

Practical Evaluation of a Spam Prevention Architecture for IMS Networks

David Waiting and Neco Ventura
University of Cape Town, South Africa
{david, neco}@crg.ee.uct.ac.za

Abstract—Telecommunications technologies are advancing rapidly particularly in the field of mobile telephony. Improved network architectures will provide cheaper, more accessible mobile communication, but unfortunately this exposes mobile users to the type of mobile phone spam and fraud that is currently prevalent in the Internet.

This work proposes a spam prevention architecture that seamlessly integrates with existing IMS networks. The architecture is implemented in a practical network testbed and is subjected to various evaluations. These evaluations demonstrate that the architecture is a viable and effective solution for mitigating spam in IMS networks.

I. INTRODUCTION

Advances in telecommunications technologies and increased competition in the mobile telephony industry have resulted in consumers enjoying rich new mobile services at lower prices. However, this reduction in cost has had an unfortunate consequence. Marketers and fraudsters that traditionally target Internet users are now increasingly targeting mobile telephone users. This mobile phone spam most frequently takes the form of a text message, but pre-recorded voice calls and missed-call spam have also been reported in some countries.

Email spam has been a problem for many years thus necessitating a range of sophisticated and creative spam blocking mechanisms. Most email end users and Internet service providers employ these mechanisms to block the daily flood of unsolicited messages touting cheap medication, stock picks, fake lottery winnings and other similar marketing and fraud attempts. At present, no spam blocking mechanisms are available to prevent mobile telephone spam.

The problem of mobile telephone spam is still small in comparison to email spam. However, new technologies that offer cheaper telephony and rich Internet-like services on the mobile phone are likely to accelerate the problem rapidly. Mobile handsets have essentially become miniature personal computers as evidenced by the multitude of powerful smart phones that run Windows, Symbian and Linux. The mobile networks themselves are also evolving to next generation packet-switched architectures that offer voice over IP and instant messaging, instead of costly circuit-switched calls and SMS. The introduction of new handsets and the reduced cost of communications are both expected to fuel the spam problem in the near future.

A next generation architecture that has gathered considerable support from both industry and academia is known as the 3GPP IP Multimedia Subsystem (IMS), and this is the

architecture on which this work is focused. The IMS has been touted as a service oriented architecture in that new services are slotted into the network with little change to the existing network elements. This is a shift from the conventional network architecture that relies on a stovepipe approach whereby every new service is built from the bottom up and requires expensive and complicated integration with existing operational and business support systems. Furthermore, in the service oriented architecture basic service enablers, for example presence and voicemail, can be combined to form rich combinational services. This approach allows network operators to provision new services easily as they become available but also provides a means to combat the inevitable problem of mobile spam.

In this work a novel spam prevention architecture is proposed that detects and mitigates spam attacks in mobile networks. The architecture is implemented in a practical IMS network testbed that demonstrates proof-of-concept and also provides a platform on which the efficacy of the architecture is evaluated.

The remainder of the paper is laid out as follows. Section II discusses related work in the field. Section III presents the spam prevention architecture. Section IV describes the implementation of a practical network testbed, the integration of the proposed architecture and the test tools utilised. Section V evaluates the proposed architecture and presents results. Section VI concludes the paper and highlights future work.

II. RELATED WORK

In the article *Progressive Multi Gray-Levelling: A Voice Spam Protection Algorithm* [10], Shin *et al.* discuss a new method of detecting voice spam based on the volume of calls made by a particular caller over a certain time period. The proposed mechanism determines whether the caller is a likely spammer or not based on a calculated gray level, a term used to highlight that the system does not categorically accept or reject calls in the way that whitelists and blacklists do. Rather, the system gives the caller a gray value that changes depending on their call patterns. Callers whose score exceeds the threshold, are blocked temporarily until their gray level returns to an acceptable level. Thus the system is seen to be superior to a blacklist that blocks senders permanently even if their behaviour improves. Through their experiments it was found that the algorithm can successfully block short term spamming attempts and over a long period is able to prevent

```

INVITE sip:bob@biloxi.com SIP/2.0
Via: SIP/2.0/UDP biloxi.com;branch=z9hG4bKKnashds8
;received=192.0.2.1
;spam=-5
;spam-detail="Hornel-1.0;whitelist=-10.call_volume=5"
Via: SIP/2.0/UDP sip.atlanta.com;branch=z9hG4bKfjzc
;received=192.0.3.2
;spam=-100
;spam-detail="Jaeger-3.3;not-a-spammer=-100"
Max-Forwards: 70
To: Bob <sip:bob@biloxi.com>
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710@pc33.atlanta.com
CSeq: 314159 INVITE
Contact: <sip:alice@pc33.atlanta.com>
Content-Type: application/sdp
Content-Length: 142

