Password management: distribution, review and revocation

Lanfranco Lopriore
Dipartimento di Ingegneria dell'Informazione, Università di Pisa, via G. Caruso 16, 56126 Pisa, Italy
E-mail: l.lopriore@iet.unipi.it

Abstract — We consider the problem of access privilege management in a classical protection environment featuring subjects attempting to access the protected objects. We express an access privilege in terms of an access right and a privilege level. The privilege level and a protection diagram associated with each given object determine whether a nominal access privilege for this object corresponds to an effective, possibly weaker access privilege, or is revoked. We associate a password system with each object; the password system takes the form of a hierarchical bidimensional one-way chain. A subject possesses a nominal access privilege for a given object if it holds a key that matches one of the passwords in the password system of this object; the protection diagram determines the extent of the corresponding effective access privilege.

The resulting protection environment has several interesting properties. A key reduction mechanism allows a subject that holds a key for a given object to distribute keys for weaker access rights at lower privilege levels. A subject that owns a given object can review or revoke the passwords for this object by simply modifying the protection diagram. The memory requirements to represent a protection diagram are negligible; as far as password storage is concerned, space-time trade-offs are possible.

Keywords: access privilege, one-way chain, one-way function, password, review, revocation.

1. INTRODUCTION

In a classical protection model, a set U of active entities, called *subjects*, attempt to access a set O of passive entities, the protected *objects* [1]. Objects are typed, and the type T of a given object G states the set $V = \{vl_0, vl_1, \ldots, vl_{v-1}\}$ of the values that the object can assume, the set $P = \{op_0, op_1, \ldots, op_{p-1}\}$ of the operations that can be applied to these values, and a set $R = \{ar_0, ar_1, \ldots, ar_{r-1}\}$ of access rights, where quantities v, p and r are specific to T. The configuration of the protection system is a triple (U, O, M) where M is an *access matrix* featuring a row for every subject in U and a column for every object in O [2]. Element M[S, G] of the access matrix specifies the access rights held by subject S on object G. Typical subjects might be processes, and typical objects might be files. Examples of access rights for files are *own*, *read*, *write*, and *execute*. For each operation in P, the definition of type T states the access rights in R that are necessary to execute this operation successfully. This means that each operation includes a test for the presence of specific access rights in specific positions of the access matrix. If this test is successful, the actions involved in the execution of the operation can be accomplished, otherwise an exception of violated protection is raised and execution of the operation fails. The owner of object G (i.e. a subject that possesses the *own* access right for G) is allowed to modify the column of the access matrix corresponding to G, e.g. to revoke the access rights held by the other subjects.[1]

1.1. Access privilege specification

In a system of this type, a basic problem is to implement the access matrix, i.e. to specify the relations existing between subjects and objects by defining the access rights that each subject holds on the objects [3]. A classical solution is based on the concept of a *capability* [4]. This is a pair (G, AR), where G is the name of an object and AR is an access privilege expressed in terms of a set of access rights. A subject that holds capability (G, AR) can access object G to perform the operations permitted by the access rights in AR.

Capabilities should be segregated in memory, so that their internal representation is inaccessible to subjects [5]. This is necessary to prevent a subject from tampering with an existing capability to amplify the access rights it contains, or even forging a capability from scratch. Several solutions have been devised to the capability segregation problem. Special memory segments, which we shall call the capability segments, can be reserved for capability storage (in contrast, data segments will be reserved for storage of ordinary information items) [6], [7], [8]. This solution leads to segment proliferation. The resulting programming paradigm is often unnatural, e.g. the representation of a data structure must include a capability segment storing the capabilities for the data segments. Alternatively, capability segregation can be obtained by taking advantage of a tagged memory system that associates a 1-bit tag with each memory cell; the value of the tag of a given cell indicates whether this cell contains a capability or an ordinary information item [9], [10]. In this approach, the instruction set of the processor includes special instructions for capability processing; if one of these instructions is executed on a memory cell tagged to contain an ordinary information item, an exception of violated protection is raised and execution fails. Contrary to hardware standardization [11], ad-hoc memory modules are necessary to support cell tags (e.g. a 65-bit cell is used to store a 64-bit memory word).

In a different approach, a set of *passwords* is associated with each protected object, and each password corresponds to a subset of all access rights [12], [13], [14], [15], [16]. In order to access an object to perform a given operation, a subject presents a *key* for this object. The access is successful only if the key matches one of the passwords associated with the object,

[1] A subject may also be a protected object. For instance, a process that creates a child process will receive the *own* access right for the child process. As a result, the parent process will be entitled to kill the child process and to control its properties, e.g. priority. If in turn the child process creates new processes, the protection system may include mechanisms for the automatic inheritance of the ownership of these processes by the parent process. In an alternative protection system design, the child process will have to transfer ownership of the new processes to the parent process by explicit actions of access right copy.

and this password includes the access right required by that operation; if this is not the case, an exception of violated protection is raised and the object access fails.

Keys are protected from forging by the password length; if passwords are large and randomly distributed, the probability of guessing a valid key is vanishingly low [17]. (Random distribution of passwords is an essential requirement; if passwords are not random, the password length may not be a determining factor to prevent key forging.) Thus, keys do not need to be segregated in memory; instead, they can be freely mixed with ordinary data. This simplifies software composition in memory. Furthermore, no need for special hardware is connected with the necessity to store cell tags.

1.2. Access right distribution

A subject S that holds a set of access rights for a given object G should be in a position to distribute these access rights, in full or in part, to other subjects. In a capability environment, a result of this type is obtained by a simple action of a capability copy (and indeed, simplicity in access right distribution was one of the original motivations for the introduction of the capability concept [4]).

