THE PUBLISHING HOUSE
OF THE ROMANIAN ACADEMY

PROCEEDINGS OF THE ROMANIAN ACADEMY, Series A,
Volume 14, Special Issue 2013, pp. 378–387

ON THE (IN)SECURITY OF 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

Group Key Transfer (GKT) protocols allow multiple parties to share a common secret key: a trusted Key Generation Center (KGC) selects a uniformly random value that has never been used before and securely distributes it to the legitimate principals. The paper restricts to GKT based on secret sharing; it briefly reviews the security goals and the existing formal security models. We motivate our work by the lack of GKT protocols based on secret sharing that are proved secure in a formal security model. Most of the recent proposals only provide informal and incomplete arguments to claim security, which makes them susceptible to known vulnerabilities. We support our affirmation by exemplifying with three different types of attacks (replay attack, insider attack and known key attack) mounted against protocols published within the last three years. We emphasize that none of these attacks would have been possible against a GKT protocol proved secure in a usual formal security model.

Key words: Group key transfer, Group key establishment, Secret sharing, Formal security model, Replay attack, Insider attack, Known key attack.

1. INTRODUCTION

Group applications have widely spread in the last years. They permit multiple users to benefit of common resources or perform collaborative tasks while they provide differentiate rights or responsibilities within the group. Group applications include text communication, audio, video or web conferences, data sharing or collaborative computing. Let's take the example of a digital conference: multiple users are simultaneously logged in the conference environment. However, they do not all communicate between themselves and should not be aware of the conversations within groups they do not belong to. Usually, the initiator of a conference is a privileged participant who can invite or exclude other parties from the meeting.

Security represents an important aspect for group applications. It is a challenging task to deal with, especially when the group size is large and the members are spread across different (location or networks) areas, with diverse protection mechanisms. In order to obtain the main cryptographic properties as *confidentiality*, *authenticity* and *integrity* it is usually required that the group members previously share a common secret group key. This is achieved as the output of a *group* (or *conference*) *key establishment protocol*.

Group Key Establishment (GKE) protocols divide into [5], [19], [20], [21]: *Group Key Transfer (GKT)* (also called *group key transport* or *group key distribution*) protocols and G*roup Key Agreement (GKA)* (also called *group key exchange*) protocols. In a GKT protocol exists a privileged party named Key Generation Center (KGC) that selects the group key and securely distributes it to the other members. In a GKA protocol each party equally contributes to the key generation, which should not be predetermined by any participant.

This paper focuses on the security of GKT protocols. Next section introduces a short description of GKT and its comparison with GKA, then restricts to GKT protocols based on secret sharing and mentions some of the recent work. Section 3 reviews the informal security goals GKT must achieve and the formal security models designed for GKE (excluding the *contributiveness* goals, which regard only GKA protocols). Section 4 summarises the adversarial and attack types GKT should stand against. Section 5 introduces three examples of attacks on recent introduced GKT protocols. Section 6 concludes.

2. PRELIMINARIES

2.1. Group Key Transfer (GKT)

GKT protocols permit multiple parties to share a common secret group key. The protocol may be executed (concurrently) for multiple times; each execution is called a *session* and it is uniquely identified by a session id. Each session permits a set of *qualified* (also called *authorized* or *legitimate*) participants from the set of registered users to establish the common key, which should remain hidden for any other party. A *registered* user is a party who has previously registered to the KGC and with whom shares a long-lived key (usually a public-secret key pair the user uses for signing).

The session key is chosen by the KGC, which must be trusted by all participants as honest in the sense that it selects a *fresh* key (a uniformly random chosen value that has never been used before). This trust assumption is not required for GKA, which do not demand the existence of a privileged party to select the key, but compute it by equal contribution of all principals. However, regardless of the GKE type, a trust relation is mandatory: the qualified participants to a session trust each other that none of them discloses the shared key. Otherwise, the confidentiality of the protocol is violated by default.

Due to the fact that parties do not necessary have to communicate between themselves (but only with the KGC), the computational and transmission costs of GKT protocols are usually lower that those of GKA protocols. Also, the design of GKT is in general less challenging.

Depending on the application needs or constraints (security requirements, computational resources and transmission costs), a GKT or a GKA protocol may suit best.

