Information Processing Letters 32 (1989) 213–219
North-Holland

THE DEMAND OPERATION IN THE SCHEMATIC PROTECTION MODEL

Ravinderpal Singh SANDHU *
Department of Computer and Information Science, The Ohio State University, Columbus, OH 43210, U.S.A.

Communicated by Fred B. Schneider
Received 2 August 1988
Revised 17 October 1988

We show that the demand operation in the schematic protection model is redundant in that it can be simulated by copy and create operations. We also consider to what extent amplification (or conditional demand) can be simulated by similar constructions.

Keywords: Information systems, operating systems, protection models, security

1. Introduction

For protection purposes computer systems are viewed as consisting of subjects and objects. Active entities such as users or processes are subjects while passive entities such as text files are objects. Protection is enforced by ensuring that subjects can execute only those operations authorized by privileges in their domains. The schematic protection model (SPM) [8] defines three operations by which a subject acquires new privileges: copy, create and demand. We show that demand is redundant in that it can be simulated by copy and create. This is surprising because demand was intended to, and indeed appears to, confer a different kind of ability than copy or create.

Redundancy of demand is an important step in understanding the minimal features needed in a protection model. The three SPM operations of copy, create and demand were thought to capture fundamentally different ways for acquiring privileges. Copy enables a subject to obtain a privilege from some other subject. For instance, the owner of a file can copy read or write privileges for that file to another user. Create introduces new privileges as the side effect of creating new subjects or objects in the system. For example, when a user creates a file he is given the owner privilege for that file. Finally, demand allows a subject to obtain certain privileges, literally by demanding them. For example, every faculty member of a department can be authorized to demand the read privilege for every departmental document.

It might appear demand can be eliminated by giving the demandable privileges to every subject. For instance, we can explicitly give every faculty member the read privilege for each departmental document. But then, whenever a new departmental document gets created, the privilege to read it must somehow be explicitly distributed to every faculty member. Automatic distribution of privileges is an unrealistic task for an operating system to manage correctly, particularly in a large distributed environment. Small and frequent incremental events such as creation of a new departmental document should not require such a sweeping impact. It moreover requires explicit representation of large numbers of privileges many of which may never be used.

* Present affiliation: Department of Information Systems and Systems Engineering, George Mason University, Fairfax, VA 22030-4444, U.S.A.

In practice the above problem can be, and often is, solved by performing the distribution in two steps as follows.

(1) Define a departmental library as a subject which contains read privileges for all departmental documents.

(2) Authorize every faculty member to copy read privileges from the departmental library, in effect demanding them.

With this set-up, when a new departmental document is created, the privilege to read it need only be placed in the departmental library.

Although this solution is quite practical, a closer look reveals we have merely pushed the problem one level further. Imagine we have several libraries in which departmental documents are made available to the faculty, perhaps organized by topic. Now, whenever a new departmental library is created, we are back to the same situation of having to explicitly give every faculty member authorization to copy read privileges from the new library. If the number of libraries is small and creation of new ones infrequent, this may be acceptable. Nevertheless, there appears to be a place for demand as a fundamental operation. After all, the above argument applies recursively to libraries of libraries and so on.

The demand operation was included in SPM on the basis of such arguments. The perception that it is fundamentally different from copy and create goes back to the very early work on this model [6]. Our main objective is to show that demand can be simulated by copy and create operations. Section 2 briefly reviews SPM followed by our construction demonstrating redundancy of demand in Section 3. In retrospect the construction is straightforward, perhaps even obvious. The result is an important correction to our intuition regarding the basic nature of demand. In Section 4 we go on to consider to what extent amplification (or conditional demand) can be simulated in SPM by similar constructions. Section 5 concludes the paper.

2. The schematic protection model

The dynamic privileges in SPM are tickets (capabilities) of the form Y/p where Y identifies some unique entity and p is a right symbol. Rights occur with or without the copy flag c. The crucial difference is that Y/p cannot be copied from one subject to another whereas Y/pc might be, if some additional conditions defined below are true. The notation $p{:}c$ denotes p or pc; with multiple occurrences in the same context either all read as p or all as pc. Each SPM ticket carries a single right. We often abbreviate sets of tickets by letting Y/P denote $\{Y/p{:}c \mid p{:}c \in P\}$.