In a password-based protection system, subject S that possesses a key *k* for object G can distribute the access privilege corresponding to this key to another subject by simply transmitting a copy of the key. A related problem is that of *key reduction*. Subject S should be given the ability to transmit only a subset of the access rights corresponding to key *k*. This means that a mechanism of the protection system should allow S to convert *k* into a weaker key *k'*. The key reduction problem can be easily solved by associating an object manager with each given object. In a situation of this type, subject S transmits key *k* to the object manager that verifies the validity of *k* and then generates a key *k'* matching a weaker password; *k'* is returned to S. Of course, the presence of an object manager tends to complicate the overall object structure, and is contrary to a main requirement of a protection system design, i.e. simplicity in access right management.

1.3. Access right review and revocation

A subject that granted an access privilege to another subject should be allowed to revise the grant and revoke the access privilege from the recipient. Revocation should extend to all the subjects that received the privilege being revoked from the first recipient, recursively. Revocation should be reversible, so that after a given access privilege has been revoked, it should be possible to confirm this access privilege and restore its validity.

Several solutions to the revocation problem have been proposed with reference to capability-based protection environments [18]. These include a reference monitor associated with the given object that manages the access permissions held by all subjects on this object [19], [20]; a propagation graph for each capability that keeps track of all successive transferals of this capability throughout the system [21]; and short-lived capabilities, whose validity must be renewed periodically, and are implicitly revoked if renewal is lacking [22]. These solutions tend to subvert the main advantage of the capability-based protection model, i.e. simplicity in access right transmission between subjects [23].

In a password-based protection environment, a subject that receives a key is free to transmit this key further, and this key diffusion process may well extend to any transition depth. This means that access rights tend to propagate throughout the system. A simple solution to the revocation problem is to modify one or more passwords associated with the given object, so that the corresponding keys lose their validity. This solution does not meet the requirement to limit revocation to a subset of the access rights associated with the given password.

This paper presents a model of a password management system that has been designed by taking the following requirements into account:

- A simple mechanism for access privilege distribution should permit effective forms of key reduction, so that a subject that holds a given access privilege can transmit this privilege only in part.
- Password review and revocation should be fully supported, so that the extent of a given password can be restricted to a subset of the original access rights. The consequence of a password review should extend to all keys matching the reviewed password. It should be possible to reverse the effect of a password review and restore full password validity.
- The memory requirements for storage of the passwords and the information items related to password management should be kept to a minimum.

The rest of this paper is organized as follows. Section 2 presents our protection model with special reference to access privileges and passwords. The concept of a protection diagram is introduced expressing a relation between the nominal access privilege connected with a password and the effective access privilege granted by possession of a key matching that password. Section 3 presents the password system with special reference to password generation and management. Section 4 discusses the protection model from a number of salient viewpoints, which include the verification, review and revocation of access rights, and the memory requirements for storage of the passwords and the protection diagram. Section 5 gives concluding remarks.

2. THE PROTECTION MODEL

Let T be an object type, and let op_0, op_1, … be the operations that can be applied to the objects of type T. Furthermore, let ar_0, ar_1, …, ar_{r-1} be the access rights, and let pl_0, pl_1, …, pl_{c-1} be the *privilege levels* defined by T, where quantities r and c are specific to T. Privilege levels are involved in password validation, review and revocation, as will be illustrated shortly.

The access rights are ordered from the weakest ar_0 to the strongest ar_{r-1}. Possession of access right ar_i implies possession of every weaker access right $ar_{i'}$, $i' < i$. Similarly, the privilege levels are ordered from the lowest pl_0 to the highest pl_{c-1}.

An *access privilege* for an object G of type T is the specification of an access right and a privilege level. Let $P_{nom} = (ar_i, pl_j)$, $0 \leq i \leq r-1$, $0 \leq j \leq c-1$, be a *nominal* access privilege expressed in terms of access right ar_i and privilege level pl_j. P_{nom} grants its holder an *effective* access privilege $P_{eff} = (ar^*, pl_j)$ expressed in terms of access right ar^* and the same privilege level pl_j, where ar^* is not stronger than ar_i. The actual extent of ar^* is stated for object G by a relation existing between the access rights and the privilege levels. This relation is specific to G, and is expressed by a diagram, which is associated with G and is called the *protection diagram* of G.

The protection diagram is defined in a two-dimensional Cartesian coordinate system. In the protection diagram, the horizontal axis refers to the access rights, from ar_0 to ar_{r-1}, and the vertical axis refers to the privilege levels, from pl_0 to pl_{c-1}. The *protection line* is a polygonal chain whose vertices are access privileges that we call the *limit privileges*, with the constraint that at least one limit privilege must be associated with each access right. Figure 1 shows examples of protection diagrams featuring four access rights, ar_0 to ar_3, and five privilege levels, pl_0 to pl_4. The protection line divides the protection diagram into two regions: a *validity region* that includes the protection line and corresponds to higher privilege levels, and a *downgrade region* that corresponds to lower privilege levels.

In the example of Figure 1a, the limit privileges are (ar_0, pl_1), (ar_1, pl_2), (ar_2, pl_2), and (ar_3, pl_4). The protection line connects the corresponding points of the protection diagram. The validity region occupies the upper-left part of the diagram and includes the protection line; the downgrade region occupies the lower-right part. In Figure 1b, the limit privileges are (ar_0, pl_4), (ar_1, pl_3), (ar_1, pl_2), (ar_2, pl_2), and (ar_3, pl_1). The validity region occupies the upper-right part of the diagram and includes the protection line; the downgrade region occupies the lower-left part. In Figure 1c, the protection line is horizontal and connects limit privileges (ar_0, pl_4), (ar_1, pl_4), (ar_2, pl_4), and (ar_3, pl_4). The validity region degenerates into the protection line. Finally,

