Cryptographic Enforcement of Access Control while Mitigating Key Sharing

Keith B. Frikken
Miami University, Oxford, U.S.A.

Keywords: Hierarchical-based Access Control, Cryptographic Enforcement, Mitigating Key Sharing.

Abstract: In this paper, we consider the well-studied problem of cryptographic enforcement of hierarchical-based access control. While this problem is well-studied, a significant drawback to prior approaches is that if a corrupt user shares his key, then any user can access the content of the corrupt user. This is particularly damaging since it is not possible to determine the identity of the corrupt user, and almost all previous schemes require some rekeying in order to revoke a key. To mitigate this key sharing attack, we propose a new model for cryptographic enforcement: Identity-based key management (IBKM). In this framework, each key is associated with an identity and this identity is required to access content. This allows the system to trace the source of key leakage and to revoke users without rekeying. The main disadvantage of this framework is the scheme does not have the ability to provide anonymous access, but it can be used to provide pseudonymous access. The main contributions of this paper are formal definitions for IBKM and schemes for achieving IBKM.

1 INTRODUCTION

In this paper we consider the problem of access control enforcement in hierarchical-based access control systems (such as RBAC) for content distribution systems. More formally, we consider access control systems with a set of roles and the following three parts: i) a user to role (U2R) mapping, ii) an object to role (O2R) mapping, and iii) a role hierarchy (RH) which is a partial order over the set of roles. Moreover, a specific user can access a specific object if and only if one of the user's roles is an ancestor in the RH of one of the object's roles. A straightforward access control enforcement mechanism is to simply distribute the U2R mapping, the O2R mapping, and the RH to each server that distributes content. Furthermore, the server needs some way to authenticate users; that is, either the server has to have authentication information for every user or there needs to be an authentication infrastructure (for example an authentication server that provides users with certificates). Clearly, with the above capabilities, such a content server would be able to enforce the access control policy.

In open systems with large numbers of users and many content servers, the "Achilles heel" of this straightforward access control enforcement mechanism is the need to distribute the U2R mapping to all servers and keeping it current when users are revoked or their rights are changed. The O2R mapping is not as cumbersome, because a server needs only the part of the O2R mapping that relates to the objects that it stores. Furthermore, in several applications the RH is much smaller and is less dynamic than the U2R mapping.

1.1 Key Derivation

To mitigate the problems described in the previous section, an enforcement mechanism based on key management has been proposed. This mechanism was introduced in (Akl and Taylor, 1983) and there has been many subsequent schemes including (Atallah et al., 2009; Sandhu, 1988)[1]. In key management, each role is assigned a cryptographic key, and every user is given the keys associated with the user's roles in the U2R mapping. To minimize the number of keys assigned to users, most key management schemes utilize key derivation. A key derivation scheme facilitates deriving a specific role's key given the key to any of that role's ancestors in the RH.

Key management provides an access control enforcement mechanism for hierarchical-based access

[1]There are many other schemes, but a detailed survey is beyond the scope of this paper. For recent surveys see (Atallah et al., 2009; Crampton et al., 2006).

B. Frikken K..
Cryptographic Enforcement of Access Control while Mitigating Key Sharing.
DOI: 10.5220/0004042602450250
In *Proceedings of the International Conference on Security and Cryptography* (SECRYPT-2012), pages 245-250
ISBN: 978-989-8565-24-2

control as follows: When a user requests an object from a server, the server uses the O2R mapping to determine the role of the object, the server then encrypts the requested object with the object's role key, and then the server sends the ciphertext to the user along with information about the object's role. If the user has permission to access the requested object, then the user will be able to derive the object's role's encryption key and the user will then be able to decrypt the ciphertext to obtain the object. Now, if the encryption scheme is secure, then it is computationally infeasible to decrypt content without the key. Thus users are prevented from obtaining objects (in plaintext form) to which they do not have the privileges. In summary, key management provides a solution to the hierarchical access control that enjoys the following benefits:

1. *U2R Mapping Ignorance:* The server does not need to know the U2R mapping.
2. *No Authentication Service Required:* The authentication is done implicitly through the key management system.
3. *Anonymous Access:* Users do not need to reveal their identities to the content servers.

