Source Authentication in Group Communication Systems

Xin Zhao
University of Michigan
1301 Beal Ave, Ann Arbor, MI, 48105, USA
zhaoxin@eecs.umich.edu

Atul Prakash
University of Michigan
1301 Beal Ave, Ann Arbor, MI, 48105, USA
aprakash@umich.edu

Abstract

Many group communication systems need to enforce a restriction that limits members are authorized to send messages to the group. Receivers therefore need to authenticate message sources before the received messages are accepted. Source authentication in peer-to-peer systems is trivial: the two communication parties can agree on one pair key and use this key to authenticate each other. However, because the group key is shared by all members in a group system, it is quite challenging to identify the sender and determine its authorization. Furthermore, if the authorization can be changed at run-time, source authentication problem can be even harder. This paper presents a source authentication technique called TTA scheme(Transitive Trust Authentication). TTA supports source authentication as well as dynamic authorization change. In addition, its computation and communication overhead is low.

1 Introduction

Many secure group communication systems only allow authorized members to send messages to the group. Therefore, a member must check the identity as well as the message source before accepting a received message. The source authentication process is trivial in a peer-to-peer communication system: the two communication parties can agree on one pair key and use it to authenticate each other. In a group system, the group key is shared by all members, which makes it challenging to identify the message source and determine its authorization. Furthermore, If the authorizations can be changed at run-time, the source authentication problem will be even harder.

Two most challenging questions are raised here:

1. **How to generate unforgeable validity proofs?** Because the group key is shared by all group members, symmetric encryption methods cannot generate unforgeable validity proofs. As an alternative, the digital signature technique could guarantee the source authenticity. However, its computational overhead is prohibitively high. How to generate unforgeable validity proofs is therefore quite challenging.
2. **How to support dynamic authorization change** It is desirable to allow dynamic authorization change. For example, a member's sending right can be revoked at run-time. However, the receiver can authenticate the message correctly only if they have the updated authorization. In a large system, it is dif£cult and expensive to distribute the policy changes to all group members in a reliable and secure way.

2 Existing Solution

Several reasonable solutions have been proposed to support single source authentication in group systems.

Canetti et al. proposed a k-MAC authentication scheme[1] which let the sender use k MAC keys to compuate k MACs for each message. Each receiver hold a subset of the k MAC keys to verify the received messages. One major problem of this scheme is its computation and communication overhead since the sender needs to compute and send multiple MACs.

Gennaro and Rohatgi proposed a MAC Chain technique[2] to embed the MAC of a packet into its previous packet. If the initial packet is authenticated, the subsequent packets can be veri£ed at low cost. Its main shortcomings are its weak robustness to packet loss and lack of dynamic authorization change support.

TESLA [3] [4], proposed by Adrian Perrig et al., is quite ef£cient. However, it requires that each receiver loosely synchronize with the sender when it joins the group. For a large group, the time synchronization may incur substantial delay. Moreover, TESLA does not support dynamic authorization change.

This paper presents the TTA(Transitive Trust Authentication) mechanism. It supports source authentication

Proceedings of the 14th International Workshop on Database and Expert Systems Applications (DEXA'03)
1529-4188/03 $17.00 © 2003 IEEE

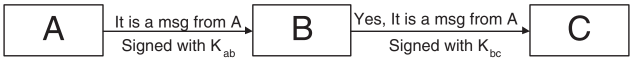


Figure 1. Transitive Trust Relation

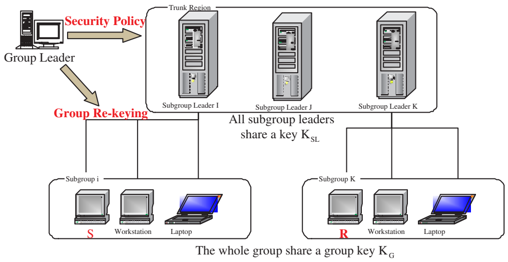


Figure 2. Infrastructure of TTA mechanism

as well as dynamic authorization change. In addition, we will show that the computation and communication overhead of TTA is low.

The rest of the paper is organized as follows: Section 3 gives the sketch of the TTA scheme, Section 4 describes one critical technique used in TTA. The performance analysis is put in Section 5. Finally, we reach the conclusion at Section 6.

