

TECHNICAL RESEARCH REPORT

Security Analysis of a Distributed Common Secret Generation Procedure

by R. Poovendran and John S. Baras

CSHCN T.R. 99-16
(ISR T.R. 99-34)



The Center for Satellite and Hybrid Communication Networks is a NASA-sponsored Commercial Space Center also supported by the Department of Defense (DOD), industry, the State of Maryland, the University of Maryland and the Institute for Systems Research. This document is a technical report in the CSHCN series originating at the University of Maryland.

Web site <http://www.isr.umd.edu/CSHCN/>

Entitled:

Security Analysis of a Distributed Common
Secret Generation Procedure

Authors:

R. Poovendran and J.S. Baras

Journal:

Journal of Cryptology

Security Analysis of a Distributed Common Secret Generation Procedure

R. Poovendran, J. S. Baras
radha, baras@isr.umd.edu
Institute for Systems Research
University of Maryland
College Park, MD 20742

May 26, 1999

Abstract

In [1, 2], a distributed scheme allowing any number of members to compute a common secret without revealing individual secret was proposed. We present a security weakness of this protocol. In doing so, we show that any two members can collude and obtain the secret contributed by *middle* member in generating the common secret.

1 Introduction

In two party computation, if there is no external collaborator, the issue of collusion does not arise. The user collusion is a problem where two or more parties collaborate and compute a certain quantity that may be otherwise not accessible to them under the protocol construct. Hence, the collusion is a “failure state” of a given security protocol. The user collusion also is a very commonly ignored but potentially costly problem due to either the lack of awareness of it.

Some of the recently proposed multiparty computing schemes [5, 6, 1, 2] suffer from collusion and often fail if two members collaborate, regardless of the size of the group. Hence, from the security view point, these schemes don't scale well.

In [7], we presented analysis of the collusion problem in the recently proposed secure multicast schemes [5, 6]. In this paper, we present the collusion problem with the distributed secret computation scheme in [1, 2].

We first present the problem using an application as in [1, 2]. We then present the solution in [1, 2]. We then show how the collusion trivially breaks this system.

1.1 Problem Description

Consider a situation where there are n members each of whom have an individual secret. Problem is to find a scheme that will allow these set of members to compute the sum of individual secret without exposing it.

To use the application context in [1, 2], consider a class where there are n students. The class is supposed to have

worked on a problem on an average of T hours. The teacher wants to know the average amount of time spent by the class on the project.

Another application is where each of the n members have some valuable information that when added will lead to a joint secret. This valuable information may be the total amount of wealth in the group, or the total score of the group etc.

Another application is the need to compute a common secret with individual shares such that at the time of combining the shares, no individual share is exposed. We note that this requirement eliminates techniques based on polynomials being candidates [4]. This is due to the fact that at the time of joint secret construction, all the shares have to be exposed for proper polynomial interpolation.

1.2 Proposed Existing Solution in [1, 2]

In [1, 2], the following solution was proposed. We present the solution as a set of steps but the idea is the same.

1. Implicit indexing among members is assumed.
2. The first member generates a random number α and adds his/her secret γ_1 to it, and securely communicates $\alpha + \gamma_1$ to the second member.
3. For $i = 2 \dots n - 1$
member i adds its secret γ_i to the quantity $\alpha + \sum_{j=1}^{i-1} \gamma_j$ securely received from member $i - 1$ and securely communicates $\alpha + \sum_{j=1}^i \gamma_j$ to member $i + 1$.
4. Member n receives the quantity $\alpha + \sum_{j=1}^{n-1} \gamma_j$ securely communicated by member $n - 1$, adds its secret γ_n and securely communicates the result $\alpha + \sum_{j=1}^n \gamma_j$ to the first member.
5. First member removes the value α and extracts the common secret $\sum_{j=1}^n \gamma_j$.

This approach prevents a member from knowing the individual secret of any other member. The computations scale linearly as a function of group size in this approach. We now show the problem with this method.

1.3 The user collusion Problem

Let members i and $i + 2$ be collaborators. After computing the quantity $\alpha + \sum_{j=1}^i \gamma_j$, it securely communicates it to member $i + 1$. It also securely communicates the quantity (without the knowledge of member i) to member $i + 2$.

The unsuspecting member $i + 1$ computes the secret $\alpha + \sum_{j=1}^i \gamma_j$ and securely communicates it to member $i + 2$. Using these two quantities, member $i + 2$ can extract the secret of member $i + 1$ in a straight forward manner. Hence, the individual secret is not guarded against collusion in this approach. Implicit assumption of this formulation is the system be free of any collusion among group members.

2 Trusted Third Party

It is possible to add verifiable secret exchange component to make sure that all the members communicate the shares that they claim to be sending. Adding verifiable secret also will prevent the first member from cheating at the time of extracting the final joint secret. However the user collusion described in this paper is removable with a trusted third party or a benign “information theoretic helper”. In this approach the following steps are needed.

- Trusted third party provides individual pads to members.
- Individual members add their secrets to the pads.
- All the members combine their padded secrets using the procedure described earlier.
- The first member then returns the padded joint secret to the trusted third party.
- The trusted third party then removes the combined padding and return the joint secret to the members.

One can add verifiable secret generation to this procedure.

3 Conclusion

We identified a security weakness due to user collusion in a recently proposed distributed secret generation procedure. We showed that the scheme fails if two members collude. We also provided a trusted third party based common secret computing.

References

- [1] N. Koblitz, Algebraic Aspects of Cryptography, pp. 11-12, Springer-Verlag, New York, 1998.
- [2] N. Koblitz, “Cryptography as a teaching tool”, In Cryptologia, October, 1998.
- [3] R. Miller, personal communication, 1999.
- [4] A. Shamir, “How to Share a Secret”, *Communications of the ACM*, 22, pp. 612-613, 1979.
- [5] G. Caronni, M. Waldvogel, D. Sun, and B. Plattner, “Efficient Security for Large and Dynamic Groups”, In *Proc. of the Seventh Workshop on Enabling Technologies*, IEEE Computer Society Press, 1998.
- [6] I. Chang, R. Engel, D. Kandlur, D. Pendarakis, D. Saha, “Key Management for Secure Internet Multicast Using Boolean Function Minimization Techniques”, in Proceedings of IEEE Infocom’99.
- [7] R. Poovendran, and J. S. Baras, “An Information Theoretic Approach for Design and Analysis of Rooted-Tree Based Multicast Key Management Schemes”, in CRYPTO’99, August 1999, Santa Barbara, USA.