M. Kolar, C. Fernandez-Gago, and J. Lopez, "A Model Specification Implementation for Trust Negotiation", *The 14th International Conference on Network and System Security (NSS 2020)* vol. 12570, pp. 327-341, 2020.
NICS Lab. Publications: https://www.nics.uma.es/publications

A Model Specification Implementation for Trust Negotiation

Martin Kolar[1], Carmen Fernandez Gago[2], Javier Lopez[1]
{kolar,mcgago,jlm}@lcc.uma.es

[1] Department of Computer Science
[2] Department of Applied Mathematics
University of Malaga, 29071 Malaga, Spain

Abstract. Trust negotiation represents a suitable approach for building trust in online environments, where the interacting entities are anonymous. It covers important criteria on security and privacy. In this work, we propose a method for implementing our model specification that handles trust negotiation. We define the structure of the trust negotiation module that is a standalone unit capable of negotiating on its own. It may be included to any software by its defined interfaces. We realise our method with a ride-sharing scenario and four trust negotiation strategies that we apply in order to validate our design and implementation. We propose a solution that is fully customisable based on different requirements. The proposal provides guidelines for developers in the process of including trust negotiation into their software.

1 Introduction

The present world offers many opportunities and challenges. People need to exchange resources, such as information, services and products. They require trust for cooperation, which may be handled by trust negotiation *(TN)*. This is a suitable approach for online environments, where unknown entities interact together. Trust is built by a mutual exchange of credentials. In this work, we propose a method for implementing TN. Since the Software Development Life Cycle *(SDLC)* does not include trust by default, we provide it for all of its phases. We analyse our TN model specification [15] from the implementation-specific point of view and integrate it into a software product. We follow the Object-oriented programming *(OOP)* and propose a trust negotiation module *(TNM)* that is an autonomous unit capable of TN on its own. It connects to other modules by defined interfaces and its logical separation makes the implementation more secure. Developers will be guided through the whole process, which may spare their time and costs when developing their own TN model.

The rest of this paper is organised as follows. Section 2 deals with the previous work on TN and the SDLC. Section 3 presents the proposal

of the TNM structure, whereas its implementation is done in Section 4. Section 5 defines a TN ride-sharing scenario that is applied in Section 6 and that validates the TNM. Finally, Section 7 concludes this paper and outlines the future work.

2 Related Work

Trust is important for Computer Science. Gambetta [1] defines it as a probability, by which an entity expects another one to perform an action, on which its welfare depends. Jøsang [2] defines two categories: the reliability trust is based on the reliability of an entity, whereas the decision trust is based on a decision to be dependent. Winsborough [3] recognises two approaches for authentication: identity-based and capability-based. Another approach is suitable for online environments: trust is built by TN, i. e., a sequential exchange of credentials [4]. TN requires a set of criteria [5], a protocol and a strategy [6]. The dynamics of trust is managed by trust models. Moyano [7] classifies them into the decision and evaluation ones. The former include TN models and TrustBuilder [8] was the first one. Some TN models implement security agents that build trust on behalf of entities [9,3]. Hess et al. [10] propose a model that provides a run-time privacy protection. This approach is suitable for dynamically-generated contents. Guo and Jiang [11] propose a TN framework with an adaptive negotiation strategy that ensures a balance between building trust and privacy protection. Seamons et al. [12] examine privacy issues in online TN and propose methods for their elimination or minimisation. Our goal is to create a general TN framework for developers. Development issues were analysed in approaches, such as the SDLC that divides the development into phases. Ruparelia [13] summarises SDLC models, such as waterfall, spiral and incremental. Driver et al. [14] include digital trust into the SDLC and integrate third-party components. Kolar et al. [15] present a model specification that guides developers in the process of including TN into software. We follow up this work for the implementation phase of the SDLC. Bresciani et al. [16] present a development of agent-oriented software systems. Casey and Richardson [17] deal with the effective globally distributed software development. Ilieva et al. [18] deal with agile methodologies that are suitable for variable requirements. We will follow the implementation phase of the SDLC and extend it by the TN capabilities. To the best of our knowledge, there is no other work following this approach. We will follow the OOP in order to make a scalable implementation. Our framework is unique since it guides developers in the process of including TN into their software.

3 Specification of the Trust Negotiation Module

In this section, we present an implementation of the general TN functionality. Our approach is to develop the TNM that covers all aspects of TN [15]. We provide a general methodology of its design and internal composition. The TNM is a self-contained unit capable of performing TN on behalf of entities. It is connected to the other modules that participate in the initial set up of TN. We will analyse its structure and functionality:

- The TNM is generally designed in order to support any topology of software modules. They are connected by four interfaces.
- A database is required for accessing user data, such as credentials and policies. This data is accessed during trust negotiation on the fly.
- Two modes of operation are possible: either a complete control over TN, for which each step is supervised, or an automated process, when the TNM is initially configured and then performs TN on its own.
- A trusted authority is supported in order to validate credentials.

