РАНДАЛ ЙЕНСЕН
ЧАРЛЗ ТОНИЗ

# ТЕХНОЛОГИЯ НА ПРОГРАМИРАНЕТО

Превели от английски език
инж. ОГНЯН СЕИЗОВ и инж. АСЕН СЕИЗОВ

ДЪРЖАВНО ИЗДАТЕЛСТВО „ТЕХНИКА“
СОФИЯ, 1987

УДК 681.3.06

Книгата е подробно въведение в технологията на програмирането. Изчерпателно са разгледани съществуващите методи, с които програмирането се превръща от изкуство в научна дисциплина. Представеният материал е илюстриран с примери на реални програмни системи. Направен е сравнителен анализ на съществуващите методи и са набелязани очертаващите се в близкото бъдеще насоки в развитието на технологията в програмното осигуряване.

Книгата е предназначена както за ръководителите на програмни колективи, така и за самите изпълнители, разработващи големи програмни системи, чието успешно изграждане не е възможно, или най-малкото не е изгодно икономически, без да се приложат съвременните методи за програмиране.



Software Engineering
Randal W. Jensen, Charles C. Tonies

c/o Jusautor, Sofia

681

# 6

# СИГУРНОСТ И НЕПРИКОСНОВЕНОСТ

*ПАТРИК А. ХАСКЪЛ*

*По-добре здрава ограда*
*при върха на канарата,*
*отколкото линейка*
*долу в долината.*

*Джозеф Мълинс*

## 6.1. ВЪВЕДЕНИЕ

В тази глава се разглеждат създаването на сигурно програмно осигуряване и запазването на неприкосновеността на личната информация, съхранена в компютърната система. Термините *сигурност* и *неприкосновеност* не са синоними; при *сигурността* проблемът е технологичен — да се осигури запазването на информацията, а при *неприкосновеността* е социологически — да се реши каква част от информацията може да се разкрие.

За да се обсъди сигурността на компютърната система, трябва да се подготви „сцената" и да се определят участниците в „играта". Информацията, която трябва да се защищава, се съдържа в компютърната система — сцената, и се съхранява като обект. Следните елементи, написани с курсив, представляват играчките. ***Обектът*** може да бъде файл от данни, файл върху магнитна лента, текст на програма, опашка от съобщения или всеки друг набор от данни, който може да се възприема като основна единица от операционната система. *Процесът* се състои от програми и файлове от данни (по същество множество обекти), които са необходими за изпълнението на дадена задача, и информация за текущото състояние на задачата. Следователно процесът е динамично представяне на изпълнението на програма. Тъй като процесът трябва да се защитава от операционната система, процесът също се счита за обект. Процесорът е апаратната част (например централният процесор) и програмното осигуряване (например операционната система), които изпълняват задачата, като използват инструкциите и данните, съдържащи се в процеса. *Потребители* на системата са тези, които поставят начало и са отговорни за процеса в системата. *Проникващи лица* са хората, които се опитват да нарушат сигурността на системата и които понякога са легалните потребители на системата, но с непочтени намерения.

Единиците, които трябва да бъдат защитени, са процеси и обекти, така че трябва да се осигури защита, когато информацията се използува и съхранява. Най-общо компютърната сигурност може да се определи като защита на обекти и процеси от проникващи лица, докато легалните потребители имат достъп до обектите и процесите по контролиран начин.

Опитът да се наруши сигурността на компютърната система приема една от следните три форми:

1. Получаване на информация от компютъра.

2. Лишаване на други потребители от възможността да използуват компютърната система.

3. Умишлено изменяне на информацията в системата.

От трите форми първата се предотвратява или открива най-трудно. Кражбата на информация от компютъра може да се сравни с влизането на крадец в канцеларията, който отваря касата, като използува шифъра и преснима съдържанието ѝ. В много компютърни системи проникналият има достъп до повече информация може много бързо да я копира и вероятността да бъде заловен е твърде нищожна.

При втората форма на проникване легалният потребител може да бъде възпрепятствуван да използува системата или части от нея от нелегално лице, проникнало по няколко начина. Проникналият може да напише програма с безкраен цикъл, като по този начин монополизира централния процесор за себе си и намали нивото на услугите, които са на разположение на другите потребители; той може да напише програма, която монополизира всички запомнящи устройства с магнитна лента в системата и блокира изпълнението на всички други задания, изискващи такива запомнящи устройства. Друг метод е включването към комуникационна линия и добавянето на лъжливи съобщения, които нарушават синхронизацията на връзката или изискват от системата да сортира и изхвърля излишните съобщения; по този начин се намалява ефективността на комуникационната система.

При третата форма на нарушаване на сигурността на компютъра проникналото лице може да успее да промени част от операционната система и да предизвика отпадане на компютърната система. Така резултатите от програмата стават невалидни, щом проникналият въведе грешки във входните данни или изчислителна грешка в програмата. Все пак това не винаги е умишлено, много често то се получава случайно от програмистки или процедурни грешки; случайните грешки обаче могат да струват толкова скъпо, колкото и умишлените. Измамата е обработката на входните данни или на съдържанието на системата, за да покажат те погрешни резултати, които облагодетелствуват проникналото лице.

Прилагането на методи за сигурност в проектирането и реализацията на програмни системи помага да се намали този вид кражба. Създаването на сигурно програмно осигуряване обаче е само част от решението; процедурните методи, административното управление, методите за възстановяване, както и правната и социалната среда също трябва да е имат предвид. За съжаление тези фактори често са неконтролируеми.

## 6.2. НИВА НА ЗАЩИТА

Основната задача на сигурността в компютърната система е да защищава информацията всистемата. Все пак в повечето многопрограмни компютърни системи информацията трябва да се споделя между множество потребители. Следователно системата за сигурност трябва да позволява съвместно ползуване на информацията и в същото време да предотвратява достъпа до нея, който не е в съответствие със схемата за сигурност. Фактически защитата в компютърната система и съвместното използуване на информацията вървят заедно. Сложните схеми за съвместно използуване изискват сложни механизми за защита.

Тук се обсъждат шесте нива, на които един потребител може да желае да споделя информация с други потребители, и се дават примери как всяко ниво помага (или не помага) на програмиста в търсенето на сигурност.

1. Най-ниското ниво на защита е „никаква защита“ — при такава система проникващият може да стигне до всяка информация в системата независимо от съществуващата схема за съвместно ползуване.

На пръв поглед изглежда, че много програми и може би някои файлове в действителност не се нуждаят от защита и в тези случаи като че ли е подходяща незащитена система. Дори ако притежателят на програмата не се интересува кой я чете или използува, все пак самоволните изменения са проблеми. Един пример е низша библиотечна подпрограма за синусова функция. Тя като че ли спада в категорията, която не се нуждае от защита; в края на краищата всеки има свободен достъп до подпрограмата, така че не е нужно да я краде. Някой обаче може да я промени така, че тя вече да не дава правилния резултат. Изменението може да е шега, но може и да не е.

2. „Система за изолиране“ — желана е за потребител, който не иска да споделя файлове или програми с друг потребител.

Един от начините да се осигури изолация е наборът от програми и файлове да се изолира физически, като се отдели един компютър за определен набор от програми и файлове.

Вторият начин да се осигури изолация е да се създаде виртуален компютър, като се направи компютърната система да работи така, като че ли даденият набор от програми и файлове е единственият набор от обекти в системата. Изолацията дава много високо ниво на защита, но не позволява съвместно ползуване.

3. Най-ниското ниво на защита, което разрешава съвместно ползуване, е системата „всичко или нищо“ — ако потребителят има достъп до произволен файл (или програма), върху файла (или програмата) може да се изпълни всякаква операция.

Притежателят на набора от файлове не може да зададе как друг потребител да работи с тях, само че другият потребител има разрешение за достъп до файловете. Не е възможно да се предотврати случайно или умишлено изменение на съвместно използуван от упълномощен потребител файл. Все пак това положение е по-добро от случая на никаква защита, тъй като потребителят може да ограничи поне броя на хората, които могат да саботират програмата или файловете му. Тази система е като системата за изолиране, ако потребителят не даде достъп на никого до файловете и програмите си.

4. Контролирано съвместно ползуване — има разрешение за ползуване на обект, като се поставят ограничения за позволения тип ползуване. Поддържа се високо ниво на защита, тъй като на потребителя на един обект се дават само минималните права за достъп, необходими за изпълнението на задачата. Обикновено правата за достъп са малък набор от общите режими за достъп, като четене, запис и четене-изпълнение. При тази форма на съвместно ползуване може да се създаде сигурна библиотека от програми.

Докато не се даде достъп за запис до програмите в библиотеката, потребителите на библиотеката са защитени от намеса. Система с база от данни и достъп само за четене може да се използува с увереност, че никой от потребителите не може да промени който и да е елемент от базата от данни.

5. Това ниво на защита позволява на потребителите да задават режимите за достъп или начините, по които могат да се ползуват обектите.

При идеята за контролирано съвместно ползуване типовете съвместно ползуване се ограничават с осигурените от програмата за съвместно ползуване, но не могат да се осъществят всичките режими за достъп, които може да пожелае потребителят. Като се разрешава на потребителя да определя режимите за достъп, се създава по-богата среда за съвместно ползуване и потребителят

създава необходимата му степен на защита. Посочените от потребителя ограничения за достъп са три форми — зависещи от потребителя, от контекста и от данните.

При контрол на достъпа, *зависещ от потребителя*, ограниченията за обработката на обект (файл или програма) зависят от това, кой се опитва да използува обекта. Пример за такава система е файлна наличностите, като някои от потребителите могат само да четат файла, други могат да изменят определени части от файла и само няколко избрани потребители могат да изменят целия файл.

При контрол на достъпа, *зависещ от контекста*, средата, в която работи потребителят на обекта, влияе върху достъпа до обекта. На потребителя може да се откаже достъп до файл или набор от файлове, ако той работи с терминал, който е извън физически сигурната зона. Работата с файл може да се ограничи например за времето от 9,00 до 17, 00 часа, като се разреши евентуално ограничено използуване от 7,00 до 9,00 часа и от 17,00 до 21,00 часа.

Третата форма е контролът, *зависещ от данните*. Не потребителя може да се разреши или откаже достъп до запис в зависимост от съдържанието на записа. Ако файлът на наличностите съдържа ценна информация за някой продукт, който все още се разработва, на повечето от потребителите на този файл може да бъде отказано да го ползуват.

6. Дотук програмите за съвместно използуване и защита включват предотвратяване или ограничаване на достъпа до информацията: потребителят обаче може да ограничи начина за изработване на информацията, след като е разрешен достъп до нея, т. е. защитните механизми могат да позволяват да се ползуват определени данни за статистически анализ, но да не се разрешава непосредствено да се отпечатват тези данни.

Една идея, тясно свързана с проблема за ограниченото ползуване, е да се проследява информацията, която се използува съвместно. В някои случаи може да е необходимо да се знае местоположението на всички копия на съвместно използуваната информация. Един от аргументите е, че всеки може да поиска да види кредита си и ако открие грешка, ще е необходимо да се открият всички копия с погрешни данни и да се поправят. Ограниченото ползуване на сметководни книги намалява проблема за проследяването на всички погрешни сметки, тъй като търсенето се ограничава само сред потребителите, на които са разрешени достъп до записа и да притежават копие от него.

Програмистът може да избере нивото на защита, което желае да ползува, за да осигури сигурността на информацията, въвеждана в компютърната система. Единственият проблем е действително да се осигури желаното ниво на защита.

## 6.3. ПРЕГЛЕД НА ГЛАВАТА

Програмистът използува няколко средства, за да осъществи схемите за съвместно ползуване. В този смисъл *средствата* са всичко , което той прилага, за да подпомогне получаването на сигурност. Тези средства включват методите за реализиране на програмата, както и съществуващите програмни и апаратни средства.

Първото и най-основно средство е операционната система, която за това обсъждане включва и апаратната част за адресиране на паметта.

Второто средство е система за установяване на самоличността. Въпреки че тайнописът, третото средство, има продължителна история, той също осигурява защита за компютърните системи.

Програмистът може да вгради в програмата контролното проследяване, механизмите за откриване на заплахи и проверките за вътрешна свързаност. Контролното проследяване и механизмите за откриване на заплахи са бариери срещу проникващите; ако проникващият знае, че неговото използуване на програмата се регистрира в контролно проследяване или че механизмът за откриване на заплахи открива и може би регистрира опита му да проникне през защитата на програмата, той може да се откаже от намеренията си. Проверките за вътрешна свързаност откриват стойности на данните извън диапазона, предвиден от проектанта на програмата.

Накрая в употреба влиза строгото ограничаване. То гарантира, че няма информационна пътечка, която дава възможност на съмнителната програма да предаде информация на програма или потребител извън областта на ограничаването.

Обикновено изброените средства се използуват заедно.

Осигуряването на сигурността в програмната система изисква ресурси и следователно „се състезава" с другите възможности в програмата; така че програмистът се изправя пред традиционната компромисна ситуация. Например зависимите от данните ограничения за достъп и утвърждаването на всеки достъп могат да изискват прекалено високи изчислителни разходи, за да са приемливи в система, работеща в реално време. Понякога ръководството решава да рискува случайното разкриване на информацията от файловете, отколкото да влоши работата на системата. Организацията на файловете и произволният достъп до тях могат да направят невъзможно ползуването на сериозна система на шифриране. Контролното проследяване често води до огромен обем данни, които могат да се използуват трудно. Сигурността трябва да се интерира по-цялостно с проекта на програмата от повечето възможности на програмата; трудно, скъпо и може би невъзможно е да се прибави сигурност към програма, която вече е проектирана и реализирана.

## 6.4. ОПЕРАЦИОННИ СИСТЕМИ

Операционната система образува основата, върху която трябва да се гради програмата. Всяка програма, създадена в дадена операционна система, може да бъде сигурна само толкова, колкото е сигурна самата операционна система. Както при всички типове системи за сигурност, включително и при физическата сигурност, всеки, който се опитва да проникне през защитата, се опитва да открие най-слабата точка, за да атакува системата. Следователно програмистът трябва да познава възможностите и пропуските в операционната система, която ползува, преди да може да проектира и реализира сигурна програмна система, ако такава изобщо може да бъде създадена.

Написването на програма за определен от потребителя контрол на достъп, работещ под управлението на операционна система, която не осигурява дори изолация, прилича на съхранение на злато в сейф с еднометрови стоманени стени и таван, но с пясъчен под. Като крадеца, който няма да тръгне през стоманените стени, за да получи съдържанието на файловете, защитени от програмите, нарушителят търси пропуските в операционната система, а не се опитва да проникне непосредствено в програмите.

Обсъждането на операционната система тук се разделя на четири основни части. Първо, методите, чрез които се реализират различните схеми на защита. Второ, начините, по които операционната система решава кои потребители са упълномощени да ползуват определени файлове и програми. Трето, начинът, по който операционната система поддържа контролите за достъп, зададени от потребителя. Четвърто, примери за операционни системи.

### 6.4.1. ЗАЩИТНИ МЕХАНИЗМИ В ОПЕРАЦИОННИТЕ СИСТЕМИ

Операционната система може да осигури схеми за защита като изолация, неограничено съвместно ползуване и контролирано съвместно ползуване, но е ограничена само с поддържането на зададените от потребителя контроли за достъп. Тук се обсъждат начините за осигуряване на първите три схеми за защита, като отначало се описва как отместващите регистри осигуряват много ниско ниво на сигурност и след това как системата на регистри може да се разшири на етапи, докато осигури високо ниво на сигурност и позволи контролирано съвместно използуване. Четвъртата схема за защита е потребителската спецификация на контролите за достъп.

#### Изолиране на програмата

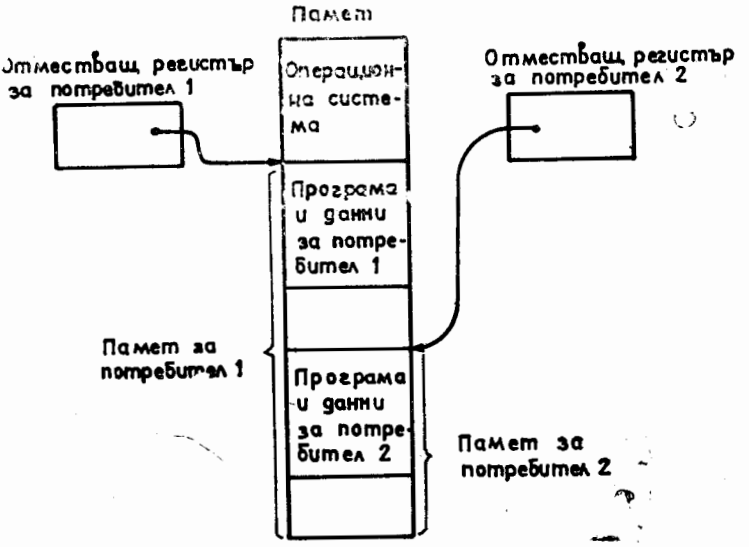
Изолирането се постига или като целият компютър се предостави на един потребител, или когато средата е многопрограмна, се изиска от операционната система да постави прегради между програмите, които се обработват едновременно в системата. Дори компютър, предоставен на един потребител, все пак има две програми, които трябва да бъдат защитени една от друга: програмата на потребителя и операционната система.

В средата на единствен потребител защитата се осигурява чрез отместващ (базов) регистър, който отделя потребителската програма от операционната система. Съдържанието на този регистър се прибавя към всеки адрес, формиран от потребителската програма, преди апаратните средства да извършат фактическия достъп до паметта. Адресите, създавани от операционната система, не се прибавят, следователно операционната система се намира в паметта на по-ниски адреси. Ако адресите, създадени от потребителския процес, не могат да бъдат отрицателни, паметта, която съдържа операционната система, е защитена от потребителския процес. От друга страна, операционната система може да генерира всякакъв адрес и така да стигне до всяка част от паметта, отредена за потребителската програма. Следователно отместващите регистри създават минимално ниво на защита, необходимо в среда с единствен потребител.

Когато няколко потребители ползуват съвместно и едновременно един компютър, простите отместващи регистри са неподходящи, тъй като регистърът осигурява само долната граница на всеки адрес, използуван от процеса. На фиг. 6.1 е пояснена невъзможността да се защищават две отделни програми, като се използуват отместващи регистри. Потребител 1 е защитен, тъй като отместващият регистър, който се съдържа в процеса на потребител 2, разрешава на потребител 2 да използува само паметта, която се съдържа от началото на неговата програма до края на оперативната памет. Съответната област на паметта, до която потребител 1 има достъп, обаче включва програмата на потребител 2.

Следователно отместващият регистър е ефективен само при работа с единствен потребител. Ето защо, за да се осигури ефективна изолация и следователно защита в многопрограмна среда, отместващият регистър трябва да бъде разширен и да включва базов адрес и дължина. Такъв регистър се нарича дескрипторен регистър. Там, където процесът образува адрес, като използува даден дескриптор, апаратните проверки гарантират, че обръщението към физическата памет се намира в раздела, определен от дескриптора. Програмата има пълен достъп до всичко в раздела по силата на притежаването на дескрипторния регистър, но няма достъп до нищо извън този раздел.

В схемата за адресиране, основаваща се на дескрипторни регистри, всеки процес може да получи раздели в оперативната памет и в областите на външната памет. Дескрипторните регистри, които се проверяват винаги, когато процесът стига до някои раздел, не разрешават достъп на процеса до който и да




Фиг. 6.1. Множество потребители в среда с отместващи регистри

е раздел, щом той принадлежи на друг процес. Дескрипторните регистри осигуряват основата на виртуалния компютър, т. е. компютърът наподобява няколко компютъра. Всеки потребител работи с един от тези „подкомпютри", а не с целия компютър. Процесорът се предоставя на един потребител за сравнително кратко време, след това превключва на друг процес и т. н. Процесорът също нанася потребителската виртуална памет в действителната памет; всеки потребител вярва, че има значителен обем памет, но операционната система зарежда повечето от информацията му върху диск и само малка част за непродължително време в оперативната памет.

