Realizing Massive-Scale Conditional Access Systems Through Attribute-Based Cryptosystems

Patrick Traynor, Kevin Butler, William Enck and Patrick McDaniel
Systems and Internet Infrastructure Security Laboratory
Department of Computer Science and Engineering
The Pennsylvania State University
University Park, PA 16802
{traynor, butler, enck, mcdaniel}@cse.psu.edu

Abstract

The enormous growth in the diversity of content services such as IPtv has highlighted the inadequacy of the accompanying content security: existing security mechanisms scale poorly, require complex and often costly dedicated hardware, or fail to meet basic security requirements. New security methods are needed. In this paper, we explore the ability of attribute-based encryption (ABE) to meet the unique performance and security requirements of conditional access systems such as subscription radio and pay-per-view television. We show through empirical study that costs of ABE make its direct application inappropriate, but present constructions that mitigate its incumbent costs. We develop an extensive simulation that allows us to explore the performance of a number of virtual hardware configurations and construction parameters over workloads developed from real subscription and television audiences. These simulations show that we can securely deliver high quality content to viewerships of the highest rated shows being broadcast today, some in excess of 26,000,000 viewers. It is through these experiments that we expose the viability of not only ABE-based content delivery, but applicability of ABE systems to large-scale distributed systems.

1 Introduction

The explosion of audio and video content diversity coupled with increasing bandwidth being delivered to the home has lead to increased options for consumers. The *conditional access systems* providing this content predicate access on a variety of arrangements with the user (i.e., subscription vs. pay-per-view). While the security of these systems has been studied for some time, many problems persist. Specifically, content providers often sacrifice simplicity, cost and security to scale delivery to viewerships that may include millions of households. Through the use of new techniques, the magnitude of these tradeoffs may be greatly reduced.

An extension of identity based cryptography, attribute-based systems (ABE) provide a semantically rich tool for implementing encryption policy. Data in ABE is encrypted under a set of identity attributes. Each user of the system is assigned a subset of the group's attributes by a trusted third party. Users possessing k out of the n those attributes can recover the plaintext, where the value of k is at the discretion of the encrypting entity. For example, one could create a system whose attributes are the states in the United States, and encrypt a particular data item under the states beginning with the letter 'A'. Assuming every person in the United States is assigned the attribute of their home state, only people in Alabama, Alaska, Arizona, and Arkansas could retrieve the data. ABE systems are no longer theoretical cryptographic constructs, but they have been implemented and their performance carefully studied [27]. Interestingly, these characterizations have shown that for small numbers of attributes, such systems can be quite efficient.

Attribute systems can address many of the problems presented by conditional access systems. In one model, the ABE would encrypt the content such that any user with a valid subscription or access code (in the pay-per-view case) could recover the plaintext. This model is significantly simpler than existing models, which typically involve dedicated hardware, implement complex key management protocols, and have limited ability to adapt to rapidly changing viewerships. However, while promising, a naïve implementation of ABE will not work well in practice. Past studies have shown that costs grow quickly with the number of attributes, and thus other techniques are needed.

In this paper, we consider the use of ABE in massive-scale conditional access systems. We begin by consider-

ing the unique requirements of current and next generation content providers, and review a number of past attempts to implement similar services. An in-depth performance characterization is used to assess the appropriateness of ABE to this application context. A novel ABE construction that addresses scalability and performance requirements of massive scale groups is introduced, and its parameterization considered. An extensive simulation study is used to develop prescriptive system models used to design conditional access systems for expected group sizes and latency budgets.

The simulations confirm that the ABE can be used to implement security in conditional access services. The major findings of that study are manifold. First, we found that adding computational power (processors) is essential to enduring instantaneous increases in viewership (such as those seen at the beginning of a broadcast). Such computational power surprisingly does little to reduce latencies observed when few membership changes can be observed. Secondly, significant performance gains can be achieved by separating viewership into independent viewer groups, and that such gains are sustainable to a fixed minimal group size. Finally, our study of large-scale groups show that major content providers such as cable companies and television networks could easily deploy sufficient hardware to support national audiences.

The remainder of this paper is organized as follows. Section 2 introduces the structure and needs of conditional access systems. Section 3 briefly describes the cryptographic foundations of ABE systems and characterizes its performance via empirical study. Section 4 introduces a novel construction that mitigates the performance costs of implementing massive-scale groups. Section 5 details an extended simulation study of ABE based conditional access systems, and Section 6 concludes.

2 Conditional Access Systems

This work explores the performance of conditional access (CA) systems[1], which operate over broadcast channels but impose access control requirements for content delivery. There are three general models for implementing CA systems: *subscription-based* services, where a periodic (often monthly) fee is paid in advance for access to service, *pay-per-view* services, where a user signs up in advance for a fixed-length program of interest, and *impulse pay-per-view* services, where the user can sign up for a program without any *a priori* setup required (e.g., the user has a set-top box for cable television service, finds an event of interest and presses a "subscribe" button on their remote control to instantaneously gain access to the program). In these systems, subscriber management may be delegated to a trusted third party because of the complexity of the operation, while subscriber authorization is dealt with by encrypting content and requiring a smart card capable of decryption in the recipient's device, or a separate descrambling box. Authorized devices will have access to the key for decrypting content, often for a fixed amount of time [15].

The totality of subscribers of a particular system is known as the *group*. These groups add members through *join* operations. Members voluntarily exit the group through *leave* operations. Although of minimal importance to the following analysis, leaves are often distinguished from *rejections* (also known as revocations) in that the latter is not voluntary. Also known as the group size and without loss of generality, the viewership is the number of members in a group.

Groups in CA systems have performance requirements specific to their environments. Join operations need to be near instantaneous–users desire to get content as soon as payment is made or the channel is tuned. Conversely, *leave* processing is often less urgent. Systems are willing to continue delivering content to a previously joined member for some period, particularly if doing so allows lower latency joins. Of course, content delivery is of paramount importance. Content playback often has real-time requirements, and thus any noticeable latency, if even for a moment, is unacceptable. We consider how to meet these often stringent performance requirements in an ABE-based secure content delivery system through, and review past attempts at securing these systems in the next subsection.

2.1 Related Work

Cryptographic mechanisms are a natural means of managing membership in large groups. In multicast systems, for example, a number of systems have attempted to use a public key infrastructure (PKI) [19] and secure group communications [20, 32, 12, 24, 34, 35] to address access control. The MARKS [10] and Nark [11] schemes reduce the impact of joins and leaves in the above schemes at the expense of frequently rekeying at regular intervals. Broadcast encryption schemes [16, 25, 9] improve over the above mechanisms by removing the requirement of bidirectional communication. However, such techniques have been considered limited in their ability to concurrently express multiple complex policies.

