The ISC Int'l Journal of Information Security

ISeCure

January 2013, Volume 5, Number 1 (pp. 37–53)
http://www.isecure-journal.org

Design and Formal Verification of DZMBE+[☆]

Mahdi Soodkhah Mohammadi [1,*], Abbas Ghaemi Bafghi [1]

[1] *Data and Communication Security Laboratory, Computer Department, Engineering Faculty, Ferdowsi University of Mashhad, Mashhad, Iran*

ARTICLE INFO.

Article history:
Received: 31 December 2012
Revised: 25 July 2013
Accepted: 31 July 2013
Published Online: 9 October 2013

Keywords:
Broadcast Encryption, Secure Multiparty Computation, Threshold Secret Sharing, Formal Methods, Applied pi Calculus.

ABSTRACT

In this paper, a new broadcast encryption scheme is presented based on threshold secret sharing and secure multiparty computation. This scheme is maintained to be dynamic in that a broadcaster can broadcast a message to any of the dynamic groups of users in the system and it is also fair in the sense that no cheater is able to gain an unfair advantage over other users. Another important feature of our scheme is collusion resistance. Using secure multiparty computation, a traitor needs k cooperators in order to create a decryption machine. The broadcaster can choose the value of k as he decides to make a trade-off between communication complexity and collusion resistance. Comparison with other Broadcast Encryption schemes indicates enhanced performance and complexity on the part of the proposed scheme (in terms of message encryption and decryption, key storage requirements, and ciphertext size) relative to similar schemes. In addition, the scheme is modeled using applied pi calculus and its security is verified by means of an automated verification tool, i.e., ProVerif.

1 Introduction

A broadcast encryption system [1, 2] enables a broadcaster to encrypt a message for an arbitrary subset of users $S \subseteq 1, ..N$ who are listening on a broadcast channel. Any user in S can decrypt the broadcast using his private information. Moreover, even if all users outside of S collude, they obtain no information about the contents of the broadcast. Such systems are said to be collusion resistant.

Since inception, Broadcast Encryption has been a hot research topic in academic world and has found its way in many sub-areas as well [1–3]. There are many applications for Broadcast Encryption including media broadcasting such as IPTV, pay TV, CD and DVD [4], Wireless Sensor Networks [5, 6], Content Protection [7, 8] to name a few.

There are two main types of Broadcast Encryption (BE) systems, namely the Public key Broadcast Encryption [9–13], which makes use of public key operations to address security requirements, and Private key Broadcast Encryption [2, 9, 14], which is also used for the purpose of the present study. It has to benoted here that the main advantage of a Private Key BE scheme lies in its use of lightweight operations to perform encryption, which makes the scheme suitable for special circumstances such as Sensor Networks and Mobile Networks.

☆ This article is an extended/revised version of an ISCISC'12 paper.
* Corresponding author.

Email addresses: mahdix@gmail.com (M. Soudkhah Mohammadi), ghaemib@ferdowsi.um.ac.ir (A. Ghaemi Bafghi)

ISSN: 2008-2045 © 2013 ISC. All rights reserved.

2 Related Literature

2.1 Broadcast Encryption

Fiat and Naor in [2] has proposed a number of private-key collusion resistant broadcast encryption schemes. In these schemes, each user will store keys, each of which is dedicated to a specific subset of users that does not include the key owner. In order for a message to be broadcast, it will be encrypted using the key dedicated to the recipient subset.

Similarly, Du *et al.* [14] have introduced an ID-based broadcast encryption scheme by which a center can distribute keys over a network, so that each member of a privileged subset of users can compute a specified key. Then a conventional private-key cryptosystem, such as AES, can be used to encrypt the subsequent broadcast with the distributed key. The main disadvantage of this process is that the center needs to setup and broadcast session keys for each new subset. In addition, when a new member is added to a recipient group, the center needs to re-establish group session key and broadcast the new key. Upon user revocation, this scenario has to be repeated and new group session keys are to be re-established and broadcast. Although this scheme has proved to be dynamic, it does not appear as efficient, since when a new user is added or removed, system parameters need to be re-established.

Elsewhere in [10], Boneh and Waters also introduced a novel concept called Augmented Broadcast Encryption which was shown to be sufficient for constructing broadcast encryption, traitor-tracing, and trace-and-revoke systems. The main problem with this scheme, however, is that, in order to add a new user, the whole system needs to be re-established. Thus the center needs to re-calculate public and private parameters of the system and user keys.

In the same vein, Delerable *et al.* [9] attempted to define two new efficient constructions for public-key broadcast encryption and a construction for private-key broadcast encryption. Takeing advantage of bilinear maps, the first construction defines three sub protocols: Join, Encryptand Decrypt. This construction supports one-time revocation, which means that revoked users are not removed permanently from the system. To provide this feature, the center needs to broadcast messages to system users to update their keys.

The second construction, however, enhances the first one by providing $O(1)$ cipher length and linear decryption keys. Yet it lacks the dynamic features of the first construction.

And the third construction enjoys $O(1)$ decryption key and a linear cipher text.

The problem with this latter one is that it is mainly designed for cases with little number of revoked users.

In another study, Danfei has proposed an authenticated multi-broadcaster Broadcast Encryption scheme which provides authentication and non-repudiation of messages [15]. This scheme is of a constant size for private key and cipher text, yet the public key size is linear depending on the number of system users.

Naor *et al.* [16] have also proposed two fully collusion secure Broadcast Encryption schemes. Recipients are assumed stateless, meaning that they do not need to update their internal information from session to session. Authors have presented two Subset-Cover revocation algorithms, which provide the user revocation feature. This method offers two improvements over similar works: first, message length is reduced to $O(r)$ while maintaining a single decryption at the recipient side. Second, it provides integration between revocation and tracing so that the tracing mechanism does not require any changes to the revocation algorithm.

It has to be noted here that in most Broadcast Encryption schemes, the list of recipients of a broadcasted message is contained in the cipher text. Nevertheless, in most practical cases, this information is considered sensitive and confidential and should not be revealed to others. In this respect, a number of studies have focused their attention on providing anonymity for Broadcast Encryption[12, 13]. Fazio and Perera, for instance, in [12] have formally defined the notion of outsider-anonymous broadcast encryption and proposed generic constructions in the standard model that can achieve outsider-anonymity under adaptive corruptions in the chosen-plaintext and chosen-ciphertext settings. In [13] too, Liber *et al.* have proposed two generic constructions for Anonymous Broadcast Encryption. The first construction can be used to transform any public-key encryption scheme with some security requirements into an Anonymous Broadcast Encryption, while the second construction uses any Identity-Based encryption system with some weak security requirements.

In line with the abovementioned studies, Wang and Liao [17] have presented an efficient encryption protocol to be used in Broadcast Encryption systems. This protocol is designed for mobile ad hoc networks and constructs an encryption mechanism with low storage requirements, useing a one-way key chain and a polynomial function.

In their study, Jeong *et al.* have defined a new security requirement for Broadcast Encryption systems [11]. This requirement is Consistency, by which each recipient can be assured that all other recipients of the message have received exactly the same data. This prevents the broadcaster to send a message to a set of recipients, which can result in different extracted data for different users. They have used Binding Encryption, Public key and private key encryption to address this requirement.

2.2 Secure Multiparty Computation

In a Secure Multiparty Computation protocol, generally two or more parties intend to conduct a computation on their private inputs. However, neither of the parties is willing to disclose his own input to anybody else. Therefore, conducting such a computation while preserving the privacy of the inputs is of paremount importance in these protocols [18]. For example, if the computation is larger than, which takes two inputs and indicates whether the first input is larger or not; the two participants each has a private number and wants to know which number is larger. This problem is called the millionaire problem and is introduced as the first SMC protocol by Andrew C. Yao in [19].

SMC protocols can also be used in cloud computing environments. Given the growing role of cloud computing infrastructure, organizations can improve their efficiency while minimizing the expenditure and the operation overhead. One of the major reasons behind switching to a cloud based service is the users' data privacy, which is supplied to the cloud provider. This problem has been discussed by Maheshwari and Kiyawat in [20], who attempted to provide SMC solution techniques that can be embedded while designing the architecture of cloud computing center.

In this regard, two types of adversaries can be defined for a SMC protocol, namely the passive type (which has access to the victim's data and cannot deviate from protocol steps) and the active one (which can additionally deviate from protocol steps e.g. by sending incorrect data) [21, 22].
Security of a SMC protocol can be discussed in the two paradigms of an ideal world and a real world. In ideal world, we assume the existence of a trusted third party (TTP) who securely takes inputs from users, performs the required calculations and distributes the result to participants. This model can also incorporate different types of adversaries. The real world paradigm, however, does not have a trusted third party. A SMC protocol is secure in real world paradigm, if whatever an adversary can do in real world can be simulated in the ideal world by creating an appropriate adversary model. This security analysis model is introduced by [23–25] and is largely used in formal analysis and proof of security of SMC protocols.

