2016 IEEE Symposium on Security and Privacy

On the Practicality of Cryptographically Enforcing Dynamic Access Control Policies in the Cloud

William C. Garrison III
University of Pittsburgh

Adam Shull
Indiana University

Steven Myers
Indiana University

Adam J. Lee
University of Pittsburgh

***Abstract*—The ability to enforce robust and dynamic access controls on cloud-hosted data while simultaneously ensuring confidentiality with respect to the cloud itself is a clear goal for many users and organizations. To this end, there has been much cryptographic research proposing the use of (hierarchical) identity-based encryption, attribute-based encryption, predicate encryption, functional encryption, and related technologies to perform robust and private access control on untrusted cloud providers. However, the vast majority of this work studies static models in which the access control policies being enforced do not change over time. This is contrary to the needs of most practical applications, which leverage dynamic data and/or policies.**

In this paper, we show that the cryptographic enforcement of dynamic access controls on untrusted platforms incurs computational costs that are likely prohibitive in practice. Specifically, we develop lightweight constructions for enforcing role-based access controls (i.e., RBAC_0) over cloud-hosted files using identity-based and traditional public-key cryptography. This is done under a threat model as close as possible to the one assumed in the cryptographic literature. We prove the correctness of these constructions, and leverage real-world RBAC datasets and recent techniques developed by the access control community to experimentally analyze, via simulation, their associated computational costs. This analysis shows that supporting revocation, file updates, and other state change functionality is likely to incur prohibitive overheads in even minimally-dynamic, realistic scenarios. We identify a number of bottlenecks in such systems, and fruitful areas for future work that will lead to more natural and efficient constructions for the cryptographic enforcement of dynamic access controls. Our findings naturally extend to the use of more expressive cryptographic primitives (e.g., HIBE or ABE) and richer access control models (e.g., RBAC_1 or ABAC).

I. Introduction

In recent years, numerous cryptographic schemes have been developed to support access control on the (untrusted) cloud. One of the most expressive of these is attribute-based encryption (ABE) [31], which is a natural fit for enforcing attribute-based access control (ABAC) policies [40]. However, the practical implications of using these types of cryptographic schemes to tackle realistic access control problems are largely unexplored. In particular, much of the literature concerns static scenarios in which data and/or access control policies are rarely, if ever, modified (e.g., [5], [30], [31], [42], [49], [52], [59]). Such scenarios are not representative of real-world systems, and oversimplify issues associated with key management and revocation that can carry substantial practical overheads. In this paper, we explore *exactly* these types of issues in an attempt to understand the computational overheads of using advanced cryptographic techniques to enforce dynamic access controls over objects stored on untrusted platforms. Our primary result is negative: we demonstrate that prohibitive computational burdens are likely to be incurred when supporting practical, dynamic workloads.

The push to develop and use cryptography to support adaptive access control on the cloud is natural. Major cloud providers such as Google, Microsoft, Apple, and Amazon are providing both large-scale, industrial services and smaller-scale, consumer services. Similarly, there are a number of user-focused cloud-based file sharing services, such as Dropbox, Box, and Flickr. However, the near-constant media coverage of data breaches has raised both consumer and enterprise concerns regarding the privacy and integrity of cloud-stored data. Among the widely-publicized stories of external hacking and data disclosure are releases of private photos [56]. Some are even state-sponsored attacks against cloud organizations themselves, such as Operation Aurora, in which Chinese hackers infiltrated providers like Google, Yahoo, and Rackspace [20], [51]. Despite the economic benefits and ease-of-use provided by outsourcing data management to the cloud, this practice raises new questions regarding the maintenance and enforcement of the access controls that users have come to expect from file sharing systems.

Although advanced cryptographic primitives seem well-suited for protecting *point states* in many access control paradigms, supporting the *transitions* between protection states that are triggered by administrative actions in a dynamic system requires addressing very subtle issues involving key management, coordination, and key/policy consistency. While there has been some work seeking to provide a level of dynamism for these types of advanced cryptographic primitives, this work is not without issues. For instance, techniques have been developed to support key revocation [8] and delegated re-encryption [32], [58]. Unfortunately, these techniques are not compatible with hybrid encryption—which is necessary from an efficiency perspective—under reasonable threat models.

In this paper, we attempt to tease out these types of critical details by exploring the cryptographic enforcement of a widely-deployed access control model: role-based access control (specifically, RBAC_0 [61]). In particular, we develop two constructions for cryptographically enforcing dynamic RBAC_0 policies in untrusted cloud environments: one based on standard public-key cryptographic techniques, and another based on identity-based encryption/signature (IBE/IBS) techniques [11], [13], [59]. By studying RBAC_0 in the context of these relatively

DOI 10.1109/SP.2016.54

efficient cryptographic schemes, we can effectively *lower-bound the costs* that would be associated with supporting richer access controls (e.g., ABAC) by using more advanced—and more expensive—cryptographic techniques exhibiting similar administrative and key delegation structures (e.g., ABE).

We use tools from the access control literature [36] to prove the correctness of our RBAC_0 constructions. To quantify the costs of using these constructions in realistic access control scenarios, we leverage a stochastic modeling and simulation-based approach developed to support access control suitability analysis [26]. Our simulations are driven by real-world RBAC datasets that allow us to explore—in a variety of environments where the RBAC_0 policy and files in the system are subject to dynamic change—the costs associated with using these constructions. In doing so, we uncover several design considerations that must be addressed, make explicit the complexities of managing the transitions that occur as policies or data are modified at runtime, and demonstrate the often excessive overheads of relying solely on advanced cryptographic techniques for enforcing dynamic access controls. This provides us with a number of insights toward the development of more effective cryptographic access controls. Through our analysis, we make the following contributions:

- We demonstrate that the cryptographic enforcement of role-based access controls on the cloud incurs overheads that are likely prohibitive in realistic dynamic workloads. For instance, we show that removing a single user from a role in a moderately-sized organization can require hundreds or thousands of IBE encryptions! Since our constructions are designed to lower-bound deployment costs (given current cryptographic techniques), this indicates that cryptographic access controls are likely to carry prohibitive costs for even mildly dynamic scenarios.
- Prior work often dismisses the need for an access control reference monitor when using cryptographically-enforced access controls (e.g., [5], [30], [31], [52]). We discuss the necessity of some minimal reference monitor on the cloud when supporting dynamic, cryptographically-enforced access controls, and we outline other design considerations that must be addressed in dynamic environments.
- We develop constructions that use either the IBE/IBS or public-key cryptographic paradigms to enable dynamic outsourced RBAC_0 access controls. In an effort to lower-bound deployment costs, our constructions exhibit design choices that emphasize efficiency over the strongest possible security (e.g., using lazy rather than online re-encryption, cf. Section IV-C), but are easily extended to support stronger security guarantees (albeit at additional costs). These constructions further highlight practical considerations that are often overlooked in the literature, or that prevent the application of techniques designed to enhance the dynamism of advanced cryptographic techniques.
- Having established the infeasibility of enforcing even the relatively simple RBAC_0 in dynamic scenarios, we discuss the increase in costs that would be associated with more expressive cryptographically-enforced access control such as hierarchical RBAC (RBAC_1) using HIBE [10], [29], or attribute-based access control (ABAC) using ABE.

The remainder of this paper is organized as follows. In Section II, we discuss relevant related work. Section III documents our system model and assumptions, and provides background on RBAC_0 and the cryptographic techniques used in this paper. In Section IV, we describe our IBE/IBS construction in detail, and overview the key differences between it and our PKI-based construction. Section V presents theorems stating the correctness of our constructions, as well as experimental results showing the overheads incurred by our constructions when applied to real-world RBAC datasets. In Section VI, we identify interesting directions for future work informed by our findings. Section VII details our conclusions.

II. Related Work

A. Access Control

Access control is one of the most fundamental aspects of computer security, with instances occurring pervasively throughout most computer systems: relational databases often provide built-in access control commands; network administrators implement access controls, e.g., firewall rules and router ACLs; operating systems provide access control primitives that enable users to protect their private files; and web applications and other frameworks typically implement purpose-specific access controls to control access to the information that they manage. The literature describes a diversity of access control systems supporting policies including basic access control lists [60], cryptographically-enforced capabilities [64], group- [43], role- [61], and attribute-based [40] controls. Despite this diversity, a central theme in most access control work is the reliance on a fully-trusted reference monitor to check compliance with the policy to be enforced prior to brokering access to protected resources. This dependency on a trusted reference monitor is problematic, however, when resources are stored on (potentially) untrusted infrastructure.

Distributed or decentralized approaches to access control have also been well studied in the literature and in practice. Work in the trust management space (e.g., [4], [7], [21], [45]) allows the specification of *declarative access control policies* for protecting resources, which are satisfied using *digital credentials* of various forms. For instance, a research portal may allow free access to publications, provided that the requester is a graduate student at an accredited university. This allows the portal to delegate trust: provided that a requestor can produce a proof-of-ownership for a “graduate student” attribute certificate issued by an accredited university, she will be permitted access. We note that these approaches need not rely on heavyweight certificate infrastructures; recent work has provided similar functionality using lightweight cryptographic bearer credentials [6]. Further, widely-deployed identity management solutions (e.g., OAuth [35]) can also be viewed as simplified trust management approaches that offload identity verification to a third party, receiving only a “token” attesting to a requestor's identity. In all cases, however, a trusted reference monitor is still required to validate that the presented credentials actually satisfy the policy protecting a resource.

In this paper, by contrast, we investigate the implications of using cryptography to enforce access controls on cloud-based storage infrastructure, where the provider is not trusted to view file contents.

B. Cryptography

We assume the reader is familiar with basic concepts from symmetric-key and public-key cryptography, and many references exist (e.g., [41]) discussing these topics. Starting with the development of practical identity-based encryption (IBE) schemes [11], there has been considerable work on the development of cryptographic systems that directly support a number of access control functionalities, with examples including hierarchical IBE [29], [37], attribute-based encryption [59], and functional encryption [57]. At a high level, these encryption schemes encrypt data to a policy, so that only those who have secret keys satisfying the policy can decrypt. What varies between these types of schemes is the expressiveness of the policies that are supported. With IBE and traditional public-key encryption, one can encrypt to a given target individual, and only that individual can decrypt. With attribute-based encryption, a ciphertext can be encrypted to a certain policy, and can be decrypted only by individuals whose secret keys satisfy that policy. With functional encryption, a certain function is embedded in the ciphertext, and when one "decrypts," one does not retrieve the underlying value, but rather a function of the encrypted value and the decryptor's secret key. One underlying motivation in all of the above work is the ability to enforce access controls on encrypted data.

Each cryptographic scheme has its own associated costs, but they can be broadly categorized as follows. Symmetric cryptography is orders of magnitude faster than traditional public-key encryption, and traditional public-key encryption is an order of magnitude faster than pairing-based cryptography, in which the pairing operation itself typically carries the largest cost.[1] The vast majority of IBE, IBS, HIBE and ABE schemes are pairing-based cryptographic schemes. IBE schemes use a small constant number of pairings in either encryption or decryption. In contrast, ABE schemes use a number of pairings that is a function of the policy being encoded, and thus, assuming minimally expressive access policies, have computational costs substantially greater than IBE.

Much of the work on these advanced cryptographic systems allows for data to be stored on the cloud, but it does not address the issue of revocation or dynamic modification of the access control structure being used to store data on the cloud. This can, of course, be done by downloading the data, decrypting it, and then re-encrypting under a new policy, but this is communication intensive, and potentially computationally intensive too. Further, for large files, clients making the changes in the access structure may not be able to support the entire file locally (e.g., smartphones). Therefore, there has been some work done in considering delegated encryption and revocation in these models (e.g., [8], [32], [33], [46], [53], [58], [62]).

C. Cryptographic Access Controls

There has been significant work on using cryptography as an access control mechanism, starting with seminal works such as that by Gudes [34]. This work describes how access controls can be enforced using cryptography, but does not address many practical issues such as key distribution and management, policy updates, and costs. Furthermore, as the work's motivation is a local file system, the access control system must be trusted with the keys (and trusted to delete them from memory as soon as possible). Work by Akl and Taylor [1] addresses some of the key management issues by proposing a *key assignment scheme*: a system for deriving keys in a hierarchical access control policy, rather than requiring users higher in the hierarchy to store many more keys than those lower in the hierarchy. Again, this work does not consider key distribution or policy updates. Later work in key hierarchies by Atallah et al. [3] proposes a method that allows policy updates, but in the case of revocation, all descendants of the affected node in the access hierarchy must be updated, and the cost of such an operation is not discussed. Continued work in key assignment schemes has improved upon the efficiency of policy updates; see [16] for a survey of such schemes that discusses tradeoffs such as how much private vs. public information must be stored and how much information must be changed for policy updates. Much of this work focuses on the use of symmetric-key cryptography, and so its use for the cloud is potentially limited.

De Capitani di Vimercati et al. [18], [19] describe a method for cryptographic access controls on outsourced data using double encryption (one layer by the administrator and one by the service). An extension to this work also enforces write privileges [17]. However, this solution requires a high degree of participation by the cloud provider or third party, and the work does not address the high cost of such operations as deleting users (which can incur cascading updates). Ibraimi's thesis [38] proposes methods for outsourcing data storage using asymmetric encryption. However, the proposed method for supporting revocation requires a trusted mediator and keyshare escrow to verify all reads against a revocation list (and does not address revoked users reusing cached keyshares). Furthermore, policy updates require an active entity to re-encrypt all affected files under the new policy. Similarly, work by Nali et al. [50] enforces RBAC using public-key cryptography, but requires a series of active security mediators.

Crampton has shown that cryptography is sufficient to enforce RBAC policies [14] and general interval-based access control policies [15], but revocation and policy updates are not considered (i.e., the constructions are shown only for static policies). Ferrara et al. [24] formally define a cryptographic game for proving the security of cryptographically-enforced RBAC systems and prove that such properties can be satisfied

[1] We will exclude lattice-based systems, due to the difficulty in determining appropriate security parameters. This, amongst other factors, makes such generic comparisons difficult.

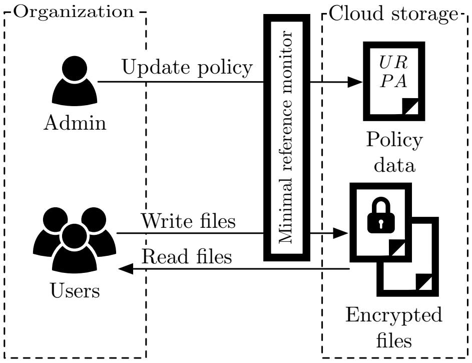


Fig. 1: Diagram of a cloud storage system

using an ABE-based construction. This construction has since been extended to provide policy privacy and support writes with less trust on the provider [23]. The latter is accomplished by eliminating the reference monitor that checks if a write is allowed and instead accepting each write as a new version; versions must then be verified when downloaded for reading to determine the most recent permitted version (the provider is trusted to provide an accurate version ordering). However, these works do not consider the costs and other practical considerations for using such a system in practice (e.g., lazy vs. active re-encryption, hybrid encryption). In this paper, we consider exactly these types of issues.

Pirretti et al. [55] have shown that distributed file systems and social networks can use ABE-based constructions to perform practical access control, but they leave dynamic revocation as future work.

III. Background and Assumptions

Our goal is to understand the practical costs of leveraging public-key cryptographic primitives to implement outsourced dynamic access controls in the cloud. In this section, we (i) define the system and threat models in which we consider this problem, (ii) specify the access control model that we propose to enforce, and (iii) define the classes of cryptographic primitives that will be used in our constructions.

A. System and Threat Models

The environment that we consider—which is based on the untrusted cloud provider typically assumed in the cryptographic literature—is depicted in Fig. 1. The system consists of three main (classes of) entities: *access control administrators*, *users/clients*, and *cloud storage providers*. In particular, we consider a model in which a single storage provider is contracted by an organization. This is analogous to companies contracting with providers like Microsoft (via OneDrive for Business) or Dropbox (via Dropbox Business) to outsource enterprise storage, or individuals making use of cloud platforms like Apple iCloud or Google Drive for hosting and sharing personal media. Further, this simplifies the overall system design by eliminating the need for a secondary mechanism that synchronizes cryptographic material and other metadata.

Assumptions. The *cloud storage provider* is contracted to manage the storage needs of a (perhaps virtual) organization. This includes storing the files hosted in the cloud, as well as any metadata associated with the access control policies protecting these files. We assume that the cloud is *not* trusted to view the contents of the files that it stores. However, it is trusted to ensure the availability of these files, and to ensure that only authorized individuals update these files. File access is assumed to occur directly though the cloud provider's API, with read access permissions being enforced cryptographically on the client side, and write access permissions being enforced by a minimal reference monitor on the cloud provider that validates client signatures that prove write privileges prior to file updates.[2] In short, the storage provider ensures file system consistency by preventing unauthorized updates, yet cannot read or make legitimate modifications to files or metadata.

