Privacy Preserving Trust Negotiation for Pervasive Healthcare

Changyu Dong and Naranker Dulay
Department of Computing
Imperial College London
London, UK SW7 2AZ
{cd04, nd}@doc.ic.ac.uk

***Abstract*— Healthcare systems are being extended to monitor patients with body sensors wirelessly linked to a mobile phone that interacts with remote healthcare services and staff. As such systems become more widespread, with multiple healthcare providers and security domains, the establishment of trust between users, providers and medical staff will become important. In this paper we implement the ETTG privacy-preserving trust negotiation protocol and show how it can be used to automatically establish mutual trust between interacting parties in compliance with access-control and disclosure policies. The protocol is implemented in Java and can be run on J2ME platforms. The trust negotiation steps are logged and the resulting trust graphs can be visualised to show how policy compliance was achieved. We also develop a new easier-to-understand syntax for ETTG and use it to define access-control and disclosure policies for a small pervasive healthcare scenario.**

I. Introduction

Future large-scale healthcare in the community will involve many different organisations cooperating in patient care, including doctors, hospitals, dentists, pharmacies, drug companies and insurance companies. With the advent of new on-body and embedded medical sensors, it is becoming feasible to contemplate new applications that offer real-time healthcare to patients, and involve complex workflows between the patient and services in different organisations. These workflows will typically require peers to establish secure connections (e.g. SSL) and have authorisation controls in place on resources. Traditional security methods are based on a shared security infrastructure, where parties are known to each other. The trust between parties is predefined. As the healthcare marketplace becomes more open and competitive, applications will need to select and interact with multiple providers and multiple security domains, trust management systems will be required to establish high levels of trust. Trust management systems [1]–[6] make trust decisions by reasoning and inferring an entity's competence, honesty and dependability from credentials, trust policy specifications, past experience and the entities reputation. Trust management is a more flexible means of establishing and evolving security and privacy in distributed, mobile and pervasive systems than the traditional centralised model because it does not require pre-knowledge of users and services nor a common security infrastructure. Such an approach will be essential in pervasive healthcare where applications will need to support legal requirements for user privacy and data protection. Trust decisions will be used to control what information should be released to another party, what resources the party should have access to, how to configure the mechanisms that will be needed to make the interaction secure etc.

Paradoxically, trust management systems, can become the source of privacy concerns. Trust management systems manage trust by collecting user information. The more accurate the information gathered, the more likely a correct decision can be reached. However, without enough justification and assurance, users may refuse to disclose information. There is a tension between respecting the users' right of controlling their data while also trying to avoid unnecessary failures caused by lack of information.

In automated trust negotiation, trust is established by incremental exchange of credentials, ideally protecting sensitive credentials in the process. In this paper we implement the Extended Trust Target Graph (ETTG) protocol proposed by Li *et al.* [7] and show how it can be used to establish trust in a small pervasive healthcare scenario. ETTG is the succesor to TTG [8] and extends TTG by adding native support for several cryptographic credential schemes and associated protocols. Our implementation is implemented in Java, uses TCP sockets for communication and can run on limited J2ME devices. It includes a new easier-to-understand syntax for specifying credentials and policies. The negotiation steps can be logged in a database for auditing and data provenance purposes and the corresponding trust graph animated to show how mutual trust was established or failed to be established.

II. Trust in Pervasive Healthcare

In the future people will have attached and embedded medical sensors as well as carry a mobile phone. The mobile phone will act as a gateway that will be able to communicate with the body sensors using a WPAN standard like ZigBee, and with remote services and medical staff using a mobile voice/video/data standard like 3G. The mobile phone will also provide a degree of local storage and processing capabilities and several interfaces (screen, keyboard, sound, voice, camera, video). Processing and communications will also need to be optimised for battery life and the pricing of cellular data transfers.

We envisage that the healthcare market will be flexible, with healthcare services being provided by both public and private organisations that collaborate behind the scenes. People will be able to decide and even negotiate which services they want. For example, a user will be able to subscribe to a medical monitoring service that can process the readings from the user's sensors forwarded via the user's mobile phone. If an emergency is detected the monitoring service could inform the user, the user's doctor and call an ambulance. The monitoring service will need access to the user's medical details that are relevant to the monitored condition as well as details of the hospital where the user may have had previous treatment for the condition. The monitoring service would then be able to liaise with the emergency services and the hospital to which the user will be taken for emergency treatment. Hospitals may need to interact with the user's doctor and possibly social services about caring for the user after treatment. In a small hospital, there may not be local expertise to evaluate medical information such as X-rays and ECG readings and so these need to be sent to a remote expert over the network. Perhaps the user's usual consultant is not available and a new one has to be chosen. In some contexts, the user's medical insurance company will need to be included in the service provisioning and workflow. The monitoring service may like to provide anonymised monitoring records of the user for medical research perhaps by offering a discounted price for the service.

Issues of trust, privacy, security and context pervade this simple scenario. Should the user trust the monitoring service? Can the user verify the credentials of the monitoring service? Can the user ensure that only those parts of his medical record pertinent to the monitored condition are disclosed by the user's doctor/hospital to the monitoring service? Should the user allow his monitored data to be anonymised and passed on? Should the hospital trust the monitoring service? Should the hospital rely on the data and assessment from the monitoring service? Should the monitoring service trust the user and the readings from the user's sensors? The list is long.

