Proof of work in mix networks

Ajinkya Rajguru, UF ID 91790974

December 9, 2015

Introduction

Active attacker for a mix network presents a challenge as it can reduce the anonymity for messages by a trickle or a flood attack on the mix. Cryptographic hash challenge can be implemented as a cost for inserting a message in the mix to discourage a $n-1$ attack on the mix.

Motivation

Flood attack

A global active adversary can reduce the anonymity size of a target message by identifying next mix in the path and attacking the mix with a flooding attack in case of a threshold mix. The adversary stops the target message from entering the mix via control of routers or other network components. Multiple messages are injected to the mix till the mix fires. The attacker then composes messages that are intended to known targets which can be tracked by the global adversary. The adversary can track and eliminate the messages that were injected by it which reveals the destination of the target messages.

The attacker has a distinct advantage over a honest user in that the message creation time for an honest user is dependent on the public key encryption for the cascade of the mixes. An honest user selects m mixes and successively encrypts the message in the reverse order that it will send it in. It is desirable to pad messages in order to make them look identical at every mix to reduce the risk of traffic analysis. This makes it easier for the attacker to cloak the messages without need for encrypting the message along the entire path. The attacker does not have to worry about maintaining anonymity and can merely create a message with 2 mixes on the path. The first mix will decrypt the message and deliver it to the second mix where the packet will not have any meaningful use to the attacker beyond it. The attacker will be still able to measure the messages received by the second mix in the path to conduct the $n-1$ attack.

The attacker can precompute a generic message to every destination through every other mix selected pairwise. The resulting message has to be encrypted only twice compared to m encryptions for an honest user. The memoized list of messages thus created can be randomized by including random padding such that it looks indistinguishable from any other traffic from honest users. This reduces the time required for an attacker to inject messages in any mix to linear time.

Proof of work to authenticate

Proof of work requires a message to be appended with a cryptographic hash challenge presented by the mix. The exploration of challenge-response pairs to find the correct response is a computationally expensive operation. Each message will have to be appended by a 'proof−of−work'.

Challenge generation strategy.

The cryptographic hash challenge are a tool that requires computational power that requires resources to access intensive resources. Hashing challenges attempt to restore the imbalance between the cost to the user for potentially abusing scarce system resources. For our example we will consider the challenge token and response token as follows.

SHA256 hashing

SHA256 is an one−directional cryptographic hashing strategy that creates a message digest of size 256 bits. The algorithm creates blocks of the input messages of size 32 bytes and padding wherever required.

Conventional cryptographic hashing challenges

Conventional hash challenges involve a strategy where only a challenge string is provided. The evaluating party appends random bits at the end of the

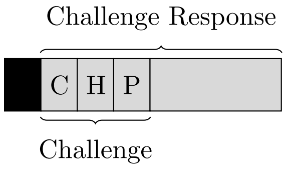
challenge and repeatedly evaluating the hash function and comparing it to a fixed characteristic in the hash.
Data : Hello,World
SHA256 : 74a3f29b1375deb29c69a3d354445050 53045e18d734cc56421b37f353a027bf
Data : Hello,World1
SHA256 : 71f9c06bb35cd446404725a5ed265cd6 e78c133656f5988ba4d3f66e60ccc38f

Changing the bits after message allows the evaluator to generate large number of these hashes. A hash with a predefined characteristic eg leading 40 0's is computed and the suffix for such a hash is determined to be the response.

Evalution

Evaluation of the hashes is a one directional function, ie there is no way to find out the message generating the digest from the digest itself. Hence no consideration can be made regarding what suffix should be tried next. It is guaranteed that over a period of time the suffix which produces an appropriate hash will be guessed.

For a k out of 256 bit hash, the probability of guessing such a hash is $\frac{2^{256-k}}{2^{256}}$ which is $1/2^k$ however there is no upper bound to the number of guesses that will return such a hash. Although this reduces the uncertainty of having an arbitrarily large search space, it has a disadvantage because it makes the attack exact.



C is the part of the challenge response of $k^{`}$ bits. H is the hash expected that satisfy the challenge. P describes the parameters of the challenge provided by the mix including the time for which the challenge is valid, number of hash bits that need to match, etc.

Verification of a token involves parsing the parameters of the challenge and evaluating the hash of the concatenation of the challenge bits and the response and comparing it with the known hash.

The entire challenge token is signed with the private key of the mix which makes it impossible for an attacker to forge challenges. The challenges will have a timeout after which the challenge response will not be accepted reducing an attackers' attack window. The challenges can be generated by multiple strategies outlined below.

a. **Self generated challenge:** The challenge-response is appended to each level of the message which leaks information about the first mix in the route. The messages are encrypted with public keys of the mixes which ensures that adversary can not read the challenge−response directly. However in case that an attacker is able to compromise a mix the information revealed by the challenge−response will reveal the first mix that the message originated from. This greatly reduces the anonymity set size of the message as it points back to the first mix. Thus compromising a mix is highly desirable for an attacker. If an attacker controls a mix, it is possible for the mix to create challenges and leak the response to the attacker who can use the responses to attack other mixes.

b. **Threshold** We are able to secure individual honest mixes from being injected with dummy messages from an attacker. However an compromised mix is still potentially able to forward appparently valid messages to the mix. A simple strategy would be to have a threshold on the messages arriving from mixes within a particular firing cycle. This strategy is recommended as multiple messages from the same mix reduces the anonymity size of the messages.

c. **Trusted challenge generating mix:** The challenge can be created via a delegated mixes that a mix trusts. The mix M will request for challenges to be generated with certain parameters and signed by another trusted mix M_t. The challenge−response can be verified by any party trivially. The trusted matrices are publicly published which allows a user to select and solve one or more challenges that are trusted on the path it chooses. However the fact that only responses from trusted mixes are allowed leads to an intersection attack where messages' anonymity set can be reduced by determining the intersection of all mixes that trust the messages' signing authority. If a well known signing mix is compromised it opens the possibility of a backdoor attack on all mixes trusted by it.

