Available trust and security in wireless networks

S. Daskapan
Researcher
Delft university of technology

J. J. Van Bergen
Researcher
Delft university of technology

M. Kaart
Researcher
Delft university of technology

Abstract

In wireless networks with unreliable connections the probability that a trust or security issuing system, and his redundant back-up systems, is cut off from the rest of the network is much higher than in common wired networks. Without this system then, critical transactions cannot take place. In this paper we introduce and test a protocol set that increases availability of trusted (security) systems in wireless networks. The protocol is among others based on secret sharing, blind signatures, group signatures and autonomous trust valuation.

1. Introduction

In any computer network where critical transactions take place, trust between entities is a prerequisite to security. An entity that issues trusted (security) services is referred to as a security/trust distribution centre (SDC).

1.1. The problem and objective

In wireless networks the availability of an SDC is of high concern. Such a *SDC might be cut-off temporary from the rest of the network due to lost connections*. Here, availability is defined by the degree to which a system is operable in a timely manner and in a committable state, when the mission is called for at an unknown random time. In figure 1a we have depicted such a wireless network containing one SDC, node 1, which contains physically a few mirroring redundants. The many other pervasive computing entities (CE's), like 2,3 and 4, rely on this SDC for their security and trust services. In figure 1b we see that due to network problems (of connection A, B and C) the SDC is not reachable anymore by the other CE's. Although CE3 still can reach the SDC, it cannot commit transactions with the others.

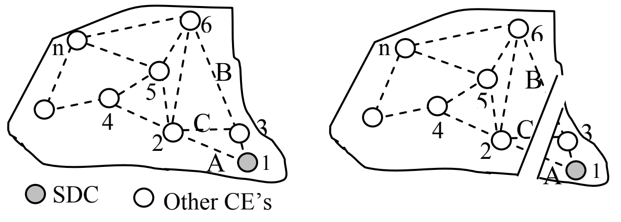

Figure 1 Advanced Preferences Design

As such, *the objective of this research is to provide a protocol by which availability of trust and security issuing systems in wireless networks can be increased.*

1.2. Our approach and other works

The current works on available SDCs can be divided in centralized and distributed approaches. On the one hand, there is the conventional way of centralizing the service, like Kerberos [1] and [2]. Their disadvantage is that they expose a/multiple single point/s of failure as was explained in previous paragraph. Even in edge networks, with many mirroring servers, persistent consecutive (or denial of service) attacks on all the follow up servers will ultimately succeed. On the other hand, decentralized and distributed approaches, like PGP [3] and Poblano [4], provide a far more dependable security/trust service. They can resist local disconnections better, since their service is the consensus of many peer SDC's. If, for example in figure 1, node 6 requests for a security service and node 1 is cut off, then trust still can be given to node 6 by consensus of the remaining nodes 2,4 and 5. However, because of the consensus protocols those approaches are complex and perform slowly in large networks.

In this paper a protocol set Pegasus, derived from Medusa [5], is proposed that reaches a compromise between the previous distributed and centralized approaches to increase availability of security and trust services in wireless networks. This survivability protocol applies the concept of volatile trust management [6], which prescribes trust to be an independent and exchangeable commodity between CE's. Survivability is defined as the capability of a system to provide its services in the presence of attacks or failures, and recover its services in a timely manner [7].

Pegasus is different from other works like MAFTIA [8], Rampart [9] and [10]. First, unlike them Pegasus does not rely on existing dedicated and thus limited number of servers, but rather on any trusted CE in the network. As such Pegasus is very scalable and its service highly survivable. Second, it optimizes the trade off between high/low performance and low/high dependability of a centralized/decentralized configuration. Third, the application of secret sharing, blind and group signatures makes consecutive attacks on each following up server on the participating CE's useless.

2. Overview of Pegasus

Pegasus consists of four protocols: bootstrapping, preparation, death and resurrection.

