World Engineering & Applied Sciences Journal 7 (4): 260-266, 2016
ISSN 2079-2204

DOI: 10.5829/idosi.weasj.2016.260.266

Green Group Key Management in Wireless Mesh Networks (WMN'S)

[1]P.S. Kanimozhy and [2]B. Dojohn Loyd

[1]Mtech(CSE), SRM University, Chennai, India
[2]Department of CSE, SRM University, Chennai, India

Abstract: Public key cryptography is well-suited to Wireless Mesh Networks, as it requires no prior secure group key distribution mechanism. Securing group communications over WMNs bring additional challenges due to member mobility and explode in the number of members. To reduce the computational and communication cost and energy for secure and efficient transmission in a secure group key communication. We propose to use a TGDH protocol where the members are arranged in a hierarchical binary tree. To reduce the re-key complexity, Queue batch algorithm is employed such that re-keying is performed on a group of join and leave requests at regular re-key intervals. We Consider a settings where master keys are preloaded on clients according to an arbitrary distribution and we present a protocol that uses session keys derived from master keys to establish a group key which is information-theoretically secure. When master keys are distributed randomly, our protocol requires O ($\log_b$ t) transmissions. This paper deals with the group key management, distribution of session keys and refreshment of the Re-keying material.

Key words: Public key cryptography · Wireless Mesh Networks · Queue-batch algorithm · TGDH protocol

INTRODUCTION

Security is main focused area in group communication. While symmetric key algorithms provide a lightweight means of establish data secure in power-constrained devices, they may be ill-suited to applications where the devices can be compromised. For an example, if a single client in a sensor network employing AES-256 with a common key is compromised, then all of the other clients must be rekeyed [1]. Public key algorithms (PKAs) have traditionally been seen as incompatible with WMNs for two reasons:

- Public key algorithms (PKAs) are much more computationally complex than symmetric key algorithms [1].
- PKAs are tailored for unicast transmission – i.e., the public encryption key and private decryption key are generated by a unique data destination [2, 3].

Public key algorithms can natively support multicast traffic by replacing *t*-destination multicast session with *t* parallel unicast sessions. It is more energy efficient to have the destination clients first securely establish a group key and then derive a common public/private key pair from that group key. This allows the data to be encrypted by the source, efficiently multicast to all destinations and then decrypted by all destinations simultaneously [4-7]. Traditional group key distribution protocols, wherein one destination generates and distributes a pair of group key to each of the other t - 1 destinations in turn, require $O(t)$ transmissions[1] [8, 9]. Existing group key agreement protocols, where the destinations cooperatively establish a key via messaging, also require $O(t)$ transmission. In the present work, group key establishment protocols that require a sub linear number of transmissions and also refreshment of the keying material are sought.

Our work on energy-efficient group key agreement was suggested by recent results on the universal recovery problem, which were review in Section II. In Section III we present a protocol that enables t multicast destinations to securely agree on a common group key pair using $O(\log_b t)$ transmissions, where b is a parameter that enables trades between energy efficiency and security. The transmissions are binary sums of session keys, which are derived from master keys that are randomly distributed amongst the nodes.

Corresponding Author: B. Dojohn Loyd, Department of CSE, SRM University, Chennai, India.

The protocol description in Section III assumes that nodes are loaded with master keys to deployment; Section IV describes a refreshment of group key using queue batch algorithm.

Our Contribution in this work, we assume that master keys are loaded on clients prior to group key agreement. Master key are preloaded to the network is deployed (i.e., preloading) or dynamically via Diffie-Hellman key exchanges. When a group of clients wants to establish a common key, they derive session keys from the subset of master keys that are shared by at least in a group for two members. The session key shared by the largest number of group members becomes the group key. That key is distributed to the rest of the group members via public multicast transmissions comprising the binary sum of the group key and session keys. By properly designing the master key distribution, the number of transmissions required for group key agreement can be made smaller than the group size. For example, if the master keys are randomly distributed such that a given client possesses a given key with probability 1-1/b, then the number of transmissions depends on the group size t as $O(\log_b t)$.

