Secure and Efficient Metering

Moni Naor* and Benny Pinkas**

Dept. of Applied Mathematics and Computer Science
Weizmann Institute of Science
Rehovot 76100, Israel

Abstract. We consider an environment in which many servers serve an even larger number of clients (e.g. the web), and it is required to meter the interaction between servers and clients. More specifically, it is desired to count the number of clients that were served by a server. A major possible application is to measure the popularity of web pages in order to decide on advertisement fees. The metering process must be very efficient and should not require extensive usage of any new communication channels. The metering should also be secure against fraud attempts by servers which inflate the number of their clients and against clients that attempt to disrupt the metering process. We suggest several secure and efficient constructions of metering systems, based on efficient cryptographic techniques. They are also very accurate and can preserve the privacy of the clients.

1 Introduction

We propose secure and efficient metering schemes to measure the interaction between clients and servers. In a representative scenario there are many clients and servers, and an audit agency should collect information about the number of clients that were served by each server. A typical server is motivated to claim that it served many more clients than it has actually served. We describe metering schemes which are cryptographically secure and prevent servers from inflating the count of their visits. The schemes are also efficient and do not add a considerable overhead to the different parties and to the overall communication. One of their important features is that they preserve the original communication pattern and do not require the clients to communicate with the audit agency (servers need to perform a short connection with the audit agency on a relatively rare basis, e.g. once a day). The raison d'être for these schemes is to measure the popularity of web pages in order to decide on advertisement fees.

A naive implementation of a metering system could give each client a certified signature key, and require it to sign a confirmation to each visit to a server. A server can present the list of signed confirmations as a proof for its operation. This system is very accurate but not too efficient: it requires clients to perform a public key signature for each visit, and the size of a server's proof (as well as the time to verify it) are of the same order as the number of visits it had (the work of the

* Research supported by a grant from the Israel Science Foundation administered by the Israeli Academy of Sciences. E-mail: naor@wisdom.weizmann.ac.il.

** Supported by an Eshkol Fellowship from the Israeli Ministry of Science. E-mail: bennyp@wisdom.weizmann.ac.il.

audit agency is of the same order as the *total* number of visits to *all* servers). The system does not preserve privacy since the audit agency obtains lists with signed confirmations for the clients and the servers actions.

We describe several metering schemes in Section 3. These schemes have the following properties: They allow a server to prove that it served a certain number of clients. The clients do relatively little additional work (evaluate a polynomial over a small field and send its result) and are not required to change their communication pattern. The amount of work performed by the server is similar: To prove that a certain number of clients visited it the server sends to the audit agency the result of an interpolation of a polynomial over a small field (and the agency can efficiently verify it). The metering scheme is secure against fraudulent attempts by servers who try to inflate the number of clients they served and protects servers from clients who attempt to disrupt the metering process (by filtering false information provided by clients). There is also the possibility of *unlinkability*, which prevents servers from linking different visits as originating from the same client.

1.1 Main Motivation – Metering the Popularity of Web Sites

There are several approaches for making money on the Internet. Advertising is currently the main source for revenues, and the current figures estimate that advertising on the Internet will be a multi-billion dollar industry in the year 2000 [18]. In order for advertising to be effective the advertisers must have a way to measure the exposure of their ads. This measurement considerably influences the fees demanded for advertising, and is common to all forms of traditional advertising (like TV or newspapers). In those channels it is usually performed via statistical sampling of clients, or a certified audit of the popularity of the channel (e.g. of newspapers). The importance of web advertising requires a means for measuring the popularity of web services which should be *impartial* and *accurate*. It should also be *efficient*, since otherwise the parties involved (clients, servers and the audit agency) would be reluctant to adopt it.

Established measurement methods seem inappropriate for web metering. There is no current method for auditing the operation of a web site without relying on the cooperation and honesty of the site. For example many audit systems require the site to install some software which monitors its activity and sends reports to an audit agency. Since sites have an obvious motivation to exaggerate their popularity it cannot be assumed that a site will not try to break the monitoring software that it installs. Software or hardware security mechanisms which might be used to protect the monitoring module should ultimately fail if strong enough commercial interests will motivate break-ins, as was the case with many software and pay-TV protection mechanisms. Statistical sampling of clients (similar to the Nielsen rating system for TV programs) is appropriate when there is a relatively small number of media channels that the client can choose from. This is certainly not the case with the Internet which offers millions of possible web pages to visit. Therefore a statistical sampling of clients is appropriate and yields useful data only for the most popular web sites (e.g. Yahoo!) , but it is meaningless regarding the majority of the sites.

It can be claimed that most web advertising is currently displayed on a small number of top popularity web sites (such as Yahoo! or CNN). Such big and established sites might not be suspected of trying to tamper an audit module that they are asked to install to meter their activities, and their popularity can also be estimated using statistical sampling of clients. However, even today a considerable amount of advertising is displayed in small sites. It can also be argued that one of the main reasons that deter advertisers from using medium and small size sites is the lack of a secure and efficient metering ability. Such metering will provide advertisers with essential feedback which is impossible to obtain today, and might increase and elaborate the overall advertising on the net. The beauty of the web is that one can set a site of interest to $10,000$ people worldwide. This number may suffice to attract some advertisers, provided there are reliable statistics. Our schemes allow such monitoring.

