Provable Secure Constant-Round Group Key Agreement Protocol based on Secret Sharing

Ruxandra F. Olimid

Department of Computer Science, University of Bucharest, Romania
ruxandra.olimid@fmi.unibuc.ro

Abstract. Group Key Agreement (GKA) allows multiple users to collaboratively compute a common secret key. Motivated by the very few existing GKA protocols based on secret sharing with formal security proofs, we propose a new method to build such protocols. We base our construction on *secret n-sharing*, an untraditional perspective of secret sharing that brings several advantages. Our proposal achieves better security than the existing work while it maintains a constant number of communication rounds regardless the group size.

Keywords: group key agreement, secret sharing, provable security.

1 Introduction

Besides popular examples like chat, digital conferences or file sharing, group applications have rapidly grown as (computational intelligent) distributed systems: grids, distributed artificial intelligence, collaborative problem solving, multi-agent systems, peer-to-peer networks. Reliable communication (as secure conversation between agents) represents a critical aspect of group applications, which may rely on a private common key obtained by all qualified participants to a Group Key Establishment (GKE) protocol. GKE divides into *Group Key Transfer* (GKT) - a privileged party selects the key and securely distributes it to the other members - and *Group Key Agreement* (GKA) - all participants collaborate to compute the key. The current work restricts to GKE based on secret sharing.

1.1 Related work

Secret Sharing Schemes. Secret sharing was introduced by Blakley [1] and Shamir [18] as a solution to backup cryptographic keys. From the multitude of existing work we recall a single secret sharing scheme, which we will later use in this paper: Karnin et al.'s scheme that permits secret reconstruction by performing a sum modulo a prime [11]. Traditionally in the literature, each participant receives a share and the secret can be recovered only when an authorized set of shares belonging to distinct users are combined together. Recently, Sun et al. proposed a different approach: to split the same secret multiple times and give each user a qualified set of shares (under the appropriate circumstances) [19]. Although their construction is insecure [14], we show that their idea can be successfully used to build strong GKE protocols.

GKE Protocols based on Secret Sharing. Pieprzyk and Li showed the benefits of using secret sharing in GKE protocols and gave a couple of examples based on Shamir's scheme [16]. Recently, Harn and Lin [9] and Yuan et. al. [20] also used Shamir's scheme for GKT. Some other examples from the current literature include: Sáez's protocol [17] (based on a family of vector space secret sharing schemes), Hsu et al.'s protocol [10] (based on linear secret sharing schemes) and Sun et al.'s protocol [19] (based on the different approach on secret sharing we have previously mentioned). We remind Bresson and Catalano [2] and Cao et al.'s [7] as protocols based on secret sharing with formal security proofs.

Provable Secure GKE. The first security model for GKE (BCPQ) [5] represents a generalization of the models designed for two or three party protocols. It was further improved to allow dynamic groups (BCP) [3] and strong corruption (BCP+) [4]. Katz and Shin were the first to consider the existence of malicious participants within the Universally Composability (UC) framework [12]. At PKC'09, Gorantla, Boyd and Nieto described a stronger model (GBG[1]) that stands against *key compromise impersonation* (KCI) attacks [8]. Two years later, Zhao et al. extended their model (eGBG) and considered *ephemeral key leakage* (EKL) to derive the session key [21].

Unfortunately, recent GKE protocols based on secret sharing lack formal security proofs and hence become susceptible to vulnerabilities: Nam et al. revealed a replay attack on Harn et Lin's protocol [13], Olimid exposed an insider attack and a known key attack on Sun et al.'s construction [14] and Olimid reported an insider attack on Yuan et al.'s scheme [15].

1.2 Our contribution

Although secret sharing brings several advantages as a building block of GKE, not much research has been done on the formal security of such protocols. This motivates our work: we give a method to build secure GKA protocols based on secret sharing in the GBG model [8]. To the best of our knowledge, no other construction was proved secure in a similar or stronger security model. In addition, our protocol maintains a constant number of rounds regardless the number of participants and its performance is comparable with the existing work.

We base our protocol on the untraditional perspective on secret sharing inspired by the work of Sun et al. [19], which we call *secret n-sharing.* We define the notion of *perfect secret n-sharing* as a natural generalization of perfect secret sharing.

