Server-Aided Password-Authenticated Key Exchange: From 3-Party to Group*

Junghyun Nam[1], Juryon Paik[2], Jeeyeon Kim[2], Youngsook Lee[3], and Dongho Won[2,**]

[1] Department of Computer Engineering, Konkuk University, Korea
jhnam@kku.ac.kr
[2] Department of Computer Engineering, Sungkyunkwan University, Korea
wise96@ece.skku.ac.kr,jeeyeonkim@paran.com,dhwon@security.re.kr
[3] Department of Cyber Investigation Police, Howon University, Korea
ysooklee@howon.ac.kr

Abstract. Protocols for group key exchange are cryptographic algorithms that describe how a group of parties communicating over a public network can come up with a common secret key. Due to their critical role in building secure multicast channels, a number of group key exchange protocols have been proposed over the years for a variety of settings. In this work, we present a new protocol for password-authenticated group key exchange in the model where the clients wishing to establish a common secret do not share any password between them but hold their individual password shared with a trusted server. This model is practical in that no matter how many different session keys for different groups a client wants to generate, he/she does not need to hold multiple passwords but only needs to remember a single password shared with the server. Our construction is generic. We assume a 3-party password-authenticated key exchange protocol and use it as a key component in building our password-authenticated GKE protocol. Our generic protocol requires no further long-term secrets than those used in the underlying 3-party protocol. This implies that if the given 3-party protocol is password-only authenticated, then our group key exchange protocol is password-only authenticated as well.

Keywords: Group key exchange, multicast, 3-party key exchange, password.

1 Introduction

The increasing ubiquity of computer networks is accelerating the development of group-oriented applications in which a group of parties communicate collaboratively to achieve their common interest or objective. Typical group-oriented applications include video/audio teleconferencing, distributed multiplayer games,

* This work was supported by Priority Research Centers Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2010-0020210).

** Corresponding author.

M.J. Smith, G. Salvendy (Eds.): Human Interface, Part I, HCII 2011, LNCS 6771, pp. 339–348, 2011.

grid computing, collaborative workspaces, and social networking services. In particular, social networking services such as Twitter [25] and Facebook [12] have recently gained tremendous popularity and are redefining our sense of community. The proliferation of group-oriented applications has led to a growing concern in security of group communications. The current Internet, by design, is an open network which might be controlled by an adversary. Today's adversaries are equipped with more powerful computing resources and attacking tools than ever before.

One valuable tool for protecting group communications is protocols for group key exchange (GKE). A group of parties communicating over a public network can generate a common secret key (called a *session key*) by running a GKE protocol. Once a session key has been established, the parties can use this key to encrypt and/or authenticate their subsequent multicast messages. This represents a typical way of communicating confidentially and with integrity over a public channel. The session key, of course, must be known only to the intended parties at the end of the protocol run, because otherwise the whole system becomes vulnerable to all manner of attacks. Roughly stated, a key exchange protocol satisfying this requirement is said to be *authenticated*. Due to their significance in building secure multicast channels, authenticated GKE protocols have been extensively studied over the last decades [13,7,8,20,17,6,18,5,15,4,16,14,2,21,1,27]. However, despite all the research efforts made so far, the design of an authenticated GKE protocol is still notoriously hard. Many GKE protocols, even some with a claimed proof of security, have been analyzed to be vulnerable to a certain kind of attack years after they were published [23,11,24,22].

Any protocol for authenticated key exchange inherently requires that the protocol participants establish their long-term authentication secrets (either low-entropy passwords or high-entropy cryptographic keys) before they ever run the protocol. Protocols for password-authenticated key exchange are designed to work even when the authentication secrets are human-memorable passwords chosen from a small known set of values. These password-based protocols, despite their practical significance in today's computing environments, are notoriously hard to design right. The major hurdle to password-authenticated key exchange is (off-line) dictionary attacks in which an adversary exhaustively enumerates all possible passwords in an off-line manner to find out the correct one. In this work, we present a new protocol for password-authenticated GKE. Like the previous protocols of [9,10,19,27], Our protocol assumes a Kerberos-like authentication model in which the clients trying to establish a common secret do not share any password between them but hold their individual password shared with a trusted server. The role of the trusted server in this model is to provide the clients with a centralized authentication service. This model enjoys the obvious practical advantage that no matter how many different session keys for different groups a client wants to generate, he/she does not need to hold multiple passwords but only needs to remember a single password shared with the server. Our protocol differs from previous designs [9,10,19] in that it is constructed generically from a password-authenticated 3-party key exchange protocol secure against active

