Bezpieczeństwo danych w sieciach komputerowych a X.500 Directory

Maja Górecka
Maja.Gorecka@cc.uni.torun.pl

Tomasz Wolniewicz
twoln@mat.uni.torun.pl

1996–03–07

Uniwersytet Mikołaja Kopernika w Toruniu

1 Wstęp

W niniejszym raporcie przestawiamy sprawozdanie z prac nad problemem bezpieczeństwa danych w sieciach komputerowych prowadzonych w dwóch kierunkach:

1. wykorzystania bazy X.500 jako zasobu wspierającego bezpieczną wymianę informacji,

2. zabezpieczenia samej bazy X.500 przed niepowołanym dostępem.

Przedstawiamy podstawy rozwijających się obecnie systemów bezpiecznej transmisji danych (PGP i PEM) i wskazujemy na rolę jaką X.500 Directory powinno odegrać w konsolidowaniu tych systemów. W części 7 opisujemy konkretną implementację systemu PEM zintegrowaną z X.500 Directory. Implementacja ta została próbnie uruchomiona na UMK w Toruniu i będzie przedmiotem dalszych testów w najbliższej przyszłości.

Dużo miejsca poświęcamy też bezpieczeństwu samej bazy X.500. Przedstawimy zabezpieczenia zdefiniowane w standardzie X.500 oraz zaimplementowane w oprogramowaniu QUIPU. Omawiamy rozszerzenia wprowadzone w standardzie X.500'93, mające na celu podniesienie bezpieczeństwa zasobów.

Nasze opracowanie opiera się w znacznej części na książce Davida Chadwicka p.t. „Understanding X.500 the Directory" [1]. Wykorzystujemy też dokumentację pakietu SecuDE [8], RFC opisujące systemy PEM i PGP, ksiażkę Simsona Garfinkela p.t. „PGP — Pretty Good Privacy" [2] oraz dyskusje prowadzone na liście administratorów X.500.

2 Zagrożenia w systemach komunikacyjnych i rodzaje ochrony danych

Istotnym wymaganiem większości systemów eksploatowanych w środowisku sieciowym jest konieczność ochrony zasobów danych przed dostępem niepożądanych użytkowników.

Informacja chroniona może być różnego typu: bazy danych, zasoby informacyjne, usługi komputerowe, czy prywatne dane, korespondencja, kontrakty, umowy. Na ogół proste metody zabezpieczenia takich danych za pomocą hasła nie są wystarczające.

Bezpieczeństwo systemu leży w interesie zarówno użytkownika, który musi mieć pewność, że stosowana usługa jest wiarygodna, jak i dostawcy serwisu, chroniącego swój system przed nieuprzywilejowanym dostępem.

Podstawowe zagrożenia, z jakimi mamy do czynienia w systemach komunikacyjnych to:

- przechwycenie danych identyfikujących nadawczę czy odbiorcę komunikatu przez osobę trzecią w celu późniejszego wykorzystania,
- nieuprawnione rozpowszechnianie informacji, czyli udostępnianie danych niepożądanym odbiorcom,
- nieuprawniona modyfikacja danych (zmiana, duplikacja, zniekształcenie),
- występowanie w imieniu innego użytkownika, „podszycie się" (*masquerade*) w celu uzyskania dostępu do zasobów danych czy zdobycia dodatkowych przywilejów,
- powtórzenie wcześniej przechwyconego komunikatu w dowolnej postaci (*replay*),
- odmówienie udziału w komunikacji lub jej części,
- zablokowanie wykonania określonej usługi czy operacji.

Systemy bezpieczeństwa w celu zminimalizowania opisanych zagrożeń realizują następujące funkcje:

- identyfikacja tożsamości stron biorących udział w komunikacji (poświadczanie proste lub silne, rozdział 6.1),
- kontrola dostępu do zasobów (rozdział 6.4),
- poufny transfer komunikatów w celu ochrony danych (szyfrowanie za pomocą dostępnych technik kryptograficznych, rozdział 3),
- kontrola integralności danych (podpisy cyfrowe, rozdział 3.3),
- uniemożliwienie nieuzasadnionej odmowy komunikacji (weryfikacja danych identyfikacyjnych na podstawie podpisów cyfrowych, rozdział 3.3),

Techniki bezpieczeństwa umożliwiają ochronę przenoszonych danych na dwa sposoby: przez ich zaszyfrowanie przed nadaniem, wówczas mamy do czynienia z poufną wymianą danych lub wyłącznie przez kontrolę spójności komunikatu, metoda ta gwarantuje, że informacja odebrana jest identyczna z nadaną.

3 Podstawy kryptografii wykorzystywane w publicznych systemach przesyłania danych

Podstawowym celem kryptografii jest przekształcenie informacji do postaci, która uniemożliwi niepowołanym osobom odtworzenie przesyłanej wiadomości.

Czasami stosuje się przekształcanie za pomocą funkcji nieodwracalnych. Ponieważ taka funkcja (w znaczeniu matematycznym) zastosowana do jakiegoś argumentu zawsze produkuje ten sam rezultat, funkcje nieodwracalne można stosować do sprawdzania czy informacja nie została zmieniona. Oczywiście dobór odpowiednich funkcji jest skomplikowanym problemem, jeżeli chce się minimalizować prawdopodobieństwo otrzymania identycznych rezultatów na różnych argumentach (funkcje takie rzadko są różnowartościowe). Kiedy mówimy o funkcji nieodwracalnej, różnowartościowej, to zdajemy sobie sprawę, że z matematycznego punktu widzenia jest to sytuacja niemożliwa, nieodwracalność rozumiemy w sensie złożoności obliczeniowej algorytmu odwracania.

Innym zastosowaniem kryptografii jest szyfrowanie wiadomości za pomocą funkcji odwracalnej. Zastosowanie funkcji odwrotnej przywraca informacje do postaci niezaszyfrowanej. Podstawowym problemem w takiej sytuacji jest to, że przy powszechnym stosowaniu kryptografii nie można stosować jednej pary funkcji. W tym celu wprowadza się do operacji szyfrowania dodatkowe argumenty (klucze). Działanie funkcji szyfrującej i deszyfrującej można opisać następującą formułą:

$$f_D(k_D, f_S(k_S, W)) = W$$

gdzie f_S i f_D oznaczają odpowiednio funkcję szyfrującą i deszyfrującą, k_S i k_D — klucze szyfrujący i deszyfrujący, W — wiadomość poddawaną szyfrowaniu. Jeżeli $k_S = k_S$, to mówimy o algorytmie symetrycznym, w przeciwnym wypadku o niesymetrycznym.

3.1 Algorytmy symetryczne

Istotnym problemem wszelkich algorytmów symetrycznych jest wymaganie współdzielenia z odbiorcą tajnego klucza, gdyż obaj partnerzy wymieniający poufne dane muszą dysponować tym samym kluczem. Ponadto w przypadku użytkowników odległych konieczna jest transmisja klucza przez sieć. Przesyłanie z kodowaniem symetrycznym stosuje metodę tworzenia niezależnych kluczy.

Pierwszy, jednorazowy (*session key, data exchange key*) stosuje się dla potrzeb jednokrotnego przesłania danych, drugi (*interchange key*) jest stosowany do szyfrowania pierwszego. Pociąga to za sobą konieczność częstego tworzenia nowych kluczy, co wprowadza wiele trudności w zarządzaniu bazą takich zasobów.

Najpopularniejszym algorytmem symetrycznym jest **Data Encription Standard** — DES, opracowany przez Amerykańskie Biuro Standardów. Ten rodzaj szyfrowania jest bardzo efektywny (istnieją układy realizujące ten algorytm sprzętowo z szybkością 10Mb/s),

Inny standard (stosowany przez system PGP), to **International Data Encription Algorithm** — IDEA, opracowany w Szwajcarii.

Obydwa standardy uważane są obecnie za bezpieczne.

3.2 Algorytmy asymetryczne

Wykorzystanie algorytmu niesymetrycznego powoduje, że każdy użytkownik dysponuje parą kluczy, jeden, zwany kluczem publicznym (*public*) jest ogólnie dostępny, drugi, prywatny (*private*), jest tajny, znany wyłącznie właścicielowi. Metoda ta zwana jest również systemem kryptografii kluczy publicznych (*Public-Key CryptoSystem* — PKCS). W klasie algorytmów asymetrycznych istnieją techniki kodowania **odwracalnego** (*reversible*) oraz **nieodwracalnego** (*irreversible*). Przy szyfrowaniu odwracalnym można zakodowaną informację odzyskać poprzez zastosowanie odpowiedniego klucza. Powszechnie stosowaną implementacją tej techniki jest algorytm RSA.