3 Transitive Trust Authentication Scheme

3.1 Transitive Trust Relation

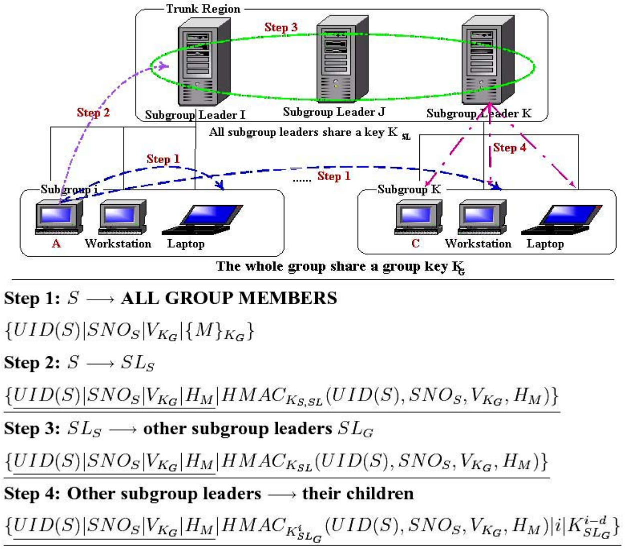
The basic idea of TTA is is illustrated in Figure 1: suppose a member A wants to send a message to C, but C accepts this message only if it believes that A is a valid sender. However, C might have no idea of A's authorization. To help C authentication A, a mutually trusted member B can be sets up to authenticate A and certify the validity of A to C. Because C trusts B, it then trust A. This kind of trust relatioin is called textittransitive trust relation. The trust transition process is illustrated in Figure 1.

TTA extends the basic idea and organizes a multicast system as shown in Figure 2, four roles exist in the TTA organized system.

1. **Group Leader:**A many-to-many multicast group has one *group leader*, the group leader takes care of group administrative operations such as group re-keying, group policy changing, etc.;

2. **Subgroup Leader:** As Figure 2 shows, the group is divided into multiple subgroups. Each subgroup has one *subgroup leader*. The subgroup leaders trust one another. They authenticate and notarize the group messages sent in/out their subgroups. One shared key K_{SL} is used by all subgroup leaders to certify the validity of messages to other subgroup leaders. Note that the subgroup leaders are not regarded as group members, they do not have the group key K_G, therefore, they cannot access the message content.

3. **Group Member:** Group members include *senders* and *receivers*. Each group member must be a receiver, but only members who are authorized to send messages are senders. Each group member belongs to one and only one subgroup, it only accept messages notarized by it own subgroup leader. Each sender S share one paired key $K_{S,SL}$ with its subgroup leader SL_S.



Step 1: $S \longrightarrow$ **ALL GROUP MEMBERS**

$\{UID(S)|SNO_S|V_{K_G}|\{M\}_{K_G}\}$

Step 2: $S \longrightarrow SL_S$

$\{\underline{UID(S)|SNO_S|V_{K_G}|H_M}|HMAC_{K_{S,SL}}(UID(S), SNO_S, V_{K_G}, H_M)\}$

Step 3: $SL_S \longrightarrow$ **other subgroup leaders** SL_G

$\{\underline{UID(S)|SNO_S|V_{K_G}|H_M}|HMAC_{K_{SL}}(UID(S), SNO_S, V_{K_G}, H_M)\}$

Step 4: Other subgroup leaders $\longrightarrow$ their children

$\{\underline{UID(S)|SNO_S|V_{K_G}|H_M}|HMAC_{K^i_{SLG}}(UID(S), SNO_S, V_{K_G}, H_M)|i|K^{i-d}_{SLG}\}$

Figure 3. Source Authentication Procedure in Many-to-Many Scenarios

The TTA system has the following trust relations:

- All group components, including senders, receivers and subgroup leaders, trust the group leader.
- The subgroup leaders trust one another.
- All group members trust their own subgroup leaders.

3.2 Sketch of TTA scheme

Figure 3 illustrates how TTA supports source authentication in many-to-many scenarios:

1. Step 1: The sender $S \longrightarrow$ other group members:

$$\{UID(S)|SNO_S|V_{K_G}|\{M\}_{K_G}\}$$

The sender S sends the message content packet to all group members. This packet includes: user ID

$UID(S)$, sequence number SNO_S, group key version number V_{K_G} and the encrypted message content $\{M\}_{K_G}$.

Upon receiving this packet, the receivers buffer this message and wait for the corresponding message certificate. To avoid unlimited waiting, a timer will be set for each buffered message, the valid message certificate must arrive before the timer expires, or else the buffered message will be dropped.

