BULETINUL ŞTIINŢIFIC al Universităţii *POLITEHNIC*A din Timişoara, România,
Seria AUTOMATICĂ ŞI CALCULATOARE
SCIENTIFIC BULLETIN of The *POLITEHNICA* University of Timişoara, Romania,
Transactions on AUTOMATIC CONTROL and COMPUTER SCIENCE, Vol. 59 (73), No. 2, Month12 2014, ISSN 1224-600X

Attacks against Group Key Transfer Protocols based on Secret Sharing

Ruxandra F. Olimid

Department of Computer Science, University of Bucharest, Romania
E-Mail: ruxandra.olimid@fmi.unibuc.ro

***Abstract* – Securing group applications usually requires a pre-established private group key, which can result as the output of a Group Key Transfer protocol (GKT). Some of the recent GKT protocols lack a security analysis and hence are susceptible to simple attacks. We analyze three such protocols and remark vulnerabilities that allow an adversary to make group members accept distinct group keys or even gain no key at all. The attacks remain hidden (until the end of the protocol execution), which prevents players to immediately ask for re-execution.**

***Keywords:* Group Key Transfer, Secret Sharing, Attack.**

I. INTRODUCTION

Group applications like text communication, video conferences, data sharing or collaborative tasks are widely used today. Securing such applications is a challenging task, especially when the group size is large and the members are spread across different locations and use diverse protection mechanisms.
To achieve security, most protocols follow a key establishment phase, which grant all users a common secret session key that is subsequently used to achieve the main goals of secure communication: *confidentiality*, *authentication* and *integrity*.
The key can be obtained as the output of a *Group Key Transfer (GKT)* protocol. In a GKT protocol, a privileged party called the *Key Generation Center (KGC)* selects a fresh group key and securely distributes it to the other parties; the initiator of the protocol might sometimes play the role of the KGC.
Secret sharing - a cryptographic method to split a secret into multiple shares that are then distributed to the participants via secure channels such that only authorized subset of participants can reconstruct the secret - are widely used as building blocks of GKT protocols. Examples include the protocols of Harn and Lin [1], Hsu et al. [2] and Yuan et al.[8], which we will refer to later in this paper.

A. Motivation and Personal Contribution

Many papers published in the last years introduce GKT protocols based on secret sharing that lack security proofs. The omission of security analysis leads to a high probability to discover vulnerabilities. Table 1 exemplifies some attacks on recent protocols.
In this paper, we analyze the possibility of theoretical attacks against the three the protocols in Table 1. The attacks permit an adversary to make distinct users end up with different keys or no key at all. We can call our attacks *Denial of Service (DoS)* because the user will not be able to benefit of the service that requires the common secret key. In addition to traditional DoS attacks (which flood the resource or restrict communications), our attacks remain hidden until the end of the protocol execution (i.e. the KGC or any other user do not realize the attack is taking place).
First, we recall the active insider attack against Hsu et al.'s construction from [5] that makes distinct users accept different group keys; the values of these keys are not random, but chosen by the adversary, on his own wish. The vulnerability is caused by an authenticity flaw that permits the attacker to impersonate the initiator.
Second, we introduce two attacks against Harn and Lin's and Yuan et al.'s protocols that restricts users to obtain the group key, while the KGC (or the initiator of the session) is unaware of this fact.

TABLE 1. Attacks against Group Key Transfer Protocols

Protocol	**Attacks**
Harn and Lin (2010) [1]	Replay Attack [3] Man-in-the-Middle [9]
Hsu et al. (2012) [2]	Active Insider Attack [5]
Yuan et al. (2013) [8]	Replay Attack [4,6] Active Insider Attack [4,6]

B. Outline

The paper is organized as follows. Next section contains the preliminaries. Section III presents three attacks mounted agianst recent GKT protocols. Last section concludes.

