Remote Revocation of Smart Cards in a Private DRM System[*]

Keith Frikken   Mikhail Atallah   Marina Bykova

CERIAS and Department of Computer Sciences
Purdue University
{kbf,mja,mbykova}@cs.purdue.edu

Abstract

We describe a DRM smartcard-based scheme in which content access requests are not linked to a user's identity or smartcard, and in which compromised cards can be revoked without the need to communicate with any card (whether revoked or not). The scheme has many other features, such as efficiency and requiring minimal interaction to process an access request (no complex interactive protocols), forward and backward security, stateless receivers, and under certain cryptographic constructions collusion-resistance. The above is achieved while requiring the smartcard to store only a single key and to perform a single modular exponentiation per revocation. Furthermore, our solution introduces a combinatorial problem that is of independent interest.

Keywords: DRM, revocation scheme, privacy preservation.

1 Introduction

One of the problems with smartcard-based DRM schemes is that, when a card is compromised and its keys are revealed to an adversary, he could distribute or sell the keys and enable massive unauthorized access to any digital content that is encrypted with these keys. Although it is not easy to "crack" a smartcard, it is more likely and inexpensive than is commonly believed (Anderson & Kuhn 1996, Anderson & Kuhn 1997). This is why it is prudent for a content distributor to plan for such occurrences: When the content distributor learns of the compromise of such a card, he must stop using the compromised keys for the delivery of encrypted content, and switch to using other, non-compromised keys. We investigate how this can be achieved when privacy requirements prevent the content owner from knowing which card is at the receiving end of the encrypted content he is sending, i.e., when all cards (whether revoked or not) must be treated in the same way and receive the same encrypted content. We also do not want the content owner to have to contact any card to effectively carry out a revocation.

We present a scheme that achieves these goals, at the cost of having to send very little additional information with each encrypted content as a result of each compromised card. In the scheme, what is sent is the encryption of the (possibly large) content M with a random (session) key k (denoted by $Enc(M, k)$), together with $Enc(k, k_1), Enc(k, k_2), \ldots, Enc(k, k_t)$ such that (i) each non-revoked card is guaranteed to be able to obtain at least one of the k_i that allows it to get k (hence the content M), (ii) none of the revoked cards can get such a k_i. This allows for "implicit revocation" by the content distributor, without having to either contact or update any of the deployed cards (whether revoked or not). A larger number of compromised cards necessitate a larger t, so one of our design goals is to keep t small; however, all the $Enc(k, k_i)$'s combined typically take up much less space and bandwidth than the (typically large) $Enc(M, k)$ that is sent by the server to the client.

Here when we say that the "client obtains M", what we mean is that a special "reader" software (e.g., a media-viewer), that interacts with (and uses) the smartcard at the client's end, obtains M and displays it to the user. It is this server-trusted reader that mediates access to M by the user at the client's end and interacts with the smartcard so as to obtain from it k (and hence M). Note that k is not stored in the smartcard. How a client-side reader can be server-trusted is achieved using techniques (software, hardware, and combinations thereof) that are well documented in the literature (Arbaugh, Farber & Smith 1997, Kuhn 2004, Lipton, Rajagopalan & Serpanos 2002) and some of these are in fact a part of the planned next generation of "trusted computing platforms".

In our scheme, it is also possible to "undo" a revocation and therefore restore the ability of a previously revoked card to use the service. This means that our scheme has the expressive power to represent every possible subset of clients in the dynamically changing environment, and is suitable for applications such as pay-TV where customers can leave and later re-join the set of users who receive the service.

Finally, our scheme does not require k to be a one-time session key: In that case there is no need to re-encrypt any content, but, on the other hand, that would make piracy easier through the pirate distributing a short key rather than the voluminous digital content. We, however, assume in the rest of this paper that k is used as a session key.

The organization of this paper is as follows: Section 2 provides a description of the problem. Section 3 gives an overview of related literature. In section 4, we provide description of cryptographic primitives used later in our protocols. Section 5 presents our basic protocol, extensions to which are given in section 6. Such extensions significantly improve the performance of the basic protocol in terms of both its computational and space requirements, as well as provide

---

generalization of the scheme to higher dimensions. In section 7, we list open problems, and finally section 8 concludes the paper.

2 Problem Description

Suppose $n + 1$ users have purchased the right to access some copyrighted media for a given period of time (e.g., for a particular month), and were given for that purpose respective smartcards $C_0, C_1, \ldots, C_n$. Access is through one or more servers operated by the content-distributors (for this paper we assume there is a single server S, but our schemes could easily be extended to multiple servers). Although a user's smartcard C_i may have been issued to them by the server S, another possibility for the privacy-conscious user is the anonymous purchase of C_i (e.g., at a bookstore using a cash payment). The properties we seek to achieve for our scheme are the following:

1. *Protected*: Only a client with a smartcard should be able to access the data.

2. *Private*: S should not be able to determine which smartcard is making an access request. Therefore, when a client requests a message, it should send the server a request for content that gives no clue as to which C_i is requesting the content. The purpose of this is to avoid the possibility of tracking and profiling by S of a C_i's content access patterns; such tracking would have the danger that if even once the C_i is associated with an event that is linked to the customer's real identity (e.g., access from a particular IP address), the whole access history of that individual becomes known to the server. The client would normally make the request either through some form of anonymous communication such as using a mix (e.g., (Chaum 1981)) or using a public PC at a cyber-cafe, but if the client occasionally makes a request from an IP address that can be traced back to him, then only this particular request is linked to him, not the whole history of requests he previously made using that C_i.

3. *Revocable:* S should be able to revoke a set of client's smartcards (remotely "shred" them by rendering them useless). This would occur when S learns that a smartcard has been cracked and the keys in it compromised, but it is crucial that this does not require reissuing of keys for any cards not in the revoked set, or communicating with any card (whether revoked or not).

4. *Non-interactive:* Upon receipt of a request, the server should send suitably encrypted content such that only non-revoked smartcards can decrypt it. The client and server do not engage in any protocol (such as a zero-knowledge proof) to determine whether the smartcard is valid or revoked.

5. *Efficient:* The system should be efficient in terms of both communication and computation.

6. *Forward and backward secure:* No newly added smartcards can read previous messages, and no users whose keys have been revoked can read future messages, even with collusion by arbitrarily many other users with revoked keys.

