Auditable Metering with Lightweight Security

Matthew K. Franklin and Dahlia Malkhi

AT&T Labs – Research, Murray Hill, New Jersey, USA
{franklin,dalia}@research.att.com

1 Introduction

In the physical realm, the new U.S. $100 bill is an example of high-grade security. Forgery of even a single bill seems to require a huge expenditure to get the necessary printing plates, paper, ink, and manpower. By contrast, it is much easier to forge a coin, especially if it only needs to be good enough to fool a vending machine. This is a good security strategy, since it tailors the security of a product to be commensurate with the risks involved. Lightweight security can be adequate for small risks, especially when (as with coins versus bills) lightweight security is cheaper and easier to provide.

In the virtual realm, an RSA digital signature is an example of high-grade security. Forgery of even a single signature, without knowing the private key, seems to require a huge computational effort. This makes digital signatures a useful tool for a wide variety of security-critical applications. However, as with $100 bills, digital signature systems can be complicated to deploy, and expensive to maintain. For applications where the risks are not so great, it is reasonable to look for alternative security measures that are easier to deploy, even if they don't provide high-grade security. A lightweight security mechanism in the virtual realm, where each individual forgery requires a modest but non-trivial computational effort, can be adequate for preventing large-scale fraud in practical applications. A number of "micro-payment" schemes have been designed in this manner (beginning with [7]).

Note also that lightweight and heavyweight security mechanisms can be combined. In the physical world, one's valuables might be protected by a cheap lock on the front door, and a burglar alarm, and recorded serial numbers, and the threat of arrest. A similar blend of security mechanisms can discourage attacks in the virtual realm.

In this work we suggest a new mechanism for providing lightweight security: the compact metering scheme. We also suggest a basic application for its use: metering the popularity of web sites. This application may be unaffected by fraud on a small-scale, since only a crude and relative measure of site popularity may be needed for purposes such as the pricing of advertisements and payment of royalties. Prior to our work, an application such as web site metering could only be protected through the use of heavyweight security mechanisms (such as digital signatures), or by using on-line trusted authorities. Our lightweight security mechanism can be used on its own, or combined with other security mechanisms to increase resistance to fraud.

1.1 Uses of Auditable Metering within the WWW

The growing popularity of the Internet and the World Wide Web (WWW) is driving various applications, several of which are commercially oriented. One such commercial application of the WWW is advertising. The difficulty of using the WWW for advertising, however, is the need for securely metering the distribution circulation to accurately price the advertisements.

The absence of secure metering greatly impacts advertising utilization. Access data is usually collected at the server site, which has control over the collecting process as well as over stored data. Since the owner of the server can charge higher rates for advertisements by showing a higher number of visits, the owner has a strong economic incentive to inflate the number of visits. The owner could accomplish this by manipulating any unsecured metered data stored on the server.

Alternatively, any individual could fraudulently increase the number of visits to a web site using a "robot" program, which is configured to generate visits to a web site. The amount of visits is theoretically limitless.

In view of the above, it is clear that a need exists for accurately and securely metering visits to a web site. Metering visits to web sites is also necessary for commercial applications other than advertising. For example, customers may choose to connect to the WWW through an Internet Service Provider (ISP). ISPs typically charge their customers for connections to the WWW by the hour. An ISP may enhance service to customers by having sites referred to as *partners* that provide information on the WWW and assume the charge of clients connections to such sites. This would be similar to companies which assume the phone charges of calls to "800 numbers" in the U.S. A secure metering scheme would enable ISPs to record the number and duration of client visits that pass through the ISP to the partner site, and use this record to reverse-charge the partner site.

Although our scheme is presented in the context of metering the WWW, it can be used in any client-server setting that is amenable to automatic metering.

1.2 Technical Approach

Metering security in the WWW framework could be achieved by several known techniques. Employing standard cryptographic methods to keep self-authenticating records of interactions on the WWW would be secure, and is enabled by some existing extensions to the WWW protocols, such as S-HTTP [10] and SSL [8]. These methods are clearly advantageous in offering a high level of auditability and non-repudiability. However, authenticating all clients has the drawback that all clients must register to obtain authentication keys. Not only is this a heavy administrative burden, but it leads to solutions that threaten the clients' privacy.

