

Wykład 11

Ochrona + Bezpieczeństwo cz. I

Ochrona (ang. protection)

Pojęcie ochrony dotyczy mechanizmu kontrolującego dostęp programów, procesów, i użytkowników do zasobów systemu operacyjnego.

Typowe zasoby to pliki i urządzenia

Cel ochrony: zapewnienie że każdy komponent systemu używa zasobów zgodnie z zadeklarowaną **polityką** (ang. policy).

Ochrona powinna dostarczać **mechanizmy** do wymuszania *polityki*.

Mechanizm określa jak coś ma być realizowane (np. bity rwx dostępu do pliku dla właściciela, grupy i całego świata w Uniksie).

Polityka określa co ma być realizowane (np. studenci nie mogą zmienić pliku wyniki_egzaminu.txt).

Mechanizm powinien być na tyle elastyczny, aby politykę można było zmienić bez konieczności zmiany mechanizmu.

Obowiązuje **zasada wiedzy koniecznej**: Proces powinien mieć dostęp tylko do tych zasobów, do których jest uprawniony, oraz których potrzebuje w danej chwili do zakończenia zadania.

Bezpieczeństwo

System jest bezpieczny, gdy zasoby są używane tak jak jest to (przez system) zamierzone ***bez względu na okoliczności.***

System może zapewniać wspaniały mechanizm ochrony gwarantujący, że użytkownik który jest studentem nigdy nie otrzyma praw zapisu do pliku wyniki_egzaminu.txt, ale

Student może się zalogować do systemu jako prof. Kowalski.

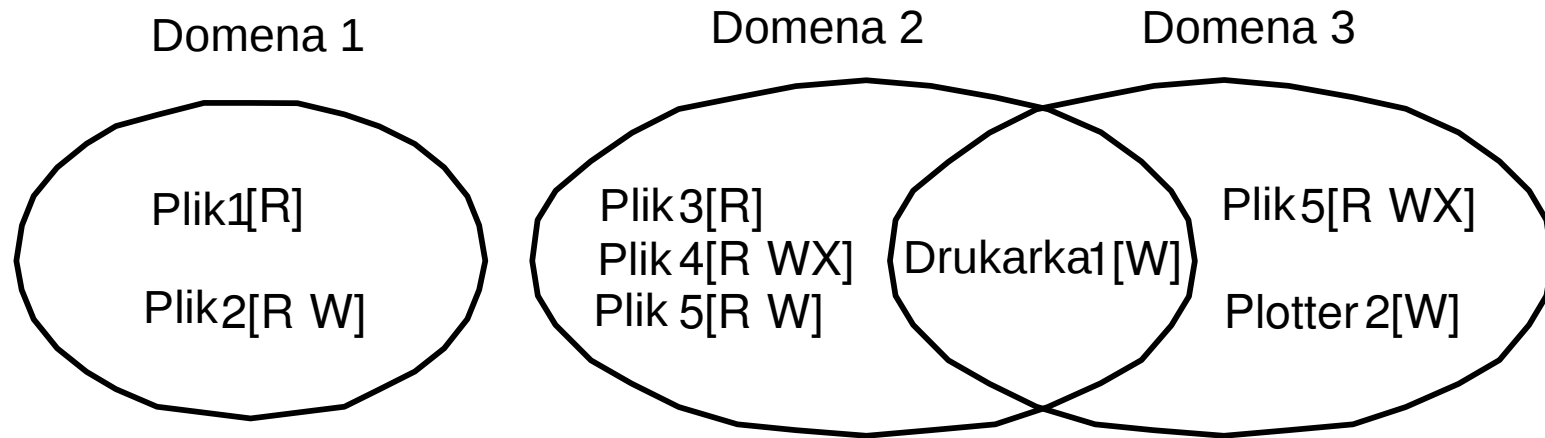
Student może poprzez złośliwe działanie (np. utwórz 10000 procesów, przydziel 10GB pamięci, inne ...) zakłócić normalną pracę systemu – atak typu *denial of service*.

Student może wystartować z dyskiety własny system operacyjny i uzyskać dostęp do wszystkich plików na dysku twardym

Student może wymontować dysk z komputera, wstawić do innego systemu, a następnie powtórzyć czynności z poprzedniego kroku.

Student mający znajomości w CIA może zarejestrować z odległości 200m wszystkie znaki wpisywane na klawiaturze.

Domeny ochrony



Obiekty sprzętowe (procesor, obszary pamięci, drukarki, dyski, ...) oraz programowe (pliki, programy, semafony).

Domena jest zbiorem obiektów, przy czym dla każdego obiektu jest zdefiniowany zbiór uprawnień.

Domeny nie muszą być rozłączne (Plik5).

Związek domeny z procesem może być statyczny albo dynamiczny (przełączenie domen).

Domeną może być:

- użytkownik.

- grupa użytkowników - zwiększa wydajność

- proces

Macierz dostępu (ang. access matrix)

		Obiekt						
		Plik1	Plik2	Plik3	Plik4	Plik5	Drukarka1	Plotter2
Domena	1	Read	Read W rite					
	2			Read	Read Write Execute	Read W rite	Write	
	3					Read Write Execute	Write	W rite

Model domenowy może być przedstawiony przy pomocy macierzy zwanej macierzą dostępu (ang. access matrix).