There are other promising applications for secure metering schemes. Some examples are measuring the interaction between a server and a predefined targeted audience (e.g. between a site with medical information and medical doctors) or deciding on royalties payments according to usage. A very important application might be a billing mechanism for usage based accounting between data networks.

1.2 Previous Work

There are many commercial enterprises that try to sell services for measuring the activity of web sites (a partial list of these includes companies like I/PRO, Nielsen, NetCount, RelevantKnowlegde, and others). The two main methods used by these companies are sampling the activities of a group of web clients, and installing an audit module in web sites. As we have argued sampling is unsuitable for the web and is very inaccurate, and an audit module is insecure.

There are attempts to solve other problems of web metering, except the security problem. Pitkow [16] discusses ways to uniquely identify users, and compensate for the the usage of proxies and caches which provide access for many clients (or many visits) which are registered currently at the server as a single visit. Novak and Hoffman [14] overview the current practice in web measurement and argue that it is crucial to standardize the web measurement process.

Franklin and Malkhi [9] were the first to consider the metering problem in a rigorous approach. Yet their solutions only offer "lightweight security": clients can refrain from helping servers count their visits, servers can improve their count, and the variance of the measurement is relatively high. Such solutions cannot be applied if there are strong commercial interests to falsify the metering results.

Micropayments are an alternative method for financing online services. Their implementations are designed to be very efficient in order for their overhead to be less than the value of the transactions. Micropayments can be used for web metering, where each visit would require the client to send a small sum of "money" to the server, which would prove many visits by showing that it earned a large sum of money. However, all the current suggestions for micropayment schemes require the communication from the merchant (i.e. the server) to the bank (i.e. the audit agency) to be of the same order as the number of payments that the merchant received. This means that the amount of information that the audit agency receives is of the order of the total number of visits to *all* the metered servers. Luckily, we were able to construct more efficient metering schemes since in such schemes there is no need to deduct "money" for clients accounts.

1.3 Organization

The following section describes the environment in which metering schemes operate and specifies the requirements from such schemes. Several approaches for constructing metering schemes are presented in App. A. They are based on hash trees, on the pricing-via-processing paradigm and on threshold cryptography. Section 3 describes our preferred metering schemes which have better properties and are based on secret sharing. Section 4 presents some open problems.

2 Definitions

2.1 Scenario

The environment in which the metering scheme operates consists of *clients* (denoted with C) and *servers* (denoted with S), and regular operation involves interaction between them, which the metering scheme should measure. The *audit agency* A is a special party responsible for measuring the interaction. Clients and servers do not necessarily trust each other (a server might not even trust other servers), but they do trust the audit agency for the purpose of metering. Nevertheless, clients might not be willing to help servers in counting their visits unless they gain something from their help (or lose something if they do not collaborate). Alternatively, some clients might help a server to claim that it had more visits, or some servers might help each other for this purpose.

The metering system measures the number of *visits* that a server receives. A visit can be defined according to the information that is of interest, e.g. it might be a page hit or any other relevant definition (it is beyond the scope of this paper to define what should be measured). The operation of metering schemes has the following general structure:

Initialization: This step is performed once, at the introduction of the system. The audit agency A chooses a random secret key α. It then generates an initialization message for every client C and server S, which is a function of α and of the identity of the receiving party. This message is sent to its receiver through a private (secure) channel, and should be kept secret. Note that this stage does not require any interaction, only one-way messages sent by the audit agency.

Beginning of a time frame: The scheme is intended to count the number of visits in a certain time frame t. At the beginning of this period the audit agency generates for every server S a *challenge* $h_{S,t}$ which is a function of α, S and t, and sends it to the server through a secure channel.

Interaction between a client C and a server S: When C approaches S it receives from it a *challenge* which is a function of $h_{S,t}$, of the initialization message that S received, and of C's identity. The client then computes and sends a *response* which is a function of the challenge and of the initialization message that it received.

End of time frame: The audit agency might send an additional *challenge* to the server. S proves that it had a certain amount of visits by answering this challenge using the responses it received from clients during the time frame, and using its initialization message.

This is the most general form of a metering scheme. It is preferable to eliminate some of the *challenge* messages, especially those sent from servers to clients, e.g.

if these challenges can be computed by the receiving parties. This is the case in our preferred scheme.

2.2 Requirements

Security: It should be impossible for a server S to inflate the count of visits that it claims to have served. The server should be able to mathematically *prove* that it had a certain number of visits. On the other hand, a server should be protected from subversive clients who might not be willing to help it in creating the proof. For example, if the server is able to detect such clients at the time that they request service then it can refrain from serving them.

Efficiency: We define efficiency as a strict requirement of metering schemes since otherwise the large scale of the metered interaction would make the schemes useless (as is the case with using micropayment schemes for metering). It is essential for scalability that the metering system would preserve the existing *communication pattern*, and in particular would not require communication between clients and the audit agency, or require mass communication between the server and the audit agency. The computation and memory overheads should be minimal, especially for the client, who does not have a direct gain from the metering system. An additional motivation for limiting the overhead of clients is to enable them to quickly compute their answers. This allows servers to adopt a policy of not serving clients until they send the required response.