```

Figure 1. SIP message with spam scores appended to the via headers.

persistent spamming behaviour. A modified version of grey-lelling is incorporated in the spam prevention architecture proposed in this paper.

In *The Session Initiation Protocol (SIP) and Spam* [7] Rosenberg and Jennings perform a thorough analysis of the similarities and differences between email and SIP spam, and the applicability of email spam solutions to SIP. Three types of spam are identified as potential targets for the SIP spammer: call spam, instant messaging spam, and presence spam. Several options are identified as possible solutions to the above problems. These include: content filtering, black-listing, whitelisting, consent-based calling, reputation systems, address obfuscation, limited-use addresses, Turing tests, computational puzzles, micro-payments and circles of trust. The architecture proposed in this paper includes several of these techniques.

In the Internet draft *Spam Score for SIP* [13], Wing *et al.* propose a mechanism for SIP proxies to communicate a spam score to downstream proxies or user agents so that alternative call handling can be performed. The spam score is appended to the *Via* header of the current SIP proxy as demonstrated in Fig. 1. This allows intermediate proxies equipped with spam detection software to warn downstream proxies of potential spam. The spam score is most beneficial when inserted by a proxy in the user's home domain, which should be trusted more than any previously traversed domains. It is recommended that only the scores generated by trusted SIP proxies be used, thus preventing untrustworthy proxies from purposefully reducing spam scores. Positive spam scores indicate that the request is considered spam whereas negative scores indicate that it is considered legitimate, and the higher the value the more likely a request is spam.

The 3GPP has proposed a new work item to be carried out with the OMA called *Protection against SMS and MMS spam* [9] in order to deal with mobile spam. The work item only covers text and multimedia messaging in the GSM system, but notes that in the future IMS messaging will also be taken into account. The objective of the study is to develop several techniques for controlling mobile spam by allowing the customer to decide which messages should be considered spam. Furthermore, it is noted that any devised solution should

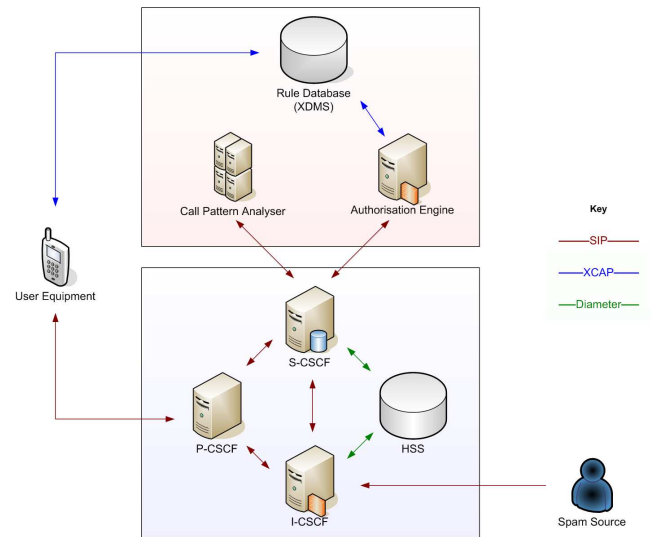


Figure 2. The solution architecture depicted with an IMS core network, user terminal and spam source.

be effective regardless of whether the customer is on the home network or roaming. The goal of the work is to allow mobile operators to provide anti-spam services hence giving their customers increased confidence in both the messaging technologies and content provisioning services.

The work item suggests several solutions to the spam problem including whitelisting, blacklisting and keyword filtering. The document also suggests a junk folder of the mobile device for storing spam temporarily, thus avoiding deleting messages that have been wrongly identified as spam but also conserving storage space for legitimate messages. The mobile may also query the user for validation every time spam is detected so that the user can gain confidence in the system. A rate limiter is also proposed that allows a user to specify the maximum frequency that they will accept messages from one source. As yet, apart from these initial recommendations, there are no outputs from this work item and no time scale has been set for completion.

