A Practical Revocation Scheme for Broadcast Encryption Using Smart Cards

Extended Abstract

Noam Kogan*† Yuval Shavitt*‡ Avishai Wool*§

Abstract

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 recent scheme of Naor and Pinkas. The 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 in novel ways among the smartcard, set-top terminal, and center. Based on this, we construct several improved schemes for many stateful revocation rounds that scale to realistic system sizes. We allow up to about 10000 pirates to be revoked using current smartcard technology before re-carding is needed. The transmission lengths of our constructions are on par with those of the best key-based schemes. However, our constructions have much lower smartcard CPU complexity: only $O(1)$ smartcard operations per revocation round, as opposed to a poly-logarithmic complexity of the best tree-based schemes.

We evaluate the system behavior via an exhaustive simulation study. Our simulations show that with mild assumptions on the piracy discovery rate, our constructions can perform effective pirate revocation for realistic broadcast encryption scenarios.

*Dept. of Electrical Engineering Systems, Tel Aviv University, Ramat Aviv 69778, Israel. This work was supported, in part, by a grant from Giesicke and Devrient, GmbH.

†E-mail: noamkogan@bezeqint.net.

‡E-mail: shavitt@eng.tau.ac.il.

§E-mail: yash@acm.org.

1 Introduction

1.1 Channel Model Definition

Many digital content and multi-media 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. Decryption is done in a set-top terminal (STT) coupled with a tamper resistant smartcard (SC). The SC stores the key material, and performs the decryption of 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[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

Since the number of users is very high, the price of the SC is not a negligible parameter. Therefore, very cheap SC's are often used. Such platforms typically have the following limitations:

- A very restricted amount of secure memory (EEPROM)—typically 1-8Kbytes. 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: 9.6-38.4Kbits/sec.

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

[1] The center can distribute keys only via the broadcast channel. If an uplink exists, user revocation is much easier.

24th IEEE Symposium on Security & Privacy, Oakland, CA, May 2003.

In the systems that we consider, there is a single key that is used by the center to encrypt the current content, and is stored in all the users' SCs. When the center needs to change this key, it broadcasts a message. Based on the content of this transmission and the secret data already on the SCs, all the legitimate 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 on the cheap SCs in use.

1.3 User Revocation

Piracy in digital content and multi-media 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 pirate ones. The scheme should allow for many revocation rounds. The scheme should also be stateful—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 are 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 the $n - R_i$ non-revoked 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.

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.

Much of the notation we use is summarized in Table 1 for quick reference.

1.4 Related Work

The study of broadcast encryption was initiated in [5, 10] and the efficiency of broadcast encryption schemes is studied in [6, 7, 16]. Broadcast encryption schemes usually fall into one of three categories: combinatorial constructions, secret sharing constructions, and tree-based constructions.

In the combinatorial approach [16, 1, 15, 11], the keys of compromised SCs are no longer used. Luby and Staddon [16] 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, Shavitt and Wool [1] allowed a factor f of free riders to enjoy the broadcast content, breaking away from the [16] bounds. The [15] 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, Staddon and Wool [11] introduced the notion of long-lived broadcast encryption, analyzing the number of recardings needed per epoch.

Naor and Pinkas [18] 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 $q = O(t^2)$ calculations at the user, or $w = 2t$ messages and only $q = O(t)$ calculations at the user. For multi-round revocation they suggested using t revocation schemes with sizes $1, 2, ..., t$, to be used one after the other. This scheme can revoke a total of up to t users before re-carding of all SCs is needed. We elaborate on this scheme later as it is our starting point.

Tree based revocation schemes were proposed in [22, 21, 8, 9, 17, 13, 19], with stateless and stateful variants, and are currently considered to have the best combination of properties. In all of them, the SC stores a poly-logarithmic number of keys. Some of the stateful variants [21, 8, 19] "re-build" the key tree after revocation takes place, while others [17, 13] "carry" the "holes" in the tree (previously revoked users) to future revocation rounds. Their transmission length is linear: $w = O(R)$ [17, 13]. In some of them [21, 8], a user which rejoins the group after being offline for a while, is required to receive and process at least some of the revocation messages that were sent in its absence, in order to update the group key.