Accuracy: The results of the metering scheme should be as accurate as possible. The requirements are of the form *"if a server S shows k hits, then with probability $1-\delta$ it had at least $(1-\varepsilon)\cdot k$ hits"*, and *"if a server S had at least $(1+\varepsilon)\cdot k$ hits, then with probability $1-\delta$ it would be able to show at least k hits"*. The parameters δ and ε should be minimized.

Privacy: The metering scheme should not degrade the privacy of clients and servers, and in particular should not require servers to store the details of every visit and send these details to the audit agency. A nice feature would be to enable client anonymity in the sense that even a server would not be able to tell whether several visits were performed by the same client. We show in Section 3.6 how this feature can be implemented with our suggested schemes.

Turnover: An important feature of a metering scheme is to measure the turnover of clients, i.e. the ratio between old and new clients who visit a server. For example, it should be possible to tell whether most of the clients who visit a server during a certain day have also visited it in previous days. Metering turnover is important for advertisers, they can tell for example whether new or returning visitors see their ads. It also measures the loyalty of clients to sites. Such metering can also prevent corrupt servers or "entrepreneurs" from organizing a large group of clients and selling their services as "visitors-per-pay". Such a group might be composed of legitimate clients and therefore their visits should be counted. However, if a server relies on a single group of clients to prove that it had many visitors then it will not be able to prove a nice turnover of clients. We demonstrate in appendix B how to check turnover of clients.

3 Secret Sharing Based Schemes

A simple k-out-of-n secret sharing scheme seems to capture some of the requirements of metering. This scheme divides a secret into n shares such that no $k-1$ of them disclose any information about the secret but any k shares suffice to recover it. Consider a naive application of secret sharing in the metering scenario: the audit agency splits a secret into n shares (where n is the number of clients) and gives each client a share. When a client visits a server it gives it its share. When the server receives k different shares from k different clients it is able to reconstruct the secret and prove that it had k visits. This system suffers from serious deficiencies: (1) It is essentially "one-time" – servers and clients should get data which suffices for metering visits to many (possibly colluding) servers and during many time periods. (2) Robustness – the server should be able to identify corrupt shares. Even a single corrupt share can prevent it from recovering the secret. In some applications clients might even have a motivation to prevent the server from proving many visits. (3) The recovery of the secret must be efficient, since the number of visits can be very large (up to millions of visits).

We base our work on a modified version of the polynomial secret sharing scheme of Shamir [19]. We propose different variations of secret sharing based schemes, which achieve different security, efficiency and accuracy properties. Next we describe a basic scheme that checks whether servers received k visits in a certain time frame, where k is a predefined parameter. (It is shown in App. B that more detailed measurement can be achieved using this type of metering). In the proceeding sections we show how to add robustness to the scheme, how to increase its efficiency, how to allow anonymity for clients and how to allow unlimited use based on a computational assumption.

Following we describe schemes which check whether a server receives k visits during a certain time frame (e.g. during a day). A different approach is that whenever a server has k new visits it proves this fact to the audit agency. We describe in App. B how to apply that approach.

3.1 The Basic Scheme

The basic metering scheme uses a bivariate polynomial rather than a univariate one, in order to share many secrets which serve as proofs for the different servers (similar ideas were used by [2, 7]). The main idea of the scheme is depicted in Fig. 1. The system has three parameters k, d and p. These parameters determine the number of visits measured in a time-frame (k) and the security (d and p).

Initialization: The audit agency A chooses a random bivariate polynomial $P(x, y)$ over a finite field Z_p, which is of degree $k-1$ in x and degree $d-1$ in y. It sends to each client C the univariate polynomial $Q_C(y) = P(C, y)$, which is constructed from P by substituting the value C for the variable x. That is, Q_C is a restriction of $P(x, y)$ to the line $x = C$, and is of degree $d-1$. (The scheme will be used to meter k visits, and the parameter d defines the number of time frames in which the scheme can be securely used).

Regular operation: When client C approaches a server S in time frame t, it sends to S the value $Q_C(S \circ t)$. The input is a concatenation of S and t, and we assume for simplicity that it is in Z_p and that no two pairs $\langle S, t\rangle, \langle S', t'\rangle$ are mapped to the same element.

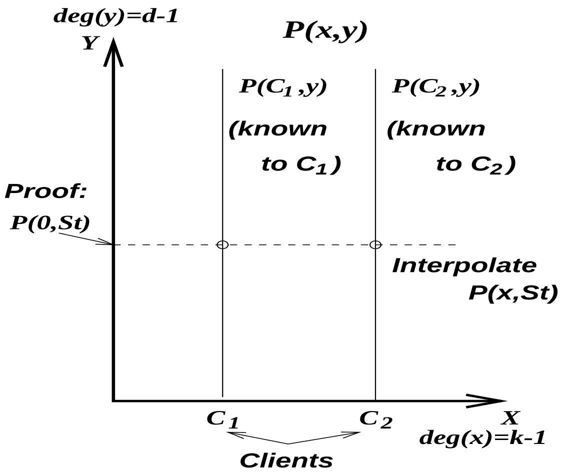

Fig. 1. The basic secret sharing based metering scheme.