Kryptografia asymetryczna wykorzystuje bardzo skomplikowane algorytmy matematyczne, co znacznie obniża efektywność. Nie zaleca się stosowania tych technik do poufnego przesyłania komunikatów o dużych rozmiarach. Korzystniej jest zastosować algorytm symetryczny z jednorazowym kluczem, a za pomocą odwracalnej kryptografii asymetrycznej przesłać poufny klucz.

Wymiana poufnej informacji między użytkownikiem A i B przebiega następująco: A i B posiadają po parze kluczy (k_S^A, k_D^A), (k_S^B, k_D^B). Klucze k_S^A i k_S^B są ogólnie dostępne.

- A generuje klucz jednorazowy k_{sym} algorytmu symetrycznego F_S^{sym}.
- A szyfruje wiadomość W kluczem k_{sym} przy pomocy algorytmu F_S^{sym} do postaci
$$Ws = F_S^{\mathrm{sym}}(k_{\mathrm{sym}}, W)$$
- A szyfruje klucz k_{sym} algorytmem asymetrycznym F_S^{asym} do postaci
$$ks = F_S^{\mathrm{asym}}(k_S^B, k_{\mathrm{sym}})$$
i wysyła komunikat w postaci: ks, Ws,
- B odkodowuje najpierw klucz k_{sym}, a następnie wiadomość W — nikt inny nie jest w stanie odszyfrować tej informacji.
- B może w identyczny sposób wysyłać poufne informacje do użytkownika A (koduje kluczem k_S^A.

.

Opisana metoda przesyłania informacji jest niezależna od konkretnego algorytmu, wydaje się jednak, że znajomość najczęściej stosowanego algorytmu RSA, znakomicie przybliża zrozumienie metody działania kryptografii z kluczem publicznym, a w szczególności możliwość zastosowania jej do tzw. podpisów cyfrowych.

Algorytm RSA

Algorytm oparty jest na następujących twierdzeniach matematycznych:

Tw. 1 Jeżeli n jest dowolną liczbą naturalną, a $\phi(n)$ oznacza ilość liczb względnie pierwszych z n, to dla dowolnej liczby naturalnej N zachodzi:
$$N^{\phi(n)} \equiv 1(\mathrm{mod}\ n).$$

Tw. 2 Jeżeli liczby naturalne e i m są względnie pierwsze to istnieje liczba naturalna d taka, że
$$ed \equiv 1(\mathrm{mod}\ m).$$

W powyższych twierdzeniach symbol $\equiv$ oznacza kongruencję (lub przystawanie). Mówimy, że $a \equiv b(\mathrm{mod}\ n)$ jezeli $a - b$ jest podzielne przez n, czy (równoważnie), że a i b dają identyczną resztę z dzielenia przez n.

Ponieważ, gdy p i q są liczbami pierwszymi, to $\phi(pq) = (p-1)(q-1)$, z Tw. 1 mamy następujący:

Wn. 1 Jeżeli p i q są liczbami pierwszymi, a $n = pq$, to dla dowolnej liczby naturalnej N zachodzi
$$N^{(p-1)(q-1)} \equiv 1(\mathrm{mod}\ n).$$

W algorytmie RSA wybiera się dwie duże liczby pierwsze p i q (obecnie zaleca się stosowanie p i q takich, aby ich iloczyn n zawierał się w zakresie 2^{512}–2^{1024}). Liczby te muszą zostać wygenerowane losowo, tak, aby było minimalne prawdopodobieństwo przydzielenia tej samej pary dwukrotnie. Następnie wybiera się liczbę e, względnie pierwszą z $(p-1)(q-1)$, standard X.509 sugeruje, aby używać $e = 2^{16}+1$ — jest to liczba pierwsza. Do e dobiera się d, tak aby zgodnie z Tw. 2

$$ed \equiv 1(\mathrm{mod}\ (p-1)(q-1). \tag{1}$$

Para (e, n) jest kluczem publicznym, a para (d, n) — kluczem prywatnym.

Szyfrowanie wiadomości polega najpierw na zamianie jej na ciąg liczb (M_i) mniejszych od n, ale dostatecznie dużych by $(M_i)^e > n$. Następnie tworzy się ciąg $L_i = (M_i)^e \mathrm{mod}\ n$, ciąg ten jest zaszyfrowaną wiadomością. Odbiorca ciągu (L_i) korzyst z równości

$$M_i = (L_i)^d \mathrm{mod}\ n \tag{2}$$

dla odzyskania ciągu (M_i), a co za tym idzie całej wysyłanej informacji.

Uzasadnienia wymaga jeszcze równość (2). Z (1) wynika, że istnieje liczba naturalna j taka, że $ed = j(p-1)(q-1)+1$. Mamy teraz następujący ciąg wynikających z siebie kongruencji (dla uproszczenia opuszczamy indeksy przy M i L):

$$M^e \equiv L(\mathrm{mod}\ n) \tag{3}$$

$$L^d \equiv (M^e)^d = M^{ed} = M^{j(p-1)(q-1)+1} = M(M^j)^{(p-1)(q-1)}(\mathrm{mod}\ n) \tag{4}$$

Ponieważ, na podstawie Wn. 1, $(M^j)^{(p-1)(q-1)} \equiv 1(\mathrm{mod}\ n)$, to po pomnożeniu obu stron przez M, dostajemy na podstawie (4):

$$L^d \equiv M(\mathrm{mod}\ n).$$

Ale $M < n$, więc $M = L^d \mathrm{mod}\ n$, co mieliśmy uzasadnić.

Analiza algorytmu RSA pozwala stwierdzić, że przy znajomości klucza publicznego wystarczy rozłożyć liczbę n na jej dwa pierwsze czynniki i rozkodowanie wiadomości jest proste (mając p, q i e możemy odtworzyć d). Bezpieczeństwo algorytmu bazuje wyłącznie na przekonaniu, że rozkład dużej liczby musi być czasochłonny. Wiara ta może się okazać nieuzasadniona, gdyby nastąpił przełom w teorii algorytmów rozkładu. Ocenia się, że od czasu wprowadzenia algorytmu RSA szybkość rozkładu (bez uwzględnienia przyrostu mocy obliczeniowej), wzrosła ok. 10 razy. Najlepszym dowodem, na niepewność co do bezpieczeństwa algorytmu może być to, że wysiłkiem ok. 1600 stacji roboczych pracujących wspólnie przez kilka miesięcy rozszyfrowano przykład przedstawiony przez twórców systemu. Nadal uważa się jednak, że generowanie dostatecznie długich kluczy jest odpowiedzią na wszelkie zagrożenia.

3.3 Podpisy cyfrowe

Opisany algorytm kodowania asymetrycznego może również służyć do **kontroli wiarygodności nadającego**. Obie strony, aby zostać uwiarygodnione muszą udowodnić posiadanie swojego tajnego klucza (*secret key*). Istotną cechą systemów kryptografii kluczy publicznych jest permutowalność (*permutability*), oznaczająca, że oba klucze mogą być stosowane zarówno do szyfrowania jak i deszyfrowania. Własność ta jest wykorzystywana w ramach identyfikacji i jest podstawą funkcjonowania podpisów cyfrowych (*digital signatures*).

Podpis cyfrowy powstaje w następujących krokach:

- Informacja przeznaczona do wysłania jest poddawana działaniu tzw. funkcji mieszającej (*hash-function*), która koduje i redukuje rozmiar informacji (RSA zaleca rozmiar do 64 bajtów), funkcja ta jest wykorzystywana do kontroli spójności informacji (*message integrity check*).
- Nadawca szyfruje przekształconą informację za pomocą swojego klucza prywatnego.
- Otrzymany ciąg znaków jest traktowany jako podpis cyfrowy i dołączany wraz ze wskazaniem zastosowanej funkcji mieszającej do komunikatu (który sam może, ale nie musi, być zaszyfrowaną wcześniej wiadomością).
- Odbiorca po otrzymaniu informacji opatrzonej podpisem cyfrowym poddaje część zawierającą jawne dane działaniu tej samej funkcji mieszającej, którą stosował nadawca.
- Odbiorca rozszyfrowuje podpis cyfrowy stosując klucz publiczny nadawcy.
- Informacja jest wiarygodna, jeżeli poddana transformacji część informacyjna jest taka sama jak rozszyfrowany podpis cyfrowy.

Należy zwrócić uwagę, że w ten sposób realizowane może być uwiarygodnienie autora komunikatu i faktu autentyczności treści wiadomości, to czy wiadomość sama jest zaszyfrowana, czy nie zależy wyłącznie od decyzji nadającego. W każdej sytuacji podpis cyfrowy gwarantuje spójność informacji oraz identyfikuje autora.

