

A Practical Revocation Scheme for Broadcast Encryption Using Smartcards

NOAM KOGAN, YUVAL SHAVITT, and AVISHAI WOOL
Tel Aviv University

We present an anti-pirate revocation scheme for broadcast encryption systems (e.g., pay TV), in which the data is encrypted to ensure payment by users. In the systems we consider, decryption of keys is done on smartcards and key management is done in-band. Our starting point is a scheme of Naor and Pinkas. Their basic scheme uses secret sharing to remove up to t parties, is information-theoretic secure against coalitions of size t , and is capable of creating a new group key. However, with current smartcard technology, this scheme is only feasible for small system parameters, allowing up to about 100 pirates to be revoked before all the smartcards need to be replaced. We first present a novel implementation method of their basic scheme that distributes the work among the smartcard, set-top terminal, and center. Based on this, we construct several improved schemes for many revocation rounds that scale to realistic system sizes. We allow up to about 10,000 pirates to be revoked using current smartcard technology before recarding is needed. The transmission lengths of our constructions are on par with those of the best tree-based schemes. However, our constructions have much lower smartcard CPU complexity: only $O(1)$ smartcard operations per revocation round (a single 10-byte field multiplication and addition), as opposed to the complexity of the best tree-based schemes, which is polylogarithmic in the number of users. We evaluate the system behavior via an exhaustive simulation study coupled with a queueing theory analysis. Our simulations show that with mild assumptions on the piracy discovery rate, our constructions can perform effective pirate revocation for realistic broadcast encryption scenarios.

Categories and Subject Descriptors: K.6.5 [**Management of Computing and Information Systems**]: Security and Protection

General Terms: Algorithms, Security

Additional Key Words and Phrases: Broadcast encryption, smart cards

A preliminary version of the paper appeared in “*IEEE Symposium on Security and Privacy, Oakland, CA, May 2003*”, under the title “A Practical Revocation Scheme for Broadcast Encryption Using Smartcards”, © 2003 IEEE.

This work was supported, in part, by a grant from Giesecke and Devrient, GmbH.

Authors’ address: N. Kogan, Y. Shavitt, and A. Wool, School of Electrical Engineering, Tel Aviv University, Ramat Aviv 69978, Israel; email: noam.kogan@gmail.com; shavitt@eng.tau.ac.il; yash@acm.org.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or direct commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.
© 2006 ACM 1094-9224/06/0800-0325 \$5.00

1. INTRODUCTION

1.1 Channel Model Definition

Many digital content and multimedia distribution systems are based on a uni-directional broadcast distribution channel, such as satellite or cable. Access to the content is regulated by encryption: only paying customers should have the keys. In typical systems, a hierarchy of keys is used for access control: the keys that encrypt the video stream are encrypted by program keys (often called key encryption keys, or KEKs), which, in turn, are encrypted by weekly keys, etc. Each user receives a set-top terminal (STT), which is coupled with a tamper-resistant smartcard (SC). The STT decrypts the video stream, while the SC stores the key material (KEKs), and uses them to perform the decryption of lower level keys. The problem known as Broadcast Encryption is how the content provider can communicate securely with a subset of the users, over the insecure broadcast channel, while minimizing the key management overhead. Scalability is crucial, since typical systems are large: tens of millions of users.

We focus on systems that do not have a reverse (uplink) channel from the STT back to the center through which key management functions can be performed,¹ which is the case for most European satellite TV systems. We assume that purchasing offers, changing a user's subscription, etc, is performed through voice/data side-band channels that are not via the STT.

1.2 Key Management Characteristics

The satellite TV industry is characterized by acute price sensitivity to the SCs used. This is because the large numbers of users in these systems. Therefore, very cheap SCs are often used. Such platforms typically have the following limitations:²

- A very restricted amount of secure memory (EEPROM)—typically 1–8 KB. This severely limits the number of keys that can be kept on the SC.
- A weak CPU: an 8-bit, 3.57MHz CPU is typical.
- The communication between the SC and the STT is extremely slow: 9.6–38.4 Kbit/sec.

Thus, in most systems the SC only decrypts keys and the STT decrypts the content (video).

In the systems we consider, there is a single key that is used by the center to encrypt the current content, and which is made known to all the users by the hierarchy of KEKs, as explained in the previous section. When the center needs to change this key, it broadcasts a message. Based on the content of this transmission and the secret data (KEKs) already on the SCs, all the legitimate

¹In our model, the center can distribute keys only via the broadcast channel. If an uplink exists, user revocation is much easier: The user needs to connect to the center once every period to obtain the keys for next period. Therefore, the center can simply refuse to provide the new keys to revoked users.

²Numbers are typical for cheap SCs circa 2000.

SCs should be able to compute the new key. We exclude solutions that use public-key cryptography, since public-key operations are typically too slow for the cheap SCs in use.

Note that the transmission overhead required by the key management is an important parameter, since key material needs to be broadcast repeatedly (not all STTs are powered up all the time; hence, the need for repetitions).

1.3 User Revocation

Piracy in digital content and multimedia distribution systems is very common: commercial pirates duplicate and sell the key material of users who have access to the content, causing a significant loss of revenue to the content providers. Our goal is to design an efficient and practical revocation scheme, especially suited for combating piracy, i.e., to render pirate decoders and SCs dysfunctional. The general approach is to create new periodic keys that would be known to all the legal users, but not to the pirates. The scheme should allow for many revocation rounds in order to cope with a monotonically expanding pirate population. The scheme should also perform permanent revocation, since the revocation of the pirate users is not temporary, but rather “for good.”

Therefore, we consider the following scenario. A group of n users shares the same key with which the content is encrypted. A center is responsible for controlling the decryption capabilities of these users. The center prepares and gives each user a set of secret data, which is stored on the user’s SC. By the beginning of revocation round i , the center learns that some new r_i pirate users are violating the terms of their usage license and wishes to disallow the subgroup of accumulated $R_i = \sum_{j=1}^i r_j$ pirate users from decrypting any new content. The center creates a new key, which should become known to the $n - R_i$ nonrevoked users. Further content should be encrypted with the new key.

For a given revocation scheme, the important factors that determine its efficiency are:

- Communication overhead: the length of the messages sent by the center to renew the keys.
- Storage overhead: the number of keys the users store in their SCs.
- Computational overhead: the computational load to renew the keys, especially by the users.

Revocation can be trivially achieved as follows: the center issues each user a unique key. To revoke R_i users (of which r_i are new), the center transmits a new periodic key via unicast messages to each of the $(n - R_i)$ nonrevoked users, encrypted with the unique key of each user. However, this scheme has a very high communication overhead, which also translates into a very high response time.

We assume that the task of disabling the viewing capabilities of users that voluntarily stop their subscription can be implemented in other ways. For instance, by forcing the users to return their SCs (to get back a deposit), or by sending each of them a “sleep” command encrypted with an individual unicast key. Thus, we only need to consider revoking pirate users, whose number is significantly smaller than that of users leaving the system voluntarily.

Table I. Notation Used in this Paper

U	the set of users in the system.
n	the total number of users (i.e., legitimate keys) in the system; $n = U $.
t	the maximal number of users that can be revoked before recarding is needed.
r_i	the number of newly revoked users (keys) in revocation round i .
R_i	the aggregate number of revoked users (keys) up to and including revocation round i .
R	the total number of revoked users (keys) in general (without referring to a particular revocation round).
k	an upper bound on the coalition size (number of keys) the adversary can assemble.
w	the transmission length of the revocation messages (in keys).
d	the dilation factor between the sizes of consecutive schemes.
s	the number of keys stored in the SC (which is also the number of revocation schemes for multiround revocation)

Much of the notation we use is summarized in Table I for quick reference.

1.4 Related Work

The study of broadcast encryption was initiated in Berkovits [1991] and Fiat and Naor [1994] and the efficiency of broadcast encryption schemes is studied in Blundo and Cresti [1994], Blundo et al. [1996], and Luby and Staddon [1998]. Broadcast encryption schemes usually fall into one of three categories: combinatorial constructions, secret-sharing constructions, and tree-based constructions.

In the combinatorial approach [Luby and Staddon 1998; Abdalla et al. 2000; Kumar et al. 1999; Garay et al. 2000], the keys of compromised SCs are no longer used. Luby and Staddon [1998] showed a trade-off between the transmission length w and the s keys in the SC, showing that these constructions either have very long transmissions or prohibitive storage requirements on the SC. Abdalla et al. [2000] allowed a factor f of free riders to enjoy the broadcast content, breaking away from the Luby and Staddon [1998] bounds. The Kumar et al. [1999] constructions are based on cover-free sets of combinatorial design theory and have transmission length $w = O(t \log n)$ and $w = O(t^2)$. Garay et al. [2000] introduced the notion of long-lived broadcast encryption, analyzing the number of recordings needed per epoch.

Naor and Pinkas [2000] used secret sharing to revoke up to t users. The scheme is information theoretic secure against coalitions of size $k = t$ with only $w = t$ messages and $O(t^2)$ calculations at the user, or $w = 2t$ messages and only $O(t)$ calculations at the user. For multiround revocation, they suggested using t revocation schemes with sizes $1, 2, \dots, t$, to be used one after the other. This scheme can revoke a total of up to t users before recarding of all the SCs is needed. We elaborate on this scheme later, as it is our starting point.

Tree-based revocation schemes were proposed in Wong et al. [2000], Wallner et al. [1998], Canetti et al. [1999a], Canetti et al. [1999b], Naor et al. [2001], Halevy and Shamir [2002], and Pinkas [2001], with stateless and stateful variants and are currently considered to have the best combination of properties. In all of them, the number of keys that the SC stores is polylogarithmic in the number n of users.

Quite a few stateful schemes are based on the LKH (Logical Key Hierarchy) algorithm and its many variants [Wallner et al. 1998; Wong et al. 2000; Canetti

et al. 1999a; Pinkas 2001]. The basic LKH scheme was suggested in the context of secure multicast, which builds secure communications on top of the Internet's multicast layer. In LKH schemes, every user keeps a personal key composed of $\log(n)$ keys and the center keeps a (binary) tree of keys. When a single user is revoked, the center must change all the keys in the tree that are in the path from the revoked user's leaf to the root. All the nonrevoked users must change their keys in the intersection between the path from their leaf to the root and the path from the revoked user's leaf to the root. Therefore, a message revoking a single user consists of $O(\log n)$ keys. Graphically speaking, the LKH schemes "rebuild" the key tree after revocation takes place.