There is another specific type of SMC protocols called server-assisted secure computation or privacy preserving security protocols. In these protocols, we assume that a lower computational power entity (client) is interested in solving a computational problem C. Yet there is another entity (called the server) which is of a much higher computational power but is not trusted by the client. In fact, the client wants to use the server to solve his problem without revealing any information about the problem. In these scenarios, the client commonly generates a different (but related) computational problem C and sends it to the server. The server solves C and sends the result $\acute{s}$ back to the client. After receiving the response $\acute{s}$, the client can apply a transformation on the response to obtain the value of s, which is a solution to the original problem C. An example of this is the solution proposed by Atallah and Frikken [26] to deal with the problem of matrix multiplication, having preserved the privacy of the client. SMC protocols make use of several cryptographic techniques (secret sharing [27], threshold homomorphic encryption [28], oblivious transfer [29]) in order to securely calculate different types of algorithms.

Keng-Pi Lin and Ming-Syan Chen [30] proposed a SMC protocol for the classification of privacy preserving using support vector machines. It is evident that the data extracted from training samples in a classification problem can impose legal and commercial threats to the privacy of customers. In this respect, the present paper is aimed at introducing an approach to post-process the SVM classifier in order to transform it to a privacy preserving classifier, which does not disclose the private content of the support vectors. The problem of privacy preserving classification is studied by Luong *et al.* [31] as well, where a cryptographic solution was presented for privacy preserving classification rules learned in two-dimension distributed data. In this paper, an algorithm is proposed for privacy preserving computation of frequencies of a tuple of values, which can ensure the privacy of users without loss of accuracy.

In their study, Yu and Zhang have tried to introduce a generic grid privacy preserving computation (G2PC) model which supports privacy preserving analysis and computation on data without compromising the privacy of raw node data or data statistics generated in intermediate computations [32]. Privacy preserving variance computation and k-means clustering algorithms are used to provide evidence for the efficacy and efficiency of the proposed scheme.

Elsewhere, Piyi Yang *et al.* in [33] have put forth an efficient multidimensional privacy preserving data aggregation scheme for Wireless Sensor Networks (WSNs) which is of high security. This scheme provides efficient countermeasures against passive and active privacy compromising attacks, coalition attacks and is robust to data loss. The security enhancements of the scheme imposes constant communication overhead to the WSN, which makes it suitable for large-scale networks.

Another SMC solution was also proposed by Mirsha and Chandwani [34] in which a user unanimously selects a trusted third party (TTP), called master TTP, from among a large number of TTPs. This master TTP can change over time, ensuring that no single TTP controls the entire system all the time. At the same time, this also indicates that no TTP knows where the computation is taking place.

2.3 Applied Pi Calculus

Using process calculi to model security protocols was first studied by Lowe [35], wherein a flaw in Needham-Schroeder key distribution protocol is found by modeling the protocol using CSP calculus. Later on, some ad hoc calculi have been proposed for modeling cryptographic protocols among which,the Spi Calculus [36] and the applied pi calculus are the most widely studied.

The applied pi calculus has been used to model security protocols in a variety of areas. Abadi and Blanchet [37], for instance, have used ProVerif [38] to model a security protocol for certified email, message secrecy and certificate receipt of the protocol. Kremer and Ryan [39] have also used applied pi calculus to model Fujioka, Okamoto, Ohta (FOO) electronic voting scheme [40] and analyze its fairness, eligibility and privacy. Similarly, Delaune, Kremer and Ryan [41] have modeled anonymity properties of electronic voting schemes, namely vote-privacy and receipt-freeness using Applied Pi Calculus and analyzed these properties for two electronic voting schemes available in the literature. In this regard, Kusters and Truderung [42] have also come up with a new definition for coercion resistance of an electronic voting scheme and analyzed three voting protocols using applied pi calculus. Kramer, Ryan and Smyth [43] have presented a formal definition of election verifiability based on boolean tests which distinguishes three aspects of verifiability, including individual, universal and eligibility verifiability. Applicability of this model is verified by analyzing three electronic voting protocols,namely the FOO [40], Helios [44] and Civitas [45]. This study is further extended by Smyth *et al.* in [46].

In a similar vein, Chen and Ryan [47] have analyzed the security of Trusted Platform Module (TPM) in case of sharing authorization values (authdata) among multiple users and showed impersonation attacks against TPM. They have also proposed a new authorization protocol called Session Key Authorization Protocol (SKAP) as a solution to this problem which allows authdata to be shared without the possibility of impersonation attack. SKAP generalizes the existing authorization protocols OIAP and OSAP. Both the new and the existing protocols are analyzed using ProVerif. As a result of these analyses, secrecy and authentication of the proposed protocol is proved.

Backes, Maffei and Unruh [48] have proposed a formal abstraction of zero-knowledge protocols using Applied Pi Calculus. A simple variant of Direct Anonymous Attestation (DAA) protocol is analyzed using ProVerif. They found a novel attack which was overlooked in its existing cryptographic security proof. A revised variant of DAA is thus proposed which has been proven to be secure.

Blanchet and Chaudhuri [49] have used ProVerif to study security properties of a state-of-the-art protocol for secure file sharing on untrusted storage. In this study, several ambiguities and some unknown attacks on the protocol are revealed. Finally, a correction to the protocol is proposed which is guaranteed to be secure.

Abadi, Blanchet, and Fournet [50] have studied Just Fast Keying protocol (JFK) which is a fast key establishment protocol to secure IP communication. The protocol is formally analyzed in Applied Pi Calculus and some ambiguities and minor problems are revealed. Hence, a number of ideas and techniques are developed which should be useful in specification and verification of security protocols.

A number of security conscious web applications use clientS to encrypt information that will be stored on a web-server. Using this mechanism, no plaintext data is kept on the server which will increase the security of sensitive information. Bansal *et al.* [51] have investigated a number of such web applications (including password managers, cloud storage providers, an e-voting website and a conference management system). In their study, an automated formal analysis is performed using ProVerif which has resulted in finding novel attacks.

With the rise of Internet, electronic auctions have recieved more popularity and are being used increasingly worldwide. Today there are numerous security protocols that address security requirements of these electronic transactions. Dreier, Lafourcade and Lakhnech [52] have proposed a formal framework based on Applied Pi Calculus to analyze and verify various numbers of security requirements of e-Auction protocols such as secrecy of bids, anonymity of the participants, receipt freeness, coercion resistance, fairness, non-repudiation and non-cancellation. Two case studies have also been conducted to show how these properties can be verified automatically using ProVerif to discover several attacks.

One critical requirement in special security protocols such as electronic voting is the privacy of users' data. With the rapid development of more powerful hardware and security attacks, one may wonder if such information can remain secret after 20 years or even forever. In their attempt to answer this question, Arapinis *et al.* [53] have proposed a novel idea of practical everlasting privacy. The key idea is that in the future, an attacker may be more powerful in terms of computation (he may be able to break the cryptography) but less powerful in terms of the data he can operate on (some transaction of the protocols may not have been stored). This notation is formalized using applied pi calculus, and with the help of ProVerif, it is shown that several variants of Helios (including Helios with Pederson commitments [54]) and a protocol by Moran and Naor [55] can indeed achieve practical everlasting privacy.

3 Applied Pi Calculus

Applied pi calculus [56] is a formal language used to model and encode security protocols and verify various security requirements such as secrecy and authentication. This language is a derivation of pi calculus [57] first introduced by Robin Milner. This formal language provides intuitive elements to describe participants of a security protocol, their internal actions (e.g., calculations, checking, and message creation), and communications. Each participant is defined as a process which can communicate with other processes through channels. A channel is a two-way communication method between multiple processes. This language is coupled with a formal semantics which enables one to reason about security protocols. A wide variety of cryptographic primitives and their relationships can be modeled using equation theory. This language allows us to define different types of security goals and to ensure whether the protocol meets those goals or not.

A signature Σ consists of a set of function symbols such as *hash*, *encrypt*, and *sign*. Arity of a function denotes the number of inputs it takes. A constant is a function with arity of 0 [56].