3.4 Certyfikaty

Jednym z podstawowych problemów stosowania szyfrów jest dystrybucja kluczy. Szyfrowanie oparte o klucze publiczne pozwala wprawdzie na publiczne ogłaszanie kluczy, ale problemem staje się zapewnienie prawdziwości informacji o tym, że dany klucz należy do danej osoby. Ponieważ klucze są jednocześnie stosowane jako metoda uwierzytelniania tożsamości, problem ich autentyczności staje się podwójnie ważny.

Rozwiązaniem jest stosowanie kluczy podpisanych cyfrowo przez zaufane osoby lub instytucje. Takie uwierzytelnione klucze nazywa się certyfikatami. Różne systemy mogą stosować własne metody wykorzystywania mechanizmu poświadczania kluczy, w następnej części podamy dwa takie przykłady, zasada pozostaje jednak niezmienna — podpis zaufanej osoby czy instytucji pozwala mieć pewność, że nikt niepowołany nie podsuwa nam falsyfikatu.

4 Zarządzanie kluczami publicznymi

W obecnej chwili funkcjonują dwa najważniejsze systemy uwierzytelniania i szyfrowania poczty elektronicznej — Pretty Good Privacy (PGP) oraz Privacy Enhanced Mail (PEM). Obydwa wykorzystują zbliżone mechanizmy zapewnienia ochrony danych, różnica polega na stosowanym przez nie symetrycznym algorytmie szyfrującym oraz, co ważniejsze, sposobie uwierzytelniania kluczy publicznych.

4.1 PGP

PGP jest systemem powstałym w typowo „internetowy” sposób — zbudowano system, który następnie w trakcie eksploatacji dociera się, jest zmieniany, aż wreszcie awansuje do roli standardu. System ten służy do bezpiecznego przesyłania poczty oraz do szyfrowania zbiorów na dyskach i zawiera cały zestaw narzędzi do generowania kluczy, wysyłania i czytania poczty, poświadczania kluczy, tworzenia własnej bazy kluczy itp. Nas interesuje wyłącznie przyjęta w PGP metoda zarządzania kluczami.

W opisywanym systemie każdy użytkownik pracuje w oparciu w własną bazę kluczy (*key-ring*). Baza taka zawiera klucze osób, z którymi wymieniana jest korespondencja. W sytuacji, kiedy konieczne jest rozszerzenie bazy robi się to albo w oparciu o klucz otrzymany bez żadnych potwierdzeń (wierząc, że przekłamanie w trakcie transmisji jest mało prawdopodobne), albo otrzymuje się certyfikat poświadczony przez osobę, której klucz już mamy i jednocześnie mamy zaufanie, że nie potwierdzi ona fałszywej informacji.

Zaletą PGP jest jego całkowita niezależność od struktur administracyjnych, wadą konieczność polegania na własnej ocenie prawdomówności innych osób oraz prawdopodobne kłopoty z otrzymaniem poświadczonych kluczy osób, których bezpośrednio nie znamy.

Aby ułatwić dystrybucję kluczy uruchomiono szereg serwerów kluczy PGP pozwalających na przeszukiwanie bazy danych i pobieranie kluczy, problemy z ich poświadczeniem nadal jednak pozostają.

Podsumowując, można ocenić, że o ile PGP jest systemem o dużym stopniu pewności i nadaje się do bezpiecznej komunikacji pomiedzy osobami, to nie można by go uznać za wystarczający w sytuacjach zwiazanych z obrotem finansowym pomiędzy nie znającymi się partnerami.

4.2 PEM

PEM, w przeciwieństwie do PGP wyrósł jako standard (RFC1421–4), na którym bazują konkretne implementacje. Sam standard opisuje mechanizmy związane z szyfrowaniem i podpisywaniem poczty, jak również bardzo dokładnie określa strukturę organizacyjną związaną z administrowaniem kluczami.

W zakresie administracyjnym PEM bazuje na zaleceniach standardu X.509, definiującego strukturę bezpieczeństwa na użytek dostępu do X.500 i innych aplikacji OSI. Directory.

Certyfikat w rozumieniu X.509, to klucz publiczny użytkownika poświadczony przez powołane specjalne urzędy wydające świadectwa (*Certification Authorities* — CA). **Urzędy certyfikacyjne** tworzą certyfikaty z upoważnienia i w imieniu użytkowników. Urząd certyfikacyjny musi być w stanie zapewnić wysoki stopień poufności działania i musi być absolutnie wiarygodny. Z tego powodu przewiduje się funkcjonowanie sieci urzędów certyfikacyjnych w ramach administracji państwowej (i z tego dokładnie powodu nie należy oczekiwać szybkiego wprowadzenia tego systemu w życie).

Certyfikat X.509 składa się z:

- nazwy urzędu wystawiającego świadectwo,
- nazwy użytkownika, do którego certyfikat należy,
- publicznego klucza użytkownika,
- identyfikatora algorytmu stosowanego do generowania podpisu oraz związanych z nim parametrów,
- czasu ważności certyfikatu,

- numeru seryjnego,
- wersji X.500 (obecnie zawsze 0).

Cała powyższa informacja jest potwierdzona podpisem cyfrowym urzędu certyfikacyjnego. Znając klucz publiczny urzędu poświadczającego, który jest odpowiedzialny za określonego użytkownika można dokonać weryfikacji poprawności certyfikatu klucza publicznego użytkownika.

Jeżeli kontaktujące się strony mają wspólnego wystawcę certyfikatu, to obie znają klucz publiczny poświadczającego, a więc może być realizowana wymiana informacji uwiarygodnionej podpisami. W przypadku, gdy mamy do czynienia z duża ilością partnerów w poufnym systemie nie wystarcza jedna organizacja uprawniona do tworzenia certyfikatów. Wówczas klucz publiczny jednego urzędu CA musi otrzymać certyfikat wydany przez inny urząd poświadczający. W ten sposób tworzone są skomplikowane struktury drzewiaste lub sieciowe zawierające opisy ścieżek (*certification path*), jakie musi przebyć certyfikat, aby został uwiarygodniony. Jeżeli urzędy certifikacyjne obu stron znajdują się w jednym drzewie, to możliwa jest weryfikacja podpisu użytkownika, poprzez kolejne weryfikowanie podpisów urzędów coraz wyższych szczebli, aż do osiągnięcia szczebla najwyższego. Podpis urzędu centralnego musi być powszechnie znany.

W szczególnych przypadkach, gdy np. wiadomo, że użytkownicy znajdujący sie pod opieką dwóch urzędów certyfikacyjnych często się komunikują można na podstawie dwustronnego uzgodnienia pomiędzy tymi urzędami ustalić certyfikat wzajemny (*cross certificate*), będący parą certyfikatów (np. klucz publiczny urzędu A poświadczony przez urząd B oraz klucz publiczny urzędu B poświadczony przez urząd A). Metoda ta musi być rownież stosowana w przypadkach komunikacji pomiędzy stronami nie posiadającymi urzędów certyfikujących formalnie umieszczonych w jednym drzewie.

Do poprawnego funkcjonowania mechanizmu potwierdzania tożsamości niezbędne jest utrzymywanie aktualnych **list unieważnionych certyfikatów** (*revocation lists*). Są to zestawy certyfikatów odwołanych, z różnych powodów, przed upływem ich terminu ważności. Gdyby nie brać pod uwagę możliwości kradzieży kluczy, to można określić pewny system komunikacji pomiędzy nie znającymi się partnerami, wystarczy, że jeden z nich przyśle drugiemu komplet certyfikatów (od swojego aż do szczytu drzewa). Niestety spójność takiej ścieżki nie daje gwarancji, że jej fragment nie został zmodyfikowany przez fałszerza będącego w posiadaniu skradzionego klucza. Pewność da dopiero skonfrontowanie wszystkich kolejnych certyfikatów z listami certyfikatów unieważnionych (oczywiście pod warunkiem, że fakt kradzieży klucza został uświadomiony).

5 Baza X.500 jako miejsce przechowywania kluczy

Oczywiste jest, że stosowanie na szeroką skalę kryptografii w przesyłaniu danych wymaga ogromnych, ogólnodostępnych baz zawierających klucze publiczne. X.500 jest bazą, która może sie do tego bardzo dobrze nadawać. Dostosowanie X.500 do systemu używanego przez PEM wynika z faktu, ze X.509 jest fragmentem standardu X.500; przechowywanie kluczy na użytek PGP jest również możliwe i aktualnie prowadzone są prace przygotowawcze mające doprowadzić do zdefiniowania odpowiednich atrybutów.

X.500 użyte do celów kryptografii publicznej spełnia równocześnie dwa zadania:

- jest systemem zapewniającym jednorodne nazewnictwo osób i obiektów, którym można przyporzadkowywać klucze identyfikacyjne,
- przechowuje certyfikaty w postaci odpowiednich atrybutów obiektów.