The TNM is depicted in Figure 1. Its interfaces are as follows:

1) This interface connects to another TNM, with whom will be carried out TN. Both must use the same interface and negotiation protocol.
2) The control interface connects to the control module that manages the TN operations. It transfers all requests and returns responses.
3) This interface connects to a database. The compliance checker uses it for requesting credentials and policies based on the actual needs.
4) The last one optionally connects to a trusted authority. The compliance checker may need to verify and validate the incoming credentials.

The following sub-modules ensure the functionality of the TNM:

- ***Negotiator*** represents the entity that wants to build trust.
- ***Compliance Checker*** performs disclosure decisions for the negotiator and enforces preserving its privacy.
- ***Trust*** represents the currently established trust used by the negotiator.
- ***Exposure*** represents the current privacy exposure level.
- ***Strategy*** and ***Protocol*** represent the chosen negotiation strategy and the communication protocol used by the negotiators, respectively.

We proposed the TNM and its main components in compliance with our defined requirements for TN [5]. The next section introduces a deeper analysis and aims to the implementation in a programming language.

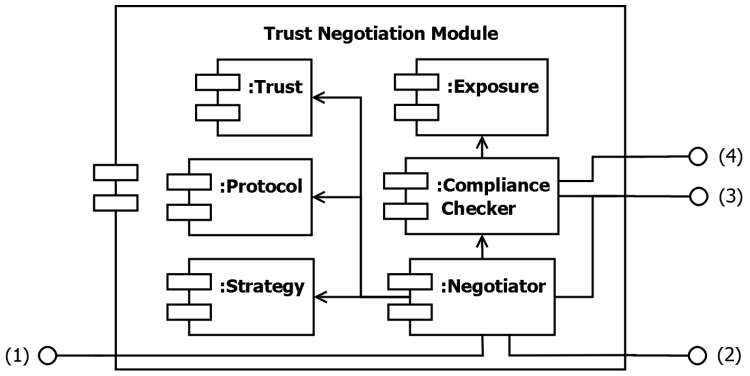

Fig. 1: The Inner Structure of the Trust Negotiation Module

4 Implementation of the Trust Negotiation Module

In this section, we decompose the TNM into the particular classes. We define attributes and methods in order to specify their features and behaviour. Figure 2 depicts the class diagram that is the core of our implementation. It depicts the design and topology of all classes.

4.1 The Core of Trust Negotiation

The *Negotiator* class is principal since it represents its owning entity that wants to build trust with another one. The negotiator carries out TN on behalf of the entity. When it is demanded for disclosing a credential, the compliance checker validates the request. Two negotiators have to use the same protocol (*protocol* attribute) that defines the exact content and order of the exchanged messages. The following methods are implemented for handling the incoming and outgoing credentials:

- ***Provide*** discloses the specified credential to the other negotiator if the compliance checker approves it.
- ***Demand*** requests the specified credential from the other negotiator.
- ***Is_negotiating*** returns the willingness of the negotiator to continue with TN. For example, *false* is returned in case of privacy violation.

The *TrustRelationship* class represents the trust relationship between two negotiators (*negotiator1*, *negotiator2* attribute). Their common goal (*goal* attribute) defines the purpose for building trust. The *trust* attribute specifies the actual trust level that has been built and is measured on a 7-degree scale. During TN, this level increases and once it is equal or higher