II. PRELIMINARIES

A. Group Key Transfer – Goals and Security

GKT protocols rely on a trusted entity called *Key Generation Center (KGC)* to select a key and privatly distribute it to the *qualified* (or *authorized*) users. The KGC can an *insider* (a group member) or an *outsider* (a non group member) .
To securerly distribute the key to the parties, the KGC first shares a long-lived key with each group member (e.g.: a symmetric key or password) during the *Users Registration Phase*. The long-lived key will be later used for multiple protocol executions, hence once the player is registered he can participate to several sessions.
The KGC is trusted by all participants as honest in the sense that it selects a *fresh* key (a uniformly random value that has not been used before) and does not reveal the key to unqualified parties. *Key confidentiality* assures that the key remains hidden for all unauthorized (group of) parties, even if the protocol runs for multiple sessions.
Although key confidentiality guarantees that unqualified members cannot obtain the group key, it does not assure that the qualified parties have actually computed the correct key or any key at all. For example, an attacker can ruin *key consistency* such that at the end of the protocol distinct users own different keys. This leads to a *Denial of Service (DoS)* attack in the sense that the victim cannot benefit of the resources or services that require the key. A similar DoS attack prevents the victims to learn any key at the end of the protocol; the attack is successful when it remains hidden (i.e. the iniator belives the victim has the key, while the victim is unaware that the protocol took place) - otherwise, any party could abort and ask for re-execution.
In this paper, we review an attack against Hsu et al.'s GKT that ruins *key consistency* in the sense that the victim will accept an incorrect key and introduce two attacks against Harn and Lin's and Yuan et al.'s constructions that prevents the victim to know the protocol is taking place and hence learns no key at all.

B. Secret Sharing

Secret sharing represents a mechanism that splits a secret into multiple components (called *shares*) such that the secret can be recovered from *authorized* (or *qualified*) sets of shares, while it remains (perfectly) hidden for *unauthorized* (or *unqualified*) sets of shares. A secret sharing scheme consists of three steps:

1. *Sharing Phase*: a dealer splits the secret into multiple shares*;*
2. *Distribution of shares*: the dealer securely sends one or more shares to each party;
3. *Reconstruction:* a qualified set of parties combines their shares to recover the secret.

The most popular secret sharing scheme is Shamir's [7]: a dealer generates a t-degree polynomial $f(x)$ such that $f(0)=S$, where S is the secret to be shared; the dealer securerly distributes $f(i)$ to the user U_i; during reconstruction phase, any $t+1$ users can recover S by Lagrange interpolation.
Secret sharing schemes are widely used as building blocks of cryptographic protocols in general and GKT in particular. All three protocols we analyze in this paper make use of secret sharing; two of them (Harn and Lin's protocol [1] and Yuan et al.'s protocol [8]) are based on Shamir's scheme.

C. Notations

We introduce next the notations we will use for the rest of the paper.
Let $\{U_1,...,U_m\}$ be the set of all possible users, $\{U_1,...,U_t\}$ the subset of authorized participants to a given session with the KGC or U_1 as initiator (after a possible reordering), $Sig_{U_i}(M)$ the signature of U_i on a message M, Ver_{U_i} the corresponding public verification and h, h_1, h_2 collision-resistant hash functions.
We consider $U_a \in \{U_2,...,U_m\}$ to be an active attacker, other than the initiator, whose goal is to break key consistency or prevent users to gain the common group key (the initiator has no interest in ruin the protocol; besides, for $U_a = U_i$ the attack is trivial). We denote by $\xleftarrow{R} X$ a uniformly random choice from a specified set of values X, $\|$ string concatenation and $A \rightarrow^{*}$ a broadcast message originating from A.
Finally, let $Z_p^* = \{1,2,...,p-1\}$ be the usual notation for the multiplicative group of order prime p and G a multiplicative group (of prime order p or composed order $n = pq$, depending of the context). For simplicity of exposure, we do not explicitly specify each time the group where the computation takes palce, since it is clear from the context.

III. DENIAL-OF-SERVICE ATTACKS AGAINST GKT PROTOCOLS

The current section describes three attacks against recent GKT protocols: an active insider attack against Hsu et al.'s protocol introduced by Olimid [5] that allows a qualified

user to make distinct users end up with different keys and two attacks on Harn and Lin's, respectivelly Yuan et al.'s constructions that prevents users to gain access to the group key. All the attacks remain hidden until the end of the protocol execution and therefore prevents users to ask for re-execution.
We allow the adversary to be *active*, in the sense that it has full control over the communication channel: U_a can eavesdrop on the communication channel, modify message content, prevent messages from reaching their destination and inject new messages of his own choice.
We also assume secure communication during registration to avoid long-lived key lekage and hence trivial win.

A. Attack against Hsu et al.'s Protocol

Fig.1 describes Hsu et al.'s construction in detail [2]. The protocol assumes that the *Discrete Logarithm Problem (DLP)* is hard in the multiplicative group G (i.e. given $pk_i = g^{sk_i}$ it is computationally infeasible to compute $sk_i, i = 1,...,m$) and that *Computational Diffie Hellman (CDH)* holds (i.e. given pk_i and pk_j it is computationally infeasible to compute $g^{sk_i sk_j}, i, j = 1,...,m, i \neq j$).
Unlike the majority of GKT protocols, the construction does not inquire an online KGC - the initiator U_1 performs this role.

