A Note on Collaborating Adversaries in the Srinathan-Kumar-Rangan Transmission Protocol

Ben Moss
School of Computer Science and Information Technology
University of Nottingham, UK

bxm@cs.nott.ac.uk

May 2003

Abstract - We show, by means of brief demonstration, that the underlying transmission protocol in [1] cannot determine dishonest paths when adversaries collaborate. Hence, this method of tolerating mixed adversaries on an asynchronous network is flawed.

Index Terms - Paths, Adversary Tracing, Secret Sharing, Collaborating Adversaries

1 Introduction

At Asiacrypt 2002, Srinathan, Kumar and Rangan presented a paper describing a method to communicate securely over a completely asynchronous incomplete network, tolerating mixed adversaries [1]. The method is based on a transmission protocol, which first identifies the set of *faulty paths* across the network, then proceeds to use the remaining *honest paths* for communication. However, there is a problem with the first part of the protocol that identifies the faulty paths.

In a network of n paths $p_0, p_1, \ldots, p_n$ from **A** to **B**, the transmission protocol allows **A** and **B** to determine the honest paths in the following way:

1. **A** splits a verifiable secret S into n shares $s_0, s_1, \ldots, s_n$
2. **A** sends s_i along path p_i to **B** (for $i = 0$ to n)
3. **B** tries to reconstruct S from $(s_0, s_1, \ldots, s_n)$
   - if S is okay then all paths $p_0, p_1, \ldots, p_n$ are honest
   - if S is not okay then **B** must assume that any number of shares may have been tampered with and so has the tuple $T = (s'_0, s'_1, \ldots, s'_n)$ where s_i may or may not be equal to s'_i (for $i = 0$ to n) and therefore sends T to **A** along all paths $p_0, p_1, \ldots, p_n$
4. Let T_i be the tuple **A** receives from **B** along the path p_i (for $i = 0$ to n)
   - if **A**'s original share s_j matches the corresponding received share s'_j (for $j = 0$ to n) in tuple T_i (for $i = 0$ to n) then path p_j is honest, and so by comparison of the shares in T_j to the original shares, the honest and fault paths are correctly determined

2 Demonstration

We give two examples; firstly to demonstrate how the protocol works, and secondly to highlight why it is flawed.

2.1 Correct Example

The following example demonstrates that this protocol gives a correct result, even if n -1 paths are faulty and they collaborate:

1. Suppose there are 3 paths (n = 3) from **A** to **B**, two of which are faulty (p_0 and p_2)
2. **A** sends s_0, s_1 and s_2 along paths p_0, p_1 and p_2 respectively
3. As p_0 and p_2 are faulty, **B** actually receives T = (s'_0, s_1 and s'_2), and so the reconstructed secret is not okay
4. **B** sends T to **A** along p_0, p_1 and p_2
5. As p_0 and p_2 are faulty, they can modify their tuples (T_0 and T_2) to indicate no changes in their shares
6. **A** receives T_0 = (s_0, s_1 and s_2), T_1 = (s'_0, s_1 and s'_2), and T_2= (s_0, s_1 and s_2), which collectively indicate that p_1 is honest and p_0 and p_2 are faulty

Despite the faulty paths collaborating (by correcting each other's shares in their tuple), the tuple sent along the single honest path cannot be changed.

2.2 Counter Example

The above example showed how collaborating faulty paths, by changing their collaborative faulty tuple shares, can still be identified. However, they can also change the shares of honest paths in their own tuples. The following example demonstrates why this protocol fails when faulty paths collaborate to incriminate honest paths:

1. Again, suppose there are 3 paths (n = 3) from **A** to **B**, two of which are faulty (p_0 and p_2)
2. **A** sends s_0, s_1 and s_2 along paths p_0, p_1 and p_2 respectively
3. As p_0 and p_2 are faulty, **B** actually receives T = (s'_0, s_1 and s'_2), and so the reconstructed secret is not okay
4. **B** sends T to **A** along p_0, p_1 and p_2
5. As p_0 and p_2 are faulty, they can modify their tuples (T_0 and T_2) to indicate no changes in their shares, and changes in the honest shares
6. **A** receives T_0 = (s_0, s'_1 and s_2), T_1 = (s'_0, s_1 and s'_2), and T_2= (s_0, s'_1 and s_2), which collectively cannot determine any honest path

The tuples from the honest paths (T_1) indicate that the true faulty paths are faulty, but the tuples from the faulty paths (T_0 and T_2) indicate that the true honest paths are faulty.

Since the number of honest paths is unknown, irrespective of the protocol, then the only information that the protocol provides is the presence of at least one faulty path.

3 Conclusion

In conclusion, we have highlighted that the transmission protocol in [1] is flawed if adversaries collaborate. In particular, when collaborating adversaries conspire to hide their own share changes and incriminate honest paths' shares.

4 References

[1] Asynchronous Secure Communication Tolerating Mixed Adversaries, K. Srinathan, M. V. N. Ashwin Kumar and C. Pandu Rangan, Advances in Cryptology – ASIACRYPT 2002 (LNCS 2501), Springer-Verlag, 2002.