How to Configure Proof-of-Work Functions to Stop Spam

Sebastian Golze, Gero Mühl, Torben Weis

Kommunikations- und Betriebssysteme
Technische Universität Berlin
Sekretariat EN-6, Einsteinufer 17
10587 Berlin
{golze, gmuehl, weis}@ivs.tu-berlin.de

Abstract. Spam email is a growing problem for today's Internet infrastructure. Besides many filtering techniques, proof-of-work functions have been proposed to fight spam email. Proof-of-work functions are moderately hard cryptographic functions which allow a computer to proof that a certain amount of resources has been spent. Since spammers have limited resources calculating proof-of-work functions can reduce the amount of email they can send out. In this paper, we put the costs of calculating proof-of-work functions in relation to the potential profit. This relation must be known in order to parameterize these functions such that a spammer makes no profit. We investigate in detail the monetary costs of different categories of proof-of-work functions. This allows us to determine how much hardware resources have to be spent per email message in order to make sending spam email unprofitable. The main result of our work is that proof-of-work functions must be a lot harder to calculate than usually assumed by other authors.

1. Introduction

Unsolicited Bulk Email (UBE) usually referred to as *spam* [DEF] is a growing problem for today's Internet infrastructure. It is difficult to find precise statistics about the spam ratio in current email traffic but some service providers claim that already half of their email volume is spam. The usual approach to fight spam is to employ filters and blacklists. One disadvantage of these measures is that legitimate messages could also be blocked. This leads to a less reliable email service than we used to know it. The other disadvantage is that a cat and mouse game between service providers and spammers arises. This means that the spam problem is not solved permanently. Spammers will try to find ways to circumvent these measures and service providers will be forced repeatedly to react to these attacks. Hence, this paper concentrates on a different approach: making spam unprofitable. The reason why people are sending spam is they want to get rich. If we stop them from making money, they will stop sending spam.

Our basic idea is to compare spam with classic post mailings. Everybody gets some commercial mail but the amount is limited because sending such mailings costs money. All we have to do is to charge for every email a small amount and spammers will stop sending email the way they do today. But if we charge that money directly, there is a huge administrative problem. Who can build up a globally accepted infrastructure? Do we want such an infrastructure which could be abused to dominate the Internet?

This means that the idea to charge small amounts of money for sending out an email is a good idea but we have to find other ways to let people pay than to send them a bill from a central email authority. At this point, the proof-of-work (POW) functions come into play. Computational resources cost money. If we can force people to pay per message by forcing them to spend a certain amount of resources on every email they send, we can achieve our goal to make spam unprofitable without any central billing infrastructure.

This idea has been published by Cynthia Dwork and Moni Naor [DN92] in 1992. Since then, some work has been published on different proof-of-work functions. Hashcash [B04] and camram [CAM] are two projects that are actually used in practice. Also Microsoft runs a project named *The Penny Black Project* [PB] researching on proof-of-work based anti-spam solutions.

Among all this work few has been published on how hard the POW will have to be to stop spam. One of the few available publications is from Laurie and Clayton [LC04]. However, they only roughly estimated the necessary hardness of the POW by setting into relation the number of email globally exchanged per day and the overall number of computers connected to the Internet. In contrast to that, we derive more precisely the necessary hardness of the POW.

2. What should be the minimum price for a message?

In order to parameterize a proof-of-work function properly, we need an estimate for the profit a spammer makes by sending out spam. Unfortunately, spammers do not publish their business results. Hence, we have to approximate these numbers. Often spam is used as an advertising medium where the spammer sells this medium to someone else. As the efficiency of every single message is relatively low, we can assume that the average spam advertising customers will not pay more per message than they would have to pay to get a full banner display on a reputable website. These numbers are published and, depending on which website we look at, prices start at about 10 EUR for 1.000 banner impressions. This corresponds to 0,01 EUR per impression which we also consider to be the upper limit for the profit that can be made by sending out spam email.

