Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 3, 2018

J. Rosenberg
C. Jennings
Cisco Systems
March 2, 2018

SIPCoin: A Cryptocurrency for Preventing RoboCalling on the PSTN
draft-rosenberg-stir-sipcoin-00

Abstract

Robocalling has become an increasing problem in the Public Switched Telephone Network (PSTN). While techniques like verified caller ID can help reduce its impact, ultimately robocalling will continue until economically it is no longer viable. This document proposes a new type of cryptocurrency, called SIPCoin, which is used to create a tax - in the form of computation - that must be paid before placing an inter-domain call on the SIP-based public telephone network. SIPCoin maintains complete anonymity of calls, is non-transferable between users avoiding its usage as an exchangeable currency, causes minimal increase call setup delays, and makes use of traditional certificate authority trust chains to validate proofs of work. SIPCoin is best used in concert with whitelist based techniques to minimize costs on known valid callers.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 3, 2018.

Copyright Notice

Table of Contents

1. Problem Statement

Robocalling (also known as SPAM, voice SPAM, and so on) has become an increasing problem in the Public Switched Telephone Network (PSTN). Efforts to prevent it - such as the do-not-call list - have so far proven ineffective. Recently, robocallers have gotten even more crafty, and are tailoring the caller ID of incoming calls to match the area codes and exchanges of the recipients in order to increase the likelihood that targets pick up the phone.

This problem is not new, and ultimately the techniques for its prevention have been known for some time. [RFC5039] outlines a number of techniques for prevention of SPAM in Session Initiation Protocol (SIP) [RFC3261] based systems.

Ultimately, SPAM calls are a matter of economics. Each call costs the spammer a certain amount of money to perform. However, a small fraction of calls produce a successful result, generating economic returns. As long as the profit is positive, spammers will continue and will likely work around legal hurdles, blacklists, reputation systems, black lists, and so on. Consequently, the only true way to end robocalling is to use economics - to make it no longer profitable.

This can be achieved in two ways. One is by the exchange of actual monies across all access and peering points in the public telephone network. As the telephone network continues to grow, this becomes increasingly difficult. Furthermore, it only requires a single point of failure at one peering point, and calls have a way to enter the network. Indeed, this is exactly why we see robocalling today despite the fact that monies are in fact exchanged within the PSTN.

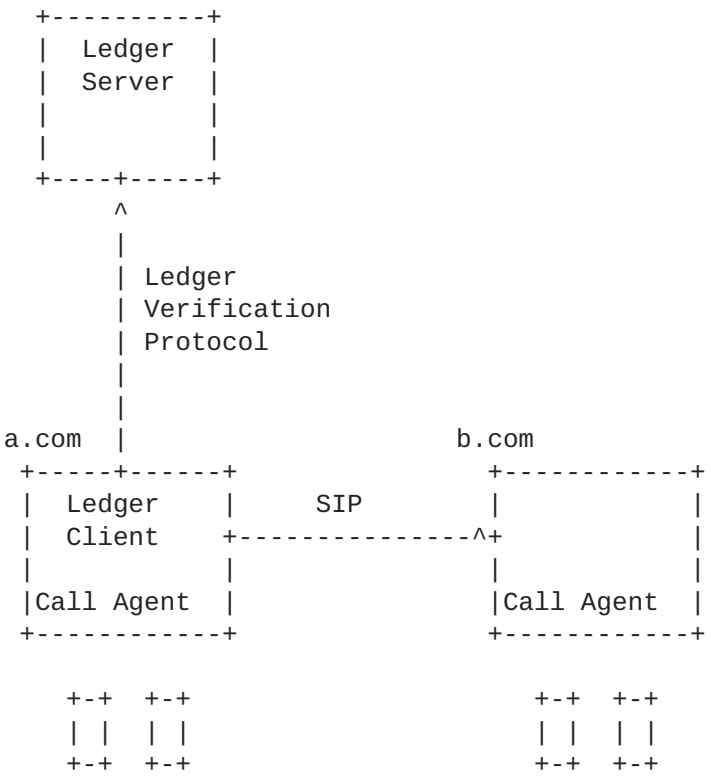
An alternative solution is to use computational puzzles, as described in Section 3.9 of [RFC5039]. The original concept described there is the a callee passes a computation test back to the caller, which performs it, and then passes the results towards the callee. This suffers from two problems. One, described in the document, is that there is high variability in the computation capabilities of individual calling devices and systems. Secondly, performing the computation at call initiation time increases call setup delays. This increase is likely to be large, owing to the amount of computation required to act as an economic disincentive.

Consequently, the problem to be solved is to provide a system that requires callers to demonstrate a proof of work towards callees in a way which does not suffer these problems. Fortunately, in the intervening years since the publication of [RFC5039], blockchain technology was invented, and along with it, a wealth of

cryptocurrencies (BitCoin, Ethereum, etc). The goal is to apply these technologies in a way to solve the unique requirements of the problem at hand.