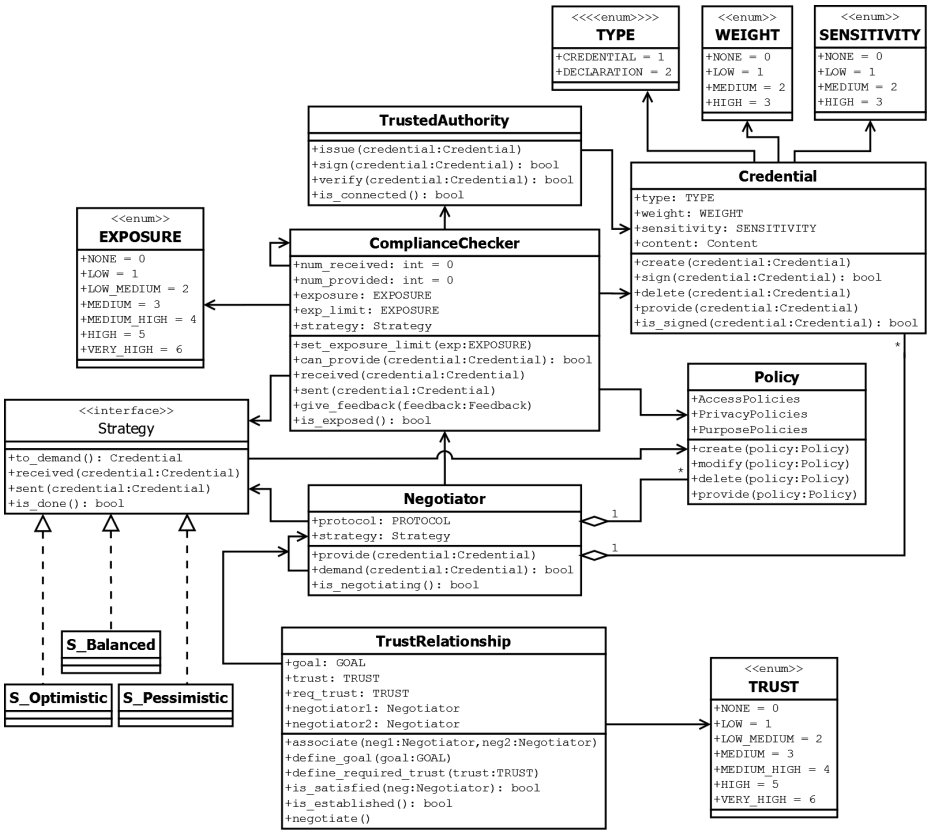

Fig. 2: The Class Diagram of the Trust Negotiation Module

than the required one (*req_trust* attribute), the negotiator is satisfied. When this happens for both negotiators, TN is successfully terminated. The TrustRelationship class implements the following methods:

- ***Associate*** links two negotiators into a relationship.
- ***Define_goal*** defines the common goal for the intended relationship.
- ***Define_required_trust*** defines the required level of trust in order to establish a successful trust relationship.
- ***Is_satisfied*** returns the satisfaction state of the negotiator that is determined by reaching its required level of trust.
- ***Is_established*** checks whether the relationship was established. It is determined by reaching the required level of trust by both negotiators.
- ***Negotiate*** starts TN and the process of exchanging credentials.

4.2 Processing Trust Negotiation

Each negotiator must specify its own negotiation strategy. The interface *Strategy* defines the methods used by the classes implementing a specific strategy. We propose the *S_Optimistic*, *S_Pessimistic* and *S_Balanced* classes that represent an optimistic, pessimistic and balanced strategy, respectively. An instantiated class represents the active strategy. Each class determines the credentials to be disclosed, by which order and also reacts to the incoming ones. The following methods are defined:

- ***To_demand*** determines the credential to be requested, if any.
- ***Received*** is called by the negotiator and informs the strategy class about the incoming credentials. It is used for planning next disclosures.
- ***Sent*** is a similar method that informs about the disclosed credentials.
- ***Is_done*** returns the information, whether all the required credentials have been obtained or there are more to be requested.

The *ComplianceChecker* class monitors all credentials flow. The current privacy exposure level (*exposure* attribute) is calculated based on the sensitivity of the disclosed credentials. Then, it is matched against the privacy policies and the exposure limit (*exp_limit* attribute) for privacy protection. The limit prevents further disclosures when reached. Just like trust, the exposure is measured on a 7-degree scale. The *num_received* and *num_provided* attributes count the obtained and disclosed credentials, respectively. A feedback that confirms the use of the obtained credentials may be provided to the other negotiator. The compliance checker implements the following methods:

- ***Set_exposure_limit*** defines the maximum privacy exposure that is tolerated during TN.
- ***Can_provide*** informs whether a credential can be disclosed. If it is locked or the current exposure is too high, the disclosure will be denied.
- ***Received*** informs the compliance checker that a credential has been received. This method is similar to the one of the interface Strategy.
- ***Sent*** informs that a credential has been disclosed. These two methods are important for calculating the current privacy exposure level.
- ***Give_feedback*** provides a feedback of how the obtained credentials were used in TN. It may help the negotiators to revise their policies.
- ***Is_exposed*** informs whether the exposure limit has been reached.

4.3 External Dependencies

The *Policy* class provides the interface for managing policies in the database. It implements the following methods:

- ***Create*** defines a new policy and then adds it to the database.
- ***Delete*** erases the specified policy from the database.
- ***Modify*** updates a policy, e. g., based on the received feedback.
- ***Provide*** reads a policy and delivers it to the compliance checker.

