Systemy operacyjne
Bezpieczeństwo

[2] Zagrożenia

Podstawowe cele bezpieczeństwa:

- **poufność** (ang. *confidentiality*),
- **integralność**, (nienaruszalność) (ang. *integrity*),
- **dostępność**.

Poufność – prawo jednostki do decydowania o tym, jakimi informacjami chce się podzielić z innymi i jakie informacje jest skłonna od nich przyjąć.

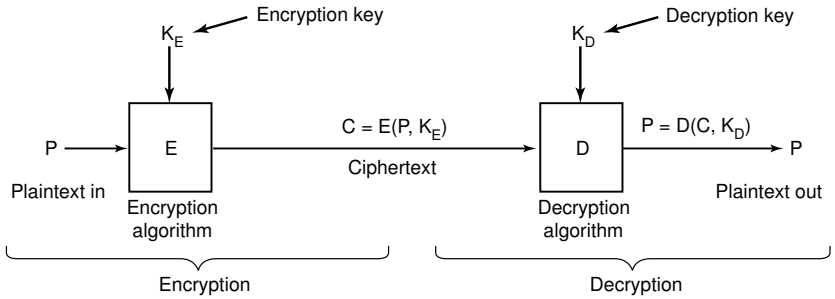
Integralność – (nienaruszalność) danych – cecha określająca, że dane nie różnią się od danych źródłowych i nie zostały przypadkowo lub umyślnie zmienione, ujawnione lub zniszczone.

[3] Podstawowe definicje

Uwierzytelnienie – proces potwierdzenia tożsamości użytkownika, rozumianego jako osoba, aplikacja bądź inny zasób komputerowy.

Autoryzacja – (uprawnienie) określa, jakich atrybutów których zasobów może używać uwierzytelniony użytkownik.

[4] Podstawy kryptografii



Zależności między tekstem jawnym a tekstem zaszyfrowanym.

- pojęcia: **test jawny**, **tekst zaszyfrowany**, **klucze**, **funkcje szyfrująca i deszyfrująca**,
- *bezpieczeństwo poprzez niedookreślanie* (ang. *security through obscurity*) - wiara, iż poziom bezpieczeństwa wzrasta poprzez utajnienie mechanizmów zabezpieczających.

[5] Szyfrowanie z kluczem tajnym

- szyfrowanie z zastosowaniem uzgodnionych funkcji do szyfrowania tekstu jawnego i klucza tajnego,
- odszyfrowanie z wykorzystaniem funkcji odwrotnej i tego samego klucza tajnego,
- przykład: standard **DES** opracowany przez IBM
  - 64-bitowy tekst jawny w 64-bitowy tekst zaszyfrowany z wykorzystaniem 56-bitowego klucza,
  - 16 zależnych od klucza etapów, okrążeń z przesunięciami bitowymi zależnymi od klucza i trzema niezależnymi od klucza transpozycjami,
  - możliwość realizacji sprzętowej,
  - 3DES metodą połączenia trzykrotnego użycia DES z dwoma kluczami, co wydłuża klucz do 112 bitów.
- główny problem: wstępna dystrybucja klucza tajnego.

[6] Szyfrowanie z kluczem jawnym

- **funkcja jednokierunkowa** - dowolna funkcja $f(X) = y$ taka, że przy danej wartości Y określenie wartości X jest bardzo trudne (brak oczywistej funkcji odwrotnej).
- każdy potencjalny odbiorca B, tworzy parę kluczy K_e i K_d oraz przechowuje w tajemnicy klucz odszyfrowania K_d, klucz szyfrowania K_e jest jawny i publicznie dostępny,
- dwie znane funkcje E (szyfrowanie) i D (deszyfrowywanie),

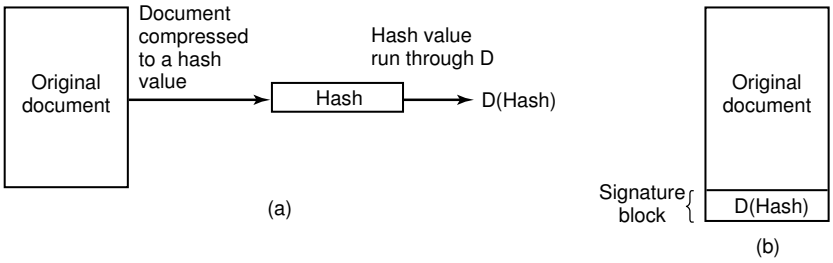
- szyfrowanie informacji przez A dla B: wykorzystanie funkcji $E(K_e, M)$ do wytworzenia komunikatu $\{M\}_{K_e}$, tylko B zna swój klucz K_d i może użyć $D(K_d, \{M\}_{K_e})$ w celu odszyfrowania,
- algorytm RSA (Rivest, Shamir, Adelman) do wytwarzania i stosowania par kluczy oparty na trudności znajdowania czynników pierwszych dużych liczb pierwszych.