A third party census may be used to independently provide measurements on web activity, as offered by the Audit Bureau of Circulations (ABC). Using this scheme, activity can be monitored by an objective authority, which can then certify the measured data and prevent the possibility of manipulating it by a

web publisher. The obvious problems with this method are the dependence on a central authority, and the deviation of census data from real activity.

We offer an alternative approach to metering, that does not rely on client authentication or on a third party. We start with a notion of a *timing scheme*, with the following properties:

1. Timing schemes can be computed with increasingly large efforts invested (incremental).
2. The output of a timing scheme need not grow with the amount of effort spent (compact).
3. The effort spent can be efficiently verified from the output with high degree of confidence, where by efficiently we mean with considerably less effort than re-computing the timing function itself (auditable).

We use the difficulty of computing a timing scheme to leverage the security of a metering method by involving each client in computing the timing function (for some given input) upon visiting a web site, and recording the result of the computation along with the record of the visit. Thus, to forge client visits requires a known investment of computational resources, which grows proportionally to the amount of fraud, and is infeasible for visit counts commonly found in the WWW.

The incremental nature of the timing function is used to create a new measure of client accesses, namely their *duration*. The term "visit" as used herein refers to the elapsed time a client examines content from a particular web site. Note that, as a timing function will be executed at the client's site, visit duration is indeed well defined despite that the WWW visiting protocol (HTTP) is stateless. A visit duration within the WWW framework is more accurately defined in Section 3. Our metering scheme engages the client in computing the timing function incrementally throughout the duration of its visit, and thus, the output captures the duration of the visit.

Dwork and Naor defined a related notion of a *pricing function* [6], whose computation requires a known lower-bound investment of resources. They use pricing functions for preventing the mass utilization of electronic mail. Pricing functions are not incremental, and cannot be utilized (compactly) to measure the duration of visits. Cai et al. define the related notion of *uncheatable benchmark* [4, 1] for efficiently verifying claims of computational power. Some of the methods proposed for uncheatable benchmarking can be used incrementally. The methods used both in pricing functions and uncheatable benchmarks contain a trapdoor that allows efficient computation of the results. If used in the context of auditable metering, a trapdoor could be used to allow efficient and definite verification of the results by an auditor, but would also provide the auditor with the means to forge results en masse. The scheme we offer below has means for efficient probabilistic auditing (verification) that does not suffer from this drawback.

1.3 Organization of Paper

The rest of this paper is organized as follows. In Section 2, we give models, definitions, and constructions of timing schemes. Our auditable metering protocols are described in Section 3. Implementation issues are discussed in Section 4, and we discuss possible extensions in section 5. Applications are given in Section 6, and some caveats are raised in Section 7.

2 Timing Schemes

A timing scheme consists of two components: The first part is a timing function, which is an incremental computation performed by a client and whose result is sent back to the server and logged there. The second part is an efficient auditing function that verifies the computation time spent producing the logged result. Computation time is expressed in terms of some agreed upon unit of computation, whose complexity is well-analyzed.

At a high level, a timing function is a computation that requires a known amount of time to compute certain outputs. This function can be computed with a reasonable investment of time by any client, but this investment of time is the only known way of producing the result. The basis of such a computation is some grain of computation h whose computational complexity is understood. In practice we can take h to be a well-known hash function such as MD-5 or SHA (producing 128 bit hash values). The timing function combines multiple applications of h to produce an output, such that the number of times that h computes represents the known required effort. Let $f_k(x)$ denote a timing function requiring k applications of h. We do not need to a-priori set k: The scheme is *incremental*, i.e., there is a way to move from $f_k(x)$ to $f_{k+1}(x)$ by applying h once. An important property of the scheme is that the output of the computation is of constant size, regardless of the number of h applications, and thus not all h results can be sent. We call this property *compact*.

For example, a timing scheme can be based on a simple min construction as follows. Let h be a hash function whose output is uniformly distributed over the domain $[0 \,..\, 2^{128} - 1]$ (e.g., take h to be MD-5). Evaluate h at $x_1, \ldots, x_k$, where $x_i = h(x_{i-1})$, $x_0 = x.r$ (concatenation) for a random seed r, and return the $(x_i, h(x_i))$ pair for which $h(x_i)$ is minimized, along with the number k of h evaluations performed. Note that this function is incremental (extending the evaluation of f_k requires one additional evaluation of h) and compact (since one pair of values is returned no matter how large k is[1]).

