Yet another insecure group key distribution scheme using secret sharing

Chris J. Mitchell
Information Security Group, Royal Holloway, University of London
www.chrismitchell.net

16th November 2020

Abstract

A recently proposed group key distribution scheme known as UMKESS, based on secret sharing, is shown to be insecure. Not only is it insecure, but it does not always work, and the rationale for its design is unsound. UMKESS is the latest in a long line of flawed group key distribution schemes based on secret sharing techniques.

1 Introduction

There is a long and sad history of insecure group (cryptographic) key establishment schemes based on secret sharing. As noted by Boyd and Mathuria, [2], the 'idea to adapt secret sharing for key broadcasting seems to have been first proposed by Laih et al. [12]', in a paper published over 30 years ago. However, the shortcomings of the approach, and of the many variants that have been proposed since 1989, have been widely discussed for almost as long, in particular that:

- as noted by Boyd and Mathuria, [2], a 'malicious principal who obtains one key gains information regarding the shares of other principals', and an outside eavesdropper can also gain this information if the old group keys are revealed;
- again as noted by Boyd and Mathuria, [2], since 'knowledge of any of the shared secrets is sufficient to construct the session key, none of these protocols provides forward secrecy';

- insider attacks of various attacks appear impossible to prevent, as many authors have observed (see, for example, [13, 14, 15], and the papers cited therein).

The history of such protocols is long and tangled, but one sequence of flawed protocol proposals, breaks, proposed fixes, and breaks of the fixes is explained very carefully in Section 5 of Liu et al. [13], and we now briefly summarise part of the story. In 2010, Harn and Lin [5] proposed an 'authenticated group key transfer protocol based on secret sharing' (itself intended to address issues in the Laih et al. scheme [12] from 1989). Unfortunately, this was shown not only to be insecure (by Nam et al. [16, 17]) but also erroneous in that it does not always work even if all parties execute it correctly (see Nam et al. [17]). Nam et al. [17] also proposed a fixed version, but this was shown to be insecure by Liu et al. [13]. Inspired by the Harn and Lin 2010 paper, Sun et al. [20] proposed another group key transfer protocol using secret sharing, and this was shown to be insecure by both Kim et al. [9] and Olimid [18]. Olimid [18] also proposed a fix, but this was shown to be insecure by Kim et al. [10]. These are not the only examples of broken schemes of this type — one common element is the lack is a rigorous proof of security in a complexity-theoretic setting, the established state of the art for such protocols for the last decade or two.

Unfortunately, despite the extensive literature pointing out these and other problems, new and fundamentally flawed schemes of this general type keep being published. One common element in the papers published over the last 31 years is that many share one of the authors of the 1989 paper. A further common element is that each new paper cites some of the previously published schemes, but many completely fail to acknowledge any of the many attacks against the previously published and often very closely related schemes. This is most unfortunate, especially given that many of the newer schemes suffer from the same problems as older schemes. As we show below, some of the above statements are also true for UMKESS, a scheme of this general type published in a very recent paper by Hsu, Harn and Zeng [7].

The remainder of the paper is structured as follows. The UMKESS scheme is summarised in §2. A detailed critique is provided in §3. A brief discussion of why proposing arbitrary fixes to such schemes is unwise is given in §4, and conclusions are drawn in §5.

2 The UMKESS scheme

2.1 Objectives

This scheme is designed to allow a single trusted authority, the *Key Generation Centre (KGC)* to simultaneously distribute a number of secret group keys to a number of distinct sets (*groups*) of entities, with each set being drawn from a larger set of entities all of which have a pre-established relationship with the KGC.

The scheme uses the Shamir secret sharing scheme [19], involving polynomials over a prime finite field $\mathrm{GF}(p) = \mathbb{Z}_p$, for large p.

2.2 Preliminaries

Prior to use a large *safe* prime p is selected. The definition of safe is not provided by the authors, but presumably it must be sufficiently large to prevent exhaustive searching for individual keys (which are elements of $\mathrm{GF}(p)$).

The protocol involves the KGC and a set of n users $\mathcal{U} = \{U_1, U_2, \ldots, U_n\}$, from which groups are created who are provided with new shared session keys by the KGC on demand. Each user $U_i \in \mathcal{U}$ is assumed to share a unique secret $x_i \in \mathrm{GF}(p)$ with the KGC.

All involved parties must also agree on a cryptographic hash-function h, whose domain and range is $\mathrm{GF}(p)$.