[7] **Porównanie kryptografii z kluczem tajnym i jawnym**

- bezpieczeństwo - obie metody mogą zapewnić wystarczający poziom bezpieczeństwa,
- wygoda - szyfrowanie z kluczem jawnym nie wymaga tajnego kanału do rozprowadzania kluczy,
- wydajność - algorytmy szyfrowania z kluczem tajnym są znacznie szybsze.

Algorytm	Działanie	Programowo (bit/sek)	Sprzętowo (bit/sek)
RSA (klucz jawny)	szyfrowanie	$0.5 * 10^3$	$220 * 10^3$
	deszyfrowanie	$32 * 10^3$	b.d.
DES (klucz tajny)	szyfrowanie i deszyfrowanie	$400 * 10^3$	$1.2 * 10^9$

[8] **Podpis cyfrowy**

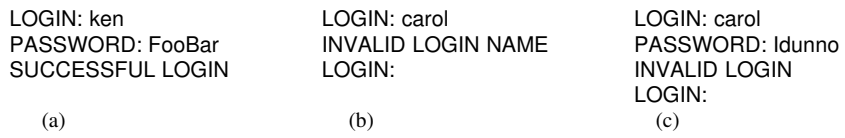


- **Podpis cyfrowy** - odpowiednik umieszczenia własnoręcznego podpisu na dokumencie - skrót dokumentu szyfrowany własnym kluczem prywatnym,
- wykorzystanie skrótu dokumentu bądź danych poprzez funkcje mieszające oraz szyfrowanie z kluczem publicznym,
- umożliwia dowiedzenie własnej tożsamości, weryfikację integralności danych oraz uniemożliwia wyparcie się podpisu,

[9] **Certyfikat cyfrowy**

- **Certyfikat cyfrowy** - podpisany cyfrowo zbiór danych binarnych, który zawiera zestaw kluczy publicznych, pewne atrybuty i wartości oraz datę ważności,
- funkcjonalność odpowiadająca funkcjonalności dokumentów typu prawo jazdy czy paszport. Certyfikat ma za zadnie poświadczyć tożsamość przed kimś, z kim wcześniej nie uzgadniano wymiany informacji.
- certyfikat wystawiany/podpisywany przez zaufaną stronę trzecią,
- ścieżki zaufania i łańcuch certfikatów,
- w sieci Internet dwa standardy: SPKI (ang. *Simple Public Key Infrastructure*) oraz PKIX (ang. *Public-Key Infrastructure X.509*),
- problem unieważnienia certyfikatu i listy unieważnionych certyfikatów.

[10] **Uwierzytelnianie**



a. udana rejestracja,

b. odrzucone po wpisaniu login name,

c. odrzucone po wpisaniu hasła.

- hasło jednorazowe,
- uwierzytelnianie metodą wyzwanie/odpowiedź (ang. *challenge-response*),

[11] **Włamania**

```
LBL> telnet elxsi
ELXSI AT LBL
LOGIN: root
PASSWORD: root
INCORRECT PASSWORD, TRY AGAIN
LOGIN: guest
PASSWORD: guest
INCORRECT PASSWORD, TRY AGAIN
LOGIN: uucp
PASSWORD: uucp
WELCOME TO THE ELXSI COMPUTER AT LBL
```

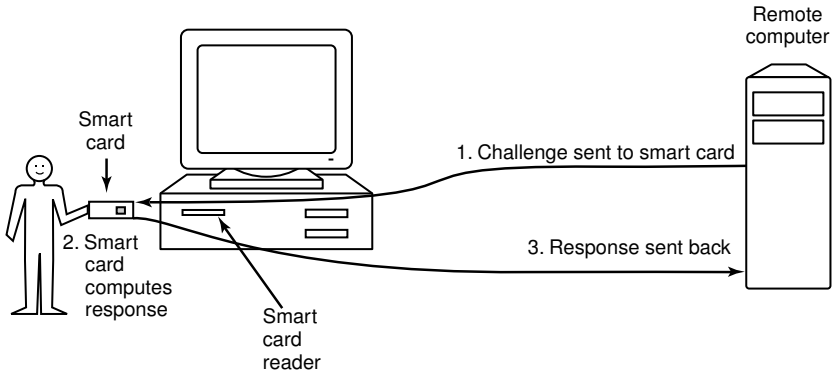
Jak włamano się do komputera Departamentu Energii USA.

