Oblivious Enforcement of Hidden Information Release Policies

Brian Wongchaowart
bpw5@cs.pitt.edu

Adam J. Lee
adamlee@cs.pitt.edu

Department of Computer Science
University of Pittsburgh
210 S. Bouquet St.
Pittsburgh, PA 15260

ABSTRACT

In a computing system, sensitive data must be protected by release policies that determine which principals are authorized to access that data. In some cases, such a release policy could refer to information about the requesting principal that is unavailable to the information provider. Furthermore, the release policy itself may contain sensitive information about the resource that it protects. In this paper we describe a scheme for enforcing information release policies whose satisfaction cannot be verified by the entity holding the protected information, but only by the entity requesting this information. Not only does our scheme prevent the information provider from learning whether the policy was satisfied, but it also hides the information release policy being enforced from the requesting principal. Unlike previous approaches, our construction requires no guesswork or wasted computation on the part of the information requester. The information release policies that we consider can contain third-party assertions that themselves have release conditions that must be satisfied; we show that our system functions correctly even when these dependencies form cycles.

Categories and Subject Descriptors

D.4.6 [**Operating Systems**]: Security and Protection—*access controls, authentication*; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection—*authentication*

General Terms

Security

Keywords

Hidden policies, hidden credentials, distributed proof

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
ASIACCS'10 April 13–16, 2010, Beijing, China.
Copyright 2010 ACM 978-1-60558-936-7 ...$10.00.

1. INTRODUCTION

In this paper we consider an access control setting in which Alice would like to obtain a secret s from Bob, but Bob has a policy requiring that a conjunction of quoted assertions made by other principals must be true before s can be disclosed to Alice. By a "quoted assertion" we mean a proposition of the form p_i says e_i, where e_i is an arbitrary expression that can be asserted by some principal p_i; the quoted assertion p_i says e_i is true if p_i is willing to assert e_i. For example, e_i may be the assertion that Alice has some attribute (e.g., that Alice is a student), and p_i may be an authority trusted by Bob to know whether Alice does in fact have the attribute in question (e.g., the registrar at an accredited university).

If Bob is able to determine the truth of each proposition p_i says e_i in his release policy for s, then he can decide whether to release s to Alice without any further interaction with her. But Bob's policy may also refer to private attributes of Alice that he is not authorized to learn about directly from the relevant authority. In this case, Bob could ask Alice to prove that she satisfies his policy by obtaining digitally signed credentials attesting to the truth of each quoted assertion that he cannot evaluate himself. This solution may be unsatisfactory for both parties, however. On the one hand, Bob may have to reveal parts of his policy that he considers sensitive to Alice, as the release policy for a secret can reveal information about the secret itself. On the other, Alice may have to disclose the truth of quoted assertions that she considers private to Bob.

If there were a universally trusted third party (TTP), solving this problem would be simple: Bob discloses his secret s and its release policy p to the TTP, which queries principals about the truth of the quoted assertions in p and releases s to Alice if and only if p is satisfied. Alice learns nothing about Bob's policy aside from whether it was satisfied (unless she is queried by the TTP herself), Bob does not learn whether Alice receives the secret, and no expensive cryptography is needed. In this paper we show that a simple and computationally efficient solution is possible even without a TTP if Bob has a basic level of trust in the principals whose quoted assertions his policy depends on. Specifically, Bob must only trust these principals not to reveal their interactions with Bob to Alice. In our approach, Bob does not have to be able to learn the truth value of a quoted assertion in his policy if Alice is allowed to do so, while Alice learns no more about Bob's policy than she would learn from in-

teracting with a TTP. Because Bob does not learn whether his policy was satisfied, we say that our protocol permits the *oblivious* enforcement of a hidden information release policy.

This paper is organized as follows. Section 2 develops the intuition behind our construction, while the full protocol details are given in Section 3. Section 4 presents a sample run. We briefly compare our approach to related work in Section 5, and summarize in Section 6.

2. PROTOCOL INTUITION

Our construction relies on a public-key encryption scheme supporting a multiplicative homomorphism. In such an encryption scheme, if $E(m_1)$ and $E(m_2)$ represent the encryptions of messages m_1 and m_2 using the same public key, then these two ciphertexts can be combined to yield $E(m_1 \cdot m_2)$ without knowledge of the private key required to decrypt $E(m_1)$ and $E(m_2)$. We further require that the encryption scheme have the property of IND-CPA security, or indistinguishability of ciphertexts under chosen-plaintext attack, which ensures that no information can be obtained about the plaintext corresponding to a given ciphertext without knowledge of the private key. A concrete example of an encryption scheme with these properties is ElGamal encryption [5] using a group in which the decision Diffie-Hellman (DDH) assumption [1] holds.