A major drawback of the LKH family is that the users must modify (update) their state whenever the group key is changed (either when users leave or join the group). This means that a user that was off-line for a while, is required to receive and process messages that enable him to "synchronize" with the group. One approach is to retransmit the original revocation messages. Another approach [Pinkas 2001] is to tailor a special message for that particular rejoining user, which corresponds to the number of updates that he missed. However, both approaches raise the bandwidth requirements of the scheme.

In the stateless version, the two best schemes are those of Naor et al. [2001] and Halevy and Shamir [2002]. Naor et al. [2001] designed the so-called *Subset Difference* scheme that enables the center to revoke any subset of users, $S \subset U$. In order to do that, they place the users in the leaves of a complete binary tree and then use the tree representation in order to define a base family of subsets of U with the property that for any $S \subset U$, its complement $U \setminus S$ may be covered by a disjoint union of $O(|S|)$ base subsets. The content is made accessible to all nonrevoked users by sending an encrypted message containing the new key to each of the base subsets in the cover of $U \setminus S$. In order to decrypt at least one of those messages, each user has to store $\frac{1}{2} \log^2(n)$ keys and perform $O(\log(n))$ computations.

Subsequently, Halevy and Shamir [2002] introduced the *Layered Subset Difference* technique, which offers an improvement of the Subset Difference scheme of Naor et al. They observed that the base family of subsets that "spans" all of 2^U may be reduced, at the expense of a factor of 2 in the number of base subsets that are required in order to cover a given subset. As a result, their scheme requires that each user holds only $\log^{3/2}(n)$ keys, while raising the revocation message length by a factor of two.

The main strength of these stateless schemes is that they have very short revocation messages, even when revoking a large part of the entire user population, as is the case in Pay-Per-View events, where revocation is temporary. When performing multiround revocation, for the scenario when the revoked user population is monotonically growing, they "carry" the "holes" in the tree (previously revoked users) to future revocation rounds.

1.5 Contributions

Our starting point is the multiround secret-sharing-based revocation scheme of Naor and Pinkas [2000]. This scheme can perform multiple revocation rounds,

revoking a total of up to t pirates. However, with current technology, this scheme is only feasible for small system parameters. Because of the SC's weak CPU, slow communication rate, and restricted RAM and EEPROM, the scheme is only useful for $t \approx 100$ after which recarding of all the SCs is required. Thus, *a priori* this scheme seems inferior to the tree-based schemes of Wong et al. [2000], Wallner et al. [1998], Canetti et al. [1999a, 1999b], Naor et al. [2001], and Halevy and Shamir [2002].

We first show how to rearrange the computations of the scheme, so the work is distributed better among the SC and the STT. Specifically, the $O(t)$ calculations at the user can be split into $O(1)$ *secret* calculations that need to be done on the SC (a single-field multiplication and addition, in a field of typically 80 bits), and $O(t)$ *nonsecret* calculations that can be done on the much more powerful STT. This allows us to bypass the CPU, RAM, and communication bottlenecks caused by the SC.

Since the scheme requires recarding of all the SCs after t pirate cards have been revoked, there is a high incentive to contemplate much larger t values, such as $t \approx 10000$. For such a large t , the SC's limited EEPROM size prevents having $s = t$ schemes as suggested by Naor and Pinkas [2000], since each scheme requires keeping a share on the SC. Hence, we separate the number of schemes s from t and use $s \ll t$, s being an unconstrained parameter.

Our first multiround construction is the *Triangular* system. It has s schemes of sizes $d, 2d, \dots, t$, with $d = t/s$. In the *Triangular* system, the center is capable of revoking an average of $d = t/s$ new users per round, but an overall total of t users, since users revoked in different rounds can collude.

Next, we recall that if a secret-sharing scheme is of degree d , but the center needs only to revoke $r < d$ pirate cards, it can use the scheme by "revoking" an additional $d - r$ dummy users with fake identities. This leads us to construct a *Rectangular* system, which has s schemes, all with size t . In this system, the overall number of revoked users is still t , but the center has the flexibility of revoking a large number of users in a single round at a cost of a single polynomial.

The transmission length of the *Triangular* scheme is $O(R)$ per round, which is on par with tree-based systems. The *Rectangular* scheme has a transmission length of $O(t)$ per round, regardless of the actual number of users revoked in that round. For s revocation rounds, the total length of transmission is $O(st)$ for both schemes. However, the total for the *Triangular* scheme is smaller by a factor of 2 and, furthermore, the *Triangular* scheme's first rounds use much shorter transmissions.

Both schemes can use various operational strategies to trigger the revocation rounds. We evaluate both schemes, with several possible strategies, under mild assumptions on the piracy discovery rate, via an exhaustive simulation study coupled with a queueing theory analysis. Our simulations show that our constructions can perform effective pirate revocation for realistic broadcast encryption scenarios, allowing several years of operation before recarding becomes necessary, arguably competing with the best tree-based schemes.

A preliminary version of this paper can be found in Kogan et al. [2003].

1.6 Organization of This Paper

In Section 2, we describe the attack model we are protecting against. In Section 3, we show how to reduce the computational, communication and memory load on the SC. In Section 4, we introduce the details of our new schemes. The simulation study and queueing analysis are described in Section 5; we conclude with Section 6.

2. ATTACK AND DEFENSE MODELS

2.1 Attack Model

Piracy in digital content and multimedia distribution systems is very common. Commercial pirates duplicate and sell the key material of users who are entitled to use (watch) the content. Content providers are estimated to suffer \$100M each year in lost revenue because piracy [cf. BBC 2001; Hackwatch 2002].

By far, the most common attack scenario is for the pirates to extract the keys from a SC and create many duplicates (clones) of that SC. This is feasible because even though the SC is supposed to be tamper resistant, in reality there are ways to compromise it—some quite cheap [cf. Anderson and Kuhn 1996; Anderson and Kuhn 1997].

We argue that the pirates are limited in the number of SCs they can obtain and hack in any given time period. Therefore, the large number of cloned SCs originate (cryptographically) from a much smaller number legal SCs, whose key material (SC) was compromised. Note that the revocation scheme is only concerned with the original compromised SCs and not with the clones: revoking the key of the original SC will also revoke all its clones. Therefore, when we speak of a *pirate card*, we are referring to a hacked (original) card rather than a clone.

2.2 Revocation Process

We assume a reactive enforcement system to combat piracy. We assume that on an ongoing basis, law enforcement agencies and content provider companies capture cloned SCs. These cards are analyzed by the content provider, who then identifies the original pirate card from which the clones were made.

As detailed in Section 1.3, the purpose of revocation is to create a new system key, which would enable all the paying users to access the content and which would be unknown to the pirate users. Such scenarios are described in Briscoe [1999].

We assume that the revocation process works in epochs: once a new pirate card is identified, its key can be scheduled for revocation in the next revocation round. Since the revocation process entails a communication overhead, the content provider has a trade-off between **communication overhead** in the broadcast channel—if the compromised SC is immediately revoked (and, consequently, all the cloned SCs originating from it), and **potential revenue loss**—if the content provider performs revocation of all the compromised SCs each predefined epoch (which lets the pirate users watch more content for free).

3. AVOIDING THE SC COMMUNICATION, RAM, AND CPU BOTTLENECKS

In this section, we show how to implement the basic Naor and Pinkas [2000] scheme for a single revocation round in a way that dramatically reduces both the computation on the SC and the data that needs to be transferred to the SC over the slow channel between it and the STT. We offload almost all of the work to the much stronger (and possibly insecure) processor in the STT and let the weak SC handle the secret part of the computation—which turns out to consist of a single-field multiplication and one addition.

3.1 The Naor and Pinkas [2000] Secret-Sharing Revocation Scheme for a Single Round

Before presenting our modifications, we briefly describe the Naor and Pinkas [2000] revocation scheme for a single revocation round. The basic Naor–Pinkas one-round scheme can revoke up to t users, with a communication overhead of $O(t)$, and is information-theoretic secure against a coalition of all the t revoked users. The scheme is based on threshold secret sharing, using Shamir’s polynomial based $(t + 1)$ -out-of- n secret sharing [Shamir 1979]. The secret sharing is performed over a field F whose elements are encryption keys and user identities: typically, 80 bits suffice for the field elements.

3.1.1 The Basic Scheme

- **Initialization:** The center generates a random polynomial P of degree t over the field F , s.t. $P(0) = S$, S being the randomly chosen key of a symmetric algorithm (e.g., $|F| \geq 80$ bit), to be used after the revocation. The center provides each user with an identifier u and a corresponding share $P(u)$. Given any $t + 1$ shares, a user can interpolate the polynomial and reveal $S = P(0)$.
- **Revocation:** The center learns the identities of t users whose keys should be revoked. Subsequently, it broadcasts the identities and the personal shares of these users:

$$\{u_1, P(u_1)\}, \dots, \{u_t, P(u_t)\}$$

Each nonrevoked user u can combine his personal share $P(u)$ with these t shares and interpolate P to compute the key $S = P(0)$ via the Lagrange interpolation formula. The center uses S as the new group key with which it encrypts further content to the nonrevoked users.

Given the identities u_i of the revoked users, and their shares $P(u_i)$ in the degree t polynomial $P()$, the Lagrange interpolation formula is

$$P(0) = \sum_{i=0}^t \prod_{j \neq i} \frac{u_j}{u_j - u_i} P(u_i)$$

where u_0 is the identity of the user performing the calculation, and u_i is the identity of the i th revoked user ($i = 1, \dots, t$). $P(0)$ is the new key that is set after the revocation. The construction of the secret sharing ensures that even the coalition of all the t revoked users will not have enough shares to compute $P(0)$, whereas every nonrevoked user will.

- **Storage and communication overhead:** The secret data that each user has to keep is just a single element of F (the identity u need not be kept secret). The revocation message is of length $t(|F| + \log(n))$, if the identities are defined in a small subset of F .