Istnienie ogólnodostępnej bazy certyfikatów umożliwia stosowanie automatyczych metod poświadczania tożsamości.

Przechowywanie certyfikatów w X.500 ma jeszcze jedną zaletę, umożliwia kontrolowany dostęp do zasobów samej bazy X.500. Temu zagadnieniu poświęcimy odrębną część niniejszego opracowania.

Stosowanie systemu PEM (i w ogóle metod poświadczania tożsamości) może wprowadzać znaczne opóźnienia systemowe w sytuacjach, kiedy trzeba kontrolować długie ścieżki poświadczeń ulokowane w odległych fragmentach drzewa informacji X.500. Aby temu zapobiec standard proponuje wprowadzanie optymalizacji (lokalne buforowanie informacji, stosowanie certyfikatów pośrednich). W ramach projektu PASSWORD opracowano system przeglądający drzewo informacji i zbierający z niego dane o urzędzach certyfikacyjnych. Zebrane dane służą następnie do utworzenia lokalnej struktury odwzorowującej system ścieżek poświadczeń. System taki zapewnia znaczne przyspieszenie poświadczeń, wprowadzajac niewielkie niebezpieczeństwo w postaci możliwości manipulowania lokaną informacją. Szczegóły systemu opisano w opracowaniu [7].

6 Ochrona danych w X.500 Directory

Część rekomendacji X.500 CCITT (obecnie ITU-T) zatytułowana *„X.509 — The Directory: Authentication Framework"*, definiuje mechanizmy, poprzez które realizowane są funkcje zapewniające ochronę danych w bezpiecznych systemach informacyjnych pracujących w środowisku OSI. Jest to poświadczanie tożsamości (w skrócie identyfikacja — *authentication*) oraz upoważnienie do dostępu do danych — (*authorization*, *access control*). Autorzy standardu w wersji 1988 nie zdążyli opracować pełnej definicji uprawnień dostępu do danych, dlatego dotychczas implementacje bazowały na nieoficjalnych uzgodnieniach. Edycja '93 zawiera kompletną definicję.

Trzeba na wstępie zaznaczyć, że standard X.500 nie przewiduje szyfrowania rezultatów operacji. Wydaje się to być bardzo silnym defektem, który musi ograniaczać dostęp do poufnych danych. Należy się spodziewać, że kolejna edycja standardu zostanie rozszerzona o ten element.

6.1 Zasady poświadczania tożsamości

Standard opisuje postać informacji wspomagającej identyfikację, umieszczonej w bazie X.500 oraz procedury stosowane przez serwery X.500 i użytkowników baz danych, mające na celu sprawdzenie tożsamości stron żądających kontaktu.

Zdefiniowany mechanizm jest na tyle ogólny, że może być stosowany nie tylko w odniesieniu do komponentów systemu X.500, ale również w przypadku konieczności identyfikacji w ramach zewnętrznych aplikacji typu FTAM (*File Transfer Access Method*), VT (*Virtual Terminal*) czy MHS (*Message Handling System*.

Standard wprowadza dwa poziomy sprawdzania tożsamości: prosty (*simple*) oraz silny (*strong*). Domyślnie przyjęta jest prosta forma poświadczania bazująca na przedstawianiu się użytkownika czy aplikacji w postaci (nazwa/hasło użytkownika). Atrybut `userPassword`, zdefiniowany jako łańcuch znaków o maksymalnej długości 128, jest przechowywany w bazie X.500. Jeżeli nie stosuje się lokalnych rozszerzeń dotyczących zasad konstrukcji drzewa informacji (*DIT schema*) atrybut zawierający hasło może być umieszczony w opisie obiektów typu: organizacja, jednostka organizacyjna, osoba w ramach organizacji — `organizationalPerson`, czy tzw. `residentialPerson`.

6.1.1 Procedura prostego poświadczania tożsamości

W przypadku stosowania prostej identyfikacji para (nazwa/hasło użytkownika) jest bezpośrednio przekazywana do serwera jako argument operacji dowiązania do X.500 (*Bind*). Serwer sprawdza zgodność otrzymanych danych z zawartością informacyjną bazy X.500 i akceptuje połączenie, gdy odebrane hasło jest prawidłowe. Jeżeli dane dotyczące obiektu żądającego dowiązania nie są dostępne lokalne inicjowana jest zdalna operacja *Compare*, która w celu rozstrzygnięcia poprawności podanego hasła przesyła je do kolejnych serwerów. Wynik tej operacji determinuje, czy proces zostanie dowiązany do serwisu X.500.

Stosując prostą metodę poświadczania powodujemy krążenie pomiędzy DUA i serwerami DSA hasła w otwartej, jawnej postaci. Dodatkowo, jeżeli informacja o obiekcie występującym o dostęp do Directory nie jest zapamiętana lokalnie, droga hasła jest nieokreślona, ponieważ trasa w bezpołączeniowym Internecie jest budowana dynamicznie. Jest to oczywiście bardzo niebezpieczne, hasło jest narażone na przechwyconie i następnie sprawca może przedstawić się jako „okradziony" użytkownik (*masquerade*).

Oczywiście wzmocnienie bezpieczeństawa można zapewnić przez szyfrowanie hasła dla celów transmisji. Ta forma nadal nie rozwiązuje wszystkich problemów, na które powinien być odporny bezpieczny system. Szyfrowane hasła nie dają zabezpieczenia przed działaniem polegającym na ponownym wykonaniu tej samej operacji w cudzym imieniu (*replay*). W celu zapobiegnięcia takiej sytuacji dodatkowo stosuje się dołączanie „stempli czasowych" lub losowej liczby, albo pary tych elementów. W ten sposób strona odbierająca powtórnie wysłane hasło, uzyskuje informację wskazującą, że informacja uległa przedawnieniu.

Tak zabezpieczone hasło wprowadzono w systemie X.500 pod nazwą **hasła chronionego** — *protected password*. Zgodnie z tą techniką stosuje się szyfrowanie hasła przed wysłaniem, poprzez użycie funkcji *one-way*, szyfrującej komunikat w sposób nieodwracalny. Otrzymany łańcuch jest przesyłany razem z nazwą obiektu oraz wskazaniem na zastosowaną do kodowania funkcję. Odbiorca po odnalezieniu w drzewie informacyjnym X.500 danych dotyczących strony zgłaszającej się koduje jej hasło i porównuje otrzymany ciąg znaków z odebranym zaszyfrowanym hasłem.

Opisane dotychczas procedury dotyczyły jednostronnego poświadczania tożsamości, tzn. identyfikowania strony wołającej. W przypadku komunikowania się dwóch serwerów na ogół obie strony powinny zostać uwiarygodnione, mamy wówczas do czynienia z **dwustronną identyfikacją**. Wołany serwer może jako wynik operacji dowiązania zwrócić swoje własne dane uwierzytelniające, które umożliwią wołającemu zatwierdzić tożsamość odbiorcy. Tego typu działania oznaczają przesyłanie haseł DSA w sieci, stanowią więc zagrożenie bezpieczeństwa systemu informacyjnego. Standard nie zabrania opcjonalnego stosowania poświadczeń dwustronnych pomiędzy serwerami, mogą one być indywidualnie negocjowane pomiędzy administratorami poszczególnych serwerów. Dotychczas X.500 w fazie pilotowej stosuje operację dowiązania bez poświadczania tożsamości DSA.

Rozproszone operacje w ramach X.500 często wymagają kontaktu z licznymi serwerami.

Każda operacja wymaga identyfikacji strony inicjującej, kolejne serwery nie uwiarygodniają rzeczywistego nadawcy zlecenia. Aplikacje o wzmocnionym zabezpieczeniu powinny wymagać od docelowego serwera kontroli tożsamości nadawcy zlecenia. W tym celu w standardzie '88 wprowadzono przesyłanie pomiędzy serwerami danych dotyczących strony inicjującej (czyli DUA) w ramach argumentów przekazywanych kolejnym serwerom operacji rozproszonej. W standardzie '93 rozszerzono ten system o dodanie do argumentów operacji poziomu identyfikacji użytkownika.

Oba rozwiązania bazują na konieczności zadeklarowania zaufania przez bieżące DSA serwerowi poprzedniemu i następnemu, tzn. przyjęcia, że serwery te nie wprowadzają do przenoszonej informacji zniekształceń. Bez tego założenia nie jest możliwe zaakceptowanie nazwy zgłaszającego się oraz jego hasła. Jeżeli nie jest stosowane silne poświadczanie pomiędzy serwerami, DSA musi lokalnie utrzymywać listy zaufanych serwerów i kontrolować każde nadchodzące zlecenie.