A promising new building block for creating distributed systems is attribute-based encryption (ABE). A generalization of identity-based cryptography [30, 8, 14], ABE systems use a collection of attributes as the basis of cryptographic primitives. Using threshold constructions such as those suggested by Sahai and Waters [28], users in possession of at least k-out-of-n attributes can gain access to

[1] The authors note the unfortunate collision between acronyms for conditional access systems and certificate authorities.

encrypted content. Such primitives have been extended by Goyal et al. [17] and Bethencourt et al. [5] to bind tree-based access control structures directly into keys and ciphertexts, respectively. Others have created protocols to limit the exposure of a principal's attributes [2, 22]. Pirretti et al. [27] were the first to demonstrate that such primitives were both expressible and efficient enough to use as the basis of real systems; however, their work did not address how the specific embodiment of policy impacts performance. Without understanding the implications of such choices, it becomes possible to build a system in which the implementation of policy prevents a system from meeting its performance requirements.

Many applications already rely on attributes as means of managing users. Attribute-based access control (ABAC) systems [7, 36] base policy decisions on the attributes assigned to users and resources. Attribute-based messaging systems [6] automatically create mailing lists by reconciling system policy with sender specified attributes. Current attribute-based systems, however, use traditional cryptographic constructions and rely upon central administration of policy. These applications therefore do not scale well to large-scale or distributed systems. Because the application of ABE primitives may significantly expand the flexibility of these and many other systems, we investigate how such constructions can be most efficiently applied.

3 Attribute-Based Systems

Before discussing the construction of efficient policy, we informally define and characterize encryption policies in an attribute-based system. An *attribute policy*, or simply policy throughout, is the specification of the attributes necessary to gain access to an object (e.g., file, session, etc). Because such policies are bound to associated objects using a series of cryptographic operations, enforcement in a distributed environment is possible.

3.1 Attribute-Based Encryption

We begin our discussion of policy by offering a high level explanation of the functionality provided by the underlying cryptographic primitives. A generalization of Identity-Based Encryption (IBE) [30, 14, 8], Attribute-Based Encryption (ABE) allows a set of strings to describe users. For example, a member of a basketball league with an online forum may be represented by a set of attributes $A = \{\ Guard,\ Over\ 7'\ Tall$ and $Left\ Handed\}$. Those users interested in recruiting teammates satisfying at least k-out-of-n of these characteristics can encrypt messages to such players using only these strings and the system's public parameters. Such exchanges can occur without the need for additional per-user public key certificates.

Systems using ABE implement four high-level algorithms. The first, *Setup*, takes a threshold value k as input and generates a master key MK and the system's public parameters. To create a user, the authority runs the *Key-Gen* algorithm with MK and the set of attributes S to generate a user's secret key SK. Note that the size of the set of attributes $S = \{A_0, A_1, \cdots, A_{x-1}\}$ assigned to a user does not necessarily match k or n; rather, the universe of attributes A can be infinite whereas the number of attributes used in an atomic expression of policy is bound by k and n. Users can *Encrypt* an object o under a set of attributes S' using the public parameters. Encrypted objects can then be accessed using the *Decrypt* function, which ensures that $|S \bigcap S'| \geq k$ before recovering o.

We rely upon a variant of the Sahai-Waters Large-Universe construction [28] to implement our systems. This construction computes a bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ between k attributes and pieces of a private key. The performance of encryption and decryption are therefore functions of n and k, respectively. Note that any cryptosystem capable of providing k-out-of-n attribute threshold semantics with resistance to collusion would be equally effective.

In order to provide additional flexibility and significantly increase performance, our variant implements a random oracle construction [3, 13]. The random oracle construction replaces the most computationally expensive component of the Sahai-Waters construction with a hash function. As long as the security of the hash function is sufficient, the random oracle construction can be used to dramatically reduce performance costs. The random oracle construction also allows for n to be variable, allowing expressions in a single cryptosystem to be expressed with as few attributes as is necessary (i.e., no padding or "default" attributes). As was demonstrated by Pirretti et al. [27], the combination of elliptic curve type and the random oracle construction can reduce encryption costs by more than 98%. We therefore use this construction throughout the remainder of this work. A more formal definition of the Sahai-Waters construction is offered in the Appendix.

Finally, because of the expense associated with ABE, we use these operations as a key encapsulation mechanism (KEM). Specifically, the cryptographic primitives protect an AES key, under which associated data is encrypted. Such techniques are standard in most public-key systems.

3.2 ABE Performance

In order to understand how to apply ABE to real systems, we must first characterize its performance. We use the ABE library created by Pirretti et al. [27] and leverage their characterization of performance as a basis for further exploration. We have provided extensive updates to improve compatibility with the most recent release (0.4.9)

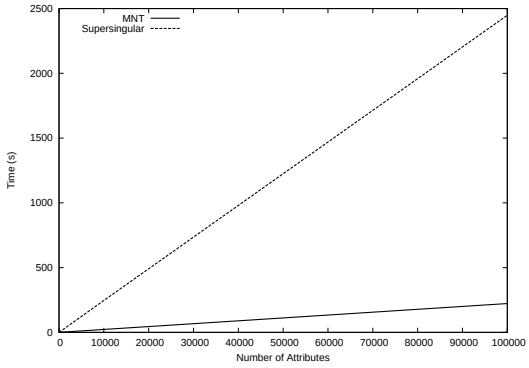

Figure 1. Cost of encryption using MNT and supersingular curves in a 1-out-of-n attribute system.

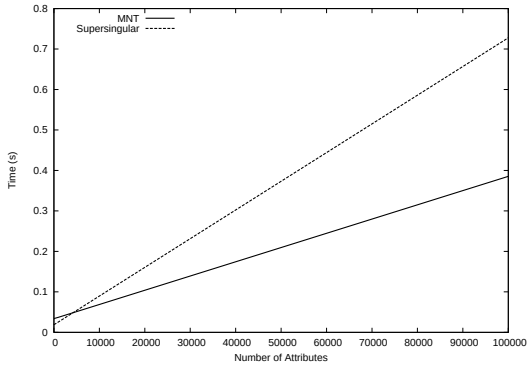

Figure 2. Cost of decryption using MNT and supersingular curves in a 1-out-of-n attribute system.

of the Pairing-Based Cryptography (PBC) library [23] and to streamline functionality. System characterization experiments were conducted on Dell workstations with Intel Core Duo 2 processors and 1 GB of RAM using Linux kernel version 2.6.20. Each experiment was conducted 500 times to ensure statistical significance.

