JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 27, 1303-1313 (2011)

Group Oriented Renewal of Secrets and Its Application to Secure Multicast*

J. A. M. NARANJO, L. G. CASADO AND J. A. LÓPEZ-RAMOS
Department of Computer Architecture and Electronics
University of Almería
Almerí, 04120 Spain

This paper introduces a multicast method for renewing secrets which are shared by a set of hosts. The method is centralized, secure, efficient, scalable to a reasonable size and compatible with any multicast topology configuration underneath. It can be used to achieve privacy in a centralized multicast overlay. Additionally, the method can be used to renew an asymmetric key pair when a cryptosystem based on discrete logarithm is used. Knowledge of the public key is then restricted to the group it is communicated to. Security and scalability are discussed, and a comparison with other well-known alternatives is shown.

Keywords: communication/networking and information technology, network operations, data encryption, secure group communication, multicast

1. INTRODUCTION

Application level multicast overlays have progressively filled the space left by IP multicast protocols, such as IGMP, since they stand as a cost-effective and easy-to-deploy alternative: they run on the OSI application layer, while IP multicast is held on the network layer [1].

According to the source of the transmitted data, multicast schemes can be divided into *one-to-many* and *many-to-many*. In the first case, the source is one entity only. A typical scenario is an IPTV or P2PTV platform, in which clients receive a TV signal from a Contents Server via Internet. In the second case, many clients (or all) act both as source and receiver. Multiconferences are examples of this.

Still, there is an important issue that needs a definitive solution: privacy in multicast communications. The typical approach in order to maintain a continuous private communication with a group of hosts is to establish a common secret key to encrypt the information. The key is then refreshed periodically to prevent attacks from outsiders, or even insiders. This solution is named *secure multicast*. Huge efforts are still directed towards finding efficient, scalable and secure enough methods. Depending on how key distribution and management are carried out, secure multicast schemes can be divided in *centralized* and *distributed*. Centralized schemes depend directly of a single entity, a key server, to release every cryptographic key. In a distributed approach the key distribution process is more complex, usually involving entities that act as subservers and manage subgroups of users. Re-encryption of the information is needed in some cases.

Received January 27, 2010; revised June 13 & September 7, 2010; accepted September 28, 2010.
Communicated by Chin-Laung Lei.
* J. A. M. Naranjo and L. G. Casado were supported by the Spanish Ministry of Science and Innovation (TIN 2008-01117), L. G. Casado was also supported by funds of Junta de Andalucía (P08-TIC-3518); J. A. López-Ramos was supported by the Spanish Ministry of Science and Innovation (TEC2009-13763-C02-02) and Junta de Andalucía (FQM 0211).

The heart of the matter in secure multicast is the usual necessity for perfect *backward* and *forward secrecy* (a client should not be able to decrypt any ciphered information transmitted before her arrival to the system and after her departure, respectively). This requirement forces to refresh all the keys used to encrypt the data whenever a client joins/leaves the system. If the members set is large and join/leave events occur frequently then the refreshment operation may become an important bottleneck. Many schemes have been proposed as a solution, and the following paragraphs summarize some of them.

The *Secure Lock* mechanism is proposed in [2]. It uses the Chinese Remainder Theorem to solve systems of congruences, each congruence corresponding to a client. The solution to the system, the *lock*, is broadcasted to clients, which obtain the key from it. The process, though smart and elegant, quickly becomes inefficient as the audience grows, due to the costly computations required at the server side [3].

RFC 2627 [4] presents several approaches to the problem. Among all, the *Hierarchical Tree Approach* (HTA) is the recommended option. It uses a logical tree arrangement of the members in order to facilitate key distribution: every member is considered to be at a leaf, and the internal nodes represent keys. Every member knows the keys in her path to the root. The root key is used as a common key for encryption of the information, since everyone knows it. The benefit of this idea is that the storage requirement for each client and the number of transmissions required for rekeying are both logarithmic in the number of members.

Contemporary to HTA, the scheme proposed in [5] is very similar in its tree approach. Its novelty relies on the proposal of three different rekeying strategies: *user-oriented*, which uses many short messages for a rekeying operation, *key-oriented*, which broadcasts more messages at a lower computational cost, and *group-oriented*, which employs one sole message of larger size.