A general mechanism for defining GKT protocols is immediate [7]: KGC generates a fresh group key and sends its encrypted value under the appropriate key to each legitimate participant. Hence, any authorized user decrypts and finds the key, while it remains secure against unauthorized parties. We have assumed that an authentication mechanism exists, such that the KGC or the users cannot be impersonated and the message cannot be modified during transmission. This trivial solution becomes inefficient for large groups: KGC must perform m encryptions and send m messages, where m is the number of qualified participants. In case a symmetric encryption scheme is used to decrease the computational costs (rather than an asymmetric encryption scheme), a supplementary assumption appears: each registered group member must previously share a secret with the KGC.

Secret sharing is used in GKE protocols to avoid such disadvantages; in addition, they introduce several benefits: a convenient way to differentiate between principals power within the group, delegation of duties by passing shares to other participants, group authentication instead of entity authentication, cheating detection and simple management of group sizing using the accepted threshold [26]. Next section briefly reminds secret sharing.

2.2. Secret Sharing

Blackley [4] and Shamir [27] independently introduced secret sharing as a solution to the key management problem. It represents a mechanism that splits a secret into multiple shares such that the secret may be recovered only from authorized sets of shares. In general, a secret sharing scheme consists of three phases: *sharing* (a dealer splits the secret into multiple parts, called *shares*), *distribution* (the dealer securely sends one or more shares to each party) and *reconstruction* (a qualified set of parties combine their shares to recover the secret).

The set of all authorized sets of shares in called the *access structure*. A secret sharing scheme whose access structure contains all sets with at least t out of m shares is called (t,m) *threshold secret sharing scheme*; a secret sharing scheme whose access structure contains only the set of all shares is called *all-or-nothing* (or *unanimous*) secret sharing scheme.

Various GKE protocols based on secret sharing exist in the literature. We only remind here some of the recent work in the field of GKT protocols: Harn and Lin's protocol based on Shamir's scheme (2010) [15], Hsu *et al.*'s protocol based on linear secret sharing (2012) [16], Sun *et al.*'s protocol based on derivative secret sharing (2012) [28] and Yuan *et al.*'s protocol based on Shamir's scheme (2013) [30]. We highlight

that even though all mentioned protocols were published in the last three years, none of them gives a security proof in a formal model, but rather incomplete and informal arguments of security. This leads to a high opportunity to reveal vulnerabilities. We emphasize that the existing work in the field of GKE protocol based on secret sharing scheme with formal security proofs is very limited.

3. SECURITY DEFINITIONS AND MODELS

3.1. Informal Security Definitions

A GKT protocol can be considered secure if it achieves the properties that we informally recall next [19], [20]. We restrict to security goals that apply to GKT and do not remind specific properties of GKA, which are beyond the scope of this paper (such as *key contributiveness* or *key control*).

Key confidentiality (also called *key privacy*, *key secrecy* or *non-disclosure*) [7], [12], [13], [17] guarantees that it is (computationally) infeasible for an adversary to compute the group key. The stronger notion of *known key security* [6], [29] assures that key confidentiality is maintained even if the attacker somehow manages to obtain group keys of previous sessions. *Backward secrecy* [11], [20] generalizes this concept and conserves the privacy of future keys regardless the adversary's actions in the past sessions. Correspondingly, *forward secrecy* [11], [13] imposes that the adversary actions in future runs of the protocol do not compromise the privacy of previous session keys (*i.e.* a key remains secure in the future).

Key selection must satisfy specific properties. *Key freshness* [21] requires that the group key is new (*i.e.* it has never been used before). The related concept of *key independence* [17], [23] imposes that no correlation exists between keys from different sessions; this means that (cooperation between) authorized participants to distinct sessions of the protocol cannot disclose session keys they are unauthorized for. In addition, *key randomness* warrants *key indistinguishability* from a random number and hence *key unpredictability*. Two other important security requirements regarding the key value exists: *key integrity* [17], which attests that no adversary can modify the group key and *key consistency,* which prevents different players to accept different keys.

Group member authentication represents a mandatory condition for group cryptographic protocols. *Entity authentication* [1] confirms the identity of a participant to the others. Similarly, *unknown key share resilience* [13] restricts a user to believe that the key is shared with one party when in fact it is shared with another. *Key compromise impersonation resilience* [3], [14] prevents an attacker who owns the long-lived key of a participant to impersonate other parties to him. The stronger property named *ephemeral key leakage resilience* [31] avoids an adversary to recover the group key even if he discloses the long-lived keys and ephemeral keys of parties involved except both these values for the participants in the test session.

(Implicit) Key authentication [21] limits the possible owners of the group key to the legitimate participants; this means that no other party except the qualified users is capable to compute the key, but it does not necessary mean that all legitimate principals actually own it. Another property, called *key confirmation* [5], [21] certifies that all authorized members actually have the key; however, it does not claim that no other party own the same key. *Explicit key authentication (or Mutual Authentication)* [2], [8], [11], [21] combines these notions and ensures that all qualified participants to the protocol have actually computed the group key and no one else except them has.