Initialization. Let G be the multiplicative cyclic group of order p, with g as generator, where p is a large safe prime (i.e. $p' = \frac{p-1}{2}$ is also a prime);

Users Registration. Each user $U_i, i = 1,...,m$ owns a pulic-private key pair (pk_i, sk_i) s.t. $pk_i = g^{sk_i}$ in G;

Round 1. User U_1:
1.1. chooses $r_1 \leftarrow^R Z_p^*$;
1.2. sends a key generation request:

$$U_1 \rightarrow^* (\{U_1,...,U_t\}, r_1, pk_1)$$

Round 2. Each user $U_i, i = 2,...,t$:
2.1. chooses $r_i \leftarrow^R Z_p^*$;
2.2. computes $S_i = pk_1^{sk_i r_i r_1}$ a shared secret with U_1 and $Auth_i = h(S_i, r_1)$;
2.3. broadcasts:

$$U_i \rightarrow^* (r_i, pk_i, Auth_i)$$

Round 3. User U_1:
3.1. computes $S_i = pk_i^{sk_1 r_i r_1}, i = 2,...,t$;
3.2. checks if $Auth_i = h(S_i, r_1), i = 2,...,t$;
If at least one equality does not hold, he quits;
3.3. chooses the group key $k \leftarrow^R Z_p^*$, splits each secret S_i into two parts $S_i = x_i \parallel y_i$ and computes $t-1$ values $K_i = k - T_i$, where $T_i = (y_i v(x_i), r)$ is the inner product of the vectors $y_i v(x_i) = y_i \sum_{j=1}^{t} x_i^{j-1} e_j$ s.t. $e_j = (0,...,1,...,0)$ with 1 on position j, $i = 2,...,t$ and $r = (r_1,...,r_t)$;
3.4. computes

$$Auth = h(k, U_1,...,U_t, r_1,...,r_t, K_2,...,K_t);$$

3.5. broadcasts:

$$U_1 \rightarrow^* (Auth, K_2,...,K_t)$$

Key Computation. Each user $U_i, i = 2,...,t$:
4.1. computes the inner product $T_i = (y_i v(x_i), r)$ and recovers the group key $k = T_i + K_i$;
4.2. checks if

$$Auth = h(k, U_1,...,U_t, r_1,...,r_t, K_2,...,K_t)$$

If the equality does not hold, he quits.

Fig. 1. Hsu et al.'s Group Key Transfer Protocol [2]

We showed in [5] that Hsu et al.'s protocol is susceptible to an attack mounted from inside. The adversary succeeds to make different authorized users accept distinct keys and therefore breaks key consistency: any qualified user $U_i, U_i \notin \{U_a, U_1\}$ (except the initiator) computes a key k_i that has been chosen in advance by the adversary.
Fig.2 reveals the attack. It is self-contained and hence we omit other comments. We only emphasize that the user U_i is unable to reveal the attack during the protocol execution: U_i recovers the correct value T_i and computes the group key as $k_i = T_i + K_i^{'}$. The verification holds in the Key Computation phase, because the attacker was able to compute $Auth_i$ based on his own choice for k_i. Hence, U_i considers k_i to be the correct group key.
We remark that U_a can mount the same attack simultaneously against multiple users to induce each one a different key. More, the adversary's identity remains hidden, which allows him to attack several times, without being detected and excluded from the group.

Step 1. U_a is qualified to participate to a protocol session and hence he computes the key:

$$k = T_a + K_a$$

Step 2. U_a intercepts the broadcast message sent in Round 3 of the protocol and prevents it from reaching $U_i, U_i \notin \{U_a, U_1\}$;

Step 3. U_a eavesdrops on K_i and computes:

$$T_i = k - K_i$$

Step 4. U_a chooses a key k_i, computes the corresponding value $K_i^{'} = k_i - T_i$ and authenticates his selection as $Auth = h(k_i, U_1, ..., U_t, r_1, ..., r_t, K_2, ..., K_i^{'}, ..., K_t)$

Step 5. U_a sends:

$$U_a \rightarrow U_i (Auth, K_2, ..., K_i^{'}, ..., K_t)$$

Step 6. U_i recovers the correct value T_i, computes the group key $k_i = T_i + K_i^{'}$ and checks that

$$Auth = h(k_i, U_1, ..., U_t, r_1, ..., r_t, K_2, ..., K_i^{'}, ..., K_t)$$

holds. U_i accepts k_i as the correct group key.

