CERIAS Tech Report 2005-36

PROVABLE BOUNDS FOR PORTABLE AND FLEXIBLE PRIVACY-PRESERVING ACCESS RIGHTS

by Marina Blanton and Mikhail Atallah

Center for Education and Research in
Information Assurance and Security,
Purdue University, West Lafayette, IN 47907-2086

Provable Bounds for Portable and Flexible Privacy-Preserving Access Rights*

Marina Blanton Mikhail J. Atallah
Department of Computer Science
Purdue University
{mbykova,mja}@cs.purdue.edu

ABSTRACT

In this work we address the problem of portable and flexible privacy-preserving access rights for large online data repositories. *Privacy-preserving* access control means that the service provider can neither learn what access rights a customer has nor link a request to access an item to a particular customer, thus maintaining privacy of both customer activity and customer access rights. *Flexible* access rights allow any customer to choose any subset of items from the repository and correspondingly be charged only for the items selected. And *portability* of access rights means that the rights themselves can be stored on small devices of limited storage space and computational capabilities, and therefore the rights must be enforced using the limited resources available.

Our main results are solutions to the problem that utilize minimal perfect hash functions and order-preserving minimal perfect hash functions. None of them use expensive cryptography, all require very little space, and they are therefore suitable for computationally weak and space-limited devices such as smartcards, sensors, etc. Performance of the schemes is measured as the probability of false positives (i.e., the probability that access to an unpurchased item will be permitted) for a given storage space bound. Using our techniques, for a data repository of size n and subscription order of $m \ll n$ items, we achieve a probability of false positives of m^{-c} using only $O(cm)$ bits of storage space, where c is an adjustable parameter (a constant or otherwise) that can be set to provide the desired performance. This is the first time that such provable bounds are established for this problem, and we believe the techniques we use are of more general interest through the unusual use we make of perfect hashing.

*Portions of this work were supported by Grants IIS-0325345, IIS-0219560, IIS-0312357, and IIS-0242421 from the National Science Foundation, Contract N00014-02-1-0364 from the Office of Naval Research, by sponsors of the Center for Education and Research in Information Assurance and Security, and by Purdue Discovery Park's e-enterprise Center.

SACMAT'05, June 1–3, 2005, Stockholm, Sweden.

Categories and Subject Descriptors

K.6.5 [**Management of Computing and Information Systems**]: Security and Protection; E.4 [**Data**]: Coding and Information Theory—*data compaction and compression*; F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems.

General Terms

Security, Design, Algorithms.

Keywords

Portable access rights, compact policy representation, minimal perfect hash functions, order-preserving minimal perfect hash functions, algorithm analysis.

1. INTRODUCTION

Consider an online data repository that contains a very large number of objects (i.e., articles, books, magazines, multimedia objects, or any other type of data items). The owner of such a data collection allows its customers to subscribe (and consequently obtain access) to items of their choice. To make the service as convenient to the customers as possible, the service provider lets the customers subscribe to any subset of the items from the data repository and charges them only for the items in that subset. With the recent growth in the number and the level of maturity of online data collections, this model has emerged as an appropriate way of addressing customer needs.

Today many customers of online services are concerned with privacy of their personal information and may not be willing to use services that do not adequately assure that personal information will not be misused. For example, access patterns to the items that a customer views might be considered confidential because they permit customer profiling. For that reason, the service provider does not store customer configurations on the server, but rather uses tamper-resistant cards to enforce access control at the user end and to anonymously send requests to the server. Under this setting, it is not possible for the service provider to link two access requests made by the same customer and therefore to profile its customers. However, because the server now needs to issue many (possibly disposable) cards, their cost must be kept at a low level. This means that such cards will be severely limited in their computational capabilities, as well as in their storage space.

If a card that stores a customer's subscription set does not have enough capacity to store at least one bit per item

in the (huge) data repository, then it becomes impossible for it to exactly represent all possible subsets of the repository items. Thus, some items or subset of items will have to share the same configuration and introduce "false positives" into the scheme — a *false positive* is an item that was not listed in the subscription but which the customer is permitted to access. This model is acceptable if the probability of a false positive (PFP) is small enough. The goal is then to design a scheme for computationally efficient access control enforcement under space constraints that minimizes the number of false positives implicit to each card. Of course, false negatives are not tolerated: a customer who has paid to subscribe to an item must always be granted access to that item.