Access control administrators are tasked with managing the protection state of the storage system. That is, they control the assignment of access permissions, which entails the creation, revocation, and distribution of cryptographic keys used to protect files in a role-based manner. Metadata to facilitate key distribution is stored in a cryptographically-protected manner on the cloud provider. *Users* may download any file stored on the storage provider, but may decrypt, read, and (possibly) modify only the files for which they have been issued the appropriate (role-based) keys. All files are encrypted and signed prior to being uploaded to the cloud storage provider. Finally, we assume that all parties can communicate via pairwise-authenticated and private channels (e.g., SSL/TLS tunnels).

Implications. To simplify presentation and analysis, the above threat model does leave some degree of trust in the cloud provider (albeit far less than is routinely placed in these providers today). In particular, the cloud provider is trusted to verify digital signatures prior to authorizing write operations. This could be avoided by using a versioning file system, allowing all writes, and relying on clients to find the most recent version of a file that has a valid signature prior to accessing that file. Similarly, it is possible—although prohibited by our threat model—for a malicious provider to "roll back" the filesystem to a prior state by replacing current files and metadata with previous versions. We note that it is possible to detect (e.g., via comparison with off-cloud metadata) or prevent (e.g., by splitting metadata and file storage across multiple providers) this issue, and thus this prohibition could be dropped. Further, we do not consider the denial-of-service threat of a user overwhelming the storage provider with spurious file downloads; in practice, this is easily addressed by using unguessable (perhaps cryptographically-produced) file names, or lightweight authorization tokens. However, all of these types of relaxations come with additional complexity. As we will demonstrate, the costs associated with cryptographic enforcement of dynamic access controls are likely prohibitive, *even under the above threat model.* This, effectively, lower-

[2] Note that this eliminates the possibility of a purely symmetric-key approach: the ability to validate, e.g., symmetric-key MACs would also allow the cloud provider to *modify* these MACs.

bounds the costs entailed by weaker threat models (which require more complex mechanisms). For the bulk of this paper, we will therefore focus on the above threat model, leaving discussion of further relaxations to Section VI.

B. Access Control Model

In this paper, we focus on cryptographic enforcement of a role-based access control (RBAC) system, given the prevalence of this type of access control system in both the research literature and commercial systems. RBAC systems simplify permission management through the use of abstraction: roles describe the access permissions associated with a particular (class of) job function, users are assigned to the set of roles entailed by their job responsibilities, and a user is granted access to an object if they are assigned to a role that is permitted to access that object. In this paper, we will investigate cryptographic implementations of the simplest RBAC formulation: RBAC$_0$ [61]. More formally, the state of an RBAC$_0$ system can be described as follows:

- U is a set of users,
- R is a set of roles,
- P is a set of permissions (e.g., $\langle file, op \rangle$),
- $PA \subseteq R \times P$ is the permission assignment relation, and
- $UR \subseteq U \times R$ is the user assignment relation.

The authorization predicate $auth : U \times P \rightarrow \mathbb{B}$ determines whether user u can use permission p and is defined as follows:

$$auth(u,p) = \exists r : [(u,r) \in UR] \wedge [(r,p) \in PA]$$

Many variants of RBAC exist, but we focus on the use of RBAC$_0$ as it is conceptually the simplest of these variants yet still provides adequate expressive power to be interesting for realistic applications. Generalizing this model to richer RBAC variants (e.g., RBAC$_1$) and attribute-based access control (ABAC) is discussed in Section VI-C.

C. Cryptographic Primitives

Both of our constructions make use of symmetric-key authenticated encryption ($\mathbf{Gen^{Sym}}$, $\mathbf{Enc^{Sym}}$, $\mathbf{Dec^{Sym}}$). Our PKI scheme uses public-key encryption and digital signatures ($\mathbf{Gen^{Pub}}$, $\mathbf{Enc^{Pub}}$, $\mathbf{Dec^{Pub}}$, $\mathbf{Gen^{Sig}}$, $\mathbf{Sign^{Sig}}$, $\mathbf{Ver^{Sig}}$). While many attribute-based encryption (ABE) schemes are being developed to support policy constructions of varying expressivity, RBAC$_0$ does not require this level of sophistication. To this end, we instead use identity-based encryption (IBE):

- $\mathbf{MSKGen^{IBE}}(1^n)$: Takes security parameter n; generates public parameters (which are implicit parameters to every other IBE algorithm) and master secret key msk.
- $\mathbf{KeyGen^{IBE}}(ID, msk)$: Generates a decryption key k_{ID} for identity ID.
- $\mathbf{Enc}^{\mathbf{IBE}}_{ID}(M)$: Encrypts message M under identity ID.
- $\mathbf{Dec}^{\mathbf{IBE}}_{k_{ID}}(C)$: Decrypts ciphertext C using key k_{ID}; correctness requires that $\forall\, ID$ if $k_{ID} = \mathbf{KeyGen^{IBE}}(ID)$ then $\forall\, M, \mathbf{Dec}^{\mathbf{IBE}}_{k_{ID}}(\mathbf{Enc}^{\mathbf{IBE}}_{ID}(M)) = M$.

We also use identity-based signature (IBS) schemes:

- $\mathbf{MSKGen^{IBS}}(1^n)$: Takes security parameter n; generates public parameters (which are implicit parameters to every other IBS algorithm) and master secret key msk.
- $\mathbf{KeyGen^{IBS}}(ID, msk)$: Generates a signing key s_{ID} for identity ID.
- $\mathbf{Sign}^{\mathbf{IBS}}_{ID,s_{ID}}(M)$: Generates a signature sig on message M if s_{ID} is a valid signing key for ID.
- $\mathbf{Ver}^{\mathbf{IBS}}_{ID}(M, sig)$: Verifies whether sig is a valid signature on message M for identity ID; requires that $\forall\, ID$ if $s_{ID} = \mathbf{KeyGen^{IBS}}(ID)$ then $\forall\, M, \mathbf{Ver}^{\mathbf{IBS}}_{ID}(M, \mathbf{Sign}^{\mathbf{IBS}}_{ID,s_{ID}}(M)) = 1$.

IBE (resp. IBS) schemes build upon traditional public-key schemes by allowing any desired string to act as one's encryption (resp. verification) key. This requires the introduction of a third party who can generate the decryption and signing keys corresponding to these identity strings. This third party, who holds the master keys, is able to produce decryption or signing keys for anyone, and thus the system has inbuilt escrow. In our use of these systems, the RBAC administrator(s) will act as this third party. Since administrators traditionally have the power to access/assign arbitrary permissions, this escrow is not a weakness. In practice, if this is still a concern, threshold/secret splitting schemes can be used to distribute trust amongst several individuals. However, such schemes would increase the cryptographic costs of operations associated with the master key.

IV. Construction

While cryptographic access control enforcement has been studied in the past, the focus has been almost entirely on techniques that are best suited for *mostly static* scenarios lacking a trusted reference monitor (e.g., [31], [49]), in which the policies to be enforced and files to be protected change very little over time. As such, the particulars associated with *securely* managing policy change and the associated overheads have been largely under-explored. In this section, we begin with a strawman construction for cryptographic access control enforcement, and use it to highlight a variety of limitations and design considerations that must be addressed. We conclude with a detailed description of our IBE/IBS and PKI constructions for RBAC$_0$, which address these issues.

A. A Strawman Construction

At first blush, it seems conceptually simple to provision a cryptographically-enforced RBAC$_0$ system. We now overview such a system, which will allow us to highlight a variety of issues that arise as a result. This strawman construction will make use of IBE/IBS; the use of a more traditional PKI is a straightforward translation. We assume that the administrator holds the master secret keys for the IBE/IBS systems.

- **Registration.** Each user, u, of the system must carry out an initial registration process with the administrator. The result of this process is that the user will obtain identity-based encryption and signing keys $k_u \leftarrow \mathbf{KeyGen^{IBE}}(u)$ and $s_u \leftarrow \mathbf{KeyGen^{IBS}}(u)$ from the administrator.

- **Role Administration.** For each role, r, the administrator will generate identity-based encryption and signing keys $k_r \leftarrow \mathbf{KeyGen}^{\mathbf{IBE}}(r)$ and $s_r \leftarrow \mathbf{KeyGen}^{\mathbf{IBS}}(r)$. For each user u that is a member of r (i.e., for each $(u, r) \in UR$ in the RBAC_0 state), the administrator will create and upload a tuple of the form:

$$\langle \mathsf{RK}, u, r, \mathbf{Enc}_u^{\mathbf{IBE}}(k_r, s_r), \mathbf{Sign}_{SU}^{\mathbf{IBS}} \rangle.$$

This tuple provides u with cryptographically-protected access to the encryption and signing keys for r, and is signed by the administrator. Here, $\mathbf{Sign}_{SU}^{\mathbf{IBS}}$ at the end of the tuple represents an IBS signature by identity SU (the administrator), and RK is a sentinel value indicating that this is a role key tuple.

- **File Administration.** For each file f to be shared with a role r (i.e., for each $(r, \langle f, op\rangle) \in PA$ in the RBAC_0 state), the administrator will create and upload a tuple:

$$\langle \mathsf{F}, r, \langle fn, op\rangle, \mathbf{Enc}_r^{\mathbf{IBE}}(f), SU, \mathbf{Sign}_{SU}^{\mathbf{IBS}} \rangle.$$

This tuple contains a copy of f that is encrypted to members of r. Here, fn represents the name of the file f, while op is the permitted operation—either Read or Write. As before, $\mathbf{Sign}_{SU}^{\mathbf{IBS}}$ is a signature by the administrator, and F is a sentinel value indicating that this is a file tuple.

- **File Access.** If a user u who is authorized to read a file f (i.e., $\exists r : (u, r) \in UR \wedge (r, \langle f, \mathsf{Read}\rangle) \in PA$) wishes to do so, she must (i) download an RK tuple for the role r and an F tuple for f; (ii) validate the signatures on both tuples; (iii) decrypt the role key k_r from the RK tuple using their personal IBE key k_u; and (iv) decrypt the file f from the F tuple using the role key k_r.

Writes to a file are handled similarly. If u is authorized to write a file f via membership in role r (i.e., $\exists r : (u, r) \in UR \wedge (r, \langle f, \mathsf{Write}\rangle) \in PA$), she can upload a new F tuple $\langle \mathsf{F}, r, \langle fn, \mathsf{Write}\rangle, \mathbf{Enc}_r^{\mathbf{IBE}}(f'), \mathbf{Sign}_r^{\mathbf{IBS}} \rangle$. If the signature authorizing the write ($\mathbf{Sign}_r^{\mathbf{IBS}}$) can be verified by the cloud provider, the existing F tuple for f will be replaced.

This construction describes a cryptographic analog to RBAC_0. The UR relation is encoded in the collection of RK tuples, while the PA relation is encoded in the collection of F tuples. The authorization relation of RBAC_0 is upheld cryptographically: to read a file f, a user u must be able to decrypt a tuple granting her the permissions associated with a role r, which can be used to decrypt a tuple containing a copy of f encrypted to role r.

B. Design Considerations

While conceptually straightforward, the strawman construction is by no means a complete solution. We now use this construction as a guide to discuss a number of design tradeoffs that must be addressed to support cryptographic enforcement of dynamic RBAC_0 states.

PKI vs. IBE. Basing an RBAC_0 system on IBE and IBS allows for a simple mapping from encryption keys to roles in RBAC_0: The name of the role is the public-key used to encrypt under that role. This is conceptually simpler than what is achieved by traditional public key or symmetric encryption, which may help limit certain key management issues in software. IBE-based constructions also generalize to richer access control models (e.g., enforced using HIBE or ABE), which we explore in Section VI. That said, rich infrastructure has been developed to support public key cryptography, which may make the systems support issues inherent in these constructions easier to manage. To this end, we present constructions based on both IBE and public key cryptography.

Inefficiency Concerns. The strawman construction exhibits two key issues with respect to efficiency. First, IBE (like public-key cryptography) is not particularly well-suited for the bulk encryption of large amounts of data. As such, the performance of this construction would suffer when large files are shared within the system. Second, this construction requires a duplication of effort when a file, say f, is to be shared with multiple roles, say r_1 and r_2. That is, f must actually be encrypted twice: once with r_1 and once with r_2. We note that this also leads to consistency issues between roles when f is updated. Fortunately, both of these concerns can be mitigated via the use of hybrid cryptography. Rather than storing F tuples of the form:

$$\langle \mathsf{F}, r, \langle fn, op\rangle, \mathbf{Enc}_r^{\mathbf{IBE}}(f), SU, \mathbf{Sign}_{SU}^{\mathbf{IBS}} \rangle$$

We can instead store the following tuples, where $k \leftarrow \mathbf{Gen}^{\mathbf{Sym}}$ is a symmetric key:

$$\langle \mathsf{FK}, r, \langle fn, op\rangle, \mathbf{Enc}_r^{\mathbf{IBE}}(k), SU, \mathbf{Sign}_{SU}^{\mathbf{IBS}} \rangle$$
$$\langle \mathsf{F}, fn, \mathbf{Enc}_k^{\mathbf{Sym}}(f), r, \mathbf{Sign}_r^{\mathbf{IBS}} \rangle$$

The FK tuples are similar to the file encryption tuples in the strawman construction, except that the ciphertext portion of the tuple now includes an IBE-encrypted symmetric key rather than an IBE-encrypted file. F tuples contain a symmetric-key-encrypted (using an authenticated mode) version of the file f, and are IBS-signed using the role key of the last authorized updater. This adjustment to the metadata improves the efficiency of bulk encryption by using symmetric-key cryptography, and greatly reduces the duplication of effort when sharing a file with multiple roles: a single F tuple can be created for the file along with multiple FK tuples (i.e., one per role).

Handling Revocation. The strawman construction can neither revoke a permission from a role, nor remove a user from a role. The former case can be handled by versioning the F and FK tuples stored within the system, and the latter case handled by adding role versioning to the role key tuples and FK tuples in the system:

$$\langle \mathsf{RK}, u, (r, v_r), \mathbf{Enc}_u^{\mathbf{IBE}}(k_{(r,v_r)}, s_{(r,v_r)}), \mathbf{Sign}_{SU}^{\mathbf{IBS}} \rangle$$
$$\langle \mathsf{FK}, r, \langle fn, op\rangle, v, \mathbf{Enc}_{(r,v_r)}^{\mathbf{IBE}}(k), SU, \mathbf{Sign}_{SU}^{\mathbf{IBS}} \rangle$$
$$\langle \mathsf{F}, fn, v, \mathbf{Enc}_k^{\mathbf{Sym}}(f), (r, v_r), \mathbf{Sign}_{(r,v_r)}^{\mathbf{IBS}} \rangle$$

Here, v represents a version number for the symmetric key used to encrypt a file. Role names have been replaced with tuples that include the role name (e.g., r), *as well as a version number (v_r).* Removing a permission from a role entails re-keying and re-encrypting the file (i.e., creating a new F tuple), and creating new FK tuples for each role whose access to the file has *not* been revoked. The roles increment their previous role number. Similarly, removing a user u from a role r entails deleting u's RK tuple for r, generating new role keys for r (with an incremented version number) and encoding these into new RK tuples for each user remaining in r, and re-versioning all files to which the role r holds some permission. We note that both of these processes must be carried out by an administrator, as only administrators can modify the RBAC_0 state. There is much nuance to these processes, and we defer a full discussion to Section IV-C.

Online, Lazy, and Proxy Re-Encryption. Supporting revocation leads to an interesting design choice: should files be re-encrypted immediately upon re-key, or lazily re-encrypted upon their next write? From a confidentiality standpoint, forcing an administrator—or some daemon process running on her behalf—to re-encrypt files immediately upon re-key is preferential, as it ensures that users who have lost the ability to access a file cannot later read its contents. On the other hand, this comes with a potentially severe efficiency penalty in the event that many files are re-keyed due to changes to some role, as access to these files must be locked while they are downloaded, re-encrypted, and uploaded. In this paper, we opt for a *lazy re-encryption* strategy, in which files are re-encrypted by the next user to write to the file (cf., Section IV-C). We note that such a scheme is not appropriate for all scenarios, but substantially reduces the computational burden on the cloud when allowing for dynamic updates to the RBAC_0 state (cf., Section V-D). Similarly, if a client is powerful enough to download a source file and decrypt it to view the material, it presumably is powerful enough to perform the roughly computationally equivalent operation of re-encrypting it. Note that a single client is unlikely to need to re-encrypt large numbers of files, unlike the cloud if a lazy re-encryption strategy were not used. Adapting our construction to instead use online re-encryption is a straightforward extension.

While appealing on the surface, IBE schemes that support proxy re-encryption, or revocation (e.g., [8], [32]) are not suitable for use in our scenario. These types of schemes would seemingly allow us to remove our reliance on lazy re-encryption, and have the cloud locally update encryptions when a permission is revoked from a role, or a role from a user. This would be done by creating an updated role name, using proxy re-encryption to move the file from the old role name to the updated one, and then revoking all keys for the old file. The *significant* issue, here, is that such schemes do not address how one would use them with hybrid encryption. We do not believe that a reasonable threat model can assume that even a limited adversary would be unable to cache all the symmetric keys for files she has access to. *Thus, using proxy re-encryption on the* RK *and* FK *tuples and not the* F *tuples would allow users to continue to access files to which their access has been revoked, and so our construction would still require online or lazy re-encryption of the files themselves.*

As a final note, we acknowledge that key-homomorphic PRFs [12] could be combined with revocation and proxy re-encryption schemes, solving the revocation problem completely on the cloud in the hybrid model. However, current technology does not solve the computational effort, as costs of current key-homomorphic PRFs are comparable or greater than the IBE and PKI technologies in consideration.