Fig. 2. Attack against Hsu et al.'s Group Key Transfer Protocol [5]

B. Attack against Harn and Lin's protocol

Fig.3 describes Harn and Lin's protocol in detail [1]. The construction is based on Shamir's scheme: the KGC selects a fresh group key k and builds a t-degree polynomial $f(x)$ such that $f(0) = k$ and $(x_i, y_i \oplus r_i)$ are on $f(x)$, where $(x_i, y_i) \in Z_n^* \times Z_n^*$ is the long-lived key U_i shares with the KGC and r_i is a random value.

Since polynomial reconstruction requires at leat $t+1$ points, an eavesdropper cannot directly find $f(0) = k$ from the public $P_1, ..., P_t$. On the other hand, a qualified user can succesfully interpolate by using his additional knowledge $(x_i, y_i \oplus r_i)$.

Initialization. KGC selects 2 large safe primes p and q (i.e. $p' = \frac{p-1}{2}$, $q' = \frac{q-1}{2}$ is also primes) and computes $n = pq$.

Users Registration. Each user $U_i, i = 1, ..., m$ shares a long-lived secret password $(x_i, y_i) \in Z_n^* \times Z_n^*$ with the KGC;

Round 1. User U_1:

1.1. sends a key generation request:

$$U_1 \rightarrow KGC : (U_1, ..., U_t)$$

Round 2. KGC:

2.1. broadcasts the list of participants as a response:

$$KGC \rightarrow^* : (U_1, ..., U_t)$$

Round 3. Each user $U_i, i = 1, ..., t$:

3.1. chooses $r_i \leftarrow^R Z_n^*$;

3.2. computes $Auth_i = h(x_i, y_i, r_i)$;

3.3. sends:

$$U_i \rightarrow KGC : (r_i, Auth_i)$$

Round 4. KGC:

4.1. checks if $Auth_i = h(x_i, y_i, r_i)$, $i = 1, ..., t$;

If at least one equality does not hold, he quits;

4.2. selects a group key $k \leftarrow^R Z_n^*$;

4.3. generates the polynomial $f(x)$ of degree t that passes through $(0, k)$ and $(x_i, y_i \oplus r_i)$, $i = 1, .., t$;

4.4. computes t additional points $P_1, ..., P_t$ on $f(x)$ and the authentification messages

$Auth = h(k, U_1, ..., U_t, r_1, ..., r_t, P_1, ..., P_t)$, $i = 1, ..., t$;

4.5. broadcasts:

$$KGC \rightarrow^* : (P_1, ..., P_t, Auth)$$

Key Computation. Each user $U_i, i = 1, ..., t$:

5.1. computes the group key $k = f(0)$ by interpolating the points $P_1, ..., P_t$ and $(x_i, y_i \oplus r_i)$;

5.2. checks if

$$Auth = h(k, U_1, ..., U_t, r_1, ..., r_t, P_1, ..., P_t)$$

If the equality does not hold, he quits.

Fig. 3. Harn and Lin's Group Key Transfer Protocol [1]

Harn and Lin's protocol was already shown susceptible to attacks that allow an adversary to disclose the session key [3,9].

We now describe a new attack that allows any qualified user to be excluded from the protocol without learning the group key or even knowing the protocol takes place. We emphasize that no participant (including the initiator and the KGC) can discover the attack during the protocol

execution because the attacker impersonates the victim by replaying in its behalf using a message from a previous session (i.e. the adversary performs a *replay attack*). An immediat attack where an adversary just prevents communication between the initiator or KGC with the victim is not enough: if the KGC does not receive a response from all users, he aborts and hence the protocol fails and the parties might request re-execution.
Fig.4 describes the attack in detail. Similar to the attack againt Hsu et al.'s protocol, the adversary's identity remains hidden.

Step 1. U_a intercepts the broadcast message sent in Round 2 of the protocol and prevents it from reaching U_i:

$$KGC \rightarrow^{*} : (U_1,...,U_t)$$

Step 2. U_a impersonates U_i and replies on his behalf in Round 3 of the protocol using a replay message U_a had eavesdropped in a previous session:

$$U_a(U_i) \rightarrow KGC : (r_i, Auth_i)$$

Step 3. KGC checks that $Auth_i = h(x_i, y_i, r_i)$ holds and therefore continues the execution of the protocol.