In the rest of this work we use the following notation: the (very large) data repository contains n items. A customer can request access to (and accordingly pay for) m items, $1 \leq m \leq n$. Access rights are stored on a card of limited capacity of k bits, where $k < n$ and $k < m \log n$.[1]

In this paper we give schemes that can be used to address the problem of flexible and privacy-preserving access control for large data repositories using weak devices. Our first (preliminary) solution utilizes minimal perfect hash functions. For a subscription order of m documents, it gives solutions of $O(cm)$ space with probability of false positives being 2^{-c}, where c is an adjustable parameter. Our second solution uses order-preserving minimal perfect hash functions to achieve significantly better asymptotic performance: with $O(cm)$ storage space available, the probability of false positives is m^{-c}.

The rest of this paper is organized as follows: section 2 gives an overview of related work. In section 3, we describe the general framework and our design goals, and list notation used throughout the paper. Section 4 gives the preliminary solution, which should be viewed as a "warmup" for the asymptotically better solution that is given in section 5. Section 6 makes further remarks on the relative merits of the schemes and discusses an approach to completely eliminate certain items from possible false positives. It also briefly covers space utilization techniques for hierarchies. Finally, section 7 concludes the paper.

2. RELATED WORK

Most literature on digital libraries does not explore the problem of access control, and many deployed systems provide only a single or otherwise very few subscription types. Payette and Lagoze [26] acknowledged this problem and introduced a spectrum of policy enforcement mechanisms that range from system-wide to object-specific. Their work, however, provides only a general framework and does not address the problem of policy assignment.

Work conducted on XML also explores the problem of access control for online data repositories, which includes securing access to XML documents and using XML as a tool for specifying security policies (see, e.g., [6, 7, 8, 16, 17]). Bertino et al. [5] use binary strings to represent both customer policy configurations and document policies, but they allocate one bit per policy on the assumption that there will be a limited number of different subscription types. Thus, their approach becomes inefficient as the data repository grows in size and each customer chooses a customized document subscription set.

The idea of achieving space efficiency at the cost of a small probability of false positives was introduced in Bloom [9]. Bloom filters support approximate membership queries and are widely used in a broad spectrum of applications ([11, 18, 25], to name a few). Such data structures achieves a better space utilization than simple hash representation, but the filter length (which in our case corresponds to the card capacity) still should be larger than the total number of items in the set n to result in a reasonable performance. This is not suitable for cards of small capacity, and even customized Bloom filters do not appear to provide acceptable results.

Other techniques for concise representation of portable access rights were used in the context of software license management [4, 2]. These solutions, however, do not apply to our problem, mainly because we cannot afford to avail ourselves of resources external to the card (as was the case in [4, 2]). The more recent work in [1, 12], on the other hand, considers the same problem of portable and flexible access rights for large data repositories. In [12], the authors consider static policy assignment to all repository documents, which makes addition of new items problematic without performing periodic policy updates (after which all smartcards must be refreshed) and also makes it possible for dishonest users to share and use information about false positives. In [1], similarly to this work, a unique policy representation is used for each subscription (even for identical subscriptions), but the solution given in that work does not have a deterministic algorithm and therefore might not be suitable in some settings. This paper is the first that describes solutions with a solid analysis that, given a threshold for the rate of false positives, are guaranteed to result in access rights specifications that do not exceed that threshold.

Some of our solutions use minimal perfect hash functions (MPHF) as their underlying building blocks. MPHFs have received significant attention, and a number of algorithms can be found in [21, 20, 15, 19]. There are MPHFs and order-preserving MPHFs (OPMPHFs) that for random m strings take a total of $O(m)$ bits to store the functions (and this is also the lower bound). See [21, 19] for more detail.

Work on unlinkability and untraceability was started by Chaum [14] and has been explored more extensively in recent years. In particular, work on unlinkability includes anonymous group authentication ([3, 10, 13, 22, 23, 24, 27, 28, 29] and others) and unlinkable serial transactions [30] for subscription-based services. Prior work, however, does not account for the fact that descriptions of access rights (or service types) may be long and required to be portable, while we describe schemes that combine compact policy representation with transaction unlinkability.