[12] **Wykorzystanie soli**

Bobbie, 4238, e(Dog4238)
Tony, 2918, e(6%%TaeFF2918)
Laura, 6902, e(Shakespeare6902)
Mark, 1694, e(XaB@Bwcz1694)
Deborah, 1092, e(LordByron,1092)

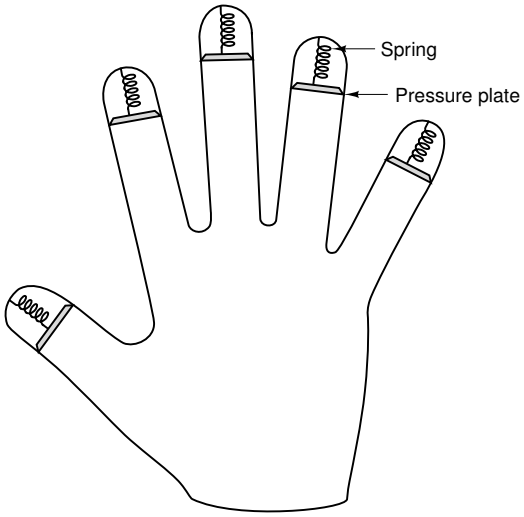
Wykorzystanie soli metodą przeciwdziałania przeciw wynajdowaniu zaszyfrowanych haseł.

[13] **Uwierzytelnianie kartą**



Wykorzystanie kart inteligentnych do uwierzytelniania.

[14] **Biometria**



Urządzenie do pomiaru długości palców.

Dobre uwierzytelnianie wymaga dwóch z trzech poniższych aspektów:

- co użytkownik zna?
- co użytkownik posiada?
- co użytkownika charakteryzuje?

[15] **Podszywanie się**

a. oryginalny ekran rejestracji,

b. podrobiony ekran rejestracji.

Konie trojańskie - przykład: rezultat wykonania poniższego zestawu poleceń zależy od zawartości zmiennej PATH:

```
cd /usr/mal
ls -l
```

[16] **Tylne furtki**

```
while (TRUE) {
        printf("login: ");
        get_string(name);
        disable_echoing( );
        printf("password: ");
        get_string(password);
        enable_echoing( );
        v = check_validity(name, password);
        if (v) break;
}
execute_shell(name);
```

(a)

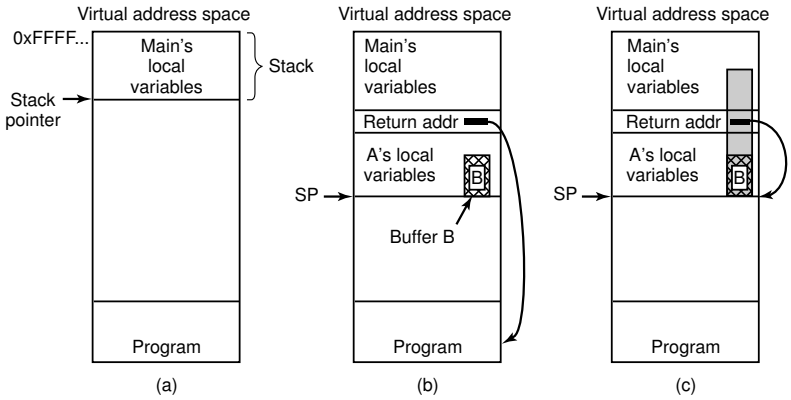
```
while (TRUE) {
        printf("login: ");
        get_string(name);
        disable_echoing( );
        printf("password: ");
        get_string(password);
        enable_echoing( );
        v = check_validity(name, password);
        if (v || strcmp(name, "zzzzz") == 0) break;
}
execute_shell(name);
```

(b)

a. kod normalny,

b. kod z tylną furtką.

