D

Dynamic Access Control Using Identity-Based Encryption

William C. Garrison III and Adam J. Lee
Department of Computer Science, University of Pittsburgh, Pittsburgh, PA, USA

Synonyms

Cryptographically enforced dynamic access control; Practical revocation in cryptographic access control; Private-key constructions for dynamic access control

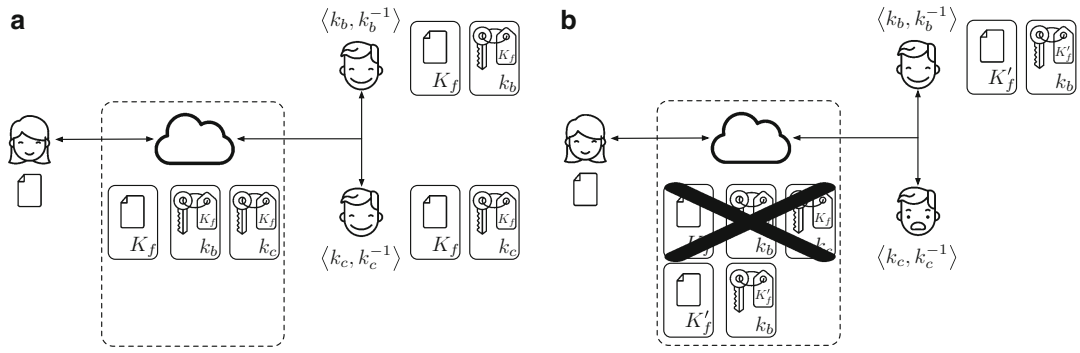
Definition

Access control describes the process of restricting the set of resources (*objects*) that some requesting entity (*subject*) is permitted to access. The information stored by an access control system for the purposes of making access decisions is referred to as the *protection state* of the access control system. While simple at its core, the modeling and analysis of access control systems is complicated when this protection state is *dynamic*: i.e., when policies can be altered by administrative users or when subjects and objects can be added to or removed from the system. As we will discuss, the state transitions induced by dynamism can add substantial complexity to both formal analyses of access control systems and the efficient enforcement of access controls using cryptographic constructions.

Background

Let us consider the enforcement of access controls using cryptographic constructions, in order to reduce the trust that must be placed in a software-based gatekeeper that otherwise makes all access decisions (e.g., a cloud storage provider). Assume that the service provider is *honest-but-curious*: it is trusted to carry out the protocols correctly but will try to eavesdrop passively on the file contents. In essence, one can prevent this eavesdropping by encrypting files and distributing file keys only to those subjects that should be granted access (compared to *capability-based* access control systems). To grant access to multiple users at once, and simplify distribution of file keys, we can share multiple copies of the file key, one encrypted to each permitted user (see Fig. 1a). When the need arises to revoke one of the users' access, though, we must consider that this user may still have a copy of the file key. This may not be an issue in the short term (since they may also have cached a copy of the file itself) but will certainly be undesirable as the file contents change. To revoke more thoroughly, we must rekey the file: generate a new key, decrypt and re-encrypt the file, and distribute the new key to all remaining permitted users (see Fig. 1b). At first glance, this may seem

S. Jajodia et al. (eds.), *Encyclopedia of Cryptography, Security and Privacy*,
https://doi.org/10.1007/978-3-642-27739-9_1452-1



Dynamic Access Control Using Identity-Based Encryption, Fig. 1 Cryptographically sharing a file using hybrid cryptography. (**a**) Sharing with two users. (**b**) Sharing with one user, after a revocation

like a satisfactory solution. Unfortunately, many issues present themselves upon closer inspection.