Proof generation: After k clients have approached the server in time frame t it has k values, $\{P(C_i, S \circ t)\}_{i=1}^{k}$, and can perform a Lagrange interpolation and compute $P(0, S \circ t)$. This value is the proof that the server sends to the audit agency. The audit agency can verify the sent value by evaluating the polynomial P at the point $(0, S \circ t)$. (The polynomial P has kd coefficients but its evaluation at this point is efficient since the x coordinate is 0 and only d terms are non-zero.)

In the Sec. 3.2 it will be shown that the probability with which a server can generate a proof without receiving k visits is $1/p$, and the system can therefore safely use $p \approx 2^{32}$ (say $2^{31} - 1$). Alternatively the system can use $GF(2^{32})$. As the typical fields are small, the basic arithmetic operations are very efficient.

3.2 Security

For a given bivariate polynomial P the server is required to find the "proof" which is the value $P(0, y)$ at a certain point $(0, y)$. The security relies on the d-wise independence of the values of P along any line parallel to the y axis, and the k-wise independence of P's values along any line parallel to the x axis. In order to be able to evaluate P everywhere the server needs to know all the kd coefficients, whereas in order to calculate P on points on the line $x = 0$ (or $x = i$ for this matter) the server should know d values of P on this line.

A corrupt server can be assisted by other corrupt clients or servers. A corrupt client C can donate his polynomial and then the server can evaluate P at every point (C, y) and needs one less client in order to prove that it had k visits at a specific time. The information that the client donates is equivalent to d coefficients of P. A corrupt server can donate the information that it received from clients in previous time frames, which is equivalent to k coefficients per time frame. The following theorem outlines the capabilities of a coalition of α_s corrupt servers and α_c corrupt clients. Its proof is straightforward.

Theorem 1. *Consider a coalition of α_s corrupt servers and α_c corrupt clients which has been operating for α_t time frames, such that $\alpha_c < k$, $\alpha_s \alpha_t < d$ and $\alpha_c d + \alpha_s \alpha_t k - \alpha_c \alpha_s \alpha_t < dk$ (the first component of the left side of the inequality is the information known to the corrupt clients, the second component is the information known to the corrupt servers, and the third is the information which was counted twice). Let S be one of the coalition members, which received less than $k - \alpha_c$ visits in one of the time frames. Then S has a probability of at most $1/p$ in finding the proof required for this time frame.*

The polynomial P should be replaced in general at least every d time frames, and typically much earlier (against coalitions of servers). A polynomial with a higher degree d can be used for a longer time, but then the storage and computational requirements from the client are also higher. One way to tackle this problem is by using classes, as we describe in Section 3.4[1].

3.3 Robustness

Even if very few corrupt or erroneous clients send incorrect shares to a server, it cannot reconstruct the secret. McEliece and Sarwate [13] pointed out that the error correction properties of Reed-Solomon codes can be used to efficiently reconstruct the secret of a k-out-of-n secret sharing scheme if there are $k + 2t$ shares and at most t of them are corrupt. However, this might not be a sufficient protection if there are many corrupt clients.

Verifiable secret sharing (VSS) enables the recipients of shares to verify that the dealer has sent them correct shares. *Non-interactive* VSS schemes (e.g. of [6, 15]) are especially useful. In our application the dealer of the shares (i.e. the audit agency) is usually trusted, but clients might send corrupt shares. VSS can be employed to prevent that. However, known non-interactive VSS schemes use large multiplicative groups (so that extracting discrete logarithms is hard), and the server should perform about $\min(d, k)$ exponentiations to verify each share it receives from a client. This is highly inefficient compared to the basic metering scheme, and non-suitable for metering.

The following verification method is much more efficient than using VSS. It is based on the following ideas from [3, 17, 20]: Suppose that A asks C to communicate to S a value $u \in Z_p$, and wants to prevent C from sending to S any different value. To authenticate the value, A can choose random values $a, b \in Z_p$, compute $v = au + b \bmod p$, and send $\langle a, b \rangle$ to S and $\langle u, v \rangle$ to C. Later C sends to S the pair $\langle u, v \rangle$ and then S can verify that $v \equiv au + b \bmod p$. The probability that S finds u before it receives the information from C, or that C can cheat S, is at most $1/p$.

[1] Another method which reduces the power of colluding servers and does not increase the *online* run time of clients is to use polynomials of the form $P(x, y, z)$ and consequently $Q_C(y, z)$, where y is substituted with the name of the server that is serving the client, and z is substituted with the time. Then at the beginning of time frame t the client can run a preprocessing stage and substitute t for z. Since this operation can be performed off-line, the degree of z can be relatively high. During run time the client would only have to substitute the identity of the server. If the system should be immune against coalitions of α_s servers for α_t time frames, then the online run time is reduced from $O(\alpha_s \alpha_t)$ to $O(\alpha_s)$.