The second part of a timing scheme is an efficient auditing function, that verifies the number k of claimed h applications probabilistically. In the example above, a good way to test an output record is to observe that for any $k' < k$, the probability $p_{k'}$ of randomly sampling a value less than y within the range

[1] We ignore the size of k itself, which can be represented by 128 bits for all practical values of k.

$[0 .. N-1]$ using k' samples (in our case, we take $N = 2^{128}$), is bounded by

$$\forall k' \leq k : p_{k'} \leq 1 - \left(\frac{N-y}{N}\right)^k .$$

Therefore, for any desired probability level $\epsilon > 0$, $p_k \leq \epsilon$ when $k \leq ln(1-\epsilon)/ln(\frac{N-y}{N})$, and thus any k within this range will be accepted by the auditor for this required level of guarantee. It should also be possible, with a more sophisticated test, to accept or reject a collection of timing records simultaneously.

Another method for auditing the output of a timing function is by estimating a most likely number k' of h applications producing this output. For the construction above, the maximum likelihood estimator is $k' = -1/ln(\frac{N-y}{N})$.

We note that a foolproof method of verifying the number of h applications for a particular output is possible, simply be repeating the entire computation. Due to its cost, this method can be applied selectively, e.g., for those records for which the efficient verification method fails, or randomly.

3 Auditable Metering with a Timing Scheme

The metering protocol involves two parties, a *client* and a *meter*. We assume that clients and meters engage in a external *visiting* protocol by which clients access meters for certain durations. Our assumptions about the (opaque) visiting protocol borrow from the HTTP protocol [2], which motivated our work. However, other protocols may suit our abstract representation as well.

3.1 Assumptions about Meters and Clients

Let m denote a meter and c a client. m and c communicate via a reliable FIFO communication channel. We assume that m and c engage in a *visiting* protocol (whose details are opaque), that generates two events:

1. A *start-visit* at m signals the beginning of the visiting protocol at m, and precedes any information sent from m to c.
2. An *end-visit* event at c signals the termination of the visiting protocol at c. Following the end-visit event, c does not send any further information to m within the visiting protocol.

Note that an end-visit event occurs at the client, hence this formulation can be realized in the (stateless) HTTP protocol. We assume that the environment supports dynamic deployment of programs from m to c, with the following properties: m is capable of sending an executable program p to c, which c may then choose to execute. c may send a termination signal to p as it executes, or can terminate it (ungracefully). Programs deployed from m to c may utilize computational resources and may communicate back and forth with m. We reiterate that these assumptions are practical, and are currently achievable in the WWW framework.

3.2 Metering Protocol

The purpose of the metering protocol is to maintain at m a record of c's visit, expressing the duration of the visit, *i.e.*, the time between the *start-visit* and the *end-visit* events. For a meter m to record the duration of a client c's visit with a timing scheme, they engage in the protocol given in Figure 1.

---

1. **Initiation:** When m incurs a *start-visit* event by c, m deploys a program p that, when executed, computes f during its runtime and, when signaled to stop, sends back the result to m and terminates. p (equivalently, f) is initiated at m prior to deployment with the following cookie:

   $$<m, ts>,$$

   where ts is a unique identifier (timestamp) used by m, and with the property that $ts_2 \neq ts_1$ for any two distinct cookies $<m, ts_1>$, $<m, ts_2>$, .
   The program p may contain a bounded computation of f or run endlessly; In the former case, if termination is reached then the computed result is sent back to m. In the latter case, the program terminates by a stop signal (see below), or when destroyed.
2. **Execution:** When a client c receives p it starts executing it.
3. **End:** When a client c incurs an *end-visit* event it signals p to stop. If p is still running it responds to this signal by terminating the computation of f and sending the computed result back to m.

Fig. 1. The metering protocol

3.3 Auditing Procedure

A visit record has the form $\text{rec} = [m, ts, r, z]$ where $z = f(m.ts)$ and r is a unique random seed chosen by the client. An auditor tests each record and accepts or rejects it according to the desired threshold of acceptance. Optionally, an auditor may selectively re-compute the timing function of certain records for guaranteed assurance, e.g., for those records failing the test.