III. PROPOSED ARCHITECTURE

The solution architecture is depicted in Fig. 2. The architecture is comprised of three reference elements, namely the Call Pattern Analyser, Rule Database and Authorisation Engine. As this is a reference architecture the actual physical architecture is implementation specific, therefore the elements may be combined or distributed over several machines. The elements are shown adjoined to an IMS core network, a user terminal and a spam source.

The Call Pattern Analyser is made up of modules that aim to detect suspicious calling patterns. The term *call* is used instead of *session* for congruency with the literature, but it should be noted that the mechanism operates on all types of IMS sessions, including voice and video calls, text messaging and push-to-talk. Two modules are used in the proposed architecture, namely a session volume analyser and concurrent

session analyser. The modular design of the CPA allows new technologies to be added when they become available. Call pattern analysis is performed on each caller's usage history whenever they initiate a session.

The session volume analysis module detects an abnormal volume of session initiations (regardless of whether or not the sessions are successfully set up) and adjusts the caller's spam probability score appropriately. The concurrent session analysis module detects an abnormal number of concurrent sessions and increases the caller's spam score if the number of concurrent sessions exceeds the amount that would be expected of a legitimate caller. This module is only applicable to sessions that can occur concurrently, i.e. multimedia sessions. The server appends an appropriate spam probability score to its own IP address and inserts this string to the SIP *Via* header, as described by Wing *et al.* [13].

The Authorisation Engine takes input from several sources in order to make an authorisation decision regarding a new session attempt. The server requires three sources of information: the XML rules, downloaded from the Rule Database; the network's default rules, a set of static rules defined by the network operator; and the SIP request itself. The Authorisation Engine may also take auxiliary information deduced by the server itself into account, such as the time of day or day of the week.

The Authorisation Engine may either block or allow the session, or challenge the caller with either a computational puzzle or a Turing test. The computational puzzle requires that the callers UA solve a computationally expensive problem and return the answer before the call is allowed [2]. The Turing test requires that the caller identify a distorted graphic or audio clip - these are also known in the Internet as CAPTCHAs.

There are several header fields that the Authorisation Engine examines in the SIP request, including the *Via*, *To*, *P-Asserted-Identity*, *Contact*, and *Content-Type* headers. Recall that the Spam Probability Score is co-located with the Call Pattern Analyser IP address in the *Via* header. The *To* header indicates which XML rules to download from the Rule Database and the *P-Asserted-Identity* header determines the identity of the user that initiated the session. The *Contact* header field shows the IP address of the user agent client that, like the *P-Asserted-Identity*, may also be subjected to rules such as blacklisting. The *Content-Type* header shows what type of session is being started; for example, multimedia, text-message, or otherwise.

The Rule Database is an XDMS (XML Document Management Server) that provides an XCAP (XML Configuration Access Protocol) [6] interface for interaction with user equipment and other network elements. 3GPP defines the connection to the XDMS as the Ut interface and the protocol as HTTP.

The XDMS is also responsible for other tasks, including authorising the users that attempt to upload or modify any documents, and checking that all uploaded documents are well formed and valid. If an unauthorised user attempts to upload or modify a document, or if the XML of an uploaded document does not parse correctly, then the XDMS is responsible for responding with an appropriate error code. The rule database

only accepts XML documents in the form of the spit-policy schema proposed by Tschofenig *et al.* [12], which is based on the well-known common-policy schema [8].

IV. IMPLEMENTATION

A. Core Proxies and HSS

The IMS core network is implemented using the Fraunhofer FOKUS OSIMS (Open Source IMS Core) [4]. The OSIMS project provides an open source implementation of the three IMS CSCFs: Proxy, Interrogating and Serving. The OSIMS CSCFs were chosen for the test-bed as they are open source, provide an excellent standards compliant solution, and can support a huge volume of SIP signalling as the project extends the highly efficient SER (SIP Express Router). The Home Subscriber Server also forms part of the project although this is written in JAVA.