Jedynym niezawodnym mechanizmem, anulującym wszystkie wskazane niedociągnięcia prostej identyfikacji i dającym pełną poufność jest silne poświadczanie tożsamości.

6.1.2 Procedura silnego poświadczania tożsamości

Metody silnej identyfikacji bazują na opisanych wcześniej technikach kryptograficznych (certyfikatach).

Generowanie podpisów cyfrowych w aplikacjach OSI nie może dotyczyć informacji w takiej postaci, z jaką mamy do czynienia w warstwie aplikacji, ponieważ lokalna reprezentacja danych jest uzależniona od implementacji i od stosowanej platformy.

Protokół OSI definiuje uniwersalną składnię danych w warstwie aplikacji (*abstract syntax*), abstrahującą od specyfiki platformy, na której system pracuje. Służy do tego formalny język *Abstract Syntax Notation One (ASN.1)*. W celu umożliwienia transmisji takich struktur informacyjnych w sieci komputerowej powstała syntaktyka transferu (*transfer syntax*), realizowana przez tzw. podstawowe reguły kodujące — *Basic Encoding Rules (BER)*.

Jeżeli wymagana jest generacja podpisu, komunikat przedstawiony w ramach aplikacji w postaci zgodnej z ASN.1 przekazywany jest do warstwy prezentacji, gdzie podlega transformacji do odpowiedniego formatu według reguł BER. Tak przekształcona informacja poddawana jest działaniu funkcji mieszającej, szyfrowana właściwym kluczem (odbywa się to w warstwie aplikacji), a następnie ponownie kodowana przez BER i przekazywana do niższych warstw w celu dostarczenia do odbiorcy. Zastosowana tu technika wykorzystuje techniki przekazywania danych pomiędzy warstwami dla celu ujednorodnienia ich przed szyfrowaniem, jeżeli na operację tą patrzy się formalnie jako na komunikację warstw, to następuje tu zaburzenie modelu OSI, ponieważ raz przekazane niżej dane trafiają ponownie (bez zastosowania zwykłego przejścia między warstwami) do warstwy aplikacji. Ta metoda wzbudzała opory u części twórców standardu, ale została ostatecznie zaakceptowana (patrz [1] str. 241).

Przy realizacji silnego poświadczania dopuszcza sie trzy metody wymiany danych uwierzytelniających:

1. **Jednostronne poświadczanie** (*one-way authentication*)
   Polega na pojedynczym transferze informacji pomiędzy użytkownikiem A i B. A przesyła do B ścieżkę certyfikacji od B do A oraz stempel czasowy t^A, liczbę losową r^A i nazwę odbiorcy B. B na podstawie ścieżki certyfikacji uzyskuje klucz publiczny A (A_p), sprawdza wiarygodność na podstawie podpisu, kontroluje zgodność nazw odbiorcy komuni-

katu, aktualność stempla czasowego oraz opcjonalnie na podstawie r^A ocenia, czy nie wystąpiło powtórzenie tego samego komunikatu (*replay*).

2. **Dwustronne poświadczanie** (*two-way authentication*)
   Pierwsza faza komunikacji przebiega jak dla jednostronnego poświadczania, następnie B wysyła element identyfikacyjny w postaci: stempel czasowy t^B, liczba losowa r^B, nazwa odbiorcy A oraz otrzymanę liczbę r^A, a użytkownik A uwierzytelnia odebrane dane.

3. **Trójstronne poświadczenie** (*tree-way authentication*)
   W pierwszej fazie wykonywane są czynności jak w przypadku dwustronnego poświadczania, z tym że nie stosuje się stempli czasowych. Strona B zwraca w drugim kroku liczbę r^A oraz swoją r^B. Strona A po kontroli poprawności danych identyfikujących B sprawdza, czy odebrany liczbę r^A jest identyczny z wysłanym. Identyczność stempla upewnia A, że otrzymał odpowiedź z autentycznego źrodła, a nie przechwyconą wcześniej i obecnie powtórzoną odpowiedź. W trzecim kroku A zwraca B jego liczbę r^B na podstawie którego B weryfikuje podpis cyfrowy oraz upewnia się, czy nie jest to komunikat powtórzony. W obecnej chwili poświadczanie trójstrone nie jest przez X.500 wykorzystywane.

6.2 Uwierzytelnianie operacji rozproszonych systemu X.500

DUA zawsze wysyła zlecenie w postaci jawnej, dzięki czemu każde DSA potrafi bez weryfikowania podpisu pośredniczyć w realizacji zapytania. Identyfikacja odbywa się w serwerze udzielającym końcowej odpowiedzi. DSA takie musi znać klucz publiczny nadającego DUA, tylko wówczas może uwiarygodnić podpis cyfrowy. Serwer może uzyskać dostęp do kluczy publicznych na kilka sposobów:

- DSA wyszukuje dane dotyczące DUA w bazie X.500 i pobiera klucz,
- administrator DSA na podstawie dwustronnych uzgodnień utrzymuje zewnętrzną listę kluczy publicznych,
- użytkownik razem ze zleceniem wysyła klucz i ścieżkę certyfikacji.

Serwery mogą również pracować na zasadzie poświadczania wiarygodności każdego partnera (*point-to-point authentication*). Wówczas każde pośredniczące DSA tworzy swój podpis cyfrowy pod otrzymanym zleceniem oraz dodanymi przez siebie argumentami łańcuchowania.

Zleceniodawca może zlecić zabezpieczenie rezultatu operacji za pomocą podpisu cyfrowego poprzez przesłanie odpowiedniej opcji w argumentach (składowa `protectionRequest` w `CommonArguments`). Końcowe DSA w tej sytuacji może, ale nie musi, zastosować się do wymagań i podpisać wygenerowany rezultat. Nigdy nie poświadcza się wyniku operacji modyfikacji oraz rezultatów wskazujących na wystąpienie błędu. Aktualizacja kończy sie wysłaniem „pustego" rezultatu, natomiast błąd sygnalizowany jest przekazaniem zwrotnym pojedynczej liczby całkowitej. Obie te postaci nie mogą zostać zaszyfrowane kluczem prywatnym nadawcy, ponieważ narażałyby bezpieczeństwo tajnego klucza.

Dodatkowe problemy wnosi skomplikowana forma wyników operacji listowania (*List*) oraz wyszukiwania (*Search*). W typowej sytuacji realizacji zapytania końcowy wynik operacji rozproszonej jest przekazywany przez jedno, pośredniczące DSA. Takie DSA jest odpowiedzialne za połączenie rezultatów otrzymanych ze wszystkich serwerów. W sytuacji, gdy rezultaty nadchodzą podpisane przez generujące je DSA, nie ma możliwości, aby dokonać scalenia wyników operacji i nie utracić poświadczenia. Oznacza to, że końcowe DSA może wyłącznie połączyć niepodpisane rezultaty. Stąd zwracany wynik może mieć postać wielu porcji danych, poświadczonych przez odpowiednie serwery i „paczki" zawierającej rezultaty połączone przez końcowy serwer, opcjonalnie poświadczonej jego podpisem.

6.3 Zawartość bazy X.500 a silne poświadczanie tożsamości

Jak wspominaliśmy poprzednio baza X.500 wykorzystuje przechowywane w niej dane również dla celów poświadczania tożsamości. Zdefiniowane są atrybuty `userPassword` (słaba identyfikacja) oraz `userCertificate` lub `CACertificate` (silna identyfikacja).

Znając klucz publiczny urzędu poświadczającego, który jest odpowiedzialny za określonego użytkownika można dokonać weryfikacji poprawności certyfikatu klucza publicznego użytkownika.

Komunikacja pomiędzy serwerami DSA i aplikacjami DUA nie nastręcza problemów w przypadku, gdy kontaktują się elementy mające wspólnego wystawcę certyfikatu. Jeżeli mamy do czynienia z łącznością pomiędzy jednostkami podlegającymi różnym, umiejscowionym w dowolnej pozycji drzewa informacji urzędom poświadczającym, konieczne jest wykorzystywanie opisanych na stronie 8 ścieżek certyfikatów, czy certyfikatów wzajemnych. Operacje według protokołu DAP (*Directory Access Protocol*, odpowiedzialny za wymianę informacji pomiędzy DUA i DSA) oraz DSP (*Directory System Protocol*, realizujący połączenia pomiędzy serwerami DSA) wykorzystują ścieżki certyfikacji zarówno w zleceniach jak i w rezultatach operacji.

Komunikaty zawierają dodatkowo wszystkie informacje niezbędne odbiorcy do weryfikacji poprawności informacji. W rzeczywistości ścieżki uwierzytelniania danych mogą być dowolnie długie. Pierwszy certyfikat jest poświadczony przez urząd certyfikacyjny odbiorcy, następne zawierają klucz publiczny urzędu certyfikacyjnego kolejnego w ścieżce, ostatni certyfikat zawiera klucz publiczny nadawcy.