First, note that most deployed access control systems are much more expressive than our simple example construction, which grants access to files individually. Consider the classical role-based access control (RBAC) scheme, which relies on a level of indirection (*roles*) to reduce the administrative overheads of granting multiple permissions that are often assigned together (e.g., access to the files associated with some job function). In RBAC, subjects are assigned to roles, and permissions over objects are granted to roles. Subject s is granted access to file f if and only if there exists a role r such that s is assigned to r and access to f is granted to r. Consider the convenience of using roles as another level of *cryptographic* indirection, much like we already use to distribute file keys. For instance, generate an IBE key per role. For each file, f_i, granted to roles $r_1, \ldots, r_k$, we can share f_i's file key, K_{f_i}, with each of the roles ($\{K_{f_i}\}K_{r_i}$, where K_{r_i} is a public key for role r_i), and share copies of the role keys with each user ($\{K_{r_i}^{-1}\}K_u$ for subject u, if u is a member of r_i). This allows convenience when granting many permissions at once: adding a user to a role can be modeled by providing them with a single private key, granting a permission to a role can be accomplished by adding another copy of its file key, etc. These ideas generalize naturally to more expressive forms of private-key encryption, such as ABE, PE, etc.

Now, imagine an RBAC protection state with many subjects, roles, and files, including many assignments between them. Removing one subject, s, from a single role, r, could have many effects: s should lose access to all files, f_i, that were granted via r (note, however, that these files might also be granted to s via some other role r'). Similarly, revoking a single file, f, from role, r, will cause any subject s assigned to r to lose access to f (again, unless they have access through another role). As we can see, one small change in the protection state can change the result to many access queries. In fact, this is one of the most significant features of RBAC compared to simpler access control systems and one that is almost entirely ignored when evaluating cryptographic constructions for enforcing RBAC unless dynamism is explicitly considered.

Regarding the choice of cryptosystem, note that using entirely symmetric-key cryptography is untenable due to key distribution issues. Using IBE to distribute file keys (perhaps indirectly via role keys) is convenient, but using entirely private-key cryptography (including for encrypting files) is similarly impractical due to efficiency concerns. Thus, realistic constructions require the use of *hybrid cryptography*, with files encrypted using per-file symmetric keys, and these file keys are then encrypted for distribution using IBE (or other public-key cryptosystem). These additional details complicate both security analysis and implementation.

Next, consider the RBAC revocation scenarios presented above, recalling that revoking access to an encrypted resource is typically accomplished by *rekeying*: decrypting the file, re-encrypting with a new key, and distributing the new key to the remaining permitted subjects. This operation can be especially burdensome in the scenarios described above, as a single state transition (revocation of a role from a subject or a permission from a role) can cause many files to be rekeyed at once. If the service provider is not trusted to view file contents (usually the case when cryptographic constructions are used), naive constructions would also require that an administrator download each file, rekey, and re-upload a new ciphertext. Such complications affect both analysis and efficiency. Regarding the former, consider proving that a cryptographic construction never reveals a file's contents to any subject except those who would have access under a gatekeeper-enforced RBAC and the complexity of making such an argument when adversaries can cache keys from previously granted permissions. Regarding the latter, consider that a single "action" could require hundreds of files to be downloaded, decrypted, re-encrypted to new keys, and re-uploaded, all before distributing new keys to potentially many still-authorized users. We will now look at each of these tasks and their feasibility with practical cryptographic constructions used to enforce dynamic access control.

Security Analysis

Much work has been done in proving that an implementation of an access control system—perhaps cryptographic, perhaps using another access control system—is correct (i.e., it allows all accesses permitted by the policy that it is enforcing) and secure (i.e., it denies all accesses prohibited by the policy that it is enforcing). One approach for this type of analysis is a *simulation-based* one, where one access control system, S, is shown to be more expressive than another, T, if S can simulate each state, query, and transition in T. There are numerous forms that such a simulation can take (Garrison and Lee 2015), and the approach has been successfully extended to access control enforced by cryptographic constructions (Garrison et al. 2016).