For more information on informal security requirements, we invite the reader to refer to [19] and [20].

3.2. Formal Security Models

Security models formalize the properties described in the previous subsection within a precise environment, specifying the trust assumptions, the relations between participants, the adversarial power, the communication medium and others relevant aspects. Similar to the informal definitions mentioned before, GKE security models were developed as a generalization of the existing two or three party security models.

Bresson *et al.* introduced the first security model for GKE protocols in 2001 [10]. Their model was rapidly extended to allow dynamic groups, meaning that the group membership may change during the protocol execution [8]. One year later, the same authors improved their model to stand against strong

corruptions, which permit an attacker to reveal the ephemeral internal state information of the user instances [9]. In 2005, Katz and Shin considered the existence of malicious users, in the sense that even if a registered party always knows the key of sessions he is qualified for, he must be restricted to perform malicious actions [18]; for example, he should not disclose session keys he is unauthorized for, modify the value of the common key as he desires or make honest users compute different keys. Recently, stronger security notions were considered: Gorantla, Boyd and González Nieto introduced in 2009 a model that deals with key compromise impersonation [14], while Zhao *et al.* enhanced it in 2011 to achieve ephemeral key leakage resilience [31]. For an extensively survey on group key security models, the reader may address to [19] and [20].

We skip the formal definitions of the security models, as they do not represent the goal of this paper. However, we briefly remind that they rely on two main security requirements known as *AKE-Security* (*Authenticated Key Exchange Security*) and *MA-Security* (*Mutual Authenticated Security*). AKE-Security's main objective is to prevent unqualified members to reveal the common group key; it aims informal security notions such as key confidentiality, forward and backward secrecy, known key security, key compromise impersonation or ephemeral key leakage resilience. MA-Security's primary goal is to make users aware of the correct identity of the other parties and compute identical keys at the end of the protocol; it unifies informal concepts such as (implicit) key authentication, key confirmation or unknown key share resilience. We bypass the notion of *key contributiveness*, which only applies to GKA protocols.

Although much research has been done in the last years in the field, the security models still have some limitations concerning the adversarial capabilities. For example, the existing models do not deal with DoS (Denial of Service) or fault tolerance [11]. Protocols remain vulnerable to this kind of attacks, even if they are proved to be secure in formal security models.

4. ADVERSARIES AND ATTACKS CLASSIFICATION

A secure GKT protocol must stand against *passive* and *active* adversaries. A passive adversary can only eavesdrop on the communication channel, while an active adversary has full control over the network (he can drop, modify or insert messages). It is immediate that an active attack is more powerful than a passive attack and therefore active attacks should be considered when formally analysing the security of a group key protocol.

Regarding the appartenance to the group, attackers split into: *outsiders* and *insiders*. An outsider is a party that has not registered as a group member (and hence does not posses a valid long-lived key within the group) and never takes part to the key establishment as a legitimate participant. An outsider attack aims to reveal the established group key and therefore to break the AKE-Security of the protocol, usually by impersonating authorized users. An insider is a valid group member, who has registered within the group at a given moment and therefore has the advantage to posses a long-lived key. He is qualified to compute session keys he is authorized for, but this should provide him no advantage in revealing other keys (of sessions he is unqualified for) or damage the protocol in any other way: find the long-term keys of other users, ruin key consistency or get control over the key value. Of course, insiders are more powerful than outsiders, because they have access to additional information. Within the definition of formal security models, an adversary is not considered to be a malicious user, but an external attacker who has access to the long-lived keys and/or ephemeral values used during the run of the protocol by making queries [11]. We illustrate an insider attack [25] on Yuan *et al.*'s protocol [30] later in this paper.

Impersonation attacks try to make messages originating from the adversary indistinguishable from messages originating from legitimate users (the adversary pretends to be a qualified group member). This may result in computing a different key than the genuine one or in establishing a common key with an attacker instead of an authorized user. A successful impersonation attack can for example break entity authentication, unknown key share resilience or key compromise impersonation resilience. A special kind of attack is the *replay attack* that consists in injecting messages from previous executions of the protocol. This can turn into a *key replication attack*, where the same (corrupted) key is derived for multiple runs of the protocol. It is immediate that a key replication attack violates key freshness. We review a replay attack [22] on Harn and Lin's construction [15] in Section 5.