Secret n-sharing brings several advantages to GKE protocols: users communicate through broadcast channels only, key computation is efficient, key confirmation is achieved by default, the number of rounds remains constant regardless of the group size, each participant restores the key from his own shares.

[1] We adopt the notation from [21].

2 Preliminaries

2.1 Background

We assume that the reader is familiar with the following notions, but (informally) remind them to introduce the notations that we will use for the rest of the paper.

Let $\{U_1, \ldots, U_n\}$ be the set of n users that may take part in the GKE protocol. We denote by (pk_i, sk_i) the public-private key pair an user U_i, $i = 1 \ldots n$ uses for signing under a signature scheme $\Sigma = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$, where: $\mathsf{G}(1^k)$ is a randomized algorithm that on input a security parameter k outputs a public-private key pair (pk_i, sk_i); $\mathsf{Sign}(sk_i, \cdot)$ is a randomized signing algorithm under the private key pair sk_i; $\mathsf{Verify}(pk_i, \cdot, \cdot)$ is a deterministic algorithm that outputs 1 for a valid message-signature pair and 0 otherwise. We require that Σ is unforgeable under chosen message attack (UF-CMA) and denote by $\mathsf{Adv}_{\mathcal{A},\Sigma}^{\mathsf{UF-CMA}}$ the advantage of an adversary $\mathcal{A}$ to win the UF-CMA game.

Let $\mathcal{F} = (\mathsf{G}, \mathsf{F}, \mathsf{F}^{-1})$ be a trapdoor function, where: $\mathsf{G}(1^k)$ is a randomized algorithm that on input a security parameter k outputs a public-private key pair (pk, sk); $\mathsf{F}(pk, \cdot)$ is a deterministic function depending on pk; $\mathsf{F}^{-1}(sk, \cdot)$ inverts $\mathsf{F}(pk, \cdot)$. We require that $\mathcal{F}$ is secure and denote by $\mathsf{Adv}_{\mathcal{A},\mathcal{F}}$ the advantage of an adversary $\mathcal{A}$ to invert $\mathsf{F}(pk, \cdot)$ without the knowledge of sk.

Let $\mathsf{H} : \{0,1\}^l \to \{0,1\}^k$ be a collision resistant hash function modeled as a random oracle and q_r the maximum number of possible queries to H.

2.2 GBG Model

In the GBG model [8], each user U has multiple instances (*oracles*) denoted by Π_U^s, where U participates in the s^{th} run of the protocol (*session*) that is unique identified by the *session id* sid_U^s. Let q_s be the upper bound for the number of sessions. Every instance Π_U^s computes a session key K_U^s and enters an *accepted* state or terminates without computing a session key. The *partner id* pid_U^s of an oracle Π_U^s is the set of parties to whom U wishes to establish a common session key, including himself. Each party U_i, $i = 1, \ldots, n$ owns a public-private *long-term key* pair (pk_i, sk_i) known to all instances $\Pi_{U_i}^{s_i}$ and maintains an internal state that contains all private ephemeral information used during the session.

An adversary $\mathcal{A}$ is a PPT (Probabilistic Polynomial Time) algorithm with full control over the communication channel (he can modify, delete or insert messages) that interacts with the group members by asking queries ($\mathsf{Execute}$, Send, $\mathsf{RevealKey}$, $\mathsf{RevealState}$, $\mathsf{Corrupt}$, Test). The model introduces revised notions of AKE (*Authenticated Key Exchange*) security, MA (*Mutual Authentication*) security and *contributiveness* (key unpredictability by equal contribution of the parties to the key establishment). We denote by $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{AKE}}$, $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{MA}}$, $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Con}}$ the advantage of an adversary $\mathcal{A}$ to win the AKE security game, MA security game and respectively the contributiveness game. We skip the definitions[2], but strongly invite the reader to address the original paper [8].

[2] Because lack of space.