Our approach is inspired by information-theoretic results on group key agreement. Building on [3], Courtade and Halford recently described the minimum number of public transmissions required for key agreement assuming an arbitrary master key distribution [4].

Our protocol applies a greedy heuristic to approximate this optimum in the polynomial time. In particular, it was shown in algorithm (2) a key agreement protocol that minimizes the number of transmissions is NP-hard.

[1]We say that a function f(n) = O (g(n)) if there exists constants n_0 and c such that f(n) $\leq$ cg(n) for all values of $n > n_0$.

The key results of this paper are:

- The specification of a protocol for group key agreement that can be measured in polynomial time (in the group size t) for each distribution of the master keys.
- A group key agreement protocol enables a group of users communicating over an entrusted, open network to come up with a common secret value called a session key.
- Refreshment of group key material using Queue-Batch Algorithm.

Information-Theoretic Secrecy Generation

Universal Recovery: Our approach is motivated by recent results on secret key agreement via Universal Recovery problem. Suppose that k packets P = {P_1, $P2$} are distributed among a network of t clients with n nodes. What is the minimum number of transmissions M required to recover all k packets at all t clients? The difference between k and M can be exploited for secret key agreement. The goal of the universal recovery problem is to distribute all k packets to all nodes with a minimum number of transmissions.

Secrecy Generation by Example: Let $P_j \subseteq P$ be the subset of packets initially available at node j such that the overlap is permitted between these sets (i.e., we don't require $P_i = P_j = \phi$).

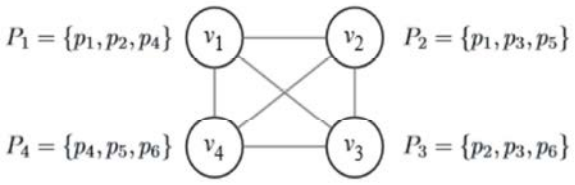


Fig. 1: Universal recovery in the fully connected network requires 4 transmissions (i.e.., M=4).

In Figure (1) each packets are shared by at least two clients (i.e., that packet must simply broadcasted to the other group members so that the difference between K and M does not change). Consider the network illustrated in Figure 1, where k = 6, 256-bit packets are distributed amongst n=4 nodes. Universal recovery of P = {p_1, $p6$} can be achieved via four transmissions:

1. Node v_1 transmits the binary sum $t_1 = p_1 + p_2$.
Node v_2 recovers $t_1 + p_1 = p_2$.
Node v_3 recovers $t_1 + p_2 = p_2$.

2. Node v_2 transmits the binary sum $t_2 = p_3 + p_5$.
Node v_3 recovers $t_2 + p_3 = p_5$.
Node v_4 recovers $t_2 + p_5 = p_3$.

3. Node v_3 transmits the binary sum $t_3 = p_2 + p_6$.
Node v_1 recovers $t_3 + p_2 = p_6$.
Node v_2 recovers $t_3 + p_2 = p_6$.
Node v_4 recovers $t_3 + p_6 = p_2$ and $t_1 + p_2 = p_1$.

4. Node v_4 transmits the binary sum $t_4 = p_4 + p_5$.
Node v_1 recovers $t_4 + p_4 = p_5$ and $t_2 + p_5 = p_3$.
Node v_2 recovers $t_4 + p_5 = p_4$.
Node v_3 recovers $t_4 + p_5 = p_4$.

A total of $k = 6$ packets have been disturbed among the clients in Fig. (1) Indicates that two packets of common randomness k_1 and K_2 can be generated. The result indicates that if, all the packets are cryptographic keys, then by this scheme the packets worth of K - M = 2 secrecy. Since the group key agreement requires only a single shared secret packet, universal recovery is not necessary. In this example two packets worth of secrecy were generated using public multicast transmissions, In general one secret packet was generated using one public multicast transmission and two packets should be recovered by destination clients.