Even when the general workflow of service interactions is known, circumstances (context) will require decisions to be made dynamically. We would like to localise and automate these decisions as much as possible. In particular we would like the decision on whether a connection (secure on or not) should be established with a particular party to be based on local policies. We also want privacy preserving policies that can be used to control what information should be disclosed (including the credentials used) in the trust negotiation process. The other party will do likewise, leading to a degree of mutual trust.

III. Automated Trust Negotiation

Automated trust negotiation (ATN), first proposed in [9] by Winsborough *et al.*, uses incremental exchange of credentials to make trust decisions. In ATN, policies are used to regulate the disclosure of credentials, in addition to the granting of system resources. *Disclosure policies* state the conditions under which a party can release a sensitive credential during a negotiation. In particular users can refuse to disclose a credential if it is not relevant to a request, or choose to disclose other credentials instead. Because there may not be enough trust to convince users to release sensitive credentials in one step, trust negotiation allows incremental exchange of credentials. At the beginning of the negotiation the two parties begin by disclosing credentials that are not sensitive. Initial trust can be established by evaluating these credentials and disclosure policies for more sensitive credentials that may be satisfied. We can then reveal these more sensitive credentials and iterate the process until eventually we satisfy or fail the policy of the desired resource.

A notable vulnerability of the trust negotiation model [10] is that the credential holder may release more information even with well defined disclosure policies. A credential may be *possession-sensitive* if knowledge of the possession of a specific credential is considered sensitive. When a request for such a credential is received, a party may reply with a counter-request if it has the credential because it needs to establish higher level of trust in order to disclose it or do nothing if it doesn't have it. However the other party can infer the possession or non-possession of the credential by observing the reply. Winsborough and Li [8] try to address this problem by introducing acknowledge policies. Here a party can define acknowledge policies on their possession-sensitive credentials such that the credential holder will not acknowledge the request until the counter-party satisfies the acknowledge policy.

A. Cryptographic Privacy Protection

Cryptographic tools and protocols can be used to improve the privacy protection and effectiveness of ATN. Conventional credentials such as X.509 certificates [11] store attribute information in clear form and the information is disclosed in an all-or-nothing fashion. If there are some sensitive attributes in the credential, the credential is not disclosed which may cause the negotiation to fail unnecessarily. Cryptographic credentials such as anonymous credentials [12], [13], privacy enhanced credentials [14] and OAcerts [15] use a commitment scheme to store attribute values in a encrypted and verifiable form. When requested, the credential holder can disclose the credential without revealing the attribute values stored in it because the counter-party learns nothing about the attribute values by looking at the commitments. The credential holder can also selectively show the attributes in a credential by only opening the commitments what he wants to disclose. Other schemes such as zero-knowledge proofs [16], hidden credentials [17], [18], OSBE [19] and OCBE [15] can also be used to validate credentials and attribute values without actually disclosing them.

IV. ETTG Trust Negotiation Language

In the ETTG, each negotiating party has a trust policy store which contains credential, attribute and policy declarations.

1. Credential declarations describe the credentials a party has. The underlying framework is designed to support various credentials, including standard X.509 certificates [11], hidden credentials [17], [18], anonymous credentials [12], [13], OAcerts [15] as well as unsigned statements. There are two kinds of credentials: *direct credentials* and *delegation credentials*. Direct credentials are statements that attest to one or more of the negotiating party's attributes. Delegation credentials are statements of delegation of authority. Both credentials take the form:

negotiator.signer.credentialname(attrlist)

The negotiator and signer fields can be substituted by **self** or **peer** when defining policies to denote each of the negotiating parties. This allows negotiator-independent policy specifications to be written. In the case of unsigned statements, the signer field is always **any**.

In the following, the direct credential is a statement signed by the NHS [1] which asserts Bob's name, date of birth and NHS registration number, while the delegation credential states that the NHS delegates the authority to certify specialists to Barts hospital.:

credential Bob.NHS.Registration(name = Bob,
DoB = 1942/02/28, number = 12345)
delegation BartsHosp.NHS.Specialist()

Dummy credentials are used as a placeholder in policy statements to ensure compliance with the TTG graph processing rules. They take the form *credentialname(attrlist)*. They don't have the negotiator and signer fields so can easily be distinguished from real credentials. Usually, we use dummy credentials to represent a resource. The goal of the corresponding policy can be understood as the signing of an access ticket to the requester. An example is the *diagnosis()* credential in the first accept policy P_1 in section IV-A.