3.1.2 Reducing the Computational Overhead at the User End. Naor and Pinkas [2000] offer a way to reduce the computational overhead at the user end. As described in the previous section, the computation of the new group key by a user involves the interpolation of the free coefficient of P and requires $O(t^2)$ multiplications using Lagrange interpolation.

Let c_i denote the inner-product function on the identities except u_i . The Lagrange interpolation formula can then be rewritten as:

$$c_i = \prod_{j \neq i} \frac{u_j}{u_j - u_i} \text{ for } i, j = 0, \dots, t; \quad P(0) = \sum_{i=0}^t c_i P(u_i)$$

In order to solve for $P(0)$, a user needs $t + 1$ shares $P(u_i)$, one of which is his own, and $t + 1$ c_i coefficients. Calculating the $t + 1$ c_i coefficients requires $O(t^2)$ calculations. However, almost all of this calculation is identical for all the users and can be precomputed by the center. Only $O(t)$ of these calculations differ between users: these are the calculations involving the identity of each nonrevoked user u_0 . Therefore, Naor and Pinkas [2000] suggests that the center should compute t coefficients, denoted by c'_i , as follows:

$$c'_i = \prod_{j \neq i} \frac{u_j}{u_j - u_i} \text{ for } i, j = 1, \dots, t$$

Each nonrevoked user u_0 who receives the broadcast completes the calculation of the c_i coefficients with $O(t)$ calculations and computes his own c_0 , as follows:

$$c_i = c'_i \frac{u_0}{u_0 - u_i} \text{ for } i = 1, \dots, t; \quad c_0 = \prod_{j=1}^t \frac{u_j}{u_j - u_0}$$

Each user then uses his own share $P(u_0)$, together with the t broadcast shares of the revoked users and the computed c_i coefficients to compute $P(0)$ according to the Lagrange interpolation formula.

3.2 Moving Computation Out of the Smartcard

Performing $O(t)$ calculations on the SC has several drawbacks. First, the STT needs to transmit $O(t)$ data (for an 80-bit field we have 14 bytes per revoked user for the identity (4 bytes) and share (10 bytes)) to the SC over the slow communication channel. Second, the SC needs enough RAM to keep this data during computation. Third, the SC CPU needs to perform $O(t)$ field operations on 80-bit numbers. With current technology, each of these bottlenecks individually constrains the maximal value of t , which is also the maximal number of pirate cards that can be revoked before recarding is needed, t , to ≈ 100 .

Our observation is that almost all the $O(t)$ calculations at the user are *nonsecret*. In fact, all the information that is received in the broadcast is not

secret: this includes the t shares $P(u_i)$ of all the revoked users, their t identities u_i , and the t coefficients c'_i (which were computed from the nonsecret identities). The only reason that the center does not compute *all* the c_i coefficients is that the interpolation performed by each user u_0 involves that user's own identity.

This means that the computation of all the c_i coefficients, including c_0 , can be performed in the untrusted STT. The SC only needs to compute the terms that involve its secret share.

Note that each STT should know the identity u_0 of its SC (but not the secret share $P(u_0)$). Thus, the STT can also sum all but one of the terms that comprise $P(0)$:

$$P(0)' = \sum_{i=1}^t c_i P(u_i)$$

The STT only needs to transmit the values $P(0)'$ and c_0 to the SC. The SC only needs to compute one term that involves its secret share $P(u_0)$:

$$P(0) = P(0)' + c_0 P(u_0)$$

Therefore, all of the computation, RAM memory, and communication with the SC are $O(1)$: The computation is a single-field multiplication and addition in the field, the communication size is 2 elements in the field, which are 10 bytes each (and possibly some additional bytes of management data), and only a few dozen bytes of SC RAM are needed, only to perform the field multiplication and addition.

This observation allows us to bypass the CPU, RAM, and communication bottlenecks caused by the SC, and we can contemplate using much larger secret-sharing scheme sizes.

4. THE NEW MULTIROUND REVOCATION SCHEMES

In this section, we present our two multiround constructions. For comparison, we first describe some details of the multiround scheme of Naor and Pinkas [2000].

4.1 Multiround Revocation

The Naor and Pinkas [2000] multiround scheme has t revocation schemes RS_1, RS_2, \dots, RS_t of secret-sharing sizes $1, 2, \dots, t$, respectively (requiring to store $s = t$ keys on the SC), which are used one after the other: in round i , revocation scheme RS_{R_i} is used for revoking the aggregated set of R_i users. If more than $r_i > 1$ new users are revoked in round i , then $r_i - 1$ schemes will be skipped ahead.

The scheme has a transmission length of $w = O(r_i)$ (the broadcast includes the IDs and shares of the r_i newly revoked users), with an additional off-line "maintenance" channel, which is assumed to enable the center to broadcast data to users at times when bandwidth is less expensive, e.g., at night. The center uses this maintenance channel to broadcast to each of the n users the $O(r_i \cdot t)$ shares of the revoked users in all the polynomials with higher degrees than RS_{R_i} . This effectively reduces the degree of the other polynomials by r_i .

If it is required to maintain the capability to revoke up to t users after each revocation round, then recarding of all the SCs is needed: the center generates new shares and sends them (in unicast) to all the nonrevoked users, in order to renew the schemes.

The authors [Naor and Pinkas 2000] also propose another scheme for multi-round revocation, which lifts Shamir's secret-sharing scheme to the exponent and relies upon the Decisional Diffie–Hellman problem for the scheme's security. We do not repeat the details of this scheme, since, as explained in Section 1.2, we focus on schemes that do not utilize public-key cryptography.

4.2 Avoiding the Maintenance Channel

At the end of each revocation round, Naor and Pinkas [2000] propose to use an “offline” maintenance channel, with a goal of keeping future revocation messages short. Using the maintenance channel, the center broadcasts the shares of the revoked users in *all* the future polynomials. The users are supposed to store these shares for future revocation rounds. This way, the revocation message in round i would be of length $O(r_i)$, the number of new pirate cards in round i , as opposed to $O(R_i)$, the aggregate number of revoked pirate cards in rounds $1, \dots, i$. Unfortunately, this idea has several shortcomings.

First, the revocation message itself does not contain all the information needed for the calculation of the new group key. Thus, the users need to receive and store all the identities and shares of the previous R_{i-1} revoked users. This requires $O(t(t-1)/2)$ nonvolatile storage on the STT, which may not be available.

Next, the “maintenance channel” idea is problematic for new users, or for users who missed one or more revocation rounds. If all the previous revocation messages are to be repeatedly retransmitted, it would raise the real length of the transmission to $O(t^2)$ for each repetition.

Finally, with this approach, the center cannot reduce the computational load at the user by precomputing most of the Lagrange interpolation formula, since broadcasting the c_i 's would raise the transmission length to be $w = O(R_i)$. Therefore, the users would need to either perform $O(R^2)$ calculations, or store $O(R_i)$ values in nonvolatile memory and perform $O(R_i r_i)$ calculations.

Instead of using a maintenance channel, we take the simpler approach, which is to broadcast the appropriate shares on the revocation round in which the polynomial is used. Using our approach, the user does not require nonvolatile memory, the computational load at the user can return to being $O(R_i)$ (of which only $O(1)$ is done on the SC), and a rejoining user is not required to process all the past revocation messages.

4.3 Dealing with a Partial Revocation

The basic Naor and Pinkas [2000] scheme is good for a single revocation. Furthermore, Naor and Pinkas [2000] explains that to revoke more than d users it is not safe to use multiple schemes with different polynomials of the same degree d , one per revocation round, since pirates revoked in different rounds can collude: A coalition of pirates that compromised d SCs in one round and

d SCs in another round, has (together with the broadcast revocation message) access to $2d$ shares in *each* of the polynomials, which suffice to compute both new keys. Instead, the authors use t polynomials P_i of degrees $1, 2, \dots, t$.

However, if the secret-sharing scheme is of degree d , but the center only needs to revoke $r < d$ pirate cards, it can use the scheme by “revoking” an additional $d - r$ dummy users with fake identities. Broadcasting such dummy shares in polynomial P_1 does not give pirates any information about shares in some other polynomial P_2 . Therefore, there is no problem in using multiple secret-sharing schemes with the same degree t , as long as the *aggregate* number of revoked users is $R \leq t$ and all the partial revocations use fake user identities.

In this scheme, if a pirate card was revoked in round i , the center has to continue revoking this card in future revocation rounds (rounds $j > i$). Note that this is required in stateless tree-based revocation schemes such as Naor et al. [2001] and Halevy and Shamir [2002], as well.

4.4 The Triangular Construction

Instead of using $s = t$ polynomials of degrees $1, 2, \dots, t$, we introduce a dilation factor d and build s schemes RS_1, \dots, RS_s of degrees $d, 2d, \dots, t$, with $d = t/s$. Scheme RS_i is capable of revoking $d \cdot i$ users. Each SC stores a share in each of the s schemes. Since we have decoupled the maximal number of revoked users t from the number s of schemes, we can use a large $t \approx 10,000$ with an s that can fit in the SCs’ limited EEPROM, e.g., $s \approx 100$. We call a scheme designed with increasing size (i.e., increasing degree of the secret-sharing polynomial) a *Triangular* system.

In the i th revocation round, the center revokes the R_i known pirate users (of which r_i are new), subject to the constraint that $r_i \leq d \cdot i - R_{i-1}$. If not all of the new r_i pirates can be revoked with scheme RS_i of degree $d \cdot i$, then we *skip ahead* one or more revocation schemes. For the i th round, we have two possible candidate schemes, RS_{j-1} and RS_j , for which $d(j-1) \leq R_i \leq d \cdot j$. We select scheme RS_j (“round up”) if $R_i - d(j-1) \geq d/2$ and select RS_{j-1} (“round down”) otherwise. Note that if we round down, then up to $d/2$ known pirates are not revoked in this round. Furthermore, in each revocation round, either the number of pirates in carry (“leftovers”) or the dummy users used (revocation “credit”³) are zero. It is possible to choose a different threshold for *skipping ahead* schemes.

