RZECZPOSPOLITA POLSKA

(12) **TŁUMACZENIE PATENTU EUROPEJSKIEGO** (19) **PL** (11) **PL/EP 1455257**

(13) **T3**

(96) Data i numer zgłoszenia patentu europejskiego:
02.03.2004 04004847.2

(51) Int. Cl.[7]
G06F1/00
G11B20/00

(97) O udzieleniu patentu europejskiego ogłoszono:
05.10.2005 Europejski Biuletyn Patentowy 2005/40

EP 1455257 B1

Urząd Patentowy Rzeczypospolitej Polskiej

(54) Tytuł wynalazku:

System wiązania sekretów kryptograficznych z systemem komputerowym tolerującym zmiany sprzętowe

(30) Pierwszeństwo:
US20030378224 03.03.2003

(43) Zgłoszenie ogłoszono:
08.09.2004 Europejski Biuletyn Patentowy 2004/37

(45) O złożeniu tłumaczenia patentu ogłoszono:
31.03.2006 Wiadomości Urzędu Patentowego 3/2006

(73) Uprawniony z patentu:
MICROSOFT CORPORATION, Redmond, US

(72) Twórca (y) wynalazku:
Aidan T. Hughes, Washington, US

(74) Pełnomocnik:
Przedsiębiorstwo Rzeczników Patentowych Patpol Sp. z o.o.
rzecz. pat. Izabela Ludwicka
02-770 Warszawa 130
skr. poczt. 37

PL/EP 1455257 T3

PRZEDMIOT WYNALAZKU

Przedmiotem wynalazku są systemy i sposoby przyporządkowywania do danego systemu komputerowego pewnego sekretu. Przedmiotem wynalazku są również systemy i sposoby generowania mocnego identyfikatora sprzętowego dla danego systemu komputerowego, przy czym mocny identyfikator sprzętowy jest sprzężony z pewnym sekretem związanym. Wynikowy mocny identyfikator sprzętowy może służyć do nadzoru wykorzystania oprogramowania na danym systemie komputerowym zależnie od stopnia zmian sprzętowych w systemie komputerowym.

PODSTAWA WYNALAZKU

W ostatnich latach czyniono poważne wysiłki dla uniemożliwiania lub minimalizowania bezprawnego korzystania z oprogramowania komputerowego. Ze względu na odtwarzalność i łatwość rozprowadzania, piractwo w zakresie programów komputerowych i nielegalne korzystanie z oprogramowania komputerowego poza zakresem umowy licencyjnej, są zjawiskami częstymi, które poważnie szkodzą producentom oprogramowania.

Opracowywano sposoby zmierzające do ograniczenia piractwa komputerowego i nielegalnego wykorzystywania oprogramowania komputerowego poza zakresem umowy licencyjnej. Jednakowoż takie sposoby często powodują problemy dla uprawnionych nabywców i użytkowników w postaci niedogodności użytkowania. Na przykład, użytkownik, który dokonał modernizacji swojego komputera powinien być w stanie legalnie ponownie zainstalować produkt programowy na zmodernizowanej maszynie. Jednakowoż obecnie dostępne sposoby mogą albo (i) nie pozwalać na ponowne zainstalowanie, albo (ii) zmuszać użytkownika (który jest z tego

niezadowolony) do zwrócenia się o pomoc do producenta oprogramowania.

W związku z tym, występuje zapotrzebowanie na opracowanie udoskonalonych metod zabezpieczeń przeciw piractwu i nielegalnemu wykorzystywaniu, lecz umożliwiających rozpoznanie i uwzględnienie potrzeb i praktycznego wykorzystania przez legalnego nabywcę i użytkownika oprogramowania.

ISTOTA WYNALAZKU

Niniejszy wynalazek pozwala rozwiązać niektóre z trudności i problemów opisanych powyżej, w wyniku opracowania sposobu wiązania z danym systemem komputerowym tak zwanego sekretu, oraz udoskonalonej identyfikacji sprzętowej związanej z tym sekretem. Identyfikacja sprzętowa według niniejszego wynalazku zapewnia sposób minimalizacji lub eliminacji piractwa oprogramowania i nielegalnego wykorzystywania oprogramowania komputerowego poza zakresem umowy licencyjnej przy jednoczesnym umożliwieniu modernizowania maszyny przez uprawnionych użytkowników oprogramowania.

Identyfikator sprzętowy, według niniejszego wynalazku, zwany w niniejszym dokumencie mocnym identyfikatorem sprzętowym (SHWID - strong hardware identification) zawiera dwa oddzielne elementy: (1) sprzętowy składnik identyfikacyjny oraz (2) częściowy składnik sekretu. Przez łączenie (1) sprzętowego składnika identyfikatora z (2) częściowym składnikiem sekretu powstaje bardziej pewny i niezawodny mocny identyfikator sprzętowy (SHWID) dla danego systemu komputerowego.

Mocny identyfikator sprzętowy może służyć do identyfikacji danej konfiguracji sprzętowej przy załadowywaniu produktu programowego do komputera. Mocny identyfikator sprzętowy może być przechowywany dla wykorzystania w przyszłości, na przykład (i) kiedy ten sam produkt progra-

mowy jest załadowywany na ten sam komputer, lub (ii) kiedy ten sam produkt programowy jest załadowywany ponownie na pewną odmianę tego samego komputera, lub na komputer całkiem inny. Na przykład, kiedy ten sam produkt programowy jest załadowywany na ten sam komputer lub na odmianę tego samego komputera, następuje sprawdzenie, czy możliwe jest wygenerowanie sekretu związanego z oryginalnym mocnym identyfikatorem sprzętowym (SHWID). Jeżeli wygenerowanie sekretu jest możliwe, to sposób według niniejszego wynalazku umożliwia wywoływanie produktu programowego. Natomiast, jeżeli wygenerowanie sekretu nie jest możliwe, to sposób według niniejszego wynalazku nie umożliwi załadowania produktu programowego wskutek zmian oryginalnego systemu sprzętowego poza wyznaczoną wartość graniczną.

Odpowiednio do tego, celem niniejszego wynalazku jest sposób przywiązywania do danego systemu komputerowego pewnego sekretu, oraz mocny identyfikator sprzętowy (SHWID - strong hardware identification) sprzężony z tym sekretem. Poza tym, celem niniejszego wynalazku jest sposób uniemożliwiający wykorzystywanie oprogramowania w systemie komputerowym, jeżeli nie jest możliwe wyszukanie w tym systemie komputerowym sekretu sprzężonego z oryginalnym mocnym identyfikatorem sprzętowym (SHWID).

Te i inne cechy charakterystyczne i zalety niniejszego wynalazku w sposób jasny wynikają z poniższego szczegółowego opisu przedstawianych odmian wykonania i załączonych zastrzeżeń.

KRÓTKI OPIS FIGUR RYSUNKU

Fig. 1 przedstawia sieć działań niektórych z głównych elementów przykładowego środowiska operacyjnego do implementacji niniejszego wynalazku;

Fig. 2 przedstawia sieć działań ukazującą przykładowe etapy wyznaczania sprzętowego elementu identyfikacyjnego należącego do mocnego identyfikatora sprzętowego (SHWID);

Fig. 3-4 przedstawiają sieć działań ukazującą przykładowe etapy wyznaczania elementu sekretu częściowego dla mocnego identyfikatora sprzętowego (SHWID);

Fig. 5 przedstawia sieć działań ukazującą przykładowe etapy łączenia elementu identyfikatora sprzętowego z elementem sekretu częściowego mocnego identyfikatora sprzętowego (SHWID); a

Fig. 6-7 przedstawiają sieć działań ukazującą przykładowe etapy stwierdzania, czy produkt programowy może być wykorzystywany w sprzętowym systemie komputerowym z wykorzystaniem wyszukiwania sekretu związanego jako czynnika odblokowującego produkt programowy.

SZCZEGÓŁOWY OPIS WYNALAZKU

Dla zapewnienia większej zrozumiałości zasad niniejszego wynalazku, poniżej zamieszczono opisy konkretnych odmian wykonania niniejszego wynalazku, a do opisu tych konkretnych odmian wykonania stosuje się specjalny język. Tym niemniej jest zrozumiałe, że zastosowanie tego specjalnego języka nie ma na celu żadnego ograniczenia zakresu wynalazku. Zmiany, dodatkowe modyfikacje i tego rodzaju zastosowanie zasad niniejszego wynalazku uważa się za oczywiste dla specjalisty w specjalności, której dotyczy wynalazek.

Celem niniejszego wynalazku jest sposób powiązania pewnego sekretu z danym systemem komputerowym, i sprzężony z tym sekretem mocny identyfikator sprzętowy (SHWID). Sekret zwykle stanowi wybraną losowo liczbę całkowitą. Pożądane jest, aby sekret miał następujące właściwości charakterystyczne:

(1) Sekret, *S*, jest obliczalny w konfiguracji sprzętowej H;

(2) Sekret, *S*, jest obliczalny w konfiguracji sprzętowej H_1, która jest konfiguracją sprzętową H po dokonaniu pewnej ilości zmian elementowych do pewnej ustalonej granicznej wartości stopnia zmiany elementowej; i

(3) Sekret, *S*, jest praktycznie niemożliwy do wyliczenia w jakiejkolwiek innej konfiguracji sprzętowej H_2.

Sekret może służyć do generowania mocnego identyfikatora sprzętowego (SHWID) dla danego systemu komputerowego zawierającego pewną rozmaitość elementów sprzętowych. Przykładowy system komputerowy może zawierać pewną liczbę klas elementów sprzętowych, obejmującą, chociaż nie wyłącznie, napędy dysków twardych, napędy dysków optycznych, na przykład napędy pamięci CD-ROM, karty sieciowe, karty graficzne, pamięć stałą (ROM), pamięć o dostępie swobodnym (RAM) i podstawowy system wejścia-wyjścia (BIOS - basic input/output system). Poniżej opisano przykładowy system komputerowy i przykładowe środowisko operacyjne do realizacji niniejszego wynalazku.

Przykładowe środowisko operacyjne

Poniżej opisano przykładowe odmiany wykonania niniejszego wynalazku, w odniesieniu do rysunków, na których wszystkich figurach takie same odnośniki liczbowe reprezentują podobne elementy. Na fig. 1 przedstawiono przykładowe środowisko operacyjne do implementacji niniejszego wynalazku. Przykładowe środowisko operacyjne obejmuje uniwersalne urządzenie liczące w postaci konwencjonalnego komputera osobistego 20. Na ogół komputer osobisty 20 zawiera procesor 21, pamięć systemową 22 i magistralę systemową 23,

która sprzęga różne elementy systemu, włącznie z pamięcią systemową 22, z procesorem 21. Magistrala systemowa 23 może być dowolną z kilku typów struktur magistralowych, włącznie z magistralą pamięci lub sterownikiem pamięci, magistralą peryferyjną oraz magistralą lokalną o dowolnej spośród różnych struktur magistralowych. Pamięć systemowa zawiera pamięć stałą (ROM) 24 i pamięć o dostępie swobodnym (RAM) 25. Podstawowy system wejścia-wyjścia (BIOS) 26 zawierający podstawowe podprogramy służące do przekazywania informacji między elementami wewnątrz komputera osobistego 20, na przykład podczas rozruchu, jest przechowywany w pamięci ROM 24.

Komputer osobisty 20 poza tym zawiera napęd 27 dysku twardego do zapisywania lub odczytywania dysku twardego, nie pokazanego, napęd 28 dysku magnetycznego do zapisywania lub odczytywania wyjmowanego dysku magnetycznego 29 i napęd 30 dysku optycznego, do odczytywania i zapisywania wyjmowanego dysku optycznego 31, na przykład dysku CD-ROM lub innego nośnika optycznego. Napęd 27 dysku twardego, napęd 28 dysku magnetycznego i napęd 30 dysku optycznego są dołączone do magistrali systemowej 23 przez, odpowiednio, interfejs 32 napędu dysku twardego, interfejs 33 dysku magnetycznego i interfejs 34 napędu dysku optycznego. Jakkolwiek w przykładowym środowisku opisanym w niniejszym dokumencie wykorzystuje się dysk twardy 27, wyjmowany dysk magnetyczny 29 i wyjmowany dysk optyczny 31, to dla specjalisty jest oczywiste, że w przykładowym środowisku operacyjnym można wykorzystywać inne typy nośników odczytywalnych komputerowo, które mogą magazynować dane dostępne dla komputera, jak na przykład kasety magnetyczne, karty pamięci flash, cyfrowe dyski wizyjne, kasety Bernoulliego, pamięci RAM, pamięci ROM i tym podobne. Napędy i odpowiada-

jące im nośniki odczytywalne komputerowo stanowią pamięć nieulotną dla wykonywalnych komputerowo instrukcji, struktur danych, modułów programowych i innych danych dla komputera osobistego 20. Na przykład, w pamięci RAM 25 i/lub w napędzie 27 z dyskiem twardym komputera osobistego 20 może być przechowywany jeden lub więcej plik 60 z danymi.

Możliwe jest przechowywanie na dysku twardym 27, dysku magnetycznym 29, dysku optycznym 31, w pamięci ROM 24 lub pamięci RAM 25, pewnej liczby modułów programowych, włącznie z systemem operacyjnym 35 pewnym aplikacyjnym modułem programowym 36 innymi modułami programowymi 37 i danymi programowymi 38. Moduły programowe obejmują, choć nie tylko, procedury, podprogramy, programy, obiekty, elementy, struktury danych itp., które realizują konkretne zadania lub implementują konkretne abstrakcyjne typy danych. Aspekty niniejszego wynalazku mogą być implementowane jako integralna część pewnego aplikacyjnego modułu programowego 36 lub jako część innego modułu programowego 37.

Użytkownik może rozkazy i informacje wprowadzać do komputera osobistego 20 przez urządzenia wejściowe, na przykład klawiaturę 40 i urządzenie wskazujące 42. Możliwe jest także stosowanie urządzeń wejściowych (nie pokazanych), takich jak mikrofon, manipulator drążkowy, podkładka game pad, antena satelitarna, skaner itp. Te i inne urządzenia wejściowe są często dołączane do procesora 22 przez interfejs 46 portu szeregowego, który jest sprzężony z magistralą systemową 23, lecz mogą być dołączane przez inne interfejsy, na przykład port równoległy, port gier, uniwersalną magistralę szeregową (USB - universal serial bus) itp. Do magistrali systemowej za pośrednictwem

interfejsu, na przykład karty graficznej 48, może być dołączony monitor 47 lub innego typu urządzenie wyświetlające. Poza monitorem, komputery osobiste zwykle zawierają wyjściowe urządzenia peryferyjne (nie przedstawione), na przykład głośniki i drukarki.

Komputer osobisty 20 może działać w środowisku sieciowym z wykorzystaniem połączeń logicznych z jednym lub więcej komputerami zdalnymi 49. Komputer zdalny 49 może być drugim komputerem osobistym, serwerem, klientem, urządzeniem trasującym, sieciowym komputerem osobistym, lub urządzeniem równorzędnym, bądź innym węzłem wspólnej sieci. Jakkolwiek komputer zdalny 49 zwykle zawiera wiele lub wszystkie, z elementów opisanych powyżej w odniesieniu do komputera osobistego 20, to na fig. 1 przedstawiono tylko pamięciowe urządzenie magazynujące 50. Połączenia logiczne przedstawione na fig. 1 obejmują sieć lokalną (LAN) 51 i sieć rozległą (WAN) 52. Takie środowiska sieciowe są powszechne w biurach, sieciach komputerowych obejmujących całe przedsiębiorstwa, sieciach Intranet, i sieci Internet.