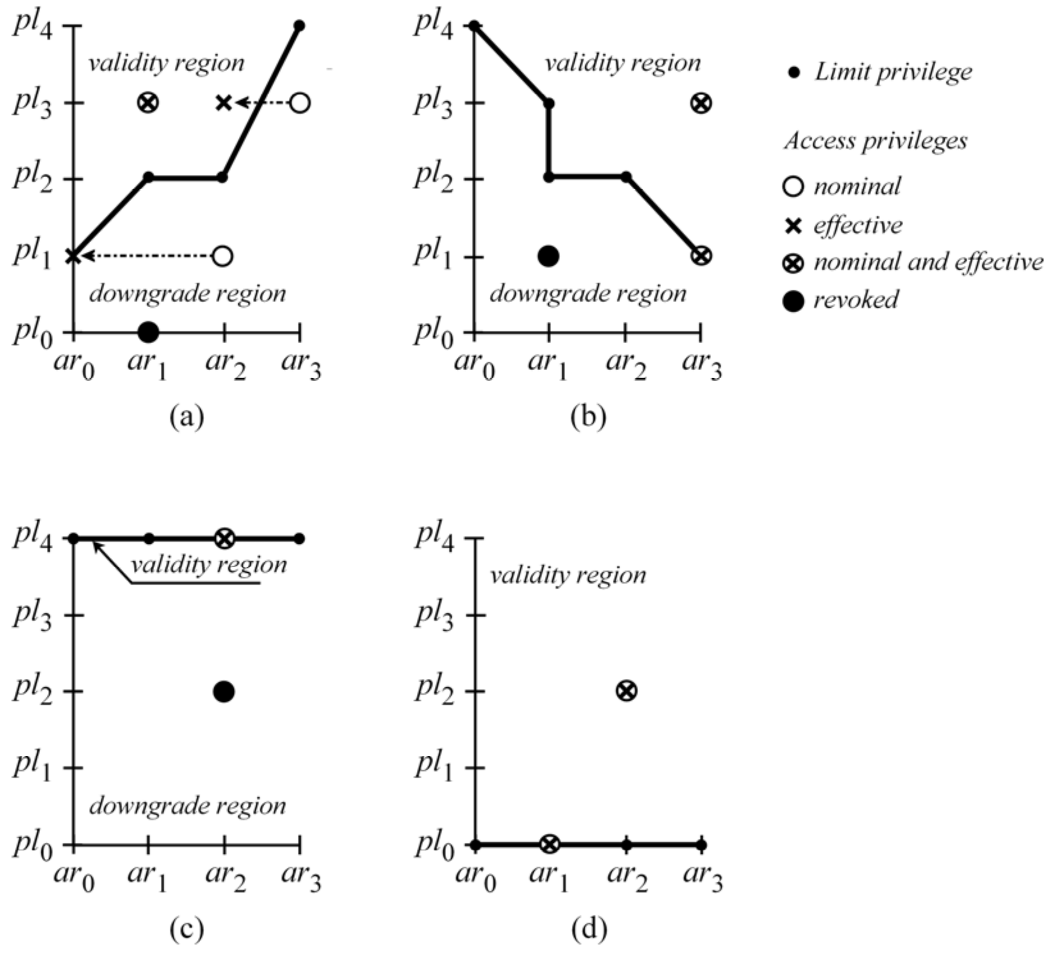

Figure 1. Protection diagrams featuring four access rights and five privilege levels, in a variety of configurations of the protection line.

in Figure 1d the protection line is horizontal and connects limit privileges (ar_0, pl_0), (ar_1, pl_0), (ar_2, pl_0), and (ar_3, pl_0). In this case, the validity region occupies the whole protection diagram.

For the given nominal access privilege, the protection line and the two protection regions determine the corresponding effective privilege. Let us refer again to nominal privilege P_{nom} = (ar_i, pl_j) and the corresponding effective privilege P_{eff} = (ar^*, pl_j). If P_{nom} is in the validity region (including the protection line), nominal access right ar_i is effective, that is, $ar^* = ar_i$ and consequently $P_{eff} = P_{nom}$; a subject S that possesses P_{nom} is granted access right ar_i. If P_{nom} is in the downgrade region, access right ar_i is downgraded to the strongest access right ar^* at protection level pl_j that is weaker than ar_i and is included in the validity region. If ar_i cannot be downgraded into the validity region, then access privilege P_{nom} grants no access right at all; in a situation of this type, we say that P_{nom} is *revoked*.

In the example of Figure 1a, nominal access privilege (ar_1, pl_3) is in the validity region, and is effective; a subject that possesses this access privilege is granted access right ar_1. Nominal privilege (ar_3, pl_3) is in the downgrade region, and the corresponding effective privilege is

(ar_2, pl_3); a subject that possesses (ar_3, pl_3) is granted access right ar_2. Similarly, nominal privilege (ar_2, pl_1) is in the downgrade region, and the corresponding effective privilege is (ar_0, pl_1). Finally, nominal privilege (ar_1, pl_0) is in the downgrade region and cannot be downgraded into the validity region, so it is revoked (this access privilege grants no access right at all). In the example of Figure 1b, nominal privileges (ar_3, pl_3) and (ar_3, pl_1) are in the validity region, and are effective. Nominal privilege (ar_1, pl_1) cannot be downgraded into the validity region, and is revoked. In the example of Figure 1c, all nominal privileges are revoked except those defined in terms of the highest privilege level, pl_4, which are effective. Finally, in the example of Figure 1d, the validity region occupies the entire protection diagram, so all nominal privileges are effective.

In every given object type, the strongest access right ar_{r-1} is called the *own* access right; possession of this access right for a given object makes it possible to delete the object and to modify its protection line (that is, to modify the position of the limit privileges in the protection diagram).[2] Access privilege (ar_{r-1}, pl_{c-1}) corresponding to the *own* access right at the highest privilege level is called the *owner privilege*. A subject that creates a new object (the *object owner*) is granted the owner privilege for this object. When an object is created, a protection diagram is associated with that object, and the shape of the protection line is determined as part of the object initialization.

An important property is that, in the protection diagram, all privileges at privilege level pl_{c-1} are always in the validity region; it follows that the owner privilege is always effective. Consequently, a subject that creates an object is always in a position to delete this object and to modify its protection line; as will be shown shortly, an action of this type corresponds to a form of review or revocation of access privileges.