For a given signature Σ, an infinite set of names and an infinite set of variables, the set of *terms* is defined as below:

L, M, N, T, U, V ::=	*terms*
a, b, c,...,k ,..., m, n, s	*names*
x, y, z	*variables*
$f(M_1, ..., M_l)$	*function application*

In the definition above, l is the arity of function f which can be any member of Σ. In order to separate different types of information such as decryption key, user identifier or bit-strings, we use a sort system containing a set of data types which can be used in a security protocol. The grammar for *processes* is as below:

P, Q, R ::=	*processes (or plain processes)*
0	*null process*
$P \mid Q$	*parallel composition*
!P	*replication*
vn.P	*name restriction*
if M=N *then* P *else* Q	*conditional*
u(x).P	*message input*
$\bar{u}\langle N\rangle$.P	*message output*

The null process 0 does nothing; $P \mid Q$ means execution of P and Q in parallel. !P denotes infinite instances of P process executing in parallel (e.g. $P \mid P \mid P \mid ...$). The process *vn*.P makes a new private name n and behaves as P. The two last processes use channel u to input a variable value or output a term and then behave as P.
These processes can be extended with *active substitutions*. Extended processes are defined as below:

A, B, C::=	*extended processes*
P	*plain processes*
$A \vert B$	*parallel execution*
vn.A	*name restriction*
vx.A	*variable restriction*
$\{\frac{M}{x}\}$	*active substitution*

$\{\frac{M}{x}\}$ is a process which replaces variable x with the term M. This definition floats and applies to any process that comes into contact with it. We can extend this definition to multiple substitutions and write:

$$\{\tfrac{M_1}{x_1}, \tfrac{M_2}{x_2}, ..., \tfrac{M_l}{x_l}\} \text{ for } \{\tfrac{M_1}{x_1}\} \mid \{\tfrac{M_2}{x_2}\} \mid ... \mid \{\tfrac{M_l}{x_l}\}$$

Like programming languages, names and variables have scopes, which are delimited by restrictions and by input. Therefore, the set of free and bound variables and free and bound names of A can be expressed by *fv(A)*, *bv(A)*, *fn(A)* and *bn(A)*, respectively.

Based on a signature Σ, we define an equational theory, which defines an equivalence relation on terms which is closed under substitution of terms for variables. For *M*=*N* in the theory associated with Σ, we write $\Sigma \vdash M = N$. When Σ is clear from context, we may only write *M*=*N* [56].

We use Dolve-Yao's [58] model to investigate protocol security. In this model, the attacker can read, intercept and create any message and is only limited by constraints of cryptographic primitives. In pi calculus, it is assumed that the attacker is the environment itself and has complete control over the public communication channel. This implies that the attacker can read any message that is transmitted between two processes using a public channel and sends any message to any process waiting to read a message from a public channel.

In order to define and investigate authentication properties, we take advantage of correspondence. Correspondence properties are used to capture the relationship between events that are raised during the execution of the security protocol. Moreover, these events can contain parameters which can be used to define the relationship between parameters of different events. In order to evaluate protocols in terms of correspondence properties, we annotate them with events which mark important stages of the protocol (e.g. *AcceptClient*, *DecryptMessage*,). These events are messages that output through an event channel.

4 The Proposed Scheme

We extend our previous protocol (DZMBE) [59] to propose a secure, fair, and dynamic broadcast encryption scheme based on secure multiparty computation (Which we call DZMBE+).

This new scheme consists of a broadcaster, which sends messages of different types to subscribers (users). When subscribing for a messages group, the user receives a group-based private key, which can be used in cooperation with other subscribers to execute a SMC protocol to rebuild the message decryption key while at the same time preserving confidentiality of the private data of the user. The group-based private key is created using a threshold secret sharing scheme. Having the message decryption key at hand, each user can decrypt the message received and access to the original data.

Given this new scheme, the recipients are assumed to be stateless. This implies that the recipients are not required to update any kind of information from session to session. Our proposed scheme consists of four phases, as described below. Two phases (message broadcast and message retrieval) are operational and the other two (group setup and user subscription) are aimed at management of user groups.

The broadcaster sends different types of messages. Each of these types is called a group. In order to define a new group of messages, group setup phase has to be executed. Adding new members to this scheme is done with no effects on other parts of the scheme. When a new member is added, user subscription phase is performed and he will receive his own identifier and private share, which can be used to decrypt broadcast messages of the related group later.

To revoke a user from a group, the broadcaster needs to re-create shares of other users and inform their new shares. Since the revoked user has a valid share, which can be used in message retrieval phase, the broadcaster needs *O(n)* communications to inform other users of their new shares. During a membership revocation, the broadcaster needs to re-execute the group setup phase and for each non-revoked user, User subscription will be re-executed to update members' subscription information.

4.1 Group Setup Phase

In order to setup a new group g, the broadcaster creates a random group identifier, ID_g, and a random private group key, k_g. It has to be noted that the value of this private group key is only known to the broadcaster.

In addition, a public random generator,h_g, is selected by the broadcaster for the new group. The broadcaster sets up Shamir's *(t,n)* threshold secret sharing scheme using an appropriate value for parameter t. This parameter depends on the level of security, which is desired for that group. For example, in the case of a group with highly confidential messages, this parameter will be much higher in comarison with that of a group about public sport news.

After selecting an appropriate value for t, the broadcaster chooses t coefficients $a_1, ..., a_{t-1}$, where $a_0 = k_g$ (group key) and $a_1, ..., a_{t-1}$ are randomly selected. Now the broadcaster has a polynomial, which can be used to generate new shares for members of this group:

$$f_g(x) = a_0 + a_1x + a_2x^2 + ... + a_{t-1}x^{t-1}$$

Using this polynomial, f_g, it is possible to generate any number of shares for the subscribers of this group.

Besides, the group identifier ID_g and a description of the group are added to a public bulletin board in order to inform the potential subscribers about the new service of the broadcaster and its price.

4.2 User Subscription Phase

When a user wants to register to receive a specific group of messages (e.g., sports and business news), he uses a secure channel to send a registration request to the broadcaster. This request contains group identifier (ID_g) and a payment token used to prove that the user has paid the appropriate amount of money to a payment server. A detail of the payment method used for this operation and generation of a payment token is beyond the scope of this paper.
After checking the validity of the payment token, the broadcaster assigns a unique user identifier, ID_u to

Figure 1. General structure of a broadcast message

ID_m	ID_g	$E_{MK_{g,m}}(SN, Data)$

the user. One method to generate this identifier is by applying a secure one-way hash function on the identity information of the user. The user's private group key is a share of group's polynomial and is calculated as below:

$$UK_{g,u} = f_g(ID_u)$$

$(ID_u, UK_{g,u})$ are sent to the new user using a secure channel after they are stored in users database.

The subscriber will store these values in his local storage and will use these data in Message retrieval phase to calculate the message decryption key. The value of ID_u is not private and can be sent to other users, but $kUK_{g,u}$ is a private user key which should be kept confidential.

4.3 Message Broadcast Phase

Suppose that the broadcaster is to broadcast a message m, which belongs to group g with ID_g as the group identifier. He first selects a random message identifier, ID_m, and calculates the message encryption key, $MK_{g,m}$ according to the formula below:

$$MK_{g,m} = z^{(h_g^{k_g})^{ID_m}}$$

Where z is a public generator. Thus the original data is encrypted under the message encryption key $MK_{g,m}$, using a secure symmetric encryption algorithm (e.g. AES). A sequence number (SN) is also used to prevent adversary from replay attacks. The final message will consist of a message identifier, a group identifier, the sequence number and the cipher text.

It is noteworthy that this final message will be broadcast to all users.

4.4 Message Retrieval Phase

After a message is broadcasted, recipients will check the message header and extract the group identifier. In fact, they check whether they have subscribed for the message group, ID_g. If the answer is positive, the member can cooperate with any other t members to calculate the value of the message decryption key. The message retrieval phase is performed in two steps, as described below:

(1) First, each member uses his private data and other members' public information to calculate a partial decryption key. While this key cannot be used for decryption operations, it will be employed in the next step to calculate a message decryption key.

(2) Then, the members execute SMC-MULT protocol to combine their partial decryption key and calculate the message decryption key.

To execute the SMC-MULT protocol between t users, we suppose that the participant users have identifiers of *1,,t.*

$$1 \leq u \leq t$$

This assumption is made to simplify notations and does not impose any limitations of the correctness on the algorithm. The target function going to be calculated by the participants is:

$$\acute{MK}_{g,m}(.) = z^{h_g^{ID_m \Sigma_{u=1}^{t} UK_{g,u} l_{g,u}(0)}}$$
$$l_{g,u}(x) = \prod_{m=1, m\neq u}^{t} \frac{x - ID_m}{ID_u - ID_m}$$

Note that in the above formula, $UK_{g,u}$ denotes each user's private group key and $l_{g,u}(x)$ is a polynomial, which is calculated according to the Lagrange's polynomial formula. It has to be remembered that each user can calculate the value of $l_{g,u}(0)$ by just having the user identifiers of other participants, i.e. the public information.

Every member can calculate his private value of $UK_{g,u} l_{g,u}(0)$ which will be denoted by Δ_u.

$$\Delta_u = UK_{g,u} l_{g,u}(0)$$

Therefore, the target function can be rewritten as:

$$\acute{MK}_{g,m}(.) = z^{h_g^{ID_m \Sigma_{u=1}^{t} \Delta_u}}$$
$$= z^{(h_g^{ID_m \Delta_1} . h_g^{ID_m \Delta_2} ... h_g^{ID_m \Delta_t})}$$
$$= z^{p_1 p_2 \cdots p_t}$$

$p_u = h_g^{ID_m \Delta_u}$ is known as a partial decryption key, which can be calculated by each user by just having the identifiers of other participants at hand.