Komputer osobisty 20, przy jego zastosowaniu w sieciowym środowisku LAN, jest dołączony do sieci lokalnej 51 przez pewien interfejs sieciowy czyli adapter 53. Przy wykorzystaniu w środowisku sieciowym WAN, komputer osobisty 20 zwykle zawiera modem 54 lub inny środek nawiązywania komunikacji w sieci WAN 52, na przykład sieci Internet. Modem 54, który może być wewnętrzny lub zewnętrzny, jest dołączony do magistrali systemowej 23 za pośrednictwem interfejsu 46 portu szeregowego. W środowisku sieciowym, moduły programowe przedstawione w odniesieniu do komputera osobistego 20 lub ich części, mogą być przechowywane w zdalnym pamięciowym urządzeniu magazynującym 50. Jest oczywiste, że przedstawione połączenia sieciowe są przykła-

dowe, i do zestawiania łącza komunikacyjnego między komputerami możliwe jest wykorzystywanie innych środków.

Ponadto, dla specjalistów jest oczywiste, że niniejszy wynalazek może być implementowany w innych konfiguracjach systemu komputerowego, włącznie z urządzeniami podręcznymi, systemami wieloprocesorowymi, mikroprocesorowymi lub programowanymi urządzeniami elektroniki użytkowej, sieciowymi komputerami osobistymi, minikomputerami, dużymi komputerami i tym podobnymi. Niniejszy wynalazek może być również realizowany w rozproszonych środowiskach obliczeniowych, gdzie zadania są realizowane przez zdalne urządzenia przetwarzające, które są połączone przez sieć komunikacyjną. W rozproszonym środowisku obliczeniowym, moduły programowe mogą znajdować się zarówno w lokalnych, jak i zdalnych pamięciowych urządzeniach magazynujących.

Implementacja przykładowych odmian wykonania niniejszego wynalazku

System komputerowy, jak opisany powyżej, zwykle zawiera wiele klas elementów sprzętowych. Poza tym, system komputerowy może w każdej klasie elementów sprzętowych zawierać wiele elementów (na przykład dwa napędy twardych dysków).

Mocny identyfikator sprzętowy (SHWID - strong hardware identification) według niniejszego wynalazku uwzględnia każdy element (zwany w niniejszym dokumencie każdym "wystąpieniem") w każdej klasie elementów sprzętowych. Mocny identyfikator sprzętowy (SHWID) według niniejszego wynalazku uwzględnia również sekret, *S*, który jest związany z danym sprzętowym systemem komputerowym.

Poniżej przedstawiono przykładowy sposób generowania mocnego identyfikatora sprzętowego (SHWID) według niniejszego wynalazku. Dodatkowo poniżej opisano również mocny

identyfikator sprzętowy (SHWID) jako narzędzie antypirackie.

I. Generowanie mocnego identyfikatora sprzętowego (SHWID – Strong Hardware Identification) dla systemu komputerowego

Mocny identyfikator sprzętowy (SHWID) danego systemu komputerowego zawiera dwa wyraźne elementy składowe: (1) element sprzętowy, i (2) element sekretu częściowego. Poniżej opisano przykładowe sposoby oznaczania takich elementów. Etapy tych przykładowych sposobów mogą być realizowane przez kod programowy w produkcie programowym na komputerze klienta, podobnym do komputera 20, opisanego powyżej w odniesieniu do fig. 1.

A. Oznaczenie elementu sprzętowego identyfikatora SHWID

Identyfikator SHWID według niniejszego wynalazku zawiera pewien produkt klasy, dla każdej klasy elementów sprzętowych. Element sprzętowy według identyfikatora SHWID może być oznaczany, jak to pokazano na fig. 2.

Przedstawiona na fig. 2 przykładowa procedura oznaczania elementu sprzętowego według identyfikatora SHWID rozpoczyna się krokiem 201, przy czym pewna liczba *n* klas elementów jest dobrana odpowiednio do potrzebnej identyfikacji danego systemu komputerowego. Jak to opisano powyżej, dany system komputerowy może zawierać różnorodne elementy sprzętowe i klasy elementów sprzętowych. Przykładowe klasy elementów sprzętowych obejmują, choć nie tylko, napędy dysków twardych, napędy dysków optycznych, karty sieciowe, karty dźwiękowe, karty graficzne, pamięć stałą (ROM), pamięć o dostępie swobodnym (RAM), i system BIOS. Korzystne jest, jeżeli *n*, liczba klas elementów sprzętowych, jest liczbą całkowitą z zakresu od 2 do około 16. Ogólnie

biorąc, jest pożądane, aby *n* było możliwie duże, dla (i) bardziej precyzyjnego identyfikowania danego systemu komputerowego, (ii) dokładniejszego określania stopnia tolerancji danego systemu komputerowego, i (iii) zapewnienia wyższego poziomu bezpieczeństwa sekretu, *S*.

Po wybraniu liczby klas elementów, *n*, w kroku 201, w kroku 202 następuje rozpoznanie każdej z klas elementów. Klasy elementów mogą obejmować dowolne spośród opisanych powyżej klas elementów, na przykład klasę napędów dysków twardych. Przykładowy wykaz klas elementów przedstawiono poniżej w Tablicy 1.

Tablica 1. Przykładowy wykaz klas elementów sprzętowych

Numer klasy elementów	Opis klasy	Identyfikator klasy
1	CdRom	Identyfikator urządzenia CdRom
2	Napęd dysku twardego	Nr kolejny partycji napędu
3	Karta sieciowa	Adres MAC
4	Urządzenie z kartą graficzną	Identyfikator

Jak to pokazano w Tablicy 1, w tym przykładzie *n* jest równe 4, a rozpoznawanymi klasami elementów sprzętowych są: (1) klasa "CdRom"; (2) klasa "Napęd dysku twardego"; (3) klasa "Karta sieciowa"; i (4) klasa "Urządzenie z kartą graficzną".

Po rozpoznaniu w kroku 202 każdej z klas elementów, w etapie 203 odbywa się rozpoznanie wszystkich wystąpień każdej z klas elementów sprzętowych. Korzystne jest, jeżeli każde wystąpienie w poszczególnej klasie jest reprezentowane przez najbardziej jednoznaczny łańcuch identyfikacyjny przyporządkowany do tego wystąpienia. Na przykład konfigu-

racja sprzętowa może zawierać napęd CdRom produkowany przez firmę NEC Corporation i mający łańcuch identyfikacyjny "NEC CDRW24 S15". Według niniejszego wynalazku do wyznaczania najbardziej jednoznacznego łańcucha identyfikacyjnego danego wystąpienia możliwe jest wykorzystywanie dowolnego sposobu, włącznie z zapytaniami urządzeń i wywoływaniem funkcji API systemu operacyjnego. Przykład konfiguracji sprzętowej komputera i wystąpienia w każdej klasie elementów sprzętowych przedstawiono w Tablicy 2, poniżej.

Tablica 2. Przykładowe wystąpienia elementów dla każdej z klas elementów

Numer klasy	Opis klasy	Wystąpienia elementów klasy
1	CdRom	{"NECCDRW24 S15", "TOSHIBA DVDR ASK-1425"}
2	Napęd dysku twardego	{1bcdff1922, 7da90024}
3	Karta sieciowa	{00b0c31b5923}
4	Urządzenie z kartą graficzną	{"NVidia GeForce2 DDR"}

Pokazana w powyższej Tablicy 2 klasa 1, klasa "CdRom", zawiera dwa wystąpienia elementów; klasa 3, klasa "Karta sieciowa" zawiera jedno wystąpienie; a klasa 4, klasa "Urządzenie z kartą graficzną" zawiera jedno wystąpienie.

W kroku 205 generuje się liczbę pierwszą wystąpienia dla każdego wystąpienia elementu, z wykorzystaniem funkcji *f*(x) generacji liczb pierwszych. Korzystne jest, jeżeli funkcja *f*(x) ma następujące właściwości:

(a) wynik *f*(x) jest dodatnią liczbą pierwszą;

(b) x może być dowolną daną o długości do około 65.000 znaków; i

(c) $f(x) > 2^t$ przy czym t jest liczbą całkowitą, korzystnie większą od około 64. Jednak na wartość t nie ma ograniczenia.

(d) wynik f(x) jest oparty deterministycznie na wartości x.

Według niniejszego wynalazku, do generowania liczby pierwszej dla każdego wystąpienia elementu można wykorzystywać dowolną funkcję $f(x)$ generowania liczb pierwszych. Do generowania liczby pierwszej dla każdego wystąpienia elementu może służyć dowolna funkcja $f(x)$ generująca liczby pierwsze. Do odpowiednich funkcji $f(x)$ generujących liczby pierwsze należą, choć nie wyłącznie, funkcje $f(x)$ generujące liczby pierwsze oparte na algorytmie Rabina-Millera, przedstawione w opracowaniu Applied Cryptography, Second Edition (Kryptografia stosowana, wyd. drugie) aut. Bruce Schneiera, str. 259-260, przy czym to przedstawienie włącza się do niniejszego dokumentu przez przywołanie w całości.

Poniższa Tablica 3 zawiera wykaz liczb pierwszych, $i_{p,q}$, dla wystąpień elementów przykładowej konfiguracji sprzętowej. Tablica 3. Przykładowe liczby pierwsze dla wystąpień elementów

Numer klasy elementów	Opis klasy	Liczby pierwsze dla wystąpień elementów
1	CdRom	{f("NEC CDRW24 S15") = $i_{1,1}$, f("TOSHIBA DVDR ASB-1425") = $i_{1,2}$
2	Napęd dysku twardego	{f(1bcdff1922) = $i_{2,1}$, f(7da90024) = $i_{2,2}$}
3	Karta sieciowa	{f(00b0c31b5923) = $i_{3,1}$}
4	Urządzenie z kartą graficzną	{f("NVidia GeForce2 DDR") = $i_{4,1}$}

Występujące w niniejszym dokumencie określenie "liczba pierwsza $i_{p,q}$ wystąpienia" służy do identyfikacji liczby

pierwszej wystąpienia dla kolejnego wystąpienia elementu w klasie 1 elementów (na przykład, p=1) a w szczególności przypadku drugiego wystąpienia elementu w klasie 1 elementów, i w konfiguracji sprzętowej komputera, q=2.

W jednej z odmian wykonania niniejszego wynalazku, do identyfikatora wystąpienia elementu może być, przed wygenerowaniem liczby pierwszej dla danego wystąpienia elementu, dodawana pewna wartość "domieszki" ("salt value"). W tej odmianie wykonania, wprowadzanie wartości domieszki umożliwia wytwarzanie, na bazie tej samej konfiguracji sprzętowej, różnych identyfikatorów SHWID. Wartości domieszek wyprowadzane z kodu aplikacyjnego lub identyfikatora użytkownika zapewniają różne identyfikatory SHWID dla różnych aplikacji lub użytkowników pracujących w tej samej konfiguracji sprzętowej, co może być korzystne przy zabezpieczaniu danych do wykorzystania tylko przez konkretną aplikację lub użytkownika.

Po wygenerowaniu liczb pierwszych wystąpień dla każdego wystąpienia elementu, w kroku 206, następuje wygenerowanie pewnego iloczynu klasy, c_p. Iloczyn klasy c_p jest tworzony przez mnożenie wzajemne liczb pierwszych dla wystąpień danej klasy. Przykładowe iloczyny klas, c_1 do c_4 podano poniżej, w Tablicy 4.

Tablica 4. Przykładowe iloczyny klas dla każdej klasy elementów

Numer klasy elementów	Opis klasy	Iloczyny klas dla każdej klasy elementów
1	CdRom	$c_1 = (i_{1,1}) \times (i_{1,2})$
2	Napęd dysku twardego	$c_2 = (i_{2,1}) \times (i_{2,2})$
3	Karta sieciowa	$c_3 = i_{3,1}$
4	Urządzenie z kartą graficzną	$c_4 = i_{4,1}$

Jak to pokazano w Tablicy 4, iloczyn c_1 klasy dla klasy "CdRom" jest równy iloczynowi dwóch wystąpień, $i_{1,1}$ i $i_{1,2}$. Należy zauważyć, że iloczyny klas otrzymywane z liczby pierwszej pojedynczego wystąpienia, na przykład iloczyn c_3 klasy, mogą być mnożone przez dodatkowe liczby pierwsze nie związane z wystąpieniami, dla zwiększenia trudności faktoryzacji danego iloczynu klasy. Jest to szczególnie użyteczne w przypadku iloczynów klas, złożonych z pojedynczej liczby pierwszej wystąpienia, jak na przykład iloczyn c_3 klasy lub iloczyn c_4 klasy, przedstawione powyżej w Tablicy 4. Przy stosowaniu dodatkowych liczb pierwszych nie związanych z wystąpieniami, dla zwiększenia wartości iloczynu klas, jest korzystne, w przypadku stosowanych liczb pierwszych nie związanych z wystąpieniem, jeżeli zawierają się one w zakresie powyżej 2 lecz poniżej 2^t, gdzie t jest dowolną liczbą całkowitą, jak to opisano powyżej. Zmniejsza to ryzyko nieumyślnej kolizji z liczbami pierwszymi wystąpień z innej konfiguracji sprzętowej.

W kroku 207, każdy iloczyn c_p klasy, jest przechowywany w celu wprowadzenia go do mocnego identyfikatora klasy, C_p, dla każdej klasy elementów, jak to opisano poniżej. Ponadto, jak to opisano poniżej, kombinacja każdego mocnego identyfikatora C_p klasy, dla każdej klasy służy do wytwa-

rzania mocnego identyfikatora sprzętowego (SHWID) dla danego systemu sprzętowego komputera. Iloczyny c_p klas reprezentują element sprzętowy mocnego identyfikatora sprzętowego (SHWID) dla danego komputerowego systemu sprzętowego.

B. Wyznaczanie składnika sekretu częściowego identyfikatora SHWID

Mocny identyfikator sprzętowy (SHWID) według niniejszego wynalazku zawiera również składnik sekretu częściowego dla każdej klasy elementów sprzętowych. Jeden z przykładowych sposobów wyznaczania składnika sekretu częściowego identyfikatora SHWID przedstawiono na fig. 3-4. Etapy przykładowego sposobu mogą być realizowane przez kod programowy wewnątrz produktu programowego na komputerze klienta, podobnym do komputera 20 opisanego powyżej w odniesieniu do fig. 1.

W przedstawionym na fig. 3 kroku 301, za pomocą generatora liczb losowych jest generowana dla każdej klasy elementów pewna liczba pierwsza r_p. Do odpowiednich do tego generatorów liczb losowych należą, choć nie wyłącznie, generatory liczb losowych przedstawione w opracowaniu Prime Numbers (Liczby pierwsze), aut. Crandella i Pomerance'a, rozdział 8, którego treść włącza się przez przywołanie w całości do niniejszego dokumentu. Pożądane jest, aby liczba losowa r_p zawierała się w zakresie od wartości równej zeru lub większej od 0 do nie więcej niż 2^u, przy czym u jest mniejsze od opisanej powyżej wielkości t. Zwykle u jest w przybliżeniu równe t podzielonemu przez 3.

Przy wykorzystywaniu liczby losowej r_p, generowanej w kroku 301 dla każdej klasy elementów sprzętowych, w kroku 302 są generowane liczby pierwsze p_p, przy wykorzystaniu funkcji $g(r_p)$ generowania liczb pierwszych. Korzystne jest,

jeżeli funkcja $g(r_p)$ generowania liczb pierwszych na następujące właściwości:

(a) wynikiem działania $g(r_p)$ jest dodatnia liczba pierwsza;

(b) $g(r_p)$ jest $< 2^V$, przy czym v jest liczbą całkowitą większą od u; i

(c) u + v = t.

(d) wynik $g(r_p)$ jest oparty deterministycznie na wartości rp.