3. THE PASSWORD SYSTEM

Let us refer again to type T, where ar_0, ar_1, …, ar_{r-1} are the access rights, pl_0, pl_1, …, pl_{c-1} are the privilege levels, and P_{nom} = (ar_i, pl_j), $0 \le i \le r-1$, $0 \le j \le c-1$, are the nominal access privileges. We shall refer to the generic operation *op* defined by T, ar_{op} being the access right necessary to execute this operation successfully. In our protection model, for each object G of type T, a password system is associated with G featuring a password for each nominal access privilege, for a total of $r \cdot c$ passwords. In order to access G to execute operation *op* successfully,

[2] In an alternative design, the two actions, to modify the protection line and to delete the object, are kept separated, and correspond to different access rights, namely *own* and *delete*. This can be useful if one wants to propagate the ability to delete the object without being allowed to modify its protection line, for instance.

$$\begin{tikzcd}
w_{0,c\text{-}1} & \cdots \arrow[l] & w_{i,c\text{-}1} \arrow[l] & \cdots \arrow[l] & w_{r\text{-}2,c\text{-}1} \arrow[l] & w_{r\text{-}1,c\text{-}1} \arrow[l] \arrow[d] \\
w_{0,c\text{-}2} & \cdots \arrow[l] & w_{i,c\text{-}2} \arrow[l] & \cdots \arrow[l] & w_{r\text{-}2,c\text{-}2} \arrow[l] & w_{r\text{-}1,c\text{-}2} \arrow[l] \arrow[d] \\
 & & & & & \cdots \arrow[d] \\
w_{0,j} & \cdots \arrow[l] & w_{i,j} \arrow[l] & \cdots \arrow[l] & w_{r\text{-}2,j} \arrow[l] & w_{r\text{-}1,j} \arrow[l] \arrow[d] \\
 & & & & & \cdots \arrow[d] \\
w_{0,0} & \cdots \arrow[l] & w_{i,0} \arrow[l] & \cdots \arrow[l] & w_{r\text{-}2,0} \arrow[l] & w_{r\text{-}1,0} \arrow[l]
\end{tikzcd}$$

Figure 2. The password system featuring a primary chain whose seed is the password $w_{r\text{-}1,c\text{-}1}$ of the owner privilege, and a secondary chain for each privilege level, whose seed is the corresponding element of the primary chain.

a given subject S must present a key k matching a password in the password system, say password $w_{i,j}$. Implementation of operation *op* includes the actions necessary to determine the effective access privilege $P_{\text{eff}} = (ar^*, pl_j)$ corresponding to the nominal access privilege $P_{\text{nom}} = (ar_i, pl_j)$ associated with $w_{i,j}$; to this aim, *op* uses the protection diagram of G. Execution of operation *op* will terminate successfully only if ar^* is stronger than or equal to access right ar_{op}; if this is not the case, execution terminates with failure.

A related problem is the generation of the set of passwords for object G. We solve this problem by taking advantage of one-way chains.

3.1. Password generation

Function F is *one-way* if it is easy to compute but hard to invert [24]. This means that given a value x it is computationally easy to compute F(x), but given a value y it is computationally unfeasible to find a value x such that y = F(x). We shall denote i successive applications of F by F^i, e.g. $F^2(x) = F(F(x))$. A *one-way chain* [25] is a collection of values $(v_0, v_1, \ldots, v_{n\text{-}1})$ such that each v_i except the last value $v_{n\text{-}1}$ is the result of applying a one-way function F to the next value, i.e. $v_i = F(v_{i+1}) = F^{n-1-i}(v_{n\text{-}1})$ for $0 \leq i < n - 1$. Value $v_{n\text{-}1}$ is called the *seed* of the chain. A hierarchical bidimensional one-way chain consists of two levels of chains, where each value of the chain at the first level (the *primary* chain) is the seed of a chain at the second level (a *secondary* chain).

In our model, the password system of object G of type T is configured as a hierarchical bidimensional one-way chain (Figure 2). To set up the password system, password $w_{r\text{-}1,c\text{-}1}$, corresponding to owner privilege $(ar_{r\text{-}1}, pl_{c\text{-}1})$, is chosen at random and is the seed of the primary chain. A one-way function PF, called the *primary function*, is used to generate the successive passwords in the primary chain, which are associated with access right $ar_{r\text{-}1}$ in the direction of decreasing privilege levels, i.e. $w_{r\text{-}1,c\text{-}2} = \mathrm{PF}(w_{r\text{-}1,c\text{-}1})$, $w_{r\text{-}1,c\text{-}3} = \mathrm{PF}(w_{r\text{-}1,c\text{-}2}) = \mathrm{PF}^2(w_{r\text{-}1,c\text{-}1})$, …, $w_{r\text{-}1,0} = \mathrm{PF}(w_{r\text{-}1,1}) = \mathrm{PF}^{c\text{-}1}(w_{r\text{-}1,c\text{-}1})$.

Each given password $w_{r\text{-}1,j}$ of the primary chain is the seed of the secondary chain associated with the corresponding privilege level, pl_j. The secondary chain features r values, one value for each access right. A further one-way function, the *secondary function* SF, is used to setup the secondary chains, so that for privilege level pl_j we have $w_{r\text{-}2,j} = \mathrm{SF}(w_{r\text{-}1,j})$, $w_{r\text{-}3,j} = \mathrm{SF}(w_{r\text{-}2,j}) = \mathrm{SF}^2(w_{r\text{-}1,j})$, …, $w_{0,j} = \mathrm{SF}(w_{1,j}) = \mathrm{SF}^{r\text{-}1}(w_{r\text{-}1,j})$.

As pointed out in Section 1.1, the fact that passwords are generated at random is essential to prevent password forging. In our password system, randomness is a salient requirement for password $w_{r\text{-}1,c\text{-}1}$, and is automatically extended to all the passwords in the password system by the mechanism of the password chains.

