SAWM: A Tool for Secure and Authenticated Web Metering

Carlo Blundo
Dipartimento di Informatica ed Applicazioni
Università di Salerno
84081 Baronissi (SA), Italy
carblu@dia.unisa.it

Stelvio Cimato
Dipartimento di Informatica ed Applicazioni
Università di Salerno
84081 Baronissi (SA), Italy
cimato@dia.unisa.it

ABSTRACT

The aim of a metering system is the accurate measure of the number of accesses to a Web page in order to have feedback on the effectiveness of the advertising on the net. At the present, there are no standard means to measure the exposure of Web pages as well as the impact of online advertising campaigns. Indeed "traditional" metering techniques are afflicted by hit inflation and hit shaving attacks. In this paper we propose a framework to accurately count the number of visits to a Web site relying on cryptographic primitives. In this way it is possible to avoid cheating by any of the agents in such a framework. Furthermore, a viable implementation of a tool to securely monitor a Web site is discussed.

Categories and Subject Descriptors

K.4.4 [**Computers and Society**]: Electronic Commerce—*Distributed commercial transactions, Security*; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection—*Authentication*; H.3.5 [**Information Storage and Retrieval**]: Online Information Services—*Web-based services, Commercial services*

Keywords

Secure metering, Auditing, E-Commerce

1. INTRODUCTION

New forms of business have been created by the spreading diffusion of the Internet and the WWW. Actually, the most diffused approaches for making money on the Internet are the exchange of physical goods or services and of commercial information through advertising. While e-commerce applications are being developed and promise to deeply affect the evolution of our society and economy in the next future, currently online advertising is largely diffused. Indeed the annual volume of money involved in online advertising is in the order of billions of dollars and is still growing. According to several forecasts, an important share of the whole advertising market will be conquered by online advertising.

Online advertising is ruled by the same mechanisms ruling other advertising channels, such as TV channels or newspapers. Advertisers exploit the popularity of the best known Web sites, typically search engines or portals for services, to advertise their products and to get in contact with the largest number of people. What is different is the way in which advertisers can measure the exposure of their ads, since the usual methods do not apply for the Internet. Traditional rating systems have poor value, due to the enormous number of "channels" where the advertisements are scattered, i.e., billions of Web pages against few TV channels or newspapers. Furthermore, due to the electronic nature of this medium, better accurate ways to count the visits can be developed.

Typically, Web servers have some logging mechanism to store and track the visits they receive. But they can be motivated to alter the log, inflating the number of visits they registered in order to request more money for displaying ads. At the same time, advertisers need reliable reports to decide on where to place the advertisements and how much to pay for this service. There is hence the need to develop secure mechanisms to measure accesses to Web pages and avoid any form of cheating.

Actually, advertising in the Internet comes into two forms. One resembles what happens in other advertising channels, where advertisers buy space on the pages of the most visited sites to publish their ads. Obviously sites that claim to have a large number of daily contacts can request higher advertisement fees to display the ads. In the other form, advertisers pay for each contact they receive when a client clicks on the banner. In practice, click-through payments programs have been developed, where a target site pays the referrer site for each visitor who accesses the target site passing through the referrer's pages. In the while, new forms of advertising have been created, exploiting the capabilities offered by the Internet.

In a typical scenario there is an *audit agency* offering the advertising service and whose task is to measure the interaction between a large number of servers and clients. The contract is between the audit agency and the servers, who agree on a certain amount of money the server earns for each received visit. In this context a visit can be defined as the displaying of a Web page on the client's browser or a client's click to the advertiser's site through the banner hosted by the server's site. As shown in [1], metering systems are afflicted by *hit inflation* and *hit shaving* attacks. In fact, servers are motivated to simulate a larger number of