Mechanizm macierzy dostępu pozwala na realizację polityki.

System operacyjny musi ten mechanizm poprawnie zaimplementować, tzn. zapewnić że proces wykonywany się w ramach domeny D_i ma dostęp tylko do tych zasobów, które są występują w wierszu i , zgodnie z zawartymi tam uprawnieniami.

Przełączanie domen – obiekty jak domeny

Domena	Obiekt									
	Plik1	Plik2	Plik3	Plik4	Plik5	Drukarka1	Plotter2	Domena1	Domena2	Domena3
1	Read	Read Write								switch
2			Read	Read Write Execute	Read Write	Write				
3					Read Write Execute	Write	Write		switch	

Założenie, że każda domena jest obiektem (występuje jako kolumna macierzy dostępu) wraz z dodatkowym uprawnieniem switch (mającym sens tylko dla obiektów reprezentujących domeny) pozwala na przełączanie domen.

W powyższym przykładzie z Domeny 1 możemy się przełączyć do Domeny 2, a z Domeny 3 do Domeny 2

Dodatkowe uprawnienia w macierzy

switch – patrz poprzedni slajd.

control, podobnie jak switch dotyczy wyłącznie obiektów będących domenami. Proces wykonujący się w Domenie która ma uprawnienie control dla Domeny i domeny, może usunąć dowolne uprawnienia z wiersza związanego z Domeną i.

owner (właściciel): możliwość zmieniania uprawnień w dowolnej kolumnie związanej z danym obiektem.

copy: możliwość przekazania swoich uprawnień innej domenie (np. gdyby Domena2 miała uprawnienie copy do obiektu Plik5, to mogłaby przekazać dowolne ze swoich uprawnień innej domenie.

transfer: podobnie jak copy, ale usuwa uprawnienia z domeny źródłowej.

Implementacja macierzy dostępu

Najprostszym rozwiązaniem byłaby tablica, przechowująca trójki postaci:

<Domena,Obiekt,Zbiór_uprawnień>

Niestety rozmiar takiej tablicy byłby olbrzymi

Wyobraźmy sobie system, w którym mamy 100 użytkowników i 10000 plików, tablica miałaby wtedy milion pozycji.

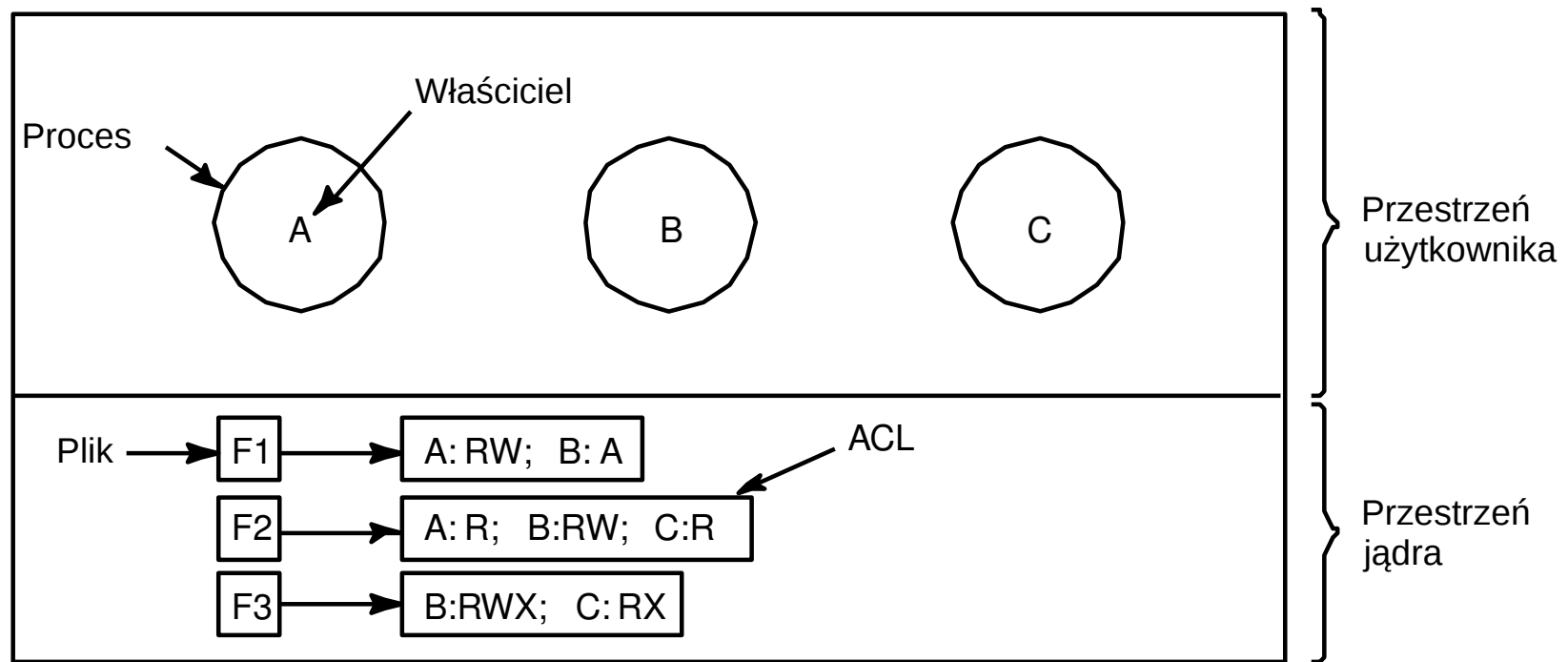
W praktyce stosuje się trzy inne inne rozwiązania:

przechowuj kolumny macierzy dostępu wraz z obiektem (tzw. listy dostępuów).