We first validated the results from Pirretti et al.'s performance analysis, due to the updated libraries and different underlying computing platform (Pirretti et al. performed experiments on Apple G5 XServe machines running Mac OS X Server). Our experiments focus on encryption and decryption operations specifically. Because we are interested in very large-scale cryptosystems, factors such as system setup and key generation are less interesting, as such operations are likely to occur infrequently. In addition, because we intend to use each attribute as a unique identifier in massive-scale systems, we limit our discussion to strictly on 1-out-of-n cryptosystems.

A major difference between our investigations and those of Pirretti et al. is the scale of considered attributes. While the original investigations considered a maximum of 32 attributes, we profile ABE systems with up to 100,000 attributes. It is thus necessary to determine whether the characteristics found by Pirretti et al. scale.

We first examined the use of MNT versus supersingular curves using the random oracles model. Pirretti et al. found that MNT curves vastly outperform supersingular curves for encryption as the number of attributes increases. We validated these results, as shown in Figure 1. To establish a relationship between encryption time and the number of attributes, we performed a least-squares regression over the MNT data. This resulted in the following equation:

$$E = 2.2214 * 10^{-3} n + 0.01804 \quad (1)$$

where E is the required encryption time and n is the number of attributes. This equation fits the data with extremely high correlation of $r^2 = 0.99999997$.

Our results for decryption of n policies stand in contrast to Pirretti et al. Decryption costs are relatively constant for the first 1000 attributes, with supersingular curves performing comparably or slightly better to MNT curves. Recall that performance of ABE systems decryption is a function of k rather than n. Since we are examining a fixed 1-out-of-n system ($k = 1$), decryption should theoretically remain constant. However, we found that as the number of attributes increases past 1000, decryption cost increased proportionally to n. More surprisingly, MNT curves perform substantially faster than supersingular curves. These results are shown in Figure 2. While Pirretti et al. demonstrated an order of magnitude performance difference for decryption in favor of supersingular curves, these results show that MNT curves perform better for large-scale systems. A regression analysis of the data yielded the function

$$D = 3.5159 * 10^{-6} n + 0.033791 \quad (2)$$

where D represents required decryption time and n is the number of attributes. Correlation was very high, yielding a value $r^2 = 0.9999992$.

To determine the cause of linearity in the decryption operation and to vet the accuracy of our operations, we extensively profiled the PBC library and our code using *gprof* [18] and *Valgrind* [26]. We found that much of the decryption costs for a large number of attributes result from AES decryption of the ciphertext; the operation necessitates

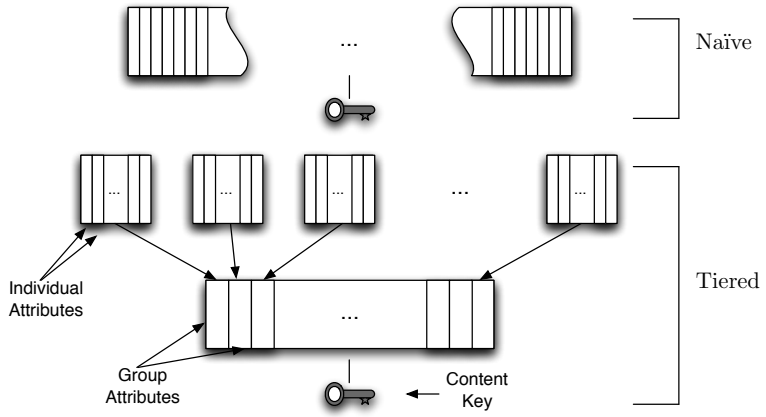

Figure 3. A comparison of the naïve and tiered constructions for providing efficient access to content keys. Because join and leave operations can be performed on groups, the tiered construction is more efficient.

use of a `for` loop based on the number of attributes as part of an HMAC calculation to validate the content before the decryption occurs. This accounts for the non-constant time profile. We also found that the increased performance of MNT versus supersingular curves may be related to changes to the PBC library that have optimized MNT curves, potentially to a larger extent than with the supersingular curves.

Decryption is a much less costly operation than encryption, with even 100,000 attributes requiring less than 500 ms to decrypt and only approximately 33 ms required for 1,000 attributes or fewer. Thus, even a low-powered device with limited computational ability, such as a mobile phone or PDA, will be able to complete attribute-based decryptions in a matter of a few seconds or less. On desktop-class machines, the cost of decryption is virtually negligible.

Based on the data we have observed, we assume that any large-scale attribute-based system will use MNT curves with a random oracle model. In the next section, we consider how these measurements relate to a real system that has practical deadlines.

4 ABE-CA Access Structure

The real-time performance requirements of CA systems mandate the use of symmetric key content encryption. Specifically, the overhead of public-key operations negates their use as the primary protector of data. However, such schemes are not without value. A number of content protection systems rely on public-key cryptography to distribute symmetric keys (e.g., AACS [1]). Because ABE greatly simplifies the management of public-keys, we seek to apply this technique to the problem of content protection.

Naïvely implemented, however, the use of such cryptographic primitives cannot meet even modest performance requirements. As shown in the previous section, the linear scaling of encryption cost with the number of attributes in the system means that large scale operations cannot be performed efficiently. For example, when encrypting an AES content key for a system with one million users, the average encryption time would require approximately 37 minutes. If we wish to change the content encryption key at any time during a half-hour broadcast, such operations would simply not be supported by the current construction.

The linearly increasing cost of encryption motivates a more efficient construction. Specifically, while encrypting data for enormous numbers of individuals fails to meet performance goals, the encryption of data for a handful of groups can be achieved within such a system. For example, if the AES content key is to be changed once per minute, an average encryption time of 2.24 seconds for groups of 1,000 attributes easily meets this performance target. Accordingly, the challenge to efficiently using ABE in this setting becomes a problem of efficiently mapping users into groups.

Figure 3 provides an overview of this tiered construction. To achieve these ends, we begin by dividing the user population into groups of size n. Because of the linear scaling of the cost of encryption with the number of users in the system, such partitioning in and of itself fails to improve performance (i.e., linear scaling means that 10 encryptions of 1-out-of-10 expression require the same amount of time as a single encryption in a 1-out-of-100 cryptosystem). Each of these groups therefore contains an encrypted "group" attribute. All users capable of decrypting this new attribute can then use it to decrypt the object containing the AES content key. When the content key is changed by the ad-

ministrator of the system, users can continue to apply their group attribute to efficiently access new AES keys. Because the number of attributes n' encrypting the content key is much smaller than the total number of users in the system, the cost of encrypting new keys is inexpensive.