The second approach is to look at the spammers that have been convicted. Usually, no detailed numbers are published on how much profit they made with their illegal business. But the profit people make spamming professionally on a large-scale seems to be in the region of a few million EUR. Howard H. Carmack, who has recently been convicted for sending spam and other related crimes, was accused having sent out over 800 million spam messages [Heise]. From the coverage we conclude that assuming that he made a total profit of 2 million EUR is reasonable. This means, that the average profit per message was about 0,0025 EUR. Another case is described in an article by Matthew Barakat [Bar04]. He writes on the case of Jeremy Jaynes, who is currently accused in Virginia for sending spam. Barakat writes that Jaynes sent out about 10 million e-mails a day while getting 10.000 to 17.000 orders a month earning him about 40 USD each. If we sum up these values and assume for our calculation 15.000 orders a month his profit per message is about 0,002 EUR. Now, we have a general idea of the profit that can be made by sending spam. Taking the average of the two approximation methods, we assume that a spammer makes a maximum of 0,005 EUR per message.

3. What does the calculation of a CPU-bound POW function cost?

The costs generated by the calculation of a CPU-bound proof-of-work function is mainly composed of electricity and hardware acquisition. We start to calculate the average electricity cost involved in the calculation of a proof-of-work function. We only take into account the normal rates for residential electricity delivery as we assume that spammers will not consume enough electricity to get the industrial rates. The average retail price for electricity in Europe is about 0,13 EUR/KWh according to [E03]. In Greece, currently having the cheapest electricity rates, only 0,065 EUR will be charged for each KWh. According to [E98] electricity costs in the United States and Canada seem to be comparable to those in Greece. As spammers will try to get the cheapest electricity available to keep their costs low, we assume that electricity can be bought for 0,05 EUR/KWh.

In order to calculate what a MHz*s costs, we have to consider the hardware acquisition costs and the power consumption of these systems multiplied with the assumed electricity price. As a simplification, we do not take the distinct performance characteristics of CPUs into account. We simply assume that all CPUs perform the same work in one MHz*s. Further considering the acquisition costs of hardware, we suppose that a spammer wants to make profit relatively fast. Hence, the hardware has to pay back in one year. In addition to the CPU, a basic computer system comprising a mainboard, some memory, and a LAN card is needed. We assume that these components consume about 40W and add that constant to the power consumption of all CPUs to obtain more realistic numbers. Table 1 shows a choice of CPUs with their parameters. According to these numbers, we assume in later calculations that the average spammer will have to pay for one MHz*s at least 6,0 E-9 EUR.

CPU	MHz	W	W/MHz incl. system	Approx. Price [EUR]	Hardware EUR/MHz *s	Electricity EUR /MHz*s	Total EUR /MHz*s
Duron	1800	53,0+40	0,052	300	5,28E-09	7,18E-10	6,00E-09
Intel P4	1300	65,2+40	0,081	200	4,88E-09	1,12E-09	6,00E-09
AMD K7	700	50,0+40	0,129	130	5,89E-09	1,79E-09	7,67E-09
Intel PII	333	20,6+40	0,182	70	6,67E-09	2,53E-09	9,19E-09
AMD K5	100	15,4+40	0,554	30	9,51E-09	7,69E-09	1,72E-08
Intel 486SX	33	3,4+40	1,316	10	9,61E-09	1,83E-08	2,79E-08

Table 1 – Power consumption of different CPU models (sources [THG] and [Hare])

4. How hard must a CPU-bound POW function be to stop spam?

Using the numbers calculated above, we are now able to calculate how hard a CPU-bound POW has to be in order to make spamming unprofitable. We said that the maximum profit out of one message is 0,005 EUR and one MHz*s costs 6,0 E-9 EUR. From the equation *POW in MHz*s = profit per message in EUR / (price per MHz*s in EUR)* we conclude, that we will have to demand for every message about 830.000 MHz*s. Table 2 summarizes how long each of our sample CPUs would have to work to deliver 830.000 MHz*s.