SEKE '02 July 15-19, Ischia, Italy

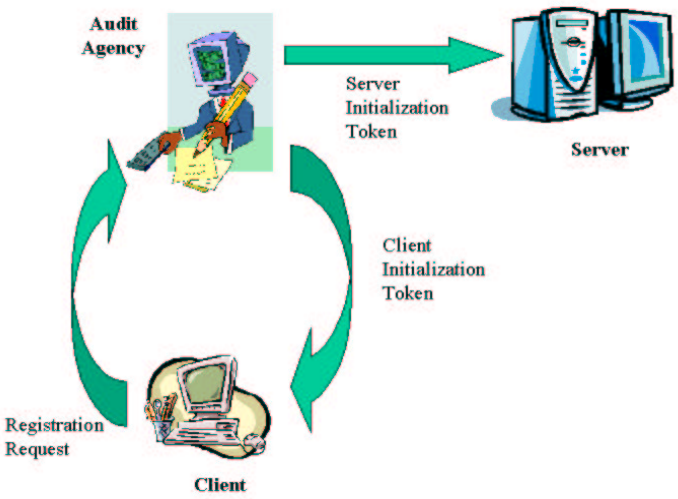

Figure 1: The Initialization Phase

visits to increase the amount of money they can claim from the audit agency (hit inflation). On the other hand, the audit agency could omit to pay the server for some number of visits (hit shaving), avoiding to pay for the visits the server claims to have received. Different settings and remedies are presented in [10].

To accurately measure the amount of visits a site receives and hence the exposure of the advertising several *metering schemes* have been produced. Franklin and Malkhi in [3] provide a theoretical approach to the metering problem, with a "lightweight" framework founded on the concept of "timing function". Naor and Pinkas [8, 9] proposed a different approach to handle coalition of corrupt servers and client, relying on secret sharing schemes. The security of click-through programs has been analyzed in [10] and in [1], where several protocols have been described to detect hit shaving and hit inflation attacks to such systems.

We propose a framework which is based on the well known idea of *hash chains* [5], previously used for user authentication and micro-payment systems [11]. Differently from other similar approaches, we minimize the overhead due to the additional communication required to implement the protocol and still provide an efficient and flexible scheme. Our framework can be used both for measuring the access to a given Web page and for counting the number of visits through a referring page.

In the next section we will overview our approach and discuss some basic requirements. In Section 3 we introduce our protocol and evaluate it in Section 4. Finally, implementation and conclusions are discussed in sections 5 and 6, respectively.

2. THE FRAMEWORK

The aim of a *metering system* is the accurate measure of the number of accesses to a Web page in order to have feedback on the effectiveness of the advertising on the net.

In a metering system, the players are the *clients* who surf the Web, the *servers* which offer services and display ads and are payed proportionally to the number of visits they receive, and an *audit agency* that counts the number of clients accessing the advertising pages. The task of the audit agency is to measure the interaction between a large number of servers and clients. Hence, the audit agency should dispose of a mechanism which ensures the validity and accuracy of usage measurements against fraud attempts by servers (Web sites) and clients (visitors).

In metering systems any server provides the audit agency with a short proof of the visits it has received. A *proof* is a value that the server can compute if and only if it has been visited by a fixed number of clients or if a client visited it a given number of times. Such a value, at fixed interval of time, is sent to the audit agency. A visit can be defined in several different ways according to the measurement context: as an example, it might be a page hit or a session lasting more than a fixed threshold of time.

Metering systems usually involve an initial interaction during which clients receive some secret information from the audit agency. Such information is used to compute a message which is sent to the visited server. After collecting these messages from different clients, each server is able to compute the proof. Clearly, such systems require clients to register with the audit agency in order to participate in the metering process. Such registration may have several advantages for clients. For example, after registration the clients may take advantage of additional services, such as receiving news on topics of interest, getting information on upcoming promotions, downloading coupons, participating in a forum, using free SMS, disposing of free disk space and mailbox, and many others. Moreover, registration does not require clients to disclose their real identity.

