Proofs of communication and its application for fighting spam

Marek Klonowski Tomasz Strumiński

Wrocław University of Technology

Nový Smokovec, January 2008

Agenda

- filtering unwanted mail
- previous work: regular proof-of-work
- proofs-of-communication (POC)
  - creating the POC
  - verifying the POC
- open problems
- conclusions

Filtering unwanted e-mails

Filtering unwanted e-mails

Spam filtering

1. content filtering
   - even the most sophisticated methods can be fooled
   - new spam types demand instant filters adjustment (image spam, pdf spam)
2. address filtering (blacklist, whitelist)
   - address spoofing/forgering
3. challenge-response systems (CAPTCHA)
4. hybrid systems – the most popular presently

Proof-of-work for spam filtering

Idea of Proof-of-work

C. Dwork and M. Naor (in 1992) proposed proof-of-work as an electronic stamp

Proof-of-work (POW)

1. the sender performs some computation to prove his honesty – computation increases costs of sending spam (it is believed that computing proper POW for every single mail is not feasible for the spammer)
2. e-mail with attached POW is sent to the recipient
3. the recipient checks if the POW is valid

Idea of Proof-of-work

C. Dwork and M. Naor (in 1992) proposed proof-of-work as an electronic stamp

Proof-of-work (POW)

1. the sender performs some computation to prove his honesty – computation increases costs of sending spam (it is believed that computing proper POW for every single mail is not feasible for the spammer)
2. e-mail with attached POW is sent to the recipient
3. the recipient checks if the POW is valid

POW essential properties

1. moderatly hard to compute
2. very easy to verify
3. any preprocessing should be useless

Example – POW for spam prevention

POW must depend on

1. sender and recipient addresses (*sender, recipient*)
2. e-mail content (*message*)
3. date and time of sending (*timestamp*)

Example – POW for spam prevention

POW must depend on

1. sender and recipient addresses (*sender, recipient*)
2. e-mail content (*message*)
3. date and time of sending (*timestamp*)

POW example – Hashcash – partial SHA-1 collision

1. find k such that the l most significant bits of SHA-1$(message||sender||receiver||timestamp||k)$ are zeros
2. 2^{l-1} tries required on average
3. one computation of SHA-1 function for verifying

Proof-of-work

- the hardness of the POW should be high enough to make a spamming too expensive
- ... but it also should not be inconvenient for honest sender

Problems

1. after one time investement spammer can still send a lot of e-mails (parallel computing of POWs)
2. effort for the recipient (checking proof)
3. POW computation can be irritating for honest senders

Our approach: Proof-of-communication

Proof-of-communication (POC)

POC are based on a sender's Internet connection bandwith

The idea

1. sender uses a particular e-mail to generate list of hosts
2. he communicates with each of the host from the list
3. communication involves some resource/documents exchanging
4. the POC is a sequence of bytes which proves that for a particular e-mail communication with hosts from list was performed
5. an e-mail with an attached POC is sent to the recipient
6. the recipient checks if attached POC is valid

Proof-of-communication (POC)

Important advantages

1. **a spammer cannot control even a significant number of hosts in the network**
2. even powerful spammer with fast computer cannot create POC significantly faster
3. proof-of-communication does not depend on CPU speed

POC requirements

POC is similar to POW therefore it also depends on *message*, *recipient* and *sender* address, *timestamp*.

Specific POC requirements

1. low traffic overhead
2. dynamic content tolerance
3. no dedicated infrastructure required
4. low connection overhead for POC verification

POC Construction – proof of concept

Our implementation

- without dedicated infrastructure
- on the top of existing Internet protocol

POC Construction – proof of concept

Our implementation

- without dedicated infrastructure
- on the top of existing Internet protocol

HTTP Based POC

1. generating a list of random webpage locations from a particular e-mail data
2. transfering all the webpages
3. making a digest from transfered documents
4. **later:** verifying generated proof

HTTP Based POC
Location generation

Transforming an e-mail to a sequence of webpage locations

1. use a collision-free hash function to generate some pseudorandom bytes
   $seq = H(body||recipient||sender||timestamp)$
2. get the ($seq \ mod \ dictionarySize$)-th word from dictionary
3. use a search service to transform word to some webpage location
4. if it is neccessery repeat the procedure from point 2 using a $seq = H(seq)$

HTTP Based POC

Preparing proofs

Preparing proof from downloaded documents

1. the proof should be in form which allows partial checking
2. the proof should be as short as possible
3. simple proposal:
   $proof = H(page_1)||H(page_2)||..||H(page_n)$
   where
   - H is a hash function with a small range
   - $page_n$ is a downloaded document/resource

HTTP Based POC

Verifying proof (1)

Partial checking

To save the verifier's resources he checks only a part of POC

1. receive an e-mail with attached *proof*
2. generate a list of webpages as described before (based on a received e-mail)
3. randomly choose a subset of k webpage locations
4. download every document from this subset
5. check if every part of the proof is correct

HTTP Based POC

Verifying proof (2)

The adversary wants to **forge** POC.

Probability of cheating

1. n - number of all parts of proof
2. k - number of parts checked by verifier
3. f - number of forged parts
4. $Pr[\text{forgery found}] = 1 - \binom{n-f}{k} / \binom{n}{k} = 1 - \frac{(n-f)!(n-k)!}{n!(n-k-f)}$

For $n = 20$, $k = 5$ and $f = 5$ (only 5 forged parts) the probability of founding a forgery is ~ 0.81 (but the adversary had to do as much as 15 correct communication parts!).

HTTP Based POC

Verifying proof (3)

Dynamic content problem

We assume that some resources can be modified between POC creation and the verification. So we accept the POC if at least a fixed fraction of proofs is correct. According to the experimental results, this strategy works.

- instead of checking if every part is correct simply **count correct parts**
- there should be some numeric treshold above which the proof is found to be correct (its value would depend on particular POC system)

Open problems

Possible application and extensions

- POC seems to be usefull in P2P networks
  - some real advantages in searching, transfering and verifying POC
  - avoiding problems with dynamic content
- maybe some other resources wolud be better in order to prove an electronic effort?
- combination with computational proof-of-work (to prevent DoS attacks)

Open problems

- high traffic overhead
- dynamic content problem in rapidly changing evironment

Conclusions

1. POCs make a spammer dependent on some external resources
2. computational POW is not the only possibility to prove an electronic effort
3. there are methods to make POW independent from the CPU speed

Thank you for your attention!