AN AGENT-BASED THRESHOLD PAYMENT MODEL FOR METERING WEB SERVICES

Akbar Siami Namin[1,2,3], Ruizhong Wei[2], Weiming Shen[3], Hamada Ghenniwa[1]

[1]Dept. of Electrical and Computer Eng., Univ. of Western Ontario, London, ON, Canada
[2]Dept. of Computer Science, Lakehead University, Thunder Bay, ON, Canada
[3]National Research Council Canada – IMTI, London, ON, Canada

asiamina@uwo.ca, Wei@ccc.cs.lakeheadu.ca, Weiming.Shen@nrc-cnrc.gc.ca, hghenniwa@eng.uwo.ca

Abstract. Service-Oriented technology is an emerging paradigm for services integration over the Internet. Use of applications as services across heterogeneous hardware and software platforms has raised new challenges. Access to Web services and the requirements to provide a secure and trusted payment model for the Web services in a malicious environment such as the Internet is among those challenges that must be addressed. Cryptographic approaches known as "Metering Schemes" provide some sophisticated solutions for access control to a Web service. This paper proposes an agent-based threshold payment model for metering Web services. The model has taken into account the abilities of software agents in reasoning and problem solving for designing some cryptographic concepts such as threshold schemes.

Keywords: Web Services, Agents, Security, Cryptography, Threshold Schemes, Trusted Payment Model.

1. INTRODUCTION

Service-Oriented technology, an emerging paradigm for integrating legacy applications over the Internet, is recognized as a major technology for the future of distributed computing. A service is a contractually defined behavior that can be implemented and provided by any component for any other component to use, based solely on the contract [12]. In Service-Oriented paradigm, the entities are treated as services and the main concern is what they can "*do*" rather than what they "*are*" and "*how*" they are constructed. In fact, by considering the

functionalities of entities, the main significant issue is to connect these services and provide new kinds of services namely as "*service integration*".

Service integration through the heterogeneous environments has introduced new challenges in software engineering. Among them is to provide a secure payment model to use a Web service while keeping the trustworthiness and privacy of Web services in a risky environment (e.g., the Internet) so that no entities trust each other without paying attention to some security issues. Metering a Web service can be considered for some other purposes as well. Counting the accurate measure of the number of access to a page, a site, or a Web service, identifying the highly requested services, and providing some mechanisms for advertising businesses are among some application domains of metering a Web service.

Threshold secret sharing schemes are conceptual cryptographic approaches to provide some mathematically proven secure solutions for distributing secure shares to participants such that by pooling a specific numbers of shares, known as threshold value, they can reconstruct the original secret value and claim for eligibility. The threshold schemes prove that any coalition of the participants lower than the threshold value is not able to reveal the original secret value. In other words, any coalition of participants of lower than the threshold value has completely "*no knowledge*" about the original secret. This concept can be useful in some areas such as metering. However, choosing a suitable numbers of the initial shares is not straightforward and it requests some knowledge about the number of participants in any dynamic environment. In other words, the dynamic changing of the number of participants in different time frames requires other supporting approaches.

Agent-Oriented paradigm has emerged as a promising technology for dealing with cooperation, coordination and decision-making in distributed applications. Software agents have been developed with sophisticated interaction patterns. They are efficient in enforcing automatic and dynamic collaborations. Agent-orientation is an appropriate design paradigm for e-Business systems with complex and distributed transactions. The "*reasoning"* and "*problem solving"* capabilities of software agents provide the feasibility of solving some

difficult problems in developing a threshold-based system. The shares generation and decision about the number of shares are considered as a case that software agents are capable of fulfilling it by means of their reasoning and problem solving capabilities.

This paper proposes an agent-based threshold payment model for metering Web services. We discuss the ability of software agents to reason and solve the problem for shares generation in metering a Web service based on the threshold schemes. The rest of the paper is structured as follows. Section 2 reviews the related work. Section 3 briefly introduces Shamir's *(k, n)* threshold scheme, which is the basic conceptual requirement of the paper. Section 4 presents an agent-based threshold payment model for metering Web services. Section 5 discusses some aspects of the model. Section 6 provides some conclusions and discusses the future work.

