $n = 2^k$ wyraża się zatem wzorem:

$$\#\max(\mathrm{DCV}_\alpha, n) = \sum_{i=0}^{k} \alpha^i = \frac{\alpha^{(k+1)} - 1}{\alpha - 1} = \frac{n\alpha - 1}{\alpha - 1}.$$

Łatwo zauważyć, że w sytuacji gdy wejściowa kolekcja jest „zanieczyszczona” w bardzo wysokim stopniu, powinno wybierać się duże α. Z wcześniejszego wzoru widać również, że w sytuacji gdy $\alpha = n$, weryfikator DCV staje się równoważny weryfikatorowi naiwnemu NV, który zawsze potrzebuje $n + 1$ testów.

Interesujące jest pytanie, przy jakim stopniu zanieczyszczenia kolekcji wejściowej weryfikator DCV_2 jest efektywniejszy (czyli wykorzystuje mniejszą liczbę testów ogólnych) od weryfikatora naiwnego.

Proste rozumowanie prowadzi do następującego stwierdzenia.

Stwierdzenie 1

Weryfikator DCV_2 jest efektywniejszy od weryfikatora naiwnego NV (czyli wykorzystuje mniejszą liczbę testów ogólnych), jeśli wejściowa kolekcja o liczności 2^k zawiera mniej niż $2^{(k-3)}$ błędnych podpisów.

Tabl. 1. Związek między optymalnym parametrem α i stopniem zanieczyszczenia kolekcji wejściowej

Liczność kolekcji n	Liczba błędnych podpisów t	Optymalny parametr α
128	1	2,4
	2,4	4,8
	8	32
	16	32,64
	32	128
256	1	2,4
	2,4,8	4
	16	8
	32,64	64
512	1	2,4
	2,4,8,16	4
	32,64	128
1024	1,2	2,4
	4,8	4
	16,32	4,8
	64	256
	128	512
2048	1	2,4
	2,4,8,16,32,64	4
	128,256	512
4096	1	2,4
	2,4,8,16,32,64,128	4
	256,512	1024

Warto zwrócić uwagę, że w ten sposób porównano weryfikator DCV_2 z weryfikatorem DCV_n (równoważnym weryfikatorowi NV). Ciekawe jest porównanie weryfikatorów DCV dla różnych parametrów podziału kolekcji wejściowej na podkolekcje. W tablicy 1 zaprezentowano opisane zależności, otrzymane za pomocą symulacji komputerowej.

Liczba testów potrzebnych do identyfikacji t błędnych podpisów

W tej części artykułu omówiono wyniki obrazujące rozkład prawdopodobieństwa liczby testów potrzebnych do identyfikacji pewnej liczby t błędnych podpisów.

Dla uproszczenia zapisów, liczba testów ogólnych – potrzebnych weryfikatorowi DCV do identyfikacji t błędnych podpisów w kolekcji liczącej n podpisów, przy założeniu podziału na α podkolekcji – będzie przedstawiona jako:

$$N_\alpha(t, n) = \#(\mathrm{DCV}_\alpha, n, t).$$

Stwierdzenie 2

Dany jest weryfikator DCV przy ustalonym $\alpha = 2$. Jeśli wejściowa kolekcja o liczności $n = 2^k$ jest zanieczyszczona przez t błędnych podpisów, wtedy liczba $N_2(t, n)$ koniecznych testów ogólnych GT jest zmienną losową o następującym rozkładzie.

1. Dla $t = 1$:

$$P\left(N_2(1, 2^k) = 2k + 1\right) = 1.$$

2. Dla $t = 2$:

$$P\left(N_2(2, 2^k) = 4k - 2j - 1\right) = \frac{n}{n-1}\,\frac{1}{2^{j+1}}$$

dla $j = 0, 1, \ldots, k-1$.

3. Dla $t = 3$:

$$P\left(N_2(3, 2^k) = 6k - 4i - 2j - 5\right) = \frac{3n^2}{(n-1)(n-2)}\,\frac{1}{2^{2i+j+3}}$$

dla $i = 0, 1, \ldots, k-1$ oraz $j = 0, 1, \ldots, k-i-2$.

Znając rozkład prawdopodobieństwa, wynikający ze stwierdzenia 2, łatwo jest znaleźć wartość oczekiwaną liczby testów. W tablicy 2 zobrazowano, obliczone za pomocą symulacji komputerowej, wartości dla $t = 0, 1, \ldots, 16$.

