

SECURITY AND PERFORMANCE OF GROUP KEY AGREEMENT PROTOCOLS*

Shanyu Zheng, David Manz, and Jim Alves-Foss
Center for Secure and Dependable Systems
University of Idaho
Moscow, ID 83844, U.S.A.
email: [zhen8299, manz6595, jimaf]@uidaho.edu

Yuzhe Chen
College of Mathematics and Information Science
Hebei Normal University
Shijiazhuang, 050016, PRC
email: hebtu_cyz@hotmail.com

ABSTRACT Several group key protocols have been presented in the literature to enable secrecy of communication among dynamic groups of participants. However, there is a lack of consistency in the literature in terms of operations supported and performance metrics of the various protocols. This makes it difficult for designers to choose the best protocol for their specific applications. To alleviate this problem we introduce a generic model of group key exchange protocols which we use to prove security properties of the protocols. In addition, we introduce new performance metrics to be used to assist in uniformly comparing these protocols and then improve the performance of five protocols based on these operations and metrics.

KEY WORDS

Group Key Agreement, Diffie-Hellman, Public Key

1 Introduction

With increased interdependence and networking of secure communication, there are numerous applications such as voice conferences, distributed computation, unmanned aerial vehicle coordination that would greatly benefit from an extended group key exchange algorithm that provides a secure session key to encipher the communication and protect confidential information for groups of participants. To provide assurance of secure and reliable communication between multiple groups, There are three categories that have been put forward to solve group key establishment: centralized, distributed, and contributory group key management [10, 2, 13] for large and dynamic groups based on the Diffie-Hellman (DH) key exchange algorithm [5].

In this paper, we focus on the contributory group key agreement protocols, the standard operations they perform, the security of the protocols and performance metrics of the protocols. In this category, every group member equally contributes to the key, so the group key is derived from all the group members. This solves the problem of a single

point of failure and trust in a centralized group key management algorithm; providing for a higher level of assurance in the continuing correct operations of the system. Throughout this paper we will address several group key management operations used by group key systems. The operations we describe are initialization, join, mass join, leave, mass leave, merge, partition, and key refresh.

The remainder of this paper is organized as follows. In Section 2 we briefly introduce the key contributions of this paper. This includes the presentation of our General Contributory Group Key Exchange algorithm (GCGKE), the set of new evaluation metrics for group-key, the cryptographic security properties supported by this approach, and a description of our proposed centralized authentication algorithm. In Section 3 we prove that extending the general secure two-party group key exchange protocol to multi-party is still secure and discuss some well-known secure two-party group key exchange protocols. In Section 4 we compare the efficiency of several notable two-party group key agreement protocols and public key algorithms. In Section 5 we evaluate the efficiency of five efficient contributory group key agreements, and further improve their performance. In Section 6 we conclude and summarize our contributions.

2 Contributory Group Key Exchange Protocol

Our General Contributory Group Key Exchange protocol (GCGKE) requires that every group member equally to contribute to the group's secret key, computed as a function of all members' contributions. This is suitable for dynamic peer groups and provides strong security properties, since each member contributes to the entropy of the key.

2.1 The GCGKE Algorithm

In the group-key protocol literature the authors explicitly describe and explain their specific algorithms for some of the operations mentioned above [10, 7, 6, 13]. Rather than going into detail for each algorithm, we emphasize a general algorithm GCGKE for these eight operations. GCGKE is described as follows:

*This material is partially supported by AFRL and DARPA under agreement number F30602-02-1-0178. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of AFRL and DARPA or the U.S. Government.

GCGKE

Step 1: Every group member of every entity (we treat a group or a group member as an abstract entity) generates its secret key K_i and corresponding public value PK_i .

Step 2: The group sponsor of every entity transmits a message including its public value to its sibling entity (if an entity has no sibling entity, the entity does nothing in that round). The two sibling entities now execute a two-party group key exchange forming a new composite entity. All members in the entity can calculate the group key and the corresponding public value. Then they choose a group member as the group sponsor in the new entity.

Step 3: Repeat step 2 until there is only one entity. Now every group member can know the shared group key.