Group Key Agreement with Preload Master Key

Protocol Specification: Given an n node network, a total of $(n, 2)$ pairwise cryptographic master keys are generated and distributed to the network nodes such that each pairwise key is shared by exactly two nodes. the pairwise keys are used to ensure that for every node pair (d_i , dj), we can efficiently find a packet P_k such that $\{ di , dj \} \subseteq O_k$ in linear time.

Suppose that node s wishes to initiate a secure multicast session to t destination nodes $D = \{d , dt\}$. This can be done using the following protocol:

Input: $D = \{d_1, ..., d_t\}$,
$O = \{O_j\}_{j?PD}$.
Output: $C \subseteq O$.
$U \leftarrow D$, $C \leftarrow \varnothing$
while $U \varnothing$ do
Select $S \in O$ that minimizes $|S \cap U|$
$U \leftarrow U \setminus S$, $C \leftarrow C \cup \{S\}$
End

Algorithm 1: Greedy, linear-time approximation of set cover.

Input: Occupancy sets $O = \{O_j\}_j \in K_G$, group $G = \{g_1, . . . , g_t\}$ and a common PRF $\phi()$.

Output: Group key $s_{j0, u}$ for session with unique identifier u.

where $j_0 \leftarrow$ index of largest occupancy set in O, $C \leftarrow O_{j0}$, $l \leftarrow 1$;
if $g_i \in O_{j0}$ then
compute the group key $s_{j0, u} \leftarrow \phi (k_{j0} , u)$;
end
while $C \neq G$ do
$j_l \leftarrow$ index of an occupancy set $O_{jl} \in O$ satisfying
$O_{jl} \cap C = \varnothing$ that maximizes $|O_{jl} \setminus C|$;
$i_l \leftarrow$ a client in $O_{jl} \cap C$;
if $g_i = i_l$ then
compute the l^{th} one-time pad s_{jl}, $u \leftarrow \phi (k_{jl} , u)$;
compute the bit-wise sum m_l, $u = s_{j0, u} \oplus s_{jl, u}$; transmit $m_{l, u}$ to all clients in $O_{jl} \setminus C$;
else if $g_i \phi O_{jl} \setminus C$ then
compute the l th one-time pad $s_{jl, u} \leftarrow \phi (k_{jl} , u)$; receive $m_{l, u}$ from client i_l;
recover the group key $s_{j0, u} = m_{l, u} \oplus s_{jl, u}$;
End
$C \leftarrow C \cup O_{jl}$, $l \cup l + 1$;
end

Algorithm 2: Proposed protocol for defining a transmission schedule from the set cover solution running at client $g_i \in G$.

Protocol Discussion: The common PRF that is employed in Algorithm 1 ensures that the public transmissions are computationally indistinguishable from random packets. Algorithm 1 determines a group key agreement protocol for any set of clients in a network that has been loaded with master keys according to a distribution. We use session keys derived from master keys in our protocol to provide forward and backward security.

Pair wise Key Agreement: If clients g_i and g_j are not previously interacted, then they establish a key for secure pair wise communications via a traditional two party Diffie-Hellman key exchange. The pair wise keys are exchanged to provide a secure random key exchange [8-10].

Simulation Results – Energy Efficiency: In our simulations, we consider master key loading schemes where R keys are distributed randomly among t clients such that any client possesses any master key with probability β when $f(n) = \log_2(n)$. When only pairwise packets are used, exactly t - 2 transmissions are required to generate the common secret packet. Figure 2 compares the average number of public multicast transmissions required for group key agreement when $t = 50$, $R = [t/\beta]$ and β varies. Observe that as β increases, the number of transmissions required for key agreement decreases. O(t) behavior is replaced by a logarithmic dependence on t. To highlight the sub linear growth achieved by our protocol, the linear growth exhibited by Burmester and Desmedt's protocol [11-17] is also shown. The difference in performance between the two protocols can be attributed largely to our use of session keys derived from preloaded master keys.