2.3 Security claims

The authors claim the protocol is secure against both insider and outsider attacks, where an insider attacker is a member of $\mathcal{U}$. The security properties are not defined formally.

2.4 Operation

As noted above, the protocol enables the KGC to simultaneously broadcast a set of group keys to a disparate collection of groups. We suppose that an instance of the protocol is being executed to distribute m group keys $K_1, K_2, \ldots, K_m$ to m distinct groups $G_1, G_2, \ldots, G_m$, where $G_i \subseteq \mathcal{U}$ and we write $|G_i = s_i|$ for every i, $1 \leq i \leq m$. For each group $G_i = \{U_{i_1}, U_{i_2}, \ldots, U_{i_{s_i}}\}$, say $(1 \leq i \leq m)$, define

$$S(G_i) = \sum_{j=1}^{s_i} i_j$$

i.e. $S(G_i)$ is the sum of the indices of the members of the group. Here as throughout addition is computed in GF(p), i.e. modulo p.

The protocol proceeds as follows, where the step numbers correspond to those given by Hsu et al. [7].

2. The KGC broadcasts the list of groups $G_1, G_2, \ldots, G_m$ and their members in a reliable way, i.e. it is assumed that these cannot be modified by a malicious insider or outsider[1].

3. Each participating user $U_i \in \mathcal{U}$ ($1 \leq i \leq n$), i.e. each user who is a member of at least one group, proceeds as follows. Suppose U_i is a member of m_i groups $G_{i_1}, G_{i_2}, \ldots, G_{i_{m_i}}$. U_i chooses m_i random values $r_{i_j} \in$ GF(p), $1 \leq j \leq m_i$, and sends them (unprotected) to the KGC, i.e. in a way that might permit them to be changed by a malicious party (this assumption is in line with the protocol specification — see, for example, the 'proof' of Theorem 5 [7]).

4. Once the KGC has received the sets of random values r_{i_j} from all the participating members of $\mathcal{U}$, it performs the following steps.

   (a) The KGC chooses m random keys $K_i \in$ GF(p), $1 \leq i \leq m$, where K_i is intended for use by group G_i, and a random value $r_0 \in$ GF(p).

   (b) For each participating user U_i ($1 \leq i \leq n$), the KGC:

   - computes the unique degree m_i polynomial f_i over GF(p) that passes through the following $m_i + 1$ points:

     $$(i, x_i + r_0) \text{ and } (S(G_{i_j}), K_{i_j} + h(x_i + r_{i_j} + r_0)),\ 1 \leq j \leq m_i;$$

   - randomly chooses a set of m_i points $\{P_1, P_2, \ldots, P_{m_i}\}$ lying on the curve defined by f_i; and
   - sends $P_1, P_2, \ldots, P_{m_i}$ to U_i (again unprotected, i.e. in a way that might permit them to be changed by a malicious party).

   (c) The KGC makes the values of r_0 and $h(K_i)$, $1 \leq i \leq m$, publicly available to all members of $\mathcal{U}$ in a reliable way, i.e. it is assumed that these cannot be modified by a malicious insider or outsider[2].

5. Each participating user U_i ($1 \leq i \leq n$) proceeds as follows.

   (a) On receipt of $P_1, P_2, \ldots, P_{m_i}$, U_i uses them together with the point $(i, x_i + r_0)$ to recover the degree m_i polynomial f_i.

[1] This integrity/authenticity assumption is implied but never explicitly made, but without it certain obvious attacks apply, as discussed in §3.3 below.

[2] Again this assumption is only implicit, but without it certain attacks apply — see §3.3.

(b) Using f_i and $S(G_{i_j})$, $1 \leq j \leq m_i$, U_i can compute $K_{i_j} + h(x_i + r_{i_j} + r_0)$ and hence K_{i_j}, for every j.

(c) Finally, U_i checks the recovered group keys K_{i_j} against the published list of values $h(K_i)$, $\leq i \leq m$, made available in a reliable way to all participants.

In essence, a separate 'secret' polynomial is computed for each participating user, and the user recovers group keys from points on this polynomial (which has degree equal to the number of group keys to be distributed to this user).

3 A critique

3.1 A definitional issue

We first observe that, in certain not unlikely cases, the system cannot work.