The following metering scheme is robust. It is depicted in Fig. 2 (together with the anonymity preserving scheme of Section 3.6). The scheme uses the following polynomials, all of them chosen at random by A over a field Z_p: $P(x,y)$, which is of degree $k-1$ in x and of degree $d-1$ in y. $A(x,y)$, of degree c_k in x and c_d in y. And $B(y)$, of degree c_d in y. The audit agency also computes the polynomial $V(x,y) = A(x,y) \cdot P(x,y) + B(y)$ in Z_p.

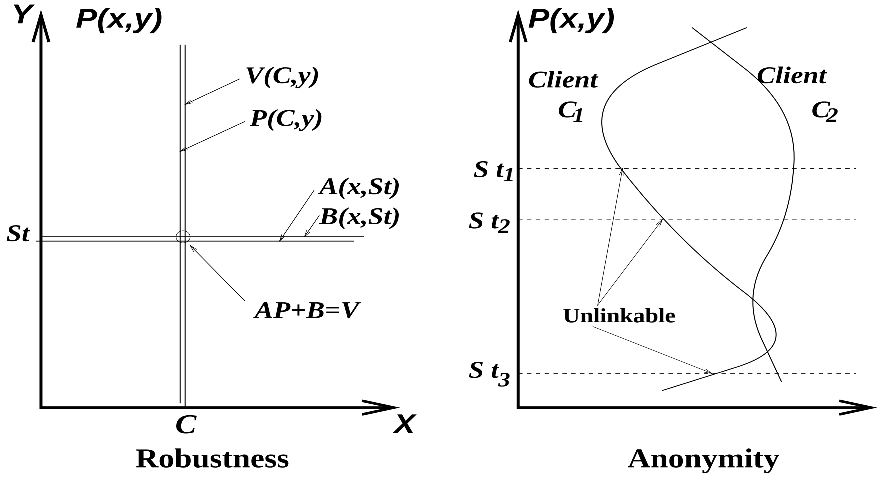

Fig. 2. The robust scheme and the anonymity preserving scheme.

Initialization: Every client C receives P and V restricted to the line $x = C$. Suppose the scheme is to be used in c_t time frames, $\{t_i\}_{i=1}^{c_t}$. Then a server S receives[2] c_t restrictions of the polynomials A and B to lines parallel to the x axis, defined by substituting $\{S \circ t_i\}_{i=1}^{c_t}$ for the value of y.

Operation: At time frame t the client C sends to S the values $P(C, S \circ t)$ and $V(C, S \circ t)$. S evaluates A and B and verifies the identity $V = AP + B$ at the point $(C, S \circ t)$. If the identity does not hold then the client is considered corrupt. As before, after receiving information from k clients the server is able to perform an interpolation and find the value $P(0, S \circ t)$.

Note that C cannot cheat S with probability better than $1/p$ without knowing the values of A and B at $\langle C, S \circ t \rangle$. The security against S finding the required value of P (with probability greater than $1/p$) is as in the non-robust scheme.

Theorem 2. *If the above scheme is used for at most c_t measurements, then a coalition of at most $c_k + 1$ clients or at most c_d/c_t servers has a probability of at most $1/p$ to succeed in sending a corrupt share to another server.*

[2] The operation of the audit agency in the initialization stage might seem to be too demanding since the polynomial V is pretty large, of degree $c_k(k-1)$ in x and degree $c_d(d-1)$ in y. However since V equals $AP+B$, the audit agency can substitute $x = C$ in A and in B (which takes $O(k + c_k)$ multiplications), and then multiply the two resulting polynomials in time $O(dc_d)$.

3.4 Increased Efficiency by Using Classes

The operation of the client and the audit agency only requires the evaluation of a d degree polynomial, and the server should interpolate[3] a polynomial of degree k. These operations are not too complex since the basic operations are performed over a small field. However, the parameters k and d are typically large and therefore it might be desirable to decrease the overhead of the parties. Following we describe how to decrease the overhead (for simplicity this is exemplified for the basic scheme of Section 3.1). A similar approach regarding micropayments, which bridges the gap between online and off-line payment systems, was suggested in [11].

The audit agency decides on a parameter k' and defines $n = k/k'$ classes by choosing n random polynomials $P_1(x, y), \ldots, P_n(x, y)$, each of degree $k' - 1$ in x and degree $d - 1$ in y. It then maps clients to classes by using a random mapping R from the set of clients to $\{1, \ldots n\}$, and giving client C the polynomial $Q_{R(C),C}(y) = P_{R(C)}(C, y)$ (the client knows to which class it is associated). Clients send to S the same messages as before, but to prove that it had k' clients from a specific class the server only needs to interpolate a k' degree polynomial.

In one possible variant of this method the audit agency should require the server to prove that it had k' clients from a specific class $r_{S,t}$ (randomly chosen by the audit agency). The proof is the value $P_{r_{S,t}}(0, S \circ t)$. An alternative option is to require the server to prove that it had k'' visits in each class (where $k'' < k'$ but $k' - k''$ is small).

The drawback of using classes is that the threshold is probabilistic, which is of course less desirable: for example, for the first variant it is possible (with low probability) that even after k clients have sent their shares the server received less than k' shares from the relevant class and does not have the required proof[4]. The waiting time for the second variant behaves according to a variant of the "coupon collector" problem.