New users gain access to content through the *join* operation. The system begins by finding a group with less than n members. The group attribute corresponding to this group is then re-encrypted using the attributes of both the new user and current members. The new user then decrypts the group attribute using his unique attribute and applies the group attribute to decrypt the content key. Note that other members of the group are unaffected by the addition of a new member as the group attribute remains unchanged. At the end of a subscription period or due to a forced revocation (e.g., illegally cloned device detected), the system executes a *leave* operation. To prevent the leaving user from accessing future content, the content key and group attribute are both updated. A new group attribute, encrypted under the set of user attributes minus the removed member, must then be decrypted by remaining members.

The benefits to this approach are numerous. The time required to perform a join operation (i.e., add a user to the system) becomes a function of n and not the size of the entire population. Moreover, the division of the population into groups allows for the parallelization of joins given the availability of multiple processors. Compared to the naïve construction, the cost of performing a leave operation is also drastically reduced. Users in the same group as the leaving party must decrypt a new group attribute before they can access a new content key. Users in other groups, however, are unaffected and can continue to apply their group attributes to recover the new content key.

As presented thus far, our tiered structure establishes a one-to-one mapping between groups and group attributes. However, as the number of users in a system grows, such a relationship may become unsustainable. While group attributes can be added to support unique new groups, application performance requirements will bound the size of n' to maintain inexpensive rekeying. Additional groups may therefore be required to share group attributes via pigeonholing. The impact of this and a number of other design tradeoffs is investigated in greater detail in the next section.

5 ABE-Enabled Systems

We now apply the constructions defined in the previous section to massive-scale systems with real-time performance requirements.

5.1 Simulation

The tiered ABE construction proposed in Section 4 provides various tunable parameters to meet performance requirements. In this section, we use simulation to consider the effects of each parameter on realistic workloads. We simulate the server side operations for our tiered construction using the encryption cost function derived in Section 3.2. Our simulator exports the following performance sensitive parameters:

n	:	The group cryptosystem size
n'	:	The content cryptosystem size
D	:	The batch duration
P	:	The processor pool size
K	:	The content attribute key pool size
KP	:	The number of content attribute key pool processors

As described in Section 4, our tiered ABE construction breaks clients into groups of size n. The group size determines both the first level encryption cost as well as the number of groups in the system. The content cryptosystem size n' directly impacts content encryption, and therefore must be kept at a minimum. As the number of groups will commonly be greater than n', multiple groups will map to each attribute position in the content cryptosystem. When a client leaves, the corresponding attribute in the content cryptosystem must change, thereby requiring all groups corresponding to that attribute position to encrypt a new key.

Client join and leave operations result in new group tasks. Task execution may be postponed with a batch duration, D. Tasks are queued for each group and ultimately scheduled on one of the P processors in the processor pool; more processors allows more tasks to execute in parallel. When a group acquires a processor, all queued tasks are aggregated and the group's content attribute key is re-encrypted for the current membership. Finally, on client leave, a new content attribute key is obtained from the content attribute key pool (initially of size K), which is maintained by KP separate processors. Unless otherwise noted, we use $K = 1000$ and $KP = 1$.

The remainder of this section analyzes the performance of our tiered ABE construction. We begin by describing the realistic workloads used as simulation inputs. We then investigate the comparatively simpler "join-only" workloads, which do not require revocation. Then, we incorporate workloads with leave operations. Finally, we incorporate our findings and consider a high-demand, performance sensitive workload.

5.2 Modeling Workloads

To profile massive-scale group management systems, we create four classes of system behavior: *Impulse Pay-Per-*

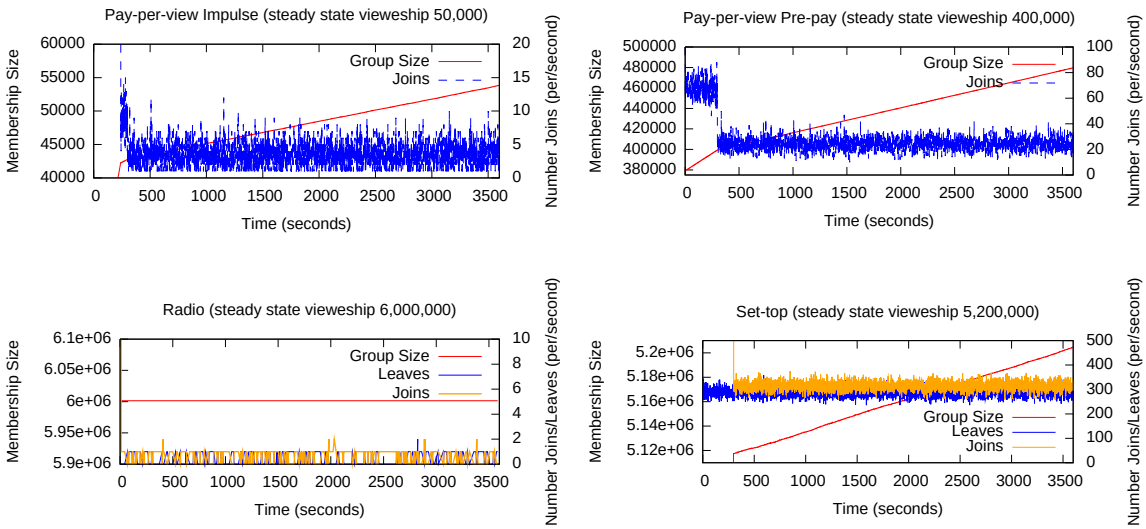

Figure 4. Sample traffic patterns (from top to bottom) Impulse Pay-Per-View, Prepaid Pay-Per-View, Radio and Set-Top Box scenarios.

View, Prepaid Pay-Per View, Radio and *Set-Top Box*. Each of these usage patterns, which are profiled in Figure 4, are described in detail below.

Impulse Pay-Per-View: Users may decide to purchase certain types of content on impulse. For example, consumption of content by guests at a nation-wide hotel chain is unlikely to planned far in advance. For some initial mean number of viewers, we model such behavior for an hour long stream as follows: 50% of users join the system in the first minute; 90% of users join within the first five minutes; 100% of the mean have joined within the first ten minutes. Throughout the remainder of the hour, the number of users grows by 2%. Because such access is sold on a per-program basis, no users leave the system during the duration of the simulation. We investigate such systems with a mean of 50k, 100k and 500k viewers.

Prepaid Pay-Per-View: The purchase of other types of pay-per-view content is more predictable. For example, sporting events such as boxing or concerts typically see the majority of viewers subscribe well before the start of an event. We model this behavior as follows: 95% of the viewership attempts to join within the first five minutes. Throughout the remainder of the hour, the number of viewers continues to grow by 2%. Like the impulse pay-per-view case, we assume that access is sold on a per-program basis and that leaves all occur after the simulation. We use both average pay-per-view boxing ratings (400,000 viewers [21]) and the most popular pay-per-view event (Tyson vs Holyfield II: 1.99 million viewers [21]) to characterize such systems.