The *bootstrapping protocol* is only executed if there are no trusted SDC's to which the CE's in the network can subscribe to. Since no CE can receive then security services no secure transactions will be committed. The bootstrapping protocol is then used to create from this, possibly chaotic, space of untrusted CE's an ordered and semi-trusted space with different types of CE's: chosen SDC's (leaders), pool members and clients. This initial election is mainly based on trust. Trust assessment happens by trust metrics [11]. Pegasus considers all CE's that have overcapacity and 'good' credentials in the network as candidate SDC's. A good credential is determined first by a long history of availability and reliable transactions and second by a reliable trust certificate issued by a (former) recognized trust authority. After that some SDC's have been elected, the SDC will invite other trusted CE's and SDC's to collaborate in a pool of mutual mirroring back-ups. In case one SDC becomes unreachable the others will take over. It is possible and likely that pools overlap each other so that the leader of one pool could be a pool member in another pool. The remaining CE's, which need security/trust services, subscribe by sending their unique secret to one of the trusted SDC's that they trust most and has a large pool; they will become clients. Furthermore, the SDC exchanges symmetric encryption keys with the clients and his pool members. This protocol has been presented in [12] and will not be explained further in detail here.

The *preparation protocol* defines a set of actions that frequently take place in a *Medused* network. Here, the SDC divides the trusted content (trust token) into partial tokens and shares those parts with his pool members. This is done in such a way that no one is able to extract information out of his piece but in the event of failure of the SDC, if they work together, they can reconstruct the token and take over the role of the now unreachable SDC. The integrity and confidentiality of the partial tokens is cryptographically secured. Failure detection of the current SDC in a pool is also done here.

The *resurrection protocol* is activated when a SDC becomes unstable or unreachable. The first action that has to take place is to reach consensus about the SDC's death by majority voting between the pool members. After that they will poll the first successor from the successor list to see if he is in good health and, if so, they will send him their partial trust token and master decryption key. If he receives the partial tokens and decryption keys from more than half of the total of pool members, the successor will be able to reconstruct the trust token. With that he will be able to serve the clients of the unreachable SDC.

3. Preparation protocol

3.1. Consolidation of trust and security

Although Pegasus is a universal secure availability protocol it is this sub protocol that makes it specifically applicable for SDC's. In this protocol the aim is to capture the current state of the security/trust data of the SDC. Three types of security/trust data are distinguished.

*Trust assertions from clients to SDC***.** In order to make fuzzy properties like trustworthiness and authority more concrete and transferable to another pool member, a trust token is defined. The basic idea of the token is the detachment of trust and authority from the SDC's identity, so that trust and the derived authority to distribute security services become a commodity that can be owned by any other assigned member. The current SDC constructs therefore this so-called trust token T, which contains a concatenation of the pairs of client identities id_m and their hashed secrets h(s): T = $Array(h(s), id)_m$. The secrets contain the specific data of the considered security/trust service: passwords for authentication service, keys for key distribution servers, etc. As long as the SDC is vital, this token is stored on his (local or remote) trusted base. The ownership of T provides any owner with a dominant position to prove undeniably his trustworthiness and capacity when interacting with the clients. The token with their secrets implicates a mandate to serve them.

*Trust assertions by leader about clients***.** Not only the by the clients granted trust authority is preserved, but also visa-versa. The asserted trust by the leader about the clients is preserved in a client list. The precautionary measure that is taken to prevent the loss of this particular information about the clients is the maintenance of a history file about the clients [13]. The history file usually contains the ID's and the 'badness' score of the misbehaving members. Some techniques are available to attribute a badness value, like [14]. The leader can differentiate from this history file three or more type of clients by adding flag records: for the good, the bad and the dubious/unknown members. Each leader sends frequently one or more of those client lists (CLs) to other leaders to preserve his experience with the clients.

*Trust assertions by leader about pool members***.** Another important issue is the assignment of a successor by the deteriorating leader so that the leader's 'knowledge' can be passed on. The SDC accomplishes this by maintaining and distributing those recommendations (trust values [11]) in a successors list (SL). Without the SL the pool members cannot have majority consensus about the successor. The execution of Pegasus could get (temporarily) in a dead lock.

3.2. Allocation

The preparation includes besides the creation also the allocation of the survivability objects to pool members.