$S \in \mathbb{Z}_q$, $q > 2$ prime, $n = 2$, $m = 2$.	
Share 1 - The dealer:	**Share 2** -The dealer:
1. chooses $s_1^1 \xleftarrow{R} \mathbb{Z}_q$;	1. chooses $s_1^2 \xleftarrow{R} \mathbb{Z}_q$;
2. computes $s_2^1 = S - s_1^1 \pmod{q}$;	2. computes $s_2^2 = S - s_1^2 \pmod{q}$;
Rec 1	**Rec 2**
$S = s_1^1 + s_2^1 \pmod{q}$;	$S = s_1^2 + s_2^2 \pmod{q}$;
$\mathcal{AS}_1 = \{\{s_1^1, s_2^1\}\}$	$\mathcal{AS}_2 = \{\{s_1^2, s_2^2\}\}$
$\mathcal{AS} = \{\{s_1^1, s_2^1\}, \{s_1^2, s_2^2\}\} = \mathcal{AS}_1 \cup \mathcal{AS}_2$	

Fig. 1. Perfect Secret 2-Sharing Scheme based on Karnin et al.'s scheme

2.3 Secret n-Sharing Schemes

Let $\mathcal{SS} = (\mathsf{Share}, \mathsf{Rec})$ be a secret sharing scheme, where: $\mathsf{Share}(S, m)$ is a randomized sharing algorithm that splits a secret S into m shares $s_1, \ldots, s_m$; $\mathsf{Rec}(s_{i_1}, \ldots, s_{i_t})$, $t \leq m$ is a deterministic reconstruction algorithm from shares that outputs S if $\{s_{i_1}, \ldots, s_{i_t}\}$ is an *authorized* subset and halts otherwise.

The set of all authorized subsets is called *access structure*. The access structure of an (m, m) *all-or-nothing* secret sharing scheme consists of the single set with cardinality m. A secret sharing scheme is *perfect* if it provides no information about the secret to unauthorized subsets.

We follow the idea of Sun et al. [19] and extend secret sharing schemes: a *secret n-sharing scheme* is a scheme that splits a secret n times into the same number of shares m using the same Share algorithm. In other words, it runs a secret sharing scheme n times on the same input. Let $\mathcal{AS}$ be the access structure of a secret n-sharing scheme and $\mathcal{AS}_i$, $i = 1, \ldots, n$, be the access structure of the i-th run of the scheme which is based on. It is immediate that an authorized subset in any of the n splits remains authorized in the extended construction:

$$\mathcal{AS}_1 \cup \ldots \cup \mathcal{AS}_n \subseteq \mathcal{AS}. \tag{1}$$

Definition 1. ***(Perfect Secret n-Sharing)*** *A secret n-sharing scheme is called perfect if the following conditions hold: (1) its access structure is $\mathcal{AS} = \mathcal{AS}_1 \cup \ldots \cup \mathcal{AS}_n$; (2) it provides no information about the secret to unauthorized subsets.*

Definition 1 states that no authorized subsets exist except the ones already authorized within the n perfect sharing instances and that combining shares originating from distinct runs give no additional information about the secret.

We affirm that perfect secret n-sharing exists. In order to support our claim we introduce in Fig.1 an example based on Karnin et al.'s scheme [11].

3 Our proposal

We introduce a new GKA protocol based on perfect secret n-sharing in Fig.2.