3.2. Password distribution

As seen in Section 1, the problem of access privilege distribution consists in allowing a subject that holds an access privilege for a given object to grant this access privilege to a different subject. A salient feature of password-based protection systems is simplicity in access privilege distribution. In a system of this type, a subject S that holds a key k for a given object is in a position to distribute the corresponding access privilege to another subject S’, by simply transmitting a copy of k to S’.

In our protection system, subject S may even distribute an access privilege weaker than the privilege in the original key. *Key reduction* is the action of transforming a key for a given access privilege into a key for a reduced privilege. Key reduction can take place in the direction of a weaker access right, a lower privilege level, or both. Key reduction is obtained by taking advantage of the structure of the password system and the password chains. In detail:

- A subject that holds key k matching password $w_{r\text{-}1,j}$ for access privilege $(ar_{r\text{-}1}, pl_j)$ expressed in terms of the strongest access right $ar_{r\text{-}1}$ and privilege level pl_j, can derive a key k' matching password $w_{r\text{-}1,j'}$ for access privilege $(ar_{r\text{-}1}, pl_{j'})$ expressed in terms of the same access right $ar_{r\text{-}1}$ and a lower privilege level $pl_{j'}$, $j' < j$. To this aim, the subject will apply primary one-way function PF iteratively $j - j'$ times, i.e. $k' = \mathrm{PF}^{j\text{-}j'}(k)$. This is a consequence of the fact that $w_{r\text{-}1,j'} = \mathrm{PF}^{j\text{-}j'}(w_{r\text{-}1,j})$.
- A subject that holds key k matching password $w_{i,j}$ for access privilege (ar_i, pl_j) expressed in terms of access right ar_i and privilege level pl_j can derive a key k' matching password $w_{i',j}$ for access privilege $(ar_{i'}, pl_j)$ expressed in terms of a weaker access right $ar_{i'}$, $i' < i$, and the same privilege level pl_j. To this aim, the subject will apply secondary one-way

function SF iteratively $i - i'$ times, i.e. $k' = \mathrm{SF}^{i-i'}(k)$. This is a consequence of the fact that $w_{i',j} = \mathrm{SF}^{i-i'}(w_{i,j})$.

3.3. Password review and revocation

As a consequence of the inherent simplicity of key distribution, in a password-based protection environment access privileges tend to spread throughout the system. A related problem is to allow the owner of a given object to control the extent of the effective access privilege granted by each password on that object, in contrast with the nominal access privilege corresponding to that password (password *review*). The object owner should be even in a position to revoke the nominal access privilege, so that the password can no longer be used for successful object access (password *revocation*).

In our design, the object owner can change the position of the limit privileges in the protection diagram, thereby modifying the protection line and the configuration of the validity region and the downgrade region. So doing, the object owner can enforce specific access privilege management strategies, by reviewing or revoking each given password, independently of the subjects that hold this password. In the following, we shall say that a password is effective, downgraded or revoked if this is the case for the corresponding nominal access privilege.

For instance, in the protection diagram of Figure 1d, the protection line is horizontal and is placed at privilege level 0. Consequently, all nominal access privileges, and the corresponding passwords, are effective. In a situation of this type, a subject that holds a key matching a password corresponding to a given access privilege can exercise the full access right included in this access privilege. By moving the protection line to a high privilege level, e.g. pl_4 (see Figure 1c), the object owner revokes all the passwords corresponding to nominal access privileges at lower privilege levels. A subject that holds a key matching one of these passwords cannot use this key for successful object access. If the protection line is that of Figure 1a, high privilege levels are necessary to exercise strong access rights. For instance, access privilege (ar_3, pl_3) is below the protection line. This access privilege is downgraded to (ar_2, pl_3); a subject that holds a key matching password $w_{3,3}$ can use this key to execute a given operation only if this operation requires access right ar_2 or lower. Finally, if the protection line is configured as shown in Figure 1b, the access privileges defined in terms of strong access rights are effective even at low privilege levels, e.g. (ar_3, pl_1) and the corresponding password $w_{3,1}$.

4. DISCUSSION

4.1. Access right verification

Let us refer again to object G of type T, let *op* be an operation defined by T, and let ar_{op} be the access right required for successful execution of *op*. The password system of G is stored as part of the object internal representation. Now suppose that subject S tries to execute operation *op* on G. To this aim, S presents a key *k* to *op*. The actions involved in the execution of *op* include key verification, which will be conceptually structured as follows:

1. Key *k* is compared with the passwords in the password system of object G. If no matching password is found, execution of operation *op* terminates with failure; otherwise, let $w_{i,j}$ be the matching password, and let $P_{nom} = (ar_i, pl_j)$ be the corresponding nominal access privilege, which is expressed in terms of access right ar_i and privilege level pl_j.
2. The protection diagram of object G is used to determine whether P_{nom} is revoked. If this is the case, execution of operation *op* terminates with failure; otherwise, let $P_{eff} = (ar^*, pl_j)$ be the effective access privilege corresponding to P_{nom}.
3. Access right ar^* is compared with access right ar_{op} (it should be recalled that access rights are ordered hierarchically, so that every given access right includes all weaker access rights). If $ar^* < ar_{op}$, execution of *op* terminates with failure; otherwise, access right verification is successful and the actions involved in the execution of *op* are subsequently accomplished.

4.2. Password review and revocation

As seen in Section 3.3, in our protection system a subject that owns a given object can review or revoke every password defined for that object by simply modifying the shape of the protection line. This mechanism results to possess a number of interesting properties:

- Password review can be limited to a subset of all passwords. This is an inherent property of the protection diagram. For instance, in the protection diagram of Figure 1a, access privilege (ar_3, pl_3) is downgraded to (ar_2, pl_3); this means that password $w_{3,3}$ grants access right ar_2 instead of the nominal ar_3. In contrast, access privilege (ar_1, pl_3) is effective, and the extent of password $w_{1,3}$ is unaltered; this password grants access right ar_1.
- Similarly, password revocation can be limited to a subset of all passwords. In the example of Figure 1b, all the access privileges in the downgrade region are revoked, and consequently, the corresponding passwords cannot be used for successful object access, e.g. password $w_{1,1}$ corresponding to access privilege (ar_1, pl_1). On the other hand, the passwords for the access privileges in the validity region maintain their access right strength, e.g.

password $w_{3,3}$ corresponding to access privilege (ar_3, pl_3).