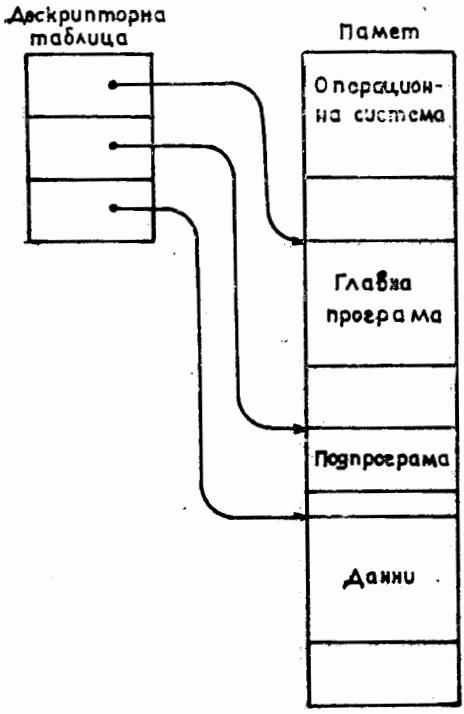
Дескрипторните регистри се управляват от процесора във виртуална компютърна среда, за да не може един процес да получи дескриптор за раздел от друг процес. Управлението на дескриптори в средата на виртуалния компютър изисква инструкции за подмяна на дескрипторните регистри, механизъм за блокиране на изпълнението на инструкции от потребителя, които изменят дескрипторните регистри, както и безопасно място за съхранение на дескрипторния регистър, когато процесорът обслужва други процеси.

Поставянето на нов дескрипторен регистър променя частта от паметта, достъпна за потребителя. Тази операция трябва да се ограничава, в противен случай всеки потребител може да зареди в дескрипторния си регистър каквато и да е стойност—неблагоприятна ситуация при опит да се осигури изолация, тъй като той може да увеличава големината на раздела, описан от дескриптора, докато включи паметта на друг потребител. Следователно операционната система трябва да запазва самостоятелното ползуване на операцията, която установява дескрипторният регистър. Индикатор, който се нарича бит на привилегированото състояние, определя дали управлението в момента е у потребителя или в операционната система и следователно дали дескрипторният ре-

гистър може да се зареди. Индикаторът се „вдига", когато управлението се предава на операционната система, и се „сваля", когато управлението се върне у потребителя. Външната памет се защищава, като се ограничава ползуването на инструкции за достъп до нея. Тези инструкции също могат да бъдат изпълнени само когато управлението е в супервайзора; в няк и случаи изпълнението на една инструкция води до автоматично предаване на управлението в операционната система.

За да се даде възможност на няколко потребители да ползуват съвместно компютъра, в паметта трябва да има място за представяне на дескрипторните регистри, които не се ползуват. При някои машини са налице няколко дескрипторни регистъра; индексен регистър определя кой действува текущо като управление на достъпа до паметта. В други случаи стойностите на дескрипторния регистър, които не се ползуват текущо, се съхраняват в някоя област, която е защитена от потребителя. Ако потребителят може да стигне до представянията на дескрипторния регистър, той може да увеличи дължината на своя участък, за да стигне до паметта на други потребители или да пречи на другите потребители като модифицира или унищожава дескрипторите им. Затова обикновено целият набор от дескриптори се защищава, като се запомнят представянията в раздела, определен за супервайзора.

В описаната програмна система с дескриптори процесорът, паметта и периферните устройства се ползуват съвместно от потребителите на системата, като се осигурява виртуална машина за всеки потребител. Употребата на дескрипторни регистри забранява взаимодействието между потребителите с изключение на съревнованието за неголемия процесор и комуникационните линии към периферните устройства. По принцип този механизъм изглежда прост и лесен за осъществяване и когато се изгради правилно, системата с дескриптори може да изолира потребителите. Тогава програмите са защитени напълно една от друга. Грешки при реализацията или пропуски в проекта при повечето съществуващи операционни системи все пак позволяват да се премине бариерата между процесите с цел да се открадне информация.



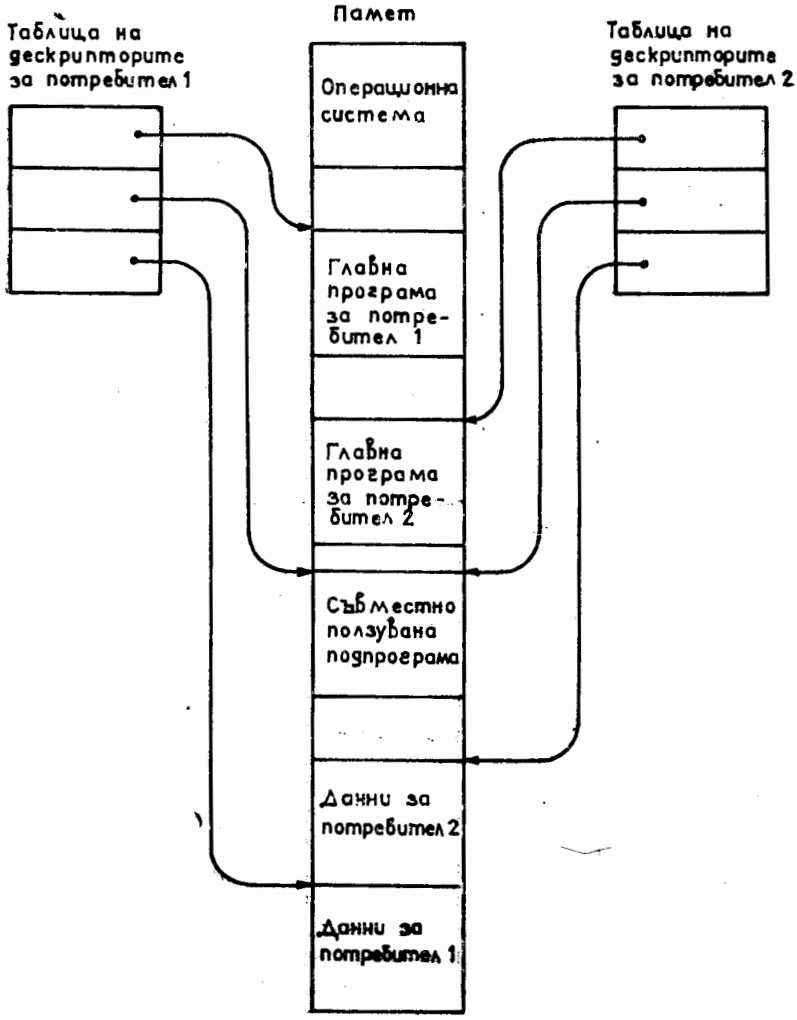
Фиг. 6.2. Защита чрез дескриптори за един потребител

### Неограничено съвместно ползуване

Същественото в защитния механизъм на описаната виртуална система е пълната изолация на потребителите, с изключение на конкуренцията за ресурсите на системата. Файловете, базите от данни и библиотечните подпрограми, създадени от един потребител, не са достъпни за другите потребители в системата. В много случаи това не е желателно. Ако потребителската група въведе обща библиотека от обслужващи програми, която да се ползува от всички членове на групата, става нужно да се разшири основаващата се на дескрип-

тори система за изолиране, за да се разреши някаква форма на съвместно ползуване.

Неограниченото съвместно ползуване е втората форма на защита, която може да се реализира чрез операционната система. Тя не се получава през раз-




Фиг. 6.3. Защита чрез дескриптори за много потребители

ширение на определението или функцията на дескрипторните регистри, а като се осигурява за всеки процес множество такива регистри в таблица. В резултат на това наличната за процеса памет се разбива на области, които се наричат сегменти с дескриптор, който адресира всеки сегмент. След това потребителят разбива този процес на определени логически единици, като поставя всяка единица в сегмент. Сегментите могат да се намират в оперативната или външната памет.

На фиг. 6.2 е изобразен потребител, който е разбил процеса на главната програма на блок от данни и на подпрограма, всички в оперативната памет.

Дескрипторите дават достъп на потребителя само до тези три области от паметта; другите области, включително и операционната система, са защитени от потребителя.

На фиг. 6.3 е показана същата система, но първият потребител ползува подпрограмата съвместно с втория. Вторият потребител има дескриптор към подпрограмата и може да я ползува, но няма достъп до главната програма или данните на първия потребител. Разбира се, само един от потребителите работи и фактически ползува дескрипторите си през всяко време; дескрипторите на другия потребител се съхраняват от операционната система. Този метод може да се разшири, за да уреди съвместно ползуване на няколко програми от няколко потребителя, като се увеличат броят на дескрипторите за всеки потребител и броят на копията от дескрипторите за всяка програма. При неограниченото съвместно ползуване процесите са защитени един от друг, но все пак им е разрешено да ползуват съвместно сегменти.

### Контролирано съвместно ползуване

Описаното ползуване на метода на множество дескриптори от операционната система означава, че когато един процес получи достъп до сегмент (програма или данни), върху сегмента може да се изпълни всякаква операция. За да се контролира достъпът до сегмента, създава се двоично поле (поле за контрол на достъпа) за всеки сегмент и начините за достъп за сегментите се поставят в двоичното поле. Обикновено задаваните начини за достъп се ограничават до четене, запис и изпълнение, следователно дължината на полето за контрол на достъпа може да бъде три бита. Всеки бит означава един начин за достъп. Сегментът може да се запише само ако е включен битът свързан с достъп за запис; другите режими се изпълняват по същия начин.

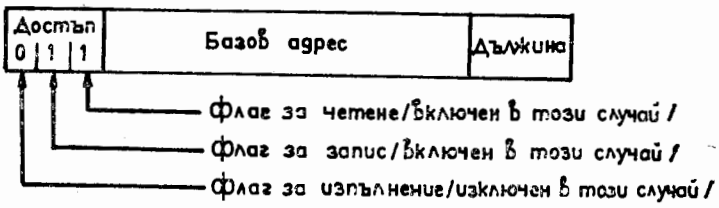
Полето за контрол на достъпа може да се съхранява заедно със сегмента или с дескрипторите, които се ползуват, за да са стигне до сегмента. Свързването на полето за контрол на достъпа със сегмента, а не с дескриптора е ограничаващо, тъй като всеки потребител има еднакви права върху сегмента; не е възможно на един потребител да се разреши да записва данни в база от данни, а на друг—само да чете данни от базата. Следователно в тази част се разглежда предимно осигуряването на контролирано съвместно ползуване, като поле за контрол на достъпа за обекта се свързва с дескриптора за обекта.

Поле за контрол на достъпа се добавя към дескриптора, което се разширява определението на дескрипторния регистър така, че да включва полета за контрол на достъпа, а също началния адрес и дължината на сегмента. На фиг. 6.4 е показан възможен формат на разширения дескрипторен регистър, при който указанията за достъп са установени така, че сегментът може да чете и записва, но не и да се изпълнява. На фиг. 6.5 е показан сегмент от данни, който се ползува съвместно от собственика и от друг потребител. Собственикът има достъп за четене /запис, е потребителят—само за четене.

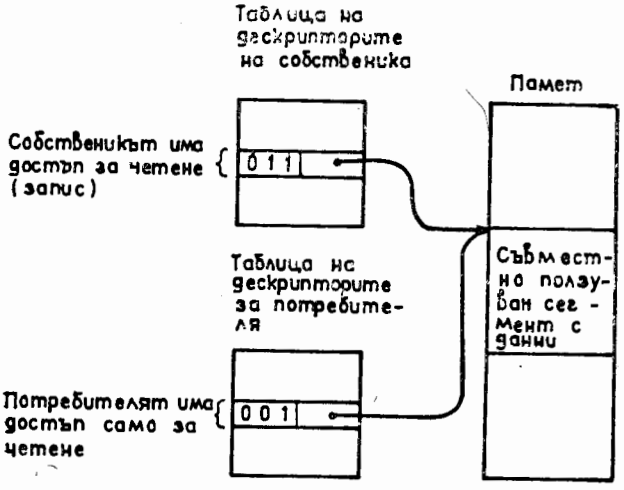
Контролираното съвместно ползуване ограничава правата за достъп до сегмента, като осигурява по-високо ниво на сигурност, отколкото при неограниченото съвместно ползуване.

Трите защитни механизма (изолация, неограничено съвместно ползуване и контролирано съвместно ползуване), които могат да се осигурят от една операционна система, се реализират, като се използуват различни форми на отместващи и дескрипторни регистри. Чрез операционната система потребителят трябва да приложи тези защитни механизми към сегмента като цяло и е ограничен да работи с начините за достъп, осигурени от компютърната система.

Четвъртата схема за защита (потребителската спецификация на контролите за достъп) може да се поддържа само от операционната система, тъй като потребителят трябва да бъде свободен да реализира произволен защитен механизъм, който може да се приложи повече към избрани части от сегмента, отколкото




Фиг. 6.4. Дескрипторен регистър с поле за достъп




Фиг. 6.5. Съвместно ползуван сегмент от данни с употреба на дескриптори

към сегмента като цяло, и да се предвиди всяка комбинация от потребителски информационни или контексно зависими ограничения за достъп.

## Контроли за достъп, специфицирани от потребителя

Тук се приема обикновеният подход към контролите за достъп, определени от потребителя, т. е. операционната система осигурява основа за защитени подсистеми. Защитената подсистема се състои от набор от програми, които могат да бъдат написани от потребителя, могат да се извикат само в определени входни точки, като иначе не са достъпни за потребителите. По тази причина информацията в подсистемата е защитена от процеси извън подсистемата. Информация, съхранена в защитената подсистема, може да се получи от потребителя само ако той повика подсистемата в една от входните ѝ точки и изиска информацията. След това защитената подсистема извършва необходимите проверки за пълномощия, преди да даде информация на потребителя, извикал подсистемата. Ако защитената подсистема получи надеждна информация за това,

кой прави заявката и откъде я прави, подсистемата може да извърши потребителски и зависещи от контекста проверки за ограничаване на достъпа. Очевидно зависещите от данните ограничения за достъп се реализират сравнително лесно, тъй като подсистемата трябва само да прегледа данните за ограничена информация, преди да я даде на заявителя.

И така, докато сегментът се ползува като първостепенен пример за обектите, които трябва да бъдат защитени, има и други типове обекти като магнитнолентови и дискови файлове, които трябва да бъдат третирани различно от операционната система. Разликите при манипулирането със сегментите и другите обекти са главно в инструкциите, които се прилагат за обработване на обектите и операциите, изпълнявани върху обекта. Начините за достъп могат да се различават за всеки тип обект. Не е възможно например да се изпълни непосредствено дисков файл, който съдържа изпълнима програма; файлът трябва да се зареди в паметта, преди програмата да може да се изпълни. Основната идея на защитния механизъм—разделяне на обектите на отделни единици посредством употребата на дескрипторни регистри и ограничаване на операциите, които могат да се изпълняват върху обектите, остава същата; само начините за достъп и подробностите, как операционната система изпълнява операциите, се променят за всеки тип обект.

### 6.4.2. СПЕЦИФИКАЦИИ ЗА ДОСТЪП

Операционната система изисква както спецификации за достъп, така и сведения на кои потребители се разрешава да имат достъп и до кои обекти. Това е необходимо, за да се наложи определена схема за съвместно ползуване, желана от програмиста. Ако потребители 1 и 2 искат да ползуват съвместно една програма, системата трябва да бъде „уведомена“ за това, както и за начина на достъп, разрешен за всеки от тях. Дори само един потребител да има достъп до дадена програма, системата трябва да „знае“, че потребителят с право на достъп е един и кой е той.

Спецификациите за достъп не осигуряват сигурност—това е работа на операционната система с описаните механизми за защита.

#### Матрици за достъп

Прилагането на матрица за достъп е много обобщен метод за съхранение на спецификации за достъп. Тя се използува като основа за описание на спецификациите за достъп.

Спецификациите за достъп се състоят от тези форми на достъп до обектите (елементите като файлове или програми, защитени от системата), които са разрешени в даден домен. Доменът е сбор от елементи (потребители, процеси или процедури), които имат достъп до набор от обекти. Всяка област може да включва набор от обекти и права за достъп до тях. Всеки елемент в дадена област има точно същите права за достъп към набора от обекти, но всеки елемент може да принадлежи на повече от един домен. Пример за домен е потребител, който има набор от файлове и програми; потребителят е целият домен, тъй като никой друг няма достъп до тези файлове и програми. Ако потребителят създаде нова програма и иска да я тества, без да рискува да повреди другите файлове и програми, той може да постави програмата в нов домен и да разреши на домена да има достъп само до допълнителните файлове и програми,

необходими за изпълнение на новата програма. Останалите програми и файлове са напълно защитени от тестваната програма.

Самите домени се възприемат като обекти, защитени от системата, тъй като правата за достъп до домена се съхраняват от системата. Достъпността

| | Обекти | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Домен 1 | Домен 2 | Домен 3 | Процес 1 | Процес 2 | Файл 1 | Файл 2 | Опашка 1 |
| Домен 1 | Собственик* Управление | | | Изпълнение* Собственик Четене* | | Собственик* Четене* Запис* | | Собственик* Добавяне* Изтриване* |
| Домен 2 | | Собственик Управление | Собственик Управление | Изпълнение Четене | Изпълнение* Собственик Четене | Собственик Четене* | Собственик* Четене* Запис* | Добавяне Изтриване |
| Домен 3 | | | | Изпълнение Четене | | | Четене | |

* Флаг COPY е включен.

Фиг. 6.6. Указания за достъп, заредени в матрица на достъпа

до домена осигурява различни обработки с правата за достъп, които доменът има за други обекти.

За да се обясни матрицата за достъп, се избира ред, който отговаря на домен. Отделните елементи в този ред показват типа на достъпа, разрешен за домена, до обекта, оглавяващ колоната, която съдържа елемента. Например на фиг. 6.6 елементът (домен 2, процес 1) показва, че домен 2 има едновременно достъп за изпълнение и четене до процес 1. Домените са представени с редове в матрицата за достъп, обектите (домен, процес, файл, опашка) — с колони, а правата за достъп (четене, запис, изпълнение) — с означения в матрицата за достъп. Собственическият и управляващият достъп означават, че доменът 2 може да добави права за достъп към означените и да измени или изтрие правата за достъп, намиращи се текущо в означението. Флагът за копиране, свързан с всяко право за достъп в матрицата, показва, че доменът, който има точно това право за достъп, може да го прехвърли на друг домен. Обратно, право за достъп без флаг за копиране, не може да бъде прехвърлено на друг домен.

Системата не интерпретира непременно значението на правата за достъп, които се съдържат в матрицата. Системата обаче осмисля правата за достъп, които трябва да наложи, например собственик и управление. Списъкът на останалите права е оставен на програмите, които обработват обектите. Например домен има право да добавя или изтрива в опашка 1. За да добави елемент към опашката, доменът повиква програма за обработка на опашката, която проверява дали доменът има права за достъп, неообходими за операцията. Системата осигурява последователност от битове на програмата, а програмата интерпретира съдържанието на полето. Системата не трябва да прави разлика между типовете обекти в този смисъл; това може да се остави на програмите, които обработват обектите.

Системата запазва копие от матриците за достъп, тъй като матрицата е разпръсната и се загубва памет. Матрицата може да се зареди в таблица като редове с три полета — домен, обект и права за достъп. Дори това е загуба на памет, тъй като само малка част от домените се ползува едновременно и е необходима само част от таблицата, за да се определят спецификациите за достъп до тези домени. Другият вариант е защитната информация да се зарежда с домена (матрицата се зарежда ред по ред като възможности) или с обекта матрицата се зарежда колона по колона в списъци за управление на достъпа).

**Възможности**

Съхраняването на защитната информация заедно с домена означава, че всеки домен има списък на обектите, до които има достъп, и правата, които има за всеки обект. При някои системи простото познаване на името или на физическото разположение на обекта разрешава на домена достъп до обекта; режимите за достъп до обекта са неограничени. Тъй като тези системи не осигуряват достатъчно ниво на сигурност, обсъждането тук се съсредоточава върху идеята за възможността, която е с по-голяма насоченост към защитата.

Възможността съдържа правата за достъп до предмета и обръщение към него и е подобна по структура на дескриптора. Разликата между възможността и дескриптора е, че дескрипторът свързва предмета с физическото му разположение, докато възможността го свързва с името му. В система с възможност операционната система поддържа таблица с имената на обектите и физическото им разположение. Тъй като възможностите се обръщат към обектите по име, операционната система може да променя свободно физическото разположение на обектите при условие, че таблицата се поддържа правилно.

За да може да ползува даден обект, потребителят получава възможност за обекта. Потребителят съхранява възможността там, където желае, и просто представя възможността на операционната система заедно със заявка за достъп до обекта. Системата проверява правата за достъп, за да е сигурна, че заявката за достъп е разрешена от възможността, и след това ползува името на обекта, за да получи физическия адрес на обекта от таблицата на паметта.

