Trust and Security: a new look at the Byzantine Generals problem

Mike Burmester, Yvo Desmedt, and Gregory Kabatianski

Abstract. Secure communication in an open dynamic network in the presence of a malicious adversary is central to secure distributed computation. We address this issue. First we link the notion of trust with security, by regarding trust as a mechanism for authentication. The trust-graph is a graph with vertices the processors, whose edges correspond to trusting pairs of processors. Our model is computational, and we assume that there is a bound u on the number of faulty (malicious) processors, and that the trust-graph is $(2u+1)$-connected.

In this paper we focus on the case when the trust depends on shared secret keys and conventional authentication is used (without signatures). We use a modular approach in which authentication is based on a two-party authentication protocol. We regard this as a primitive. It is easy to see that if the sender and receiver both know the trust-graph then they can communicate securely (as securely as the primitive). However if only the sender knows the trust-graph, then we do not know of any way of finding secure paths, other than by exhaustive search. The problem of secure communication for this case is related to two **NP**-complete problems. It is not clear if it is **NP**-complete.

1. Introduction

Secure communication in an open and dynamic network system (such as the internet) in the presence of a malicious adversary is central to secure distributed computation. By using an authentication protocol the power of the adversary can be restricted, however the problem is that the sender may not share a secret key with (or know the public-key of) the receiver. Finding authentication paths through intermediary processors may not be possible if the sender does not know which pairs of processors can authenticate messages. In this paper we address this issue. We motivate our discussion by considering an extension of the classic Byzantine generals problem which is open and dynamic: generals come and go, with newcomers not recognized by some, those staying not knowing who left.

First we discuss the notion of trust, from an abstract point of view, by regarding it as key for authentication mechanisms, or as a probability to forward such keys, or more generally as a boolean expression. We represent the trust relationship between processors by a trust-graph whose edges correspond to trusting pairs. We then consider the problem of secure communication in an open dynamic network

1991 *Mathematics Subject Classification.* Primary 68M10, 90B12, 68M15; Secondary 68R10, 68Q25.

system which is reliable and synchronous. We use a computational model, and assume that the number of faulty (malicious) processors is bounded by u, and that the trust-graph is $(2u + 1)$-connected.

In this paper we focus on the case when the trust depends on shared secret keys and conventional authentication is used. We use a modular approach, with the authentication based on a two-party authentication protocol which we regard as a primitive. The reason for this is practicality: there are currently available several two-party authentication protocols for which we have a good tradeoff between efficiency and security (*e.g.* [**1**]). It is easy to see that the sender and receiver can communicate securely if they both know the trust-graph. In this case the security is as strong as the primitive. However if the trust-graph is known to the sender but not the receiver, then the problem of secure communication is related to two **NP**-complete problems. It is not clear whether this problem is **NP**-complete. Indeed we do not know of any way of solving it, other than by exhaustive search.

The organization of this paper is as follows. In Section 2 we motivate our general model of trust and secure communication using an extended version of the Byzantine generals paradigm. In Section 3 we consider various models of trust, and in Section 4 we address the issue of secure communication for various scenaria. We conclude in Section 5 with remarks.

Related work. There are many papers in the literature which analyze various aspects the notion of trust. Recent work, similar to ours, is by Beth-Borcherding-Klein [**3**] and Maurer [**12**]. However there are differences with our model, and our results are more general. Reiter-Stubblebine [**15**] consider trust-paths, for authentication, as we do. Our results are for a general model. Part of our work is based on earlier results presented at DIMACS [**5**].

Perlman [**10**, pp. 462–465] describes a different jamming attack and a solution based on certificates originating from a certifying authority. We do not have a certifying authority and so her solution can not be used.

2. The Byzantine generals paradigm: a motivation for a general model of trust and secure communication

In the classic Byzantine generals paradigm [**14, 11**] a city is surrounded by Byzantine forces and their generals. Generals can communicate either directly, or indirectly through other generals, orally or in writing using messengers. The messengers are reliable and loyal. A general wants to send a message to the others. However some generals are traitorous and want to prevent the other generals from agreeing on which message was sent. These generals may conspire and follow a malevolent plan.