In our framework, the metering scheme consists of three phases: an *Initialization Phase*, an *Interaction Phase*, and a *Verification Phase*. During the Initialization Phase, depicted in Figure 1, the client contacts the Audit Agency and registers himself to gain access, for a fixed number of times, to services offered by a Web server. After the registration, the audit agency sends to the client a *Client Initialization Token*, thta will be used by the client to compute an *Authentication Token* to get access to restricted services. The audit agency sends, as well, to the Web server a *Server Ini-*

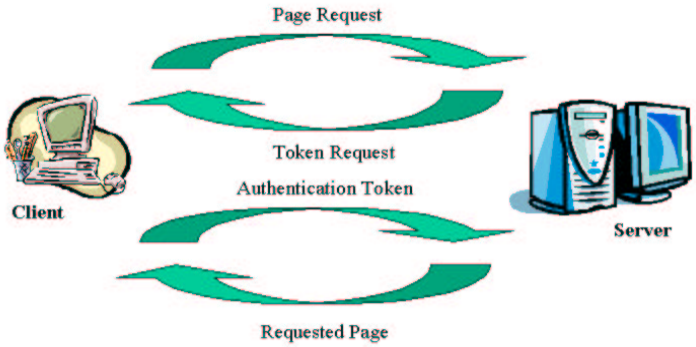

Figure 2: The Interaction Phase

tialization Token. Such a token will be used by the server to verify that the user has right to access to restricted services.

The *Interaction Phase*, depicted in Figure 2, runs similarly to "traditional" Web surfing. The client asks for a page, if such a page belongs to the restricted services provided by the Web server, then the server sends a *Token Request* message to the client asking him to authenticate himself. If the client is authorized (i.e., he registered with the audit agency), then he will be able to send an *Authentication Token* to the server. If so, the server will grant access to the client. The Authentication Token will be used by the server to prove to the audit agency how many times he has been visited by the given client. Finally, during the *Verification Phase*, depicted in Figure 3, the server contacts the audit agency in order to get paid according to the number of times he has been visited. The server sends to the audit agency the *Proof of Visits* token, which can be computed by the server if and only if it has been visited by a client a given number of times.

2.1 Requirements

A metering system is evaluated against a number of basic requirements which reflect the needs of the different actors of the framework [8].

Security. Since the aim of a metering system is the measure of the number of visits to the server's Web site, the protocol should protect both the agency and the server from malicious behaviors. In particular, the server should not be able to claim he received more visits than he effectively had and should be protected against non collaborative clients not willing to help him to count the visits made.

Non repudiation. The audit agency should not doubt on the proof that the server presents. In the case of dispute, the server should be able to show the evidence of the visits he received and claim for the payment.

Accuracy. A metering system should provide as result a number as strict as possible to the real number of the visits effectuated by the clients.

Efficiency. The computation and the storage requirements for the participants to the protocol should be as limited as possible. In particular, the clients should not be overwhelmed by a computational burden needed to access the provided service since both the augmented computational resources and the longer connection time translates into additional costs.

Privacy. The system should preserve the privacy of the clients, avoiding the possibility of tracking and retrieving too much details on the clients' behavior.

3. THE AUTHENTICATED METERING PROTOCOL

In our framework, the *metering system* consists of n clients, say $\mathcal{C}_1, \ldots, \mathcal{C}_n$, interacting with an audit agency $\mathcal{A}$ and a server $\mathcal{S}$. The audit agency wants to hold a measure of the number of times the clients visit the Web pages hosted by the server $\mathcal{S}$. The players agree on a one-way hash function $\mathcal{H}$ with the following additional properties [7]:

- *pre-image resistance:* for every output $y = \mathcal{H}(x)$ it is computationally infeasible to find any pre-image x' such that $\mathcal{H}(x') = y$;
- *2nd-pre-image resistance:* for every input x it is computationally infeasible to find another input $x' \neq x$ such that $\mathcal{H}(x') = \mathcal{H}(x)$;
- *collision resistance:* it is computationally infeasible to find any two distinct inputs x, x' such that $\mathcal{H}(x) = \mathcal{H}(x')$.