The main idea is that each user U_i uniformly selects a random r_i (later a share in a $(2, 2)$ all-or-nothing scheme) and broadcasts its secured value through

```
**Round 1** - User $U_1$:
  1.1. runs $\mathcal{F}.\mathsf{G}(1^k)$ to obtain a public-private key pair $(pk, sk)$;
  1.2. chooses $r_1 \leftarrow^R \{0,1\}^k$;
  1.3. broadcatsts $U_1 \to^*$: $(\mathcal{U}, pk, \mathsf{F}(pk, r_1), \sigma = \Sigma.\mathsf{Sign}(sk_1, \mathcal{U}||pk||\mathsf{F}(pk, r_1)))$;
**Round 2** - Each user $U_i$, $i = 2, \ldots, n$:
  2.1. checks if $\Sigma.\mathsf{Verify}(pk_1, \mathcal{U}||pk||\mathsf{F}(pk, r_1), \sigma) = 1$.
       If the equality does not hold, he quits;
  2.2. chooses $r_i \leftarrow^R \{0,1\}^k$;
  2.3. broadcasts $U_i \to^*$: $(\mathsf{F}(pk, r_i)$, $\sigma_i = \Sigma.\mathsf{Sign}(sk_i, \mathcal{U}||pk||\mathsf{F}(pk, r_i)))$;
**Round 3** - User $U_1$:
  3.1. checks if $\Sigma.\mathsf{Verify}(pk_i, \mathcal{U}||pk||\mathsf{F}(pk, r_i), \sigma_i) = 1$, $i = 2, \ldots, n$.
       If at least one equality does not hold, he restarts the protocol;
  3.2. computes $\mathsf{sid}_{U_1} = \mathsf{F}(pk, r_1)|| \ldots ||\mathsf{F}(pk, r_n)$, $r_i = \mathsf{F}^{-1}(sk, \mathsf{F}(pk, r_i))$, the session key
       $K = \mathsf{H}(\mathsf{sid}_{U_1}||r_1||r_2|| \ldots ||r_n)$ and $r'_i$ such that $\mathcal{SS}.\mathsf{Share}(K, 2) = \{r_i, r'_i\}$, $i = 2, \ldots, n$;
  3.3. broadcasts $U_1 \to^*$: $(r'_2, \ldots r'_n, \sigma' = \Sigma.\mathsf{Sign}(sk_1, \mathcal{U}||pk||r'_2|| \ldots ||r'_n||\mathsf{sid}_{U_1})$;
**Key Computation** - Each user $U_i$, $i = 2, \ldots, n$:
  4.1. checks if $\Sigma.\mathsf{Verify}(pk_j, \mathcal{U}||pk||\mathsf{F}(pk, r_j), \sigma_j) = 1$, $j = 2, \ldots n$, $j \neq i$.
       If at least one equality does not hold, he quits;
  4.2. computes $\mathsf{sid}_{U_i} = \mathsf{F}(pk, r_1)|| \ldots ||\mathsf{F}(pk, r_n)$ and $K = \mathcal{SS}.\mathsf{Rec}(r_i, r'_i)$;
  4.3. checks if $\Sigma.\mathsf{Verify}(pk_1, \mathcal{U}||pk||r'_2|| \ldots ||r'_n||\mathsf{sid}_{U_i}, \sigma') = 1$.
       If the equality holds, he accepts the key $K$; otherwise he quits;
**Key Confirmation** - Each user $U_i$, $i = 2, \ldots, n$:
  5.1. computes $r_j$ such that $\mathcal{SS}.\mathsf{Share}(K, 2) = \{r_j, r'_j\}$, $j = 2, \ldots n$, $j \neq i$;
  5.2. checks if $\mathsf{F}(pk, r_j)$ equals the one sent in step 2.3.
       If at least one equality does not hold, he quits.
```

Fig. 2. GKA Protocol based on Secret n-Sharing

a trapdoor function F (Rounds 1 and 2). The initiator (U_1, without loss of generality) computes the session key K based on the received values and the session id, invokes secret n-sharing on inputs K and $r_2, \ldots, r_n$ and extracts the second shares $r'_2, \ldots, r'_n$, which he then broadcasts (Round 3). Each user U_i can recover the agreed session key K from only his own shares r_i and r'_i (Key Computation).

Key confirmation is achieved by default due to secret n-sharing: U_i eavesdrops r'_j (step 3.3), computes the corresponding r_j such that $\{r_j, r'_j\}$ represents a valid set of shares for K (step 5.1) and verifies that r_j is the genuine value chosen by U_j (step 5.2). Hence, U_i is sure that U_j uses the correct values $\{r_j, r'_j\}$ as inputs for the reconstruction algorithm and obtains the same key. However, an adversary may prevent the last broadcast message (step 3.3) to arrive to one or more users, who become unable to recover the key. This represents a DoS (Denial of Service) attack, which is always detected, but leads to the futility of the protocol. We do not consider this as a weakness of our proposal, since GKE security models ignore DoS scenarios [6] and therefore all existing provable secure GKA protocols are susceptible to such attacks.

The construction requires a secret n-sharing scheme that given as input a secret K and the shares $r_2, \ldots, r_n$ permits to compute the second shares $r'_2, \ldots, r'_n$

such that $\mathsf{Share}(K, 2) = \{r_i, r'_i\}$ (or, equivalent $\mathsf{Rec}(r_i, r'_i) = K$), $i = 2, \dots, n$. We emphasize that this does not restrict the applicability, since efficient schemes with such property exist - for example the secret n-sharing scheme in Fig.1.