Assignment of successor(s). Since in wireless networks the number and place of the constituent pervasive devices changes quickly, pools are also changing. Subsequently, the trustworthiness rank of the pool members is also changing frequently. To have reliable succession it is essential that a SDC creates and distributes the SL frequently. Not a single successor is pointed out in advance, but a list of candidate successors. The SL usually contains the ID of the pool members, possibly added with a key escrow stealth entity. After that the current leader collapses all the members will check this list and select the first ranked member (with the highest score) or the next member if the first selected appears also to be corrupted or malfunctioning and so on. Since all members receive the same SL, the trusted and connected majority will unanimously recognize the successors. Even if the lists of some pool members are not fresh due to disconnections a majority will determine the next successor. Furthermore, each leader sends besides the SL also the CLs to other pool members to ensure preservation of his trust assertions about the clients.

Distribution and planning. When the leader at a sudden moment collapses, not only discontinuity of the security service, but also the established trust relations he has with many clients would be lost. This means that even after that the leader has been recovered technically, trust has to be established again with the clients. After all, his authority is doubtful after the collapse and many clients have unsubscribed. Since trust negotiations are time consuming this will damage service continuity of the clients even more. Therefore the idea is not to prevent, but to mask his failure. A SDC leader pawns his encrypted trust objects to other members in the pool. When he is malfunctioning the other pool members cooperate to restore al the trusted relations and to continue his security service. According to *twisted secret sharing* [15] and Shamir [16] the token can be split in v pieces, so that each (eventually also shuffled) piece contains some of the secret data. Each piece is encrypted with a key of one of the poolmembers. Each piece is once or multiply distributed among the other pool members, such that that each receiving member cannot open it with his keys.

4. Resurrection protocol

4.1. The moment between live and death

The state of a collapse is usually hard to detect. First boundaries have to be given of this state; when is an entity considered as unreliable or malfunctioning? Where is the frontier between death and alive? Many fault detection techniques are known to fill this gap, like [17,18]. In Pegasus the detection is the responsibility of both sides. The leader should notify the clients when he 'feels sick' and the pool members should check his condition (i.e. heart beats) regularly.

The leader sends a default time stamped heart beat messages according to a quasi-random frequency fr to the pool leaders and other successors. In case of trouble this (ALIVE, fr) message is not send (correctly) and, if not too late, subsequently he sends an explicit (SOS, ts, fr) message. When the pool members sense this unavailability due to lacking heartbeats or SOS, they request an explicit ALIVE message. If the leader still does not respond correctly in time or not at all (if overloaded after a DOS for example) or again with an SOS, he is discarded further as a leader/pool member (until new elections). Conflict resolution about his death is achieved by majority consensus (i.e. >2/3 of pool). The threshold values for this declaration of death should exceed average disorders, like the mean network traffic delays due to congestions.

4.2. Resurrection

The resurrection process can be divided in three sub protocols: the settlement of a successor, the reconstruction of the token and the re-keying of master keys.

Selecting the successor. The successor list has been distributed among the pool members previously. Now this list becomes the main source for the group to determine (by majority consensus) the successor unless the leader has pointed out already a stealth successor (first ranked in the list is then blank). Before granting him the token pieces and the keys his condition is checked like in the previous protocol is done with the leader.

Reconstruction of token. The token has been scattered over the pool members in the previous sub protocol and now the several scatters have to be gathered by a majority of pool members to reconstruct the original token.
The appointed successor, who receives the different parts from this majority, parses subsequently the token T in id_m and $h(s_m)$ and restores the secrets in a secure and trusted local environment. With those secrets this successor can now claim security/trust authority and be recognized as

the new SDC of the abandoned group of clients. After all the clients only recognize that authority that can identify itself with their secrets. The successor has also received previously the client lists and eventually other logs as a final breath from former leader. The successor tries to trace the malicious clients and discards them from the group, so that future attacks from them are prevented.

Refreshing master keys and secrets. After this empowerment the critical anarchistic moment is passed and the successor starts re-keying all the remaining master keys to guarantee backward group secrecy. He distributes the new master keys, authenticated with the old hash, to them. The pool master keys are also replaced.