3.1 The protocol

The metering system we present here is based on the authentication of clients. Indeed, clients which are previously registered by the audit agency can access the services provided by the server $\mathcal{S}$. Our basic protocol can be easily

Figure 3: The Verification Phase

extended to many servers, by constructing different hash chains for each different server.

The operations of the basic system are structured in the following way:

Initialization The Initialization phase starts when a client $\mathcal{C}$ contacts the audit agency and requests to access the services provided by $\mathcal{S}$. A number of subsequent operations are started by each of the players:

- The Audit Agency calculates a random seed, say w, the value of the k-th application of the hashing function $\mathcal{H}$, $w^k = \mathcal{H}^k(s)$, and stores the tuple $[id_C, k, w]$ holding the Client Identifier id_C, the number of granted accesses k to $\mathcal{S}$, and the initial seed w. Then, it sends the two tuples $[id_C, k, w]$ and $[id_C, w^k]$ to the client $\mathcal{C}$ and the server $\mathcal{S}$, respectively.
- The Client stores the tuple sent by $\mathcal{A}$and retrieves the initial seed w, the number k of accesses he has registered for, and then he generates and stores the k values $\mathcal{H}^1 = \mathcal{H}(w), \mathcal{H}^2 = \mathcal{H}(\mathcal{H}(w)), \ldots, \mathcal{H}^k = \mathcal{H}(\mathcal{H}^{k-1}(w))$. Actually, the client $\mathcal{C}$ could just store the seed w and compute $\mathcal{H}^{k-j}$ when he wants to access the j-th time the services provided by $\mathcal{S}$. But, in this way the value $\mathcal{H}^{k-j}$ will be computed $k-j$ times during the "lifespan" of the metering system.
- The Server stores the tuple received from $\mathcal{A}$in its database of registered clients, associating it with a counter L_C initially set to 0.

Interaction Interaction happens when the client $\mathcal{C}$ visits the server site $\mathcal{S}$ and wants to access the restricted services offered by $\mathcal{S}$,

- The client $\mathcal{C}$ sends the token $\mathcal{H}^{k-j} = \mathcal{H}^{k-j}(w)$ for the j-th access, and updates the access counter incrementing its value;
- Upon the reception of the token, the server $\mathcal{S}$ performs the access control, by verifying that the stored value matches with $\mathcal{H}(\mathcal{H}^{k-j})$.If so, he will update its stored value with the new value $\mathcal{H}(\mathcal{H}^{k-j})$ and he will increment the client's counter L_C;

Verification The verification phase begins when the server $\mathcal{S}$claims the payment after a certain number of visits received in a certain period of time previously agreed with the agency.

- For each client $\mathcal{C}$ the server sends to $\mathcal{A}$a tuple (id_C, v_C, L_C), containing the client's identifier associated with the last authentication token v_C, and the visits counter L_C;
- The agency $\mathcal{A}$verifies that the value v_C, returned by $\mathcal{S}$for the client $\mathcal{C}$ that effectuated L_C visits, is equal to $\mathcal{H}^{k-j}(w)$.

4. EVALUATION

The system we presented requires that the clients interact with the audit agency in the initialization phase. The situation in the reality is acceptable whenever the audit agency sells a service and the agency agrees with the servers which effectively provide the service sharing the revenues according to the number of visits they receive.

Security. As regards security, the proposed scheme is robust against the common problems of hit shaving and hit inflation. The server $\mathcal{S}$cannot inflate the number of visits he receives because of the property of the function $\mathcal{H}$. Furthermore, unregistered users cannot access the restricted services since they are not able to provide a valid token. An attack to this system should be of type "man in the middle", where a fake server requests the token from the client and is able to get the services in place of $\mathcal{C}$. Even in this case only one access if lost by $\mathcal{C}$.

Non repudiation. The proof which $\mathcal{S}$sends to $\mathcal{A}$cannot be repudiated, since $\mathcal{S}$is able to prove that the last token is in the hash chain whose initial value has been provided by

$\mathcal{A}$itself. Furthermore $\mathcal{S}$should not be able to reconstruct the hash chain as far as he doesn't know the starting random seed for a given client. Only $\mathcal{A}$and $\mathcal{C}$ know the seed, but both of them are willing to protect and maintain this secret for obvious reasons.

