Group Key Management Using Public Key Exchange

Babak Daghighi, Laiha Mat Kiah, and Saeedeh Afkhami Rad

***Abstract*—Group based applications have gained popularity in recent years. As these applications typically involve communication over open network, security is an important requirement. Group key management is one of main block in securing group communication. Though, centralized and distributed group key schemes rely on at least one explicit key manager for generating traffic key, contributory schemes employ the contribution of all members of a group for initiating new traffic key. This work presents an overview of existing contributory group key management schemes which specifically employ public key exchange algorithms for generating traffic key among members of a group. The initial operation which is used to generate first traffic key among members of group is demonstrated. Subsequently, required operations for provision of backward and forward secrecy during upon any changes in group membership are explored. And finally, the investigated schemes are analyzed and then compared in term of their communication cost.**

***Index Terms*—Group communication, group key managemen, cryptographic algorithms**

I. Introduction

In recent years, many group based applications such as multicast conferencing, information dissemination service, satellite TV distribution services and, more generally, applications supporting collaborative work have significantly gained in popularity. Since most of group based applications take place over insecure network, basic security services such as confidentiality, data integrity and entity authentication are necessary for them. These services can be established by a shared common key between entities in group. This key is used to encrypt all traffic that destined to a particular group. As a result, only members of the group can decrypt the received message.

One of fundamental challenges in designing secure and reliable group communication system is to manage a group key. There are several group key management schemes for disseminating group key to members of a group which can be classified into three categories as: 1) Centralized 2) Distributed 3) Contributory.

Centralized group key management involves a single entity (i.e. a group controller GC) which is responsible to generate, distribute and update the group key. One of the famous schemes in this category Logical Key Hierarch (LKH) that was proposed by several research groups nearly at the same time [1], [2]. Other existing approach can be found in, [3], [4], [5]. This approach has two problems: 1) dependencies on a key server leave a single point of failure and 2) it must be constantly available and present during group operation.

In another approach, distributed or also known as decentralized group key management, a large group is split into small subgroups. Each subgroup has its own subgroup controller which is responsible for key management in its subgroup. The first scheme was IOLUS [6] and then followed by some improvements schemes such as [7]-[9].

This approach can minimize the problem of concentrating on a single entity as a key server.

In contrast to these approaches, contributory group key management has no explicit key distributor center and all members contribute to manage the key(s) generation. This scheme helps to uniform distribution of the work load for key management and eliminates the need for central entity. This approach alleviates the problem of single points of failures in centralized group key management. Some contributory group key management schemes have been presented in [10], [11] [12], [13].

The goal of this paper is providing an overview of existing contributory group key management schemes. We will demonstrate the mechanisms which are used by these protocols that prevent a new joining member to access previous information as well as a leaving member to access future data communication within group communication in order to provide backward and forward secrecy.

The protocol that have been selected for analysis are ING, GDH1, GDH2, GDH3 [13], [14], Octopus [15], STR [11], [16], TGDH [10], [17] and CRTDH [18], which present variation of n-party Diffie Hellman group key agreement schemes.

The rest of paper is organized as follows: section II introduces public two party key exchanges. The group key management schemes are presented in section III. Section IV addresses a qualitative comparison of the discussed protocol. And finally, conclusion is presented in Section V.

II. Public Two Party Key Exchange

A two party key exchange protocol is a protocol that allows two entities who have not contacted each other before, publicly exchange their information so that at the end of protocol they have a same share secret key. Any eavesdropper that listens to communication is unable to obtain this key. One of the first published public key crypto algorithm was Diffie Hellman (DH) [19].

If two entities are interested in using the DH for secure communication, they agree on Diffie Hellman group, a large prime number p and a generator number g. Next, the two entities choose secret private key x_a and x_b. They compute the public (blinded) key y_a and y_b by using secret private key and send them to each other.

Manuscript received November 11, 2012; revised January 20, 2013.

Babak Daghighi and Laiha Mat Kiah are with the Faculty of Computer Science & IT, University of Malaya, KL, CO 50603 Malaysia (e-mail: babak@siswa.um.edu.my, misslaiha@um.edu.my).

Saeedeh Afkhami Rad is with the Faculty of Computer Science, Azad University, Mashad, CO 80309 Iran (e-mail: s.afkhamirad@gmail.com).