Suppose that a principal p_0 wishes to obtain a secret s from principal p_1, but that p_1 is only willing to disclose s to p_0 if certain conditions—the *release policy* for s—hold. We model p_1's release policy for s as a set of quoted assertions of the form p_i says e_i, where e_i is any expression that can be evaluated to a Boolean value by p_i. If p_1's release policy requires that some of these assertions must be true simultaneously, then each relevant e_i should include the constraint that the expression must evaluate to true throughout a time interval long enough to ensure that the expressions are simultaneously true at some point in time despite any clock differences between the principals.

Since p_1 can trivially enforce the part of a release policy that depends only on information that it can obtain from its local knowledge, credentials provided by p_0, or other principals in the system, we will assume in the following discussion that the release policy for s can be evaluated as satisfied or unsatisfied entirely on the basis of the truth values of quoted assertions p_2 says e_2, ..., p_n says e_n, where $p_2, \ldots, p_n$ are willing to disclose the truth values of these assertions to p_0, but not to p_1. (A release policy that depends on information available to neither p_0 nor p_1 cannot be evaluated without the help of a third party.) We also assume that s can be encoded as a short binary string of perhaps 128 bits, since this is sufficient for a symmetric key that can be used by p_0 to decrypt additional data or as proof of authorization.

Given these assumptions, p_1 can ask each $p_i \in \{p_2, \ldots, p_n\}$ to evaluate the corresponding expression e_i from p_1's release policy. This expression may involve information that p_i is willing to reveal to p_0, but not to p_1. In order to assert the truth of the expression e_i, p_i encrypts the value 1 for p_0's public key using an IND-CPA-secure homomorphic encryption scheme and returns the ciphertext to p_1; otherwise, p_i encrypts a random value and returns the resulting ciphertext. These ciphertexts reveal no information at all to p_1 because of the IND-CPA security of the encryption scheme.

Now p_1 encrypts the secret s using p_0's public key, homomorphically combines the encryption of s with each of the ciphertexts received from $p_2, \ldots, p_n$, and finally sends the combined ciphertext to p_0. If the quoted assertions p_2 says e_2, ..., p_n says e_n from p_1's release policy were all true, then p_0 receives the encryption of s, since homomorphically combining the encryption of s with the encryption of 1 has no effect. Otherwise, one of the ciphertexts that p_1 received from $p_2, \ldots, p_n$ must have been the encryption of a random value, so p_0 receives the encryption of s multiplied by a random value, which contains no more information than the random value. In either case, p_0 learns nothing about the structure of p_1's policy or the fact that $p_2, \ldots, p_n$ were involved in enforcing it, since p_0 always receives a single ciphertext from p_1. If p_0 fails to obtain s by decrypting the ciphertext, then this fact may simply indicate that p_1 determined that the release policy for s was unsatisfied based on p_1's local knowledge. In the following section, we describe this solution in more detail, including how $p_2, \ldots, p_n$ can enforce release policies of their own on the truth values of the assertions that they evaluate.

3. PROTOCOL DETAILS

Let $\mathcal{M}$ denote the message space of the IND-CPA-secure homomorphic encryption scheme, $E_{p_i}(m)$ denote the encryption of message m using principal p_i's public key, and $E_{p_i}(m_1) \otimes E_{p_i}(m_2) = E_{p_i}(m_1 \cdot m_2)$ denote the homomorphic combination of two ciphertexts encrypted using p_i's public key. We will assume that principals can obtain one another's public keys and that all communication takes place over secure and authenticated channels. As before, we use p_0 to denote the principal who wishes to obtain a secret $s_1 \in \mathcal{M}$ and p_1 to denote the principal in possession of this secret.

3.1 Core Protocol

The secret requester p_0 first sends a message to p_1 asking for its secret s_1, along with a newly generated globally unique session identifier *sid*. Upon receiving this request, p_1 contacts each principal p_i listed in its release policy for s_1 and asks it to evaluate the corresponding assertion e_i. The session identifier *sid* generated by p_0 is passed along with this request. At this point, p_1 may replace s_1 with a random value if p_1 decides that p_0 is not authorized to receive s_1 based on information available to p_1.