Accuracy. As regards accuracy, the number of visits presented by $\mathcal{S}$is the real number of times that that clients access the services, since no cheating is possible from any of the players acting in the framework.

Efficiency. As regards efficiency, the system overhead on the normal communication should be minimized. In this case, clients have only an initial registration phase and no communication overload deriving from interaction with the audit agency during the regular phases of the protocol. Interaction between the clients and the servers is affected by a minimal overload: the client has to retrieve and send the token, the server $\mathcal{S}$has to verify the proof that $\mathcal{C}$sends. The operations involved are very simple and the time spent and the size of the additional message do not alter the original communication pattern. Despite of other proposed metering systems exploiting micro-payments schemes, the number of proofs that the server $\mathcal{S}$sends to the audit agency is of the order of the number of registered clients (not of the number of the visits).

Privacy. As regards privacy, only the audit agency can reconstruct the number of visits that a client $\mathcal{C}$performs. But the information is poor since $\mathcal{A}$ already knows that the client bought a certain number of access to services. When $\mathcal{S}$ claims the payment for a certain number of visits, he sends to $\mathcal{A}$only the association between the client's identifier and the stored tokens.

5. IMPLEMENTATION

In this section we will discuss a prototype implementation of the protocol presented in Section 3.1.

The most important requirements that were taken into account in the design of both the protocol and its implementation are efficiency and transparency. Indeed, since the monitoring of the visits is a problem for a large part of service providers, a valid solution should be practically implementable. For this reason the computational resources requested to execute and the additional infrastructures should be minimized. At the same time the changes to the usual client-server communication over the Internet should be contained as far as possible. A solution which consists of severe modifications to a Web site could be un-practical for server who hosts thousands of Web pages.

The implementation of the protocol has two main parts:

- computation performed on the client's computer each time the client wants to access a Web page controlled by the metering system;
- computation performed by the server who wants to control the access to its resources and has to get the proof that will be presented to the audit agency to claim for the payment.

In the following we will consider as "visit", to a Web site, the set of accesses that a client performs during a session. In our framework, a Web server can decide to give access to its resources only to registered clients. The clients are motivated to register since in this way they get access to the services provided by $\mathcal{S}$. The framework can be efficiently used also in the case where $\mathcal{S}$ requests payment for its services. In this case, clients can buy a number of accesses paying a fee to the audit agency during the registration phase. The prototype implementation has been developed for the Linux platform, using Netscape Navigator (ver. 4.76) and the Apache Web Server (ver. 1.3.19).

5.1 Client Side Computation

The need of transparency for the client side of computation lead us to consider a plug-in for the realization of the protocol. Plug-ins are modules which extend the browser's functionalities. Once that a plug-in is downloaded and installed on the client's computer, no additional operations are requested to the clients. The life cycle of a plug-in is the following:

`Registration:` whenever the browser is started, the installed plug-in is retrieved and registered, and it is associated with the supported MIME-Type;

`Activation:` The plug-in is activated whenever an HTTP request holding the particular MIME-Type which the plug-in is registered for arrives. The plug-in is loaded into the memory and an instance is created to handle the data flow in input;

`Computation:` The plug-in handles the data in input, performing some operations involving the display of areas in the browser or the exchange of data with other applications through data streams;

`Destruction:` Whenever the plug-in terminates its computation or the browser requests a new Web page, the plug-in instance is destroyed and all the allocated resources are released. If no other plug-in instance is in execution, the plug-in is unloaded from memory.

In our case, the plug-in has to handle the interaction between the browser and the server each time that a protected Web page is requested. All the messages are encapsulated in HTTP messages. To activate the plug-in, a particular MIME-Type has to be created. In our case each request to the Web server for a protected Web page will create a MIME-Type response with type "application/x-meter". Upon reception of the x-meter MIME-Type, the browser will load and execute the plug-in registered to handle that particular request type. Whenever the plug-in is activated, it creates an HTTP message which contains a "meter-request" used to send the necessary data for the authentication to the server. In particular, the message will contain the client id, the current visit number, and the value of the authentication token, as shown in Figure 4-(a).