Tabl. 2. Wartość oczekiwana liczby testów GT potrzebnych do identyfikacji t błędnych podpisów w kolekcjach o liczności n

t	$n = 16$	$n = 32$	$n = 64$	$n = 128$	$n = 256$	$n = 512$	$n = 1024$
1	9,0	11,0	13,0	15,0	17,0	19,0	21,0
2	13,5	17,3	21,2	25,1	29,1	33,0	37,0
3	17,1	22,5	28,2	34,0	39,8	45,8	51,7
4	19,9	26,9	34,3	41,9	49,7	57,6	65,5
5	22,2	30,8	39,8	49,3	58,9	68,7	78,5
6	24,1	34,2	44,9	56,0	67,5	79,2	91,0
7	25,6	37,2	49,5	62,4	75,7	89,3	103,0
8	27,0	39,9	53,8	68,4	83,5	99,0	114,7
9	28,0	42,4	57,8	74,1	91,0	108,3	125,9
10	28,9	44,6	61,5	79,5	98,2	117,4	136,9
11	29,6	46,6	65,1	84,7	105,1	126,1	147,6
12	30,2	48,5	68,4	89,6	111,8	134,7	158,0
13	30,6	50,2	71,5	94,3	118,3	143,0	168,2
14	30,9	51,7	74,5	98,9	124,5	151,1	178,2
15	31,0	53,1	77,3	103,3	130,6	159,0	188,0
16	31,0	54,4	80,0	107,5	136,6	166,7	197,6

Optymalizacja weryfikatora „dziel i rządź”

Warto podkreślić, że weryfikator DCV_2 może być zoptymalizowany (tzn. można zmniejszyć liczbę koniecznych testów GT), jeśli wiadomo, że liczba błędnych podpisów wynosi dokładnie $t = 1$. Wtedy w każdym kroku algorytmu wystarczy sprawdzić tylko jedną podkolekcję (jeśli pierwsza jest „czysta”, druga musi zawierać jeden błędny podpis i na odwrót). Prowadzi to do redukcji liczby testów z $N_2(1, 2k) = 2k+1$ do $k+1$.

Opisana obserwacja umożliwia również definicję szybkiego weryfikatora DCV.

Definicja 6. Szybki weryfikator „dziel i rządź” (Fast DCV)

Dana jest wejściowa kolekcja $x = \big((m_1, s_1), \ldots, (m_n, s_n)\big)$ z $n = 2^k$ podpisami. Weryfikator $\mathrm{FastDCV}_\alpha$ jest zdefiniowany następująco.

1. Warunek stopu. Jeśli instancja wejściowa zawiera $n = 1$ podpis, wtedy jest uruchamiany test ogólny $\mathrm{GT}(x, 1)$. Jeśli $\mathrm{GT}(x, 1) = 0$, weryfikator zwraca „0”, w przeciwnym przypadku zwraca błędny podpis i kończy działanie.
2. Jeśli instancja wejściowa zawiera $n > 1$ podpisów, wtedy jest uruchamiany test ogólny $\mathrm{GT}(x, n)$. Jeśli $\mathrm{GT}(x, n) = 0$, weryfikator zwraca „0” i kończy działanie, w przeciwnym przypadku weryfikator przechodzi do kroku rekurencyjnego.
3. Krok rekurencyjny. Wejściowa kolekcja jest dzielona na α podkolekcji $(x_1, \ldots, x_a)$, zawierających n/α podpisów każda, przy czym podział jest wykonany losowo. Dla każdej z $\alpha - 1$ podkolekcji jest uruchamiany weryfikator $\mathrm{FastDCV}_\alpha\ (x_1, n/\alpha), \ldots, \mathrm{FastDCV}_{\alpha-1}, (x_{\alpha-1}, n/\alpha)$. Jeśli występuje przynajmniej jedna zanieczyszczona podkolekcja, jest uruchamiany test $\mathrm{FastDCV}_\alpha\,(x_\alpha, n/\alpha)$. W przeciwnym przypadku (wszystkie podkolekcje są czyste) jest uruchamiany weryfikator $\mathrm{FastDCV}_\alpha\,(x_\alpha, n/\alpha)$, w którym pomija się punkty 1 oraz 2 algorytmu (jeśli zawiera więcej niż jeden podpis) lub zwraca błędny podpis (jeśli zawiera jeden podpis) i kończy działanie.