Each principal p_i contacted by p_1 selects a local secret s_i based on the result of evaluating e_i. If e_i is true and p_i is (conditionally) willing to disclose this to p_0 via p_1, then s_i is set to 1; otherwise, s_i is an element of the message space $\mathcal{M}$ chosen uniformly at random. If p_i is willing to unconditionally reveal s_i to p_0, then p_i simply returns $E_{p_0}(s_i)$ to p_1. Otherwise, p_i can make the disclosure of the truth value of e_i contingent upon the truth of a set of quoted assertions of the same form as p_1's release policy for s_1 by homomorphically combining $E_{p_0}(s_i)$ with additional ciphertexts as described below. In either case, p_1 homomorphically combines the ciphertext returned by each p_i with $E_{p_0}(s_1)$, the encryption of the secret requested by p_0, and returns the final ciphertext to p_0.

It may initially seem as though any principal p_i contacted by p_1 can enforce its release policy for s_i in exactly the same manner as p_1 enforces its release policy for s_1. This is almost the case, but a problem arises when the release policies of several principals create a cycle of quoted assertions in which the disclosure of each assertion depends on the disclosure of another assertion in the cycle. The purpose of the session

identifier passed along with each request is to enable principals to detect and break such cycles.

3.2 Policy Cycle Resolution

The basic idea of the algorithm that we will present below is that when a requesting principal p_r contacts a principal p_i to ask about some assertion e_i, p_i must ensure that the ciphertext that is returned to p_r is the encryption of a random value if any quoted assertion p_j says e_j in p_i's release policy for the truth value of e_i is false. If p_i always waits for p_j to return a ciphertext corresponding to the truth value of e_j before p_i returns an answer to p_r, however, a policy cycle will cause a deadlock.

Our solution is for p_i to *provisionally* act as though the required ciphertext c_j from p_j was the encryption of a random value if a request for c_j was previously sent in the current session but no reply has yet been received. That is, p_i homomorphically combines the ciphertext c_i that it will return to the requesting principal p_r with the encryption of a newly generated random value. When the real response c_j from p_j is eventually received, a number of these random values, $r_1, r_2, \ldots, r_n$, may have been used, but p_i can undo the effect of $r_1, r_2, \ldots, r_n$ by computing $(r_1 r_2 \cdots r_n)^{-1}$ and homomorphically combining the encryption of this quantity with c_j. Since every ciphertext encrypted for p_0 in a session is eventually combined into a single ciphertext, $(r_1 r_2 \cdots r_n)^{-1}$ will cancel out $r_1 r_2 \cdots r_n$ in the final ciphertext received by p_0.

Specifically, each principal p_i maintains a *product* table indexed by the name of a principal, an assertion, and a session identifier. The product $r_1 r_2 \cdots r_n$ of all the random values that p_i generates while waiting for an answer from p_j concerning assertion e_j in session *sid* is stored in table entry $product[p_j, e_j, sid]$ and p_j's answer c_j is combined with the encryption of $product[p_j, e_j, sid]^{-1}$ when c_j becomes available. Upon receiving a query concerning e_i from a requesting principal p_r in a session initiated by p_0 and identified by *sid*, p_i initializes the ciphertext c_i that will be returned to p_r with $E_{p_0}(s_i)$, where $s_i = 1$ if e_i is true and p_i is (conditionally) willing to disclose this to p_0 via p_r, and s_i is a freshly chosen random element of the message space $\mathcal{M}$ otherwise. Then p_i consults its release policy for disclosing the truth value of e_i to p_0 via principal p_r to obtain a list of quoted assertions of the form p_j says e_j. For each assertion p_j says e_j in this list, there are now two possible cases:

- If p_i is not currently waiting for a reply from p_j about e_j in the current session, then p_i queries p_j about e_j (also sending the session identifier *sid*) and assigns 1 to $product[p_j, e_j, sid]$. When p_j eventually replies with a ciphertext c_j, p_i homomorphically combines both c_j and the encryption of $product[p_j, e_j, sid]^{-1}$ with the ciphertext c_i that will be returned to p_r: $c_i \leftarrow c_i \otimes c_j \otimes E_{p_0}(product[p_j, e_j, sid]^{-1})$.