Baza X.500 moża zawierać również artybuty (`crossCertificatePair`), będące parą certyfikatów (np. klucz publiczny urzędu A poświadczony przez urząd B oraz klucz publiczny urzędu B poświadczony przez urząd A), takie wzajemne cerytfikaty skracają (jak już pisaliśmy) scieżki certyfikacji pomiędzy często komunikującymi sie obiektami.

Warto zauważyć, że jeżeli mamy do czynienia ze ścieżkami certyfikacji użytkownik, bądź jego DUA musi znać dane dotyczące zdalnego docelowego DSA. Silnie poświadczane zlecenie nie może nadejść z nieznanego źródła. Ten typ identyfikacji wymaga administracyjnego ustalenia sposobu certyfikacji, tak by mogła zostać zbudowana ścieżka poświadczania. Niezbędne wydaje się opracowanie metod automatycznego budowania ścieżek certyfikacji na podstawie hierarchicznej struktury urzędów poświadczających, standard dopuszcze dowolne ich rozmieszczenie w drzewie informacji X.500 Directory.

Wspomniane uprzednio listy unieważnionych certyfikatów mogą być przechowywane w X.500 poprzez atrybuty `certificateRevocationList`. Taka obsługa list daje łatwy dostęp do informacji, która ma podstawowe znaczenie w systemie potwierdzeń.

6.4 Kontrola dostępu do danych

Jak wspomnieliśmy, standard '88 nie sprecyzował systemu zabezpieczeń danych w X.500 Directory. Z tego powodu implementatorzy zastosowali swoje własne rozszerzenia. Oprogramowanie QUIPU stało sie „de facto” standardem, ponieważ na nim oparte jest ok. 95% pracujących na świecie systemów X.500. Wydaje się zatem uzasadnione opisanie mechnizmów zaimplementowanych w QUIPU, a następnie porównanie ich z tymi określonymi przez standard X.500 ’93.

6.4.1 Kontrola dostępu w QUIPU

Każde hasło umieszczane w bazie X.500 posiada atrybut *access control list* (ACL) lub *extended access control list* (XACL). Lista kontroli dostępu jest podzielona na trzy części:

1. *Child ACL.* Dotyczy kontroli dostępu do haseł umieszczanych w drzewie informacji DIT bezpośrednio pod bieżącym hasłem.
2. *Entry ACL.* Kontroluje dostęp do informacji umieszczonych w bieżącym haśle.
3. *Attribute ACL.* Każdy atrybut bieżącego hasła może posiadać własną listę kontroli dostępu.

Każdy element listy składa się z selektora dostępu oraz poziomu dostępu. Selektor określa zakres uprawnień:

- *User* — dotyczy wyłącznie bieżącego obiektu.
- *Other* — ogólne prawo dostępu.
- *Prefix* <*name*> — dotyczy obiektu o podanej nazwie *name* i wszystkiego poniżej w drzewie informacji.
- *Group* <*name*> — przyznaje uprawnienia obiektom należącym do grupy o podanej nazwie *name*, w praktyce przydziela się w ten sposób prawa pojedynczej osobie (administratorowi) do modyfikowania określonych pozycji w bazie.
- *Attribute* <*dnattr*> — tylko w przypadku XACL; daje uprawnienia obiektom wymienionym w atrybucie o syntaksie DN (*Distinguished Name*), np. `owner`, `member` hasła.

Poziom dostępu może być określony następująco:

- *None* — zabroniony dostęp.
- *Compare* — filtr porównania stosowany do obiektu, typowo używany do atrybutu `userPassword`.
- *Read* — odczyt zawarości informacyjnej obiektu.
- *Add* — informacja może być dodana do obiektu, ale nie usunięta.
- *Write* — zawartość informacyjna obiektu może być modyfikowana w dowolny sposób.
- *Remove* — tylko w przypadku XACL; obiekt może być usuwany lub modyfikowany w dowolny sposób.

Atrybut ACL decyduje o sposobie wykonywania operacji w ramach X.500. Np., aby można było pomyślnie zrealizować operację wylistowania (*List*), *Child ACL*, czyli lista ACL dotycząca położonych poniżej bieżącego haseł musi co najmniej dawać dostęp typu *read.* Dodatkowo, każde poniższe hasło będzie listowane tylko wówczas, gdy jego własna lista ACL *EntryACL* będzie uprawniała do dostępu typu *read.*

6.4.2 Kontrola dostępu w standardzie '93

Nowa wersja standardu daje bardzo skomplikowany mechanizm ochrony danych, różniący się istotnie od stosowanego przez QUIPU. Mechanizm ten jest na tyle skomplikowany, że sam standard przewiduje jego uproszczoną wersję (przypuszczalnie początkowo ona właśnie będzie implementowana). Szczegółowy opis tych mechanizmów zajmuje w książce [1] 40 stron, a opiera się na obszernym materiale wprowadzonym wcześniej. Z konieczności potraktujemy zatem ten temat pobieżnie przedstawiając jedynie jego główne założenia.

Pierwszą poważną różnicą w stosunku do QUIPU jest to, że standard '93 domyślnie przyjmuje blokowanie wszelkiego dostępu, podczas gdy QUIPU przyznaje powszechny dostęp. Jest to spowodowane tym, że QUIPU napisano z myślą o pilotowym, szeroko dostępnym systemie, podczas gdy standard '93 musi na pierwszym miejscu stawiać ochronę danych.

W standardzie na użytek kontroli dostępu wprowadzono możliwość stosowania dodatkowych identyfikatorów obiektów. Zadaniem ich jest rozróżnianie kolejnych wystąpień obiektu o tej samej nazwie (`distinguishedName`). Dokładniej chodzi o sytuacje, kiedy likwiduje się jakiś obiekt (np. osoba A przestaje pracować w danej instytucji), a następnie tworzy się go ponownie (zatrudniono osobę B o identycznym imieniu i nazwisku jak A). W takiej wypadku mogłoby się zdarzyć, ze osoba B uzyskałaby dostęp do danych poprzednio dostępnych osobie A (jeżeli przez niedopatrzenie zapomniano gdzieś w drzewie informacji skasować odpowiednie przywileje). Zastosowanie dodatkowych identyfikatorów rozróżniających kolejne wystąpienia tej samej nazwy zagwarantuje, że takie zagrożenie nie wystąpi. Stosowanie identyfikatorów jest opcjonalne.

W stosunku do przedstawionego mechanizmu zaimplementowanego w QUIPU, wprowadzono ochronę obszarów danych, a nie pojedynczych haseł. Rozbudowano też listę poziomów dostępu o *detect*, *search*, *modify_DN* oraz *import*, wprowadzono poziom identyfikacji wymagany do uzyskania odpowiednich praw dostępu, istnieje możliwość jawnego blokowania dostępu pewnym grupom użytkowników.

Ustalenie zaostrzonych wymagań w stosunku do określonego hasła czy poddrzewa informacji może implikować pewne anomalie w dostępie do tych danych przez innych użytkowników. Na przykład, jeżeli użytkownik A ma prawo dostępu do obiektu E po potwierdzeniu tożsamości na słabym poziomie, ale użytkownik B ma zabroniony dostęp do tego samego obiektu i wyspecyfikowany został silny stopień identyfikacji, to użytkownikowi A nie wystarczy słaby poziom identyfikacji, aby uzyskać dostęp do E, musi on bowiem **udowodnić**, że nie jest użytkownikiem B, z wykorzystaniem silnej identyfikacji, a więc musi dokonać w tym przypadku własnej identyfikacji na tym poziomie.

Wprowadzono pojęcie obszaru ochrony danych (Access Control Area — ACA). Pojęcie to jest następnie poddawane jeszcze drobniejszemu podziałowi, ale nie będziemy w to wnikać. ACA może być określone jako podobszar w drzewie zaczynający się od określonego punktu i rozciągający w dół do granic administrowanego obszaru, być może z pominięciem pewnych poddrzew. Można dodatkowo określić, że ACA zawiera wyłącznie hasła o zadanym atrybucie `objectClass`, a zatem możemy na przykład zdefiniować w ACA wszystkie osoby w ramach uniwersytetu, z wyłączeniem Wydziału Matematyki.

Następnym określeniem są listy chronionych obiektów, typowo zawierające listy typów atrybutów, istnieje możliwość chronienia samych wartości atrybutów, tzn. np. można udostępnić informacje o istnieniu atrybutu, ale zabronić jakichkolwiek akcji na jego wartości, można też chronić konkretne wartości atrybutów.