- Two or more passwords corresponding to the same access right at different privilege levels can be revoked independently of each other. In the example of Figure 1a, password $w_{1,3}$ corresponding to access privilege (ar_1, pl_3) for access right ar_1 at privilege level pl_3 is in the validity region and is effective, whereas password $w_{1,0}$ corresponding to access privilege (ar_1, pl_0) for the same access right ar_1 at privilege level pl_0 is in the downgrade region and is revoked.
- The effects of a review or revocation are *transitive* [21], that is, they propagate to all the keys matching a given password independently of the subjects that hold these keys and the propagation paths followed by the keys to reach these subjects. If a given password is revoked or downgraded to a given extent, all keys matching this password are automatically revoked or downgraded to the same extent. In fact, a copy of a given key is indistinguishable from the original, and keys have no memory of the consecutive copy actions.
- The effects of a review or revocation are *temporal* [21], that is, they can be reversed through the same mechanism used for the revocation. With reference to Figure 1c, password $w_{2,2}$ corresponds to a nominal access privilege in the downgrade region, and is revoked. If the protection line is moved to privilege level pl_2, for instance, the nominal access privilege will be in the validity region, and the validity of $w_{2,2}$ will be restored to its full extent.[3]

In Table 1, we compare the different implementations of the access matrix, which have been introduced in Section 1: capabilities, passwords and the password systems proposed in this paper. A number of salient properties of these implementations are considered: segregation (i.e. the method used to prevent forging), reduction (i.e. the ability to transmit only a subset of the access rights), review and revocation. As seen in Section 1, in capability systems, segregation can be obtained by taking advantage of capability segments or tagged storage; whereas a suitable password length is sufficient to prevent password forging if passwords are randomly distributed. In their original formulation, capabilities cannot be reviewed; whereas the extent of a password can be modified by changing the set of access rights associated with that password.

[3] The merits of temporal revocation are debatable. Consider a key that has been distributed by subject S to a recipient subject S', and suppose that S' distributes this key further. Revocation involves all the copies of the original key, independently of their present location in memory and the distribution path followed by each of them. Again, this is a consequence of the fact that keys have no memory of the consecutive copy actions, and a key copy is indistinguishable from the original. Now suppose that subject S decides to restore validity of the key distributed to S'. So doing, it may well be the case that S unintentionally restores an unknown key copy resulting from recursive propagation. Of course, an alternative is to carry out a new action of key distribution to S', e.g. for a key for the same access right at a different privilege level, corresponding to a password in the validity region. Indeed, the main merit of temporal revocation is to avoid the cost of this type of repeated activities of key distribution [21].

Table 1. Properties of different implementations of the access matrix.

Capabilities
- *Segregation*: capability segments; tagged storage
- *Reduction*: modification of the access right field
- *Review*: not supported
- *Revocation*: reference monitor; propagation graph; short-lived capabilities; etc.

Passwords
- *Segregation*: password length and random distribution
- *Reduction*: object manager
- *Review*: modification of the corresponding set of access rights
- *Revocation*: password modification

Password systems
- *Segregation*: password length and random distribution
- *Reduction*: application of the one-way functions
- *Review*: modification of the protection diagram
- *Revocation*: modification of the protection diagram

Password reduction requires the intervention of an object manager. In contrast, in a password system, reduction and review are supported natively by one-way functions and the protection diagram. Finally, as far as revocation is concerned, complicated ad-hoc mechanisms have been proposed for capability systems, and a password replacement can be used for password invalidation; in a password system, similar effects can be obtained by simply modifying the shape of the protection line in the protection diagram.

4.3. Considerations concerning performance

The protection line

As seen in Section 2, the shape of the protection line is completely determined by the coordinates of the limit privileges; at least one, and at most two limit privileges correspond to each access right (two limit privileges are necessary to delimit a vertical protection line segment). In a possible representation for up to 16 privilege levels, we shall reserve one byte for each access right. The two 4-bit nibbles of the *i*-th byte encode the privilege levels of the two limit privileges for the *i*-th access right; if only one limit privilege corresponds to a given access right, the two nibbles will specify the same privilege level.

Let us refer to a protection diagram featuring four access rights, ar_0 to ar_3, and five privilege levels, pl_0 to pl_4; in this case, the shape of the protection line can be encoded in 4 bytes. Examples are given in Figures 1a to 1d; in these examples, the corresponding hexadecimal configurations are 11 22 22 44, 44 32 22 11, 44 44 44 44 and 00 00 00 00, respectively. We may conclude that the memory requirements to represent a protection line are negligible.

As seen in Section 3.3, access right revocation for a given object is simply obtained by modifying the shape of the protection line of this object. The cost of the revocation process is that of determining of the new position of each limit privilege in the protection diagram and expressing this new position in terms of the chosen form of protection line representation. This cost will be paid for each object involved in the revocation. Let us refer, for instance, to the protection line of Figure 1d, and suppose that we are aimed at revoking all access rights at privilege levels lower than pl_4. The resulting protection line is that of Figure 1c. In our representation of the protection line, a result of this type will be obtained by replacing hexadecimal quantity 00 00 00 00 with hexadecimal quantity 44 44 44 44.

Passwords

The passwords in the password system of a given object will be evaluated when this object is initialized. These passwords will be stored in a *password array* that is part of the internal representation of the object. As seen in Section 4.1, the password system is used when an action of access right verification takes place, to find the password $w_{i,j}$ matching a given key k. To this aim, a sequential search in the password array will check the array elements until the matching password is found. The expected number of comparisons is $(r \cdot c + 1) / 2$, r and c being the number of access rights and the number of privilege levels, respectively.

