Information Security Using Threshold Cryptography With Paillier Algorithm

1Machha.Narender, 2G.N.Ramesh, 3P.Ranganath

Abstract—The dynamic and cooperative nature of ad hoc networks present challenges in securing these networks. There are recent research efforts in securing ad hoc networks. Among these, security approaches, there are threshold cryptography and authentication. In this paper, we survey the threshold cryptography based schemes and the authentication schemes that have been proposed to secure ad hoc networks. We conclude this paper and identify the challenges and open research areas associated with each of these approaches. The idea of threshold cryptography is to protect information (or computation) by fault-tolerantly distributing it among a cluster of cooperating computers. First consider the fundamental problem of threshold cryptography, a problem of secure sharing of a secret. A secret sharing scheme allows one to distribute a piece of secret information among several servers in a way that meets the following requirements: (1) no group of corrupt servers (smaller than a given threshold) can figure out what the secret is, even if they cooperate; (2) when it becomes necessary that the secret information be reconstructed, a large enough number of servers (a number larger than the above threshold) can always do it.

I. INTRODUCTION

Threshold Cryptography is the art of chopping a secret into little bits. Only by possessing more than a threshold number of bits of the secret can the secret be determined. Algorithms exist to break any secret up such that at least and exactly M out of N holders of pieces of the secret must get approval (and their partial secret or key) inorder to compute the total secret (e.g. 3 of 5, 3 of 12, 5 of 12, etc.). Removing probability has acost, though... a secret must be broken into C(N,M-1) pieces and each holder carries (NM+1)/N parts of the whole key... so '3 of 12' is more expensive per-node than '5 of 12'. (These numbers come from the pigeonhole principle and constraints: any piece 'pK' must be found on NM+1 holders so that access to a full M secret holders guarantees 'pK' will be known, whilst accessing to M-1 computers must guarantee that there is at least one piece not found, so 'pK' must NOT be with the other M-1 computers. The minimum number of components 'pK' elements todo this is (N) Choose (M-1). Individual pElements can be made artificially large in order to subvert guessing of one or two missing pieces; the combinatorial function needn't be straightforward appended. However, the computation and storage cost of this approach is high, and it may do well to combine it with some straightforward split-and-distribute as listed above; e.g. splitting the 'require 5 of 7 pieces' to 'more than 7' people is the natural extension to splitting 'require 3 of 3 pieces' to '12 people'. The combined effect can avoid the massive costs of splitting and storing, say '5 of 20' parts (C(20,4) unique parts, every node holding ~16/20ths of the secret, vs. C(7,4) parts with each node holding 3/7ths of the total secret). The main advantage of mixing in this algorithmic division approach is in achieving better guarantees as to redundancy and survivability while simultaneously increasing the number of users some must access to possess the whole secret. E.g. for the other approach, to require 5 users would require splitting the key into 5 pieces and dividing that up among, say, 15 people; it wouldn't be possible to access to 5 people to gain the secret, and the secret could be lost by losing 3 people. Splitting it to '5 of 7 first, then dividing the 7 chunks among 14 people results in 2 different people having a copy of any given chunk, and the secret won't be lost before losing 6 people (losing three whole chunks).

As a security measure, Threshold Cryptography requires that many systems must be compromised prior to taking control of a secret, inherently including resistance to snooping or abuse by any super users of the computation resource (who would have the ability to do so if the secret were wholly on one system). It also provides inherent redundancy of the secret... e.g. if you can guarantee that it takes at least and at most 5 of 12 secret holders to build the secret, you can guarantee that a failure of up to 7 systems is tolerable without failure. With a probabilistic split, you can easily calculate a percentage chance that the data is unavailable for at least a given length of the network... and, with some intelligent split of components, you can guarantee that at least some count of nodes must be lost before the data has any chance of being lost. In the case of authorization to access a different system (e.g. to control a power plant), security can be increased further by demanding that a few parts of the approval come from particular people that are known to be accessible... and changing these people at regular intervals. This makes it much more difficult to gain access even by compromising the systems... because you can't easily know which particular systems are under your control.