We chose the MD5 function as one-way function producing a 128 bit long output message. Since we encode the hash value in hexadecimal notation and use 16 bytes for both the client id and the counter, we get a 64 bytes long "meter request" message. The data necessary for the construction of the request are held in a configuration file which is set up during the registration of the plug-in. The file contains the client-id, the index of the next visit, the max number of visits that the client can perform and the initial token.

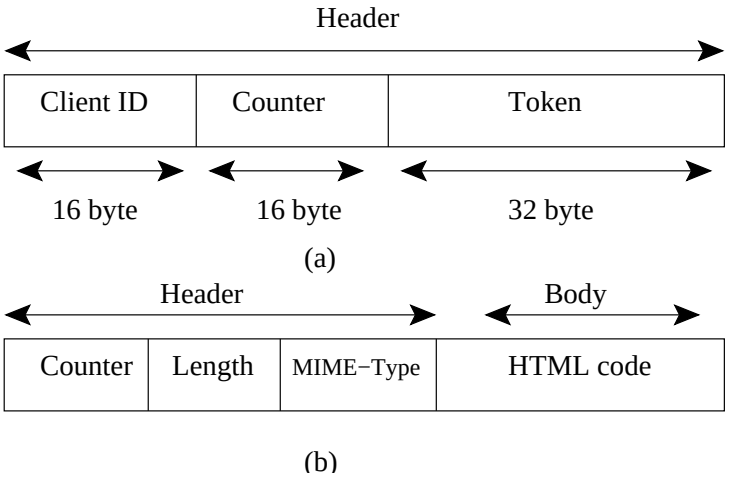

Figure 4: (a) meter request; (b) meter response

If the meter request is valid, the server allows the client to access the page. In this case he will elaborate a meter-response which contains a notification of the transaction and the document containing the HTML code. In particular, the message is composed of a counter field, holding the current visit number, the length and the MIME-type of the attached document, and at the end the document content, as shown in Figure 4-(b).

If the meter request is not valid (for instance, this could be caused by any modification of the local data file), the server will construct a valid response containing an error page informing of the impossibility to access the requested resource.

5.2 Server Side Computation

The server side of the implementation has the task to authenticate the clients and provide in the positive case the requested resources.

The main functionality of the server module is to control the data contained in the meter request, retrieving the client id contained in the request and the corresponding entry stored in its database. The verification of the access is performed by checking the validity of the authentication token. This is done by comparing its value with the one resulting from the application of the hash function to the value contained in its database.

An important requirement to satisfy for the server-side of the implementation is to avoid or reduce as far as possible the interventions on the Web pages hosted by the server. In the following we will discuss two alternative implementation and evaluate the achievements.

5.2.1 CGI Script

A first implementation of our protocol used a CGI script to perform server-side operations. The CGI script has the task to control the meter-requests and return the requested resources after the authentication. The architecture of the solution is shown in Figure 5

Whenever the browser forwards an HTTP request for a protected Web page, the Web server answers with a message which activates the plug-in. At this point the plug-in constructs the meter-request which is handled by the CGI module. The data contained in the request are stored and successively retrieved for the authentication. If the authentication is performed successfully the CGI program returns the home page of the monitored Web site.

To avoid that each time the user returns back to the home page of the site, a token is consumed by the protocol, cookies have been used to identify sessions. The meter response contains then a session cookie, which expires whenever the browser is stopped. The value of the cookie is constructed by associating the id of the client with the current token. Each time that the browser requests to access a protected resource, the cookie contained in the header of the HTTP request is checked by the CGI script.

The CGI based solution, however, poses some problems deriving form the fact that all the protected resources should be filtered through the CGI program. In this case the URL of these resources and all the referring links should be changed to avoid that unauthorized users access them by directly invoking them in an HTTP request. This in turn means that it is necessary to break the essential requirement of avoiding changes to the Web pages already hosted on the server Web site.