Kolejne pojęcie, to lista użytkowników, do której stosuje się prawa dostępu, podobnie jak w QUIPU wyróżnia się właściciela, grupę, poddrzewo, wszystkich i, dodatkowo, pojedynczego użytkownika (np. administratora). Zasadnicza, w stosunku do QUIPU różnica pojawia się w pojęciu poddrzewa, które specyfikuje się nie tylko za pomocą wierzchołka, ale również przy pomocy dodatkowych operacji odcinania fragmentów drzewa (*chop*).

Wprowadzono priorytety nadawanych uprawnień. Priorytety te mogą być jawne (określane numerycznie) oraz domyślne (prawa określone dla mniejszego ACA mają wyższy priorytet niż prawa określonego dla ACA szerszego). Jeżeli z dwóch praw dostępu wynika zablokowanie dostępu i jego udzielenie z takimi samymi priorytetami, to zablokowanie przeważa nad przydzieleniem.

Jawne priorytety mają za zadanie umożliwienie częściowego delegowania kontroli nad prawami dostępu. Główny administrator poddrzewa może zastrzec (za pomocą mechanizmów spoza standardu X.500), aby administratorzy poddomen stosowali priorytety nie wyższe niż pewien limit. W takiej sytuacji wprowadzenie nowego ograniczenia o wyższym limicie może unieważnić dowolne upoważnienia przyznane w poddomenach.

Prawa dostępu zapisywane są w postaci struktur `ACIItem` - *Access Control Informatiom Item*, które w symbolice ASN.1 przestawione są jako:

```
ACIItem     ::=              SEQUENCE {
    identificationTag        DirectoryString,
    precedence               Precedence,
    authenticationLevel      AuthenticationLevel.
    itemOrUserFirst          CHOICE {
         itemFirst               SEQUENCE {
             protectedItems           ProtectedItems,
             itemPermissions          SET OF ItemPermission },
         userFirst               SEQUENCE {
             userClasses         UserClasses,
             userPermissions     SET OF UserPermission }}}
```

Struktury `ACIItem` przechowywane są w specjalnych hasłach (`subentries`) określających poddrzewa, do których się odnoszą.

Aby zilustrować omówione mechanizmy, spróbujmy rozważyć przykład, kiedy na uniwersytecie udostępniamy lokalnie telefony służbowe pracowników, za wyjątkiem telefonów określonej grupy pracowników (np. dyrektorów instytutów, dziekanów itp.) Telefony tej wydzielonej grupy udostępnimy pracownikom administracji centralnej oraz członkom samej grupy. Dodatkowo, telefony dyrektorów będą udostępniane pracownikom podległych im jednostek.

W tej sytuacji:

1. W `subentry` drzewa uniwersytetu wprowadzamy `ACIItem` dający prawo odczytu wartości atrybutu `telephoneNumber` wszystkim zarejestrowanym w poddrzewie uniwersytetu. Wprowadzamy priorytet 50.

2. Wprowadzamy lokalną klasę (`objectClass`) o nazwie `dyrektor`.

3. Definiujemy grupę użytkowników (jako hasło w poddrzewie uniwersytetu) i wpisujemy do niego wszystkich „dyrektorów”.

4. Wprowadzamy definicję poddrzewa zawierającą wszystkie wystąpienia haseł klasy `dyrektor` w uniwersytecie i do `subentry` wpisujemy `ACIItem` nadający prawo odczytu atrybutu `telephoneNumber` członkom grupy dyrektorów (priorytet 100), dopisujemy drugi `ACIItem` nadający prawa odczytu osobom ulokowanym w poddrzewie administracji centralnej, oczywiście widać, że ważne jest jak skonstruowane jest samo drzewo, przy niewłaściwej strukturze niemożliwa może być definicja poddrzewa zawierającego dokładnie uprawnione osoby (priorytet 100). Wreszcie wpisujemy kolejny `ACIItem` zamykający dostęp do `telephoneNumber` wszystkim (priorytet 50).

5. W każym poddrzewie podjednostki uniwersytetu (wydziału, instytutu) wpowadzamy prawo odczytu `telephoneNumber` wszystkim zarejestrowanym w poddrzewie tej jednostki (priorytet 50). Powinniśmy jednocześnie zakazać dostępu do numeru kierownika jednostki, osobom spoza jednostki (priorytet 50).

Zauważmy, że grupy, którym nadano prawa z priorytetem 100 z pewnością mają dostęp do informacji, bo jawny priorytet jest najważniejszy. Wszyscy na terenie uniwersytetu mają dostęp do informacji o osobach spoza grupy „dyrektorzy". Pracownicy podjednostki mają dostęp do telefonu kierownika tej jednostki, wprawdzie mamy zakaz i udostępnienie z priorytetem 50, ale ponieważ dostęp jest udzielony węższej grupie niż zakaz, dostęp uzyskuje przewage. Zwroćmy jednocześnie uwagę, ze wszyscy pracownicy jednego wydziału mają dostęp do telefonu swojego dziekana, ale pracownicy instytutu A tego wydziału nie mają dostępu do telefonu dyrektora instytutu B tego samego wydziału, bo w instytucie B odebrano dostęp do tego konkretnego telefonu wszystkim spoza tego Instytutu.

Opisany przykład miał na celu zilustrowanie ile elementów trzeba brać pod uwagę przy określaniu prawa dostępu. Standard definiuje Funkcję Decyzji (*Access Control Decission Function*), która generuje decyzję o przydzieleniu dostępu. Funkcja ta pracuje na zbiorze piątek postaci:

```
(userClasses, autenticationLevel, protectedItems, grantsAndDenials,
 precedence)
```

W opisanym przykładzie zablokowanie prawa dostępu do atrybutu pojedynczej osoby (dyrektora w ostatnim kroku) można zrealizować na dwa sposoby, albo poprzez wyspecyfikowanie poddrzewa zawierającego tylko jego, albo poprzez wprowadzenie do samego hasła odpowiedniej ochrony. Ten pierwszy sposób mieści się w ramach uproszczonych reguł dostępu, drugi nie.

Przeprowadzone porównanie metod kontroli dostępu w QUIPU i standardzie '93 pozwala stwierdzić, że podobieństwa są jedynie powuerzchowne. Fakt ten będzie z pewnością utrudniał wprowadzenie nowego standardu w życie.

7 Pakiet SecuDE jako implementacja technologii kontroli bezpieczeństwa systemów informacyjnych

Oprogramowane o nazwie SecuDE (Security Development Environment) jest rozwijane przez niemiecką organizację GMD (Gesellschaft für Mathematik und Datenverbereitung) jako zestaw narzędzi programowych wspomagających tworzenie poufnych systemów. Początkowo sponsorem projectu *SecuDE* była narodowa organizacja sieci komputerowych — DFN (Deutsches Forschungsnetz).

Dystrybucja SecuDE zawiera pakiet kryptograficzny DES oraz oprogramowanie implementujące algorytmy kodujące MD2, MD4, MD5, pochodzące z RSA Data Security Inc.

Obecnie, wersja SecuDE opiera się na technologiach pochodzących z pakietu ISODE (IC R1.1v3), m.in. w celu implementacji funkcji translacji formatu ASN.1.

Oprogramowanie SecuDE może zosać zainstalowane na systemach SunOS 4.1.x, Solaris 2.x, HP-UX, OSF1, IRIX, LinuX, NetBSD, SCO Unix jak również w systemie MS-DOS, a w trakcie opracowywania są wersje na IBM OS/2 i Microsoft Windows NT/95.

Na kwiecień 1996 zapowiadana jest edycja SecuDE-5.0, która będzie dystrybuowana w dwóch wersjach: jako „public domain" oraz komercyjnie. Pakiet komercyjny będzie zawierał dodatkowo podręczniki instalacyjne, szczegółową dokumentację, GSS-API (Generic Security Service Application Program Interface), biblioteki DLL oraz nowe interfejsy użytkowe.