2. Attribute declarations are used to define the sensitivity of individual credential attributes. The keyword **nonsensitive** means that the attribute can be revealed to anyone, while the keyword **sensitive** indicates that at least one of the disclosure policies for the attribute must be satisfied before the attribute can be disclosed. The declarations also define a more convenient shorter name for the credential attribute for use in disclosure policies. For example:

nonsensitive
speciality = Alice.BartsHosp.Specialist(speciality)

means that the speciality attribute is considered as nonsensitive and refers to the attribute speciality in the credential *Alice.BartsHosp.specialist*. If a party has several credentials with the same attribute, they can be listed in the same attribute declaration. For example:

nonsensitive DoB = Alice.Gov.Passport(DoB),
Alice.Gov.IDCard(DateOfBirth)

3. Policy declarations are used to control access to resources, credentials and the disclosure of information. Policies can either be declared unconditionally **true** and are always satisfied or can be defined as a conjunction of credentials in which case, the counter party must provide all the stated credentials. Policies can have a precondition to control the disclosure of the policy. The precondition can be **false** or a credential. If it is false, then the precondition will never be satisfied and the protected part will not be revealed. If the precondition is a credential, then the counter-party must provide the credential to satisfy the precondition and learn the policy.

[1]National Health Service in the UK

There are six types of policy:

- **accept** policies are used to control access to a resource. An example of access control policy is the policy P_1 in section IV-A.
- **disclose ac** policies are used to control access to a credential. If there are disclose ac policies associated with a requested credential, we will issue a counter-request and wait until it is satisfied before sending a copy of the requested credential to the counter-party.
- **disclose ack** policies are used to control the disclosure of the information that we possess such a credential. If there is a disclose ack policy associated with a requested credential, we will not acknowledge the request, e.g. in the form of disclosing the credential or sending a counter-request, until the counter-party satisfies the disclose ack policy.
- **disclose full** policies are used to control the disclosure of an attribute value. When a disclose full policy is satisfied, the attribute value will always be disclosed to the counter-party. This can be done by sending the credential linked to the attribute reference to the counter-party, opening the commitment or simply sending a value in the case of an unsigned credential.
- **disclose range** policies are used to control the disclosure of an attribute value. When a disclose range policy is satisfied, the counter-party can learn a range in which the attribute value exists. The range is determined by the precision given in the policy.
- **disclose bit** policies are used to control the disclosure of an attribute value. If a disclose bit policy is satisfied, the counter-party is entitled to receive one bit of information about the attribute value. This information can be the result of an evaluation performed on the attribute value, for example, whether it is greater than a particular number.

A. Example

Alice is a doctor of Hospital BartsHosp who provides a remote diagnostic service. To perform a remote diagnosis, Alice requires the patient to provide two credentials. The first is a digital credential from the NHS containing the patient's registration number which can be used to locate the patient's Electronic Medical Record. The second is the current body sensor data needed for the diagnosis. The policy P_1 offers the diagnosis service. In the policy **peer** denotes the requesting party, while **any** is used when a credential does not need to be signed. Note: **self** could be used instead of Alice in the body of this trust policy.

Alice has following credentials: C_1, C_2 and C_3. The first credential C_1 is signed by General Medical Council (GMC) which certifies that Alice is a doctor in good standing. The

```
trustpolicy: Alice
    accept diagnosis() if
        peer.NHS.Registration(number) and peer.any.MonitorData()                              [P1]
    credential Alice.GMC.Doctor()                                                             [C1]
    credential Alice.BartsHosp.Specialist(speciality = Heart)                                 [C2]
    delegation BartsHosp.NHS.Specialist()                                                     [C3]
    nonsensitive speciality = Alice.BartsHosp.Specialist(speciality)                          [A1]

trustpolicy: Bob
    credential Bob.NHS.Registration(name = Bob, DoB = 1942/02/28, number = 12345)             [C4]
    sensitive number = Bob.NHS.Registration(number)                                           [A2]
    sensitive DoB = Bob.NHS.Registration(DoB)                                                 [A3]
    nonsensitive name = Bob.NHS.Registration(name)                                            [A4]
    disclose full number if peer.GMC.Doctor()                                                 [P2]
    disclose full DoB if peer.GMC.Doctor()                                                    [P3]
    disclose bit DoB if true                                                                  [P4]
    disclose ac Bob.BodySensors.MonitorData() if peer.NHS.Specialist(speciality = Heart)      [P5]

trustpolicy: MedMon
    accept discount() if peer.NHS.Registration(DoB ≤ 1946/01/01)                              [P6]
```

Fig. 1. The Policies for the Example

second credential C_2 is signed by the hospital which certifies Alice is a heart specialist. There is also a delegation credential C_3 which is issued by the NHS that delegates to the hospital the authority to certify specialists.

The attribute speciality in C_2 is not considered sensitive, so we can declare it in A_1.

Bob is a patient who wants to use the remote diagnosis service (e.g. provided by Alice). Bob has a credential C_4 which is the registration certificate issued by the NHS containing his name, date of birth and registration number.

Bob considers his number and DoB as sensitive, so in his trust policy store, he adds two sensitive declarations, one for the registration number A_2 and the other for the DoB A_3. The name attribute is considered as non-sensitive, so Bob also has a declaration A_4 for it.

The disclosure of the registration number and DoB is controlled by disclosure policies P_2, P_3 and P_4. The policies P_2 and P_3 state that the attributes can be disclosed if the peer (Alice in this scenario) is a GMC certified doctor. P_4 states that Bob is willing to tell the peer whether his DoB attribute satisfies the peer's policy in any situation. Bob will run a zero-knowledge proof protocol to convince the peer without revealing the exact value. Bob also has body sensors which can provide real time monitoring data. The data can be deemed as a unsigned credential therefore can be protected by an ac policy P_5 which states that the monitoring data can be disclosed if the peer is certified by the NHS as a heart specialist.