[17] **Przepełnienie bufora**

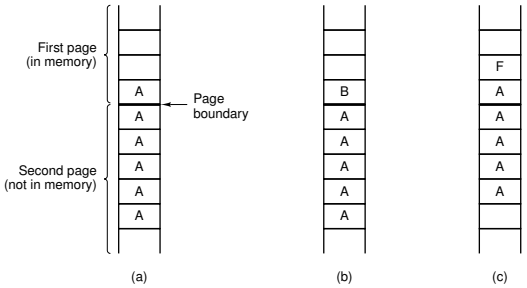


a. uruchomiona funkcja *main()*,

b. po wywołaniu procedury A,

c. przepełnienie bufora zaznaczone szarym kolorem.

[18] **Przykład ataku w systemie TENEX**

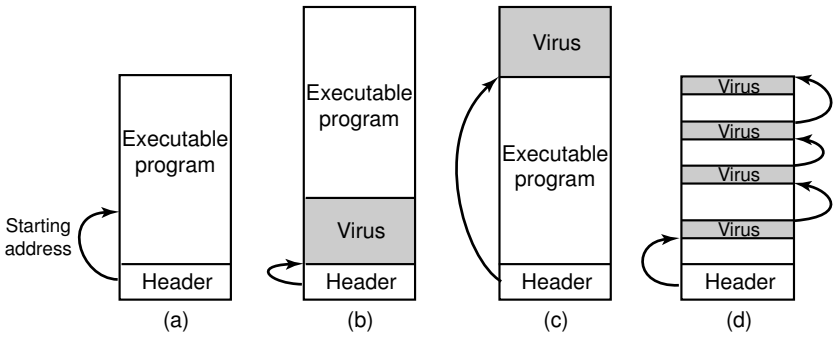


- otwarcie pliku chronione hasłem,
- system operacyjny sprawdzał hasło po literce, kończąc natychmiast, gdy hasło było niepoprawne,
- możliwość wywoływania funkcji użytkownika w przypadku wystąpienia błędu strony,
- odpowiednie ustawianie hasła plus analiza występowania błędu strony metodą uzyskania hasła,
- statystycznie $128 * n$ prób zamiast 128^n dla hasła o n znakach.

[19] **Przenikanie**

- **wirus** - program dołączony do prawomocnego programu-nosiciela, instalujący się w docelowym środowisku podczas każdego wykonania programu-nosiciela.
- **robak** - program korzystający z możliwości zdalnego uruchamiania procesów w systemach rozproszonych.
- **koń trojański** - program oferowany użytkownikom jako wykonujący pożyteczną funkcję, lecz skrywający w sobie inne cele.

[20] **Wykonywalne wirusy**



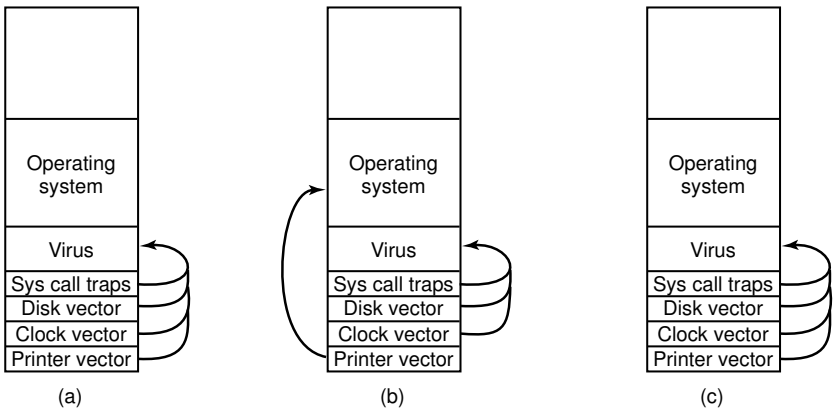
a. wykonywalny program,

b. wirus na początku,

c. wirus na końcu,

d. wirus rozproszony w wolnych obszarach programu.