3. PROBLEM SPECIFICATION

3.1 General Framework

The general model used in our work is depicted in Figure 1, and consists of two stages: During the initialization stage — which can take place in a bookstore or at a public library — a customer chooses items of his choice, pays for the items selected, and receives a customized card that subsequently permits access to these items. During the card usage stage — which can be done from a home computer, library,

[1] If $k \geq m \log n$ then the card can explicitly store the m items. So we henceforth assume that k is less than $m \log n$, i.e., that space on the card is tight.

Initialization

1. Customer chooses m items for purchasing.
2. Server processes the order and creates a card.
3. Customer pays and receives the card.

Operation

	Card		Server
1.	Receives from the user a request to access item i		
2.	Checks if access can be granted		
3.	Anonymously authenticates to the server and requests item i on behalf of the user	$\longrightarrow$	Verifies authentication credentials
4.		$\longleftarrow$	Sends item i to the requester

Figure 1: General model.

etc. — the customer can request access to any items from the repository. If the card permits access, it uses the built-in anonymous authentication protocol to prove its authenticity to the server, and obtains the item from the server.

Throughout this work, we assume that a card is authentic and can *anonymously* and at low computational cost authenticate itself to the server. A number of solutions that range from trivial secret key schemes to more complex and provably secure ones can be used to achieve this goal (see, for instance, [23]). Card unforgeability is also achieved through other, standard techniques described in prior literature and is out of scope of this work.

Here by "server" we do not necessarily mean a remote server. Instead, it could be a local (trusted) content player at the client end or any other mechanism used by the content owners to enforce their policies. In that case, the encrypted content is available at the client's end and the server grants access by decrypting and then displaying it. Therefore the model does not necessarily assume network connectivity for data access.

3.2 Design goals

The design goals that we require any solution to have are as follows:

Low rate of false positives The probability (or rate) of false positives — the probability of a random document to be among the documents to which access is authorized — is the main evaluation criterion of any approach, and the goal of this work is to minimize such a PFP. The PFP depends, of course, on the storage space available on the card.

Transaction untraceability and unlinkability For customer privacy, we require that after a customer buys an access card and uses it to retrieve an item from the repository, it is not possible to use the data sent in the request to tell with probability significantly greater than a random guess which customer is making this request. Similarly for transaction unlinkability, we require that given two access requests it is not possible to tell with probability significantly greater than a random guess whether these two requests originated with the same user.

Unique policy representation It is also a design requirement that every policy representation stored on a card is unique. More precisely, given two subscription requests that contains identical sets S_1 and S_2 of items to be purchased, their representation stored on access cards C_1 and C_2 will be different and the false positives implicit to each card will be different. We require this to eliminate the possibility of sharing information about false positives by dishonest customers. When this is not the case and a fixed set of items triggers the same set of false positives, dishonest users might share this information through, possibly, public channels such as the Internet, making the scheme unusable to the data provider.

No additional sources of information The schemes we design are for online data repositories that, using a card, can be accessed from a number of places such as terminals at public libraries, bookstores, home workstations, and other places connected to the Internet. Therefore, if a scheme requires some additional information to be stored on external storage, in our case there is no reasonable place at which such information can be stored (no user information can be stored at the server itself by the above requirement of untraceability). Thus, the access card itself should contain all information necessary to perform access verification.

Fast access verification, fast card generation time These parameters also serve as evaluation criteria of each scheme, and in general we require card generation time to be bounded by a low-degree polynomial in n or, preferably, by a polynomial in m. Access verification time should be bounded by $O(k)$ (where k is the space available on the card) because each card is assumed to be a computationally weak device.

Forward compatibility In any proposed solution, if a card is created at time t_1 when the data repository contained n_1 documents, it also should stay operational at time $t_2 > t_1$ when the data repository contains $n_2 > n_1$ documents. In other words, the scheme should stay operational as new documents are added to the data repository.