przechowuj wiersze macierzy dostępu wraz z domeną (listy uprawnień)

mechanizm klucza-zamka (rozwiązanie mieszane)

Listy dostępu dla obiektu (ang. access control list - ACL)



Lista dostępu przechowuje pary <Domena, Zbiór_uprawnień>.

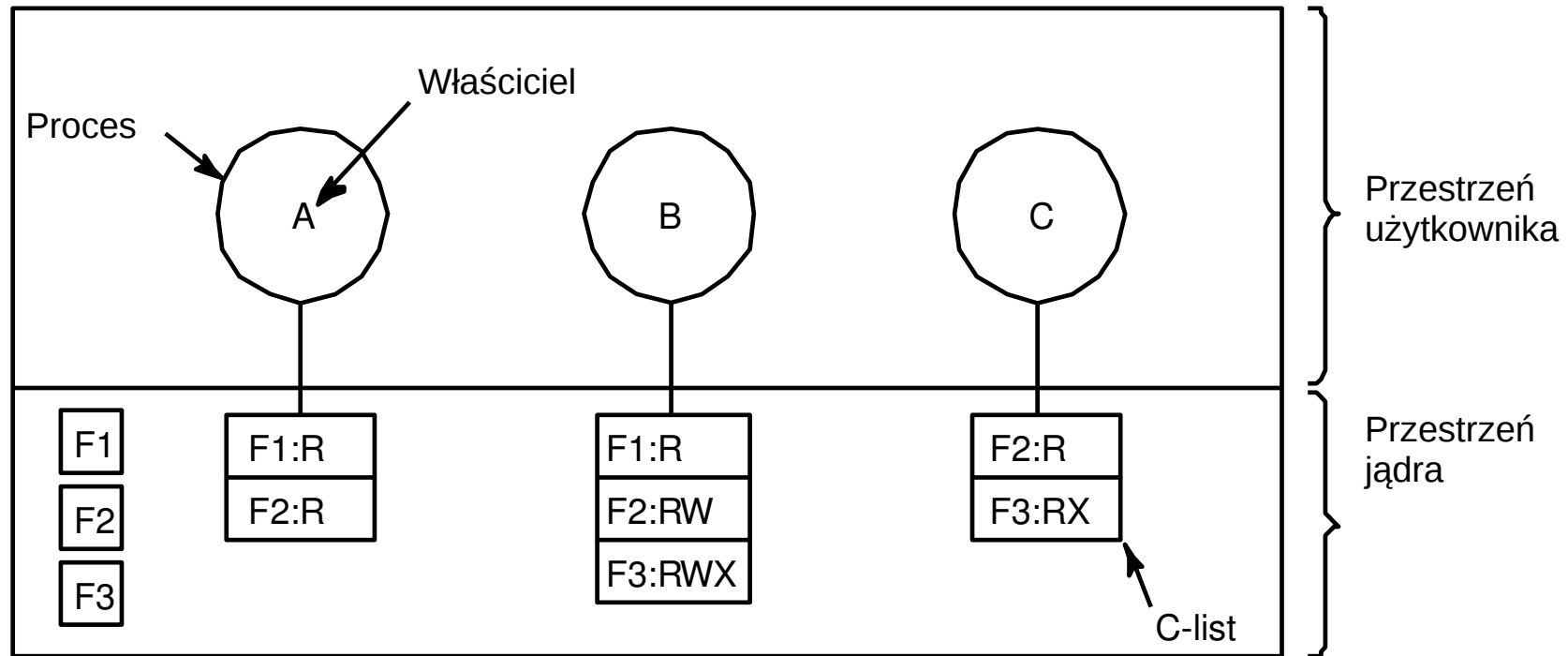
Pomijamy domeny, które nie mają żadnych uprawnień.

Wady: a) trudno jest określić zbiór praw dla konkretnej domeny b) każda próba dostępu wymaga przeszukania listy

Często do listy dostępu dodaje się domyślny zbiór uprawnień.

np *:R (każdy może czytać)

Listy uprawnień dla domeny (ang. capability lists)



Proces podejmujący próbę dostępu musi wykazać się, że ma uprawnienie.

Lista uprawnień również musi być obiektem podlegającym ochronie. Np. przestrzeń adresowa procesu może mieć odrębny segment niedostępny z poziomu użytkownika.

Proces ma wskaźnik do listy uprawnień, ale nie może jej modyfikować.