B. Call Pattern Analyser

The design of the Call Pattern Analyser (CPA) in the previous section calls for two modules to collectively assign a spam probability score to a particular session. The modules are the session volume analyser and concurrent session analyser. Each module is responsible for assigning an integer Spam Probability Score (SPS) to the call history in the range [0, 100].

In this implementation the session volume analysis module is configured with two parameters for each media type: a sliding window time period, and the maximum number of sessions that are allowed during this window. The module subtracts the time window from current time and calculates how many sessions of that type were initiated during this time. Equation 1 shows how the spam probability score for the call volume analysis module is calculated. The numerator is reduced by one so that a user who makes exactly the allowable volume of calls is not given a spam score. The division is a modulus division so any non-integer remainder is discarded.

$$SPS_{Call-Volume} = 10 \left\lfloor \frac{InitiatedSessions - 1}{AllowedSessions} \right\rfloor \quad (1)$$

The implementation of the concurrent session analysis module is not difficult because the CPA stores the number of active calls for each user. The module has only one adjustable parameter - the allowable number of concurrent sessions. The spam probability score calculation shown in Equation 2 is similar to that for the session volume analysis module.

$$SPS_{Concurrent} = 10 \left\lfloor \frac{ConcurrentSessions - 1}{AllowedSessions} \right\rfloor \quad (2)$$

A weighted average of all the module scores is then calculated and appended to the SIP message as it traverses the proxy. The CPA is implemented in the C programming language using the oSIP library. The server is located in the user's home network and is invoked only on terminating calls.

C. Authorisation Engine

In the proposed architecture the Authorisation Engine acts in either proxy or user agent server mode depending on the outcome of the authorisation decision. If, for example, a caller appears on the target user's whitelist then the call must be allowed to continue uninhibited. In this case the server will simply act as a proxy, similar to the CPA, with the exception that the Authorisation Engine does not need to add a *Record-Route* header as it is not interested in any future transactions in the SIP dialog. However, if the authorisation decision requires that either a computational puzzle or Turing test is required then the server is required to terminate the call itself, and thus act as a UAS.

D. Computational Puzzle

The proposed spam prevention architecture requires that there be a method by which to challenge the caller with a computational puzzle. Such puzzles are also commonly referred to as hashcash puzzles as they are often based on hashing algorithms. The implementation of the computational puzzle is based primarily on the Jennings' Internet draft [5]. The draft discusses a method by which a server can calculate a computational puzzle of variable difficulty using a SHA1 hash [3].

The server constructs a string by concatenating a random number, the current time and various header values from the SIP request. The string is then hashed with SHA1 to create what is known as the pre-image. The pre-image is concatenated with the string *z9hG4bK* and then hashed again with SHA1 to create the image. The SHA1 hash is used since it is very quick to create a hash but almost impossible to find the input value of the hash without trying every possible combination of alphanumeric characters. The puzzle difficulty is determined by a parameter known as *work*. The value of work determines how many bits of the pre-image are set to zero. The image, pre-image and work values are then sent to the user agent client that must attempt to recreate the pre-image using a brute-force method. The difficulty of the puzzle increases exponentially as the number of work bits increases.

If a rule states that a computational puzzle is required the Authorisation Engine proxies the request to the UAS, which then generates a random string. The string and the amount of work required are sent back to the UAC in a *419 with puzzle* SIP response. In the proposed architecture the difficulty of the puzzle is related to the Spam Probability Score extracted from the session request. Thus a session with a high spam probability will be required to solve a harder puzzle than one with a lower score.

E. Turing Test

The Turing test server is implemented as a SIP UAS using the eXosip library for signalling and the Gstreamer framework for media delivery. The Gstreamer framework encodes the video in the H.263 codec and the audio in the MPEG layer 1 codec and transmits the stream over RTP to the UAC. The codecs were chosen for their high quality so as to reduce the

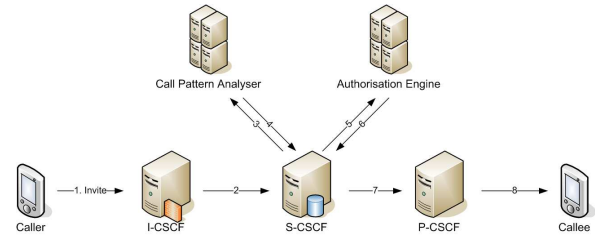


Figure 3. The Call Pattern Analyser and the Authorisation Engine are traversed in a serial fashion.

chance of the test being illegible or inaudible for legitimate callers.