From the above GCGKE, treating a group as an abstract entity, we expand the traditional two-party key exchange algorithms into the group setting. With this new point of view we build a new group, add members and merge groups through an iterative process using two-party exchange as a primitive operation. For example, in the initialization operation, current contributory group key agreements such as Efficient Group Key Protocol (EGK) [2], Tree-Based Group Diffie-Hellman (TGDH) [7], Steer et al.(STR) [6], and Communication-Computation Efficient Group Key (CEGK) [13] treat a group member as an entity in the first round and each group member begins the algorithm using two party key exchange. After that, each entity executes a two-party key exchange with its sibling group to create a new group and compute the group key. The process is repeated until there is only one group. However, before we can use GCGKE to provide reliable and secure communication, we must prove that this form of extending two-party secure group key exchange to multiparty is secure. We present our proof of this extension for three different two-party key exchange algorithms in Section 3.1.

2.2 Evaluation Metrics

Several evaluation metrics that are used to evaluate the performance of the contributory key agreements are discussed in current literature [3, 13]. The performance of the contributory key agreement can be divided into two categories: computation cost and communication cost. Communication costs include the *total number of rounds*, and *total number of messages* (both unicast and broadcast messages). Computation costs consist of the *total number of cryptographic signatures*, *total sequential exponentiations*, and *total number of verifications*.

In the group key protocols it is often necessary for

siblings or groups to have to wait for the results of a sibling's or group's cryptographic operation before conducting their own. The sequencing of these operations are denoted by the *number of sequential exponentiations*. It is noted that several of the published results are based on two-party DH [10, 2, 7, 6, 13]¹ which uses exponentiation as its most computational expensive cryptographic operation, hence the term sequential exponentiation. A more accurate term may be sequential keying operation, O_p in the general secure two-party group key exchange, so we treat total sequential O_p s as a parameter of the computation cost. Moreover, we introduce three new evaluation metrics to evaluate the performance of the contributory group key agreement protocols: *message size*, the execution time of every O_p , and the *execution time of signing/verification* in every round. We use these metrics in Section 4 when we discuss improving performance of the algorithms.

When the size of the message being transmitted is large, the network may fragment the message, which will cause network congestion, increasing the cost of communication. In addition, data transmission volume is of key concern for battery-powered mobile devices. Kim et. al. [7, 6] put forward that in every algorithm, in every operation, every group sponsor broadcasts all the blinded keys (not just one key) to the other group members, which causes the message size to become quite large. This large message size is largely ignored when they evaluate communication costs.

During an O_p there will be a series of computationally expensive cryptographic operations. When the execution time of every O_p is slow, the computation cost becomes expensive. Note that the execution time of every O_p is related to the efficiency of the two-party group key exchange protocols, as discussed in Section 4.

In order to transmit the messages securely, we should implement an authentication algorithm in the contributory group key agreement. If the execution time of signing/verification in every round is slow, the computation cost becomes expensive. Note that the time of signing/verification in every round is related to the efficiency of public key algorithms that can be used for signing/verification.

In the literature, researchers improve the costs of computation and communication by putting forward different algorithms for every operation and using different structures such as linked lists and binary trees [10, 2, 7, 6, 13]. They also put forward a centralized authentication scheme and use RSA to sign/verify the messages. In this paper, the focus is on finding a more efficient two-party group key protocol to replace DH rather than detailing the specified algorithms for every operation. Additionally we explore the use of more efficient public-key algorithms for signing/verification to take the place of RSA to furthermore improve the performance of the general contributory group

¹Although both Alves-Foss [2] and the authors [13] briefly mention that any two party group key algorithm including Elliptic curves can be used, they only use DH group key exchange in their papers.

key agreement.

2.3 Cryptographic Properties

There are some desired properties of any contributory group key agreement protocol such as group key secrecy, weak backward secrecy, backward secrecy, weak forward secrecy, forward secrecy, perfect forward secrecy, and key independence discussed in the literature [10, 7, 6, 13]. Group key secrecy is that any group key is indistinguishable in polynomial time from a random number. This is a group key variant of the Decision Diffie-Hellman problem [4] that we discuss in Section 3. It is related to the security of multi-party group key exchange. If multi-party group key exchange is secure, then contributory group key agreement based on it guarantees group key secrecy. Forward secrecy, backward secrecy, key independency and perfect forward and backward secrecy are guaranteed by the specified algorithms of specified operations. In this paper, we only focus on how to provide group key secrecy when we extend the secure two-party group key exchange to multi-party. We prove whether extending the secure two-party group key exchange to multi-party is secure. If it is true, we know we can use the most efficient two-party group key exchange to improve the performance of contributory group key agreement (see Section 2.2). Note that the difference between our paper and [10, 2, 7, 6] is that they only prove extending two-party DH to multi-party is still secure. Our paper will prove extending two-party secure group key exchange based on the assumption is equivalent to Decisional Diffie-Hellman (DDH) or Elliptic Curve Decisional Diffie-Hellman (ECDDH) assumption to multi-party is still secure (See the definitions of DDH and ECDDH in Section 3).