2. RELATED WORK

Visiting Web pages and payment methods to Web servers are considered as cryptographic approaches, namely as "*Metering Schemes*", rather than implementing some simple models.

A tool for secure and authenticated Web metering based on a hash function is proposed in [3]. The hash function counts the numbers of visits to a Web sever.

Auditable metering with lightweight security based namely as compact metering scheme is proposed in [5]. The paper proposes a timing function that can be performed incrementally and the output is compact.

An auditable metering scheme for Web advertisement applications is proposed in [4]. The approach is based on cryptographic mechanism for metering the duration and the number of instances of running a data process.

Obviously, the main approach for considering cryptographic aspects of metering schemes, in which a secret key has to be revealed by a set of participants, is the Shamir's simple threshold secret sharing scheme [11].

Naor and Pinkas were probably the first who proposed the secret sharing approach based on Shamir's threshold scheme [11] for metering schemes [9]. Their approach considers an environment, in which many servers serve a large number of clients and it is desired to meter the number of clients that served by the server.

Ogata and Kurosawa presented a revised version of Naor and Pinkas' approach by considering one more dimension [10].

Masucci has proposed some interesting conceptual cryptographic models for metering schemes [8]. In [2] they have discussed their main ideas, which are based on the information theory.

Efficient metering schemes with pricing are discussed in [7]. They have provided lower bounds on the size of the information distributed to clients and servers and on the number of random bits needed to set up a metering scheme with pricing.

Dynamic multi-threshold metering scheme was discussed in [1], in which for each server, an associated threshold value is considered for any time frame.

Metering schemes for general access structure is proposed in [6]. The paper constructs a metering scheme for general access structure, which includes multilevel and compartmented access structure.

In this paper we propose a metering model for Web services. The main concept is to use the Shamir's threshold scheme for secret sharing. However, our approach is different from the approaches mentioned above. The proposed model employs the two emerging paradigms (service orientation and agent orientation), as enabling technologies, in designing a threshold-based model for metering.

3. SHAMIR'S *(k, n)* THRESHOLD SCHEME.

Shamir's threshold scheme [11] is based on the mathematical fact that each polynomial $y = f(x)$ of degree $k-1$ is uniquely determined by k points (x_i, y_i) with distinct x_i, where $y_i = f(x_i)$.

The main purpose of a threshold scheme is to share a secret value, or a key, among participants such that any small group of participants cannot recover the key, but certain number of participants can rebuild the key. The main steps in the scheme are:

- The shares generator chooses a random number D as a secret key.
- The shares generator chooses a "*large enough*" prime number $p > Max(D, n)$ such that D, as a secret, should be divided into n shares and n is the number of participants. The purpose of picking a large prime number p is for interpolation in modular arithmetic rather than in real field. The set of integer numbers (module a prime number p) forms a *field* Z_p in which interpolation is possible [11].
- The shares generator defines $a_0 = D$
- The shares generator chooses $k-1$ random, independent coefficients $a_1,...,a_{k-1}$ in Z_p that form a polynomial:

$$f(x) = a_0 + a_1 x + a_2 x^2 + ... + a_{k-1} x^{k-1} \pmod{p}$$

- The shares generator chooses n random numbers x_i and computes n values D_i as $D_i = f(x_i)$ for $1 \le i \le n$. The pairs (x_i, D_i) are n secret shares.
- Any group of k or more users pool their shares and compute coefficients a_j, $0 \le j \le k-1$ of $f(x)$, using Lagrange interpolation.
- Therefore, they can compute $f(0) = a_0 = D$ and reveal the secret.
- Any group of $k-1$ or less users has no information about D [11] (perfectly secrecy).
- The security of the scheme is proven by the fact that Shamir's scheme is "*unconditionally*" secure scheme.

4. AN AGENT-BASED *(k, n)* THRESHOLD PAYMENT MODEL FOR METERING WEB SERVICES

We propose an agent-based payment model for visiting a Web service. Figure 1 depicts a simple scenario, in which a service requester is interested in using a Web service, by searching through the discovery service, achieves some knowledge about the services, policies, and the financial agency.