Known key attacks aim to disclose a session key when the adversary knows at least one key from a previous run of the protocol. All insiders satisfy the assumption by default, as they may legitimate take part to protocol executions. We exemplify a known key attack [24] on Sun *et al.*'s proposal [28] in the next section.

Attacks can be classified based on the information the adversary has access to: the long-lived key of the registered users or the ephemeral secrets used during protocol execution. *Opening attacks* allow the attacker to learn the ephemeral secrets without revealing the long-lived secrets, while *weak corruption attacks* allow the attacker to learn the long-lived secrets without revealing ephemeral secrets. *Strong corruption attacks* combine these two attacks and give the adversary tremendous power: he can corrupt a user and obtain both his long-lived secret and his ephemeral secret values [11].

DoS (Denial of Service) attacks lead to the futility of GKT protocols: they prevent legitimate users to establish common secret keys that they would have later use for application purposes. A DoS attack may inhibit users to compute any key at all or may force users to end up with different keys. Although the attack is discovered at the latest during the execution of the application (the group members realise that they cannot properly communicate between themselves) it represents an important aspect of network security. In contrast to the previously mentioned attacks, we highlight that the DoS attacks are not considered within the existing formal security models for GKE protocols [11].

In formal security models, the adversary is modelled as a PPT (Probabilistic Polynomial Time) algorithm with full control over the communication channel (hence active in the sense that it can inject, delete or modify exchanged messages). He interacts with the legitimate group members by asking queries with the scope to reveal information that he may use in breaking the AKE-Security or MA-Security. For example, he makes Corrupt queries to obtain the long-lived key of registered participants, RevealState queries to retrieve ephemeral secret used by parties during the protocol execution or RevealKey queries to retrieve the common shared key of particular sessions. Each security model defines the queries an attacker is allowed to make (which model the power of the attacker) as well as the security game he plays against the AKE-Security or MA-Security of the protocol (which he breaks if he wins the game with non-negligible probability). We will skip the formal definitions of particular formal models, but strongly invite the reader to address some of the recent original papers [11], [14], [31] or surveys [19], [20].

5. ATTACKS ON RECENT GKT PROTOCOLS

The current section describes three types of attacks against recent GKT protocols: a replay attack on Harn and Lin's protocol [15] introduced by Nam *et al.* [22], an insider attack on Yuan *et al.*'s proposal [30] defined by Olimid [25] and a known key attack on Sun *et al.*'s construction [28] revealed by Olimid [24]. We emphasize that all mentioned protocols lack a formal security proof and hence the vulnerabilities arise natural.

For the rest of the paper, we will use the following notations: m the number of possible users, $\{U_1,...,U_t\}, t \le m$ the set of participants to a given session (after reordering) with U_1 as initiator, h, h_1 and h_2 collision-resistant hash functions, $\leftarrow^R$ a random choice from a specified set of values, $||$ string concatenation, $(s_j), j = 1..4$ specific protocol sessions.

Let U_a be the attacker. His main goal is to break the AKE-Security or MA-Security of the protocol. U_a may be an insider ($U_a \in \{U_1,...,U_m\}$) or an outsider ($U_a \notin \{U_1,...,U_m\}$), depending the adversarial scenario.

We proceed with the description of the protocols and the attacks. For more details, we invite the reader to address the original papers.

Protocol 1. Harn and Lin [15]

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

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

Round 1. User U_1 sends a key generation request:

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

Round 2. KGC broadcasts the list of participants as a response:

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

Round 3. Each user $U_i, i = 1..t$ chooses $r_i \leftarrow^R Z_n^*$, computes $Auth_i = h(x_i, y_i, r_i)$ and sends:

$$U_i \to KGC : (r_i, Auth_i).$$

Round 4. KGC checks if $Auth_i = h(x_i, y_i, r_i), i = 1..t$ (otherwise he aborts), selects a group key $k \leftarrow^R Z_n^*$, generates the polynomial $f(x)$ of degree t that passes through $(0,k)$ and $(x_i, y_i \oplus r_i), i = 1..t$, computes t additional points $P_1,...,P_t$ on $f(x)$ and the authentication message $Auth = h(k, U_1,...,U_t, r_1,...,r_t, P_1,...,P_t)$, then broadcasts:

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

Key Computation. Each user $U_i, i = 1..t$ computes the group key $f(0)$ by interpolating the points $P_1,..,P_t$ and $(x_i, y_i \oplus r_i)$ and checks if $Auth = h(k, U_1,...,U_t, r_1,...,r_t, P_1,...,P_t)$ (otherwise he aborts).