The *Credential* class provides the interface for accessing credentials in the database. The type of a credential (*type* attribute) is either *CREDENTIAL* or *DECLARATION*. The former is signed, whereas the latter is not. The *weight* and *sensitivity* attributes specify its importance for building trust and its confidentiality level, respectively. They are measured on a 4-degree scale. The *content* attribute is the actual information of the credential. The following methods are implemented:

- ***Create*** adds a new unsigned credential of type DECLARATION.
- ***Sign*** serves for signing a credential, but requires a trusted authority. Its type is changed from DECLARATION to CREDENTIAL.
- ***Delete*** erases a credential. It is used for the no longer valid ones.
- ***Provide*** ensures the access to the credentials during TN.
- ***Is_signed*** informs whether a credential is signed.

The *TrustedAuthority* class represents the optional interface for connecting to a trusted authority. Its purpose is to issue and verify credentials, so that they may be more valuable for TN. These methods are implemented:

- ***Issue*** issues a new credential of type CREDENTIAL by default.
- ***Sign*** inputs an existing credential of type DECLARATION and signs it. In case of success, its type will change to CREDENTIAL.
- ***Verify*** inputs a signed credential by this authority and verifies its authenticity. This is intended for unknown or untrusted credentials.
- ***Is_connected*** informs whether the authority is connected.

5 Trust Negotiation Ride-sharing Scenario

In this section, we propose a TN ride-sharing scenario implemented by using the TNM. Later, we will use it for validating the TNM. This demonstrative scenario is easy to follow: the web portal is a popular service that enables its users to share a ride. The drivers offer rides to the passengers that pay for it. Two of them want to travel to the same destination, which is their common goal. They have to establish a trust relationship and especially the passenger has to be confident that the driver is experienced and responsible. Their TN is as follows:

1. The passenger sends a message to the driver by the web portal. He requests a contact information.

2. The driver replies by providing his e-mail address and telephone number. Then, he asks for a contact information in return.
3. The passenger only discloses his telephone number as other information is too sensitive. He requests the driving experience and price.
4. The driver discloses the history of his rides and his price list.
5. Now, the passenger is more confident in the driving experience of the driver. He also demands a detailed information about the latest ride.
6. The driver discloses the exact route information and his usual speed. Then, he asks the passenger for the luggage he wants to carry.
7. The passenger is satisfied and acquires a sufficient confidence. He responds that the luggage is large and asks whether it is a problem.
8. The driver responds that he has a space. The passenger agrees with the offered price and decides to book the ride.

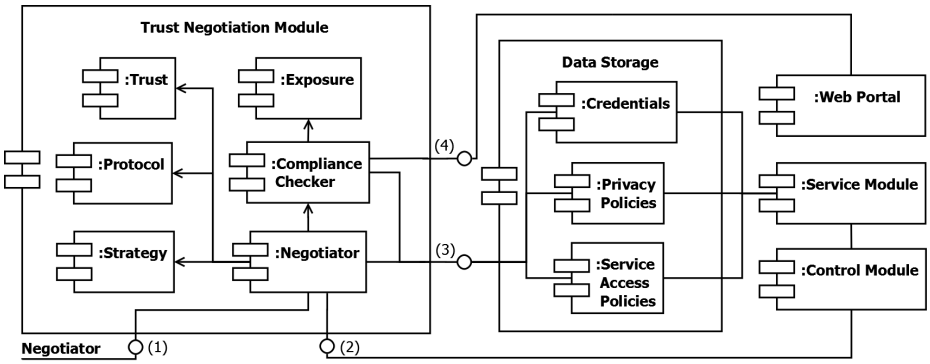

Fig. 3: The Ride-sharing Scenario Implementation

We implement this scenario by following our proposal and using the OOP paradigm. We specify modules that are depicted in Figure 3. The TNM is the core unit that is connected to the following entities:

1. The other TNM. Two such modules are connected by a communication channel. They act on behalf of the driver and the passenger.
2. The control module managing TN and processing its results.
3. The data storage unit containing credentials and policies.
4. The web portal providing the online service for ride-sharing.

Both users have their own instances of the TNM, the control module, the service one and the data storage unit. They perform the following actions:

- ***The service module*** manages data for its user, such as updating the credentials and policies in the data storage unit. The module has full access. Then, it configures system variables, e. g., encryption keys.
- ***The control module*** manages the TNM. It initiates TN, supervises the exchange process and obtains its result. The module is acknowledged from the service one when all data is configured for TN.
- ***The trust negotiation module*** carries out TN. Both users request the TNM to build trust for them. Their credentials are exchanged based on the defined policies and the chosen negotiation strategy. Finally, TN is evaluated and the users are informed about its result.

This ride-sharing scenario has shown the method for implementing TN using the proposed specification. It has explained the purpose of the specific modules and their cooperation in the process of building trust.