In Step 4(b), the KGC generates the following m points:

$$(S(G_{i_j}), K_{i_j} + h(x_i + r_{i_j} + r_0)),\ 1 \leq j \leq m_i;$$

Clearly, if the values r_{i_j} are all distinct, $1 \leq j \leq m_i$, then the y coordinates will all be distinct. However, there is nothing to prevent the possibility that $S(G_{i_j}) = S(G_{i_{j'}})$ for two distinct groups G_{i_j} and $G_{i_{j'}}$. This could happen very easily, e.g. if $G_{i_j} = \{U_1, U_5\}$ and $G_{i_{j'}} = \{U_1, U_2, U_3\}$, where we have $S(G_{i_j}) = S(G_{i_{j'}}) = 6$. In such a case, the polynomial f_1 for user U_1 cannot exist, since it cannot pass through two points with the same x coordinate but distinct y coordinates.

This issue could, of course, be fixed, e.g. by replacing $S(G_i)$ throughout by a unique numeric identifier for the group G_i. Indeed, it would seem reasonable to require the KGC to devise a new (and unique) set of group identifiers for every instance of the protocol, and to distribute them as part of Step 2 of the protocol. However, given that there are more serious issues with the security of, and rationale for, the protocol, we do not explore such fixes further here.

3.2 A serious security weakness

We now demonstrate that a much more serious security issue exists, in that the long-term secret x_i of one user can be recovered by another user (an insider attacker), who needs only make a small modification to one message sent to the KGC by the 'victim' user and then intercept the response. We use the same notation as employed in the protocol description in §2.4.

We suppose that the insider attacker (U_a, say) is a member of (at least) two groups in common with the victim user U_v. Suppose that U_a intercepts the set of random values $\{r_1, r_2, \ldots, r_{m_v}\}$ sent by user U_v to the KGC in Step 3, and prevents them reaching the KGC; we suppose also, without loss of generality, that U_a is a member of the two groups G_{v_1} and G_{v_2}. We further suppose that T modifies the set of random values sent by U_v to $\{r_1, r_2', r_3, \ldots, r_{m_v}\}$ before forwarding them to the KGC, where $r_2' = r_1$.

The protocol proceeds exactly as specified and we observe that U_a, as a legitimate protocol participant, will be able to learn K_{v_1} and K_{v_2} from the set of points it is sent by the KGC (since we assumed that U_a is a member of the two groups G_{v_1} and G_{v_2}).

We further suppose that K_a intercepts the set of points $P_1, P_2, \ldots, P_{m_v}$ sent to U_v — these points will all lie on the polynomial f_v generated by the KGC in Step 4. This polynomial will also pass through the points:

$$(S(G_{v_1}), K_{v_1} + H) \text{ and } (S(G_{v_2}), K_{v_2} + H)$$

(amongst others), where $H = h(x_v + r_{v_1} + r_0)$. That is, apart from the m_i points $P_1, P_2, \ldots, P_{m_v}$, U_a will know the difference between the y values for two other points on the curve defined by f_v (with known x values). That is, if we let $z_1 = S(G_{v_1})$ and $z_2 = S(G_{v_2})$, U_a will know the following equation holds:

$$f_v(z_1) - f_v(z_2) = K_{v_1} - K_{v_2}$$

where all the values (apart from the coefficients of f_v are known. This yields a linear equation in the coefficients of f_v.

The m_v points $P_1, P_2, \ldots, P_{m_v}$ can be used to yield a set of m_v further linear equations in the $m_i + 1$ coefficients of f_v, i.e. U_a will have a set of $m_v + 1$ linear equations in the $m_v + 1$ coefficients of f_v, which will almost certainly be independent given that $P_1, P_2, \ldots, P_{m_v}$ are randomly chosen and p is very large. These can very easily be solved to yield f_v. Finally, U_a simply evaluates $f_v(v)$ to yield $x_v + r_0$, i.e. U_a has the long-term secret of U_v (since r_0 is public).

That is, using this simple attack, one legitimate user can obtain the secret belonging to another user, and can thereafter learn all the group keys issued to this user. This clearly invalidates Theorem 5 of Hsu et al. [7]; this is not so surprising since the 'proof' offered is a series of heuristic arguments rather than a rigorous proof.