The transmission length of the *Triangular* scheme is linear: $w_i = O(d \cdot i)$. The center is capable of revoking an average of $d = t/s$ new users per round, but an overall total of t users, since users revoked in different rounds can collude.

4.5 The Rectangular Construction

Our second construction, called the *Rectangular* construction, has s revocation schemes, of equal secret-sharing sizes t each. In this system, the overall number of revoked users is still t , but the center has the flexibility of revoking up to t users in a single round.

³Dummy users can be replaced with real ones in future rounds.

In the i th revocation round, the center revokes all the known pirate users, subject to the constraint that $R_i \leq t$, and completes the round by also “revoking” $t - R_i$ fake users.

The *Rectangular* scheme has a transmission length of $O(t)$ per round regardless of the actual number of users revoked in that round. For s revocation rounds, the total length of transmission is $O(st)$ for both schemes, however, the total for the *Triangular* scheme is smaller by a factor of 2 and *Triangular* scheme’s first rounds use much shorter transmissions.

4.6 Incremental Computation at The Center

We observe that for a large t , the $O(t^2)$ calculations required at the center can become a bottleneck as well. However, the $O(t^2)$ calculations at the center can be performed *incrementally*: performing only $O(rt)$ computations for a revocation round with r new users to revoke. The steps needed in the *Rectangular* scheme follow. The *Triangular* scheme uses a similar procedure.

- At initialization of the revocation scheme, the center chooses $(t + 1)$ dummy values for all the identities (u_i ’s), and performs $O(t^2)$ calculations to obtain the t c_i ’s, that are ready to be broadcast for this choice of revoked users. Note that this calculation can be carried out outside the system (for example, in a strong off-line computer), and be imported to the center.
- With each new user u_a to revoke, the center takes out a dummy user u_d from the list2 and updates the t c_i ’s, by performing for each of them :

$$c_i^{new} = c_i^{old} \frac{u_a}{u_a - u_i} / \frac{u_d}{u_d - u_i}$$

The new c_i coefficients are now ready to be broadcast for the new revocation list. This requires $O(t)$ calculations.

This incremental approach allows us to bypass the potential CPU bottleneck at the center, for large values of t . In addition, if significantly less than t users are revoked, then the calculations the center actually performs are correspondingly lowered, since the initial calculations involving the identities of the dummy users remain part of the revocation message.

5. SIMULATION

In this section we evaluate the system’s behavior via an exhaustive simulation study. We examine different piracy rates, explore the operational aspects of several revocation triggers, and evaluate the following parameters: transmission length, scheme lifetime (without recarding at all), average piracy levels, a pirate’s lifetime, sensitivity to bursts of pirates, and pirate “leftovers” after revocation.

5.1 Model Definition

In our model, the number of pirate users that exist in the system, and the number of hacked SCs from which they originate (pirate cards), are unknown to the center. What the center does know is the number of pirate cards that are

discovered by enforcement authorities and the content provider companies per time period.

We set the total number of pirate cards that the schemes need to be able to revoke at $R = 10,000$. This seems to be a reasonably large number of hacked cards for a population of $n \approx 10^8$: $R \approx \sqrt{n}$. Therefore, the maximal degree of the polynomials that the schemes use is $t = 10,000$. We chose the number of shares each SC needs to store to be $s = 40$, which is a value that even the cheapest SCs can handle (all the shares can fit in about 640 bytes of EEPROM).

We simulated both the *Rectangular* construction, which has $s = 40$ schemes of a secret-sharing size of $t = 10,000$, and the *Triangular* construction, which has $s = 40$ schemes with linearly increasing secret-sharing sizes 250, 500, 750, ..., 9750, 10,000.

For each scheme, we evaluated two possible triggers for performing the revocation. The first trigger for a revocation round is the accumulation of 250 pirate cards. The second trigger is time: perform a revocation round if 3 months have passed from the last revocation round (i.e., four revocation rounds per year, one round each quarter). The two constructions and two trigger mechanisms give us four variants to compare:

1. Scheme A: *Rectangular* construction with a revocation trigger of each 3 months.
2. Scheme B: *Rectangular* construction with a revocation trigger of accumulating 250 pirate cards.
3. Scheme C: *Triangular* construction with a revocation trigger of each 3 months.
4. Scheme D: *Triangular* construction with a revocation trigger of accumulating 250 pirate cards.

Note that the two trigger types are essentially equivalent for a discovery rate of $\lambda = 250/13 \approx 19.2$ pirate cards per week. At such a rate (250 pirate cards per quarter), all four variants finish 40 revocation rounds, on average, in 10 years' time.

We model the number of discovered pirate cards as a random variable having a Poisson distribution with an average value of λ pirate cards discovered per week. We varied λ between $2 \leq \lambda \leq 50$ pirate cards per week. The simulation's time unit is one day. For each simulated day, we randomly generate new pirate cards (according to their weekly discovery rate) and check whether a revocation round should be performed.

5.2 Results

5.2.1 The Number of Known Pirate Cards in the System. Figure 1 shows the number of known pirate cards as a function of time, for a discovery rate of $\lambda = 14$ and $\lambda = 28$ pirate cards per week. In all four schemes we see that the number of known pirate cards in the system grows linearly and then drops at each revocation point. In Figure 1(top), schemes A and C, which revoke each quarter, accumulate $13 \times 14 = 182$ pirate cards on average before revocation occurs, while schemes B and D have $250/14 \times 7 = 125$ days on average between

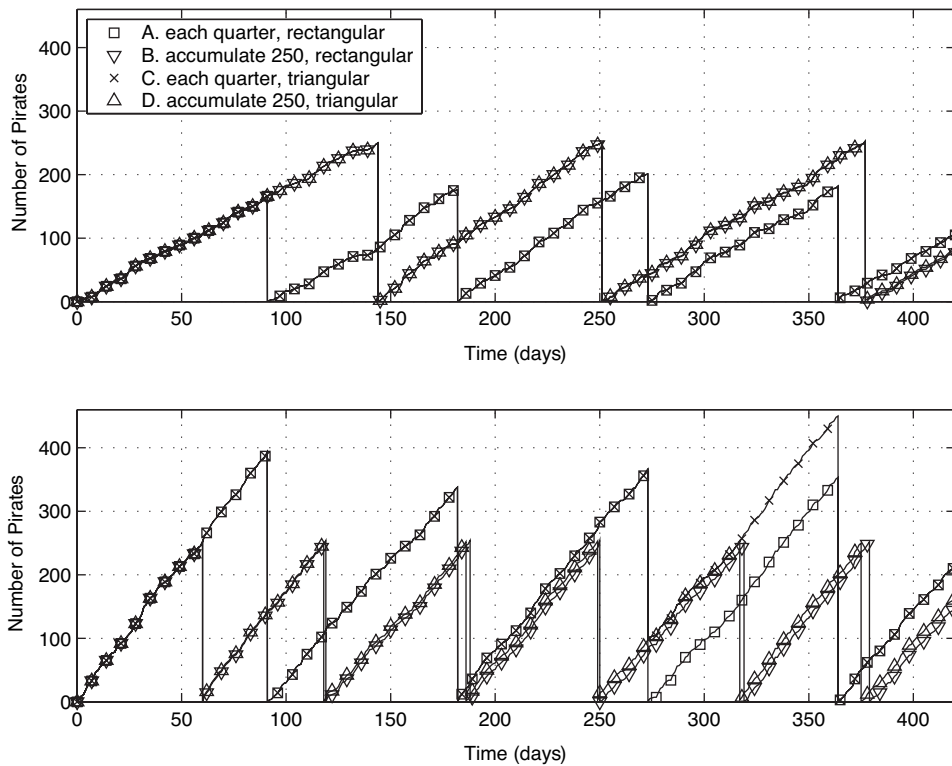


Fig. 1. The number of pirates in the system as a function of time: (top) for $\lambda = 14$, (bottom) for $\lambda = 28$.

revocations. Thus, for a relatively low discovery rate, the time-based trigger (every 3 months) occurs more frequently—but causes partial revocations (of less than 250 per round).

In Figure 1 (bottom), the discovery rate is $\lambda = 28$. The figure shows that for such a discovery rate, the *Triangular* construction with a revocation trigger each quarter (scheme C) maintains the number of pirates after revocation below $d/2$ and zeroes it in case either the rounding operation (Section 4.4) is up or $r_i < d$. By comparison, scheme A zeroes the number of pirate cards after each revocation, since all its secret-sharing schemes are of degree $t = 10,000$. Schemes B and D have $250/28 \times 7 = 62.5$ days, on average, between revocations.

5.2.2 The Revocation Message Length. Figure 2 shows the length of the revocation messages in each revocation round. The same figure also shows the system’s lifetime: i.e., the time by which either all $s = 40$ revocation rounds are done, or $t = 10,000$ pirate cards have been revoked, whichever arrives sooner.

The figure shows how the transmission length of the *Triangular* constructions grows linearly with the round number. The *Rectangular* constructions have a fixed transmission length of $w = t = 10,000$. (We omitted the constant factor of 2 incurred by precomputing the c_i coefficients.)