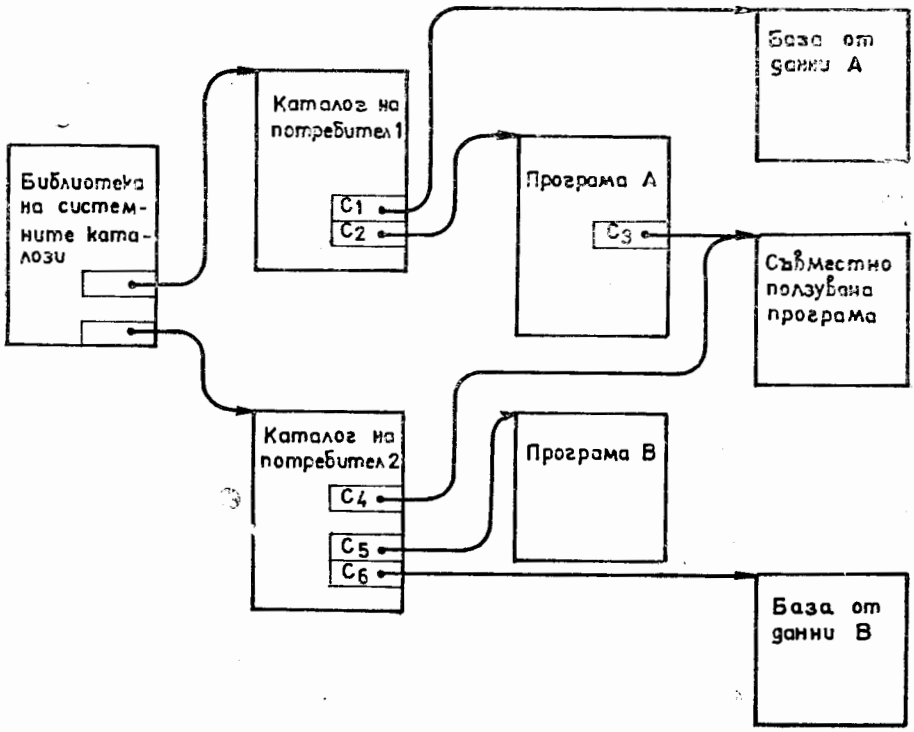
Възможностите трябва да се защищават точно както дескрипторите, така че потребителите да не могат да изменят нелегално спецификациите за достъп, представяни от възможностите. Тази защита може да се осигури от техническите средства (с белязана архитектура) или от програмните средства (операционната система отказва на потребителя пряк достъп до паметта, която съдържа възможностите).

При белязаната архитектура към всяка дума в паметта е прикрепен допълнителен бит, но битът не е достъпен за потребителя. Ако битът, свързан с дадена дума, е нула, тогава тя се счита за обикновена инструкция или дума от данните. Ако обаче битът е единица, счита се, че думата съдържа възможност и не може да се обработва от процес с обикновени инструкции. Когато процесът желае да има достъп до сегмент той повиква супервайзора и му предава адреса на възможността за сегмента, заедно с адрес в сегмента и искания тип достъп. Супервайзорът проверява дали битът е единица, а думата възможност и след това ползува съдържанието на възможността, за да получи достъп до сегмента. Когато системата поставя възможностите в паметта на потребителя, операцията установява автоматично бит в единица, за да покаже, че думата съдържа възможност. Потребителят не трябва да може да обработва възможността. Това може да се постигне или като се спре изпълнението на процеса, когато към белязана като възможност дума се подходи неправилно, или като се изключи маркиращият бит, когато други инструкции за паметта записват в думата. Следователно, от една страна, инструкциите, които имат достъп до възможностите, не могат да се използват за обикновени данни или думи на инструкции, а от друга страна, инструкциите, които се ползуват за данни или думи на инструкции, не могат да се ползуват за думи на възможностите.

Друг метод за защита на възможностите е да се разбие всеки сегмент на две части — едната за данни и (или) инструкции и другата за възможности. Тогава всяко обръщение към възможностите ползува частта за възможност на сегмента, а обръщенията към данни или инструкции ползуват другата част.

Независимо дали възможностите са защитени от белязана архитектура или са в защитена част от сегмента, потребителят контролира клетката с възможността, въпреки че не може да задава съдържанието ѝ.

На фиг. 6.7 е изобразена проста система от възможности. Предполага се,




Фиг. 6.7. Съвместно ползуване въз основа на възможности

че потребител 1 влиза в системата, може би от терминал, и самоличността му е установена. Системата намира каталога (сегмент който съдържа списък на възможностите), който принадлежи на потребител 1, като ползува възможността, получена от каталога на системата при установяване на самоличността, и *следователно има списък на сегментите, достъпни за потребител 1*. След това потребител 1 изисква от системата да изпълни програма А. В тази точка процесорът има възможност за програма А и за каталога на потребител 1. Програма А ползува възможност $C_1$ в католога, за да получи достъп до базата от данни А. Той може да ползува възможността $C_3$, която се намира в програма А, за да получи достъп до съвместно ползуваната програма. Когато потребител 2 работи и тази програма е на негово разположение, тя изпълнява програма В и също може да ползува съвместно ползуваната програма. Потребител 2 решава да постави възможността за съвместно ползуваната програма в своя каталог, а не в прогрограма В, което показва, че мястото на възможността зависи от предпочитанието на потребителя.

## Списъци за контролиране на достъпа

При система, която се основава на списъци за контролиране на достъпа, защитната информация се съхранява в самия обект. В основни линии списъкът на упълномощените потребители и правата на всеки потребител се свързват с всеки обект. Винаги когато към обекта се осъществи достъп, списъкът се проверява от системата, за да се установи дали потребителят е упълномощен да ползува обекта и дали има нужните права за изпълнение на заявената операция.

Необходимо е потребителите да знаят само името или някоя друга форма на указател за обекта, за да се опитат да стигнат до него. Тъй като сигурността се налага от системата (като се ползува списъкът за контролиране на достъпа), тези указатели могат да се фалшифицират свободно от потребителите. Дори ако потребителят изработи указател за обект, за който няма разрешен достъп, системата предотвратява достъпа.

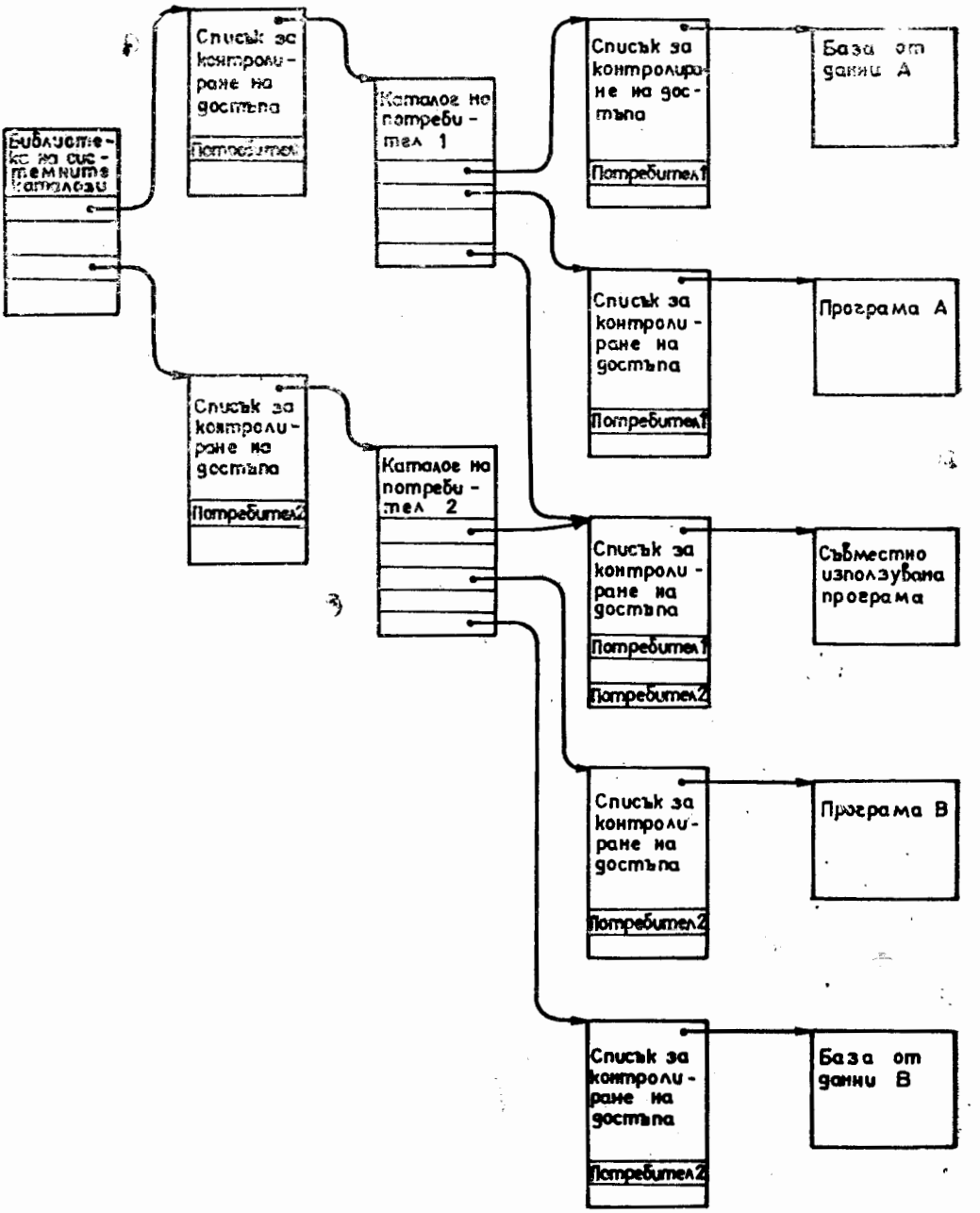
За да се улесни изменението на списъците за контролиране на достъпа, обикновено списъкът се поставя сам в сегмент заедно с дескриптор на защищаващия от него обект. Имената, с които обикновено потребителите получават достъп, насочват операционната система към сегмента, съдържащ списъка за контролиране на достъпа, тогава за достъп до обекта се използува дескрипторът, който се съдържа в този сегмент. Следователно списъкът за контролиране на достъпа се ползува като непряко обръщение към предмета.

Една допълнителна информация трябва да бъде защитена по такъв начин, че списъците за контролиране на достъпа да действуват така, че да се идентифицира потребителят. Когато потребителят влезе в системата и се установи самоличността му, идентификацията му трябва да се съхрани грижливо с процеса и да се провери в списъка за контролиране на достъпа, когато се прави опит за достъп до обекта. Ако потребителят може да променя идентификатора след установяването на самоличността му, системата не ще може да приложи правилно спецификациите за достъп. Следователно потребителската идентификация трябва да се съхранява в недостъпна област.

На фиг. 6.8 е изобразена същата организация на потребител и обект както на фиг. 6.7, разликата е само в ползуването на списъци, а не на възможността. На фиг. 6.8 потребител 1 изпълнява програма А и прави опит да чете от базата от данни. Указателят към списъка за контролиране на достъпа за базата от данни А, който се съдържа в каталога на потребител 1, се представя на системата заедно със заявката за четене от базата от данни. Системата ползува указател, за да открие списъка за контролиране на достъпа до базата от данни и получава правата за достъп на потребител 1 от списъка. След проверка за установяване, че правата за достъп отговарят на операцията, системата проверява дескриптора на обекта, който се съдържа в списъка, за да осъществи заявката за четене в базата от данни. В схема за съвместно ползуване списъкът за контролиране на достъпа за съвместно ползуваната програма трябва да съдържа идентификаторите за всички, които изпълняват съвместно ползуваната програма.

Ако списъците за реализиране на достъпа се реализират, както е показано тук, разходите са много високи поради претърсването на списъка за всеки достъп до обект. Един от начините да се снижат разходите е списъкът да се проверява за контролиране на достъпа само при първия достъп до сегмента. Ако е разрешен достъп, на процеса се дава някакъв временен указател. Този указател има пряк достъп до обекта, без да се проверява списъкът за контролиране на достъпа, но процесът не може да запазва указателя между повикванията.

Временен указател може да бъде дескриптор в списъка за контролиране на достъпа или възможност; прост указател не е подходящ, тъй като правата за достъп трябва да се съхраняват и проверяват при всеки достъп. Резултатът от ползуването на временен указател е смесена система, с мощността на списъците за контролиране на достъпа и бързодействието на възможностите. Някои дискови файлови системи са примери за прилагането на този метод. Списъкът




Фиг. 6.8. Съвместно ползуване със списъци за контролиране на достъпа

за контролиране на достъпа се проверява, когато дисковият файл се отваря, но не и за следващите операции с файла. Начинът за достъп се запазва и проверява за всяка операция; ако например файлът е отворен само за четене, опитът да се записва във файла води до съобщение за грешка от файловата система

Като разширение на системите със списъци за контролиране на достъпа е полезно няколко потребители да се поставят в група и да се осигурят еднакви права за достъп на всеки член от групата. Потребителите може да се групират като се прибавят един или повече идентификатори към полето за идентифициране на потребителите едновременно в списъка за контролиране на достъпа и в идентификатора на потребителя, свързан с всеки процес. Когато се проверява списъкът за контролиране на достъпа, системата търси или съответствие на идентификатор на потребителя, или съответствие на групов идентификатор. При поредицата за установяване на самоличността системата трябва да свърже потребителя с дадена група, може би посредством формата на списък за контролиране на достъпа.

Контролирането на достъпа с ползуване на възможности е аналогично на раздаването на ключове за дадена врата, като всеки, който има ключ, може да я отключва, докато ползуването на списъци е аналогично на поставянето на пазач до вратата, държащ списък на всички, на които е разрешено да влизат през нея.

### Изменение на спецификациите за достъп

Дотук се описват статични спецификации за упълномощаване. В повечето случаи обаче матрицата за достъп на системата е динамична, като постоянно се добавят, изменят и унищожават въвеждания. Следователно матрицата трябва да може да се изменя. Докато възможностите са по-ефикасни през време на изпълнение, списъците печелят, когато спецификациите за упълномощаване трябва да се променят.

Обсъждането на спецификациите за достъп обхваща три теми:

- Как се получава упълномощаване за съвместно ползуване?
- Кой може да изменя упълномощаването за съвместно ползуване?
- Как да се отменя упълномощаването за съвместно ползуване?

Списъците за контролиране на достъпа и възможностите се разглеждат във всяка тема и се сравняват за удобство при ползуването им.

Ето пример как може да се даде упълномощаване за съвместно ползуване. Потребител А желае да позволи на потребител В да изпълни програма, която е притежание на потребител А. В система, която ползува списъци за контролиране на достъпа, потребител А просто „казва“ на системата да добави потребител В към списъка за контролиране на достъпа за програмата и да ограничи начина за достъп само за четене-изпълнение. Потребител А може да получи идентификатора на потребител В, а потребител В— името на обекта.

Проблемът за разрешаване на достъпа е много по-сложен в система, основаваща се на възможности. Ако потребител А може да запише възможността в каталога на потребител В, потребител А може да записва и върху всяка възможност в каталога. Обратно, ако потребител В може да прочете възможност от католога на потребител А, потребител В също може да прочете останалите възможности в каталога. Може да се организира операция от типа пощенска кутия за за защита на двама потребители, участвуващи в обмена. Тази операция се състои от три основни стъпки:

- Потребителите А и В си разменят идентификатори по някакъв път, който е извън системата.

● Потребител А дава идентификатора на потребител В на задание, което се изпълнява с идентификатора на потребител А. Заданието прехвърля възможността на потребител В, като я поставя в пощенската кутия, определена за потребител В.

● Потребител В дава идентификатора на потребител А на задание, което след това търси в пощенската кутия възможността, означена като идваща от потребител А.

Двама потребители организират пощенска кутия, като ползуват съвместно един сегмент само с цел да предават съобщения и възможности. При много потребители това не е практично, тъй като са необходими много сегменти; затова за пощенска кутия може да служи системата, а не потребителите. За всеки потребител се отделя сегмент и съобщенията за потребителя се поставят в пощенската кутия. Системата прибавя към съобщението идентификатора на изпращача така, че получателят може да провери дали възможността, получена от сегмента-пощенска кутия, идва от правилния потребител.

Сега възниква въпросът, кой е упълномощен (или способен) да дава достъп или да изменя текущите права за достъп до обекта. Всеки, който има възможност за обект, може да я предаде на друг потребител. Възможно изключение е, ако способността да се прехвърля възможността на друг потребител е право (право на копиране), което се съдържа във възможността. Ако собственикът на програма даде възможността, без да вдигне флага за копиране, той знае, че възможността няма да се предаде на неупълномощен потребител. Ако системата не поддържа флаг за копиране, винаги когато собственикът на програми дава възможност за програмата на друг потребител, собственикът губи контрола върху достъпа до програмата.

При система, която ползува списъци за контрол на достъпа, са възможни самоконтролиране и йерархично контролиране на списъците за контролиране на достъпа. Даването на достъп или изменението на текущите права за достъп до обекта се извършват, като се изменят списъците за контролиране на достъпа; правото да се прави промяната трябва да бъде съхранено в някой списък за контролиране на достъпа. Ако това право се съхранява в самия списък за контролиране на достъпа, образува се самоконтролираща се система. Потребителят създава обект заедно със съответния списък за контролиране на достъпа и може да даде само на себе си способността да изменя списъка. В системата с йерархичен контрол правото за промяна на списъка за контролиране на достъпа се съхранява в отделен списък за контролиране на достъпа. При една форма на йерархичен контрол списъкът за контролиране на достъпа до даден обект се съхранява в потребителски каталог (също обект) и способността за изменение на каталога означава и способност да се изменя списъкът за контролиране на достъпа. Резултатът е дървовидна структура от каталози.

Динамичното изменение на спецификациите за достъп изисква също способността да се отказва достъп до обекта. При списък за контролиране на достъпа всеки, който може да изменя списъка, просто отстранява идентификатора на потребителя от списъка, който блокира по-нататъшен достъп. Отказването на достъп в система с възможности е много по-сложно. Потребителят няма достъп до вече дадена възможност и може да не знае колко копия са направени от нея. Следователно схемите, основаващи се на възможности, обикновено не допускат отказване на достъп.

Достъп в система, която се основава на възможности, може да се откаже, ако се разрешат непреки възможности. При това потребителят не дава възможност за обекта на други потребители; вместо това се дава възможност на възможността за обекта. Системата трябва да се проектира така, че да осигурява

непреки обръщения, необходими за работата ѝ. Началният потребител запазва контрола върху единствената възможност към обекта и може да откаже всеки достъп до него, като я унищожи. Разбира се, той първо прави копие от възможността, така че да не се загуби обектът. Тази схема може да се обобщи така, че потребителят да притежава списък за достъп на възможностите; това води до смесена система, която клони към възможностите.

### Подотчетност на ползването на информация

Подотчетност е установяването кой може да ползва дадена информация в системата. Подотчетност се изисква, за да може да се прегледа списъкът на потребителите с право на достъп до даден обект, да се направи списък на онези, които ползват даден обект, или да се установи, че никой няма достъп до обекта и следователно обектът може да отпадне от системата.

При системи със списък на контролиране на достъпа има списък на потребителите с текущ достъп до даден сегмент, но има и други потребители, които заслужават да бъдат разгледани. Ако списъците за достъп образуват йерархично дърво (не форма на самоконтрол), всеки потребител, чийто каталог се намира по-високо в дървото, но все пак в същия клон, може да наложи достъп до сегмента. Необходимо е търсене по дървото (връщане назад от списъка за контролиране на достъпа за сегмента), за да се преброят тези потребители.

Подотчетността е труден проблем при система, основаваща се на възможности, тъй като потребителят може да копира много пъти възможността и да разпръсне копията из цялата памет. Подотчетността изисква всички тези копия да могат да се откриват. Единственият метод да се откриват всички случаи на възможност за даден сегмент е да се претърсва цялата памет, а това е сериозно нарушение на неприкосновеността на потребителите на системата. Дори ако се открият всички копия на възможността, проблемът все още не е решен. Потребителят може да има възможност да прочете сегмент, който съдържа едно от копията и следователно може да е способен да го ползува. Може да има извънредно много потребители, които могат да стигнат до някое от копията по обиколен път. Търсенето на всички потребители, които могат да стигнат до даден обект, е изключително скъпо. Следователно подотчетността е много по-голяма при системи, основаващи се на списъци за контрол на достъпа, поради простите процедури на вписване в списъка, и малка — при системи, основаващи се на възможности, поради разходите и потенциалното нарушаване на сигурността при търсенето на всички възможности за даден обект.

Концептуалното разглеждане на механизмите за съвместно ползуване и упълномощаване дава два основни противоположни възгледа. Възможностите са по-ефективни при изпълнение, но се контролират много по-трудно. Списъците работят много по-бавно, но отказването, подотчетността и ползуването се улесняват. Смесени системи като списъци за контролиране на достъпа с подчинена система с възможности, дескриптори или непреки възможности могат да съчетаят предимствата и на двете системи.

### 6.4.3. ЗАЩИТЕНИ ПОДСИСТЕМИ

В някои случаи схемите за съвместно ползуване и защита, осигурени от операционната система, може да не са подходящи и потребителят да иска да има по-сложна схема за защита. В система с база от данни например достъпът до дадени записи в данните или дори до дадени полета в записа е ограничен.