Radio: A number of subscriber-based systems operate over longer periods of time. Satellite-radio subscriptions, for instance, can be purchased for intervals of months or years. Accordingly, such systems present an interesting case for steady state analysis as they do not exhibit the initial spikes seen in the previous examples. We use subscriber data from Sirius Satellite Radio [31] as the basis for modeling long-term subscription services. Specifically, we assume a mean of six million users with a 2.8% join rate and a 2% leave rate.

Set-top Boxes: In our final model, we examine the "Pay Per Channel" subscription approach. In this model, viewers are only charged for the channels they watch. From the perspective of the set-top box, each change of a channel becomes equivalent to a join operation. Accordingly, such a model would need to support an extremely large number of users. We characterize this model by using recent Nielsen ratings for average (The Tonight Show: 5.22 million viewers [4]) and extremely popular (American Idol: 26.9 million viewers [33]) broadcast numbers. We assume that 100% of the mean number of users tune in to such shows uniformly over the first five minutes and that joins and leaves occur evenly at a rate of 2% throughout.

5.3 "Join-only" Systems

The *join-only* philosophy allows for the creation of systems with simple billing policies. Users are charged for the duration of a program, regardless the actual amount of con-

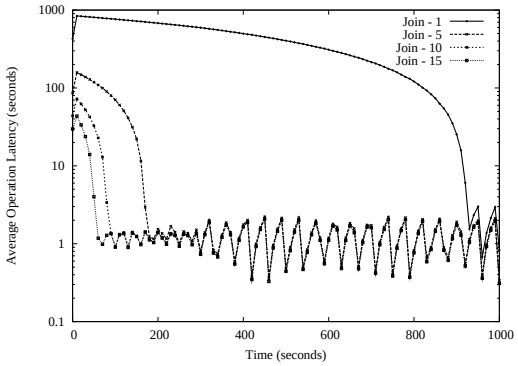

Figure 5. The approach toward steady state for a varying number of processors. Once all groups reach steady state, the extra processors used to decrease quiescence time remain idle.

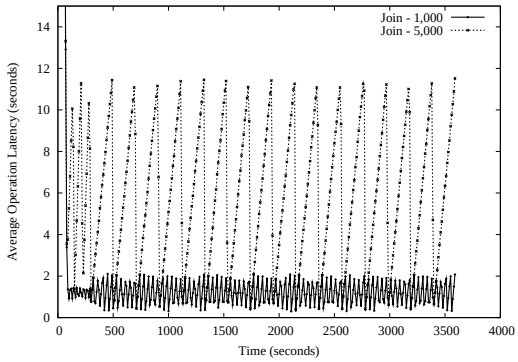

Figure 6. The cost of a user join at steady state for $n = 1000$ and $n = 5000$. The saw-like pattern is the result of the group growing from 1 to n, after which a new group is created.

tent consumed. For instance, a hotel patron purchasing a movie is charged for the entire program whether or not they watch it in its entirety. Accordingly, users in such a system perform join but not leave operations during content distribution. Both the Impulse and Prepaid Pay-Per-View models described in the previous subsection fall into this category.

We begin with an analysis of the latency caused by the initial burst of joins in the system. Because the vast majority of viewers enter within the first few minutes of operation, resource allocation for such systems must minimize join latency. Figure 5 shows the time required to reach quiescence (i.e., steady state) as a function of the number of processors in the system. Because the number of users joining far exceeds the number of processors, nearly all requests are initially queued in the system. This initial queuing has a number of significant repercussions on the system. For instance, changing the number of users in each user-layer expression has no measurable initial impact on system performance given a fixed number of processors. Because of the linear cost of encryption discussed in Section 3.2, the time required to encrypt the initial rush of users is the same for a single large group or a number of smaller subgroups. As the number of processors is increase, so too does the speed with which the system reaches quiescence.

Batching, as described in the previous section, also has no measurable benefit to the system. In the presence of multiple processors, batching in fact degrades performance. This phenomenon is a result of the inherent batching that occurs while waiting for a free processor. When a processor becomes available, the size of the initial burst ensures that at least n users are waiting to join the system. Accordingly, pausing for s seconds simply adds s extra seconds to join latency.

While the addition of processors dramatically reduces time to quiescence, the presence of additional processors does not necessarily benefit a system in steady state. As is shown in Figure 5, the low join rate throughout the remainder of a program can often typically be handled by a single processor. As derived from our performance evaluation, if the number of steady state joins is less than 435 per minute, the additional processors added to reduce the initial rush simply lay idle for the remainder of a program. A system designer must therefore decide between high latency using a small number of processors and low utilization when using more than one. If the start of programs can be offset (i.e., movies begin every 15 minutes), a compromise between these two extremes can be achieved.

While the size of user-layer groups does not affect the network behavior before quiescence, its impact on join latency becomes apparent during steady state operation. Figure 6 compares the joins for $n = 1000$ and $n = 5000$. Accordingly, in systems where mid-program leaves are unlikely or impossible, there is no clear advantage to using large groups.

5.4 "Join and Leave" Systems

Leave operations are computationally expensive. As discussed in Section 4, a system leave requires every group containing that attribute to re-encrypt (i.e., if 100 groups contain the attribute, 100 re-encryptions are necessary). We consider the effect of leaves and methods of minimizing their impact as part of our exploration into the radio and

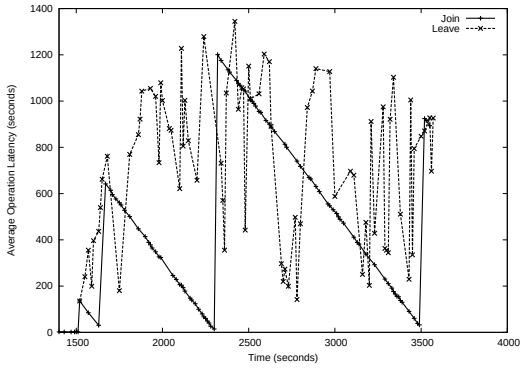

Figure 7. A satellite radio subscription model for 6 million users. With 10 processors dedicated to computation, there is insufficient ability to process all requests for a reasonable deadline.