If a rule dictates that a Turing test must be completed then the Authorisation Engine UAS answers the session on behalf of the intended recipient. If the session contains a video component then a visual CAPTCHA is streamed to the UAC, or if no video component is defined then an audio CAPTCHA is used instead. Alternatively, a combined audio/video stream can be used. In all cases the caller is required to identify the visual or audio characters and type them into their IMS client. These characters are transferred to the Authorisation Engine using the SIP NOTIFY request, in a similar fashion that DTMF tones are transmitted over SIP signalling.

If the caller correctly solves the CAPTCHA then the UAS sends a SIP REFER message to the UAC. The URI in the REFER method includes in the *Refer-To* header [11] the original SIP URI of the intended recipient plus an additional token made up of pseudo-random characters included as a URI parameter. This token indicates to the Authorisation Engine that the caller has previously passed the Turing test. The supplied token can only be used once and expires after 3 minutes, thus the UAC is required to re-initiate the call immediately or otherwise risks having to pass another Turing test if the token expires.

F. Session Routing

The two application servers of the proposed spam prevention architecture must be traversed in a serial fashion, as shown in Fig. 3. Every session must traverse the CPA so that the element correctly identify the amount of calls originating from each user but only sessions destined for users who have subscribed to the spam prevention service need to traverse the Authorisation Engine.

Every IMPU in an IMS network is associated with a service profile that contains zero or more initial filter criteria that determine which services are invoked for the different session types. In the case of the spam prevention architecture the user requires two initial filter criteria. The first specifies that all INVITE and MESSAGE requests must traverse the CPA. The second initial filter criterion states that these same requests must traverse the Authorisation Engine for those users who have requested it.

The order in which these application servers are traversed is vitally important, as the Authorisation Engine relies on information inserted into the SIP message by the CPA. For

this reason the initial filter criteria are given integer priority values, zero being the highest priority. Thus it is essential that the CPA initial filter criterion priority value is lower than that of the Authorisation Engine.

The initial filter criteria are assigned trigger points that match particular SIP messages. In the test-bed the trigger points match SIP INVITE and MESSAGE requests on the terminating leg of the session. The initial filter criteria also specify the URI of the application server to which the request should be forwarded.

The S-CSCF adds the address of the next application server to be traversed to a *Route* header, hence ensuring the request will be transmitted to the server using regular SIP loose routing rules. The S-CSCF also adds its own address to a second *Route* header so that, once the application server has been traversed, the request will be routed back to itself. State information is added to this second *Route* header in order to inform the S-CSCF which filter criteria have already been processed, thus avoiding a routing loop.

V. EVALUATIONS AND RESULTS

A. Session Volume Analysis Module

The evaluation is conducted with the SIPp traffic generator [1] acting as both UAC (User Agent Client) and UAS (User Agent Server). The network is configured so that only the CPA is invoked and not the Authorisation Engine.

In order to make reliable comparisons the maximum sessions per minute parameter is held constant at a value of 6 SPM (sessions per minute) throughout these evaluations. This value was chosen because it was found in the available call records that legitimate callers rarely exceed this limit. The evaluations run for a total of 10 minutes each from a cold start, i.e. the CPA has no previous record of the caller.

For each of these sessions the UAC sends an INVITE addressed to the UAS. The UAS immediately responds with a 180 Ringing provisional response followed by a two second delay and then a 200 OK final response. On receiving the final response the UAC sends an ACK request, followed by a five second delay and then a BYE request. The UAS then replies with a 200 OK message confirming the end of the session.

For brevity only two different scenarios are considered. The scenarios are chosen so as to test the range of spam scores, therefore in each of the scenarios the UAC initially generates sessions at 10 times the allowed rate. The sliding window periods are chosen to fit within the evaluation time.

- 1) The UAC generates sessions at 60 SPM. After five minutes the UAC slows its rate to 30 SPM. The sliding window period is four minutes.
- 2) The UAC generates sessions at 60 SPM. After five minutes the UAC slows its rate to 30 SPM. The sliding window period is two minutes.

The results of the evaluations are graphed in Fig. 4 and Fig. 5. The graphs show the time scale in seconds on the horizontal axis, the total session volume scale on the left side axis and the spam score scale on the right side axis.