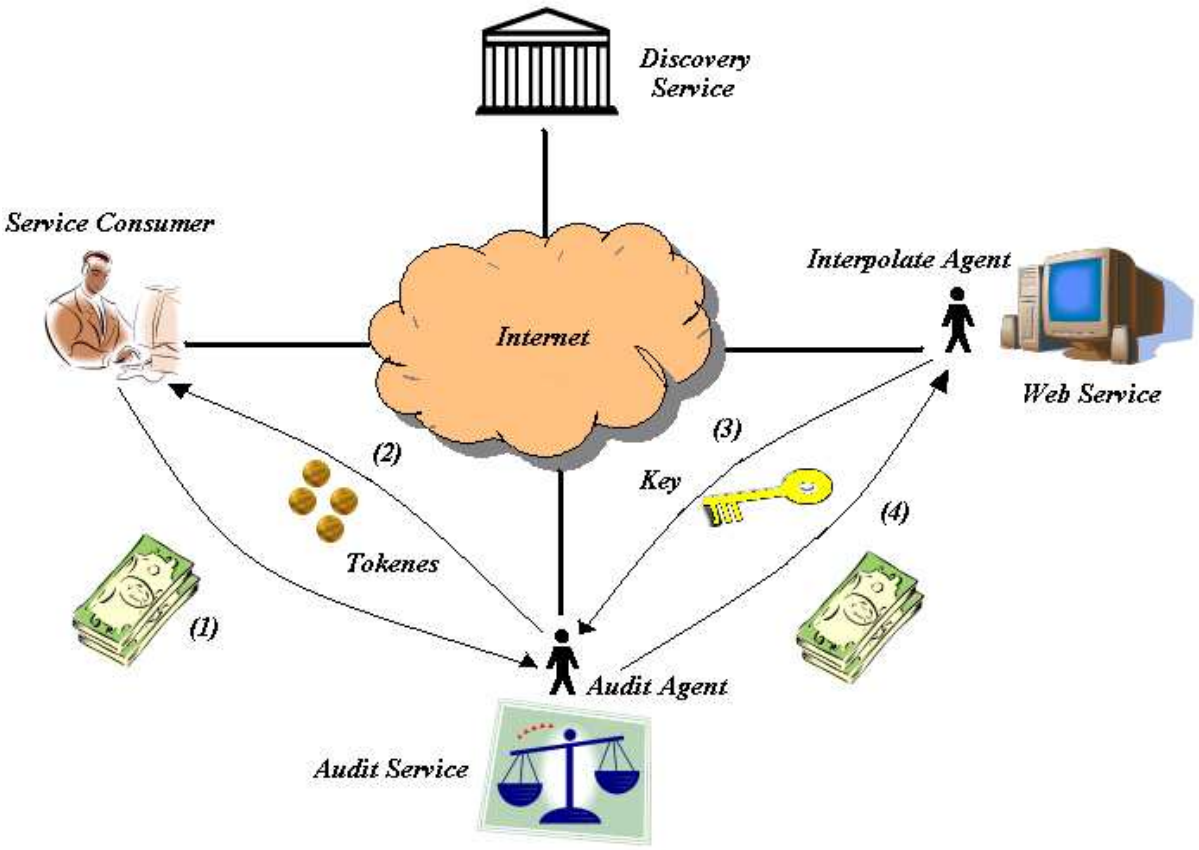

Figure 1 - A simple payment scenario for metering a Web service

The financial service (audit service) may be a registered Web service discoverable through the discovery service. It provides some facilities such as "*tokens*", or shares to sell for using a specific service. A service requester by buying some tokens from the audit service is able to visit and use a specific service. During each visit, a token must be submitted to the Web

service. In other words, each client by buying *m* tokens is able to use the target service for at most *m* times. The visited Web service, by achieving *k* visits may reconstruct a secret key and asks audit service for cost and payment. The audit service verifies the Web service's claim by comparing the received key with the real value, which is kept secret at the audit service.

The model is raising some security aspects such as the trust relationship between participants. As a matter of fact, no entities trust other entities; however, all participants trust the audit service, as a trusted financial agency.