4 Security Proofs

Theorem 1. ***(AKE Security)*** *If the signature scheme Σ is UF-CMA, the trapdoor function $\mathcal{F}$ is secure, the hash function H is a random oracle and the secret n-sharing scheme $\mathcal{SS}$ is perfect then our protocol is AKE secure and*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{AKE}} \leq 2n^2 \mathsf{Adv}_{\mathcal{A},\Sigma}^{\mathsf{UF-CMA}} + \frac{(q_s + q_r)^2}{2^{k-1}} + \frac{(n+1){q_s}^2}{2^{k-1}} + 2\mathsf{Adv}_{\mathcal{A},\mathcal{F}}.$$

Proof. We prove by a sequence of games. Let $\mathsf{Win}_{\mathsf{i}}^{\mathsf{AKE}}$ be the event that the adversary $\mathcal{A}$ wins $\mathsf{Game\ i}, \mathsf{i} = 0, \dots, 4$.

Let **Game 0** be the original AKE security game. By definition, we have:

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{AKE}} = |2Pr[\mathsf{Win}_0^{\mathsf{AKE}}] - 1|. \tag{2}$$

Let **Game 1** be the same as $\mathsf{Game\ 0}$, except that the simulation fails if an event Forge occurs, where Forge simulates a successful forgery on an honest user signature. We follow the idea from [8] to estimate $Pr[\mathsf{Forge}]$ and obtain:

$$|Pr[\mathsf{Win}_1^{\mathsf{AKE}}] - Pr[\mathsf{Win}_0^{\mathsf{AKE}}]| \leq Pr[\mathsf{Forge}] \leq n^2 \mathsf{Adv}_{\mathcal{A},\Sigma}^{\mathsf{UF-CMA}}. \tag{3}$$

Let **Game 2** be the same as $\mathsf{Game\ 1}$, except that the simulation fails if an event $\mathsf{Collision}$ occurs, meaning that the random oracle H produces a collision for any of its inputs. Since the total number of random oracle queries is bounded by $q_s + q_r$:

$$|Pr[\mathsf{Win}_2^{\mathsf{AKE}}] - Pr[\mathsf{Win}_1^{\mathsf{AKE}}]| \leq Pr[\mathsf{Collision}] \leq \frac{(q_s + q_r)^2}{2^k}. \tag{4}$$

Let **Game 3** be the same as $\mathsf{Game\ 2}$, except that the simulation fails if an event Repeat occurs, where Repeat simulates a replay attack: it appears when the same public-private key pair (pk, sk) is used in different sessions (event bounded by $q_s^2/2^k$) or when a party uses the same value r_i in different sessions (event bounded by $nq_s^2/2^k$). Hence, we get:

$$|Pr[\mathsf{Win}_3^{\mathsf{AKE}}] - Pr[\mathsf{Win}_2^{\mathsf{AKE}}]| \leq Pr[\mathsf{Repeat}] \leq \frac{(n+1){q_s}^2}{2^k}. \tag{5}$$

Note that this last game eliminates forgeries and replay attacks.

Let **Game 4** be the same as $\mathsf{Game\ 3}$, except that the simulation fails if $\mathcal{A}$ is able to compute at least one value r_i, $i = 1, \dots, n$. Hence:

$$|Pr[\mathsf{Win}_4^{\mathsf{AKE}}] - Pr[\mathsf{Win}_3^{\mathsf{AKE}}]| \leq \mathsf{Adv}_{\mathcal{A},\mathcal{F}}. \tag{6}$$

Since only r'_i, $i = 2, \dots, n$ are available for the adversary in this last game and $\mathcal{SS}$ is perfect, $\mathcal{A}$ has no advantage in finding the secret and $|Pr[\mathsf{Win}_4^{\mathsf{AKE}}]| = 0$.

We conclude by combining (2) - (6).