3 Related Work

Our work is closely related to other work on broadcast encryption. A *broadcast encryption scheme* (which, in some sources, is also referred to as *revocation scheme*) allows a distribution center to securely broadcast data to a dynamically changing set of users (so called *privileged users*) over an insecure channel. In other words, the scheme should allow to selectively exclude a subset of users from receiving the data by carefully constructing keys used to encrypt broadcast material. Broadcast encryption was motivated by applications such as pay-TV where users who fail to pay for the service leave the set of privileged users.

The idea of broadcasting of a secret was first explored in (Berkovitz 1991), and the formal study of broadcast encryption was initiated in (Fiat & Naor 1994). In (Fiat & Naor 1994), a scheme is given that requires each user to store $O(k \log k \log n)$ keys and the center to broadcast $O(k^2 \log^2 k \log n)$ messages, where n is the total number of users and k is the maximum number of excluded users (so called *revocation threshold*). The scheme is based on the idea of constructing a scheme that is 1-resilient (i.e., works when a single user is excluded) and applying multi-level hashing to subgroups of users to provide resilience against a larger number of excluded users. Subsequent work of (Blundo & Cresti 1994, Blundo, Mattos & Stinson 1996, Luby & Staddon 1998) and others further explore and provide analysis of broadcast encryption schemes. Many other broadcast encryption schemes based on hierarchies, secret sharing, cover-free set systems, etc. have been proposed in the literature. We categorize them based on their intended use and the underlying methods used.

Practical schemes for *multicast security* were first designed in (Wong, Gouda & Lam 1998) and (Wallner, Harder & Agee 1999). Both (Wong et al. 1998) and (Wallner et al. 1999) are tree-based approaches that use key hierarchies and achieve $O(\log n)$ keys per user and $O(r \log n)$ communication overhead to revoke r users. The work in (Canetti, Garay, Itkis, Micciancio, Naor & Pinkas 1999) and (McGrew & Sherman 1998) improve (by a constant factor) on the schemes, and another work (Canetti, Malkin & Nissim 1999) studies tradeoffs between communication overhead and space requirements of the schemes. All of the work above, however, require *stateful* receivers (i.e., keys of all of the users change after a leave/join) and therefore are not appropriate in our setting where clients do not always stay online and the keys of cards not in the revoked set should stay unchanged.

Another broadcast encryption scheme based on trees (using layered subset difference) was developed (Halevy & Shamir 2002). The scheme is stateless and requires $O(\log^{1+\epsilon} n)$ keys per user, $O(r)$ message length, and $O(\log n)$ computation per message received.

Combinatorial approaches to broadcast encryption include (Kumar, Rajagopalan & Sahai 1999) who use error-correcting codes and cover-free set systems to give solutions with $O(k \log n)$ keys per user and $O(k^2)$, $O(k \log n)$, and $O(k)$ message lengths. Their scheme is threshold-based, i.e., provides k-resilience. Another work (Garay, Staddon & Wool 2000) attempts to minimize the amount of re-carding needed per epoch by introducing so called long-lived broadcast encryption for a combinatorial threshold-based scheme. Note that schemes that provide full collusion resilience (including our own scheme) do not require re-carding at all. In (Abdalla, Shavitt & Wool 2000) the theoretical lower bounds of (Luby & Staddon 1998) were lowered by providing a relaxed definition of security. We, however, seek a solution where an adversary has a negligible probability of success.

Another related line of work goes under the

name of traitor tracing. Traitor tracing schemes (such as (Chor, Fiat, Naor & Pinkas 2000, Fiat & Tassa 1999, Naor & Pinkas 1998) among many others) were introduced to help to trace pirate smartcards or decoders back to the users who produced and distributed those pirate copies to illegal subscribers. An interesting class of schemes combine tracing with revocation capabilities, thus producing new revocation schemes. For instance, (Naor & Pinkas 2000) and later (Kogan, Shavitt & Wool 2003) describe revocation schemes based on secret sharing; their systems are, however, stateful and have a revocation threshold. Another scheme, (Naor, Naor & Lotspiech 2001), gives a tree-based stateless revocation scheme with no upper bound on the number of revocations. In this scheme, the number of keys per user is $O(\log n)$ and $O(\log^2 n)$, and communication overhead is $O(r \log \frac{n}{r})$ and $O(r)$, respectively. There is also work on trace and revocation schemes in the public setting (Naor & Pinkas 2000, Tzeng & Tzeng 2001, Dodis & Fazio 2003), where any entity can play the role of the sender, but we concentrate only on the centralized setting in this work.

Recent work, (Attrapadung, Kobara & Imai 2003), provides a stateless revocation scheme based on one-way accumulators. In the scheme, each user needs to store $O(1)$ keys and there is no transmission overhead because additional non-secret storage is used to store keys. This scheme allows to trade security against collusion for storage size and computational power. However, to make this scheme resilient to collusions of any size, it would require to store $O(2^n)$ keys and each client to process $O(2^n)$ keys in order to decrypt a message. The authors provide another scheme for non-decomposely-one-way accumulators that is more efficient than their other scheme but no construction for which is known to exist.

Other recent work, (Boneh & Franklin 1999), introduced a scheme that provides anonymous authentication of an arbitrary subset of smartcards. This scheme had the added property of identity escrow, but it requires an interactive protocol to be run between the smartcard and the server.

Our scheme is based on commutative one-way functions. It requires each card to store only 1 key and has $O(r)$ message overhead (where r, as before, is the number of revoked keys) which can further be lowered if higher dimensions are used. The scheme is stateless and fully collusion resilient. Our scheme also preserves customer privacy, which is usually not addressed in other revocation schemes.

4 Cryptographic Primitives

We use a pseudorandom function (such as AES) and, as was done above, when a message M is being encrypted with k, we write this as $Enc(M, k)$. The decryption function is represented by $Dec()$.

Our primary cryptographic primitive is that of a commutative one-way function. Two functions G and H are said to be *commutative one-way functions* if: (i) both G and H are one-way functions and (ii) $G(H(x)) = H(G(x))$. We now present a candidate for commutative one-way functions: We use RSA in a private key manner; it is commonly believed that this function is one-way, i.e., we assume the Strong RSA Assumption (Baric & Pfitzmann 1997).