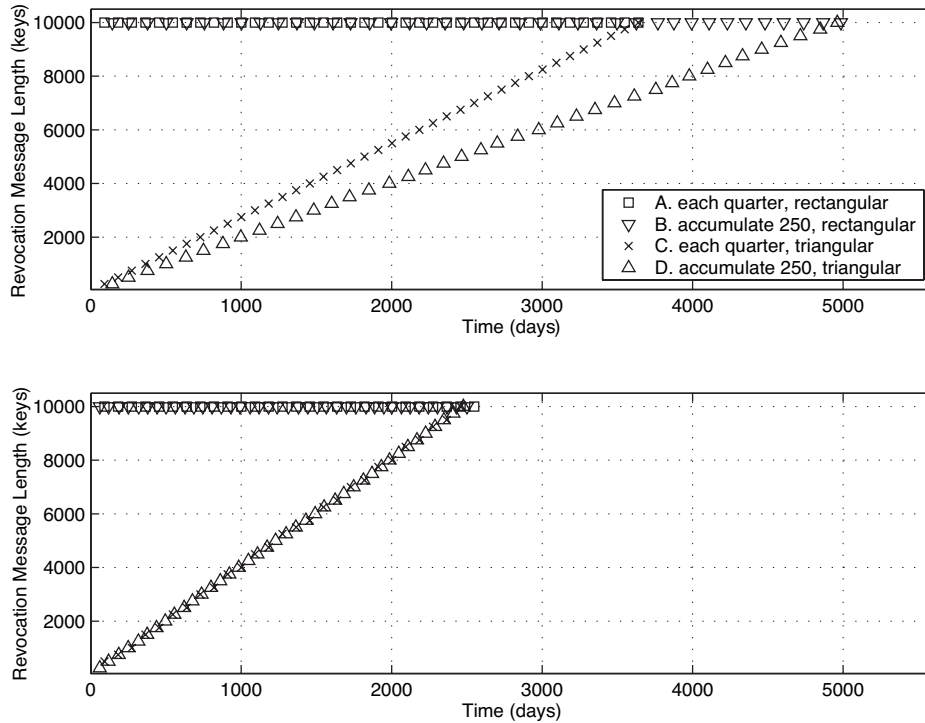


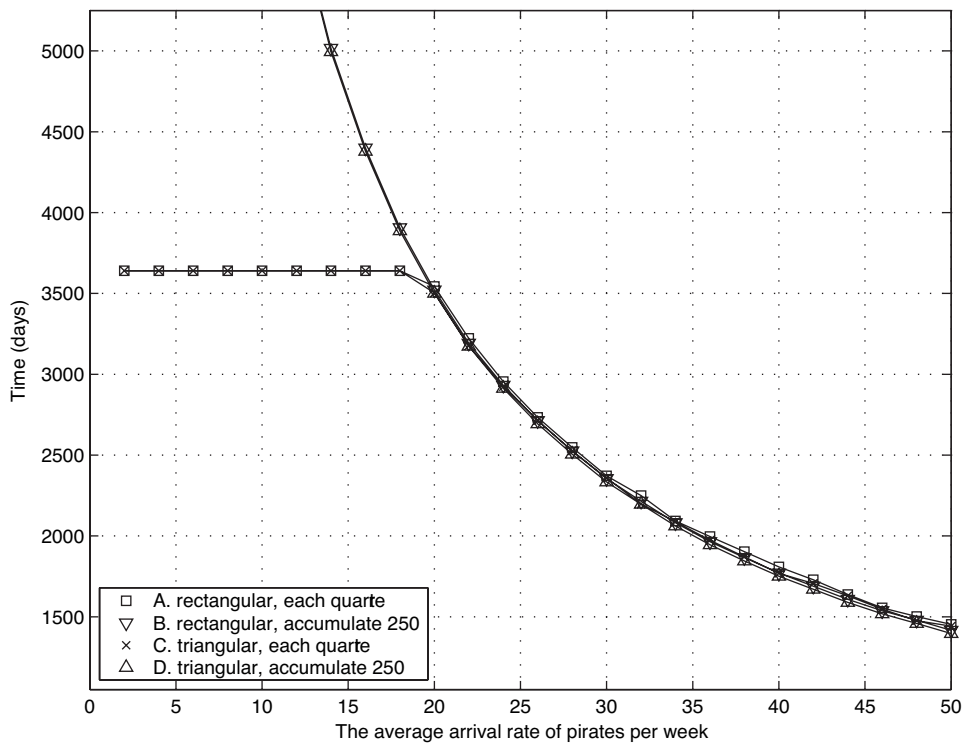
Fig. 2. Revocation message length (in keys) as a function of time, for (top) $\lambda = 14$, (bottom) $\lambda = 28$.

At a low discovery rate of $\lambda = 14$ (Figure 2, top) the schemes that revoke only when 250 pirate cards are discovered (B and D) complete their rounds much later than the 10 years the systems were designed for: the figures show these schemes to last about 13.7 years. For a discovery rate of $\lambda = 28$ (Figure 2, bottom), the system lifetime for all schemes is shortened to an average of 6.9 years, since 10,000 pirate cards would already be revoked by then.

5.2.3 Varying the Pirate Discovery Rate λ . To check the scheme's sensitivity to the pirate discovery rate, we increased λ from 2 to 50 and plotted the system lifetime, average number of known pirates, and average pirate's lifetime. Each curve in Figures 3, 4–5, shows the average of 32 simulation runs. For each curve, we also computed the 95% confidence interval values (cf. [Jain 1991] for the definition). The confidence intervals were roughly as small as the symbols on the curves, so we omit them for clarity.

Figure 3 depicts the average system lifetime as a function of the discovery rate for each of the four schemes. Schemes B and D, performing revocation upon accumulating 250 pirate cards, have a very high lifetime for a low discovery rate, while schemes A and C, performing revocation upon a time trigger, have a constant lifetime of 10 years for a low discovery rate. For high discovery rates, all the schemes have a lifetime that behaves approximately as $10,000/\lambda$.

5.2.4 Levels of Piracy. Figure 4 shows the average number of known pirate cards over the system lifetime as a function of the discovery rate for each of the

Fig. 3. System lifetime as a function of λ .

four schemes. Schemes B and D, performing revocation upon accumulating 250 pirate cards, have a constant average piracy level of 125 pirates, which is one-half of the accumulation trigger, independent of the discovery rate. Schemes A and C, performing revocation upon a time trigger, have a linear average piracy level of $\approx 13/2\lambda$ for a low discovery rate. For high discovery rates, scheme A retains the same linear average piracy level, since the *Rectangular* construction can revoke all existing pirate cards. Scheme C however, has an additional offset of size $d/8$, since the *Triangular* construction does not revoke all the existing pirate cards when the rounding operation (Section 4.4) is down. The number of pirate cards left after revocation can be approximated by a random variable uniformly distributed in $[0 \dots d/2]$;⁴ thus its average is $d/4$. The rounding down operation is done with probability $1/2$, giving us an average offset of $d/8$.

Another interesting parameter is the length of time between the discovery of a new pirate card and its revocation, which we call the “pirate lifetime.” Figure 5 shows the average lifetime of a pirate card, as a function of the discovery rate. Schemes B and D, performing revocation upon accumulating 250 pirate cards, suffer a high pirate lifetime for low discovery rates (since revocations are well

⁴The offset at $\lambda = 38$ on the curve is smaller than $d/8$ due to integrality conditions. For $\lambda = 38$, the quarterly rate is $13 * 38 = 494$ new pirates, which is numerically close to $500 = 2 * 250$. This causes the approximation of a uniform distribution to be inaccurate when only 40 revocation rounds are simulated (in this case, the average number of leftover pirates stabilizes more slowly).

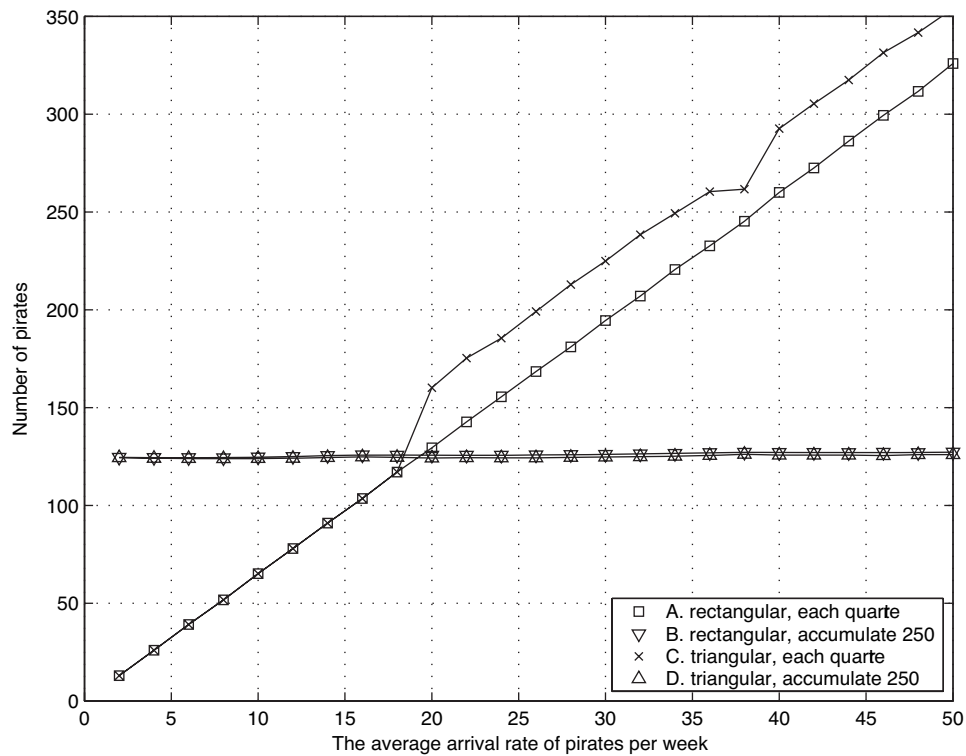


Fig. 4. The average number of pirates throughout the system lifetime, as a function of λ .

spaced in time). The pirate lifetime decreases like $250/2\lambda$ when the discovery rate grows under these schemes. Schemes A and C have a fixed pirate card average lifetime of $6.5 = 13/2$ weeks for low discovery rates. For high discovery rates, scheme A retains the same average, while, as we have already seen, scheme C has a higher average pirate lifetime because of all the rounds that have pirate “leftovers.”

5.2.5 Hybrid Triggers. Neither the time trigger or number-of-pirates trigger is universally superior. Revoking only upon a time trigger causes the accumulation of pirate cards at high discovery rates. Revoking only upon the accumulation of pirate cards has the disadvantage of a high pirate lifetime for low discovery rates.

However, it is straight-forward for the center to perform revocation upon both triggers, whichever arrives first. If revocation is performed upon the combination of both triggers, then the *Triangular* construction has the same system lifetime, average pirate card number, and average pirate lifetime as the *Rectangular* construction (the lower parts of Figures 3, 4–5). The aggregate transmission length (over all revocation rounds) of the *Triangular* schemes is $ts/2$, while for the *Rectangular* schemes it is ts . Thus, overall, the *Triangular* scheme is better by a factor 2. Note, though, that the savings are much higher during the early revocation rounds.