Porównanie list dostępów z listami uprawnień

Listy dostępów odpowiadają potrzebom użytkowników. Użytkownik tworzy obiekt, a określa uprawnienia dla tego obiektu.

List uprawnień nadają się szczególnie do implementacji piaskownic (ang. sandbox) np. dla kodu mobilnego (applety) któremu nie możemy zaufać.

Większość systemów stosuje kombinację tych dwóch technik.

Na przykład w Uniksie przy otwieraniu pliku (open) sprawdzana jest lista dostępów związana z danym plikiem.

Po pomyślnym otwarciu proces otrzymuje deskryptor (uchwyt) pliku - liczba 32bitowa. Deskryptor jest indeksem do tablicy otwartych plików danego procesu. W tablicy jest przechowywane m.in. tryb otwarcia pliku (odczyt/zapis). Proces nie ma bezpośredniego dostępu do tej tablicy.

Operacje read/write wymagają podania deskryptoru pliku. Nie są już sprawdzane uprawnienia związane z obiektem.

Tablica otwartych plików pełni więc funkcję listy uprawnień.

Po zamknięciu pliku (close) pliki odpowiadający deskryptorowi jest usuwany z tablicy. (Wycofanie uprawnień).

Mechanizm klucza i zamka (ang. lock-key scheme)

Każdy obiekt posiada wykaz jednoznacznych wzorców binarnych – **zamek**ów (locks)

Każda domena posiada wykaz jednoznacznych wzorców binarnych – **klucze** (keys)

Proces działający w danej domenie może mieć dostęp do obiektu, jeżeli jeden z kluczy tej domeny pasuje do któregoś zamka obiektu.

Wycofanie (ang. revocation) praw dostępu

Czasami musimy usunąć pewne prawa dostępu z domeny.

- Wycofanie natychmiastowe albo opóźnione

- Wycofanie ogólne (dla wszystkich domen) albo selektywne (dla wybranych)

- Wycofanie całkowite (tracone wszystkie prawa dostępu) albo częściowe

- Wycofanie permanentne albo tymczasowe.

W przypadku mechanizmów list dostępu oraz zamka-klucza nie ma problemów z implementacją.

W przypadku list uprawnień problem jest bardziej skomplikowany, ponieważ uprawnienia dostępu do jednego obiektu są rozproszone po wielu listach.

- Wskaźniki zwrotne (back-pointers): każdy obiekt posiada listę wskaźników do wszystkich list uprawnień związanych z danym obiektem.

- Powtórna akwizycja: co jakiś czas listu uprawnień są automatycznie kasowane. W takim przypadku proces musi ponownie spróbować uzyskać uprawnienie (i może go nie uzyskać jeżeli zostanie cofnięte).

- Każde uprawnienie nie wskazuje bezpośrednio na obiekt, ale na pozycję w globalnej tablicy, która z kolei wskazuje na obiekt.

Mechanizmy ochrony w systemie Unix - dostęp do plików

Superużytkownik (ang. root, superuser) może wszystko

Każdy plik ma właściciela w postaci użytkownik.grupa.

Użytkownik może należeć do wielu grup.

Każdy plik i katalog ma określone uprawnienia do odczytu [r], zapisu [w] oraz wykonania. Uprawnienia są określane osobno dla właściciela grupy oraz pozostałych użytkowników. Np.

```
-rwxr-xr-x 1 wkwedlo users plik5
```

Dla katalogu odczyt jest równoznaczny z przeglądaniem katalogu (ls), zapis z usunięciem albo dodaniem nowej pozycji lub zmianą nazwy a wykonanie z uczynieniem katalogu katalogiem bieżącym.

Aby usunąć plik trzeba mieć uprawnienia do pisania w katalogu w którym plik się znajduje.

Bit suid w Uniksie

Pytanie: W czym imieniu wykonuje się proces załadowany z pewnego programu ?
Czy w imieniu właściciela programu, czy w imieniu tego użytkownika który uruchomił program ?

Odpowiedź domyślnie z uprawnieniami użytkownika który uruchomił program.
Ale rozpatrzmy program passwd (właściciel root) uruchomiony przez użytkownika wkwedlo w celu zmiany hasła. Oczywiście modyfikacje pliku haseł mogą być wykonane tylko przez superużytkownika.

W Uniksie ten problem rozwiązano przez wprowadzenie dodatkowej flagi suid.
Plik mający tą flagę wykonuje się z uprawnieniami właściciela.

Programy których właścicielem jest root i z bitem suid są potencjalnie ****olbrzymim**** źródłem problemów związanych z bezpieczeństwem.

Co się stanie gdy w wyniku (być może złośliwego) podania przez użytkownika wkwedlo hasła o długości 0 znaków program passwd w wyniku błędu w kodzie skasuje plik z hasłami ?

Bit suid stosuje się także do katalogów

Naruszenia bezpieczeństwa

Całkowite bezpieczeństwo jest niemożliwe do osiągnięcia - ale należy projektować mechanizmy sprawiające, aby naruszenia były jak najrzadsze.

Ogólnie można podzielić na przypadkowe bądź celowe, złośliwe (ang. *malicious*) - obrona przed złośliwymi jest dużo trudniejsza.