CPU	MHz	POW [s]	POW [min]	POW [h]
AMD Duron	1800	461	8	0,1
Intel P4	1300	638	11	0,2
AMD K7	700	1186	20	0,3
Intel PII	333	2492	42	0,7
AMD K5	100	8300	138	2,3
Intel 486SX	33	25152	419	7,0

Table 2 - Times different CPUs need to deliver a POW of 830.000 MHz*s

5. Memory size bound POW functions

An alternative to CPU bound functions could be memory size bound functions. We will not describe any detailed POW function but simply investigate the economic parameters of this possibility. We already have numbers for the electricity and acquisition costs of computer systems. In the CPU bound approach the systems would be equipped with a minimum of memory to reduce costs. For this approach we have to find out what today's minimal cost for one megabyte of memory is. Contrary to complete computer systems old used memory is usually more expensive per MB than new memory. So we will just consider the current retail prices for memory. The average price for 512 MB is a little over 100 Euro so we can assume that one MB is currently worth about 0.20 Euro. Further we assume that we can pack about 2 GB of memory into one computer system worth 200 Euro bare of memory. This means that 1 MB of running memory (including the computer system) costs *2000/(200+2000*0.20)= 0.30* Euro. As for the CPU-bound approach we will assume that the hardware will have to pay back in one year.

Knowing the acquisition costs of a running MB of memory we will have to determine its power consumption. For the basic computer system without memory we assume a power consumption of 100 W. Modern memory consumes about 0.1 W per MB. So our computer with 2 GB memory will consume a total of *100W+2000*0.1W= 300W*. This means that the total energy consumption for one MB is *300W/2000MB=0.15W/MB* which allows us to calculate the total electricity costs for one MB*s: *(0.15W/MB) * (0.05EUR/3600,000W*s) = 2.1E-9 EUR/MB*s.*

These numbers can be combined to calculate how many MB*s are necessary to generate an effective proof-of-work. *POW in MB*s = profit per message in EUR / (price per MB*s in EUR)* which means: *2.4 E6 MB*s = 0.005 EUR / (2.1E -9 EUR/MB*s).*

Table 3 shows how long this memory size bound POW would take to calculate with different memory configurations. The numbers show that the calculation of a sufficient POW would take up to several hours on current machines.

Memory [MB]	**POW [s]**	**POW [min]**	**POW [h]**
64	37500	625	10,4
128	18750	313	5,2
256	9375	156	2,6
348	6897	115	1,9
512	4688	78	1,3

Table 3 - Duration of POW with different memory configurations.

6. Memory cache bound POW functions

This kind of POW function is based on the fact that cache misses slow down an algorithm. The basic observation is that the main memory latency is a lot more uniform across computer systems than CPU speed. According to Abadi et al. [AB03] this type of POW function solves the problem that users of older computers would have to wait too long for their machine to compute the necessary CPU-bound POW. Similar to the CPU-bound functions, we will now calculate how hard the memory cache bound POW must be in order to generate a cost of 0,005 EUR. Table 4 shows a selection of machines and the times they need to calculate a given memory cache bound function called MBound. The measurements are taken from [DGN03] and we added an approximate purchase price for each system.

System	Price [EUR]	MBound [s]	MBound / s	Depreciation/ MBound [EUR]	Power [W]	Electricity/ MBound [EUR]	Price/MBound [EUR]
P2-266	60,00	2,67	0,37	5,08E-06	60	2,23E-06	7,30E-06
P3-1200	200,00	1,00	1,00	6,34E-06	100	1,39E-06	7,73E-06
P4-3060	450,00	1,01	0,99	1,44E-05	200	2,81E-06	1,72E-05
P4-2000	350,00	1,33	0,75	1,48E-05	150	2,77E-06	1,75E-05

Table 4 - Times different systems need to calculate the MBound function

Using these values, we calculated the price for one execution of the MBound function assuming again that electricity costs 0,05 EUR/kWh. These numbers show that we have to assume that one calculation of the MBound function costs at least 7,0E-6 EUR. From this result and the equation *POW in MBound = Profit per message in EUR / (price per MBound in EUR)* we conclude that 714 MBound functions must be calculated to generate a minimal cost of 0,005 EUR. Table 5 depicts how long the calculation of 714 MBound functions takes using different systems.

System	MBound [s]	714 MBound [s]	714 MBound [min]
P3-1200	1,00	714	11,9
P4-3060	1,01	721	12,0
P4-2000	1,33	950	15,8
P2-266	2,67	1906	31,8

Table 5 - Times needed by different systems to calculate 714 MBound functions

7. Turing tests

An automated turing test is an interactive test to distinguish between humans and computers. These tests are also known under the name of CAPTCHA (Completely Automated Public Turing Test to Tell Computers and Humans Apart). These tests require the solution of a puzzle that usually humans can easily find while machines cannot. The most commonly used technique is to distort text, render it into an image and let the client recognize the text. In Figure 1 you see a sample test. In this case the user would have to recognize the four words "CAPTCHA", "spam", "turing test", and "Proof-of-work". [ABL04] and [ABHL03] give a more detailed introduction to these tests. Currently these tests are used by webmail providers in order to prevent automated mass registrations of email accounts.