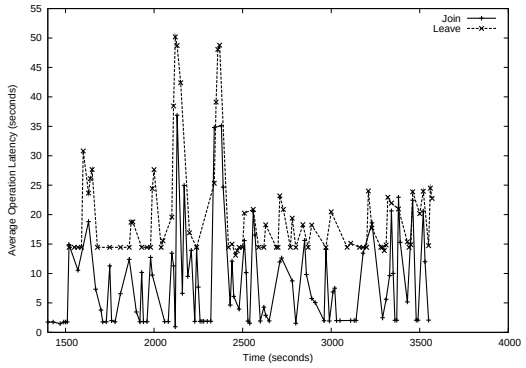

Figure 8. The satellite radio model with 100 processors deployed. The costs of joining and leaving the system are now under one minute.

set-top box models.

Radio Model: Recall that in our satellite radio model, we consider a monthly subscriber to the system. We assume that joins and leaves, corresponding to new subscriptions and account cancellations, are uniformly distributed. To model this steady state behavior we "warm" the system by simulating the initial addition of the 6 million users into the system. We wait until the system quiesces to steady state behavior before performing measurements. Figure 7 shows that given 10 processors and the same assumptions as previously considered for the pay-per-view situations, the system is not stable. Regardless of the number of subgroups, because the cost of leaving the system is so high, the latency to join the system spikes to intolerable levels, over 20 minutes in some cases. We can mitigate this behavior by adding extra processors into the system; Figure 8 shows that a ten-fold increase of processors to 100 brings the latency bounds down considerably. Joins in this case require less than 40 seconds even when considerable load exists on the system, while a user is fully revoked from the system in less than one minute. Given that these are essentially one-time operations from the point of view of the user (i.e., the user's subscription becomes active), an activation time of less than one minute should be within a user's expectations for product activation.

To minimize the reliance on extra processors, an alternate method of maintaining a latency budget is to increase the size of n', the number of attributes in the content cryptosystem. To minimize the cost of decryption to the user, we have used $n' = 10$ in our simulations to this point; as discussed in Section 3.2, this entails a cost of under 34 ms on a desktop-class machine. By increasing n' to 100, the cost of a decryption increases by less than 1 ms. Simply put, decryption costs on even consumer equipment will be minimal. [2] Figure 9 shows that increasing n' provides a similar performance benefit to increasing processors; even under load, joins will occur in under 40 seconds, while leaves will occur in less than 50 seconds.

If even more stringent performance requirements are mandated, we can combine the optimizations of increasing processors and increasing the size of the content cryptosystem. As shown in Figure 10, the time required for joins and leaves is reduced by a factor of more than 10 over either solution used singly. While the previous solutions may be feasible for solutions where up to a minute in latency is acceptable, this optimization allows for tight latency budgets.

Set-Top Box Model: In contrast to satellite radio with its relatively few joins and leaves, our proposed model for set-top box usage has millions of user joins in the first few minutes of a program's commencement, and additional joins and leaves throughout the broadcast. Thus, this model is much more resource-intensive than any previously considered situation. Both optimizations from the satellite radio model are required; Figure 11 demonstrates that for a broadcast of with 5.2 million users, both an increase in n' to 1000 and an increase in the number of deployed processors to 1000 are required to ensure that joins and leaves achieve a tolerable latency.

Our largest dataset involved modeling the viewership of

[2] Satellite radio receivers already perform buffering to ensure a constant stream of content to the user; these buffers may also be used to mask any decryption costs incurred by the receiver.

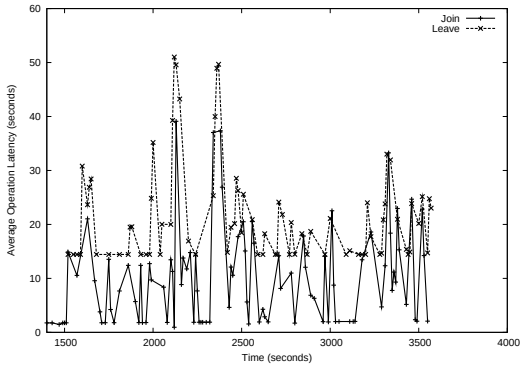

Figure 9. Increasing the size of n', the number of attributes in the content cryptosystem, reduces peak latency from minutes to seconds in the satellite radio model while incurring a nominal increase in decryption costs.

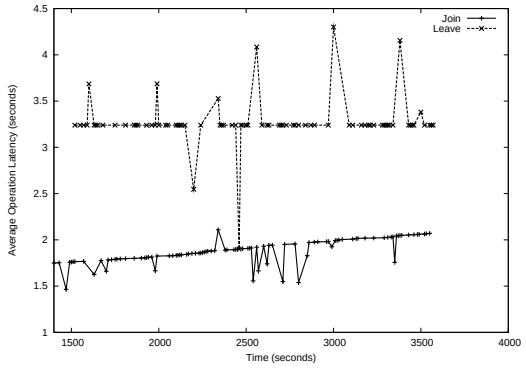

Figure 10. Combining the optimizations of increased processors and increased content cryptosystem attributes in the satellite radio model allows operations to be bounded by tight latency requirements.

"American Idol", with 26.9 million viewers.[3] To provide similar latency results necessitated increasing the number of processors to 5000, as shown in Figure 12. Interestingly, this increased user base is slightly more than 5 times as large as the previous 5.2 million user case, and increasing the number of processors by a factor of 5 results in latencies that are similar, if slightly higher. This attests to the overall relative linearity between the addition of processors and a decrease in latency for systems under high load.

An additional factor that required consideration for this model is the size of the available key pool. Because of the tremendous number of users joining the system, during our simulations, we ran into issues of key exhaustion. One key processor is capable of generating a key every 46.3 ms (a value obtained during the ABE performance characterization in Section 3.2) until the key pool is filled. Up until this set of experiments, designating one server to be the key processor was sufficient; in this case, however, the key pool was depleted so rapidly that the key processor could not generate additional keys in time. As a result, we delegated 100 key pool processors for this experiment to prevent exhaustion. Determining an exact number of required key pool processors is a situationally-dependent optimization that we defer for future work.

[3] We make the simplifying assumption that each viewer has their own set-top box. This represents an upper-bound on the particular broadcast, but for special events, this number of set-top boxes tuned to an special event is certainly feasible.

5.5 Summary

Based on our experiences understanding the operation of ABE-based CA systems across many hundreds of tests, we have determined a set of guiding principles to assist system architects with implementing a performance solution optimized for their unique constraints.

Adding processors helps get to steady-state, but no further. The addition of processors decreases the amount of time required for a system to quiesce after a large series of joins (e.g., the pay-per-view case), and is necessary to augment a system that cannot manage its load. Once steady state is reached, however, extra processors do not provide any additional benefit above what is necessary for maintaining the steady state. The upshot is that deploying large amounts of hardware will not garner additional gains, and a system designer should be cognizant of offered and potential loads before making large-scale hardware purchases.