6 Validation of the Trust Negotiation Module

In this section, we validate our proposal. We create four variations based on the scenario while each implements a different negotiation strategy. We compare their attributes for building trust. We assemble three equal devices with different roles: two negotiators and a trusted authority. They are made of the following components:

- ***Arduino Nano*** is a single-board electronics platform using the 8-bit ATmega328 microcontroller with 32 KB flash memory and 2 KB RAM. It performs the deployed TN scenario and drives the other components.
- ***ESP8266*** is a Wi-Fi microcontroller with the full TCP/IP stack ability. We use it for interconnecting the devices by a wireless network.
- ***ST7735*** is a small colour TFT display. It is not required for TN, however, we use it for outputting messages of its progress to the user.

We defined a general TN algorithm (*Negotiate* function) that is shown in Algorithm 1. TN is performed until both negotiators are willing to do so: the negotiator (1) checks his actual trust level and if it is lower than required, he demands a credential from the negotiator (2). Then, (2) checks whether he may disclose it. This depends on his privacy exposure level that, increased by the sensitivity of the credential, must be lower than his exposure limit. If it was disclosed, (1) increases his trust level by his defined weight of the credential and (2) increases his exposure level by the defined sensitivity. Otherwise, (1) demands another credential. The negotiators change their roles after each step so that trust is built evenly. TN continues until both establish the required trust and while

their privacy is preserved. TN fails if the policies are improperly specified, the available credentials are not sufficient or privacy would be violated.

Algorithm 1 Trust negotiation

```
1: function NEGOTIATE(negotiator1, negotiator2)
2:     while negotiator1 is negotiating and negotiator2 is negotiating do
3:         if negotiator1.current_trust < negotiator1.required_trust then
4:             negotiator1 demands credential from negotiator2
5:             if negotiator2 discloses credential then
6:                 increase negotiator1.current_trust about credential.weight
7:                 increase negotiator2.current_exposure about credential.sensitivity
8:         if negotiator2.current_trust < negotiator2.required_trust then
9:             negotiator2 demands credential from negotiator1
10:            if negotiator1 discloses credential then
11:                increase negotiator2.current_trust about credential.weight
12:                increase negotiator1.current_exposure about credential.sensitivity
13:        if negotiator1.current_exposure > negotiator1.exposure_limit or
               negotiator2.current_exposure > negotiator2.exposure_limit then
14:            return failure
15:        if negotiator1.current_trust >= negotiator1.required_trust and
               negotiator2.current_trust >= negotiator2.required_trust then
16:            return success
17:    return failure
```

The algorithm is applied in four trust negotiation strategies. Each strategy extends it by specific features. We have implemented the following ones:

- ***Optimistic strategy*** basically follows the algorithm. The negotiators define their required trust and exposure limit. This strategy is marked as optimistic since credentials may be freely exchanged. It is fast, efficient for building trust and not demanding for computing resources.
- ***Pessimistic strategy*** is more restrictive. The unbalance level and limit are implemented: the former is a difference between the current exposure and established trust. The latter defines its maximum value that changes as trust increases. Then, credentials use a locking mechanism. These features ensure a balance in exchanging credentials. The strategy preserves privacy well, however, is less efficient.
- ***Balanced strategy*** measures trust by the couple (t, c), for which t is the trust value and c is the confidence value [19]. This metrics provides a certainty to the established trust in TN. A trusted authority is used and the signed credentials are assigned high confidence levels. The strategy represents a good compromise in efficiency, so we mark it as balanced.

- ***Improved balanced strategy*** uses trust intervals *(TI)* [7], where the trust value lies on $< t_a, t_b >$. TI is a more intuitive alternative for (t, c) and the confidence value is hidden from the negotiator. Credentials are assigned the confidence levels as previously and the resulting trust value is computed as the arithmetic mean of TI.

All strategies successfully established trust for both negotiators. The TN process is depicted in appendices. Our approach provides a suitable procedure for implementing TN that a developer may utilise for his own TN scenario. It may significantly save his time and costs.

7 Conclusion

In this work, we have proposed a method for implementing our TN model specification. We have specified the TNM that is capable of automated TN. The TNM isolates the TN functionality, which increases security. Other software modules are connected by controlled interfaces. This versatile concept facilitates adding and removing modules based on requirements. We have provided a detailed TNM implementation by specifying its classes and their topology. Then, we have defined a ride-sharing scenario that is a practical use-case for TN. We have specified a general algorithm for TN that cares about privacy and that is used in four TN strategies. We have deployed the scenario for validating the TNM. We demonstrated how to implement a given TN scenario using the proposed TNM, connect it to the other entities and finally create a functional model.

In the future work, the proposed model specification will become a part of a TN framework and will be fully integrated into all phases of the SDLC. It will form a complete guide for the developers that want to include TN into their systems from the early phases of development to the successful end.