   Replay attack: A mix can be trusted by multiple mixes which opens up the probability of attacking multiple mixes simultaneously by using the same challenge−response pair on all mixes that trust the challenge generating mix.

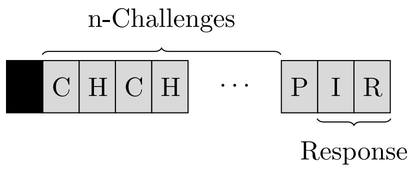
d. **Token based authentication:** The token based authentication involves appending a signed token from the mix when it receives a message from the

previous mix. The mix will only accept signed tokens from mixes it trusts. Each mix strips the previous mix token and forwards the message after appending its own token. The first mix will only evaluate the proof-of-work challenge that it issued. This apparently breaks message unlinkability by providing a back link to the previous mix which is undesirable. However this case is similar to that of crowd networks where local adversary can detect the message for upto one hop. A presence of a single non−compromised mix along the path can ensure that the sender and receiver cannot be linked via the message.
The condition of a mix only requiring to trust the mix before it reduces the potency of an intersection attack. An adversary can create multiple messages from a compromised mix however each mix can verify the previous mix that the message originated from and cap the number of messages accepted from a mix.

e. **Trusted mix with fixed origin:** In order to prevent the replay attack on the mix, each mix will request that k challenges be issued with a secret nonce included in the challenge. Each mix has a strategy that accepts messages only if it detects the same secret nonce in the challenge−response. This eliminates a possibility of a replay attack as the first mix in the message path is determined by the nonce.

f. **Single challenge generating mix** In order to prevent an intersection attack, we can consider a single challenge distributing node that will issue the challenge that can be included in level of the mix messages. The challenges are generated and signed by the generating mix. This allows scalability as multiple nodes can be deployed that are signed by a single private key. However this requires a central trusted party and thus a central point of failure.

The generation of challenges involves digital signatures which may be costly for the mix producing it. Hence multiple challenges with a similar timeout interval and parameters for a mix can be generated with a single digital signature. The response will include the index of the challenge that it is intended to solve.



This also reduces the number of control message overhead that are required to ensure that enough challenges are issued.

Selecting appropriate timeout

The timeout for the challenges needs to be carefully chosen where it should allow a normal user to evaluate and respond to a challenge response without permitting an attacker with reasonable computational resources.

The mix will also ensure that the timeout of the requests overlaps over each epoch of the last batch of requests becoming invalid. The user will pick a challenge with a timeout such that it can be solved. A mix selecting a short timeout may make it impossible for some clients with limited computational power to be unable to send any messages.

A longer timeout on the other hand will create a longer attack window and may cause collisions where two users compute the same challlenge independently.

Practical challenges and implications

Due to the advent of bitcoin miners, cryptographic hashing has been optimized to the level of multiple thousand Gigahashes per second first using GPU optimization and then dedicated standalone FGPA miners. A normal desktop computer can not outperform the dedicated miners in terms of their hashing speeds. This provides a severe challenge to design a proof of work system where an attacker will have economically viable computational resources to defeat the challenges.

One approach would dictate using intentionally slower hashing algorithm that is not optimized for dedicated bitcoin miners.

Conversely, we can use the available hashing power through cloud based Hashing-as-a-Service (HaaS). The business model of such a service would include utilizing bitcoin miners to solve the hash challenges and provide the response over a secure channel. The hashing service will charge a bounty associated with each successful challenge that it responds to. The cost is tied to each message that is sent which does not affect a normal user but may prove to be prohibitive for an attacker who has to send multiple times the number of messages.

Challenge utilization in case of delegated trusted challenge generating mixes

The challenge utilization metric is the number of challenges sucessfully solved within the interval to the number of total challenges issued in that interval.

This ratio will depend on multiple factors like timeout, complexity, number of users attempting the challenges. However a higher ratio may indicate that the mix may have been compromised and the attacker is using the mix to launch the attack. An honest mix M_h will likely reduce the number of challenges that requests to be issued from it's behalf from any mix that shows anomalous behavior $M_a n$.

Because the challenges are visible to all participants, any other honest mix $M_h^{`}$ can monitor the number of challenges being requested from M_h and understand the trust relation between $M_a n$ and M_h and may choose to reduce the challenges it requests from $M_a n$s. This could have a self correcting effect where a mix that has greater than average response rate is considered potentially malicious and is consequently used by fewer honest mixes.

Dummy messages generated by the mix

Even with the proof-of-work in place, it may be required for a mix to generate dummy messages for various diagnostic reasons or as a security against potential attack from an attacker with greater resources than anticipated. The dummy messages can be created and either attached with an authentication token or a valid self $-$generated challenge response.

This increases the complexity of the attack asymptotically combined.