In [6], a divide-and-conquer extension to Secure Lock is proposed. It combines the Hierarchical Tree Approach and the Secure Lock: members are arranged in a HTA fashion while Secure Lock is used to refresh keys on each tree level. Therefore, the number of computations required by Secure Lock is reduced.

One-way function trees (OFT) [7] are an extension to the hierarchical tree approach. A key tree is also used, but in this case every internal key is built depending on its two descendant keys: both descendants are *blinded* by means of a one-way function and the results are wired to the input of a mixing function. The tree, therefore, is built on a bottom-top fashion. Members, placed at the leaves, know their own key, the keys in the path to the root and the blinded sibling keys of the path. Thanks to that, the amount of information needed by a member to recompute the whole path of keys to the root is smaller and rekeying messages are shorter (approximately half of HTA's).

The ELK protocol [8] is also an improvement of HTA. It is similar to OFTs in the sense that intermediate keys are generated from its children, but pseudo-random functions (PRFs) are used rather than one-way functions. Thanks to PRFs and to timely rekeying no broadcast of information is needed in join events (only unicast messages for tree maintenance). Additionally, ELK addresses message loss tolerance by introducing the concept of *hints*: small pieces of information attached to broadcast data packets that allow recovering lost rekey information.

An IETF Working Group, MSEC [9], is currently working in a set of protocols to standardize secure multicast. They've focused, on an initial stage, in IP-layer centralized

multicast, assuming the presence of groups and a single trusted entity in each one.

This paper presents a centralized secure multicast scheme with the following features:

- Suitable for all topologies. No need for node hierarchies, though they can be supported.
- No need for re-encryption.
- Only one secret piece of info is held by each client. We call these pieces *member tickets*.
- Cost-effective and easy to deploy.

An additional and interesting feature shows up when using the scheme along with a discrete logarithm based cryptosystem (for example ElGamal [10]). In this scenario, the scheme can be used to refresh the server's key pair in a semi-private manner: publication of the public key is restricted to the group it is sent to, remaining obscure to outsiders.

The rest of the paper is organized as follows. Section 2 introduces the scheme and the theory underneath, while sections 3 and 4 discuss some security and efficiency considerations and compare the scheme with other well-known alternatives mentioned above, respectively. Finally, section 5 presents the conclusions of the paper.

2. DESCRIPTION OF THE SCHEME

The scenario assumed is a Key Server and a set of members (other hosts) that either send or receive multicast messages. Data communications can therefore be either one-to-many or many-to-many, and are encrypted with a symmetric key. Communications related to key refreshment are always one-to-many, from the Key Server to the members. Members can enter and leave the system at any time, and the key must be refreshed upon member arrival or departure to achieve perfect backward and forward secrecy, respectively. In addition, a key refreshment must be performed periodically to prevent statistical or brute force attacks. Any multicast topology can be used underneath: that is out of the scope of this paper.

Let us assume r is the symmetric encryption key to be multicast, and that there are n members at a given time. The following paragraphs explain how the scheme works.

When a member i joins, the Key Server assigns it a member ticket, x_i. Every ticket is a large prime[1] and is communicated to the corresponding member under a secure channel: SSL/TLS, for example. This communication is made once per member only, so it does not affect global efficiency. All tickets must be different from each other, at least during a relatively wide period of time. Note that x_i is known only by its owner (and the Key Server), and r is shared by all members (and the Key Server).

The distribution of r is done as follows.

1. The Key Server selects:

- m and p, large prime numbers, such that p divides $m - 1$.
- k and δ, such that $\delta = k + p$ and $\delta < x_i$, $i = 1, \ldots, n$.
- g that verifies $g^p = 1 \bmod m$ (such a value is easy to find[2]).

[1] Strictly, it is sufficient that all x_i are coprime and greater than δ. Having all x_i prime makes the factorization of L harder (L will be introduced immediately).

[2] Once the Key Server has chosen $m = p \cdot q + 1$, a primitive element from Z_m, say a, is chosen. Then $g = a^q \bmod m$.

The encryption key r consists of $r = g^k \bmod m$.