adversaries. Although Yi et al.'s protocol [27] features generic construction as well, it is different from ours in two aspects: (1) it can be constructed from a GKE protocol secure against passive adversaries and (2) it employs identity-based cryptography where an arbitrary identity like an email address can serve as a public key. Hence, the best way to describe our protocol is the first server-aided password-authenticated GKE protocol that builds on a password-authenticated 3-party key exchange protocol.

In the next section, we give some preliminaries required for the security proof of the proposed protocol, including a communication and adversary model with an associated definition of security. Then, in Section 3, we present our protocol for server-aided password-authenticated GKE protocol and consider its efficiency and security.

2 Formal Setting

Any form of security analysis of a cryptographic construction should be preceded by clear definitions of its security goals and tools. In this section we provide such a preliminary formalism for server-aided password-authenticated GKE.

2.1 Communication and Adversary Model

Participants/Passwords. There are two types of participants in a server-aided password-authenticated GKE protocol: clients and servers. Let $\mathcal{C}$ be the set of all clients and $\mathcal{S}$ be the set of all servers. $\mathcal{C}$ and $\mathcal{S}$ are assumed to be of polynomial size. Before the protocol is executed for the first time, each client $C_i \in \mathcal{C}$ chooses a password pw_i from a dictionary D of size d and then registers it on a server $S \in \mathcal{S}$. The clients in any subset of $\mathcal{C}$ may run the protocol with a server $S \in \mathcal{S}$ to establish a common key, as long as all the clients (in the subset) have registered their individual passwords with the same server S. Each client may run the protocol multiple times either serially or concurrently, with possibly different sets of participants. Thus, at a given time, there could be many instances of a single client. We use C_i^π to denote the π-th instance of client C_i. All instances of a client C_i use the same password pw_i even if they participate in their respective sessions independently.

Partners. Intuitively, the *partners* of an instance is the set of all instances that should compute the same session key as the instance in an execution of the protocol. Like most of previous works, we use the notion of *session IDs* to define partnership between instances. Literally, a session ID (denoted as sid) is a unique identifier of a communication session. In our protocol, session IDs are constructed during protocol runs. We also need the notion of *group IDs* to define partnership properly. A group ID (denoted as gid) is a set consisting of the identities of the protocol participants. This notion is clearly natural because it is impossible (not even defined) to ever execute a group key exchange protocol without participants. Indeed, a group ID is a both necessary and important input to any protocol execution. We use sid_i^π and gid_i^π to denote respectively sid

and gid of instance C_i^π. Note that gid_i^π includes C_i itself and its trusted server S. Session IDs and group IDs are public and assumed to be available to the adversary.

An instance is said to *accept* when it successfully computes a session key in a protocol execution. Let acc_i^π be a boolean variable that evaluates to TRUE if C_i^π has accepted, and FALSE otherwise. We say that any two instances C_i^π and C_j^ω are *partners* of each other, or equivalently, *partnered* iff all the following three conditions are satisfied: (1) $\mathsf{sid}_i^\pi = \mathsf{sid}_j^\omega$, (2) $\mathsf{gid}_i^\pi = \mathsf{gid}_j^\omega$, and (3) $\mathsf{acc}_i^\pi = \mathsf{acc}_j^\omega =$ TRUE. We also say that two instances C_i^π and C_j^ω are *potential partners* of each other, or equivalently, *potentially partnered* iff the first two conditions above hold. We use pid_i^π and ppid_i^π to denote respectively the partners and the potential partners of the instance C_i^π. Then it follows by the definitions that $\mathsf{pid}_i^\pi \subseteq \mathsf{ppid}_i^\pi$.