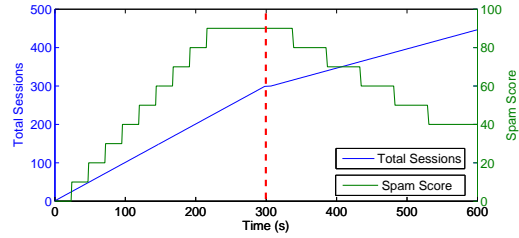


Figure 4. Session volume analysis module evaluation results - Scenario 1.

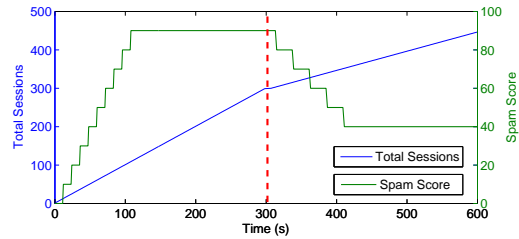


Figure 5. Session volume analysis module evaluation results - Scenario 2.

B. Concurrent Session Analysis Module

The allowable concurrent session parameter is kept constant throughout the evaluations at a value of four so that each scenario can be compared fairly. This value is chosen arbitrarily, but it is unlikely that a legitimate caller would be involved in more than four concurrent calls or text conversations. Each evaluation is allowed to run for ten minutes from a cold start.

The signalling is identical to the session volume analysis module evaluations in that the UAC sends an INVITE to which the UAS responds with a 180 Ringing response followed two seconds later by a 200 OK response. However, in these evaluations the length of the call is adjusted in the various scenarios in order to show how the session length affects the number of concurrent sessions and the corresponding spam score.

Two scenarios are considered for evaluation:

- 1) The UAC generates sessions at a rate of 100 SPM. The session length is 15 seconds.
- 2) The UAC generates sessions at a rate of 30 SPM that increases by 1 SPM every 5 seconds. The session length is 15 seconds.

The results of the evaluations are depicted in Fig. 6 and

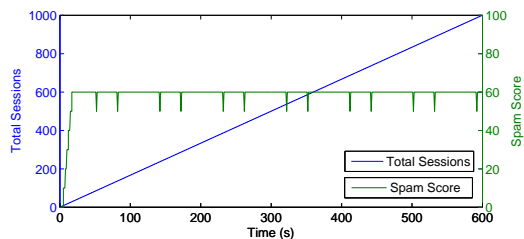


Figure 6. Concurrent session analysis module evaluation results - Scenario 1.

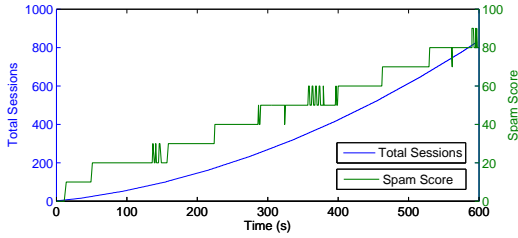


Figure 7. Concurrent session analysis module evaluation results - Scenario 2.

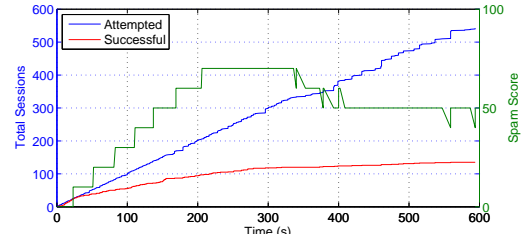


Figure 9. Results of computational puzzle evaluation - Scenario 2.

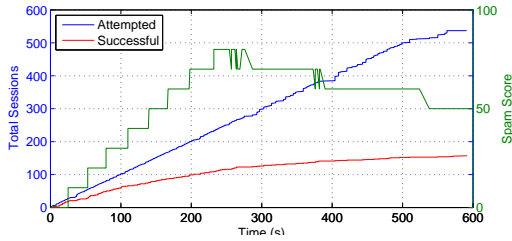


Figure 8. Results of computational puzzle evaluation - Scenario 1.

Fig. 7. The horizontal axis shows the time scale in seconds, the left vertical axis shows the total session scale and the right vertical axis shows the spam score scale.