An outside party (in our case this is the server S) chooses two large primes p and q, and computes $N = pq$. S also creates two values h and g that are encryption keys for N, that is, $G(x) = x^g \bmod N$ and $H(x) = x^h \bmod N$. Note that any number of commutative functions could be generated with this scheme.

As a shorthand notation, we introduce $H^i(G^j(x))$ to be i applications of H and j applications of G; note that since the hash functions are invertible (given the factorization of N) it is possible to have negative superscripts. Also, it is obvious that this scheme is commutative.

We now state an assumption for two commutative one-way functions that is key to the security properties of our paper; this assumption can easily be extended to more than two hash functions. Note that the RSA scheme described above is not such a scheme.

Assumption 1 *Given two commutative hash functions G and H and any number of triplets:*
$(i_1, j_1, H^{i_1}(G^{j_1}(x))), \ldots, (i_k, j_k, H^{i_k}(G^{j_k}(x)))$, *it is tractable to compute $H^i(G^j(x))$ only for (i, j) such that $\exists t \in [1, k] : i_t \leq i$ and $j_t \leq j$. In all other cases it is computationally intractable.*

5 A Preliminary Protocol

In this section we present a preliminary protocol that should be viewed as a "warmup" for later ones, and postpone discussion of extended versions of the protocol until Section 6. We describe the protocol in phases, then show that is satisfies the properties outlined in Section 2, and then discuss the complexity of the protocol.

The Protocol

Server Initialization
   The server keeps the following state information $(\mathcal{C}, \mathcal{R}, H, G, x, \mathcal{K})$, where these state variables are initialized to:

1. $\mathcal{C}$ is the set of all cards and is initialized to $C_0, C_1, \ldots, C_n$. Note that the server typically does not know the identity of the user who has a particular C_i (because it may have been purchased anonymously at a public store).

2. $\mathcal{R}$ is the set of all revoked smartcards and is initialized to $\emptyset$.

3. H and G are commutative one-way functions that are chosen randomly by the server.

4. x is a random element in the domain of H and G.

5. $\mathcal{K}$ is the set of keys that are used to encrypt the messages. This is initialized to the single-element set $\{H^n(G^n(x))\}$.

We now define a shorthand notation for the possible keys in the system: We use $K_{i,j}$ to denote $H^i(G^j(x))$. Note that anyone who has key K_{x_1,y_1} can use it to generate key K_{x_2,y_2} iff $x_1 \leq x_2$ and $y_1 \leq y_2$ (the "only if" part is a special case of Assumption 1). We say that a tuple $(x_1, \ldots, x_d)$ *dominates* another tuple $(y_1, \ldots, y_d)$ iff $\forall i \in \{1, \ldots, d\}, x_i \leq y_i$.

Smartcard Initialization
   The smartcards are given an ordering $C_0, C_1, \ldots, C_n$. Smartcard C_i contains key $K_{i,n-i}$. Note that each smartcard can initially generate the key $K_{n,n}$. This is depicted in Figure 1.

Sending a Message
   When the server receives a request for digital content M, it sends back the following information:

1. $Enc(M, k)$ where k is a random encryption key generated and used to encrypt M as a result of the request for M.
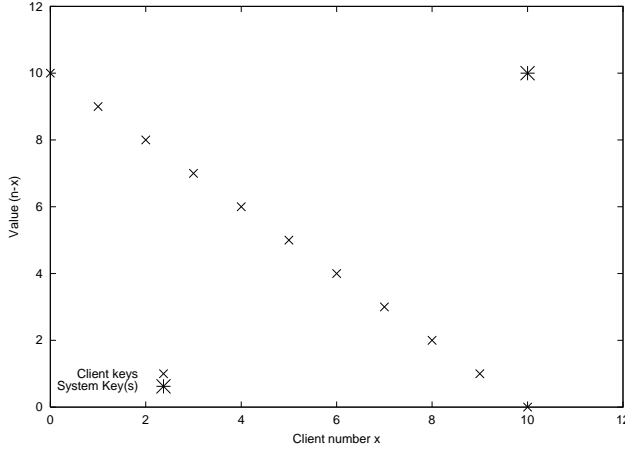
Figure 1: The keys in a system with 11 clients ($n = 10$), along with the system key $K_{n,n}$.

2. For each key $K_{i,j} \in \mathcal{K}$, the server sends $(i,j), Enc(k, K_{i,j})$.

Revoking Keys

Suppose a card containing key $K_{i,j}$ is to be revoked, i.e., it no longer should be able to reach any key in $\mathcal{K}$. Then the server does the following: Find all keys $K_{x,y}$ in $\mathcal{K}$ such that (i,j) dominates (x,y). Delete each such $K_{x,y}$ from $\mathcal{K}$ and replace it (in $\mathcal{K}$) with keys $K_{i-1,y}$ and $K_{x,j-1}$. Figure 2 illustrates changes in the key set $\mathcal{K}$ after a single revocation.

Note that it is possible that one of the indices of the keys is -1, in which case the key is ignored; we further discuss the issue of key elimination in Section 6.4.
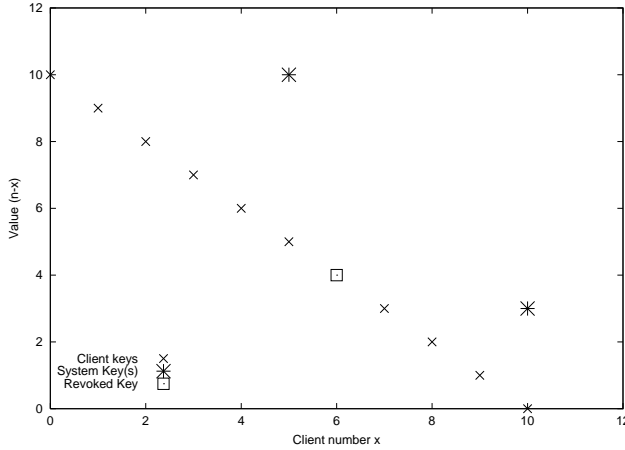

Figure 2: System in Figure 1 after client C_6 is revoked.