7.1 Opis funkcjonalny pakietu

Pakiet SecuDE oferuje zestaw programów, interfejsów typu API oraz funkcji bibliotecznych, przeznaczonych do tworzenia aplikacji ukierunkowanych na zapewnienie bezpieczeństwa komunikacji. W jego skład wchodzą następujące funkcje biblioteczne oraz programy użytkowe realizujące techniki bezpiecznego systemu:

- podstawowe funkcje kryptograficzne implementujące algorytmy RSA, DSA, DES, funkcje mieszające itp.,
- funkcje zabezpieczające realizujące identyfikację użytkowników, spójność danych, poufność przesyłanych komunikatów na podstawie podpisów cyfrowych oraz kryptografii symetrycznej i asymetrycznej,
- funkcje poświadczania (certyfikacji), zaimplementowane według standardu X.509, obsługujące ścieżki certyfikacji, certyfikaty wzajemne oraz listy unieważnionych certyfikatów,
- programy i funkcje zarządzające urzędami certyfikacyjnymi oraz zapewniające współpracę między tymi urzędami a użytkownikiem, którego tożsamość jest poświadczana,
- programy służące do tworzenia podpisów cyfrowych, weryfikacji, kodowania i dekodowania,
- implementację systemu PEM według RFC1421–4,
- metody przetwarzania list odwołanych certyfikatów według definicji z RFC1422,
- obsługa bezpiecznego dostępu do danych w ramach X.500 Directory, gromadzenie w ramach bazy X.500 informacji związanych z bezpieczeństwem, realizacja silnie poświadczanych operacji pomiędzy interfejsem użytkowym i serwerem oraz pomiędzy serwerami X.500 (w oparciu o pakiet IC R1.1v3),

- zarządzanie danymi dotyczącymi użytkowników w prywatnym otoczeniu PSE (*Private Security Environment*).

SecuDE dostarcza dwie możliwości realizacji środowiska PSE:

- poprzez specjalne karty (*smartcards*), będące formą sprzętowej realizacji zapewnienia bezpieczeństwa tajnych danych (SC-PSE),

- w ramach dyskowej bazy danych, umieszczonej w zaszyfrowanej postaci (według algorytmu DES) w strukturze katalogowej sytemu Unix (*software*-PSE, SW-PSE).

Obie metody implementują dostęp do środowiska PSE za pomosą specjalnego identyfikatora zwanego PIN (*Personal Identification Number*).

7.2 Realizacja bezpiecznej poczty elektronicznej PEM w pakiecie SecuDE

SecuDE dostarcza filtry transformujące dowolne zbiory wejściowe do formatu PEM i odwrotnie. Narzędzia te mogą zostać zintegrowane z interfejsami poczty elektronicznej (np. *MH*, *elm*), czy programami administrującymi urzędami certyfikacyjnymi CA.

SecuDE realizuje wszystkie formaty zdefiniowane w RFC1421–4, ale implementuje wyłącznie zarządzanie kluczami według kryptografii asymetrycznej. Udostępnia określone w RFC1424 metody certyfikacji oraz administrowania listami odwołanych certyfikatów CRL. Dodatkowo, jest to rozszerzenie w stosunku do definicji standardu, SecuDE-PEM może zostać tak skonfigurowany, aby współpracował z X.500-DUA m.in. w celu automatycznego wyszukiwania certyfikatów, czy list odwołań w bazie danych.

7.3 Integracja SecuDE z X.500

Jedną z istotniejszych cech pakietu jest możliwość wykorzystywania go w połączeniu z X.500 Directory.

Pakiet SecuDE dopuszcza konfigurację na jeden z trzech możliwych sposobów:

- Bez możliwości współpracy z X.500.

- Jako zintegrowany z X.500-DUA, przy czym dostęp do X.500 następuje poprzez proste poświadczanie oraz niechronione operacje typu DAP; w tej sytuacji musimy w trakcie kompilacji SecuDE wskazać miejsce, w którym wcześniej zainstalowano biblioteki ISODE.

- Jako zintegrowany z X.500-DUA z dostępem na zasadzie silnego poświadczania oraz możliwością poświadczania (*sign*) operacji DAP; SecuDE używa w tym celu zmodyfikowanej wersji oprogramowania QUIPU, dystrybuowanego przez GMD.

  Wersja QUIPU zawierająca rozserzenia związane z bezpieczeństwem systemu umożliwia również instalację tzw. bezpiecznych serwerów DSA.

7.4 Wykaz narzędzi programowych pakietu SecuDE

7.4.1 Programy użytkowe

Są to programy służące do tworzenia i weryfikacji podpisów cyfrowych, szyfrowania, deszyfrowania informacji, realizowania funkcji mieszającej na danych, dokonywania transformacji zgodnej z protokołem PEM. Wszystkie te narzędzia wymagają istnienia prywatnego otoczenia systemu bezpieczeństwa, zwanego PSE, traktowanego jako baza danych gromadząca niezbędne dane bezpiecznego systemu.

Specjalny program o nazwie `psecreate` służy do tworzenia PSE, `psemaint` pozwala natomiast wszechstronnie zarządzać PSE.

7.4.2 Programy do obsługi urzędów certyfikacyjnych

SecuDE udostępnia dwie metody wymiany informacji między użytkownikami SecuDE a zewnętrznym urzędem certyfikacyjnym:

- przez przesłanie zbiorów zakodowanych zgodniem z ASN.1,
- za pomocą specjalnych procedur zgodnych z protokołem PEM.

Dostarczone są programy do generacji kluczy oraz certyfikatów, a także narzędzia tworzące hierarchiczną strukturę urzędów poświadczających.

7.4.3 Dostęp do X.500 Directory

SecuDE umożliwia wyszukanie oraz aktualizację w zasobach bazy X.500 atrybutów związanych z bezpieczeństwem, takich jak:
`userCertificate`, `CaCertificate`, `crossCertificatePair`, `CertificateRevocationList`.

Funkcje te są zrealizowane na bazie oprogramowania QUIPU w wersji IC R1.1v3 i wymagają instalacji tej edycji pakietu. Zostały one również zintegrowane w ramach innych programów użytkowych SecuDE (np. w postaci opcjonalnego korzystania z danych w X.500).

7.5 Funkcje biblioteczne pakietu SecuDE

7.5.1 Kryptografia i PSE

Moduł *Secure* zawiera podstawowe funkcje kryptograficzne do generowania i weryfikacji podpisów cyfrowych, szyfrowania i deszyfrowania danych oraz funkcje dostępu do środowiska PSE. Implementuje kryptografię RSA, DES, algorytmy SqmodN, MD2, MD4, MD5, SHA oraz mechanizm podpisów cyfrowych według PKCS. Funkcje kryptograficzne są ściśle związane ze środowiskiem PSE w oprogramowaniu SecuDE. Stosowane klucze są adresowane jako obiekty PSE.

7.5.2 Zasady identyfikacji i certyfikacji

Moduł *AF* to zestaw funkcji realizujących zasady poświadczania i certyfikacji określone w standardzie X.509. Dotyczą one obsługi zarówno certyfikatów gromadzonych lokalnie, jak i certyfikatów z bazy X.500. Moduły *AF* i *Secure* dostarczają razem mechanizmy do tworzenia podpisów, weryfikacji, szyfrowania, deszyfrowania za pomocą kluczy z PSE lub X.500 oraz narzędzia zarządzania obiektami w ramach obu baz, a także sprawdzania prawdziwości i konsystencji PSE.

7.5.3 Standard poczty elektronicznej PEM

Moduł *PEM* implementuje funkcje zdefiniowane w RFC 1421–4. Standard PEM określa zasady szyfrowania i identyfikacji w celu realizacji bezpiecznej poczty elektronicznej bez konieczności interakcji z systemem transportu komunikatów (*Message Transfer System* — MTS).

7.5.4 Funkcje realizujące konwersję według ASN.1

Obiekty informacyjne przeznaczone do wykorzystywania nielokalnego mogą być przekształcane do postaci uniwersalnej za pomocą funkcji implementujących standard ASN.1 Odwrotne funkcje umożliwiają uzyskanie na podstawie odebranych danych uniwersalnych informacji czytelnej lokalnie.

Bibliografia

[1] David Chadwick (1994) *Understanding X.500 The Directory*

[2] Simson Garfinkel (1995) *PGP — Pretty Good Privacy*

[3] J. Linn (February 1993) *Privacy Enhancement for Internet Electronic Mail: Part I: Message Encryption and Authentication Procedure*, Request for Comments RFC1421

[4] S. Kent (February 1993) *Privacy Enhancement for Internet Electronic Mail: Part II: Certificate-Based Key Management* Request for Comments RFC1422

[5] D. Balenson (February 1993) *Privacy Enhancement for Internet Electronic Mail: Part III: Algorithms, Modes, and Identifiers* Request for Comments RFC1423

[6] B. Kaliski (February 1993) *Privacy Enhancement for Internet Electronic Mail: Part IV: Key Certification and Related Services* Request for Comments RFC1424

[7] B. Jerman-Blažič, D. Trček, T. Klobučar, F. Bračun, *A Tool Support of Key Distribution and Validy Certyficate Check in Global Directory Service*, Ljubljana, Slovenia

[8] W. Schneider, (March 1995) *SecuDE, Overview, Vol. 1: Principles of Security Operations, Vol. 2: Security Commands, Functions and Interfaces, Vol. 3: Security Applications' Guide*, GMD

[9] ITU-T Recommendation X.500–X.521 (93) *Information Technology — Open Systems Interconnection — The Directory*