2.4 Centralized Authentication Algorithm

Kim et. al. [7, 6, 13] didn't discuss the centralized authentication scheme in detail and they use RSA to sign/verify the messages. In this paper we present a general public key algorithm and general hash function to implement the centralized authentication algorithm:

1. The group sponsor enciphers the original message using a secure hash function. The output is a fixed size. The output is enciphered text with the private key of the group sponsor. It, plus the original message, is broadcast to all other group members.
2. The other group members receive the message, they use the public key of the group sponsor to decipher the output. They also use the hash function of the original message to compute the result and compare it with the output. If they are same, the message is verified; otherwise, the message is rejected.

From the above algorithm, we know the execution time of signing/verification in every round depends on the ef-

iciency of the public key algorithm. In order to improve the execution time of signing/verification in every round we should choose the most efficient public key algorithm. In Section 4.2 we compare the efficiency of some well-known public key algorithms.

3 General Secure Contributory Group Key Exchange Protocol

Before we discuss the general secure contributory group key exchange, we give some definitions used in this paper.

Definition 1 *Decisional Diffie-Hellman (DDH) Assumption:* G is a finite cyclic group and g is a generator. Given (g^a, g^b, g^{ab}) and (g^a, g^b, g^c) for random $a, b, c \in [1, |G|]$, no efficient algorithm can decide that $c = ab$ in G . In the other word, the value g^{ab} is indistinguishable in polynomial time from a random number of G .

Definition 2 *Elliptic Curve Decisional Diffie-Hellman (ECDDH) Assumption:* E is an elliptic curve over a finite field \mathbb{F}_q and P is a generator point on E . Given P, aP, bP, cP for random a, b, c , no efficient algorithm can decide that $c = ab$.

A two-party group key exchange protocol is used when two participants generate their own private value and a corresponding public value, exchange their public values, and use key exchange operations to create the same new shared keys independently. Given the public value and some other information, it is infeasible to learn the private key in polynomial time [4]. There are some secure two-party group key exchange protocols such as DH and MQV, based on discrete logarithms, and Elliptic Curves over a finite field that support the DDH assumption, or its equivalent. Our General Contributory Group Key Exchange algorithm extends two-party secure group key exchange to multi-party. This leads us to ask if the key generated from multi-party group key exchange is as secure. In Section 3.1 we will prove that extending two-party group key exchange to multi-party is still secure.

3.1 Security of Extending Two-Party Group Key Exchange Protocols

Assumption 1: Using the DDH Assumption, let K_1 and K_2 be random numbers, and PK_i be the corresponding public values. The shared value $K = PK_1 \text{ Op } K_2 = PK_2 \text{ Op } K_1$ using two-party secure group key exchange is indistinguishable in polynomial time from a random value.

Assumption 2: Using the ECDDH Assumption, let K_1 and K_2 be random numbers, and PK_i be the corresponding public values. The shared value $K = PK_1 \text{ Op } K_2 = PK_2 \text{ Op } K_1$ using two-party secure group key exchange is indistinguishable in polynomial time from a random value.

They are very strong assumptions. These assumptions lead to many useful cryptographic operations, including many of the group key agreement protocols found in the literature. The interested reader is referred to Boneh's summary of this problem [4]. Assumptions assuming the general two-party group key exchange is indistinguishable in polynomial time from a random value and the operation in group key exchange is performed by Op , is the generated group key secure?

Theorem 1 *Let G be a group of size $n = 2^m$ and let K_1, K_2, \dots, K_n be random values. The shared group key K derived in the application of GCGKE using the values K_1, K_2, \dots, K_n is indistinguishable in polynomial time from a random value.*

Proof. The proof is based on mathematical induction over m .

