Using Commutative Encryption to Share a Secret

Saied Hosseini Khayat[*]

August 18, 2008

Abstract

It is shown how to use commutative encryption to share a secret. Suppose Alice wants to share a secret with Bob such that Bob cannot decrypt the secret unless a group of trustees agree. It is assumed that Alice, Bob and the trustees communicate over insecure channels. This paper presents a scheme that uses modular exponentiation and does not require key exchange. The security of the scheme rest of the difficulty of the discrete logarithm problem.

Key words: secret sharing, commutative cryptography.

1 Introduction

Suppose Alice has a secret (e.g., a bank password) and wants to share it securely with Bob such that Bob cannot decrypt the secret without the consent of a group of trustees. In addition, consider the following assumptions:

1. Alice, Bob and the trustees can communicate only over insecure public channels.
2. The secret is not to be revealed to the trustees, nor to anyone but Bob.
3. The involved parties can be trusted in following a prescribed protocol.

The above problem can arise in commercial, corporate or military settings, and can be solved by means of the well-known secret sharing schemes (as described in Section 4), such as the Shamir threshold scheme [4]. In those schemes, the shareholders must "pool" their shares *securely* in order to unlock the secret. If the parties are remotely located, that normally requires a key exchange before execution of the protocol. In a distributed application, avoiding the hassle of secure key distribution prior to a secret sharing scheme is a worthwhile goal. In this paper, we show how a commutative encryption function, such as the modular exponentiation operation, can be used to create a keyless scheme that solves the stated problem. The security of this scheme rests on the intractability of the discrete logarithm problem. This paper is organized as follows: Section 2 describes the key idea of this paper. In Section 3, our proposed scheme is presented and its correctness and security is shown. In Section 4, the merits of the proposed scheme are discussed, and in the last section, the contribution of the paper is summarized.

[*]Electrical Engineering Department, Ferdowsi University of Mashhad, Iran. Email: *shk@alum.wustl.edu*

2 Key Idea

This paper presents a secret sharing scheme that is inspired by Shamir's keyless secret communication [3, pp.500], which utilizes the commutative property of modular exponentiation. The idea, however, works with any commutative encryption function meeting certain conditions. Shamir in [5] explored the power of commutativity in cryptography. References [1] and [2], respectively, use commutative cryptography for information sharing across databases, and for privacy in distributed data mining. The topic has received much recent fundamental consideration in [6].

In our proposed scheme, commutativity plays an important role: it allows the secret owner to undo her encryption after the trusted parties have encrypted the secret. It also allows the trusted parties to join or leave the group in any order without the sharing procedure having to be restarted.

Definition 1 *Let $\mathcal{M}$ be denote a message space and $\mathcal{K}$ denote a key space. A* **commutative encryption function** *is a family of bijections $f : \mathcal{M} \times \mathcal{K} \to \mathcal{M}$ such that for a given $m \in \mathcal{M}$ we have $f_a \circ f_b(m) = f_b \circ f_a(m)$, for any $a, b \in \mathcal{K}$.*

It is an easy exercise to show that modular exponentiation under certain conditions is a commutative encryption function.

Fact 1 *Fix a prime p. Define $f_a(m) \overset{\text{def}}{=} m^a \pmod p$. Let $m \in \mathbb{Z}_p$ and $a \in \mathbb{Z}_{p-1}$, such that $\gcd(a, p-1) = 1$. Then*

a) $f_a(m)$ is a bijection. Proof: There exists $b \in \mathbb{Z}_{p-1}$ such that $f_b \circ f_a(m) = m$. This is easily seen as follows: Given that $a \in \mathbb{Z}_{p-1}$ and $\gcd(a, p-1) = 1$, we can find $b \in \mathbb{Z}_{p-1}$ (using the extended Euclid algorithm) such that $ab = 1 \pmod{p-1}$. As a result:

$$f_b \circ f_a(m) = m^{ab} \pmod p = m^{1+k(p-1)} \pmod p = m,$$

where we have used the Fermat's theorem (if $m \in \mathbb{Z}_p$, then $m^{(p-1)} = 1 \pmod p$).

b) $f_a(m)$ is commutative. Proof: For all positive integers a, b, m, we have

$$f_a \circ f_b(m) = m^{ab} \pmod p = f_b \circ f_a(m).$$

The above fact is the key to the correctness of our keyless secret sharing scheme.