[21] **Wirusy rezydujące w pamięci**

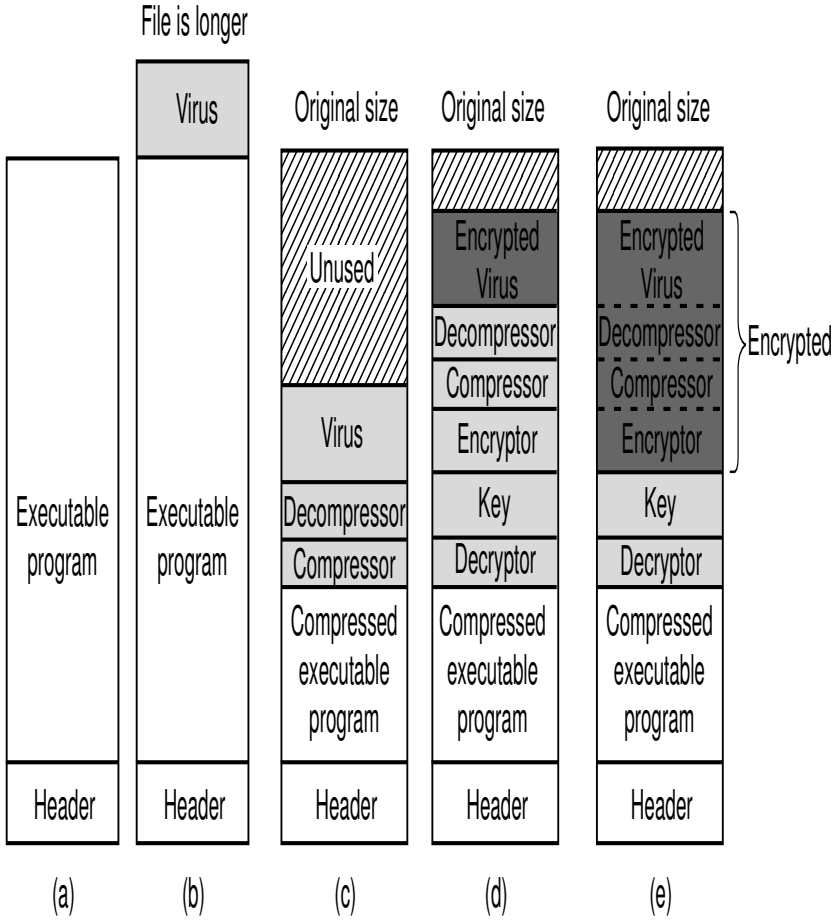


a. wirus podmienił wektor przerwań,

b. system ponownie ustawił wektor przerwania drukarki,

c. wirus ponownie przejął przerwanie drukarki.

[22] **Techniki antywirusowe i antyantywirusowe**



a. program,

b. zainfekowany program,

c. skompresowany zainfekowany program,

d. zaszyfrowany wirus,

e. skompresowany wirus z zaszyfrowanym kodem kompresji.

[23] **Wirusy polimorficzne**

```
MOV A,R1        MOV A,R1        MOV A,R1        MOV A,R1        MOV A,R1
ADD B,R1        NOP             ADD #0,R1       OR R1,R1        TST R1
ADD C,R1        ADD B,R1        ADD B,R1        ADD B,R1        ADD C,R1
SUB #4,R1       NOP             OR R1,R1        MOV R1,R5       MOV R1,R5
MOV R1,X        ADD C,R1        ADD C,R1        ADD C,R1        ADD B,R1
                NOP             SHL #0,R1       SHL R1,0        CMP R2,R5
                SUB #4,R1       SUB #4,R1       SUB #4,R1       SUB #4,R1
                NOP             JMP .+1         ADD R5,R5       JMP .+1
                MOV R1,X        MOV R1,X        MOV R1,X        MOV R1,X
                                                MOV R5,Y        MOV R5,Y
    (a)             (b)             (c)             (d)             (e)
```

Przykład wirusa polimorficznego.

[24] **Bezpieczny system operacyjny**

Wskazówki odnośnie przygotowania rozwiązania o dużym stopniu bezpieczeństwa:

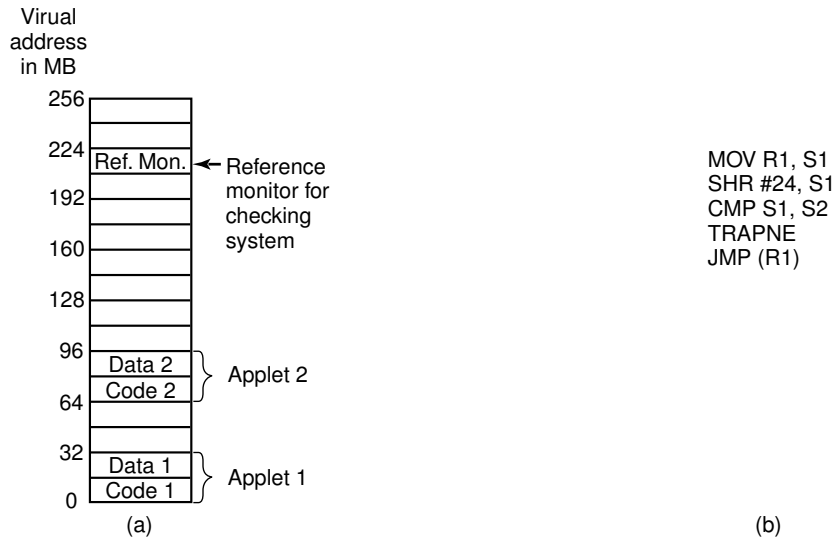
- projekt systemu powinien być prosty,
- domyślnym stanem powinien być brak dostępu,
- uwierzytelnianie w miarę na bieżąco, a nie tylko inicjalne,
- mechanizm ochrony powinien być prosty, zunifikowany i wbudowany w najniższą możliwą warstwę systemu,
- wybrany schemat musi być psychologicznie akceptowalny.
- zasada KISS - im system łatwiejszy i spójniejszy, tym mniejsza szansa na lukę w zabezpieczeniach.

[25] **Piaskownica (I)**

- podział przestrzeni adresowej na piaskownice,
- każdy aplet otrzymuje dwie piaskownice - na kod i na dane,

- idea: aplet nie może skoczyć ani odwołać się do informacji będącej poza przydzielonymi mu piaskownicami,
- sprawdzenie wstępne po załadowaniu (Java),
  1. Czy aplet próbuje preparować wskaźniki?
  2. Czy nie ma prób obejścia systemu kontroli dostępu do metod prywatnych?
  3. Czy nie ma próby użycia zmiennej jednego typu jako zmiennej innego typu?
  4. Czy nie ma prób wywołania przepełnienia stosu?
  5. Czy nie ma prób nielegalnej konwersji?
- kontrola wywołań JMP i CALL, wyzwaniem kontrola wywołań dynamicznych,

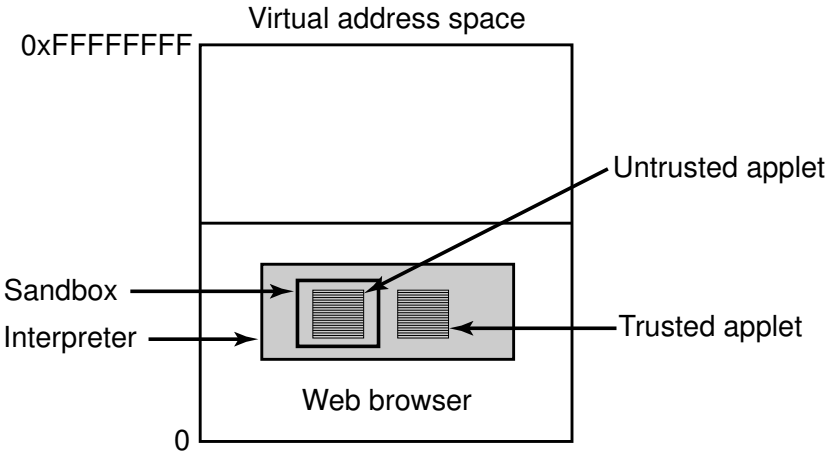
[26] **Piaskownica (II)**



a. pamięć podzielona na piaskownice po 16MB,

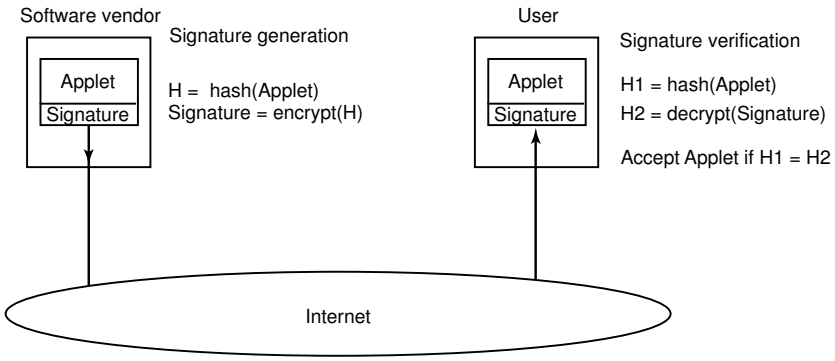
b. jedna z metod sprawdzania poprawności instrukcji.