Basis: $m = 1$. This is the general two-party group key exchange protocol, which is assumed to be indistinguishable in polynomial time from a random number.

Induction Hypothesis: The shared group key K derived in the application of GCGKE, for any group of size $2^m (m \geq 1)$ is indistinguishable in polynomial time from a random number.

Induction Step: GCGKE creates a group size $2^{(m+1)}$ by merging two subgroups of size 2^m . Suppose that K_1 and K_2 are the shared subgroup keys of their respective subgroups. GCGKE uses these shared subgroup keys to create their own public value PK_i , and then executes the two-party group key exchange protocol to compute the new group shared key $K = PK_1 \text{Op} K_2 = PK_2 \text{Op} K_1$ independently. From the *induction hypothesis*, K_1 and K_2 are indistinguishable in polynomial time from random values. From the **Assumption 1 or Assumption 2**, we know the new shared group key K is indistinguishable from a random number.

Theorem 1 directly derives the corollary that the key created from the merge of two groups (where group size is a power of 2) is secure.

Corollary 1 *Let G_1 and G_2 be two groups of size $n_1 = 2^{m_1}$ and $n_2 = 2^{m_2}$ respectively. G_1 and G_2 have their own secure shared group keys K_1 and K_2 . The new shared group value K derived from merging two subgroups using GCGKE is indistinguishable in polynomial time from a random value.*

Proof. G_1 and G_2 generate their own public value PK_1 and PK_2 respectively. Then they execute the two-party group key exchange protocol to compute the new shared group key $K = PK_1 \text{Op} K_2 = PK_2 \text{Op} K_1$ independently. Because K_1 and K_2 is indistinguishable in polynomial time from random values and by the assumption, the value K is indistinguishable from a random number.

In general the group size n does not have to be a power of 2. We can break down any size group into a collection of the groups whose size is the power of 2 and merge them.

Theorem 2 *Let G be a group of size n and let K_1, K_2, \dots, K_n be random values. The shared group key K derived from the application of GCGKE using initial values K_1, K_2, \dots, K_n is indistinguishable in polynomial time from a random value.*

Proof. This proof is based on the binary decomposition of the size of the group.

Case 1: if $n = 2^m$ for some integer m , by **Theorem 1** the conclusion holds.

Case 2: if $n \neq 2^m$ for any integer m , we can always find the values $a_i \in [0, 1]$ that is satisfied with the equation $n = a_m 2^m + a_{m-1} 2^{m-1} + \dots + a_1 2^1 + a_0 2^0$. We treat every subgroup in which a_i is equal to 1 as an entity. From **Theorem 1** we know every entity has a secure shared key K_i . We continue to collapse the number of groups by pairing two subgroups (siblings in the tree) and having them execute the two-party group key exchange protocols to create a new subgroup and generate a new secure shared key by **Corollary 1** until there is only one group. The final new secure shared key is K . So K is indistinguishable in polynomial time from a random value.

After we extend the general two-party group key exchange to multi-party and prove the security of multi-party group key exchange protocol, we discuss three secure popular two-party group key exchange protocols and prove the security of their extending to multi-party.

3.2 Diffie-Hellman (DH) Key Exchange

The DH Key Exchange with indistinguishable security is base on DDH assumption. It is presented as follows: Let g, p and q be publicly known values, where p, q are large primes and where $p = 2q + 1$. g is a generator of the group \mathbb{QR}_p that is a subgroup of \mathbb{Z}_p^* and the order is $|\mathbb{QR}_p| = q$. Let Alice and Bob choose secret private value $K_A \in \mathbb{Z}_q$ and $K_B \in \mathbb{Z}_q$, respectively. They create their corresponding public values (**blinded keys**), r_A and r_B , and send them to each other.

$$\begin{aligned} \text{Alice} \rightarrow \text{Bob} & : r_A = g^{K_A} \text{ mod } p \\ \text{Bob} \rightarrow \text{Alice} & : r_B = g^{K_B} \text{ mod } p \end{aligned}$$

Bob and Alice then compute a shared secret key using their partner's public value and their own private value:

$$K = (r_A)^{K_B} \text{ mod } p = (r_B)^{K_A} \text{ mod } p = g^{K_A * K_B} \text{ mod } p$$

Corollary 2 *Let G be a group of size n and let K_1, K_2, \dots, K_n be random values in \mathbb{Z}_p^* . The shared group key, K , derived from the application of GCGKE based on two-party DH Key Exchange with indistinguishable security using initial values K_1, K_2, \dots, K_n is indistinguishable in polynomial time from a random value in \mathbb{Z}_p^* .*

Proof. Because two-party DH key exchange with indistinguishable has power operation that is commutative and it is indistinguishable in polynomial time from a random value. By **Theorem 2**, we can directly prove that multi-party DH group key exchange is secure.