2. The Key Server calculates $L = \prod_{i=1}^{n} x_i$. L is kept private in the Key Server.
3. The Key Server finds u, v, by means of the Extended Euclidean Algorithm [11], such that

$$u \cdot \delta + v \cdot L = 1. \tag{1}$$

4. The Key Server multicasts (makes public) g, m and u on plain text.
5. Each member i calculates $u^{-1} \bmod x_i = \delta$ and $g^{\delta} \bmod m = g^k \bmod m = r$. The length of r, by definition, cannot exceed that of m.

New values for m, g, p and/or k must be chosen for each refreshment of r. Note that δ, u and v depend on them and will change as they do.

In an event-driven refreshments scenario, the refreshment operation must be performed at least in the following two cases:

Member *j* joins: x_j is included in the product L.
Member *j* leaves: The leaving member should not be able to decrypt contents anymore. This is achieved by dividing L by x_j and refreshing r afterwards.

Note that it is not necessary to recomputed L from scratch: only a single multiplication or division is required for a join or a leave, respectively. Finally, for security reasons, the Key Server might decide to refresh r after a long period of time with no members joining or leaving.

2.1 Proof of Correctness

Given that $\delta < x_i$, $i = 1, \ldots, n$ and with every x_i prime (or coprime at least), it is clear that:

$$gcd(\delta, x_i) = 1,\ i = 1, \ldots, n \tag{2}$$

and hence,

$$gcd(\delta, L) = 1. \tag{3}$$

Eq. (3) ensures the existence of $u, v \in \mathbb{Z}$ such that $\delta \cdot u + v \cdot L = 1$, from where it is deduced that $\delta \cdot u \equiv_{x_i} 1 \Rightarrow u^{-1} \equiv_{x_i} \delta$, $i = 1, \ldots, n$. The Chinese Remainder Theorem guarantees that the solution for $u^{-1} \bmod x_i = \delta$ and $\delta < x_i$, $i = 1, \ldots, n$, is unique.

The value $r = g^k \bmod m$ is obtained as shown next:

$$g^{\delta} \equiv_m g^{k+p} \equiv_m g^k \cdot g^p \equiv_m g^k \cdot 1 \equiv_m g^k, \tag{4}$$

g is public, but the use of δ assures that an outsider will not be able to guess k and, therefore, r.

2.2 Disclosure of an Asymmetric Key Pair

The scheme proposed in this paper can be used for an additional purpose: the refreshment of asymmetric key pairs and the disclosure of the public part. Recall, from the algorithm presented at the beginning of this section, that the encryption key delivered to the audience has the form $r = g^k \bmod m$. This is similar to an ElGamal public key. An ElGamal key pair has the form:

K_{pub}: g, m, $g^k \bmod m$,
K_{priv}: k.

Therefore the method can be seen as a way of *controlling the disclosure of discrete logarithm based public keys*: the public key is only communicated to a closed group of recipients. The key pair can then be used for signature purposes (only the Key Sever knows the private part k) or for encryption of messages addressed to the Key Server.

3. SCALABILITY AND SECURITY CONSIDERATIONS

3.1 Scalability

Kruus [12] suggests five issues that a multicast key management protocol must address. They are:

1. efficiency in initial keying,
2. efficiency in rekeying,
3. computational requirements,
4. storage requirements,
5. scalability.

There is no difference in our scheme between first time keying (requirement 1) and further rekeying operations. Rekeying operations are simple (requirement 2): the Key Server generates a single message which is injected into the multicast network on plain text, since only authorized members will be able to process it correctly. Requirements 3, 4 and 5 are discussed next.

We can observe that L will be large, given that $L = \prod_{i=1}^{n} x_i$. So will be u (recall Eq. (1)). To estimate it, assume for the rest of the paper that every x_i value is stored in an unsigned binary data type of b bits. The greatest value that can be represented is $2^b - 1$. Assume also that there are n members. The maximum length of L is then $n \cdot b$ bits. That is also the maximum length of u.

As an example, for $b = 64$ and $n = 1000$ the maximum length of u is 64000 bits $\approx$ 8 KBs. Though that is an affordable message length for many devices (requirement 4), a shorter message would be desirable.