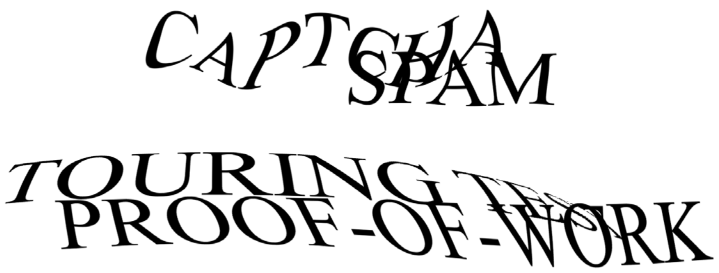

Figure 1 – A sample CAPTCHA test

The disadvantage of these tests is that they are based on open problems in artificial intelligence. There is no warranty that these problems remain unsolved and in the past people have already found new mechanisms to solve certain CATCHPA tests automatically. Some say this means at least a progress in artificial intelligence and the tests can be changed but nevertheless this can be a problem.

We now calculate the cost for a turing test and give an approximation how many tests a user would have to pass to produce a value of 0,005 EUR. Human labor is the most important factor for solving turing tests. Therefore, we solely look at the labor costs involved. Since spammers are criminals and businessman, we assume they do not pay any legal minimum salary or social security etc. They simply try to get the cheapest people available who can read and write because these skills are necessary to solve the commonly used turing tests. We assume that globally such people can be employed for around 40 EUR a month. If they work 160 h each month this means that an hour costs about 0,25 EUR. If we assume that one turing test takes 15 seconds, this means that solving one test costs *0,25EUR* 15 s/ 3600s = 0,001 EUR*. Since a single spam message earns about 0,005 EUR, about 5 turing tests are necessary.

8. Comparison

In the previous sections we have calculated how hard different types of POW functions have to be to stop spammers. Each of these functions could be applied if it is parameterized as described above. Hence, the most important question is how much the introduction of the different functions harms an average legitimate user. Table 6 shows how long the different types of POW functions would take considering two common computer configurations. Note that some values are approximated because we do not have the exact values for all types of POW functions on identical machines.

POW function	**Duration with 1,2GHz, 256MB RAM [min]**	**Duration with 333MHz, 64MB RAM [min]**
CPU	12	42
Memory Size	156	625
Memory Cache	12	30
Turing Test	1,25	1,25

Table 6 – Durations of different POW functions on different machines

Turing tests have the advantage that they do not require a powerful hardware. Moreover, they can be carried out fast in comparison to the other two alternatives (while generating the same cost) because the costs of human labor are relatively high (even in the third world) in comparison to hardware resources. CPU bound and memory cache bound functions are, at least for state of the art hardware, rather similar in their performance while memory size bound functions perform very poorly. On older machines the memory cache miss bound functions have some advantage because an appropriate POW function can be calculated a bit faster than a corresponding CPU bound function.

On first sight, turing tests seem to be the most effective type of POW function. However, they will be harder to introduce to users because they will ask why they have to solve such strange puzzles each time they want to send email. Therefore, we propose a hybrid model which lets the user decide which type of POW function he wants to use: Either the user has got a rather powerful machine and performs a CPU or memory cache bound POW function or he has got a rather weak machine and prefers to pass an appropriate turing test.

9. Conclusion and Future work

Before we conclude we would like to talk about the most critical variables in our calculations which could change and influence the results significantly. First, if the profit per spam message differs a lot from 0,005 EUR all POW calculation times etc. will scale linearly. Second, CPU power could get cheaper than it is today. Third, very old computers could be accessible free of charge. Especially for a memory cache based POW function, large clusters of free machines would be a very cheap way for spammers to send mail. Last, POW functions are based on moderately hard problems. If a spammer finds an efficient shortcut to the POW function, it will no longer be effective. Some people argue that in such a case, the POW function will have to be changed and that the spammers did at least a good thing by solving an open problem in computer science.