C. Computational Puzzle

A modified version of the UCT IMS Client is used as a UAC, as the SIPp traffic generator is not capable of solving computational puzzles. The authorisation engine acting in user agent mode answers all incoming INVITE requests that do not contain a correct puzzle header with the 419 With Puzzle error response. The response contains a work parameter that tells the UAC how much work must be performed in order to create an appropriate pre-image.

On receiving the 419 response the UAC sends an ACK and begins solving the computational puzzle. Once the puzzle is solved the UAC sends a new INVITE request, this time with the *Puzzle* header correctly filled, thus proving to the Authorisation Engine that the puzzle has been solved. If the Authorisation Engine receives an Invite with a correct *Puzzle* header it proxies the request to a dummy UAS that answers the request with a 200 OK response, to which the UAC replies with an ACK and then immediately terminates the session.

Two scenarios are considered for the purposes of evaluation:

- 1) The UAC generates sessions at 60 SPM. The Authorisation Engine challenges the UAC with a puzzle of variable difficulty depending on the spam score, where $work = 15 + SpamScore/10$.
- 2) The UAC generates sessions at 60 SPM. The Authorisation Engine challenges the UAC with a puzzle of variable difficulty depending on the spam score, where $work = 16 + SpamScore/10$.

The results from the evaluation scenarios are shown in Fig. 8 and Fig. 9.

VI. CONCLUSIONS AND FUTURE WORK

This work has proposed a spam prevention architecture for IMS networks and implemented it in a practical network testbed thus demonstrating proof of concept. The elements of the architecture were submitted to several evaluations and it was found that the elements behave correctly under testing. Furthermore, it was found that the computational puzzle was able to dynamically slow the rate of session initiation showing that this is a feasible solution for practical networks.

Due to limited space only a brief number of evaluations could be presented. In future work these evaluations will be extended and the proposed elements subjected to more vigorous testing. Furthermore, the overheads associated with the proposed solution were not discussed and this too is left for future publications.

REFERENCES

- [1] SIPp. <http://sipp.sourceforge.net/>.
- [2] Adam Back. Hashcash - A Denial of Service Counter-Measure. *Tech Report*, August 2002.
- [3] D. Eastlake and P. Jones. US Secure Hash Algorithm 1 (SHA1). *IETF RFC 3174*, September 2001.
- [4] Fraunhofer Institute FOKUS. The Open Source IMS Core Project. <http://www.openimscore.org/>.
- [5] C. Jennings. Computational Puzzles for SPAM Reduction in SIP - draft-jennings-sip-hashcash-06. *IETF Internet Draft (work in progress)*, July 2007.
- [6] J. Rosenberg. The Extensible Markup Language (XML) Configuration Access Protocol (XCAP). *IETF RFC 4825*, May 2007.
- [7] J. Rosenberg and C. Jennings. The Session Initiation Protocol (SIP) and Spam. *IETF RFC 5039*, January 2008.
- [8] H. Schulzrinne, H. Tschofenig, J. Morris, J. Cuellar, J. Polk, and J. Rosenberg. Common Policy: A Document Format for Expressing Privacy Preferences - RFC 4745. *IETF Request for Comments*, February 2007.
- [9] 3GPP Technical Specification Group Services and System Aspects. Study Item: Protection against SMS and MMS spam. <http://www.3gpp.org/specs/WorkItem-info/WI-320026.htm>, 2006.
- [10] D. Shin, J. Ahn, and C. Shim. Progressive Multi Gray-Leveling: A Voice Spam Protection Algorithm. *IEEE Network Magazine*, 20(5):18–24, September 2006.
- [11] R. Sparks. The Session Initiation Protocol (SIP) Refer Method. *IETF RFC 3515*, April 2003.
- [12] H. Tschofenig, D. Wing, H. Schulzrinne, T. Froment, and G. Dawirs. A Document Format for Expressing Authorization Policies to tackle Spam and Unwanted Communication for Internet Telephony - draft-tschofenig-sipping-spit-policy-01. *IETF Internet Draft (work in progress)*, July 2007.
- [13] D. Wing, S. Niccolini, H. Tschofenig, and M. Stiemerling. Spam Score for SIP - draft-wing-sipping-spam-score-00. *IETF Internet Draft (work in progress)*, August 2007.