1.5 Contributions

Our starting point is the multi-round secret-sharing-based revocation scheme of [18]. This scheme can perform many stateful revocation rounds, revoking a total of up to t pirates. However, with current technology, this scheme is

n	the total number of users (i.e., legitimate keys) in the system.
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.
D_i	the aggregate number of discovered pirate users (keys) up to and including revocation round i.
t	the maximal number of users that can be revoked before recarding is needed.
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).
s	the number of keys stored in the SC.
q	the computational load at the user (decryptions, key derivations, etc).

Table 1. The notation used in this paper.

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 $t \approx 100$. Thus, *a-priori* this scheme seems inferior to the tree-based schemes of [22, 21, 8, 9, 17, 13].

We first show how to re-arrange the computations of the scheme, so the work is distributed better among the SC and the STT. Specifically, the $q = O(t)$ calculations at the user can be split into $O(1)$ *secret* calculations that need to be done on the SC, and $O(t)$ *non-secret* 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. Furthermore, note that tree-based schemes require $\Omega(\log^{\alpha} n)$ secret calculations on the SC (the exponent $1 \leq \alpha \leq 2$ depends on the particular scheme).

Since the scheme requires re-carding 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 us from having $s = t$ schemes as suggested by [18], 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 multi-round construction is the *Triangular* system. It has s schemes of sizes $d, 2d, \ldots t$, with $d = t/s$. In the *Triangular* system, the center is capable of revoking an average of $d = t/s$ 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. Broadcasting such dummy shares in polynomial P_1 does not give pirates any information about shares in some other polynomial P_2. 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 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 *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. 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.

Organization: In the next 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 is described in Section 5, and we conclude with Section 6. In an appendix we show how the center's computation can be made incrementally, and we also include the salient details of the [18] schemes for completeness.

2 Attack Model

Piracy in digital content and multi-media 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 due to piracy (cf. [4, 12]).

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. [2, 3]).

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.

We assume a reactive enforcement system to combat piracy. We assume that on an ongoing basis, law enforcement agencies capture cloned SCs. These cards are analyzed by the content provider, which identifies the original pirate card from which the clone was made. Once a new pirate card is identified, its key can be scheduled for revocation in the next revocation round.

3 Avoiding the SC Communication, RAM and CPU Bottlenecks

In this section, we show how to implement the basic [18] 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 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 operation and one addition.

3.1 The Basic [18] Secret-Sharing Revocation Scheme

Before presenting our modifications, we briefly describe the [18] revocation scheme for a single revocation round. More detail can be found in Appendix B. The scheme uses Shamir's polynomial secret sharing [20], over a field F whose elements are encryption keys and user identities: typically, 64–128 bits suffice for the field elements. The [18] scheme utilizes the Lagrange interpolation formula. Given the identities u_i of the revoked users, and their shares $P(u_i)$ in the degree d polynomial $P()$, the formula is

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

where u_0 is the identity of the user performing the calculation, and u_i is the identity of the i'th revoked user ($i = 1, \ldots, d$). $P(0)$ is the new key that is set at the end of the revocation round. The construction of the secret sharing ensures that the coalition of d revoked users will not have enough shares to compute $P(0)$, whereas every non-revoked user will.

Denote the inter-product function on the identities by c_i. Then the Lagrange interpolation formula can be re-written as:

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

In order to solve for $P(0)$, a user needs $d+1$ shares $P(u_i)$, one of which is his own, and $d+1$ c_i coefficients. Calculating the $d+1$ c_i coefficients requires $O(d^2)$ calculations. However, almost all of this calculation is identical for all the users, and can be pre-computed by the center. Only $O(d)$ calculations differ between users: these are the calculations involving the identity of each non-revoked user u_0. Specifically, the center computes t coefficients, denoted by c'_i, as follows:

$$c'_i = \prod_{j \neq i} \frac{u_j}{u_j - u_i} \text{ for } i = 1, \ldots, d.$$

The center then broadcasts the d identities u_i of the revoked users, their d shares $P(u_i)$, and the d coefficients c'_i.

Each non-revoked user u_0 that receives the broadcast completes the calculation of the c_i coefficients with $O(d)$ calculations:

$$c_i = c'_i \cdot \frac{u_0}{u_0 - u_i} \text{ for } i = 1, \ldots, d\,; \qquad c_0 = \prod_{j=1}^{d} \frac{u_j}{u_j - u_0}$$

Then each user uses his own share $P(u_0)$, together with the d 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 $O(d)$ calculations on the SC has several drawbacks. First, the STT needs to transmit $O(d)$ data (with some 12-16 bytes per revoked user for the identity and share) 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 $O(d)$ field operations on 80-bit numbers. With current technology, each of these bottlenecks individually constrains the maximal value of d, which is also the maximal number of pirate cards that can be revoked, t, to ≈ 100.