Acknowledgements
This research has been partially supported by the Spanish Ministry of Science and Innovation through the project SecurEdge (PID2019-110565RB-I00) and by the European Commission through the project EU H2020-SU-ICT-03-2018 Project No. 830929 CyberSec4Europe (cybersec4europe.eu).

References

1. D. Gambetta, Can We Trust Trust?, D. Gambetta (Ed.), Trust: Making and Breaking Cooperative Relations, B. Blackwell, Oxford, pp. 213-238, 1990.

2. A. Jøsang, R. Ismail and C. Boyd, A Survey of Trust and Reputation Systems for Online Service Provision, Decision Support Systems, vol. 43, pp. 618-644, 2007.
3. W. H. Winsborough, K. E. Seamons and V. E. Jones, Automated Trust Negotiation, DARPA Information Survivability Conference and Exposition, DISCEX '00 Proceedings, vol. 1, pp. 88-102, 2000.
4. W. H. Winsborough and N. Li, Towards Practical Automated Trust Negotiation, Proceedings of the 3rd International Workshop on Policies for Distributed Systems and Networks, pp. 92-103, 2002.
5. M. Kolar, C. Fernandez-Gago and J. Lopez, Policy Languages and their Suitability for Trust Negotiation, 32nd Annual IFIP WG 11.3 Conference on Data and Applications Security and Privacy XXXII, vol. 10980, Springer, Cham, 69-84, 2018.
6. T. Yu, M. Winslett and K. E. Seamons, Interoperable Strategies in Automated Trust Negotiation, Proceedings of the 8th ACM Conference on Computer and Communications Security, 2001.
7. F. Moyano, Trust Engineering Framework for Software Services, PhD thesis, Lenguajes y Ciencias de la Computación, Universidad de Málaga, 2015.
8. M. Winslett, T. Yu, K. E. Seamons, A. Hess, J. Jacobson, R. Jarvis, B. Smith and L. Yu, Negotiating Trust in the Web, IEEE Internet Computing, vol. 6, no. 6, pp. 30-37, 2002.
9. P. A. Bonatti, J. L. De Coi, D. Olmedilla and L. Sauro, A Rule-Based Trust Negotiation System, IEEE Transactions on Knowledge and Data Engineering, vol. 22, no. 11, 2010.
10. A. Hess, J. Holt, J. Jacobson and K. E. Seamons, Content-Triggered Trust Negotiation, ACM Transactions on Information and System Security, 7(3), pp. 428-456, 2004.
11. S. Guo and W. Jiang, An Adaptive Automated Trust Negotiation Model and Algorithm, International Conference on Communications and Intelligence Information Security, Nanning, 2010, pp. 130-134, 2010.
12. K. E. Seamons, M. Winslett, T. Yu, L. Yu and R. Jarvis, Protecting Privacy During On-line Trust Negotiation, In International Workshop on Privacy Enhancing Technologies, pp. 129-143, Springer, 2002.
13. N. B. Ruparelia, Software Development Lifecycle Models, ACM SIGSOFT Software Engineering Notes, 35(3), pp. 8-13, 2010.
14. M. Driver, F. Gaehtgens and M. O'Neill, Managing Digital Trust in the Software Development Life Cycle, ID G00326944, Gartner, 26 May 2017.
15. M. Kolar, C. Fernandez-Gago and J. Lopez, A Model Specification for the Design of Trust Negotiations, Computers & Security, vol. 84, Elsevier, pp. 288-300, 2019.
16. P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia and J. Mylopoulos, TROPOS: An Agent-Oriented Software Development Methodology, Autonomous Agents and Multi-Agent Systems, 8(3), pp. 203-236, 2004.
17. V. Casey and I. Richardson, Implementation of Global Software Development: A Structured Approach, Software Process: Improvement and Practice, 14(5), pp. 247-262., 2009.
18. S. Ilieva, P. Ivanov, E. Stefanova, Analyses of an Agile Methodology Implementation, In Proceedings of the 30th Euromicro Conference, pp. 326-333, IEEE, 2004.
19. G. Theodorakopoulos and J. S. Baras, Trust Evaluation in Ad-Hoc Networks, In Proceedings of the 3rd ACM Workshop on Wireless Security (WiSe '04), ACM, New York, NY, USA, pp. 1-10., 2004.