Step 4. U_a intercepts the broadcast message sent in Round 4 of the protocol and prevents it from reaching U_i:

$$KGC \rightarrow^{*} : (P_1,...,P_t, Auth)$$

The adversary successfully prevents U_i to compute the group key, while all parties remain unaware of the attack.

Fig. 4. Attack against the Harn and Lin's Group Key Transfer Protocol

The attack assumes that the victim had previously participate to another protocol session, a natural assumption in practice. Also, note that $Auth_i$ is independent of the set of participants to the session, which means that the adversary can replicate any message originating from U_i in a previous session, no matter the group componency, as long as the long-lived key is unchanged.

C. Attack against Yuan et al.'s protocol

Fig.5 describes Yuan et al.'s protocol [8]. The construction is very similar to Harn and Lin's, except a few differences.
A first difference is that in Yuan et al.'s proposal the long-lived key consists in a password $pw_i = pw_{ix} \parallel pw_{iy}$, not a random pair $(x_i, y_i) \in Z_n^* \times Z_n^*$.
A second difference is that in Yuan et al.'s protocol the values M_i used to authenticate the random numbers k_i depend on the group members of the current session, while in Harn and Lin's protocol $Auth_i$ is independent of the current session group.

Initialization. KGC selects 2 large primes p and q and computes $n = pq$.

Users Registration. Each user $U_i, i = 1,...,m$ shares a long-lived secret password $pw_i = pw_{ix} \parallel pw_{iy}$ with the KGC;

Round 1. User U_1:
1.1. chooses $k_1 \leftarrow^{R} Z_n$;
1.2. computes $K_1 = pw_{1x} + k_1$ and

$$M_1 = h_1(U_1,...,U_t, k_1),$$

1.3. sends a key generation request:

$$U_1 \rightarrow KGC : (U_1, \{U_1,...,U_t\}, K_1, M_1)$$

Round 2. KGC:
2.1. computes $k_1 = K_1 - pw_{1x}$;
2.2. checks if $M_1 = h_1(U_1,...,U_t, k_1)$;
If the equality does not hold, he quits.
2.3. broadcasts the list of participants as response:

$$KGC \rightarrow^{*} : (U_1,...,U_t)$$

Round 3. Each user $U_i, i = 2,...,t$:
3.1. chooses $k_i \leftarrow^{R} Z_n$
3.2. computes $K_i = pw_{ix} + k_i$ and

$$M_i = h_1(U_1,...,U_t, k_i),$$

3.3. sends:

$$U_i \rightarrow KGC : (U_i, \{U_1,...,U_t\}, K_i, M_i)$$

Round 4. KGC:
4.1. computes $k_i = K_i - pw_{ix}$;
4.2. checks if $M_i = h_1(U_1,...,U_t, k_i), i = 2,...,t$;
If at least one equality does not hold, he quits;
4.3. selects 2 random numbers x_{ta} and y_{ta} of length equal to pw_{ix} and pw_{iy};
4.4. generates the polynomial $f(x)$ of degree t that passes through (x_{ta}, y_{ta}) and $(pw_{ix}, pw_{iy} + k_i)$, $i = 1,..,t$;
4.5. computes t additional points $P_1,..., P_t$ on $f(x)$ and

the verification messages

$$V_i = h_2(U_1,...,U_t,P_1,...,P_t,k_i),\ i=1,...,t;$$

4.6. sends:

$$KGC \rightarrow U_i : (P_1,...,P_t,V_i)$$

Key Computation. Each user $U_i, i=1,...,t$:

5.1. checks if $V_i = h_2(U_1,...,U_t,P_1,...,P_t,k_i)$;

If at least one equality does not hold, he quits;

5.2. computes the group key $k = f(0)$ by interpolating the points $P_1,...,P_t$ and $(pw_{ix}, pw_{iy}+k_i)$.

Fig. 5. Yuan et al.'s Group Key Transfer Protocol [8]

Yuan et al.'s protocol was already shown susceptible to attacks that break key confidentiality [4,6].

We now show it is vulnerable to another attack, similar to the one we have exposed against Harn and Lin's protocol. The only difference is that the attack now succeeds if the adversary had previously eavesdropped on a protocol session within the same group: $M_i = h_1(U_1,...,U_t,k_i)$, thus depends on $\{U_1,...,U_t\}$; this means that if the adeversary replys with a replayed message from a session within a distinct group, the verification will not hold and the KGC detects the attack. Fig.6 describes the attack.