Łatwo zauważyć, że szybki weryfikator DCV_2 potrzebuje około $(1,5k+1)$ testów (zamiast $2k+1$) do wykrycia jednego błędnego podpisu. W ogólności szybki weryfikator $\mathrm{DCV}\alpha$ potrzebuje (przy założeniu występowania pojedynczego błędnego podpisu) następującą liczbę testów:

$$\left(\left(\alpha - 1 + \frac{1}{\alpha}\right) k + 1\right).$$

Kolejnym rozwiązaniem, które zmniejsza liczbę koniecznych testów ogólnych GT, jest zmiana losowego podziału na α podkolekcji na podział ustalony oraz uważne zaprojektowanie testów GT.

Dla zilustrowania idei można założyć, że weryfikator DCV używa wcześniej zdefiniowanego testu $V_e(x)$, który z kolei dla wejściowej kolekcji musi obliczyć iloczyny wszystkich wiadomości m_i oraz podpisów s_i. Zanim zostanie uruchomiony test $V_e(x)$, tworzy się dwie tablice mnożeń (osobno dla wiadomości, osobno dla podpisów).

Na przykład, dla wejściowej kolekcji o liczności 16 – tablica mnożeń będzie miała następującą postać:

kolekcja wejściowa:	1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16
pierwszy poziom iloczynów:	(1,2),(3,4),(5,6),(7,8),(9,10),(11,12),(13,14),(15,16)
drugi poziom iloczynów:	(1,2,3,4),(5,6,7,8),(9,10,11,12),(13,14,15,16)
trzeci poziom iloczynów:	(1,2,3,4,5,6,7,8),(9,10,11,12,13,14,15,16)
czwarty poziom iloczynów:	(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16),

gdzie (i, j) oznacza iloczyn $m_i \, x \, m_j$. Wszystkie potrzebne mnożenia zostały w ten sposób obliczone i są przechowywane w tablicy. W celu uruchomienia testów $V_e(x)$ dla kolejnych podkolekcji wystarczy już tylko wykonać pojedyncze potęgowanie dla każdego wykonywanego testu ogólnego GT.

Weryfikatory oparte na kodach Hamminga

Można założyć, iż kolekcja wejściowa jest zanieczyszczona przez co najwyżej jeden błędny podpis, co może być prawdą, np. w przypadku pojawiających się błędów w trakcie transmisji. Zakłada się dalej, iż jest kolekcja wejściowa x o liczności $n = 2^k - 1$. W celu identyfikacji pojedynczego błędnego podpisu wystarczy zaprojektować kod Hamminga o bloku długości n z k równaniami błędów.

Niech zatem H będzie macierzą korygującą kodu Hamminga, zawierającą k wierszy oraz n kolumn. Jeśli macierz H ma postać:

$$H = \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_k \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 & \dots & 2^k - 1 \end{bmatrix},$$

gdzie $h_i = (h_{i,1}, \dots, h_{i,n})$ jest binarną tablicą o długości n i wadze 2^{k-1}, a liczby całkowite i w tablicy reprezentują kolumny, będące odpowiednikami binarnymi danej liczby. Należy podkreślić, iż tak zdefiniowany kod Hamminga umożliwia szybką identyfikację błędnego podpisu przez identyfikację pozycji, na której znajduje się błędny podpis (szczegółowe informacje można znaleźć np. w [4]).

Definicja 7. Weryfikator Hamminga (*Hamming Verifier* – HV)

Dana jest wejściowa kolekcja $x = \big((m_1, s_1), \dots, (m_n, s_n)\big)$ z $n = 2^k - 1$ podpisami dla pewnego całkowitego, dodatniego k.