Adversary. The adversary in our model controls all message flows of the protocol and can ask participants to open up access to any long-term secrets and session keys. These capabilities of the adversary are modeled via various oracles to which the adversary is allowed to make queries.

- $\mathsf{Execute}(\mathsf{gid})$: This query prompts the clients and the server in gid (precisely, their instances) to execute the protocol. The transcript of the execution is returned to the adversary as the output of the query. This models passive attacks on the protocol.
- $\mathsf{Send}(U^\pi, M)$: Let U^π be an instance of either a client or a server. This query sends message M to instance U^π. The instance U^π proceeds as it would in the protocol upon receiving message M; the instance updates its state performing any required computation, and generates and sends out a response message as needed. The response message, if any, is the output of this query and is returned to the adversary. This models active attacks on the protocol, allowing the adversary to control at will all message flows between instances. A query of the form $\mathsf{Send}(U^\pi,$ "`start`":$\mathsf{gid})$ prompts U^π to initiate an execution of the protocol in which the participants are the clients and the server specified in gid.
- $\mathsf{Reveal}(C_i^\pi)$: This query returns to the adversary the session key held by C_i^π. This oracle call captures the idea that exposure of some session keys should not affect the security of other session keys. The adversary is not allowed to ask this query if it has already queried $\mathsf{Test}(C_j^\omega)$ for some $C_j^\omega \in \mathsf{pid}_i^\pi$ (see below for the description of the Test oracle).
- $\mathsf{Corrupt}(U)$: Let U be a client or a server. This query returns to the adversary all long-term secrets of U. This models the adversary's capability of breaking into U's machine and gaining access to the long-term data set stored there. The adversary can issue this query at any time regardless of whether U is currently executing the protocol or not. This oracle call captures the idea that damage due to loss of U's long-term secrets should be restricted to those sessions where U will participate in the future.
- $\mathsf{Test}(C_i^\pi)$: This query provides a means of defining security. The output of this query depends on the hidden bit b that the Test oracle chooses uniformly

at random from $\{0,1\}$ during its initialization phase. The Test oracle returns the real session key held by C_i^π if $b = 1$, or returns a random session key drawn from the key space if $b = 0$. The query can be asked only when instance C_i^π is *fresh* (see Section 2.2 for the definition of freshness). The adversary is allowed a single Test query, at any time during its execution.

Definition 1. *An adversary is called* active *iff it is allowed to access all the oracles described above, and called* passive *iff it is allowed to access all but the* Send *oracle.*

We represent the amount of queries used by an adversary as an ordered sequence of five non-negative integers, $Q = (q_{\text{exec}}, q_{\text{send}}, q_{\text{reve}}, q_{\text{corr}}, q_{\text{test}})$, where the five elements refer to the numbers of queries that the adversary made respectively to its Execute, Send, Reveal, Corrupt, and Test oracles. We call this usage of queries by an adversary the *query complexity* of the adversary. Note that by Definition 1, the query complexity of a passive adversary is always represented as a sequence of the form $Q = (q_{\text{exec}}, 0, q_{\text{reve}}, q_{\text{corr}}, q_{\text{test}})$.

2.2 Security Definition and Assumptions

Freshness. The notion of *freshness* is used in the definition of security to prohibit the adversary from asking the Test query against an instance whose session key (or some information about the key) can be exposed by trivial means.

Definition 2. *The instance C_i^π is considered unfresh iff any of the following conditions hold:*

1. $\mathsf{acc}_i^\pi = \text{TRUE}$.
2. *The adversary queried* $\mathsf{Corrupt}(U)$ *for some* $U \in \mathsf{gid}_i^\pi$ *before some instance in* ppid_i^π *accepts.*
3. *The adversary queried* $\mathsf{Reveal}(C_j^\omega)$ *for some* $C_j^\omega \in \mathsf{pid}_i^\pi$.