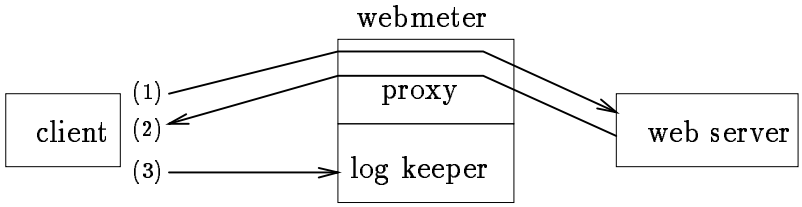
The following technique can reduce the number of visit records that need to be stored at a meter site. Only keep those visit records which satisfy some predicate P. The predicate should have a predictable success rate, but be hard to predict for individual inputs, e.g., $P_{h,q}(\text{rec})$ is true if and only if $h(\text{rec}) \bmod q = 0$. The auditor verifies that all stored timing records satisfy the predicate, and expands the timing interpretation according to the success rate (e.g., multiply by q for $P \equiv P_{h,q}$).

4 Implementation

We have implemented *webmeter*, a prototype metering tool that can monitor a web site. Figure 2 depicts the structure of webmeter. The tool consists of two modules:

proxy: A proxy module intercepts HTTP traffic to and from the designated web-site, and appends a metering *applet* to the body of each content leaving the web-site. The applet code is written in Java, and is deployed to the client after the applet is sent, as part of the standard HTTP protocol. The proxy module may be placed on the same host as the web server and *hide* the real server from the world, or it can be placed elsewhere on the communication path between clients and the server.

log keeper: A log keeper accepts the results sent by metering applets and keeps them on stable storage.



(1) *webmeter* "hides" web server. (2) webmeter appends a *metering applet* to contents. (3) metering applet at client sends result to *log keeper*.

Fig. 2. The *webmeter* metering tool

5 Discussion

The metering protocol uses the difficulty to compute visit records to leverage auditability. Our timing scheme requires an investment of Thus, the amount of possible fraud is proportional to the amount of computational resources invested in it.

In addition to verifying the correctness of the timing scheme values in a meter, we suggest strengthening the auditing of our metering data by several accompanying mechanisms.

Network Authentication: For a visit record to be accepted by an auditor, the meter m must be involved in the generation of the cookie timestamp

ts. This hand-shake protocol prevents a batch of visit records being forged off-line, and implies that visit records represent real visits made at the meter. A meter can verify the origin of a visit at the network level to some extent. One method of achieving a high degree of assurance is to use cryptographically secure authentication codes, as supported in various extensions to the WWW protocols (*e.g.*, S-HTTP [10], SSL [8]), to authenticate origins. Another, more manageable way, is to position the meter on the gateway entering an organization's network and filter out visits generated from within the network. Most known routing tricks can be prevented by a meter thus positioned (see [5]). These types of network verifications prevent some small number of clients from masquerading as a larger and/or broader set of clients. Thus almost all visit records in the log file will represent visits by external clients.

Checkpoints: An auditor can use the timestamped cookies (used for initializing the timing function f) to limit the *rate* of possible fraud by preventing the mass-generation of fraudulent visit records off-line. This can be done in the following way: An auditor designates epochs at which it requests a checkpoint of the meter log, and re-initializes the timestamp value for the next epoch. Any cookies produced during an epoch must be utilized within this epoch, and later become unusable. Thus, for example, computations performed during off-peak hours cannot produce fraudulent visit records for peak hours.

Census: An independent third party may perform a census of web activity concerning any particular web site, given any means at its disposal. The records held by the meter should agree with the census approximations.

Third party: On-line monitoring by a third party can be required selectively for some pre-determined portion of the visits. This could be done by transparently deferring designated requests to the third party site. Selection of visits should be done according to a deterministic predicate with a known success rate, applied on the cookies. In this way, the expected number of visits in which the third party is involved would be known, approximately, as a portion of the total number of logged visits at the web server.

6 Applications

Many applications may benefit from auditability of web metering. We list some possibilities here.

The primary application motivating our work is the metering of a web site to measure its popularity. Our metering scheme has the advantages of being auditable, as well as accounting every visit to pages containing contents from the web site, even in the case of repeated visits to cache proxy servers.