The final target function will thus be:

$$\acute{MK}_{g,m}(.) = z^{\prod_{u=1}^{t} p_u}$$

In order to calculate the value of $\acute{MK}_{g,m}$, the users participate in execution of SMC-MULT$(z, p_1, p_2, ..., p_t)$. After running this sub-protocol, each user will have access to the value of $\acute{MK}_{g,m}$. This sub-protocol is being described in details in the following section. Having calculated the value of $\acute{MK}_{g,m}$, the message decryption key can be used to decrypt the message data (In other words, in case the protocol proceeds properly, it is expected that $\acute{MK}_{g,m} = MK_{g,m}$.

The SMC protocol proposed in this paper enjoys the following advantages:

- The SMC protocol is correct. It calculates the result correctly if inputs are correct (This can be proved according to the abovementioned formulas).

- No confidential information for any participant user may be exposed to an attacker or other participants. The information sent by each member to others is his partial decryption key (p_u) which can be expressed using the formula below:
  $$p_u = h_g^{ID_m \Delta_u} = h_g^{ID_m UK_{g,u} l_{g,u}(0)}$$
  In the above formula, the value of $UK_{g,u}$ is regarded as sensitive information and should remain private. The only publicly available data is p_u, however, to calculate $UK_{g,u}$, one needs to solve the discrete logarithm which has no polynomial-time algorithm [60].

The message retrieval phase can be performed in one of two following modes: *Interactive* vs. *Offline*. In the interactive mode, t parties cooperate and execute *SMC-MULT* protocol and exchange data with each other to calculate the message decryption key (as explained above). Yet this mode is of one major disadvantage in that it may be difficult for a member to find other *t-1* parties who are ready for participation in the protocol with each other at the same time.

In order to deal with this issue, members can use the *offline* mode, in which a member sends requests to all other members asking them to provide him with their calculated partial decryption keys ($h_g^{ID_m UK_{g,u}}$) and user identifiers. Having gathered sufficient partial decryption keys of other members, the user will be able to combine the keys and calculate the message decryption key without having to find *t-1* participants willing to execute the appropriate protocol at the same time.

Finally the last step is to check the monotonicity of the received sequence number (SN). In case that the sequence number of the message is not greater than the most recent received SN, the message is discarded.

5 SMC-MULT Protocol

This protocol has t participants $(U_1, U_2, ..., U_t)$, each of which having $p_i, 1 \leq i \leq t$, respectively. Participants are to calculate the value of $y = z^{\prod_{u=1}^{t} p_u}$, where z is a public group generator. Each p_i is a secret piece of information and cannot be revealed to any other party, even a trusted third party. To execute this protocol, users make a ring, meaning that the user after U_i will be U_{i+1} if $i < t$ or U_1 if $i = t$. The user after U_i in the ring will be denoted by U_{i+1} for simplicity.

This protocol is executed in t rounds. A definition of round i is provded below:

Figure 2. Message structure for a fair broadcast protocol

ID_m	ID_g	$E_{T_{g,m,b-1}}(SN, Data)^{T_{g,m,b}}$
$CV_1 = H(T_{g,m,1}), CV_2 = H(T_{g,m,2})$		
...		
$CV_{b-1} = H(T_{g,m,b-1}), CV_{b+1} = H(T_{g,m,b+1})$		
...		
$CV_w = H(T_{g,m,w}), CV_b = H(T_{g,m,b})$		

(1) U_i is the round starter, which selects a random number b and calculates b^{-1}.
(2) U_i sends z^{bp_i} to the next user in the ring. The recipient user is known as U_{i+1}.
(3) The recipient user raises the received data to the power of his private information (p_{i+1}) and sends the result $(z^{bp_i})^{p_{i+1}}$ to the next user in the ring.
(4) The previous step is repeated until the data is returned to the starter user (U_i).
(5) U_i raises the input to the power of b^{-1}.
(6) The result is $z^{p_i p_{i+1} p_{i+2} \cdots p_t p_1 \cdots p_{i-1}} = z^{p_1 p_2 \cdots p_t}$.

This round is executed t times. It has to be noted that in each execution i, user U_i will be the starter.

Once the protocol is finished, the value of the target function will be available to each participant, while no confidential information is sent to any external entity.

6 Improved Scheme

It is noteworthy here that there is one major problem with the proposed scheme in that it fails to stop cheaters. A cheater is a user that sends incorrect data to others while receives the others' correct data and thus is able to calculate the value of the target function. Therefore, it is necessary to ensure that either all or none of the participants will be able to calculate the message decryption key.

In order to eliminate cheaters and add fairness to our SMC-MULT protocol, an improved version of the scheme is proposed based on the protocol explained in [61]. As for this new scheme, the group setup phase and user subscription phase are similar to the original scheme, and hence their explanation is not included here. The message broadcast and message retrieval phases are being described below though.

6.1 Message Broadcast Phase

Given the improved scheme, the message is created the same as the original scheme with a check vector is added to the end of the message, which contains w items.

In this new structure $T_{g,m,i} = ((z^{h_g{}^{k_g}})^{ID_m})^l$, where k_g is the private key of the message group. H denotes a secure hash function. The broadcaster selects a fairness length parameter w, which indicates the number of $H(T_{g,m,i})$ values that will be added to the message. In order to prepare a message for broadcasting, the broadcaster calculates $T_{g,m,i}$ values and selects a random $b \subset \{1, ..., w-1\}$. The original data is encrypted under $T_{g,m,b-1}$ and the result is raised to power $T_{g,m,b}$.

6.2 Message Retrieval Phase

In order to decrypt this data, recipients need to find out the values of both $T_{g,m,b}$ and $T_{g,m,b-1}$; neverthelss, they will not know value of b beforehand. This phase is executed in a similar way to the original scheme, with SMC-MULT being executed multiple times (at most w times). The steps of this phase are the same as the basic scheme with the only difference that, it is executed more than once and during the i^{th} execution of the protocol, the users' inputs are raised to power i.

If $T_{g,m,i}$ will be the result of the i^{th} execution of SMC-MULT protocol, each participant will have this value at the end of the protocol execution. Now, the users perform a check operation. Through this step, they will realize if there is a cheater among them and if not, they can assure whether they have found out the value of the message decryption key or not.

Then each user hashes their value of $T_{g,m,i}$ and compares it with CV_i and CV_b. At this stage, three possible scenarios can occur:

(1) $H(\acute{T}_{g,m,i}) = CV_i$
(2) $H(\acute{T}_{g,m,i}) \neq CV_i, H(\acute{T}_{g,m,i}) \neq CV_b$
(3) $H(\acute{T}_{g,m,i}) \neq CV_i, H(\acute{T}_{g,m,i}) = CV_b$

In scenario 1, there is no cheater and users have not yet calculated the message decryption key.

In scenario 2, it is indicated that there is a cheater among the participants. Thus, the users stop executing the protocol at this step.

In scenario 3, $T_{g,m,b}$ is the users' calculated candidate message decryption key and the actual message decryption key is $T_{g,m,b-1}$ which was previously calculated.

Having the values of $T_{g,m,b}$ and $T_{g,m,b-1}$ at hand, the users can raise the encrypted message to $T^{-1}_{g,m,b}$ and decrypt the result using $T_{g,m,b-1}$ as shown below:

$$(E_{T_{g,m,b-1}}(Data)^{T_{g,m,b}})^{T_{g,m,b}{}^{-1}} = E_{T_{g,m,b-1}}(Data)$$
$$Data_{T_{g,m,b-1}}(E_{T_{g,m,b-1}}(Data)) = Data$$

After each round of execution of SMC-MULT, the parties hash the result $T_{g,m,i}$ and compare it with the value provided in the message. In case any conflict is observed, they stop the protocol execution.

7 Analysis

7.1 Privacy

With ever increasing online communications, privacy is considered as an important factor in today's networks. In fact, it is desired to decrease the number of entities that have access to users' information (including their identity, billing address, and postal address) to the least possible. In addition, user's preferences (what messages or contents each user likes) need to be confidential. The method proposed in this paper provides this type of privacy. In our scheme, each user's identity information is hidden from all parties. When a new user is subscribed in a group to receive certain types of messages, he sends a subscription request to the broadcaster. This request contains no identity information (only a randomly chosen identifier and a payment receipt). As a result, the broadcaster will not be able to deduce his customer's identity.

Another feature of this scheme is that no one will be able to link preferences of a user to his identity. As users subscribe for a message channel anonymously, even the broadcaster is not be able to determine which groups of messages a user is interested in.

Additionally, when broadcasting a message, only the group identifier is broadcast, and thus no external entity with any amount of wiretapping transmitted information knows the identity of members of a group, which in turn will increase the privacy of users.

7.2 Collusion Resistance

In a broadcast encryption system, a center sends a message to a set S of users U, where $S \subseteq U$. Any member of S can decrypt and read Center messages. However, even if all members of U collude, they can obtain no information about the contents of the message. Such systems are said to be collusion resistant [62].