All other instances are considered fresh.

Security. The security of a server-aided password-authenticated GKE protocol P against an adversary $\mathcal{A}$ is defined in terms of the probability that $\mathcal{A}$ succeeds in distinguishing random session keys from real session keys established by the protocol P. That is, the adversary $\mathcal{A}$ is considered successful in attacking P if it breaks the semantic security of session keys generated by P. This notion of security is defined in the context of the following two-stage game, where the goal of adversary $\mathcal{A}$ is to correctly guess the value of the hidden bit b chosen by the Test oracle.

- **Stage 1:** $\mathcal{A}$ makes any allowed oracle queries at will as many times as it wishes.
- **Stage 2:** Once $\mathcal{A}$ decides that Stage 1 is over, it outputs a bit b' as a guess for the value of the hidden bit b used by the Test oracle. $\mathcal{A}$ wins the game if $b = b'$.

In the game above, the adversary can keep querying the oracles even after it asked some Test queries. However, when there was the query $\mathsf{Test}(C_i^\pi)$ asked, the adversary is prohibited from querying $\mathsf{Reveal}(C_j^\omega)$ for some $C_j^\omega \in \mathsf{pid}_i^\pi$. This restriction reflects the fact that the adversary can win the game unfairly by using the information obtained via the query $\mathsf{Reveal}(C_j^\omega)$.

Given the game above, the advantage of $\mathcal{A}$ in attacking the protocol P is defined as $\mathsf{Adv}_P(\mathcal{A}) = |2 \cdot \Pr[b = b'] - 1|$. Note that this definition is equivalent to say that the advantage of $\mathcal{A}$ is the difference between the probabilities that $\mathcal{A}$ outputs 1 in the following two experiments constituting the game: the *real experiment* where the query to the Test oracle is answered with the real session key, and the *random experiment* where the Test query is answered with a random session key. Thus, if we denote the real and the random experiments respectively as $\mathsf{Exp}_P^{\mathsf{real}}(\mathcal{A})$ and $\mathsf{Exp}_P^{\mathsf{rand}}(\mathcal{A})$, the advantage of $\mathcal{A}$ can be equivalently defined as $\mathsf{Adv}_P(\mathcal{A}) = |\Pr[\mathsf{Exp}_P^{\mathsf{real}}(\mathcal{A}) = 1] - \Pr[\mathsf{Exp}_P^{\mathsf{rand}}(\mathcal{A}) = 1]|$, where the outcomes of the experiments is the bit output by $\mathcal{A}$.

We say that the group key exchange protocol P is *secure* if $\mathsf{Adv}_P(\mathcal{A})$ is negligible for all probabilistic polynomial time adversaries $\mathcal{A}$. To quantify the security of protocol P in terms of the amount of resources expended by adversaries, we let $\mathsf{Adv}_P(t, Q)$ denote the maximum value of $\mathsf{Adv}_P(\mathcal{A})$ over all $\mathcal{A}$ with time complexity at most t and query complexity at most Q.

3 Our Protocol

We here present a password-authenticated GKE protocol n-PAKE.

Participants. The protocol participants consist of: (1) a set of clients $C_1, \ldots, C_n \in \mathcal{C}$ who wish to establish a common session key and (2) a server $S \in \mathcal{S}$ who provides the clients with a centralized authentication service. The server S is trusted to behave in an "honest but curious" manner; that is, S may attempt to learn the session key only by passive eavesdropping.

Building Blocks. The cryptographic building blocks of n-PAKE include:

- *a password-authenticated 3-party key exchange protocol* 3-PAKE that, given two users $C_i, C_j \in \mathcal{U}$ and a server $S \in \mathcal{S}$ (or rather their identities), returns either a secret key $\kappa \in \{0,1\}^k$ or a special symbol $\top$ (indicating failure of the key establishment).
- *a collision-resistant pseudorandom function family* $\mathcal{F} = \{F^\ell\}_{\ell \in \mathbb{N}}$ with $F^\ell = \{F_s^\ell\}_{s \in \{0,1\}^L}$. Collision-resistance means, informally, that there exists a value v such that no efficient adversary can find two different indices $s, s' \in \{0,1\}^L$ such that $F_s(v) = F_{s'}(v)$. We assume three publicly known values v_1, v_2 and v_3 that satisfies the collision-resistance condition.
- *a hash function* $H : \{0,1\}^* \to \{0,1\}^L$ from a family of universal hash functions. H is used to select an index within the aforementioned collision-resistant pseudorandom function family.