1. Uruchamia się test ogólny $\mathrm{GT}(x, n)$. Jeśli $\mathrm{GT}(x, n) = 0$, weryfikator kończy działanie, w przeciwnym przypadku jest wykonywany kolejny krok.
2. Tworzy się k podinstancji:
$$x_1 = \{(m_j, s_j) \,|\, h_{i,j} = 1\}$$
dla $i = 1, \dots, k$, gdzie x_i jest podkolekcją, zbudowaną z elementów kolekcji x wybranych zawsze, gdy $h_{i,j} = 1$ (ignoruje się elementy, dla których $h_{i,j} = 0$).
3. Uruchamia się testy ogólne $\mathrm{GT}(x_i, 2^{k-1}) = \sigma_i$ dla $i = 1, \dots, k$, przy czym $\sigma_i = 0$ oznacza, że test akceptuje wejściową podkolekcję x_i, a w przeciwnym przypadku – odrzuca. Syndrom:
$$\sigma = (\sigma_1, \dots, \sigma_k)$$
identyfikuje pozycję błędnego podpisu.
4. Uruchamia się test ogólny na wejściowej kolekcji bez błędnego podpisu. Jeśli wejściowa kolekcja przechodzi test pomyślnie, weryfikator kończy działanie i zwraca numer pozycji błędnego podpisu, w przeciwnym przypadku działanie kończy się niepowodzeniem i weryfikator zwraca „1”.

Działanie weryfikatora Hamminga kończy się pomyślnie zawsze, gdy wejściowa kolekcja o liczności $2^k - 1$ zawiera co najwyżej jeden błędny podpis i potrzebuje do tego $k + 2$ testów ogólnych GT (jest

to prawie dokładnie tyle, ile potrzebuje weryfikator DCV_2 w sytuacji, gdy wiadomo, iż w wejściowej kolekcji znajduje się dokładnie jeden błędny podpis – wspomniano już o tym wcześniej, omawiając optymalizację weryfikatora „dziel i rządź").

Jeżeli działanie weryfikatora HV kończy się niepowodzeniem, są możliwe przynajmniej dwa sposoby dalszego postępowania.

1. Usuwa się z wejściowej kolekcji wszystkie zweryfikowane przez weryfikator HV poprawne podpisy (czyli takie, dla których $\sigma_i = 0$), a błędne podpisy identyfikuje się za pomocą weryfikatora DCV na pozostałości wejściowej kolekcji.
2. Używa się kodów BCH korygujących dwa błędy, co może być szczególnie atrakcyjne, jeśli będzie możliwe ponowne wykorzystanie wyników testów uzyskanych przez weryfikator HV. Ta konkluzja ułatwia zdefiniowanie dwupoziomowego kodu Hamminga.

Niestety nie ma możliwości bezpośredniego zastosowania kodów BCH do identyfikacji błędnych podpisów w kolekcji. Wynika to z faktu, iż błędy pojawiające się w kanałach transmisyjnych zachowują się zgodnie z zasadami operacji XOR (czyli np. dwa błędy znoszą się w równaniach błędu), natomiast błędne podpisy (a właściwie ich zakodowane indeksy) zachowują się zgodnie z zasadami binarnej operacji OR.

Należy zatem zdefiniować dwupoziomowy kod Hamminga. Zakłada się, iż jest kolekcja wejściowa x o liczności $n = 2^k - 2$ zanieczyszczona przez dwa błędne podpisy. Trzeba zdefiniować najpierw macierz korygującą kodu Hamminga:

$$H_1 = \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_k \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 & \dots & 2^k - 2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 1 \\ & \vdots & \dots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}.$$

Warto zwrócić uwagę, że przedstawiona macierz nie zawiera żadnej kolumny z samymi jedynkami.

Należy teraz w następujący sposób zdefiniować kolejną macierz dla uprzednio zdefiniowanej macierzy H_1 oraz $i, j = 1, \dots, k$:

$$H = \begin{bmatrix} H_1 \\ H_2 \end{bmatrix}$$

$$H_2(i\,j) = \overline{H_1(i, j)}$$

Definicja 8. Dwupoziomowy weryfikator Hamminga (*Two-Layer Hamming Verifier* – 2HV)