Multiple Levels of Encryption. We note that our construction has levels of indirection between RK, FK, and F tuples that mirror the indirection between users, roles, and permissions in RBAC_0. This indirection could be flattened to decrease the number of cryptographic operations on the critical path to file access; this would be akin to using an access matrix to encode RBAC_0 states. While this is possible, it has been shown to cause computational inefficiencies when roles' memberships or permissions are altered [27]; in our case this inefficiency would be amplified due to the cryptographic costs associated with these updates.

Other Issues and Considerations. Our constructions are measured without concern for concurrency-related issues that would need to be addressed in practice. We note, however, that features to handle concurrency would be largely independent of the proposed cryptography used to enforce the RBAC_0 policies. As such, we opt for the analysis of the conceptually-simpler schemes presented in this paper. Finally, our analysis is agnostic to the underlying achieved security guarantees and hardness assumptions of the public-key and IBE schemes. Production implementations would need to consider these issues.

C. Detailed IBE/IBS Construction

We now flesh out the strawman and previously-discussed enhancements. This produces a full construction for enforcing RBAC_0 protections over an evolving collection managed by a minimally-trusted cloud storage provider.

1) Overview and Preliminaries: We reiterate that the administrators act as the Master Secret Key Generator of the IBE/IBS schemes. Users add files to the system by IBE-encrypting these files to the administrators, using hybrid cryptography and F tuples. Administrators assign permissions (i.e., $\langle file, op\rangle$ pairs) to roles by distributing symmetric keys using FK tuples. Role keys are distributed to users using RK tuples. Recall the format of these tuples is as follows:

$$\langle \mathsf{RK}, u, (r, v_r), \mathbf{Enc}^{\mathbf{IBE}}_{u}(k_{(r,v_r)}, s_{(r,v_r)}), \mathbf{Sign}^{\mathbf{IBS}}_{SU}\rangle$$
$$\langle \mathsf{FK}, r, \langle fn, op\rangle, v, \mathbf{Enc}^{\mathbf{IBE}}_{(r,v_r)}(k), SU, \mathbf{Sign}^{\mathbf{IBS}}_{SU}\rangle$$
$$\langle \mathsf{F}, fn, v, \mathbf{Enc}^{\mathbf{Sym}}_{k}(f), (r, v_r), \mathbf{Sign}^{\mathbf{IBS}}_{(r,v_r)}\rangle$$

Note that symmetric keys and role keys are associated with version information to handle the cases where a user is removed from a role or a permission is revoked from a role.

We assume that files have both read and write permissions associated with them. However, we cannot have write without read, since writing requires decrypting the file's symmetric key, which then can be used to decrypt and read the stored file. Thus we only assign either Read or RW, and only revoke Write (Read is retained) or RW (nothing is retained). When a user wishes to access a file, she determines which of her roles has access to the permission in question. She then decrypts the role's secret key using her identity, and then decrypts the symmetric key for the file using the role's secret key, and finally uses the symmetric key to decrypt the symmetrically-encrypted ciphertext in question.

2) Full Construction: Figure 2 lists every RBAC_0 operation and shows how each can be implemented using IBE, IBS, and the metadata structures described previously. This figure uses the following notation: u is a user, r and q are roles, p is a permission, fn is a file name, f is a file, c is a ciphertext (either IBE or symmetric), sig is an IBS signature, and v is a version number. Users are listed in a file called USERS. The identity corresponding to a role r is (r, v), where v is a positive integer representing the version number. We use v_r to denote the latest version number for role r. Roles and versions are stored as (r, v_r) pairs in a file called ROLES, which is publicly viewable and can only be changed by the administrator. Similarly, we use v_{fn} to denote the latest version number for the file with name fn. Filenames and versions are stored as (fn, v_{fn}) pairs in a file called FILES, which is publicly viewable and can only be changed by the admin or reference monitor (R.M.). SU is the superuser identity possessed by the administrators. We use "$-$" to represent a wildcard. $\mathbf{Sign}^{\mathbf{IBS}}_{id}$ at the end of a tuple represents an IBS signature by identity id over the rest of the tuple. The subscript after an operation name identifies who performs the operation if it is not performed by an administrator.

Many operations described in Fig. 2 are straightforward given the discussion earlier in this section. To demonstrate some of the more complicated aspects of this construction, we now describe the procedure to revoke a role from a user, which demonstrates several types of re-keys as well as our notion of lazy re-encryption. The procedure for removing a user u from a role r consists of three steps: (i) re-keying r, (ii) re-encrypting existing file keys stored in FK tuples to the new role key, and (iii) re-keying all files accessible by r.

To re-key a role r, we must transition from (r, v_r) to $(r, v_r + 1)$, generating new IBE keys for this new role version. The old RK tuples for r are deleted, and each remaining member u' of role r is given the new RK tuples of the form of $\langle \mathsf{RK}, u', (r, v_r + 1), c, \mathbf{Sign}^{\mathbf{IBS}}_{SU} \rangle$, where c contains the new IBE/IBS keys encrypted to u''s identity key. Next, all (symmetric) file keys encrypted to (r, v_r) in FK tuples are replaced with file keys encrypted to $(r, v_r + 1)$. This allows the remaining members of r to retain access to existing files, while preventing the revoked user u from accessing any file keys that he has not already decrypted and cached.

Finally, each file to which r has access must be re-keyed to prevent u from accessing future updates to this file *using cached symmetric keys*. For each file f, a new symmetric key is generated via $\mathbf{Gen}^{\mathbf{Sym}}$. This key is then encrypted for each role r' that has access to f (including r), and new FK tuples $\langle \mathsf{FK}, r', \langle f, op \rangle, v + 1, c', \mathbf{Sign}^{\mathbf{IBS}}_{SU} \rangle$ are uploaded *alongside* existing $\langle \mathsf{FK}, r', \langle f, op \rangle, v, c, \mathbf{Sign}^{\mathbf{IBS}}_{SU} \rangle$ tuples. Here, $v + 1$ is the new file key version, c is the existing encrypted file key, and c' is the new file key IBE-encrypted to identity r'. The next time f is read, the key contained in c will be used for decryption; the next time f is written, the key contained in c' will be used for encryption. This process obviates the need for a daemon to re-encrypt all files at revocation time, but prevents the revoked user u from accessing any future modifications to these files using cached symmetric file keys.

D. PKI Construction Overview

We now provide an intuition for how traditional public-key cryptography can be used in place of IBE/IBS to implement RBAC_0. Due to space limitations, full details will be left to a technical report version [28] of this paper. In our PKI construction, public-key encryption and signatures take the place of IBE and IBS. Each role is assigned a public/private key pair rather than IBE/IBS keys. The primary difference between the IBE and PKI constructions is that IBE/IBS clients are given *escrowed* IBE/IBS identity private keys by the role administrator, while PKI clients generate their own public/private key pairs and upload their public keys. Note that in both systems, the administrators have access to all of the roles' private keys.

V. Analysis

We now describe our evaluation of the *suitability* of IBE/IBS and PKI constructions for enforcing RBAC_0 access controls. We utilize a workflow similar to that proposed in [26], in which we first evaluate the candidates' *expressive power* (i.e., ability to represent the desired policy as it evolves), then evaluate the *cost* of using each candidate using Monte Carlo simulation based on initial states obtained from real-world datasets.

A. Qualitative Analysis

We analyze the correctness and security guarantees of our implementations using the access control expressiveness framework known as *parameterized expressiveness* [36]. In particular, we ensure that the implementation properties of correctness, AC-preservation, and safety are preserved by these constructions. *Correctness* ensures that the RBAC_0 state's image in our constructions answers queries exactly as the original RBAC_0 system would, and that the same end state is reached by either executing an RBAC_0 action natively and mapping the result into our construction or by mapping the initial RBAC_0 state and executing the action's image in our construction. *AC-preservation* says that the RBAC_0 system's authorization requests must be asked directly in the simulating system. For instance, the policy must be simulated in such a way that the RBAC_0 request "Can subject s read file f?" is asked directly in the simulated state rather than being translated to any other queries. Finally, *safety* ensures that our constructions do not grant or revoke unnecessary permissions during the simulation of a single RBAC_0 command. That is, the intermediate states through which our constructions travel while implementing an RBAC_0 command do not add or remove any granted requests except those that must be added or removed as determined by the start and end states of the RBAC_0 command.

```
$addU(u)$
- Add $u$ to USERS
- Generate IBE private key $k_u \leftarrow \mathbf{KeyGen}^{\mathbf{IBE}}(u)$ and IBS private key $s_u \leftarrow \mathbf{KeyGen}^{\mathbf{IBS}}(u)$ for the new user $u$
- Give $k_u$ and $s_u$ to $u$ over private and authenticated channel

$delU(u)$
- For every role $r$ that $u$ is a member of:
  * $revokeU(u, r)$

$addP_u(fn, f)$
- Generate symmetric key $k \leftarrow \mathbf{Gen}^{\mathbf{Sym}}$
- Send $\langle \mathsf{F}, fn, 1, \mathbf{Enc}^{\mathbf{Sym}}_k(f), u, \mathbf{Sign}^{\mathbf{IBS}}_u \rangle$ and $\langle \mathsf{FK}$, $SU$, $\langle fn, \mathsf{RW} \rangle$, 1, $\mathbf{Enc}^{\mathbf{IBE}}_{SU}(k)$, $u$, $\mathbf{Sign}^{\mathbf{IBS}}_u \rangle$ to R.M.
- The R.M. receives $\langle \mathsf{F}, fn, 1, c, u, sig \rangle$ and $\langle \mathsf{FK}$, $SU$, $\langle fn, \mathsf{RW} \rangle$, 1, $c'$, $u$, $sig' \rangle$ and verifies that the tuples are well-formed and the signatures are valid, i.e., $\mathbf{Ver}^{\mathbf{IBS}}_u(\langle \mathsf{F}, fn, 1, c, u \rangle, sig) = 1$ and $\mathbf{Ver}^{\mathbf{IBS}}_u(\langle \mathsf{FK}$, $SU$, $\langle fn, \mathsf{RW} \rangle$, 1, $c'$, $u \rangle$, $sig') = 1$.
- If verification is successful, the R.M. adds $(fn, 1)$ to FILES and stores $\langle \mathsf{F}, fn, 1, c, u, sig \rangle$ and $\langle \mathsf{FK}$, $SU$, $\langle fn, \mathsf{RW} \rangle$, 1, $c'$, $u$, $sig' \rangle$

$delP(fn)$
- Remove $(fn, v_{fn})$ from FILES
- Delete $\langle \mathsf{F}, fn, -, -, -, - \rangle$ and all $\langle \mathsf{FK}, -, \langle fn, - \rangle, -, -, -, - \rangle$

$addR(r)$
- Add $(r, 1)$ to ROLES
- Generate IBE private key $k_{(r,1)} \leftarrow \mathbf{KeyGen}^{\mathbf{IBE}}((r, 1))$ and IBS private key $s_{(r,1)} \leftarrow \mathbf{KeyGen}^{\mathbf{IBS}}((r, 1))$ for role $(r, 1)$
- Send $\langle \mathsf{RK}$, $SU$, $(r, 1)$, $\mathbf{Enc}^{\mathbf{IBE}}_{SU}\left(k_{(r,1)}, s_{(r,1)}\right)$, $\mathbf{Sign}^{\mathbf{IBS}}_{SU} \rangle$ to R.M.

$delR(r)$
- Remove $(r, v_r)$ from ROLES
- Delete all $\langle \mathsf{RK}, -, (r, v_r), -, - \rangle$
- For all permissions $p = \langle fn, op \rangle$ that $r$ has access to:
  * $revokeP(r, \langle fn, \mathsf{RW} \rangle)$

$assignU(u, r)$
- Find $\langle \mathsf{RK}$, $SU$, $(r, v_r)$, $c$, $sig \rangle$ with $\mathbf{Ver}^{\mathbf{IBS}}_{SU}(\langle \mathsf{RK}$, $SU$, $(r, v_r)$, $c \rangle$, $sig) = 1$
- Decrypt keys $(k_{(r,v_r)}, s_{(r,v_r)}) = \mathbf{Dec}^{\mathbf{IBE}}_{k_{SU}}(c)$
- Send $\langle \mathsf{RK}$, $u$, $(r, v_r)$, $\mathbf{Enc}^{\mathbf{IBE}}_u\left(k_{(r,v_r)}, s_{(r,v_r)}\right)$, $\mathbf{Sign}^{\mathbf{IBS}}_{SU} \rangle$ to R.M.

$revokeU(u, r)$
- Generate new role keys $k_{(r,v_r+1)} \leftarrow \mathbf{KeyGen}^{\mathbf{IBE}}((r, v_r + 1))$, $s_{(r,v_r+1)} \leftarrow \mathbf{KeyGen}^{\mathbf{IBS}}((r, v_r + 1))$
- For all $\langle \mathsf{RK}$, $u'$, $(r, v_r)$, $c$, $sig \rangle$ with $u' \neq u$ and $\mathbf{Ver}^{\mathbf{IBS}}_{SU}(\langle \mathsf{RK}$, $u'$, $(r, v_r)$, $c \rangle$, $sig) = 1$:
  * Send $\langle \mathsf{RK}$, $u'$, $(r, v_r + 1)$, $\mathbf{Enc}^{\mathbf{IBE}}_{u'}\left(k_{(r,v_r+1)}, s_{(r,v_r+1)}\right)$, $\mathbf{Sign}^{\mathbf{IBS}}_{SU} \rangle$ to R.M.
- For every $fn$ such that there exists $\langle \mathsf{FK}$, $(r, v_r)$, $\langle fn, op \rangle$, $v_{fn}$, $c$, $SU$, $sig \rangle$ with $\mathbf{Ver}^{\mathbf{IBS}}_{SU}(\langle \mathsf{FK}$, $(r, v_r)$, $p$, $v_{fn}$, $c$, $SU \rangle$, $sig) = 1$:
  * For every $\langle \mathsf{FK}$, $(r, v_r)$, $\langle fn, op' \rangle$, $v$, $c'$, $SU$, $sig \rangle$ with $\mathbf{Ver}^{\mathbf{IBS}}_{SU}(\langle \mathsf{FK}$, $(r, v_r)$, $\langle fn, op' \rangle$, $v$, $c'$, $SU \rangle$, $sig) = 1$:
    · Decrypt key $k = \mathbf{Dec}^{\mathbf{IBE}}_{k_{(r,v_r)}}(c')$
    · Send $\langle \mathsf{FK}$, $(r, v_r + 1)$, $\langle fn, op' \rangle$, $v$, $\mathbf{Enc}^{\mathbf{IBE}}_{(r,v_r+1)}(k)$, $SU$, $\mathbf{Sign}^{\mathbf{IBS}}_{SU} \rangle$ to R.M.
  * Generate new symmetric key $k' \leftarrow \mathbf{Gen}^{\mathbf{Sym}}$ for $p$
  * For all $\langle \mathsf{FK}$, $id$, $\langle fn, op' \rangle$, $v_{fn}$, $c''$, $SU$, $sig \rangle$ with $\mathbf{Ver}^{\mathbf{IBS}}_{SU}(\langle \mathsf{FK}$, $id$, $\langle fn, op' \rangle$, $v_{fn}$, $c''$, $SU \rangle$, $sig) = 1$:
    · Send $\langle \mathsf{FK}$, $id$, $\langle fn, op' \rangle$, $v_{fn} + 1$, $\mathbf{Enc}^{\mathbf{IBE}}_{id}(k')$, $SU$, $\mathbf{Sign}^{\mathbf{IBS}}_{SU} \rangle$ to R.M.
  * Increment $v_{fn}$ in FILES, i.e., set $v_{fn} := v_{fn} + 1$
- Increment $v_r$ in ROLES, i.e., set $v_r := v_r + 1$
- Delete all $\langle \mathsf{RK}$, $-$, $(r, v_r)$, $-$, $- \rangle$
- Delete all $\langle \mathsf{FK}$, $(r, v_r)$, $-$, $-$, $-$, $-$, $- \rangle$

$assignP(r, \langle fn, op \rangle)$
- For all $\langle \mathsf{FK}$, $SU$, $\langle fn, \mathsf{RW} \rangle$, $v$, $c$, $id$, $sig \rangle$ with $\mathbf{Ver}^{\mathbf{IBS}}_{id}(\langle \mathsf{FK}$, $SU$, $\langle fn, \mathsf{RW} \rangle$, $v$, $c$, $id \rangle$, $sig) = 1$:
  * If this adds Write permission to existing Read permission, i.e., $op = \mathsf{RW}$ and there exists $\langle \mathsf{FK}$, $(r, v_r)$, $\langle fn, \mathsf{Read} \rangle$, $v$, $c'$, $SU$, $sig \rangle$ with $\mathbf{Ver}^{\mathbf{IBS}}_{SU}(\langle \mathsf{FK}$, $(r, v_r)$, $\langle fn, op' \rangle$, $v$, $c'$, $SU \rangle$, $sig) = 1$:
    · Send $\langle \mathsf{FK}$, $(r, v_r)$, $\langle fn, \mathsf{RW} \rangle$, $v$, $c'$, $SU$, $\mathbf{Sign}^{\mathbf{IBS}}_{SU} \rangle$ to R.M.
    · Delete $\langle \mathsf{FK}$, $(r, v_r)$, $\langle fn, \mathsf{Read} \rangle$, $v$, $c'$, $SU$, $sig \rangle$
  * If the role has no existing permission for the file, i.e., there does not exist $\langle \mathsf{FK}$, $(r, v_r)$, $\langle fn, op' \rangle$, $v$, $c'$, $SU$, $sig \rangle$ with $\mathbf{Ver}^{\mathbf{IBS}}_{SU}(\langle \mathsf{FK}$, $(r, v_r)$, $\langle fn, op' \rangle$, $v$, $c$, $SU \rangle$, $sig) = 1$:
    · Decrypt key $k = \mathbf{Dec}^{\mathbf{IBE}}_{k_{SU}}(c)$
    · Send $\langle \mathsf{FK}$, $(r, v_r)$, $\langle fn, op \rangle$, $v$, $\mathbf{Enc}^{\mathbf{IBE}}_{(r,v_r)}(k)$, $SU$, $\mathbf{Sign}^{\mathbf{IBS}}_{SU} \rangle$ to R.M.

$revokeP(r, \langle fn, op \rangle)$
- If $op = \mathsf{Write}$:
  * For all $\langle \mathsf{FK}$, $(r, v_r)$, $\langle fn, \mathsf{RW} \rangle$, $v$, $c$, $SU$, $sig \rangle$ with $\mathbf{Ver}^{\mathbf{IBS}}_{SU}(\langle \mathsf{FK}$, $(r, v_r)$, $\langle fn, \mathsf{RW} \rangle$, $v$, $c$, $SU \rangle$, $sig) = 1$:
    · Send $\langle \mathsf{FK}$, $(r, v_r)$, $\langle fn, \mathsf{Read} \rangle$, $v$, $c$, $SU$, $\mathbf{Sign}^{\mathbf{IBS}}_{SU} \rangle$ to R.M.
    · Delete $\langle \mathsf{FK}$, $(r, v_r)$, $\langle fn, \mathsf{RW} \rangle$, $v$, $c$, $SU$, $sig \rangle$
- If $op = \mathsf{RW}$:
  * Delete all $\langle \mathsf{FK}$, $(r, v_r)$, $\langle fn, - \rangle$, $-$, $-$, $- \rangle$
  * Generate new symmetric key $k' \leftarrow \mathbf{Gen}^{\mathbf{Sym}}$
  * For all $\langle \mathsf{FK}$, $r'$, $\langle fn, op' \rangle$, $v_{fn}$, $c$, $SU$, $sig \rangle$ with $\mathbf{Ver}^{\mathbf{IBS}}_{SU}(\langle \mathsf{FK}$, $r'$, $\langle fn, op' \rangle$, $v$, $c$, $SU \rangle$, $sig) = 1$:
    · Send $\langle \mathsf{FK}$, $r'$, $\langle fn, op' \rangle$, $v_{fn} + 1$, $\mathbf{Enc}^{\mathbf{IBE}}_{id}(k')$, $SU$, $\mathbf{Sign}^{\mathbf{IBS}}_{SU} \rangle$ to R.M.
  * Increment $v_{fn}$ in FILES, i.e., set $v_{fn} := v_{fn} + 1$

$read_u(fn)$
- Find $\langle \mathsf{F}, fn, v, c, id, sig \rangle$ with valid ciphertext $c$ and valid signature $sig$, i.e., $\mathbf{Ver}^{\mathbf{IBS}}_{id}(\langle \mathsf{F}, fn, 1, c, id \rangle, sig) = 1$
- Find a role $r$ such that the following hold:
  * $u$ is in role $r$, i.e., there exists $\langle \mathsf{RK}$, $u$, $(r, v_r)$, $c'$, $sig \rangle$ with $\mathbf{Ver}^{\mathbf{IBS}}_{SU}(\langle \mathsf{RK}$, $u$, $(r, v_r)$, $c' \rangle$, $sig) = 1$
  * $r$ has read access to version $v$ of $fn$, i.e., there exists $\langle \mathsf{FK}$, $(r, v_r)$, $\langle fn, op \rangle$, $v$, $c''$, $SU$, $sig' \rangle$ with $\mathbf{Ver}^{\mathbf{IBS}}_{SU}(\langle \mathsf{FK}$, $(r, v_r)$, $\langle fn, op \rangle$, $v$, $c''$, $SU \rangle$, $sig') = 1$
- Decrypt role key $k_{(r,v_r)} = \mathbf{Dec}^{\mathbf{IBE}}_{k_u}(c')$
- Decrypt file key $k = \mathbf{Dec}^{\mathbf{IBE}}_{k_{(r,v_r)}}(c'')$
- Decrypt file $f = \mathbf{Dec}^{\mathbf{Sym}}_k(c)$

$write_u(fn, f)$
- Find a role $r$ such that the following hold:
  * $u$ is in role $r$, i.e., there exists $\langle \mathsf{RK}$, $u$, $(r, v_r)$, $c$, $sig \rangle$ with $\mathbf{Ver}^{\mathbf{IBS}}_{SU}(\langle \mathsf{RK}$, $u$, $(r, v_r)$, $c \rangle$, $sig) = 1$
  * $r$ has write access to the newest version of $fn$, i.e., there exists $\langle \mathsf{FK}$, $(r, v_r)$, $\langle fn, \mathsf{RW} \rangle$, $v_{fn}$, $c'$, $SU$, $sig' \rangle$ and $\mathbf{Ver}^{\mathbf{IBS}}_{SU}(\langle \mathsf{FK}$, $(r, v_r)$, $\langle fn, \mathsf{RW} \rangle$, $v$, $c'$, $SU \rangle$, $sig') = 1$
- Decrypt role key $k_{(r,v_r)} = \mathbf{Dec}^{\mathbf{IBE}}_{k_u}(c)$
- Decrypt file key $k = \mathbf{Dec}^{\mathbf{IBE}}_{k_{(r,v_r)}}(c')$
- Send $\langle \mathsf{F}, fn, v_{fn}, \mathbf{Enc}^{\mathbf{Sym}}_k(f), (r, v_r), \mathbf{Sign}^{\mathbf{IBS}}_{(r,v_r)} \rangle$ to R.M.
- The R.M. receives $r$ and $\langle \mathsf{F}, fn, v, c'', (r, v_r), sig'' \rangle$ and verifies the following:
  * The tuple is well-formed with $v = v_{fn}$
  * The signature is valid, i.e., $\mathbf{Ver}^{\mathbf{IBS}}_{(r,v_r)}(\langle \mathsf{F}$, $fn$, $v$, $c'', (r, v_r) \rangle$, $sig'') = 1$
  * $r$ has write access to the newest version of $fn$, i.e., there exists $\langle \mathsf{FK}$, $(r, v_r)$, $\langle fn, \mathsf{RW} \rangle$, $v_{fn}$, $c'$, $SU$, $sig' \rangle$ and $\mathbf{Ver}^{\mathbf{IBS}}_{SU}(\langle \mathsf{FK}$, $(r, v_r)$, $\langle fn, \mathsf{RW} \rangle$, $v_{fn}$, $c'$, $SU \rangle$, $sig') = 1$
- If verification is successful, the R.M. replaces $\langle \mathsf{F}, fn, -, -, -, - \rangle$ with $\langle \mathsf{F}, fn, v_{fn}, c'', (r, v_r), sig'' \rangle$
```