[27] **Interpretacja**



Aplety mogą być interpretowane przez przeglądarkę WWW.

[28] **Podpisywanie kodu**



Jak działa podpisywanie kodu.

[29] **Bezpieczeństwo w środowisku Java**

- JDK 1.0 - aplety pdozielone na zaufane i niezaufane,
- JDK 1.1 - wprowadzenie podpisywania kodu,

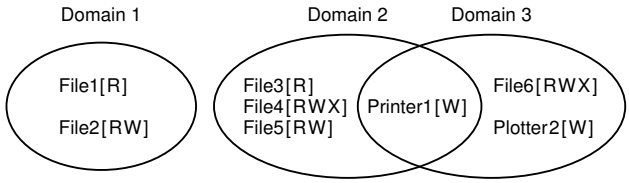
- JDK 1.2 - konfigurowalny drobnoziarnisty model oparty o pliki konfiguracyjne.

URL	Signer	Object	Action
www.taxprep.com	TaxPrep	/usr/susan/1040.xls	Read
*		/usr/tmp/*	Read, Write
www.microsoft.com	Microsoft	/usr/susan/Office/–	Read, Write, Delete

Przykłady ochrony jakie można wyspecyfikować w JDK 1.2.

[30] **Domeny ochrony**

- ochrona kontrolowana przez monitor odwołań,
- domena - zbiór par (obiekt,prawa),
- prawo - pozwolenie do wykonania pewnej operacji,
- w systemie Unix domeny ochrony determinowane przez UID i GID.



Trzy domeny ochrony.

[31] **Macierz ochrony (I)**

	Object							
Domain	File1	File2	File3	File4	File5	File6	Printer1	Plotter2
1	Read	Read Write						
2			Read	Read Write Execute	Read Write		Write	
3						Read Write Execute	Write	Write

Macierz ochrony.

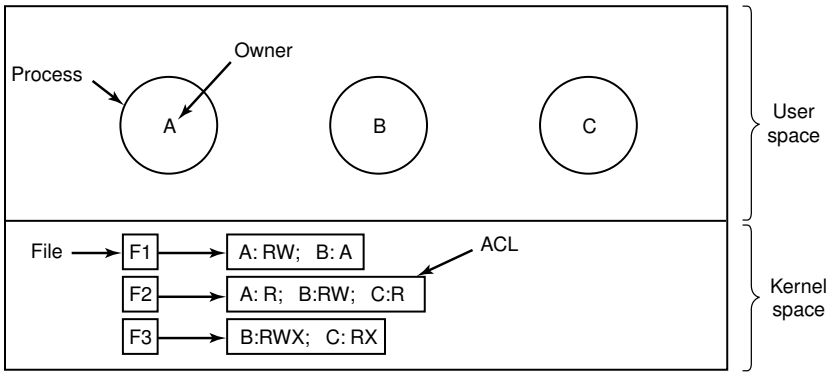
[32] **Macierz ochrony (II)**

	Object										
	File1	File2	File3	File4	File5	File6	Printer1	Plotter2	Domain1	Domain2	Domain3
Domain 1	Read	Read Write								Enter	
2			Read	Read Write Execute	Read Write		Write				
3						Read Write Execute	Write	Write			

Macierz ochrony z domenami traktowanymi jako obiekty.

[33] **Listy kontroli dostępu (I)**

ACL (ang. *access control list*)



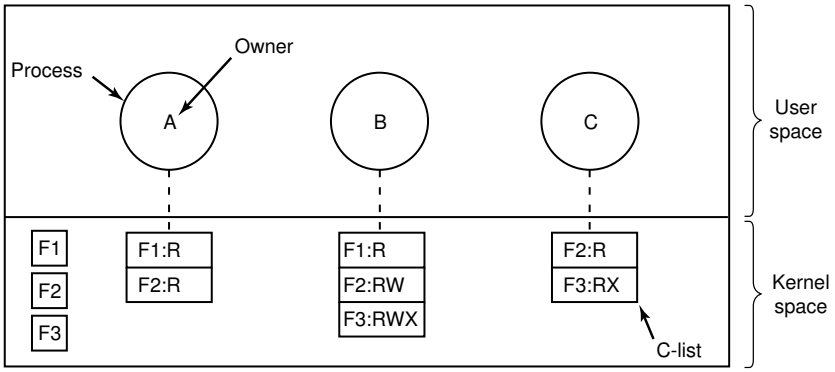
Wykorzystanie list kontroli dostępu do zarządzania dostępem do plików.