The model is adopting the threshold secret sharing schemes as the main concept. However, designing a threshold scheme needs some knowledge in the number of generated shares and changing the number of generated tokens. The changing must be dynamic enough to handle the market requests.

The software agent and its unique capabilities can be considered as a supplementary technology for implementing a threshold scheme. The model considers two agents namely as "*interpolator agent*" and "*audit agent*". The interpolator agent is responsible for estimating the number of visits known as value of threshold, accepting the tokens, verifying them, reconstructing the secret key, and sending it to the audit service. However, the main responsibility of the interpolator agent is reasoning about the number of visits in any specific time frame and informing the audit agent to modify system with the estimated number of visits. The reasoning takes place based on some knowledge of previous time frames. The audit agent is responsible for setting up the threshold scheme, generating shares, selling tokens to requesters, accepting, verifying the revealed key, and modifying the threshold system with respect to the number of visits to a Web service.

The approach is based on Shamir's *(k, n)* simple threshold scheme, such that a secret key can be revealed by achieving knowledge about *k* values of *n* shares. To describe the model, we divide the scenario into some stages as following.

4.1. Initialization Stage

In order to provide a payment model for using a Web service, the "*Audit Service*", which is trusted by all participants, takes the responsibility. The "*Audit Agent*", which is employed by the audit service, picks up a "*random*" value D as the secret key for the Web service as well as a large prime number p such that $p > Max(D, n)$. D is the secret key that the Web service, in order to claim any payment, must reveal. The value of n, or the number of generated tokens, can be asked from the interpolator agent, or can be estimated by the reasoning ability of the audit agent, which represents the maximum bound of clients for the Web service. To estimate a reasonable value for n, the audit agent may consider a time frame t, in which the estimation takes place. The audit agent also chooses the value of k that depends on the payment scheme. Eventually, the audit agent chooses a random polynomial in Z_p.

The Audit Agent sends out the value of p that is necessary for interpolation, to the Web service and consequently to the interpolator agent sited on the Web service.

The Audit Agent, by achieving some information from the Web service, is capable of *reasoning* about the minimum, maximum, minimum upper bound, or maximum lower bound of the number of visits to the Web service. The required information is achievable by keeping track of each Web service or asking the Web service itself to reason about the number of visits dynamically and collaborate with the audit agent in any time frame.

4.2. Generating Tokens (Shares)

Suppose that a client, or a group of clients, is/are interested in using a specific Web service. The client contacts the audit service, and consequently the audit agent to buy some tokens. The audit agent, based on some knowledge of the Web service, generates some shares, or tokens that are a function of secret key D. In other words, the audit agent divides the value of D into some pieces namely as tokens. The generation of shares is the same as generating

shares in Shamir's *(k, n)* Threshold Scheme. So the audit agent picks up a random $k - 1$ degree polynomial as:

$$q(x) = a_0 + a_1 x + ... + a_{k-1} x^{k-1} \quad (mod\ p)$$

In which $a_0 = D$ (the secret key), that is supposed to be revealed by the Web service. Furthermore, the audit agent evaluates the following *n* values:

$$D_i = q(x_i) \quad (mod\ p) \quad \text{for } 1 \le i \le n$$

The *n* values of x_i are distinct random numbers for $1 \le i \le n$. The audit agent computes *n* values of $D_i = f(x_i)$. The pairs (x_i, D_i) are the *n* secret tokens supposed to sell.

4.3. Using a Hash Function for Verifying Tokens by Audit Agent

In order to verify the validity of each token (x_i, D_i) by the interpolator agent, the audit agent uses a hash function rather than a digital signature on tokens and sends the hash values to the interpolator agent. The reason for using a hash function instead of a digital signature is that signatures schemes use public key system, which is not efficient while a hash function is known very fast. Moreover, digital signatures are useful when the receiver needs to verify the visible content of the message. In other words, to verify a signature it is necessary to see the original message. However, in our mode, the Web service is interested in the validity of the received tokens rather than contents. Thus, the validity can be proved by a hash function quickly. The interpolator agent, by receiving hash values from the audit agent, stores all the hash values in a table or file. Therefore, the interpolator agent, by achieving any token, uses the hash function on the received token and compares with the hash value received by the audit agent. The identity of both values guarantees the validity of the token and allows the Web service to offer the service to clients. The interpolate agent, by offering service to the

client, removes the hash value from the table, which corresponds to the token. In fact, removing the hash value from the table emphasizes that a token can be used only one time. Hashing a token is essential, since both the Web service and the service requester do not trust each other, though both of them trust the audit service. The hashing algorithm can be any one of well known hash functions such as MD5 or SHA-1.