Unfortunately, there is a significant downside to this key management enforcement mechanism, because malicious clients may distribute their keys to unauthorized users. In the worst case, these keys could be published, and thus anyone would be able to access all content linked to the leaked keys. Furthermore, even if the server determines that such a leakage has occurred, the server cannot trace the key leakage back to the culprit. In the prior schemes, this is typically handled by requiring the keys be stored on tamper-resistant hardware. However, achieving tamper-resistance in harder may be difficult. Furthermore, even if the hardware is tamper-resistant, then these schemes still suffer the problem that a single local security failure in the hardware leads to a global security failure of the system. Finally, if the required level of trust in such hardware could be reduced, then this would reduce the cost of deploying such systems. To exacerbate this key leakage problem, key revocation in such systems is highly impractical. That is, most key management schemes require that a large portion of users to re-key themselves every time a key is revoked. One scheme that suggests a way to avoid this re-keying is (Atallah et al., 2009), but it does so by requiring public information that is linear in the number of users, which is clearly not practical for large systems.

1.2 Id-based Key Management

The main contribution of this paper is a new form of key management that mitigates the key leakage and key revocation problems, while sacrificing only the anonymous access property. The motivation for this scheme is loosely based on identity-based encryption (Shamir, 1985), in that each identity will have a different key for every role. We call such a scheme an Identity-Based Key Management (IBKM) scheme. In order for an IBKM scheme to be secure, it must be collusion resistant. That is, a group of users should not be able to obtain any keys that are not derivable from their individual keys; this includes preventing users from deriving the keys of other users. Access control enforcement can be achieved using similar techniques to the traditional key management approaches. The only difference is that when a user requests an object the user sends his identity to the server, and then the server will encrypt the requested content with that identity's key for the requested object's role.

Enforcement by IBKM enjoys the benefits of U2R mapping ignorance and also does not require an authentication service. Clearly, IBKM does not provide anonymous access, but it does allow for pseudonymous access since users can choose pseudonyms that do not reveal their true identity. While this is a disadvantage of the proposed scheme, there are some applications where anonymous access is not desired– for example, when external requirements require that user access to the content be logged.

In addition to the above benefits, the key leakage problem is now mitigated. That is, since a user has to use her identity to access content, leaking a key now has to reveal the identity of the leaked key. Thus the scheme has key leakage tracing built into it. Furthermore, key revocation can be achieved by revealing revoked identities to the content servers. These servers could then forbid access to content requests that use this identity. This key revocation scheme has the same problems as certificate revocation lists, but is a substantial improvement over the previous key management schemes. Furthermore, expiration dates can easily be incorporated into the scheme by adding the expiration date to the identities. For example, the identity 'Bob, Dec 31, 2011' could be used to denote that Bob's key expires on Dec 31, 2011. This idea can be extended to support temporal key management schemes such as (Atallah et al., 2007; Ateniese et al., 2006; Santis et al., 2008) where the keys have start and end times.

The main contributions of this paper are: i) A scheme for tree access hierarchies that assumes only

pseudorandom functions, and ii) A scheme for arbitrary access hierarchies that assumes pseudorandom functions and the Strong RSA assumption.

2 PRELIMINARIES

2.1 Notation

Generally we model adversaries as Probabilistic Polynomial Time (PPT) algorithms. A function $\varepsilon(n)$ is negligible if for sufficiently large n, $\varepsilon(n) < \frac{1}{p(n)}$ for all polynomials p. We use $[x,y]$ to denote the set $\{v \in \mathbb{Z} : x \leq v \leq y\}$. The notation $v \leftarrow S$ for a set S denotes randomly selecting an element from S.

We model the access hierarchy as a partial order that is represented by a DAG $G = (V,E)$. We denote the vertices as $v_1, \ldots, v_m$. We say that v_i is an ancestor of v_j if there is a path from v_i to v_j in G. In this case we also say that v_j is a descendant of v_i.

A prime p is a safe prime if $(p-1)/2$ is also prime. We use $\phi(N)$ to denote the Euler phi function of N. We use $RSAModGen(1^\kappa)$ to choose two safe primes p and q (each with κ bits), and then return $N = pq$.

2.2 Problem Description

The environment that we consider has three actors: i) clients, ii) content servers, and iii) a content owner. The content owner establishes the access control policy (i.e., the U2R mapping, the O2R mapping, and the RH), it also posts content on a subset of the content servers. The content owner performs a setup phase with the clients. The clients request access to information from the content servers.