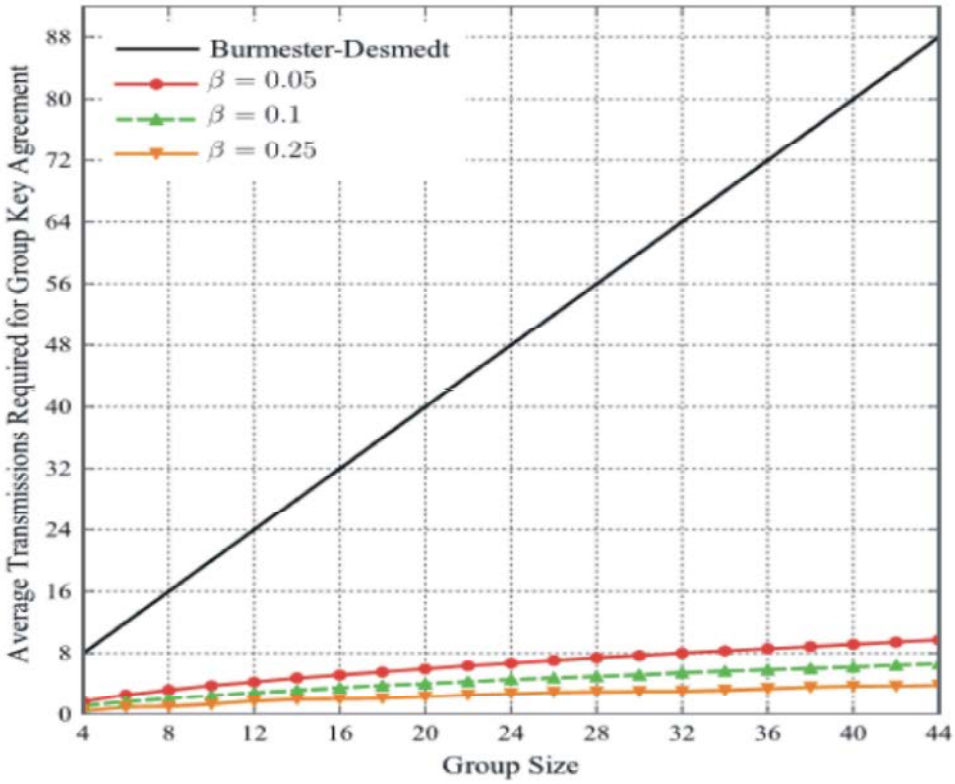


Fig. 2: Number of public multicast transmissions required for group key agreement via the proposed protocol when [50/β] master keys are loaded randomly in a 50-client network. Our protocol exhibits a sub linear growth in the number of transmissions as the group size increases