Fig. 2: Implementation of RBAC_0 using IBE and IBS

For formal definitions of these properties, see [36]. Using parameterized expressiveness, we get the following results:

Theorem 1: The implementation of RBAC_0 using IBE and IBS detailed in Fig. 2 is correct, AC-preserving, and safe.

Theorem 2: The implementation of RBAC_0 using public key cryptographic techniques is correct, AC-preserving, and safe.

We now give an overview of the structure of and ideas behind the proof of Theorem 1. This proof begins by formalizing the IBE/IBS construction presented in Section IV using the parameterized expressiveness framework. We then provide a formal mapping from RBAC_0 to our IBE/IBS system. We show that this mapping preserves user authorization, meaning that a user is authorized for a permission in RBAC_0 if and only if the user is also authorized by the IBE/IBS construction.

The tricky part of this proof involves showing that changes to the RBAC_0 state map correctly as changes to the IBE/IBS state. This means that changing the RBAC_0 state and then mapping to IBE/IBS has the same effect as mapping to IBE/IBS and then changing the state there in an equivalent way. Our use of version numbers in IBE/IBS means that a single RBAC_0 state may map to multiple IBE/IBS states; i.e., if a user is granted permissions that are later revoked, the resulting RBAC_0 state will be the same as if the permissions were never granted, but the IBE/IBS state will have different version numbers as a result of the revocation. Therefore, we consider IBE/IBS states that only differ in version numbers to be *congruent*. We show that the IBE/IBS state resulting from a change to the RBAC_0 state, followed by mapping to IBE/IBS, is congruent to one crafted by first mapping to IBE/IBS, and then changing the IBE/IBS state in a corresponding way.

The full proof of Theorem 1 can be found in Appendix A. We note that the proof of Theorem 2 is very similar in structure, and an accompanying technical report [28] provides the details.

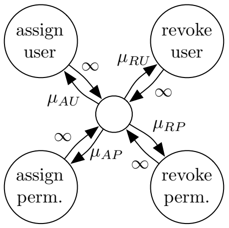
B. Algebraic Costs

Table I lists the costs for each RBAC operation based on the system state. All costs are incurred by the user or administrator running the operation unless otherwise noted. In order to simplify the formulas, we employ a slight abuse of notation: we use the operation itself to represent its cost (e.g., $\mathbf{Enc}^{\mathbf{IBE}}$ is used to represent the cost of one $\mathbf{Enc}^{\mathbf{IBE}}$ operation). We use the following notation:

- $roles(u)$ is the set of roles to which user u is assigned
- $perms(r)$ is the set of permissions to which role r is assigned
- $users(r)$ is the set of users to which role r is assigned
- $roles(p)$ is the set of roles to which permission p is assigned
- $versions(p)$ is the number of versions of permission p

C. Experimental Setup

To evaluate the costs of using our constructions to enforce RBAC_0, we utilize the simulation framework proposed in [26]. We encode RBAC_0 as a workload, with implementations in IBE/IBS and PKI as described in Sections IV-C and IV-D. Simulations are initialized from start states extracted from real-world RBAC datasets. We then generate traces of access control actions using actor-specific continuous-time Markov chains, or *actor machines*. While this is a fairly simple model of actors' behaviors, it allows us to easily investigate trends in costs. In particular, we are able to investigate changes in the relative frequencies of the various administrative actions, and the costs resulting from these changes.



var	semantics	value
R	administrative rate	$0.1 \times \sqrt{\lvert U \rvert}/\text{day}$
μ_A	add bias	$[0.7, 1.0]$
μ_U	UR bias	$[0.3, 0.7]$
μ_{AU}	Rate of assignUser	$\mu_A \times \mu_U \times R$
μ_{RU}	Rate of revokeUser	$(1 - \mu_A) \times \mu_U \times R$
μ_{AP}	Rate of assignPermission	$\mu_A \times (1 - \mu_U) \times R$
μ_{RP}	Rate of revokePermission	$(1 - \mu_A) \times (1 - \mu_U) \times R$

Fig. 3: Administrative actions in our experiments

We simulate one-month periods in which the administrator of the system behaves as described in the actor machine depicted in Fig. 3. The administrative workload increases with the number of users in the system, and we randomly sample an *add bias* parameter that describes the relative proportion of assignment vs. revocation operations. We do not include administrative actions that add or remove users or roles, due to the unlikely occurrence of these actions on such short timescales (one-month simulations).

This administrative behavior model describes a range of realistic scenarios and thus allows us to investigate the interactions in which we are interested. The overall administrative rate is approximately $\sqrt{|U|}$ (with $|U|$ the number of users), ranging from about 0.6 administrative actions per day on our smallest dataset to 2.2 on the largest. We consider the range of 0% to 30% of the administrative load consisting of revocations, since in realistic scenarios permissions tend to be assigned at a greater rate than they are revoked [63].

To quantify the costs associated with our cryptographic constructions, we record the number of instances of each cryptographic operation executed, including counts or averages for traces of related operations (e.g., the average number of IBE encryptions needed to revoke a role from a user).

As mentioned above, simulation start states are extracted from real-world RBAC datasets. These datasets are summarized in Table II. All of these datasets, aside from university, were originally provided by HP [22]. The domino dataset is from a Lotus Domino server, emea is from a set of Cisco firewalls, firewall1 and firewall2 are generated from network reachability analysis, and healthcare is a list of

$$
\begin{aligned}
\mathit{addU}(u)\colon\ & \mathbf{KeyGen^{IBE}} + \mathbf{KeyGen^{IBS}} \\
\mathit{delU}(u)\colon\ & \textstyle\sum_{r \in roles(u)} \mathit{revokeU}(u,r) \\
\mathit{addP}(p)\colon\ & \mathbf{Enc^{IBE}} + 2 \cdot \mathbf{Sign^{IBS}} \text{ and } 2 \cdot \mathbf{Ver^{IBS}} \text{ by R.M.} \\
\mathit{delP}(p)\colon\ & \text{None} \\
\mathit{addR}(r)\colon\ & \mathbf{KeyGen^{IBE}} + \mathbf{Enc^{IBE}} + \mathbf{KeyGen^{IBS}} + \mathbf{Sign^{IBS}} \\
\mathit{delR}(r)\colon\ & \textstyle\sum_{p \in perms(r)} \mathit{revokeP}(p,r) \\
\mathit{assignU}(u,r)\colon\ & \mathbf{Enc^{IBE}} + \mathbf{Dec^{IBE}} + \mathbf{Sign^{IBS}} + \mathbf{Ver^{IBS}} \\
\mathit{revokeU}(u,r)\colon\ & \mathbf{KeyGen^{IBE}} + \mathbf{KeyGen^{IBS}} + \Big(|users(r)| + \textstyle\sum_{p \in perms(r)} (versions(p) + |roles(p)|)\Big) \big(\mathbf{Enc^{IBE}} + \mathbf{Sign^{IBS}} + \\
& \mathbf{Ver^{IBS}}\big) + \Big(\mathbf{Dec^{IBE}} \cdot \textstyle\sum_{p \in perms(r)} versions(p)\Big) \\
\mathit{assignP}(p,r)\colon\ & versions(p) \cdot \big(\mathbf{Sign^{IBS}} + \mathbf{Ver^{IBS}}\big)\text{; if } r \text{ has no permissions for the file then also } versions(p) \cdot \big(\mathbf{Enc^{IBE}} + \mathbf{Dec^{IBE}}\big) \\
\mathit{revokeP}(p,r)\colon\ & \text{Revokes all access: } |roles(p)| \cdot \big(\mathbf{Enc^{IBE}} + \mathbf{Sign^{IBS}} + \mathbf{Ver^{IBS}}\big)\text{;} \\
& \text{Revokes only write access: } |versions(p)| \cdot \big(\mathbf{Sign^{IBS}} + \mathbf{Ver^{IBS}}\big) \\
\mathit{read}(fn)\colon\ & 2 \cdot \big(\mathbf{Dec^{IBE}} + \mathbf{Ver^{IBS}}\big) \\
\mathit{write}(fn,f)\colon\ & \mathbf{Sign^{IBS}} + 2 \cdot \big(\mathbf{Dec^{IBE}} + \mathbf{Ver^{IBS}}\big) \text{ and } 2 \cdot \mathbf{Ver^{IBS}} \text{ by R.M.}
\end{aligned}
$$

TABLE I: Algebraic costs of RBAC_0 operations in our IBE/IBS implementation

						roles/user		users/role		perm./role		roles/perm.	
set	users	|P|	|R|	|UR|	|PA|	max	min	max	min	max	min	max	min
domino	79	231	20	75	629	3	0	30	1	209	1	10	1
emea	35	3046	34	35	7211	1	1	2	1	554	9	31	1
firewall1	365	709	60	1130	3455	14	0	174	1	617	1	25	1
firewall2	325	590	10	325	1136	1	1	222	1	590	6	8	1
healthcare	46	46	13	55	359	5	1	17	1	45	7	12	1
university	493	56	16	495	202	2	1	288	1	40	2	12	1

TABLE II: Overview of the datasets used in our experiments

healthcare permissions from the US Veteran's Administration. The university dataset describes a university's access control system, and was developed by IBM [48], [65].