After receiving the package, the members decrypt and verify the hashed secret. Only when the verification is positive the successor's authority is recognized. The members ratify their subordination by sending a new hashed secret, authenticated with the old hash. In this way they also provide backward secrecy in case the former corrupted leader awakes.

5. Testing by Simulation

5.1. Test set up

The Pegasus concept will be tested by simulation. Simulation has some important advantages above other test methods [19]. It allows the study of a system in compressed time, to have control over experimental conditions, to analyze the dynamic behavior of systems and to visualize the model.

The simulation environment that is used is NS-2. NS-2 is a discrete event driven and object oriented network simulator, written in C++ and TCL, developed at UC Berkeley. The assumed wireless network is not uniform, i.e. the shape, size, capacity of routing queues and bandwidth varies greatly from one segment to another. A simple wireless network topology is used: ring for backbone and star for each network segments. This has some redundancy to prevent catastrophic failures when a link between two nodes goes down. A few nodes are appointed as SDC's and the rest is either a client and/or pool member.

The aim of the test is to show that the security service of a 'Medused' SDC will survive as long as there are successors reachable. The security service of the SDC is here a simple authentication service. The amount of successors is related to the amount of participating SDC's. To create the maximum number of pools, the pool size (the number of pool members serving as possible successors) is set to a minimum: three. As such now many interconnected small pools are created.

The test (or independent) variables are the network size and number of intact connections. As such to test Pegasus, for each test randomly a wireless network with a different size is created. Then gradually the available bandwidth of some parts of this network is decreased until zero, i.e. disconnected the individual SDC one by one. The performance indicator (or dependent variable) is availability time, i.e. time that the security service is available for its clients.

5.2. Test results

The following graph displays the output from the simulation model when it was run with 6 SDC's participating in Pegasus.

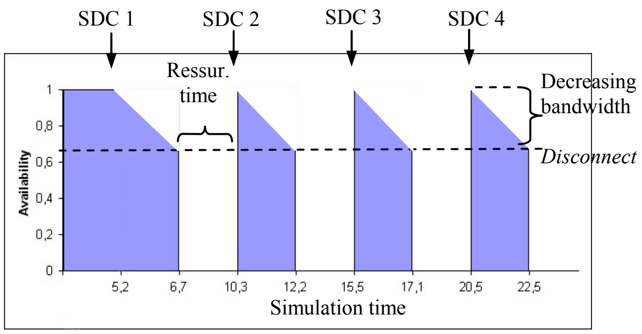

Figure 2. Test Results with six 'Medused' SDCs

This figure shows the availability of the security service in (simulation) time. The availability is indicated by two thresholds. Availability equals one if the SDC service is instantly available, i.e. it responds within a given minimum acceptable time slot, and zero if it does not respond within a given maximum acceptable time slot. The time it takes for a successor to resurrect the unreachable leader's service depends highly on the chosen input parameters for fault detection and pool size (time for consensus). After 5.2 seconds, the bandwidth starts decreasing. At 6.7 seconds in the simulation time, the original leader is unreachable. After some time a resurrection takes place and at 10.3 seconds the successor takes over the role of the unreachable leader. The bandwidth problem is persistent so that now the new follow-up leader is suffering from decreasing bandwidth and so on. Because this scenario was executed with 6 SDC's participating in pools, Pegasus was able to resurrect the security service three times. After cutting out four out of six SDC's, the remaining two SDC's wouldn't be able to get a majority vote. The security service is then permanently unavailable after the disconnection of the fourth SDC at t = 22,5 seconds.

The simulation was performed with increasing number of participating SDC's. The outcome of the tests with more SDC's showed the same trend, namely that Pegasus is capable of resurrecting the services of the leader as long as there are enough SDC's to form pools of a minimum size of 3. For example, with 15 participants, Pegasus was able to resurrect the security management process 13 times and thereby extending the availability to approximately 67,8 seconds in simulation time.