2. Step 2: The sender $S \longrightarrow$ its subgroup leader SL_S:

$$\{\underline{C}|HMAC_{K_{S,SL}}(\underline{C})\}$$

$$\underline{C} = \{UID(S)|SNO_S|V_{K_G}|H_M\}$$

S first generates the ***Message Certificate*** $\underline{C}$ of this message. The message certificate uniquely maps to the corresponding message. If the source of a message certificate is valid, the source of the corresponding group message must be valid. Receivers then use the hashed value H_M to verify the integrity of message content. After generating the message certificate, it uses the paired key $K_{S,SL}$ to generate $HMAC_{K_{S,SL}}(\underline{C})$ which can be used by its subgroup leader to identify S.

The message certificate as well as its HMAC is sent to S's subgroup leader SL_S. SL_S authenticates the validity of S through two steps: first, SL_S checks S's authorization record to see whether it has appropriate sending right. second, SL_S identify the sender S by checking the attached HMAC value. Through these two checks, SL_S not only identifies the data source, but also enforces dynamic authorization control.

3. Step 3: The subgroup leader $SL_S \longrightarrow$ other subgroup leaders SL_G:

$$\{\underline{C}|HMAC_{K_{SL}}(\underline{C})\}$$

If the message certificate is verified to be valid, the subgroup leader SL_S will certify the validity of the message certificate to all other subgroup leaders. SL_S replace the HMAC value $HMAC_{K_{S,SL}}(\underline{C})$ with a new HMAC $HMAC_{K_{SL}}(\underline{C})$, which is computed with the K_{SL} that is known by all subgroup leaders only.

The certificate as well as the new HMAC will be distributed to all other subgroup leaders. Other subgroup leaders can check the HMAC to see whether the certificate is notarized by a subgroup leader. Because all subgroup leaders trust one another, if HMAC is verified to be generated by one subgroup leader, the corresponding message certificate will be regarded as valid by all other subgroup leaders.

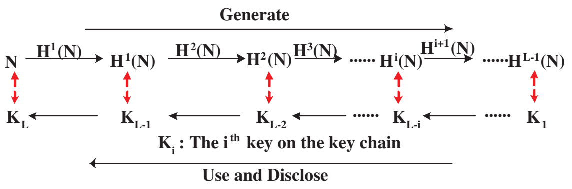


Figure 4. Self-Authenticated One Way Key Chain

4. Step 4: Other subgroup leaders $\longrightarrow$ their children:

$$\{\underline{C}|HMAC_{K^i_{SL_G}}(\underline{C})|i|K^{i-d}_{SL_G}\}$$

If the certificate is verified to valid, other subgroup leaders will certify the validity of the message certificate to their children. Upon receiving the notarized certificate, the receivers first check whether this notarized certificate came from their subgroup leaders. How can receivers authenticate whether the certificate originated from their subgroup leader is actually a one-to-many source authentication problem and will be discussed in next section.

If the notarized message certificate is verified to be from its subgroup leader, receivers will be confident that the source of the certificate is valid. The source of the corresponding message will then be regarded as valid. The receivers can then use the hash value H_M, which is included in the message certificate, to verify the integrity of message content. That finalizes the whole message verification procedure.

4 Source Authentication between Receivers and their Subgroup Leader

TTA uses TESLA[3, 4, 5] protocol to achieve source authentication between Receivers and their subgroup leader. It requires that each receiver synchronize with its subgroup leader. After the synchronization, the receivers can use their local time infer the upper bound of the subgroup leader's local time.

Bootstrap Stage As illustrated in Figure 4 shows, at the subgroup leader's bootstrap stage, the subgroup leader uses the self authenticated one-way key chain to generate a one-way key chain: the subgroup leader randomly pick a number N and repeatedly applying a one-way hash function H for $\ell - 1$ times. It then gets a hash value chain: $H^0(N), H^1(N), ...H^{\ell-1}(N).$, where $H^i(N) = H(H...(H(N))$. The self-authenticated key chain is generated by putting the one-way hashed chain in reverse order. If we use K_i to represent the i^{th} key on the key chain,

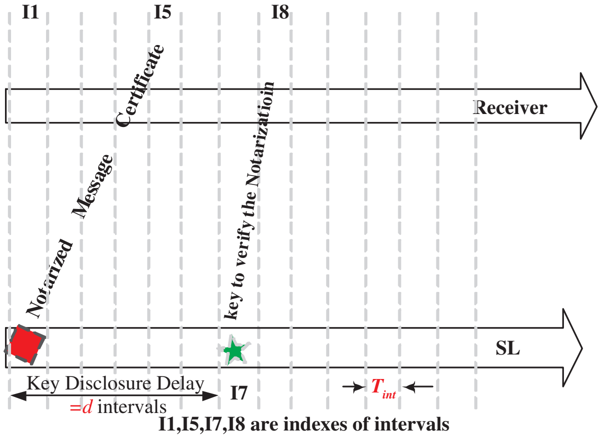


Figure 5. Source Authentication between a receiver and its subgroup leader

then $K_i = H^{\ell-i}(N)$. If K_i is verified to be valid, the validity of the key K_l can be verifed by check whether $K_l = H^{i-l}(K_i) \quad l < i$. Therefore, the cost of key authentication is very low.