Един от подходите е да се създаде програма, която приема и интерпретира заявките на потребителя, утвърждава и (въз основа на някаква спецификация за защита) чете в базата от данни, а и връща само исканата информация. Програмата заедно с базата от данни образува т. нар. защитена подсистема. Това означава, че базата от данни е единствено достъпна за програмата и че самата програма е защитена.

Със списък за контролиране на достъпа или методите на възможностите не може да се получи достъп до програмата, но може да се получи достъп до файловете, необходими за програмата. При системи със списък за контролиране на достъпа например потребителят може да получи точно определен достъп до базата от данни в списъка за контролиране на достъпа за базата от данни, иначе достъп не е разрешен. В системи въз основа на възможности потребителят няма възможност за базата от данни, така че възможността е поставена в програма, която чете в базата от данни. Дори това не защищава достатъчно възможността, защото потребителят трябва да има достъп за четене и изпълнение до програмата, преди самата програма да може да ползува възможността. Ако потребителят обаче има достъп за четене и изпълнение, възможността може да се придобие и базата от данни да се прочете непосредствено.

За да се създаде полезна защитена подсистема, системата трябва да осигурява известно увеличение на мощностите, когато се повика подсистемата. С други думи, трябва да се промени средата. Пример е супервайзорът на повечето системи. Когато потребителят повика супервайзора, системата минава в режим на супервайзор и за процесора се осигуряват допълнителни инструкции и памет. Затова един начин да се осигуряват защитени подсистеми е да се разширяват битовете за потребителския и супервайзорния режим до няколко отделни състояния, а не само до две. Това е реализирано в системата Multics, където са осигурени осем нива.

### 6.4.4. РАЗПРОСТРАНЕНИ ОПЕРАЦИОННИ СИСТЕМИ

Тук са дадени примери за операционни системи. Първата причина са методите за защита и съвместно ползуване, а втората—анализирането на самите операционни системи.

Разгледани са три операционни системи-OS/360, Multics и HYDRA. Системата OS/360 дава ниско ниво на сигурност; Multics използува списъци за контролиране на достъпа, а HYDRA се основава на възможности. При Multics и HYDRA сигурността се увеличава значително в сравнение с OS/360.

#### OS/360

Операционната система OS/360 е широко разпространена. Тя причина на много търговски системи не толкова по реализацията, колкото по осигуреното ниво на сигурност.

Системата на паметта се разделя на блокове; с всеки блок се свързва 4-битов образец за блокировка. Думата за състоянието на програмата съдържа съответствуващ ключ от четири бита; разрешава се запис в паметта само ако ключът в думата за състоянието на програмата съвпада с блокировката на блока от паметта, към който се извършва достъп, или ако ключът е нула. В новите варианти към блокировката е прибавен бит, за да има защита от четене и запис; в резултат на това паметта се разделя на отделни области. Системата има бит

потребител/супервайзор; системата от инструкции, с която разполага потребителят, е ограничена и не обхваща нито инструкции за въвеждане и извеждане, нито инструкции, които променят бита потребител/супервайзор или блокировките и ключовете за паметта. Въпреки че системата съдържа достатъчно механизми, за да осигурява изолационна форма на защита, пропуските в проекта заедно с грешките при реализацията са довели до несигурна система. Вместо да се обсъждат формите на защита, които системата може да осигури, ако е била реализирана надеждно, тук се отбелязват някои пропуски, които я правят несигурна.

Един от основните пропуски се проявява в именуването на файловете, програмите и съглашенията за достъп. Ако потребителят налучка името на файла, той може да получи достъп до него. Вторият и по-деликатен проблем е начинът, по който системата търси програма, към която има обръщение. В някои случаи системата очаква дадена програма да бъде заредена и ползувана; ако обаче потребителят създаде програма със същото име както името на системната програма, тогава ще бъде повикана потребителската, а не системната програма. Тъй като програмата се повиква от операционната система, потребителската програма има контрол върху процесора и си присвоява режима на супервайзора. Разновидност на тази атака е довела до успешното проникване в системата Multics, а също и в OS/360.

Вторият основен пропуск в проекта на системата засяга супервайзора, който зарежда в паметта информация, отнасяща се за сигурността, достъпна за потребителя. Програмистът може да се възползува от този факт по няколко начина, за да компрометира системата.

В един случай на проникване потребителят има пряк достъп до паметта непосредствено преди повикването на супервайзора за отваряне на файл. Отначало супервайзорът проверява дали потребителят може да отвори файла и след това прилага информация, заредена в пространството на потребителя, за да извърши отварянето. Ако синхронизирането е правилно, с прекия достъп до паметта може да се осигури да се записва върху информацията след извършване на проверката за сигурност, но преди системата да стигне до информацията, за да изпълни отварянето. Обикновено отварянето на файл изисква файл, нормално достъпен за потребителя, докато прекият достъп до паметта заменя заявката за файл, който се предполага, че е защитен от потребителя.

В друг случай на проникване адресът на системната програма се съхранява в пространството на потребителя. Потребителят зарежда свой адрес върху адреса на системната програма и вероятно ще бъде повикан от супервайзора в режим на супервайзор. Този маршрут е бил блокиран с подходящо изменение на системата, но с леко изменен метод отново е осъществено проникване в системата. Направено е второ изменение за блокиране на втория маршрут; отново малка промяна е осигурила нов път за проникване. Тук системният програмист се е предал. Това е пояснение как понякога е невъзможно да се въведе допълнително сигурност в система, която не е проектирана като сигурна.

## Multics

Системата Multics е първостепенен кандидат за изследване от интересуващите се от сигурност. Тук системата не се разглежда напълно. Вместо това се представя основната система—отначало с кратък преглед и след това с по-подробен анализ.

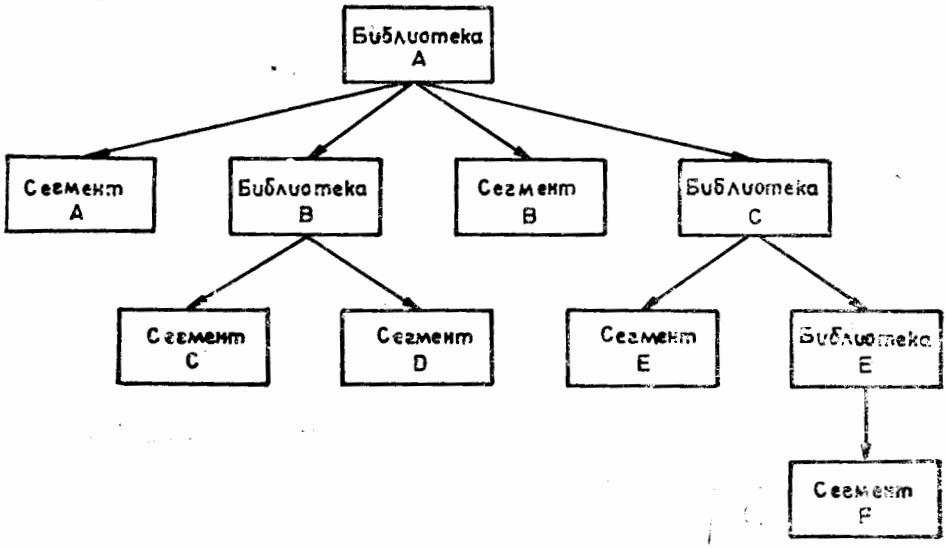
Системата осигурява съвместно ползуване и сигурност на обектите чрез списъци за контролиране на достъпа при взаимодействието с потребителя.

Самоличността на потребителя на системата се установява с пароли. Контролът върху списъците за контролиране на достъпа е йерархичен, имитирайки структурата на повечето търговски организации. Потребителите са снабдени със сегментирана виртуална памет, докато методи за разделяне на страници нанасят сегментите във виртуалната памет на потребителя през време на изпълнението. Спецификациите за сигурност, посочени от списъка за контролиране на достъпа, се изпълняват при работа от дескрипторна структура.

### Спецификации за достъп

Съхраняват се в списъци за контролиране на достъпа. Тези списъци се пазят в каталози, които също са обекти, защитени от системата. Тук се набляга само върху защитата на сегменти и каталози, независимо че системата поддържа и други обекти. Другите типове обекти получават защита от същите основни механизми като сегментите и каталозите, като основната разлика е в това, че действията, които могат да се извършат върху обектите, може да не са същите.

Всеки обект трябва да се запише в каталога. Вписването в каталога включва името на обекта и списъка му за контролиране на достъпа. Тъй като всеки каталог трябва да бъде описан в някой друг каталог, образува се йерархична




Фиг. 6.9. Йерархична структура от библиотеки в Multics

структура на спецификациите за достъп. Прост пример за такава система е даден на фиг. 6.9. Клоните на дървото са изолирани, но в действителност съвместното ползуване на файловете се контролира от списъците за контролиране на достъпа, а не от йерархичната структура на спецификациите за достъп. Списъкът за контролиране на достъпа (съдържащ се в каталог D) за сегмент E например може да съдържа вписване, което позволява на всеки, допуснат в системата, да ползува сегмент E.

За да се ползуват правилно списъците за контролиране на достъпа, системата свързва някакъв идентификатор с всеки процес. Потребителят, който носи отговорността за даден процес, се нарича ръководител, а процесът получава

име (то не може да се фалшифицира), означено като идентификатор на ръководителя.

Индентификаторът на ръководителя включва три части. Всяка част отговаря на различно разделяне на потребителите на системата. Първата част дава името на лицето, втората задава името на проекта, а третата определя името на група. Първият идентификатор разделя потребителите на отделни личности. Индентификаторът на проекта разрешава работата на групи от потребители, които се занимават с една и съща задача. Идентификаторът на проекта доставя третата форма на именувана група, която служи за такива цели като защитаване на файлове от ненастроени или съмнителни програми.

Списъците за контролиране на достъпа действуват така, както в предните описания, с две изключения. Първо, ако идентификаторът на процес, който се опитва да получи достъп до сегмент, се съдържа в списъка за контролиране на достъпа до сегмента, сегментът се нанася във виртуалната памет на процеса. Второ, използува се идентификатор от три части.

За да се пояснят трите части на идентификатора на потребителя Смит, който работи върху проекта Счетоводство и който използува група Z, дава се идентификатор на ръководител

SMITH. ACCOUNTING.Z

Идентификаторът от три части, който се сравнява с елементите на списъка за контролиране на достъпа, определя дали на процеса се разрешава да използува сегмента.

Списъците за контролиране на достъпа също включват подробни идентификатори от три части, но за да се разрешават по-обобщени форми на съвместно ползуване, всяка от трите части на даден идентификатор може да съдържа указател „без значение". Ако списъкът за контролиране на достъпа е съдържал

SMITH. ACCOUNTING.Z

на процеса със същото име се дава достъп до сегмента. Вписването в списъка за контролиране на достъп като

*.ACCOUNTING.*

разрешава достъп на всеки, който работи върху счетоводния проект, независимо в коя група се намира. Ако Смит е създал сегмент, към който иска да има достъп от няколко групи и (или) проекти, той ползува

SMITH.*.*

като вписване в списъка за контролиране на достъпа. Употребата на указателя „без значение" ползува групиранията, осигурени от идентификатор от три части. Ако например идентификаторът на хората и групите са отбелязани с „без значение", а проектът е указан за вписване в списък за контролиране на достъпа, всеки член на този проектантски екип може да ползува обекта, защитен от този списък.

Начинът за достъп, разрешен на потребителя, е свързан с всеки идентификатор в списъка за достъп. Възможните начини за достъп до сегмента са четене, запис и изпълнение. Съчетанията на начините за достъп до сегмента, които се разрешени, са: никакви, четене само на данни, четене и изпълнение за „чиста" процедура („чиста" означава, че процедурата не изменя себе си), четене и и запис на данни, които могат да се записват, и четене—изпълнение—запис за „нечиста" процедура. Каталозите могат да се поставят в списъка; съществуващото вписване може да се изменя или изтрива; могат да се добавят нови вписвания.

Начинът, по който се претърсват списъците за контролиране на достъпа, съществува като средствоза даване на отделен член на проект или група различни права за достъп до останалата част от проекта или групата, тъй като търсенето в списъка за достъп се прекратява при първото съвпадане на иден-

тификатори. Ако вписванията в списъка за контролиране на достъпа за лицето се поставят преди вписването за проекта, правата за достъп се вземат от вписването в списъка за контролиране на достъпа на лицето, а не от вписването за проекта. Например Смит и Браун могат да бъдат единствените членове на проекта Ведомости, на които е разрешено да променят файла Заплати, докато на останалите потребители на проекта е разрешено само да го четат. Необходимият списък за достъп е:

SMITH.PAYROLL. * четене—запис
BROWN.PAYROLL.* четене—запис
*.PAYROLL.* четене

Ако Смит и Браун са външни лица, а не привилегировани потребители в проекта Ведомости, техният начин на достъп може да бъде „никакъв", като по този начин им се отказва всякакъв достъп до файла Заплати. Списъците за достъп отначало са сортирани по първото поле, така че указателите „без значение" да бъдат на края на списъка. След това всяка подгрупа се сортира по същия начин по второто поле и т. н. Резултатът е, че по-специфичните елементи в списъка за достъп са най-напред и се проверяват първи, когато се прави опит за достъп.

Изменянето на списъците за контролиране на достъпа се защищава от същия механизъм, който защищава другите обекти в системата. Тъй като списъците за контролиране на достъпа се съхраняват в каталози, възможността да се добавят, изтриват или изменят вписвания в каталога позволява да се изменят спецификациите за достъп, които се съдържат в списъците за контролиране на достъпа в каталога. Правото за изменяне на каталога се съдържа в списъка за контролиране на достъпа за самия каталог. На фиг. 6.9 възможността да се изменя (или дори използва) каталог С се контролира от списъка за контролиране на достъпа в каталог А; тази зависимост създава йерархичната структура от каталози.

Човекът, който може да изменя каталога най-високо в йерархията, има значителни възможности. Първо, той може да получи достъп до всеки обект от системата. Второ, той или други потребители (обикновено системни администратори), които могат да изменят каталога на върха на йерархията, могат да създадат каталози на следващото по-ниско ниво и да диктуват правата за достъп на онези, които ползуват каталога. Това заедно с факта, че всеки каталог може да съдържа начален списък за контролиране на достъпа, който се дава автоматично на всеки обект при началното му въвеждане в каталога, позволява на системния администратор да постави под много строг контрол потребителите на системата. Например администраторът може да създава каталог и да дава на бъдещия потребител на каталога разрешение да добавя вписвания в него. Началният списък за контролиране на достъпа в каталога се ползува само за новите вписвания, като потребителят не може да контролира от кого да бъдат ползувани обектите, които той добавя в каталога. Следователно администраторът контролира списъците за достъп за всеки обект в системата, като гарантира, че нито един член на даден проект няма да ползува съвместно информация с някой извън проекта. Поради йерархичния характер на каталозите всеки, който има пълен достъп до даден каталог, може да създава структура под него и има същото влияние върху нея, каквото има системният администратор върху цялата система.

За да може системата да ползува правилно списъците за контролиране на достъпа, важно е всеки потребител да бъде снабден с правилния основен идентификатор, когато се опитва да влезе в системата. В противен случай потребителят може да се представи за системен администратор (или по-малко амбициозното — за който и да е друг потребител) и да получава достъп до обекта, за който не е упълномощен.

**Установяване на самоличността на потребителя.** Системата ползува схема с пароли за установяване на самоличността на всеки потребител. Всеки потребител в системата Multics се регистрира под уникално име. Обикновено то се състои от фамилното му име и няколко инициала. Винаги когато потребителят се опитва да влезе в системата, се изисква парола. Ако не може да даде правилна парола, не му се разрешава да ползува системата.

Всяко задание в пакетен или в диалогов режим може да се стартира само от потребител с установена самоличност. Задания в пакетен режим, след като веднъж са прочетени от четеца на перфокарти, се задържат в системата, докато се поиска изпълнение посредством процес с установена автентичност. Всяко така започнато пакетно задание след това става задание с установена автентичност и може също да стартира задания. Може да се образуват вериги от задания, но веригата трябва да се стартира от потребител, чиято самоличност е установена от системата. За да направи работата на пакетни задания по-лесна, потребителят може да поиска изпълнението на задание, преди да бъде прочетена колодата от перфокарти, която е заданието, като системата задържа заявката само докато се четат картите.

Паролите са защитени, когато се използват и съхраняват от системата. Когато потребителят се включи, веднага го запитват за паролата му. Преди той да въведе паролата си на терминала, или терминалът се изключва, или зоната, на която той ще напечата паролата си, се затъмнява, като на всяка позиция се извеждат по няколко букви. Това действие позволява да се предотврати случайното разкриване на паролата. След като се установи автентичността на потребителя, той може да сменя паролата си толкова пъти, колкото поиска. Една от наличните програми следва 8-буквени пароли с англоезични характеристики. Тези пароли са лесни за запомняне и произнасяне и намаляват вероятността да бъдат записани и компрометирани. Паролите се шифрират, като се използува еднопосочна схема за шифриране преди запомнянето им в системата. Следователно дори системният администратор не може да открие паролата на потребителя.

Системата осигурява няколко метода за откриване на опити за проникване чрез налучкване на паролата. При всеки опит за влизане в системата с неправилна парола въвеждането се поставя в дневник за проследяване на грешките. На потребителя се разрешават до десет опита за влизане, след което комуникационната връзка се прекъсва от системата; това помага да се предотвратят автоматизирани опити за проникване.

Когато потребителят се включи, напечатват се местоположението и времето на последното му ползуване на системата, така че потребителят може да открие проникване, ако някой преди това се е включил в системата с неговата парола, без да е бил упълномощен за това.

Системата е проектирана така, че да осигурява допълнителна процедура за установяване на самоличността (извън установяването на автентичност посредством парола — това се осигурява от системата), която се прилага към потребители при даден проект. Администраторът на проекта установява процедура, която се изпълнява винаги когато потребител се регистрира под името на проекта. Тази процедура изисква допълнително установяване на автентичността и започва да води дневник, ако то не се извърши правилно. Например процедурата може да провери списък на потребители, които се включват в проекта, и да отхвърли потребители, които не участвуват в списъка. Администраторът на проекта може да разреши на потребители без установена автентичност да се включат под името на проекта. Това дава възможност на „анонимни“ потребители да ползуват системата, без да са регистрирани в нея.

Докато спецификациите за достъп се съхраняват в списъци за контролиране на достъпа, те се изпълняват при работа от дескрипторен механизъм.

**Налагане на време за изпълнение на спецификациите за сигурност.** При изпълнение защитата се осигурява от:

дескрипторен механизъм;

разширение на бита потребител/супервайзор, използуван при OS/360;

ограничение на възможностите за достъп на супервайзора;

асинхронни входно-изходни буфери;

защита на остатъците в паметта (данни, останали в сегмента, след като потребителят върне сегмента в системата).

При изпълнение на всеки процес в системата се дава дескрипторен сегмент, съдържащ набор от дескриптори, който служи, за да наложи спецификациите за достъп на системата. На един процес се дава дескриптор за сегмент само ако списъкът за контролиране на достъпа за сегмента съдържа основния идентификатор на процеса. Защитната информация, която се съдържа в списъка за контролиране на достъпа, се използува за създаване на дескриптор. Набор от индикатори гарантира, че промените в списъка за контролиране на достъпа ще бъдат отразени незабавно във всички дескриптори за този сегмент. Всеки достъп до сегмент в процеса се превръща в номер на сегмент (който се ползува като индекс в дескрипторния сегмент) и номер на думата в този сегмент. Следователно за всеки достъп до паметта трябва да се ползува наложения от апаратните средства дескрипторен защитен механизъм. Тъй като процесът има свой собствен дескрипторен сегмент и място за обръщане, той може да има достъп само до сегментите, за които му е дадено разрешение. Ако сегментът се ползува съвместно от два или повече процеси, всеки процес има собствен дескриптор за сегмента и може да има различни права за достъп.

Супервайзорът е запазен от потребителите посредством кръгова структура, разширение на бита потребител/супервайзор при OS/360, който дава няколко нива на привилегироване. При Multics индикаторът на нивото на привилегироване за сегмента се съдържа в неговите дескриптори. Има осем кръга (номера от 0 до 7), като този с най-ниския номер има най-големи привилегии. Апаратните средства разрешават на процеса да ползува само дескрипторите, които имат кръгови номера, по-големи или равни на кръга, зададен от дескриптора, ползуван за достъп до процедурата, която в момента се ползува от процеса. Супервайзорът работи в най-долните кръгове; само по-горните кръгове са достъпни за потребителите. Процесите на потребителя имат дескриптори, които му дават възможност да повика супервайзора, но само със специфицирани входни точки, които се наричат входове. Тези входове гарантират, че управлението се предава правилно на супервайзора и осигуряват единственото средство, с което процесът може да стигне до супервайзора.