Subjects and objects are collectively called entities in SPM. Subjects can possess tickets whereas objects cannot. Each entity is created to be of a specific type which does not change. The type of a ticket is determined by the type of entity it addresses and the right symbol it carries, that is, $\text{type}(Y/p{:}c)$ is the ordered pair $\text{type}(Y)/p{:}c$. By convention, types are named in lower case while entities are named in upper case. Notation for tickets extends in an obvious way to ticket types.

Tickets are acquired in accordance with specific rules. These rules are specified by a scheme defined by the following (finite) components, explained below:

(1) A set of entity types T partitioned into subject types TS and object types TO.

(2) A set of rights R; the set of ticket types is thereby $T \times R$.

(3) A demand function $d: \text{TS} \to 2^{T \times R}$.

(4) A can-create relation $\text{cc} \subseteq \text{TS} \times T$, equivalently viewed as a function $\text{cc}: \text{TS} \to 2^T$.

(5) A create rule $\text{cr}_p(u, v) = c/R_1 \cup p/R_2$, $\text{cr}_c(u, v) = c/R_3 \cup p/R_3$ for each $\text{cc}(u, v)$.

(6) A collection of link predicates $\{\text{link}_i\}$.

(7) A filter function $f_i: \text{TS} \times \text{TS} \to 2^{T \times R}$ for each predicate link_i.

An SPM system is defined by a scheme and an initial state, i.e., the initial set of objects, subjects and their initial domains. The scheme is static and controls how the state can evolve by demand, create and copy operations. Demand and create are authorized entirely by the scheme whereas copy is partially authorized by the scheme and partially by tickets.

The demand operation: Subjects of type u are authorized to demand all types of tickets specified by $d(u)$, i.e., if $v/p{:}c \in d(u)$ then every subject U of type u can demand $V/p{:}c$ for every entity V of

type v. For instance, if departmental documents and faculty members are entities of type ddoc and fac respectively, ddoc/$r \in d$(fac) authorizes the faculty to read every departmental document, including those which will be created in future.

The create operation: Subjects of type u can create entities of type v if and only if cc(u, v). For instance cc(fac, ddoc) authorizes faculty members to create departmental documents. Tickets introduced as the side effect of creation are specified by a different create-rule for every cc(u, v). Each rule has the two components shown above, where p and c respectively denote parent and child and the R_i's are subsets of R. When subject U of type u creates entity V of type v, the parent U gets the tickets specified by $\mathrm{cr}_p(u, v)$, i.e., V/R_1 and U/R_2. The child V similarly gets the tickets specified by $\mathrm{cr}_c(u, v)$, i.e., V/R_3 and U/R_4. For instance, cr_p(fac, ddoc) = c/rw and cr_c(fac, ddoc) = $\emptyset$ gives the creator read and write tickets for the created document. In general, if v is an object type then $\mathrm{cr}_c(u, v)$ should be empty.

The copy operation. A copy of a ticket can be transferred from one subject to another leaving the original ticket intact. Three independent pieces of authorization are required for copying $Y/p{:}c$ from U to V. Firstly U must possess Y/pc. Note that U is required to have Y/pc for copying either Y/pc or Y/p. Secondly there must be a link connecting U to V. Links are established by tickets for U and V in the domains of U and V. Links are specified in the scheme by predicates which take a pair of subjects as arguments. Let dom(U) denote the set of tickets possessed by U. Each predicate $\mathrm{link}_i(U, V)$ is defined as a conjunction or disjunction, but not negation, of one or more of the following terms for any $z \in R$: $U/z \in \mathrm{dom}(U)$, $U/z \in \mathrm{dom}(V)$, $V/z \in \mathrm{dom}(U)$, $V/z \in \mathrm{dom}(V)$, and **true**. For instance, the following are legitimate link predicates:

$\mathrm{link}_g(U, V) \equiv V/g \in \mathrm{dom}(U)$
(grant link [3]),

$\mathrm{link}_t(U, V) \equiv U/t \in \mathrm{dom}(V)$
(take link [3]),

$\mathrm{link}_{sr}(U, V) \equiv V/s \in \mathrm{dom}(U) \wedge U/r \in \mathrm{dom}(V)$
(send-receive link [5]),

$\mathrm{link}_b(U, V) \equiv U/b \in \mathrm{dom}(U)$
(broadcast link [8]),

$\mathrm{link}_u(U, V) \equiv$ **true**
(universal link [7]).