Attack 1. Replay Attack [22]

Attack scenario. U_a is an insider whose goal is to break the AKE-Security of Protocol 1: he obtains the key of any session a user $U_i \in \{U_1,..,U_{a-1},U_{a+1},..,U_m\}$ is qualified for (even if U_a is unqualified for) by disclosing the long-lived secret of U_i.

Step 1. U_a eavesdrops $(r_i, Auth_i)$ from a session U_i is qualified for.

Step 2. U_a initiates two legitimate sessions of the protocol with U_i, denoted by $(s_j), j = 1,2$.

Step 3. In both sessions, U_a impersonate the legitimate user U_i by sending the eavesdropped message $(r_i, Auth_i)$ and behaves honestly in sending his own message $(r_a, Auth_a)$.

Step 4. U_a recovers the coefficients of the polynomials $f(x)_{(s_j)} = a_{(s_j)}x^2 + b_{(s_j)}x + c_{(s_j)}, j = 1,2$, as being a qualified participant for sessions $(s_j), j = 1,2$.

Step 5. As $(x_i, y_i \oplus r_i)$ and $(x_a, y_a \oplus r_a)$ are valid points on $f(x)_{(s_j)}, j = 1,2$, x_i and x_a are two roots of the quadratic equation:

$$(a_{(s_1)} - a_{(s_2)})x^2 + (b_{(s_1)} - b_{(s_2)})x + c_{(s_1)} - c_{(s_2)} = 0. \quad (1)$$

Step 6. U_a computes the secret key of U_i as:

$$\begin{aligned} x_i &= x_a^{-1}(a_{(s_1)} - a_{(s_2)})^{-1}(c_{(s_1)} - c_{(s_2)}) \\ y_i &= f(x_i)_{(s_1)} \oplus r_1 = f(x_i)_{(s_2)} \oplus r_2. \end{aligned} \quad (2)$$

Step 7. U_a discloses all keys of the sessions U_i is qualified for by using the long-lived secret (x_i, y_i).

Protocol 2. Yuan *et al.* [30]

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

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

Round 1. User U_1 chooses $k_1 \leftarrow^R Z_n$, computes $K_1 = pw_{1x} + k_1$ and $M_1 = h_1(U_1,...,U_t, k_1)$, then sends a key generation request:

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

Round 2. KGC computes $k_1 = K_1 - pw_{1x}$, checks if $M_1 = h_1(U_1,..,U_t,k_1)$ (otherwise he aborts) and broadcasts the list of participants as a response:

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

Round 3. Each user $U_i, i = 2..t$ chooses $k_i \leftarrow^R Z_n$, computes $K_i = pw_{ix} + k_i$ and $M_i = h_i(U_1,..,U_t,k_i)$, then sends:

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

Round 4. KGC computes $k_i = K_i - pw_{ix}$, checks if $M_i = h_i(U_1,..,U_t,k_i), i = 2,..,t$ (otherwise he aborts), selects 2 random numbers x_{ta} and y_{ta} of lengths equal to pw_{ix} and pw_{iy}, generates the polynomial $f(x)$ of degree t that passes through (x_{ta}, y_{ta}) and $(pw_{ix}, pw_{iy} + k_i), i = 1..t$, computes t additional points $P_1,...,P_t$ on $f(x)$ and the verification messages $V_i = h_2(U_1,...,U_t,P_1,..,P_t,k_i), i = 1..t$, then sends:

$$KGC \to^* : (P_1,...,P_t,V_i).$$

Key Computation. Each user $U_i, i = 1..t$ checks if $V_i = h_2(U_1,...,U_t,P_1,..,P_t,k_i)$ (otherwise he aborts) and computes the group key $k = f(0)$ by interpolating the points $P_1,...,P_t$ and $(pw_{ix}, pw_{iy} + k_i)$.

Attack 2. Insider Attack [25]

Attack scenario. U_a is an insider whose goal is to break the AKE-Security of Protocol 2: he obtains the key of any session a user $U_i \in \{U_1,..,U_{a-1},U_{a+1},..,U_m\}$ is qualified for (even if U_a is unqualified for) by disclosing the long-lived secret password of U_i.

Step 1. U_a initiates four legitimate sessions of the protocol with U_i, denoted by $(s_j), j = 1..4$.

Step 2. U_a recovers the coefficients of the polynomials $f(x)_{(s_j)} = a_{(s_j)}x^2 + b_{(s_j)}x + c_{(s_j)}, j = 1..4$, as being a qualified participant for sessions $(s_j), j = 1..4$.