Recognition, whether oral or written, can be regarded as a bond of *trust* which links the parties involved: if general g_i recognizes general g_j (or his signature), then g_i trusts that g_j is who he is suppose to be (whether loyal or traitorous). The trust among the generals can be represented by a *trust-graph* whose vertices g_i correspond to the generals, and for which (g_i, g_j) is an edge if g_i recognizes g_j (the edge is directed if the trust is not mutual). In the classic Byzantine paradigm, each general recognizes the others (or their signatures) and there is complete interconnection. That is, the trust-graph is a complete graph. A weaker requirement would be near-completeness: one or more of the generals may be "trusted third parties", *i.e.* known to be loyal and recognized by (almost) all the others. However even this setting is not suitable for large open and dynamic networks such as the internet. In our extended paradigm we allow for the following aspects:

- *Open network*: No *a priori* assumption is made about the trust-graph of the generals, *i.e.* about who (or whose signature) is recognized by whom. There may be several generals which a particular general has not met before (or whose signature he does not recognize).
- *Dynamic setting*: No *a priori* assumption is made about knowledge of the trust-graph by the generals. New generals may join the task force. A general may not recognize them (or their signatures), and may not know any other general who recognizes them (or their signatures). If generals leave, others may not be aware of this.
- *Subset consensus*: A consensus may be required from a subset of loyal generals. If the subset consists of a sender and receiver only, and both are loyal, we require secure communication (this is not trivial in the extended paradigm).

In this paradigm, communication between generals g_{i_1} and g_{i_n} may only be possible indirectly through intermediary generals who trust each other. In an oral communication (authentication without signatures) the messages sent are of type: "general $g_{i_{n-2}}$ told me that, general $g_{i_{n-3}}$ told him that, ... , general g_{i_1} (the sender) said, attack". These should include the path to be followed: $g_{i_1}, g_{i_2}, \ldots, g_{i_n}$, as specified by the sender (and a sequence number). The message would be delivered orally to general g_{i_n} (the receiver) by general $g_{i_{n-1}}$. All pairs of generals $(g_{i_j}, g_{i_{j+1}})$ in the trust-path should recognize each other. With written messages, general g_{i_1} (the sender) signs his message, and then in turn, each general g_{i_j} in the path signs the message, to confirm its authenticity. In this case, each general $g_{i_{j+1}}$ must recognize the signature of the previous signer g_{i_j}. In both cases, only the last authentication step can be verified by the receiver. So traitorous generals may create bogus paths.

Applications.

1. *Secure network communication.*

2. *Fault-tolerant network systems.* The classic example is of a flight control system which uses the feedback from various physical sensors. The sensor values have to be distributed to each processor. The data can be replicated, but the replicates may not be identical. For fault-tolerance, the system should operate in the presence of faults (which could be malicious).

3. Models for trust

The notion of trust is intangible, even though it is an essential ingredient for the design of secure protocols. Trust comes in different flavors. If Bob shares a secret key with Alice then he trusts that all messages authenticated with this key are Alice's. If Bob knows Alice's public-key then he trusts that all signed messages that he can verify with this key are Alice's. In both cases the trust of Bob is based on the knowledge of a key: it is symmetric in the first case, but not necessarily so in the second. Although trust is usually assumed to be transitive [**4**], there is no reason why this should be so in the general case. In the models we consider, we deal with a very general kind of trust which is not absolute but *relative*. For example, if the trust is based on physical recognition then the fact that Bob recognizes Alice, and that Carol recognizes Bob, does not imply that Carol recognizes Alice. If Alice tells Bob that she believes statement x, then Bob trusts that: "Alice believes x". Bob's trust is based on his ability to recognize others. If Bob tells Carol that "Alice believes x" then Carol trusts that: "Bob said that, Alice believes x" is true. Not that: "Alice believes x" is true. Indeed Bob may have lied.

The trust-graph. It is futile to attempt to define in any particular way the trust relationship. Instead we describe several models for it. First we represent it by a graph, the *trust-graph*, whose vertices $A, B, \ldots$ correspond to the processors, with (A, B) an edge if B trusts A. The edge is undirected if the relation is symmetric, otherwise it is directed, see Figure 1. In Figure 2 there are two trust paths from A to B.

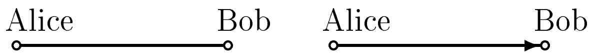

FIGURE 1. Bob trusts Alice: the symmetric and non-symmetric case.

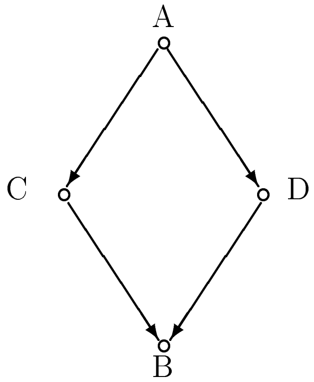

FIGURE 2. The trust of Bob in Alice is sustained by two trust-paths.