Luckily, almost all the $O(d)$ calculations at the user are *non-secret*. In fact, all the information that is received in the broadcast is not secret: this includes the d shares $P(u_i)$ of all the revoked users, their d identities u_i, and the d coefficients c'_i (which were computed from the non-secret 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 STT. Each STT should know the identity u_0 of it's SC. Thus, the STT can also sum all but one of the terms that comprise $P(0)$:

$$P(0)' = \sum_{i=1}^{d} c_i \cdot P(u_i).$$

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 $P(u_0)$:

$$P(0) = P(0)' + c_0 \cdot P(u_0).$$

Therefore, both the computation, RAM memory, and communication with the SC are $O(1)$.

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 schemes.

4 The New Multi-Round Revocation Schemes

In this section, we present our two multi-round constructions. For comparison, we first describe some details of the multi-round [18]. We refer the reader to Appendix B for more detail.

4.1 Avoiding the Maintenance Channel

The [18] multi-round scheme uses t secret sharing schemes $RS_1, RS_2, ..., RS_t$ of sizes (degrees) $1, 2, ..., t$ respectively. In round i, revocation scheme RS_{R_i} is used for revoking the aggregated set of R_i users. At the end of each revocation round, the authors propose to use an "offline" maintenance channel. The idea is to broadcast the shares of the revoked users in *all* the future polynomials. Users are supposed to store these shares for future revocation rounds. The idea is to keep future broadcast transmissions short: their length would be $O(r_i)$, the number of new pirate cards in round i, as opposed to $O(R_i)$, the aggregate number of revoked pirate cards in rounds $1, \ldots, i$. 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 $O(st)$ non-volatile 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 retransmitted repeatedly, it would raise the real length of the transmission to $O(t^2)$ for each repetition.

Finally, with this approach the center has a diminished ability to reduce the computational load at the user by pre-computing most of the Lagrange interpolation formula. Note that broadcasting the c'_i's would be $w = O(R)$. Therefore, the users would need to either perform $O(R^2)$ calculations, or store $O(R)$ values in non-volatile memory and perform $O(Rr)$ 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. In this way, each user does not require non-volatile memory, the computational load at the user can return to be $q = O(R)$ (of which only $O(1)$ is done on the SC), and a rejoining user is not required to process all the past revocation messages.

4.2 Dealing with a Partial Revocation

The basic [18] scheme is good for a single revocation. Furthermore, [18] explains that it is not safe to use multiple schemes with the same degree, one per revocation round, since pirates revoked in different rounds can collude: A coalition of pirates that compromised d SCs has access to d shares in *each* of the polynomials. Instead, the authors use t polynomials P_i of degrees $1, 2, \ldots, t$.

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. Note that if a pirate card was revoked in round i it needs to be revoked again in every round $j > i$: this is required in stateless tree-based systems [17, 13] as well.

4.3 The Triangular Construction

Instead of using $s = t$ polynomials of degrees $1, 2, ..., t$, we introduce a dilation factor d, and build s schemes $RS_1, \ldots, RS_s$ of degrees $d, 2d, ...t$, with $d = t/s$. Scheme RS_i is capable of revoking di 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 of schemes

s, we can use large $t \approx 10000$ 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 $r_i \leq di - R_{i-1}$. If not all of the new r_i pirates can be revoked in this round, then some will carry over and be revoked in future revocation rounds. To prevent carry accumulation, it is possible to *skip ahead* one or more revocation schemes. For the i'th round we choose RS_j, s.t.,

$$RS_j = RS_{i-1} + d + d \cdot round(\frac{D_i - RS_{i-1} - d}{d})$$

where D_i is the aggregate number of pirate users up to and including the i'th round. Note that in each round, either the pirates in carry or the dummy users used are zero. It is possible to choose a different threshold for *skipping ahead* schemes.

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

4.4 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.

In the i'th revocation round, the center revokes all the known pirate users, namely $R_i = D_i$, 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.

5 Simulation

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 re-carding at all), average piracy levels and a pirate's lifetime.

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 every week.

We set the total number of pirate cards that the schemes need to be able to revoke at $R = 10000$. 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 $t = 10000$. 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 $t = 10000$, and the *Triangular* construction which has $s = 40$ schemes with linearly increasing secret sharing sizes $250, 500, 750, ..., 9750, 10000$.