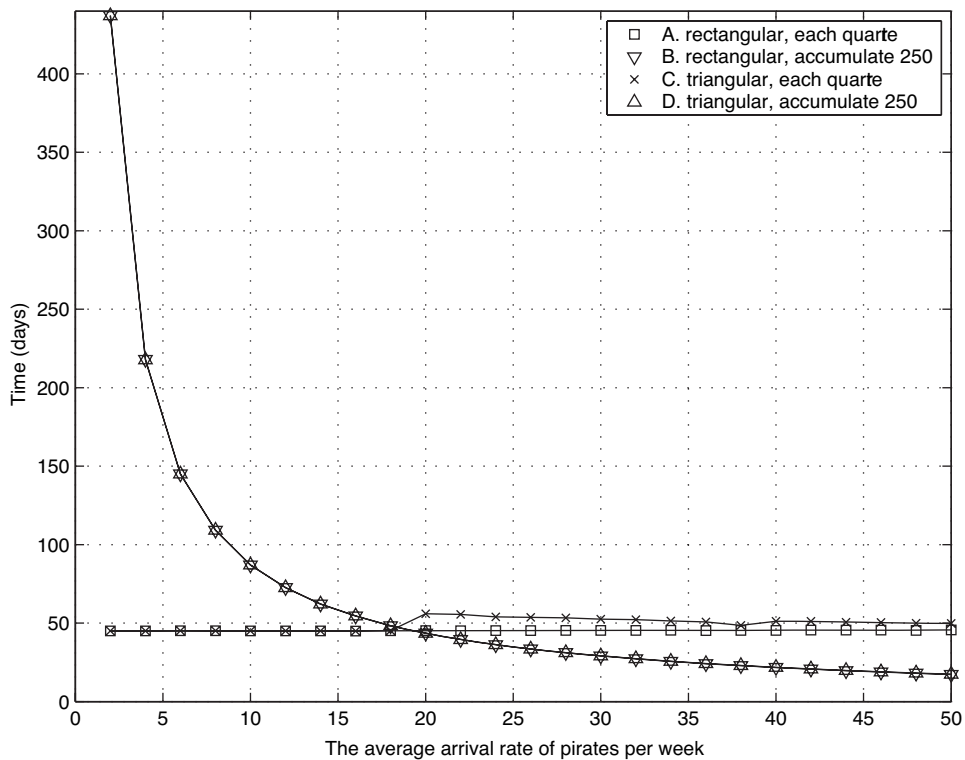


Fig. 5. The average lifetime of a Pirate (from discovery to revocation), as a function of λ .

We can conclude that of the variants we have examined, the best solution is to use the *Triangular* scheme, using a hybrid revocation-round trigger: A revocation round should be triggered by the earlier of a time period and exceeding a threshold of discovered pirate cards.

5.2.6 Sensitivity to Bursts of Pirates. Modeling the discovery of pirates as a random Poisson process can be somewhat limiting. In order to check the sensitivity of both schemes (*Rectangular* and *Triangular*) and both trigger types (time and pirate number) to fluctuations in the pirate discovery rate, we have generated a burst of pirates on the 245th day of the simulation, instead of the pirates that are generated by the Poisson process. The burst of pirates was set to be of size 1630 pirate users. Although these two values are arbitrary, the behavior of the system is essentially the same for other values.

Figure 6 shows the average system lifetime including the burst, as a function of the pirate discovery rate, for each of the four schemes. For the sake of comparison, the figure also shows the approximate lifetime of the schemes without the burst (Figure 3), shown in the dash-dot line. For low discovery rates, when the lifetime of the system is limited by the number of polynomials ($s = 40$), the *Triangular* scheme C has a shortened lifetime compared to both its *Rectangular* counterpart (scheme A), whose lifetime is unchanged, and to scheme C without the burst. This is due to “skipping ahead” polynomials to

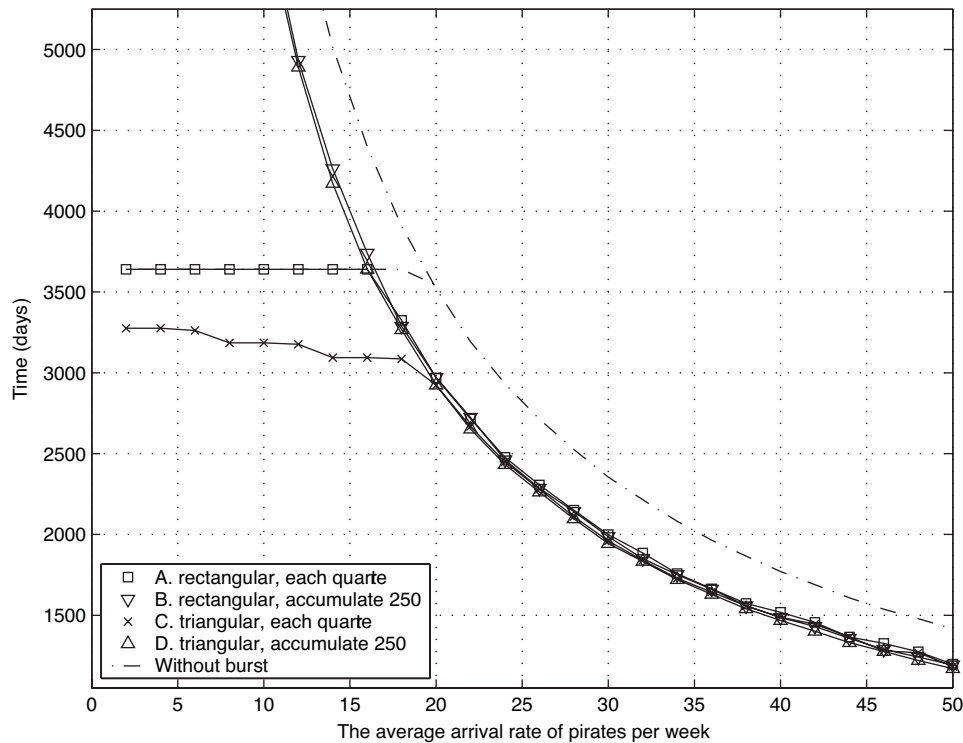


Fig. 6. System lifetime as a function of λ , including a burst of 1630 pirates on day 245.

handle the pirate burst. Schemes B and D have also a shortened lifetime compared to Figure 3, since more revocation capacity is spent on handling the burst. For high discovery rates, when the lifetime is limited by the total revocation capacity $t = 10,000$, all four schemes have a shortened lifetime compared to Figure 3, since more revocation capacity is spent on handling the burst.

Figure 7 shows the total number of pirate cards revoked during the lifetime of the system, as a function of the discovery rate, for each of the four schemes. Schemes B and D, performing revocation upon accumulating 250 pirate cards, revoke $\approx 10,000$ pirate cards during the lifetime of the system, independent of the discovery rate. For low discovery rates, schemes A and C, performing revocation upon a time trigger, revoke a linearly increasing number of pirates (with respect to λ) during the system's lifetime. Scheme A behaves like the Poisson arrival, to which the burst is added. Scheme C, however, manages to revoke less pirate cards, since the burst shortens the usable number of polynomials, which translates also to a shortened system lifetime. For high discovery rates, schemes A and C also revoke $\approx 10,000$ pirate cards during the lifetime of the system.

5.3 Queueing Theory Analysis of Pirate “Leftovers”

In this section, we study the behavior of the pirate “leftovers” using queueing theory. As explained in Section 4.4, pirate “leftovers” exist after revocation in

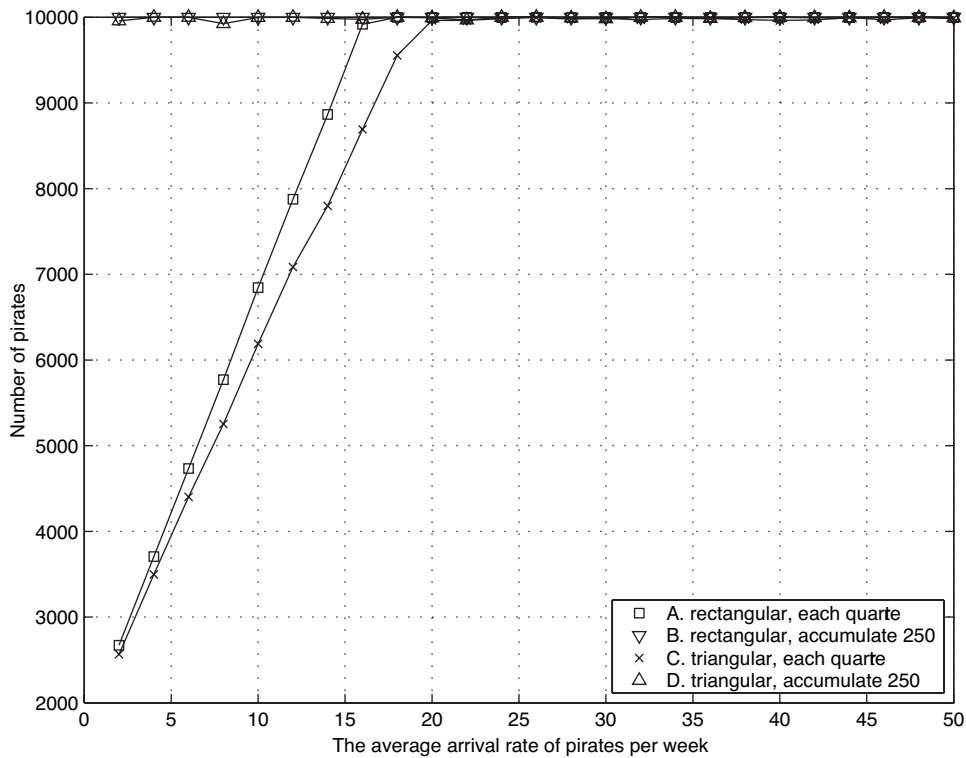


Fig. 7. The aggregate number of pirates that were revoked during the Lifetime of the system, as a function of λ , including a burst of 1630 pirates on day 245.

the *Triangular* constructions when the round operation is “down” and represent pirates that were not “served” (revoked). Note that the *Rectangular* constructions (A and B) do not suffer from pirate “leftovers,” so its queue analysis conforms to Little’s theorem [Jain 1991]: $N = \lambda t$, where t is the average lifetime of a pirate, N is the average number of pirates, and λ is their average arrival rate. Our analysis is performed on scheme C. We assume scheme D can check the revocation trigger in a continuous manner and thus refrain from having pirate “leftovers.”