Our proposed scheme is also collusion resistant as no user outside the eligible recipient group can extract any knowledge from the broadcast message. That is because the message is encrypted with a key that can only be extracted if a user has a valid share, which belongs to the message group. As an example, suppose that e, who is not a member of S, wants to decrypt a received message. In order to obtain the message decryption key, e needs to cooperate with *(t-1)* other eligible recipients and execute the *message retrieval* protocol. During the execution of the first round of this protocol, *t-1* members will find out that there is a cheater among them and hence, will stop proceeding the execution. As a result, it can be viewed that no one outside S will be able to calculate the message decryption keys. It has to be added that increasing the size of collusion of illegible users will not change the situation.

7.3 Flexibility

It is known that the cooperation among the message recipients is costly and requires a certain number of communications. On the other hand, this feature assures the security of the scheme and prevents unauthorized users to access messages. As we are using *(t,n)* threshold secret sharing protocol, the broadcaster can choose a value of t adaptive to the security level and communication requirements of its own network.

Besides, using a threshold secret sharing scheme makes our scheme more flexible, because when some recipients are not available or when communication problems occur, users are free to choose alternative recipients to communicate and build the message decryption key. Actually, if a message has n recipients and the scheme is based on *(t,n)*-threshold secret sharing, each of n recipients is required to select only t-1 other recipients to decrypt the message. And in fact, the user can use any factor (such as the communication cost or trust to other users) to select these t-1 users. This parameter makes the scheme even more flexible for system users.

In Wireless Sensor Networks (WSNs), one major performance factor is communication cost. In these systems, a communication with a distant sensor consume a lot of power which is a valuable resource. Our method enables such systems to adapt and select t near neighbors to perform OTK reconstruction. This reduces power consumption while the system will remain secure at the same time.

7.4 Fairness

Our improved scheme also provides fairness in that it is able to eliminate cheaters. This scheme ensures all participants that either all of them will get the final key or no one will be able to calculate anything.

Suppose that one of the participants is a cheater who wants to send out incorrect values to others and use their correct values to build the secret and decrypt the received message. In case of successful cheating, other participants will not be able to calculate the decryption key while the cheater will have this data. This will provide an unfair advantage to cheaters, which makes the scheme impractical in sensitive implementation scenarios. The cheater needs cooperation of t-1 other users in order to calculate any of T_i values. Additionally, he needs to guess the correct value of. As a result, the probability of successful cheating will be $\frac{1}{w}$ where w is the number of T_i values in the check vector. Hence, in order to decrease chances of cheating, the broadcaster needs to select an appropriate value for w. Higher values for w will decrease the probability of cheating while communication costs increase. Lower values will facilitate the cheater's job while imposing less burden on honest users.

7.5 Forward and Backward Secrecy

A group communication system provides *perfect forward secrecy* if a member leaving the group at time t does not gain any information about the content of the messages communicated at times $\acute{t} > t$ [63]. In order to provide perfect forward secrecy, the center needs to update the privacy of group members, for each group in which the revoked user was subscribed. To this end, the following steps are required to be followed for each of those groups:

(1) First, the center selects a random r.
(2) For each group member, Center broadcasts $M = \{ID_u, ID_g, E_{UK_{g,u}}(r)\}$ as a special group management message. Although this message is broadcast, the recipient of the message is only one user. If a user is a member of more than one group, all required group management messages for that user can be combined to form a single message. This will reduce the communication cost.
(3) Upon receiving message M, the user decrypts the message using his private user key ($UK_{g,u}$) to find the value of r.
(4) The recipient user updates his private user key by adding it to r, and therefore, the updated user key will be $(UK_{g,u} + r)$.
(5) Once the broadcasting operation is finished, the center updates the group key k_g to $(k_g + r)$.

A group communication system provides *perfect backward secrecy* if a member joining the group at time t does not gain any information about the content of the messages communicated at times $\acute{t} < t$ [63]. In order to provide backward secrecy, the center needs to inform members of the group g in which a new user is going to be subscribed, to update their private user keys ($UK_{g,u}$). To this end, the center selects a random r and updates the group key from k_g to $(k_g + r)$. Hence, the group polynomial will be:

$$f_g(x) = (k_g + r) + a_1x + a_2x^2 + ... + a_{t-1}x^{t-1}$$

In order to update private user keys for all member of the group, the only required action is to add r to the user keys. The value of r is randomly chosen and does not reveal any information regarding f_g or k_g. Thus, the center can broadcast a special group management message to a member of group g and declares the value of r. Each recipient only needs to add his private user key with r to have a correct key. Having updated the process, the center can use the new group polynomial to calculate the subscription information for the new user.

7.6 Comparison

In our broadcast encryption scheme, adding users is done in *O(1)*, however, revoking an existing member requires *O(n)* complexity. Moreover, the cipher text length is *O(1)*, meaning that the length of the cipher text is independent of the number of recipient users. In most broadcast encryption schemes, one constant part of the messages is a list of recipients, which is linear to the size of recipient users, yet this is not usually counted towards the cipher text length. In our scheme, the notion *user groups'* is used. A user group is a group of recipients all of whom are interested in receiving specific message types (e.g., different channels in IPTV systems). When a sender wants to broadcast a message, a group id is attached to message data according to the message type. Therefore, the users can decide whether they are recipients of the message or not.

Table 1 summarizes some other well-known broadcast encryption schemes in terms of the following parameters:

(1) Cipher text Length: length of the cipher text of a message that is broadcast to a set of n recipients.
(2) Key length: Length of the key material that each user of the system needs to store.
(3) Cost of new members: How much new data is to be communicated when a new member is introduced to the system.
(4) Cost of revoking: How much data is to be communicated in order to update the system when a user is revoked.
(5) Consistency [11]: which is a feature of a secret broadcast system and means that each receiver can assure that all of the receivers have received the same message. In fact, if sender cannot send a message to a group of recipients in a way that each recipient gets a different message, then the system is consistent.
(6) Encryption/Decryption complexity: which refers to the computation complexity the sender (or broadcaster) and each of recipients have to perform in order to act according the system.

The first protocol is a trivial protocol which is a general construct describing the worst case in Broadcast Encryption. In this protocol, each user has its own key and the broadcaster broadcasts n different messages for n recipients. In this scheme, a new member/revoke membership requires no extra communication as each user stores his private key. However, the main problem in this scheme is that the length of the cipher text is linear to the size of recipient set. This problem eliminates the real benefits of broadcast encryption because as the number of message recipients increases, so does the cipher-length and the overhead of the broadcaster grows as well. Thus the scheme is not scalable and is only viable for a little number of users. That is why it is called trivial scheme.

Table 1. Comparison of some private-key broadcast encryption schemes

Scheme	Cipher text Length	Key Length	Cost of new member	Cost of revoking	Consistency	Encryption Complexity	Decryption Complexity
Trivial	O(n)	O(1)	O(1)	O(1)	No	O(n)	O(1)
FN1 [2]	O(1)	O(logn)	O(logn)	O(logn)	No	O(1)	O(1)
FN2 [2]	O(1)	O(1)	N/A	N/A	Yes	O(n)	O(n)
DPP-3 [9]	O(r)	O(1)	O(n)	O(n)	No	O(r)	O(r)
DWGW [14]	O(n)	O(2^n)	O(n)	O(n)	No	O(n)	O(n)
Proposed Method	O(1)	O(1)	O(1)	O(n)	Yes	O(1)	O(t)

In Table 1 below, the last row belongs to the method introduced in this paper. These results provide evidence for the advantages of our scheme (in terms of encryption/decryption complexity) over most of similar schemes, which makes our scheme highly scalable for large scale applications.

It can be observed from the table that generally speaking, the only other scheme that provides consistency is FN2 with a lower performance in encryption complexity. This scheme also does not support member addition and revoking. It is viewed that the proposed scheme outperforms DPP-3 in terms of the cipher text length, cost of new member, consistency and encryption complexity. A comparison with DWGW in terms of providing consistency, cost of new members, key length, cipher text length and encryption complexity also indicates the superiority of the proposed scheme.

8 Formal Modelling and Analysis

In order to model our protocol and verify its security requirements, applied pi calculus was used. Since we are not interested in verifying the fairness of the scheme (due to limitations of our tools), we just aim to model the basic version of our scheme.

In order to automate the verification step, ProVerif [38] is being employed. Four security properties of the scheme are thus verified and proved to be satisfied:

(1) Confidentiality of the broadcasted message
(2) Secrecy of the users' private keys
(3) Anonymity of the users
(4) Authentication of the sender

Due to special syntax of ProVerif, we have modelled our protocol for *t=3*, four recipient users and a single group. This model can easily be extended for a larger number of users and groups or different values of *t*.

In this regard, two public channels are used to model our protocol:

- *Broadcast Channel*: used by the broadcaster to send messages to recipients.
- *Smc Channel*: Used among recipients to perform SMC protocol to calculate the message decryption key.

Below is a list of free and constant names that are used in the protocol:

- *uNamei*: Identity information of the i^{th} user.
- *groupSecretKey*: Secert group key which is private and only the broadcaster knows (k_g).
- *groupIdentifier*: Public identifier of the group.
- *messageIdentifier*: Public identifier of the broadcast message.
- *groupGenerator*: Public generator of the group (denoted by z in the scheme).
- *tag*: A tag is used to enable users to check whether they have decrypted the message with the right key or not. If the decryption result has this special tag at the beginning, that key and/or message is known as not corrupt. This tag is prefixed to the messages by the broadcaster.