Jako opisanej powyżej funkcji $f(x)$ generującej liczby pierwsze, według niniejszego wynalazku można użyć dowolnej funkcji $g(x)$ do generowania liczby pierwszej dla liczby losowej r_p każdej klasy elementów. Jak to opisano powyżej, korzystne jest, jeżeli funkcja $g(x)$ generowania liczb pierwszych posiada powyższe właściwości charakterystyczne. Do odpowiednich funkcji $f(x)$ generujących liczby pierwsze należą, choć nie wyłącznie, funkcje $f(x)$ generujące liczby pierwsze przedstawione w opracowaniu Applied Cryptography, Second Edition aut. Bruce Schneiera, str. 259-260, przy czym to przedstawienie włącza się do niniejszego dokumentu przez przywołanie w całości.

Poniżej podano przykład zależności między $g(r_p)$, t, i oraz v:

$$t = 64$$

$$u = 20$$

$$v = 44$$

$$0 < r_p < 2^u (2^{20} = 1.048.576)$$

$$2 < g(r_p) < 2^v$$

Przykładowy wykaz dla liczb pierwszych p_p klas dla każdej z n klas (na przykład, n=4) przykładowego systemu komputerowego przedstawiono poniżej w Tablicy 5.

Tablica 5. Przykładowe liczby pierwsze dla każdej klasy elementów

Numer klasy elementów	Opis klasy	Liczby pierwsze dla każdej klasy elementów
1	CdRom	$p_1=g(r_1):0<r_1<2^u$
2	Napęd dysku twardego	$p_2=g(r_2):0<r_2<2^u$
3	Karta sieciowa	$p_3=g(r_3):0<r_3<2^u$
4	Urządzenie z kartą graficzną	$p_4=g(r_4):0<r_4<2^u$

Przykładowy wynik wyjściowy obrazujący zależność między liczbami pierwszymi klas dla danej konfiguracji sprzętowej może być wyrażony jako:

$$2 < p_2 < p_3 < p_1 < p_4 < 2^v$$

W kroku 303 liczba wymaganych zgodności klas elementów, m, jest dobierana zależnie od pożądanego stopnia tolerancji dla zmian elementowych konfiguracji sprzętowej. Liczba wymaganych zgodności, *m*, klas elementów, może być równa aż *n*, ogólnej liczbie klas elementów, lub wynosić zaledwie jeden. W miarę wzrostu *m*, zmniejsza się stopień tolerancji na zmiany konfiguracji sprzętowej komputera. Na przykład, jeżeli łączna liczba klas elementów, *n*, jest równa 4 i *m* jest równe 3, to dla otrzymania sekretu S, który umożliwia załadowanie lub wykonywanie produktu programowego, muszą być zgodne z przynajmniej jednym wystąpieniem elementu przynajmniej 3 z ogółem 4 klas elementów. Jeżeli liczba zgodności klas elementów jest mniejsza od 3, to sekret S nie zostanie otrzymany, i produkt programowy nie będzie działał lub nie będzie załadowywany w tej konfiguracji sprzętowej komputera.

Liczba wymaganych zgodności klas elementów, *m*, może być wyznaczona wstępnie przez producenta oprogramowania i

kodowana w sposobie generacji identyfikatora SHWID według niniejszego wynalazku. Po wybraniu *m*, określa się dodatkowe parametry, co pokazano w krokach 304 i 305.

W kroku 304 wyznacza się parametr *N*, przy czym *N* jest równe iloczynowi *m* najmniejszych liczb pierwszych p_p. Na przykład, w wyjściowych przykładowych liczbach pierwszych dla klas dwiema najmniejszymi liczbami pierwszymi klas są p_2 i p_3. Jeżeli *m* jest równe 2, to *N* jest równe $(p_2)x(p_3)$.

W kroku 305 wyznaczany jest parametr *M*, przy czym *M* jest równe iloczynowi (*m*−1) największych liczb pierwszych p_p klas. Na przykład, w podanych powyżej przykładowych klasach, p_4 jest największą liczbą pierwszą klasy. Jeżeli *m* jest równe 2, to wtedy *M* jest równe iloczynowi pojedynczej liczbie pierwszej, p_4, klasy (to znaczy, (*M*−1)=1). Należy zauważyć, że *M* musi być mniejsze od *N*, dla zapewnienia, że dany zbiór liczb pierwszych klas ma pewien próg *m*. Jest to pewna implementacja metody współużytkowania sekretu z wartością progową, opisanej w opracowaniu Het Mathematics of Ciphers (Matematyka szyfrów) aut. S.C. Coutinho, rozdział 7, którego treść włącza się w całości do niniejszego dokumentu przez przywołanie.

Po wyznaczeniu parametrów *N* i *M*, w kroku 306 odbywa się wybranie sekretu *S*. Sekret *S* jest większy od *M* lecz mniejszy od *N*. Poza tym sekret S jest dowolną liczba losową między *M* a *N*.

W kroku 307 oblicza się reszty d_p klas stosując następujące równanie:

$$d_p = S \bmod p_p$$

Poniżej, w Tablicy 6 przedstawiono przykładowy zbiór reszt d_p klas.

Tablica 6. Przykładowe reszty klas dla każdej klasy elementów

Numer klasy elementów	Opis klasy	Reszty klas dla każdej klasy elementów
1	CdRom	$d_1 = S \bmod p_1$
2	Napęd dysku twardego	$d_2 = S \bmod p_2$
3	Karta sieciowa	$d_3 = S \bmod p_3$
4	Urządzenie z kartą graficzną	$d_4 = S \bmod p_4$

W kroku 308, przedstawionym na fig. 4, liczby pierwsze p_p klas, dla każdej klasy elementów, są kodowane w postaci pierwszych wartości binarnych dla każdej klasy elementów. Każda z pierwszych wartości binarnych ma u bitów. Należy zauważyć, że każda liczba pierwsza p_p klasy może być reprezentowana przez u bitów zgodnie z następującymi zależnościami:

$p_p = g(r_p)$, gdzie $0 <= r_p < 2^u$

p_p może być reprezentowane przez r_p jeżeli w czasie pobierania jest dostępne $g(r_p)$

r_p może być reprezentowane w u bitach

p_p może być reprezentowane przez r_p jeżeli w czasie pobierania jest dostępne $g(r_p)$ o ile znana jest funkcja $g()$ generowania liczb pierwszych z powodu następującego. Kiedy p_p jest równe $g(r_p)$ i znana jest funkcja $g()$ generowania liczb pierwszych, to wtedy znajomość r_p jest wystarczająca do regenerowania p_p przez wykonanie g() z parametrem r_p. Kodowanie r_p wymaga u bitów (czyli w powyższym przykładzie 20 bitów), podczas gdy p_p wymaga v bitów (czyli w powyższym przykładzie 44 bity). Oszczędności w odniesieniu do żądanej liczby bitów realizowane są przez reprezentowanie p_p jako r_p.

W kroku 309, każda z reszt d_p klas jest kodowana w postaci drugich wartości binarnych dla każdej klasy elementów. Te drugie wartości binarne bitów mogą być reprezentowane przez v bitów. Należy zauważyć, że reszty d_p klas mogą być reprezentowane przez v bitów, w wyniku następujących zależności:

$$d_p = S \bmod p_p$$

$$0 < P_p < 2^v$$

$$\text{Zatem, } d_p < 2^v$$

W kroku 310, odbywa się konkatenacja pierwszej wartości binarnej wygenerowanej w kroku 308 z drugą wartością binarną z kroku 309 z utworzeniem zakodowanego sekretu częściowego klasy elementów, P_p, mającego w całości t bitów (to znaczy, t = u + v). Sekret częściowy, P_p, klasy elementów jest generowany dla każdej klasy elementów.

Należy zauważyć, że sekret częściowy, P_p, klasy elementów dla danej klasy elementów może zawierać bity nie wykorzystywane, z, spowodowane tym, że druga wartość binarna ma mniej, niż v bitów. W tym przypadku, te bity niewykorzystywane, z, mogą być rozproszone z pewnym losowym szumem, dla uniemożliwienia agresorowi, który zna właściwości $g(r_p)$, analizowanie zakodowanego seretu częściowego P_p, przy próbie wyznaczenia sekretu częściowego klasy P_p. Na przykład, kiedy p_p mieści się w zakresie 2 - 2^v, dp jest zawsze < p_p. Jeżeli p_p jest znacznie mniejsze od 2^v, to wtedy d_p wymaga znacznie mniej, niż v bitów przy kodowaniu. Agresor mógłby na podstawie wartości d_p odgadnąć rozmiar p_p. Wprowadzenie szumu losowego dla zapełnienia niewykorzystywanych [v-(rozmiar (d_p))] bitów d_p pomaga w ukryciu rozmiaru p_p.

C. Identyfikator SHWID w przypadku systemu komputerowego

Mocny identyfikator sprzętowy (SHWID) może teraz być skonfigurowany z wykorzystaniem iloczynów klas, c_p, otrzymanych w kroku 207 i częściowego sekretu P_p klasy, otrzymanego w kroku 310. Jak to pokazano w kroku 401 na fig. 5, mocne identyfikatory klas (ID), C_p, są tworzone dla każdej klasy elementów, przy czym $C_p = c_p + P_p$. W kroku 402, wszystkie spośród mocnych identyfikatorów klas, C_p, dla klas elementów są wiązane z utworzeniem mocnego identyfikatora sprzętowego (SHWID). Wynikowy identyfikator SHWID jest przechowywany do pobrania w przyszłości. Identyfikator SHWID może być przechowywany lokalnie (na przykład w rejestrze, systemie plikowym, lub bezpiecznej pamięci) albo w dostępnym usytuowaniu zdalnym (na przykład bazie danych).

Należy zauważyć, że zwiększenie bezpieczeństwa można osiągnąć przez zwiększenie wartości t w celu utworzenia sekretu częściowego klasy mającego większą liczbę bitów.

II. Pobieranie sekretu z systemu komputerowego z wykorzystaniem mocnego identyfikatora sprzętowego (SHWID)

Celem niniejszego wynalazku jest ponadto sposób pobierania lub próby pobierania sekretu, *S*, w z danej sprzętowej konfiguracji komputerowej. W jednej z odmian wykonania niniejszego wynalazku, sposób podejmowania próby pobrania sekretu związanego *S* z danej komputerowej konfiguracji sprzętowej jest inicjowany (i) podczas instalowania produktu programowego, (ii) podczas załadowywania aplikacji programowej już istniejącej w elemencie konfiguracji sprzętowej, lub (iii) w obu tych przypadkach. Przykładowy sposób pobierania sekretu związanego *S* z konfiguracji sprzętowej przedstawiono na fig. 6-7. Etapy tego przykładowego sposobu mogą być realizowane przez kod programowy wewnątrz produktu programowego na komputerze klienta,

podobnym do komputera 20 opisanego powyżej w odniesieniu do fig. 1.

W kroku 501, jak to pokazano na fig. 6, dla danej komputerowej konfiguracji sprzętowej są rozpoznawane. mocne identyfikatory klas, C_p, mocnego identyfikatora sprzętowego (SHWID). Dla celów zilustrowania niniejszego wynalazku poniżej opisano sposób pobierania sekretu związanego *S* z trzech oddzielnych konfiguracji sprzętowych, H, H_1 i H_2, z wykorzystaniem uprzednio zapisanego mocnego identyfikatora sprzętowego (SHWID) wyznaczonego z konfiguracji sprzętowej, H. Te trzy różne konfiguracje sprzętowe stanowią (i) dokładnie tę konfigurację sprzętową H, dla której wyemitowano identyfikator SHWID; (ii) konfigurację sprzętową H_1, która stanowi konfigurację sprzętową H ze zmianą jednego lub więcej elementów w dopuszczalnym zakresie tolerancji; i (iii) konfigurację sprzętową H_2, która reprezentuje konfigurację sprzętową H z na tyle dużymi zmianami elementów, że ta konfiguracja sprzętowa H_2 wychodzi poza tolerancje względem konfiguracji sprzętowej H.

Przykładowy zbiór mocnych identyfikatorów klas dla konfiguracji sprzętowych H, H_1 i H_2 podano poniżej w Tablicy 7.

Tablica 7. Przykładowe identyfikatory klas mocnych dla pewnej konfiguracji sprzętowej

Numer klasy elementów	Opis klasy	Identyfikatory klas mocnych
1	CdRom	C_1
2	Napęd dysku twardego	C_2
3	Karta sieciowa	C_3
4	Urządzenie z kartą graficzną	C_4

W kroku 502, rozpoznawane są wszystkie wystąpienia każdej klasy elementów danej konfiguracji sprzętowej. Jak to opisano powyżej, do rozpoznawania każdego wystąpienia elementu można wykorzystywać dowolną metodę konwencjonalną. Zwykle wystąpienie elementu jest rozpoznawane na podstawie najbardziej jednoznacznego łańcucha identyfikacyjnego dla tego elementu. Przykładowe łańcuchy identyfikacyjne dla każdego wystąpienia elementu w przykładowych konfiguracjach H, H_1 i H_2 przedstawiono poniżej w Tablicach 8-10.

Tablica 8. Przykładowe wystąpienia elementów dla konfiguracji sprzętowej H

Konfiguracja H		
Numer klasy elementów	Opis klasy	Wystąpienia elementów
1	CdRom	{"NEC CDRW24 S15", "TOSHIBA DVDR ASB-1425"}
2	Napęd dysku twardego	{1bcdff19, 7da90024}
3	Karta sieciowa	{00b0c31b5923}
4	Urządzenie z kartą graficzną	{"NVidia GeForce2 DDR"}

Tablica 9. Przykładowe wystąpienia elementów dla konfiguracji sprzętowej H_1

Konfiguracja H_1		
Numer klasy elementów	Opis klasy	Wystąpienia elementów
1	CdRom	{"NEC CDRW24 S15," "SONY DVD 1221"}
2	Napęd dysku twardego	{8146af92}
3	Karta sieciowa	{00c0c21b5933}
4	Urządzenie z kartą graficzną	{"NVidia GeForce2 DDR"}

Tablica 10. Przykładowe wystąpienia elementów dla konfiguracji sprzętowej H_2

Konfiguracja H_2		
Numer klasy elementów	Opis klasy	Przypadki elementów
1	CdRom	{"SONY DVD 1221"}
2	Napęd dysku twardego	{8146af92}
3	Karta sieciowa	{00c0c21b5933}
4	Urządzenie z kartą graficzną	{"NVidia GeForce2 DDR"}

Po rozpoznaniu wszystkich wystąpień wewnątrz każdej klasy elementów danej konfiguracji sprzętowej, pobierany jest najbardziej jednoznaczny łańcuch identyfikacyjny dla każdego wystąpienia, jak to pokazano w kroku 503. Łańcuchy identyfikacyjne dla każdego wystąpienia służą do generowania liczb pierwszych wystąpienia, dla każdego wystąpienia elementu, z wykorzystaniem funkcji generowania liczby pierwszej, *f*(identyfikator wystąpienia elementu), co pokazano w kroku 504. Funkcja generowania liczby pierwszej, *f*(identyfikator wystąpienia elementu) może być dowolną

funkcją generowania liczb pierwszych, znaną specjalistom, opisaną powyżej. W Tablicach 11-13 poniżej zamieszczono przykładowe liczby pierwsze dla każdego z wystąpień elementów w przykładowych konfiguracjach sprzętowych H, H_1 i H_2.