3.5 A Scheme for Unlimited Use

The following scheme is based on secret sharing, but uses a single polynomial for a virtually unlimited number of time frames. Still, a server which receives $k - 1$ visits cannot learn (even after many time frames) the proof for k visits.

Let Z_p^* be the cyclic group modulo p, and let g be a generator of a subgroup of Z_p^* of order q, such that extracting discrete logarithms to the base g in this subgroup is hard. The audit agency chooses a random polynomial $P(x)$ of degree $k - 1$ over Z_q.

The basic protocol is depicted in Fig. 3. All values and operations are in Z_p^*.

Initialization: every client C receives from the audit agency a message which includes $P(C)$ and A's signature on $g^{P(C)}$.

---

3 Polynomial interpolation is a relatively efficient operation, the complexity of interpolation between k points is $O(k \log^2 k)$ multiplications (see e.g. [1] p. 299).

4 It follows from the Chernoff bound that the probability that after $k'n + cn$ random visits there are less than k' clients from a certain class is at most $2\exp(-\frac{1}{2}\frac{c^2}{c+k'})$. This means for example that if it is required that this probability be less than 1% then c should be approximately $\sqrt{10k'}$, and then the relative size of the "grey area" is $c/k' \approx \sqrt{10/k'}$.

C	**S**	**A**
		Beginning of time frame $S \xleftarrow{g^r} A$
C approaches S $C \xleftarrow{g^r} S$ $C \xrightarrow{g^{rP(C)},\text{proof}} S$	proof verified	
	End of time frame S calculates $g^{rP(0)}$	$S \xrightarrow{g^{rP(0)}} A$

Fig. 3. A scheme for unlimited use.

Operation: At the beginning of a time frame a server S receives from the audit agency a challenge g^r. The proof that it has to return, to show that it had k visits, is $g^{r \cdot P(0)}$.

A client C that approaches S in this time frame receives from it the challenge g^r. The client should answer with $g^{rP(C)}$, $g^{P(C)}$ signed by the audit agency, and a proof that the discrete logarithms of $g^{P(C)}$ to the base g and $g^{rP(C)}$ to the base g^r are equal. The server verifies the signature and the correctness of the proof. (We discuss the details of this proof later in this section)

Proof: At the end of the time frame a server which received k visits can perform an interpolation of $g^{rP(\cdot)}$ and find the value $g^{rP(0)}$. This is possible since a Lagrange interpolation uses additions and multiplications of elements which are known to S, and they can be replaced with multiplications and exponentiations, respectively.

The following theorem asserts that the system is as secure as the computational Diffie-Hellman assumption[5].

Theorem 3. *Consider a polynomial P of degree $k-1$ and a server S which knows a polynomial number of challenges and their answers, $\{g^{r_i}, g^{r_iP(0)}, \{g^{r_iP(C_{j_\ell})}\}_{\ell=1}^{k}\}$, and which received a new challenge g^r and less than k answers of the form $g^{rP(C_j)}$. If S can compute $g^{rP(0)}$ it can also break the computational Diffie-Hellman assumption: given g, g^a, g^b it can compute g^{ab}.*

The main advantage of this scheme is that the same polynomial can be used for an unlimited number of time frames. However, the complexity of the basic operations is higher since both the client and the server should perform exponentiations. Classes can be used with this method too, and reduce the degree of the polynomial that is used.

Robustness: The proof that the client performs to convince the server in the authenticity of its share can be either interactive or non-interactive (and then the security is heuristic), but typically such proofs are complex and require relatively lengthy computations. They can be replaced with the same technique that was used

[5] The proof is rather straightforward and appears in the full version of the paper. Note however that the reduction assumes knowledge of the identities of the (less than k) clients from which the server obtains the values $g^{rP(C)}$.

in section 3.3 to achieve robustness, which is very efficient. The main difference is that now the server verifies that the *exponents* of the two sides of the equation are equal. In short, a client C receives the polynomials $P(C,y)$ and $V(C,y)$, whereas each server S receives the polynomial $A(x,S)$ and the value $B(S)$. For a query $h = g^r$ the client sends to the server the values $\langle h^{P(C,S)}, h^{V(C,S)} \rangle$. The server verifies this answer by checking that $h^{V(C,S)} \equiv (h^{P(C,S)})^{A(C,S)} \cdot h^{B(S)} \bmod p$.

3.6 Anonymity

Anonymity is desired by many clients. An even stronger property is *unlinkability*, which prevents servers from linking different visits as originating from the same client. At first it seems that secret sharing based metering schemes do not support this property since a client C always sends values of P at points in which $x = C$. Following we describe how to achieve unlinkability of different visits by the same client (exemplified for the basic system).

The anonymity preserving scheme is depicted in figure 2, and is as follows:

Initialization: As before the audit agency generates a random polynomial P over the field that is used. It also generates for every client C a random polynomial $Q_C(y)$ of degree u. Consider the polynomial $P(Q_C(y), y)$, which is of degree $d - 1 + u(k-1)$. It is a restriction of P to the curve defined by $x = Q_C(y)$. The audit agency sends to C the coefficients which enable it to calculate values of $P(Q_C(y), y)$.