4.4. Selling the Encrypted Tokens

In order to sell the generated tokens, the audit agent may use an encryption method to encrypt the tokens (x_i, D_i) and protect them through passing the Internet. The encryption takes place by using the service requesters' public keys over a secure channel such as SSL (*Secure Socket Layer*). Then only the client that has the secret key can decrypt the encrypted token. The encryption prevents any eavesdroppers to get any information about the shares. Consequently, the service requesters, in order to visit the desired Web service, may sends a token though the Internet to the Web service. Then the Web service, by using its hash values, may verify the token and let the clients use the service.

4.5. Visiting the Web service

Any service requester is allowed to use a Web service as many as it has paid for it. For instance a service requester by buying *m* tokens is allowed to use a Web service *m* times. By any visiting, the client submits a token (x_i, D_i) to the Web service, or consequently, to the "*Interpolator Agent*". The interpolator agent by using hash function on the received tokens and comparing the computed hash value with the hash value that has been received by the audit agent, allows the client to use the Web service. The interpolator agent, by collecting some data regarding the activities of clients, may decide to offer some discounts or deals.

4.6. Interpolating the Value of Secret *D* by Interpolator Agent

The interpolator agent, by comparing the hashed values on the token, allows the client to use the Web service. Using Web service may be based on some policies, globally accepted by all parties, such as: one token per session, hour, day, etc.

We argue that the interpolator agent by achieving at least *k* tokens, is able to solve the system of equations and reveal the secret key D.

From a mathematical point of view, the audit agent creates a system of a polynomial of degree *k -1* and the values of (x_i, D_i) are the second part of each equations. Since the degree of the polynomial is $k - 1$, k numbers of (x_i, D_i) are enough to solve the system of linear equations and find out the values of coefficient $a_0, a_1, ..., a_{k-1}$.

The interpolator agent by achieving at least k number of tokens, (x_i, D_i), for $1 \le i \le n$ by interpolating, can compute the values of coefficient $a_0, a_1, ..., a_{k-1}$ and eventually the value of the secret key $D = a_0$ generated by the audit agent.

4.7. Verifying the Secret Key and the Value Computed by Interpolator Agent

The interpolator agent, by computing the secret key, submits to the audit agent in order to claim for payment. Therefore, the audit agent, by comparing the original value of the secret key with the value achieved from interpolator agent, decides whether to pay the Web service. The payment is based on the proving the number of visits grater than the threshold value *k*.

5. DISCUSSION

As Shamir's threshold scheme, in case of needs to change the number of tokens or shares, the only requirement is defining a new polynomial with the same free terms [11]. The changes in the number of generated tokens, or in fact the number of visits to the Web service, can be achieved by the cooperation between the interpolator agent and the audit agent. The

interpolator agent may ask the audit agent of changing the number of offered services to service requesters. The interpolator agent may achieve this result by keeping a histogram of the Webs service's activities in any arbitrary time frame and reason about the coming time frames. The audit agent, by receiving the information of changes in the number of tokens for the coming time frame, may create a new random polynomial for the Web service and repeats the same scenario for the coming time frame.

As a very *interesting* result, both the audit agent and interpolator agent may come to an agreement in order to change the number of tokens for the current time frame. In other words, the interpolator agent by reasoning about the number of visiting clients may decide to increase the offering services for the existing time frame. Thus, the interpolator agent asks the audit agent to increase the number of generated tokens without changing any generated tokens or the secret value. The audit agent, as previous scenario, may create a new random polynomial such that not only both values for the previously generated tokens and the secret value are not changed and can be used for interpolation but also the new generated tokens are injected to the system.