Increasing n' gives the same benefit as adding processors. The radio and set-top models illustrate that while leaves exact a computational load on the system, this can be mitigated by increasing n', the number of attributes in the content cryptosystem. A 10-fold increase in n' provides similar results to a factor of 10 increase in the number of processors deployed in the system. The cost of this optimization is a corresponding increase in attribute decryption time by the client. However, as we previously showed in section 3.2, the costs of decryption are sufficiently small

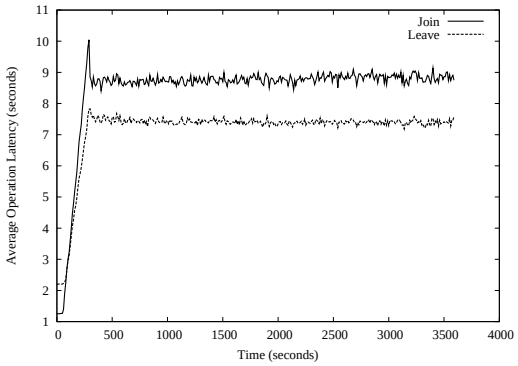

Figure 11. Set-top box model with 5.2 million users. 1000 processors and an increase in n' are necessary to minimize latency.

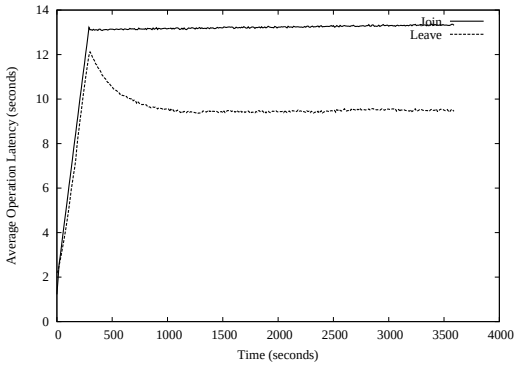

Figure 12. Set-top box model with 26.9 million users. To achieve similar latency bounds to the 5.2 million user case, 5000 processors are necessary.

that we can safely increase n' to any value less than or equal to 1,000 without any meaningful noticeable effect on client performance.

National audiences can be supported by major content providers. Our large-scale simulations show that a system capable of supporting tens of millions of users is easily deployable by major providers such as cable companies and television networks. The costs of adding users is often front-loaded at the beginning of a program, and the corresponding costs in hardware incurred by major providers is amortized by reusing processors for multiple programs. A number of server racks in a data center to support a massive scale audience is a feasible assumption for back-end computational power.

We determined some secondary results as well during the course of our investigations. They can be summarized as follows.

Be aware of key exhaustion for massive systems. It is important that the key pool not be exhausted due to a high rate of user leaves from a system. If this behavior among users is expected in a short time period, we recommend additional investment in dedicated key pool processors.

Let the system do the batching. Our initial intuition indicated that adding batching delays would be beneficial, as less overall computation would result. As we discovered, the system performs its own implicit batching due to processors being unavailable during their encryption cycle; when they return, they process the batch of requests that have been queued. Adding explicit batch delays only caused the system to perform more slowly when it reaches steady state.

Optimize groups for increased benefit. Our solutions often modeled a worst-case scenario, as we did not attempt to optimize group membership. For example, we can mitigate the effect of leaves by ensuring that, for example, users who have monthly subscriptions are binned into different groups than those who have annual subscriptions. This strategy allows for a minimal number of groups to be affected. We defer an analysis of group scheduling strategies for future work.

6 Conclusion

In this paper, we have explored the ability of ABE to meet the unique requirements of conditional access systems. Such an investigation would seem to be doomed from the start: ABE systems employ heavyweight constructions that appear at odds with the enormous and often fluid viewerships of the target content groups. Quite in contrast, our simulations of realistic, massive-scale programming shows that through novel constructions, we can meet this challenge using inexpensive commodity hardware.

What remains is a more direct investigation. We have already built prototype interfaces of ABE systems, and have begun the process of integrating these systems with content delivery services. It is through these latter experiments that we hope to further establish the viability of not only conditional content systems, but promote the use of ABE to implement massive scale distributed systems.

References

[1] Advanced Access Content System. Advance Access Content System Home. http://www.aacsla.com/home, 2007.

[2] G. Antenise, M. Blanton, and J. Kirsch. Secret Handshakes with Dynamic and Fuzzy Matching. In *Proceedings of the ISOC Network & Distributed System Security Symposium (NDSS)*, 2007.

[3] M. Bellare and P. Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, pages 62–73, 1993.

[4] M. Berman. The Programming Insider. www.mediaweek.com/mw/newsletters/proginsider/index.jsp, Accessed May 4th, 2007.

[5] J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-Policy Attribute-Based Encryption. In *Proceedings of the IEEE Symposium on Security & Privacy (S&P)*, 2007.

[6] R. Bobba, O. Fatemieh, F. Khan, C. A. Gunter, and H. Khurana. Using Attribute-Based Access Control to Enable Attribute-Based Messaging. In *Annual Computer Security Applications Conference (ACSAC)*, 2006.

[7] P. A. Bonatti and P. Samarati. A uniform framework for regulating service access and information release on the web. *Journal of Computer Security*, 10(3), 2002.

[8] D. Boneh and M. K. Franklin. Identity-based encryption from the Weil pairing. In *Proceedings of Advances in Cryptology (CRYPTO)*, 2001.

[9] D. Boneh, C. Gentry, and B. Waters. Collusion Resistant Broadcast Encryption With Short Ciphertexts and Private Keys. In *Proceedings of Advances in Cryptology (CRYPTO)*, 2005.

[10] B. Briscoe. MARKS: Zero Side Effect Multicast Key Management Using Arbitrarily Revealed Key Sequences. In *Proceedings of the International Workshop on Networked Group Communication*, 1999.

[11] B. Briscoe and I. Fairman. Nark: Receiver-Based Multicast Non-Repudiation and Key Management. In *Proceedings of the ACM conference on Electronic commerce*, 1999.

[12] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas. Multicast security: A taxonomy and some efficient constructions. In *Proceedings of IEEE INFOCOM'99*, 1999.

[13] R. Canetti, O. Goldreich, and S. Halevi. The Random Oracle Methodology, Revisited (Preliminary Version). In *STOC*, pages 209–218, 1998.

[14] C. Cocks. An identity based encryption scheme based on quadratic residues. In *IMA Int. Conf.*, pages 360–363, 2001.

[15] EBU Project Group B/C/A. Functional model of a conditional access system. *EBU Technical Review*, (266):64–77, Winter 1995.