We model the number of pirates in the system by a queue, whose size increases with the discovery of new pirates (modeled by the Poisson process), and decreases upon revocation (by an integer multiple of the dilation factor d). Next, we develop the equations governing the queue of scheme C, solve them numerically, and present the results.

We denote the state of the system *after* revocation round i by n_i : positive values of n_i represent pirate “leftovers,” while negative values represent revocation “credit.” In order to write compact equations, we use the additive property of a Poisson process (an average of λ per week equals an average of $13 \times \lambda$ per quarter). Recalling that the number of newly discovered pirates per quarter is r_i , and setting δ_i to be the difference in size between the previous scheme and

the current one, we get:

$$n_i = n_{i-1} + r_i - \delta_i = n_{i-1} + r_i - y \times d \quad ; \quad y = \max \left\{ 1, \left\lfloor \frac{n_{i-1} + r_i}{d} + 0.5 \right\rfloor \right\}$$

The numerical solution is based on steady-state analysis, i.e., each value of the queue, after revocation, is assumed to have a fixed probability and, for each state, the probability of entering it and leaving it are equal. This means that scheme C is analyzed as if it has an unlimited number of polynomials.

We identify two regions the system can be in. In the first, $\lambda < 250/13 \approx 19.2$, the system has a high chance of emptying the queue upon each revocation, in which case the pirates average number and lifetime can be trivially derived by using Little's theorem [Jain 1991], as before. In the second region, where $\lambda > 250/13 \approx 19.2$, the phenomenon of pirate "leftovers" becomes interesting. From here on, we focus on this region.

Intuitively the state of the queue after revocation varies between a "credit" of $-d/2$ to $d/2-1$ "leftovers." However, the accumulated "credit" may occasionally go below $-d/2$. This happens when the queue is already in "credit," for example, of $-d/2$, and the following round has less than d discovered pirates (which can happen even for $\lambda > 19.2$). Thus, in principle, the accumulated credit can be unbounded. In order to have a finite (and compact) state-machine, we bound the accumulated "credit" by $-d$, since the probability of going below it is negligibl.⁵

5.3.1 Numerical Solution. Define $q_{u,v}$ as the probability to pass from state n_u to state n_v , after r_i new pirates have been discovered and revocation round i has taken place. Then:

$$q_{u,v} = \sum_{r_i} \frac{\lambda^{r_i} e^{-\lambda}}{r_i!} ; \quad \left\{ r_i \mid r_i = v - u + d \max \left(1, \left\lfloor \frac{u + r_i}{d} + 0.5 \right\rfloor \right) \right\}$$

Let ψ_u be the steady-state probability of state u . Therefore, the probability of entering state v has to be equal to the probability of leaving state v :

$$\forall v : \sum_{u=-d}^{d/2-1} \psi_u q_{u,v} = \psi_v \sum_{s=-d}^{d/2-1} q_{v,s} \simeq \psi_v$$

This yields $3d/2$ equations with $3d/2$ variables. In vector notation, we make the following standard manipulation:

$$\begin{aligned} \vec{\psi}_{1 \times 3d/2} \times \vec{q}_{3d/2 \times 3d/2} &= \vec{\psi}_{1 \times 3d/2} \\ \vec{\psi}_{1 \times 3d/2} \times (\vec{q} - I)_{3d/2 \times 3d/2} &= \vec{0}_{1 \times 3d/2} \\ \vec{\psi}_{1 \times 3d/2} \times \vec{q}'_{3d/2 \times 3d/2} &= \vec{0}_{1 \times 3d/2} \end{aligned}$$

⁵Assume that the queue is in a "credit" state of $n_i = -d/2$, the discovery rate is $\lambda_{quarter} = d$, and the additional $d/2$ "credit" accumulates consecutively and linearly in z rounds. Then, $Prob(n_{i+z} < -d) \leq (\sum_{k=0}^{d/(2z)} \frac{d^k \cdot e^{-d}}{k!})^z$. The maximal value of this expression is obtained for $5 \leq z \leq 10$ and it is upper bounded by 10^{-6} .

where q' is defined as

$$q'_{i,j} = \begin{cases} q_{i,j} & \text{for } i \neq j \\ q_{i,i} - 1 & \text{for } i = j \end{cases}$$

The $3d/2$ equations are dependent and thus one can be omitted. In order to solve the system, we replace the omitted equation with the probability conservation rule, namely:

$$\sum_{u=-d}^{d/2-1} \psi_u = 1$$

which finally yields the following set of equations:

$$\vec{\psi}_{1 \times 3d/2} \times \vec{q}''_{3d/2 \times 2d/2} = \overrightarrow{(0 \ 0 \ \dots \ 0 \ 1)}_{1 \times 3d/2}$$

where

$$q''_{i,j} = \begin{cases} q'_{i,j} & \text{for } i \neq 3d/2 \\ 1 & \text{for } i = 3d/2 \end{cases}$$

We solved this set of equations numerically using Matlab. Formally, each $q_{u,v}$ is a sum of an infinite number of terms (r_i is formally unlimited). However, since the probability of a Poisson variable to be greater than a few times its average is negligibly small,⁶ a limit on r_i was used.

5.3.2 Results. Figure 8 shows the probability to be in each of the states of the queue after revocation for different λ values. The figure shows that when $\lambda > 22$ pirates per week, the steady-state size of the queue after revocation behaves almost as a uniform random variable between $[-d/2 \dots d/2 - 1]$ ⁷ as was assumed in Section 5.2.4. However, for $19.2 < \lambda \leq 22$ (the six values of λ that are visibly distinguishable in this resolution), we notice that the states between $[-d \dots -d/2 - 1]$ have a nonnegligible probability, and the probability of the states of the queue with high pirate “leftovers” is correspondingly lowered.

The pirate “leftovers” contribute to the average number of pirates in the system and to the lifetime of a pirate. Figure 9 shows the average number of pirate “leftovers” after revocation, for different λ values. For $\lambda > 22$ pirates per week, we see that this average reaches 31, which is exactly the expected value for a uniformly distributed random variable.⁸ For lower λ values, the average number of leftovers grows rapidly with λ , until the approximation holds.

6. CONCLUSIONS

Over the last few years, it has been generally accepted that tree-based broadcast encryption schemes, such as those of Wong et al. [2000], Wallner et al. [1998], Canetti et al. [1999a, 1999b], Naor et al. [2001] and Halevy and Shamir [2002], offer the best mix of features, and are superior to combinatorial or secret-sharing schemes for the same problem. We have shown that this superiority is

⁶For example $Prob(r_i > 3\lambda) < 2 \times 10^{-13}$; $19.2 < \lambda < 50$.

⁷Which creates the impression of a step function because of the y axis resolution.

⁸ $\sum_{i=1}^{d/2-1} i = \frac{d}{8} - \frac{1}{4} = 31$, when $d = 250$.

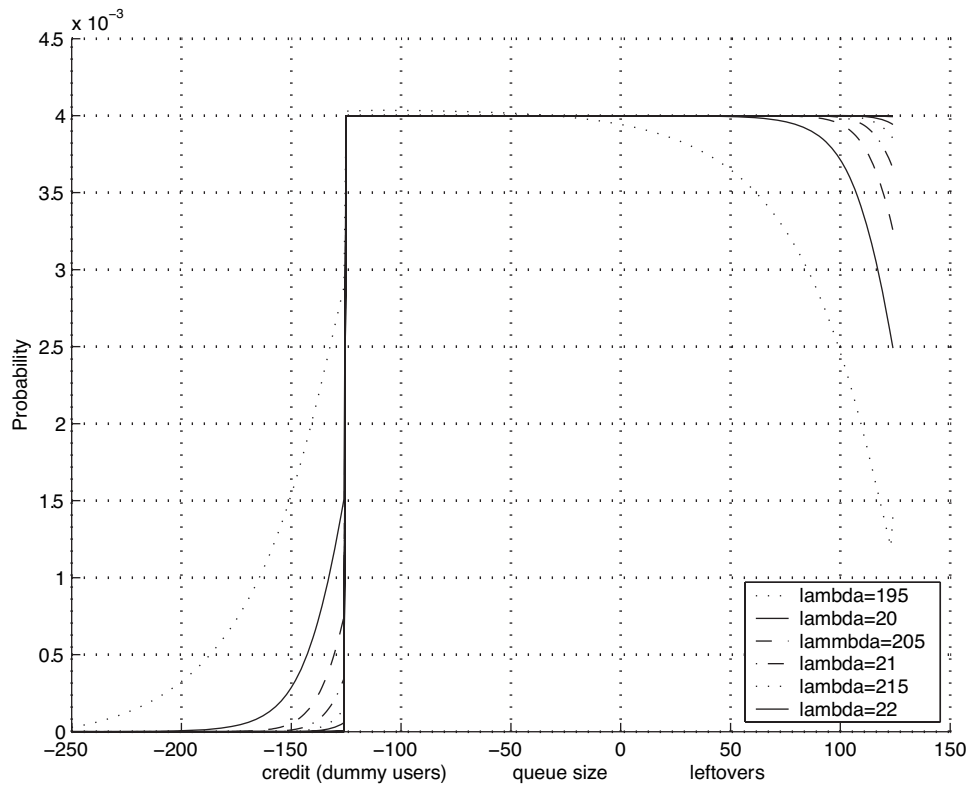


Fig. 8. Queue size probability in steady state, for different λ values.

less clear-cut than was previously assumed, by demonstrating a secret-sharing-based broadcast-encryption revocation scheme, that is competitive with the best known tree-based solutions in combating piracy. Specifically, our best scheme enjoys the following properties:

- **Computational load:** Our *Triangular* construction uses minimal resources on the SC: CPU, RAM, and STT-to-SC communication. The SC has to perform only a single-field multiplication and addition in a field. This is significantly better than the $O(\log^2 n)$ or $O(\log^{3/2} n)$ computation required for tree-based schemes. Although harder to quantify, we argue that, because the scheme's simplicity, the SC-resident code of our scheme should be much simpler to write and will require much less system ROM than that of tree-based schemes. In fact it is hard to imagine anything simpler.
- **Transmission length:** Our *Triangular* construction has linear transmission lengths that are on par with the best tree-based schemes ([Naor et al. 2001; Halevy and Shamir 2002]). These latter schemes offer much shorter transmission lengths under favorable conditions. However, when the application is pirate card revocation, we argue that favorable conditions (many pirates in a small number of subtrees) are highly unlikely.