II. MOTIVATION

The strongest reason for using this mechanism is that a secret might need to be available to users that can only provide a -certificate-authorizing access to a file or service, and the primary...
encryption isn't against any key with which individuals share long-term access (there is no shared key). E.g., one can use Threshold Cryptography to encrypt files or split keys requiring, say, either 'Secret' clearance with 'Power Grid' specialization, or 'Top Secret' clearance, represented as a certificate signed by a government master key in not in expiration, and any individual that can prove to M of N systems that he or she possesses the necessary clearances will be provided the capability to actually perform the task. Key distribution is a difficult problem, doubly so when you won't trust that any one key distribution server hasn't been compromised; Threshold Cryptography is one of the more elegant answers to that particular problem. Any useful extension of secret sharing is function sharing. Its main idea is that a highly sensitive operation, such as decryption or signing, can be performed by a group of cooperating servers in such a way that no minority of servers is able to perform this operation by themselves, nor would they be able to prevent the other servers from performing the operation when it is required.

In many real-life situations, we don't believe that any given person can be trusted, and we may even suspect that a big fraction of all people are dishonest, yet it is reasonable to assume that the majority of people are trustworthy. Similarly, in on-line transactions, we may doubt that a given server can be trusted, but we hope that the majority of servers are working properly. Based on this assumption, we can create trusted entities. A good example of an application whose security could be greatly improved with a threshold solution is a network Certification Authority, a trusted entity that certifies that a given public key corresponds to a given user. If we trust one server to perform this operation, then it is possible that as a result of just one break-in, no certificate can any longer be trusted. Thus it is a good idea to distribute the functionality of the certification authority between many servers, so that an adversary would need to corrupt half of them before he can forge a certificate on some public key.

Goals: In the threshold setting, we would like to implement, via efficient protocols, the most secure cryptosystems and signature schemes. We would also like to make our protocols secure in the strongest possible model of faults. The following are some of the various considerations we make when modeling computer faults.

A. The Size Of The Threshold

What fraction of the servers can be corrupted by the adversary without any harm to the service (e.g., signature or decryption) that these servers implement?

B. Efficiency Considerations

How much communication, storage, and computation do these fault-tolerant protocols require?

C. Model Of Communication

How realistic are the requirements we place on it? Do we require synchronous or partially synchronous communication, authenticated broadcast and secure links between servers?

D. Type Of Adversary We Tolerate

How does the adversary choose which players to corrupt? Can a server securely erase its local data so that it cannot be retrieved by the adversary once the server is infiltrated?

III. PAILLIER CRYPTOSYSTEM ALGORITHM

A. Key Generation

i. Choose two large prime numbers p and q randomly and independently of each such that gcd(p, q - 1) = 1. This property is ensured if both primes are of equivalent length, i.e., p, q ∈ [0, 1]^{security parameter}_{security parameter}_{security parameter}_{security parameter}.-1 for security parameter_{security parameter}_{security parameter}_{security parameter}. Compute n = pq and \( \lambda = \text{lcm}(p - 1, q - 1) \)

ii. Select random integer g where \( g \in \mathbb{Z}_n^{\ast} \)

iii. Ensure \( n \) divides the order of \( g \) by checking the existence of the following modular multiplicative inverse \( \mu = (L(g^\lambda \mod n^2))^{-1} \mod n \) Where function L is defined as \( L(u) = \frac{u - 1}{n} \). Note that the notation does not denote the modular multiplication of \( a \) times the modular multiplicative inverse of \( b \) but rather the quotient of \( a \) divided by \( b \), i.e., the largest integer value \( v \geq 0 \) to satisfy the relation \( a \geq vb \).

a. The Public (Encryption) Key Is \((N, G)\).
b. The private (decryption) key is \((\lambda, \mu)\).

If using p, q of equivalent length, a simpler variant of the above key generation steps would be to set \( n = p + 1, \lambda = \varphi(n) \), and \( \mu = \varphi(n)^{\mu} \mod n \). where \( \varphi(n) = (p - 1)(q - 1) \).