5.2.2 Apache Module

A better solution has been obtained by developing an Apache module. Indeed, the Apache Web Server allows the programming of external modules which can extend its basic functionalities. The modules allow the modification of the basic request loop of the server introducing handlers which are activated according to the different request type. In particular it is possible to give directives in the server configuration file, allowing all the requests for a Web page contained in a particular server directory to be redirected to a new installed handler. For our purposes, this is important since a server site which would like to adopt the proposed metering system, can reduce the necessary intervention to few modifications on the Web server.

Exploiting the PERL API of the Apache Web server, we developed a module which contains an handler responsible to respond to each request for the Web pages of the protected site. The main functionalities of the module resemble the behavior of the CGI program we described in the previous Section. It relies on a session cookie to perform users authentication after the initial exchange of the token is occurred.

Whenever the Apache Web Server receives a request for a Web page which is contained in the directory which the module has been registered for, it exits from the normal execution cycle and releases the control to the developed module.

The module answers differently according to the method type of the HTTP request. If the request is of "GET" type, the handler retrieves from the HTTP header the field containing the cookie. If the cookie is valid, the handler returns an "OK" state code, which means that the request is authorized to access the resource and that the server can continue to execute the request loop returning the resource addressed by the HTTP request.

If the cookie has expired or contains invalid data, the "FORBIDDEN" state code is returned by the handler, meaning that the user cannot access the resource. In this case the HTML code which activates the plug-in is returned giving the possibility to the user to construct a valid meter request.

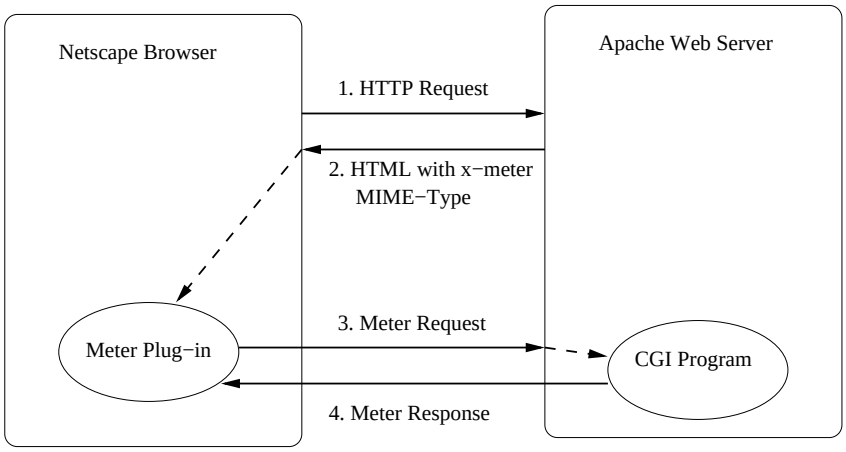

Figure 5: (a) The architecture of the CGI based solution

If the request is of "POST" type, the handler looks for a meter request in the body of the message. In this case, the response will contains in its body the resource specified in the previous message. Obviously the resource is returned after having verified the validity of the data contained into the meter request. Otherwise a response containing an error page is returned to the browser.

6. RELATED WORKS/CONCLUSIONS

As the Internet is more and more exploited as a market place, where it is possible to do business, advertising and e-commerce, numerous Web based applications are being developed. A central issue in several commercial applications, is the management of personal information. While accounting is required to access restricted services and accomplish transactions on the Web, users are exposed to significant risks when disclosing private informations on the Web. Regarding online advertising, metering systems enabling the monitoring of client activities without violating their privacy are required. Our framework begins with a registration phase which is accomplished once for all between the client and the audit agency. The centralized management of personal information enable data protection from merchants' sites as far as clients trust in the loyalty of the audit agency. Furthermore the risks due to an attacker intercepting the personal data during the transmission are reduced. As discussed in section 4, our system has little overhead on the usual communication patterns, among clients and service providers.