The above requirements make our problem very different from a mere data compression one. Another difference from data compression, is that here each representation on the card must be usable "as is" without uncompressing it first: There is no room in the card for that, and using server memory to uncompress would reveal enough about the card to make profiling of the card's usage patterns possible (because two cards' contents are different even if both have the same subscription set. They are in some sense an implicit ID for the card and should therefore not be revealed to the server). Client memory cannot be trusted either.

3.3 Notation

Throughout this work, we use the following notation. The data repository contains n items that are numbered 1 through

n. A customer subscription order contains m items and is denoted as $\{i_1, \ldots, i_m\}$ where $1 \leq i_1 < \ldots < i_m \leq n$.

A hash function $h : X \rightarrow Y$ that is 1–1, i.e. $\forall x_1, x_2 \in X$, $h(x_1) \neq h(x_2)$ iff $x_1 \neq x_2$, is called a *perfect hash function*. In other words, perfect hash functions never result in collisions. A hash function $h : X \rightarrow Y$ that is 1–1 and for which $|X| = |Y|$ is a *minimal perfect hash function* (MPHF). An *order-preserving* MPHF (OPMPHF) also has the property that it maps the ith smallest element of X into the integer i.

In what follows we use f to denote a minimal perfect hash function that maps $\{i_1, \ldots, i_m\}$ into $\{1, \ldots, m\}$ without collisions. Also, functions f', f'' denote order-preserving MPHFs each of which maps $\{i_1, \ldots, i_m\}$ into $\{1, \ldots, m\}$ without collisions and in an order-preserving manner (i.e., $f'(i_j) = j$).

4. A PRELIMINARY SOLUTION

Given a card that can store $k = O(cm)$ bits, this first approach gives us: (i) card creation time polynomial in m, and (ii) probability of false positives 2^{-c}. Note that it is reasonable to assume that cards can store cm bits. The reason is that this space will be small for relatively small orders; for larger, more expensive orders one can use cards of larger capacity, the cost of which can be offset by the amount charged for the subscription order.

In what follows, H is a keyed cryptographic one-way hash, whose key is unique to each card (this is so as to make false-positive information sharing impossible); the key's purpose is not cryptographic security, but rather making each card unique. The k bits available do not include the bits needed for storing the key for the hash H, which would be small in practice. For instance, a 20-bit key would result in a million different cards that can request identical m items yet be different; and the possibility of such sharing when the two cards correspond to different sets of m documents significantly decreases. An alternative to a keyed H would be that H is the same hash function for all cards, but the random choices made during the computation of a suitable minimal perfect hash function would vary from card to card.

4.1 Card creation

1. Compute a minimal perfect hash function f for $\{i_1, \ldots, i_m\}$. Store f in the card, using $O(m)$ bits (according to [20], a MPHF for m random strings can be stored using bm bits, where b is a constant and can normally be 2). This leaves $k' = O(cm)$ bits available for what follows.

2. Partition these k' bits into m blocks of c bits each; call them blocks B_i $(i = 1, \ldots, m)$.

3. Let the hash function $H(x)$ produce a c bits long hash of x (e.g., by considering only c of the 160 bits it produces in case of SHA-1). For every item i from the m items $(i \in \{i_1, \ldots, i_m\})$, if $f(i) = j$, then set the bits of block B_j on the card equal to $H(i)$.

4.2 Access verification

Every time a customer uses his card to request access to an item i, the card performs the following:

1. Compute $f(i)$, assume $f(i) = j$.

2. Compare the c bits of the card's block B_j to the corresponding computed c bits of $H(i)$.

3. Access is allowed if these c bits match, denied otherwise.

4.3 Analysis

THEOREM 1. *Given $k = O(cm)$ storage space, the above MPHF-based approach produces in time polynomial in m a solution with the properties of (a) transaction unlinkability and untraceability, (b) unique policy representation, (c) no additional sources of information, (d) forward compatibility, and (e) probability of false positives 2^{-c}.*

Proof Card creation takes time polynomial in m because a MPHF can be generated in polynomial time [21]. Given $k = (c + b)m = O(cm)$ space, the probability of a false positive is less than 2^{-c} (see, e.g., [20] for more detail).

Transaction untraceability is achieved because the card anonymously authenticates to the server and then everything else it sends is a request for a specific data item with no personal or card-specific information. By the same argument, any two transactions are also unlinkable.