Finally, consider a monitoring service provider MedMon which offers a discount to senior citizens (over 60 years old). To receive the discount, the subscriber must provide a digital credential from the NHS containing his Date of Birth. The policy store for MedMon contains only one policy P_6. Bob negotiates with MedMon using the same credentials and policies as described before.

V. Trust Negotiation

The trust negotiation process involves two parties working together to successfully construct a trust target graph (TTG). The negotiation begins when the requesting parties requests access to a resource. The resource holder creates a TTG containing one node which represents the request. The first node is called the *primary target*. The resource holder then tries to process the TTG by adding more nodes into the graph according to its local trust policy specifications. If it cannot process the graph further, it sends the partially processed graph to the requesting party.

In each subsequent round, one party tries to process the graph, making changes and sends these changes to the counter-party. If necessary and allowed, credentials are released to justify the changes. The counter-party verifies the changes against the credentials. It then updates its local copy of the TTG to reflect these changes. The negotiation succeeds when the primary target is satisfied. It fails when the primary target fails, or when neither party can no longer change the graph.

A. Trust Target Graphs

The trust-target graph is directed graph that represents the negotiation process. Nodes in a TTG are called *trust targets* and are used to represent each parties questions and replies. In any step, the party that asks a question is called the *verifier* and the counter-party is called the *opponent*. There are five kinds of trust target nodes in a TTG:

1) A *credential target* means that the verifier wants to see proof that the opponent has a particular credential.
2) A *policy target* means the verifier wants to see proof that the opponent satisfies a policy.
3) A *conjunction target* means the verifier wants to see a conjunction of credentials.
4) An *attribute goal* means that the verifier wants the opponent to show him the value of an attribute.

	Credential Target	Policy Target	Conjunction Target	Attribute Goal
Credential Edge	at least 1	N/A	N/A	N/A
Policy Edge	at least 1	N/A	N/A	N/A
Policy Control Edge	N/A	N/R	N/A	N/A
Policy Expansion Edge	N/A	at least 1	N/A	N/A
Conjunction Edge	all	N/A	all	N/A
Attribute Control Edge	N/R	N/A	N/A	at least 1

TABLE I
SATISFACTION CONDITIONS

5) A *trivial target* has no particular meaning and is used as a place holder when the parties need to add an edge into the graph.

Nodes in a TTG are connected by edges. An edge in TTG is always directed from the child node to the parent node. There are 6 types of edges in a TTG:

1) A *credential edge* points from a credential target or a trivial target to another credential target. It needs to be justified before being added into the TTG. For trivial targets, a direct credential must be attached to show that the opponent has the credential requested in the parent node. For credential targets, a delegation credential must be attached to prove the delegation of authority. The credential will be revealed to the verifier if the child node is satisfied.
2) A *policy edge* points from a policy target to a credential target and the credential specified in the credential target is specified in the policy target.
3) A *policy control edge* points from a credential target to a policy target. The credential specified in the credential target is the precondition specified for the policy target.
4) A *policy expansion edge* points from a conjunction target to a policy target in which case the conjunction is in the policy body. If the policy body is the keyword *true*, then the child should be a trivial target.
5) An *conjunction edge* points from a credential target to a conjunction target where the credential is part of the conjunction. It can also points from an attribute goal to a credential target where the attribute is contained in the credential.
6) An *attribute control edge* points from a policy target to a credential target or an attribute goal. The policy is a disclosure policy which controls the disclosure of the credential or the attribute. Each attribute control edge has a tag consisting of ac, ack, full, bit or range which denote the type of the disclosure policy.

B. Graph Construction

Each node has a processing state which is a pair of boolean states: *verifier-processed* and *opponent-processed*. A node will be *fully processed* if both states are true.

When a new node is added to a TTG, its processing state is initialised. For a trivial target, it will be initialised as fully processed. For a policy target or an conjunction target, it will be initialised as opponent-processed. For an attribute goal, it will be initialised as verifier-processed. For a credential target, the initialisation is more involved: if the credential is a *dummy credential*, which means the credential is not listed in the credential declarations and is created only for the purpose of defining a policy statement, then the corresponding credential target will be initialized as opponent-processed. Otherwise, it will be initialized as verifier-processed.

The parties use the processing state to distinguish which nodes have been processed by them. If the node is verifier-processed, the verifier will not process it further. If the node is opponent-processed, the opponent will not process it further. If the node is fully-processed, then neither party will process it. In each round, the party finds those nodes which haven't been completely processed by it, processes each of them and marks them as processed when it has finished all the processing steps on them.

Each node also has a *satisfaction state* which can be satisfied, failed or unknown. A trivial target will be created as satisfied. All the other nodes will be created with a satisfaction state of unknown. The satisfaction state of these trust targets changes depending on the satisfaction state of their children and the edges between them and the children.

A target will fail if all its children failed, or one of the children connected to it using the conjunction edge failed, or it is fully processed but has no child. In table I, we describe the conditions under which the target will become satisfied. The first row shows the type of a target, the first column shows the type of edge between the target and its children. *N/A* means that the target cannot be connect with the child using this kind of edge. *N/R* means that the target can be connected with the child using this kind of edge, but the satisfaction state of the child has no influence on the satisfaction state of the parent. *At least 1* means if there are children connected to the target using this kind of edge, then the target will become satisfied if at least one of them is satisfied. *All* means if there are children connected to the target using this kind of edge, then the target will become satisfied if all of them are satisfied.