Tablica 11. Przykładowe liczby pierwsze $i_{p,q}$, wystąpień, dla przykładowej konfiguracji sprzętowej H

Konfiguracja H		
Numer klasy elementów	Opis klasy	Liczby pierwsze wystąpień
1	CdRom	$\{f($"NEC CDRW24 S15") = $i_{1,1}$, $f($"TOSHIBA DVDR ASB-1425")= $i_{1,2}\}$ -
2	Napęd dysku twardego	$\{f($1bcdff19) = $i_{2,1}$, $f($7da90024) = $i_{2,2}\}$
3	Karta sieciowa	$\{f($00b0c31b5923) = $i_{3,1}\}$
4	Urządzenie z kartą graficzną	$\{f($"NVidia GeForce2 DDR") = $i_{4,1}\}$

Tablica 12. Przykładowe liczby pierwsze $i_{p,q}$, wystąpień, dla przykładowej konfiguracji sprzętowej H_1

Konfiguracja H_1		
Numer klasy elementów	Opis klasy	Liczby pierwsze wystąpień
1	CdRom	$\{f$("NEC CDRW24 S15") = $i_{1,1}$ f("SONY DVD 1221") = $i_{1,3}\}$
2	Napęd dysku twardego	$\{f$(8146af92) = $i_{2,3}\}$
3	Karta sieciowa	$\{f$(00c0c21b5933) = $i_{3,1}\}$
4	Urządzenie z kartą graficzną	$\{f$("NVidia GeForce2 DDR") = $i_{4,1}\}$

Tablica 13. Przykładowe liczby pierwsze $i_{p,q}$, wystąpień, dla przykładowej konfiguracji sprzętowej H_2

Konfiguracja H_2		
Numer klasy elementów	Opis klasy	Liczby pierwsze wystąpień
1	CdRom	{*f*("SONY DVD 1221") = $i_{1,3}$}
2	Napęd dysku twardego	{*f*(8146af92) = $i_{2,3}$}
3	Karta sieciowa	{*f*(00c0c21b5933) = $i_{3,2}$}
4	Urządzenie z kartą graficzną	{*f*("NVidia GeForce2 DDR") = $i_{4,1}$}

Należy zauważyć, że wystąpienie elementu {*f*("SONY DVD 1221"} jest oznaczone $i_{1,3}$ co wskazuje, że ten typ pamięci CdRom jest trzecim typem pamięci CdRom uwzględnianym w powyższych konfiguracjach sprzętowych (to znaczy H, H_1 i H_2).

W kroku 505, wszystkie z potencjalnych sekretów częściowych $P_{p,q}$, klas są wyznaczone, przy czym $P_{p,q}$ jest równe C_p mod $i_{p,q}$. Jak to opisano powyżej, mocny identyfikator klasy, C_p, dla każdej klasy elementów jest wynikiem sumy iloczynu c_p klasy i częściowego sekretu P_p dla każdej klasy. Przy dzieleniu mocnego identyfikatora klasy, C_p, przez liczbę pierwszą $i_{p,q}$ wystąpienia, która występuje w oryginalnej konfiguracji H, która była podstawą mocnego identyfikatora sprzętowego (SHWID), to reszta po operacji (mod) stanowi potencjalny sekret częściowy $P_{p,q}$ klasy. Przykładowe potencjalne sekrety częściowe $P_{p,q}$ klas dla konfiguracji sprzętowych H, H_1 i H_2 podano poniżej w Tablicach 14-16.

Tablica 14. Przykładowe potencjalne sekrety częściowe $P_{p,q}$, klas dla każdego wystąpienia elementu w przykładowej konfiguracji sprzętowej H

Konfiguracja H		
Numer klasy elementów	Opis klasy	Potencjalne sekrety częściowe klasy
1	CdRom	$P_{1,1}=C_1 \bmod i_{1,1}$, $P_{.}=C_1 \bmod i_{1,2}$
2	Napęd dysku twardego	$P_{2,1}=C_2 \bmod i_{2,1}$, $P_{2,2}= C_2 \bmod i_{2,2}$
3	Karta sieciowa	$P_{3,1}=C_3 \bmod i_{3,1}$
4	Urządzenie z kartą graficzną	$P_{4,1} = C_4 \bmod i_{4,1}$

Tablica 15. Przykładowe potencjalne sekrety częściowe $P_{p,q}$, klas dla każdego wystąpienia elementu w przykładowej konfiguracji sprzętowej H_1

Konfiguracja H_1		
Numer klasy elementów	Opis klasy	Potencjalne sekrety częściowe klasy
1	CdRom	$P_{1,1}=C_1 \bmod i_{1,1}$ $P_{1,3}=C_1 \bmod i_{2,3}$
2	Napęd dysku twardego	$P_{2,3}=C_2 \bmod i_{2,3}$
3	Karta sieciowa	$P_{3,2}=C_3 \bmod i_{3,2}$
4	Urządzenie z kartą graficzną	$P_{4,1}=C_4 \bmod i_{4,1}$

Tablica 16. Przykładowe potencjalne sekrety częściowe $P_{p,q}$, klas dla każdego wystąpienia elementu w przykładowej konfiguracji sprzętowej H_2

Konfiguracja H_2		
Numer klasy elementów	Opis klasy	Potencjalne sekrety częściowe klasy
1	CdRom	$P_{1,3}=C_1 \bmod i_{1,3}$
2	Napęd dysku twardego	$P_{2,3}=C_2 \bmod i_{2,3}$
3	Karta sieciowa	$P_{3,2}=C_3 \bmod i_{3,2}$
4	Urządzenie z kartą graficzną	$P_{4,1}=C_4 \bmod i_{4,1}$

Z każdej z klas potencjalnych sekretów częściowych $P_{p,q}$ można wydzielić, jak to pokazano w kroku 506, liczby losowe r_p i reszty d_p klas. Jak to opisano powyżej, liczby pierwsze $p_{p,q}$ klas, mogą być pobieralne z użyciem funkcji $g(r_p)$, gdzie r_p jest to pierwszych u bitów potencjalnego sekretu częściowego $P_{p,q}$ klasy. Reszty $d_{p,q}$ klas mogą być pobrane z ostatnich v bitów sekretu częściowego $P_{p,q}$ klasy. Przykładową listę wszystkich prawdopodobnych liczb pierwszych $p_{p,q}$ klas i reszt $d_{p,q}$ klas dla przykładowych konfiguracji sprzętowych H, H_1 i H_2 podano poniżej w Tablicach 17-19.

Tablica 17. Przykładowe liczby pierwsze $P_{p,q}$ klas i reszty $d_{p,q}$ klas dla każdego potencjalnego sekretu częściowego $P_{p,q}$ przykładowej konfiguracji sprzętowej H

Konfiguracja H		
Numer klasy elementów	Opis klasy	Potencjalne sekrety częściowe klasy
1	CdRom	$P_{1,1}=>p_{1,1}, d_{1,1}$, $P_{1,2}=>p_{1,2}, d_{1,2}$
2	Napęd dysku twardego	$P_{2,1}=>p_{2,1}, d_{2,1}$, $P_{2,2}=>p_{2,2}, d_{2,2}$
3	Karta sieciowa	$P_{3,1}=>p_{3,1}, d_{3,1}$,
4	Urządzenie z kartą graficzną	$P_{4,1}=>p_{4,1}, d_{4,1}$,

5

Tablica 18. Przykładowe liczby pierwsze $P_{p,q}$ klas i reszty $d_{p,q}$ klas dla każdego potencjalnego sekretu częściowego $P_{p,q}$ przykładowej konfiguracji sprzętowej H_1

Konfiguracja H_1		
Numer klasy elementów	Opis klasy	Potencjalne sekrety częściowe klasy
1	CdRom	$P_{1,1}=>p_{1,1}, d_{1,1}$, $P_{1,3}=>p_{1,3}, d_{1,3}$
2	Napęd dysku twardego	$P_{2,3}=>p_{2,3}, d_{2,3}$
3	Karta sieciowa	$P_{3,2}=>p_{3,2}, d_{3,2}$
4	Urządzenie z kartą graficzną	$P_{4,1}=>p_{4,1}, d_{4,1}$

Tablica 19. Przykładowe liczby pierwsze $P_{p,q}$ klas i reszty $d_{p,q}$ klas dla każdego potencjalnego sekretu częściowego $P_{p,q}$ przykładowej konfiguracji sprzętowej H_2

Konfiguracja H_2		
Numer klasy elementów	Opis klasy	Potencjalne sekrety częściowe klasy
1	CdRom	$P_{1,3}=>p_{1,3},d_{1,3}$
2	Napęd dysku twardego	$P_{2,3}=>p_{2,3},d_{2,3}$
3	Karta sieciowa	$P_{3,3}=>p_{3,2},d_{3,2},$
4	Urządzenie z kartą graficzną	$P_{4,1}=>p_{4,1},d_{4,1},$

Po określeniu w kroku 506 wszystkich prawdopodobnych liczb pierwszych klas i reszt klas, reprezentują one pewną liczbę zbiorów kongruencji. Wartość *S* jest dużą liczbą, która przy dzieleniu przez prawdopodobne liczby pierwsze $p_{p,q}$ klas daje dla danego zbioru kongruencji reszty $d_{p,q}$ klas. Ponieważ *S* jest dobrane starannie (to znaczy, *S* zawiera się między M a N) i wszystkie dzielniki są liczbami pierwszymi, to rozwiązaniem dla tego zbioru kongruencji, przy wykorzystaniu prawdopodobnych liczb pierwszych klas i reszt klas, przypadających między M i N musi być wartość *S*. Staranny wybór *S* w kroku 306 zapewnia, że do utworzenia poprawnej wartości *S* potrzebnych jest tylko N elementów zgodnych zbioru kongruencji. Jest to klasyczna progowa metoda współdzielenia sekretu, jak opisana w opracowaniu Het Mathematics of Ciphers (Matematyka szyfrów) aut. S.C. Coutinho, rozdział 7, którego treść włącza się w całości do niniejszego dokumentu przez przywołanie.

Niemożliwe jest ustalenie zawczasu, które, jeżeli w ogóle, prawdopodobne liczby pierwsze i reszty klas są zgodne z pożądaną konfiguracją sprzętową, tak że jest konieczne generowanie potencjalnych sekretów dla każdej permutacji prawdopodobnych liczb pierwszych i reszt klas

przez rozwiązanie, w kroku 507, dyskretnego zbioru kongruencji prezentowanych przy każdej permutacji. Jak to pokazano w kroku 508, wynikowe potencjalne sekrety można sprawdzać z użyciem tekstu zaszyfrowanego tworzonego dla celów weryfikacji. Taki proces opisano poniżej.

Według niniejszego wynalazku, znany tekst jawny jest kodowany z użyciem sekretu *S* jako klucza do tworzenia tekstu zaszyfrowanego. Zwykle zaszyfrowanemu komunikatowi (to znaczy tekstowi zaszyfrowanemu) towarzyszy token weryfikacyjny pozwalający deszyfrantowi się zorientować, czy komunikat został zdeszyfrowany poprawnie. Jest to zwykle albo skrót tekstu jawnego komunikatu albo pewien wybrany tekst jawny. Według niniejszego wynalazku dla uproszczenia korzystne jest stosowanie wybranego tekstu jawnego. Tak, że przy generowaniu identyfikatora SHWID, przy otrzymywaniu tekstu zaszyfrowanego, szyfrowany jest, z użyciem wartości *S* (to znaczy jako klucza), wybrany tekst jawny (na przykład "To jest wybrany tekst jawny"). Dekoder zna zarówno wybrany tekst jawny, jak i tekst zaszyfrowany.

W tej sytuacji możliwe jest zweryfikowanie ważności potencjalnie możliwej wartości *S* (to znaczy każdy z wynikowych potencjalnych sekretów), przez deszyfrowanie tekstu zaszyfrowanego z wykorzystaniem jako klucza tej potencjalnej wartości *S* (to znaczy każdego z wynikowych potencjalnych sekretów). Jeżeli otrzymany tekst jawny jest zgodny z wybranym tekstem jawnym, to wtedy potencjalna wartość *S* (to znaczy jeden z wynikowych potencjalnych sekretów) jest rzeczywiście wartością *S*. Jeżeli wynikowy tekst jawny nie jest zgodny z wybranym tekstem jawnym, to wtedy potencjalna wartość *S* (to znaczy jeden z wynikowych potencjalnych sekretów) nie jest wartością *S*.

Jak to pokazano na fig. 7, jeżeli znany tekst jawny jest ujawniany przez dekodowanie tekstu zaszyfrowanego z zastosowaniem potencjalnego sekretu, wtedy oznacza to, że sekret *S* został znaleziony, i sposób przechodzi do kroku 510, przy czym program umożliwia załadowanie lub zainstalowanie danego produktu programowego w danej konfiguracji sprzętowej komputera. W przeciwnym przypadku sposób przechodzi do kroku 509. Jeżeli istnieje więcej permutacji, które nie zostały sprawdzone, to sposób wraca do kroku 507. W przeciwnym przypadku identyfikator SHWID nie jest zgodny, i sposób przechodzi do kroku 511, który uniemożliwia załadowanie bądź zainstalowanie tego produktu programowego.

Przykładowe wyniki dla przykładowych konfiguracji H, H_1 i H_2 podano poniżej w Tablicach 20-22.

Tablica 20. Przykładowe wyniki ukazujące liczbę zgodności między sekretem potencjalnym $S_{p,q}$ i sekretem rzeczywistym *S* dla przykładowej konfiguracji sprzętowej H

Konfiguracja H		
Numer klasy elementów	Opis klasy	Możliwe zgodności między sekretem potencjalnym $S_{p,q}$ a sekretem rzeczywistym *S*
1	CdRom	$S=d_{1,1} (\text{mod } p_{1,1})$ LUB $S=d_{1,2} (\text{mod } p_{1,2})$
2	Napęd dysku twardego	$S=d_{2,1} (\text{mod } p_{2,1})$ LUB $S=d_{2,2} (\text{mod } p_{2,2})$
3	Karta sieciowa	$S=d_{3,1} (\text{mod } p_{3,1})$
4	Urządzenie z kartą graficzną	$S=d_{4,1} (\text{mod } p_{4,1})$
Wynik – rozwiązanie pojedyncze, zgodności 4 na 4 = S		

Tablica 21. Przykładowe wyniki ukazujące liczbę zgodności między sekretem potencjalnym $S_{p,q}$ i sekretem rzeczywistym S dla przykładowej konfiguracji sprzętowej H_1

Konfiguracja H_1		
Numer klasy elementów	Opis klasy	Możliwe zgodności między sekretem potencjalnym $S_{p,q}$ a sekretem rzeczywistym S
1	CdRom	$S=d_{1,1}(\text{mod } p_{1,1})$ LUB $S=d_{1,3}(\text{mod } p_{1,3})$
2	Napęd dysku twardego	$S=d_{2,3}(\text{mod } p_{2,3})$
3	Karta sieciowa	$S=d_{3,2}(\text{mod } p_{3,2})$
4	Urządzenie z kartą graficzną	$S=d_{4,1}(\text{mod } p_{4,1})$
Wynik – dwa rozwiązanie zależnie od wykorzystania $d_{1,1}$(zgodności 2 na 4, znalezienie S) lub $d_{1,3}$(zgodność 1 na 4, znalezienie Z_1)		

Tablica 22. Przykładowe wyniki ukazujące liczbę zgodności między sekretem potencjalnym $S_{p,q}$ i sekretem rzeczywistym S dla przykładowej konfiguracji sprzętowej H_2

Konfiguracja H_2		
Numer klasy elementów	Opis klasy	Możliwe zgodności między sekretem potencjalnym $S_{p,q}$ a sekretem rzeczywistym S
1	CdRom	$S=d_{1,3}(\text{mod } p_{1,3})$
2	Napęd dysku twardego	$S=d_{2,3}(\text{mod } p_{2,3})$
3	Karta sieciowa	$S=d_{3,2}(\text{mod } p_{3,2})$
4	Urządzenie z kartą graficzną	$S=d_{4,1}(\text{mod } p_{4,1})$
Wynik - Rozwiązanie pojedyncze, zgodność 1 na 4, znalezienie Z_1		

Jak to pokazano w powyższej Tablicy 20, oryginalna konfiguracja sprzętowa H daje w wyniku cztery na cztery zgodności między potencjalnymi sekretami $S_{p,q}$ a sekretem rzeczywistym S. Jak pokazano w powyższej Tablicy 21, konfiguracja sprzętowa H_1 ma najwyżej dwie zgodności na cztery zgodności możliwe, zależnie od tego, która reszta

$d_{p,q}$ klas jest wykorzystywana do wyznaczania potencjalnego sekretu *S*. W tej przykładowej konfiguracji sprzętowej, jeżeli m jest równe 2, to program umożliwia pobranie sekretu związanego S, i załadowanie lub zainstalowanie produktu programowego w konfiguracji sprzętowej H_1. Natomiast w konfiguracji sprzętowej H_2, jak to przedstawiono w Tablicy 22 powyżej, występuje tylko jedna z czterech możliwych zgodności. Jeżeli *m* jest równe 2, to powstają fałszywe nie-sekrety Z1, i sposób uniemożliwia załadowanie lub zainstalowanie konkretnego produktu programowego w konfiguracji sprzętowej H_2.