Theorem 2. ***(MA Security)*** *If the signature scheme* Σ *is UF-CMA and the hash function* H *is a random oracle then our protocol is MA secure and*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{MA}} \leq n^2 \mathsf{Adv}_{\mathcal{A},\Sigma}^{\mathsf{UF-CMA}} + \frac{(q_s + q_r)^2}{2^k} + \frac{(n+1){q_s}^2}{2^k}.$$

Proof. We prove by a sequence of games. Let $\mathsf{Win}_{\mathsf{i}}^{\mathsf{MA}}$ be the event that the adversary $\mathcal{A}$ wins $\mathsf{Game\ i}$, $\mathsf{i} = 0, \ldots, 3$.

Let **Game 0** be the original MA security game. By definition, we have:

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{MA}} = Pr[\mathsf{Win}_0^{\mathsf{MA}}]. \tag{7}$$

Let **Game 1** be the same as $\mathsf{Game\ 0}$, except that the simulation fails if the event Forge defined in $\mathsf{Game\ 1}$ of Theorem 1 occurs:

$$|Pr[\mathsf{Win}_1^{\mathsf{MA}}] - Pr[\mathsf{Win}_0^{\mathsf{MA}}]| \leq Pr[\mathsf{Forge}] \leq n^2 \mathsf{Adv}_{\mathcal{A},\Sigma}^{\mathsf{UF-CMA}}. \tag{8}$$

Let **Game 2** be the same as $\mathsf{Game\ 1}$, except that the simulation fails if the event $\mathsf{Collision}$ defined in $\mathsf{Game\ 2}$ of Theorem 1 occurs:

$$|Pr[\mathsf{Win}_2^{\mathsf{MA}}] - Pr[\mathsf{Win}_1^{\mathsf{MA}}]| \leq Pr[\mathsf{Collision}] \leq \frac{(q_s + q_r)^2}{2^k}. \tag{9}$$

Let **Game 3** be the same as $\mathsf{Game\ 2}$, except that the simulation fails if the event Repeat defined in $\mathsf{Game\ 3}$ of Theorem 1 occurs:

$$|\mathsf{Pr}[\mathsf{Win}_3^{\mathsf{MA}}] - Pr[\mathsf{Win}_2^{\mathsf{MA}}]| \leq Pr[\mathsf{Repeat}] \leq \frac{(n+1){q_s}^2}{2^k}. \tag{10}$$

This last game excludes both forgeries and replay attacks. If $\mathsf{Game\ 3}$ does not aboard, it is impossible for honest partnered parties to accept with different keys. Hence $Pr[\mathsf{Win}_3^{\mathsf{MA}}] = 0$.

We conclude by combing (7) - (10).

Theorem 3. ***(Contributiveness)*** *If the trapdoor function* $\mathcal{F}$ *is secure and the hash function* H *is a random oracle then our protocol is contributive and*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Con}} \leq \frac{(n+1){q_s}^2}{2^k} + \frac{q_r}{2^k}.$$

Proof. We prove by a sequence of games. Let $\mathsf{Win}_{\mathsf{i}}^{\mathsf{Con}}$ be the event that the adversary $\mathcal{A}$ wins $\mathsf{Game\ i}, \mathsf{i} = 0 \ldots 2$.

Let **Game 0** be the original contributiveness game. By definition, we have:

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Con}} = Pr[\mathsf{Win}_0^{\mathsf{Con}}]. \tag{11}$$

Let **Game 1** be the same as $\mathsf{Game\ 0}$, except that the simulation fails if the event Repeat defined in $\mathsf{Game\ 3}$ of Theorem 1 occurs:

$$|Pr[\mathsf{Win}_1^{\mathsf{Con}}] - Pr[\mathsf{Win}_0^{\mathsf{Con}}]| \leq Pr[\mathsf{Repeat}] \leq \frac{(n+1){q_s}^2}{2^k}. \tag{12}$$

Let **Game 2** be the same as $\mathsf{Game\ 1}$, except that the simulation fails if $\mathcal{A}$ can find a collision for $K = \mathsf{H}(\mathsf{sid}_{U_1}||r_1||r_2||\dots||r_n)$ on an input r_i, $i = 1, \dots, n$:

$$|Pr[\mathsf{Win}_2^{\mathsf{Con}}] - Pr[\mathsf{Win}_1^{\mathsf{Con}}]| \leq \frac{q_r}{2^k}. \tag{13}$$

If $\mathsf{Game\ 2}$ does not abort, the output of the random oracle is uniformly distributed. Hence $Pr[\mathsf{Win}_2^{\mathsf{Con}}] = 0$.