Dana jest wejściowa kolekcja $x = \big((m_1, s_1), \dots, (m_n, s_n)\big)$ z $n = 2^k - 2$ podpisami dla pewnego całkowitego, dodatniego k oraz z kodem liniowym, reprezentowanym przez macierz błędów H z $2k$ kolumnami oraz n macierzami. Zakłada się dalej, iż wejściowa kolekcja jest zanieczyszczona przez dwa błędne podpisy o indeksach:

$$I_1 = (i_{1,1}, \dots, i_{1,k}),$$
$$I_2 = (i_{2,1}, \dots, i_{2,k}).$$

1. Uruchamia się test ogólny dla całej kolekcji wejściowej. Jeśli $\mathrm{GT}(x,n)=0$, weryfikator kończy działanie, w przeciwnym przypadku jest wykonywany kolejny krok.
2. Tworzy się $2k$ podkolekcji (grup kontrolnych) odpowiadających wierszom w macierzy H:

$$x_{1,i}=\{(m_j,s_j)\,|\,H_1(i,j)=1,\ j+1,\ldots,n\}$$
$$x_{2,i}=\{(m_j,s_j)\,|\,H_2(i,j)=1,\ j+1,\ldots,n\}$$

   dla $i=1,\ldots,k$.
3. Uruchamia się testy ogólne $\mathrm{GT}(x_{1,i},2^{k-1}-1)=\sigma_i$ oraz $\mathrm{GT}(x_{2,i},2^{k-1}-1)=\sigma_i'$ dla $i=1,\ldots,k$. Tworzy się dwa syndromy:

$$\sigma=(\sigma_1,\ldots,\sigma_k),$$
$$\sigma'=(\sigma_1',\ldots,\sigma_k').$$

4. Znajduje się pewien indeks m taki, że $\sigma_m=1$ oraz $\sigma_m'=1$, co oznacza, że w m-tej grupie kontrolnej znajduje się dokładnie jeden błędny podpis.
5. Uruchamia się weryfikator HV dla podkolekcji $x_{1,m}$, w wyniku czego jest identyfikowany indeks pierwszego błędnego podpisu I_1.
6. Oblicza się drugi indeks według wzoru:

$$I_2=I_1\oplus\sigma\oplus\overline{\sigma'}.$$

7. Uruchamia się test ogólny na wejściowej kolekcji bez dwóch zidentyfikowanych błędnych podpisów o indeksach (I_1,I_2), jeśli test przechodzi, weryfikator zwraca dwa znalezione indeksy, w przeciwnym przypadku działanie kończy się niepowodzeniem i weryfikator zwraca „1”.

Wytłumaczenia wymaga jedynie wzór z kroku 6:

$$\begin{cases} I_1+I_2=\sigma \\ \overline{I_1}+\overline{I_2}=\sigma' \end{cases}$$

$$\begin{cases} I_1\oplus I_2\oplus I_1I_2=\sigma \\ I_1I_2=\overline{\sigma'} \end{cases}$$

$$I_2=I_1\oplus\sigma\oplus\overline{\sigma'}.$$

Na podstawie analizy złożoności dwupoziomowego weryfikatora 2HV można sformułować następującą propozycję.

Propozycja 1

Dana jest wejściowa kolekcja o liczności 2^k-2 zanieczyszczona przez 2 błędne podpisy. Wtedy dwupoziomowy weryfikator Hamminga zawsze prawidłowo zidentyfikuje 2 błędne podpisy kosztem $3k+3$ testów ogólnych GT.

Warto podkreślić, że weryfikator 2HV poprawnie zidentyfikuje jeden błędny podpis, zwracając indeksy $I_1 = I_2$. Inną możliwością jest zastosowanie najpierw weryfikatora HV, a dopiero później 2HV, bowiem część potrzebnych do identyfikacji testów jest już wykonana.

Można rozważyć prosty przykład. Niech kolekcja wejściowa zawiera $n = 2^4 - 2 = 14$ podpisów. Zakłada się dalej, że błędne podpisy występują na 6 pozycji (0110) oraz na 10 pozycji (1010). Macierz H będzie miała postać:

$$H = \begin{bmatrix} H_1 \\ H_2 \end{bmatrix} = \left[\begin{array}{cccccccccccccc} 1 & 0 & 1 & 0 & 1 & \mathbf{0} & 1 & 0 & 1 & \mathbf{0} & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & \mathbf{1} & 1 & 0 & 0 & \mathbf{1} & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & \mathbf{1} & 1 & 0 & 0 & \mathbf{0} & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & \mathbf{0} & 0 & 1 & 1 & \mathbf{1} & 1 & 1 & 1 & 1 \\ & & & & & & & & & & & & & \\ 0 & 1 & 0 & 1 & 0 & \mathbf{1} & 0 & 1 & 0 & \mathbf{1} & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & \mathbf{0} & 0 & 1 & 1 & \mathbf{0} & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & \mathbf{0} & 0 & 1 & 1 & \mathbf{1} & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & \mathbf{1} & 1 & 0 & 0 & \mathbf{0} & 0 & 0 & 0 & 0 \end{array}\right]$$