3.3 Reliable broadcasts

In the protocol description in §2, there are four main communications flows:

- two broadcasts to all participants from the KGC: a broadcast of the list of groups (Step 2), and a broadcast of the values r_0 and $h(K_i)$ $(1 \leq i \leq m)$ (Step 4c);

- transmission of m_i random values r_{i_j} from each participating user U_i to the KGC (Step 3);
- for every participating user U_i, transmission from the KGC to U_i of the set of points $\{P_1, P_2, \ldots, P_{m_i}\}$ (Step 4b).

Hsu et al. [7] do not made clear the degree to which these communications flows need to be protected. They variously refer to a 'broadcast channel', 'broadcasts', and making information 'publicly known'. However they do claim (in the 'proof' of Theorem 5), that 'service requests from group members are not authenticated', and also that 'an adversary (insider) can ... forge challenges of other group member'. They also explicitly refer to the possibility that one of the m_i values r_{i_j} is modified by an adversary.

We have therefore assumed throughout this paper that the transmission of the r_{i_j} values to the KGC in Step 3 is unprotected. This enables the attack described in §3.2. We have correspondingly assumed that the transmission of the points $P_1, P_2, \ldots, P_{m_i}$ from the KGC to each participating user U_i in Step 4b is unprotected, although we do not discuss this further here.

There is no substantive discussion of the security requirements for the two broadcasts made by the KGC to all participants. On reflection, and to be as fair as possible to the protocol designers, we have assumed that these are protected in some way, e.g. by being posted on a KGC website which can be authenticated (e.g. using TLS). Of course, this adds an 'invisible' overhead to the protocol, but it is a necessary assumption, since if either of these broadcasts can be manipulated then attacks are possible, as we now briefly describe.

- If the list of groups can be manipulated then a simple outsider attack is possible which we describe in the form of a short example. Suppose group G_i in the list includes the users U_1, U_2 and U_3. Then, clearly, $S(G_i) = 6$. Suppose that the version of the group list sent to U_1 is modified to G'_i so that G'_i includes U_1 and U_5. Then $S(G'_i) = 6$, i.e. the polynomial f_1 computed by the KGC would be exactly the same in both cases; this means that, when performing the protocol, user U_1 will recover key K_i correctly, but will believe it is shared with user U_5 when it is in fact shared with users U_2 and U_3. This is clearly not a desirable situation.
- If the list of hashed keys $h(K_i)$ ($1 \leq i \leq m$) can be manipulated, then in this case an insider attack is possible, which we again describe in the form of a simple example. Suppose a 'victim' user U_v is in the same group, G_{v_t} say (for some t satisfying $1 \leq t \leq m_v$), as an attacker user U_a. Both users perform the protocol correctly, except U_a prevents the correct list of hashed keys $\{h(K_1), h(K_2), \ldots, h(K_m)\}$ and the correct

set of points $\{P_1, P_2, ..., P_{m_v}\}$ reaching U_v. U_A completes the protocol correctly, and learns K_{v_t} (since both U_a and U_v are in group G_{v_t}). U_a now chooses a key K'_{v_t} which will be accepted by U_v instead of K_{v_t}.

U_a next computes the unique polynomial δ of degree m_v passing through the $m_v + 1$ points:

$$(i, 0),\ (S(G_{v_t}), K'_{v_t} - K_{v_t}) \text{ and } (S(G_{v_j}), 0),\ 1 \leq j \leq m_i\ (j \neq t).$$

Suppose the points $P_1, P_2, ..., P_{m_v}$ sent by the KGC to U_v (but which did not reach U_v) satisfy $P_i = (x_i, y_i)$. U_a now computes a new set of points $D_i = (x_i, d_i)$, $1 \leq i \leq m_v$, which lie on δ, and puts $P'_i = (x_i, y_i + d_i)$, $1 \leq i \leq m_v$. It should be clear that the points P'_i all lie on the curve defined by the polynomial $f_v + \delta$; it should also be clear that the point $(v, x_v + r_0)$ also lies on this curve, although the y value is of course not known to U_a.

U_a now sends to U_v (masquerading as the KGC), the correct set of hashed keys except that $h(K_{v_i})$ is replaced by $h(K'_{v_i})$, and the new set of points $P'_1, P'_2, ..., P'_{m_v}$. Since $P'_1, P'_2, ..., P'_{m_v}$ and $(v, x_v + r_0)$ all lie on the curve defined by $f_v + \delta$, this is the polynomial that will be recovered by U_v (instead of f_v). U_v now evaluates this polynomial and it is simple to see that U_v will recover the correct set of keys except that K_{v_i} will be replaced by K'_{v_i} — this is consistent with the manipulated set of hashed group keys, and hence U_v will accept the recovered keys as valid.