Кръговата структура, която е проектирана за супервайзора, е достъпна и за процеса на потребителя. Ако например кръговете 5, 6 и 7 са достъпни за потребителя, той може да постави основната си програма в кръг 5 и да изпълнява подпрограми, в които няма доверие, в кръгове 6 или 7. Основната му програма е защитена от подпрограмите по същия начин както супервайзорът е защитен от целия процес. Следователно потребителят може да създава защитени подсистеми.

Кръговата структура в Multics затваря пътя за проникване, на който OS/360 дължи уязвимостта си. Вместо да ползува бит в паметта, за да определи дали работят супервайзорът или потребителят (както е при OS/360), при Multics привилегиите, представени на текущата операция, зависят от номера на кръга, който се съдържа в дескриптора, използуван за получаване на последната инструкция. Индикаторът на супервайзора не може да бъде

оставен включен, тъй като винаги когато управлението премине в сегмент на потребителя, номерът на кръга се сменя автоматично и за да се получат инструкции от потребителския сегмент, трябва да се разгледа различен дескриптор. Реализирането на дескриптори с базов адрес и дължина при Multics не позволява на потребителя да предава на супервайзора аргумент, който ще го накара (както е при OS/360) да направи преход към адрес извън сегмента му и следователно в сегмента на друг потребител, като системата все още ще счита, че е в сегмента на супервайзора.

Супервайзорът няма повече „власт", отколкото е необходимо за системата. Независимо че супервайзорът поддържа средствата за достъп до всички сегменти на потребителя, тъй като те са в по-горен кръг, все пак кръгът има само правата за достъп, дадени в дескриптора на системата. Следователно бази от данни могат да се направят недостъпни и записът в чистите процедури — невъзможен. Това защищава потребителя от собствените му грешки при програмирането, тъй като заявката до супервайзора за изпълнение на сегмент от базата от данни би довела до нарушаване на защитата.

Въвеждането и извеждането се извършват така, че да се избягнат конфликти между работещ процес и канални програми за пряк достъп, който може да бъде използван от нарушител. Всичките входно-изходни операции се извършват в буфер в областта на супервайзора. Когато завърши въвеждането, супервайзорът предава данните на потребителския процес. Освен това супервайзорът не приема от потребителя данни за въвеждане, докато не се запълни буферът. Когато асинхронните входно-изходни операции се реализират по този начин, те не могат да се използват за проникване в системата.

Накрая трябва да се защитава информацията, наречена остатък в паметта. Тя се съхранява в сегмент, който се връща обратно в системата. При Multics целият достъп до паметта е посредством виртуалната система на паметта. Следователно никой не може да има достъп до каквато и да е памет, която не е част от собствената му виртуална памет. Единственият начин, по който потребителят може да получи достъп до остатъка в паметтта, е да изиска нов сегмент и след това да се опита да прочете информацията, оставена в него от предишния му собственик, с надеждата, че може да получи важни данни. Системата предотвратява това действие, като връща нули при четене на виртуалната памет, която не е установена начално. Дори ако има остатъци в паметта, те остават недостъпни.

**Слабости на системата.** Произлизат от йерархичния й характер (при структурата на каталозите, а и при кръговата структура, която осигурява защитени подсистеми), нейната сложност (както в големината на програмите на операционната система, така и на потребителския интерфейс) и от защитените й комуникационни линии.

Възможностите за защита през време на изпълнение имат йерархичен характер. Изпълняваната програма винаги може да ползува всеки сегмент, който се намира в кръг със същия или по-голям номер, като се приеме, че основният идентификатор на процеса, който съдържа програмата, съществува в списъка за контролиране на достъпа до сегментите. Типът достъп може да бъде ограничен, но ограничението важи за всички програми в процеса.

Ако потребителят иска да защити предоставените на даден процес сегменти от заета (или наета) чужда програма, той трябва да изпълнява чуждата програма в кръг с по-голям номер от останалата част от процеса. Това може да не бъде приемливо за притежателя на чуждата програма, който ще предпочете програмата да бъде защитена от процеса, който я ползува. Следователно изключват се няколко типа съвместно ползуване като току-що описаните изоб-

що съмнителни подсистеми, като по този начин се ограничават формите на защита, които системата може да осигури.

Йерархичният характер на структурата от каталози също създава трудности със сигурността. Приема се, че потребителят има два каталога, като списъкът за контролиране на достъпа за втория се съдържа в първия.

В системата Multics няма списък, който да изброява всички потребители, които могат да изискват достъп до сегмент, тъй като имат възможност за изменение на каталог, независимо че такава информация е важна за потребителя.

Една от слабостите, която потребителят не забелязва лесно, е сложността на супервайзора. Системата се състои от около 2000 програмни модула, от които 300 работят в най-защитената област. Всеки от тези модули може да създава дескриптори и така има възможността да компрометира сигурността. При средно 200 реда за модул това се равнява на 60000 програмни реда. Това дава доста възможности за грешки или неправилни реализации.

Взаимодействието с потребителя е доста сложно. Когато създава сегмент, той трябва да реши кого да постави в списъка за контролиране на достъпа. В тази работа го подпомага автоматично създавания начален списък за контролиране на достъпа, но има още няколко допълнителни елементи, които трябва да бъдат взети предвид. Например възможностите за достъп до процедурен сегмент и сегмент от данни вероятно ще бъдат различни. Потребителят трябва да реши дали сегментът ще бъде защитен чрез изискването той да се изпълнява или чрез ползуването му в кръг с малък номер. Потребителят трябва да внимава много, когато отстранява име от списъка за контролиране на достъпа, тъй като потребител, чието име е отстранено, може да получи по-големи възможности от друг елемент в списъка за контролиране на достъпа. Ако например каталогът е

SMITH.ACCOUNTING *: read
*.ACCOUNTING.*read—write

отстраняването на първото вписване не предотвратява ползуването му от Смит. Вместо това той има достъп за четене и запис. Това може да бъде желаната цел, но може и да не бъде. Сложностите на потребителския интерфейс могат да накарат потребителите да направят грешки в решения, които са свързани със сигурността, или да използуват съкращения, за да направят по-лесна реализацията на системата. Ако потребителят направи въвеждане в списъка за контролиране на достъпа, което разрешава на всеки да ползува сегмента, той не трябва да се тревожи за сложностите на системата, но не ще има и защита за сегмента. Във всеки случай сложностите на системата могат да доведат непряко до пропуски в сигурността.

Комуникационните линии при Multics не са защитени. Нарушителят може да подслушва телефонна линия, да получи парола на потребител и да проникне в системата, като се представи за легален потребител. Този тип проникване може да се открие или ако се изследват дневниците за ползуване на системата, или ако потребителят открие при следващ сеанс, че времето и местоположението на последното му ползуване на паролата е ненормално. Все пак такова разкритие само предупреждава потребителя и може да се почви, след като вредата е налице.

**Заключение.** Системата Multics е проектирана и реализирана сравнително добре. Осигурен е доста широк обхват от възможности за съвместно ползуване. Разходите за ползуване на системата не са прекалено големи; реализирането на защитните механизми в апаратните средства е запазило разходите за сигурността ниски. Много от пътищата, използувани за проникване в операционната система OS/360, са отстранени. В общи линии Multics работи така, както се рекламира. Някои потребители обаче могат да изискват по-сложни

схеми за съвместно ползуване, като съвместна работа между две взаимно поддържащи се подсистеми в един процес, и следователно не могат да ползуват Multics. За съжаление никоя друга от предлаганите операционни системи не поддържа взаимно подозрителни подсистеми.

## HYDRA

Целта на системата HYDRA не е да осигурява пълни възможности на операционната система, а да даде основа, върху която да се изграждат и проучват операционни системи. Затова тя дава набор от примитивни операции (наричани ядро), с които операционната система обслужва и защищава потребителите. Тук се проучва основната система HYDRA. Интелигентният нарушител (а повечето са такива) използува всеки пропуск на сигурността в основата на HYDRA, за да проникне в операционната система (изградена на основата на HYDRA), която иначе няма пропуски в сигурността си.

Системата HYDRA е реализирана на мултиминипроцесор. Този процесор е конструиран като система от миникомпютри. Системата се състои наймного от 16 миникомпютъра PDP-11 с 32 милиона байта обща оперативна памет. Часовник осигурява общо време; миникомпютрите могат да се прекъсват помежду си. Всеки миникомпютър има собствена оперативна и външна памет както и входно-изходни устройства.

В основни линии защитата и съвместното ползуване са осъществени чрез възможности. Възможностите са вградени в програмното осигуряване; структурата на паметта и механизмите за достъп не разрешават на потребителите да обработват или фалшифицират възможностите.

**Представяне на обектите.** Както при Multics основните единици, които се обработват и защищават от HYDRA се наричат обекти и се използуват за представяне на наличните ресурси в системата. Всеки обект има име, тип, брояч на достъпите и представяне. Представянето може да се разбие на данни и списък на възможностите. Всички обекти се придържат към една и съща обща форма, но за всеки определен обект някои от основните части могат да бъдат нулеви.

Името на обекта е уникална 64-битова стойност, създавана от централния часовник на системата; така системата не може да объркa обектите.

Полето за типа на обекта поставя обекта в група с подобни нему обекти. Системата не тълкува полето за типа, с него само се групират обектите. Съдържанието на това поле всъщност е името на дадения обект, който представя цялата група.

За всеки обект се поддържа брояч на достъпа, за да се определи кога да се отстрани от системата. Броячът на достъпа регистрира броя на възможностите, които съдържат обръщения към обекта. Когато този брой стигне до нула, обектът се отстранява автоматично.

Представянето на обекта е от две части. Първи е списъкът на възможностите, който дава обръщенията към другите обекти. Списъкът на възможностите за даден обект не може да се променя по какъвто и да е начин от друг обект, независимо колко свободни са правата за достъп до него; списъкът на възможностите може да се променя само от системата HYDRA. Втората част от представянето на обекта е частта от данни, която съдържа описателна информация за обекта. Частта от данни за една процедура осигурява памет за програмата и вътрешните данни, необходими за изпълнението на процедурата.

Обектите са едната от трите части в структурата на защитния механизъм.

**Структура на защитния механизъм.** При HYDRA защитният механизъм оперира с три основни обекта — процедури, места за локални имена и обекти. Важно е да се пояснят връзките между тези обекти, за да се разбере работата на системата HYDRA.

Процедурата е равностойна на подпрограма по това, че изпълнява функция или набор от свързани функции. Процедурата заедно със списъка на параметрите също определя връзките между процедурата и останалите обекти в системата. Списъкът на възможностите на процедурата, който може да се разширява с възможности в списъка на параметрите, определя какви други обекти в системата може да ползува процедурата и какъв тип ползуване е разрешен. С изключение на правото WALK, което е режим на достъп, процедурата може да получи достъп до обектите само като ползува възможностите в списъка. Частта на данните за процедурата съхранява кода и локалните данни на процедурата.

Областта на локалните имена съдържа всички възможности, които могат да се ползуват от процедурата при дадено повикване. Области на локалните имена се създават от системата, като се комбинират възможностите в списъка на възможностите с възможностите в списъка на аргументите (наречени зависими от повикването възможности). Независимите от повикването възможности отговарят на обекти, към които процедурата може да се обръща, без разлика кой я е повикал, докато възможностите, които зависят от повикването, отговарят на обекти, които могат да са различни, в зависимост от това кой повиква процедурата. Веднъж създадена областта на локалните имена става независим обект, който се ползува от процедурата и съществува, докато процедурата върне управлението в точката на повикването. С един процес може да са свързани повече от една област на локални имена, което поддържа възможността за рекурсивна програма (т. е. процедура, която може да повиква себе си) или за съвместно ползуване на чисти процедури.

Процесът е динамичен сбор от области на локалните имена. Сборът е текущото състояние на последователна група от процедурни повиквания. Процедурата, в която е управлението на процесора, винаги ползува област на локалните имена от върха на стека, за да адресира обектите, до които процедурата има достъп. С повикването на процедурата, нейната област на локални имена се вкарва в стека на процеса и със завършването на процедурата нейната област на локалните имена отпада от върха на стека.

**Работа на защитния механизъм.** Работата на защитния механизъм тук се обсъжда с действия, които се изпълняват от системата — създаване на нови области на локалните имена и разширяване на броя на възможностите, представени на процеса чрез правото WALK. Системата осигурява много повече функции; тези две обаче са с най-голямо значение за защитните механизми при HYDRA.

Най-важната работа на защитния механизъм е смяната на областите на защита чрез създаване на нова област на локалните имена. Системата осигурява CALL (повикване) и RETURN (връщане) за осъществяване на смяната. При CALL за процедура се създава нова област на локалните имена за тази процедура и областта на локалните имена се поставя в стека на процеса. При RETURN областта на локалните имена се изтрива от стека. Тъй като процедурата ползува само най-горната област на локалните имена в стека на процеса, действията CALL и RETURN сменят областите на защитата.

Създаването на новата област на локалните имена за дадена процедура включва съчетаването както на независими от повикването възможности в списъка на възможностите на повикваната процедура, така и на зависими от повикването възможности, представени като аргументи на процедурата. Неза-

висимите от повикването възможности преминават непосредствено в областта на локалните имена; зависимите от повикването възможности се проверяват и трансформират, преди да бъдат поставени в областта на локалните имена. Списъкът на възможностите за повикваната процедура съдържа шаблон за всяка зависима от повикването възможност. Шаблонът посочва какви проверки трябва да се направят на възможността-параметър и отчасти определя формата на възможността, която ще бъде поставена в областта на локалните имена. Шаблонът съдържа типовото поле, което определя какъв тип обект трябва да е възможността. Може да има шаблон, който приема всеки тип обект. Шаблонът също съдържа поле за проверка на правата, което определя минимума права за обекта, които трябва да се съдържат във възможността-параметър. Ако полето за тип или за проверка на правата даде грешка, връща се код за грешка и повикваната процедура не се изпълнява. Ако проверките не дадат грешка, създава се нова възможност за обекта и се поставя в новата област на локалните имена. Правата за достъп в новата възможност се извеждат от полето за правата за достъп на шаблона, а не от полето за проверка на правата на шаблона или от правата за достъп на възможността-параметър. Тогава повикваната процедура може да има по-големи права за достъп до обекта, отколкото процедурата, прехвърлила като параметър възможността на обекта.

Процедурата може да има достъп до възможности от списъка на възможностите на друг обект, като ползува правото WALK. Правото WALK както всички други права, е посочено в полето на правата за достъп във възможността на обекта. При дадено право WALK за обект и цяло число може да се ползува възможността, определена от цялото число на възможностите за обекта. С правото WALK един обект може да получи достъп до обекти, които не са изброени в неговата област на локалните имена; той може да получи достъп до всеки наличен обект с еднократна или неколкократна употреба на правото WALK.

**Слабости на системата.** В системата HYDRA процесът увеличава възможностите си за достъп, като разширява полето на правата за достъп, когато възможността за даден обект се предава на процедурата и ползуването на правото WALK върху възможностите, които има процедурата. Това осигурява среда за обобщено съвместно ползуване, но големите възможности, достъпни за процедурата, създават затруднения за сигурността. Причината е в това, че този който повиква процедурата, трябва да приеме, че разширението на правата за обекта не дава на процедурата изключителен достъп до обекта, че възможностите се предават само на други упълномощени процедури, и че правото WALK не се ползува за получаване на нелегален достъп до някой обект.

Правото WALK създава друг път за проникване, като разрешава на процедурата да ползува възможности в списъка на възможностите на друга процедура. Такова ползуване изисква правото WALK за процедурата, но отново правото WALK може да се получи от нарушител чрез предаване на възможността за процедурата на друга процедура, която той е написал, и разширяване на правата за достъп така, че да включват и правото WALK. Следователно, когато се дава възможност за процедура, достъпни стават не само процедурата, а и възможностите в списъка на възможностите за процедурата.

Системата HYDRA не е проектирана да поддържа собствеността на какъвто и да е обект. След като потребителят даде възможност за обект, той губи собственическият контрол върху този обект. Това е нормално за всяка система, основаваща се на възможности. При HYDRA обаче обектът се премахва само когато броят на обръщенията към него стане равен на нула. Създателят на обекта не може да го премахне дори когато открие, че с него се злоупотребява. Разбира се, полезността на обекта може да се унищожи, като върху

него се извърши операция, която променя основните му свойства; такава операция е записването на нули върху обектния код на процедурата. Поради възможността да се разширят правата за достъп до обекта през време на процедурата повикване, създателят на обекта не може да ползува съвместно обекта и да запазва някакви специални права върху него.

**Заключение.** Системата HYDRA осигурява мощна и ефективна схема на съвместно ползуване, но само в безопасна среда. Разширяването на правата за достъп може да се ползува за предаване на възможности през процедури, като едновременно се предотвратява погрешното ползуване на възможностите от погрешно настроена програма. Разширяването на правата за достъп обаче, пречи да бъде защитен съвместно ползуван байт в застрашена среда, като разрешава на нарушителя да получи всякакви права за достъп.

Основните операции на системата HYDRA дават наистина възможност за разнообразни схеми за съвместно ползуване и защита, като позволяват да се изграждат много и различни операционни системи. Могат да се изграждат системи със списъци за контролиране на достъпа както при Multics. Сигурността на такава система зависи от възможността на операционната система да ограничава ползуването на разширени права за достъп при процедурно повикване и осигурява надеждна схема за именуване. Операционна система, която се основава на възможности, се реализира много лесно. Многостранността на системата създава трудностите при осъществяването на защитната схема. Създателят на системата трябва да ограничава ползуването на процедурното повикване, поне за разширението на правата за достъп, за да може да осигури достатъчна сигурност. Системата HYDRA дава гъвкавост, но не подпомага създателя на системата да ограничи възможностите на потребителя.

## 6.5. УСТАНОВЯВАНЕ НА САМОЛИЧНОСТТА

Установяването на самоличността (т.е. осигуряването, че потребителите са точно тези, за които те се представят) е второто средство, предоставено за създаването на сигурна компютърна програма. Ако самоличността на потребителя е установена неправилно, нарушителят може да се представи за системен администратор (или за всеки друг потребител) и да получи широки права за достъп. Обикновено самоличността се установява от операционната система. Все пак програмистът трябва да осигури някаква форма за установяване на самоличността в програмата.

### 6.5.1. УСТАНОВЯВАНЕ НА САМОЛИЧНОСТТА НА ПОТРЕБИТЕЛЯ ОТ СИСТЕМАТА

Компютърната система може да установи самоличността на потребителя по три основни метода — по местоположението на потребителя; по нещо, което потребителят притежава; по нещо, което потребителят знае.

#### Установяване на самоличността по местоположението на потребителя

При тази най-проста форма за установяване на самоличността операционната система приема, че щом потребителят работи на даден терминал, той е упълномощено лице или е член на привилегирована група с разрешен достъп

до компютъра. Операторският пулт на компютъра е пример за такова установяване на самоличността. За да се предотврати достъпът на нарушител до пулта, самият пулт е защитен физически; в случая компютърът не осигурява никаква сигурност. Нелегален достъп до пулта може да се предотвратява с пазачи и (или) заключени врати. Може да бъде заключен и самият терминал.

Този метод за установяване на самоличността има две основни слабости. Първо, трябва да се осигури физическата сигурност на цялата линия от терминала до компютъра. В противен случай нарушителят може да се включи към линията, за да записва данните или за да стимулира действителния терминал. Второ, физическата сигурност може да се окаже голямо препятствие за проникващия; тя обаче рядко защищава обекта напълно. Ключалката на терминала може да бъде счупена, лесно или трудно отключена или да бъде направен втори ключ.

### Установяване на самоличността чрез характерни черти на потребителя

Потребителят може да притежава някакъв предмет или характерна черта, които се проверяват и се установява самоличността му. Предметите са ключове и карти с магнитни ивици, които се възприемат от входното устройство. Физически характерни черти като отпечатъци от пръсти, гласови записи, подписи и снимки на ретината също могат да се използват.

В тази методика човек рискува да предаде своите характерни черти по несигурни линии. Линиите могат да бъдат подслушвани, информацията записана и записът пуснат от нарушителя, за да симулира оригиналния потребител. Загубата на ключ или карта означава, че всеки, който ги намери може да проникне в системата. Тази възможност може да се ограничи чрез допълнителен механизъм за установяване на самоличността, като например програма за паролите.