We define *encrypt* and *decrypt* functions to model the symmetric-key encryption of the messages. In order to indicate the relation between these two functions, the reduction rule below is used:

reduc forall m: bitstring, k: Key;
decrypt(encrypt(m, k), k)=m.

The above rule implies that for all messages and keys, decryption of encryption of those messages results in the same message, when done under common keys.

In this respect, *hashName* function is used to calculate unique user identifiers based on user identity information.

calculatePDK function is also used to calculate partial decryption keys for each user. This function receives a user key and a message id and the resulted value will be $h_g^{ID_m \Delta_i}$, according to the protocol definition provided in the *Proposed Scheme* section.
groupFunction is used to calculate private user keys. This function uses the group secret key and the appropriate polynomial for Shamir's secret sharing to create a share for a user.

getSequenceNumber and *checkSequenceNumber* methods are used by the broadcaster and receivers respectively to make sure that a received message is not a replay of an old message.
In the same vein, *Power* and *unpower* functions are used for power operations. *Power* raises a number to a given power and *unpower* raises a number to the inverse of the given parameter. This function is used to implement *SMC-MULT* protocol functionality. A set of reduction rules are defined to model threshold secret sharing. These rules express that any combination of the three users' shares can be used to calculate the message decryption key.

In order to model the scheme, the two following processes are defined:

- *centerProcess*: This process models the behavior of the broadcaster. After selecting a random message identifier, a message encryption key is created by applying *calculateMK* function. Then the message identifier, group identifier and the encrypted message are broadcast using the *broadcastChannel*.
- *userProcess*: This process models a message recipient. After receiving data from the *broadcastChannel* and checking it's group, a partial decryption key is calculated using *calculatePDK* method. In the next steps, *SMC_MULT* protocol is executed through sending the partial decryption key powered to a random number *b* and sending *(originator, data)* using *smcChannel*. Originator is the identifier of the user that has sent the message. After receiving his message, the user can *unpower* the payload and the result will be the message decryption key. Any received message on *smcChannel* whose originator is not the user himself will be powered to the partial decryption key and sent out on the channel.

The input script to ProVerif too is provided below:

```
const tag: bitstring.

free broadcastChannel: channel.
free smcChannel: channel.

type UK.
type UID.
type GK.
type MessageId.
type Key.
type sequenceNumber.

const groupGenerator: Key.
```

```

free uName1: bitstring [private].
free uName2: bitstring [private].
free uName3: bitstring [private].
free uName4: bitstring [private].

free groupSecretKey: GK [private].
free secretMessage: bitstring [private].
free groupIdentifier: bitstring.

query attacker(secretMessage).
query attacker(groupSecretKey).

query attacker(uName1).
query attacker(uName2).
query attacker(uName3).
query attacker(uName4).

query mid: MessageId; event(clientGetMessage(mid)) ==¿
event(serverSentMessage(mid)).


event serverSentMessage(MessageId).
event clientGetMessage(MessageId).

fun groupFunction(GK, UID):UK.
fun calculatePDK(UK, MessageId):Key.
fun encrypt(bitstring, Key): bitstring.
fun hashName(bitstring): UID.
fun checkSequenceNumber(sequenceNumber):bool.
fun getSequenceNumber(): sequenceNumber.
fun power(Key, Key): Key.

reduc forall p1: Key, p2: Key;
unpower(power(p1, p2), p2)=p1.

reduc forall m: bitstring, k: Key;
decrypt(encrypt(m, k), k)=m.

reduc
forall id1: UID, id2: UID, id3: UID, mid: MessageId, gsk: GK;
calculateMK(gsk, mid, id1, id2, id3 ) =
power(power(power(groupGenerator,
calculatePDK(groupFunction(gsk, id1), mid)),
calculatePDK(groupFunction(gsk, id2), mid)),
calculatePDK(groupFunction(gsk, id3), mid));
forall id1: UID, id2: UID, id4: UID, mid: MessageId, gsk: GK;
calculateMK(gsk, mid, id1, id2, id4 ) =
power(power(power(groupGenerator,
calculatePDK(groupFunction(gsk, id1), mid)),
calculatePDK(groupFunction(gsk, id2), mid)),
calculatePDK(groupFunction(gsk, id4), mid));
forall id2: UID, id3: UID, id4: UID, mid: MessageId, gsk: GK;
calculateMK(gsk, mid, id2, id3, id4 ) =
power(power(power(groupGenerator,
calculatePDK(groupFunction(gsk, id2), mid)),
calculatePDK(groupFunction(gsk, id3), mid)),
calculatePDK(groupFunction(gsk, id4), mid));
forall id1: UID, id3: UID, id4: UID, mid: MessageId, gsk: GK;
calculateMK(gsk, mid, id1, id3, id4 ) =
power(power(power(groupGenerator,
calculatePDK(groupFunction(gsk, id1), mid)),
calculatePDK(groupFunction(gsk, id3), mid)),
calculatePDK(groupFunction(gsk, id4), mid)).



let centerProcess() =
new msgId: MessageId;
let mk = calculateMK(groupSecretKey, msgId, hashName(uName1),
hashName(uName2), hashName(uName3)) in
out(broadcastChannel, (msgId, groupIdentifier,
encrypt((tag, getSequenceNumber(), secretMessage), mk)));
event serverSentMessage(msgId);
0.


let userProcess(myUserKey: UK, userId: UID, myGroup: bitstring) =
in(broadcastChannel, (msgId: MessageId, gId: bitstring,
encryptedMessage: bitstring));
if ( gId = myGroup ) then
let pdk0 = calculatePDK(myUserKey, msgId) in
new bFactor: Key;
out(smcChannel, (userId, power(power(groupGenerator,pdk0), bFactor)));
in (smcChannel, (originator: UID, payload: Key));
if ( originator = userId ) then
(
let mdk = unpower(payload, bFactor) in
let (mtag: bitstring, ts: sequenceNumber, msg: bitstring) =
decrypt(encryptedMessage, mdk) in
if ( mtag = tag AND checkSequenceNumber(ts) ) then
event clientGetMessage(msgId)
)
else if ( payload ¡¿ groupGenerator ) then
out(smcChannel, (originator, power(payload, pdk0)));
0.

process
let uid1 = hashName(uName1) in
let uid2 = hashName(uName2) in
let uid3 = hashName(uName3) in
let uid4 = hashName(uName4) in
let uk1 = groupFunction(groupSecretKey, uid1) in
let uk2 = groupFunction(groupSecretKey, uid2) in
let uk3 = groupFunction(groupSecretKey, uid3) in
let uk4 = groupFunction(groupSecretKey, uid4) in
(centerProcess())
—
!(
userProcess(uk1, uid1, groupIdentifier) —
124. userProcess(uk2, uid2, groupIdentifier) —
125. userProcess(uk3, uid3, groupIdentifier) —
126. userProcess(uk4, uid4, groupIdentifier)
127. )
```

Given the above data, lines #23 through #32 relate to queries that check the security properties of the scheme. In the same vein, the first six queries check the confidentiality of *secretMessage*, *groupSecretKey* and identities of the users.
The last query, checks the authentication of the server. This query verifies that in every possible execution, if event *clientGetMessage(id)* is fired, there must be a correspondent event *serverSentMessage(id)* with the same message identifier.

9 Further Work

One important feature of a Broadcast Encryption system is traitor tracing. Suppose that a member of a system builds a Decoder machine and sells it to other illegal users. This machine participates in the message extraction protocol and behaves exactly like a normal user. This will enable illegal users to extract messages. However, the method proposed in this paper does not address traitor-tracing concerns, which is regarded as an open problem to be tackled in future studies.

Another related open problem is the broadcaster's ability to convince a third party (usually an arbitrator) that a user of the system is also a member of a traitor set. Most traitor tracing protocols enable a broadcaster to find out if a user is a traitor, yet they appear not to be able to transfer this knowledge to another third party as a proof.

In addition, one practical feature of a Broadcast Encryption is its ability to support multiple broadcasters. For example, in an IPTV scenario, there might be multiple content providers, that want to use a shared broadcasting network to send information to their subscribers. Addressing this problem in our method requires some modification. A trivial solution would be to share user information (user keys and identifiers) among all broadcasters but this will increase the maintenance cost and decrease the security of the system. Providing more robust and secure methods is thus required to deal with this issue.

10 Conclusion

The present paper was intended to define a new dynamic and fair scheme for broadcast encryption based on secure multiparty computation and threshold secret sharing which is at the same time flexible and lightweight. The proposed scheme is indeed dynamic in the sense that adding a new member has no effect on other members of the system. In addition, we defined a security parameter t (used in threshold secret sharing) which makes the scheme more flexible. A secure multiparty computation protocol is also proposed which enables each set of t group members to calculate the message decryption key without revealing their private information. The security of the scheme is verified using applied pi calculus and ProVerif tool as well.