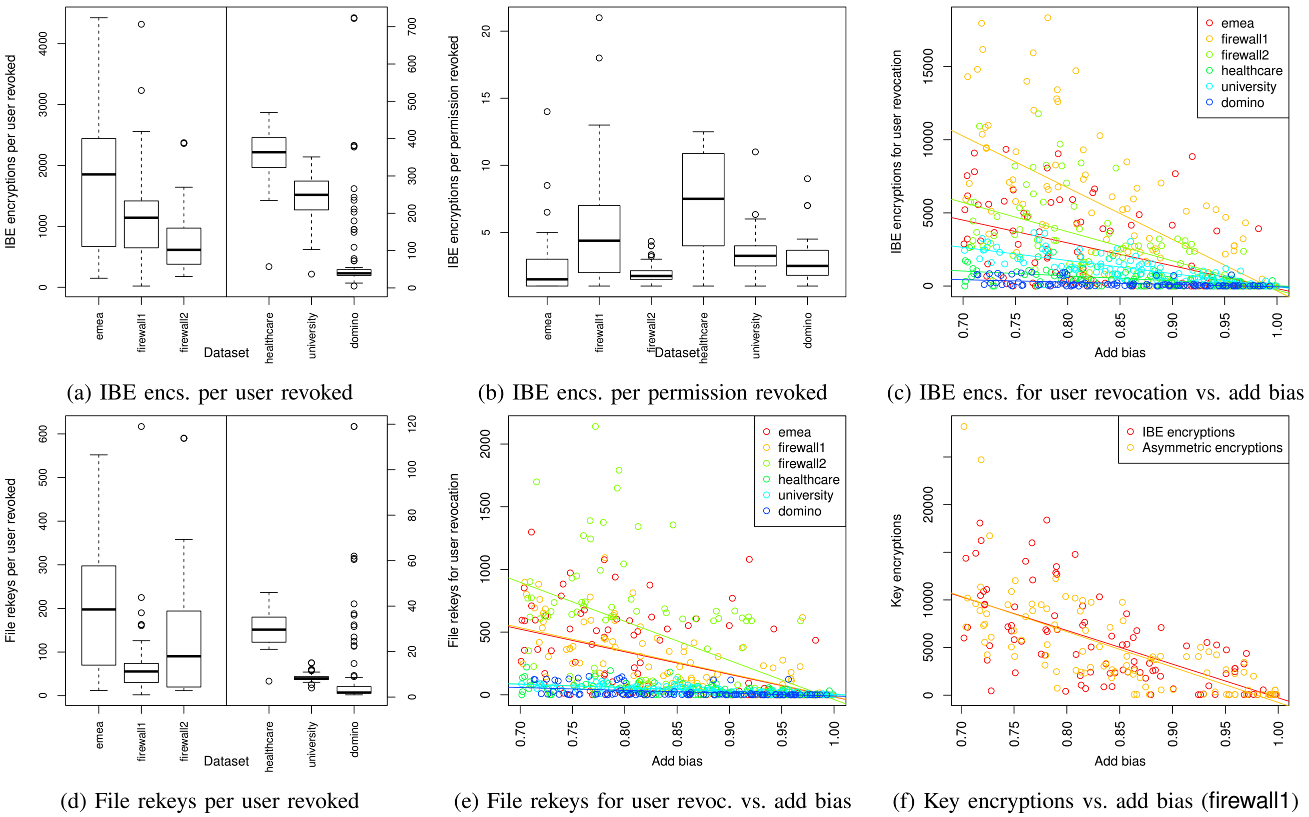
D. Experimental Results

Figure 4 presents a sampling of our results. First, we consider the cost of performing revocations in our implementation of RBAC_0 using IBE/IBS. Figure 4a shows the average number of IBE encryptions needed for a single user revocation (i.e., removing a user from a role), and Fig. 4b shows the same for permission revocation (i.e., revoking a permission from a role). This shows that revoking a permission can cost several IBE encryptions, while user revocation incurs hundreds or thousands of IBE encryptions, on average. We note that, by inspection of the code in Fig. 2, a user revocation also requires an equal number of IBS signatures and verifications, a smaller number of IBE decryptions, and the generation of new IBE and IBS keys for the role.

For our chosen distribution of administrative actions, Fig. 4c shows the total number of IBE encryptions performed over a month for *all* user revocations. As the add bias approaches 1, the number of revocations (and thus the total number of IBE encryptions for user revocation) approaches 0. However, even when only 5–10% of administrative actions are revocation, the number of monthly IBE encryptions under this parameterization is often in the thousands.

In Fig. 4d, we show the number of files that must be re-keyed for a single user revocation. This highlights the benefit of utilizing lazy re-encryption; if we had instead utilized *active* re-encryption, each of these files would need to be locked, downloaded, decrypted, re-encrypted, and re-uploaded *immediately* following revocation. In certain scenarios, active re-encryption may be computationally feasible. For instance, in university, only ≈ 10 files must be re-encrypted for the average user revocation, adding less than 1% to the total number of file encryptions executed over the entire simulation, even at the highest rate of revocations that we consider. However, in most other scenarios, a user revocation triggers the re-key of tens or hundreds of files, such as in emea or firewall2, where active re-encryption increases the total number of file encryptions by 63% and 12%, respectively (at 20–30% revocation rate). Thus, in most scenarios, active re-encryption is likely to be infeasible, as discussed in Section IV-B.

Given the administrative behavior model depicted in Fig. 3, Fig. 4e shows the total number of file re-keys that take place over a month for the purpose of user revocation. For scenarios with very user- and permission-dense roles (e.g., firewall1 and firewall2), we see several times as many re-keys as *total files*, indicating that, on average, each file is re-keyed multiple times per month for the purposes of user revocation. This further enforces that inefficiencies that active re-encryption would bring, as each file (on average) would be locked and



(a) IBE encs. per user revoked

(b) IBE encs. per permission revoked

(c) IBE encs. for user revocation vs. add bias

(d) File rekeys per user revoked

(e) File rekeys for user revoc. vs. add bias

(f) Key encryptions vs. add bias (firewall1)

Fig. 4: Results of running 100 one-month simulations on each dataset (each data point is a simulation)

re-encrypted by the administrator multiple times per month.

Finally, we note that the costs for our IBE/IBS- and PKI-based constructions for RBAC_0 are not notably different. For instance, Fig. 4f compares, for scenario firewall1, the number of IBE encryptions with the number of asymmetric encryptions executed over each simulated month and reveals the same distribution in both IBE/IBS- and PKI-based constructions. Given the similarity in the cost of these classes of operations, we can conclude that these constructions are similarly expensive from a computational standpoint.

E. Converting Experimental Results to Real Costs

We now demonstrate how the costs of generic IBE encryptions turn into actual computational costs for given schemes. Since any implementation's running time is contingent on a myriad of variables (e.g., processor speed, memory, etc.) we focus on the number of (pairing friendly) elliptic curve cryptographic operations that need to be performed. We assume schemes are implemented using an asymmetric (Type 3) pairing: $e : \mathbb{G} \times \hat{\mathbb{G}} \rightarrow \mathbb{G}_T$, where $\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T$ are groups of prime order; this is more efficient than a symmetric (Type 1) pairing [25]. Additive notation is used in $\mathbb{G}$ and $\hat{G}$, while multiplicative notation is used in $\mathbb{G}_T$.

We use multiplication in $\mathbb{G}$ as our cost unit, expressing the relative costs of other operations in terms of this operation. The relative costs should be somewhat stable across hardware and reasonable implementations. These relative costs are given in Table III and are based on data provided by Ayo Akinyele, an ABE/pairing implementation expert at Zeutro LLC (personal communication). Costs of addition in $\mathbb{G}, \hat{\mathbb{G}}$, and multiplication in $\mathbb{G}_T$ are so low that we ignore them. These relative costs are based on the implementation of RELIC v0.4 [2], using a Barreto-Naehrig curve with a 256-bit base field, GMP for big number operations, and standard configuration options for prime field arithmetic. For a point of reference, a reasonable modern workstation running RELIC v0.4 on such curves will take approximately 0.2 ms on average to compute a multiplication in $\mathbb{G}$.

Operation	$\hat{\mathbb{G}}$ Multiply	$\mathbb{G}_T$ Exp.	Pairing (e)
$\mathbb{G}$ Multiplies	4.5	9	9

TABLE III: Relative cost of Type 3 pairing operations in terms of multiplication in $\mathbb{G}$ in RELIC v0.4

To determine concrete costs, we consider three representative combinations of IBE and IBS algorithms:

BF+CC: The IBE scheme from [11, Sec. 4.1] and the IBS scheme from [13, Sec. 2]. Both are efficient and are proven secure in the random oracle model.

$\mathbf{BB}_1$+PS: The IBE scheme from [9, Sec. 4] and the IBS scheme from [54, Sec. 4]. These schemes are less efficient than BF+CC but are proven secure in the standard model.

LW+PS: The IBE scheme from [44, App. C] and the IBS scheme from [54, Sec. 4]. The IBE scheme here is less efficient but has stronger security properties.

Table IV lists the cost of each additive RBAC_0, read and write operation in terms of total "multiplication units" in $\mathbb{G}$. That is, we sum the cost of cryptographic operations in terms of multiplication units using the conversion factor in Table III. Table IV specifies the costs incurred by the invoker of the operation (either the admin or the user) as well as the reference monitor. For completeness, a table documenting the individual costs of each basic IBE/IBS operation for these schemes can be found in [28], an accompanying technical report.

Incurred by	Operation	BF+CC	BB_1+PS	LW+PS
Invoker	addU	5.5	14.5	32.5
	addP	15	25	29
	addR	18.5	33	55
	assignU	41	63.5	103.5
	assignP[3]	41	63.5	103.5
	read	56	90	162
	write	58	96.5	168.5
R.M.	addP	38	54	54
	write	38	54	54

TABLE IV: Costs of operations in terms of $\mathbb{G}$ multiplications

The cost to delete a user/role or to revoke a user/permission depends on the RBAC state at the time of revocation, so we cannot give definite costs for these operations. Instead, we use the experimental results from Section V-D to get an idea of how expensive revocation can be. The results of this are in Fig. 5, where we plot the costs for each dataset using the three IBE/IBS combinations listed above. Figure 5a shows the cost of revoking a user in terms of multiplications in $\mathbb{G}$; Fig. 5b does the same for revoking a permission. Note that for our datasets, a single user revocation usually costs more than 10,000 multiplications in $\mathbb{G}$ ($\approx$ 2 s. on a modern workstation), and often costs more than 100,000 multiplications ($\approx$ 20 s.) for some datasets. While not exceedingly huge, we remind the reader that our costing does not account for many costs, such as concurrency, communication, and storage costs. Further, our construction minimizes other costs through the use of lazy re-encryption and hybrid encryption.

VI. Discussion

There is no doubt that IBE and ABE can enable various forms of cryptographic access control for data in the cloud. In fact, the results presented in Figs. 4c, 4e and 4f show that in situations in which the system grows in a monotonic manner (i.e., users and files are *added* to the system and roles are provisioned with *new* permissions), there is no need for revocation, re-keying, or complicated metadata management: IBE alone can enforce RBAC access controls on the cloud. In fact, there are even implications or direct claims in the literature that, in the static setting, the reference monitor can be removed entirely (e.g., [30], [31], [49]). However, this does not imply that IBE or ABE alone can entirely replace the use of a reference monitor when implementing outsourced access controls: *it is not the case when dynamic controls are required.*

Specifically, this paper shows that IBE and PKI systems are well-suited for implementing *point states* of an RBAC_0 system. However, managing *transitions* between these states—specifically, supporting the removal of a user from a role, the revocation of a permission from a role, and efficient updates to files shared with multiple roles—requires non-trivial metadata management and a small, minimally-trusted reference monitor that verifies signatures prior to file deletion and replacement. In some of the datasets that we analyzed, this could lead to thousands of IBE encryptions (Fig. 4a) and over one hundred file re-keys/re-encryptions (Fig. 4d) when a single user is removed from a role.

The above considerations lead to a tradeoff between confidentiality and efficiency that must be weighed by both cryptographers and system designers. There are two obvious ways that this can be accomplished: by altering the threat model assumed, or developing cryptographic approaches that are more amenable to the dynamic setting. We now discuss both of these approaches, and comment on lessons learned during our analysis that can be applied to richer cryptographic access control, such as using HIBE to support RBAC_1, or ABE to support ABAC.

A. Alternate Threat Models

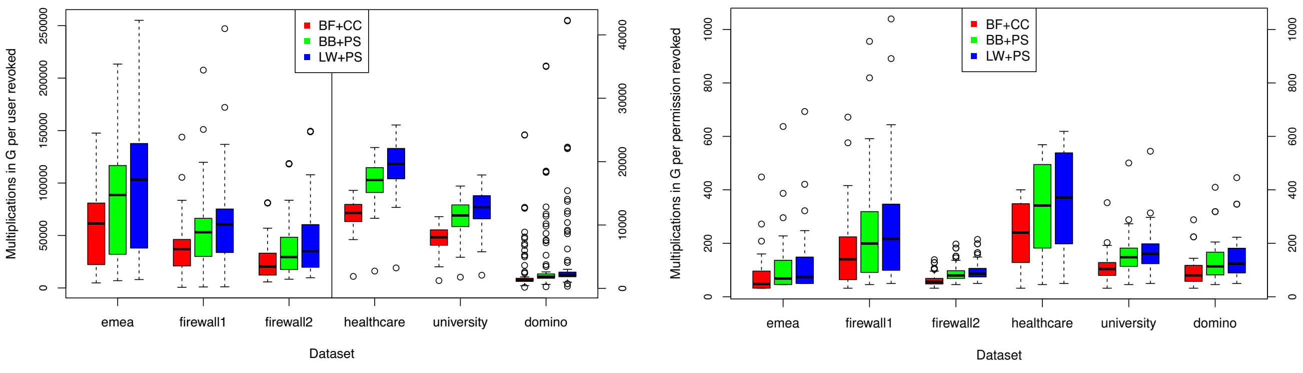
Many of the overheads that we report on in the previous section result from the threat model often implied by the cryptographic literature (i.e., untrusted storage server, minimal client-side infrastructure). Altering this model can reduce the cryptographic costs of enforcing dynamic access controls on the cloud. Here we consider two such alternate models.

Encryption/Decryption Proxy. A large amount of overhead comes from relying the cloud storage provider to act as a (cryptographic) metadata broker, as well as a file store. An alternative approach might make use of an encryption/decryption proxy server situated within an organization, using the cloud provider solely as a backing store for encrypted files. This proxy would act as a traditional reference monitor, mediating all file access requests, downloading and decrypting files for authorized readers[4], and returning plaintext to the user. This would obviate the need for any cryptography beyond authenticated symmetric key encryption, and could make use of tried-and-true access control reference monitors. However, this approach carries an extra infrastructure overhead (the proxy server, itself) that could make it unappealing to *individuals* hoping to enforce access controls over cloud hosted files. Large *organizations* may also have to deal with synchronizing access control policies and key material across multiple proxies in the event that file I/O demands outpace the abilities of a single server.

Trusted Hardware. A more extreme approach to simplifying the cryptographic overheads of access control enforcement would be to use, e.g., an SGX enclave [39], [47] to carry out the work of the encryption/decryption proxy discussed above.

[3] Assumes permission is for new file; cost is per version of the file

[4] Writes could be handled symmetrically.



(a) Multiplications in $\mathbb{G}$ per user revoked

(b) Multiplications in $\mathbb{G}$ per permission revoked

Fig. 5: Costs of revocation in different IBE/IBS schemes, in terms of elliptic curve point multiplications in group $\mathbb{G}$

In this scenario, files could be stored encrypted on the cloud server, while file encryption keys and the access control policy to be enforced would be managed by a process running within an SGX enclave. To access a file, a user would negotiate an authenticated channel (e.g., using public key cryptography) with this trusted process/reference monitor. The reference monitor could then check the user's permission to access the file, and transmit the encrypted file and its associated key to the user using a session key that is unknown to any process outside of the SGX enclave. This approach frees organizations from the overheads of running their own encryption/decryption proxies, but is not without its limitations. For instance, this approach will not work on commonly-used, storage-only services (e.g., Dropbox). Further, this approach may be subject to architectural compromises or flaws (e.g., memory integrity vulnerabilities) that cryptography-only solutions are not.

While these and other alterations to the threat model that we consider can lead to decreased cryptographic overheads, each incurs other costs or tradeoffs. We now consider future research directions that may decrease the costs associated with cryptography-only solutions to the problem of outsourcing dynamic access controls.

B. Future Directions

Our experimentation and analysis has led to a number of interesting directions for future work:

- **Revocation.** It is unclear how to use IBE to enforce even RBAC_0 without incurring high costs associated with revocation-based state changes. Given our use of hybrid cryptography for efficiency reasons, existing schemes for revocation or proxy re-encryption (e.g., [8], [32]) cannot solve the problem. Developing techniques to better facilitate these forms of revocation and efficient use of hybrid encryption is an important area of future work.
- **Trust Minimization.** Our construction makes use of a reference monitor on the cloud to validate signatures prior to file replacement or metadata update. Moving to file versioning (e.g., based on trusted timestamping or block-chaining) rather than file replacement may result in a minimization of the trust placed in this reference monitor, but at the cost of potential confidentiality loss, since old key material may remain accessible to former role members. It is important to better explore this tradeoff between reference monitor trust and confidentiality guarantees.
- **"Wrapper" Minimization.** Our construction required the management and use of three types of metadata structures to correctly implement RBAC_0 using IBE or PKI technologies. It would be worth exploring whether the core cryptography used to support outsourced access controls could be enhanced to reduce the use of trusted management code needed to maintain these sorts of structures.
- **Deployability/Usability Costs.** We did not consider issues related to the *use* of the cryptographic tools underlying our constructions. Further, our simulations do not separate our IBE- and PKI-based constructions[5] on the basis of RBAC_0 implementation complexity. However, it may be the case that the maturity of tools to support the use of PKIs or the conceptual simplicity of IBE techniques tips the scales in one direction or the other. Developing reasonable approaches for considering these types of tradeoffs would greatly inform future analyses.