The subgroup leader's side As Figure 5 shows, the subgroup leader SL splits time into even intervals, each interval is T_{int} long. Interval 0 starts at time T_0, interval i starts at $T_i = T_0 + i * T_{int}$. At any time point, the subgroup leader should be in one interval i. Next,the subgroup leader has a self-authenticated one-way key chain $K^0_{SL}, K^1_{SL}, ...K^{\ell-1}_{SL}$. Each key K^i_{SL} will be used to compute HMACs for message certificates only within time interval i. Each used key will be kept secret for several(for example, d) intervals. After d intervals, SL discloses the key. The validity of the disclosed key can be verifed with the self-authenticated key chain technique.

The receiver's side $\underline{C}$ represent the message certificate $\{UID(S)|SNO_S|V_{K_G}|H_M\}$. When the receiver receives notarized certificate N_j sent in interval i at local time t_r, the packet N_j should be like this:

$$\{\underline{C}|\underline{HMAC_{K^i_{SL}}(C)|i|K^{i-d}_{SL}}\}$$

The receiver takes the following steps to check the notarized certificate arrived *safely*:

1. Infer the time interval i the subgroup leader was in when it sent out the notarized certificate. Since i is included in the packet, the receiver can verify the i by checking the authenticity of the disclosed key K^{i-d}_{SL}. If the last authenticated key is K^l_{SL}, the authenticity of K^{i-d}_{SL} can be determined through checking whether $K^l_{SL} = H^{i-d-l}(K^{i-d}_{SL})$.
2. Compute the upper bound of the sender's current time interval x. Because the receiver is loosely time synchronized with the sender, the receiver can compute the upper bound on the sender's clock $t_s = t_r + \Delta$, and consequently calculate the maximal interval with $x = \lfloor (t_s - T_0)/T_{int} \rfloor$.
3. Compare $i + d$ and x, it $x < i + d$, this key used to notarize this message certificate must be still a secret and the notarized certificate arrived *safely*. Otherwise, probably the key have been disclosed, and the certificate notarization could be forged with the revealed key.

If the notarized certificate is assured to arrive safely, the receivers will buffer it and wait for disclosure of the key K^i_{SL} to authenticate the packet source.

After the key K^i_{SL} is disclosed, the receiver first check the validity of K^i_{SL}. If the key K^i_{SL} is valid, it can be used to check the message certificate's HMAC $HMAC_{K^i_{SL}}(C)$. If it is valid, the source of the message certificate is valid, and the source of the corresponding message is also valid. We follow TELSA's suggestion to set d as $\lceil RTT/T_{int} \rceil + 1$, here, RTT is a reasonable upper bound on the round trip time between the receiver and the subgroup leader.

5 Performance analysis

Communication Overhead If TTA scheme uses 80 bit HMAC-MD5 to generate HMACs, both the disclosed key and hashed value are 10 bytes long. The user ID, sequence number, key version number, and interval index are 4 bytes long. The sender's subgroup leader needs to send/receive message certificates for 3 times. The communication cost is 128bytes/message. Other subgroup leaders only need to send/receive the certificate twice. The communication cost is 90bytes/message.

Computation Overhead We also use simuation program to evaluate TTA's computation overhead. The simulation program is written in C++ and runs on a 1.4GHz Pentium IV Linux PC, cryptography library is OpenSSL[6]. The simulation results show that the experimental machine can process 61900 messages per second. If the average length of messages is 1000 bytes, The experimental machine can process the group messages of around 61.9M(Bytes/second).

The generation of key chain is efficient too. If the interval length is 0.1 seconds, computing a key chain long enough to notarize certificate for one hour only takes the subgroup leader around $0.12(seconds)$.

Verification Latency and Probability Verification latency is defined as follows: when a group member receives a group message, it records this time as the starting point.