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., 4 revocation rounds per year, one round each quarter). The two constructions and two trigger mechanisms gave 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 $\lambda = 250/13 = 19.2$ 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 average value λ of 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.

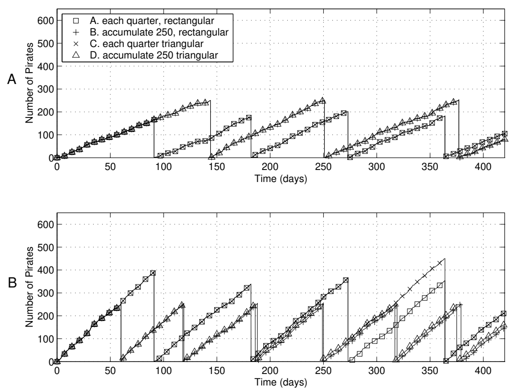

Figure 1. The number of pirates in the system as a function of time: (A) for $\lambda = 14$, (B) for $\lambda = 28$

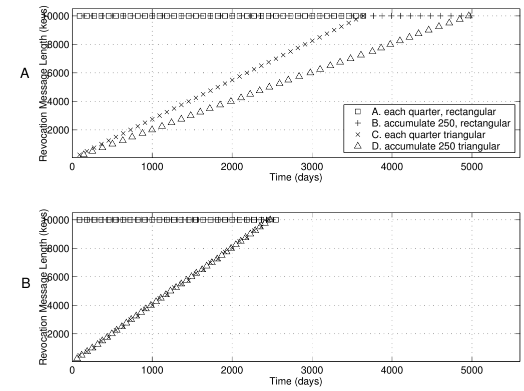

Figure 2. Revocation message length (in keys) as a function of time, for (A) $\lambda = 14$, (B) $\lambda = 28$.

5.2 Results

5.2.1 The Number of Known Pirate Cards in the System

Figure 1 depicts 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-A, schemes A and C, which revoke each quarter, accumulate $13 \cdot 14 = 182$ pirate cards on average before revocation occurs, while schemes B and D have $250/14 \cdot 7 = 125$ days on average between 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-B, 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, namely $D_i - R_i$, below $d/2$, and zeroes it in case either the round operation (section 4.3) was 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 $t = 10000$. Schemes B and D have $250/28 \cdot 7 = 62.5$ days on average between revocations.

5.2.2 The Revocation Message Length

Figure 2 depicts 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 $t = 10000$ pirate cards have been revoked — whichever arrives sooner.

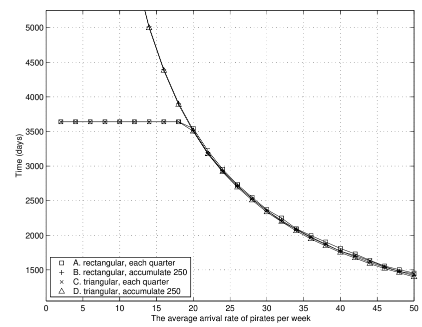

Figure 3. System lifetime as a function of λ.

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 $w = t = 10000$ (we omitted the constant factor incurred by pre-computing the c_i coefficients).

Schemes that revoke more frequently and use more partial revocation rounds cause their lifetime to be reduced, because they "waste" some of their s rounds on fewer than 250 pirates.

At a low discovery rate of $\lambda = 14$ (Figure 2-A) the schemes that revoke only when 250 pirate cards are discovered (B and D) complete their s 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-B), the system lifetime for all schemes is shortened to an average of $10000/364 \cdot 13/52 = 6.9$ years, since 10000 pirate cards would already be revoked by then.

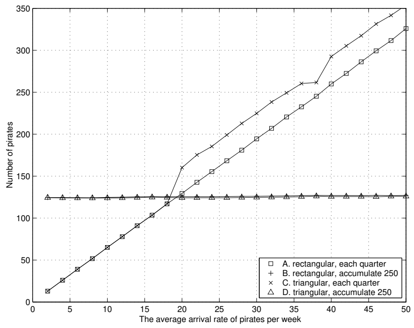

Figure 4. The average number of pirates throughout the system lifetime, as a function of λ.