The third and final condition required for a copy operation is defined by the filter functions f_i : TS $\times$ TS $\rightarrow 2^{T \times R}$, one per predicate link_i. The value of $f_i(u, v)$ specifies the types of tickets that may be copied from subjects of type u to subjects of type v over a link_i. Example values are $T \times R$, TO $\times R$ and $\emptyset$ respectively authorizing all tickets, object tickets and no tickets to be copied. To summarize $Y/p{:}c$ can be copied from U to V if and only if

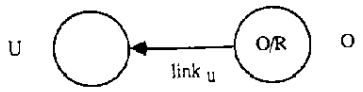
$$Y/pc \in \mathrm{dom}(U)$$
$$\wedge\, (\exists \mathrm{link}_i)\big[\mathrm{link}_i(U, V) \wedge y/p{:}c \in f_i(u, v)\big]$$

where the types of U, V and Y are respectively u, v and y. Note that f_i determines whether or not the copied ticket can have the copy flag.

3. Redundancy of demand

We now show that the SPM demand operation can be simulated by copy and create operations. For any given system we will construct an equivalent system whose scheme has empty values for its demand function. The central idea is to mimic demand by a copy operation over the universal link, i.e., $\mathrm{link}_u(U, V) \equiv$ **true**. However the SPM copy operation only copies existing tickets, whereas demand can introduce new tickets which did not previously exist in any subject's domain. This aspect of demand is mimicked by creation where the create-rules have the power to introduce new tickets.

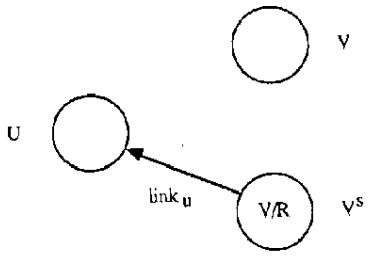
First consider demand of object tickets. We simulate each object in the original system as a subject in the modified system. Specifically let O be an object of type o in the original system. In the modified system it is simulated by a subject O of type o which possesses all tickets for itself, i.e., dom(O) = O/R. By virtue of possessing these self-tickets, O is technically an SPM subject in the modified system. We continue to assume that O is



Original System: U demands O/P

Modified System: U copies O/P over $\text{link}_u(O,U)$

(a) For object tickets



Original System: U demands V/P

Modified System: U copies V/P over $\text{link}_u(V^s,U)$

(b) For subject tickets

Fig. 1. Simulation of demand.

passive and cannot exercise the self-tickets it possesses. Moreover, these are the only tickets O will ever possess. To emphasize the limited sense in which O is a subject we call it a pseudosubject. We can then visualize mimicking demand by copy as shown in Fig. 1(a). Let $o/P \in d(u)$, so subject U of type u can demand O/P in the original system. In the modified system we simulate this by authorizing U to copy O/P from dom(O) over $\text{link}_u(O, U)$, i.e., $f_u(o, u) = o/P$. It remains to consider how pseudosubjects acquire self-tickets in the modified system. For objects present in the initial state of the original system, we can introduce these self-tickets in the initial state of the modified system. For objects created subsequently in the original system, self-tickets are introduced by create-rules in the modified system. For this purpose cr_p components of the create-rules for object creation are left unchanged while cr_c components are set to c/R rather than $\emptyset$, thereby introducing self-tickets in every pseudosubject's domain when it is created. In this manner demand for object tickets reduces to self-tickets introduced by creation and copy over the universal link.

Next consider demand of subject tickets. We cannot assume that subjects in the original system possess self-tickets in the modified system. Such self-tickets can result in copy operations not possible in the original system. Also the original subjects are potentially active and can exercise these tickets. Instead we replace each original subject by a pair of subjects in the modified system as shown in Fig. 1(b). Specifically, if V is a subject of type v in the original system we have a pair of subjects V and V^s in the modified system respectively of type v and v^s. V^s is called a shadow subject for the real subject V. We arrange for dom(V^s) to be V/R, i.e., the shadow subject possesses all tickets for the corresponding real subject. Moreover, these are the only tickets a shadow subject will ever possess. Let $v/P \in d(u)$, so subject U of type u can demand V/P in the original system. In the modified system we simulate this by authorizing U to copy V/P from dom(V^s) over $\text{link}_u(V^s, U)$, i.e., $f_u(v^s, u) = v/P$. It is easy to arrange for shadow subjects to possess tickets for their real counterparts. We authorize each real subject to create its shadow subject by $cc(v, v^s)$. The create-rules for creation of shadow subjects have their cr_p component as empty and cr_c as p/R, thereby giving each shadow subject tickets for its real parent. Demand for subject tickets thus reduces to creation of shadow subjects and copy over the universal link.