### Установяване на самоличността по познания на потребителя

Обикновено се измисля някаква форма на парола, въпреки че са възможни и по-сложни схеми. Когато потребителят влиза в допир със системата той съобщава името си и бива запитан за паролата. Ако името и паролата съответстват на онези, които се съдържат в компютърната система, автентичността е установена. За да се защити паролата, обикновено тя не се изписва върху терминала или терминалът осигурява блокирана зона за отпечатване на паролата. При по-сложни пароли на потребителя се задават по-нататъшни въпроси, като например бащиното име на леля му. Програмата може да бъде усилена, като се задават повече такива въпроси.

Установяването на автентичността чрез пароли има няколко сериозни недостатъци. Първо, изборът на пароли обикновено е малък, тъй като потребителят иска парола, която да се запомня и набира лесно. Обикновено за пароли се използват име на роднина, на приятел, рождена дата, име на герой от любим роман или дума, свързана с хобито на ползуващия паролата. Потребителите се стремят да имат една и съща парола при различните системи. Всичко това ограничава броя на паролите, които трябва да претърси нарушителят. Някои системи подпомагат потребителя, като осигуряват програма, която генерира случайни пароли. Ако паролите, създадени от системата, са прекалено трудни за запомняне (като LBTXHP) потребителите понякога записват паролата, вместо да рискуват да я забравят. Съществуването на написано

копие увеличава вероятността за компрометиране. Някои програми създават случайни пароли, които имитират лесни за произнасяне и запомняне думи, затова не е нужно да бъдат записвани.

Фактът, че паролата се използва извън компютърната система, е друг недостатък, който дава няколко възможности за компрометиране. Нарушителят може да надникне зад рамото на потребителя, когато паролата се отпечатва върху терминала, или може да вземе лентата от терминала, за да получи паролата. Нарушителят може да подслуша линията от терминала до компютъра и да запише паролата.

Системата с парола е уязвима и поради списъка на пароли, който трябва да се съхранява в компютъра. Ако този файл не е защитен достатъчно, той очевидно ще бъде цел за проникващия. Дори защитеният файл е уязвим. Не е необходимо обаче някой да има достъп до файла с паролите, не е нужно дори файлът да съдържа паролата. Когато паролата се поставя за първи път в системата, тя може да се трансформира с някаква функция (която е трудно или невъзможно да бъде променена) и да се съхранява трансформацията, а не оригиналната парола. Когато потребителят въведе паролата от терминала, използва се същата трансформация и резултатът се сравнява с трансформацията, съхранена във файла с паролите. Тъй като е трудно да се обърне функцията, съдържанието на файла с паролите не може да се преведе с която и да е проста функция за получаване на оригиналната парола. Нарушителят може да симулира функцията с компютъра и да опита всички възможни пароли. Той все пак може да открие една, съответствуваща на трансформирания резултат, който се съхранява във файла с паролите, но разумният избор на дължината на паролата и сложността на функцията може да блокира всяко подобно действие, като направи средното време за решението няколко години. . .

Остава проблемът с потребителя, който забравя паролата. Трябва да има механизъм, чрез който потребителят да може да получи нова парола. Може да се разреши на потребителя да отиде до системния администратор, да представи някаква идентификация и след това да може да въведе нова парола в системата. Администраторът все още няма да може да наблюдава новата парола. Потребителската парола може да бъде променена от администратора без разрешението на потребителя, но той най-малкото ще бъде уведомен за промяната, като не му се разреши да ползува системата при следващия му опит за включване.

Гарантираната от паролите сигурност може да се разшири с правила за ползуване на паролите. Понякога може да се поддържа списък на паролите и от потребителя, и от компютърната система. Всяка парола се прилага веднъж (или ограничен брой пъти), премахва се от списъка и след това се използува следващата дума от списъка. Дори ако потребителят има само една парола, продължителността или броят на ползуванията ѝ могат да бъдат ограничени. Когато се стигне до такъв край, потребителят или сменя паролата, или спира да ползува системата. Нарушител, който узнае паролата, е ограничен в ползуването ѝ.

### 6.5.2. УСТАНОВЯВАНЕ НА АВТЕНТИЧНОСТТА НА СИСТЕМАТА ОТ ПОТРЕБИТЕЛЯ

Понякога потребителят не е уверен, че влиза във връзка с правилната компютърна система. Ако нарушителят има достъп до съобщителната линия, той може да включи към нея свой миникомпютър, така че потребителят да се свързва с него, а не с желаната компютърна система. Миникомпютърът ими-

тира действията на компютърната система, с която потребителят вярва, че е във връзка. Миникомпютърът записва цялата информация на потребителя за установяване на автентичността му. В този момент нарушителят може да използва записаната информация, за да влезе в компютърната система.

В система, където паролите се използват само веднъж, е възможна същата методика. Потребителят вярва, че включването е успешно. Чрез миникомпютъра нарушителят го уведомява, че компютърната система се поврежда и няма да работи за определен период от време. Сега нарушителят има парола и време, през което може да ползува системата, тъй като може да се вярва, че потребителят няма да се опита да получи достъп до системата през този период. Щом нарушителят използува паролата, тя се заличава от списъка на валидните пароли за системата, ето защо следващия път, когато потребителят се включи, неговият и системният списък от пароли пак са еднакви.

За да се предотврати заобикалянето на механизма за установяване на автентичността чрез имитиране, може да се създаде сложна поредица от взаимни опознавания. При този метод потребителят осигурява частична идентификация, системата осигурява частична идентификация и т. н., докато потребителят и системата бъдат задоволени. Поне част от тази поредица трябва да бъде уникална за всеки потребител, т. е. системата трябва да снабдява всеки потребител с уникална информация. Ако обаче взаимното опознаване не е проектирано внимателно, например с еднократна парола и от двете страни, нарушителят може да използува миникомпютър в предавателната линия, за да влезе в компютърната система. В този случай обаче нарушителят трябва да имитира и потребителя, и системата.

Като механизъм за установяване на автентичност тайнописът има едно предимство пред методите за установяване на автентичност — автентичността на потребителя се установява едновременно с установяването на автентичността на системата за потребителя.

### 6.5.3. ОТКРИВАНЕ И ЗАПИСВАНЕ НА ЗАПЛАХИ КАТО ПРЕПЯТСТВИЯ ЗА НАРУШИТЕЛИТЕ

Ако нарушителят си даде сметка, че опитите му се проследяват и записват, той може да реши да проникне по друг начин. Откриването на заплахи и компрометирано записване, два метода, които служат като препятствия за нарушителите, са обсъдени в раздела за сигурността на програмите и файловете.

Слабостта и на трите форми за установяване на автентичността е използването на несигурни комуникационни линии. Тайнописът може да компенсира тази слабост, да даде сигурност и други допълнителни изгоди за комуникационните системи изобщо, като осигури още едно защитено ниво в компютърната система.

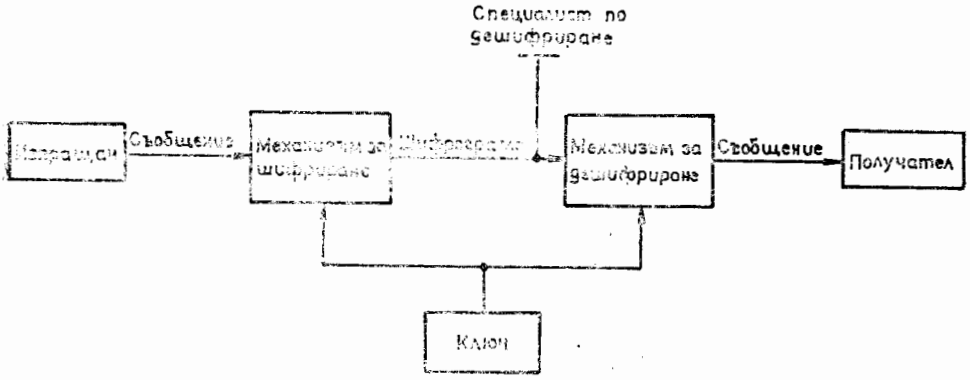
## 6.6. ТАЙНОПИС

Тайнописът, третото средство в арсенала на програмиста, може да се ползува за защита на информацията дори ако тя попадне в чужди ръце. Много компютърни системи предават данни по телефонни линии, радиолинии или други несигурни пътища, които могат да бъдат подслушани лесно от нарушителя. Нарушителят може да има пълен запис от двете страни на взаимодействието между компютър и потребител. Затова тайнописът се прилага в компютърните комуникации.

Необходимостта да се защищава информацията от любопитни е призната от много време. Един от по-простите шифри се приписва на Цезар.

### 6.6.1. МЕТОДИ ЗА ШИФРИРАНЕ

*Тайнописът* е такова преобразуване на съобщението, че дори ако нарушителят го види, да не може да го разбере. Оригиналното съобщение се нарича *ясно*. Процесът на преобразуване се нарича *шифриране*. Обикновено се прилага ключ, който определя коя от многото възможни трансформации на съобще-




Фиг. 6.10. Обобщена система за тайнопис

нието всъщност е извършена. Преобразуваното съобщение е *шифрограмата* която трябва да се *дешифрира* (със същия ключ), за да се получи същото съобщение. Нарушителят (или *противникът)* се опитва да *дешифрира* шифрограмата, за да получи оригиналното съобщение, без да знае ключа или приложения метод.

Схемата на обобщената тайнописна система е показана на фиг. 6.10. Първо се определя ключът и се изпраща по безопасен начин и на изпращача, и на получателя. Изпращачът създава нешифрирано съобщение. То се шифрира и и след това се предава на получателя. Получателят дешифрира шифрограмата, като използва ключа и получава оригиналния текст. Приема се, че информацията може да бъде разкрита само между механизмите за шифриране и дешифриране. Нешифрираното съобщение от двете страни трябва да се защити с други мерки.

Методите за шифриране (*шифрите)* могат да се разделят на два класа — прости и съставни. Простите шифри са елементи на съставните шифри.

#### Прости шифри

Простият шифър има три основни форми — заместване, разместване и Vigenére. Освен някои случаи на шифъра Vigenére обикновено простите шифри не осигуряват добра защита.

● **Шифри със заместване.** Знаците или групи от знаци се заменят със заместителни знаци или групи от знаци. Най-просто е всяка отделна буква да се замести с определен заместител. При по-прости форми на тайнописа често интервалите се елиминират.

По-сложните системи заместват буквите с групи от два знака, три знака или по-големи групи от букви. Ключът за такива замени нараства бързо с увеличаването на броя на буквите в групата. Може да се използват аритметични действия за n-грамно заместване.

Друга разновидност на заместването е множественото смесено азбучно заместване. Това е просто набор от n прости замествания, в определен ред при последователните букви в съобщението. Ключът се състои от ключовете на отделните замествания и реда, по който те трябва да се прилагат.

● **Шифри с разместване.** За този тип шифър съобщението се разделя на групи от по n знака, след това всяка група се размества по един и същи начин. Периодът на разместването е големината на групата. Ключът е действително използваното разместване и може да се представи удобно като разместване на първите n знака. Разместването може да се прилага последователно, като се създава съставно разместване. Резултатът е разместване, чийто период е най-малкият общ множител на използваните периоди на разместване.

● **Шифри Vigenère.** Най-простият шифър е използваният от Цезар. Всяка буква в съобщението се пренася напред в азбуката с определен брой букви. Пренасянето напред е ключът, който за латиницата е в границите от 0 до 25. За ключ 1 A става B, B — C и т. н., *Z* — A, като веригата се затваря. За съобщения, които не са азбучни, множеството от знаци е подредено така, че всеки знак в съобщението се пренася напред с толкова знака, колкото е ключът.

При по-обобщена система Vigenére ключът се състои от поредица от букви, периодът на шифъра е броят на буквите в ключа. Всяка последователна буква в съобщението се шифрира, като за ключ се използва местоположението на съответствуващия знак от ключа за множество от знаци.

Могат да се приложат два или повече шифри Vigenére последователно. Както при съставното разместване, периодът на резултата е най-простият общ множител на периодите на отделните шифри Vigenére.

Особен случай на шифъра Vigenére възниква, когато ключът е безкраен (или изглежда безкраен). Това е системата Vernam. Този шифър е прост. Той дава пълна сигурност, ако ключът се състои от произволни букви. Ключ, който се състои от текст със значение, произвежда шифър с бягащ ключ.

Интересна разновидност на шифър от типа Vigenére (шифър с автоключ) използва или самото съобщение, или шифрограмата за ключ. Използва се начален ключ (от n знака) за шифриране на първите n знака от съобщението и шифрирането продължава, като се използва съобщението или шифрограмата (изместена с n знака) като ключ.

● **Кодове.** Прилагат се отдавна в компютърните системи, но обикновено, за да се пести място, а не за сигурност. В кодовата система се използват символ или група от символи за задаване на съобщение. Кодовете са форма на заместване на семантично ниво, като ключът установява съответствие между кодираните символи и значенията им.

Формата на ключа за кодовата система изяснява основната разлика между тайнописа и кодовете. В тайнописната система всяко съобщение се шифрира; кодовата система се прилага само за изпращане на съобщения, като те ползуват само семантиката в ключа.

### Сложни шифри

Простите методи за тайнопис могат да се съчетават по два начина, за да образуват съставни шифри. Първо, прави се динамичен избор кой прост шифър да се ползува. Възможни са няколко от по-простите методи на шифриране,

като ключът за съобщението съдържа идентификация за използувания метод, а също и ключа за този метод. При по-сложна система може да се ползува схема като тази при множество смесено азбучно заместване. Тогава ключът съдържа реда, по който се използуват методите, а също и ключа за всеки метод.

Второ, два или повече методи се прилагат последователно върху едно и също съобщение. Това образува система, която е продукт от всичките методи. Пример за такава система е дробният шифър, при който всяка буква се заменя с две цифри; полученият цифров поток се размества и след това двуцифровите групи се заместват с буквн.

Изборът на прост шифър или на група от прости шифри, които образуват съставен шифър, зависи от желаните характеристики на шифъра.

### Оценяване на системите за тайнопис

Клод Е. Шенън изброява пет критерии за оценяване на всяка система за тайнопис:

● *Ниво на секретност.* Необходимата сигурност за дадена система за тайнопис зависи от разрешавания проблем. При някои системи нивото на секретност е отлично. При тези системи шифрограмата може да се дешифрира само с ключа. При други системи са възможни няколко решения на шифрограмата. Сред системите, които се решават по единствен начин, големината на усилията, положени за решението, се изменя в широки граници.

● *Големина на ключа.* Ключът трябва да бъде създаден и предаден до местата за изпращане и получаване. Методът на предаване трябва да бъде напълно сигурен. Ключът трябва да се съхранява или може би да се запомня от потребителите на системата. Независимо че запомнянето върху дискове и ленти е намалило проблемите при съхранението, може би все още е за предпочитане ключът да бъде къс и да се изменя и запомня лесно.

● *Лесно ползуване.* Когато шифрирането или дешифрирането се извършват ръчно, очевидна е необходимостта от проста работа. В компютърната система не е така. Главните съображения са скоростта на операцията (изчисленията или предаванията да не се забавят прекомерно) и необходимата компютърна памет.

● *Разпространяване на грешките.* При някои системи за тайнопис грешка в една буква от шифрограмата създава множество грешки при дешифрирането ѝ. Обикновено тази характеристика е нежелателна, но понякога високата степен на разпространяване на грешката може да се използува за разкриване на грешки при предаване и на опити за проникване.

● *Разширяване на съобщението.* Някои системи за секретност увеличават големината на шифрограмата в сравнение с големината на съобщението. Използуват се нулеви съобщения или знаци за объркване чрез променяне на статистиката на езика.

Шенън подчертава, че тези критерии не са съвместими, когато се работи с естествени езици. (Естествени езици тук са човешките и алгоритмичните, за разлика от изкуствените, предназначени да намалят възможността за разкриване на шифрограмата.) Може да се направи компромис в зависимост от характера на даденото приложение:

ако необходимата секретност не е много голяма (както е при системи, където трябва да се предотвратят само случайни разкрития), може да се ползува всеки от представените прости методи;

ако е достъпен безкраен (или практически много голям ключ), системата Vernam с изцяло случаен ключ дава абсолютна сигурност;

ако няма ограничение за сложността на работата (няма ограничения за време и оперативна памет), може да се ползува изключително сложен шифър;

ако разпространението на грешки е разрешено, могат да се ползуват няколко добри смесителни преобразувания;

ако е разрешено съобщението да се разширява, към шифрограмата на специфични места може да се добавят произволни букви и да се изпратят безсмислени съобщения за объркване.

**Разшифриране на шифрограми**

При идеална система за секретност единствената информация, която може да получи противниковия шифроаналитик (който се опитва да установи оригиналния текст на шифрограмата или ключа, без предварително да ги знае) от шифрограмата е, че е изпратено съобщение.

При не съвсем идеални системи, достатъчно време и достатъчно голям обем от шифрирания материал анализиращият шифрите може да намали броя на възможните съобщения до малък набор или до единствено решение. Тъй като идеалните системи не са практични за всички случаи, трябва да се използуват и не съвсем идеални системи. Много важен показател на неидеалната система е коефициентът на работа. Анализиращият шифъра може да е в състояние да открие единственото решение на шифрограмата, но ако средното време, необходимо за това решение е 50 години, от резултата няма голяма полза. Това е значително по-добре от шифрограма, която дава не единствено, а пет решения след неколкодневни усилия.

Анализиращият шифъра разчита на статистическите свойства на езика. Те се дължат на излишъци в езика. В английския език пример за такъв излишък е буквената двойка qu. Буквата u е излишна, тъй като при обикновените английски думи след q винаги идва u. Един от основните статистически набори за езика е честотата на отделни букви, дву- и трибуквени съчетания.

Ако анализиращият шифъра има достатъчно дълга шифрограма, той може да опита няколко статистически теста върху нея, което може да ограничи работата за определяне на оригиналното съобщение или на неговия ключ. Прост шифър със заместване може да бъде разрешен лесно (ако е достатъчно дълъг), като се установят честотите на буквите в шифрограмата. Най-често срещаната буква вероятно замества *e*, тъй като *e* се използува в английския език по-често от която и да е друга буква. По същия начин честотата на останалите букви в шифрограмата помага на аналитика да определи останалата част от ключа.

Два са методите за отнемане на статистическия анализ от аналитика, когато не се ползува идеална тайнописна система. При първия се отстраняват излишъците в езика; за целта се използуват кодове.

При втория метод се прекъсват зависимостите, които използува аналитикът. Шенън нарича това прекъсване объркване или разсейване в зависимост от това кои взаимоотношения са засегнати. При метода, с който се постига прекъсването, всяка буква в шифрограмата зависи от много букви в съобщението, а не от няколко или само от една. В резултат се създава система на секретност с голямо разпространение на грешката; ако тази особеност се окаже приемлива, може да се създаде много сигурна система. Идеята е да се приложи прост шифър, след него добър смесващ шифър и след него друг прост шифър. В общи линии смесващите шифри са блокови шифри, т.е. блок с фиксирана дължина от знаци се шифрира в една операция. Поради сложния

характер на необходимите за смесващите шифри операции, реализацията на смесващия шифър е по-лесна (и с по-малко грешки), когато се използва компютърна система.

### Сравнение между компютърното и ръчното шифриране

Реализираният от компютъра тайнопис се различава от традиционния тайнопис в три области — сложност на операцията, опростяване на азбуката и увеличаване на използването на ключа. Първата от тях (сложността на операцията) се основава на точността и скоростта на компютъра — неудобните за ръчна работа методи за тайнопис могат да се прилагат успешно при компютрите.

За компютърно ползуване тайнописните методи със заместване и Vigenére могат да се опростят поради намалената големина на компютърната азбука в сравнение с по-големите буквени азбуки—нали всички данни се представят с цифрите 1 и 0. Това опростява простото заместване в два случая, като или 0 замества 1 и 1 замества 0, или 0 замества 0 и 1 замества 1. Ключът може да се представи като двоично число, като 1 представя обръщането на цифрите, а 0— втория случай. В табл. 6.1 е изяснена връзката между ясния текст, ключа и

Таблица 6.1

**Просто заместване с двоична азбука**

| Ясен текст | Ключ | Шифрограма |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 1 | 0 |

шифрограмата. Резултатът от заместването може да се постигне чрез събиране по модул две на ключа със съобщението и следователно става идентичен на Vigenére с период от 1.

При множествено смесено азбучно заместване ключът се състои от поредица от битове. Всеки бит е избор от една от двете възможни таблици на заместване. За реализирането на тази система ключът (повторен толкова пъти, колкото е необходимо) се прибавя по модул две към съобщението. Това прилича твърде много на шифър Vigenére с ключ със същата дължина. В таблицата се пояснява, че простото заместване или Vigenére могат да се реализират, като се използва изключващо ИЛИ. Едно от основните предимства при събирането по модул две (изключващо ИЛИ) за тайнописа е, че операцията по същество е реципрочна на самата себе си. Например:

```
Ясен текст:      1100
    Ключ:      ⊕1010
Шифрограма:      0110
Шифрограма:      0110
    Ключ:      ⊕1010
Ясен текст:      1100
```

Така при ползуването на шифър Vigenére може да се употреби проста програма за шифриране и дешифриране.