DOI: 10.7763/IJCTE.2013.V5.801

$$A \rightarrow B: y_a = g^{x_a} \bmod p$$

$$B \rightarrow A: y_b = g^{x_b} \bmod p$$

Now, they can compute secret share key by using partner's public key

$$K = (y_b)^{x_a} \bmod p = (y_a)^{x_b} \bmod p = g^{x_a * x_b} \bmod p$$

III. Contributory Group Key Schemes

For simplicity, the following notations are used in this section:

M_i ($1 \leq i \leq n$): Group members;

p: Large prime;

g: A primitive root or generator of $\mathbb{Z}_p^*$;

x: Denotes a random integer or secret exponent;

$<L, i>$: The ith node at level L in the binary key tree;

$K_{<L, i>}$: The secret key of the node $<L, i>$;

$bk_{<L, i>}$: Blinded key of the node $<L, i>$;

bk_i: Blinded key of the ith group member;

K_i: Secret key of the member M_i;

CRT: Chinese Remainder Theorem;

LCM: Least Common Multiple.

A. ING

The earliest attempt for extending two parties Diffie Hellman key exchange to a group has been done by Ingemarsson et al. (1982), and is called ING [12]. This protocol consists of (n-1) round and group members perform every round in synchronization. The member must be arranged in logical ring.

During the first round, each member M_i computes g^{x_i} and forwards it to its next neighbor or member M_{i+1}. In next rounds, each member raises the previously received intermediate key value to the power of its own exponent and forwards result to its logical neighbor. After (n-1) rounds, all participants possess a same key K_n.

B. GDH 1

GDH1 [13] consists of two stages: upflow and downflow. The purpose of upflow stage is to collect contribution from all group members. In round i, member M_i receives a collection of intermediate value from M_{i-1}. Member M_i raises the last value in received collection, i.e. $g^{x_1 \cdots x_{i-1}}$, to the power of its secret component x_i and computes $g^{x_1 \cdots x_i}$. After that member M_i appends new intermediate value to incoming flow and forwards it to next member M_{i+1}. The intended group key K_n will be produced when the highest numbered group member compute $g^{x_1 \cdots x_n}$.

In downflow stage, member M_i receives an ordered list of intermediate values from member M_{i+1}. It raises all values to the power of its secret component x_i. The last value in the set of computed intermediate values is K_n. Other values are sent to the member M_{i-1}.

C. GDH2

GDH2 generates shared key in n rounds [13], [14]. The protocol collects contribution from group members in upflow stage. In this stage, each member M_i receives a sequence of intermediate values and a cardinal value from M_{i-1}. The M_i raises these values to its secret component x_i and sends them *to* M_{i+1}.

In this protocol, the highest indexed group member M_n plays the role of group controller. It receives the some values and raises them to its secret component. It keeps a cardinal value as an actual group key and broadcasts other intermediate values to the entire group. Each member M_i recognizes its intermediate value from the receiving values and powers it by its x_i. As a result it computes the final group key.

When a new member is interested in joining to the group, the group controller M_n generates a new secret component x_n and creates a new upflow message by raising the contents of the previously received upflow message. It then sends the message to new member. The role of group controller is thus passed to new member of group.

In leave event, the member M_n generates a new secret component x_n. Now, the M_n compute a new set of n-2 sub keys (exclude sub key of leaving member M_p and M_n) and broadcast them to all members of group. Since, $g^{x_1 \cdots x_{p-1} x_{p+1} \cdots x_{n-1}}$ is missing from the set of broadcast sub keys, the departure member M_p is unable to compute new group key.

D. GDH3

The GDH3 [13] consist of four stages. In first stage, it collects the contribution from members $M_{1, \ldots,} M_{n-1}$ similar to GDH1. In the second stage, M_{n-1} broadcasts $g^{x_1 \cdots x_{n-1}}$ to all members of group. In next stage, each member factors out its exponent x_i from the receive value and sends the result to M_n. In the final stage, M_n powers the received values from group members with its exponent x_n and broadcasts the results to rest of members in the group. After receiving these values, each member can compute the group key.