Opisane powyżej i przedstawione na fig. 2, 3-4, 5 i 6-7 kroki sposobu mogą być realizowane lokalnie lub w usytuowaniu zdalnym. Zwykle klient zakupuje produkt programowy, który może działać na danym komputerze, na przykład komputerze 20 przedstawionym na fig. 1. Produktem programowym może być opakowany w folię termokurczliwą produkt zawierający pewien program komputerowy przechowywany na przenośnym nośniku odczytywalnym przez komputer, na przykład dysku CD-ROM lub dyskietce elastycznej. W rozwiązaniu alternatywnym, produkt programowy może być dostarczany elektronicznie przez sieć, na przykład sieć lokalną (LAN) 51 lub rozległą sieć komputerową (WAN) 52. Klient załadowuje produkt programowy do komputera 20 jako program przechowywany w pamięci systemowej 22.

Podczas instalowania produktu programowego, klient zwykle jest wzywany do wprowadzenia pewnej części identyfikatora (PID - product identification) tego produktu programowego do komputera 20. Identyfikator PID może być, na przykład kluczem dysku CD wydrukowanym na etykiecie opakowania z folią termokurczliwą. Klient wprowadza identyfikator PID, który jest skojarzony z programem tego

produktu programowego. Identyfikator PID jest przechowywany lokalnie na komputerze 20 i/lub zdalnie w miejscu dostępnym, albo w sieci lokalnej (LAN) 51 albo rozległej sieci komputerowej (WAN) 52 przez stronę trzecią, na przykład organ uprawniony do aktywowania.

Jak to opisano powyżej, podczas instalowania produktu programowego, jest generowany również mocny identyfikator sprzętowy (SHWID), przy wykorzystaniu kodu wewnątrz produktu programowego, lub wyzwalany przy instalowaniu produktu programowego. Mocny identyfikator sprzętowy (SHWID) generowany sposobem według niniejszego wynalazku jest skojarzony z identyfikatorem produktu programowego (PID) i przechowywany wraz z identyfikatorem produktu programowego (PID) lokalnie w komputerze 20 i/lub zdalnie w miejscu dostępnym, albo w sieci lokalnej (LAN - local area network) 51 albo w sieci rozległej (WAN) 52, ze stroną trzecią uprawniona do aktywacji.

Jako część procesu instalacji może pojawić się żądanie skierowane do użytkownika, aktywowania produktu programowego przez organ uprawniony. Tym organem uprawnionym mógłby być na przykład producent produktu lub autoryzowana strona trzecia. Proces aktywowania ma na celu zmuszenie klienta do aktywowania produktu programowego (i) w przypadku instalacji i wykorzystywania na konkretnym komputerze, lub (ii) w przypadku instalacji i wykorzystywania zgodnie z warunkami umowy o licencjonowaniu produktu. Taki proces aktywowania opisano szczegółowo w patencie USA nr 6.243.468, udzielonym firmie Microsoft Corporation (z Redmond, WA).

Mocny identyfikator sprzętowy (SHWID) generowany sposobem według niniejszego wynalazku i identyfikator produktu programowego (PID - product identification) mogą być przechowywane lokalnie w komputerze 20 i/lub zdalnie w

miejscu dostępnym, albo w sieci lokalnej (LAN - local area network) 51 albo w sieci rozległej (WAN) 52, z uprawnieniem do aktywacji. Korzystne jest, jeżeli produkt programowy przy pierwszym wywołaniu automatycznie wyświetla okno dialogowe graficznego interfejsu użytkownika (UI - user interface), które wzywa użytkownika do zainicjowania połączenia w serwerem aktywacyjnym dla własnej aktywacji. Serwer aktywacyjny utrzymuje bazę danych do przechowywania odbieranych mocnych identyfikatorów sprzętowych (identyfikatorów SHWID) i przyporządkowanych do nich identyfikatorów produktu programowego (PID).

Mocny identyfikator sprzętowy (SHWID) i przyporządkowany do niego identyfikator (PID) produktu programowego dla danego produktu programowego może być przechowywany w nieograniczonym okresie czasu aż do ponownego zainstalowania na innym komputerze lub wywołania na pierwszym komputerze (to znaczy komputerze wykorzystywanym podczas instalacji początkowej). Kiedy ten sam produkt programowy jest ponownie instalowany na innym komputerze lub wywoływany w pierwszym komputerze, kod produktu programowego inicjuje sposób sprawdzenia, czy produkt programowy może być wykorzystywany w systemie komputerowym według niniejszego wynalazku. Produkt programowy pobiera uprzednio zapisany mocny identyfikator sprzętowy (SHWID) przyporządkowany do identyfikatora produktu programowego (PID), należącego do tego produktu, albo z komputera lokalnego 20 albo z usytuowania zdalnego za pośrednictwem sieci lokalnej (LAN) 51 albo sieci rozległej (WAN) 52. Z wykorzystaniem poprzednio zapisanego mocnego identyfikatora sprzętowego (SHWID) odbywa się sprawdzenie, czy produkt programowy może być wykorzystywany w konfiguracji sprzętowej komputera, jak opisana powyżej.

Kiedy następuje odmowa zgody na wykorzystywanie produktu programowego wskutek znacznych zmian w konfiguracji sprzętowej pierwszego komputera (to znaczy komputera wykorzystywanego podczas instalowania wstępnego), może ukazywać się użytkownikowi okienko dialogowe, informujące, że następuje odmowa dopuszczenia produktu programowego, i że dodatkową informację dotyczącą przyszłego wykorzystania produktu programowego można uzyskać z danego źródła.

III. Inne zastosowania mocnego identyfikatora sprzętowego (SHWID)

Poza zastosowaniami opisanymi powyżej, mocny identyfikator sprzętowy (SHWID) według niniejszego wynalazku może być wykorzystywany przy szyfrowaniu/deszyfrowaniu danych przy zastosowaniu specjalnej konfiguracji sprzętowej.

Jakkolwiek wynalazek opisano w odniesieniu do jego konkretnych odmian wykonania, to jest oczywiste, że specjalista, po otrzymaniu i zrozumieniu powyższego opisu może z łatwością dokonać zmian, opracować warianty i rozwiązania równoważne względem tych odmian wykonania. Odpowiednio do tego zakres niniejszego wynalazku należy oceniać jako zakres załączonych zastrzeżeń niezależnych i zależnych.

Zastrzeżenia patentowe

1. Sposób generowania mocnego identyfikatora sprzętowego (SHWID) dla pierwszego systemu komputerowego o pierwszej konfiguracji sprzętowej, znamienny tym, że sposób obejmuje następujące etapy:

rozpoznanie każdej klasy elementów w pierwszej konfiguracji sprzętowej, przy czym liczba klas jest równa *n*;

określenie iloczynu klasy, c_p, dla każdej klasy elementów;

określenie sekretu częściowego, P_p, klasy dla każdej klasy elementów;

wprowadzenie iloczynu klasy, c_p i sekretu częściowego, P_p, do każdej klasy elementów z utworzeniem *n* mocnych identyfikatorów klas,

przy czym tych *n* mocnych identyfikatorów klas w połączeniu tworzy mocny identyfikator sprzętowy (SHWID) pierwszego systemu komputerowego.

2. Sposób według zastrz. 1, przy czym *n* jest liczbą całkowitą do 16.

3. Sposób według zastrz. 1, przy czym iloczyn klasy, c_p, dla każdej klasy elementów jest wyznaczany w następujących etapach:

wybranie liczby *n* klas elementów;

rozpoznanie *n* klas elementów;

rozpoznanie wszystkich wystąpień w każdej klasie elementów;

przyporządkowanie łańcucha identyfikacyjnego do każdego wystąpienia elementu;

generowanie liczb pierwszych, $P_{p,q}$, wystąpień dla każdego wystąpienia elementu;

przy czym każda liczba pierwsza wystąpienia jest dodatnią liczbą pierwszą, i przy czym p reprezentuje numer danej klasy elementów w zakresie od 1 do *n*, a q reprezentuje q-ty typ elementu w pierwszej konfiguracji sprzętowej; i

mnożenie liczb pierwszych wystąpień w każdej klasie elementów z utworzeniem iloczynu klasy, c_p, dla każdej klasy elementów.

4. Sposób według zastrz. 3, przy czym etap generowania liczb pierwszych wystąpień obejmuje podawanie łańcucha identyfikacyjnego dla każdego wystąpienia elementu do funkcji generacji liczb pierwszych, *f*(x), przy czym x jest łańcuchem identyfikacyjnym o długości do 65.000 znaków.

5. Sposób według zastrz. 4, przy czym *f*(x) generuje dodatnią liczbę pierwszą o wartości większej, niż 2^t, przy czym t jest liczbą całkowitą w zakresie od 32 do 2.048.

6. Sposób według zastrz. 1, przy czym sekret częściowy, P_p, dla każdej klasy elementów jest wyznaczany w następujących etapach:

generowanie liczby losowej, r_p, dla każdej klasy elementów;

generowanie liczb pierwszych p_p klas, dla każdej klasy elementów, przy czym każda liczba pierwsza klasy jest dodatnią liczbą pierwszą

wybór pewnej liczby, *m*, wymaganych zgodności klas elementów, przy czym *m* jest mniejsze od lub równe *n*;

wyznaczenie *N*, przy czym *N* jest równe iloczynowi *m* najmniejszych liczb pierwszych, p_p, klas;

wyznaczenie *M*, przy czym *M* jest równe iloczynowi (*m*-1) największych liczb pierwszych, p_p, klas, a *M* jest mniejsze *N*;

wybranie pewnej liczby sekretu *S*, przy czym $M < S < N$;

wyznaczenie reszt d_p klas dla każdej klasy elementów, przy czym d_p jest równe $[S(\text{mod } p_p)]$;

utworzenie pierwszej wartości binarnej dla każdej klasy elementów, przy czym ta pierwsza wartość binarna jest liczbą pierwszą p_p dla każdej klasy zakodowaną w pierwszej liczbie binarnej mającej do u bitów, i przy czym u jest mniejsze od t, a zarówno u, jak i t są mniejsze od 2.048;

utworzenie drugiej wartości binarnej dla każdej klasy elementów, przy czym ta druga wartość binarna jest resztą d_p dla każdej klasy zakodowaną w drugiej liczbie binarnej

mającej do v bitów, przy czym v jest mniejsze od t, a (u + v = t); i

konkatenacja pierwszej wartości binarnej i drugiej wartości binarnej z utworzeniem sekretu częściowego, P_p, dla każdej klasy elementów, mającego ogólną liczbę bitów równą t.

7. Sposób według zastrz. 6, przy czym każda liczba losowa, r_p, jest generowana z wykorzystaniem generatora liczb losowych, i ma wartość w zakresie od 0 do poniżej 2^u.

8. Sposób według zastrz. 6, przy czym etap generowania liczb pierwszych klas, p_p, dla każdej klasy elementów, obejmuje podawanie liczby losowej dla każdej klasy elementów do funkcji generacji liczb pierwszych, $g(x)$, przy czym x jest liczbą losową o długości do 65.000 znaków.

9. Sposób według zastrz. 8, przy czym $g(x)$ generuje dodatnią liczbę pierwszą o wartości w zakresie od 0 do poniżej 2^v.

10. Sposób według zastrz. 1, przy czym sposób jest inicjowany podczas etapu załadowywania produktu programowego do pierwszego systemu komputerowego.

11. Sposób określania, czy produkt programowy może być wykorzystywany w drugim systemie komputerowym, o drugiej konfiguracji sprzętowej, przy czym drugi system komputerowy jest identyczny lub inny, niż pierwszy system komputerowy, znamienny tym, że sposób obejmuje następujące etapy:

rozpoznanie *n* klas elementów służących do wyznaczenia mocnego identyfikatora sprzętowego (SHWID) generowanego sposobem według zastrz. 1;

rozpoznanie wszystkich wystąpień w każdej klasie elementów drugiej konfiguracji sprzętowej;

pobranie łańcucha identyfikacyjnego, który reprezentuje każde oddzielne wystąpienie elementu;

generowanie liczb pierwszych, $i_{p,q}$, wystąpień dla każdego wystąpienia elementu, przy czym każda liczba pierwsza wystąpienia jest dodatnią liczbą pierwszą, i przy czym p reprezentuje numer danej klasy elementów w zakresie od 1 do *n*, a q reprezentuje q-ty typ elementu w pierwszej konfiguracji sprzętowej lub drugiej konfiguracji sprzętowej;

pobranie potencjalnych sekretów częściowych klas, $P_{p,q}$, przy czym:

$$P_{p,q} = [C_p \pmod{i_{p,q}}];$$

wydzielanie prawdopodobnych liczb pierwszych, $p_{p,q}$, klas, i prawdopodobnych reszt, $d_{p,q}$, klas, z potencjalnych sekretów częściowych $P_{p,q}$, klas;

rozwiązanie zbioru kongruencji otrzymanych z permutacji prawdopodobnych liczb pierwszych klas i prawdopodobnych reszt klas, z utworzeniem potencjalnego sekretu; i

sprawdzanie potencjalnego sekretu przez dekodowanie danego zaszyfrowanego tekstu i porównanie wyniku tego sprawdzenia z odpowiednim znanym tekstem jawnym;

oraz:

załadowanie produktu programowego do drugiego systemu komputerowego jeżeli wynikowy tekst jawny jest zgodny ze znanym tekstem jawnym; i

uniemożliwienie załadowania produktu programowego do drugiego systemu komputerowego jeżeli wynikowy tekst jawny nie jest zgodny ze znanym tekstem jawnym.

12. Sposób według zastrz. 11, przy czym etap generowania liczb pierwszych wystąpień obejmuje podawanie łańcucha identyfikacyjnego, który reprezentuje każde indywidualne wystąpienie elementu do funkcji generacji liczb pierwszych, *f*(x), przy czym x jest łańcuchem identyfikacyjnym o długości do 65.000 znaków.

13. Sposób według zastrz. 12, przy czym *f*(x) generuje dodatnią liczbę pierwszą o wartości większej, niż 2^t, przy czym t jest liczbą całkowitą w zakresie od 32 do 2.048.

14. System obliczeniowy zawierający przynajmniej jeden moduł aplikacyjny nadający się do stosowania w tym systemie obliczeniowym, przy czym ten przynajmniej jeden moduł aplikacyjny zawiera kod aplikacyjny do realizacji sposobu według zastrz. 1.

15. Nośnik odczytywalny komputerowo z zapisanymi na nim wykonywalnymi komputerowo instrukcjami do realizacji sposobu według zastrz. 1.

16. Nośnik odczytywalny komputerowo z zapisanymi na nim wykonywalnymi komputerowo instrukcjami do realizacji sposobu generowania mocnego identyfikatora sprzętowego (SHWID) dla pierwszego systemu komputerowego o pierwszej konfiguracji sprzętowej, znamienny tym, że sposób obejmuje następujące etapy:

rozpoznanie każdej klasy elementów w pierwszej konfiguracji sprzętowej, przy czym liczba klas jest równa *n*;

określenie iloczynu klasy, c_p, dla każdej klasy elementów;

określenie sekretu częściowego, P_p, klasy dla każdej klasy elementów;

wprowadzenie iloczynu klasy, c_p i sekretu częściowego, P_p, do każdej klasy elementów z utworzeniem *n* mocnych identyfikatorów klas,

przy czym tych *n* mocnych identyfikatorów klas tworzy mocny identyfikator sprzętowy (SHWID) pierwszego systemu komputerowego.