Operation: When the client C visits a server S at time t it sends it the values $\langle Q_C(h), P(Q_C(h), h) \rangle$, where $h = S \circ t$. After receiving k such values the server can interpolate the polynomial $P(x,h)$ and calculate the proof $P(0,h)$.

The information that a client sends in $u+1$ visits is unlinkable since any $u+1$ points can be fit to a curve of degree u. Therefore examining this information does not reveal whether these visits were from the same client[6] . Furthermore, consider a server which received k visits in each of the first $u+1$ time frames, and in time frame $u+2$ receives a visit from a client who made one visit in every previous time frame. How can the server check which are the previous $u+1$ visits of this client? Each visit is hidden among the k visits of its time frame. An obvious algorithm requires $O(k^u)$ operations, and therefore might not be practical. For some choice of parameters this problem might not be easy, to say the least.

4 Open Problems

There are two main properties that can be improved in our metering schemes. First, the more efficient schemes can be used for only a limited number of measurements (though by applying classes this limit can be quite large), and the schemes that can be used for an unlimited number of measurements require public key operations and are less efficient. It is therefore important to design private key based systems

[6] Note that a corrupt audit agency cooperating with the servers can find out the activity of a client. A possible way around that is for the client to choose its polynomial $Q_C(y)$ itself and conduct the initialization process via an (inefficient) secure function evaluation [21, 10]. Improving the efficiency of such a protocol is an interesting open problem.

that can be used more than a linear number of times. Second, in the schemes of Section 3 we had to preset a certain number k and for each time frame and the server proves (at least) k visits. This is acceptable whenever there is a long-term relationship between the audit agency and the server. However, for other settings it would be preferable to have a totally dynamic metering scheme, that can measure any number of visits in any granularity.

A somewhat related problem is the problem of *licensing*. Here some software or content is bought by a client under a license which limits the maximum allowed usage. For example no more than four copies of the software are allowed to be run simultaneously, or the total number of times in which the software or content can be used is limited. It would be very interesting to design a cryptographic scheme that can enforce such a policy.

Acknowledgments: We thank Omer Reingold for suggesting the method of section 3.5.

References

1. Aho A., Hopcroft J. and Ullman J., *The design and analysis of computer algorithms*, Addison-Wesley, 1974.
2. Ben-Or M., Goldwasser S. and Wigderson A., Completeness theorems for noncryptographic fault tolerant distributed computation, *20th STOC*, 1988, 1-9.
3. Carter L. and Wegman M., Universal hash functions, *J. of Computer and System Sciences*, Vol. 18, 1979, 143–154.
4. Desmedt Y. and Frankel Y., Threshold cryptosystems, *Crypto '89*, LNCS 435, 1990, 307–315.
5. Dwork C. and Naor M., Pricing via Processing or Combating Junk Mail, *Crypto '92*, LNCS 576, 1992, 114–128.
6. Feldman P., A practical scheme for non-interactive verifiable secret sharing, *28th FOCS*, 1987, 427–437.
7. Feldman P. and Micali S., An Optimal Probabilistic Protocol for Synchronous Byzantine Agreement, *SIAM J. on Comp.*, Vol. 26, No. 4, 1997, 873–933.
8. Frankel Y., Gemmell P., MacKenzie P. D. and Yung M., Optimal-resilience proactive public-key cryptosystems, *38th FOCS*, 1997, 384–393.
9. Franklin M. K. and Malkhi D., Auditable metering with lightweight security, *Financial Cryptography '97*, 1997.
10. J. Kilian, Founding Cryptography on Oblivious Transfer, *20th STOC*, 1988, 20–31.
11. Jarecki S. and Odlyzko A., An efficient micropayment system based on probabilistic polling, *Financial Cryptography '97*, 1997.
12. Merkle R., A certified digital signature, *Crypto '89*, LNCS 435, 1990, 218–238.
13. McEliece R. J. and Sarwate D. V., On sharing secrets and Reed-Solomon Codes, *Comm. ACM*, Vol. 24, No. 9, 1981, 583–584.
14. Novak T. and Hoffman D., New metrics for web media: toward the development of web measurement standards, Sep. 1996. Manuscript available at `http://www2000.ogsm.vanderbilt.edu/novak/web.standards/webstand.html`
15. Pedersen T. P., Non-interactive and information-theoretic secure verifiable secret sharing, *Crypto '91*, LNCS 576, 1991, 129–140.
16. Pitkow J., In search of reliable usage data on the WWW, *Sixth International WWW Conf.*, Apr. 1997.
17. Rabin T. and Ben-Or M., Verifiable secret sharing and multiparty protocols with honest majority, *21st STOC*, 1989, 73–85.

18. Kinsman M., *Web advertising 1997: market analysis and forecast*, Cowles/Simba Information, Stamford, Connecticut. May 1997.
19. Shamir A., How to share a secret, *Comm. ACM* Vol. 22, No. 11, 1979, 612–613.
20. Wegman M. and Carter L., New hash functions and their use in authentication and set equality, *J. of Computer and System Sciences*, vol. 20, 1981, 265–279.
21. Yao A. C., How to generate and exchange secretes, *27th FOCS*, 1986, 162-167.