Proof of Scheme

At this point we state a system Invariant (1):

1. $\forall C_i \in \mathcal{C} - \mathcal{R}$ with key $K_{i,n-i}$, $\exists K_{i,j} \in \mathcal{K}$ such that $(i, n-i)$ dominates (i,j). This means that every smartcard that has not been revoked has a key that can generate a key in the master key set $\mathcal{K}$.

2. $\forall C_i \in \mathcal{R}$ with key $K_{i,n-i}$, $\forall K_{x,y} \in \mathcal{K}$, $(i, n-i)$ does not dominate (x,y). This means that every smartcard that has been revoked has a key that cannot generate any key in the master key set.

Lemma 1 *The system invariant (1) holds regardless of the number of revoked keys.*

Proof We show this by induction of the number of revocations. Suppose that there are no revocations, then property 2 holds trivially, since $\mathcal{R} = \emptyset$. Property 1 also holds since for any smartcard $C_i \in \{C_0, \ldots, C_n\}$, that smartcard has key $K_{i,n-i}$, and thus can generate $K_{n,n}$, which is in $\mathcal{K}$.

For the induction step, suppose that f smartcards have been revoked and that the state of the system is $(\mathcal{C}, \mathcal{R}, H, G, x, \mathcal{K})$. Suppose that a new client $C_r \in \mathcal{C} - \mathcal{R}$ is to be revoked. We partition the set of keys $\mathcal{K}$ into two groups $\mathcal{K}_1$ and $\mathcal{K}_2$, where $\mathcal{K}_1$ are the keys reachable from $K_{r,n-r}$ and $\mathcal{K}_2$ is the rest of the keys. Note that $\mathcal{K}_1$ is not empty, and let $|\mathcal{K}_1| = k_1$. Suppose that the keys in $\mathcal{K}_1$ are represented $\{K_{x_1,y_1}, \ldots, K_{x_{k_1},y_{k_1}}\}$. Now the revocation protocol replaces each of these keys by two sets of k_1 keys: $\{K_{r-1,y_1}, \ldots, K_{r-1,y_{k_1}}\}$ and $\{K_{x_1,n-r-1}, \ldots, K_{x_{k_1},n-r-1}\}$. We must show that if a C_j's key could reach key K_{x_i,y_i} then it can reach either K_{r-1,y_i} or $K_{x_i,n-r-1}$. Now C_j's key is $K_{j,n-j}$; suppose $j < r$, then $j \le r-1$ and $n-j \le y_i$ (since it could reach K_{x_i,y_i}). A similar argument can be made when $j > r$, and it is known that $j \ne r$. □

We now prove that the above mentioned protocol satisfies properties (1)–(4) outlined in Section 2.

Theorem 1 *The above protocol is protected, private, revocable, non-interactive, and backward secure.*

Proof The system is protected because without a smartcard, an adversary is limited to guessing the keys. The system is trivially private, because the server sends the same message to every item request. The system is trivially non-interactive by the design of the system. The invariant property proves that the system is revocable. Finally, backward security is a natural consequence of Assumption 1, but note that this requires the usage of a family of commutative hash functions. Essentially, the assumption states that, given a set of keys, one can only generate keys that are dominated by one of the known keys. □

Forward security of the scheme is shown in Section 6.5.

Efficiency

There are many efficiency metrics, we state these for f revocations:

1. *Server initialization*: The server needs to generate $n+1$ cards where each requires n commutative hash functions. The apparently $O(n^2)$ cost can easily be reduced to $O(n \log n)$ by using a divide and conquer approach. This is improved to $O(n)$ in Section 6.3.

2. *Smartcard initialization*: The card needs to do n commutative hash functions, which might be too expensive for a smartcard. We discuss extensions to the scheme that address the problem in Section 6.1 and Section 6.2, and reduce the card's workload to one modular exponentiation if an RSA-like scheme is used. Also, notice that a card needs to perform this computation only once at initialization and once per revocation.

3. *Sending a message after f revocations*: The server must send out at most $f+1$ different encryptions of the "session key" k along with each encrypted message $Enc(M,k)$. This is stated more formally later in Theorem 2.

4. *Smartcard work after key revocation*: The card must again do $O(n)$ work (of course, the more revocations in the system the less work that needs to be done). The smartcard's work, however, in this case can also be reduced by applying the same techniques as those used during smartcard initialization.

Lemma 2 *At any time in the system, any smartcard in $\mathcal{C} - \mathcal{R}$ will have exactly one key in $\mathcal{K}$ that it can reach.*

Proof We prove this by induction on the number of revocations. Clearly, when there are no revocations, there is only one key in $\mathcal{K}$ and thus the statement holds.

Suppose it also holds for f revocations. Now, suppose that another card C_i that can reach key $K_{x,y} \in \mathcal{K}$ (we know there is exactly one by the induction hypothesis) is being revoked. Clearly, any card in the range $C_{n-y}, \ldots, C_x$ can reach this key (by system invariant each of these clients must be in $\mathcal{C} - \mathcal{R}$), and thus $n - y \leq i \leq x$. Now the key $K_{x,y}$ will be split into $K_{i-1,y}$ and $K_{x,n-i-1}$. Clearly clients $C_{n-y}, \ldots, C_{i-1}$ can only reach the first of these keys and clients $C_{i+1}, \ldots, C_x$ can reach the second of these keys. $\square$

Theorem 2 *After f revocations, the worst-case number of keys in $\mathcal{K}$ is $f+1$, and the expected number is $f - (f^2/n)$ with a variance of $(2(n-f)^2 f^2 - n(n-f)f)/2n^2(n-1)$.*

Proof By Lemma 2, each key will have at most one key that it can reach, and therefore after each revocation there will only be one key that is modified in the set. Each key that is modified produces at most two keys, which means that the number of keys increases by 1, and the worst-case bound of $f + 1$ follows. For the average-case claim, the crucial observation is that the expected number of keys is 1+(the number of contiguous runs of ones in a random sequence containing f ones and $n - f$ zeros). Runs of ones in random binary sequences are a well studied topic in probability theory, and this theorem's claims follow without much difficulty from the "theorems on runs" found in (Feller 1968). $\square$

6 Extensions

6.1 Grouping

In the previous scheme, before any revocations take place (i.e., when $\mathcal{K} = \{K_{n,n}\}$), a smartcard having key $K_{a,b}$ does not need to re-compute $K_{n,n}$ from $K_{a,b}$ every time it makes an access request from the server: Instead, it computes $K_{n,n}$ only the first time, and stores it for subsequent accesses. But if at some point in time a smartcard $K_{i,j}$ is cracked and its key is posted on rogue bulletin boards on the Internet, then the server will revoke it by replacing $K_{n,n}$ with $K_{i-1,n}$ and $K_{n,j-1}$ in $\mathcal{K}$. An access request after such a change in $\mathcal{K}$ could cause the requesting smartcard to have to do $O(n)$ commutative hash function computations to compute its new access key (which is one of the two new keys in $\mathcal{K}$). This is not efficient for smartcards; thus in this section we introduce a tradeoff, which consists of the server performing more pseudorandom function evaluations and an increased message size it sends to the smartcards in response to an access request.