IV. ENCRYPTION

i. Let \( m \) be a message to be encrypted where \( m \in \mathbb{Z}_n \)

ii. Select random \( r \) where \( r \in \mathbb{Z}_n^{\ast} \)

iii. Compute cipher text as \( c = g^m \cdot r^n \mod n^2 \)

V. DECRYPTION

i. Cipher text \( t \in \mathbb{Z}_n^{\ast} \)

ii. Compute message \( m = L(c^\lambda \mod n^2) \cdot \mu \mod n \)

As the original paper points out, decryption is "essentially one exponentiation modulo \( n^3 \)."

VI. HOMOMORPHIC PROPERTIES

A notable feature of the Paillier cryptosystem is its homomorphic properties. As the encryption function is additively homomorphic, the following identities can be described:
A. Homomorphic Addition Of Plaintexts

The product of two cipher texts will decrypt to the sum of their corresponding plaintexts,
\[ D(E(m_1, r_1) \cdot E(m_2, r_2)) \mod n^2 = m_1 + m_2 \mod n. \]

The product of a cipher text with a plaintext raising \( g \) will decrypt to the sum of the corresponding plaintexts,
\[ D(E(m_1, r_1) \cdot g^{m_2} \mod n^2) = m_1 + m_2 \mod n. \]

B. Homomorphic Multiplication Of Plaintexts

An encrypted plaintext raised to the power of another plaintext will decrypt to the product of the two plaintexts,
\[ D(E(m_1, r_1)^{m_2} \mod n^2) = m_1m_2 \mod n. \]

More generally, an encrypted plaintext raised to a constant \( k \) will decrypt to the product of the plaintext and the constant,
\[ D(E(m_1, r_1)^k \mod n^2) = km_1 \mod n. \]

However, given the Paillier encryptions of two messages there is no known way to compute an encryption of the product of these messages without knowing the private key.

VII. Semantic Security

The original cryptosystem as shown above does provide semantic security against chosen plaintext attacks (IND-CPA). The ability to successfully distinguish the challenge cipher text essentially amounts to the ability to decide composite residuosity. The so-called decisional composite residuosity assumption (DCRA) is believed to be intractable. Because of the aforementioned homomorphic properties however, the system is malleable, and therefore does not enjoy the highest echelon of semantic security that protects against adaptive chosen-cipher text attacks (IND-CCA2). Usually in cryptography the notion of malleability is not seen as an "advantage," but under certain applications such as secure electronic voting and threshold cryptosystems, this property may indeed be necessary. Paillier and Pointcheval however went on to propose an improved cryptosystem that incorporates the combined hashing of message m with random r. Similar in intent to the Cramer-Shoup cryptosystem, the hashing prevents an attacker, given only c, from being able to change m in a meaningful way. Through this adaptation the improved scheme can be shown to be IND-CCA2 secure in the random oracle model.

VIII. Applications

A. Electronic Voting

Semantic security is not the only consideration. There are situations under which malleability may be desirable. The above homomorphic properties can be utilized by secure electronic voting systems. Consider a simple binary ("for" or "against") vote. Let m voters cast a vote of either 1 (for) or 0 (against). Each voter encrypts their choice before casting their vote. The election official then knows that n people voted for and \( m-n \) people voted against. The role of the random r ensures that two equivalent votes will encrypt to the same value only with negligible likelihood, hence ensuring voter privacy.

B. Electronic Cash

Another feature named in paper is the notion of self-blinding. This is the ability to change one cipher text into another without changing the content of its decryption. This has application to the development of electronic cash, an effort originally spear-headed by David Chaum. Imagine paying for an item online without the vendor needing to know your credit card number, and hence your identity. The goal in both electronic cash and electronic voting is to ensure the e-coin (likewise e-vote) is valid, while at the same time not disclosing the identity of the person with whom it is currently associated.

IX. Conclusion

A new threshold signing scheme is proposed in this project that when combined with Shared Paillier secret keys generation will lead us to a complete solution for the Threshold Paillier problem. The complete solution has also been implemented successfully in this project.

X. References