Member M_n has to save the content of original broadcast message from M_{n+1} and response messages from other members (during stage2 and 3). In join event, member M_n generates a new secret exponent x_n and computes a new sub keys which will be forward to new member. New member M_{n+1} computes a new group key K_{n+1}. It also raises received sub keys to its own secret exponent x_{n+1} and broadcasts them to all members in the group. Now, the group members can perform the group key K_{n+1}.

Generation key in leave event is similar to GDH2.

E. Octopus Protocol

In this protocol [15], each large group split into 4 subgroups. Each subgroup performs its intermediary DH value ($I_{subgroup} = g^{x_1 \cdots x_n/4}$) and then subgroups exchange their intermediary DH value with each other. All group members are then able to generate the group key. The contribution from each subgroup members is collected by leader of each subgroup in order to perform intermediary DH value.

F. STR

Kim et.al has proposed STR [11], [16] protocol to extend Steer protocol to deal with dynamic group and network failure. They used a logical key hierarchy to minimize the number of key that held by group members.

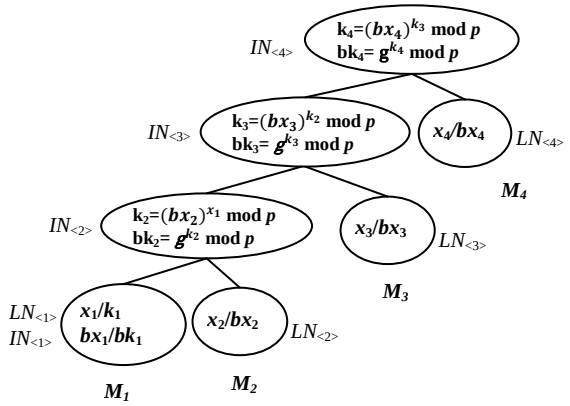

Fig. 1. Example of STR key tree

STR key tree has two type of node, Leaf and Internal. Each member group is associated with a leaf node. An internal node $IN_{<i>}$ always has two nodes, leaf node and another internal node $IN_{<i-1>}$. The internal node $IN_{<1>}$ is exception. It's also a leaf node.

Each leaf node $LN_{<i>}$ has a session secret key x_i which is kept secret by member M_i. The blinded (public) key bx_i of each leaf node calculate as $g^{x_i} \bmod p$. Every internal node has secret key $K_{<i>}$ and blinded key $bk_{<i>} = g^{k_i} \bmod p$. The internal node $IN_{<i>}$ computes its secret key K_i (i > 1) by using of Diffie Hellman key agreement of its two children. The K_i ($i > 1$) is generated recursively as follow:

$K_i = (bk_{i-1})^{x_i} \bmod p = (bk_i)^{k_{i-1}} \bmod p = g^{x_i k_{i-1}} \bmod p$ if $i > 1$.

For example in Fig. 1, $K_1 = x_1$, $K_2 = g^{k_1 x_2} \bmod p$, $K_3 = g^{k_2 x_3} \bmod p$ and $K_3 = g^{k_3 x_4} \bmod p$.

In join event, the new user broadcast his join request message that contains its own blinded key bk_{n+1}. The group's sponsor node M_n, top most node in tree, concurrently computes a blinded version of the current group key (bk_n) and sends the current tree $BT_{<n>}$ to new member with all blinded keys and blinded session random. The pre-existing members increase the number of node from n to $n + 1$. They also create a new tree so that the left node is prior tree and the right node is new member. Now, the current members can compute the group key when they receive the new member's blinded (public) key.

If member M_d decides to leave the group, the other group members upon hearing the leave event begin to update its key tree by deleting the node $LN_{<d>}$ corresponding to leaving node and its parent node $IN_{<d>}$. The nodes above the leaving member M_d are renumbered and the sibling of M_d would be replaced by M_d's parent. The sponsor M_s is a leaf node directly below the M_d (if $d > 1$). Otherwise the sponsor is M_2. The sponsor selects a new secret session random and computes all key up to the root and broadcast to $BT_{<s>}$. Now, all members can regenerate the new group key.

G. TGDH

In Tree based Group Diffie Hellman (TGDH) [10], [17], all members maintain a virtual binary tree. The nodes in tree are denoted by $<L, V>$, where $0 \le V \le 2^L - 1$ since each level L has at most 2^L nodes. V indicates the sequential index of node at level L in tree. Each parent node $<L, V>$ has two children, which are labeled as $<L+1, 2V>$ and $<L+1, 2V+1>$, respectively. The root of tree indicates by label <0, 0> as it's located at level 0.