Tworzy się podkolekcje, zgodnie z krokiem drugim algorytmu weryfikatora 2HV, a także oblicza syndromy:

$$\begin{aligned} \sigma &= (1110), \\ \sigma' &= (1101), \\ \sigma_3 &= \sigma'_3. \end{aligned}$$

Z tego wynika, że trzecia grupa kontrolna zawiera wyłącznie jeden błędny podpis. Teraz używa się weryfikatora HV do trzeciej grupy kontrolnej macierzy H_1, uzyskując $I_1 = (0110)$. Drugi indeks oblicza się następująco:

$$I_2 = I_1 \oplus \sigma \oplus \overline{\sigma'} = (0110) \oplus (1110) \oplus (0010) = (1010).$$

Ogólny model kodów identyfikujących błędne podpisy

Należy rozważyć ponownie weryfikator 2HV. Wydawałoby się, że jest możliwe zoptymalizowanie weryfikatora 2HV, czyli uzyskanie indeksów dwóch błędnych podpisów za pomocą $2k$ testów. Innymi słowy, powstaje problem, czy w następującym układzie równań jest możliwe obliczenie indeksów I_1, I_2:

$$\begin{cases} I_1 \oplus I_2 \oplus I_1 I_2 = \sigma \\ f(I_1) \oplus f(I_2) \oplus f(I_1) f(I_2) = \sigma'. \end{cases}$$

Prawdziwe jest twierdzenie 1.

Twierdzenie 1

Jeśli są dane cztery ciągi binarne oraz funkcja binarna f

$$I_1, I_2, \sigma, \sigma' \in \Sigma^k$$

$$f : \Sigma^k \to \Sigma^k$$

$$\Sigma^k : \{0,1\}^k$$

używane w weryfikatorze 2HV, spełniające opisany układ równań, to nie istnieje funkcja binarna f, dla której ten układ równań ma unikalne rozwiązanie dla I_1 i I_2.

Można teraz zdefiniować ogólny model kodów identyfikujących błędne podpisy.

Definicja 9. Weryfikator ogólny (*Generic Verifier* – GV)

Dana jest wejściowa kolekcja $x = \big((m_1, s_1), \ldots, (m_n, s_n)\big)$ o liczności $n = 2^k$ dla pewnego całkowitego, dodatniego k oraz pewien kod liniowy reprezentowany przez macierz błędów H z $l > 2k$ wierszami i n kolumnami. Można założyć, że kolekcja wejściowa jest zanieczyszczona przez t błędnych podpisów o indeksach $I_1, \ldots, I_t$, które są kolumnami macierzy H. Syndrom $\sigma = I_1 + \ldots + I_t$, będący bitową sumą logiczną indeksów $I_1, \ldots, I_t$, jednoznacznie identyfikuje indeksy $I_1, \ldots, I_t$.

1. Uruchamia się test ogólny GT(x, n) na wejściowej kolekcji. Jeśli GT$(x, n) = 0$, weryfikator kończy działanie, w przeciwnym przypadku jest wykonywany kolejny krok.
2. Tworzy się l podkolekcji (lub grup kontrolnych) odpowiadających wierszom macierzy H.
3. Uruchamia się l razy test ogólny GT i formuje się syndrom σ.
4. Z syndromu σ identyfikuje się t błędnych podpisów $I_1, \ldots, I_t$.
5. Uruchamia się test ogólny GT na wejściowej kolekcji x bez t błędnych, zidentyfikowanych podpisów. Jeśli test akceptuje wejściową kolekcję, są zwracane indeksy błędnych podpisów, w przeciwnym przypadku działanie kończy się niepowodzeniem i weryfikator zwraca „1”.

Wnioski końcowe

Zdefiniowany weryfikator ogólny stanowi dobrą podstawę do dalszych badań, w szczególności należy sprecyzować kilka otwartych zagadnień, takich jak:

– ograniczenie dolne lub zależność funkcyjna, określająca związek między liczbą testów ogólnych GT a liczbą błędnych podpisów w kolekcji t;
– zasady projektowania macierzy H tak, aby syndrom poprawnie identyfikował pozycje błędnych podpisów w kolekcji;
– zasady projektowania efektywnych kodów weryfikujących;
– określenie t, dla którego kod weryfikujący staje się gorszy od testu naiwnego NV;
– konstrukcje kodów weryfikujących oparte na podejściu kombinatorycznym (Pastuszak, Michałek, Pieprzyk oraz Seberry w [7] podali konstrukcję wykorzystującą macierz SBIBD).