C. Negotiation Process for Example

In this section, we illustrate the negotiation process for the example discussed in section IV-A.

1) Negotiation between Alice and Bob: Figure 2 is the screenshot from our negotiation visualizer which shows the final result of the negotiation between Alice (doctor) and

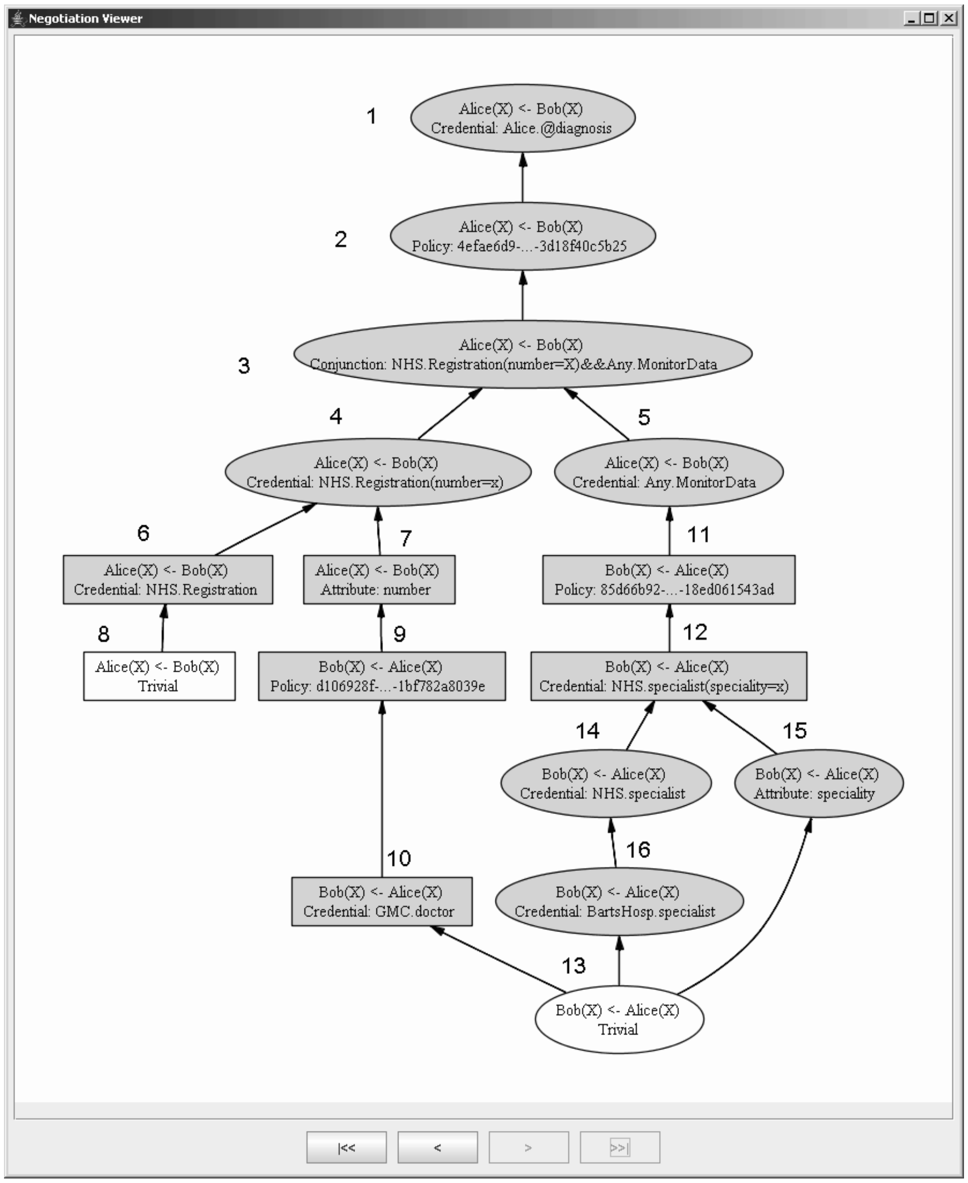

Fig. 2. Trust Negotiation Between Alice and Bob

Bob (patient). The ellipse nodes are added by Alice and the rectangle nodes are added by Bob. Each node shows the role of the parties, processing states, node type and the content of the node. For example, in node 1, $Alice(X) \leftarrow Bob(X)$ means that Alice is the verifier and Bob is the opponent. The "X" in the brackets next to a party's name means that the party has processed the node. $Credential : diagnosis$ means that this node is a credential target for the dummy credential diagnosis.