This is done by breaking the smartcards into c groups with at most $\lceil \frac{n}{c} \rceil$ cards per group and then creating a key for each group. Clearly, the cards would need to perform at most $O(\frac{n}{c})$ commutative function computations, but the message size would be $O(f+c)$ and the server would need to perform $O(f + c)$ pseudorandom function computations.

6.2 Offloading smartcard work

As smartcards are weak computational devices, it would be desirable for the smartcard to be able offload its expensive computations to a workstation. This cannot be done by just sending the workstation the key and having them do the work, as this would provide an attacker with the ability to easily retrieve the key. In this section we introduce a scheme that allows the smartcard to do a single exponentiation per key update. This technique can only be used if the base scheme being used is the RSA-based scheme (i.e., no collusion protection) or is a family of commutative trapdoor functions (rather than being one-way functions).

When issuing a smartcard C_i, the server, instead of sending $H^i(G^{n-i}(x))$, creates another function and its inverse (E, D), creates a key $E((H^i(G^{n-i}(x))))$, and also puts D on the smartcard. Now the smartcard can send $E((H^i(G^{n-i}(x))))$ to the workstation along with H and G. The workstation can perform the computation and send the result back to the smartcard, which can then decrypt it with D to obtain the actual key.

6.3 Reducing server's load

When the underlying scheme is RSA (no collusion-resilience), the server's load at the card creation time can be reduced to $O(1)$ modular exponentiations per smartcard, resulting in the total of $O(n)$ exponentiations. This can be done by utilizing the fact that the server knows p and q: For smartcard C_i, the server is to compute $H^i(G^{n-i}(x)) = x^{h^i \cdot g^{n-i}} \bmod N$. Consequently, the server computes $y = h^i \cdot g^{(n-i)} \bmod (p-1)(q-1)$ and performs one exponentiation $x^y \bmod N$, where $x^y \bmod N \equiv H^i(G^{n-i}(x))$.

6.4 Filtering keys

Another way to improve the efficiency of the protocol is to reduce the number of keys in $\mathcal{K}$ by filtering out unnecessary keys. There are two types of filtering:

1. *Reachability:* A key in $\mathcal{K}$ can be filtered out if it is not reachable by any of the smartcards in $\mathcal{C} - \mathcal{R}$. In our previous protocol, a key $K_{x,y}$ can be reached iff: (i) $x \geq 0$, (ii) $y \geq 0$, and (iii) $x + y \geq n$. This possibly reduces the number of keys from the bound in Theorem 2, but if the number of keys revoked is smaller than $\frac{n}{2}$ then the bound is still tight in the worst case. We omit the proof in this version of the paper.

2. *Dominance:* Suppose that there are two keys K_{x_1,y_1} and K_{x_2,y_2} where (x_1,y_1) dominates (x_2,y_2). Any key that could reach K_{x_1,y_1} could also reach K_{x_2,y_2} and thus K_{x_1,y_1} can be removed. This type of filtering does not occur in our previous protocols (trivial consequence of Lemma 2), however our subsequent protocols use this type of filtering (see sections 6.7 and 6.8).

The computational complexity of each of the above types of filtering will be discussed in section 6.7.

6.5 Adding new smartcards

In this section, we show how with slight changes to our protocol we can allow addition of new smartcards to the system. Before describing the scheme we note that in the modified protocol we assume that key filtering based on key reachability is in place.

Server Initialization
 The server keeps the following state information $(\mathcal{C}, \mathcal{R}, H, G, x, \mathcal{K}, m)$, where $\mathcal{C}$, $\mathcal{R}$, H, G, x, and $\mathcal{K}$ are defined as before. The new state variable $m \geq n$ defines the total number of cards that the system can accommodate. Now the set of keys $\mathcal{K}$ is initialized to $\{K_{n,m}\}$.

Smartcard Initialization
 Smartcard C_i contains key $K_{i,m-i}$.

Adding a Smartcard
 When a new smartcard C_{n+1} is created by S, it is added to the set of smartcards $\mathcal{C}$ and a key $K_{n+1,m-(n+1)}$ is stored in it. If $K_{n,x} \in \mathcal{K}$ for some $m - n \leq x \leq m$, then $\mathcal{K}$ is updated to $(\mathcal{K} - \{K_{n,x}\}) \cup \{K_{n+1,x}\}$. Otherwise, a key $K_{n+1,x}$ has been filtered out (after revocation of card C_n) and $\mathcal{K}$ is updated to $\mathcal{K} \cup \{K_{n+1,m-(n+1)}\}$. Notice that this does not invalidate Theorem 2's result.

The message encryption and key revocation mechanisms stay unchanged. Now the reachability rule for key filtering is slightly different: a key $K_{x,y}$ can be reached by any of the smartcards in $\mathcal{C} - \mathcal{R}$, $\mathcal{C} = C_0, C_1, \ldots, C_n$, iff (i) $0 \leq x \leq n$, (ii) $m - n \leq y \leq m$, and (iii) $x + y \geq m$.