The incremental timing scheme is novel in providing a measure of the duration of client accesses to a server. A profile of the times spent with a particular content can be of enormous value to content service providers, as well as to advertisers seeking maximum exposure. Information on the time spent viewing the

contents of a document from a web site is valuable for pricing advertisements accompanying the document.

We also propose a novel application called **1-800-HTTP**: An Internet Service Provider (ISP) used for connecting private clients to the WWW charges clients for the duration of usage (typically, after some initial quota of flat-rate usage). This charge may prevent potential customers from browsing sites freely. An ISP may enhance its service to clients by partnering with certain commercial sites and *reverse the charges* of connection time spent by clients visiting the partner sites. The motivation to such a paradigm is similar to that of companies offering '800' service phone numbers in the U.S., that reverse the phone charges of customer calls. The scheme we suggest allows metering such visits and reversing the charges for them.

In all of the examples above, the auditable metering scheme was used to prevent (mass) forgery by the meter. Forging mass (and diverse) visits is just as hard with our scheme for the client. This property is useful when metering is employed for royalty payments on copyright material, since it inhibits owners of the copyrights from inflating their royalties.

7 Caveats

The incremental timing scheme uses the difficulty of computing the timing function to leverage auditability. This difficulty (likewise, the computation time invested) may differ significantly between different clients. At one extreme, clients may be smart Internet terminals with very limited computational power, and at the other extreme, clients may be top of the line workstations. Thus, the timing scheme should be tuned carefully to the environment. When metering is deployed by an ISP (e.g., for reverse charging), this problem may be alleviated since an ISP has direct link into the client's environment.

Currently, dynamic deployment of software is supported in the Internet only for Java programs running on a Java virtual machine. Experience with computationally intensive Java applications indicates that it is inefficient [3]. This suggests that a forger can gain a significant advantage, even on a machine of comparable power, by coding the timing function in a more suitable language. We believe that the Java environment will mature in the future, will become efficient, and will include accompanying mathematical libraries (such as CryptoLib [9]), enabling efficient cryptographic software in Java.

Clients can avoid being metered in our approach in many ways, e.g., by disabling Java. This will be detectable by the meter after the client has finished its visit, and of no real benefit to the client. The future will tell whether Java (or a similar mechanism for mobile code) will be largely accepted by WWW users or not.

References

1. S. Ar, J. Cai, Reliable benchmarks using numerical instability. In *ACM Symposium on Discrete Algorithms (SODA)*, pages 34–43, 1993.

2. T. Berners-Lee and D. Connolly. Hypertext Markup Language - 2.0. RFC-1866, SRI Network Information Center, 1995.
3. W. Bradley, J. B. Lacy and R. Wright. Porting CryptoLib to Java: preliminary performance results. private communication, 1996.
4. J. Cai, R. Lipton, R. Sedgewick and A. Yao, Towards uncheatable benchmarks. *IEEE Structures*, pages 2–11, 1993.
5. W. R. Cheswick and S. M. Bellovin. *Firewalls and Internet Security, repelling the wily hacker*. Addison-Wesley, 1994.
6. C. Dwork and M. Naor. Pricing via Processing or Combatting Junk Mail. In *Advances in Cryptology—CRYPTO '92, volume 740 of Lecture Notes in Computer Science*, pages 139–147, 1993.
7. S. Glassman, M. Manasse, M. Abadi, P. Gauthier, and P. Sobalvarro. The Millicent Protocol for Inexpensive Electronic Commerce. In *Proc. 4th International World Wide Web Conference*, pages 603–618, December 1995. http://www.research.digital.com/SRC/millicent.
8. K. E. B. Hickman. The SSL Protocol. Technical report, Netscape Communications Corp, 1995.
9. J. B. Lacy, D. P. Mitchell, and W. M. Schell. CryptoLib: Cryptography in Software. In *Proceedings of the 4th USENIX Security Workshop*, pages 1–17, October 1993.
10. A. Schiffman E. Rescorla. The Secure HyperText Transfer Protocol. Technical report, Web Transaction Security Working Group, Enterprise Integration Technologies, July 1995.
11. E. I. Schwartz. Advertising Webonomics 101. *Wired*, 4(02):74–82, 1996.
12. Gary Welz. The Internet World Guide to Multimedia on the Internet. http://found.cs.nyu.edu/found.a/CAT/misc/welz/internetmm/index.html; to be published.

This article was processed using the LaTeX macro package with LLNCS style