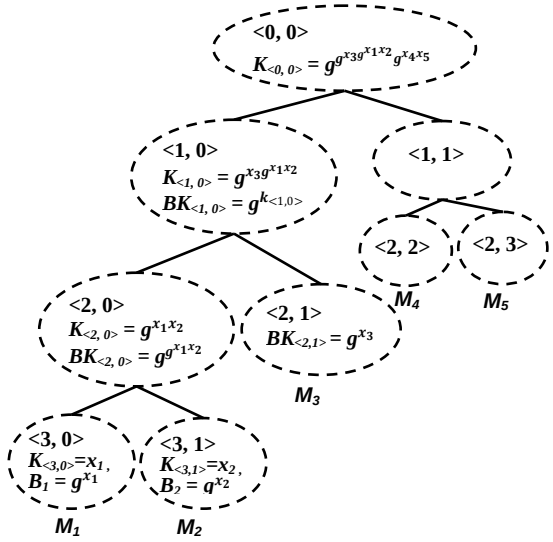

Fig. 2. Notation of TGDH

In this protocol, all group members are hosted on leaf nodes respectively. Each node in tree except root is associated to a secret key $K_{<L, V>}$, which is a Diffie Hellman private key, and a blinded (public) key $BK_{<L, V>} = f(K_{<L, V>})$, where $f(x) = g^x \bmod p$. The parent node computes its key $K_{<L, V>}$ by using private keys of its two children, i.e. $K_{<L, V>} = f(K_{<L+1, 2V>} K_{<L+1, 2V+1>})$. It means that each node's secret key can be computed from secret key of one of its two children and blinded key of other child by using the Diffie Hellman key exchange protocol. All blinded key are broadcasted to all members of group.

Each member M_i at $<L, V>$ knows the secret key $K_{<L, V>}$ along path from itself to the root of tree. $K_{<0,0>}$ at the root node is group secret key which is shared by all members in the group. Fig. 2 demonstrates a key tree with five members.

When a new user is interested in joining to group, it broadcasts its request message that contains its blinded key. All pre-existing members receive this request and determine the join point in the tree and its sponsor. The sponsor is rightmost leaf in the subtree rooted at the insertion node. Each member updates the key by adding a new node for new member, new intermediate node and removing all secret key and blinded key from sponsor's leaf to the root. Then, sponsor node computes all secret key and blinded key along path from its leaf node to the root. Now, sponsor node broadcast all new blinded key to all members. All members upon receiving new blinded key would be able to generate the new root key.

If one of existing member M_i decides to leave the group, each member updates its key tree by deleting key corresponding to M_i. the former sibling of M_i replace by M_i's parent. The sponsor generate new secret key and computes all key and blinded key from itself to the root. After generating key, sponsor broadcasts new blinded keys BK to all members, which allows all members to compute the new group key. Sponsor at this case is the shallowest rightmost leaf of the subtree rooted at M_i's sibling node.

H. CRTDH

In Chinese Remainder Theorem and Diffie Hellman

(CRDTH) scheme, all members exchange their contributed key share by using the Diffie Hellman key exchange mechanism, and then the members independently but mutually generate the group key based on the chinese remainder theorem (CRT) [18].

Each node computes its Deffie Hellman public key y_i, and broadcast it to all nodes. Next, Each node which receives others nodes public key, computes secret shared key with the sender of public key. After that each node selects a random key K_i which is less than all the received public keys in order to solve CRT value. The CRT value is broadcasted to other members. Each member calculates other members' random key by using received CRT value. Every member now extracts other members' random key from received CRT values, and then XOR them to make traffic key.

In join occurrence, one of closest member transmit hash value of traffic key as well as all group members' Diffie Hellman public key to new member. New member computes DH and CRT and broadcasts new CRT to all group members. The pre-existing users can calculate the received CRT and extract the new member key *K*. They XOR new *K* with current traffic key for providing new key.

In leave event from group, one of the remaining members selects a new key *K* and computes new CRT, and then broadcasts it to other members. Upon receiving new CRT, the new key *K* will be extracted from new CRT by other members. New traffic key will be obtained by XORing existing traffic key with new key *K*.