Appendices

A Approaches for Designing Metering Schemes

In addition to secret sharing, there are several other directions that seem helpful for designing efficient and secure metering schemes.

Hash trees: In this solution each client signs a confirmation for its visit. The server arranges these confirmations in a hash tree [12] and sends its root to the audit agency, which later verifies the values of random leaves. Additional care should be taken to prevent the server from storing the same value at different leaves (e.g. by using families of perfect hash functions, or by requiring the server to sort the leaves).

Pricing via processing: This approach is similar to the suggestion of Dwork and Naor [5] for combating junk email. The server is given a large computational task by the audit agency. It should ask each client to perform a small part of this task, whose final completion proves the visit of k clients. Special care should be taken to prevent the server from performing the task by itself, to prevent clients from sending incorrect results, and to minimize the variance of the stopping time.

Threshold computation of a function (e.g. threshold computation of the RSA function): In order to compute a function f each client C receives a share f_C, and $f(x)$ can only be computed by a party which gets k of the clients to compute their partial functions $f_C(x)$ and send her the results. The notion of a threshold computation of a function was introduced in [4], and the most recent implementation of threshold RSA is suggested in [8]. However known implementations were not designed for large values of n and k, and are far too inefficient in terms of computation and communication to be applicable for metering.

B Variants

THE METERING PERIOD: For the simplicity of the exposition we demonstrated in the paper metering schemes which check whether a server had k visits in a certain time frame, e.g. during a day. A different approach is that whenever the server has k visits, it proves this to the audit agency (e.g. a popular server might send such proofs several times a day, whereas a less popular server might do so every few days). In such schemes, the proof for k visits cannot be the value $P(0, S \circ t)$, where t is the date. Rather, for every proof the audit agency should provide the server with a new challenge h, and the server should then ask clients to send it values $P(C, h)$ and supply the proof $P(0, h)$.

Corrupt servers might try to send to clients false challenges h' in order to obtain values $P(C, h')$ they are not entitled to receive. (This can be done in order to receive several values from a client which has several visits in the duration of a single challenge, or to obtain values that might assist another server in computing its proof). A simple solution to this problem is that challenges h start with the identity of the server and are always even numbers. Then a server which should answer the challenge h receives the polynomial $P(\cdot, h+1)$ by the audit agency. The server should send to client C the challenge h and the value $P(C, h+1)$ as a proof for the validity of the challenge.

CHECKING TURNOVER OF CLIENTS: If a server knows $k' \leq k$ shares they enable it to wait for just $k - k'$ clients before it can provide the proof for being visited by k clients. It is possible to detect a server which operates in this manner by a system which estimates the intersection of the groups of clients that contributed to different proofs. Advertisers might have additional motivations for checking the turnover of clients.

The following simple method can estimate the turnover of a server's clients. It is demonstrated for the system proposed in section 3.5. Assume that in each time frame the server proves that it had k clients, after receiving from clients values in the form $g^{rP(C)}$. The audit agency should choose a random challenge t from a domain of size ck (e.g. $c = 10$), and a "cryptographic" hash function h whose range is of size ck. It should send to the server the challenge t and ask it to send back as soon as possible a value $g^{r_i P(C)}$ which it received from a client C, such that $h(g^{r_i P(C)}) = t$. The parameter g^{r_i} should be one of the succeeding challenges that the audit agency sends to the server. The server is expected to find a suitable answer after receiving ck new clients (about c time frames if the clients keep changing). In contrast, a server which receives $k'' \ll k$ new clients per time frame (even if it had acquired the complete shares of k corrupt clients), is not expected to answer the challenge in t time frames[7]. Although the number of values that are needed to hit the target has a relatively large variance, this scheme is useful to estimate the turnover of clients.

ADAPTABILITY: The secret sharing based metering schemes we proposed check whether a server received k clients, where k is a predefined quota. It is of course preferable to have a more flexible measurement unit which enables to count the exact number of visits that a server received. A more fine grained system can be achieved by setting the quota k to be smaller (e.g. $k = 1000$ for measuring web advertising).

A server which received almost k visits cannot provide the required proof and appears to be in the same situation as a server who received very few visits. However, if a server received $k' < k$ visits and $k - k'$ is small it can inform the audit agency of this situation and ask to receive k' values of the polynomial that it has to interpolate. After receiving these values the server should be able to perform the interpolation and compute the required proof.

[7] The system as it was described here degrades the privacy of clients and servers since it allows the audit agency to verify whether a certain client visits a certain server: to check whether client C visits S it should set the target to $t = h(g^{r_i P(C)})$ and then if C is the first client that visits S and is mapped to t then S sends $g^{r_i P(C)}$ to A and A learns about this visit. Furthermore, A can send this challenge to all the servers and trace many of C's visits in a certain time frame. To eliminate this option the target t should be defined in the following way: the audit agency uses a publicly known universal one-way hash function h. The target will be $h(x)$, but A will also send to the servers the value of x . The reply that the servers should return is a value $g^{r_i P(C)}$ that is mapped by h to the target, but subject to an additional requirement that $g^{rP(C)} \neq x$.