3.1. A Boolean model. We view the trust between two processors as a Boolean expression induced by the trust-graph. In this model the vertices are Boolean variables. If B and A are adjacent, the trust of B in A is 1. In Figure 3 the trust of B in A is: $C \wedge D$.

FIGURE 3. The trust of B in A with one trust-path.

For Figure 2 the trust of B in A is the Boolean expression: $C \vee D$. In Figure 4 there are three intersecting paths from A to B. In this case the trust of B in A is:

$$(C \wedge E) \vee (D \wedge F) \vee (C \wedge F).$$

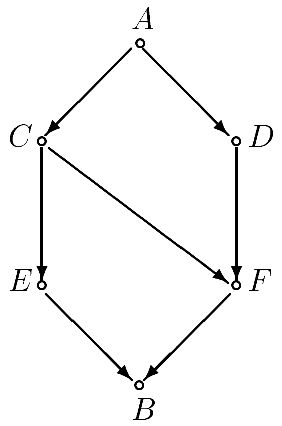

FIGURE 4. The trust of B in A with overlapping trust-paths.

3.2. A Probabilistic model. In this model trust is a probability. We assign to each vertex X of the trust-graph, independently, a probability p_X. We regard p_X as the probability that process X will forward correctly the public-key of a neighbor. In this model the trust of B in A is the probability that B will get the correct key from A. For the graph in Figure 2, the probability that B will be get the correct key from A is: $p_C + (1 - p_C)p_D$. Similarly the probability that B will be get the correct key from A in Figure 3 is: $p_C \cdot p_D$. In Figure 4 it is: $p_C p_F + (1 - p_C)p_D p_F + (1 - p_F)p_C p_E$.

3.3. A Deterministic model. For this model the trust is restricted to the trust-graph. As in the other models, the trust may be based on possession of a key. For example, B shall trust A if, B possesses a certified public-key of A. We have an undirected edge if A, B share a secret key.

We shall use this basic model in the following sections to study the security of networks. Our task will be to investigate the possibility of achieving secure communication in the presence of a malicious adversary.

4. Secure communication

It is claimed in the literature that secure communication can be achieved in a reliable synchronous network system if appropriate cryptographic tools are used (*e.g.*, [**11**]). The proof of this result uses techniques from network reliability, and deals with the case when the trust-graph is a complete graph, or a nearly complete graph.

In this paper we address the general case when the trust-graph is not necessarily known to the processors in the network, and when the only assumption on its topology is a certain degree of connectivity. We focus on the case when the trust depends on conventional authentication without signatures, *e.g.* with Message Authentication Codes [**13**], and use a modular approach in which authentication is based on a two-party authentication protocol. We take this as a primitive. The security of the scheme is therefore based on the security of the primitive.

Before we proceed with our analysis, we clarify our setting and the role of the adversary. In the next sections we shall make the following assumptions on the network system.

- *Reliability*: All processors can communicate reliably.[1]
- *Synchrony*: There is an upper bound on the possible delay of sending or receiving a message. We allow for messages to be delivered in different order than sent (communication asynchrony), but assume that all non-faulty processors will respond within a certain time (process synchrony). Processors which go to sleep are treated as faulty, and under the control of the adversary.[2]
- *Bounds on resources*: Authentication can be established in polynomial time (in the number of processors in the network and a security parameter). Also, all processors, including the adversary are polynomially time bounded.

The adversary E controls faulty processors, and may use a malicious plan. Her goal is to prevent the non-faulty processors from communicating securely. For this purpose E can employ various strategies. If a message is to be authenticated through a path from processor A to processor B involving several intermediary processors, and if one of these is faulty, E can create bogus paths to B through which a bogus message is authenticated.

To achieve secure communication in this scenario, we put a bound u on the number of faulty processors, and have the sender authenticate the message through more than $2u$ vertex-disjoint paths. This is a standard technique used in network reliability. In the following sections we shall see how, and when, this technique can be used for our purpose.

4.1. The trust-graph is known to the sender and receiver. In this case we get secure communication if the connectivity of the trust-graph is greater than $2u$. This result holds for both the public-key case and the secret-key case. For the sake of simplicity we assume that the trust is based on sharing secret keys, and that a conventional cryptosystem is used as the primitive for authentication. Let k_{XY} be the secret key shared by X and Y.