Theorem 3 *The above protocol is forward secure.*

Proof To show that no new smartcard can generate a key to read previous messages, we use induction on the number of new smartcards. For the basic step, when there are no new smartcards in the system, the protocol is trivially forward secure.
 Now suppose that k new smartcards have been added to the system, i.e., $\mathcal{C} = C_0, C_1, \ldots, C_n, C_{n+1}, \ldots, C_{n+k}$, and the induction hypothesis (i.e., no new smartcard can read previous messages) is true. Suppose another smartcard C_l, $l = n + k + 1$, is added to the system with key $K_{l,m-l}$. The set $\mathcal{K}$ is updated to include key $K_{l,j}$ reachable by C_l. Notice that no other key $K_{x,y}$ currently in or that previously has been in $\mathcal{K}$ is reachable by C_l, because prior to addition of C_l, $\forall K_{x,y} \in \mathcal{K}$, $0 \leq x \leq n + k$ and thus $(l, m - l)$ cannot dominate (x, y). $\qquad\square$

This modification does not affect backward security of the scheme, and Theorem 1 still holds (with the proof omitted).

6.6 "Undo"ing a revocation

There are two primary reasons for revocations: (i) a smartcard was cracked and (ii) the smartcard's user did not pay some fee. In the first case, the server would never want to "undo" a revocation, but suppose that in the second case the smartcard owner paid the appropriate fee. In this case it would be desirable to undo the revocation without requiring communication with the smartcard or the user to obtain a new smartcard. A consequence of such an ability would be that the server could send messages to arbitrary subsets of the smartcards.
 To see that it is possible to undo a revocation, the server can easily compute a set of keys required as if the revocation did not happen, and use the resulting set for the new keys. A naive implementation of this would require the server to perform $O(f^2)$ work as it would have to compute $f - 1$ revocations, but this can be improved to $O(\log f)$ in a straightforward manner (we leave the details for the full version of the paper).

6.7 Higher dimensions

A three-dimensional version of the scheme has the advantage of requiring a smaller number of applications of the hashes when computing a new key after a change in $\mathcal{K}$ due to a revocation. Specifically, instead of two hash functions G and H, there would now be three such functions H_1, H_2, H_3. (As stated in the earlier discussion of the use of RSA for such commutative functions, any number of them could be generated for the same pair of primes p, q.) A smartcard would now contain key $K_{x,y,z}$ where $x + y + z = n$ and $0 \leq x, y, z \leq n$; hence the number of initial (i.e., before any revocation) smartcards m would be quadratic in n (whereas in the two-dimensional case m was n). The advantage here is that whereas in the two-dimensional case computing a new key (one that is dominated by the original key in the card) requires $O(m)$ applications of the hashes, for a three-dimensional scheme this drops to $O(\sqrt{m})$. It is not hard to see that, for a d-dimensional scheme ($d \geq 2$) with m initial keys, this worst-case work is $O(dm^{1/(d-1)})$.
 Higher dimensions therefore seem to be better in terms of cutting down on computation time. However, this advantage would be much less appealing if a higher d resulted in an uncontrolled growth in the number of keys in $\mathcal{K}$ as a function of the number of revocations f. The revocation of a key in higher dimensions is done in a natural generalization of the two-dimensional case, e.g., when a three-dimensional key $K_{i,j,\ell} \in \mathcal{K}$ is to be revoked, the server finds all keys $K_{x,y,z}$ in $\mathcal{K}$ such that (i, j, ℓ) dominates (x, y, z) and for each such key it does the following:

1. The server deletes each such $K_{x,y,z}$ from $\mathcal{K}$ and replaces it (in $\mathcal{K}$) with keys $K_{i-1,y,z}$, $K_{x,j-1,z}$, and $K_{x,y,\ell-1}$.

2. The server filters out of $\mathcal{K}$ the keys that dominate any other key of $\mathcal{K}$, and also those that are not reachable from any of the non-revoked smartcards. Filtering eliminates keys on a massive scale (more on this later). We now briefly discuss the algorithmics of such filtering.

   (a) Dominance-filtering is easily seen to be a computation of the minimal ("not dominating any other") vectors in a set of vectors, for which very efficient algorithms are known: Computing the minimal vectors among a set of μ d-dimensional vectors can be done in a simple recursive manner in time $O(\mu(\log \mu)^{d-2} + \mu \log \mu)$ (Kung, Luccio & Preparata 1975), and in time $O(\mu(\log \mu)^{d-3} \log \log \mu + \mu \log \mu)$ using a more complicated technique (Gabow, Bentley & Tarjan 1984). There are also dynamic data structuring techniques (Overmars & Leeuwen 1981) for maintaining the minimal vectors as insertions and deletions occur.

   (b) Reachability-filtering can be solved using dominance range counting data structures, which dynamically maintain a set of points so as to efficiently answer queries of the type "how many points in the data structure dominate the query point". The d-dimensional points in the data structure correspond to the active (not revoked) card

keys, and a new key $K_{i,j,\ell}$ is tested for reachability by means of a query that counts how many points in the structure dominate $K_{i,j,\ell}$ (if that number is zero then the key is filtered out). If λ denotes the number of points in the data structure, then insertions, deletions, and queries can be done in time $O(\log \lambda^d)$ each, using a variant of the usual $O(\lambda(\log \lambda)^{d-1})$-space range-query data structure (Willard 1985), somewhat faster using more complicated techniques (see (Agarwal 1997) for a survey).

In the special case where the original set of client keys is full, i.e., in d dimensions it consists of all keys $K_{x_1,x_2,...,x_d}$ where x_i are all non-negative and the $\sum_{i=1}^{d} x_i = n$ for a fixed value n, it is possible to determine if a key is reachable in $O(d)$ time. We now present how this can be done for three dimensions, which easily extends to d dimensions. A key $K_{x,y,z}$ is reachable iff (i) x, y, and z are non-negative, and (ii) $x + y + z \geq n$. The proof that this is a necessary and sufficient condition will be given in the full version of the paper.

