Pricing via Processing
or
Combatting Junk Mail*

Cynthia Dwork * Moni Naor †

Abstract

We present a computational technique for combatting junk mail, in particular, and controlling access to a shared resource, in general. The main idea is to require a user to compute a moderately hard, but not intractable, function in order to gain access to the resource, thus preventing frivolous use. To this end we suggest several *pricing functions*, based on, respectively, extracting square roots modulo a prime, the Fiat-Shamir signature scheme, and the Ong-Schnorr-Shamir (cracked) signature scheme.

*A Preliminary version of this paper was presented at Crypto'92

*IBM Research Division, Almaden Research Center, 650 Harry Road, San Jose, CA 95120. E-mail: dwork@almaden.ibm.com.

†Incumbent of the Morris and Rose Goldman Career Development Chair, Dept. of Applied Mathematics and Computer Science, Weizmann Institute of Science, Rehovot 76100, Israel. Most of this work performed while at the IBM Almaden Research Center. Research supported by an Alon Fellowship and a grant from the Israel Science Foundation administered by the Israeli Academy of Sciences. E-mail: naor@wisdom.weizmann.ac.il.

1 Introduction

Some time ago one of us returned from a brief vacation, only to find 241 messages in our reader. While junk mail has long been a nuisance in hard (snail) mail, we believe that electronic junk mail presents a much greater problem. In particular, the ease and low cost of sending electronic mail, and in particular the simplicity of sending the same message to many parties, all but invite abuse. In this paper we suggest a computational approach to combatting the proliferation of electronic mail. More generally, we have designed an *access control mechanism* that can be used whenever it is desirable to restrain, but not prohibit, access to a resource.

Two general approaches have been used for limiting access to a resource: legislation and usage fees. For example, it has been suggested that sending an unsolicited FAX message should be a misdemeanor. This approach encounters obvious definitional problems. Usage fees may be a deterrent; however, we do not want a system in which to send a letter or note between friends should have a cost similar to that of a postage stamp; similarly we do not wish to charge a high fee to transmit long files between professional collaborators. Such an approach could lead to *underutilization* of the electronic medium.

Since we believe the real cost of using the medium (plus the profit to the provider) will not serve as a deterrent to junk mail, we propose a system that imposes another type of cost on transmissions. These costs will deter junk mail but will not interfere with other uses of the system. The main idea is for the mail system to require the sender to compute some moderately expensive, but not intractable, function of the message and some additional information. Such a function is called a *pricing function.*

In the more general setting, in which we have an arbitrary resource and a resource manager, a user desiring access to the resource would compute a moderately hard function of the *request id.* (The request id could be composed of the user's identifier together with, say the date and time of the request.)

The pricing function may be chosen to have something like a trap door: given some additional information the computation would be considerably less expensive. We call this a *shortcut.* The shortcut may be used by the resource manager to allocate cheap access to the resource, as the manager sees fit, by bypassing the control mechanism. For example, in the case of electronic mail the shortcut permits the post office to grant bulk mailings at a price chosen by the post office, circumventing the cost of directly evaluating the pricing function for each recipient.

We believe our approach to be of practical interest. It also raises the point that, unlike the situation with one-way functions (functions that are easy to compute but hard to invert) and Cryptography, there is virtually no complexity theory of moderately hard functions, and therefore yields excellent motivation for the development of such a theory.

The rest of this paper is organized as follows. Section 2 contains a description of the properties we require of pricing functions. Section 3 focusses on combatting junk mail. Section 4 describes three possible candidates for pricing functions. We require a family of hash functions satisfying certain properties. Potentially suitable hash functions are discussed in more detail in Section 5. Section 6 contains conclusions and open problems.

2 Definitions and Properties

We must distinguish between several grades of difficulty of computation. Rather than describe the hardness of computing a function in terms of asymptotic growth, or in terms of times on a particular machine, we focus on the *relative* difficulty of certain computational tasks.

We require three classes of difficulty: *easy*, *moderate*, and *hard*. The term *moderate* can be viewed in two different ways. As an upper bound, it means that computation should be at *most* moderately hard (as opposed to hard); as a lower bound it means that computation should be at *least* moderately easy (as opposed to easy). The precise definition of easy and moderate and hard will depend on the particular implementation. However, there must be some significant gap between easy and moderately easy. As usual, *hard* means intractable in reasonable time, such as factoring a 1024-bit product of two large primes.