[16] A. Fiat and M. Naor. Broadcast Encryption. In *Proceedings of Advances in Cryptology (CRYPTO)*, 1993.

[17] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, 2006.

[18] S. L. Graham, P. B. Kessler, and M. K. Mckusick. Gprof: A Call Graph Execution Profiler. In *Proceedings of the 1982 SIGPLAN Symposium on Compiler Construction*, Boston, MA, USA, June 1982.

[19] T. Hardjono and B. Weis. The Multicast Group Security Architecture. RFC 3740 (Informational), Mar. 2004.

[20] H. Harney and C. Muckenhirn. RFC 2093: Group Key Management Protocol (GKMP) Specification. http://www.faqs.org/rfcs/rfc2093.html, 1997.

[21] C. Jay. Can De La Hoya-Mayweather fix boxing's ills? http://msn.foxsports.com/boxing/story/6772742, 2007.

[22] A. Kapadia, P. P. Tsang, and S. W. Smith. Attribute-Based Publishing with Hidden Credentials and Hidden Policies. In *Proceedings of the ISOC Network & Distributed System Security Symposium (NDSS)*, 2007.

[23] B. Lynn. PBC library. http://rooster.stanford.edu/~ben/pbc/, 2007.

[24] P. McDaniel, A. Prakash, and P. Honeyman. A flexible framework for secure group communication. In *USENIX Security Symposium*, pages 99–114, 1999.

[25] D. Naor, M. Naor, and J. Lotspiech. Revocation and Tracing Schemes for Stateless Receivers. 2001.

[26] N. Nethercote and J. Seward. Valgrind: A Program Supervision Framework. *Electronic Notes in Theoretical Computer Science*, 89(2):44–66, Oct. 2003.

[27] M. Pirretti, P. Traynor, P. McDaniel, and B. Waters. Secure Attribute-Based Systems. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, 2006.

[28] A. Sahai and B. Waters. Fuzzy identity based encryption. In *Proceedings of International Cryptology Conference (Eurocrypt)*, 2005.

[29] A. Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

[30] A. Shamir. Identity-based cryptosystems and signature schemes. In *Proceedings of Advances in Cryptology (CRYPTO)*, 1985.

[31] SIRIUS, Inc. SIRIUS Satellite Radio Reports Strong First Quarter 2007 Results. http://investor.sirius.com/ReleaseDetail.cfm?ReleaseID=240128, 2007.

[32] M. Steiner, G. Tsudik, and M. Waidner. Diffie-Hellman Key Distribution Extended to Group Communication. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, 1996.

[33] USA TODAY. Nielsens ratings for April 23 – April 29. http://www.usatoday.com/life/television/news/nielsens-charts.htm, 2007.

[34] D. M. Wallner, E. J. Harder, and R. C. Agee. RFC 2627: Key management for multicast: Issues and architectures. http://www.faqs.org/rfcs/rfc2627.html, 1997.

[35] C. K. Wong, M. G. Gouda, and S. S. Lam. Secure Group Communications Using Key Graphs. In *Proceedings of the ACM SIGCOMM Conference on Applications, technologies, architectures, and protocols for computer communication*, 1998.

[36] T. Yu, M. Winslett, and K. E. Seamons. Supporting structured credentials and sensitive policies through interoperable strategies for automated trust negotiation. *ACM Trans. Inf. Syst. Secur.*, 6(1):1–42, 2003.

Appendix

The Sahai-Waters construction [28] computes a bilinear map between k components of the ciphertext with corresponding pieces of the private key. The resulting key is derived by interpolation over these pieces using Shamir's secret sharing. Lagrangian coefficients are computed in the domain $\mathbb{Z}_p$ using:

$$\Delta_{i,S}(X) = \prod_{j \in S, j \neq i} \frac{x-j}{i-j}.$$

Additionally, we assume all systems will work in some predetermined bilinear group $\mathbb{G}$ of appropriate size.

The operations performed by a cryptosystem are:

Setup(k): The setup algorithm chooses a random exponent $y \in \mathbb{Z}_p$, creates a public parameter $Y = e(g,g)^y$ and sets the threshold value k. The public key and the secret exponent y become the master key.

Key-Gen(S, MK): Let $H : \{0,1\}^* \rightarrow \mathbb{Z}_p$ be a collision-resistant hash function and let $T : \mathbb{Z}_p \rightarrow \mathbb{G}$ be a function that we will model as a random oracle [3].

We define $\Gamma = \bigcup_{s \in S} H(s)$ as the set of all hashed attributes assigned to the user. The authority then generates a new random polynomial $q(x)$ with degree $k-1$ over $\mathbb{Z}_p$ such that $q(0) = y$. For all $i \in \Gamma$, the authority selects a random r_i, yielding the private keys components:

$$D_i = g^{q(i)} T(i)^{r_i}, d_i = g^{r_i}$$

Encrypt(M, S', PK): The encryption algorithm begins by computing the set of hashes for the attributes over which encryption will occur ($\Gamma' = \bigcup_{s \in S'} H(s)$). The algorithm then selects a random exponent $t \in \mathbb{Z}_p$. The ciphertext is output as:

$$C = \left(C' = MY^t, C'' = g^t, \{C_i = T(i)^t : i \in \Gamma'\}\right)$$

, where $T(i)$ is defined as:

$$T(i) = g^{x^i} \prod_{j=1}^{n+1} t_j^{\Delta_{j,N}(i)}$$

where N is the set $\{1, \ldots, n+1\}$. Note that $T(i)$ is replaced by the random oracle construction (i.e., hash function) in this work for reasons of performance.

Decrypt(C, S', S, SK): Like the encryption algorithm, the decryption algorithm begins by computing sets of hashes for the ciphertext (Γ') and the client attempting to access the encrypted content (Γ). If $|\Gamma \bigcup \Gamma'| \geq k$, the algorithm possesses a sufficient number of attributes to decrypt the ciphertext. For each attribute i in the shared set of attributes U, where $|U| = k$, the algorithm computes a temporary value:

$$A_i = \frac{e(D_i, C'')}{e(d_i, C_i)} = \frac{e(g^{q(i)} T(i)^{r_i}, g^t)}{e(g^{r_i}, T(i)^t)} = e(g,g)^{tq(i)}.$$

This computation gives k shares of the polynomial $tq(i)$ in the exponent. Using polynomial interpolation [29], the algorithm recovers the blinding value $e(g,g)^{yt}$ and divides it out by computing:

$$M = C' / \left(A_i^{\Delta_{i,U}(0)}\right) = C' / e(g,g)^{tq(0)} = C' / e(g,g)^{ty} = M.$$

Because a new random polynomial is chosen for each private key, the system is secure against attempts to collude and pool the attributes of different adversaries.