Besides the work on better POW functions, means to increase user acceptance should be investigated. For example, algorithms limiting the usage of POW functions to unknown senders could be investigated. Another important point is security. Today, we see at lot of Internet worms etc. which can steal computer power from infected machines to send spam. This would circumvent the idea of POW functions because spammers could get them calculated for free. According to Heise Technology Review [HTR04] this is a growing problem and already today huge "bot networks" can be rented from hacker organizations for about 100 EUR per hour. Another weak point in the proof-of-work concept is that the computing resources used to compute the puzzles are lost besides their proof-of-work effect. In fact this is only a minor problem as the POW that is sometimes used in real life (e.g. waiting in line) does not always have another positive effect besides its regulating function, but nevertheless this point is often objected to POW functions. Therefore, Jakobsson and Juels [JJ99] have published their work on bread pudding protocols and further research in this domain should be done.

References

[AB03] ABADI Martin, BURROWS Mike, MANASSE Mark, WOBBER Ted: *Moderately Hard, Memory-bound Functions*, Feb 2003, NDSS 03 *http://www.hashcash.org/papers/memory-bound-ndss.pdf*

[ABHL03] AHN Lous von, BLUM Manuel, HOPPER Nicholas J., LANGFORD John: *CAPTCHA: Using Hard AI Problems For* Security In: Advances in Cryptology, Eurocrypt 2003

[ABL04] AHN Luis von, BLUM Manuel, LANGFORD John: *Telling Humans and Computers Apart (Automatically)* In: Communications of the ACM, Volume 47, Issue 2, Pages 56 – 60, 2004

[B04] BACK Adam: *Hashcash - A Denial-of-Service Counter-Measure* http://www.hashcash.org/

[Bar04] BARAKAT Matthew: *Trial Shows How Spammers operate* http://apnews1.iwon.com/article/20041114/D86BQ2JO0.html

[CAM]	Camram http://www.camram.org/
[DEF]	The Spamhaus Project: *The Definition of Spam* http://www.spamhaus.org/definition.html
[DGN03]	DWORK Cynthia, GOLDBERG Andrew, NAOR Moni: *On Memory-Bound Functions for Fighting Spam*, Microsoft Research In: Proceedings of the 23rd Annual International Cryptology Conference (CRYPTO 2003), Pages 426-444, Springer 2003 http://research.microsoft.com/research/sv/PennyBlack/demo/lbdgn.pdf
[DN92]	DWORK Cynthia, NAOR Moni: *Pricing via Processing or Combatting Junk Mail* in: CRYPTO 92 http://www.hashcash.org/papers/pvp.pdf
[E03]	Direction Générale de l'Énergie et des Matières Premières*: Prix du gaz et de l'électricité en Europe au 1 er juillet 2003* http://www.industrie.gouv.fr/energie/statisti/pdf/hanprix2.pdf
[E98]	New Zealand Official Yearbook 2000: *International comparison of electricity prices, 1998* http://www.stats.govt.nz/domino/external/web/nzstories.nsf/Response/Electricity
[Hare]	HARE Chris: Processor Electrical Specifications, 1997-2004 http://users.erols.com/chare/elec.htm
[Heise]:	*"Buffalo Spammer" schuldig gesprochen* in: Heise News http://www.heise.de/newsticker/meldung/46237
[HTR04]	*Angriff der ferngesteuerten Computer* in: Heise Technology Review 2004 http://www.heise.de/tr/aktuell/meldung/51686
[JJ99]	JAKOBSSON Markus, JUELS Ari: *Proofs of Work and Bread Pudding Protocols* In: B. Communications and Multimedia Security, Seiten 258-272, Kluwer Academic Publishers, 1999
[LC04]	LAURIE Ben, CLAYTON Richard: *"Proof-of-Work" Proves Not to Work* 2004
[PB]	Microsoft: The Penny Black Project http://research.microsoft.com/research/sv/PennyBlack/
[THG]	Tom's Hardware Guide: *Power Consumption of a High-Power PC* http://www6.tomshardware.com/howto/20021021/powersupplies-02.html