IV. Analysis and Comparison

This section summarizes and compares cost of communication of the contributory key management protocols presented in section III during group operation: initial, join and leave.

The initial operation is the initial creation of the group key and organization of the key management infrastructure.

Join operation is adding a new member to the group during group communication and participate in present and future communication.

Leave operation is used to remove a member from group either voluntary or enforcing expulsion and won't be able to decrypt next group communication messages.

Cost of communication contains number of rounds and number of exchange messages between members during group operation such as initial, join and leave. The exchange messages can include either unicast or broadcast messages.

We look at the number of round as a time unit that shows the number of steps taken in an operation.

The number of unicast message is sum of the messages every member sends to another single member in group.

The number of broadcast message is sum of all messages which is sent by a member to other members in a group for operation. This greatly affects the communication cost in compare with unicast.

The total number of message is sum of all unicast and broadcast messages which are sent in group during operation. This number is used to determine the total communication time.

Table I summarizes communication cost for the eight protocols which are investigated in section III. The numbers of current group member is denoted by *n*. The height of key tree constructed by TGDH is also denoted by *h* that is equal$\log_2 n$.

ING, GDH1 were originally designed to support only group formation. They cannot support join or leave operation in group. These protocols have to restart anew upon every changing in group membership. Octopus was designed to exchange key without using broadcast message. We don't consider join and leave operation in this protocol.

From Table I, the number of rounds of all protocols except CRTDH in initial operation is relative to number of members *n* in a group. GDH1 in compare with other protocols has the biggest number of rounds with 2*n*-2. This means that GDH1 needs to take 2*n*-2 steps and consume more time to perform group key. The CRTDH has a fixed round number, which means that the number of interaction between members is independent from the number of group members.

In term of number of exchange messages, ING with *n*-1 messages in each round is the most expensive protocol. However, STR and TGDH exploit 2(*n*-1) broadcast messages. As every broadcast message is sent to all members of group, It makes sense that each broadcast message will consume more bandwidth than a given unicast message.

TABLE I: Comparison of Contributory Key Agreement Scheme

Scheme		No round	Message		
			Unicast	Broadcast	Total
ING [12]	Initial	n-1	*n*(*n*-1)	0	*n*(*n*-1)
GDH1 [13]	Initial	2(*n*-1)	2(*n*-1)	0	2(*n*-1)
GDH2 [13-14]	Initial	*n*	*n*-1	1	*n*
	Join	2	1	1	2
	Leave	1	0	1	1
GDH3 [13]	Initial	*n*+1	2*n*-3	2	2*n*-1
	Join	2	1	1	2
	Leave	1	0	1	1
Octopus [15]	Initial	2(*n*-4)/4 + 2	3*n* - 4	0	3*n* - 4
STR [11, 16]	Initial	*n*-1	0	2(*n*-1)	2(*n*-1)
	Join	2	0	3	3
	Leave	1	0	1	1
TGDH [10, 17]	Initial	*h*	0	2(*n*-1)	2(*n*-1)
	Join	2	0	3	3
	Leave	1	0	1	1
CRTDH [18]	Initial	2	2*n*-2	2	2*n*
	Join	2	1	1	2
	Leave	1	0	1	1

In join operation, all protocols require two rounds. All of them use a constant number of messages, which are independent from number of members in the group. TGDH and STR employ three broadcast messages that in contrast to other protocols probably consume more bandwidth.

All protocols have the same communication cost in leave event. They have one round consisting one message.

V. Conclusion

This paper has presented a review of group key management, particularly contributory schemes which

employ public key exchange algorithm. It looked at the initial operation when a number of members of a group are going to generate a shared traffic key. Meanwhile, the used method for provision of backward secrecy after joining a new member to the group as well as forward secrecy after leaving a pre-existing member from the group have been investigated. It also provided a qualitative evaluation of the presented approach. The investigated schemes then summarized and analyzed in term of communication cost. In the initial operation, communication requirement of all contributory schemes (except CRTDH) are raised by increasing number of members. Otherwise, they are independent from number of members in join or leave operation.

REFERENCES

[1] C. K. Wong *et al.*, "Secure group communications using key graphs," *SIGCOMM Comput. Commun. Rev.*, vol. 28, pp. 68-79, 1998.

[2] D. Wallner *et al.*, "Key Management for Multicast: Issues and Architectures. RFC 2627," *Internet Engineering Task Force*, 1999.