Najważniejsze celowe naruszenia to:

- Nieautoryzowany dostęp do danych lub kradzież informacji.

- Nieautoryzowana modyfikacja danych.

- Nieautoryzowane zniszczenie danych.

- Uniemożliwienie normalnego korzystania z systemu (ang. denial of service, DOS).

Niezbędne jest podjęcie niezbędnych środków na poziomie:

- Fizycznym: kluczyki, kasa pancerna, ...

- Ludzkim: może nie warto przyznawać konta osobie, która jest spędza cały wolny czas w kasynie i jest zadłużona.

- Sieci: Komputery nie są izolowane, dane przesyłane są liniami nad którymi nie mamy kontroli: możliwość podsłuchu, ukrycia tożsamości, zmiany tożsamości

- Systemu operacyjnego.

Na dzisiejszym (i następnym) wykładzie dwa ostatnie punkty

Uwierzytelnianie (ang. authentication) użytkowników

Jak sprawdzić, czy identyfikator użytkownika jest autentyczny ? Istniejące podejścia oparte są na sprawdzeniu wiedzy (hasła), pomiarach biometrycznych (linie papilarne, wzór siatkówki), czy stanu posiadania (np. karty chipowe).

Hasła są najpopularniejsza, ale sprawiają wiele problemów.

Ludzie wybierają jako hasła słowa łatwe do odgadnięcia (np. swoje imię).

Użytkownik może zapisać trudne hasło na kartce i pozostawić w łatwo dostępnym miejscu.

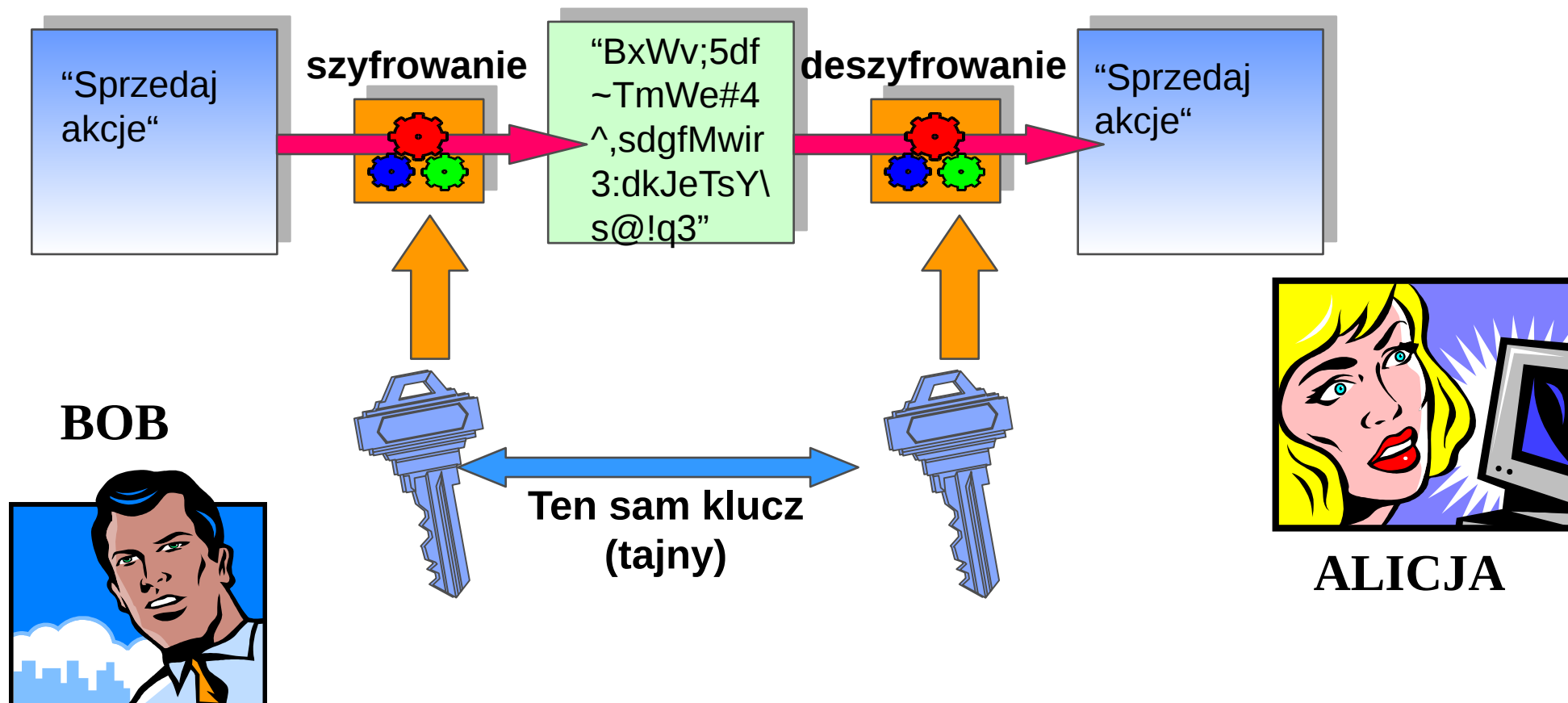
Hasła są narażone na ujawnienie w wyniku monitoringu (np. spoglądanie zza ramienia, przechwyt pakietów w sieci (o ile nie są szyfrowane)).