While this paper focused on the use of IBE/IBS and PKI schemes to enforce RBAC_0 access controls, our findings translate in a straightforward manner to the use of other cryptographic tools (e.g., HIBE or ABE/ABS) to implement more complex access control policies (e.g., RBAC_1 or ABAC). We now discuss some lessons learned when considering these richer access control models.

C. Lessons Learned for More Expressive Systems

RBAC_0 and IBE were natural choices for our initial exploration of the costs associated with using cryptography to implement dynamic access control: RBAC_0 is a simple, but widely used, access control system; roles in RBAC_0 have a natural correspondence to identities in IBE; and the use of hybrid encryption allows us to easily share resources between

[5] Our PKI construction and its corresponding simulations were omitted from this paper due to space limitations.

roles. Further, it seemed like an implementation of RBAC_0 using IBE would be a jumping-off point for exploring the use of hierarchical roles in RBAC_1 via an analogous use of HIBE. However, many of the costs that we see with our IBE implementation of RBAC_0 have analogues (or worse) in any reasonable RBAC_1 or ABAC implementation that we foresee based on respective cryptographic operations.

We first note that we assume that any reasonable cryptographic access control system must make use of hybrid encryption. Without hybrid encryption, we would need to continuously apply expensive asymmetric operations to small "blocks" of a file that is to be encrypted. Given the complexity of IBE/ABE encryption operations, the associated overheads of this approach would be prohibitive, even for moderately-sized files. Additionally, depending on the security requirements of the application (e.g., Chosen Ciphertext Attack security), even more complicated constructions than this simple blocking will be required. The following observations may not apply to an access control scheme where all files are small enough to do away with the need of hybrid-encryption. However, the use cases for such schemes seem limited.

A seemingly natural extension of our IBE-based RBAC_0 scheme to a HIBE based RBAC_1 scheme exploits the fact that the HIBE can be used to encode hierarchical relationships, such as those that exist between roles in a RBAC_1 role hierarchy. However, the costs of this implementation proved to be considerable. A large initial problem is that an RBAC_1 role hierarchy can be an arbitrary DAG structure, while HIBE only supports trees. Yet, even limiting RBAC_1 to role hierarchies that form a tree structure comes with serious costs. For example, removing non-leaf roles in the hierarchy cascades re-encryption down to all files at descendant leaves of the role, the creation of new roles for each descendant node, and associated rekeying. Similarly, practical operations like moving sub-trees in the access structure can only be achieved by breaking the operation down into addition and deletion of roles, which comes with the associated costs of these primitive operations. We note that we have developed a full RBAC_1 implementation using HIBE, which attempts to minimize costs. Unfortunately, a simple inspection of this implementation shows that it would incur significantly more computational expense than the RBAC_0 scheme discussed herein.

Similarly, one might hope that the expressiveness of the ABE encryption schemes would allow us to naturally implement ABAC access control schemes. Further, there has been some initial work [58] supporting dynamic (restrictive) credentials and revocations. However, there is still significant work associated with making a practical ABE implementation of ABAC, and such schemes will still have significant costs and meta-data to manage (as in our IBE/RBAC_0 implementation). For example, revoking a secret-key in an KP-ABE/ABAC setting requires the dynamic re-encryption of every ciphertext whose attributes satisfy the policy in the revoked user's key. Each attribute in each ciphertext that is re-encrypted must given a new version, and then finally all users whose keys have policies affected by the re-versioning of the attributes must be re-issued. Further, there are ABAC design decisions that must be informed by the ABE scheme being implemented. For example, suppose a single file is to be accessed by multiple policies in a CP-ABE scheme. One can support multiple policies $p_1, \ldots, p_n$ as individual public-key encryptions all encrypting the same hybrid key, or as a single encryption supporting the disjunction of all previous policies, $p_1 \vee p_2 \vee \cdots \vee p_n$. The cost tradeoffs are completely dependent on the ABE scheme used for the implementation, as the cost of ABE encryption is highly dependent on the policy encoded into the ciphertext.

VII. Conclusions

Advanced cryptographic techniques (e.g., IBE and ABE) are promising approaches for cryptographically enforcing rich access controls in the cloud. While prior work has focused on the *types* of policies that can be represented by these approaches, little attention has been given to how policies may *evolve* over time. In this paper, we move beyond cryptographically representing point states in an access control system for cloud-hosted data, and study constructions that cryptographically enforce dynamic (role-based) access controls. We provide evidence that, given the current state of the art, in situations involving even a minimal amount of policy dynamism, the cryptographic enforcement of access controls is likely to carry prohibitive costs. Further, these costs are seemingly amplified when enforcing richer policies (e.g., RBAC_1 or ABAC), requiring more stringent security guarantees (e.g., online, rather than lazy, re-encryption), or assuming more relaxed threat models.

To conduct our analysis, we developed IBE- and PKI-based constructions that use hybrid cryptography to enforce dynamic RBAC_0 access controls over files hosted on a third-party cloud storage provider. In addition to proving the correctness of our constructions, we used real-word RBAC datasets to experimentally analyze their associated cryptographic costs. Our findings indicate that IBE and ABE are a natural fit to this problem in instances where users, roles, and permissions increase monotonically, but incur very high overheads—e.g., sometimes exceeding thousands of encryption operations to support a single revocation—when updates and revocation must be supported. In doing so, we have identified a number of fruitful areas for future work that could lead to more natural constructions for cryptographic enforcement of access control policies in cloud environments.

Acknowledgements. We would like to offer our thanks to our paper shepherd, Úlfar Erlingsson, for his guidance in refining this paper. This work was supported, in part, by the National Science Foundation under awards CNS–1111149, CNS–1228697, and CNS–1253204.

References

[1] S. G. Akl and P. D. Taylor, "Cryptographic solution to a problem of access control in a hierarchy," *TOCS*, vol. 1, no. 3, 1983.

[2] D. F. Aranha and C. P. L. Gouvêa, "RELIC is an efficient library for cryptography," http://code.google.com/p/relic-toolkit/.

[3] M. J. Atallah, M. Blanton, N. Fazio, and K. B. Frikken, “Dynamic and efficient key management for access hierarchies,” *TISSEC*, vol. 12, no. 3, 2009.

[4] M. Y. Becker, C. Fournet, and A. D. Gordon, “Secpal: Design and semantics of a decentralized authorization language,” *Journal of Computer Security*, vol. 18, no. 4, pp. 619–665, 2010.

[5] J. Bethencourt, A. Sahai, and B. Waters, “Ciphertext-policy attribute-based encryption,” in *S&P*, 2007.

[6] A. Birgisson, J. G. Politz, Ú. Erlingsson, A. Taly, M. Vrable, and M. Lentczner, “Macaroons: Cookies with contextual caveats for decentralized authorization in the cloud,” in *NDSS*, 2014.

[7] M. Blaze, J. Feigenbaum, and J. Lacy, “Decentralized trust management,” in *S&P*, 1996.

[8] A. Boldyreva, V. Goyal, and V. Kumar, “Identity-based encryption with efficient revocation,” in *CCS*, 2008.

[9] D. Boneh and X. Boyen, “Efficient selective identity-based encryption without random oracles,” *Journal of Cryptology*, vol. 24, no. 4, 2011.

[10] D. Boneh, X. Boyen, and E.-J. Goh, “Hierarchical identity based encryption with constant size ciphertext,” in *EUROCRYPT*, 2005.

[11] D. Boneh and M. Franklin, “Identity-based encryption from the Weil pairing,” *SIAM Journal on Computing*, vol. 32, no. 3, 2003.

[12] D. Boneh, K. Lewi, H. W. Montgomery, and A. Raghunathan, “Key homomorphic prfs and their applications,” in *CRYPTO*, 2013.

[13] J. C. Cha and J. H. Cheon, “An identity-based signature from gap diffie-hellman groups,” in *PKC*, 2003.

[14] J. Crampton, “Cryptographic enforcement of role-based access control,” in *FAST*, 2010.

[15] ——, “Practical and efficient cryptographic enforcement of interval-based access control policies,” *TISSEC*, vol. 14, no. 1, 2011.

[16] J. Crampton, K. M. Martin, and P. R. Wild, “On key assignment for hierarchical access control,” in *CSFW*, 2006.

[17] S. De Capitani di Vimercati, S. Foresti, S. Jajodia, G. Livraga, S. Paraboschi, and P. Samarati, “Enforcing dynamic write privileges in data outsourcing,” *Computers & Security*, vol. 39, 2013.

[18] S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati, “Over-encryption: Management of access control evolution on outsourced data,” in *VLDB*, 2007.

[19] S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati, “Encryption policies for regulating access to outsourced data,” *TODS*, vol. 35, no. 2, 2010.

[20] D. Drummond, “A new approach to China,” Jan. 2010, http://googleblog.blogspot.com/2010/01/new-approach-to-china.html.

[21] C. M. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen, “SPKI certificate theory,” IETF RFC 2693, Sep. 1999, http://www.ietf.org/rfc/rfc2693.txt.

[22] A. Ene, W. Horne, N. Milosavljevic, P. Rao, R. Schreiber, and R. E. Tarjan, “Fast exact and heuristic methods for role minimization problems,” in *SACMAT*, 2008.

[23] A. L. Ferrara, G. Fuchsbauer, B. Liu, and B. Warinschi, “Policy privacy in cryptographic access control,” in *CSF*, 2015.

[24] A. L. Ferrara, G. Fuchsbauer, and B. Warinschi, “Cryptographically enforced RBAC,” in *CSF*, 2013.

[25] S. D. Galbraith, K. G. Paterson, and N. P. Smart, “Pairings for cryptographers,” *Discrete Appl. Math.*, vol. 156, no. 16, 2008.

[26] W. C. Garrison III, A. J. Lee, and T. L. Hinrichs, “An actor-based, application-aware access control evaluation framework,” in *SACMAT*, 2014.

[27] W. C. Garrison III, Y. Qiao, and A. J. Lee, “On the suitability of dissemination-centric access control systems for group-centric sharing,” in *CODASPY*, 2014.

[28] W. C. Garrison III, A. Shull, S. Myers, and A. J. Lee, “On the practicality of cryptographically enforcing dynamic access control policies in the cloud (extended version),” Computing Research Repository, Tech. Rep. arXiv:1602.09069, Feb. 2016.

[29] C. Gentry and A. Silverberg, “Hierarchical ID-based cryptography,” in *ASIACRYPT*, 2002.

[30] V. Goyal, A. Jain, O. Pandey, and A. Sahai, “Bounded ciphertext policy attribute based encryption,” in *ICALP*, 2008.

[31] V. Goyal, O. Pandey, A. Sahai, and B. Waters, “Attribute-based encryption for fine-grained access control of encrypted data,” in *CCS*, 2006.

[32] M. Green and G. Ateniese, “Identity-based proxy re-encryption,” in *ACNS*, 2007.

[33] M. Green, S. Hohenberger, and B. Waters, “Outsourcing the decryption of abe ciphertexts,” in *USENIX Security*, 2011.

[34] E. Gudes, “The Design of a Cryptography Based Secure File System,” *IEEE Transactions on Software Engineering*, vol. 6, no. 5, 1980.

[35] D. Hardt, “The OAuth 2.0 authorization framework,” IETF RFC 6749, Oct. 2012, https://tools.ietf.org/html/rfc6749.

[36] T. L. Hinrichs, D. Martinoia, W. C. Garrison III, A. J. Lee, A. Panebianco, and L. Zuck, “Application-sensitive access control evaluation using parameterized expressiveness,” in *CSF*, 2013.

[37] J. Horwitz and B. Lynn, “Toward hierarchical identity-based encryption,” in *EUROCRYPT*, 2002.

[38] L. Ibraimi, “Cryptographically enforced distributed data access control,” Ph.D. dissertation, University of Twente, 2011.

[39] “Intel software guard extensions programming references,” Intel, Tech. Rep. 329298–002, Oct. 2014.

[40] X. Jin, R. Krishnan, and R. S. Sandhu, “A unified attribute-based access control model covering DAC, MAC and RBAC,” in *DDBSec*, 2012.

[41] J. Katz and Y. Lindell, *Introduction to Modern Cryptography*, 2nd ed. Chapman & Hall/CRC, 2014.

[42] J. Katz, A. Sahai, and B. Waters, “Predicate encryption supporting disjunctions, polynomial equations, and inner products,” in *EUROCRYPT*, 2008.

[43] R. Krishnan, J. Niu, R. S. Sandhu, and W. H. Winsborough, “Group-centric secure information-sharing models for isolated groups,” *TISSEC*, vol. 14, no. 3, 2011.

[44] A. Lewko and B. Waters, “New techniques for dual system encryption and fully secure hibe with short ciphertexts,” in *TCC*, 2010.

[45] N. Li, J. C. Mitchell, and W. H. Winsborough, “Design of a role-based trust-management framework,” in *IEEE Symposium on Security and Privacy*, 2002.

[46] B. Libert and D. Vergnaud, “Adaptive-id secure revocable identity-based encryption,” in *CT-RSA*, 2009.

[47] F. McKeen, I. Alexandrovich, A. Berenzon, C. Rozas, H. Shafi, V. Shanbhogue, and U. Savaganokar, “Innovative instructions and software model for isolated execution,” in *HASP*, 2013.

[48] I. Molloy, H. Chen, T. Li, Q. Wang, N. Li, E. Bertino, S. B. Calo, and J. Lobo, “Mining roles with semantic meanings,” in *SACMAT*, 2008.

[49] S. Müller and S. Katzenbeisser, “Hiding the policy in cryptographic access control,” in *STM*, 2011.

[50] D. Nali, C. M. Adams, and A. Miri, “Using mediated identity-based cryptography to support role-based access control,” in *Information Security, 7th International Conference, ISC 2004*, 2004.

[51] “Operation Aurora,” Jun. 2015, https://en.wikipedia.org/wiki/Operation_Aurora.

[52] R. Ostrovsky, A. Sahai, and B. Waters, “Attribute-based encryption with non-monotonic access structures,” in *CCS*, 2007.

[53] S. Park, K. Lee, and D. Lee, “New constructions of revocable identity-based encryption from multilinear maps,” *TIFS*, 2015.

[54] K. G. Paterson and J. C. N. Schuldt, “Efficient identity-based signatures secure in the standard model,” in *ACISP*, 2006.

[55] M. Pirretti, P. Traynor, P. McDaniel, and B. Waters, “Secure attribute-based systems,” in *CCS*, ser. CCS ’06. ACM, 2006, pp. 99–112.

[56] T. Ring, “Cloud computing hit by celebgate,” http://www.scmagazineuk.com/cloud-computing-hit-by-celebgate/article/370815/, 2015.

[57] A. Sahai and H. Seyalioglu, “Worry-free encryption: functional encryption with public keys,” in *CCS*, 2010.

[58] A. Sahai, H. Seyalioglu, and B. Waters, “Dynamic credentials and ciphertext delegation for attribute-based encryption,” in *CRYPTO*, 2012.

[59] A. Sahai and B. Waters, “Fuzzy identity-based encryption,” in *EUROCRYPT*, 2005.

[60] J. H. Saltzer and M. D. Schroeder, “The protection of information in computer systems,” *Proceedings of the IEEE*, vol. 63, no. 9, 1975.

[61] R. S. Sandhu, “Rationale for the RBAC96 family of access control models,” in *ACM Workshop on RBAC*, 1995.

[62] J. H. Seo and K. Emura, “Revocable identity-based encryption revisited: Security model and construction,” in *PKC*, 2013.

[63] S. Sinclair, S. W. Smith, S. Trudeau, M. E. Johnson, and A. Portera, “Information risk in financial institutions: Field study and research roadmap,” in *FinanceCom*, 2007.

[64] J. G. Steiner, B. C. Neuman, and J. I. Schiller, “Kerberos: An authentication service for open network systems,” in *USENIX Winter Conference*, 1988.

[65] S. D. Stoller, P. Yang, C. R. Ramakrishnan, and M. I. Gofman, “Efficient policy analysis for administrative role based access control,” in *CCS*, 2007.

Appendix A
IBE/IBS Proof

We first provide a formal definition of an access control system that uses IBE, IBS, and symmetric-key cryptography, and then show it implements RBAC_0, proving Theorem 1. The proof of Theorem 2 is analogous, replacing IBE/IBS with public-key equivalents.

A. Our IBE/IBS System

1) Preliminaries:

- We use m as the symmetric-key size, which is also the size of the IBE and IBS message spaces.
- For signatures, we assume that hash-and-sign is used, where the message is hashed with a collision-resistant hash function and then signed using IBS.