3.4 A questionable rationale

We further point out that the rationale for the scheme is highly questionable. One instance of the scheme costs each participant a total of m_i executions of the hash function h, together with solving for the coefficients of a degree $m_i + 1$ polynomial and a few modular additions, i.e. on average one hash execution plus some minor computations for each key.

Hsu et al. [7] compare the cost of their scheme with two other protocols. The first uses a public key cryptosystem, and the second involves a number of parallel executions of another secret sharing based scheme proposed by Harn and Lin [5]. Neither of these are sensible comparisons. The public key scheme is designed with different assumptions, and it would be expected to be significantly more costly. The comparison with the scheme of Harn and Lin makes no sense at all because, as discussed in §1, it is known to be insecure. Moreover, the comparisons avoid the cost of providing publicly verifiable lists of the groups and of group key hashes.

Even more importantly, there are very well-established protocols which achieve the same goal in a provably secure way at comparable computational and communications cost, and which avoid the need for a publicly verifiable publication of group key hashes. The authors completely ignore the huge and very well-established literature in the area, e.g. as summarised in the excellent Boyd and Mathuria [2] (and the recent second edition, [1]). Indeed, there is even an international standard for group key establishment — ISO/IEC 11770-5 [8], which was published in 2011.

4 Pointless fixes

In §1, some of the sad history of group key distribution schemes based on secret sharing was described. It seems clear that the cycle of design, break and fix is itself broken, at least until and unless a 'fixed' protocol is proven secure in a rigorous way. This point is made by Liu et al. [13].

> The security proof for each vulnerable group key distribution protocol only relies on incomplete or informal arguments. It can be expected that they would suffer from attacks.

Sadly, this lesson has not yet been recognised by everyone. Apart from the cases mentioned in §1, we should also mention the secret-sharing-based group key transfer scheme proposed by Hsu et al. in 2017 [6]. This was shown to be insecure [14] shortly after its publication. In a response published shortly afterwards, Kisty and Saputra [11] proposed a fixed version of the 2017 protocol. Sadly this 'fix' completely lacks a rigorous security analysis. As a result, it too may be insecure. However, perhaps more significantly, the fix involves the addition of digital signatures to enable recipients of certain messages to verify their origin and integrity. Whilst this may well prevent attacks, it completely negates any rationale for the design of the protocol by greatly increasing the computational complexity. Distributing group keys using public key techniques is a well known and solved problem, and thus the Kisty-Saputra scheme is not a valuable contribution to the literature.

Finally we observe that there are well-established formal security models within which properties of group key establishment protocols can be established — see in particular Bresson et al. [3] and Gorantla et al. [4]. A helpful summary of the scope of the various models for group key establishment protocols can be found in §2.7.1 of Boyd et al. [1].

5 Concluding remarks

In this paper we have discussed two related themes: the (sad) history of insecure group key distribution schemes based on secret sharing, and the details of why a specific example of a recently proposed scheme of this type is insecure. Perhaps the saddest point is that the literature reviewed here is only a small sample of a very extensive literature on secret-sharing-based group key distribution, including a number of other sagas involving schemes repeatedly broken and fixed.

In conclusion, this evidence strongly argues in favour of two recommendations. Firstly, the academic world should stop publishing security schemes for which there is a lack of robust evidence of security. Secondly, academia should stop attempting to publish fixed schemes which are pointless either because there is no proof of security or because, whilst they may be secure, they invalidate the rationale of the original unfixed scheme.

References

[1] C. Boyd, A. Mathuria, and D. Stebila. *Protocols for Authentication and Key Establishment*. Information Security and Cryptography. Springer, 2nd edition, 2020.

[2] C. A. Boyd and A. Mathuria. *Protocols for key establishment and authentication*. Springer-Verlag, 2003.

[3] E. Bresson, O. Chevassut, and D. Pointcheval. Dynamic group Diffie-Hellman key exchange under standard assumptions. In L. R. Knudsen, editor, *Advances in Cryptology — EUROCRYPT 2002, International Conference on the Theory and Applications of Cryptographic Techniques, Amsterdam, The Netherlands, April 28 – May 2, 2002, Proceedings*, volume 2332 of *Lecture Notes in Computer Science*, pages 321–336. Springer, 2002.