3.3 MQV Key Exchange

MQV is based on DH and improves DH. It derives a shared secret value from one participant's two key pairs and another participant's two key pairs. As a derivative of DH it supports the DDH assumption. The algorithm is as follows [9]:

1. Suppose a key exchange between Alice and Bob. Alice possesses a key pairs (PK_A, K_A) with PK_A being her public key and K_A being her private key. Similarly, Bob possesses a key pair (PK_B, K_B) with PK_B being his public and K_B being his private key.
2. Alice generates a session key pair (X, x) by randomly generating x and calculating $X = x * P$ where x is an integer and X is a point on an elliptic curve and P is the generating point. Similarly, Bob generates a session key pair (Y, y) .
3. Alice calculates $S_A = (x + \overline{X} * K_A) \bmod n$ and Bob calculates $S_B = (y + \overline{Y} * K_B) \bmod n$. In here n is the order of P , \overline{X} (or \overline{Y}) represents the first L bits of the point X (or Y) where $L = \lceil \frac{\lceil \log_2 n \rceil + 1}{2} \rceil$.
4. Alice and Bob send their public key PK_A and PK_B to each other. Alice and Bob can compute the shared group key $K = h * S_A * (y + \overline{Y} * PK_B) = h * S_B * (x + \overline{X} * PK_A)$ where h is the co-factor defined in P1363.

Corollary 3 *Let G be a group of size n and let K_1, K_2, \dots, K_n be random values. The shared group key, K derived from the application of GCGKE based on two-party MQV key exchange protocol using initial values K_1, K_2, \dots, K_n , is indistinguishable in polynomial time from a random value.*

Proof. We treat $(y + \overline{Y} * PK_B)$ as the public value of B , S_A as the private value of A , $(x + \overline{X} * PK_A)$ as the public value of A , and S_B as the private value of B , so the operation in two-party MQV key exchange protocol is commutative. Because two-party MQV key exchange protocol is secure, By **Theorem 2**, we can directly prove multi-party MQV key exchange protocol is secure.

3.4 Elliptic Curve Key Exchange

The Elliptic curve key exchange algorithm is as follows [8, 1]:

1. Pick a prime number p , two nonnegative integers a and b that satisfy $(4a^2 + 27b^2) \bmod p \neq 0$, and elliptic curve $E_p(a, b)$ whose elements (a, b) are satisfied with $y^2 \doteq (x^3 + ax + b) \bmod q$, a generator point $G(x_1, y_1)$ in $E_p(a, b)$.
2. Suppose a key exchange between two participants A and B . A selects a private key K_A less than n . Then

A calculates its public value $P_A \doteq K_A * G$. P_A is a point in $E_p(a, b)$. B selects its secret key K_B less than n . Then B generates its public value $P_B = K_B * G$. P_B is a point in $E_p(a, b)$.

3. A and B send their own public values to each other, respectively. After A and B receive the other's public value, they can calculate the shared key K . The shared key $K = K_A * P_B = K_B * P_A = K_A * K_B * G$

In the two-party elliptic curve key exchange, even if an attacker knows G and KG , it is still infeasible to compute K .

Corollary 4 *Let G be a group of size n and let K_1, K_2, \dots, K_n be random values. The shared group key K derived from the application of GCGKE based on two-party elliptic curve key exchange protocol using initial values K_1, K_2, \dots, K_n is indistinguishable in polynomial time from a random value.*

Proof. Because the operation in two-party Elliptic curve key exchange protocol uses multiplication, it is commutative. Additionally, two-party Elliptic curve key exchange protocol is secure. By **Theorem 2**, we can directly prove that multi-party elliptic curve key exchange protocol is secure.