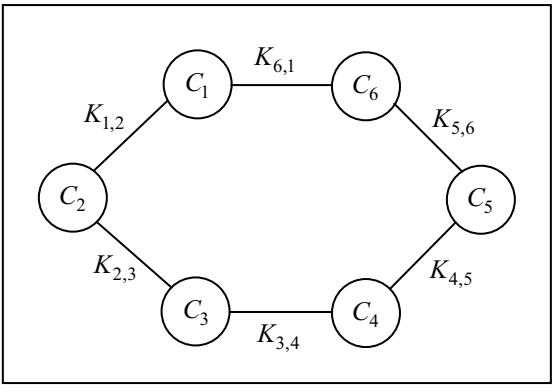

Fig. 1. Pairwise key establishments among $C_1, \ldots, C_6$

Initialization. Before n-PAKE is ever executed, the protocol participants generate public parameters and long-term secrets. The server S determines the above-mentioned building blocks and their related parameters. Each client C_i chooses a password pw_i and stores it on the server S.

Protocol Execution. If the underlying 3-party protocol 3-PAKE takes r rounds of communications, then the protocol n-PAKE takes $r + 2$ rounds. Recall that each C_i receives as input a group ID $\mathsf{gid}_i = \{C_1, \ldots, C_n\} \cup \{S\}$ to start to run the protocol. n-PAKE runs as follows (throughout the protocol description, all indices are to be taken in a cycle, i.e., $C_{n+1} = C_1$, etc.):

[Round $1 \sim r$]: Each neighboring pair of clients C_i and C_{i+1}, for $i = 1, \ldots, n$, generates a pairwise key $K_{i,i+1}$ by running 3-PAKE with the server S. Accordingly, at the end of the r^{th} round, each C_i holds two pairwise keys $K_{i-1,i}$ and $K_{i,i+1}$ shared respectively with C_{i-1} and C_{i+1}. (Fig. 1 shows pairwise key establishments among $C_1, \ldots, C_6$.)

[Round $r + 1$]: Each C_i computes

$$\overleftarrow{\sigma}_i = F_{H(C_i \| K_{i-1,i} \| \mathsf{gid}_i)}(v_1),$$
$$\overrightarrow{\sigma}_i = F_{H(C_i \| K_{i,i+1} \| \mathsf{gid}_i)}(v_1),$$

and sends $\overleftarrow{\text{Auth}}_i = \langle C_i \| \overleftarrow{\sigma}_i \rangle$ and $\overrightarrow{\text{Auth}}_i = \langle C_i \| \overrightarrow{\sigma}_i \rangle$ respectively to C_{i-1} and C_{i+1}. Upon receiving $\overrightarrow{\text{Auth}}_{i-1}$ and $\overleftarrow{\text{Auth}}_{i+1}$, C_i verifies the correctness of both $\overrightarrow{\sigma}_{i-1}$ and $\overleftarrow{\sigma}_{i+1}$ in the straightforward way. If either one of the verifications fails, C_i aborts the protocol.

[Round $r + 2$]: Each C_i computes

$$X_i = K_{i-1,i} \oplus K_{i,i+1}.$$

and broadcasts $\text{Xor}_i = \langle C_i \| X_i \rangle$. Upon receiving the XOR-values, C_i checks that $X_1 \oplus X_2 \oplus \cdots \oplus X_n = 0$. If the check fails, C_i aborts the protocol.