[34] **Listy kontroli dostępu (II)**

Linux/ Solaris: man **acl**, **getfacl**, **setfacl**.

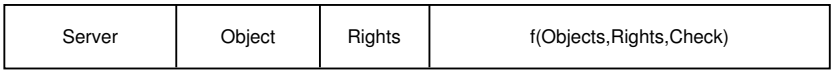
```
# file: somedir/
# owner: lisa
# group: staff
user::rwx
user:joe:rwx         #effective:r-x
group::rwx           #effective:r-x
group:cool:r-x
mask:r-x
other:r-x
default:user::rwx
default:user:joe:rwx         #effective:r-x
default:group::r-x
default:mask:r-x
default:other:---
```

[35] **Uprawnienia**



Procesy z własnymi listami uprawnień (ang. *capabilities*).

[36] **Budowa uprawnień**



Kryptograficznie zabezpieczone uprawnienie.

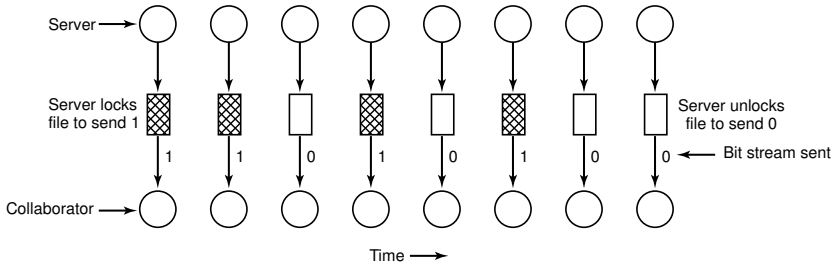
- uprawnienie atrybutem procesu odwołującego się a nie zasobu,
- uprawnienie zawiera adres oraz listę operacji, jakie można za pomocą tego uprawnienia wykonać,

- wykorzystanie funkcji jednokierunkowej do tworzenia sumy kontrolnej do weryfikacji integralności.

[37] **Bezpieczeństwo wg Orange Book**

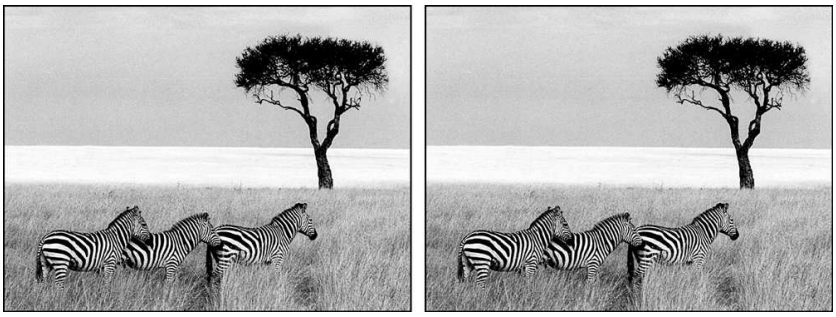
Criterion	**D**	**C1**	**C2**	**B1**	**B2**	**B3**	**A1**
Security policy							
Discretionary access control		X	X	→	→	X	→
Object reuse			X	→	→	→	→
Labels				X	X	→	→
Label integrity				X	→	→	→
Exportation of labeled information				X	→	→	→
Labeling human readable output				X	→	→	→
Mandatory access control				X	X	→	→
Subject sensitivity labels					X	→	→
Device labels					X	→	→
Accountability							
Identification and authentication		X	X	X	→	→	→
Audit			X	X	X	X	→
Trusted path					X	X	→
Assurance							
System architecture		X	X	X	X	X	→
System integrity		X	→	→	→	→	→
Security testing		X	X	X	X	X	X
Design specification and verification				X	X	X	X
Covert channel analysis					X	X	X
Trusted facility management					X	X	→
Configuration management					X	→	X
Trusted recovery						X	→
Trusted distribution							X
Documentation							
Security features user's guide		X	→	→	→	→	→
Trusted facility manual		X	X	X	X	X	→
Test documentation		X	→	→	X	→	X
Design documentation		X	→	X	X	X	X

[38] **Ukryte kanały (I)**



Ukryty kanał wykorzystujący blokowanie pliku.

[39] **Ukryte kanały (II)**



a. trzy zebry i drzewo,

b. trzy zebry, drzewo i komplet pięciu sztuk Williama Shakespeare'a.