Security Improvement on a Password-Authenticated Group Key Exchange Protocol

Junghyun Nam[†]

August 1, 2010

Abstract

A group key exchange (GKE) protocol is designed to allow a group of parties communicating over a public network to establish a common secret key. As group-oriented applications gain popularity over the Internet, a number of GKE protocols have been suggested to provide those applications with a secure multicast channel. Among the many protocols is Yi et al.'s password-authenticated GKE protocol in which each participant is assumed to hold their individual password registered with a trusted server. A fundamental requirement for password-authenticated key exchange is security against off-line dictionary attacks. However, Yi et al.'s protocol fails to meet the requirement. In this work, we report this security problem with Yi et al.'s protocol and show how to solve it.

Keywords: Group key exchange, password, dictionary attack, identity-based cryptography.

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, grid computing, collaborative workspaces, and social networking services. In particular, social networking services such as Twitter [5] and Facebook [3] 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. The situation gets even

[†]Department of Computer Science, Konkuk University, 322 Danwol-dong, Chungju-si, Chungcheongbuk-do 380-701, Republic of Korea.
E-mail: `jhnam@kku.ac.kr`

worse when we consider malicious insiders. In general, we cannot expect complete trust among all group members just because they collaborate to achieve a specific purpose; collaboration does not imply full trust. Perhaps malicious insiders pose the most serious security threat to many organizations and enterprises.

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*. 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. Indeed, many protocols, even some with a claimed proof of security, have been found to be vulnerable to a dictionary attack years after they were published. In the current work, we present another instance of the vulnerability that can be identified in the password-authenticated GKE protocol proposed recently by Yi et al. [6]. Like the previous protocols of [1, 2, 4], Yi et al.'s protocol assumes a Kerberos-like authentication model in which each client, who is a potential participant of the protocol, shares a password with a trusted server but not with any other clients. 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. Yi et al.'s protocol differs from previous designs [1, 2, 4] in two aspects: (1) it can be constructed generically from any 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. Despite its practicality and uniqueness, Yi et al.'s protocol should not be adopted in its present form. Due to a fatal flaw in its design, Yi et al.'s protocol fails to protect the passwords of its participants against a dictionary attack. We here report this critical problem with Yi et al.'s protocol and present how to solve it.

2 Yi et al.'s Group Key Exchange

This section reviews Yi et al.'s password-authenticated GKE protocol PGKE [6]. There are three kinds of entities involved in PGKE: (1) a set of n clients $C_1, \ldots, C_n$ who wish to establish a common session key; (2) a server S who provides the clients with a centralized authentication service; (3) a private key generator PKG who generates global system parameters as well as S's long-term private keys. Both S and PKG are trusted to behave in an "honest but curious" manner; that is, S and PKG may attempt to learn the session key only by passive eavesdropping.

Building Blocks. The cryptographic building blocks of PGKE include:

- *a group key exchange protocol* GKE which is secure against a passive adversary. Every message of GKE is assumed to be sent — via point-to-point links — to all protocol participants. This assumption implies that in GKE, the set of all messages sent and received by each participant is expected to be the same.

- *an identity-based encryption scheme* IBE which is secure against an adaptive chosen ciphertext attack. We let Ecrt and Dcrt be the encryption and decryption algorithms of IBE. The plaintext space of IBE is $\mathcal{M} = \{0,1\}^\ell$ for some ℓ.

- *an identity-based signature scheme* IBS which is existentially unforgeable under an adaptive chosen message attack. We let Sign and Vrfy be the signing and verification algorithms of IBS.

Initialization. Before the protocol PGKE is ever executed, the following initialization is performed to generate public parameters and long-term secrets.

PUBLIC PARAMETERS. PKG chooses: (1) a large cyclic group $\mathbb{G}$ of prime order q and a generator g of $\mathbb{G}$ and (2) two collision-resistant hash functions $H_1 : \{0,1\}^* \rightarrow \mathcal{M}$ and $H_2 : \{0,1\}^* \rightarrow \{0,1\}^\lambda$. (Here, λ is the security parameter that determines the length of session identifiers constructed during protocol runs.) This is in addition to generating any public parameters needed for GKE, IBE and IBS.

LONG-TERM SECRETS. The server S obtains from PKG its private decryption/signing keys (DK_S, SK_S) corresponding to its public key ID_S. (Here, the public key ID_S is an arbitrary identity of S, and is used both for encryption and verification purposes.) Each client C_i chooses a password pw_i and stores it on the server S.

Protocol Execution. If the protocol GKE takes r rounds of communications, then the protocol PGKE runs in $r + 2$ rounds as follows:

[ROUND $1 \sim r$]: The clients $C_1, \ldots, C_n$ execute the protocol GKE. Let k_i be the key computed by C_i as a result of the execution of GKE. Let sid_i be the (ordered) concatenation of all messages sent and received by C_i during the course of the execution.

[ROUND $r+1$]: Each client C_i computes $SID_i = H_2(g^{k_i}|sid_i)$ and sets $PID_i = (C_1, C_2, \ldots, C_n, S)$. Then C_i computes

$$Auth_i = \mathsf{Ecrt}_{ID_S}(H_1(SID_i|PID_i|pw_i))$$

and sends the message $M_i = C_i|SID_i|Auth_i$ to the server S. Upon receiving all of $M_1, \ldots, M_n$, the server S sets $SID_S = SID_1$ and $PID_S = (C_1, C_2, \ldots, C_n, S)$ and checks that the following equation holds for all $i = 1, \ldots, n$:

$$\mathsf{Dcrt}_{DK_S}(Auth_i) = H_1(SID_S|PID_S|pw_i).$$

If any of the checks fails, S terminates the protocol execution.