17. Nośnik odczytywalny komputerowo według zastrz. 16, przy czym *n* jest liczbą całkowitą do 16.

18. Nośnik odczytywalny komputerowo według zastrz. 16, przy czym iloczyn klasy, c_p, dla każdej klasy elementów jest wyznaczany w następujących etapach:

wybranie liczby *n* klas elementów;

rozpoznanie *n* klas elementów;

rozpoznanie wszystkich wystąpień w każdej klasie elementów;

przyporządkowanie łańcucha identyfikacyjnego do każdego wystąpienia elementu;

generowanie liczb pierwszych, $P_{p,q}$, wystąpień dla każdego wystąpienia elementu;

przy czym każda liczba pierwsza wystąpienia jest dodatnią liczbą pierwszą, i przy czym p reprezentuje numer

danej klasy elementów w zakresie od 1 do *n*, a q reprezentuje q-ty typ elementu w pierwszej konfiguracji sprzętowej; i

mnożenie liczb pierwszych wystąpień w każdej klasie elementów z utworzeniem iloczynu klasy, c_p, dla każdej klasy elementów.

19. Nośnik odczytywalny komputerowo według zastrz. 18, przy czym etap generowania liczb pierwszych wystąpień obejmuje podawanie łańcucha identyfikacyjnego dla każdego wystąpienia elementu do funkcji generacji liczb pierwszych, *f(x)*, *przy czym* x jest łańcuchem identyfikacyjnym o długości do 65.000 znaków.

20. Nośnik odczytywalny komputerowo według zastrz. 19, przy czym *f*(x) generuje dodatnią liczbę pierwszą o wartości większej, niż 2^t, przy czym t jest liczbą całkowitą w zakresie od 32 do 2.048.

21. Nośnik odczytywalny komputerowo według zastrz. 16, przy czym sekret częściowy, P_p, dla każdej klasy elementów jest wyznaczany w następujących etapach:

generowanie liczby losowej, r_p, dla każdej klasy elementów;

generowanie liczb pierwszych p_p klas, dla każdej klasy elementów, przy czym każda liczba pierwsza klasy jest dodatnią liczbą pierwszą

wybór pewnej liczby, *m*, wymaganych zgodności klas elementów, przy czym *m* jest mniejsze od lub równe *n*;

wyznaczenie *N*, przy czym *N* jest równe iloczynowi *m* najmniejszych liczb pierwszych, p_p, klas;

wyznaczenie *M*, przy czym *M* jest równe iloczynowi (*m*-1) największych liczb pierwszych, p_p, klas, a *M* jest mniejsze *N*;

wybranie pewnej liczby sekretu *S*, przy czym $M < S < N$;

wyznaczenie reszt d_p klas dla każdej klasy elementów, przy czym d_p jest równe $[S(\text{mod } p_p)]$;

utworzenie pierwszej wartości binarnej dla każdej klasy elementów, przy czym ta pierwsza wartość binarna jest liczbą pierwszą p_p dla każdej klasy zakodowaną w pierwszej liczbie binarnej mającej do u bitów, i przy czym u jest mniejsze od t, a zarówno u, jak i t są mniejsze od 2.048;

utworzenie drugiej wartości binarnej dla każdej klasy elementów, przy czym ta druga wartość binarna jest resztą d_p dla każdej klasy, zakodowaną w drugiej liczbie binarnej mającej do v bitów, przy czym v jest mniejsze od t, a (u + v = t); i

konkatenacja pierwszej wartości binarnej i drugiej wartości binarnej z utworzeniem sekretu częściowego, P_p, dla każdej klasy elementów, mającego ogólną liczbę bitów równą t.

22. Nośnik odczytywalny komputerowo według zastrz. 21, przy czym każda liczba losowa, r_p, jest generowana z wykorzystaniem generatora liczb losowych, i ma wartość w zakresie od 0 do poniżej 2^u.

23. Nośnik odczytywalny komputerowo według zastrz. 21, przy czym etap generowania liczb pierwszych klas, p_p, dla każdej klasy elementów, obejmuje podawanie liczby losowej dla każdej klasy elementów do funkcji generacji liczb pierw-

szych, $g(x)$, przy czym x jest liczbą losową o długości do 65.000 znaków.

24. Nośnik odczytywalny komputerowo według zastrz. 23, przy czym $g(x)$ generuje dodatnią liczbę pierwszą o wartości w zakresie od 0 do poniżej 2^v.

25. Nośnik odczytywalny komputerowo według zastrz. 16, przy czym sposób jest inicjowany podczas etapu załadowywania produktu programowego do pierwszego systemu komputerowego.

26. Nośnik odczytywalny komputerowo z zapisanymi na nim instrukcjami wykonalnymi komputerowo do realizacji sposobu określania, czy produkt programowy może być wykorzystywany w drugim systemie komputerowym, o drugiej konfiguracji sprzętowej, przy czym drugi system komputerowy jest identyczny lub inny, niż pierwszy system komputerowy, znamienny tym, że sposób obejmuje następujące etapy:

rozpoznanie *n* klas elementów służących do wyznaczenia mocnego identyfikatora sprzętowego (SHWID) generowanego sposobem według zastrz. 16;

rozpoznanie wszystkich wystąpień w każdej klasie elementów drugiej konfiguracji sprzętowej;

pobranie łańcucha identyfikacyjnego, który reprezentuje każde oddzielne wystąpienie elementu;

generowanie liczb pierwszych, $i_{p,q}$, wystąpień dla każdego wystąpienia elementu, przy czym każda liczba pierwsza wystąpienia jest dodatnią liczbą pierwszą, i przy czym p reprezentuje numer danej klasy elementów w zakresie od 1 do *n*, a q reprezentuje q-ty typ elementu w pierwszej konfiguracji sprzętowej lub drugiej konfiguracji sprzętowej;

pobranie potencjalnych sekretów częściowych klas, $P_{p,q}$, przy czym:

$$P_{p,q} = [C_p \pmod{i_{p,q}}];$$

wydzielanie prawdopodobnych liczb pierwszych, $p_{p,q}$, klas, i prawdopodobnych reszt, $d_{p,q}$, klas z potencjalnych sekretów częściowych $P_{p,q}$, klas;

rozwiązanie zbioru kongruencji otrzymanych z permutacji prawdopodobnych liczb pierwszych klas i prawdopodobnych reszt klas, z utworzeniem potencjalnego sekretu; i

sprawdzanie potencjalnego sekretu przez dekodowanie danego zaszyfrowanego tekstu i porównanie wyniku tego sprawdzenia z odpowiednim znanym tekstem jawnym;

oraz:

załadowanie produktu programowego do drugiego systemu komputerowego jeżeli wynikowy tekst jawny jest zgodny ze znanym tekstem jawnym; i

uniemożliwienie załadowania produktu programowego do drugiego systemu komputerowego jeżeli wynikowy tekst jawny nie jest zgodny ze znanym tekstem jawnym.

27. Nośnik odczytywalny komputerowo według zastrz. 26, przy czym etap generowania liczb pierwszych wystąpień obejmuje podawanie łańcucha identyfikacyjnego, który reprezentuje każde indywidualne wystąpienie elementu do funkcji generacji liczb pierwszych, *f(x), przy czym* x jest łańcuchem identyfikacyjnym o długości do 65.000 znaków.

28. Nośnik odczytywalny komputerowo według zastrz. 27, przy czym *f*(x) generuje dodatnią liczbę pierwszą o wartości

większej, niż 2^t, przy czym t jest liczbą całkowitą w zakresie od 32 do 2.048.

29. System obliczeniowy zawierający przynajmniej jeden moduł aplikacyjny nadający się do stosowania w tym systemie obliczeniowym, przy czym ten przynajmniej jeden moduł aplikacyjny zawiera kod aplikacyjny do realizacji sposobu generowania mocnego identyfikatora sprzętowego (SHWID) dla pierwszego systemu komputerowego o pierwszej konfiguracji sprzętowej, znamienny tym, że sposób obejmuje następujące etapy:

rozpoznanie każdej klasy elementów w pierwszej konfiguracji sprzętowej, przy czym liczba klas jest równa *n*;

określenie iloczynu klasy, c_p, dla każdej klasy elementów;

określenie sekretu częściowego, P_p, klasy dla każdej klasy elementów;

wprowadzenie iloczynu klasy, c_p i sekretu częściowego, P_p, do każdej klasy elementów z utworzeniem *n* mocnych identyfikatorów klas,

przy czym tych *n* mocnych identyfikatorów klas tworzy mocny identyfikator sprzętowy (SHWID) pierwszego systemu komputerowego.

30. System obliczeniowy według zastrz. 29, przy czym *n* jest liczbą całkowitą do 16.

31. System obliczeniowy według zastrz. 29, przy czym iloczyn klasy, c_p, dla każdej klasy elementów jest wyznaczany w następujących etapach:

wybranie liczby *n* klas elementów;

rozpoznanie *n* klas elementów;

rozpoznanie wszystkich wystąpień w każdej klasie elementów;

przyporządkowanie łańcucha identyfikacyjnego do każdego wystąpienia elementu;

generowanie liczb pierwszych, $P_{p,q}$, wystąpień dla każdego wystąpienia elementu;

przy czym każda liczba pierwsza wystąpienia jest dodatnią liczbą pierwszą, i przy czym p reprezentuje numer danej klasy elementów w zakresie od 1 do *n*, a q reprezentuje q-ty typ elementu w pierwszej konfiguracji sprzętowej; i

mnożenie liczb pierwszych wystąpień w każdej klasie elementów z utworzeniem iloczynu klasy, c_p, dla każdej klasy elementów.

32. System obliczeniowy według zastrz. 31, przy czym etap generowania liczb pierwszych wystąpień obejmuje podawanie łańcucha identyfikacyjnego dla każdego wystąpienia elementu do funkcji generacji liczb pierwszych, *f(x), przy czym* x jest łańcuchem identyfikacyjnym o długości do 65.000 znaków.

33. System obliczeniowy według zastrz. 32, przy czym *f*(x) generuje dodatnią liczbę pierwszą o wartości większej, niż 2^t, przy czym t jest liczbą całkowitą w zakresie od 32 do 2.048.

34. System obliczeniowy według zastrz. 29, przy czym sekret częściowy, P_p, dla każdej klasy elementów jest wyznaczany w następujących etapach:

generowanie liczby losowej, r_p, dla każdej klasy elementów;

generowanie liczb pierwszych p_p klas, dla każdej klasy elementów, przy czym każda liczba pierwsza klasy jest dodatnią liczbą pierwszą

wybór pewnej liczby, *m*, wymaganych zgodności klas elementów, przy czym *m* jest mniejsze od lub równe *n*;

wyznaczenie *N*, przy czym *N* jest równe iloczynowi *m* najmniejszych liczb pierwszych, p_p, klas;

wyznaczenie *M*, przy czym *M* jest równe iloczynowi (*m*-1) największych liczb pierwszych, p_p, klas, a *M* jest mniejsze *N*;

wybranie pewnej liczby sekretu *S*, przy czym $M < S < N$;

wyznaczenie reszt d_p klas dla każdej klasy elementów, przy czym d_p jest równe $[S(\text{mod } p_p)]$;

utworzenie pierwszej wartości binarnej dla każdej klasy elementów, przy czym ta pierwsza wartość binarna jest liczbą pierwszą p_p dla każdej klasy zakodowaną w pierwszej liczbie binarnej mającej do u bitów, i przy czym u jest mniejsze od t, a zarówno u, jak i t są mniejsze od 2.048;

utworzenie drugiej wartości binarnej dla każdej klasy elementów, przy czym ta druga wartość binarna jest resztą klasy, d_p, dla każdej klasy, zakodowaną w drugiej liczbie

binarnej mającej do v bitów, przy czym v jest mniejsze od t, a (u + v = t); i

konkatenacja pierwszej wartości binarnej i drugiej wartości binarnej z utworzeniem sekretu częściowego, P_p, dla każdej klasy elementów, mającego ogólną liczbę bitów równą t.

35. System obliczeniowy według zastrz. 34, przy czym każda liczba losowa, r_p, jest generowana z wykorzystaniem generatora liczb losowych, i ma wartość w zakresie od 0 do poniżej 2^u.

36. System obliczeniowy według zastrz. 34, przy czym etap generowania liczb pierwszych klas, p_p, dla każdej klasy elementów, obejmuje podawanie liczby losowej dla każdej klasy elementów do funkcji generacji liczb pierwszych, $g(x)$, przy czym x jest liczbą losową o długości do 65.000 znaków.

37. System obliczeniowy według zastrz. 36, przy czym $g(x)$ generuje dodatnią liczbę pierwszą o wartości w zakresie od 0 do poniżej 2^v.

38. System obliczeniowy według zastrz. 29, przy czym sposób jest inicjowany podczas etapu załadowywania produktu programowego do pierwszego systemu komputerowego.

39. System obliczeniowy zawierający przynajmniej jeden moduł aplikacyjny nadający się do stosowania w tym systemie obliczeniowym, przy czym ten przynajmniej jeden moduł aplikacyjny zawiera kod aplikacyjny do realizacji sposobu określania, czy produkt programowy może być wykorzystywany w drugim systemie komputerowym, o drugiej konfiguracji

sprzętowej, przy czym drugi system komputerowy jest identyczny lub inny, niż pierwszy system komputerowy, znamienny tym, że sposób obejmuje następujące etapy:

rozpoznanie *n* klas elementów służących do wyznaczenia mocnego identyfikatora sprzętowego (SHWID) generowanego sposobem według zastrz. 29;

rozpoznanie wszystkich wystąpień w każdej klasie elementów drugiej konfiguracji sprzętowej;

pobranie łańcucha identyfikacyjnego, który reprezentuje każde oddzielne wystąpienie elementu;

generowanie liczb pierwszych, $i_{p,q}$, wystąpień dla każdego wystąpienia elementu, przy czym każda liczba pierwsza wystąpienia jest dodatnią liczbą pierwszą, i przy czym p reprezentuje numer danej klasy elementów w zakresie od 1 do *n*, a q reprezentuje q-ty typ elementu w pierwszej konfiguracji sprzętowej lub drugiej konfiguracji sprzętowej;

pobranie potencjalnych sekretów częściowych klas, $P_{p,q}$, przy czym:

$$P_{p,q} = [C_p \pmod{i_{p,q}}];$$

wydzielanie prawdopodobnych liczb pierwszych, $p_{p,q}$, klas, i prawdopodobnych reszt, $d_{p,q}$, klas, z potencjalnych sekretów częściowych $P_{p,q}$, klas;

rozwiązanie zbioru kongruencji otrzymanych z permutacji prawdopodobnych liczb pierwszych klas i prawdopodobnych reszt klas, z utworzeniem potencjalnego sekretu; i

sprawdzanie potencjalnego sekretu przez dekodowanie danego zaszyfrowanego tekstu i porównanie wyniku tego sprawdzenia z odpowiednim znanym tekstem jawnym;

oraz:

załadowanie produktu programowego do drugiego systemu komputerowego jeżeli wynikowy tekst jawny jest zgodny ze znanym tekstem jawnym; i

uniemożliwienie załadowania produktu programowego do drugiego systemu komputerowego jeżeli wynikowy tekst jawny nie jest zgodny ze znanym tekstem jawnym.

40. System obliczeniowy według zastrz. 39, przy czym etap generowania liczb pierwszych wystąpień obejmuje podawanie łańcucha identyfikacyjnego, który reprezentuje każde indywidualne wystąpienie elementu do funkcji generacji liczb pierwszych, $f(x)$, przy czym x jest łańcuchem identyfikacyjnym o długości do 65.000 znaków.