Step 3. U_a eavesdrops the values $K_{i(s_j)}, j = 1..4$; as $(pw_{ix}, pw_{iy} + k_{i(s_j)})$ are valid points on $f(x)_{(s_j)}, j = 1..4$ and $K_{i(s_j)} = pw_{ix} + k_{i(s_j)}$, U_a obtains:

$$pw_{iy} = a_{(s_j)} pw_{ix}^{\ 2} + (b_{(s_j)} + 1)\, pw_{ix} + c_{(s_j)} - K_{i(s_j)}, j = 1..4. \tag{3}$$

Step 4. U_a eliminates pw_{iy} from the first two equations ($j = 1,2$), respectively from the last two equations ($j = 3,4$) in (3) and obtains:

$$\begin{cases} A_{(s_1,s_2)} pw_{ix}^{\ 2} + B_{(s_1,s_2)}\, pw_{ix} + C_{(s_1,s_2)} = 0 \\ A_{(s_3,s_4)} pw_{ix}^{\ 2} + B_{(s_3,s_4)}\, pw_{ix} + C_{(s_3,s_4)} = 0, \end{cases} \tag{4}$$

where:

$$\begin{aligned} A_{(s_1,s_2)} &= a_{(s_1)} - a_{(s_2)} & A_{(s_3,s_4)} &= a_{(s_3)} - a_{(s_4)} \\ B_{(s_1,s_2)} &= b_{(s_1)} - b_{(s_2)} \quad ; & B_{(s_3,s_4)} &= b_{(s_3)} - b_{(s_4)} \\ C_{(s_1,s_2)} &= c_{(s_1)} - c_{(s_2)} - (K_{i(s_1)} - K_{i(s_2)}) & C_{(s_3,s_4)} &= c_{(s_3)} - c_{(s_4)} - (K_{i(s_3)} - K_{i(s_4)}) \end{aligned} \tag{5}$$

Step 4. U_a computes the secret key of U_i as:

$$pw_{ix} = (A_{(s_1,s_2)}C_{(s_3,s_4)} - A_{(s_3,s_4)}C_{(s_1,s_2)})(A_{(s_3,s_4)}B_{(s_1,s_2)} - A_{(s_1,s_2)}B_{(s_3,s_4)})^{-1}$$
$$pw_{iy} = f(pw_{ix})_{(s_j)} + pw_{ix} - K_{i(s_j)}, j = 1..4. \quad (6)$$

Step 5. U_a discloses all group keys of the sessions the user U_i is qualified for by using the long-lived password $pw_i = pw_{ix} \| pw_{iy}$.

Protocol 3. Sun *et al.* [28]

Initialization. KGC selects a multiplicative cyclic group G of prime order p with g as generator.

Users Registration. Each user $U_i, i = 1..m$ shares a long-lived secret $s_i' \in G$ with the KGC.

Round 1. User U_1 sends a key generation request:

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

Round 2. KGC broadcasts the list of participants as a response:

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

Round 3. Each user $U_i, i = 1..t$ chooses $r_i \leftarrow^R Z_p^*$ and sends it to the KGC:

$$U_i \to^* r_i.$$

Round 4. KGC selects a value $S \leftarrow^R G$, invokes the derivative secret sharing scheme to split S into 2 parts t times such that $S = s_i + s_i'$ (in G), $i = 1,..,t$, computes the session group key as $k = g^S$ (in G), t messages $M_i = (g^{s_i+r_i}, U_i, h(U_i, g^{s_i+r_i}, s_i', r_i)), i = 1..t$ and $Auth = h(k, g^{s_1+r_1},..., g^{s_t+r_t}, U_1,...,U_t, r_1,..., r_t)$, then broadcasts:

$$KGC \to^* : (M_1,..., M_t, Auth).$$

Key Computation. Each user $U_i, i = 1..t$ checks if $h(U_i, g^{s_i+r_i}, s_i', r_i)$ equals the corresponding value in M_i (otherwise he aborts), computes $k = g^{s_i'} g^{s_i+r_i} (g^{r_i})^{-1}$, checks if $Auth = h(k, g^{s_1+r_1},..., g^{s_t+r_t}, U_1,...,U_t, r_1,..., r_t)$ (otherwise he aborts), accepts k as the group key and sends:

$$U_i \to KGC : h_i = h(s_i', k, U_1,...,U_t, r_1,..., r_t).$$

Key Confirmation. KGC checks if $h_i = h(s_i', k, U_1,...,U_t, r_1,..., r_t)$ using his knowledge on s_i' and k, certifying that all users posses the same key.