We conclude by combining (11) - (13).

5 Protocol Analysis and Future Work

Table 1 analysis the complexity of the proposed protocol from three different perspectives: storage, computational cost and overall transmission cost. Let l_x be the length (in bits) of x and c_y be the cost to execute y. First, our proposal is storage efficient: each user maintains his long-lived secret key and a session ephemeral value r_i; in addition, the initiator keeps secret a session trapdoor key. Second, the computational cost is acceptable for a regular party. In case Key Confirmation phase is performed, extra cost is required. However, we stress that $c_{\mathcal{SS}.\mathsf{Share}}$ and $c_{\mathcal{SS}.\mathsf{Rec}}$ can be neglected as they reduce to a sum modulo a prime when the scheme in Fig.1 is used. We also remark that the untraditional perspective on secret n-sharing scheme permits the parties to efficiently recover the key by themselves (with no need to interact with the other parties in Key Computation Phase). Third, the overall dimension of the exchanged messages during one session of the protocol is cost-efficient.

Our proposal introduces different storage and computational costs for the initiator and the rest of the users - a property of GKT rather than GKA protocols. This suggests to introduce an online high performance entity that runs the computation instead of the initiator, while the initiator plays the role of a regular party (after he requests the key establishment). Therefore, the costs of the initiator become lower, while key contributiveness is maintained.

Table 2 compares our proposal with the existing work (we restrict the comparison to GKA protocols based on secret sharing with formal security proofs).

Our protocol maintains a constant number of rounds regardless the group size, while it achieves better security: Bresson and Catalano [2] miss the strong corruption (the adversary is not allowed to reveal private internal state information of participants) and Cao et al. [7] miss a contributiveness proof. Our proposal is secure in the GBG model and hence it stands against KCI attacks.[3]

We emphasize two more advantages of our work: the session id is computed at runtime (the environment of the protocol does not generate it in advance) and participants communicate through broadcast channels only (highly appreciated in distributed and multi-agent systems as well as ad-hoc mobile networks due to concurrency increase and transmission overhead reduction).

We consider as topics for further research the improvements of the protocol such that it admits dynamic groups, anonymity and robustness.

[3] We are confident that it can be improved to become secure in the eGBG model; because of space limitations we consider this for an extended version of the paper.

Table 1. Complexity Analysis of the Proposed Protocol

	Storage	Computation	Transmission
U_1	$l_{sk} + l_{sk_1} + k$	$c_{\mathcal{F}.\mathsf{G}} + c_{\mathsf{F}} + (n-1)c_{\mathsf{F}^{-1}} + c_{\mathsf{H}}$ $2c_{\Sigma.\mathsf{Sign}} + (n-1)c_{\Sigma.\mathsf{Verify}} + (n-1)c_{\mathcal{SS}.\mathsf{Share}}$	$l_{\mathcal{U}} + l_{pk} + nl_{\mathsf{F}} +$ $+(n+1)l_{\Sigma.\mathsf{Sign}} + (n-1)k$
U_i $(i \neq 1)$	$l_{sk_i} + k$	$c_{\mathsf{F}} + c_{\Sigma.\mathsf{Sign}} + nc_{\Sigma.\mathsf{Verify}} + c_{\mathcal{SS}.\mathsf{Rec}}$ $(+(n-2)c_{\mathcal{SS}.\mathsf{Share}})$	

Table 2. Comparison to the Existing Work

	No. of Rounds	Transmission Type	Group Type	sid Generation	Security Model
Our Protocol	3	broadcast	static	at runtime	GBG / ROM
Bresson and Catalano [2]	3	unicast, broadcast	static	in advance	BCP / ROM
Cao et al. [7]	3	broadcast	static	in advance	UC / ROM

6 Conclusions

We proposed a new method to build GKA protocols based on secret sharing. We relied our construction on a slightly different perspective of secret sharing inspired by the idea of Sun et al. [19], which we call *secret n-sharing*. We introduced the notion of *perfect n-sharing* as a natural generalization of perfect sharing. Secret n-sharing brings several advantages as a building block of GKA protocols: broadcast communication is sufficient, key computation is efficient, key confirmation is achieved by default, users recover the key from their own shares. Our protocol achieves better security than the existing work, while it maintains the same number of communication rounds regardless the number of participants.