2.3 Assumptions

We utilize two assumptions in this paper: i) the existence of pseudorandom functions and ii) the Strong RSA assumption (Baric and Pfitzmann, 1997). We review both of these assumptions below.

A function $F : \{0,1\}^\kappa \times \{0,1\}^\kappa \rightarrow \{0,1\}^\kappa$ is a pseudorandom function if for all PPT algorithms D:

$$Pr_{k \leftarrow \{0,1\}^\kappa}[D^{F(k,\cdot)}(1^\kappa) = 1] - Pr[D^{f(\cdot)}(1^\kappa) = 1]$$

is negligible in κ where f is a truly random function.

Recall that the Strong RSA assumption states that given y, N where N chosen using $RSAModGen(1^\kappa)$ that it is hard to produce a pair (x,e) such that $x^e \bmod N = y$ and $e \neq 1$. More specifically, we are assuming that no PPT algorithm A can produce such a pair except with negligible probability (in κ).

2.4 Formal Definition

An identity-based key management (IBKM) system consists of three algorithms $(Setup, GenKey, Derive)$ where:

- $Setup(G, 1^\kappa)$ takes as input a security parameter 1^κ and an access graph $G = (V,E)$. The output of this method is public information *pub* and a secret key *sk*.
- $GenKey(id, v, pub, sk)$ takes as input an identity, *id*, an access vertex $v \in V$, the public information *pub*, and the secret key *sk*. The output of this method is the key $k_{id,v}$.
- $Derive(G, pub, id, v_s, v_d, k_{id,v_s})$ takes as input the access graph G, the public information *pub*, an identity *id*, a source vertex $v_s \in V$, a destination vertex $v_d \in V$, and the secret key k_{id,v_s} If there is a path from v_s to v_d in the access graph, then the output of this method is k_{id,v_d} and otherwise it is $\perp$.

Two notions of security for traditional key management schemes were introduced in (Atallah et al., 2009). We initially discuss the weaker of these two notions, security against key recovery, but describe the stronger version, key indistinguishability, in section 4.1. Security against key recovery states that a coalition of corrupt users can recover a key if and only if one of the corrupt users can recover the key without collusion. The main difference between IBKM and traditional key management is that the corrupt users must be prevented from deriving any key for a non-corrupted identity. Formally, security against key recovery for IBKM is defined by the following experiment, $Exp^{\mathcal{A},\Pi}_{IdKeyReco}(1^\kappa, G)$:

1. $(pub, sk) \leftarrow Setup(G, 1^\kappa)$.
2. $\mathcal{A}$ is given *pub* and G and is provided oracle access to $GenKey(\cdot,\cdot,sk)$. Store all query tuples in the set L.
3. Eventually, $\mathcal{A}$ outputs a triple (id, v, k). We say that $Exp^{\mathcal{A},\Pi}_{IdKeyReco}(1^\kappa, G) = 1$ if the following two conditions hold:
   (a) $k = GenKey(id, v, pub, sk)$. That is, the key is correct for the identity-vertex pair that the adversary output.
   (b) There is no tuple of the form $(id, v^\star) \in L$ such that $v^\star$ is an ancestor of v. That is the adversary has not used its oracle to obtain *id*'s key for any of v's ancestors.

The advantage of an adversary $\mathcal{A}$ is defined as:

$$Adv^{\mathcal{A},\Pi}_{IdKeyReco}(1^\kappa, G) = Pr[Exp^{\mathcal{A},\Pi}_{IdKeyReco}(1^\kappa, G) = 1]$$

The scheme Π is secure against key recovery if for all PPT adversaries $\mathcal{A}$, $Adv^{\mathcal{A},\Pi}_{IdKeyReco}(1^\kappa, G)$ is negligible in κ.

Due to the usage of the IBKM scheme, the goal is to design such a scheme while minimizing: i) the size of the public key, ii) the size of the secret key, iii) the cost of *GenKey*, and iv) the cost of *Derive*.

3 IBKM SCHEMES

We first introduce an IBKM scheme for trees and then introduce an IBKM scheme for arbitrary access graphs. The proofs of security are omitted due to page constraints.