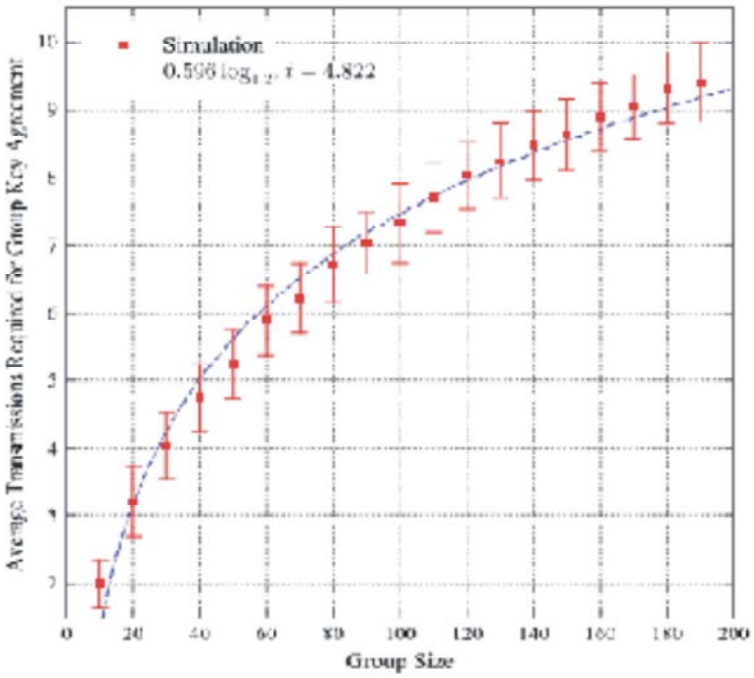


Fig. 3: Number of public multicast transmissions for group key agreement when $n = 200$, $R = 100$, $\beta = 0.2$ and $b = 1.25$ for the proposed protocol. The number of transmissions required for key agreement grows as the logarithm of the group size

Simulation Results – Energy vs. Security Trades: Fig. 3 illustrates the tradeoff between energy effciency and security against undetected compromised clients in the proposed protocol. Energy efficiency is measured in terms of the number of public multicast transmissions. For different values of β, we measured the average number of public multicast transmissions required for group key agreement among 10, 15, 20 and 25 clients in a 50-client network. The total number of keys *R* was set to [50/β] so that the average number of keys per client remained constant.

Queue Batch Re-key Agreement Protocol: The problems with the existing system are Group Key information depends on centralized key server and also both the Computational and Communication cost is enormous. The proposed approach uses TGDH protocol where the members are arranged in a hierarchical binary tree which uses Queue-batch algorithm for re-keying. This algorithm can substantially reduce the computation, communication, Latency and workload in a highly dynamic environment.

Tree Based Group Diffie Hellman (TGDH) Protocol

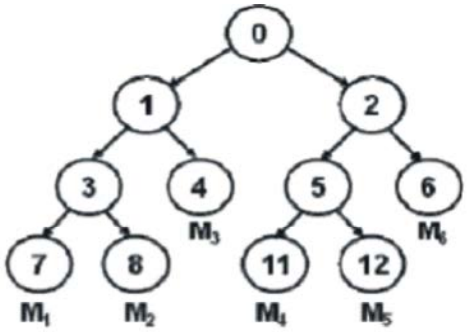
Group Key Agreement Schemes: Based on the Diffie-Hellman protocol [2], each member maintains a set of keys, which are arranged in a hierarchical binary tree. We assign a node ID to every tree node. For a given node, we associate a master (or private) key K*v* and a slave (or public) key BK*v* are performed in a group of prime order *p* with generator a: the blinded key of node *v* can be generated by;

$$BK_V = a^{K_V} \bmod P$$

The following is the minimal requirement for computing the group key.

TGDH: Group Key Generation: Each leaf node in the tree corresponds to the individual master and slave keys of a group member Mi. Every member holds all the master keys along its key path starting from its associated leaf node up to the root node. Therefore, the master key held by the root node is shared by all the members and is regarded as the group key.

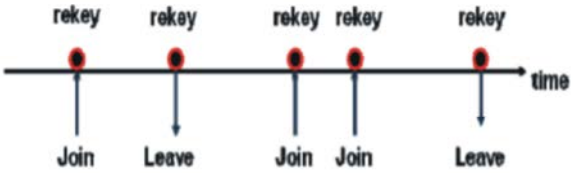
The figure below illustrates a possible key tree with six members M1 to M6. For example, member M1 and holds the keys at nodes 7, 3, 1 and 0. The master key at node 0 is the group key of this peer group.



Eg., M1 generates the group key via

- K_7, $BK_8 = K_3$
- K_3, $BK_4 = K_1$
- K_1, $BK_2 = K_0$ (Group Key)

TGDH: Membership Events: Rekeying (renewing the keys of the nodes) is performed at every single join/leave event to ensure backward and forward Secrecy.



Queue-Batch Algorithm: In the existing system the rekeying is at the beginning of every rekey interval, resultant in a high processing load during the update instance and thereby delay the start of the secure group communication. The processing load includes the computation and communication cost of the exponentiation operations to generate the keys.