4 Performance Improvement of CGKE Protocols

As we have stated before, the published group-key protocols are typically based on DH. However, as we have shown in Section 3, if we replace DH with the more secure and efficient two-party group key exchange in the contributory group key, the contributory group key agreement will still provide group key secrecy. From Section 2.2 we know that replacing the older two-party group key exchange with the new more efficient two-party group key exchange can improve the performance of the contributory group key agreement. Moreover, we know that using the new more efficient public key algorithms for signing/verification in place of older algorithms can improve the efficiency of the contributory group key agreement. In the next sections, we compare the efficiency of secure two-party group key exchange protocols, and public key algorithms for signing/verification.

4.1 Efficiency of Secure Two-Party Group Key Exchange Protocols

There are several secure two-party group key agreement protocols that have been published. In this section we discuss the most well-known examples DH, MQV, Elliptic Curve (ECDH), and Elliptic Curve MQV (ECMQV) [5, 9]. According to Adams and Lloyd [1], these protocols can be compared when we adjust the key length to obtain an equivalent level of cryptographic protection. Table 1 summarizes

these results; where speed indicates the relative computation cost of executing the given algorithm for the specified key size.

Group Key Exchange	Key Length (bits)	Speed
DH	1024	Slow
MQV	1024	Slow
ECDH	192	Faster
ECMQV	192	Fast

Table 1: Efficiency of Two-Party Group Key Exchange Protocols

The key size of DH, MQV should be 1024 bits long to ensure adequate security while the key size of ECDH, ECMQV only needs 192 bits to provide the same level of security and execute faster for these key sizes. It is apparent that ECDH is overall the most efficient among these four group key agreements. From Section 3, we know these four group key exchange protocols can be extended to multi-party secure group key agreements. Contributory group key agreements based on them guarantee the group key secrecy, so contributory group key protocols using ECDH can improve the time of every \mathcal{O}_P and improve the computation cost.

4.2 Efficiency of Public Key Algorithms for Signing/Verification

To provide for authentication in key agreement for dynamic groups, the CGKE protocols need to implement some form of authentication. For dynamic groups, this authentication is often public-key based. There have are several public key algorithms published. We summarize three of the more common ones RSA, DSA, and ECDSA [1]. In Table 2 we summarize the key length and implementation speed of these three algorithms as documented in the literature [1]. The key length is chosen to provide comparable levels of cryptographic strength in the signatures.

Public Key Algorithms	Key Length(bits)	Speed
RSA	1024	Slow
DSA	1024	Slow
ECDSA	192	Fast

Table 2: Comparison of Efficiency of Public Key Algorithms

The key size of RSA, and DSA should be 1024 bits long to provide adequate security while the key size of ECDSA only needs to be 192 bits to provide the same security. Therefore, ECDSA needs a much smaller keys than RSA, this is beneficial if transmission of each byte is a concern to system designers. ECDSA is the most efficient among these three public key algorithms. In Section 2.2, we discussed that Signing/Verification used in the centralized authentication of contributory group key protocols can

be improved by using ECDSA in every round as well as improving the total computation cost. Although we propose the use of ECDSA in this paper, any new developments and improvements in public key encryption can and should be used by GCEGE.

5 Improve Performance of Five Contributory Group Key Agreement Protocols

Table 3 summarizes the cost of communication and computation for five protocols in the worst case situation. Because Averages are based on specific network behavior and are documented in [11, 12], this is not as easy as it first appears. This paper followed the literature uses worst case analysis. In one or two instances we found some discrepancies or inconsistencies in the reported results and have corrected them here. n , K_a , K_p , K_m , m , and P_i are denoted the sizes of current group members, mass join members, mass leave members, merging groups, merging members, and leaving members for subgroup G_i , respectively. We use h and h_i to denote the height of the updating key tree and the height of the key tree of the i_{th} subgroup, respectively. $h'_a = \lceil \log_2 K_a \rceil$, $h'_m = \lceil \log_2 K_m \rceil$, $\rho = \min(\lceil \log_2 K_p \rceil + 1, h)$, $\mu = \min(2K_p, n - K_p)$, $\varphi = \max(\min(\lceil \log_2 P_i \rceil + 1, h_i))$, and $\omega = \max(\min(2P_i, n - P_i))$.

From Table 3, it is hard to determine which protocol is better, because we do not know what would happen when we execute multiple operations together. In [11, 12] We have already compared the average costs of multiple instances of these operations among these five protocols. In summary, the average phases and messages of CCEGK and STR are the most efficient, followed by TGDH, GDH3.0, EGK. The average sequential exponentiations of EGK is the most efficient, followed by TGDH, CCEGK, GDH3.0 and STR [11, 12].