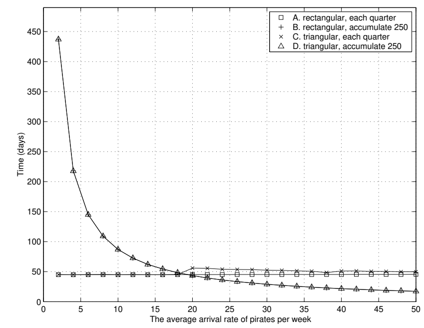

Figure 5. The average lifetime of a Pirate (from discovery to revocation), as a function of λ.

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, and 5, shows the average of 32 simulation runs. For each curve we also computed the 95% confidence interval values (cf. [14] 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 4 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 $10000/\lambda$.

5.2.4 Levels of Piracy

Figure 4 depicts the average number of known pirate cards over the system lifetime as a function of the discovery rate, for each of the 4 schemes. Schemes B and D, performing revocation upon accumulating 250 pirate cards, have a constant average piracy level of 125 pirates, which is 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 $\approx 13/2 \cdot \lambda$ 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, while scheme C has an additional offset of size $d/8$, since the *Triangular* construction does not revoke all the existing pirate cards when one or more schemes are *skipped ahead* and the round operation (section 4.3) is down. The number of pirate cards left after revocation, $D_i - R_i$, can be approximated by a random variable uniformly distributed in $[0..d/2]^2$, 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 depicts 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 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 which had pirates in carry.

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 *Triangular* scheme to accumulate 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

[2] The point for $\lambda = 38$, with a quarterly rate of $13 * 38 = 494$ new pirates, is closer to scheme A since the dummy users / carry accumulation rate is relatively slow compared to the 40 revocation rounds (the system lifetime) upon which they are averaged, refuting the assumed uniform distribution.

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.

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.

6 Conclusions

Over the last few years, it has been generally accepted that tree-based broadcast encryption schemes, such as those of [22, 21, 8, 9, 17, 13], 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 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 are all $O(1)$, requiring only a single field multiplication operation and single addition. This is significantly better than the $O(\log^{\alpha} n)$ computation required for tree-based schemes. Although harder to quantify, we argue that, due to 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 ([17, 13]). 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.

- **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 [17] may exceed the capacity on weak SCs when $n = 10^8$ (although, in fairness, the $O(\log^{3/2} n)$ of [13] is probably small enough for realistic values of n).

- **Scheme lifetime** While tree schemes are not limited in their lifetime before re-carding is needed, the *Triangular* construction can be used with realistic parameter settings (t, s and d) that allow several years of use before re-carding 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 and known pirate number. Our simulations identified a good operational strategy, under which the *Triangular* scheme can perform effective pirate revocation for realistic broadcast encryption scenarios.

References

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

[2] R. Anderson and M. Kuhn. Tamper resistance - a cautionary note. In *Proc. 2nd USENIX Workshop on Electronic Commerce*, pages 1–11, Oakland, California, Nov. 1996. Springer-Verlag.

[3] R. Anderson and M. Kuhn. Low cost attacks on tamper resistant devices. In *5th Security Protocols Workshop, LNCS 1361*, pages 125–136, Paris, France, Apr. 1997. Springer-Verlag.

[4] Toasting the crackers. BBC news on Science and Technology, reporter Mark Ward, Front Page, Friday 26 January 2001. `http://www.bbc.co.uk/hi/english/sci/tech/newsid_1138000/1138550.stm`.

[5] S. Berkovits. How to broadcast a secret. In *Advances in Cryptology – EUROCRYPT'91, LNCS 547*, pages 535–541. Springer-Verlag, 1991.

[6] C. Blundo and A. Cresti. Space requirements for broadcast encryption. In A. D. Santis, editor, *Advances in Cryptology – EUROCRYPT'94, LNCS 950*, pages 287–298. Springer-Verlag, 1994.

[7] C. Blundo, L. A. Frota Mattos, and D. R. Stinson. Tradeoffs between communication and storage in unconditionally secure schemes for broadcast encryption and interactive key distribution. In *Advances in Cryptology – CRYPTO'96, LNCS 1109*, pages 387–400. Springer-Verlag, 1996.