The functions we consider for implementing our scheme have a *difference parameter* that serves a role analogous to that of a *security parameter* in a cryptosystem. A larger difference parameter stretches the difference between easy and moderate. Thus, if it is desired that, on a given machine, checking that a function has been correctly evaluated should require only, say, 10^{-2} seconds of CPU time, while evaluating the function directly, without access to the shortcut information, should require 10 seconds, the difference parameter can be chosen appropriately.

A function f is a *pricing function* if

1. f is moderately easy to compute;
2. f is not amenable to amortization: given ℓ values $m_1, \ldots m_\ell$, the *amortized* cost of computing $f(m_1), \ldots, f(m_\ell)$ is comparable to computing $f(m_i)$ for any $1 \leq i \leq \ell$;
3. given x and y it is easy to determine if $y = f(x)$.

We use the term "function" loosely: sometimes f will be a relation. That is, given x it should be moderately easy to find a y such that the pair (x, y) satisfies the relation, but given (x, y) it should be easy to determine whether it satisfies the relation.

Let $S \subseteq \{0,1\}^*$ be a set that can be easily sampled (i.e. there is an efficient algorithm for selecting a random $s \in S$). $F = \{f_s | s \in S\}$ is a *family* of pricing functions indexed by $s \in S \subseteq \{0,1\}^*$ if, given s, f_s is a pricing function. We will be interested in a collection of families of pricing function $\mathcal{F} = \{F_k | k \geq 1\}$, indexed by a difference parameter k, where the hardness of evaluating $f_s \in F_k$ should increase with k.

Remark 2.1 *It is important not to choose a function that after some preprocessing can be computed very efficiently. Consider the following family of pricing functions F, based on subset sum. The index s is a set of ℓ numbers $a_1, a_2, \ldots a_\ell$, $1 \leq a_i \leq 2^\ell$, such that 2^ℓ is moderately large. For a given request x, $f_s(x)$ is a subset of $a_1, a_2, \ldots a_\ell$ that sums to x. Computing f_s seems to require time proportional to 2^ℓ. As was shown by Schroepel and Shamir [24], after preprocessing, using only a moderate amount of storage, such problems can be solved much more efficiently. Thus, there could be large difference between the time spent evaluating f_s on a large number k of different inputs, such as would be necessary for sending bulk mail, and k individual computations of f_s from scratch. This is clearly undesirable.*

We now introduce the notion of a *shortcut*, similar in spirit to a trapdoor one-way permutation, introduced by Diffie and Hellman [10]. A pricing function with a shortcut is easy to evaluate given the shortcut. In particular, the shortcut is used for bypassing the access control mechanism, at the discretion of the resource manager.

A collection $\mathcal{F}$ of families of pricing functions is said to have the *shortcut property* if for $k \geq 1$ there exists an efficient algorithm that generates a pair (s, c) where

1. s is uniformly distributed in S.

2. given s (but not c) f_s is a function in $\mathcal{F}$.

3. c is a *shortcut*: computing f_s is easy given s *and* c.

Note that since f_s is a pricing function, it is not amenable to amortization. Thus, given s, finding c or an equivalent shortcut, should be hard.

Remark 2.2 *The consequences of a "broken" function are not severe. For example, if a cheating sender actually sends few messages, then little harm is done; if it sends many messages then the cheating will be suspected, if not actually detected, and the pricing function or its key can be changed.*

In the context of junk mail we use hash functions so that we never apply the pricing function to a message, which may be long, but only to its hash value. Ideally, the hash function should be very easy to compute. However, given m, h, and m', it should not be easy to find m'' closely related to m' such that $h(m'') = h(m)$. For example, if Macy's sends an announcement m of a sale, and later wishes to send an announcement m' of another sale, it should not be easy to find a suffix z such that $h(m' \cdot z) = h(m)$.

Suitable hash functions could be based on DES, subset sum, $MD4$, $MD5$, and Snefru. We briefly discuss each of these in Section 5.