Acknowledgments. 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. Blakley, G.: Safeguarding Cryptographic Keys. In: Proceedings of the 1979 AFIPS National Computer Conference. pp. 313–317 (1979)
2. Bresson, E., Catalano, D.: Constant Round Authenticated Group Key Agreement via Distributed Computation. In: Public Key Cryptography. pp. 115–129 (2004)
3. Bresson, E., Chevassut, O., Pointcheval, D.: Provably Authenticated Group Diffie-Hellman Key Exchange - The Dynamic Case. In: Proceedings of ASIACRYPT '01. pp. 290–309 (2001)

4. Bresson, E., Chevassut, O., Pointcheval, D.: Dynamic Group Diffie-Hellman Key Exchange under Standard Assumptions. In: Proceedings of EUROCRYPT '02. pp. 321–336 (2002)
5. Bresson, E., Chevassut, O., Pointcheval, D., Quisquater, J.J.: 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)
6. Bresson, E., Manulis, M.: Securing group key exchange against strong corruptions. In: Proceedings of ASIA CSS'08, pp.249-260 (2008)
7. Cao, C., Yang, C., Ma, J., Moon, S.J.: Constructing UC Secure and Constant-Round Group Key Exchange Protocols via Secret Sharing. EURASIP J. Wireless Comm. and Networking (2008)
8. Gorantla, M.C., Boyd, C., Nieto, J.M.G.: Modeling Key Compromise Impersonation Attacks on Group Key Exchange Protocols. In: Public Key Cryptography. pp. 105–123 (2009)
9. Harn, L., Lin, C.: Authenticated Group Key Transfer Protocol based on Secret Sharing. IEEE Trans. Comput. 59(6), 842–846 (2010)
10. Hsu, C., Zeng, B., Cheng, Q., Cui, G.: A Novel Group Key Transfer Protocol. Cryptology ePrint Archive, Report 2012/043 (2012)
11. Karnin, E.D., Greene, J.W., Hellman, M.E.: On Secret Sharing Systems. IEEE Transactions on Information Theory 29(1), 35–41 (1983)
12. Katz, J., Shin, J.S.: Modeling Insider Attacks on Group Key-Exchange Protocols. In: Proceedings of the 12th ACM Conference on Computer and Communications Security (CCS05). pp. 180–189 (2005)
13. Nam, J., Kim, M., Paik, J., Jeon, W., Lee, B., Won, D.: Cryptanalysis of a Group Key Transfer Protocol based on Secret Sharing. In: Proceedings of the Third international conference on Future Generation Information Technology. pp. 309–315 (2011)
14. Olimid, R.F.: On the Security of an Authenticated Group Key Transfer Protocol Based on Secret Sharing. In: Proceedings of ICT-EurAsia 2013 (AsiaARES'13). pp. 399–408 (2013)
15. Olimid, R.F.: Cryptanalysis of a Password-based Group Key Exchange Protocol Using Secret Sharing. Appl. Math. Inf. Sci. 7(4), 1585–1590 (2013)
16. Pieprzyk, J., Li, C.H.: Multiparty Key Agreement Protocols. In: IEEE Proceedings - Computers and Digital Techniques. pp. 229–236 (2000)
17. Sáez, G.: Generation of Key Predistribution Schemes using Secret Sharing Schemes. Discrete Applied Mathematics 128(1), 239–249 (2003)
18. Shamir, A.: How to Share a Secret. Commun. ACM 22(11), 612–613 (1979)
19. Sun, Y., Wen, Q., Sun, H., Li, W., Jin, Z., Zhang, H.: An Authenticated Group Key Transfer Protocol based on Secret Sharing. Procedia Engineering 29(0), 403 – 408 (2012)
20. Yuan, W., Hu L., Li H, Chu J.: An Efficient Password-based Group Key Exchange Protocol Using Secret Sharing, Appl. Math. Inf. Sci. 7(1), 145–150 (2013)
21. Zhao, J., Gu, D., Gorantla, M.C.: 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)