Step 1. U_a intercepts the broadcast message sent in Round 2 of the protocol and prevents it from reaching U_i:

$$KGC \rightarrow^{*} : (U_1,...,U_t)$$

Step 2. U_a impersonates U_i and replies on his behalf in Round 3 of the protocol using a replay message U_a had eavesdropped in a previous session run for the same group of users $\{U_1,...,U_t\}$:

$$U_a(U_i) \rightarrow KGC : (U_i,\{U_1,...,U_t\},K_i,M_i)$$

Step 3. KGC checks that $M_i = h_1(U_1,...,U_t,k_i)$ holds and therefore continues the execution of the protocol.

Step 4. U_a intercepts the broadcast message sent to U_i in Round 4 of the protocol and prevents it from reaching its destination:

$$KGC \rightarrow U_i : (P_1,...,P_t,V_i)$$

The adversary successfully prevents U_i to compute the group key, while all parties remain unaware of the attack.

Fig. 6. Attack against the Yuan et al.'s Group Key Transfer Protocol

A simple method can be used to eliminate the presented attacks – an additional *Key Confirmation* phase that certifies all authorized members actually own the correct key. For completeness only, we review in Fig.7 the Key Confirmation phase for Hsu et al.'s protocol from [5].

Key Confirmation. Each user $U_i, i=1,...,t$:

5.1. computes:

$$V_i = Sig_{U_i}(h(k,U_1,...,U_t,r_1,...,r_t,K_2,...,K_t))$$

5.2. broadcasts:

$$U_i \rightarrow^{*} : V_i$$

5.3. checks that

$$V_j = Ver_{U_j}(V_i, h(k,U_1,...,U_t,r_1,...,r_t,K_2,...,K_t))$$

holds for all $j=1,...,t,\ j \neq i$;

If at least one equality does not hold, he quits.

Fig. 7. Key Confirmation Phase for Hsu et al.'s protocol [5]

IV. CONCLUSIONS

The paper analyses three recent GKT protocols and considers some vulnerabilities that inhibit users to compute the correct key or any key at all. The attacks remain hidden throughout the execution of the protocol and can possible be detected only subsequently, when the group key is required to access resources or services.

ACKNOWLEDGMENTS

This work was supported by the strategic grant POSDRU/159/1.5/S/137750, "Project Doctoral and Postdoctoral programs support for increased competitiveness in Exact Sciences research" cofinanced by the European Social Found within the Sectorial Operational Program Human Resources Development 2007 – 2013.

REFERENCES

[1] L. Harn and C. Lin, "Authenticated group key transfer protocol based on secret sharing," IEEE Transactions on Computers, vol. 59, no. 6, pp. 842–846, 2010.

[2] C. Hsu, B. Zeng, Q. Cheng, and G. Cui, "A novel group key transfer protocol," Cryptology ePrint Archive, Report 2012/043, 2012, http://eprint.iacr.org/.

[3] J. Nam, M. Kim, J. Paik, W. Jeon, B. Lee, and D. Won, "Cryptanalysis of a group key transfer protocol based on secret sharing," in FGIT, 2011, pp. 309–315.

[4] R.F. Olimid, "Cryptanalysis of a password-based group key exchange protocol using secret sharing," Appl. Math. Inf. Sci, vol. 7, no. 4, pp. 1585–1590, 2013.

[5] R. F. Olimid, "On the vulnerability of a group key transfer protocol based on secret sharing," in 9th IEEE International Symposium on

Applied Computational Intelligence and Informatics, SACI 2014, Timisoara, Romania, May 15-17, 2014, 2014, pp. 159–163.

[6] R. F. Olimid, “A chain of attacks and countermeasures applied to a group key transfer protocol,” in International Joint Conference SOCO14-CISIS14- ICEUTE14, vol. 299.

[7] A. Shamir, “How to share a secret,” Commun. ACM, vol. 22, no. 11, pp.612–613, 1979.

[8] W. Yuan, L. Hu, H. Li, and J. Chu, “An Efficient password-group key exchange protocol using secret sharing,” Appl. Math. Inf. Sci, vol. 7, no.1, pp. 145–150, 2013.

[9] W. Yuan, L. Hu, H. Li, and J. Chu, “Security and improvement of an authenticated group key transfer protocol based on secret sharing,” Appl. Math. Inf. Sci, vol. 7, no. 5, pp. 1943–1949, 2013.