3 Junk Mail

The primary motivation for our work is combatting electronic junk mail. We envision an environment in which people have computers that are connected to a communication network. The computers may be used for various anticipated activities, such as, for example, updating one's personal database (learning that a check has cleared), subscribing to a news service, and so on. This communication requires no human participation. This is different from the situation when one receives a personal letter, or an advertisement of a product in which one is likely to be interested, which clearly require one's attention. Our interest is in controlling mail of this second kind.

The system requires a single pricing function f_s, with shortcut c, and a hash function h. The selection of the pricing function and the setting of usage fees are controlled by a *pricing authority*. All users agree to obey the authority. There can be any number of trusted agents that receive the shortcut information from the pricing authority. The functions h and f_s are known to all users, but only the pricing authority and its trusted agents know c.

To send a message m at time t to destination d, the sender computes $y = f_s(h(\langle m, t, d\rangle)$ and sends $\langle y, m, t\rangle$ to d. The recipient's mail program verifies that $y = f_s(h(\langle m, t, d\rangle)$. If the verification fails, or if t is significantly different from the current time, then the message is discarded and (optionally) the sender is notified that transmission failed. If the verification succeeds and the message is timely, then the message is routed to the reader.

Suppose the pricing function f has no short-cut. In this case, if one wants to write a personal letter, the computation of f_s may take time proportional to the time taken to compose the letter. For typical private use that may be acceptable. In contrast, the computational cost of a bulk mailing, even a "desirable" (not junk) mailing, would be prohibitive, defeating the whole point of high bandwidth communication.

In our approach bulk mail, such as a call for papers for a professional conference, or an announcement of a new product, is sent using the shortcut c, which necessarily requires the participation of the system manager. The sender pays a fee and prepares a set of letters, and one of the trusted agents evaluates the pricing function as needed for all the letters, using the shortcut. Since the fee is levied to deter junk mail, and not to cover the actual costs of the mailing, it can simply be turned over to the recipients of the message (and used to pay for the services of the authority) .

Finally, each user can have a *frequent correspondent list* of senders from whom messages are accepted without verification. Thus, friends and relatives could circumvent the system entirely. Moreover, one could join a mailing list by adding the name of the distributor to one's list of frequent correspondents[1]. The list, which is maintained locally by the recipient, can be changed as needed. Thus, when submitting a paper to a conference, an author can add the name of the conference to the list of frequent corresponders. In this way the conference is spared the fees of bulk mailing.

[1]Similarly, one could have a list of senders to whom access is categorically denied.

4 Pricing Functions

In this section we list three candidate families of pricing functions. All the candidates use number-theoretic algorithms. For a good introduction to this area see [9, 13]. The first pricing function is the simplest, but has no shortcut. The other two do have good shortcuts.

4.1 Extracting Square Roots

The simplest implementation of our idea is to base the difficulty of sending on the difficulty (but not infeasibility) of extracting square roots modulo a prime p. Again, there is no known shortcut for this function.

- **Index:** A prime p of length depending on the difference parameter; a reasonable length would be 1024 bits.
- **Definition of f_p:** The domain of f_p is Z_p. $f_p(x) = \sqrt{x} \bmod p$.
- **Verification:** Given x, y, check that $y^2 \equiv x \bmod p$.

The checking step requires only one multiplication. In contrast, no method of extracting square roots $\bmod p$ is known that requires fewer than about $\log p$ multiplications. Thus, the larger we take the length of p, the larger the difference between the time needed to evaluate f_p and the time needed for verification.

4.2 A Fiat-Shamir Based Scheme

This pricing function described in this section is based on the signature scheme of Fiat and Shamir [11]. The idea is to reduce the difficulty of forging signatures in that scheme. The security of the Fiat-Shamir signature Scheme is based on

- The difficulty of factoring large numbers (or equivalently of extracting square-roots modulo a composite).
- A hash function whose range size is (exponential in) the security parameter. Ideally, this hash function should behave as a random function and the time it takes to forge a message should be proportional to the range size.