Problem: W jaki sposób nie przechowywać hasła w sposób jawny, ale jednocześnie umożliwić uwierzytelnienie ? Odp: Wykorzystując szyfrowanie w postaci jednokierunkowej funkcji skrótu f , np.

$f(\text{"Wojtek"}) = \text{"aqw36vbuyolrw6yefgnmyulocdxnmueq98i7rtfmuo76w"}$

Postać zaszyfrowana jest przechowywana w pliku z hasłami. Przy uwierzytelnianiu jest obliczana funkcja skrótu, a wynik porównywany z postacią zaszyfrowaną. Jeżeli funkcja skrótu jest na tyle bezpieczna, że (a) pomimo znajomości algorytmu i (b) znajomości postaci zakodowanej, nie da się analitycznie (bez przeszukania wyczerpującego) odgadnąć hasła, dostęp do postaci zaszyfrowanej niewiele daje.

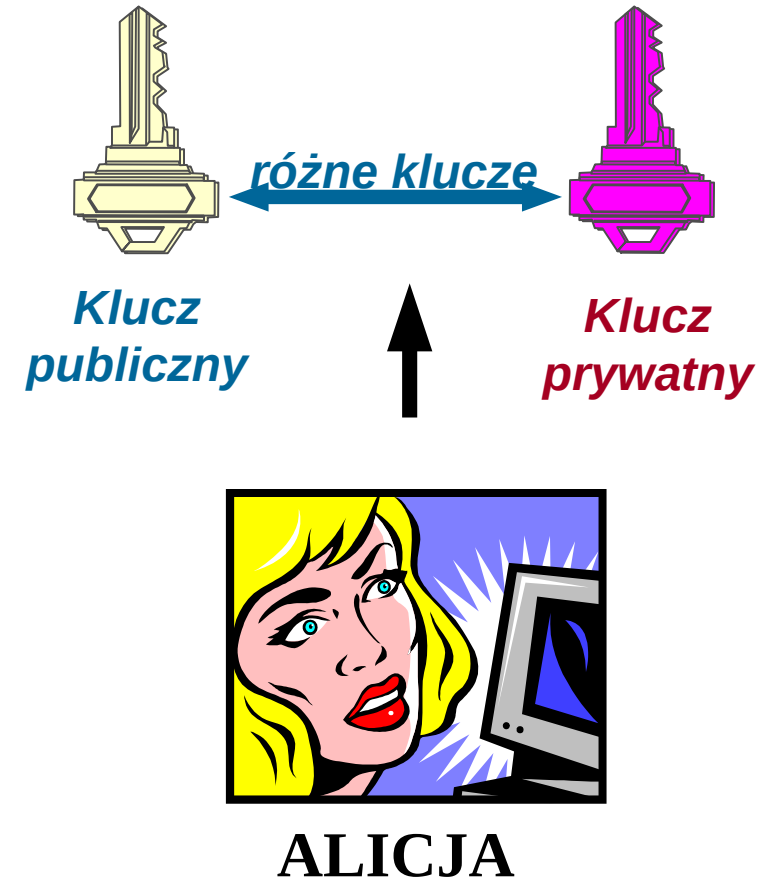
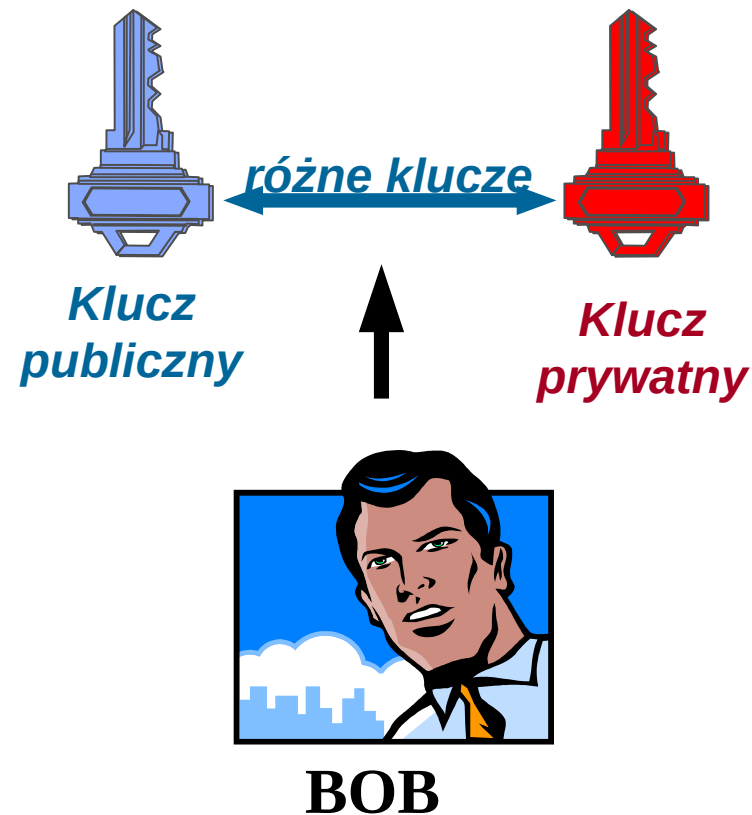
Szyfrowanie symetryczne



Algorytmy: DES, 3-DES, IDEA, Blowfish, ... (szybkie !!!)