This simulation is formally summarized as follows.

Step I: Modify the given scheme to have an empty demand function as follows:

(1) $\text{TS}' = \{u, u^s \mid u \in \text{TS}\} \cup \{o \mid o \in \text{TO}\}$, $\text{TO}' = \emptyset$;

(2) $R' = R$;

(3) for all $o \in \text{TO}$: $d'(o) = \emptyset$,
for all $u \in \text{TS}$: $d'(u) = \emptyset$, $d'(u^s) = \emptyset$;

(4) for all $o \in$ TO: $cc'(o) = \emptyset$,
for all $u \in$ TS: $cc'(u) = cc(u) \cup \{u^s\}$, $cc(u^s) = \emptyset$;
(5) for all $u \in$ TS, $o \in$ TO:
$cr'_p(u, o) = cr_p(u, o)$, $cr'_c(u, o) = c/R$,
for all $u \in$ TS:
$cr'_p(u, u^s) = \emptyset$, $cr'_c(u, u^s) = p/R$,
for all $u, v \in$ TS:
$cr'_p(u, v) = cr_p(u, v)$, $cr'_c(u, v) = cr_c(u, v)$;
(6) retain the link predicates of the original scheme,
define $link_u(U, V) \equiv$ **true** if not already defined;
(7) for all $u \in$ TS, $o \in$ TO:
$f'_u(o, u) = \{o/p{:}c \mid o/p{:}c \in d(u)\}$,
for all $u, v \in$ TS:
$f'_u(v^s, u) = \{v/p{:}c \mid v/p{:}c \in d(u)\}$,
for all $u, v \in$ TS and all $link_i$:
$f'_i(u, v) = f_i(u, v)$,
all other values of the filter functions are empty.

Step II: Modify the initial state so that each object O of type o is replaced by a subject O of type o with $dom(O) = O/R$.

Step III: Assume each real subject creates at least one shadow subject. [1]

A formal inductive proof of equivalence between the original and modified systems is easily obtained by mimicking the operations of one in the other along the lines sketched in Fig. 1.

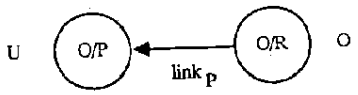
4. Amplification

Amplification is used in several capability-based systems [1,2] to support data abstraction. For example, let O be an object of type queue with dequeue and enqueue operations respectively authorized by d and e rights. Only the type manager for queues can execute read and write operations on O. Other subjects are restricted to possessing $O/de{:}c$ tickets. So if the type manager subject is given O/d or O/e, it is necessary to amplify these to obtain O/rw. Amplification can be viewed as a conditional demand in the following way. Let U and V be of type u and v respectively. We want U to be able to demand V/Q provided U already possesses V/P. It is therefore natural to ask whether the constructions of the previous section can be extended to simulate amplification.

We assume the ability to amplify is independent of the copy flag, i.e., O/d and O/dc are equivalent so far as the ability to amplify them is concerned. This is consistent with the viewpoint that the copy flag plays a role only in the copy operation. We do allow the amplified rights to be copiable. This is useful for layered abstractions. For example, if queues are implemented as lists, the type manager for queues needs to amplify O/d to O/htc where h and t respectively authorize head and tail operations on lists. The O/htc tickets obtained by amplification carry the copy flag so they can be passed on to the type manager for lists, who in turn amplifies O/h or O/t to O/rw. In short, if V/P is amplified to V/Q we require the rights in P to be without the copy flag whereas Q can have rights with or without the copy flag.