[4] M. C. Gorantla, C. Boyd, J. M. González Nieto, and M. Manulis. Modeling key compromise impersonation attacks on group key exchange protocols. *ACM Trans. Inf. Syst. Secur.*, 14(4):no. 28, 2011.

[5] L. Harn and C. Lin. Authenticated group key transfer protocol based on secret sharing. *IEEE Transactions on Computers*, 59:842–846, 2010.

[6] C.-F. Hsu, L. Harn, Y. Mu, M. Zhang, and X. Zhu. Computation-efficient key establishment in wireless group communications. *Wireless Networks*, 23:289–297, 2017.

[7] C.-F. Hsu, L. Harn, and B. Zeng. UMKESS: user-oriented multi-group key establishments using secret sharing. *Wireless Networks*, 26(1):421–430, 2020.

[8] International Organization for Standardization, Genève, Switzerland. *ISO/IEC 11770-5:2011, Information technology — Security techniques — Key management — Part 5: Group key management*, December 2011.

[9] M. Kim, N. Park, and D. Won. Cryptanalysis of an authenticated group key transfer protocol based on secret sharing. In J. J. Park, H. R. Arabnia, C. Kim, W. Shi, and J.-M. Gil, editors, *Grid and Pervasive Computing — 8th International Conference, GPC 2013 and Colocated Workshops, Seoul, Korea, May 9-11, 2013. Proceedings*, volume 7861 of *Lecture Notes in Computer Science*, pages 761–766. Springer, 2013.

[10] M. Kim, N. Park, and D. Won. Security analysis on a group key transfer protocol based on secret sharing. In J. J. Park, H. Adeli, N. Park, and I. Woungang, editors, *Mobile, Ubiquitous, and Intelligent Computing — MUSIC 2013, FTRA 4th International Conference on Mobile, Ubiquitous, and Intelligent Computing, September 4–6, 2013, Gwangju, Korea*, volume 274 of *Lecture Notes in Electrical Engineering*, pages 483–488. Springer, 2014.

[11] C. O. Kisty and S. R. Y. Saputra. Modification the Hsu-Harn-Mu-Zhang-Zu group key establishment protocol. `https://iceecs2018.org/wp-content/uploads/2019/10/paper_048.pdf`, 2018.

[12] C.-S. Laih, J.-Y. Lee, and L. Harn. A new threshold scheme and its application in designing the conference key distribution cryptosystem. *Information Processing Letters*, 32(3):95–99, 1989.

[13] J. Liu, Y. Wu, X. Liu, Y. Zhang, G. Xue, W. Zhou, and S. Yao. On the (in)security of recent group key establishment protocols. *The Computer Journal*, 60:507–526, 2017.

[14] C. J. Mitchell. The Hsu-Harn-Mu-Zhang-Zhu group key establishment protocol is insecure. arXiv:1803.05365 [cs.CY], `http://arxiv.org/abs/1803.05365`, March 2018.

[15] C. J. Mitchell. Security issues in a group key establishment protocol. *The Computer Journal*, 62(3):373–376, 2019.

[16] 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 T.-H. Kim, H. Adeli, D. Slezak, F. Eika Sandnes, X. Song, K.-I. Chung, and K. P. Arnett, editors, *Future Generation Information Technology — Third International Conference, FGIT 2011 in Conjunction with GDC 2011, Jeju Island, Korea, December 8–10, 2011. Proceedings*, volume 7105 of *Lecture Notes in Computer Science*, pages 309–315. Springer-Verlag, Berlin, 2011.

[17] J. Nam, M. Kim, J. Paik, and D. Won. Security weaknesses in Harn-Lin and Dutta-Barua protocols for group key establishment. *KSII Transactions on Internet and Information Systems*, 6(2):751–765, 2012.

[18] R. F. Olimid. On the security of an authenticated group key transfer protocol based on secret sharing. In K. Mustofa, E. J. Neuhold, A M. Tjoa, E. R. Weippl, and I. You, editors, *Information and Communicatiaon Technology — International Conference, ICT-EurAsia 2013, Yogyakarta, Indonesia, March 25–29, 2013. Proceedings*, volume 7804 of *Lecture Notes in Computer Science*, pages 399–408. Springer, 2013.

[19] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[20] Y. Sun, Q. Wen, H. Sun, W. Li, Z. Jin, and H. Zhang. An authenticated group key transfer protocol based on secret sharing. *Procedia Engineering*, 29:403–408, 2012.