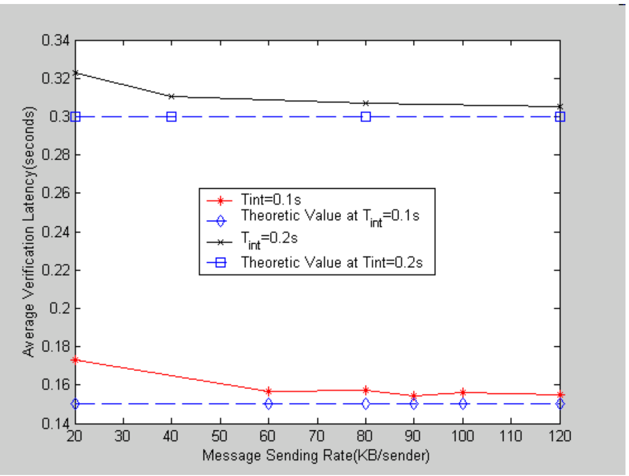


Figure 6. Average Authentication Latency

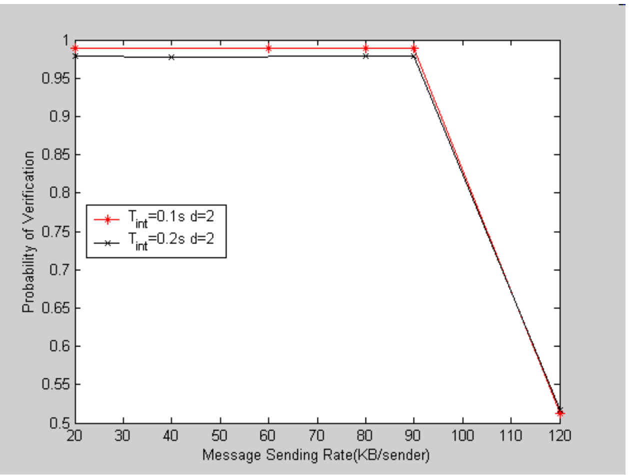


Figure 7. Verification Probability

The time that the message source is verified is recorded as the ending point. The verification latency is the time elapsed between the starting point and ending point. The dominate factor of verification latency is the notarization key disclosure delay. Assuming that the notarized message certificates arrive in constant rate. Under such condition, the average key disclosure latency should be equal to $T_{int} * (d - 0.5)$.

The simulation program is built up on the NS2(Network Simulator 2) [7] platform, the hardware is a 1.4GHz Pentium IV Linux PC. In the simulation environment, 200 members are divided evenly into two subgroups. 20 out of these 200 members are senders. 10 senders for each group. Each member has a 5MB connection with its subgroup leader, and leaders maintain an 1MB connection with each other. The network transmission latency is 3ms.

The simulation result shown in figure 6 confirms our theoretic analysis. Moreover, we can that different message sending rates has little effects on average verification latencies. The dominant factor is still the key disclosure delay.

The verification probability is the ratio of the number of messages verified to the number of messages sent. Because the successful verifications of TTA scheme rely on the delivery of both the message and its certificate, when network is congested, the verification failure rate will be doubled. We increase the message sending rate to saturate the network and expect to see that TTA's verification probability will drop fast when network is congested. Figure 7 confirmed that the verification probability is very high when network bandwidth is still available. However, when the traffic saturates the network, the verification probability drops fast. How to make the TTA system more fault tolerable will be our next research topic.

6 Conclusion

TTA converts the procedure of many-to-many source authentication into two peer-to-peer source authentication procedures and a one-to-many source authentication procedure. With this conversion, TTA can offer source authentication, dynamic security policy enforcement, and minimal overhead.

References

[1] Ran Canetti, Juan Garay, Gene Itkis, Daniele Micciancio, Moni Naor, and Benny Pinkas. Multicast security: A taxonomy and some efficient constructions. In *INFOCOMM'99*, 1999.

[2] Rosario Gennaro and Pankaj Rohatgi. How to sign digital streams. In *Advances in Cryptology C CRYPTO 97*, 1997.

[3] Adrian Perrig, Ran Canetti, J.D̃. Tygar, and Dawn Song. The tesla broadcast authentication protocol. *RSA CryptoBytes*, 5(Summer), 2002.

[4] Adrian Perrig, Ran Canetti, J. D. Tygar, and Dawn Xiaodong Song. Efficient authentication and signing of multicast streams over lossy channels. In *IEEE Symposium on Security and Privacy*, pages 56–73, 2000.

[5] A. Perrig, R. Canetti, D. Song, and J.D. Tygar. Efficient and secure source authentication for multicast. In *Network and Distributed System Security Symposium, NDSS '01*, February 2001.

[6] The OpenSSL Project. http://www.openssl.org.

[7] The Network Simulator - ns-2. http://www.isi.edu/nsnam/ns.