References

[1] Shimshon Berkovits. How to broadcast a secret. In DonaldW. Davies, editor, *Advances in Cryptology EUROCRYPT 91*, volume 547 of *Lecture Notes in Computer Science*, pages 535–541. Springer Berlin Heidelberg, 1991. ISBN 978-3-540-54620-7.

[2] Amos Fiat and Moni Naor. Broadcast encryption. In DouglasR. Stinson, editor, *Advances in Cryptology CRYPTO 93*, volume 773 of *Lecture Notes in Computer Science*, pages 480–491. Springer Berlin Heidelberg, 1994. ISBN 978-3-540-57766-9. doi: 10.1007/3-540-48329-2_40.

[3] Paolo DArco and DouglasR. Stinson. Fault tolerant and distributed broadcast encryption. In Marc Joye, editor, *Topics in Cryptology CT-RSA 2003*, volume 2612 of *Lecture Notes in Computer Science*, pages 263–280. Springer Berlin Heidelberg, 2003. ISBN 978-3-540-00847-7. doi: 10.1007/3-540-36563-X_18.

[4] Jung Hee Cheon, Nam-Su Jho, Myung-Hwan Kim, and Eun Sun Yoo. Skipping, cascade, and combined chain schemes for broadcast encryption. *IEEE Transactions on Information Theory*, 54 (11):5155–5171, 2008.

[5] Yanli Chen and Geng Yang. An efficient broadcast encryption scheme for wireless sensor network. pages 3138–3141. IEEE Press, 2009.

[6] In Tae Kim and Seong Oun Hwang. An efficient identity-based broadcast signcryption scheme for wireless sensor networks. In *Wireless and Pervasive Computing (ISWPC), 2011 6th International Symposium on*, pages 1–6. 2011. doi: 10.1109/ISWPC.2011.5751323.

[7] Jeffrey B. Lotspiech. Broadcast encryption versus public key cryptography in content protection systems. In *Proceedings of the nineth ACM workshop on Digital rights management*, DRM '09, pages 39–46. ACM, New York, NY, USA, 2009. ISBN 978-1-60558-779-0. doi: 10.1145/1655048.1655055.

[8] Hongxia Jin and Jeffrey Lotspiech. Efficient traitor tracing for clone attack in content protection. In *Proceedings of the 2011 ACM Symposium*

on Applied Computing, SAC '11, pages 1544–1549. ACM, New York, NY, USA, 2011. ISBN 978-1-4503-0113-8. doi: 10.1145/1982185.1982513.

[9] Ccile Delerable, Pascal Paillier, and David Pointcheval. *Fully Collusion Secure Dynamic Broadcast Encryption with Constant-Size Ciphertexts or Decryption Keys*, volume 4575 of *Lecture Notes in Computer Science*, pages 39–59. Springer Berlin Heidelberg, 2007. ISBN 978-3-540-73488-8.

[10] Dan Boneh and Brent Waters. A fully collusion resistant broadcast, trace, and revoke system. In *Proceedings of the 13th ACM conference on Computer and communications security*, CCS '06, pages 211–220. ACM, New York, NY, USA, 2006. ISBN 1-59593-518-5. doi: 10.1145/1180405.1180432.

[11] Ik Rae Jeong. Efficient secret broadcast in the broadcasting networks. *Communications Letters, IEEE*, 13(12):1001 – 1003, 2009.

[12] Nelly Fazio and IrippugeMilinda Perera. Outsider-anonymous broadcast encryption with sublinear ciphertexts. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *Public Key Cryptography PKC 2012*, volume 7293 of *Lecture Notes in Computer Science*, pages 225–242. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-30056-1. doi: 10.1007/978-3-642-30057-8_14.

[13] Benot Libert, KennethG. Paterson, and ElizabethA. Quaglia. Anonymous broadcast encryption: Adaptive security and efficient constructions in the standard model. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *Public Key Cryptography PKC 2012*, volume 7293 of *Lecture Notes in Computer Science*, pages 206–224. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-30056-1. doi: 10.1007/978-3-642-30057-8_13.

[14] Xinjun Du, Ying Wang, Jianhua Ge, and Yumin Wang. An id-based broadcast encryption scheme for key distribution. *Broadcasting, IEEE Transactions on*, 51(2):264–266, 2005.

[15] Wu Danfei and Zhang Weimin. Authenticated broadcast encryption with short ciphertexts and private keys. In *Multimedia Technology (ICMT), 2011 International Conference on*, pages 218–221. 2011. doi: 10.1109/ICMT.2011.6001984.

[16] Dalit Naor, Moni Naor, and Jeff Lotspiech. Revocation and tracing schemes for stateless receivers. In Joe Kilian, editor, *Advances in Cryptology CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 41–62. Springer Berlin Heidelberg, 2001. ISBN 978-3-540-42456-7. doi: 10.1007/3-540-44647-8_3.

[17] Xiaoming Wang and Zhiwei Liao. A secure encryption protocol for ad hoc networks. In *Information Science and Engineering (ISISE), 2010 International Symposium on*, pages 578–581. 2010. doi: 10.1109/ISISE.2010.144.

[18] Wenliang Du and Mikhail J. Atallah. Secure multi-party computation problems and their applications: a review and open problems. In *Proceedings of the 2001 workshop on New security paradigms*, NSPW '01, pages 13–22. ACM, New York, NY, USA, 2001. ISBN 1-58113-457-6. doi: 10.1145/508171.508174.

[19] Andrew C. Yao, Andrew C. Yao, Andrew C. Yao, and Andrew C. Yao. Protocols for secure computations. In *Foundations of Computer Science, 1982. SFCS '08. 23rd Annual Symposium on*, pages 160–164. 1982. doi: 10.1109/SFCS.1982.38.

[20] N. Maheshwari and K. Kiyawat. Structural framing of protocol for secure multiparty cloud computation. In *Modelling Symposium (AMS), 2011 Fifth Asia*, pages 187–192.

[21] Ueli Maurer. Secure multi-party computation made simple. In Stelvio Cimato, Giuseppe Persiano, and Clemente Galdi, editors, *Security in Communication Networks*, volume 2576 of *Lecture Notes in Computer Science*, pages 14–28. Springer Berlin Heidelberg, 2003. ISBN 978-3-540-00420-2. doi: 10.1007/3-540-36413-7_2.

[22] Ronald Cramer, Ivan Damgrd, Stefan Dziembowski, Martin Hirt, and Tal Rabin. Efficient multiparty computations secure against an adaptive adversary. In Jacques Stern, editor, *Advances in Cryptology EUROCRYPT 99*, volume 1592 of *Lecture Notes in Computer Science*, pages 311–326. Springer Berlin Heidelberg, 1999. ISBN 978-3-540-65889-4. doi: 10.1007/3-540-48910-X_22.

[23] R. Canetti. Universally composable security: a new paradigm for cryptographic protocols. In *Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on*, pages 136–145. 2001. doi: 10.1109/SFCS.2001.959888.

[24] B. Pfitzmann and M. Waidner. A model for asynchronous reactive systems and its application to secure message transmission. In *Security and Privacy, 2001. S P 2001. Proceedings. 2001 IEEE Symposium on*, pages 184–200. 2001. doi: 10.1109/SECPRI.2001.924298.

[25] Ran Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology: the journal of the International Association for Cryptologic Research*, 13(1):143–202, 2000.

[26] Mikhail J. Atallah and Keith B. Frikken. Securely outsourcing linear algebra computations. In *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*, ASIACCS '10, pages 48–59. ACM, New

York, NY, USA, 2010. ISBN 978-1-60558-936-7. doi: 10.1145/1755688.1755695.

[27] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

[28] Ronald Cramer, Ivan Damgrd, and JesperB. Nielsen. Multiparty computation from threshold homomorphic encryption. In Birgit Pfitzmann, editor, *Advances in Cryptology EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 280–300. Springer Berlin Heidelberg, 2001. ISBN 978-3-540-42070-5. doi: 10.1007/3-540-44987-6_18.

[29] Moni Naor and Benny Pinkas. Efficient oblivious transfer protocols. In *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, SODA '01, pages 448–457. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2001. ISBN 0-89871-490-7.

[30] Lin Keng-Pei and Chen Ming-Syan. On the design and analysis of the privacy-preserving svm classifier. *Knowledge and Data Engineering, IEEE Transactions on*, 23(11):1704–1717, 2011.

[31] Dung Luong The, Bao Ho Tu, Binh Nguyen The, and Hoang Tuan-Hao. Privacy preserving classification in two-dimension distributed data. In *Knowledge and Systems Engineering (KSE), 2010 Second International Conference on*, pages 96–103.

[32] Z. Yu and N. Zhang. Achieving privacy-preserving computation on data grids. In *Computers and Communications, 2007. ISCC 2007. 12th IEEE Symposium on*, pages 763–768.

[33] Yang Piyi, Cao Zhenfu, Dong Xiaolei, and T. A. Zia. An efficient privacy preserving data aggregation scheme with constant communication overheads for wireless sensor networks. *Communications Letters, IEEE*, 15(11):1205–1207, 2011.