Bibliografia

[1] Bellare M., Garay J., Rabin T.: *Fast batch verification for modular exponentiation and digital signatures*. W: *Advances in Cryptology – EUROCRYPT'98*. Ed. K. Nyberg. Springer, Lecture Notes in Computer Science, 1998, no. 1403, s. 236–250

[2] Bellare M., Yacobi Y.: *Batch Diffie-Hellman key agreement systems and their application to portable communications*. W: *Advances in Cryptology – EUROCRYPT'92*. Ed. R. Rueppel. Springer, Lecture Notes in Computer Science, 1993, no. 658, s. 208–220

[3] Berlekamp E.: *Algebraic Coding Theory*. New York, McGraw-Hill, 1968

[4] Coron J.-S., Naccache D.: *On the security of RSA screening*. W: *Public Key Cryptography – Second International Workshop on Practice and Theory in Public Key Cryptography (PKC'99)*. Eds. H. Imai, Y. Zheng. Springer, Lecture Notes in Computer Science, 1999, no. 1560, s. 197–203

[5] Harn L.: *Batch verifying multiple DSA-type digital signatures*. Electronics Letters, 1998, vol. 34(9), s. 870–871

[6] Naccache D., M'Raihi D., Vaudenay S., Raphaeli D.: *Can DSA be improved complexity trade-offs with the digital signature standard*. W: *Advances in Cryptology – EUROCRYPT'94*. Ed. A. De Santis. Springer, Lecture Notes in Computer Science, 1995, no. 950, s. 77–85

[7] Pastuszak J., Michałek D., Pieprzyk J., Seberry J.: *Identification of bad signatures in batches*. W: *Public Key Cryptography – Third International Workshop on Practice and Theory in Public Key Cryptography (PKC'2000)*. Eds. H. Imai, Y. Zheng. Springer, Lecture Notes in Computer Science, 2000, no. 1751, s. 28–45

[8] Street A.P., Wallis W.D.: *Combinatorics: A First Course*. Winnipeg, CBRC, 1982

[9] Yen S., Laih C.: *Improved digital signature suitable for batch certification*. IEEE Transactions on Computers, 1995, vol. 44(7), s. 957–959

Józef Pieprzyk

Prof. dr inż. Józef Pieprzyk (1949) – absolwent Wydziału Elektroniki i Elektrotechniki Wyższej Szkoły Inżynierskiej w Bydgoszczy (1972) oraz Wydziału Matematyki, Fizyki i Chemii Uniwersytetu Mikołaja Kopernika w Toruniu (1975); nauczyciel akademicki i pracownik naukowy Akademii Techniczno-Rolniczej w Bydgoszczy (1972–1986) oraz wielu australijskich uczelni: Sydney University (1986–1988), University of New South Wales, Australian Defence Force Academy w Canberze (1988–1992), Wollongong University (1992–2001), Macquarie University w Sydney (od 2001); autor ponad 160 artykułów; zainteresowania naukowe: projektowanie i analiza algorytmów kryptograficznych, kryptografia rozproszona, rozproszone protokoły kryptograficzne, systemy z podziałem sekretu, ochrona dokumentów elektronicznych, ochrona sieci komputerowych.
e-mail: josef@comp.mq.edu.au

Jarosław Pastuszak

Mgr inż. Jarosław Pastuszak (1972) – absolwent Wydziału Telekomunikacji i Elektroniki Akademii Techniczno-Rolniczej w Bydgoszczy (1996); doktorant w Instytucie Badań Systemowych PAN; pracownik firmy Bazy i Systemy Bankowe Sp. z o.o. w Bydgoszczy (od 1995); autor kilkunastu artykułów naukowych oraz popularnonaukowych; zainteresowania naukowe: protokoły kryptograficzne, systemy z podziałem sekretu, ochrona dokumentów elektronicznych, infrastruktura PKI.
e-mail: jarek.pastuszak@bsb.com.pl