- If p_i has previously contacted p_j about e_j but has not yet received a reply (possibly because this is the second time around a cycle of quoted assertions), then p_i chooses a random element r from $\mathcal{M}$ and multiplies $product[p_j, e_j, sid]$ by r: $product[p_j, e_j, sid] \leftarrow product[p_j, e_j, sid] \cdot r$. Then p_i homomorphically combines the encryption of r with the ciphertext c_i that will be returned to p_r: $c_i \leftarrow c_i \otimes E_{p_0}(r)$.

Finally, p_i returns the resulting ciphertext c_i to p_r.

While p_i may be tempted to cache a ciphertext c_j received from principal p_j so that p_i will never need to query p_j more than once about any assertion within a single session, this is inadvisable because the same random value would be reused multiple times if c_j was the encryption of a randomly chosen element of the message space. That is, reusing c_j would not be indistinguishable from using a fresh element of the message space chosen uniformly at random.

4. AN EXAMPLE

Suppose that Alice is a reporter who has heard a rumor about a certain government agency. She asks Bob, who works at this agency, whether the rumor is true (this is Bob's secret s_B). Bob is willing to tell s_B to Alice if his superior Carol says that it is all right (Carol says e_C). Carol is not willing to say that it is all right unless David also approves (David says e_D), while David is not willing to say that it is all right unless Carol approves (Carol says e_C). Suppose that Carol and David both approve of Bob telling Alice the secret, so e_C and e_D are true.

In our protocol, Alice first contacts Bob, who contacts Carol, which causes Carol to contact David, who then contacts Carol again. At this point the chain ends, since Carol has already contacted David concerning e_D in the current session. The ciphertexts passed back up the chain from Carol to Alice are shown in Figure 1. At the end of the chain, Carol has not yet received a reply from David concerning e_D, so she chooses a random element r from $\mathcal{M}$ and sets $product[David, e_D, sid] \leftarrow r$. The encryption of r is combined with the encryption of Carol's secret s_C (which is 1, since e_C is true), and the resulting ciphertext is then returned to David. At this stage Carol has revealed no information about her secret value s_C, since it is completely obscured by multiplication with the random value r.

Upon receiving $E_{\text{Alice}}(rs_C)$ from Carol, David combines it with the encryption of his secret s_D (which is also 1 in this case, because e_D is true) and passes the resulting ciphertext up the chain to Carol. While David does not know that this ciphertext contains the random factor r, he can assume that it contains the factor s_C, which would be a random value that conceals David's secret s_D unless e_C is true. Upon receiving the combined ciphertext from David, Carol cancels out the random factor r that she added earlier by combining $E_{\text{Alice}}(rs_C s_D)$ with $E_{\text{Alice}}(product[David, e_D, sid]^{-1}) = E_{\text{Alice}}(r^{-1})$. Carol also multiplies the combined secret by s_C for a second time, but this has no effect since $s_C = 1$ (if e_C were false, then Carol would use a new random value for s_C each time she is queried about e_C). Although r has now been removed from the ciphertext, Carol's secret s_C is protected by the inclusion of the factor s_D. When Bob receives the combined ciphertext from Carol, he contributes s_B, yielding $E_{\text{Alice}}(rr^{-1}s_B s_C^2 s_D)$. Since $rr^{-1} = 1$ and $s_C^2 s_D = 1$, Bob's response to Alice is just $E_{\text{Alice}}(s_B)$. Thus Alice has no reason to believe that Bob has consulted anyone about releasing his secret value s_B.

5. RELATED WORK

Hidden credentials [7, 2], oblivious signature-based envelope (OSBE) [9], and multiauthority attribute-based encryption [3, 4] are cryptographic mechanisms that allow a message to be protected by a release policy whose satisfaction is verified by the recipient of the message. In hidden creden-