3 Proposed Scheme

3.1 Description

The proposed algorithm uses the assumption that a large prime p has been made publicly known to all (even adversaries). The prime can be made part of a technical standard for this type of applications and then be published to the world. As a result, the prime p is assumed to be accessible to everyone in an authenticated way.

In what follows, for convenience, the secret owner (Alice) is denoted by P_0. The trustees are denoted by $P_1, P_2, \cdots, P_{n-1}$. Finally, P_n denotes the recipient of the secret (Bob). We also use the word "lock" to mean "encrypt."

Our protocol has three phases: (1) Setup, (2) Locking, and (3) Unlocking. The phases are described as follows:

Setup: A large prime p made public. A secret $s \in \mathbb{Z}_p$ owned by party P_0. Each party P_i, for $i = 0, 1, \cdots, n$, has his/her own private key pair (a_i, b_i) such that $a_i, b_i \in \mathbb{Z}_p$, $\gcd(a_i, p-1) = 1$, and $a_i b_i = 1 \pmod{p-1}$.

Locking: In this phase, the secret is locked by all parties in a cascaded manner, in the following way:

1. The secret owner, P_0 (Alice), locks the secret s by doing $c_0 = s^{a_0} \pmod{p}$, and then sends c_0 to P_1.
2. For $i = 1, 2, \cdots, n-1$, the party P_i locks the secret by doing $c_i = c_{i-1}^{a_i} \pmod{p}$, and then sends the secret c_i to P_{i+1}.
3. P_n locks the secret by doing $c_n = c_{n-1}^{a_n} \pmod{p}$, and then sends c_n to P_0.
4. The secret owner P_0 now removes her lock from the secret. The result is the shared secret $s' = c_n^{b_0} \pmod{p}$. She sends s' to P_n (Bob).

Unlocking: In this phase, the secret is first unlocked by all trustees (one after another in an arbitrary order), and then in last step by P_n, in the following way:

1. P_n sets $d_i = s'$ and sends d_i to a trustee P_i, $i \in \{1, 2, \cdots, n-1\}$.
2. Each trustee P_i, in $\{P_1, P_2, \cdots, P_{n-1}\}$ (in an arbitrary sequential order), removes his lock from the shared secret by doing $d_i^{b_i} \pmod{p}$, and then sends the result to the next trustee. The last trustee in $\{P_1, P_2, \cdots, P_{n-1}\}$ removes his lock from the shared secret and sends the result to P_n.
3. At this point, P_n unlocks the secret by doing $s = d_n^{b_n} \pmod{p}$. Since at this point no one has locked the secret s. Bob (P_n) finds out the secret.

The execution of the above protocol results in the transfer of a secret from a secret owner to a second party under unanimous consent of a group of trustees, in a distributed way.

3.2 Correctness

The correctness of the scheme can be easily established by using Fact 1, as follows: At the end of a locking phase, the shared secret s' is

$$s' = s^{a_0 a_1 a_2 \cdots a_n b_0} \pmod{p} = s^{a_1 a_2 \cdots a_n} \pmod{p},$$

and at the end of the unlocking phase we have:

$$s'^{(b_1 b_2 \cdots b_n)} \pmod{p} = s^{(a_1 a_2 \cdots a_n)(b_1 b_2 \cdots b_n)} \pmod{p} = s.$$

As can be seen, the commutativity of modular exponentiation makes the order of unlocking operations unimportant. The secret s at the end is obtained by P_n.

3.3 Security

The security of the scheme rests on the computational difficulty of discrete logarithm problem (DLP). An adversary may pursue one of the following goals in breaking the scheme:

- Total break: He finds out the secret, thereby bypassing the unanimous consent requirement altogether.
- Partial break: He finds out the private keys (a_i or equivalently b_i) of one or more of the parties. This enables him to play the part of some trustees, stealing their right to consent.