[8] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas. Multicast security: A taxonomy and efficient constructions. In *IEEE INFOCOM'99*, Mar. 1999.
[9] R. Canetti, T. Malkin, and K. Nissim. Efficient communication-storage tradeoffs for multicast encryption. In *Advances in Cryptology – EUROCRYPT'99, LNCS 1592*, pages 459–474. Springer-Verlag, 1999.
[10] A. Fiat and M. Naor. Broadcast encryption. In *Advances in Cryptology – CRYPTO'93, LNCS 773*, pages 480–491. Springer-Verlag, 1994.
[11] J. A. Garay, J. Staddon, and A. Wool. Long-lived broadcast encryption. In M. Bellare, editor, *Advances in Cryptology – CRYPTO'2000, LNCS 1880*, pages 333–352. Springer-Verlag, 2000.
[12] Canal plus claims murdoch operation pirated canal plus cards. Hackwatch, March 13 2002. `http://www.hackwatch.com/~kooltek/`.
[13] D. Halevy and A. Shamir. The LSD broadcast encryption scheme. In *Advances in Cryptology – CRYPTO'02*, 2002.
[14] R. Jain. *The Art of Computer Systems Performance Analysis*. John Wiley & Sons, 1991.
[15] R. Kumar, S. Rajagopalan, and A. Sahai. Coding constructions for blacklisting problems without computational assumptions. In *Advances in Cryptology – CRYPTO'99, LNCS 1666*, pages 609–623, 1999.
[16] M. Luby and J. Staddon. Combinatorial bounds for broadcast encryption. In K. Nyberg, editor, *Advances in Cryptology – EUROCRYPT'98, LNCS 1403*, pages 512–526, Espoo, Finland, 1998. Springer-Verlag.
[17] D. Naor, M. Naor, and J. B. Lotspiech. Revocation and tracing schemes for stateless receivers. In *Advances in Cryptology – CRYPTO'2001, LNCS 2139*, pages 41–62, 2001.
[18] M. Naor and B. Pinkas. Efficient trace and revoke schemes. In *Financial Cryptography'00, LNCS 1962*, pages 1–20. Springer-Verlag, 2000.
[19] B. Pinkas. Efficient state updates for key management. In *Digital Rights Management Workshop'2001, LNCS 2320*, pages 40–56, 2001.
[20] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.
[21] D. M. Wallner, E. J. Harder, and R. C. Agee. Key management for multicast: Issues and architectures. Internet Draft, Sept. 1998. Available from `http://www.ietf.org/ID.html`.
[22] C. K. Wong, M. Gouda, and S. S. Lam. Secure group communications using key graphs. In *Proc. ACM SIGCOMM*, pages 68–79, Vancouver, BC, 1998.

Appendix

A Incremental Computation at The Center

We observe that for a large t, the $q = O(t^2)$ 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 $q = O(rt)$ 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 offline computer), and be imported to the center.

- With each new user to revoke u_a, the center takes out a dummy user u_d from the list, and updates the t $c_i^{'}$'s, by performing for each of them :

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

  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. If in addition, significantly less than t users are actually revoked, than 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.

B Extracts from [18]

B.1 Basic Revocation Scheme for a Single Round

The basic 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 [20].

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 that the identities of t users whose keys should be revoked. The center broadcast the identities and the personal keys of these users:

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

Each non-revoked user u can combine his personal share $P(u)$ with these t shares and interpolate P to compute the key $S = P(0)$. The center uses S as the new group key with which it encrypts messages to the non-revoked users.

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.

Reducing the computation overhead: The computation of the new group key by a user involves the interpolation of the free coefficient of P, and requires $O(t \log^2 t)$ multiplication using FFT, or $O(t^2)$ multiplications using Lagrange interpolation.

B.2 Multi-Round Revocation

The scheme has t revocation schemes $RS_1, RS_2, ..., RS_t$ of linearly increasing secret sharing sizes $1, 2, ..., t$ 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, than $r_i - 1$ schemes will be discarded.

The scheme has a transmission length of $w = O(r)$ (the broadcast includes the ID's and shares of the newly r_i revoked users), with an additional offline "maintenance" channel of $O(rt)$, used for the transmission of all the shares of the revoked users in all the polynomials with higher degrees than RS_{R_i}, effectively reducing the degree of the other polynomials by r_i. The center cannot reduce the computational load at the user to $O(R)$ by pre-computing most of the Lagrange interpolation formula - since broadcasting the c_i''s would be $w = O(R)$. Therefore, the users need to perform $O(R^2)$ calculations.

If it is required to maintain the capability to revoke up to t users after each revocation round, than re-carding of all the SCs is needed: the center generates new shares and sends them (in unicast) in "offline" ("maintenance") to all the users, in order to renew the schemes.