Unique policy representation is achieved through the use of the keyed hash function H or, alternatively, by randomizing f itself. Each card will also stay operational as we add more items to the data repository because the card is dependent on the purchased items and contains no information about other items or the size of the data repository. This means that the forward compatibility requirement is satisfied. Finally, by design this scheme does not use any additional sources of information. □

4.4 Case where $c = \log m$

If in the above $c = c' \log m$ where c' is constant, then the scheme has $k = O(c' m \log m)$ bits of storage and an $m^{-c'}$ probability of false positive. In such a case, however, the following simpler scheme that achieves the same bounds can be used. We use a keyed hash function F (not a perfect one — collisions can occur) that maps items in the range $[1, n]$ into $[1, m^{c'+1}]$. An example of such a function that we use in our further discussion is $F(i) = H(i) \bmod m^{c'+1}$, where H is a keyed cryptographic one-way hash function. What the card stores is the (at most m) elements of $[1, m^{c'+1}]$ to which the subscription items map. It allows access to a requested item i iff $F(i)$ is stored on the card. Since each of the (at most) m numbers stored is $(1 + c') \log m$ bits long, the total space needed is $O(c' m \log m)$ bits. The probability of a false positive is no greater than $m/m^{c'+1} = m^{-c'}$. This matches the MPHF scheme's performance if $k = m \log m$, but it cannot be used if $k = o(m \log m)$. When it can be used, however, it has the potential for the following heuristic improvement in its space usage: The m stored elements from $[1, m^{c'+1}]$ could be such that the trie implied by their bit representations makes further space savings possible (by storing common prefixes or other common bit patterns only once). The expected space needed to store the trie remains $O(m \log m)$ bits, however, so the savings are by no more than a constant factor, and the multiplicative factor of $\log m$ in the space complexity remains.

The scheme in the next section achieves the same false positives probability performance of m^{-c} but without the multiplicative factor of $\log m$ in the space used.

5. AN ASYMPTOTICALLY BETTER SOLUTION

Given $k = O(cm)$ space available on the card, the approach described in this section and which is based on usage of order-preserving MPHFs gives us: (i) card creation time polynomial (in fact, linear) in m, and (ii) probability of false positives m^{-c}, where c is an integer parameter that can be chosen so as to achieve a desired PFP. For large enough m (which we assume is the case since $m > k/\log n$), however, it is sufficient to have $c = 1$. We start with describing the $c = 1$ version of the scheme, after which we extend it to larger values of c.

5.1 Card creation

As usual, we deal with a subscription order $\{i_1, \ldots, i_m\}$ where $i_1 < i_2 < \ldots < i_m$. We use two order-preserving minimal perfect hash functions f' and f'', each computed for this subscription order: $f'(i_j) = j$ and $f''(i_j) = j$ for all $j \in [1, m]$. To see why we use different functions f' and f'', we first need to recall that the construction of an order-preserving minimal perfect hash function involves many random choices along the way, and f' and f'' will differ through those different random choices. While the effect of such functions on the elements of the set $\{i_1, \ldots, i_m\}$ is fixed and well known for all i_j (i.e., $f'(i_j) = f''(i_j) = j$), their effect on elements not in the set $\{i_1, \ldots, i_m\}$ is arbitrary. Consequently, we use two different functions f' and f'' for their different effects on randoms r *that are not in set* $\{i_1, \ldots, i_m\}$. This is an unusual use of such functions because we care about their *random effect* on an r that is *not* in the set, as much as about their predictable effect on an i_j from the set. The card hence stores f' and f'', which take $O(m)$ bits of space.

While the effect of those random choices on a random $r \notin \{i_1, \ldots, i_m\}$ has not been investigated in the literature, we postulate that the existing OPMPHF schemes can be used to hash such an r uniformly on the interval $[1, m]$. That is, each of $f'(r)$ and $f''(r)$ is random and uniformly distributed over $[1, m]$. What follows is subject to this assumption[2].