2) States:

- USERS: a list of user names
- ROLES: a list of (r, v_r) pairs containing role names and version numbers
- FILES: a list of (fn, v_{fn}) pairs containing file names and version numbers
- FS: the set of tuples (RK, FK, or F) stored on the filestore

3) Request:

- u, p for whether user u has permission p

4) Queries:

- RK returns whether a user is in a role. Note that we do not verify the validity of the encrypted keys because the encryption is performed by the trusted admin, and the signature ensures integrity.

$$\begin{aligned} RK(u,r) \triangleq \exists(c, sig).(&\langle \mathsf{RK}, u, (r, v_r), c, sig\rangle \in FS \\ &\wedge sig = \mathbf{Sign}^{\mathbf{IBS}}_{SU}(\langle \mathsf{RK}, u, (r, v_r), c\rangle)) \end{aligned}$$

  Checking RK requires one instance of $\mathbf{Ver}^{\mathbf{IBS}}$.

- FK returns whether a role has a permission for the latest version of a file. As is the case RK, we do not need to verify the validity of the encrypted key.

$$\begin{aligned} FK(r, \langle fn, op\rangle) \triangleq\ &\exists(c, sig).(\\ &\langle \mathsf{FK}, r, \langle fn, op\rangle, v_{fn}, c, SU, sig\rangle \in F \\ &\wedge sig = \mathbf{Sign}^{\mathbf{IBS}}_{SU}(\langle \mathsf{FK}, r, \langle fn, op\rangle, v_r, c, SU\rangle)) \end{aligned}$$

  Checking FK requires one instance of $\mathbf{Ver}^{\mathbf{IBS}}$.

- $Role(r) \triangleq \exists v.((r, v) \in ROLES)$
- $auth$ returns whether a user has a permission.

$$auth(u,p) \triangleq \exists r.(RK(u,r) \wedge FK(r,p))$$

  Checking $auth$ requires two instances of $\mathbf{Ver}^{\mathbf{IBS}}$.

5) Labels: The labels used in this system are simply the operations in Fig. 2.

B. Implementing RBAC_0 *using IBE/IBS*

Because our IBE/IBS system uses versioning to handle revocation, assigning and then revoking a user/permission will not result in the same state as if the user/permission were never assigned. However, it will result in the same set of users having access to the latest versions of the same files, so the results of authorization requests will not be changed. We consider such states, which are equal except for differences in versioning, to be *congruent*, and represent this with the $\cong$ relation. We also say that state mappings σ and σ' are congruent if $\sigma(x) \cong \sigma'(x)$ for all states x.

The definition of correctness from [36] requires that α preserves σ, which means the following: For all $n \in \mathbb{N}$, states x_0, and labels $\ell_1, \ldots, \ell_n$, let $y_0 = \sigma(x_0)$, $x_i = next(x_{i-1}, \ell_i)$ for $i = 1, \ldots, n$, and $y_i = terminal(y_{i-1}, \alpha(y_{i-1}, \ell_i))$ for $i = 1, \ldots, n$. Then α preserves σ means that $y_i = \sigma(x_i)$ for all $i = 1, \ldots, n$.

We cannot achieve this in our system because of version numbers, e.g., if ℓ_1 assigns a user to a role and then ℓ_2 revokes that user from the role, x_2 will be equal to x_0 (and thus $\sigma(x_2)$ will be equal to $\sigma(x_0)$), but y_2 will have version numbers different from y_0. Thus instead we will show that $y_i \cong \sigma(x_i)$ for all $i = 1, \ldots, n$, which we define as α *congruence-preserves* σ.

In [27], α preserves σ is defined as

$$\sigma\Big(next(x, \ell)\Big) = terminal\Big(\sigma(x), \alpha\big(\sigma(x), \ell\big)\Big) \qquad (1)$$

for every state x and label ℓ. This implies the definition from [36] by the following inductive argument:

Proof: By definition, $y_0 = \sigma(x_0)$. Now assume that $y_i = \sigma(x_i)$. Then by Eq. (1),

$$\begin{aligned} y_{i+1} &= terminal\Big(y_i, \alpha\big(y_i, \ell_{i+1}\big)\Big) \\ &= terminal\Big(\sigma(x_i), \alpha\big(\sigma(x_i), \ell_{i+1}\big)\Big) \\ &= \sigma\Big(next\big(x_i, \ell_{i+1}\big)\Big) = \sigma\big(x_{i+1}\big). \end{aligned}$$

■

However, an analogous proof with congruence instead of equality does not work because we cannot substitute $\sigma(x_i)$ for y_i if they are not equal. Thus

$$\sigma\Big(next(x, \ell)\Big) \cong terminal\Big(\sigma(x), \alpha\big(\sigma(x), \ell\big)\Big)$$

does not imply that α congruence-preserves σ. This may occur, for instance, if one of the IBE/IBS labels does not work correctly when multiple versions of a file are present.

Instead we will show that

$$\sigma'\Big(next(x, \ell)\Big) \cong terminal\Big(\sigma'(x), \alpha\big(\sigma'(x), \ell\big)\Big) \qquad (2)$$

for all states x, labels ℓ, and state mappings σ' congruent to σ. This proves that α congruence-preserves σ by the following inductive argument:

Proof: By definition, $y_0 \cong \sigma(x_0)$. Now assume that $y_i \cong \sigma(x_i)$. Let σ^* be the state mapping equivalent to σ except that $\sigma^*(x_i) = y_i$. Since $\sigma^*(x) = \sigma(x)$ for all $x \neq x_i$ and $\sigma^*(x_i) \cong \sigma(x_i)$, $\sigma^* \cong \sigma$. Thus by Eq. (2),

$$\begin{aligned} y_{i+1} &= \mathit{terminal}\Big(y_i, \alpha\big(y_i, \ell_{i+1}\big)\Big) \\ &= \mathit{terminal}\Big(\sigma^*(x_i), \alpha\big(\sigma^*(x_i), \ell_{i+1}\big)\Big) \\ &\cong \sigma^*\Big(\mathit{next}\big(x_i, \ell_{i+1}\big)\Big) = \sigma^*\big(x_{i+1}\big) \cong \sigma\big(x_{i+1}\big). \end{aligned}$$

■

If we have an implementation $\langle \alpha, \sigma, \pi \rangle$ such that α congruence-preserves σ and σ preserves π, we say that the implementation is *congruence-correct.*

Theorem 3: There exists an implementation $\langle \alpha, \sigma, \pi \rangle$ of RBAC_0 using IBE and IBS where:

- α congruence-preserves σ and preserves safety
- σ preserves π
- π is AC-preserving

Thus there exists a congruence-correct, AC-preserving, safe implementation of RBAC_0 using IBE and IBS.

Proof:

The notation and conventions used here are listed in Section IV-C2.

1) State mapping σ:

For each $u \in U \cup \{SU\}$:

- Add u to USERS.
- Generate $k_u \leftarrow \mathbf{KeyGen}^{\mathbf{IBE}}(u)$ and $s_u \leftarrow \mathbf{KeyGen}^{\mathbf{IBS}}(u)$.

Let $FS = \{\}$.

Let ROLES and FILES be blank.

Run $\mathbf{MSKGen}^{\mathbf{IBE}}(m)$ to get IBE system parameters and master secret key msk.

Run $\mathbf{MSKGen}^{\mathbf{IBS}}(m)$ to get IBS system parameters and master secret key msk'.

For each $R(r) \in M$:

- Add $(r, 1)$ to ROLES.
- Let $FS = FS \cup \{\langle \mathsf{RK}, SU, (r, 1), \mathbf{Enc}^{\mathbf{IBE}}_{SU}\Big(\mathbf{KeyGen}^{\mathbf{IBE}}_{msk}((r, 1)), \mathbf{KeyGen}^{\mathbf{IBS}}_{msk'}((r, 1))\Big), \mathbf{Sign}^{\mathbf{IBS}}_{SU} \rangle\}$.

For each $P(fn) \in M$ where fn is the name of file f:

- Add $(fn, 1)$ to FILES.
- Produce a symmetric key $k = \mathbf{Gen}^{\mathbf{Sym}}(m)$.
- Let $FS = FS \cup \{\langle \mathsf{F}, fn, 1, \mathbf{Enc}^{\mathbf{Sym}}_k(f), SU, \mathbf{Sign}^{\mathbf{IBS}}_{SU} \rangle\}$.
- Let $FS = FS \cup \{\langle \mathsf{FK}, SU, \langle fn, \mathsf{RW} \rangle, 1, \mathbf{Enc}^{\mathbf{IBE}}_{SU}(k), SU, \mathbf{Sign}^{\mathbf{IBS}}_{SU} \rangle\}$.

For each $UR(u, r) \in M$:

- Find $\langle \mathsf{RK}, SU, (r, 1), c, sig \rangle \in FS$.
- Let $FS = FS \cup \{\langle \mathsf{RK}, SU, (r, 1), \mathbf{Enc}^{\mathbf{IBE}}_u\Big(\mathbf{Dec}^{\mathbf{IBE}}_{k_{SU}}(c)\Big), \mathbf{Sign}^{\mathbf{IBS}}_{SU} \rangle\}$.

For each $PA(r, \langle fn, op \rangle)$:

- Find $\langle \mathsf{FK}, SU, \langle fn, \mathsf{RW} \rangle, 1, c, SU, sig \rangle$.
- Let $FS = FS \cup \{\langle \mathsf{FK}, (r, 1), \langle fn, op \rangle, 1, \mathbf{Enc}^{\mathbf{IBE}}_{(r,1)}\Big(\mathbf{Dec}^{\mathbf{IBE}}_{k_{SU}}(c)\Big), SU, \mathbf{Sign}^{\mathbf{IBS}}_{SU} \rangle\}$.

$\mathit{output}(FS, \text{ROLES}, \text{FILES})$

2) Query mapping π:

$$\begin{aligned} \pi_{UR(u,r)}(T) &= RK(u, r) \in T \\ \pi_{PA(r,p)}(T) &= FK(r, p) \in T \\ \pi_{R(r)}(T) &= \mathit{Role}(r) \in T \\ \pi_{auth(u,p)}(T) &= \mathit{auth}(u, p) \in T \end{aligned}$$

The query mapping π is AC-preserving because it maps $\mathit{auth}(u, p)$ to TRUE for theory T if and only if T contains $\mathit{auth}(u, p)$.

3) σ preserves π: This means that for every $RBAC_0$ state x, $Th(x) = \pi(Th(\sigma(x)))$. To prove this, we show that for each $RBAC_0$ state x and query q, $x \vdash q$ if and only if $\pi_q(Th(\sigma(x))) = \text{TRUE}$.

We consider each type of query separately.

- **UR:** If $x \vdash UR(u, r)$ then $UR(u, r) \in Th(x)$, meaning that in x, $\langle u, r \rangle \in UR$. Thus in $\sigma(x)$, $v_r = 1$ and $\exists (c, sig).(\langle \mathsf{RK}, u, (r, 1), c, sig \rangle \in FS \wedge sig = \mathbf{Sign}^{\mathbf{IBS}}_{SU}(\langle \mathsf{RK}, u, (r, v_r), c \rangle))$. Hence $RK(u, r) \in Th(\sigma(x))$, so $\pi_{UR(u,r)}(Th(\sigma(x))) = \text{TRUE}$.
  If $x \nvdash UR(u, r)$ then $UR(u, r) \notin Th(x)$, meaning that in x, $\langle u, r \rangle \notin UR$. Thus in $\sigma(x)$, $v_r = 1$ and $\nexists (c, sig).(\langle \mathsf{RK}, u, (r, 1), c, sig \rangle \in FS)$. Hence $RK(u, r) \notin Th(\sigma(x))$, so $\pi_{UR(u,r)}(Th(\sigma(x))) = \text{FALSE}$.
- **PA:** If $x \vdash PA(r, p)$ with $p = \langle fn, op \rangle$, then $PA(r, p) \in Th(x)$, meaning that in x, $\langle r, p \rangle \in PA$. Thus in $\sigma(x)$, $v_{fn} = 1$ and $\exists (c, sig).(\langle \mathsf{FK}, r, \langle fn, op \rangle, v_{fn}, c, SU, sig \rangle \in FS \wedge sig = \mathbf{Sign}^{\mathbf{IBS}}_{SU}(\langle \mathsf{FK}, r, \langle fn, op \rangle, v_{fn}, c, SU \rangle))$. Hence $FK(r, p) \in Th(\sigma(x))$, so $\pi_{PA(r,p)}(Th(\sigma(x))) = \text{TRUE}$.
  If $x \nvdash PA(r, p)$ with $p = \langle fn, op \rangle$, then $PA(r, p) \notin Th(x)$, meaning that in x, $\langle r, p \rangle \notin PA$. Thus in $\sigma(x)$, $v_{fn} = 1$ and $\nexists (c, sig).(\langle \mathsf{FK}, r, \langle fn, op \rangle, v_{fn}, c, SU, sig \rangle \in FS)$. Hence $FK(r, p) \notin Th(\sigma(x))$, so $\pi_{PA(r,p)}(Th(\sigma(x))) = \text{FALSE}$.
- **R:** If $x \vdash R(r)$ then $R(r) \in Th(x)$, meaning that in x, $r \in R$. Thus in $\sigma(x)$, $(r, 1) \in$ ROLES. Hence $\mathit{Role}(r) \in Th(\sigma(x))$, so $\pi_{R(r)}(Th(\sigma(x))) = \text{TRUE}$.
  If $x \nvdash R(r)$, then $R(r) \notin Th(x)$, meaning that in x, $r \notin R$. Thus in $\sigma(x)$, $\nexists v.((r, v) \in \text{ROLES})$. Hence $\mathit{Role}(r) \notin Th(\sigma(x))$, so $\pi_{R(r)}(Th(\sigma(x))) = \text{FALSE}$.
- **auth:** If $x \vdash \mathit{auth}(u, p)$ then $\mathit{auth}(u, p) \in Th(x)$, so there exists r such that $UR(u, r) \in Th(x) \wedge PA(r, p) \in Th(x)$. Since σ preserves π for UR and PA queries, $RK(u, r) \in Th(\sigma(x)) \wedge FK(r, p) \in Th(\sigma(x))$. Hence $\mathit{auth}(u, p) \in Th(\sigma(x))$, so $\pi_{auth(u,p)}(Th(\sigma(x))) = \text{TRUE}$.
  If $x \nvdash \mathit{auth}(u, p)$ then $\mathit{auth}(u, p) \notin Th(x)$, so $\nexists r.(UR(u, r) \in Th(x) \wedge PA(r, p) \in Th(x))$. Since σ preserves π for UR and PA queries, $\nexists r.(RK(u, r) \in Th(\sigma(x)) \wedge FK(r, p) \in Th(\sigma(x)))$. Hence $\mathit{auth}(u, p) \notin Th(\sigma(x))$, so $\pi_{auth(u,p)}(Th(\sigma(x))) = \text{FALSE}$.

4) Label mapping α: The label mapping α simply maps any RBAC_0 label, regardless of the state, to the IBE/IBS label of the same name found in Fig. 2. The only difference is that in IBE/IBS, $addP$ takes as input a filename and file instead of a permission and $delP$ takes as input a filename instead of a permission.

5) α congruence-preserves σ: We consider each type of $RBAC_0$ label separately. We let σ' be a state mapping congruent to σ and let $x' = next(x, \ell)$ be the result of executing label ℓ in state x. While key generation and encryption algorithms are normally randomized, for determining equality of states we assume that they are deterministic.