[34] D. K. Mishra and M. Chandwani. A zero-hacking protocol for secure multiparty computation using multiple ttp. In *TENCON 2008 - 2008 IEEE Region 10 Conference*, pages 1–6.

[35] Gavin Lowe. *Breaking and fixing the Needham-Schroeder Public-Key Protocol using FDR*, volume 1055 of *Lecture Notes in Computer Science*, chapter 10, pages 147–166. Springer Berlin Heidelberg, 1996.

[36] Martín Abadi and Andrew D. Gordon. A calculus for cryptographic protocols: the spi calculus. In *Proceedings of the 4th ACM conference on Computer and communications security*, CCS '97, pages 36–47. ACM, New York, NY, USA, 1997. ISBN 0-89791-912-2. doi: 10.1145/266420.266432.

[37] Martn Abadi and Bruno Blanchet. *Computer-Assisted Verification of a Protocol for Certified Email*, volume 2694 of *Lecture Notes in Computer Science*, chapter 17, pages 316–335. Springer Berlin Heidelberg, 2003.

[38] B. Blanchet. An efficient cryptographic protocol verifier based on prolog rules. In *Computer Security Foundations Workshop, 2001. Proceedings. 14th IEEE*, pages 82–96. 2001. doi: 10.1109/CSFW.2001.930138.

[39] Steve Kremer and Mark Ryan. Analysis of an electronic voting protocol in the applied pi calculus. In Mooly Sagiv, editor, *Programming Languages and Systems*, volume 3444 of *Lecture Notes in Computer Science*, pages 186–200. Springer Berlin Heidelberg, 2005. ISBN 978-3-540-25435-5. doi: 10.1007/978-3-540-31987-0_14.

[40] Atsushi Fujioka, Tatsuaki Okamoto, and Kazuo Ohta. A practical secret voting scheme for large scale elections. In Jennifer Seberry and Yuliang Zheng, editors, *Advances in Cryptology - AUSCRYPT '92*, volume 718 of *Lecture Notes in Computer Science*, pages 244–251. Springer Berlin Heidelberg, 1993. ISBN 978-3-540-57220-6. doi: 10.1007/3-540-57220-1_66.

[41] Stphanie Delaune, Steve Kremer, and Mark Ryan. *Verifying Privacy-Type Properties of Electronic Voting Protocols: A Taster*, volume 6000 of *Lecture Notes in Computer Science*, chapter 18, pages 289–309. Springer Berlin Heidelberg, 2010.

[42] R. Kusters and T. Truderung. An epistemic approach to coercion-resistance for electronic voting protocols. In *Security and Privacy, 2009 30th IEEE Symposium on*, pages 251–266. 2009. doi: 10.1109/SP.2009.13.

[43] Steve Kremer, Mark Ryan, and Ben Smyth. Election verifiability in electronic voting protocols. In Dimitris Gritzalis, Bart Preneel, and Marianthi Theoharidou, editors, *Computer Security ESORICS 2010*, volume 6345 of *Lecture Notes in Computer Science*, pages 389–404. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-15496-6. doi: 10.1007/978-3-642-15497-3_24.

[44] Ben Adida. Helios: web-based open-audit voting. pages 335–348. USENIX Association, 2008.

[45] M.R. Clarkson, S. Chong, and A.C. Myers. Civitas: Toward a secure voting system. In *Security and Privacy, 2008. SP 2008. IEEE Symposium on*, pages 354–368. 2008. doi: 10.1109/SP.2008.32.

[46] Ben Smyth, Mark Ryan, Steve Kremer, and Mounira Kourjieh. Towards automatic analysis of election verifiability properties. In Alessandro Armando and Gavin Lowe, editors, *Automated Reasoning for Security Protocol Analysis and Issues in the Theory of Security*, volume 6186 of *Lecture Notes in Computer Science*, pages 146–163. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-16073-8. doi: 10.1007/978-3-642-16074-5_11.

[47] Liqun Chen and Mark Ryan. Attack, solution and verification for shared authorisation data in tcg tpm. In Pierpaolo Degano and JoshuaD. Guttman, editors, *Formal Aspects in Security and Trust*, volume 5983 of *Lecture Notes in Computer Science*, pages 201–216. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-12458-7. doi: 10.1007/978-3-642-12459-4_15.

[48] M. Backes, M. Maffei, and D. Unruh. Zero-knowledge in the applied pi-calculus and automated verification of the direct anonymous attestation protocol. In *Security and Privacy, 2008. SP 2008. IEEE Symposium on*, pages 202–215. 2008. doi: 10.1109/SP.2008.23.

[49] Bruno Blanchet and Avik Chaudhuri. Automated formal analysis of a protocol for secure file sharing on untrusted storage. In *Proceedings of the 29th IEEE Symposium on Security and Privacy (S&P'08)*, pages 417–431. IEEE, 2008.

[50] Martin Abadi, Bruno Blanchet, and Cedric Fournet. Just fast keying in the pi calculus. *ACM Trans. Inf. Syst. Secur.*, 10(3):9, 2007.

[51] Chetan Bansal, Karthikeyan Bhargavan, Antoine Delignat-Lavaud, and Sergio Maffeis. Keys to the cloud: formal analysis and concrete attacks on encrypted web storage. In *Proceedings of the Second international conference on Principles of Security and Trust*, POST'13, pages 126–146. Springer-Verlag, Berlin, Heidelberg, 2013. ISBN 978-3-642-36829-5. doi: 10.1007/978-3-642-36830-1_7.

[52] Jannik Dreier, Pascal Lafourcade, and Yassine Lakhnech. *Formal Verification of e-Auction Protocols*, volume 7796 of *Lecture Notes in Computer Science*, chapter 13, pages 247–266. Springer Berlin Heidelberg, 2013.

[53] Myrto Arapinis, Vronique Cortier, Steve Kremer, and Mark Ryan. *Practical Everlasting Privacy*, volume 7796 of *Lecture Notes in Computer Science*, chapter 2, pages 21–40. Springer Berlin Heidelberg, 2013.

[54] Denise Demirel, Jeroen Van De Graaf, and Roberto Arajo. Improving helios with everlasting privacy towards the public. In *Proceedings of the 2012 international conference on Electronic Voting Technology/Workshop on Trustworthy Elections*, EVT/WOTE'12, pages 8–8. USENIX Association, Berkeley, CA, USA, 2012.

[55] Tal Moran and Moni Naor. *Receipt-Free Universally-Verifiable Voting with Everlasting Privacy*, volume 4117 of *Lecture Notes in Computer Science*, chapter 22, pages 373–392. Springer Berlin Heidelberg, 2006.

[56] Martin Abadi and Cedric Fournet. Mobile values, new names, and secure communication. *SIGPLAN Not.*, 36(3):104–115, 2001.

[57] Robin Milner. *Communicating and mobile systems: the pi-calculus*. Cambridge University Press, 1999.

[58] Danny Dolev and Andrew C. Yao. On the security of public key protocols. Technical report, Stanford University, 1981.

[59] M. S. Mohammadi and A. G. Bafghi. A dynamic, zero-message broadcast encryption scheme based on secure multiparty computation. In *Information Security and Cryptology (ISCISC), 2012 9th International ISC Conference on*, pages 12–17.

[60] V. I. Nechaev. Complexity of a determinate algorithm for the discrete logarithm. *Mathematical Notes*, 55(2):165–172, 1994.

[61] Hung-Yu Lin and Lein Harn. Fair reconstruction of a secret. *Inf. Process. Lett.*, 55(1):45–47, 1995.

[62] Dan Boneh, Craig Gentry, and Brent Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In Victor Shoup, editor, *Advances in Cryptology CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 258–275. Springer Berlin Heidelberg, 2005. ISBN 978-3-540-28114-6.

[63] Adrian Perrig. Efficient collaborative key management protocols for secure autonomous group communication. In *International Workshop on Cryptographic Techniques and E-Commerce CrypTEC '99*, pages 192–202.

Mahdi Soodkhah Mohammadi has received his B.S. degree in software engineering from Amirkabir University of Technology. His M.S. degree was received in the same university in the field of information technology. He is currently a Ph.D. student in Ferdowsi University of Mashhad. His research interests are cryptography, data security and electronic commerce. He is constantly faced with different challenges in information security field, and interested in finding efficient solutions to them.

Abbas Ghaemi Bafghi received his B.S. degree in Applied Mathematics in Computer from Ferdowsi University of Mashhad, Iran, in 1995, and the M.S. degree in Computer engineering from Amirkabir (Tehran Polytechnique) University of Technology, Iran, in 1997. He received his Ph.D. degree in Computer engineering from Amirkabir (Tehran Polytechnique) University of Technology, Iran in 2004. He is a member of the Computer Society of Iran (CSI) and Iranian Society of Cryptology (ISC). He is an assistant professor in the Department of Computer Engineering, Ferdowsi University of Mashhad, Iran. His research interests are cryptology and security. He has published more than 50 conference and journal papers.