5.2 Access verification

To verify a request for access to an item i, the card needs to perform the following steps:

1. It computes $f'(i)$ and $f''(i)$.
2. Access is granted if $f'(i) = f''(i)$, denied otherwise.

5.3 Extension to higher values of c

To obtain versions of the scheme with $c > 1$, instead of using two functions f' and f'', we use $c + 1$ such functions: access to i is granted if all $c+1$ functions map i into the same value, and is denied otherwise. Of course different random parameters are selected when constructing each of these $c+1$ functions, and the space complexity becomes $O(cm)$ bits.

[2]It is possible that OPMPHF representations that use an optimally small number of bits bm will not involve many random choices for certain elements of the set $\{i_1, \ldots, i_m\}$. This means that $f'(r)$ and $f''(r)$ might not be truly independent for some values of r. To magnify the randomization effect of the functions on such r's, we might want to increase the space occupied by the functions by increasing the value of the constant b, and make sure that the number of random choices during function generation is large. Obviously, this topic deserves further investigation and formal treatment.

Note that if the value of c is relatively large compared to m, it might be difficult or even impossible to generate $c + 1$ different functions for those m items. In such a case, either the value of c might be lowered, or the space needed to store these $c + 1$ functions might have to be increased (each function will still require $O(m)$ bits but with a larger than optimal constant).

5.4 Analysis

THEOREM 2. *Given $k = O(cm)$ space, the above OPMPHF-based approach produces in time polynomial in m a solution with the properties of (a) transaction unlinkability and untraceability, (b) unique policy representation, (c) no additional sources of information, (d) forward compatibility, and (e) probability of false positives m^{-c}.*

Proof Sketch Each of the $c+1$ functions can be computed in linear time and space [19], whence the claimed card creation time. We now argue that the probability of a false positive is m^{-c}. First we note that, for an $r \notin \{i_1, \ldots, i_m\}$ to be a false positive, all of the $c+1$ functions must map r into the same value. Recall from our above assumption that the choices of different random parameters for each such function f' randomizes $f'(r)$ uniformly over $[1, m]$, and choosing the different functions' random parameters independently effectively makes this $f'(r)$ independent of the other $f''(r)$ values. The probability that the $c + 1$ functions map r into the same value is therefore m^{-c}.

Property (b) is ensured through the random choices in selection of the functions, and properties (a), (c), and (d) are by the same arguments as in the proof of Theorem 1. □

6. FURTHER REMARKS

The $k = O(m \log m)$ space approach described in Section 4.4 is the simplest to implement, and its space usage can be heuristically lowered as described in that section. This approach, however, cannot be used if $k = o(m \log m)$, whereas the MPHF-based scheme can work in cases when $k = o(m \log m)$, e.g., when $k = O(m)$. In general, both of these approaches give the same rate of false positives of m^{-c} if $k = O(cm \log m)$. The OPMPHF-based approach of Section 5 achieves the same false positives rate of m^{-c}, but with an asymptotically lower requirement for space: $O(cm)$ bits.

6.1 Decreasing the value of false positives

The performance of any of the schemes we described can be further improved with respect to the cost of false positives if we make sure that certain unpurchased but generally popular items are not among the false positives. We refer to such items in high demand as "hot" items, and for each customer they can be either system-wide (the same for everyone), card-specific (based on the subscription order at card-creation time), or both.

Note that, from the privacy point of view, it is acceptable for the data owner to determine the hot items for a card based on the card's subscription order (which must be given anyway at the time of purchase, e.g., during anonymous card purchase at a vending machine or bookstore). Later on, as the card is used, the card does not give away data about the subscription order or the card-specific forbidden hot items.

There are various ways of ensuring that such hot items are not among the false positives of a customer order. One

approach is to map them to a region different from the legitimate subscription items on the card, or to place a special mark on where they map to indicate that access to them should be denied. We could in fact use all of the techniques described in this paper for the hot items, as long as we consider them to be a "negative subscription list" in the sense that access is denied for any item on that list. This of course increases the space needed to represent a subscription order, but the increase in the space may be worthwhile if it is less than the cost associated with the hot items being among the false positives.

Another way of isolating such hot items is by weaving their isolation into the random choices made in the schemes we described: After we make random choices, we can evaluate the card encoding by counting how many hot items are among the false positives as the result, and make another set of random choices if necessary.