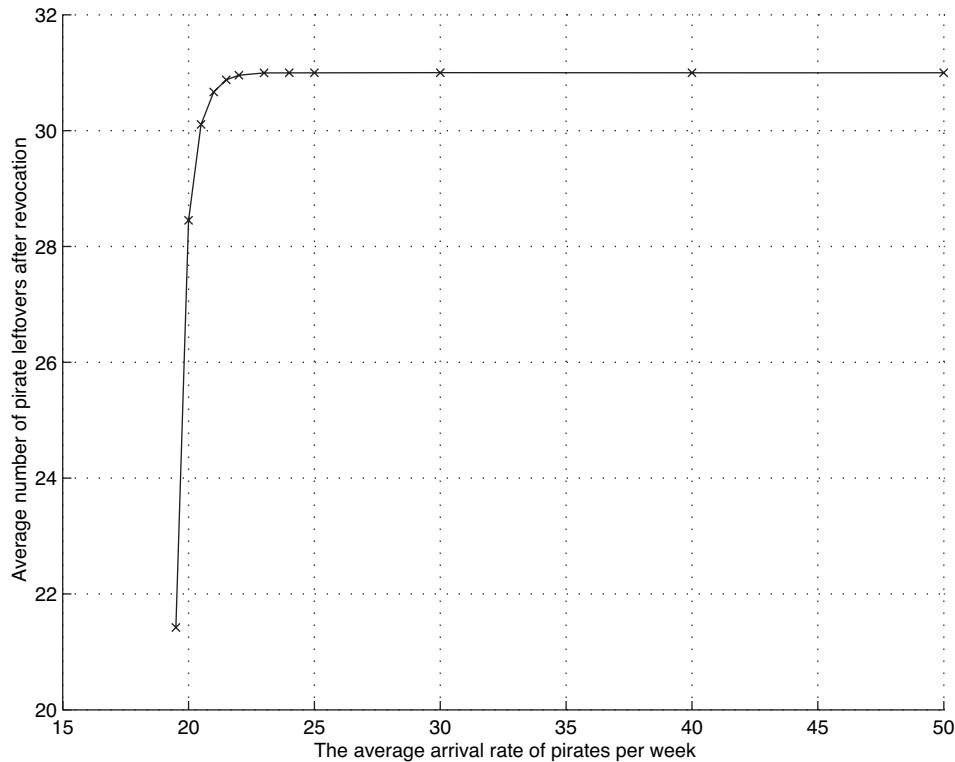


Fig. 9. The average number of pirate “leftovers” after revocation, as a function of λ .

- **Secure EEPROM usage:** In the *Triangular* construction the number of shares stored in EEPROM, s , is an unconstrained parameter, which can be chosen to balance the amount of EEPROM that is available, against the desired lifetime of the system and the granularity of revocation. In tree schemes, EEPROM usage is dictated by n and is not flexible: e.g., the $O(\log^2 n)$ keys needed by Naor et al. [2001] may exceed the capacity on weak SCs when $n = 10^8$ (although, in fairness, the $O(\log^{3/2} n)$ of Halevy and Shamir [2002] is probably small enough for realistic values of n).
- **Scheme lifetime:** While tree schemes are not limited in their lifetime before recarding is needed, the *Triangular* construction can be used with realistic parameter settings (t , s , and d) that allow several years of use before recarding becomes necessary, effectively obtaining the same behavior (the lifetime of the system itself, including the SC, is also limited).
- **Pirate cards revoked:** While tree schemes can revoke an unlimited number of pirates, the *Triangular* construction can have $t = R \approx 10000$, which seems to be a reasonably large number of *different* hacked SCs for a population of $n \approx 10^8$: $t \approx \sqrt{n}$. Furthermore, the bottlenecks for increasing t further are no longer in the SC: they are the center’s ability to compute $O(t^2)$ values and the STT’s need to compute and store $O(t)$ values.

- Late entry: In the *Triangular* construction, as in the stateless tree schemes, a rejoining or new user is not required to process past revocation messages.

Finally, we have highlighted the parameters of an operational revocation strategy, namely, revocation round trigger, system lifetime, pirate lifetime, known pirate number, and sensitivity to bursts of pirates. Our simulations identified a good operational strategy, under which the *Triangular* scheme can perform effective pirate revocation for realistic broadcast encryption scenarios.

REFERENCES

- ABDALLA, M., SHAVITT, Y., AND WOOL, A. 2000. Key management for restricted multicast using broadcast encryption. *IEEE/ACM Transactions on Networking* 8, 4, 443–454.
- ANDERSON, R. AND KUHN, M. 1996. Tamper resistance—a cautionary note. In *Proc. 2nd USENIX Workshop on Electronic Commerce*. USENIX, Oakland, CA. 1–11.
- ANDERSON, R. AND KUHN, M. 1997. Low cost attacks on tamper resistant devices. In *5th Security Protocols Workshop, LNCS 1361*. Paris, France. Springer-Verlag, New York. 125–136.
- BBC. 26 Jan. 2001. Toasting the crackers. BBC news on Science and Technology, reporter Mark Ward, Front Page. <http://news.bc.co.uk/hi/science/nature/1138550.stm>.
- BERKOVITS, S. 1991. How to broadcast a secret. In *Advances in Cryptology—EUROCRYPT ’91, LNCS 547*. Springer-Verlag, New York. 535–541.
- BLUNDO, C. AND CRESTI, A. 1994. Space requirements for broadcast encryption. In *Advances in Cryptology—EUROCRYPT ’94, LNCS 950*, A. D. Santis, Ed. Springer-Verlag, New York. 287–298.
- BLUNDO, C., FROTA MATTOS, L. A., AND STINSON, D. R. 1996. Trade-offs between communication and storage in unconditionally secure schemes for broadcast encryption and interactive key distribution. In *Advances in Cryptology—CRYPTO ’96, LNCS 1109*. Springer-Verlag, New York. 387–400.
- BRISCOE, B. 1999. MARKS: Zero side-effect multicast key management using arbitrarily revealed key sequences. In *Proc 1st International Workshop on Networked Group Communication (NGC ’99), LNCS 1736*, L. Rizzo and S. Fdida, Eds. Pisa, Italy. Springer-Verlag, New York.
- CANETTI, R., GARAY, J., ITKIS, G., MICCIANCIO, D., NAOR, M., AND PINKAS, B. 1999a. Multicast security: A taxonomy and efficient constructions. In *Proc. IEEE INFOCOM ’99*. 708–716.
- CANETTI, R., MALKIN, T., AND NISSIM, K. 1999b. Efficient communication-storage tradeoffs for multicast encryption. In *Advances in Cryptology—EUROCRYPT ’99, LNCS 1592*. Springer-Verlag, New York. 459–474.
- FIAT, A. AND NAOR, M. 1994. Broadcast encryption. In *Advances in Cryptology—CRYPTO ’93, LNCS 773*. Springer-Verlag, New York. 480–491.
- GARAY, J. A., STADDON, J., AND WOOL, A. 2000. Long-lived broadcast encryption. In *Advances in Cryptology—CRYPTO ’2000, LNCS 1880*, M. Bellare, Ed. Springer-Verlag, New York. 333–352.
- Hackwatch 2002. Canal plus claims Murdoch operation pirated canal plus cards. Hackwatch. <http://www.hackwatch.com/~kooltek/>.
- HALEVY, D. AND SHAMIR, A. 2002. The LSD broadcast encryption scheme. In *Advances in Cryptology—CRYPTO ’02, LNCS 2442*, M. Yung, Ed. Springer-Verlag, New York.
- JAIN, R. 1991. *The Art of Computer Systems Performance Analysis*. Wiley, New York.
- KOGAN, N., SHAVITT, Y., AND WOOL, A. 2003. A practical revocation scheme for broadcast encryption using smart cards. In *Proc. IEEE Symp. on Security and Privacy*. Oakland, CA. 225–235.
- KUMAR, R., RAJAGOPALAN, S., AND SAHAI, A. 1999. Coding constructions for blacklisting problems without computational assumptions. In *Advances in Cryptology—CRYPTO ’99, LNCS 1666*. Springer-Verlag, New York. 609–623.
- LUBY, M. AND STADDON, J. 1998. Combinatorial bounds for broadcast encryption. In *Advances in Cryptology—EUROCRYPT ’98, LNCS 1403*, K. Nyberg, Ed. Espoo, Finland. Springer-Verlag, New York. 512–526.
- NAOR, M. AND PINKAS, B. 2000. Efficient trace and revoke schemes. In *Financial Cryptography ’00, LNCS 1962*. Springer-Verlag, New York. 1–20.

- NAOR, D., NAOR, M., AND LOTSPIECH, J. B. 2001. Revocation and tracing schemes for stateless receivers. In *Advances in Cryptology—CRYPTO '2001, LNCS 2139*. Springer-Verlag, New York. 41–62.
- PINKAS, B. 2001. Efficient state updates for key management. In *Digital Rights Management Workshop '2001, LNCS 2320*. Springer-Verlag, New York. 40–56.
- SHAMIR, A. 1979. How to share a secret. *Communications of the ACM* 22, 11, 612–613.
- WALLNER, D. M., HARDER, E. J., AND AGEE, R. C. 1998. Key management for multicast: Issues and architectures. Internet Draft. Available from <http://www.ietf.org/ID.html>.
- WONG, C. K., GOUDA, M., AND LAM, S. S. 2000. Secure group communications using key graphs. *IEEE/ACM Transactions on Networking* 8, 1 (Feb.), 16–30.

Received July 2003; revised May 2005 and March 2006; accepted April 2006