[Key Computation]: Using $K_{i-1,i}$ and the XOR-values, each C_i computes

$$\begin{aligned} K_{i-2,i-1} &= X_{i-1} \oplus K_{i-1,i}, \\ K_{i-3,i-2} &= X_{i-2} \oplus K_{i-2,i-1}, \\ &\vdots \\ K_{i,i+1} &= X_{i+1} \oplus K_{i+1,i+2}. \end{aligned}$$

Then, C_i defines a master key

$$K = \langle K_{n,1} \| K_{1,2} \| \ldots \| K_{n-1,n} \| \mathsf{gid}_i \rangle,$$

computes the session key $SK_i = F_{H(K)}(v_2)$, and sets $\mathsf{sid}_i = F_{H(K)}(v_3)$.

For the instantiation of 3-PAKE, any particular choice that is, informally speaking, secure against an active adversary will do. Possible candidates include the generic 3-party protocols from [3] and [26].

Remark 1. We assume that the number of clients participating in n-PAKE is greater than 2, i.e., $n \geq 3$. It is because in case of $n = 2$, running the 3-party protocol 3-PAKE suffices for two clients to establish a session key.

Efficiency Consideration. Our 3-to-n compiler is quite efficient both in terms of computation cost and communication cost. The transformation process of the compiler adds $O(n)$ computation complexity and $O(1)$ round complexity to those of the underlying 3-party protocol. From the practical point of view, the $O(n)$ increase in computation overhead is not that significant because the increase is purely due to exclusive-or (XOR) operations, which can be implemented efficiently in hardware and/or software.

Security Result. Here we claim that the GKE protocol n-PAKE is secure against active adversaries under the security of the 3-party key exchange protocol 3-PAKE against active adversaries. The following theorem makes this claim precise.

Theorem 1. *Let* $Q = (q_{\text{exec}}, q_{\text{send}}, q_{\text{reve}}, q_{\text{corr}}, q_{\text{test}})$ *and* $Q' = (n \cdot q_{\text{exec}}, 2 \cdot q_{\text{send}}, q_{\text{reve}}, q_{\text{corr}}, q_{\text{test}})$. *For any adversary with time complexity at most* t *and query complexity at most* Q, *its advantage in breaking the security of protocol n-PAKE is upper bounded by:*

$$\mathsf{Adv}_{n\text{-}PAKE}(t, Q) \leq \mathsf{Adv}_{3\text{-}PAKE}(t', Q'),$$

where $t' = t + O(n \cdot q_{\text{exec}} t_{3\text{-}PAKE} + 2 \cdot q_{\text{send}} t_{3\text{-}PAKE})$ *and* $t_{3\text{-}PAKE}$ *is the time required for execution of 3-PAKE by any party.*

References

1. Abdalla, M., Bohli, J.-M., González Vasco, M.I., Steinwandt, R. (Password) authenticated key establishment: From 2-party to group. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 499–514. Springer, Heidelberg (2007)