The negotiation begins when Bob sends a request for the remote diagnosis. Alice creates a new TTG and adds the request into the graph as a credential node (node 1). Node 1 is the primary node which means if it is satisfied, then the negotiation succeeds. Alice searches her trust policy store and finds that there is an access control policy P_1 for offering diagnosis. Alice then expands the graph by adding nodes 2, 3, 4, 5. Node 2 is a policy node which contains the reference to the policy. It is connected to node 1 by a policy edge. Node 3 is a conjunction node contains the policy body of the policy stated in node 2. Nodes 4 and 5 are derived from node 3 and each represents a credential requirement in the conjunction contained in node 3. The two nodes connect to node 3 using conjunction edges, which means both of them need to be satisfied in order to satisfy node 3. Alice cannot process the graph any more, so she passes the graph to Bob.

Bob then adds nodes 6 - 12 to the graph. Nodes 6 and 7 are derived from node 4. The framework is designed to support cryptographic credentials such as anonymous credentials and OAcerts which allow separation of credential disclosure from attribute disclosure. Node 4 is a request for a credential plus the attribute contained in it. Node 6 and node 7 are used to separate the disclosure of these two pieces of information. Node 6 which is a credential node requests only the credential, while node 7 is an attribute node which requests only the attribute. These two nodes are also connected to the parent by conjunction edges. Node 7 is further expanded because there is a disclosure policy for the attribute value. Node 9 contains the reference to the policy and node 10 contains the body of the policy. Similarly, nodes 11 and 12 are added. A trivial node (node 8) is added as child of node 6 because there is no policy controls for the disclosure of the credential requested in node 6. The credential is attached to the credential edge between node 6 and node 8. Since node 8 is a trivial node and is satisfied when created, the credential is released to the verifier. The verifier gets the requested credential thereby node 6 becomes satisfied. Bob then passes the graph back to Alice.

Alice adds nodes 13 - 16 and the negotiation succeeds in this round. Node 14 and 15 are added to separate the disclosure of the credential and the attribute requested in node 12. Node 10 and node 15 are trivially satisfied since there are no policies to control the disclosure of the credential and attribute requested in these two nodes. The satisfaction of node 10 also satisfies node 9 and node 7 in turn. Now both nodes 6 and 7 are satisfied, node 4 is also satisfied. On the other hand, Alice doesn't have the credential requested in node 14, but she has a direct credential and a delegation credential which are effectively equivalent to the credential requested. So she adds node 16 and a credential edge between node 16 and 14 which the delegation credential is attached to. Node 16 is trivially satisfied and both credentials are revealed to Bob. Consequently, node 14 is satisfied. Node 12 is now satisfied because both children are satisfied and the attribute value for node 15 satisfies the requirement in node 12. Node 11 and node 5 are satisfied as a consequence. Now both nodes 4 and 5 are satisfied, so node 3 is satisfied, then node 2, and then finally node 1, the primary node, is satisfied.

2) Negotiation between MedMon and Bob: The negotiation process is shown in figure 3.

MedMon first creates nodes 1, 2, 3 which represent its requirement for the subscribers to get the discount. Nodes 4 and 5 are added by Bob to represent the two sub-requests derived from node 3. Node 4 is trivially satisfied since there are no disclosure policies related to the required credential. Node 5 has two related disclosure policies, one is a **disclose full** policy which requires the counter-party to have a doctor certificate from the GMC, the other is a **disclose bit** policy which is unconditionally true. Since MedMon does not have the credential required, it cannot satisfy node 9 to learn the exact Date of Birth of Bob. But MedMon can run a zero-knowledge proof protocol with Bob to see whether the attribute value satisfies its policy or not. The evaluation result turns out to be true, so node 5 is satisfied. The success also triggers the satisfication of node3, node 2 and at last, the primary node.

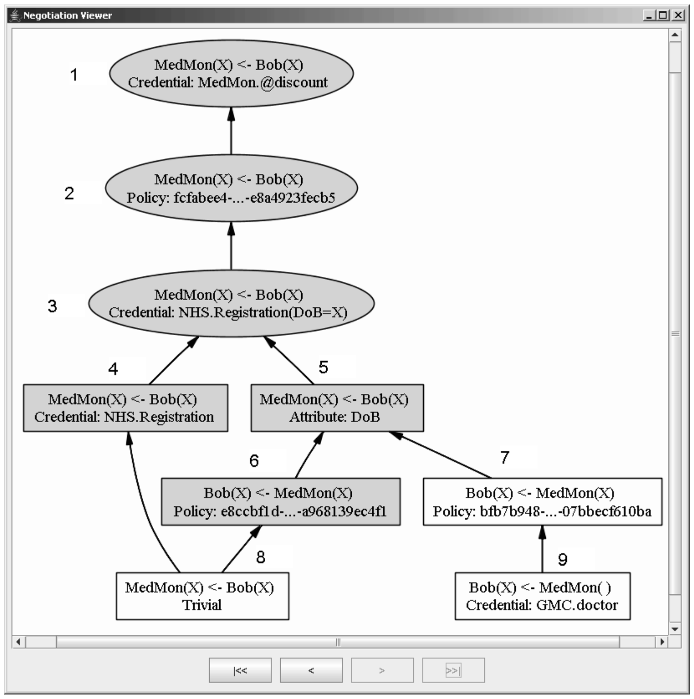

Fig. 3. Trust Negotiation Between MedMon and Bob

VI. Related Work

A number of trust negotiation schemes have recently been developed.