3.1 Scheme 1: For Trees Only

In this section we modify the scheme in (Sandhu, 1988) to incorporate the user identity. This scheme works only for tree access structures, but only requires pseudorandom functions (PRF). The main idea of this scheme is that the PRF will be used to derive the key associated with the root node for each identity, and a very similar scheme to that in (Sandhu, 1988) will be used to assign keys to the other nodes in the access hierarchy. More specifically, the following is the IBKE scheme where $F : \{0,1\}^\kappa \times \{0,1\}^\kappa \rightarrow \{0,1\}^\kappa$ is a pseudorandom function:

1. $Setup(G = (V,E), 1^\kappa)$: For each edge $(v_i, v_j) \in E$, choose a label $\ell_{i,j} \leftarrow \{0,1\}^\kappa$ and assign it to that edge. Denote as $\hat{G}$ the graph G where each edge has this label. Finally, choose a random secret $s \leftarrow \{0,1\}^\kappa$. Then set $pub = \hat{G}$ and $sk = s$.
2. $GenKey(id, v, pub, sk)$. Let v_0 be the root of the tree G, then set $k_{id,v_0} = F_s(id)$. Return the value from $Derive(pub, v_0, v, k_{id,v_0})$.
3. $Derive(pub, id, v_s, v_d, k_{id,v_s})$:
   (a) If there is no path from v_s to v_d in G, then return $\perp$ else if $v_s = v_d$ return k_{id,v_s}.
   (b) Find a path from v_s to v_d in G. Let the node after v_s be v_i. Compute $k_{id,v_i} = F_{k_{id,v_s}}(\ell_{s,i})$. Return $Derive(pub, id, v_i, v_d, k_{id,v_i})$.

The costs of this scheme are as follows: the size of the public parameters are $O(|V|)$. The secret key size and the user key size are both $O(\kappa)$. The computational cost of *GenKey* for node v_i is $O(height(v_i))$ where $height(x)$ is the height of node x in the access hierarchy. In the worst case (i.e., a linear access hierarchy) this is $O(|V|)$, but if the access hierarchy is balanced, then this is $O(\log|V|)$. Finally, the computational cost of *Derive* for node v_s to v_d is $O(height(v_d) - height(v_s))$ PRF functions, which is $O(|V|)$ in the worst case (but is $O(\log|V|)$ in the balanced case).

3.2 Scheme 2: For Arbitrary Graphs

In this section, we introduce an IBKM scheme for arbitrary graphs. This scheme is a modification of the scheme introduced in (Akl and Taylor, 1983) that provided traditional key derivation for arbitrary graphs. At a high level, the scheme in (Akl and Taylor, 1983) used a public RSA modulus N (in the remainder of this section all math is done modulo N), a secret value $x \in \mathbb{Z}^*_N$, and assigned a public random prime to every node in the access graph. Each node in the graph will also have an exponent that is the product of all primes that are associated with non-descendants of the node. That is the exponent $e_i = \prod_{v_j \in ND(v_i)} p_j$, and the key for vertex v_i is $x^{e_i} \bmod N$ (where $ND(v_i)$ corresponds to the non-descendants of v_i). This scheme is collusion resistant. That is, suppose that a set of users collude where none of these users have the key for any ancestor of v_i, then the exponents for all of these user keys will contain the prime associated with v_i. Now, e_i does not contain this prime, and thus in order to generate this key, the adversary must be able to compute the multiplicative inverse of this value which can be used invalidate the Strong-RSA assumption. We refer the reader to (Akl and Taylor, 1983) for a more detailed discussion of this previous scheme. To convert this scheme into an IBKM, we simply assign a different x value to each identity using a pseudorandom function.

1. $Setup(G = (V,E), 1^\kappa)$: Set $N \leftarrow RSAModGen(1^k)$. For each node $v_i \in V$, choose a unique prime p_i and set $pub = (G, N, p_1, \ldots, p_m)$. It is important to note that any set of unique primes that are relatively prime to $\phi(N)$ will be sufficient for this scheme. Thus it makes sense to choose the primes as small as possible, and if N is the product of two safe primes, then the prime values can be the first m odd prime numbers. As a shorthand we will let $e_i = \prod_{v_j \in ND(v_i)} p_j$; note that these values won't be stored in the key but can be derived from the public information. Let $F : \{0,1\}^\kappa \times \{0,1\}^\kappa \rightarrow \mathbb{Z}^*_N$ be a pseudorandom function. Choose $s \leftarrow \{0,1\}^\kappa$ and set $sk = s$.
2. $GenKey(id, v, pub, sk)$: Set $X_{id} = F_s(id)$. Return $k_{id,v} = (x_{id})^{e_i}$.