ALGORITHM 4.1. Processor A authenticates a message m to B
Suppose that the number of faulty processors is no more than u and that, if A, B are non-adjacent, there are at least $(2u+1)$ vertex-disjoint paths from A to B.

1. If A, B are adjacent, A authenticates m to B directly. Otherwise A finds $2u+1$ vertex-disjoint paths using an appropriate deterministic (max-flow based) algorithm [**8**], and chooses the first $2u+1$ paths (*e.g.* in lexicographical order),

$$\pi_i = (A = A_{i_0}, A_{i_1}, A_{i_2}, \ldots, A_{i_{t(i)+1}} = B), \quad i = 1, 2, \ldots, 2u+1,$$

and authenticates to each processor A_{i_1} the pair (π_i, m).[3] The message m includes a sequence number.

[1] We assume that appropriate anti-jamming techniques are used to prevent the adversary from jamming messages.

[2] The adversary can always arrange for faulty processors to sent bogus messages. However he cannot corrupt a message sent by a non-faulty processor, or prevent it from being received.

[3] If Message Authentication Codes [**13**] are used, A sends $(\pi, m, MAC(k_{A,A_1}, \pi_i, m))$ to A_{i_1}.

2. Each processor A_{i_j} different from A and B which receives (π_i, m) authenticated by $A_{i_{j-1}}$ accepts it, if it has been authenticated[4] with the shared key $k_{A_{i_{j-1}},A_{i_j}}$, and if π_i is a path in the trust-graph, and will authenticate (π_i, m) to the next processor $A_{i_{j+1}}$ in the path. If A_{i_j} does not accept (π_i, m), it will not forward it.
3. The receiver B finds the same $2u+1$ vertex-disjoint paths as A, using the deterministic algorithm of A. When B receives (π_i, m) authenticated by $A_{i_{t(i)}}$, it will accept it, if it was authenticated with the key $k_{A_{i_{t(i)}},B}$ that B shares with it, and if π_i is one of the computed paths. B obtains the message by majority vote on the received authenticated messages.

CLAIM 4.2. *Suppose that the conditions in Section 4 hold, and that the trust-graph is known to A and B. If A, B are adjacent, or if there are $2u+1$ vertex-disjoint paths from A to B, where u is an upper bound on the number of faulty processors, then A and B can communicate securely.*

REMARK 4.3. The communication is as secure as the primitive two-party authentication system. It is interesting to compare this protocol with other protocols for this scenario (when the trust-graph is known), *e.g.* the Secure Message Transmission protocol in [**7**], for which we have perfect secrecy and perfect resiliency.

4.2. Conventional authentication (without signatures): the trust-graph is known to the sender but not the receiver. Suppose that the number of faulty processors is no more than u, and that a sender A authenticates to (a non-adjacent) B through $2u+1$ vertex-disjoint paths a message m. Then B will receive pairs of type (π, m), where $\pi = (A = A_0, A_1, \ldots, A_{t+1} = B)$ is a path from A to B in the trust-graph, authenticated as in Algorithm 4.1. In particular, non-faulty processors A_i in the path π will verify that (π, m) came from or via A_{i-1} by checking its authenticity, using the shared key. If this fails, (π, m) will not be forwarded nor accepted.

Let us now discuss bogus paths, *i.e.*, paths originating from a faulty processor, say A_{i-1}, which claims that it originated from A_0, via some A_1, etc (or some other subpath leading to A_{i-1}). A message traveling via a bogus path will be forwarded by a non-faulty processor A_i provided that it is followed by A_{i+1} in the specified bogus path where (A_i, A_{i+1}) is an edge, that (A_{i-1}, A_i) is an edge in the trust path, and that A_{i-1} has authenticated (π, m) to A_i using the key they share. The actual message m (without the path specification) may be identical to an authentic one, but sent via a bogus path, or may be different. If the message m originated from the sender A, *i.e.* if *it is authentic*, then it will be received by B through at least $u+1$ disjoint paths, and possibly many other bogus paths. Bogus paths must include at least one faulty processor. So, if the message did not originate from A, *i.e.* if it *is not authentic*, its path must go through at least one of a set of at most u vertices corresponding to the faulty processors.

The problem of detecting authentic messages corresponds to the **NP**-complete Set Packing problem [**9**, #SP3 p.221]. The problem of detecting non-authentic messages corresponds to the **NP**-complete Hitting Set problem [**9**, #SP8 p.222]. However, since we know that the paths will be restricted to one type or the other

[4] With Message Authentication Codes, processor $A_{i_{j-1}}$ checks that $k_{A_{i_{j-1}},A_{i_j}}$ was used in $MAC(k_{A_{i_{j-1}},A_{i_j}}, \pi_i, m)$.