In an alternative approach, a key has the form of a triple (k, i, j), that is, key value k and the indexes i and j of the matching password $w_{i,j}$. In this case, a single comparison is necessary in the password array, at element $(r \cdot i + j)$ (here, we hypothesize that passwords are stored in the password array by privilege levels); if the value of this element does not match quantity k, key verification fails. The memory requirements for storage of quantities i and j are negligible, e.g. a single byte for up to 16 access rights and 16 privilege levels.

If memory is an extremely scarce resource, we shall store a single password instead of the entire password system, i.e. password $w_{r-1,c-1}$ corresponding to the seed of the primary chain (see Section 3.1 and Figure 2). In this case, verification of the validity of a given key requires recalculation of several passwords, which will be accomplished by using primary one-way function PF and secondary one-way function SF. Specifically, we shall use PF to evaluate passwords $w_{r-1,c-2}$, $w_{r-1,c-3}$, …, to find the seed $w_{r-1,j}$ of the j-th secondary chain; then, we shall use SF to evaluate passwords $w_{r-2,j}$, $w_{r-3,j}$, …, to find the value of password $w_{i,j}$. The expected number of password evaluations is $(c + r) / 2$.

A space-time trade-off is possible, by using the password array to permanently store only the passwords $w_{r-1,c-1}$, $w_{r-1,c-2}$, …, $w_{r-1,0}$ that form the primary chain of the password system. In this case, verification of the validity of a key matching password $w_{i,j}$ will require recalculation of the first $r - i - 1$ passwords in the j-th secondary chain, whose seed is password $w_{r-1,j}$ in the primary chain. In this case, the expected number of password evaluations decreases to r / 2.

One-way functions

The total number of one-way functions is a parameter of the system design. If a single (PF, SF) pair of one-way functions is used throughout the system (see Section 3.1), all the subjects know these functions, and a subject that holds a given key is always in a position to reduce this key. This means that when a subject receives a key, it can transform this key into a key for a reduced privilege, and distribute the new key instead of the original key. Alternatively, we can have several one-way functions, e.g. one (PF, SF) pair for each subject (in a protection system where subjects are processes, this means that each process has its own one-way functions). In this case, when a given subject S creates a new object, the password system of this object is generated by using the one-way functions of this subject. If S distributes a key for the new object, the recipient subject S' will not be able to reduce the key, either in the direction of a weaker access right or in the direction of a lower privilege level, as it does not possess the original one-way functions. Implications follow in terms of access right review. In particular, the original object owner S has tighter control over revocation, as follows from the fact that only those keys originally distributed by S will exist throughout the system.

Multiple one-way functions can be obtained as follows. A function H is a *parametric one-way function* if given a value z and a parameter p, it is computationally unfeasible to find a value x so that $z = \mathrm{H}(x, p)$ [26]. An example of practical implementation is $\mathrm{H}(x, p) = \mathrm{G}(\mathrm{E}_x(p))$ where G is a hash function and $\mathrm{E}_x(p)$ denotes the encryption of p using a symmetric encryption cipher E with key x. Thus, a parametric one-way function corresponds to a family of one-way functions, a function for each value of the parameter. In an implementation of our protection system reserving a (PF, SF) pair for each subject, the parametric one-way function will be known to all subjects, and a pair of parameters, corresponding to PF and SF, will be reserved for each subject.

We wish to point out that the implementation of the password system based on one-way functions is an essential feature of our protection system design. In particular, we generate only one initial random password, and all the other passwords are produced by application of the PF and SF functions. This mechanism is the basis of key reduction. If all passwords were generated at random, for instance, key reduction would require intervention of the object owner; a subject possessing a given key k and wishing to generate a key k' matching a weaker password would have to interact with the owner, which verifies validity of k and returns the weaker key. As seen in Section 1.2, this is a complication of the overall key distribution mechanism that we have been aimed at avoiding.

Of course, it may well be the case that a subject receives two or more keys for the same object from independent sources. This is not a security hole; indeed, in a one-way chain, possession of an arbitrary number of values cannot be used to calculate a previous value. One could argue that using two arbitrary PF and SF functions could reduce the overall system security unless these functions are guaranteed not to affect password randomness. On the other hand, if we use a single parametric one-way function $H = G(E_x(p))$ with different values for parameter p, and if symmetric cipher E is secure against known plaintext attacks, it is not necessary that function G has any cryptographic property since the required strength is guaranteed by E [26].

5. CONCLUDING REMARKS

We have considered the problem of access privilege management in a classical protection environment featuring subjects attempting to access the protected objects. We express an access privilege in terms of an access right and a privilege level. The privilege level and a protection diagram associated with each given object determine whether a nominal access privilege for this object corresponds to an effective, possibly weaker access privilege, or is revoked.

We associate a password system with each object. The password system takes the form of a hierarchical bidimensional one-way chain: a primary chain contains the passwords for the strongest access right at the different privilege levels; a secondary chain for each privilege level contains the passwords for the access rights at that privilege level. A subject possesses a nominal access privilege for a given object if it holds a key matching one of the passwords in the password system of that object; the protection diagram determines the extent of the corresponding effective access privilege.

The following is a brief summary of the main results we have obtained:

- A subject that holds a key for a given object is free to distribute this key to other subjects, which acquire the corresponding access privilege. A key reduction mechanism allows a subject that holds a key for the strongest access right at a given privilege level to distribute keys for the same access right or weaker access rights, at the same privilege level or at lower privilege levels. Furthermore, a subject that holds a key for an access right which is not the strongest access right can distribute keys for weaker access rights at the same privilege level.
- A subject that owns a given object can review or even revoke the passwords for this object by simply modifying the protection diagram. Review and revocation can be limited to a

subset of all the passwords. Two or more passwords for the same access right at different privilege levels can be revoked independently of each other. We have shown that the effects of a password review or revocation are transitive and temporal.