2. Abdalla, M., Bresson, E., Chevassut, O., Pointcheval, D.: Password-based group key exchange in a constant number of rounds. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T. (eds.) PKC 2006. LNCS, vol. 3958, pp. 427–442. Springer, Heidelberg (2006)
3. Abdalla, M., Fouque, P.-A., Pointcheval, D.: Password-based authenticated key exchange in the three-party setting. In: Vaudenay, S. (ed.) PKC 2005. LNCS, vol. 3386, pp. 65–84. Springer, Heidelberg (2005)
4. Boyd, C., Nieto, J.: Round-optimal contributory conference key agreement. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 161–174. Springer, Heidelberg (2002)
5. Bresson, E., Chevassut, O., Pointcheval, D.: Group diffie-hellman key exchange secure against dictionary attacks. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 497–514. Springer, Heidelberg (2002)
6. Bresson, E., Chevassut, O., Pointcheval, D., Quisquater, J.-J.: Provably authenticated group Diffie-Hellman key exchange. In: 8th ACM Conference on Computer and Communications Security (CCS 2001), pp. 255–264 (2001)
7. Burmester, M., Desmedt, Y.G.: A secure and efficient conference key distribution system. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 275–286. Springer, Heidelberg (1995)
8. Burmester, M., Desmedt, Y.: Efficient and secure conference-key distribution. In: Lomas, M. (ed.) Security Protocols 1996. LNCS, vol. 1189, pp. 119–129. Springer, Heidelberg (1997)
9. Byun, J.W., Lee, D.-H.: N-party encrypted diffie-hellman key exchange using different passwords. In: Ioannidis, J., Keromytis, A.D., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 75–90. Springer, Heidelberg (2005)
10. Byun, J.W., Lee, S.-M., Lee, D.-H., Hong, D.: Constant-round password-based group key generation for multi-layer ad-hoc networks. In: Clark, J.A., Paige, R.F., Polack, F.A.C., Brooke, P.J. (eds.) SPC 2006. LNCS, vol. 3934, pp. 3–17. Springer, Heidelberg (2006)
11. Choo, K.-K.R., Boyd, C., Hitchcock, Y.: Errors in computational complexity proofs for protocols. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 624–643. Springer, Heidelberg (2005)
12. Facebook, `http://www.facebook.com`
13. Ingemarsson, I., Tang, D., Wong, C.: A conference key distribution system. IEEE Transactions on Information Theory 28(5), 714–720 (1982)
14. Katz, J., Shin, J.: Modeling insider attacks on group key-exchange protocols. In: 12th ACM Conference on Computer and Communications Security (CCS 2005), pp. 180–189 (2005)
15. Katz, J., Yung, M.: Scalable protocols for authenticated group key exchange. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 110–125. Springer, Heidelberg (2003)
16. Kim, H., Lee, S., Lee, D.: Constant-round authenticated group key exchange for dynamic groups. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 245–259. Springer, Heidelberg (2004)
17. Kim, Y., Perrig, A., Tsudik, G.: Simple and fault-tolerant key agreement for dynamic collaborative groups. In: 7th ACM Conference on Computer and Communications Security (CCS 2000), pp. 235–244 (2000)
18. Kim, Y., Perrig, A., Tsudik, G.: Communication-efficient group key agreement. In: IFIP SEC 2001, pp. 229–244 (2001)

19. Kwon, J., Jeong, I., Sakurai, K., Lee, D.: Password-authenticated multi-party key exchange with different passwords. Cryptology ePrint Archive, Report 2006/476 (2006)
20. Mayer, M., Yung, M.: Secure protocol transformation via "Expansion": From two-party to groups. In: 6th ACM Conference on Computer and Communications Security (CCS 1999), pp. 83–92 (1999)
21. Nam, J., Paik, J., Kim, U.-M., Won, D.H.: Constant-round authenticated group key exchange with logarithmic computation complexity. In: Katz, J., Yung, M. (eds.) ACNS 2007. LNCS, vol. 4521, pp. 158–176. Springer, Heidelberg (2007)
22. Nam, J., Paik, J., Kim, U., Won, D.: Security enhancement to a password-authenticated group key exchange protocol for mobile ad-hoc networks. IEEE Communications Letters 12(2), 127–129 (2008)
23. Pereira, O., Quisquater, J.-J.: A security analysis of the Cliques protocols suites. In: 14th IEEE Computer Security Foundations Workshop, pp. 73–81 (2001)
24. Shim, K., Woo, S.: Cryptanalysis of tripartite and multi-party authenticated key agreement protocols. Information Sciences 177(4), 1143–1151 (2007)
25. Twitter, `http://twitter.com`
26. Wang, W., Hu, L.: Efficient and provably secure generic construction of three-party password-based authenticated key exchange protocols. In: Barua, R., Lange, T. (eds.) INDOCRYPT 2006. LNCS, vol. 4329, pp. 118–132. Springer, Heidelberg (2006)
27. Yi, X., Tso, R., Okamoto, E.: ID-Based group password-authenticated key exchange. In: Takagi, T., Mambo, M. (eds.) IWSEC 2009. LNCS, vol. 5824, pp. 192–211. Springer, Heidelberg (2009)