- **addU:** If ℓ is an instance of $addU(u)$, then $x' = x \cup U(u)$. Thus

$$\begin{aligned}\sigma'(x') = \sigma'\big(x \cup U(u)\big) &= \sigma'(x) \cup \mathrm{USERS}(u)\\ &= next\big(\sigma'(x), addU(u)\big)\\ &= terminal\big(\sigma'(x), \alpha(\sigma'(x), \ell)\big).\end{aligned}$$

- **delU:** If ℓ is an instance of $delU(u)$, then $x' = x \setminus (U(u) \cup \{UR(u,r) \mid UR(u,r) \in x\})$. Let $T = \{(r, c, sig) \mid \langle \mathsf{RK}, u, (r, v_r), c, sig\rangle \in FS\}$ and $T' = \{r \mid \exists(c, sig).((r, c, sig) \in T)\}$. Let $\{r_1, r_2, \ldots, r_n\}$ be the elements of T' in arbitrary order. Then

$$\begin{aligned}\sigma'(x') &= \sigma'\big(x \setminus (U(u) \cup \{UR(u,r) \mid UR(u,r) \in x\})\big)\\ &= \sigma'(x) \setminus \mathrm{USERS}(u)\\ &\quad \setminus \big\{FS\big(\langle \mathsf{RK}, u, (r, v_r), c, sig\rangle\big) \mid (r, c, sig) \in T\big\}\\ &\cong terminal\big(\sigma'(x) \setminus \mathrm{USERS}(u),\\ &\quad revokeU(u, r_1) \circ revokeU(u, r_2)\\ &\quad \circ \cdots \circ revokeU(u, r_n)\big)\\ &= next\big(\sigma'(x), delU(u)\big)\\ &= terminal\big(\sigma'(x), \alpha(\sigma'(x), \ell)\big).\end{aligned}$$

- **addR:** If ℓ is an instance of $addR(r)$, then $x' = x \cup R(r)$. Thus

$$\begin{aligned}\sigma'(x') &= \sigma'\big(x \cup R(r)\big)\\ &= \sigma'(x) \cup \mathrm{ROLES}(r, 1) \cup FS\Big(\Big\langle \mathsf{RK}, SU, (r, 1),\\ &\quad \mathbf{Enc}^{\mathbf{IBE}}_{SU}\Big(\mathbf{KeyGen}^{\mathbf{IBE}}_{msk}((r, 1)),\\ &\quad \mathbf{KeyGen}^{\mathbf{IBS}}_{msk'}((r, 1))\Big), \mathbf{Sign}^{\mathbf{IBS}}_{SU}\Big\rangle\Big)\\ &= next\big(\sigma'(x), addR(r)\big)\\ &= terminal\big(\sigma'(x), \alpha(\sigma'(x), \ell)\big).\end{aligned}$$

- **delR:** If ℓ is an instance of $delR(r)$, then $x' = x \setminus (R(r) \cup \{UR(u, r) \mid UR(u, r) \in x\} \cup \{PA(r, p) \mid PA(r, p) \in x\})$. Let $T = \{(u, c, sig) \mid \langle \mathsf{RK}, u, (r, v_r), c, sig\rangle \in FS\}$ and $F = \{fn \mid \exists(op, v_{fn}, c_{fn}, sig).(\langle \mathsf{FK}, (r, v_r), \langle fn, op\rangle, v_{fn}, c_{fn}, SU, sig\rangle \in FS)\}$. For each $fn \in F$, let $T_{fn} = \{(op', v, c_v, sig) \mid \langle \mathsf{FK}, (r, v_r), \langle fn, op'\rangle, v, c_v, SU, sig\rangle \in FS\}$. Let $\{fn_1, fn_2, \ldots, fn_n\}$ be the elements of F in arbitrary order. Then

$$\begin{aligned}\sigma'(x') &= \sigma'\big(x \setminus (R(r) \cup \{UR(u, r) \mid UR(u, r) \in x\}\\ &\qquad \cup \{PA(r, p) \mid PA(r, p) \in x\}\big)\\ &= \sigma'(x) \setminus ROLES(r, v_r) \setminus \big\{FS\big(\langle \mathsf{RK}, u, (r, v_r),\\ &\qquad c, sig\rangle\big) \mid (u, c, sig) \in T\big\} \setminus \big\{FS\big(\langle \mathsf{FK}, (r, v_r),\\ &\qquad \langle fn, op'\rangle, v, c_v, SU, sig\rangle\big) \mid \big(fn \in F\\ &\qquad \wedge (op', v, c_v, sig) \in T_{fn}\big)\big\}\\ &\cong terminal\Big(\sigma'(x) \setminus ROLES(r, v_r)\\ &\quad \setminus \big\{FS\big(\langle \mathsf{RK}, u, (r, v_r), c, sig\rangle\big) \mid (u, c, sig) \in T\big\},\\ &\qquad revokeP\big(r, \langle fn_1, \mathsf{RW}\rangle\big) \circ revokeP\big(r, \langle fn_2, \mathsf{RW}\rangle\big)\\ &\qquad \circ \cdots \circ revokeP\big(r, \langle fn_n, \mathsf{RW}\rangle\big)\Big)\\ &= next\big(\sigma'(x), delR(r)\big)\\ &= terminal\big(\sigma'(x), \alpha(\sigma'(x), \ell)\big).\end{aligned}$$

- **addP:** If ℓ is an instance of $addP(p)$ with $p = \langle fn, op\rangle$ and fn the name of file f, then $x' = x \cup P(p)$. Thus for $k \leftarrow \mathbf{Gen}^{\mathbf{Sym}}(m)$,

$$\begin{aligned}\sigma'(x') &= \sigma'\big(x \cup P(p)\big)\\ &= \sigma'(x) \cup \mathrm{FILES}(fn, 1)\\ &\quad \cup FS\Big(\Big\langle \mathsf{F}, fn, 1, \mathbf{Enc}^{\mathbf{Sym}}_{k}(f)\Big\rangle\Big) \cup FS\Big(\Big\langle \mathsf{FK}, SU,\\ &\quad \langle fn, \mathsf{RW}\rangle, 1, \mathbf{Enc}^{\mathbf{IBE}}_{SU}(k), SU, \mathbf{Sign}^{\mathbf{IBS}}_{SU}\Big\rangle\Big)\\ &= next\big(\sigma'(x), addP(fn, f)\big)\\ &= terminal\big(\sigma'(x), \alpha(\sigma'(x), \ell)\big).\end{aligned}$$

- **delP:** If ℓ is an instance of $delP(p)$ with $p = \langle fn, op\rangle$, then $x' = x \setminus (P(p) \cup \{PA(r, p) \mid PA(r, p) \in x\})$. Let $T = \{(v, c) \mid \langle \mathsf{F},\ fn,\ v,\ c\rangle \in FS\}$ and $T' = \{(r, op', v, c', id, sig) \mid \langle \mathsf{FK}, r, \langle fn, op'\rangle, v, c', id, sig\rangle \in FS\}$. Then

$$\begin{aligned}\sigma'(x') &= \sigma'\big(x \setminus (P(p) \cup \{PA(r, p) \mid PA(r, p) \in x\})\big)\\ &= \sigma'(x) \setminus \mathrm{FILES}(fn, v_{fn})\\ &\quad \setminus \big\{FS\big(\langle \mathsf{F}, fn, v, c\rangle\big) \mid (v, c) \in T\big\}\\ &\quad \setminus \big\{FS\big(\langle \mathsf{FK}, r, \langle fn, op'\rangle, v, c', id, sig\rangle\big) \mid (r, op',\\ &\qquad v, c', id, sig) \in T\big\}\\ &= next\big(\sigma'(x), delP(fn)\big)\\ &= terminal\big(\sigma'(x), \alpha(\sigma'(x), \ell)\big).\end{aligned}$$

- **assignU:** If ℓ is an instance of $assignU(u, r)$, then $x' = x \cup UR(u, r)$. Thus for $\langle \mathsf{RK}, SU, (r, 1), c, sig\rangle \in FS$ in $\sigma'(x)$,

$$\begin{aligned}\sigma'(x') &= \sigma'\big(x \cup UR(u, r)\big)\\ &= \sigma'(x) \cup FS\Big(\Big\langle \mathsf{RK}, SU, (r, 1),\\ &\quad \mathbf{Enc}^{\mathbf{IBE}}_{u}\left(\mathbf{Dec}^{\mathbf{IBE}}_{k_{SU}}(c)\right), \mathbf{Sign}^{\mathbf{IBS}}_{SU}\Big\rangle\Big)\\ &= next\big(\sigma'(x), assignU(u, r)\big)\\ &= terminal\big(\sigma'(x), \alpha(\sigma'(x), \ell)\big).\end{aligned}$$

- **revokeU:** If ℓ is an instance of $revokeUser(u, r)$, then $x' = x \setminus UR(u, r)$. Let $k_{(r, v_r+1)} \leftarrow \mathbf{KeyGen}^{\mathbf{IBE}}((r, v_r + 1))$ and $s_{(r, v_r+1)} \leftarrow \mathbf{KeyGen}^{\mathbf{IBS}}((r, v_r + 1))$. Let $T = \{(u', c_{u'}, sig) \mid \langle \mathsf{RK},\ u',\ (r, v_r),\ c_{u'},\ sig\rangle \in FS\}$ and

$F = \{fn \mid \exists(op, v_{fn}, c_{fn}, sig).(\langle \mathsf{FK}, (r, v_r), \langle fn, op\rangle, v_{fn}, c_{fn}, SU, sig\rangle \in FS)$. For each $fn \in F$, let $k_{fn} \leftarrow \mathbf{Gen}^{\mathbf{Sym}}$, $T_{fn} = \{(op', v, c_v, sig) \mid \langle \mathsf{FK}, (r, v_r), \langle fn, op'\rangle, v, c_v, SU, sig\rangle \in FS\}$ and $T'_{fn} = \{id, op', c_{id}, sig) \mid \langle \mathsf{FK}, id, \langle fn, op'\rangle, v_{fn}, c_{id}, SU, sig\rangle \in FS\}$. Then

$$\begin{aligned}
\sigma'(x') &= \sigma'\big(x \setminus UR(u,r)\big)\\
&= \sigma'(x) \setminus \big\{FS\big(\langle \mathsf{RK}, u, (r, v_r), c_u,\\
&\qquad sig\rangle\big) \mid (u, c_u, sig) \in T\big\}\\
&\cong \sigma'(x) \setminus \big\{FS\big(\langle \mathsf{RK}, u', (r, v_r), c_{u'},\\
&\qquad sig\rangle\big) \mid (u', c_{u'}, sig) \in T\big\}\\
&\quad \cup \Big\{FS\Big(\Big\langle \mathsf{RK}, u', (r, v_r), \mathbf{Enc}^{\mathbf{IBE}}_{u'}\left(k_{(r,v_r+1)}, s_{(r,v_r+1)}\right),\\
&\qquad \mathbf{Sign}^{\mathbf{IBS}}_{SU}\Big\rangle\Big) \mid (u', c_{u'}, sig) \in T \wedge u' \neq u\Big\}\\
&\quad \setminus \big\{FS\big(\langle \mathsf{FK}, (r, v_r), \langle fn, op'\rangle, v, c_v,\\
&\qquad SU, sig\rangle\big) \mid fn \in F \wedge (op', v, c_v, sig) \in T_{fn}\big\}\\
&\quad \cup \Big\{FS\Big(\Big\langle \mathsf{FK}, (r, v_r+1), \langle fn, op'\rangle, v,\\
&\qquad \mathbf{Enc}^{\mathbf{IBE}}_{(r,v_r+1)}\left(\mathbf{Dec}^{\mathbf{IBE}}_{k_{(r,v_r)}}(c_v)\right), SU,\\
&\qquad \mathbf{Sign}^{\mathbf{IBS}}_{SU}\Big\rangle\Big) \mid fn \in F \wedge (op', v, c_v, sig) \in T_{fn}\Big\}\\
&\quad \cup \Big\{FS\Big(\Big\langle \mathsf{FK}, id, \langle fn, op'\rangle, v_{fn}+1, \mathbf{Enc}^{\mathbf{IBE}}_{id}(k'_p),\\
&\qquad SU, \mathbf{Sign}^{\mathbf{IBS}}_{SU}\Big\rangle\Big) \mid fn \in F \wedge (id, c_{id}, sig) \in T'_{fn}\Big\}\\
&\quad \cup \big\{\mathrm{FILES}(fn, v_{fn}+1) \mid fn \in F\big\}\\
&\quad \setminus \big\{\mathrm{FILES}(fn, v_{fn}) \mid fn \in F\big\}\\
&\quad \cup \mathrm{ROLES}(r, v_r+1) \setminus \mathrm{ROLES}(r, v_r)\\
&= next\big(\sigma'(x), revokeU(u,r)\big)\\
&= terminal\big(\sigma'(x), \alpha(\sigma'(x), \ell)\big).
\end{aligned}$$

- **assignP:** If ℓ is an instance of $assignP(r,p)$ with $p = \langle fn, op\rangle$, then $x' = x \cup PA(r,p)$. We have two cases where $assignP(r,p)$ has an effect on x:
  - If $op = \mathsf{RW}$ and there exists $\langle \mathsf{FK}, (r, v_r), \langle fn, \mathsf{Read}\rangle, v_{fn}, c, SU, sig\rangle$, then let $T = \{(v, c_v, sig) \mid \langle \mathsf{FK}, (r, v_r), \langle fn, \mathsf{Read}\rangle, v, c_v, SU, sig\rangle \in FS\}$. Then

$$\begin{aligned}
\sigma'(x') &= \sigma'\big(x \cup PA(r,p)\big)\\
&= \sigma'(x) \setminus \big\{FS\big(\langle \mathsf{FK}, (r, v_r), \langle fn, \mathsf{Read}\rangle, v, c_v,\\
&\qquad SU, sig\rangle\big) \mid (v, c_v, sig) \in T\big\}\\
&\quad \cup \big\{FS\Big(\Big\langle \mathsf{FK}, (r, v_r), \langle fn, \mathsf{RW}\rangle, v, c_v, SU,\\
&\qquad \mathbf{Sign}^{\mathbf{IBS}}_{SU}\Big\rangle\Big) \mid (v, c_v, sig) \in T\big\}\\
&= next\big(\sigma'(x), assignP(r,p)\big)\\
&= terminal\big(\sigma'(x), \alpha(\sigma'(x), \ell)\big).
\end{aligned}$$

  - If there does not exist $\langle \mathsf{FK}, (r, v_r), \langle fn, op'\rangle, v_{fn}, c, SU, sig\rangle$, then let $T = \{(v, c_v) \mid \exists(id, sig).(\langle \mathsf{FK}, SU, \langle fn, \mathsf{RW}\rangle, v, c_v, id, sig\rangle \in FS)\}$. Then

$$\begin{aligned}
\sigma'(x') &= \sigma'\big(x \cup PA(r,p)\big)\\
&= \sigma'(x) \cup \Big\{FS\Big(\Big\langle \mathsf{FK}, (r, v_r), \langle fn, op\rangle, v,\\
&\qquad \mathbf{Enc}^{\mathbf{IBE}}_{(r,v_r)}\left(\mathbf{Dec}^{\mathbf{IBE}}_{k_{SU}}(c_v)\right), SU,\\
&\qquad \mathbf{Sign}^{\mathbf{IBS}}_{SU}\Big\rangle\Big) \mid (v, c_v) \in T\Big\}\\
&= next\big(\sigma'(x), assignP(r,p)\big)\\
&= terminal\big(\sigma'(x), \alpha(\sigma'(x), \ell)\big).
\end{aligned}$$

- **revokeP:** If ℓ is an instance of $revokeP(r,p)$ with $p = \langle fn, op\rangle$, then $x' = x \setminus PA(r,p)$.
  - If $op = \mathsf{Write}$, then let $T = \{(v, c_v, sig) \mid \langle \mathsf{FK}, (r, v_r), \langle fn, \mathsf{RW}\rangle, v, c_v, SU, sig\rangle \in FS\}$. Then

$$\begin{aligned}
\sigma'(x') &= \sigma'\big(x \setminus PA(r,p)\big)\\
&= \sigma'(x) \setminus \big\{FS\big(\langle \mathsf{FK}, (r, v_r), \langle fn, \mathsf{RW}\rangle, v, c_v,\\
&\qquad SU, sig\rangle\big) \mid (v, c_v, sig) \in T\big\}\\
&\quad \cup \Big\{FS\Big(\Big\langle \mathsf{FK}, (r, v_r), \langle fn, \mathsf{Read}\rangle, v, c_v, SU,\\
&\qquad \mathbf{Sign}^{\mathbf{IBS}}_{SU}\Big\rangle\Big) \mid (v, c_v, sig) \in T\Big\}\\
&= next\big(\sigma'(x), assignP(r,p)\big)\\
&= terminal\big(\sigma'(x), \alpha(\sigma'(x), \ell)\big).
\end{aligned}$$

  - If $op = \mathsf{Read}$, then let $k' \leftarrow \mathbf{Gen}^{\mathbf{Sym}}$, $T = \{(op', v, c_v, sig) \mid \langle \mathsf{FK}, (r, v_r), \langle fn, op'\rangle, v, c_v, SU, sig\rangle \in FS\}$, and $T' = \{(id, op') \mid id \neq r \wedge \exists(c_{id}, sig).(\langle \mathsf{FK}, id, \langle fn, op'\rangle, v_{fn}, c_{id}, SU, sig\rangle \in FS)\}$. Then

$$\begin{aligned}
\sigma'(x') &= \sigma'\big(x \setminus PA(r,p)\big)\\
&= \sigma'(x) \setminus \big\{FS\big(\langle \mathsf{FK}, (r, v_r), \langle fn, op'\rangle, v, c_v,\\
&\qquad SU, sig\rangle\big) \mid (op', v, c_v, sig) \in T\big\}\\
&\cong \sigma'(x) \setminus \big\{FS\big(\langle \mathsf{FK}, (r, v_r), \langle fn, op'\rangle, v, c_v,\\
&\qquad SU, sig\rangle\big) \mid (op', v, c_v, sig) \in T\big\}\\
&\quad \cup \Big\{FS\Big(\Big\langle \mathsf{FK}, id, \langle fn, op'\rangle, v_{fn}+1,\\
&\qquad \mathbf{Enc}^{\mathbf{IBE}}_{id}(k'), SU, \mathbf{Sign}^{\mathbf{IBS}}_{SU}\Big\rangle\Big) \mid (id, op') \in T\Big\}\\
&\quad \cup \mathrm{FILES}(fn, v_{fn}+1) \setminus \mathrm{FILES}(fn, v_{fn})\\
&= next\big(\sigma'(x), assignP(r,p)\big)\\
&= terminal\big(\sigma'(x), \alpha(\sigma'(x), \ell)\big).
\end{aligned}$$

6) Safety: The label mapping α is safe by inspection—for any RBAC_0 state x and label ℓ, the IBE/IBS label $\alpha(\sigma(x), \ell)$ never revokes or grants authorizations except the images of those that are revoked or granted by ℓ. ■