An alternate approach borrows from the proof techniques typically used in the cryptographic literature, using the concept of multiplayer games assessing an adversary's ability to circumvent the controls imposed by the system. Ferrara et al. (2013) point out that, "few of the existing cryptographic access-control schemes come with precise guarantees, and the gap between the policy specification and the implementation being analyzed only informally, if at all." They thus define a cryptographic game where the adversary attempts to create a scenario in which they have access to a file, f, even though the RBAC policy being enforced symbolically would not allow such an access. Specifically, Ferrara et al. presented a cryptographic construction to enforce RBAC access controls faithfully. Intuitively, their proof imagines an adversary who can be given access to any role at any time, except for those that can access a target file, f. Then, viewing leaked (encrypted) files from the service provider, the adversary must guess whether the ciphertext file f represents plaintext file f_0 or f_1 (of the same length). If the adversary is unable to succeed (more often than chance), then the construction is said to have securely enforced the RBAC policy: despite being able to manipulate their own role membership nearly arbitrarily, the adversary is unable to exploit this to gain a permission that the underlying policy would not permit.

Proofs of either form can seem straightforward, perhaps even obvious, once observed. Showing that a construction that *symbolically* enforces a policy also *computationally* enforces it may seem an unnecessary step. However, it is extremely important that such proofs are written regarding cryptographic enforcement, without which no concrete guarantees can be made. Ferrara et al. argue that, "[In order] to present a rigorous approach to the analysis of cryptographic enforcement of access control... a formal definition of computational enforcement of a policy is an important (if not the most important) part." A similar point is made by Garrison et al. (2016) when tying changes in

the symbolic access control policy to changes in the computational feasibility of recovering a plaintext. Consider, for example, the following subtle scenario. An adversary joins, and then leaves, role r, which is granted no permissions. The user temporarily gains the role key for r, but since it is not used to encrypt symmetric keys for any files, this may seem harmless (notably, naive constructions may choose not to rekey anything, since no permissions were lost). However, later, if access to a file is granted to r, a copy of its file key will *then* be encrypted to role r, and the adversary may use a cached copy of the role key to access the file. Thus, a secure implementation must rekey role r even when doing so is seemingly free of effects (i.e., does not protect any current resources).

Despite the significant effort of both works, however, it becomes apparent that there is much to be done in proving the security of *practical* cryptographic constructions for enforcing *dynamic* access control. Note that the construction presented by Ferrara et al. (2013) does not utilize hybrid cryptography, modeling a system where files are encrypted using (expensive) private-key cryptography (specifically, predicate encryption as a generalization of attribute-based encryption). As we have discussed, this is crucial to practical implementation, as PE is infeasible for most scenarios with large files. Adding this additional level of indirection would no doubt complicate an already fragile proof.

Furthermore, rekeying is done in the naive way: to generate a new role key, a new role is effectively created (e.g., with a sequence number added to the name) and the new key is distributed to all remaining members. More principled solutions for revocation in modern private-key encryption have been proposed, including those using broadcast encryption (e.g., Park et al. 2015) and key homomorphism (e.g., Wang et al. 2016). Each adds additional complexity and security assumptions and would necessitate substantial further work to prove security to the level of rigor presented by Ferrara et al. (2013) and Garrison and Lee (2015). In fact, more practical (i.e., efficient) constructions, such as that presented by Qi and Zheng (2019), assume a significant weakening of the system model compared to that used in the aforementioned works, in that any single user can reveal a single key to the provider and allow the latter to read all files. Such subtle issues are indicative of the more wide-reaching requirement that cryptographic constructions are formally proved to enforce the desired policy, *within the precise desired threat model*.

Efficiency

In addition to the correctness and security of cryptographically enforced access controls, it is also important to consider the efficiency of these constructions. While protection of static states is straightforward, the previous section showed that much care must be taken to correctly enforce cryptographic access controls in an evolving, dynamic system. One example, presented by Garrison et al. (2016), attempts to quantify the costs of revocation within a cryptographically enforced RBAC system. Unlike Ferrara et al. (2013), which focused on proving security and largely ignored efficiency, Garrison et al. made many security concessions in an effort to provide a very conservative view of the costs of using these types of cryptographic access control constructions. Hybrid cryptography was utilized, with identity-based encryption (or, alternately, standard public-key encryption) used to encrypt per-file symmetric keys. Lazy revocation allowed a re-encryption to be deferred to the next write (which is not a vulnerability if it is assuming that a motivated attacker caches the files to which they have access).