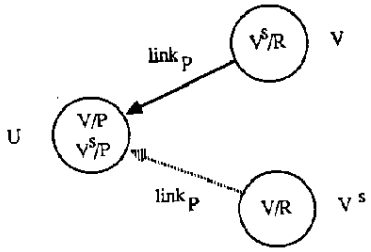
Let us again first consider amplification of object tickets. Recall Fig. 1(a) which shows the simulation of demand by copying self-tickets from a pseudosubject's domain over the universal link. Amplification requires that such copying be permitted only if U already possesses certain tickets for O. The SPM mechanism of links and filter functions can easily model this condition as illustrated in Fig. 2(a). Let U be a subject of type u and O an object of type o in the original system. Suppose that subjects of type u can amplify the P rights for objects of type o to obtain Q rights. The requirement that $O/P \in dom(U)$ is expressed in SPM by defining $link_P(O, U) \equiv O/P \in dom(U)$. The amplification of O/P to O/Q is achieved by authorizing U to copy O/Q from $dom(O)$ over $link_P(O, U)$, i.e., $f_P(o, u) = o/Q$. The similarity between Figs. 1(a) and 2(a) is striking. All we are doing is replacing a copy operation over $link_u(O, U)$ by a copy over $link_P(O, U)$.

[1] In principle, it is a simple matter to arrange for the operating system to create a shadow subject whenever needed. Note that creation of a shadow subject is a local incremental event far simpler than automatic, and possibly widespread, explicit distribution of privileges.



Original System: U amplifies O/P to O/Q
Modified System: U copies O/Q over link$_P$(O,U)

(a) For object tickets



Original System: U amplifies V/P to V/Q

Modified System: U copies V^S/P over link$_P$(V,U)
U copies V/Q over link$_P$(V^S,U)

(b) For subject tickets

Fig. 2. Simulation of amplification.

Next consider amplification of subject tickets. In Fig. 1(b) demand was simulated by copying a real subject's ticket from the domain of its shadow subject over the universal link. Now amplification is predicated on U possessing tickets for the real subject. Specifically, let U and V be subjects of type u and v respectively. Suppose subjects of type u can amplify the P rights for subject of type v to obtain Q rights. The requirement that $V/P \in \mathrm{dom}(U)$ is again expressed by defining $\mathrm{link}_P(V, U) \equiv V/P \in \mathrm{dom}(U)$ indicated by the solid edge in Fig. 2(b). However, V^s, not V, is the source for amplified tickets. To enable U to copy V/Q from V^s we first establish $\mathrm{link}_P(V^s, U)$ as indicated by the broken edge in Fig. 2(b). To achieve this, we modify the create-rules for creation of shadow subjects so as to introduce tickets for the shadow subject in its real parent's domain, i.e., $\mathrm{cr}'_p(v, v^s) = c/R$ rather than $\emptyset$. Amplification is then simulated by two copy operations. First copy V^s/P from V to U over $\mathrm{link}_P(V, U)$ to establish $\mathrm{link}_P(V^s, U)$. Then copy V/Q from V^s to U over $\mathrm{link}_P(V^s, U)$. For this purpose we define f'_P as follows,

$$f'_P(v, u) = \begin{cases} v^s/P \cup f_P(v, u) \\ \quad \text{if } \mathrm{link}_P \text{ was already defined,} \\ v^s/P \quad \text{otherwise;} \end{cases}$$

$$f'_P(v^s, u) = v/P.$$

This simulation works provided u and v are distinct types. However, it fails for peer amplification with $u = v$. In the latter case, because $V^s/R \in \mathrm{dom}(V)$, we have $\mathrm{link}_P(V^s, V)$, so V can copy V/Q from V^s even if $V/P \notin \mathrm{dom}(V)$. In this manner V can acquire self-tickets not obtainable in the original system. Fortunately, amplification is most useful when $u \neq v$. Nevertheless, the failure raises serious questions about the modeling power of SPM. By basing the model so strongly on types, have we lost the ability to do peer amplification? The problem is that V possesses tickets for its shadow subject V^s, but for peer amplification these tickets establish $\mathrm{link}_P(V^s, V)$ with nonempty $f'_P(v^s, v)$. For non-peer amplification there is no problem since $f'_P(v^s, u)$ is nonempty while $f'_P(v^s, v)$ is empty. The impasse arises for $u = v$ because we cannot simultaneously have nonempty $f'_P(v^s, u)$ and empty $f'_P(v^s, v)$.