Because these five contributory group key agreements are based on DH and use RSA to sign/verify message we can further improve the performance, based on our results from Sections 2 and 3. We keep the algorithms of seven operations in contributory group key agreements unchanged, but replace two-party DH with ECCDH, and use ECCDSA to sign/verify the messages, we can further improve their communication and computation costs. Our goal is to give a direction to improve efficiency of the General Contributory Group Key Exchange algorithm (which would reflect any and all other group key algorithms). For example, if there is a new, more efficient, contributory group key algorithm, a new secure two-party group key protocol based on the assumption that is equivalent to DDH or ECDDH assumption, and a new more efficient public key algorithm that can be used for signing/verification, we can create the contributory group key based on the new secure two-party group key protocol using the new efficient public key algorithm to authenticate the messages. The usefulness of GCGKE is that it can be modularly upgraded and used

Protocols	Communication Costs						Computation Costs		
	Rounds	Messages	Unicast	Broadcast	Message size	Sequential Exponentiations	Signatures.	Verifications.	
CCEGK	Initialize	h	2n-2	n	n-2	2n-2	2h-2	h	2n-2
	Join	1	2	1	1	1	1	1	2
	Mass Join	1	$K_a + 1$	0	$K_a + 1$	$K_a + 1$	K_a	1	$K_a + 1$
	Merge	1	K_m	0	K_m	K_m	$K_m - 2$	1	K_m
	Leave	1	1	0	1	$h - 1$	$3h - 3$	1	1
	Mass Leave	ρ	μ	0	μ	$\rho * h - \frac{(1+\rho)*\rho}{2}$	3h-3	ρ	μ
	Partition	φ	ω	0	ω	$\varphi * h - \frac{(1+\varphi)*\varphi}{2}$	$\max(3h_i) - 3$	φ	ω
EGK	Initialize	h	2n-2	0	2n-2	2n-2	2h-2	h	2n-2
	Join	1	2	0	2	2	1	1	2
	Mass Join	$h'+1$	$2K_a$	0	$2K_a$	$2K_a$	$2h'_a$	$h'_a + 1$	$2K_a$
	Merge	K_m	$2K_m - 2$	0	$2K_m - 2$	$2K_m - 2$	$2K_m$	K_m	$2K_m - 2$
	Leave	h	$2(n-1)$	0	$2(n-1)$	$2n-1$	2h	h	$2(n-1)$
	Mass Leave	h	$2(n - K_p)$	0	$2(n - K_p)$	$2(n - K_p)$	2h	h	$2(n - K_p)$
	Partition	$\max(h_i)$	$\max 2(n - P_i)$	0	$\max 2(n - P_i)$	$\max 2(n - P_i)$	$\max(2h_i)$	$\max(h_i)$	$\max 2(n - P_i)$
TGDH	Initialize	h	2n-2	0	2n-2	2n-2	2h-2	h	2n-2
	Join	2	3	0	3	$h + 1$	3h-3	2	3
	Mass Join	$h'_a + 1$	$2K_a$	0	$2K_a$	$(h'_a + 1) * h - \frac{(h'_a + 2)(h'_a + 1)}{2}$	3h-3	$h'_a + 1$	$2K_a$
	Merge	$h'_m + 1$	$2K_m$	0	$2K_m$	$(h'_m + 1) * h - \frac{(h'_m + 2)(h'_m + 1)}{2}$	3h-3	$h'_m + 1$	$2K_m$
	Leave	1	1	0	1	$h - 1$	3h-3	1	1
	Mass Leave	ρ	μ	0	μ	$\rho * h - \frac{(1+\rho)*\rho}{2}$	3h-3	ρ	μ
	Partition	φ	ω	0	ω	$\varphi * h - \frac{(1+\varphi)*\varphi}{2}$	$\max(3h_i) - 3$	φ	ω
STR	Initialize	n-1	2n-2	0	2n-2	2n-2	$2(n-1)$	n-1	2n
	Join	2	3	0	3	3	4	2	3
	Mass Join	2	$K_a + 2$	0	$K_a + 2$	$K_a + 3$	$3K_a + 1$	2	$K_a + 2$
	Merge	2	$K_m + 1$	0	$K_m + 1$	$K_m + 2$	$3m+1$	2	$K_m + 1$
	Leave	1	1	0	1	$n - 1$	$\frac{3n}{2} + 2$	1	1
	Mass Leave	1	1	0	1	$n - 1$	$\frac{3n}{2} + 2$	1	1
	Partition	1	1	0	1	$n - 1$	$\frac{3n}{2} + 2$	1	1
GDH3.0	Initialize	n+1	2n-1	2n-3	2	$3n - 6$	$5n-6$	n+1	2n-1
	Join	4	n+3	0	n+3	2n+1	n+3	4	n+3
	Mass Join	$K_a + 3$	$n + 2K_a + 1$	0	$n + 2K_a + 1$	$2n + 3K_a - 2$	$n + 2K_a + 1$	$K_a + 3$	$n + 2K_a + 1$
	Merge	m+3	n+2m+1	0	n+2m+1	$2n + 3m - 2$	n+2m+1	m+3	n+2m+1
	Leave	1	1	0	1	n	n-1	1	1
	Mass Leave	1	K_p	0	K_p	$n - K_p + 1$	$n - K_p$	1	1
	Partition	1	1	0	1	$n - \min(P_i) + 1$	$\max(n - P_i)$	1	1