41. System obliczeniowy według zastrz. 40, przy czym f(x) generuje dodatnią liczbę pierwszą o wartości większej, niż 2^t, przy czym t jest liczbą całkowitą w zakresie od 32 do 2.048.

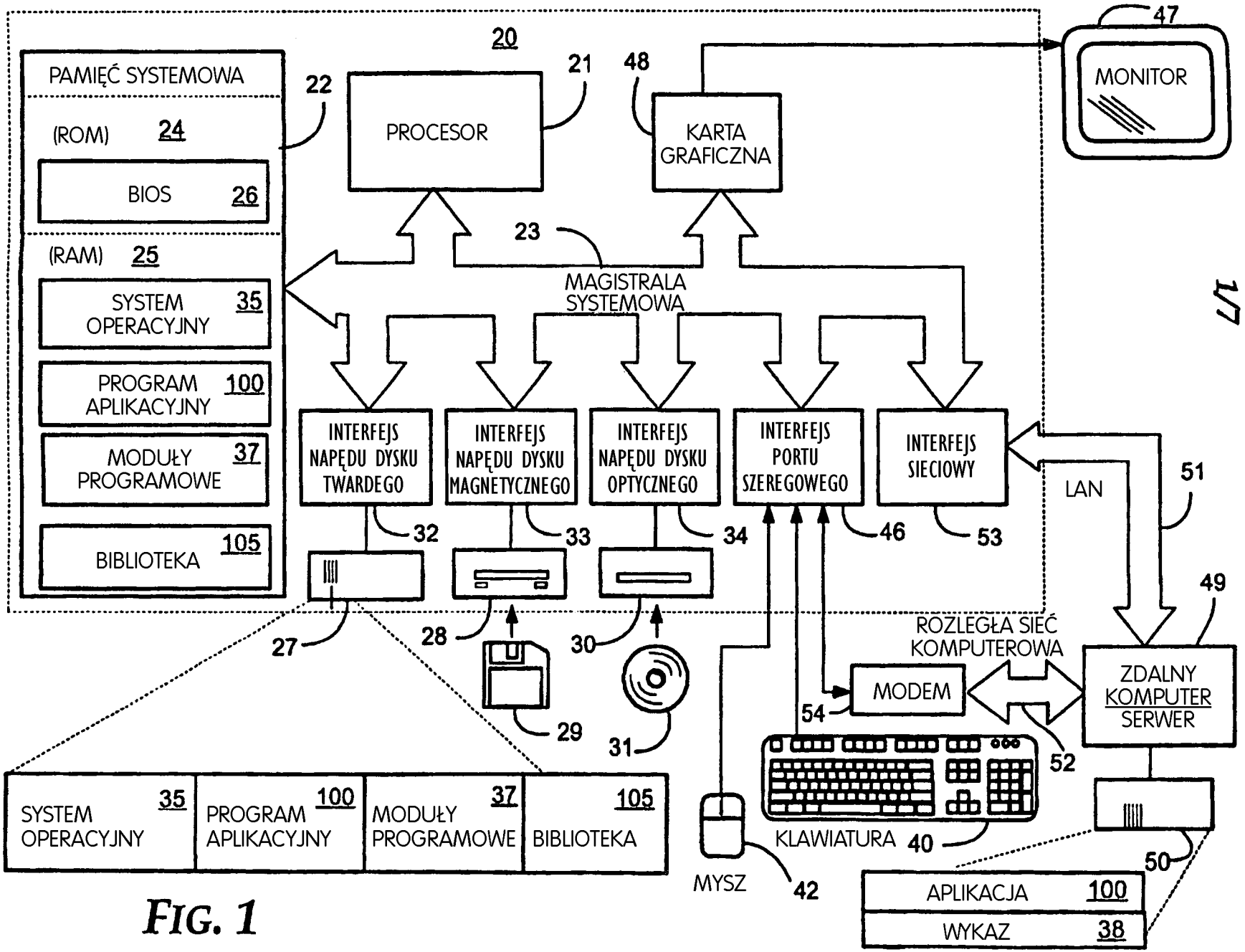
20
PAMIĘĆ SYSTEMOWA
22
(ROM) 24
BIOS 26
(RAM) 25
SYSTEM OPERACYJNY 35
PROGRAM APLIKACYJNY 100
MODUŁY PROGRAMOWE 37
BIBLIOTEKA 105
PROCESOR 21
48
KARTA GRAFICZNA
47
MONITOR
23
MAGISTRALA SYSTEMOWA
INTERFEJS NAPĘDU DYSKU TWARDEGO
INTERFEJS NAPĘDU DYSKU MAGNETYCZNEGO
INTERFEJS NAPĘDU DYSKU OPTYCZNEGO
INTERFEJS PORTU SZEREGOWEGO
INTERFEJS SIECIOWY
LAN
51
32
33
34
46
53
27
28
30
29
31
49
ROZLEGŁA SIEĆ KOMPUTEROWA
MODEM
54
ZDALNY KOMPUTER SERWER
52
KLAWIATURA 40
MYSZ 42
50
APLIKACJA 100
WYKAZ 38
SYSTEM OPERACYJNY 35
PROGRAM APLIKACYJNY 100
MODUŁY PROGRAMOWE 37
BIBLIOTEKA 105

FIG. 1

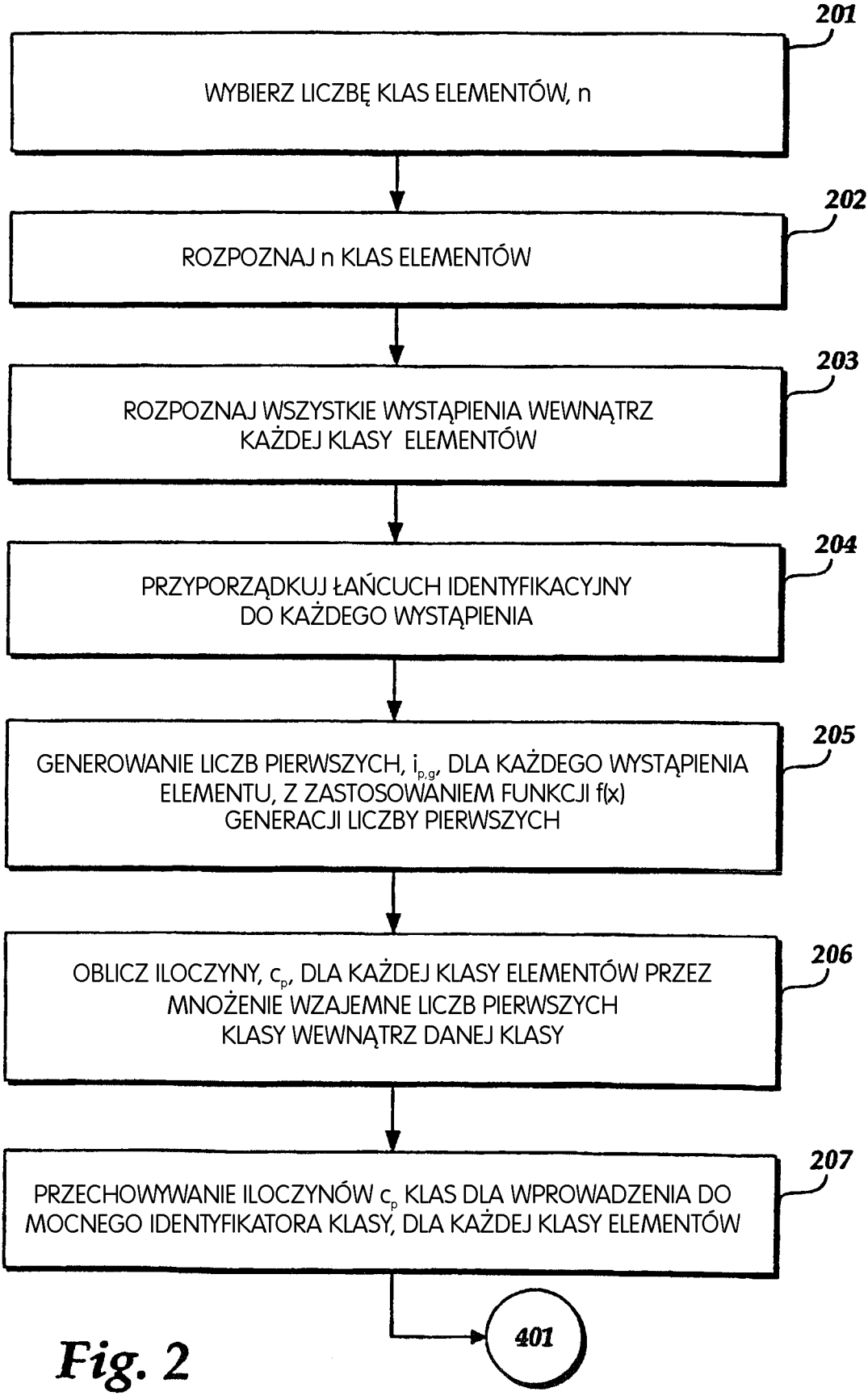
201
WYBIERZ LICZBĘ KLAS ELEMENTÓW, n
202
ROZPOZNAJ n KLAS ELEMENTÓW
203
ROZPOZNAJ WSZYSTKIE WYSTĄPIENIA WEWNĄTRZ KAŻDEJ KLASY ELEMENTÓW
204
PRZYPORZĄDKUJ ŁAŃCUCH IDENTYFIKACYJNY DO KAŻDEGO WYSTĄPIENIA
205
GENEROWANIE LICZB PIERWSZYCH, $i_{p,g}$, DLA KAŻDEGO WYSTĄPIENIA ELEMENTU, Z ZASTOSOWANIEM FUNKCJI f(x) GENERACJI LICZBY PIERWSZYCH
206
OBLICZ ILOCZYNY, c_p, DLA KAŻDEJ KLASY ELEMENTÓW PRZEZ MNOŻENIE WZAJEMNE LICZB PIERWSZYCH KLASY WEWNĄTRZ DANEJ KLASY
207
PRZECHOWYWANIE ILOCZYNÓW c_p KLAS DLA WPROWADZENIA DO MOCNEGO IDENTYFIKATORA KLASY, DLA KAŻDEJ KLASY ELEMENTÓW
401

Fig. 2

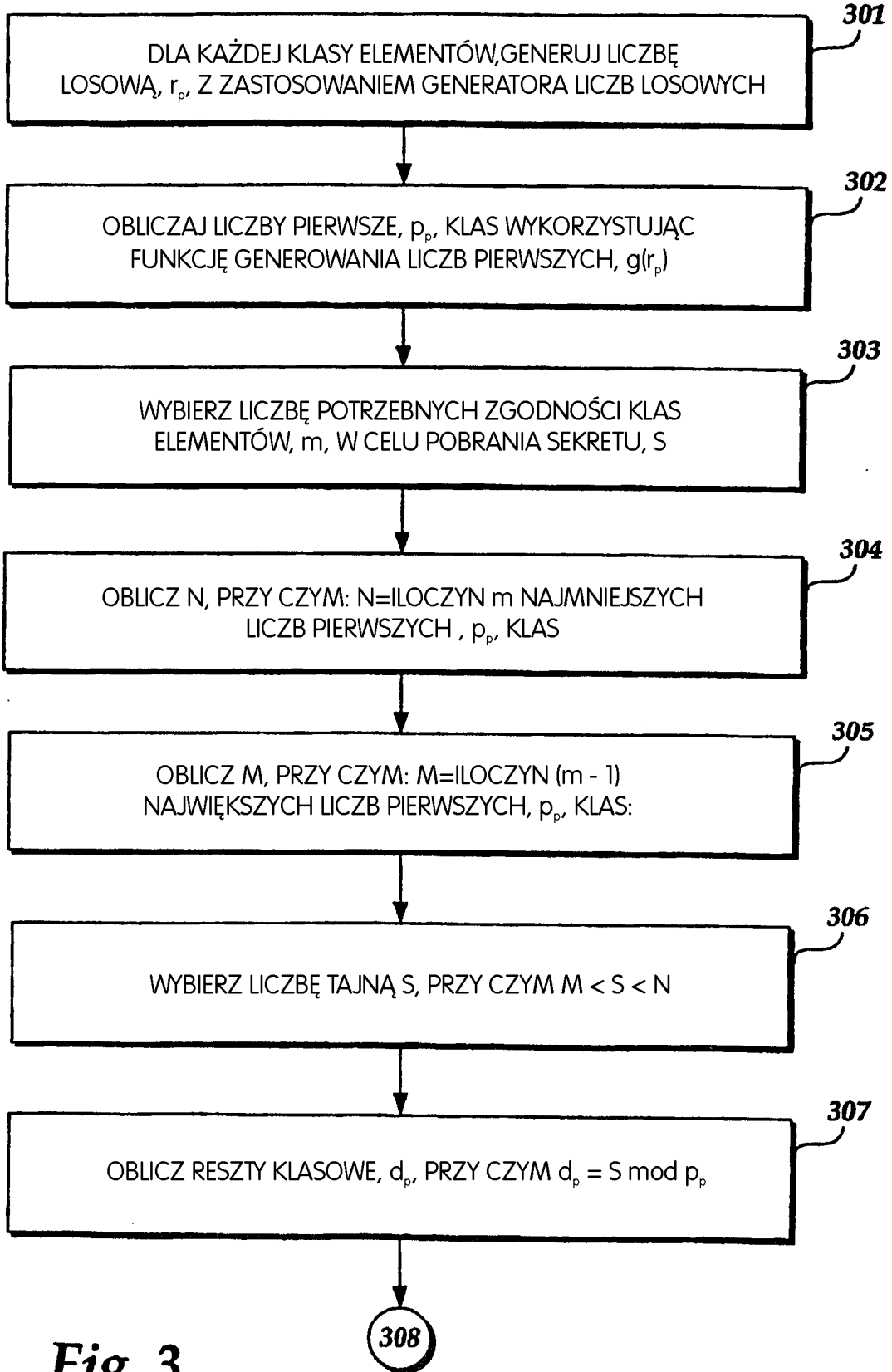
301
DLA KAŻDEJ KLASY ELEMENTÓW,GENERUJ LICZBĘ LOSOWĄ, r_p, Z ZASTOSOWANIEM GENERATORA LICZB LOSOWYCH
302
OBLICZAJ LICZBY PIERWSZE, p_p, KLAS WYKORZYSTUJĄC FUNKCJĘ GENEROWANIA LICZB PIERWSZYCH, $g(r_p)$
303
WYBIERZ LICZBĘ POTRZEBNYCH ZGODNOŚCI KLAS ELEMENTÓW, m, W CELU POBRANIA SEKRETU, S
304
OBLICZ N, PRZY CZYM: N=ILOCZYN m NAJMNIEJSZYCH LICZB PIERWSZYCH , p_p, KLAS
305
OBLICZ M, PRZY CZYM: M=ILOCZYN (m - 1) NAJWIĘKSZYCH LICZB PIERWSZYCH, p_p, KLAS:
306
WYBIERZ LICZBĘ TAJNĄ S, PRZY CZYM M < S < N
307
OBLICZ RESZTY KLASOWE, d_p, PRZY CZYM $d_p = S \bmod p_p$
308