6. Conclusion and future work

In this paper we described the Pegasus survivability protocol. In wireless networks, security and trust systems that implement the Pegasus concept can achieve high availability. Pegasus consists of four main sub protocols to provide survivability: bootstrapping, preparation, death and resurrection. The simulation tests showed that Pegasus could indeed increase availability given a sufficient number of reachable participants. Pegasus works as long as there is a trusted well functioning qualified majority in the network. No other related work delivers this feature and does not provide this kind of survivability without making substantial investments in fixed redundants. In future work we intend to develop a light version of Pegasus and to test it on very small sensor networks.

7. References

[1] J.G. Steiner and B.C. Neuman and J.I. Schiller, "Kerberos: An Authentication Service for Open Network Systems", *Usenix Conference Proceedings*, Dallas, 1988, pp. 191-202.

[2] L. Gong, "Increasing Availability and Security of an Authentication Service", *IEEE Journal on Selected Areas in Communications*, Vol. 11(5), 1993, pp. 657-662.

[3] P. Zimmermann, *PGP User's Guide*, MIT Press, USA,1994.

[4] R Chen, "Poblano-A Distributed Trust Model for Peer-to-Peer Networks", *Sun white paper*, 2002.

[5] Semir Daskapan, Willem G. Vree and Rene W. Wagenaar, Emergent information security in critical infrastructures, to appear in *Int. Journal of Critical Infrastructures*, Inderscience, Vol.x(x), 2006, ISSN: 1475-3219.

[6] S. Daskapan, A. Verbraeckand and W. G Vree, "The merge of computing paradigms", *5th International Conf. on computer and information technology,* Dhaka, 2002, pp. 553-558.

[7] R. Ellison, D. Fisher, R. Linger, H. Lipson, T. Longstaff and N. Mead, "Survivable network systems: An emerging discipline", *Technical Report CMU/SEI-97-153,* Carnegie Mellon University, Pittsburgh, 1997.

[8] C. Cachin, and J. Poritz, "Secure Intrusion Tolerant Replication on the Internet", In *Proc. of the International Conference on Dependable Systems and Networks*, Washington, 2002, pp.167-176.

[9] M. Reiter, "Secure Agreement Protocols: Reliable and Atomic Group Multicast in Rampart", In *Proc. of 2nd ACM Conference on Computer and Communications Security*, Fairfax, 1994, pp.68–80.

[10] Srikant Sharma, Jiawu Chen, Wei Li, Kartik Gopalan, Tzi-cker Chiueh, Duplex, "A Reusable Fault Tolerance Extension Framework for Network Access Devices", *International Conference on Dependable Systems and Networks (DSN'03)*, San Francisco, 2003, pp. 501.

[11] S Daskapan., W. G. Vree and A Ali Eldin, "Trust metrics for survivable security systems", In *Proc. of the IEEE International Conference on Systems, Man & Cybernetics*, Washington, 2003.

[12] Wiebe Wiechers, Semir Daskapan, Willem G. Vree, "Simulating the Establishment of Trust Infrastructures in Multi-Agent Systems", *Sixth International Conference on Electronic Commerce*, Delft, The Netherlands, 2004.

[13] P. Maniatis and M. Baker, "Secure History Preservation Through Timeline Entanglement", In *Proc. of the 11th USENIX Security Symposium*, San Francisco, 2002.

[14] P. J. Mosterman and G. Biswas, "Diagnosis of continuous valued systems in transient operating regions", *IEEE Trans. Syst., Man, Cybern. A*, vol. 29, 1999, pp.554–565.

[15] S. Daskapan, "Dependable security by twisted secret sharing", *19th IFIP Information Security Conference*, Toulouse, 2004.

[16] A. Shamir, "How to Share a Secret." Communications of the ACM, Vol. 22(11), 1979, pp. 612-613.

[17] M. Larrea, *Efficient Algorithms to Implement Failure Detectors and Solve Consensus in Distributed Systems*. PhD thesis, University of the Basque Country, San Sebastian, 2000.

[18] E. J. Manders, L. A. Barford and G. Biswas, "An Approach for Fault Detection and Isolation, Dynamic Systems From Distributed Measurements", *IEEE transactions on instrumentation and measurement*, Vol.51(2), 2002,pp.235-240.

[19] A. M. Law, and W. D. Kelton, *Simulation Modeling & Analysis*, *McGraw-Hill Inc.*, 1991.