[ROUND $r+2$]: S generates a signature

$$Auth_S = \mathsf{Sign}_{SK_S}(PID_S|SID_S)$$

and broadcasts the message $M_S = S|Auth_S$. After receiving M_S, each client C_i checks that

$$\mathsf{Vrfy}_{ID_S}(PID_i|SID_i, Auth_S) = 1.$$

If the verification fails, C_i aborts the protocol. Otherwise, C_i computes the session key $K_i = g^{k_i^2}$.

3 Security Analysis

Resistance against dictionary attacks is the fundamental security requirement that should be satisfied by any password-based protocols for authenticated key exchange. However, the PGKE protocol described above fails to meet the requirement. In this section, we reveal this security problem with PGKE and then suggest a countermeasure to the attack.

Dictionary Attack. Consider an adversary $\mathcal{A}$ whose goal is to find out the password of client C_i. Then, the following describes a dictionary attack mounted by $\mathcal{A}$ to achieve its goal.

1. As the $(r+1)^{\text{th}}$ round of PGKE proceeds, $\mathcal{A}$ eavesdrops on the message $M_i = C_i|SID_i|Auth_i$ sent from C_i to S.

2. $\mathcal{A}$ next makes a guess pw_i' for the password pw_i and computes

$$Auth_i' = \mathsf{Ecrt}_{ID_S}(H_1(SID_i|PID_i|pw_i')).$$

3. $\mathcal{A}$ then verifies the correctness of pw_i' by checking that $Auth_i'$ is equal to $Auth_i$. If pw_i' and pw_i are equal, then the equality $Auth_i' = Auth_i$ ought to be satisfied.

4. $\mathcal{A}$ repeats steps 2 and 3 until a correct password is found.

This dictionary attack may lead to devastating losses of passwords because: (1) it can be mounted against any of the clients and (2) the steps for verifying password guesses can be performed in an off-line manner by an automated program.

Of course, there is a possibility in the dictionary attack that the adversary $\mathcal{A}$ comes up with a password guess pw'_i such that $pw'_i \neq pw_i$ but $H_1(SID_i|PID_i|pw'_i) = H_1(SID_i|PID_i|pw_i)$ and thus $Auth'_i = Auth_i$. However, this possibility should be negligible because otherwise H_1 is not collision-resistant.

Countermeasure. The security failure of PGKE is attributed to one obvious flaw in the protocol design: the password pw_i is the only secret included in the computation of $Auth_i = \mathsf{Ecrt}_{ID_S}(H_1(SID_i|PID_i|pw_i))$. SID_i can be obtained directly from the message M_i since it is transmitted in the clear. PID_i represents the identities of protocol participants and is generally assumed to be available to the adversary. (However, this assumption is not necessary for our dictionary attack if we think of the adversary $\mathcal{A}$ as a malicious client $C_j (\neq C_i)$ who also is a protocol participant.) On the basis of this observation, one may suggest that a simple defense against the attack is to transmit SID_i in an encrypted form. This suggestion, of course, is valid if the adversary $\mathcal{A}$ does not know the key k_i from which SID_i can be derived. However, notice that $\mathcal{A}$ could be any (malicious) client C_j who runs the protocol with client C_i. Hiding SID_i from the public makes no difference to such an inside adversary.

As the discussion above highlights, a proper defense to the dictionary attack must ensure that the password of a client should not be disclosed even to other clients participating in the same protocol run. Keeping this in mind, we recommend to change the $(r+1)^{\text{th}}$ round of PGKE as follows:

[Round $r+1$] (revision): Each client C_i chooses a random $x_i \in \{0,1\}^\ell$, computes $X_i = \mathsf{Ecrt}_{ID_S}(x_i)$ and $SID_i = H_2(g^{k_i}|sid_i)$, and sets $PID_i = (C_1, C_2, \ldots, C_n, S)$. Then C_i computes

$$Auth_i = \mathsf{Ecrt}_{ID_S}(H_1(SID_i|PID_i|pw_i|x_i))$$

and sends the message $M_i = C_i|SID_i|X_i|Auth_i$ to the server S. After receiving the messages $M_1, \ldots, M_n$, the server S sets $SID_S = SID_1$ and $PID_S = (C_1, C_2, \ldots, C_n, S)$ and checks that $H_1(SID_S|PID_S|pw_i|\mathsf{Dcrt}_{DK_S}(X_i))$ is equal to $\mathsf{Dcrt}_{DK_S}(Auth_i)$ for all $i = 1, \ldots, n$. If any of the checks fails, S terminates the protocol execution.

The other rounds of the protocol remain unchanged.

The key change made in our revision is the inclusion of the confounder x_i into the computation of $Auth_i$. This change prevents $Auth_i$ from being used as a password verifier. Hence, the dictionary attack is no longer valid against the improved protocol.

References

[1] J. Byun and D. Lee, "N-party encrypted Diffie-Hellman key exchange using different passwords," in *Proc. 3rd International Conference on Applied Cryptography and Network Security*, LNCS vol. 3531, pp. 75–90, 2005.

[2] J. Byun, S. Lee, D. Lee, and D. Hong, "Constant-round password-based group key generation for multi-layer ad-hoc networks," in *Proc. 3rd International Conference on Security in Pervasive Computing*, LNCS vol. 3934, pp. 3–17, 2006.

[3] Facebook, http://www.facebook.com.

[4] J. Kwon, I. Jeong, K. Sakurai, and D. Lee, "Password-authenticated multi-party key exchange with different passwords," *Cryptology ePrint Archive*, Report 2006/476, 2006.

[5] Twitter, http://twitter.com.

[6] X. Yi, R. Tso, and E. Okamoto, "ID-Based group password-authenticated key exchange," *Advances in Information and Computer Security — 4th International Workshop on Security*, LNCS vol. 5824, pp. 192–211, 2009.