In general, in our schemes the rate of false positives is very small (e.g., for $m = 100$ and $c = 3$ the PFP is one in a billion), and therefore even if the list of hot items is long, only a tiny fraction of them will be among the false positives to be isolated. This permits us to use the above special treatment for those few items, if we want absolutely no hot items to be among the false positives.

6.2 Improving space utilization for hierarchies

Hierarchically structured data repositories provide additional possibilities for efficiently utilizing storage space on the cards and therefore minimizing the rate of false positives. For tree-like hierarchies, the objects in the repository corresponds to the leaves of the tree. In addition, all internal nodes are marked with unique identifiers. Then great space savings can be achieved if now instead of storing all m items we store nodes of the tree the entire sub-trees of which are among the m items.

Our techniques trivially apply to such hierarchical repositories if we assume that for every item i its "path to the root" can be obtained from the server. This, however, means more interaction with the server. It is not clear how to avoid this extra interaction: While it is well known that ancestral relationships in a tree are completely described by two linear listings of its nodes (e.g., preorder and postorder), this is not immediately exploitable because of the randomization introduced by the hashes. This clearly deserves further investigation.

An additional difficulty is that, in hierarchically structured objects, care must be exercised to ensure that certain nodes are not among the false positives. For instance, during the card creation process we must ensure that the root of the tree and other nodes high in the hierarchy are not among the false positives. Techniques of section 6.1 then can be applied to this case as well to exclude the special items from the possible false positives.

7. CONCLUSIONS

In this paper, we presented schemes for minimizing space requirements to permit user access to items of their choice from a large data repository in a privacy-preserving manner. Our schemes are based on the use of minimal perfect hash functions and comply with the design goals of: transaction untraceability and unlinkability, unique policy representation, single storage device, fast operation, and forward compatibility of the scheme. The primary goal is to minimize the rate of false positives; and for both of our schemes, given the bounds on the available storage space, we proved bounds on the false positives rates. These properties of the schemes are summarized in Table 1.

Scheme / Space	$k = O(cm)$	$k = O(cm \log m)$
Scheme I (MPHF-based)	2^{-c}	m^{-c}
Scheme II (OPMPHF-based)	m^{-c}	$m^{-c \log m}$

Table 1: The rates of false positives of the schemes for different storage space bounds. Here m is the number of items in the subscription, k is the storage space, and c is a constant.

The solutions presented in this work can be used for different applications, with the most intuitive ones being digital libraries that might contain books, articles, magazines, and also music, video, and other objects. With such systems in place, a customer can purchase a subscription to the items of interest from home, stores, or libraries, and have anonymous access to the media from many convenient locations (that support the model) as well. Other usages include access to locally stored (encrypted) objects, where trusted software plays the role of the server and on demand decrypts the objects that the user is authorized to access.

Our approach can also be combined with access mechanisms based on temporal constraints, but this topic is a direction of future research.

8. REFERENCES

[1] M. Atallah and M. Bykova. Portable and flexible document access control mechanisms. In *Computer Security – ESORICS 2004*, volume 3193, pages 193–208. Springer–Verlag, September 2004.

[2] M. Atallah and J. Li. Enhanced smart-card based license management. In *IEEE International Conference on E-Commerce (CEC'03)*, pages 111–119, June 2003.

[3] G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A practical and provably secure coalition-resistant group signature scheme. In *Advances in Cryptology – CRYPTO'00*, volume 1880 of *LNCS*, pages 255–270, 2000.

[4] T. Aura and D. Gollmann. Software license management with smart cards. In *USENIX Workshop on Smart Card Technology*, May 1999.

[5] E. Bertino, B. Carminati, E. Ferrari, B. Thuraisingham, and A. Gupta. Selective and authentic third-party distribution of XML documents. Working Paper, Sloan School of Management, MIT, 2002. `http://papers.ssrn.com/sol3/papers.cfm?abstract_id=299935`.

[6] E. Bertino, S. Castano, and E. Ferrari. On specifying security policies for web documents with an XML-based language. In *ACM Symposium on Access Control Models and Technologies (SACMAT'01)*, May 2001.

[7] E. Bertino, S. Castano, and E. Ferrari. Securing XML documents with author-$\mathcal{X}$. *IEEE Internet Computing*, 5(3):21–31, 2001.