Regardless of these concessions, the costs (in terms of the number of private-key encryptions and other metrics) of deploying such a construction were found to be intractable for most organizations. In a realistic scenario, for instance, revoking a single user from a single role could incur a cost of hundreds to thousands of IBE (or similar) encryptions (to distribute new "role keys"), with dozens to hundreds of files being marked for re-encryption. It was also necessary to trust the reference monitor to enforce write permissions (i.e., refuse to save a new version

of a file uploaded by a user without the relevant access permission), an assumption also made by Ferrara et al. (2013) and Qi and Zheng (2019). An alternative is identified that allows reading users to verify a write themselves, which requires only that the service provider can timestamp writes reliably. However, as with the additional efficiency tricks discussed above in *Security Analysis*, this adds complexity to any security analysis of the construction and modifies the system model in a nontrivial way. Using onion encryption layers (Qi and Zheng 2019) further improves efficiency by offloading re-encryption to the provider but also modifies the threat model as noted above.

Open Problems and Future Directions

The biggest open question in this space revolves around making cryptographic access control enforcement constructions more efficient without sacrificing the formally proved security of the existing constructions. Several primitives are discussed above in *Security Analysis* that may help, but none have been included in a full security proof a la (Ferrara et al. 2013; Garrison et al. 2016), nor have they been evaluated for absolute efficiency relative to the alternatives. We note that primitives for broadcast encryption presented up to now incur a variety of additional costs relative to other cryptosystems. For instance, depending on the scheme, the length of broadcast updates, key sizes, and/or the sizes of other data components can grow as the number of users increases. Similarly, implementations of key homomorphism thus far are dependent on the problem of Learning with Errors, leaving it an open question whether such primitives are possible with more standard assumptions. Interestingly, recent work has explored the implementation of efficient FE systems by leveraging trusted hardware primitives while also establishing formal foundations for these types of constructions (Fisch et al. 2017). However, the literature currently lacks a cryptographically enforced dynamic access control system that is both practical and formally proven to be correct and secure.

Cross-References

- Access Control Policies, Models, and Mechanisms
- Attribute-Based Access Control
- Broadcast Encryption
- Identity Based Encryption
- Proxy Re-encryption
- User Revocation in Identity Based Encryption and Attribute Based Encryption

References

Ferrara AL, Fuchsbauer G, Warinschi B (2013) Cryptographically enforced RBAC. In: 2013 IEEE 26th Computer Security Foundations Symposium, pp 115–129. https://doi.org/10.1109/CSF.2013.15

Fisch B, Vinayagamurthy D, Boneh D, Gorbunov S (2017) Iron: functional encryption using Intel SGX. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS'17. Association for Computing Machinery, New York, pp 765–782. https://doi.org/10.1145/3133956.3134106

Garrison WC, Lee AJ (2015) Decomposing, comparing, and synthesizing access control expressiveness simulations. In: 2015 IEEE 28th Computer Security Foundations Symposium, pp 18–32. https://doi.org/10.1109/CSF.2015.9

Garrison WC, Shull A, Myers S, Lee AJ (2016) On the practicality of cryptographically enforcing dynamic access control policies in the cloud. In: 2016 IEEE Symposium on Security and Privacy (SP), pp 819–838. https://doi.org/10.1109/SP.2016.54

Park S, Lee K, Lee DH (2015) New constructions of revocable identity-based encryption from multilinear maps. IEEE Trans Inf Forensics Secur 10(8):1564–1577. https://doi.org/10.1109/TIFS.2015.2419180

Qi S, Zheng Y (2019) Crypt-DAC: cryptographically enforced dynamic access control in the cloud. IEEE Trans Dependable Secure Comput 1–1. https://doi.org/10.1109/TDSC.2019.2908164

Wang F, Mickens J, Zeldovich N, Vaikuntanathan V (2016) Sieve: cryptographically enforced access control for user data in untrusted clouds. In: 13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16). USENIX Association, Santa Clara, pp 611–626. https://www.usenix.org/conference/nsdi16/technical-sessions/presentation/wang-frank