TrustBuilder [20], [21] aims to build a ubiquitous and scalable trust negotiation system which enables interaction across security domain boundaries in any situation. A variety of strategies are defined to allow strangers to establish trust through the exchange of digital credentials and the use of access control policies that specify what combinations of credentials a stranger must disclose in order to gain access to each local service or credential. TrustBuilder also provides a trust negotiation protocol, Trust Negotiation in TLS (TNT) protocol [22], which extends the trust negotiation in TLS handshake protocol. A problem of TrustBuilder is the lack of adequate protection on possession-sensitive credentials. The two strategies proposed in [10] either increases the risk of the negotiation failing or enables the negotiator to gather excessive information about the other party.

Trust Target Graph (TTG) [8] uses RT_0 [4] for specifying credentials and policies. By using RT_0, the system takes advantage of its expressive power in attribute-based reasoning, delegation support and credential discovery. It also exploits new approaches to protect sensitive credentials and attributes by introducing the notion of attribute acknowledgment policies. In [23], the authors also introduced a formal notion of safety for ATN which is based on the possibility of third parties inferring information on the negotiating parties profiles. Based on the formal notion, the authors present a family of negotiation strategies that uses the TTG protocol that support a credential system with delegation and show that these strategies provide credential-combination hiding with probabilistic indistinguishability.

Trust-X [24] proposed by Bertino *et al.* is an XML-based framework for trust negotiations. It was designed specifically for peer-to-peer interactions in which both negotiating parties are equally responsible for negotiation management. An XML-based language, X-TNL, is used to specify certificates and policies. The system supports both CA-signed credentials and self-signed declarations. Trust-X allows *trust tickets* which are generated at the end of a successful negotiation to be used in negotiation. If similar negotiations are executed repeatedly between the same parties, trust tickets efficiently reduce the negotiation time and traffic. A further extension of Trust-X enables it to support partial attribute disclosure and negotiation based on P3P policies [14].

VII. Summary and Future Directions

In this paper we described the implementation of a privacy-preserving trust negotiation protocol and showed how it can be used to automatically establish mutual trust between interacting parties if their access-control and disclosure policies comply. The protocol is implemented in Java and can be run on J2ME platforms. The trust negotiation steps are recorded and the resulting trust graphs can be visualised to show how policy compliance was achieved. We also developed a new syntax for ETTG trust negotiation and illustrated its use with a simple pervasive healthcare scenario.

We are currently experimenting with more complex examples involving several parties and hope to trial them on them on our WPAN comprising a Nokia 6680 phone running Symbian OS and locally developed body-sensor nodes running TinyOS[2]. We are also looking at optimisations to reduce the communication overheads of trust negotiation, particular for servers that concurrently process many negotiations.

One aspect of our future work is to use cryptography to achieve "perfect privacy". Cryptography can dramatically reduce the unnecessary disclosure, but not fully. Commitment schemes minimise the disclosure of information, but must reveal the commitment in order to convince the other party. Zero-knowledge proofs leak the possession of credentials and are subject to probing attack. OSBE and Hidden Credentials can be used to protect possession-sensitive credentials, but their oblivious property is only valid when the policies don't need to test attribute values. OCBE enables the oblivious use of attributes which prevent probing attack on the sensitive attributes, but we must show the credentials containing the attributes to the other party. We are looking at whether we can achieve oblivious usage of credentials and oblivious usage of attribute values simultaneously. We are also looking at the use of secure computation techniques and anonymising proxies in our negotiation framework to enhance security and privacy in pervasive healthcare.

[2]A Gumstix currently bridges the body-sensor network with the phone

Acknowledgments

This research was supported by the UK's EPSRC through grant EP/C537181/1 and forms a part of CareGrid, a collaborative EPSRC project between Imperial College and the University of Cambridge that is developing middleware support for trust, privacy, security and context in pervasive healthcare systems. The authors would also like to thank the members of the Policy Research Group for their continuing support.

References

[1] M. Blaze, J. Feigenbaum, and J. Lacy, "Decentralized trust management," in *SP '96: Proceedings of the 1996 IEEE Symposium on Security and Privacy*. Washington, DC, USA: IEEE Computer Society, 1996, p. 164.

[2] M. Blaze, J. Feigenbaum, and A. D. Keromytis, "Keynote: Trust management for public-key infrastructures (position paper)," in *Proceedings of the 6th International Workshop on Security Protocols*. London, UK: Springer-Verlag, 1999, pp. 59–63.

[3] A. Herzberg, Y. Mass, J. Michaeli, Y. Ravid, and D. Naor, "Access control meets public key infrastructure, or: Assigning roles to strangers," in *SP '00: Proceedings of the 2000 IEEE Symposium on Security and Privacy*. Washington, DC, USA: IEEE Computer Society, 2000, p. 2.

[4] N. Li, J. C. Mitchell, and W. H. Winsborough, "Design of a role-based trust-management framework," in *SP '02: Proceedings of the 2002 IEEE Symposium on Security and Privacy*. Washington, DC, USA: IEEE Computer Society, 2002, p. 114.

[5] B. Yu and M. P. Singh, "An evidential model of distributed reputation management," in *AAMAS '02: Proceedings of the first international joint conference on Autonomous agents and multiagent systems*. New York, NY, USA: ACM Press, 2002, pp. 294–301.