Attack 3. Known Key Attack [24]

Attack scenario. U_a is an adversary who knows the key $k_{(s_1)}$ of a session (s_1) and his goal is to break the AKE-Security of Protocol 3: he obtains the key $k_{(s_2)}$ of another session (s_2) (he is unauthorized for) that has at least one common qualified participant U_i with (s_1).

Step 1. U_a eavesdrops $r_{i_{(s_j)}}$ and $g^{s_{i(s_j)}+r_{i(s_j)}}, j = 1,2$ and computes:

$$g^{s_{i(s_j)}} = g^{s_{i(s_j)}+r_{i(s_j)}} (g^{r_{i(s_j)}})^{-1}, j = 1,2. \quad (7)$$

Step 2. U_a knows the session key $k_{(s_1)}$ and uses the previously computed value $g^{s_{i(s_1)}}$ to obtain:

$$g^{s_i'} = k_{(s_1)} (g^{s_{i(s_1)}})^{-1} \quad (8)$$

Step 3. U_a recovers the session key $k_{(s_2)}$ from (7) and (8) as:

$$k_{(s_2)} = g^{s_{i(s2)}} g^{s_i} \tag{9}$$

6. CONCLUSIONS

The paper considers GKT protocols based on secret sharing schemes. It provides a brief survey on the informal security notions that GKT protocols should satisfy and reviews the adversarial models and types of attacks that GKT protocols should stand against. Security models formalize such requirements within a specific environment. We highlight the necessity of electing a protocol secure within a suitable model, considering the best trade-off between efficiency and the required security level for each application.

Unfortunately, current work on GKT protocols based on secret sharing neglect this aspect. Recent published protocols ignore formal security and only rely on incomplete and informal arguments. This makes them susceptible to known vulnerabilities that would have been excluded by a proper proof under a convenient security model. In order to support our claim, we recall three different types of attacks against GKT protocols using secret sharing that were published within the last three years: a replay attack, an insider attack and a known key attack. We mention that all these attacks are considered, and therefore impossible to succeed, in all usual security models.

Although much work has been done in the field of formal security lately, we must admit some limitations. The GKE formal security models do not deal with all known vulnerabilities. For example, they do not stand against DoS attacks. Therefore, even if a GKT protocol is proved secure in a strong security model, it can become useless if participants are restricted to compute the session key. This can be easily achieved by blocking the messages to arrive to the qualified participants: since a user misses mandatory information, he can no longer recover the session key and therefore halts. We admit that DoS attacks are probably impossible to handle only by cryptographic techniques and therefore require proper protection mechanisms.

To conclude, we highlight the problems a protocol without a formal security proof may raise and emphasize the deficiency of provable secure GKT protocols based on secret sharing in the literature, which we consider a direction for further research.

ACKNOWLEDGEMENTS

This paper is supported by the Sectorial Operational Program Human Resources Development (SOP HRD), financed from the European Social Fund and by the Romanian Government under the contract number SOP HDR/107/1.5/S/82514.

REFERENCES

1. M. Bellare, P. Rogaway, *Entity Authentication and Key Distribution,* In Advances in Cryptology (CRYPTO'93), pp. 232–249, 1993.
2. M. Bellare, P. Rogaway, *Random Oracles are Practical: A Paradigm for Designing Efficient Protocols*, In Proceedings of the 1st ACM Conference on Computer and Communications Security (CCS'93), pp. 62–73, 1993.
3. S. Blake-Wilson, D. Johnson, A. Menezes, *Key Agreement Protocols and Their Security Analysis*, In Proceedings of the 6th IMA International Conference on Cryptography and Coding, pp. 30–45, 1997.
4. G. Blakley, *Safeguarding Cryptographic Keys,* In Proceedings of the AFIPS'79, pp. 313–317, 1979.
5. C. Boyd, *On Key Agreement and Conference Key Agreement*, In Information Security and Privacy: Australasian Conference, pp. 294–302, 1997.
6. M. Burmester, *On the Risk of Opening Distributed Keys*, In Advances in Cryptology (CRYPTO'94), pp.308–317, 1994.
7. M. Burmester, Y. Desmedt, *A Secure and Efficient Conference Key Distribution System,* In Proceedings of EUROCRYPT'94, pp. 275–286, 1994.
8. E. Bresson, O. Chevassut, D. Pointcheval, *Provably Authenticated Group Diffie-Hellman Key Exchange - The Dynamic* Case, In Proceedings of ASIACRYPT'01, pp. 290–309, 2001.
9. E. Bresson, O. Chevassut, D. Pointcheval, *Dynamic Group Diffie-Hellman Key Exchange under Standard Assumptions*, In Proceedings of EUROCRYPT'02, pp. 321–336, 2002.