- The memory requirements to represent a protection diagram are negligible. As far as password storage is concerned, space-time trade-offs are possible. If we store all the passwords of the password system as part of the internal representation of the given object, a single comparison is necessary to ascertain the validity of a given key. Alternatively, we can store only the passwords in the primary chain, or even a single password, i.e. the seed of the primary chain, at the cost of recalculating several passwords to validate a key.

ACKNOWLEDGMENTS

The author thanks the anonymous reviewers for their insightful comments and constructive suggestions.

This work has been partially supported by the TENACE PRIN Project (Grant no. 20103P34XC_008) funded by the Italian Ministry of Education, University and Research.

REFERENCES

[1] Samarati, P. and de Capitani di Vimercati, S. (2001) Access control: policies, models, and mechanisms, in: Focardi, R. and Gorrieri R. (eds.), Foundations of Security Analysis and Design. Springer, Berlin, Heidelberg, pp. 137–196.

[2] Harrison, M. A. and Ruzzo, W. L. (1976) Protection in operating systems. Communications of the ACM, 19, 8, 461–471.

[3] Seitz, L., Pierson, J. M. and Brunie, L. (2003) Key management for encrypted data storage in distributed systems. Proceedings of the Second IEEE International Security in Storage Workshop, Washington, DC, USA, October 2003. IEEE Computer Society, Washington, DC, USA.

[4] Levy, H. M. (1984) Capability-Based Computer Systems. Digital Press, Bedford, Mass, USA.

[5] de Vivo, M., de Vivo, G. O. and Gonzalez, L. (1995) A brief essay on capabilities. SIGPLAN Notices, 30, 7, 29–36.

[6] England, D. M. (1974) Capability concept mechanisms and structure in System 250. Proceedings of the International Workshop on Protection in Operating Systems, IRIA, Paris, pp. 63–82. IRIA, Paris.

[7] Klein G. et al. (2009) seL4: formal verification of an OS kernel. Proceedings of the 22nd ACM Symposium on Operating Systems Principles, Big Sky, MT, USA, pp. 207–220. ACM, New York, NY, USA.

[8] Wilkes, M. V. and Needham, R. M. (1979) The Cambridge CAP Computer and Its Operating System. North Holland, New York, USA.

[9] Houdek, M. E., Soltis, F. G. and Hoffman, R. L. (1981) IBM System/38 support for capability-based addressing. Proceedings of the 8th Annual Symposium on Computer Architecture, Minneapolis, Minnesota, USA, pp. 341–348. IEEE Computer Society Press, Los Alamitos, CA, USA.

[10] Neumann, P. G. and Feiertag, R. J. (2003) PSOS revisited. Proceedings of the 19th Annual Computer Security Applications Conference, Las Vegas, NV, USA, pp. 208–216. IEEE, Los Alamitos, CA, USA.

[11] Meyer, M. (2004) A novel processor architecture with exact tag-free pointers. IEEE Micro, 24, 3, 46–55.

[12] Castro, M. D., Pose, R. D. and Kopp, C. (2008) Password-capabilities and the Walnut kernel. The Computer Journal, 51, 5, 595–607.

[13] Chase, J. S., Levy, H. M., Lazowska, E. D. and Raker-Harvey, M. (1992) Lightweight shared objects in a 64-bit operating system. Proceeding of the Conference on Object-Oriented Programming Systems, Languages, and Applications, Vancouver; in: SIGPLAN Notices, 27, 10, 397–413.

[14] Heiser, G., Elphinstone, K., Vochteloo, J., Russell, S. and Liedtke, J. (1998) The Mungi single-address-space operating system. Software — Practice and Experience, 28, 9, 901–928.

[15] Lopriore, L. (2013) Password capabilities revisited. The Computer Journal, first published online November 11, doi:10.1093/comjnl/bxt131

[16] Pose, R. (2001) Password-capabilities: their evolution from the Password-Capability System into Walnut and beyond. Proceedings of the Sixth Australasian Computer Systems Architecture Conference, Gold Coast, Australia, pp. 105–113. IEEE Computer Society, Washington, DC, USA.

[17] Anderson, M., Pose, R. D. and Wallace, C. S. (1986) A password-capability system. The Computer Journal, 29, 1, 1–8.

[18] Lopriore, L. (2013) Protection structures in multithreaded systems. The Computer Journal, 56, 4, 478–496.

[19] Miller, M. S., Yee, K.-P. and Shapiro, J. (2003) Capability Myths Demolished. Technical Report SRL2003-02, Systems Research Laboratory, Johns Hopkins University. Available at http://srl.cs.jhu.edu/pubs/SRL2003-02.pdf

[20] Shapiro, J. S., Smith, J. M. and Farber, D. J. (1999) EROS: a fast capability system. Proceedings of the Seventeenth ACM Symposium on Operating Systems Principles, Kiawah Island Resort, SC, December 1999; in: Operating Systems Review, 34, 5, 170–185.

[21] Gligor, V. D. (1979) Review and revocation of access privileges distributed through capabilities. IEEE Transactions on Software Engineering, SE-5, 6, 575–586.

[22] Leung, A. W. and Miller, E. L. (2006) Scalable security for large, high performance storage systems. Proceedings of the Second ACM Workshop on Storage Security and Survivability, Alexandria, Virginia, USA, pp. 29–40. ACM, New York, NY, USA.

[23] Lopriore, L. (2012) Encrypted pointers in protection system design. The Computer Journal, 55, 4, 497–507.

[24] Lamport, L. (1981) Password authentication with insecure communication. Communications of the ACM, 24, 11, 770–772.

[25] Hu, Y.-C., Jakobsson, M. and Perrig, A. (2005) Efficient constructions for one-way hash chains. Proceedings of the Third International Conference on Applied Cryptography and Network Security, New York, NY, USA; in: Lecture Notes in Computer Sciences, 3531, Springer-Verlag, Berlin, Heidelberg.

[26] Trappe,W., Song, J., Poovendran, R. and Ray Liu, K. J. (2003) Key management and distribution for secure multimedia multicast. IEEE Transactions on Multimedia, 5, 4, 544–557.