[6] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, "The eigentrust algorithm for reputation management in p2p networks," in *WWW '03: Proceedings of the 12th international conference on World Wide Web*. New York, NY, USA: ACM Press, 2003, pp. 640–651.

[7] J. Li, N. Li, and W. H. Winsborough, "Automated trust negotiation using cryptographic credentials," in *CCS '05: Proceedings of the 12th ACM conference on Computer and communications security*. New York, NY, USA: ACM Press, 2005, pp. 46–57.

[8] W. Winsborough and N. Li, "Towards practical automated trust negotiation," in *POLICY '02: Proceedings of the 3rd International Workshop on Policies for Distributed Systems and Networks (POLICY'02)*. Washington, DC, USA: IEEE Computer Society, 2002, p. 92.

[9] W. H. Winsborough, K. E. Seamons, and V. E. Jones, "Automated trust negotiation," in *DARPA Information Survivability Conference and Exposition, 2000. DISCEX '00*, vol. 1. Hilton Head, SC: IEEE Press, 2000, pp. 88–102 vol.1.

[10] K. E. Seamons, M. Winslett, T. Yu, L. Yu, and R. Jarvis, "Protecting privacy during on-line trust negotiation." in *Privacy Enhancing Technologies*, ser. Lecture Notes in Computer Science, R. Dingledine and P. F. Syverson, Eds., vol. 2482. Springer, 2002, pp. 129–143.

[11] R. Housley, W. Polk, W. Ford, and D. Solo, "RFC3280: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile," IETF RFC Publication, April 2002. [Online]. Available: http://www.ietf.org/rfc/rfc3280.txt

[12] J. Camenisch and E. V. Herreweghen, "Design and implementation of the idemix anonymous credential system," in *CCS '02: Proceedings of the 9th ACM conference on Computer and communications security*. New York, NY, USA: ACM Press, 2002, pp. 21–30.

[13] J. Camenisch and A. Lysyanskaya, "An efficient system for non-transferable anonymous credentials with optional anonymity revocation." in *EUROCRYPT*, ser. Lecture Notes in Computer Science, B. Pfitzmann, Ed., vol. 2045. Springer, 2001, pp. 93–118.

[14] E. Bertino, E. Ferrari, and A. C. Squicciarini, "Privacy-preserving trust negotiations." in *Privacy Enhancing Technologies*, ser. Lecture Notes in Computer Science, D. Martin and A. Serjantov, Eds., vol. 3424. Springer, 2004, pp. 283–301.

[15] J. Li and N. Li, "Oacerts: Oblivious attribute certificates." in *ACNS*, ser. Lecture Notes in Computer Science, J. Ioannidis, A. D. Keromytis, and M. Yung, Eds., vol. 3531, 2005, pp. 301–317.

[16] R. Cramer and I. Damgård, "Zero-knowledge proofs for finite field arithmetic; or: Can zero-knowledge be for free?" in *CRYPTO*, ser. Lecture Notes in Computer Science, H. Krawczyk, Ed., vol. 1462. Springer, 1998, pp. 424–441.

[17] J. E. Holt, R. W. Bradshaw, K. E. Seamons, and H. Orman, "Hidden credentials," in *WPES '03: Proceedings of the 2003 ACM workshop on Privacy in the electronic society*. New York, NY, USA: ACM Press, 2003, pp. 1–8.

[18] R. W. Bradshaw, J. E. Holt, and K. E. Seamons, "Concealing complex policies with hidden credentials," in *CCS '04: Proceedings of the 11th ACM conference on Computer and communications security*. New York, NY, USA: ACM Press, 2004, pp. 146–157.

[19] N. Li, W. Du, and D. Boneh, "Oblivious signature-based envelope," in *PODC '03: Proceedings of the twenty-second annual symposium on Principles of distributed computing*. New York, NY, USA: ACM Press, 2003, pp. 182–189.

[20] M. Winslett, T. Yu, K. E. Seamons, A. Hess, J. Jacobson, R. Jarvis, B. Smith, and L. Yu, "Negotiating trust on the web." *IEEE Internet Computing*, vol. 6, no. 6, pp. 30–37, 2002.

[21] K. E. Seamons, T. Chan, E. Child, M. Halcrow, A. Hess, J. Holt, J. Jacobson, R. Jarvis, A. Patty, B. Smith, T. Sundelin, and Y. Lina, "Trustbuilder: negotiating trust in dynamic coalitions," in *DARPA Information Survivability Conference and Exposition, 2003.*, vol. 2. IEEE Press, 2003, pp. 49–51 vol.2.

[22] A. Hess, J. Jacobson, H. Mills, R. Wamsley, K. E. Seamons, and B. Smith, "Advanced client/server authentication in tls." in *NDSS*. The Internet Society, 2002.

[23] W. H. Winsborough and N. Li, "Safety in automated trust negotiation." in *IEEE Symposium on Security and Privacy*. IEEE Computer Society, 2004, pp. 147–160.

[24] E. Bertino, E. Ferrari, and A. C. Squicciarini, "Trust-x: A peer-to-peer framework for trust establishment." *IEEE Trans. Knowl. Data Eng.*, vol. 16, no. 7, pp. 827–842, 2004.