The aim of this algorithm is to reduce the rekeying load by pre-processing the joining members in the queue during the idle rekey interval. The Queue-batch algorithm is divided into two phases; they are Queue-sub tree phase and the Queue-merge phase. The first phase occurs whenever a new member joins the communication group during the rekey interval. In this case, we append this new member in a temporary key tree T'.

Queue-sub tree (T')
1. If (a new member joins) {
2. If (T'==NULL)/*no new member in T'*/
3 Create a new tree T' with the only new member;
3. Else {/*there are new members in T'*/
4. Find the insertion node;
5. Add the new member to T';

6. Elect the rightmost member under the sub tree rooted at the sibling of the joining node to be the sponsor;
7. If (sponsor)/* sponsor's responsibility*/
8. Re-key renewed nodes and broadcast new blinded keys;
9.}
10.}

The second phase occurs at the beginning of every rekey interval and we merge the temporary tree T' (which contains all newly joining members) to an existing key tree T.

Queue-merge (T, T', Ml, L)

1. If (L==0) {/* There is no leave*/
2. Add T' to either the shallowest node (which need to be the leaf node) of T such that merge will not increase the resulting tree height, or the root node of T if the merge to any location will increase the resulting tree height;
3.} else {/* there are leaves*/
4. Add T' to the highest leave position of the key tree T;
5. Remove remaining L-1 leaving leaf nodes and promote their siblings;
6.}
7. Elect members to be sponsors if they are the rightmost members of the sub tree model rooted at the sibling nodes of the departed leaf nodes in T, or they are the rightmost member of T';
8. If (sponsor) / *sponsor's responsibility*/
9. Re-key renewed nodes and broadcast new blinded keys;

Analysis of the Queue-Batch Algorithm: The main aim of the Queue-batch algorithm is that the idle rekey interval to pre-process certain rekeying operations. When we compare its performance with the other algorithms like Rebuild/Batch algorithms, we only need to consider the rekey operations occurring at the beginning of each rekey interval operations.

When J = 0, Queue-batch is equals to batch in the leave scenario completely. In case J > 0, the number of renewed nodes during the Queue-merge phase is equivalent to that of batch when J = 1.

The Queue-batch algorithm is illustrated in the Figure. 4, where the members M2, M5 and M7 wish to leave the communication group, while M8 wish to leave. Then the re-keying process is as follows:

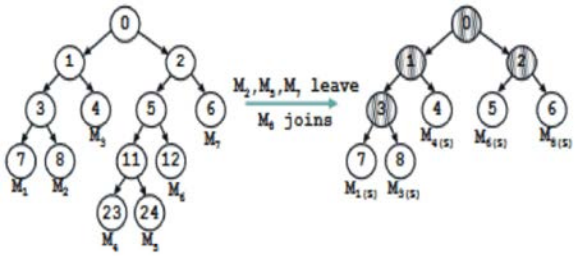


Fig. 4: Example for Queue Batch Algorithm

- In the Queue-sub tree phase, the three new members M2, M5 and M7 first form a sub tree T'. M7, in this case, is elected to be the sponsor.
- In the Queue-merge phase, the tree T' is added at the highest departed position, which is at node 6. Also, the blinded key of the root node of T, which is BK6, is broadcast by M7.
- The sponsors M1, M6 and M7 are elected. M1 renews the secret key K1 and broadcasts the blinded key BK1. M6 renews the secret key K2 and broadcasts the blinded key BK2.
- Finally, all members can compute the group key.

CONCLUSION

Motivated by a desire to employ public key cryptography in WMNs' networks, this paper described an energy-efficient group key agreement protocol. Whereas existing approaches require $O(t)$ transmissions to establish a group key among t nodes, our protocols require $O(\log_b t)$ transmissions at steady state, where b is a parameter that enables trades between energy efficiency and security against compromised nodes. When combined with an energy-efficient public key algorithm. Our approach was inspired by recent results on the universal recovery problem the protocols described herein generate a single packet of secrecy via public multicast transmission.