Appendix A Optimistic Strategy

	Driver		Trust Negotiation	**Passenger**	
	Required trust: 6 Exposure limit: 10			Required trust: 12 Exposure limit: 8	
Step	Trust	Exposure		Trust	Exposure
1.	0	0	⟵ request: phone number ⟵	0	0
2.	0	1	⟶ disclosure: phone number ⟶	3	0
3.	0	1	⟶ request: phone number ⟶	3	0
4.	2	1	⟵ disclosure: phone number ⟵	3	3
5.	2	1	⟵ request: price list ⟵	3	3
6.	2	2	⟶ disclosure: price list ⟶	5	3
7.	2	2	⟶ request: address ⟶	5	3
8.	5	2	⟵ disclosure: address ⟵	5	6
9.	5	2	⟵ request: driving history ⟵	5	6
10.	5	6	⟶ disclosure: driving history ⟶	8	6
11.	5	6	⟶ request: luggage info ⟶	8	6
12.	6	6	⟵ disclosure: luggage info ⟵	8	7
13.	6	6	⟵ request: space info ⟵	8	7
14.	6	7	⟶ disclosure: space info ⟶	10	7
15.	6	7	⟵ request: address ⟵	10	7
16.	6	7	⟶ **disclosure denied** ⟶	10	7
17.	6	7	⟵ request: latest ride ⟵	10	7
18.	6	8	⟶ disclosure: latest ride ⟶	12	7
19.	6	8	⟵ **successful termination** ⟶	12	7

Appendix B Pessimistic Strategy

	Driver				Trust Negotiation	**Passenger**			
	Required trust: 6 Exposure limit: 10					Required trust: 12 Exposure limit: 9			
Step	Trust	Exp.	Unb. Level	Unb. Limit		Trust	Exp.	Unb. Level	Unb. Limit
1.	0	0	0	2	⟵ request: phone number ⟵	0	0	0	0
2.	0	1	1	2	⟶ disclosure: phone number ⟶	2	0	-2	0
3.	0	1	1	2	⟶ request: phone number ⟶	2	0	-2	0
4.	0	1	1	2	⟵ **disclosure denied** ⟵	2	0	-2	0
5.	0	1	1	2	⟶ request: e-mail ⟶	2	0	-2	0
6.	1	1	0	2	⟵ disclosure: e-mail ⟵	2	2	0	0
7.	1	1	0	2	⟵ request: price list ⟵	2	2	0	0
8.	1	2	1	2	⟶ disclosure: price list ⟶	4	2	-2	2
9.	1	2	1	2	⟶ request: address ⟶	4	2	-2	2
10.	1	2	1	2	⟵ **disclosure denied** ⟵	4	2	-2	2
11.	1	2	1	2	⟶ request: phone number ⟶	4	2	-2	2
12.	3	2	-1	2	⟵ disclosure: phone number ⟵	4	5	1	2
13.	3	2	-1	2	⟵ request: driving history ⟵	4	5	1	2
14.	3	2	-1	2	⟶ **disclosure denied** ⟶	4	5	1	2
15.	3	2	-1	2	⟵ request: space info ⟵	4	5	1	2
16.	3	3	0	2	⟶ disclosure: space info ⟶	6	5	-1	2
17.	3	3	0	2	⟶ request: address ⟶	6	5	-1	2
18.	3	3	0	2	⟵ **disclosure denied** ⟵	6	5	-1	2
19.	3	3	0	2	⟶ request: luggage info ⟶	6	5	-1	2
20.	4	3	-1	4	⟵ disclosure: luggage info ⟵	6	6	0	2
21.	4	3	-1	4	⟵ request: latest ride ⟵	6	6	0	2
22.	4	4	0	4	⟶ disclosure: latest ride ⟶	8	6	-2	4
23.	4	4	0	4	⟶ request: address ⟶	8	6	-2	4
24.	7	4	-3	5	⟵ disclosure: address ⟵	8	9	1	4
25.	7	4	-3	5	⟵ request: driving history ⟵	8	9	1	4
26.	7	8	1	5	⟶ disclosure: driving history ⟶	11	9	-2	4
27.	7	8	1	5	⟵ request: address ⟵	11	9	-2	4
28.	7	10	3	5	⟶ disclosure: address ⟶	13	9	-4	4
29.	7	10	3	5	⟵ **successful termination** ⟶	13	9	-4	4