3. $Derive(pub, id, v_s, v_d, k_{id,v_s})$:
   (a) If there is no path from v_s to v_d in G, then return $\perp$ else if $v_s = v_d$ return k_{id,v_s}.
   (b) Let ND_s (resp. ND_d) be the set of non-descendants for v_s (resp v_d). It must be that $ND_d \supseteq ND_s$, since the descendants of v_d are a subset of v_s. Hence it must be that e_s is a divisor of e_d. Hence let $q = \frac{e_d}{e_s}$, and return $k_{id,v_d} = (k_{id,v_s})^q$

The public parameters of the above scheme are the graph and the m prime numbers. Since it is well know that the mth prime is about $m \log m$ each of these primes can be represented with $O(\log m)$ bits. Hence the public key has size $O(|G| + m \log m)$. The computational cost of *KeyDerive* is a single PRF, and then a modular exponentiation for every non-descendant of the vertex. In the worst case this requires $O(m)$ modular exponentiations. It is worth noting that if the repeated squares method of modular exponentiation is used then this can be accomplished with $O(m \log m)$ modular multiplications since each prime has $O(\log m)$ bits. This cost is reduced to $O(1)$ in section 3.3. Finally, the cost of key derivations is a modular exponentiation for each additional node in the set of non-descendants for v_d compared to v_s, in the worst case this is $O(m)$ modular exponentiations (or more precisely $O(m \log m)$ modular multiplications).

3.3 Reducing *KeyGen* to $O(1)$

The protocol for *KeyGen* requires $O(m \log m)$ modular multiplications, because the server had to raise the identity key to all of the exponents associated with non-descendant nodes. This can be reduced to $O(1)$ modular exponentiations by making the following modifications. Since the outsourced server is semi-trusted, it can learn the factorization of N. Thus it can learn $e_j \bmod \phi(N)$ for all nodes v_i as part of its secret key. With this information a key for a specific identity can be derived by raising the identity information to this value, and hence the computation for *KeyGen* is reduced to $O(1)$. The specific changes are omitted due to page constraints.

4 EXTENSIONS

4.1 Providing Key Indistinguishability

The previous schemes have provided security against key recovery. However, as described in (Atallah et al., 2009) this is not always enough. In many cases, a stronger notion key indistinguishability is required in that the adversary cannot distinguish the real key from a randomly generated key. Key indistinguishability is required because most definitions from cryptographic primitives assume that the adversary has no information about the key. At first glance, key indistinguishability would seem impossible, because the adversary could simply apply key derivation on the key to determine if it can derive a weaker key (i.e., one from a descendant role) that it already knows. To mitigate this problem, we use a similar idea to that in (Atallah et al., 2009) by creating two keys for every access vertex: i) a derivation key and ii) an encryption key. The derivation keys will be used to derive both the derivation keys and the encryption keys, but the encryption keys will be used to encrypt content. Note that the same modification proposed in (Atallah et al., 2009) for converting traditional key management schemes with key recovery into a scheme that provides key indistinguishability. One way to do this is to choose a random label, ℓ_v for each vertex v, and then set the encryption key for (id, v) to $F_k(\ell_v)$ where k is the derivation key for (id, v) and F is a pseudorandom function. A formal definition is omitted due to page constraints.

4.2 Hiding Content From Servers

A variation of key-based access control was used in (De Capitani di Vimercati et al., 2007). In this variation the content is encrypted before placing it on the servers. That is, the content owners do not trust the content servers with the actual content, however the owners trust the content servers to enforce the access control policies. By itself, IBKM does not facilitate hiding content from the server, however this can be achieved by using similar ideas to (De Capitani di Vimercati et al., 2007). To support this feature, content is encrypted twice, once with a traditional key management scheme and then with an IBKM scheme, where both schemes use the same RH. The first encryption is applied before sending it to the server, and thus the content is protected from the server. The server would then apply the IBKM scheme to prevent unauthorized users from accessing the content. In order to access content a user would have to decrypt the message using the IBKM key and then by using the traditional key management key.

5 RELATED WORK

There has been a substantial number of schemes in the traditional key management enforcement model (Atallah et al., 2009; Sandhu, 1988). These schemes all have the problem that if a user shares its key, that

the culprit is not traceable and rekeying the users requires the system to rekey everyone with this key or a key for a descendant.