Different metering systems have been proposed in literature. In [8, 9] a metering system relying on a secret sharing scheme is presented. The proofs returned by the clients consist of a polynomial Q of degree $k-1$ evaluated in a particular point. Such a polynomial is provided by the audit agency during the initialization phase. The server can demonstrate he received k visits, by interpolating the polynomial Q using the k points provided by the clients during their visits. The main disadvantage of this approach is that the number of clients must be fixed at the beginning of the protocol. If new clients join the framework, then it could be necessary to reinitialize the whole system since a new polynomial must be constructed and exchanged with all the participants.

Other approaches monitor the time spent on a Web page measuring the number of complex operations that the client's browser is able to perform. Franklin and Malkhi [3] use timing functions to calculate a value which will be returned to the auditing proxy. Similarly, in [2] a system is provided which is based on the calculation of *metering evidences* which are successively verified by an auditing algorithm. The measures returned by such metering systems are not so much meaningful, since they depend on the hardware platform where the complex function is computed.

An authentication protocol widely deployed is the Passport protocol, which is the Microsoft proposal to support user authentication and sign on mechanisms. The Passport model is based on a central entity, the Passport server, which stores the clients' personal data and gives the merchants access when permitted by customers. The protocol uses encrypted cookies to store authentication information on the client side. Such information are then provided to the merchant sites which verify user credentials decrypting the data contained in the cookie. Passport protocol has been analyzed in [4] and recently some bugs afflicting the Microsoft platform have been discovered [6].

7. ACKNOWLEDGMENT

We would like to thank Rocco Scappatura for fruitful comments and discussions on the topics of this paper. This work is partially supported by National Council for Research (C.N.R.) under grant CNRRG008BF3: "Pubblicitá Online: Nuove Misure per Nuovi Media. Auditing e Accounting sicuro sul WEB".

8. REFERENCES

[1] V. Anupam, A. Mayer, K. Nissim, B. Pinkas, and M. K. Reiter. On the security of pay-per-click and other Web advertising schemes. In *8th World Wide Web Conference (WWW8)*, 1999.

[2] L. Chen and W. Mao. An auditable metering schemes for Web advertisement applications. In G. D. nad Y. Frankel, editor, *4th International Conference on Information Security (ISC 2001)*, volume 2200 of *Lecture Notes in Computer Science*, Malaga, Spain, 2001. Springer-Verlag, Berlin.

[3] M. Franklin and D. Malkhi. Auditable metering with ligthweigth security. In R. Hirschfeld, editor, *Financial Criptography (FC '97)*, volume 1318 of *Lecture Notes in Computer Science*, pages 151–160. Springer-Verlag, Berlin, 1997.

[4] D. Kormann and A. D. Rubin. Risks of the passport single signon protocol. *Computer Networks and ISDN Systems*, 33(2):51–58, 2000.

[5] L. Lamport. Password authentication with insecure communication. *Communications of the ACM*, 24(11):770–771, 1981.

[6] B. McWilliams. Stealing ms passport's wallet. `http://www.wired.com/`, 2001.

[7] A. Menezes, P. van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[8] M. Naor and B. Pinkas. Secure and efficient metering. In K. Nyberg, editor, *International Conference on the Theory and Application of Cryptographic Techniques (Eurocrypt '98)*, volume 1403 of *Lecture Notes in Computer Science*, pages 576–590, Espoo, Finland, 1998. Springer-Verlag, Berlin.

[9] M. Naor and B. Pinkas. Secure accounting and auditing on the Web. In *8th World Wide Web Conference (WWW9)*, 1999.

[10] M. K. Reiter, V. Anupam, and A. Mayer. Detecting hit shaving in click-through payment scheme. In *3th USENIX Workshop on Electronic Commerce*, pages 155–166, September 1998.

[11] R. Rivest and A. Shamir. Payword and micromint: Two simple micropayment schemes. In *International Workshop on Security Protocols*, 1996.