Appendix C Balanced Strategy

	Driver			Trust Negotiation	**Passenger**		
	Required trust: 6 Exposure limit: 10 Required conf.: 0.8				Required trust: 12 Exposure limit: 8 Required conf.: 0.6		
Step	Trust	Exp.	Conf.		Trust	Exp.	Conf.
1.	0	0	0	⟵ request: phone number ⟵	0	0	0
2.	0	1	0	⟶ disclosure: phone number ⟶	3	0	0.6
3.	0	1	0	⟶ request: phone number ⟶	3	0	0.6
4.	2	1	0.7	⟵ disclosure: phone number ⟵	3	3	0.6
5.	2	1	0.7	⟵ request: price list ⟵	3	3	0.6
6.	2	2	0.7	⟶ disclosure: price list ⟶	5	3	0.6
7.	2	2	0.7	⟶ request: address ⟶	5	3	0.6
8.	5	2	0.88	⟵ disclosure: address ⟵	5	6	0.6
9.	5	2	0.88	⟷ **authority: confirmed**	5	6	0.6
10.	5	2	0.88	⟵ request: driving history ⟵	5	6	0.6
11.	5	6	0.88	⟶ disclosure: driving history ⟶	8	6	0.75
12.	5	6	0.88	**authority: confirmed** ⟷	8	6	0.75
13.	5	6	0.88	⟶ request: luggage info ⟶	8	6	0.75
14.	6	6	0.85	⟵ disclosure: luggage info ⟵	8	7	0.75
15.	6	6	0.85	⟵ request: space info ⟵	8	7	0.75
16.	6	7	0.85	⟶ disclosure: space info ⟶	10	7	0.72
17.	6	7	0.85	⟵ request: address ⟵	10	7	0.72
18.	6	9	0.85	⟶ disclosure: address ⟶	12	7	0.77
19.	6	9	0.85	**authority: confirmed** ⟷	12	7	0.77
20.	6	9	0.85	⟵ **successful termination** ⟶	12	7	0.77

Appendix D Improved Balanced Strategy

	Driver				Trust Negotiation	**Passenger**			
	Required trust: 0.6 Exposure limit: 0.8					Required trust: 0.75 Exposure limit: 0.55			
St.	Trust	Exp.	Conf.	TI		Trust	Exp.	Conf.	TI
1.	0	0	0	<0, 1>	⟵ req: phone number ⟵	0	0	0	<0, 1>
2.	0	0.05	0	<0, 1>	⟶ dis: phone number ⟶	0.15	0	0.6	<0.09, 0.49>
3.	0	0.05	0	<0, 1>	⟶ req: phone number ⟶	0.15	0	0.6	<0.09, 0.49>
4.	0.20	0.05	0.7	<0.14, 0.44>	⟵ dis: phone number ⟵	0.15	0.15	0.6	<0.09, 0.49>
5.	0.20	0.05	0.7	<0.14, 0.44>	⟵ req: price list ⟵	0.15	0.15	0.6	<0.09, 0.49>
6.	0.20	0.15	0.7	<0.14, 0.44>	⟶ dis: price list ⟶	0.30	0.15	0.6	<0.18, 0.58>
7.	0.20	0.15	0.7	<0.14, 0.44>	⟶ req: address ⟶	0.30	0.15	0.6	<0.18, 0.58>
8.	0.50	0.15	0.88	<0.44, 0.56>	⟵ dis: address ⟵	0.30	0.30	0.6	<0.18, 0.58>
9.	0.50	0.15	0.88	<0.44, 0.56>	⟷ **auth: confirmed**	0.30	0.30	0.6	<0.18, 0.58>
10.	0.50	0.15	0.88	<0.44, 0.56>	⟵ req: driving history ⟵	0.30	0.30	0.6	<0.18, 0.58>
11.	0.50	0.35	0.88	<0.44, 0.56>	⟶ dis: driving history ⟶	0.55	0.30	0.78	<0.43, 0.65>
12.	0.50	0.35	0.88	<0.44, 0.56>	**auth: confirmed** ⟷	0.55	0.30	0.78	<0.43, 0.65>
13.	0.50	0.35	0.88	<0.44, 0.56>	⟶ req: luggage info ⟶	0.55	0.30	0.78	<0.43, 0.65>
14.	0.60	0.35	0.85	<0.51, 0.66>	⟵ dis: luggage info ⟵	0.55	0.35	0.78	<0.43, 0.65>
15.	0.60	0.35	0.85	<0.51, 0.66>	⟵ req: space info ⟵	0.55	0.35	0.78	<0.43, 0.65>
16.	0.60	0.45	0.85	<0.51, 0.66>	⟶ dis: space info ⟶	0.65	0.35	0.75	<0.49, 0.74>
17.	0.60	0.45	0.85	<0.51, 0.66>	⟶ req: e-mail ⟶	0.65	0.35	0.75	<0.49, 0.74>
18.	0.70	0.45	0.83	<0.58, 0.75>	⟵ dis: e-mail ⟵	0.65	0.45	0.75	<0.49, 0.74>
19.	0.70	0.45	0.83	<0.58, 0.75>	⟵ req: address ⟵	0.65	0.45	0.75	<0.49, 0.74>
20.	0.70	0.65	0.83	<0.58, 0.75>	⟶ dis: address ⟶	0.85	0.45	0.81	<0.69, 0.88>
21.	0.70	0.65	0.83	<0.58, 0.75>	**auth: confirmed** ⟷	0.85	0.45	0.81	<0.69, 0.88>
22.	0.70	0.65	0.83	<0.58, 0.75>	⟵ **success** ⟶	0.85	0.45	0.81	<0.69, 0.88>