Table 3: Performance Metrics for Five Group Key Agreement Protocols

when and if newer components come up, without compromising or favoring one algorithm over another.

6 Conclusion and Future Work

In this paper we first presented the General Contributory Group Key Exchange algorithm (GCGKE) as a general algorithmic description of group key protocols to provide a framework for comparison. Proofs of security for GCGKE can be reused for instances of it. In addition, to allow for a more uniform analysis of the performance of these protocols we introduced new evaluation metrics (the time of Signing/Verification in every round, message size sent in every round, and the time of every Op). In future, we will elaborate more strict methodology for group key protocol evaluation, e.g. formulate with a proper weighting system reflecting the various importances of different metrics instead of presenting an empiric evaluation of CCEGK protocols.

References

- [1] C. Adams and S. Lloyd. *Understanding PKI*. Person Education, Inc., third edition, 2003.
- [2] J. Alves-Foss. An efficient secure authenticated group key exchange algorithm for large and dynamic groups. In *Proc. 23rd National Information Systems Security Conference*, pages 254–256, Oct. 2000.
- [3] K. Becker and U. Willie. Communication complexity of group key distribution. In *Proc. 5th Conference on Computers and Communication Security*, pages 1–6, 1998.
- [4] D. Boneh and R. Venkatesan. Breaking rsa may not be equivalent to factoring. In *Advances in Cryptology - EUROCRYPT'98*, pages 59–71, 1998.
- [5] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–652, Nov. 1976.
- [6] Y. Kim, A. Perrig, and G. Tsudik. Group key agreement efficient in communication. *IEEE Transactions on Computers*, 53(7):905–921, Jul. 2004.
- [7] Y. Kim, A. Perrig, and G. Tsudik. Tree-based group key agreement. *ACM Transactions on Information and System Security*, 7(1):60–96, Feb. 2004.
- [8] N. Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48(177):203–209, 1987.
- [9] L. Law, Z. Menezes, M. Qu, J. Solinas, and S. Vanstone. An efficient protocol for authenticated key agreement. Technical report, CORR 98-05, Department of CO, University of Waterloo, Mar. 1998.
- [10] M. Steiner, G. Tsudik, and M. Waidner. Key agreement in dynamic peer groups. Technical report, Information Sciences Institute, Jan. 1999.
- [11] S. Zheng, J. Alves-Foss, and S. Lee. Exploring average performance of group key management algorithms over multiple operations. In *Proc. 4th IASTED International Conference on Communications, Internet and Information Technology*, pages 47–52, Nov. 2005.
- [12] S. Zheng, J. Alves-Foss, and S. Lee. Performance of group key agreement protocols over multiple operations. In *Proc. 17th IASTED International Conference on Parallel and Distributed Computing and Systems*, pages 600–606, Nov. 2005.
- [13] S. Zheng, D. Manz, and J. Alves-Foss. A communication-computation efficient group key algorithm for large and dynamic groups. Technical report, University of Idaho, Sep. 2004.