10. E. Bresson, O. Chevassut, D. Pointcheval, J.J. Quisquater, *Provably Authenticated Group Diffie-Hellman Key Exchange*, In Proceedings of the 8th ACM Conference on Computer and Communications Security (CCS'01), pp. 255–264, 2001.
11. E. Bresson, M. Manulis, *Securing Group Key Exchange against Strong Corruptions*, In Proceedings of ASIACSS'08, pp. 249–260, 2008.
12. W. Diffie, M.E. Hellman, *New Directions in Cryptography*, IEEE Transactions on Information Theory, IT-**22**(6), pp. 644–654, 1976.
13. W. Diffie, P.C. van Oorschot, M.J. Wiener, *Authentication and Authenticated Key Exchanges*, Designs, Codes and Cryptography, **2**(2), pp. 107–125, 1992.
14. M.C. Gorantla, C. Boyd, J.M. González Nieto, *Modeling Key Compromise Impersonation Attacks on Group Key Exchange Protocols,* In Proceedings of Public Key Cryptography (PKC'09), pp. 105–123, 2009.
15. L. Harn, C. Lin, *Authenticated Group Key Transfer Protocol based on Secret Sharing*, IEEE Trans. Comput. **59**(6), pp. 842–846, 2010.
16. C. Hsu, B. Zeng, Q. Cheng, G. Cui, *A Novel Group Key Transfer Protocol*, Cryptology ePrint Archive, Report 2012/043, 2012.
17. P. JANSON, G. TSUDIK, *Secure and Minimal Protocols for Authenticated Key Distribution*, Computer Communications, **18**(9), pp. 645–653, 1993.
18. J. Katz, J.S. Shin, *Modeling Insider Attacks on Group Key-Exchange Protocols*, In Proceedings of the 12th ACM Conference on Computer and Communications Security (CCS'05), pp. 180–189, 2005.
19. M. Manulis, *Survey on Security Requirements and Models for Group Key Exchange*, Technical Report 2006/02, 2006.
20. M. Manulis, *Provably Secure Group Key Exchange*, European University Press, 2007.
21. A. Menezes, P. van Oorschot, S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1996.
22. J. Nam, M. Kim, J. Paik, W. Jeon, B. Lee, D. Won, *Cryptanalysis of a Group Key Transfer Protocol based on Secret Sharing*, In Proceedings of the 3rd International Conference on Future Generation Information Technology, pp. 309–315, 2011.
23. M. Steiner, G. Tsudik, M. Waidner, *CLIQUES: A New Approach to Group Key Agreement,* In Proceedings of the 18th International Conference on Distributed Computing Systems (ICDCS'98), pp. 380–387, 1998.
24. R.F. Olimid, *On the Security of an Authenticated Group Key Transfer Protocol Based on Secret Sharing*, In Proceedings of ICT-EurAsia 2013 (AsiaARES 2013), pp. 399–408, 2013.
25. R.F. Olimid, *Cryptanalysis of a Password-based Group Key Exchange Protocol Using Secret Sharing*, Appl. Math. Inf. Sci.**7**(4), pp. 1585–1590, 2013.
26. J. Pieprzyk, C.H. Li, *Multiparty Key Agreement Protocols*, In IEEE Proceedings - Computers and Digital Techniques, pp. 229–236, 2000.
27. A. Shamir, *How to Share a Secret*, Commun. ACM **22**(11), pp. 612–613, 1979.
28. Y. Sun, Q. Wen, H. Sun, W. Li, Z. Jin, H. Zhang, *An Authenticated Group Key Transfer Protocol based on Secret Sharing,* Procedia Engineering **29**(0), pp. 403–408, 2012.
29. Y. Yacobi, Z. Shmuely, *On Key Distribution Systems*, In Advances in Cryptology (CRYPTO'89), pp. 344–355, 1990.
30. W. Yuan, L. Hu, H. Li, J. Chu, *An Efficient Password-based Group Key Exchange Protocol Using Secret Sharing*, Appl. Math. Inf. Sci. **7**(1), pp. 145–150, 2013.
31. J. Zhao, D. Gu, M.C. Gorantla, *Stronger Security Model of Group Key Agreement*, In Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security (ASIACCS'11), pp. 435–440, 2011.

Received June 1, 2013