За увеличаване на секретността, осигурена от ключ, могат да се прилагат

числени методи. Ключ от 32 бита, ползуван при шифър Vigenére не ще устои на анализ. Ако обаче този ключ се ползува като начало на генератор на псевдослучайни числа, а псевдослучайният двоичен низ — за ключ, голям обем информация може да се шифрира, без да се повтаря ключът. Получената система не е толкова силна, като тази с напълно случаен ключ, тъй като псевдослучайният низ има статистически свойства, които могат да се ползуват от аналитик, но е значително подобрена в сравнение с началния ключ от 32 бита.

Най-подходящите методи за компютърно шифриране са системата Vigenére и съчетанието прост—смесващ—прост шифър (ПСП). Възможните методи са:

ползуване на единствен ключ;

координирано ползуване на списък от ключове;

координирано създаване на ключ от база от ключове или псевдослучаен генератор на числа;

ползуване на единичен ключ, който се изменя от съдържанието на всеки блок от съобщението.

Съчетанието ПСП е отстъпление от изискванията за леснота при работа и разпространение на грешката, докато шифърът Vegenére е отстъпление от изискването за големината на ключа. Тъй като всеки от двата шифъра има свои особености, изборът на шифъра (и на метода за ползуване на ключа) зависи от характерните особености на даденото приложение.

Системата ПСП може да ползува и четирите метода за създаване и употреба на ключ. Поради фактора работа, включен при разшифрирането, може да се работи дори с единствен ключ. По-безопасно е обаче да се ползува един от другите три метода за създаване на ключ независимо от това че са по-трудопоглъщащи. Тъй като ключът не трябва да бъде създаден бит по бит със съобщението, което трябва да се шифрира, работата, необходима за създаването на ключа, не е толкова много както при системата Vigenére. Системата ПСП има голямо разпространение на грешката. Следователно грешката в единствен бит от шифрограмата създава множество грешки в разшифрованото съобщение. Откриването на грешката се увеличава с цената на изопачено съобщение, ако не е възможно повторното му предаване. Системата ПСП изисква повече компютърно време от Vigenére поради сложността на процеса на шифриране. При системи с голям обмен на данните това може да бъде сериозен недостатък.

При шифъра Vigenére големият обем на обмена в компютърната система изключва ползуването на единствен ключ или къс списък от ключове, тъй като ключът трябва да се прилага само веднаж (в най-лошия случай рядко). Останалите два метода за създаване на ключове са подходящи за системата Vigenére и с двата може да се получи безкраен ключ, който изглежда произволен. Основният проблем е при откриването на грешки. Грешка в единствен бит при предаването остава грешка в единствен бит след разшифрирането, а това намалява вероятността за откриването й. Системата Vigenére е много бърза и осигурява отлична защита, ако ключът е напълно произволен.

При метод на шифриране Vigenére или ПСП могат да възникнат проблеми при координираното ползуване на ключа (независимо от това, дали е създаден и запомнен предварително или е създаван непрекъснато), ако се загуби координацията. Последиците зависят от възможността на изпращача и получателя да я открият и компенсират. Ако ключът се изменя от съдържанието на всеки блок от съобщението и възникне неоткрита грешка в единичен бит, получателят не може да създаде следващия ключ в серията и комуникационната връзка фактически се прекъсва.

### 6.6.2. ПРИЛОЖЕНИЯ НА ТАЙНОПИСА В КОМПЮТЪРНИТЕ СИСТЕМИ

Тайнописът може да се приложи в три основни области в компютърните системи — установяване на самоличността, защита на комуникациите и защита на информацията във файловете.

#### Установяване на самоличността с тайнопис

Установяването на самоличността с тайнопис е проста операция, тъй като при шифрирането и дешифрирането трябва да се ползува един и същ ключ. Типична *поредица за установяване на самоличността на потребителя* при терминал (приема се, че терминалът може да шифрира) е:

- Потребителят изпраща нешифриран идентификатор (ID) на компютъра.
- Компютърът получава ключа, свързан с потребителския идентификатор, и с него дешифрира всички следващи предавания.
- Потребителят въвежда ключа в терминала и продължава обикновената връзка с компютъра.

Ако нарушителят се опита да се представи за потребител, има вероятност той да въведе от терминала неверен ключ, компютърът (който работи с верния ключ) дешифрира съобщенията в безсмислици и разкрива опита за проникване. Основните предимства на този метод са: необходимата информация (ключът) не се предава по несигурни линии и самоличността може да се установи на всякакво устройство (терминал или друг компютър), което е във връзка с основния компютър.

За установяване на самоличността могат да се ползуват всякакви шифри или методи с ключове. Нужен е обаче сравнително сигурен тайнописен метод, тъй като нарушителят може да анализира комуникациите между терминала и компютъра, да разкрие ключа на потребителя и да проникне в компютъра.

#### Тайнопис в компютърните системи

Значителен обем информация се предава по телефонни линии като комуникационна връзка между два компютъра или между компютър и терминал. Тази информация може да бъде подслушвана сравнително лесно. Други форми на комуникации (ултракъсовълновите) също могат да бъдат подслушвани. Тайнописът може да осигури ефективна сигурност, когато се прилагат несигурни средства за връзка.

Нарушителят, който подслушва комуникационната мрежа, има няколко метода, за да я компрометира. Той може да запише информацията, за да я използува по-късно, може да прихване блокове от съобщението (част от съобщението с фиксирана големина, която се шифрира като едно цяло), може да добавя фалшиви или да променя съществуващи блокове от съобщението. Шифрирането на съобщението не допуска ползуването на информацията от нарушителя, но не предпазва съобщението от прекъсване на връзката. Прекъсването може да бъде предотвратено, като се открият блоковете от съобщението, които не са в последователен ред, като се извърши специална обработка, за да се отхвърлят фалшивите блокове, или като се поиска повторно предаване на блоковете, които са били прихванати или променени.

За да се улесни откриването на опити за намеса, част от всеки блок от съобщението трябва да съдържа служебни данни. Важна част от тези данни е

поредният номер, или някакъв друг двоичен образец, който се съдържа само в един блок от съобщението. Ако някой постави подправен блок от съобщението в потока, вероятно служебният двоичен образец няма да съвпадне. Вероятността фалшификацията да бъде успешна зависи от броя на битовете, отделени за установяване на автентичността на всеки блок. Ако за установяването на автентичността на блока са отредени 10 бита, фалшифициран блок ще бъде приеман веднъж на 1024 пъти с вероятност по-малка от $10^{-3}$.

Служебните данни могат да се получат лесно чрез по-дълъг ключ от необходимото (като се приема, че ключът се създава непрекъснато) и като се използват допълнителни битове за идентификатор на блока от съобщението. Ако се работи по този метод, не е нужно да се шифрират служебните данни, а грешката може да се открие преди дешифрирането на блока от съобщението. Този метод не помага да се откриват подправени блокове, освен ако не са засегнати в служебните данни. За да се увеличи вероятността за откриване на променени блокове, служебните данни могат да се шифрират като част от блока на съобщението. Употребата на шифър с голямо разпространение на грешката, гарантира в повечето случаи, че промяната на която и да е част от шифрограмата води до грешки в служебните данни при дешифриране на съобщението. Следователно идентификацията на блока от съобщението и шифъра с голямо разпространение на грешката може да се ползуват за откриване на променени или дефектни блокове от съобщението, като вероятността се определя от броя на битовете, предназначени за служебна информация.

Лекотата, с която блок от съобщението може да бъде предаден повторно, зависи от метода на ползуване на ключа. Ако се работи с ключов списък или псевдослучаен поток от числа, при повторно предаване може да се ползува следващият ключ от поредицата. Ако обаче ключът се изменя от съдържанието на съобщението, получателят няма да може да обнови достатъчно ключа си, за да получи каквито и да са по-нататъшни съобщения. В този случай изпращачът трябва да ползува един от следващите четири метода:

ключът не трябва да се обновява, докато не отпадне всяка вероятност за повторно предаване;

всеки ключ трябва да се пази, докато не отпадне всяка вероятност, че ще бъде необходим при повторно предаване;

трябва да се ползува обратима трансформация на ключа;

трябва да се ползува буферна методика, така че да не е нужно повторно шифриране на блоковете.

Всеки от тези методи увеличава сложността на изчислението и запомнянето на процеса на шифриране.

### Тайнопис за защита на файлове

Записването на данните във файла може да се сравни с изпращането на съобщението, а прочитането на данните от файла е подобно на получаването на съобщението. Информацията във файла може да се прочете винаги, а информацията в комуникационната система обикновено се получава веднага след предаването. Тъй като повечето компютърни системи, които работят с файлове, се считат за несигурни (точно, както и комуникационните линии), за защита на информацията във файловете се използва тайнопис. Закъснението между записването и прочитането на информацията от файла води до няколко характерни свойства и проблеми, които не се срещат при традиционната комуникационна система.

Една от основните разлики между файловата и комуникационната систе-

ма е обемът на данните, достъпни за анализиращия шифъра. При комуникационната система подслушвачът има само една възможност да прихване блок от съобщението. При файловата система той може да прекопира целия файл. Тъй като трябва да се приеме, че нарушителят има на разположение за анализ целия файл (подобно на дълга поредица от блокове от съобщението), тук се засяга методът за създаване и използуване на ключове. При комуникационната система честата смяна на ключа означава, че с даден ключ са шифрирани ограничен брой блокове от съобщението. Във файловата система смяната на ключа изисква повторно шифриране на целия файл с новия ключ, което дава на аналитика много повече материал за работа, тъй като той разполага с целия файл, а не само с няколко блока от съобщението. Работата на анализиращия шифъра е по-лесна, ако той може да получи повече от един екземпляр от същия материал, шифриран с различни ключове. Следователно на честата смяна на ключа липсват някои от предимствата за сигурност, каквито има в комуникационната система.

При файловата система, а и при комуникационните системи шифърът Vegenére с голям ключ гарантира пълна сигурност на файла, ако проникващият не познава ключа. Във файловата система употребата на предварително създаден, напълно случаен ключ изисква ключов файл с дължина колкото на файла за шифриране. Това не само не е желателно поради необходимото допълнително място; този файл с ключове също трябва да бъде защитен. Ако се приеме, че нарушителят може да получи достъп до файла с данните, трябва също да се приеме, че той може да получи достъп и до файла с ключовете. Следователно трябва да се използува ключ, представляващ псевдопроизволен цифров низ, или ключ, който се изменя от съдържанието на всеки запис.

При последователен достъп до файл по вид системата се доближава много до комуникационната система с изключение на закъснението по време. Когато се изисква произволен достъп, проблемът е много по-труден, тъй като много от методите за създаване на ключ не са приемливи. Например псевдослучайният низ за ключ изисква низ до и включително частта, необходима за шифриране на желания запис. Използуването на ключ, който се изменя от съдържанието на файла, изисква дешифрирането на файла до необходимия запис. За да бъде шифрираната файлова система с произволен достъп практична, необходимо е ключът за дадена област от файла да бъде създаден, като се използува адресът на областта. За шифъра Vigenére като основна единица е удобно да се прилага запис във файла и с номера на записа да се създаде ключ. За блоковия шифър ПСП номерът на блока може да се използува за генериране на ключа. Методът с номер на блок или запис за създаване на ключ е много чувствителен, тъй като номерата следват в строг ред. Аналитикът може да възпроизведе ключа за всеки блок, освен ако създаването му не е достатъчно сложно. Един от възможните методи да се създава ключ е да се притежава основен ключ за файла, да се използува шифър ПСП за шифриране на номера на записа чрез ключа на файла и резултатът да се прилага или като ключ на записа, или като входни данни за някакъв механизъм за генериране на ключове. Шифърът ПСП се прилага поради сложността и високия работен коефициент.

Фактът, че файловата система действува като закъснителна линия, създава трудности за файл, който е шифриран с шифър ПСП. Закъснението между „изпращането“ и „получаването“ на данните елиминира възможността за повторно предаване на блок, ако се открие грешка. Тъй като шифърът ПСП има голямо разпространение на грешката, грешка в единствен бит от записа може да превърне целия запис в безсмислица. Ако не се предприемат съответни стъпки, данните в записа не могат да бъдат възстановени. Първо, копие от

данните върху магнитен носител може да се съхрани във физически защитена област. Този метод не е приемлив за файл, който се използува много често, по-добре е да се прилага схема за откриване и коригиране на грешките. Силното разпространение на грешката при шифър ПСП спомага да се открие грешката, а удобен код за поправка на грешката позволява да се възстанови началният запис, ако броят на битовете с грешка не е прекалено голям.

Непрекъснато обновяваните файлове с произволен достъп са друг проблем. Тъй като ключът за всеки запис трябва да се изчисли от номера на записа, анализиращият шифъра може да проследи записа в развитие и да получи няколко копия от същия запис, които са били шифрирани със същия ключ. Това е много опасна ситуация. За да се сведе този проблем до минимум, файлът може да се шифрира на подходящи интервали с нов ключ.

Изборът на тайнописния метод е само част от проблема, който стои пред програмиста.

### 6.6.3. ДИСЦИПЛИНА

Прилагането на всяка тайнописна система трябва да се съпровожда със строга дисциплина в проектирането, осъществяването и използуването на системата. В проекта трябва да се държи сметка не само за нормалния шум по предавателната линия, а също и за предумишлени опити за промяна. Това включва откриването на прехванати, фалшифицирани или подправени блокове от съобщението и възможността за възстановяването им. Процесът на шифриране не трябва да се вплита в проекта на системата. По-добре е това да е относително независим раздел. Правилният контрол върху тайнописния процес е достатъчно труден, без да се добавя и сложността на голямата степен на взаимодействие с други части на системата.

Недисциплинираната система, която използува тайнопис, може да доведе бързо до хаос. Ключът за всеки файл трябва да бъде записан и съхранен внимателно; ако ключът се загуби, потребителят не е в по-добра ситуация от този, от когото се опитва за защити файла. Ако ключовете за даден файл се променят често, трябва да се поддържа архив, така че файлът да може да бъде реконструиран. Преди всичко трябва да се уреди оперирането с ключовете и те да бъдат защитени. Прекалено сложната тайнописна програма има малка стойност, ако проникващият може да влезе в канцеларията и да вземе копие от ключа от нечие бюро или от незаключена картотека. Както и при всички други форми на сигурност, нарушителят търси и намира най-слабото звено в тайнописната система.

### 6.6.4. ПРИНЦИПИ НА ТАЙНОПИСНОТО ПРОЕКТИРАНЕ

Първо, програмистът трябва да приеме, че анализиращият шифъра ще успее да открие каква тайнописна методика е използувана; той може да открадне програмата или подруг начин —да узнае точния метод. Следователно не трябва да се работи със строго определен алгоритъм (без променлив ключ). Програмите трябва да са написани така, че лесно да могат да бъдат изменяни, ако сигурността се компрометира.

Системата трябва да се проектира така, че ключът да може да се сменя лесно. Ако смяната на ключа е прекалено трудна, потребителите ще се стремят да работят с един и същи ключ прекалено дълго. Това ще даде на анализиращия шифъра повече време, за да открие и използува ключа за дешифриране

на последователните шифрограми. Ключът трябва да се променя толкова често, че да не се дава достатъчно време за откриване на ключа. За да се увеличи полезният живот на ключа, той трябва да е достатъчно сложен. Ключът трябва да е избран внимателно, тъй като буквени или цифрови комбинации, като име на човек или рождена дата, могат да бъдат налучкани лесно. Трябва да се избират ключове, създадени с някакъв случаен или добър псевдослучаен процес.

Когато е възможно, всяко лице в системата трябва да получава само минимално необходимите знания за изпълнение на задълженията му. Потребителят може да получи ключа, но за него трябва да е трудно да получи копия от програмите за шифриране и дешифриране. Това ще предпази потребителя от изпълнението на тези програми извън нормалната защита на системата. Обратно, програмистът, който има достъп до програмите, не трябва да получава ключ за никой от файловете. Сигурността на системата не трябва да зависи от разделянето на достъпа до ключовете и тайнописните програми, но цената за получаване на ключа заедно с програмата за дешифриране може да се повиши.

На аналитика трябва да се даде възможно най-малко материал за работа. Програмистът не трябва да използува системния генератор на случайни числа в тайнописната система. Ако се приложи необичаен генератор, за аналитика ще е много по-трудно да определи неговите резултати. Колкото се може повече от статистическите зависимости в дадения език трябва да се премахнат. Трябва да се избягва шифриране и предаване на съобщения тогава, когато аналитикът може да се досети за значението на съобщението, освен ако информацията е без стойност и след това бъде изоставена. Предаването или запомнянето на несъществени данни или знаци, като номера на записите, край на записа или край на предаването на нешифрирани знаци, е също предпазна мярка срещу нарушителя. Ако информацията за шифриране има дълъг и ползотворен жизнен цикъл, трябва да се използуват сложна шифровъчна методика и еднократен ключ, и (като допълнителна защитна мярка) описанието трябва да се заключва в каса.

Анализиращият шифъра не трябва да има възможност да получи част от съобщението в шифрирана и ясна форма, защото запознат с метода за шифриране на съобщението и подпомогнат от компютри, ще снижи работния коефициент на тайнописната система. Файл, който е шифриран, никога не трябва дори частично да се предоставя нешифриран. Файл, който може би частично е компрометиран, никога не трябва да се шифрира. Ако аналитикът успее да получи образци от текст в шифрирана и нешифрирана форма, системата трябва да бъде проектирана така, че ключът да не може да бъде открит навреме, за да се приложи.

Всяка тайнописна система трябва да се проектира, като се имат предвид активните подслушвачи. Системата трябва да може да открива фалшифицирани или подправени съобщения и трябва да може да се възстановява. Тайнописната методика и методологията за предаването трябва да са устойчиви на грешки по линията, въведени от анализиращия шифъра с цел да се наблюдават определени или повторени съобщения.

Накрая, но не последно по важност — всеки, който проектира тайнописна система, трябва да познава методите, които използуват анализиращите шифрите. Много алгоритми изглеждат сигурни за непосветените, а всъщност се разчитат лесно от аналитика.

### 6.6.5. АПАРАТЕН ТАЙНОПИС

Най-ефективният метод да се реализира тайнопис в комуникационната линия е с апаратно тайнописно устройство. Апаратните средства са много по-бързи за сложни тайнописни методи и не може да се променят от системните програмисти. Един от примерите за апаратен тайнопис е системата „Луцифер". Независимо че „Луцифер" работи само във връзката между терминал и компютър, тя илюстрира някои от възможностите на апаратната тайнописна система.

Ключът в системата „Луцифер" е низ от 128 бита, който се въвежда чрез магнитна карта или чрез постоянно запомнящо устройство от 16 бита само за четене. В първия случай идентифицирането на потребителя предхожда всякакво предаване на шифрирани данни, така че компютърът може да избере подходящия ключ от файловете си. „Луцифер" може да изпраща ясен и шифриран текст, така че е възможен начален диалог за идентифициране. Във втория случай, когато се работи с постоянно запомнящо устройство, терминалът винаги употребява същия ключ, така че компютърът проверява идентификацията на терминала, за да избере съответния ключ.

Независимо че системата „Луцифер" има единствен ключ за продължителен период от време, все пак тя се счита за много сигурна. Шифърът ПСП увеличава много работния коефициент, поради което опитите да бъде разкрит ключът стават непрактични, дори ако има налице образци от нешифриран и шифриран материал. Единичният ключ елиминира трудностите, които се срещат при системи със синхронизиран списък.

### 6.6.6. ТАЙНОПИСНИ ПРОГРАМИ

Тайнописните програми могат да осигурят високо ниво на сигурност на компютърните файлове и комуникации, но само ако се прилагат правилно. Това не означава, че те осигуряват пълна сигурност. Ако тайнописната програма е избрана внимателно и правилните процедурни методи се спазват, нарушителят разбира, че е по-добре да открадне шифровъчната програма и ключа, отколкото да анализира шифрите на защитените файлове.

## 6.7. ОГРАНИЧАВАНЕ

Предотвратяването на нелегално разкриване или изтичане на информация от несигурна програма се нарича ограничаване. Всеки програмист се сблъсква с проблема да работи с програма (написана от друг), която може да сметне за несигурна. Ако програмата не работи правилно, програмистът може да установи грешката чрез тестване. Тук се разглеждат методите за ограничаване на програмата така, че да се предотврати изтичането на информация към нарушител чрез блокиране на информационните пътечки извън програмата.