From the previous consideration we assume that Kruus' requirements 3 and 5 are the weakest points of our scheme. The solution that allows to overcome these problems consists of dividing the audience into disjoint subgroups while delivering the same encrypt-

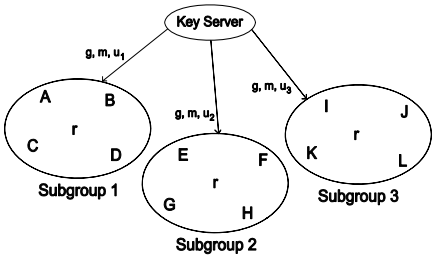

Fig. 1. The subgroup extension to the scheme. Capital letters denote members. *r* is the multicasted encryption key.

tion key to all of them. This can be done by running a different instance of the algorithm for every subgroup, with some variations: values *m*, *g*, *p* and *k* are the same for every instance (and so is δ therefore), but each *L* contains only the tickets of the members within the given subgroup. With this modification the same key is delivered to every subgroup by means of several shorter messages, which is more convenient in terms of efficiency.

For the rest of the paper we assume there are *s* subgroups, each one with a similar number of members. Still, the join and leave operations require the whole set of members to obtain a new key, therefore *s* refreshment messages (*g*, *m* and the corresponding *u*) must be computed and multicasted now; each one for a different subgroup. Fig. 1 shows an example. Note that all subgroups receive the same *g* and *m* (which guarantees that the same key is obtained) but a different *u*, due to the use of different *L* values: $L_1 = x_A x_B x_C x_D$, $L_2 = x_E x_F x_G x_H$ and $L_3 = x_I x_J x_K x_L$. Adopting this approach brings many benefits, even though the final bandwidth requirements do not change.

First, it is obvious that, for a fixed number of members, the length of *u* values decreases linearly as the number of subgroups increases. In the previous example, arranging the same audience in 20 groups of 50 members would yield 20 messages of 3200 bits = 400 Bs maximum, each one shorter than a typical X.509 certificate. Shorter messages can be handled more easily and quickly by the recipients. This means less hardware requirements.

Second, the message generation process that takes place at the Key Server can be sped up. Every different *u* can now be computed by a separate process, which may run concurrently with the others. That is specially appropriate for current multi-core processors. The whole process can be sped up by almost *s* times if the software is properly tuned.

This subgroup approach provides a better scalability, allowing to increase the maximum number of clients that can be handled. As a remark, users should be assigned to subgroups in a balanced way in order to keep refreshment messages as short as possible. That raises other issues, such as the problem of rebalancing subgroups after a leave avalanche, for example.

3.2 Security

Security in the distribution of *r* relies on the unfeasibility of calculating the right δ in a reasonable time if a valid x_i is not known by the attacker (recall that values for Eq. (1) are unique). The privacy of *k* and *p* is guaranteed if:

- a sufficiently large value is chosen for m,
- p and q have a similar bitlength (recall that $m - 1 = p \cdot q$).

In that case factorizing $m - 1$ will be more difficult. Additionally, a strong prime can be chosen for m. Next, security is discussed considering three different types of attacker.

Security against a passive adversary: First, we assume the presence of an adversary who neither has nor had a valid ticket. Her intention is to learn about the secret value that is being disclosed to legal members. Those members, as was explained in step 5 of the algorithm in section 2, compute a modular inverse by using their corresponding ticket as the modulo. It is clear that the adversary cannot compute that inverse, therefore she cannot gain access to the secret. Her only option is to brute-force the scheme by trying all the 2^b possibilities (recall b is the ticket bitlength).

Security against an active adversary: Next we consider two types of active adversary. The first one is a legal member, say i, who aims at obtaining one or more legal tickets from other members. Prior to discussing this situation, note that the product L is not public in order to make attackers' work more difficult. Now, in case L was discovered and factorized by the adversary, she would gain access to every member ticket. But such a factorization is impractical by means of a brute force attack. Still, she might be tempted to factorize $\frac{u \cdot \delta - 1}{x_i}$, but the problem of factorizing such a value is equivalent to factorizing L. Having said this we can claim that our scheme rejects security by obscurity, since it is resilient even with L public.

The second type of active adversary is a former user who still holds her old ticket. It is straight that she might try to intercept a refreshment message and use her own ticket expecting that it has been reassigned to a new member. If that was the case it is clear that she would be successful in recovering the secret r. The way to prevent this attack is to never reuse tickets (or for a large period of time at least).

3.3 Avoiding Man in the Middle and Denial of Service Attacks

Man in the middle attacks are a serious threat that must be carefully taken into account when a communication protocol is designed. In our scenario, an attack of this kind means impersonating the Key Server by intercepting key refreshment messages and forging new ones.