(the number of faulty processors is at most u), it is not obvious to conclude whether our problem is **NP**-complete.

We do not know how to solve this problem, other than by exhaustive search, and definitely not when the sender also does not know the network graph, as is often the case in large open networks. We doubt if secure communication is possible when conventional authentication is used with symmetric keys, if the trust-graph is not known and if several insiders are dishonest.

5. Conclusion

We have analyzed the notion of trust in the context of network security by regarding it as a mechanism for authentication, and addressed the issue of secure communication in an open and dynamic network, using a computational model.

We have shown that if trust depends on shared secret keys and there are no more than u faulty processors, and if the trust-graph is $(2u+1)$-connected and known to the sender but not the receiver, then the problem of secure communication is related to two **NP**-complete problems. We do not know of any way of solving this problem, other than by exhaustive search.

The case when the trust is based on public keys and the trust-graph is not known to the receiver will be discussed elsewhere.

Acknowledgement. We thank Mike Reiter and Stuart Stubblebine for discussions about certificates.

References

[1] M. Bellare and P Rogaway, Entity Authentication and Key Distribution, Proceedings, *Advances in Cryptology - Crypto '93, Lecture Notes in Comput. Sci. vol. 773*, D. R. Stinson, Ed., Springer-Verlag, 1994, pp. 232–249.
[2] D. Bertsekas and R. Gallager, Data Networks, Prentice-Hall, NJ, 1992.
[3] T. Beth, M. Borcherding, and B. Klein, Valuation of trust in open networks, Proceedings, *Computer Security - Esorics 94, Lecture Notes in Comput. Sci. vol. 875*, D. Gollmann, Ed., Springer-Verlag, 1994, pp. 3–18.
[4] M. Burrows, M. Abadi and R. Needham, A logic of authentication, *ACM Trans. on Comput. Systems*, **8**(1) (1990), 18–36.
[5] Y. Desmedt and M. Burmester. Linking trust with Network reliability. Presented at DIMACS Workshop on Trust Management in Networks, South Plainfield, New Jersey, Sept 30 - Oct 2, 1996.
[6] D. Dolev. Unanimity in Unknown and Unreliable Environment, *IEEE 22nd Symp. on Found. of Comput. Science*, Oct 1981, pp. 159–168.
[7] D. Dolev, C. Dwork, O. Waarts and M. Yung. Perfectly Secure Message Transmission. *Journal of ACM*, **40**(1) (1993), 17–47.
[8] S. Even. *Graph algorithms.* Computer Science Press, Rockville, Maryland, 1979.
[9] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman and Company, San Francisco, 1979.
[10] C. Kaufman, R. Perlman and M. Speciner, *Network Security*, Prentice-Hall, Englewood Cliffs, New Jersey, 1995.
[11] L. Lamport, R. Shostak and M. Pease, The Byzantine Generals Problem, *Journal of ACM* **32**(2) (1982), 374–382.
[12] U. Maurer. Modeling public-key infrastructure, Proceedings, *Computer Security - Esorics 96, Lecture Notes in Comput. Sci. vol. 1146* E. Bertino, H. Kurth, G. Martella, E. Montolivo, Eds., Springer-Verlag, 1996, pp. 325–350.
[13] A. Menezes, P.C. van Oorschot and S.A. Vanstone, Handbook of Applied Cryptography, CRC Press, 1996.

[14] M. Pease, R. Shostak and L. Lamport, Reaching Agreement in the Presence of Faults, *Journal of ACM* **27**(2) (1980), 228–234.
[15] M. Reiter and S. Stubblebine, PathServer. Presented at DIMACS Workshop on Trust Management in Networks, South Plainfield, New Jersey, Sept 30 - Oct 2, 1996.

Information Security Group, Dept. of Mathematics, Royal Holloway, University of London, Egham, Surrey TW20 OEX, U.K.
E-mail address: m.burmester@rhbnc.ac.uk

Center for Cryptography, Computer and Network Security, Dept. EE and CS, University of Wisconsin – Milwaukee, P.O. Box 784, WI 53201 Milwaukee
E-mail address: desmedt@cs.uwm.edu

Institute for Problems of Information Transmission, Russian Academy of Sciences, Ermolovoy 19, Moscow, GSP-4, Russia
E-mail address: kaba@ippi.ac.msk.su