We can also prove that $\| \mathcal{K} \| = O(df)$; the proof will be given in the full paper. Figure 3 gives experimental data on how $\| \mathcal{K} \|$ changes with f for random revocations — note that df is an over-estimate of $\| \mathcal{K} \|$, especially for larger values of f. A probabilistic analysis similar to the one we did for the case $d = 2$ clearly needs to be done, and is left for future work.
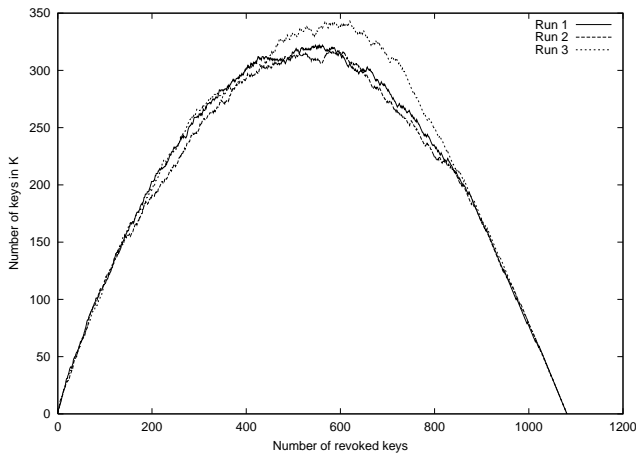

Figure 3: Number of Keys in System in (3-D).

6.8 Hypercube

It is interesting to see what happens when $n = 1$ and the d-dimensional "grid graph" of keys is one where the length $n + 1$ of each dimension is 2 (because each key's dimension is now 0 or 1). The graph so-defined is then a 2^d-node hypercube. Recall that a hypercube of dimension d is a graph that consists of 2^d vertices which are uniquely labeled with bitstrings of length d, where two vertices are connected by an edge iff their respective bitstrings differ from each other in exactly one bit position (i.e., they are equal for $d-1$ bits). A vertex has therefore degree d.

In this case of a hypercube, it is not interesting to define the initial set of keys as those for which $K_{i_1,...,i_d}$ satisfies $i_1 + \ldots + i_d = n$: This is because $n = 2$

and there would only be $m = d$ initial keys, which is too small a number to be useful. The number of initial keys would be exponential in d if we defined the initial set of keys to be those for which $K_{i_1,...,i_d}$ satisfies $i_1 + \ldots + i_d = d/2$:

$$ m = \binom{d}{d/2} \sim 2^d \sqrt{2/(\pi d)} $$

where Stirling's approximation was used.

With such an exponential (in d) number of initial keys, the work done (for still-valid smartcards) as a result of a revocation is now $O(d)$ and therefore asymptotically $O(\log m)$. Contrast this with, for example, the three-dimensional case where it was $O(\sqrt{m})$. The cost of this improvement, in terms of how $\| \mathcal{K} \|$ changes with f, is quite acceptable: Figure 4 gives experimental data on how $\| \mathcal{K} \|$ changes with f for random revocations.
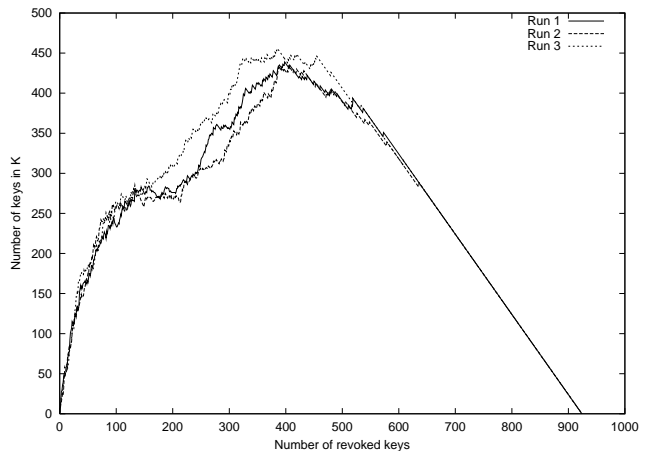

Figure 4: Number of Keys in System in a Hypercube.

7 Open Problems

In this section we list several open problems as directions for future work.

Open Problem 1 *In the higher dimensional schemes for d dimensions, what is a tight upper bound for the number of keys after f failures? What is the expected number of keys after f failures?*

Open Problem 2 *In the hypercube schemes of d dimensions, what is a tight upper bound for the number of keys after f failures? What is the expected number of keys after f failures?*

Open Problem 3 *Is there a way to achieve similar results without requiring the smartcard to perform any modular exponentiations?*

8 Conclusions

In this work we described a new revocation scheme based on commutative one-way functions that prevents owners of revoked smartcards from using the service and at the same time preserves privacy of legitimate cardholders. This scheme has the same expressive power as other revocation schemes previously described in the literature: it allows to revoke smartcards and then later "undo" a revocation, which lets the scheme to represent every possible subset of the

smartcards and be suitable for a wide range of applications.

Our scheme works with stateless receivers, i.e., no interaction between smartcards and the center is required when a smartcard is revoked or, in turn, added to the set of cards who have access to the service. Our scheme is also fully backward and forward secure in the presence of commutative one-way hash functions: no newly added cards can read previous messages, and no revoked cards can obtain access to future messages, even if they collude with any number of other revoked cards.

Our base scheme requires each smartcard to store only one key, has $O(r)$ communication overhead where r is the current number of revoked cards, and one modular exponentiation at the smartcard's end per revocation if an RSA-like scheme is used. Hence, this scheme outperforms many other known revocations schemes in terms of the number of keys stored per smartcard, communication overhead, or both.

In order to make our scheme more efficient, we introduce higher dimensions into the base model. Higher dimensions allow for a reduced number of keys, i.e., smaller communications overhead (as low as $O(\log r)$ in a hypercube) and less work performed by a smartcard. Higher dimensions are, however, generally more difficult to analyze: our scheme is not fully explored in higher dimensions, thus providing directions for future work.

References

Abdalla, M., Shavitt, Y. & Wool, A. (2000), 'Key management for restricted multicast using broadcast encryption', *IEEE/ACM Transactions on Networking* **8**(4), 443–454.

Agarwal, P. K. (1997), Range searching, *in* J. E. Goodman & J. O'Rourke, eds, 'Handbook of Discrete and Computational Geometry', CRC Press LLC, Boca Raton, FL, chapter 31, pp. 575–598.

Anderson, R. & Kuhn, M. (1996), Tamper resistance - a cautionary note, *in* 'USENIX Workshop on Electronic Commerce', pp. 1–11.

Anderson, R. & Kuhn, M. (1997), Low cost attacks on tamper resistant devices, *in* 'International Workshop on Security Protocols', pp. 125–136.

Arbaugh, W., Farber, D. & Smith, J. (1997), A secure and reliable bootstrap architecture, *in* 'IEEE Symposium on Security and Privacy', pp. 65–71.