The following arguments can establish the security of the scheme:

a) In the first locking hop, from P_0 to P_1, the secret owner has locked the secret s ($c_0 = s^{a_0} \mod p$) before sending it to P_1. Note that the adversary has two unknowns to search for: a_0 and s. The private keys are chosen from a huge space ($a_i, b_i \in \mathbb{Z}_p$), therefore an exhaustive search is infeasible. The adversary has no way of guessing s without knowing a_0.

b) In the subsequent hops, the adversary can eavesdrop and read c_{i-1} in transit from P_{i-1} to P_i, and he can read c_i in transit from P_i to P_{i+1}. To find out a_i, the adversary has to solve a DLP ($c_i = c_{i-1}^{a_i} \mod p$), which is infeasible. If he succeeds, he could then compute b_i, which is one of the keys necessary for unlocking the shared secret s'. He has to solve DLP for all hops in order to obtain all b_i's. On the other hand, if the adversary determines any of the b_i's, he can take the role of the corresponding parties and conspire for unauthorized unlocking of the secret. Since solving DLP is computationally hard, the above possibilities are vanishingly improbable.

c) In the last locking hop, s' is sent to P_n. The adversary now knows c_n and s'. But again he has to solve a DLP ($s' = c_n^{b_0} \mod p$) in order to find out b_0.

e) The same type of arguments are valid in the unlocking phase: The shared secret is always locked by at least one party when in transit.

f) There is a nonzero possibility that the private keys are by chance selected such that

$$\begin{aligned} a_0 a_1 &= 1 \pmod{p-1}, \text{ or} \\ a_0 a_1 a_2 &= 1 \pmod{p-1}, \text{ or} \\ \cdots &= \cdots \\ a_0 a_1 \cdots a_n &= 1 \pmod{p-1} \end{aligned}$$

This will result in having $c_i = s$ for some i. In that case, the secret s is exposed in transit from P_i to P_{i+1}. However, given the fact that the keys must be chosen from a huge space, the probability of any of the above events is vanishingly small.

Therefore, the security of the scheme was shown to rely on DLP, as claimed in the introduction.

4 Discussion

The proposed scheme has several merits.

a) It requires no keys—private or public—to be exchanged prior to the protocol. This is the main advantage of this scheme because otherwise the same goal (i.e., transfer of a secret under group consent) can be achieved using the well-known secret sharing schemes such as Shamir's, in the following way:

Alice uses the secret to produce as many shares as there are trustees. Then she encrypts all shares with the same private key K and sends the encrypted shares to her trustees. She also sends K to Bob. When Bob needs to unlock the secret, and when all the trustees agree, the trustees will send the encrypted shares to Bob. He then decrypts all the shares and uses the associated pooling algorithm to calculate the secret.

In the above protocol, the channel from Alice to Bob needs to be either private-key encrypted, or public-key encrypted. The scheme proposed in this paper requires only authenticated channels—to make sure the adversary cannot impersonate a party to take part in the protocol. (An adversary's attempt to modify messages in transit is not a concern here because that can easily be detected at the end if the secret is given a special structure.)

b) The group of trustees can be dynamic, i.e., trusted parties may join or leave the group without a need to restart the scheme. The commutative property of modular exponentiation allows one to lock or unlock the secret independent of others. The shared secret s' needs to be sent to the party who is joining (leaving) so that he can lock (unlock) it in exactly the same way as the other parties.

c) The security of the scheme relies upon the DLP which is believed to be more difficult than factoring. As a result, the scheme is expected to be more secure than a scheme using, for example, the Shamir threshold scheme together with RSA (for the private link between Alice and Bob).

d) Choosing who will receive the secret (i.e. choosing who will play Bob) can be delayed until after the locking phase. In that case, the recipient can be chosen from amongst the people we call trustees. The scheme remains essentially unchanged.

5 Conclusion

It was shown how to use the commutative property of the modular exponentiation operation to transfer a secret securely from one party to another under unanimous consent of a group of trustees. The main advantage of the proposed scheme is the lack of requirement for a pre-exchanged key. The scheme is expected to offer good security because it relies on the discrete logarithm problem.

References

[1] Agrawal, R., Evfimievski, A., and Srikant, R., "Information sharing across private databases," *International Conference on Management of Data (ACM SIGMOD),* ACM Press, 2003.

[2] Clifton, C., Kantarcioglu, M., Vaidya, J., Lin, X., and Zhu, M. Y. "Tools for privacy preserving distributed data mining," *SIGKDD Explorations* 4, 2, January 2003.

[3] Menezes, A., Oorschot, P., Vanstone, S., *Handbook of Applied Cryptography,* CRC Press, 1997.

[4] Shamir, A., "How to Share a Secret," *Communications of The ACM,* Vol. 22, No. 11, November 1979.

[5] Shamir, A., “On the Power of Commutativity in Cryptography,” *ICALP80,* July 1980.

[6] Weis, Stephen A., *New Foundations for Efficient Authentication, Commutative Cryptography, and Private Disjointness Testing,* MIT PhD Dissertation, May 2006.