[8] E. Bertino and E. Ferrari. Secure and selective dissemination of XML documents. *ACM Transactions on Information and System Security*, 5(3):290–331, August 2002.

[9] B. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.

[10] D. Boneh and M. Franklin. Anonymous authentication with subset queries. In *ACM Conference on Computer and Communication Security (CCS'99)*, pages 113–119, November 1999.

[11] A. Broder and M. Mitzenmacher. Network applications of bloom filters: A survey. Allerton Conference, 2002.

[12] M. Bykova and M. Atallah. Succinct specifications of portable document access policies. In *ACM Symposium on Access Control Models and Technologies (SACMAT'04)*, June 2004.

[13] J. Camenisch and M. Michels. A group signature scheme with improved efficiency. In *Advances in Cryptology – ASIACRYPT'98*, volume 1514 of *LNCS*, pages 160–174, 1998.

[14] D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–88, February 1981.

[15] Z. Czech, G. Havas, and B. Majewski. An optimal algorithm for generating minimal perfect hash functions. *Information Processing Letters*, 43(5):257–264, October 1992.

[16] D. Damiani, S. De Capitani Di Vimercati, S. Paraboschi, and P. Samarati. A fine-grained access control system for XML documents. *ACM Transactions on Information and System Security*, 5(2):169–202, May 2002.

[17] P. Devanbu, M. Gertz, A. Kwong, C. Martel, and G. Nuckolls. Flexible authentication of XML documents. In *ACM Conference on Computer and Communications Security (CCS'01)*, November 2001.

[18] L. Fan, P. Cao, J. Almeida, and A. Broder. Summary cache: A scalable wide-area web cache sharing protocol. *IEEE/ACM Transactions on Networking*, 8(3):281–293, 2000.

[19] E. Fox, Q. Chen, A. Daoud, and L. Heath. Order-preserving minimal perfect hash functions and information retrieval. *ACM Transactions on Information Systems*, 9(3):281–308, July 1991.

[20] E. Fox, Q. Chen, and L. Heath. A faster algorithm for constructing minimal perfect hash functions. In *Annual International ACM SIGIR*, pages 266–273, 1992.

[21] E. Fox, L. Heath, Q. Chen, and A. Daoud. Practical minimal perfect hash functions for large databases. *Communications of the ACM*, 35(1):105–121, January 1992.

[22] J. Kilian and E. Petrank. Identity escrow. In *Advances in Cryptology – CRYPTO'98*, volume 1462 of *LNCS*, pages 169–185, August 1998.

[23] J. Kim, S. Choi, K. Kim, and C. Boyd. Anonymous authentication protocol for dynamic groups with power-limited devices. In *Symposium on Cryptography and Information Security (SCIS'03)*, volume 1/2, pages 405–410, January 2003.

[24] C. Lee, X. Deng, and H. Zhu. Design and security analysis of anonymous group identification protocols. In *Public Key Cryptography (PKC'02)*, volume 2274 of *LNCS*, pages 188–198, February 2002.

[25] M. Mitzenmacher. Compressed bloom filters. In *ACM Symposium on Principles of Distributed Computing*, August 2001.

[26] S. Payette and C. Lagoze. Policy-carrying, policy-enforcing digital objects. In *European Conference on Research and Advanced Technology for Digital Libraries (ECDL'00)*, volume 1923, pages 144–157, 2000.

[27] P. Persiano and I. Visconti. A secure and private system for subscription-based remote services. *ACM Transactions on Information and System Security*, 6(4):472–500, November 2003.

[28] A. Santis, G. Cresenzo, and G. Persiano. Communication-efficient anonymous group identification. In *ACM Conference on Computer and Communication Security (CCS'98)*, pages 73–82, November 1998.

[29] S. Schechter, T. Parnell, and A. Hartemink. Anonymous authentication of membership in dynamic groups. In *Financial Cryptography*, volume 1648 of *LNCS*, pages 184–195, 1999.

[30] S. Stubblebine, P. Syverson, and D. Goldschlag. Unlinkable serial transactions. *ACM Transactions on Information and System Security*, 2(4):354–389, November 1999.