[3] H. Harney and C. Muckenhirn, "Group Key Management Protocol (GKMP) Specification. RFC 2093," *Internet Engineering Task Force*, 1997.

[4] H. Harney and C. Muckenhirn, "Group Key Management Protocol (GKMP) Architecture. RFC 2094.," *Internet Engineering Task Force*, 1997.

[5] M. Baugher *et al.*, "Multicast Security (MSEC) Group Key Management Architecture. RFC 4046," *Internet Engineering Task Force*, 2005.

[6] S. Mittra, "Iolus: a framework for scalable secure multicasting," *SIGCOMM Comput. Commun. Rev.*, vol. 27, pp. 277-288, 1997.

[7] T. Hardjono *et al.* (2000, *Intra-Domain Group Key Management Protocol.* [Online]. Available: http://www.ietf.org/proceedings/98dec/I-D/draft-ietf-ipsec-intragkm-00.txt

[8] L. R. Dondeti *et al.*, "A Dual Encryption Protocol for Scalable Secure Multicasting," in *Proc. IEEE International Symposium on Computers and Communications*, 1999, pp. 2-8.

[9] M. L. M. Kiah and K. M. Martin, "Host Mobility Protocol for Secure Group Communication in Wireless Mobile Environments," in *Proceedings of the Future Generation Communication and Networking* 2007.

[10] Y. Kim *et al.*, "Tree-based group key agreement," *ACM Trans. Inf. Syst. Secur.*, vol. 7, pp. 60-96, 2004.

[11] Y. Kim *et al.*, "Group Key Agreement Efficient in Communication," *IEEE Trans. Comput.*, vol. 53, pp. 905-921, 2004.

[12] I. Ingemarsson *et al.*, "A conference key distribution system," *Information Theory, IEEE Transactions on*, vol. 28, pp. 714-720, 1982.

[13] M. Steiner *et al.*, "Diffie-Hellman key distribution extended to group communication," in *Proceedings of the 3rd ACM conference on Computer and communications security*, New Delhi, India, 1996, pp. 31-37.

[14] M. Steiner *et al.*, "CLIQUES: A New Approach to Group Key Agreement," in *Proceedings of the The 18th International Conference on Distributed Computing Systems*, Amsterdam, Netherlands, 1998, pp. 380-387.

[15] K. Becker and U. Wille, "Communication complexity of group key distribution," in *Proceedings of the 5th ACM Conference on Computer and Communications Security*, San Francisco, California, United States, 1998.

[16] Y. Kim *et al.*, "Communication-efficient group key agreement," in *Proceedings of the 16th International Conference on Information Security: Trusted Information: The New Decade Challenge*, Paris, France, 2001, pp. 229 - 244.

[17] Y. Kim *et al.*, "Simple and fault-tolerant key agreement for dynamic collaborative groups," in *Proceedings of the 7th ACM Conference on Computer and Communications Security*, Athens, Greece, 2000, pp. 235 - 244.

[18] R. K. Balachandran *et al.*, "CRTDH: an efficient key agreement scheme for secure group communications in wireless ad hoc networks," presented at the Communications, 2005. 2005 IEEE International Conference on, 2005.

[19] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. 22, pp. 644-654, 1976.

Babak Daghighi received his B.Sc. in Computer Science from Azad University (Tehran branch), Iran, and M.Sc. from University of Malaya (UM), Malaysia. He is now continuing his Ph.D studies in Computer Science in University of Malaya. His research interests include cryptography, network security, and wireless system with an emphasis on group key management in wired and wireless networks.

Laiha Mat Kiah received her B.Sc. (Hons) in Computer Science from University of Malaya (UM), Malaysia in 1997, M.Sc. in 1998 and Ph.D. in 2007 from Royal Holloway, University of London UK. She is currently the Deputy Dean of Faculty of Computer Science and Information Technology, University of Malaya. Her current research interests include wireless/mobile/cloud security, secure group communication and key management. She is also interested in routing protocols, ad-hoc networks and secure algorithms.

Saeedeh Afkhami Rad received her B.Sc. in Computer Science from Azad University (Mashad branch), Iran. She is now continuing her master studies in Computer Science in Shahid Beheshti University, Iran. Her research interests include database.