The limitation of the Threshold schemes is that the Web service cannot provide the proof of exact number of visitors, but can prove that there are at least k visitors (the Threshold value). It may be considered as a shortcoming of this model. However the ability of the interpolator agent and audit agent in reasoning about choosing a suitable value for the Threshold and changing it dynamically is inevitable. We take advantages of the software agents to solve the problem of choosing a suitable value as Threshold.

There exist some secure electronic transaction protocols such as SET, which takes care of payment over the Internet. However, SET is a complicated protocol and not flexible enough to use by involved parties. Basically, in order to use SET, every party should be on-line. However, in our model, none of the defined participants are requested to be on-line all the time. This is the result of the efficiency of both Web services and software agents technologies.

6. CONCLUSION AND FUTURE WORK

The emerging Service-Oriented paradigm, as a new technology for service integration over the Internet, has opened up new view of distributed computing such that the participants in an E-business may offer their services over the Internet regardless of developed platforms. By introducing any entity as an available service, new challenges emerged. The most important issue on the use of a service is related security and financial concerns. Visiting a Web service and paying the service provider, namely as metering the Web service, request a security proven approach rather than a simple model.

Cryptographic approaches for metering a Web service namely as "Metering Schemes", based on their mathematical concepts, provide a sophisticated solution to address the measuring of a Web service. Among cryptographic approaches, secret sharing schemes are believed to provide some mathematically proven secure metering mechanisms. Threshold schemes by dividing a secure value into some secure pieces, allow the reconstruction of the original secret value by composing a specific number of divided pieces, known as threshold, and revealing the secret. However, generating the initial tokens and revealing the secret value need some further investigations. Among them, the number of generated shares and reconstructing the secret keys are significant issues.

Agent-Oriented paradigm has introduced a new technology for dealing with decision-making in distributed environments. Intelligent software agents are software components capable of reasoning and problem solving. The capability of an agent in solving a mathematical problem such as the ability to reason about the number of generated shares is the main concern of this paper.

This paper proposes an agent-based threshold payment model for metering Web services. The main idea is taken from the Shamir's simple threshold scheme. We discuss how agents are able to reason about dividing a secret value into some secret shares and reason about the number of generated shares.

As future work, we are concerned with proposing an agent-based partially payment model for Web services. The proposed model in this paper provides a payment for service provider if the Web service has been used by the specific number of visitors known as threshold. In other words, if the Web service has been visited lower than the threshold value, it is unable to prove the secret. Masucci [8] has provided some information theory based approaches to address this issue; however, our approach is to take advantages of unique properties of software agents in solving the mathematical aspects of this problem.

REFERENCES

[1] C. Blundo, A. D. Bonis, and B. Masucci: "Dynamic Multi-Threshold Metering Schemes", Selected Areas in Cryptography, pp. 130-143, 2000

[2] C. Blundo, A. D. Bonis, and B. Masucci: "Bounds and Constructions for Metering Schemes", Communications in Information and Systems, 2(1), pp. 1-28, 2002.

[3] C. Blundo and S. Cimato: "SAWM: A Tool for Secure and Authenticated Web Metering", SEKE, pp. 641-648, 2002.

[4] L. Chen and W. Mao: "An Auditable Metering Scheme for Web Advertisement Applications", ISC, pp. 475-485, 2001.

[5] M. K. Franklin and D. Malkhi: "Auditable Metering with Lightweight Security", Journal of Computer Security, 6(4), pp. 237-256, 1998.

[6] B. Masucci and D. Stinson: "Metering Schemes for General Access Structures", ESORICS, pp. 72-87, 2000.

[7] B. Masucci and D. Stinson: "Efficient Metering Schemes with Pricing", DISC, pp. 194-208, 2000.

[8] B. Masucci: "Secure Metering on the Web", PhD Thesis, http://udsab.dia.unisa.it/dottorato/TESI/tesi-masucci.pdf. 2000.

[9] M. Naor and B. Pinkas: "Secure and Efficient Metering", EUROCRYPT, pp. 576-590, 1998.

[10] W. Ogata and K. Kurosawa: "Provably Secure Metering Scheme", ASIACRYPT, pp. 388-398, 2000.

[11] A. Shamir: "How to Share a Secret", Communication of ACM 22(11), pp 612-613, 1979.

[12] http://www.openwings.org