Wada: Jak uzgodnić tajny klucz (włamywacz może podsłuchiwać !!!)

Kryptografia asymetryczna



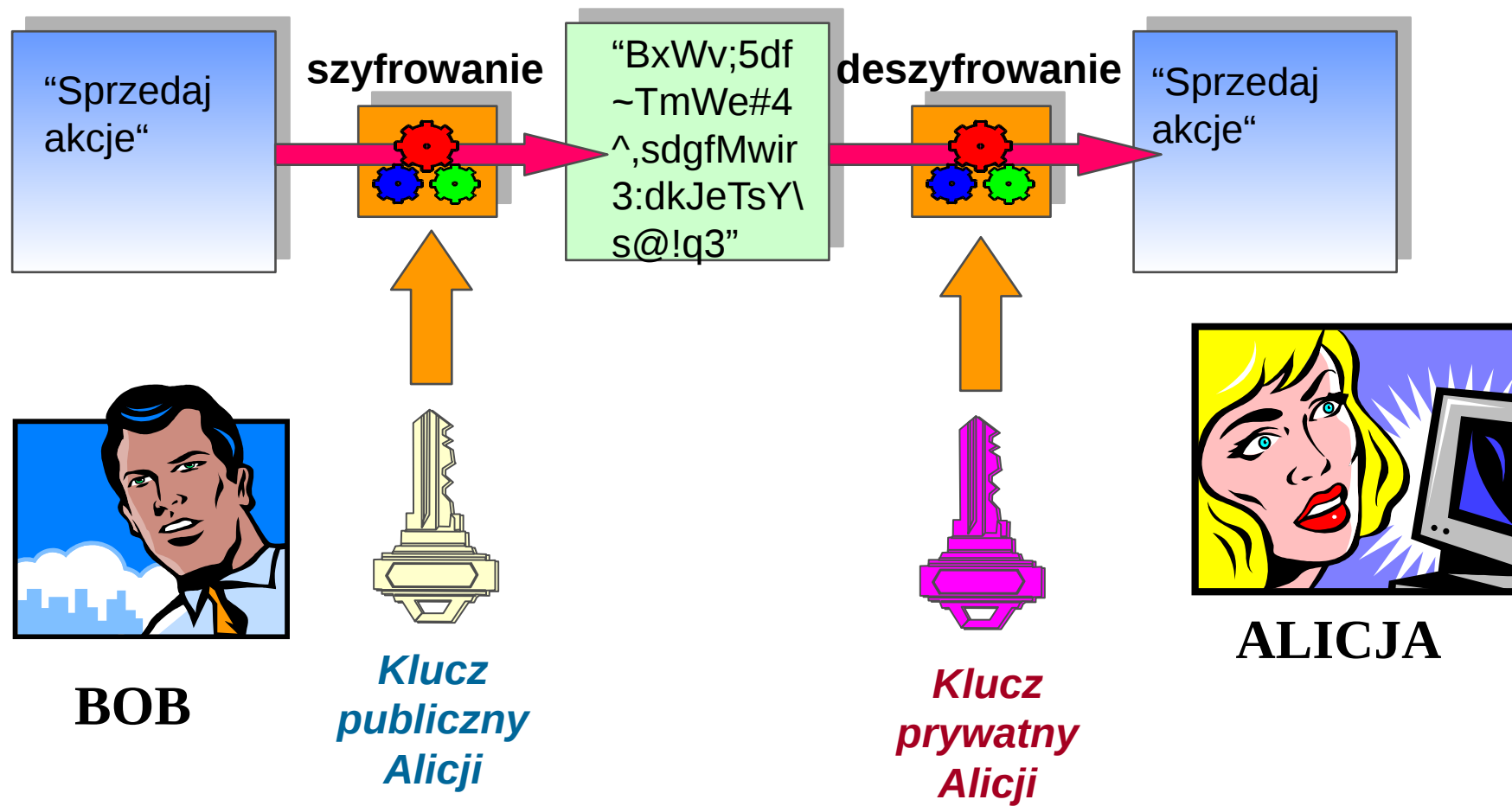
Zarówno Bob jak i Alicja generują pary kluczy:

Klucz publiczny - dostępny dla wszystkich np. w książce telefonicznej

Klucz prywatny - tajny

Algorytm działa w sposób następujący: Komunikat zaszyfrowany jednym kluczem z pary można odszyfrować wyłącznie drugim kluczem.