Using the scheme "as is", a fake u can be forged if the attacker knows the ticket x_i of one user at least: she may compute her own L by multiplying the known x_is. After that, she may generate fake u, g and m values and send them to the owners of the compromised tickets. Those owners would not be able to detect the fraud and would mistake the attacker for the Key Server. Obviously, the attack would only work against the exposed owners.

In case an attacker does not know any ticket she can always perform a denial of service attack (DoS), by replacing g and m by random values so the secret computed by members is different from the actual one.

It is clear then that an authentication mechanism is needed for Key Server messages (this problem is common to every communication protocol). The simplest and most straight is digital signature. In that case the Key Server should own a public/private key pair to sign refreshment messages. The public part of this new key pair should be known by all members in the system, independently of the subgroup they belong to.

4. COMPARISON WITH OTHER MULTICAST SCHEMES

Table 1 compares our scheme with other well-known centralized alternatives, which were briefly presented in section 1. More concretely, we consider the Hierarchical Tree Approach (HTA) in its "multiple keys per message" version [4], the Secure Lock extended with HTA [6], OFT [7] and ELK [8]. Our scheme is analyzed in its subgroups version (see section 3.1). Some of the information shown in the Table 1 was taken from [13, 14], while the notation used is explained in the Table 1 itself. In order to make a fair comparison three assumptions were made:

- both backward and forward secrecy are provided,
- trees (where applicable) are balanced and full,
- subgroups (where applicable) are balanced and full.

Join and leave events are for a single member. Unicast communications are not shown in the table, since they do not affect global efficiency.

Table 1. Secure multicast schemes comparison. n is the number of members, s is the number of subgroups where applicable, d is the tree degree and h is the tree depth where applicable.

	HTA	Secure Lock + HTA	OFT	ELK	Ours
Number of keys stored in Key Server	$d^h - 1$	$d^h - 1$	$2n - 1$	$2n - 1$	n
Number of keys stored in member	$h + 1$	$h + 1$	$h + 1$	$h + 1$	1
Number of messages per Join	$(d - 1)h$	h	$h + 1$	0	s
Number of messages per Leave	$(d - 1)h$	h	$h + 1$	h	s

Results depend on several variables. The number of members is, obviously, the most important of them all. For tree-based schemes the degree is also a decisive factor since it determines the number of intermediate keys. In our scheme, the size and number of subgroups are the main variables to pay attention to.

Regarding storage, our scheme has the lowest requirements: the amount of information to be stored by the Key Server is directly proportional to the number of users. The tree based schemes must maintain a set of keys for the intermediate logical nodes of the tree (note that $2n - 1$ is equivalent to $d^h - 1$ for the case $d = 2$). Subsets of those keys must be held by the corresponding members, too. In our scheme members only need to store their own ticket.

ELK is ahead of the rest in join events. The combination of a logical tree layout and PRFs allows all members to recompute their key path to the root without any broadcast

communication, though some unicast messages are occasionally required for member reallocation. Our scheme sends a broadcast message per group. HTA, SecureLock + HTA and OFT need a message per tree level.

Leave operations require, in most cases, the same number of messages than joins. Tree-based schemes may need, again, some unicast messages for layout maintenance purposes. Our scheme needs no unicast messages.

It is worthy to remark that tree-based schemes need members to be aware of their location within the logical tree: they must held the intermediate keys that connect them to the tree root (the group key), and recompute all the way back whenever the tree is renewed. Members in our scheme only need to know which subgroup they belong to. This makes member computations simple and fast.

Finally, flash crowds should be considered too. They occur when large sets of users enter or leave the system in a short interval of time. A typical example are pay-per-view massive interest TV events (*e.g.* soccer matches). At the beginning of the event a great number of users enter the system, which forces to constantly refresh the group key. At the end of the event those users leave the system, therefore a great number of leave operations must be performed.

Tree-oriented schemes may suffer in flash crowd scenarios since the tree structure needs to be continuously rearranged to avoid degeneration, specially at leave events. On the contrary, our scheme can handle subgroups management very efficiently: only multiplications or divisions are needed and there is no structure to maintain. At the start of an event new subgroups can be established if needed, while at the end, the remaining users can be quickly reallocated so the number of subgroups is reduced.