A related area is public key traitor tracing (Boneh and Franklin, 1999). While this appears to solve the same problem as the current work, there is a crucial difference. Mainly, these schemes do not take into account the role hierarchy and the need to derive keys according to the hierarchy.

Another area of related work is broadcast encryption (Fiat and Naor, 1994). These problems are similar in that they both attempt to give access to content to a set of users according to an access policy. However, in order to user broadcast encryption for the problems considered in this paper the content servers would need to know the RH and the U2R mapping.

6 CONCLUSIONS

In this paper we introduced a new framework for cryptographic enforcement of access control. In this new framework each user has its own key for every access level, and the content providers use the identity to encrypt content. This new framework mitigates the problems posed by key sharing, because this scheme has built-in traitor tracing and allows revocation without rekeying. These benefits come at the price of allowing anonymous access, but pseudonymous access is still possible. We give two schemes in this new framework, one for tree access hierarchies and one for arbitrary hierarchies. This new framework is an interesting first step towards mitigating the problems with cryptographic enforcement of access control.

ACKNOWLEDGEMENTS

The authors would like to thank the anonymous reviewers for their comments and useful suggestions. Portions of this work were supported by Grant CNS-0915843 from the National Science Foundation.

REFERENCES

Akl, S. and Taylor, P. (1983). Cryptographic solution to a problem of access control in a hierarchy. *ACM Transactions on Computer Systems*, 1(3):239–248.

Atallah, M., Blanton, M., and Frikken, K. (2007). Incorporating temporal capabilities in existing key management schemes. In Biskup, J. and Lopez, J., editors, *Computer Security (ESORICS 2007)*, volume 4734 of *Lecture Notes in Computer Science*, pages 515–530. Springer Berlin / Heidelberg.

Atallah, M. J., Blanton, M., Fazio, N., and Frikken, K. B. (2009). Dynamic and efficient key management for access hierarchies. *ACM Trans. Inf. Syst. Secur.*, 12:18:1–18:43.

Ateniese, G., De Santis, A., Ferrara, A. L., and Masucci, B. (2006). Provably-secure time-bound hierarchical key assignment schemes. In *Proceedings of the 13th ACM conference on Computer and communications security*, CCS '06, pages 288–297, New York, NY, USA. ACM.

Baric, N. and Pfitzmann, B. (1997). Collision-free accumulators and fail-stop signature schemes without trees. In Fumy, W., editor, *Advances in Cryptology (EUROCRYPT 1997)*, volume 1233 of *Lecture Notes in Computer Science*, pages 480–494. Springer Berlin / Heidelberg.

Boneh, D. and Franklin, M. (1999). An efficient public key traitor tracing scheme. In Wiener, M., editor, *Advances in Cryptology (CRYPTO 1999)*, volume 1666 of *Lecture Notes in Computer Science*, pages 783–783. Springer Berlin / Heidelberg.

Crampton, J., Martin, K., and Wild, P. (2006). On key assignment for hierarchical access control. In *Computer Security Foundations Workshop, 2006. 19th IEEE*.

di Vimercati, S. D. C., Foresti, S., Jajodia, S., Paraboschi, S., and Samarati, P. (2007). Over-encryption: management of access control evolution on outsourced data. In *Proceedings of the 33rd international conference on Very large data bases*, VLDB '07, pages 123–134. VLDB Endowment.

Fiat, A. and Naor, M. (1994). Broadcast encryption. In Stinson, D., editor, *Advances in Cryptology CRYPTO 93*, volume 773 of *Lecture Notes in Computer Science*, pages 480–491. Springer Berlin / Heidelberg.

Sandhu, R. (1987). On some cryptographic solutions for access control in a tree hierarchy. In *Fall Joint Computer Conference on Exploring technology: today and tomorrow*, pages 405–410.

Sandhu, R. (1988). Cryptographic implementation of a tree hierarchy for access control. *Information Processing Letters*, 27(2):95–98.

Santis, A. D., Ferrara, A. L., and Masucci, B. (2008). New constructions for provably-secure time-bound hierarchical key assignment schemes. *Theoretical Computer Science*, 407(1-3):213 – 230.

Shamir, A. (1985). Identity-based cryptosystems and signature schemes. In Blakley, G. and Chaum, D., editors, *Advances in Cryptology*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53. Springer Berlin / Heidelberg.