Kryptografia asymetryczna - przesyłanie tajnej wiadomości

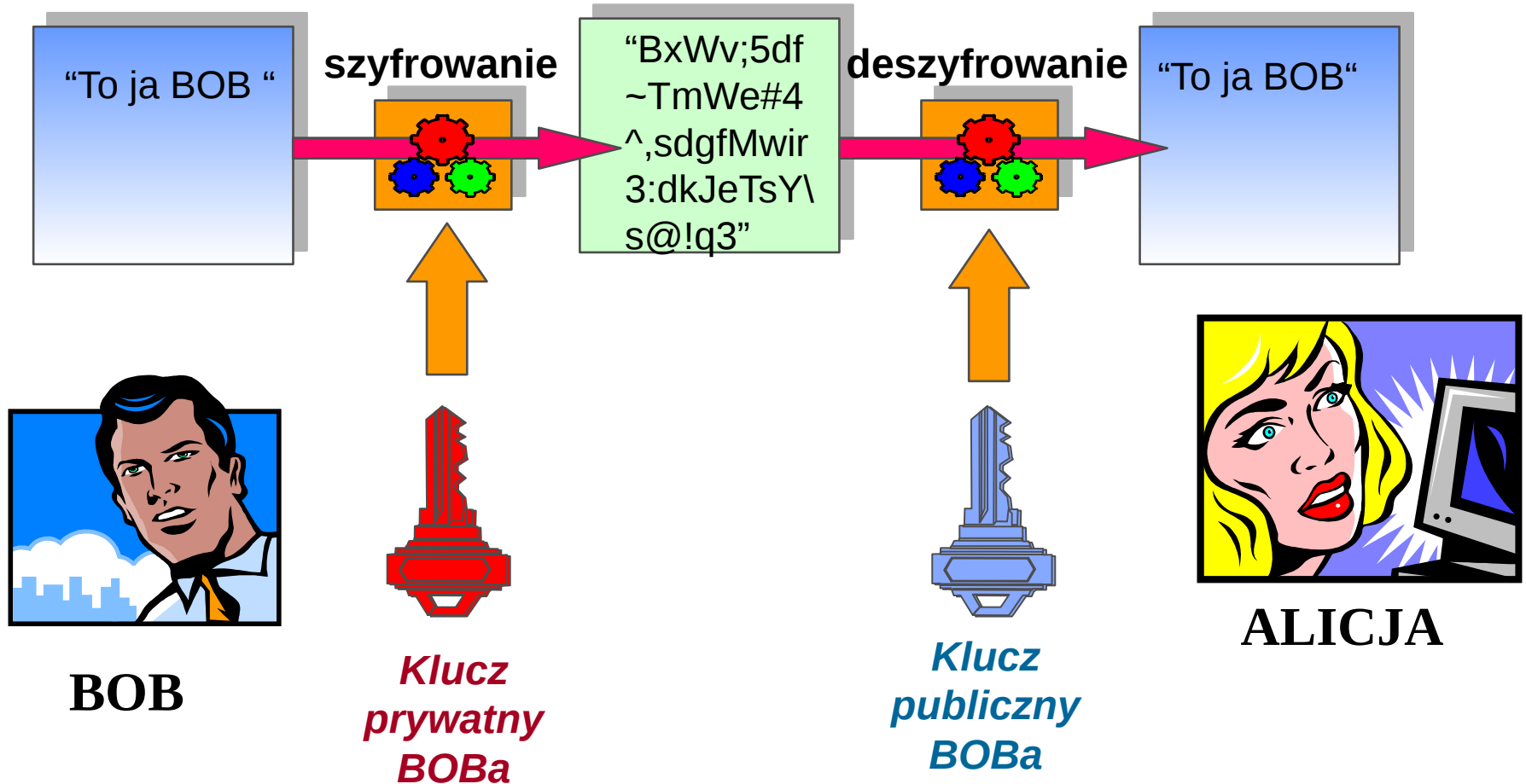


Tylko Alicja może odczytać wiadomość.

Nie trzeba uzgadniać tajnego klucza !!!

Alicja nie wie, czy nadawcą jest Bob

Kryptografia asymetryczna - podpis cyfrowy



Każdy może odczytać wiadomość.

Mamy gwarancję że napisał ją BOB.

Dobre narzędzie do uwierzytelnienia np. program ssh.

Kryptografia asymetryczna

Możemy łączyć dwie techniki np. BOB szyfruje wiadomość najpierw kluczem publicznym Alicji a potem swoim kluczem prywatnym. Alicja dekoduje używając najpierw klucza publicznego BOBa, a następnie swojego klucza prywatnego.

Alicja ma gwarancję, że komunikat wysłała BOB i jednocześnie treść jest tajna.

Techniki asymetryczne są niezwykle wolne (1000 razy wolniejsze od technik symetrycznych) - problem w przypadku długich wiadomości, ruchu w sieci, ...

W praktyce używa się rozwiązań hybrydowych, w którym techniki asymetryczne wykorzystuje się do wygenerowania i uzgodnienia klucza dla szyfru symetrycznego - ale dla użytkownika nie ma to znaczenia.

Problem: Jak zagwarantować, że publiczny klucz Alicji umieszczony w książce telefonicznej jest naprawdę kluczem Alicji. Rozwiązania: certyfikaty. Klucz publiczny jest podpisany przez organizację której ufamy: VeriSign, PCSS, Microsoft (- to żart).

W ten sposób powstaje PKI (Public Key Infrastructure) - e-banki, etc