There does not appear to be any simple way out of this impasse. However it turns out to relate, rather unexpectedly, to a larger question concerning the SPM copy flag. SPM assumes the copy flag is used solely for controlling the copy operation, so $V/pc \in \mathrm{dom}(U)$ implies $V/p \in \mathrm{dom}(U)$. It has been suggested that the copy flag might be redundant [4] in that for every system there is an equivalent system in which all tickets have the copy flag and it is carried along on every copy operation. In this equivalent system we would have the converse property that $V/p \in \mathrm{dom}(U)$ implies $V/pc \in \mathrm{dom}(U)$. If this conjecture is true,

we can simulate peer amplification by non-peer amplification as follows. Let U and V be subjects of type u in the original system and consider peer amplification of V/P to V/Q in U's domain. Define u^a to be a new type of subject whose instances, such as U^a, are called amplification partners of the parent U. The basic idea is that instead of amplifying V/P to V/Q in U's domain we copy V/P from U to U^a, let U^a do the amplification and copy the amplified tickets back to U from U^a. Since $u^a \neq u$, the amplification of V/P by U^a is non-peer amplification which we know how to simulate. The catch is that to copy V/P from U to U^a requires U to possess the tickets $V/Pc \equiv \{V/pc \mid p \in P\}$, whereas for amplification U is only required to possess V/P. Now if the SPM copy flag is redundant, we can replace the original system by an equivalent system in which $V/P \in \mathrm{dom}(U)$ implies $V/Pc \in \mathrm{dom}(U)$. We can then reduce peer amplification of V/P to V/Q in U's domain to non-peer amplification as follows:

(1) Let $k \notin R$ be a new right symbol and define $\mathrm{link}_k(X, Y) \equiv Y/k \in \mathrm{dom}(X)$.

(2) U creates its amplification partner U^a with link_k's from U to U^a and vice versa. This is achieved by $\mathrm{cc}'(u, u^a)$, $\mathrm{cr}'_p(u, u^a) = c/k$ and $\mathrm{cr}'_p(u^a, u) = p/k$.

(3) U copies V/P to U^a over link_k, authorized by $f'_k(u, u^a) = T \times R$. It is this step which requires redundancy of the copy flag.

(4) U^a amplifies V/P to V/Qc. This requires $f'_p(u, u^a) = u^s/P$ and $f'_p(u^s, u^a) = u/Qc$. The crucial point is $f'_p(u^s, u) = \emptyset$, so U or V can no longer copy their self-tickets from U^s or V^s respectively.

(5) Finally, U copies V/Q from U^a over link_k, authorized by $f'_k(u^a, u) = T \times R$.

To summarize, we have shown that peer amplification can be reduced to non-peer amplification provided the SPM copy flag is redundant.

5. Conclusion

The demand operation in SPM is theoretically redundant since it can be simulated by copy and create operations. To a large extent amplification (or conditional demand) can be simulated by similar constructions. It is noteworthy that these constructions do not introduce cycles in can-create. Therefore, amplification is achieved without compromising safety analysis [8]. For peer amplification however, our construction works only if the SPM copy flag is redundant. Redundancy of the SPM copy flag is an important open question in its own right. Its significance is enhanced by this unexpected connection to peer amplification.

Acknowledgment

I would like to express my sincere thanks to both referees for their valuable comments.

References

[1] E. Cohen and D. Jefferson, Protection in the Hydra Operating System, in: *Proc. 5th ACM Symp. on Operating Systems Principles* (1975) 141–160.

[2] H.M. Levy, *Capability-Based Computer Systems* (Digital Press, Belford, MA, 1984).

[3] R.J. Lipton and L. Snyder, A linear time algorithm for deciding subject security, *J. ACM* **24** (1977) 455–464.

[4] J.K. Millen, Private communication.

[5] N. Minsky, Selective and locally controlled transport of privileges, *ACM Trans. Programming Lang. Systems* **6** (1984) 573–602.

[6] R.S. Sandhu, Design and analysis of protection schemes based on the send–receive transport mechanism, Ph.D. Thesis, Department of Computer Science, Rutgers University (1983).

[7] R.S. Sandhu and M.E. Share, Some owner-based schemes with dynamic groups in the schematic protection model, in: *Proc. IEEE Symp. on Security and Privacy* (1986) 61–70.

[8] R.S. Sandhu, The schematic protection model: its definition and analysis for acyclic attenuating schemes, *J. ACM* **35** (1988) 404–432.