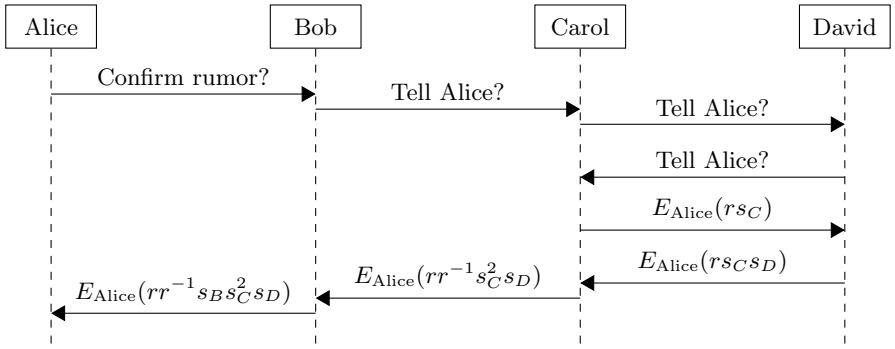

Figure 1: Queries and replies in the example of Section 4.

tials and OSBE, a message is encrypted in such a way that only a recipient who possesses certain digital credentials can decrypt the message. The identity of the intended recipient needs to be known to the message sender when this information is included in the recipient's digital credentials, as is usually the case. In multiauthority attribute-based encryption, a ciphertext is associated with a set of attributes such that any user who has been issued decryption keys (possibly by different authorities) that correspond to a satisfying set of attributes can decrypt the message. Thus no knowledge of the identities of potential recipients is needed at the time of message encryption. Unlike our work, however, in all of these schemes decrypting the message implies knowledge of at least one way of satisfying the sender's release policy.

Protocols based on scrambled circuit evaluation can be used to allow an information provider to keep a release policy partially hidden even when a message recipient satisfies the provider's policy. The three schemes of progressively greater complexity presented in [6] respectively reveal a superset of the attributes in the policy, the number of attributes in the policy that are satisfied, and an upper bound on the total number of attributes in the policy. Nevertheless, these "hidden policies with hidden credentials" protocols still require the message recipient to supply a set of credentials that potentially satisfies the unknown policy. In our system, the issuers of the credentials would be contacted directly by the information provider without any participation from the recipient; the trade-off, of course, is that partial information about the policy is revealed to the credential issuers.

Our work is also related to the notion of confidentiality-preserving distributed proof introduced in [8], which allows information providers to place release conditions that are verified by the querier on facts used in distributed inference. On the one hand, these release conditions need to be known to the querier, unlike the hidden dependencies in our protocol, but on the other, information providers communicate only with the querier and not with one another, which hides the source of dependency relationships from the providers of facts that satisfy those dependencies.

6. CONCLUSION

In this paper we have described a scheme for enforcing information release policies whose satisfaction cannot be verified by the principal holding the protected information, but only by the principal requesting this information. Our construction hides the information release policy being enforced from the requesting principal and at the same time hides whether the release policy was satisfied from the information provider. The quoted assertions in the information release policy can themselves have release conditions that must be satisfied and our system functions correctly even when these dependencies form cycles.

Acknowledgments

This research was supported by the National Science Foundation under grant number CCF–0916015.

7. REFERENCES

[1] D. Boneh. The decision Diffie-Hellman problem. In *Proceedings of the Third International Symposium on Algorithmic Number Theory*, pages 48–63, 1998.

[2] R. W. Bradshaw, J. E. Holt, and K. E. Seamons. Concealing complex policies with hidden credentials. In *Proceedings of the 11th ACM Conference on Computer and Communications Security*, pages 146–157, 2004.

[3] M. Chase. Multi-authority attribute based encryption. In *Proceedings of the Fourth Theory of Cryptography Conference*, pages 515–534, 2007.

[4] M. Chase and S. S. M. Chow. Improving privacy and security in multi-authority attribute-based encryption. In *Proceedings of the 16th ACM Conference on Computer and Communications Security*, pages 121–130, 2009.

[5] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.

[6] K. Frikken, M. Atallah, and J. Li. Attribute-based access control with hidden policies and hidden credentials. *IEEE Transactions on Computers*, 55(10):1259–1270, 2006.

[7] J. E. Holt, R. W. Bradshaw, K. E. Seamons, and H. Orman. Hidden credentials. In *Proceedings of the 2003 ACM Workshop on Privacy in the Electronic Society*, pages 1–8, 2003.

[8] A. J. Lee, K. Minami, and N. Borisov. Confidentiality-preserving distributed proofs of conjunctive queries. In *Proceedings of the Fourth ACM Symposium on Information, Computer, and Communications Security*, pages 287–297, 2009.

[9] N. Li, W. Du, and D. Boneh. Oblivious signature-based envelope. *Distributed Computing*, 17(4):293–302, 2005.