The proposed pricing function is obtained by taking the Fiat-Shamir signature scheme with a smaller security parameter for the hash function. Searching a range of size exponential in the security parameter should be feasible, but time-consuming. The scheme is as follows:

- **Index:** Let $N = pq$, where p and q are primes of sufficient length to make factoring N infeasible (currently 512 bits each suffices, but if there is further progress in factoring algorithms, then 1024 bits should be used). Let $y_1 = x_1^2, \ldots, y_k = x_k^2$ be k squares modulo N, where k depends on the difference parameter. Finally, let h be a hash function whose domain is $Z_N^* \times Z_N^*$, and whose range is $\{0,1\}^k$. h can be obtained from any of the hash functions described in Section 5 by taking the k least significant bits of the output. The index s is the $(k+2)$-tuple $(N, y_1, \ldots, y_k, h)$.

- **Shortcut:** The square roots $x_1, \ldots, x_k$.

- **Definition of f_s:** The domain of f_s is Z_N^*. Below, we describe a moderately easy algorithm for finding z and r^2 satisfying the following conditions. Let us write $h(x, r^2) = b_1 \ldots b_k$, where each b_i is a single bit. Then z and r^2 must satisfy

  $$z^2 = r^2 x^2 \prod_{i=1}^{k} {y_i}^{b_i} \bmod N.$$

  $f_s(x) = (z, r^2)$ (note that f_s is a relation).

- **Verification:** Given x, z, r^2, compute $b_1 \ldots b_k = h(x, r^2)$ and check that

  $$z^2 = r^2 x^2 \prod_{i=1}^{k} {y_i}^{b_i} \bmod N.$$

- **To Evaluate f_s with Shortcut Information:** Choose an r at random, compute $h(x, r^2) = b_1 \ldots b_k$, and set $z = rx \prod_{i=1}^{k} {x_i}^{b_i}$. $f_s(x) = (z, r^2)$.

- **Evaluating f_s without Shortcut Information:**

  $f_s(x) = (z, r^2)$ can be computed as follows.

  Guess $b_1 \ldots b_k \in \{0,1\}^k$.
  Compute $B = \prod_{i=1}^{k} {y_i}^{b_i} \bmod N$.
  Repeat:
      Choose random $z \in Z_N^*$
      Define r^2 to be $r^2 = (z^2/Bx^2) \bmod N$
  Until $h(x, r^2) = b_1 \ldots b_k$.

In the evaluation of f_s without the shortcut the expected number of iterations is 2^k, which, based on the intuition driving the Fiat-Shamir signature scheme, seems to be the best one can hope for. In particular, if h is random, then one can do no better. In particular, retrieving the shortcut $x_1, \ldots, x_k$ is as hard as factoring [21]. In contrast, the verification procedure involves about k multiplications (actually $k/2+1$ expected multiplications) and one evaluation of the hash function. Similarly, given the shortcut the function can be evaluated using about k multiplications and one evaluation of the hash function. Thus, k is the difference parameter. A reasonable choice is $k = 10$.

4.3 An Ong-Schnorr-Shamir Based Scheme, or, Recycling Broken Signature Schemes

A source of suggestions for pricing functions with short cuts is signature schemes that have been broken. The "right" type of breaking applicable for our purposes is one that does not retrieve the private signature key (analogous to factoring N in the previous subsection), but nevertheless allows forging signatures by some moderately easy algorithm.

In this section we describe an implementation based on the proposed signature scheme of Ong, Schnorr and Shamir and the Pollard algorithm for breaking it. In [18, 19] Ong, Schnorr, and Shamir suggested a very efficient signature scheme based on quadratic equations modulo a composite: the public key is a modulus N (whose factorization remains secret) and an element $\ell \in Z_N^*$. The private key is u such that $u^2 = -\ell^{-1} \bmod N$, (i.e a square root of the inverse of $-\ell$ modulo N). A signature for a message m (which we assume is in the range $0 \ldots N-1$) is a solution (x_1, x_2) of the equation $x_1^2 + \ell \cdot x_2^2 = m \bmod N$. There is an efficient signing algorithm, requiring knowledge of the private key:

- choose random $r_1, r_2 \in Z_n^*$ such that $r_1 \cdot r_2 = m \bmod N$
- set $x_1 = \frac{1}{2} \cdot (r_1 + r_2) \bmod N$ and $x_2 = \frac{1}{2} \cdot u \cdot (r_1 - r_2) \bmod N$.

Note that verifying a signature is extremely easy, requiring only 3 modular multiplication.