### 6.7.1. ПЪТЕЧКИ ЗА ИЗТИЧАНЕ НА ДАННИ

Програмата може да „изпуска" данни по няколко начина през запомнящите устройства на компютърната система. Най-очевидното изтичане е поставянето на информация в постоянен файл, собственост на нарушителя. Нарушителят може да прочете файла и да получи данните. Ако програмата постави данните

във временен файл, нарушителят трябва да поработи малко повече, за да получи информацията.

Информация може да изтече до нарушителя по комуникационните канали. Изтичането може да бъде явно, когато програмата изпраща нелегално съобщение до нарушителя, или скрито, когато програмата скрива информацията в легално съобщение.

Последната форма за изтичане е по пътечки, които обикновено не се използват за предаване на информация. Тези пътечки могат да бъдат много скрити и да се установяват трудно.

Пътечките, по които обикновено не се предават данни, се използват, като получателят наблюдава какво въздействие оказва върху системата изпращането на заявки за обслужване на програмата. Някои от каналите за данни са много „шумни", но програми за откриване и коригиране на грешки могат да се използват да подобряват точността на комуникационния канал. Основният проблем е, че програмата, която изпраща данните, изисква обслужване от системата и собственикът ѝ може да открие въздействието на заявката.

### 6.7.2. ПРАВИЛА НА ОГРАНИЧАВАНЕТО

Основната идея за сигурност при ограничаването е да се определят пътечки на данните и да се блокират онези от тях, които могат да се използват от нарушителя. Ограничаване само на съмнителната програма може да не е достатъчно, тъй като тя повиква други програми, от които може да изтече информация. Всички програми, които се повикват от съмнителната програма, трябва да се класифицират като сигурни или като несигурни и несигурните да се ограничават. Супервайзорът трябва да бъде включен в този процес, независимо че ограничаването му не е възможно. След като определи набора от програми за ограничаване, програмистът може да продължи ограничаването, като блокира пътечките на данните.

Ограничената програма не трябва да ползува никакъв файл съвместно с програма, която не е ограничена. Това блокира прякото предаване на информация през файла и това, че файлът е отворен или затворен за предаване на информация. Следователно програмата трябва да може да ползува само файловете, определени от потребителя, а системата трябва да може да ги защищава. Пишещият програмата може да се противопостави на използването на файловете, определени от потребителя, тъй като информацията, поставена във файловете на програмата, може да има частен характер. Потребителят и пишещият програмата трябва заедно да определят схема, която да задоволява изискванията за защита и на двамата.

Проблемът при програмата, която съхранява вътрешно информация, може да бъде решен по два начина. За потребителя е най-лесно да поиска копие от програмата с ограничен достъп. Това копие може да е защитено така, че пишещият да не може да го изпълнява. Тогава, дори ако програмата съхранява данни, те не са достъпни за пишещия. По-трудно за налагане е изискването програмата да няма място за съхраняване на данните, така че те да не бъдат унищожени в края на изпълнението на програмата. Много програми се предават без листинг и е трудно да се определи дали имат област за съхраняване на данните.

Обикновените канали за връзка между междупроцесни комуникации се откриват и блокират по-трудно. Забележително изключение е механизмът за повикване на подпрограми. Разкриването на информация чрез подпрограмно повикване се предотвратява, тъй като програмистът ограничава всички съмнителни програми, които могат да бъдат повикани. Другите канали обаче не

са така прости. Програмистът може да забрани използването на каквито и да са междупроцесни комуникации, ако програмата може все още да работи при такива условия. От друга страна, може да е трудно да се определи дали програмата ще използва междупроцесните комуникационни възможности в зависимост от операционната система. Съобщенията , изпращани от програмата, може да се проследяват периодично.

Съобщенията, скрити в легалната комуникация между програмата и пишещия, се откриват много трудно без експерт по тайнопис. Форматът, съдържанието и честотата на такива съобщения трябва да бъдат ограничени.

Предаването на информация по канали, по които обикновено не се предават данни (скрити канали), най-често включва влияние на програмата върху системата и следователно въвлича супервайзора. Съобразно със строгите правила на ограничаването, след като програмата, която трябва да бъде ограничена, използва супервайозора,той или трябва да бъде ограничен, или трябва да му се има доверие. При това, а и при други съображения за сигурност, които включват супервайзора, програмистът обикновено няма избор; той трябва да се задоволява с възможностите на съществуващите супервайзори.

Ако супервайзорът е проектиран правилно, той може да блокира скритите канали или най-малкото силно да ограничи скоростта на предаване на данните.

Когато се опитва да ограничи програмата, програмистът трябва да се погрижи за паметта, легалните канали за данни, каналите, предназначени за предаване на данни, и другите програми, които могат да бъдат повикани от съмнителна програма. Възможно е програмата да се ограничи напълно, но е скъпо и не всички програми могат да работят правилно при пълно ограничаване. При подходящи ограничения на работата и с малко помощ от супервайзора много програми могат да бъдат ограничени, независимо че не всички пътечки на данните са блокирани.

## 6.8. МЕТОДИ ЗА СИГУРНОСТ, ПРОГРАМИРУЕМИ ОТ ПОТРЕБИТЕЛЯ

Пред програмиста има два основни проблема на сигурността —да защити файловете и програмите и да създаде сигурни програми. Тук се засяга само вторият проблем, тъй като файловете и програмите могат да се защитят само от операционната система (или чрез тайнопис) Два са аспектите за създаване на сигурна програма—предотвратяване на опитите за проникване и налагане на ограничения в достъпа, които не се поддържат от операционната система.

### 6.8.1. ПРЕДОТВРАТЯВАНЕ НА ОПИТИТЕ ЗА ПРОНИКВАНЕ

Проникването може да се предотврати, като се блокират пътищата за проникване или като се използва програма, която възпира нарушителя да използва съществуващ път за проникване.

Програмата трябва да установява сама автентичността, ако операционната система не го прави или ако установяването ѝ от системата не е достатъчно за изискванията за сигурност на програмата. Ако програмата трябва да установи пълната автентичност на потребителя, могат да се ползуват описаните методи (например установяване на автентичност чрез парола). Програмата може да включва проверка на местонахождението на потребителя или времето, за да бъдат наложени ограничения за достъп, зависещи от потребителя. Прог-

рамата трябва да зависи от системата, която осигурява допълнителната информация; ако системата не е надеждна в това отношение, не могат да се наложат ограничения за достъп, зависещи от потребителя.

Една от простите форми на защита с парола изисква да се постави единствена парола в оператор за данни от програмата, тъй като се използват знаци, невъзможни за отпечатване, така че листингът да не може да издаде паролата. При програма, която работи с перфокарти за входните данни, полето, което трябва да съдържа паролата, може да бъде скрито сред другите полета на една от първите перфокарти с данни. Именуването на паролата в програмата е по-нататъшно усложняване. Тогава сравняването на входната парола със запомнената парола изглежда като проверка на свързаността на входните данни, особено поради невъзможните за отпечатване знаци. Тази проста система блокира случайния читател, но решителният нарушител вероятно няма да бъда спрян за дълго; основният пропуск е, че схемата зависи от собствената си секретност, а не от секретността на паролата.

Откриването и записването на заплахи е важно средство за предотвратяване на нелегалното ползуване на програмата. При този метод програмата прави запис на всяка подозрителна дейност или прекъсва изпълнението, щом забележи такава. Ако опитите да се установи автентичността на потребителя пропадат непрекъснато, той може да бъде уличен за нарушител, който опитва произволни пароли. Друга форма за откриване на заплаха е да се прави проверка на свързаността, за да се установи, че входните данни и (или) вътрешните променливи са в допустимите обхвати. Подходящият набор от проверки за установяване на заплаха ограничава силно възможността нарушителят да ползува програмата злонамерено. Ако възможностите на системата за откриване на заплаха се пазят в относителна тайна, нарушителят може да не бъде особено внимателен и да попадне на някой от механизмите за откриване. Ако е добре известно, че системата може да открива заплахи, нарушителят може да прецени, че рисковете надхвърлят облагите и може да се откаже от проникване.

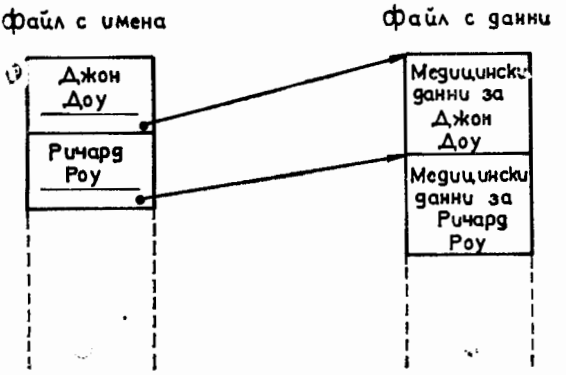
Наличността на подходящи ревизионни процедури също е препятствие за нарушителите, тъй като информацията, съхранена за ревизионно предаване, по-късно може да посочи потребителя, отговорен за злонамерената дейност.

Програмистът трябва да работи в тясно сътрудничество с ревизор при проектирането на всяка програма, представляваща част от система, която трябва да се ревизира. За да се предотвратят компютърни измами, програмата трябва да бъде проектирана да извежда данни за нормални ревизионни процедури и за всякакви специални ревизии, които потребителите на програмата могат да поискат.

### 6.8.2. НАЛАГАНЕ НА ОГРАНИЧЕНИЯ В ДОСТЪПА

Програмистът трябва да наложи ограничения в достъпа, които не се налагат от операционната система. Операционната система ограничава достъпа до файловете или програмите, а самата програма трябва да ограничава достъпа до записите във файла или до полетата в записа. Програмата, която извлича данни от база от данни, може да бъде процедурна (написва се специална процедура за задоволяване на дадена заявка) или непроцедурна (потребителят получава списък на операциите, реализирани от програмата, и може да изисква всяка от тях). От друга страна, необходим е подходящ механизъм за установяване на автентичността, за да могат да се приложат правилно ограниченията за достъп.

Най-лесният начин да се осъществи достъпът до база от данни, поне за целите на сигурността, е да се ползуват непроцедурни методи; потребителят дава списък на операциите, които иска да бъдат изпълнени от програмата. Програмата на базата от данни интерпретира този списък и връща резултатите на




Фиг. 6.11. Защита чрез контролиране на пътечките за достъп

потребителя, ако няма нарушения на сигурността. В тази система се виждат ясно разликите между зависимите и независимите от данните ограничения за достъп. Независимите от данните ограничения могат да се осъществят чрез начално сканиране на списъка от програмата. Потребителят или има, или няма достъп до дадено поле. Следователно всяка заявка може да се отхвърли незабавно, щом правата за достъп са нарушени. Зависимо от данните ограничение за достъп се изпълнява при обработване на заявката. Независимо че непроцедурният метод ограничава потребителя до определен брой операции, такава система може да поддържа много сложни заявки, ако е проектирана правилно.

Непроцедурната информационно-търсеща система не е подходяща за някои форми на достъп. Може да има заявка, която не отговаря на списъка, тъй като потребителят не е снабден с достатъчно голям речник. Дадена заявка може да се прави много често и разходите за непроцедурната система могат да бъдат много високи. Може да е нужно да се обработва всеки запис, извлечен от базата от данни, и да няма удобна връзка между информационно-търсещата система и обработващата програма. За тази цел информационно-търсещата система осигурява необходимите възможности, а също и допълнителни затруднения в сигурността. В основни линии за изпълнение на заявката се пише нова или се изменя старата програма.

Най-сигурната форма на процедурен достъп до базата от данни е администраторът на базата от данни да пише всички програми, тъй като той носи отговорност за базата от данни и е фактически част от защитената система. Ако обаче системата работи много, това може да претовари администратора или информационно-търсещата система да бъде затрупана с програми, някои от тях с минимални разлики.

За да се намали това натоварване, на потребителите може да се разреши да пишат собствените си програми. Тази ситуация е много „нездравословна" за сигурността, освен ако се наложат много строги правила. Един от начините да има сигурност и все пак да се разрешава на потребителите да имат пряк достъп до базата от данни е да се контролират пътищата за достъп до данните, като всеки потребител получава права да работи само с някои от тях. Напри-

мер информацията за пациентите на един лекар може да се организира така, както е показано на фиг. 6.11. Имената на пациентите отиват в един файл, а данните за тях — в друг файл. Лекарят има достъп само до файла с имената. Изследователят има достъп до групата данни и може да извършва статистически анализ, но не може да свърже данните с име, тъй като няма право за достъп до файла с имената. Това приложение на контрола на пътищата за достъп осигурява специална форма на защита — личната неприкосновеност.

## 6.9. НЕПРИКОСНОВЕНОСТ

Въпросът за неприкосновеността възниква при всяка компютърна система, която съхранява лична информация за хора. Програмистът трябва да познава юридическите права на лицата, за които се съхранява информация в базата от данни. Юридическото ниво на сигурност е най-ниското ниво, допустимо при всяка информационно-търсеща система; желателно е обаче по-високо ниво на сигурност. Програмистът не решава тези въпроси, той само създава програми и реализира правилно определени решения.

### 6.9.1. БАЗИ ОТ ДАННИ

Основните форми на базите от данни са две — база от данни с досиета и база от статистически данни. Базата от данни с досиета може да подаде данни за самоличността на лицата — така може да се наруши неприкосновеността им.

В системата за сигурност на базата от данни програмистът трябва да вземе предвид значението на данните за лицето, за собственика на данните (собственика на базата от данни) и за нарушителя.

## 6.10. ИЗМЕРВАНЕ НА СИГУРНОСТТА

Категоризирането (удостоверяването) на нивото на сигурност на дадена програма или операционна система е субективен проблем. Не е възможно например да се даде оценка от едно до десет, тъй като нивото на сигурност зависи от ползуването на системата и от средата, в която работи тя. Следователно всеки, който оценява сигурността на системата, трябва да има предвид набор от изисквания за сигурността. Три от методите за определяне на нивото на сигурността на системата са — познаването на проектните и програмните методи за изграждане на програмата; използуването на екипи по проникването; програмните доказателства.

Нивото на сигурността може да се определи, като се наблюдава качеството на програмата в системата. Например програмата трябва да е модулна и всички програми, свързани със сигурността, трябва да са изолирани. Всички автоматични възприемания в системата трябва да са на най-ниското ниво на привилегии, като осигуряването на сигурността се проектира така, че да се ползува лесно. Този метод дава само бегла представа за нивото на сигурността, но може да е отправна точка за по-сериозно търсене на пропуски.

Екипите по проникването могат да установят приблизителните разходи за проникване в системата. Допълнителен резултат от техните усилия е откриването на системни грешки, които могат да бъдат поправени; понякога обаче грешката не може да бъде поправена без повторно написване на основните

части на системата. Възможно е екипът по проникването да изолира определени форми на грешките и да създаде автоматизирани средства за други проявления на тези типове грешки.

Най-силната форма на удостоверяване е официалното доказване, че програмното осигуряване работи точно както е специфицирано и че формите на сигурност, за които се претендира, действително съществуват. Това се прави трудно дори и за малки програми и обикновено е непрактично за набор от програми с размерите на операционна система. Един от подходите към този проблем е да се създаде ядро на сигурността и да се докаже правилността му.

## 6.11. ОБОБЩЕНИЕ

Програмистът може да получи сигурност, като блокира изцяло нелегалния достъп до данните или като увеличи разходите за такъв достъп толкова, че цената на проникването да стане по-висока от цената на официално получаваната информация. Достъпните средства за получаване на сигурност включват:

- Операционната система — определя как потребителите да подхождат към обектите в системата и осигурява основата за програмираните от програмиста мерки за сигурност.
- Установяването на автентичността — не позволява на нарушителя да се представи за легален потребител.
- Тайнописът — защищава информацията, независимо че нарушителят може да има достъп до нея и може да осигури автентичност.
- Ограничаването — блокира пътечките на данните, водещи извън програмата.
- Механизмите, програмирани от програмиста — представляват пречки срещу проникването и ограничават възможността нарушителят да злоупотребява със системата, без да бъде открит.

## ЛИТЕРАТУРА


*Общо*

Martin, J. Security, Accuracy and Privacy in Computer Systems. Englewood Cliffs, N. J., Prentice-Hall, 1973.

Saltzer, J. H., M. D. Schroeder. The Protection of Information in Computer Systems. Proceedings of the IEEE, 16, 9 (септември 1975), 1278—1308.

Van Tassel, D. Computer Security Management. Englewood Cliffs, N. J., Prentice-Hall, 1972.

*Сигурност на операционната система*

Fabry, R. S. Capability-Based Addressing. *Communications of the ACM, 17,* 7 (юли 1974), 403—412.

Fabry, R. S. Dynamic Verification of Operating System Decisions. *Communications of the ACM, 16,* 11 (ноември 1973), 659—668.

Graham, G. S., P. J. Denning. Protection — Principles and Practice. *AFIPS Conference Proceedings, 40* (1972 SJCC), 417—429.

Graham, R. M. Protection in an Information Processing Utility. *Communications of the ACM, 11,* 5 (май 1968), 365—369.

Lampson, B. W. Protection. Proc. Fifth Princeton Symposium on Information Sciences and Systems, Princeton University, март 1971, 437—443, препечатано в *Operating Systems Review, 8,* 1 (януари 1974), 18—24.

Watson, R. W. *Timesharing System Design Concepts.* New York, McGraw Hill, 1970.

*Установяване на самоличност*
Evans, A., W. Kantrowitz. A User Authentication Scheme Not Requiring Secrecy in the Computer. *Communications of the ACM, 17,* 8 (август 1974), 437—442.
Purdy, G. B. A High Security Log-in Procedure. *Communications of the ACM, 17,* 8 (август 1974), 442—445.
*Тайнопис*
Baran, P. On Distributed Communications: IX. Security, Secrecy and Tamper-Free Considerations. Doc. RM-3765-PR, RAND Corp., Santa Monica, Calif., август 1974.
Chesson, F. W. Computers and Cryptology. *Datamation, 19,* 1 (януари 1973), 62—64, 77—81.
Feistel, H. Cryptography and Computer Privacy. *Scientific American, 228,* 5 (май 1973), 15—23.
Friedman, T. D., L. J. Hoffman. Execution Time Requirements for Encipherment Programs. *Communications of the ACM., 17,* 8 (август 1974), 445—449.
Kahn, D. The Codebreakers. New York, The MacMillan Company, 1967.
Shannon, C. E. Communication Theory of Secrecy Systems. *Bell Systems Technical Journal, 28* (октомври 1949), 656—716.
Wolfe, J. M. A First Course in Cryptanalysis, 3 тома. Brooklyn, N. Y. Brooklyn College Press, 1954.
*Изолиране*
Lampson, B. W. A Note on the Confinement Problem. *Communications of the ACM, 16,* 10 (октомври 1973), 613—615.
*Неприкосновеност*
Haq, M. I. Insuring Individuals Privacy From Statistical Data Base Users. *AFIPS Conference Proceedings, 44* (SJCC 1975), 941—946.
Hoffman, L. J., W. F. Miller. Getting a Personal Dossier From a Statistical Data Bank. *Datamation, 16,* 5 (май 1970), 74—75.
Minsky, N. Intentional Resolution of Privacy Protection in Database Systems. *Communications of the ACM, 19,* 3 (март 1976), 148—159.
Turn, R., N. Z. Shapiro. Privacy and Security in Databank Systems — Measures of Effectiveness, Costs and Protector — Intruder Interactions. *AFIPS Conference Proceedings* (FJCC 1972), 435—444.

# СЪДЪРЖАНИЕ

**4. Структурно програмиране**



**5. Проверка и утвърждаване**



**6. Сигурност и неприкосновеност**

**ТЕХНОЛОГИЯ НА ПРОГРАМИРАНЕТО**

Автори: *Рандал Йенсен*
*Чарлз Тониз*

Преводачи: инж. *Огнян Асенов Сеизов*
инж. *Асен Огнянов Сеизов*

Националиост американска (САЩ)

Първо издание

Код 03 $\frac{9533172511}{3207-10-87}$

Изд. № 14521

Научен редактор инж. *Пламен Гъмизов*
Художник *Марин Держонков*
Художествен редактор *Досю Досев*
Технически редактор *Любчо Иванчев*
Коректор *Елена Ласкина*

Дадена за набор на 16.V.1986 г.
Подписана за печат м. февруари 1987 г.
Излязла от печат м. февруари 1987 г.
Формат 70×100/16
Печ. коли 22,75
Изд. коли 29,48
УИК 33,19
Цена 5,97 лв.

Държавно издателство „Техника", бул. Руски 6, София
Държавна печатница „Атанас Стратиев" — Хасково