5. CONCLUSIONS

This paper introduces a novel secure multicast rekeying scheme. It is centralized, secure and reasonably efficient. Any overlay topology is supported, and there is no need for intermediate trusted nodes. Its correctness is proven, while efficiency and security are discussed: a solution for achieving a better scalability is provided and several types of adversary are considered. Finally, a comparison with other well-known alternatives has been discussed.

An interesting feature of the scheme is also presented: it can be used as a way to *control the disclosure of public keys*. The secret delivered has the form of an ElGamal public key, and the scheme guarantees that no entity other than the legal recipients can gain access to it as long as there is no information leakage.

Our plans for the future include implementing and running the scheme on a test bed so proper performance analyses can be done. Additionally, other possible applications, such as mobile networking, ubiquitous computing and related technologies will be considered.

REFERENCES

1. H. Zimmermann, “OSI reference model – The ISO model of architecture for open systems interconnection,” *IEEE Transactions on Communications*, 1980, pp. 2-9.

2. G. H. Chiou and W. T. Chen, "Secure broadcasting using the secure lock," *IEEE Transactions on Software Engineering*, Vol. 15, 1989, pp. 929-934.
3. P. S. Kruus and J. P. Macker, "Techniques and issues in multicast security," in *Proceedings of Military Communications Conference*, 1998, pp. 1028-1032.
4. D. Wallner, E. Harder, and R. Agee, "Key management for multicast: Issues and architectures," RFC 2627, 1999.
5. C. K. Wong, M. Gouda, and S. S. Lam, "Secure group communications using key graphs," *IEEE/ACM Transactions on Networking*, Vol. 8, 2000, pp. 16-30.
6. O. Scheikl, J. Lane, R. Boyer, and M. Eltoweissy, "Multi-level secure multicast: The rethinking of secure locks," in *Proceedings of International Conference on Parallel Processing Workshops*, 2002, pp. 17-24.
7. A. T. Sherman and D. A. McGrew, "Key establishment in large dynamic groups using one-way function trees," *IEEE Transactions on Software Engineering*, Vol. 29, 2003, pp. 444-458.
8. A. Perrig, D. Song, and J. D. Tygar, "Elk, a new protocol for efficient large-group key distribution," in *Proceedings of IEEE Symposium on Security and Privacy*, 2001, pp. 247-262.
9. Msec working group, http://www.ietf.org/dyn/wg/charter/msec-charter.html.
10. TaherElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," in *Proceedings of Crypto on Advances in Cryptology*, 1985, pp. 10- 18.
11. A. Menezes, P. van Oorschot, and S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, Boca Raton, FL, 1996.
12. P. S. Kruus, "A survey of multicast security issues and architectures," in *Proceedings of the 21st National Information Systems Security Conference*, 1998, pp. 5-8.
13. K. C. Chan and S. H. G. Chan, "Key management approaches to offer data confidentiality for secure multicast," *IEEE Network*, Vol. 17, 2003, pp. 30-39.
14. S. Rafaeli and D. Hutchison, "A survey of key management for secure group communication," *ACM Computing Surveys*, Vol. 35, 2003, pp. 309-329.

J. A. M. Naranjo is currently a Ph.D. candidate at the Department of Computers Architecture and Electronics, University of Almería. He received his M.S. from University of Almería in 2008, and a grant from the Spanish Ministry of Science and Research to in 2009. His research interests include applied cryptography and information privacy and security, specially in communications.

L. G. Casado received his B.S. degree in Computer Sciences in 1992 from University of Granada and his Ph.D. degree in 1999 from University of Málaga. He is currently an Associate Professor at the University of Almería, Department of Computers Architecture and Electronics, and the leader of TIC-146 (PAI 7069) research group (http://www.hpca.ual.es). Additionally he participates in several national research projects. His main research interests are global optimization, parallel architectures and algorithms and security in communications.

J. A. López-Ramos is an Associate Professor at the University of Almeria. He received a Ph.D. in Mathematics in the University of Almeria in 1995. He is specialist in Homological Algebra and Number Theory and his current researching interest includes applications of Algebra to Cryptography and Coding Theory. He has participated in several researching projects financed by some international organisms, including NATO and UE and collaborates as a consultant for some private companies.