Pollard (reported and extended in [20]) suggested a method of solving the equation without prior knowledge of the private key (finding the private key itself is hard – equivalent to factoring [21]). The method requires roughly $\log N$ iterations, and thus can be considered moderately hard, as compared with the verification and signing algorithms, which require only a constant number of multiplications and inversions. For excellent descriptions of Pollard's method and related work see [6, 14].

We now describe how to use the Ong-Schnorr-Shamir signature scheme as a pricing function.

- **Index:** Let $N = pq$ where p and q are primes let $\ell \in Z_n^*$. Then $s = (N, \ell)$.
- **Shortcut:** u such that $u^2 = \ell^{-1} \bmod N$
- **Definition of f_s:** The domain of f_s is Z_N^*. Then $f_s(x) = (x_1, x_2)$, where $x_1^2 + \ell x_2^2 = x \bmod N$. f_s is computed using Pollard's algorithm, as described above.

- **Verification:** Given x_1, x_2, x, verify that $x = x_1^2 + \ell x_2^2$.
- **To Evaluate f_s with Shortcut Information:** Use the Ong-Schnorr-Shamir algorithm for signing.

5 Hash Functions

Recall that we need hash functions for two purposes. First, in the context of junk mail, we hash messages down to some reasonable length, say 512 bits, and apply the pricing function to the hashed value of the message. In addition, we need hashing in the pricing function based on the signature scheme of Fiat-Shamir.

We briefly discuss four candidate hash functions. Each of these can be computed very quickly.

- **DES:** Several methods have been suggested for creating a one-way hash function based on DES (*e.g.* [16] and the references contained therein). Since DES is implemented in VLSI, and such a chip might become widely used for other purposes, this approach would be very efficient. Note that various attacks based on the "birthday paradox" [8] are not really relevant to our application since the effort needed to carry out such attacks is moderately hard.
- **MD4 & MD5 :** $MD4$ and $MD5$ are candidate one-way hash functions proposed by Rivest [22, 23]. They were designed explicitly to have a high speed *software* implementation and are in wide use. The length of the output is either 128 or 256 bits. Although a simplified version of MD4 has been successfully attacked [3], we know of no attack on the full MD4. Also, [4] finds "pseudo-collisions" in $MD5$, but it is not clear whether this can be converted into a collision finding algorithm.
- **Subset Sum:** Impagliazzo and Naor [12] have proposed using "high density" subset sum problems as one-way hash functions. They showed that finding colliding pairs is as hard as solving the subset sum problem for this density. Although this approach is probably less efficient than the others mentioned here, the function enjoys many useful statistical properties (*viz.* [12]). Moreover, it is parameterized and therefore flexible.
- **Snefru:** Snefru was proposed by Merkle [17] as a one-way hash function suitable for software, and was broken by Biham and Shamir [2]. However, the Biham and Shamir attack still requires about 2^{24} operations to find a partner of a given message. Thus, it may still be viable for our purposes.

6 Discussion and Further Research

Of the three pricing functions described in Section 4, the Fiat-Shamir is the most flexible and enjoys the greatest difference function: changing k by 1 doubles the difference. The disadvantage is that this function, like the Fiat-Shamir scheme, requires the "extra" hash function.

As mentioned in the Introduction, there is no theory of moderately hard functions. The most obvious theoretical open question is to develop such a theory, analogous, perhaps, to the theory of one-way functions. Another area of research is to find additional candidates for pricing functions. Fortunately, a trial and error approach here is not so risky as in cryptography, since as discussed earlier, the consequences of a "broken" pricing function are not severe. If someone tries to make money from having found cheaper ways of evaluating the pricing function, then he or she underprices the pricing authority. Either few people will know about this, in which case the damage is slight, or it will become public.

A growing area of research is the economics of networks [15, 7, 5] where issues such as the effect of pricing on the network behavior are investigated. It is interesting to see whether there are connection between this direction and the ideas suggested in this paper.

Finally, the evaluation of the pricing function serves no useful purpose, except serving as a deterrent. It would be exciting to come up with a scheme in which evaluating the pricing function serves some additional purpose.

References

[1] T. A. Berson, *Differential cryptanalysis mod* 2^{32} *with applications to MD5*, Advances in Cryptology — Eurocrypt '92, Springer-Verlag, 1992.