To reduce the rekey complexity, we propose to use an interval-based rekey approach so that we can group multiple join/leave requests and process them at the same time. In particular, we show that the Queue-batch algorithm can significantly reduce both computational and communication costs. This reduction enables a more efficient way to manage secure group communication. Future work will address the translation of the abstract protocols described in this paper to an Application Layer solution suitable for implementation in power-constrained

wireless networks. The key issue to address in that translation is protocol scalability. For example, a pseudo-random master key distribution that can be derived from a single seed could be used in place of the random distributions considered herein. This would enable low-over head network join/leave operations while maintaining energy-efficient group key agreement.

REFERENCES

1. Carman David W., Peter S. Kruus and Brian J. Matt, 2000. Constraints and approaches for distributed sensor network security. NAI Labs Technical Report #00-010.
2. Tyagi, H., 2013. Common information and secret key capacity, IEEE Trans. Information Theory, 59(9): 5627-5640.
3 Steiner Michael, Gene Tsudik and Michael Waidner, 2000. Key Agreement in Dynamic Peer Groups, IEEE Transactions on Parallel and Distributed Systems, 11(8): 769-780.
4. Burmester, M. and Y. Desmedt, 1995. A secure and efficient conference key distribution system. In Advances in Cryptology – EUROCRYPT '94, of Lecture Notes in Computer Science, Springer-Verlag, 950: 275-286.
5. Jarrah Omar Al and Ramzy Saifan, 2008. A novel key management algorithm in sensor networks, Proceedings of the 6th International Conference on Advances in Mobile Computing and Multimedia.
6. Kong PJiejun, Petros Zerfos, Haiyun Luo, Songwu Lu and Lixia Zhang, 2001. Providing Robust and Ubiquitous Security Support for Mobile Ad-Hoc Networks. International Conference on Network Protocols (ICNP).
7. Halford, T.R., T.A. Courtade and K.M. Chugg, 2013. Energy-efficient, secure group key agreement for ad hoc networks, in Proc. IEEE Communications and Network Security Conf., Washington, DC, pp: 181-188.
8. Courtade, T.A. and T.R. Halford, 2014. Coded cooperative data exchange for a secrect key, in Proc. IEEE International Sympon Information Theory, Honolulu, HI, pp: 776-780.
9. Courtade, T.A. and R.D. Wesel, 2014. Coded cooperative data exchange in multichip networks, IEEE Trans. Information Theory, 60(2): 1136-1158.
10. Diffie, W. and M. Hellman, 1976. New directions in cryptography.IEEE Transactions on Information Theory, IT, 22(6): 644-654.
11. Kim, Y., A. Perrig and G. Tsudik, 2000. Simple and fault-tolerant key agreement for dynamic collaborative groups. In ACM CCS 2000.
12. Tseng, Y.M. and J.K. Jan, 2001.Anonymous conference key distribution systems based on the discrete logarithm problem, Computer Communications, 22(8): 749-754.
13. Horng, G., An efficient and secure protocol for multi-party key establishment, Computer J., 44(5): 463-470.
14. Vercellis, C., 1984. A probabilistic analysis of the set covering problem, Annals of Operations Research, 1(3): 255-271.
15. Telelis, O.A. and V. Zissimopoulos, 2005. Absolute (logm) error in approximating random set cover: An average case analysis, Information Processing Letters, 94(4): 171-177.
16. Anastasi, G., M. Conti, M. Di Francesco and A. Passarella, 2009. Energy conservation in wireless sensor networks: A survey, Elsevier Ad Hoc Networks, 7(3): 537-568.
17. Sherman, A.T. and D.A. McGrew, 2003. Key establishment in large dynamic groups using one-way function trees, IEEE Transactions on Software Engineering, 29(5): 444-458.
18. Li, X.S., Y.R. Yang, M.G. Gouda and S.S. Lam, 2001. Batch re-keying for secure group communications, (WWW'10), 525-534.