Fig. 3

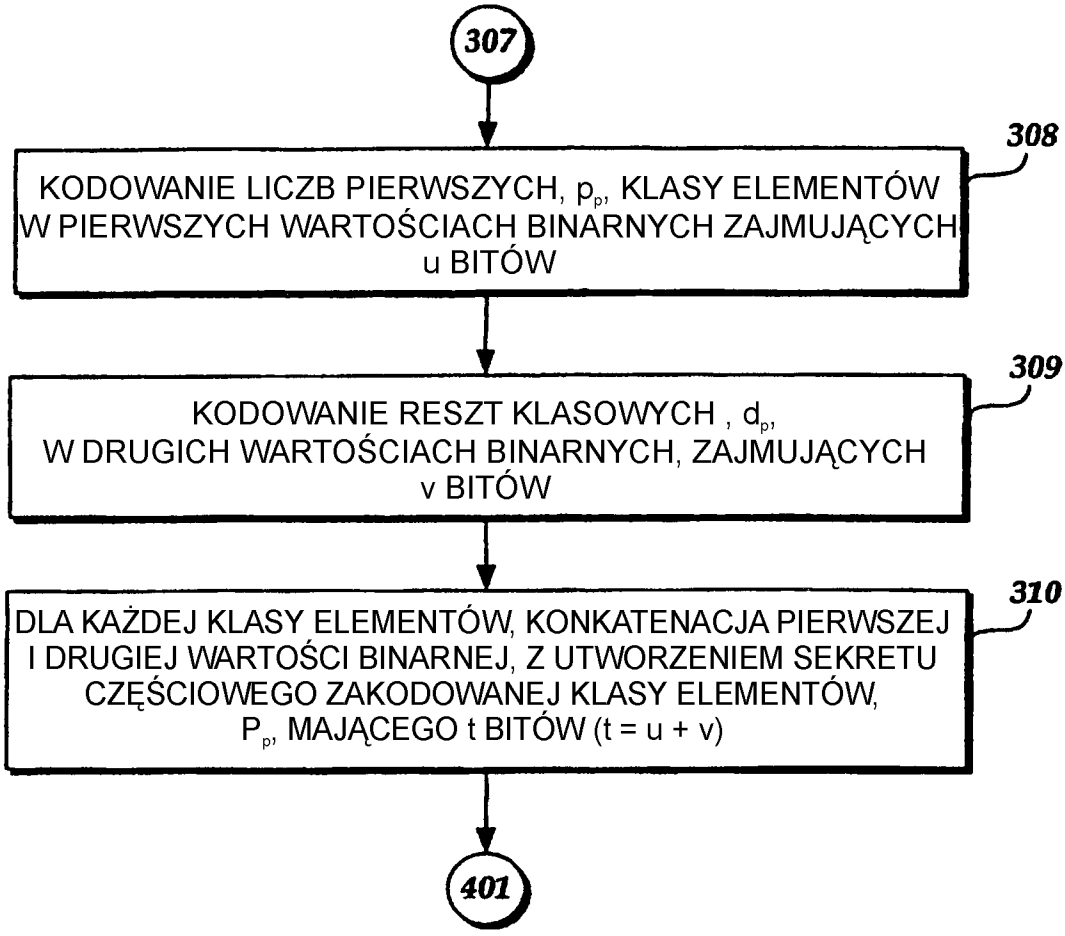
307
308
KODOWANIE LICZB PIERWSZYCH, p_p, KLASY ELEMENTÓW W PIERWSZYCH WARTOŚCIACH BINARNYCH ZAJMUJĄCYCH u BITÓW
309
KODOWANIE RESZT KLASOWYCH , d_p, W DRUGICH WARTOŚCIACH BINARNYCH, ZAJMUJĄCYCH v BITÓW
310
DLA KAŻDEJ KLASY ELEMENTÓW, KONKATENACJA PIERWSZEJ I DRUGIEJ WARTOŚCI BINARNEJ, Z UTWORZENIEM SEKRETU CZĘŚCIOWEGO ZAKODOWANEJ KLASY ELEMENTÓW, P_p, MAJĄCEGO t BITÓW (t = u + v)
401

Fig. 4

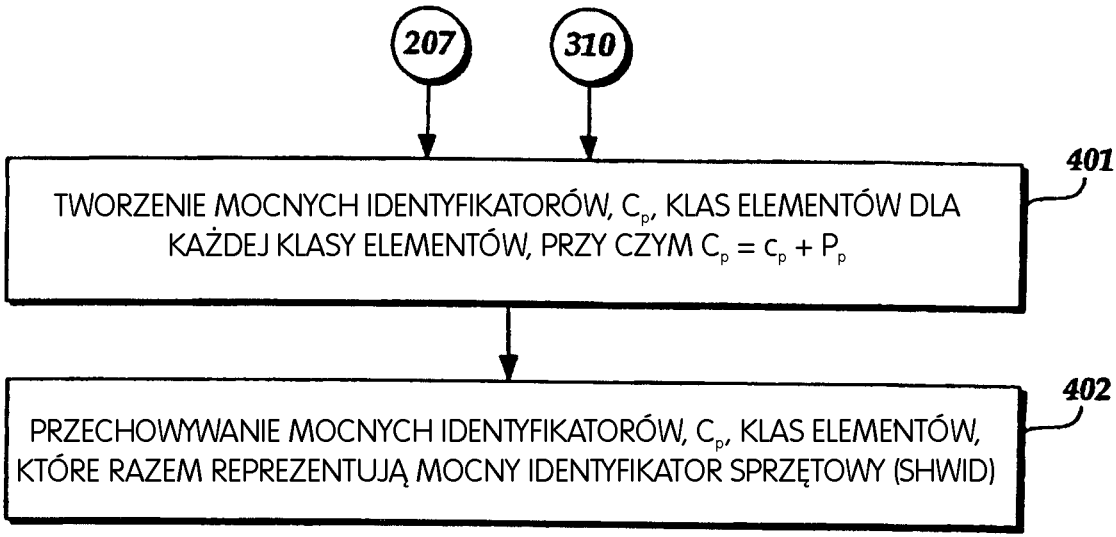
207
310
401
TWORZENIE MOCNYCH IDENTYFIKATORÓW, C_p, KLAS ELEMENTÓW DLA KAŻDEJ KLASY ELEMENTÓW, PRZY CZYM $C_p = c_p + P_p$
402
PRZECHOWYWANIE MOCNYCH IDENTYFIKATORÓW, C_p, KLAS ELEMENTÓW, KTÓRE RAZEM REPREZENTUJĄ MOCNY IDENTYFIKATOR SPRZĘTOWY (SHWID)

Fig. 5

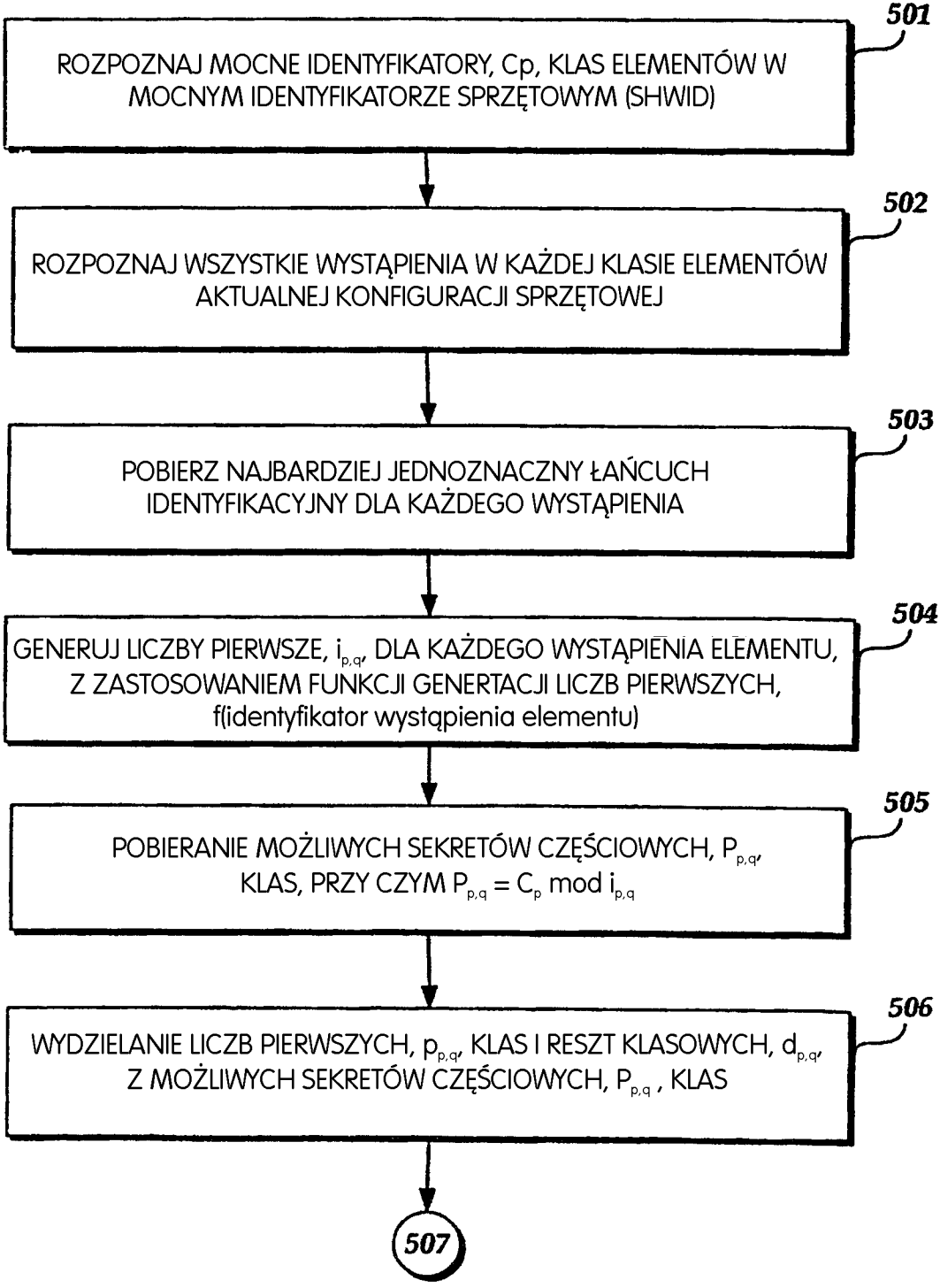
501
ROZPOZNAJ MOCNE IDENTYFIKATORY, Cp, KLAS ELEMENTÓW W MOCNYM IDENTYFIKATORZE SPRZĘTOWYM (SHWID)
502
ROZPOZNAJ WSZYSTKIE WYSTĄPIENIA W KAŻDEJ KLASIE ELEMENTÓW AKTUALNEJ KONFIGURACJI SPRZĘTOWEJ
503
POBIERZ NAJBARDZIEJ JEDNOZNACZNY ŁAŃCUCH IDENTYFIKACYJNY DLA KAŻDEGO WYSTĄPIENIA
504
GENERUJ LICZBY PIERWSZE, $i_{p,q}$, DLA KAŻDEGO WYSTĄPIENIA ELEMENTU, Z ZASTOSOWANIEM FUNKCJI GENERTACJI LICZB PIERWSZYCH, f(identyfikator wystąpienia elementu)
505
POBIERANIE MOŻLIWYCH SEKRETÓW CZĘŚCIOWYCH, $P_{p,q}$, KLAS, PRZY CZYM $P_{p,q} = C_p \bmod i_{p,q}$
506
WYDZIELANIE LICZB PIERWSZYCH, $p_{p,q}$, KLAS I RESZT KLASOWYCH, $d_{p,q}$, Z MOŻLIWYCH SEKRETÓW CZĘŚCIOWYCH, $P_{p,q}$, KLAS
507

Fig. 6

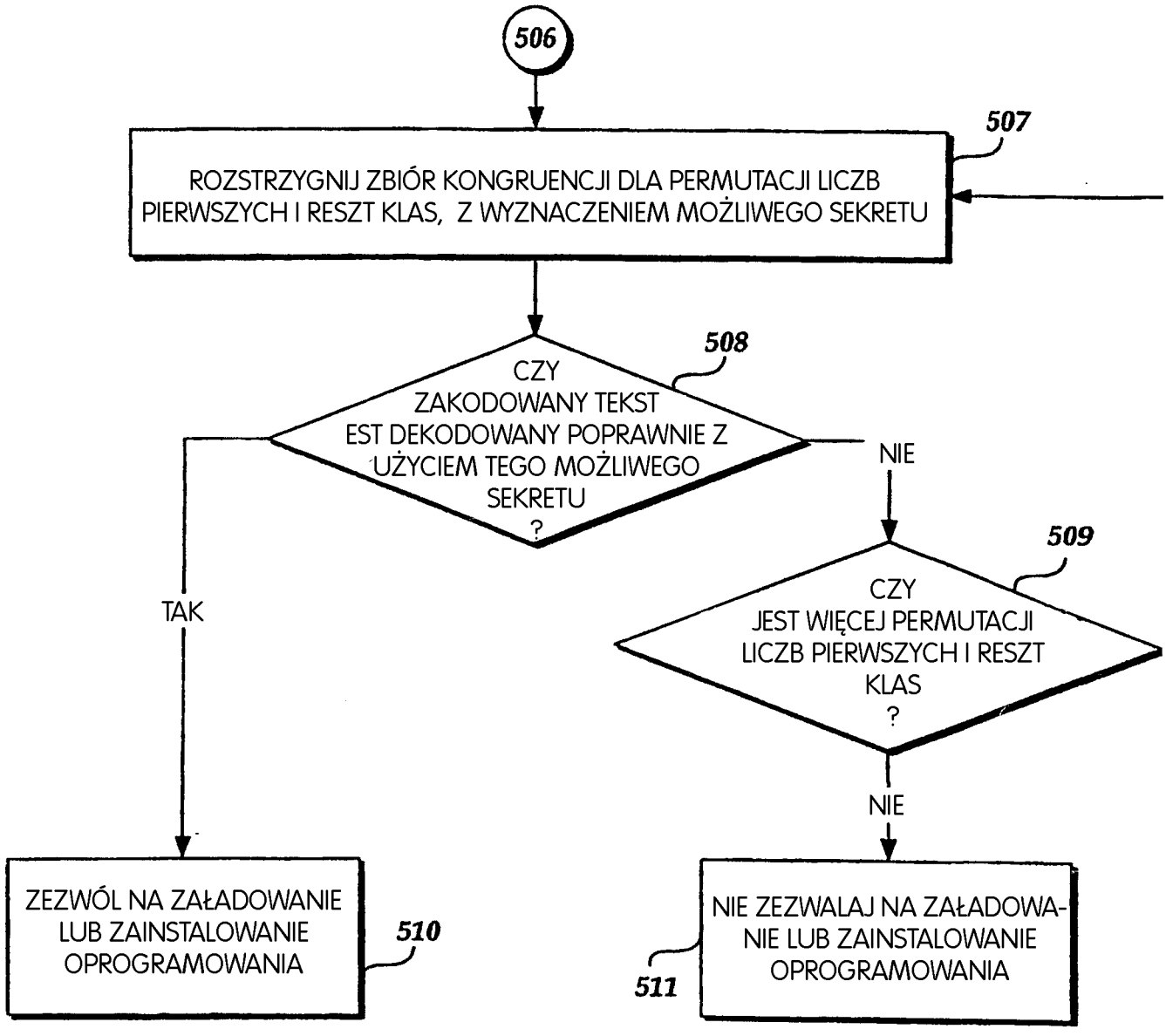
506
507
ROZSTRZYGNIJ ZBIÓR KONGRUENCJI DLA PERMUTACJI LICZB PIERWSZYCH I RESZT KLAS, Z WYZNACZENIEM MOŻLIWEGO SEKRETU
508
CZY ZAKODOWANY TEKST EST DEKODOWANY POPRAWNIE Z UŻYCIEM TEGO MOŻLIWEGO SEKRETU ?
NIE
TAK
509
CZY JEST WIĘCEJ PERMUTACJI LICZB PIERWSZYCH I RESZT KLAS ?
NIE
ZEZWÓL NA ZAŁADOWANIE LUB ZAINSTALOWANIE OPROGRAMOWANIA
510
NIE ZEZWALAJ NA ZAŁADOWANIE LUB ZAINSTALOWANIE OPROGRAMOWANIA
511

Fig. 7