Attrapadung, N., Kobara, K. & Imai, H. (2003), Broadcast encryption with short keys and transmissions, *in* 'ACM Workshop on Digital Rights Management (DRM'03)', pp. 55–66.

Baric, N. & Pfitzmann, B. (1997), Collision-free accumulators and fail-stop signature schemes without trees, *in* 'Advances in Cryptology – EUROCRYPT'97', LNCS, Springer–Verlag, pp. 480–494.

Berkovitz, S. (1991), How to broadcast a secret, *in* 'Advances in Cryptology – EUROCRYPT'91', Vol. 547 of *LNCS*, Springer–Verlag, pp. 535–541.

Blundo, C. & Cresti, A. (1994), Space requirements for broadcast encryption, *in* 'Advances in Cryptology – CRYPTO'94', Vol. 950 of *LNCS*, Springer–Verlag, pp. 287–298.

Blundo, C., Mattos, L. F. & Stinson, D. (1996), Trade-offs between communication and storage in unconditionally secure schemes for broadcast

encryption and interactive key distribution, *in* 'Advances in Cryptology – CRYPTO'96', Vol. 1109 of *LNCS*, Springer–Verlag, pp. 387–400.

Boneh, D. & Franklin, M. (1999), Anonymous authentication with subset queries (extended abstract), *in* 'ACM Conference on Computer and Communications Security', ACM Press, pp. 113–119.

Canetti, R., Garay, J., Itkis, G., Micciancio, D., Naor, M. & Pinkas, B. (1999), Multicast security: a taxonomy and some efficient constructions, *in* 'IEEE INFOCOM'99', pp. 708–716.

Canetti, R., Malkin, T. & Nissim, K. (1999), Efficient communication-storage tradeoffs for multicast encryption, *in* 'Advances in Cryptology – EUROCRYPT'99', Vol. 1592, Springer–Verlag, pp. 459–474.

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

Chor, B., Fiat, A., Naor, M. & Pinkas, B. (2000), 'Tracing traitors', *IEEE Transactions on Information Theory* **46**(3), 893–910.

Dodis, Y. & Fazio, N. (2003), Public key trace and revoke scheme secure against adaptive chosen ciphertext, *in* 'Public Key Cryptography (PKC'03)', Vol. 2567 of *LNCS*, pp. 100–115.

Feller, W. (1968), *An Introduction to Probability Theory and Its Applications, Volume 1*, Wiley.

Fiat, A. & Naor, M. (1994), Broadcast encryption, *in* 'Advances in Cryptology – CRYPTO'93', Vol. 773 of *LNCS*, Springer–Verlag, pp. 480–491.

Fiat, A. & Tassa, T. (1999), Dynamic traitor tracing, *in* 'Annual International Cryptology Conference on Advances in Cryptology', Springer–Verlag, pp. 354–371.

Gabow, H., Bentley, J. & Tarjan, R. (1984), Scaling and related techniques for geometry problems, *in* 'Annual ACM Symposium on Theory of Computating', pp. 135–143.

Garay, J., Staddon, J. & Wool, A. (2000), Long-lived broadcast encryption, *in* 'Advances in Cryptology – CRYPTO'00', Vol. 1880 of *LNCS*, Springer–Verlag, pp. 333–352.

Halevy, D. & Shamir, A. (2002), The lsd broadcast encryption scheme, *in* 'Advances in Cryptology – CRYPTO'02', Vol. 2442 of *LNCS*, Springer–Verlag, pp. 47–60.

Kogan, N., Shavitt, Y. & Wool, A. (2003), A practical revocation scheme for broadcast encryption using smart cards, *in* 'IEEE Symposium on Security and Privacy', pp. 225–235.

Kuhn, M. (2004), The trustno1 cryptoprocessor concept, Technical report, Purdue University.

Kumar, R., Rajagopalan, S. & Sahai, A. (1999), Coding constructions for blacklisting problems without computational assumptions, *in* 'Advances in Cryptology – CRYPTO'99', Vol. 1666 of *LNCS*, Springer–Verlag, pp. 609–623.

Kung, H., Luccio, F. & Preparata, F. (1975), 'On finding the maxima of a set of vectors', *Journal of the ACM* **22**, 469–476.

Lipton, R., Rajagopalan, S. & Serpanos, D. (2002), Spy: a method to secure clients for network services, *in* 'IEEE Symposium on Security and Privacy Systems Workshops', pp. 23–28.

Luby, M. & Staddon, J. (1998), Combinatorial bounds for broadcast encryption, *in* 'Advances in Cryptology – EUROCRYPT'98', Vol. 1403 of *LNCS*, Springer–Verlag, pp. 512–526.

McGrew, D. & Sherman, A. (1998), 'Key establishment in large dynamic groups using one-way function trees', Manuscript.

Naor, D., Naor, M. & Lotspiech, J. (2001), Revocation and tracing schemes for stateless receivers, *in* 'Advances in Cryptology – CRYPTO'01', Vol. 2139 of *LNCS*, Springer–Verlag, pp. 41–62.

Naor, M. & Pinkas, B. (1998), Threshold traitor tracing, *in* 'Advances in Cryptology – CRYPTO'98', Vol. 1462 of *LNCS*, Springer–Verlag, pp. 502–517.

Naor, M. & Pinkas, B. (2000), Efficient trace and revoke schemes, *in* 'Financial Cryptography'00', Vol. 1962 of *LNCS*, Springer–Verlag, pp. 1–20.

Overmars, M. & Leeuwen, J. V. (1981), 'Maintenance of configurations in the plane', *Journal of Computer and Systems Sciences* **23**, 166–204.

Tzeng, W. & Tzeng, J. (2001), A public-key traitor tracing scheme with revocation using dynamic shares, *in* 'Public Key Cryptography (PKC'01)', Vol. 1992 of *LNCS*, Springer–Verlag, pp. 207–224.

Wallner, D., Harder, E. & Agee, R. (1999), 'Key management for multicast: Issues and architectures', RFC 2627.

Willard, D. (1985), 'New data structures for orthogonal range queries', *SIAM Journal on Computing* **14**, 232–253.

Wong, C., Gouda, M. & Lam, S. (1998), Secure group communications using key graphs, *in* 'ACM SIGCOMM', pp. 68–79.