[2] E. Biham and A. Shamir, *Differential Cryptanalysis of Snefru, Khafre, REDOC-II, LOKI, and Lucifer*, Advances in Cryptology - Proceedings of Crypto'91, Lecture Notes in Computer Science No. 576, Springer Verlag, 1992, pp. 156–171.

[3] B. den Boer and A. Bosselaers, *An attack on the last two rounds of MD4*, Advances in Cryptology - Proceedings of Crypto'91, Lecture Notes in Computer Science No. 576, Springer Verlag, 1992, pp. 194–203.

[4] B. den Boer and A. Bosselaers, *Collisions for the compression function of MD5*, Advances in Cryptology — Eurocrypt '93, Springer-Verlag, Lecture Notes in Computer Science 765, 1994, pp. 293–304.

[5] R. Bohn, H-W. Braun, K. C. Claffy and S. Wolff, *Mitigating the coming Internet crunch: multiple service levels via Precedence*, Journal on High Speed Networks, 1994.

[6] E. F. Brickell and A. M. Odlyzko, *Cryptanalysis: A Survey of Recent Results*, Proceedings of the IEEE, vol. 76, pp. 578-593, May 1988.

[7] R. Cochi, D. Estrin, S. Shenkar and L. Zhang, *Pricing in computer networks; motivation, formulation, and Exmaple*, ACM-IEEE Trans. on Computer Networks.

[8] D. Coppersmith, *Another Birthday Attack*, Advances in Cryptology - Proc. Crypto '85, Springer Verlag, LNCS, Vol. 218, pp. 369–378.

[9] T. H. Cormen, C. E. Leiserson and R. L. Rivest, **Introduction to Algorithms**, MIT Press/McGraw-Hill, 1990

[10] W. Diffie and M. E. Hellman, *New Directions in Cryptography*, IEEE Trans. Inform. Theory, IT-22, 1976, pp. 644–654

[11] A. Fiat and A. Shamir, *How to prove yourself*, Advances in Cryptology - Proc. of Crypto 86, pp. 641-654.

[12] R. Impagliazzo and M. Naor, *Cryptographic schemes provably secure as subset sum*, Proc. of the 30th IEEE Symp. on Foundation of Computer Science 1989.

[13] N. Koblitz, **A Course in Number Theory and Cryptography**, Springer-Verlag, New York, 1987.

[14] K. McCurley, *Odd and ends from cryptology and computational number theory*, in **Cryptology and computational number theory**, edited by C. Pomerance, AMS short course, 1990, pp. 145-166.

[15] J. Mackie-Mason and H. Varian, *pricing the Internet*, Technical Report, U. Michigan, 1993.

[16] R. C. Merkle, *One Way Functions and DES*, Advances in Cryptology - Proc. of Crypto'89, LNCS 435, Springer Verlag, 1990, pp. 428–446.

[17] R. C. Merkle, *Fast Software One-Way Hash Function*, J. of Cryptology Vol 3, No. 1, pp. 43–58, 1990.

[18] H. Ong, C. P. Schnorr and A. Shamir, *An efficient signature scheme based on quadratic equations*, Proc 16th ACM Symp. of Theory of Computing, 1984, pp. 208–216.

[19] H. Ong, C. P. Schnorr and A. Shamir, *Efficient signature scheme based on polynomial equations*, Advances in Cryptology - Proc. of Crypto 84, LNCS 196, 1985, pp. 37–46.

[20] J. M. Pollard and C. P. Schnorr, *Solution of* $X^2 + ky^2 = m \bmod n$, IEEE Trans. on Information Theory., 1988.

[21] M. O. Rabin, *Digital Signatures and Public Key Functions as Intractable as Factoring* Technical Memo TM-212, Lab. for Computer Science, MIT, 1979.

[22] R. L. Rivest, *The MD4 Message Digest Algorithm*, Advances in Cryptology - Proc of Crypto'90, LNCS 537, Springer Verlag, 1991, pp. 303–311.

[23] R. L. Rivest, *The MD5 Message Digest Algorithm*, Internet Request for Comments, April 1992, RFC 1321.

[24] R. Schroepel and A. Shamir, *A* $T = O(2^{n/2})$, $S = O(2^{n/4})$ *algorithm for certain NP-complete problems.* SIAM J. Computing, 10 (1981), pp. 456-464.