2. Reference Architecture

The reference architecture for SIPCoin is:



In this architecture, users associated with one call agent (representing a.com) wish to communicate with users associated with a different agent, reachable through b.com, using the Session Initiation Protocol (SIP) [RFC3261]. The b.com agent wishes to gate incoming calls based on proof of computational work provided by the a.com call agent. To perform this, the a.com agent implements the client component of the Ledger Verification Protocol (LVP). In LVP, clients - in this case embedded into the call agent - perform hashing operations, and maintain a self-generated ledger of transactions. To validate pages in the ledger, the ledger client accesses a ledger server through LVP. Through this protocol, the ledger client can obtain information to include int the SIP INVITE. A call agent will typically implement many instances of the ledger client, since each instance has an upper bound on the amount of calls per second it can perform.

In this architecture, there are two call agent roles - the generating agent and the receiving agent. Though, in the picture as shown, they represent the registrar of record for the caller and callee respectively, this need not be the case. Rather, the two roles can be implemented at differing paths along the actual call setup, and indeed occur multiple times along the call. Later sections in this document map the architecture to recommended points of physical implementation.

3. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

Generating Agent
: The SIP proxy, user agent or B2BUA which wishes to demonstrate proof of work in order to pass a call downstream towards a receiving agent which will ultimately validate the proof of work.

Receiving Agent
: The SIP proxy, user agent or B2BUA which will only accept incoming calls under demonstration of proof of work.

4. Requirements

- Unlimited Participants: The system must allow for an unlimited number of call agents to participate. New agents should be able to come and go on demand. This allows the system to extend to agents representing carriers, enterprises, home networks, and so on.

- Low Latency: The system should not significantly increase the call setup delay for calls. This is a big constraint, since it means that proof-of-work computations must be performed in advance of placing the actual call. One to two seconds is acceptable, but not more than that.

- Privacy Protection: There must not be any sharing of logs of calls, personally identifiable information (PII), phone numbers, or similar information. Sharing includes passing this information between entites which would otherwise not have access to it, or storing it in some kind of ledger.

- Non-Transferrable: Any currency used for placing calls must be limited in scope to only allow placing of calls, and not be transferrable amongst participants in the system, or exchangeable for traditional or crypto currencies. This is a significant

requirement since it rules out all existing cryptocurrencies by definition. Why is this requirement important for this use case?

* Enable small players: SIP was designed to enable an open interconnection amongst anyone on the Internet. A SIP domain can be a single device supporting a single user. It can be a home network. It can be a small business. It can be a large enterprise. It can be a small telco, a large telo, or a massive global provider. In order to enable the most open access possible, barriers to entry must be small. Consequently, we want to retain the property of SIP that a two person domain can install an open source SIP server, and be off and able to make calls. Transferability would mean that the currency has real value, and thus to operate a system, the agent must be able to connect to currency exchange systems, payment processing platforms, and so on, in order to obtain the currency before being able to place the first call. This makes it difficult for small players to participate.

* Fraud: The entire purpose of this system is to prevent fraudulent entities from placing calls into the global SIP network. If it was based on transferrable cryptocurrencies, it would likely be susceptible to fraud and thus benefit the very entities we are trying to stop.

* Managed Costs: Today's cryptocurrencies have highly variable exchange rates, sufficiently variable that they are difficult to use as a payment vehicle, and even more difficult to use for microtransactions. However, that is exactly the opposite of our case - we require high volume, extremely low cost microtransactions, at a price point which hits a particular operating point that is just high enough to make it unprofitable for spammers yet not overly expensive for real callers. Consequently, by tying the cost strictly to the price of computation, we reduce (though certainly do not fully eliminate!) the risks of highly variable currency and allow for relatively low cost microtransactions.

o Non Privileged: The system should not require centralized entities to have access to telecom databases or other information which requires governmental or regulatory access. This constraint in the system makes it incrementally deployable without waiting for the centralized bureaucracy of telco operations. Any centralized capabilities must be an easy incremental add to existing services (e.g., a change to current cerificate authorities).

- Phone Numbers or SIP URI: The system should not require phone numbers to operate. It should work with traditional domain-based SIP URI as tell as tel URI phone numbers.

- Predictable Cost: The system must enable a call agent to perform a certain amount of computation and be able to predict the amount of calling which it can perform for a given amount of computation performed in advance of the call. Without this property, an agent runs a risk it cannot service real-time requests for calls from its users because it doesn't have enough crypto currency. This property is related to the non-transferability requirement; if the crypto currencies were transferrable, an agent could instantly purchase crypto currency to place a call. Without transferability, predictable computation is required to ensure the ability to place a call.

- Managed Governance: Since adjustments will need to be made in the computational costs required, the system must support a managed governance model under the authority of a standards body, such as the IETF or ITU.

5. Applicability of Traditional Cryptocurrencies

One immediate question is - why not just use Bitcoin or one of the other crypto currencies? This would be easy to do. Each SIP INVITE would contain a reference to a transaction that passes the required costs from the caller to the callee.

Putting aside for a moment the non-transferability requirement - which rules out all existing cryptocurrency - other requirements make Bitcoin and similar cryptocurrencies non viable.

Firstly, they fail on the privacy requirement. Usage of Bitcoin would require transactions in the ledger to identify the caller and called parties, thus leaking information about who is calling who.

Secondly, the systems do not provide predictable or managed costs, which are essential for this application. The cost of Bitcoin is highly variable, and subject to (sometimes wild) market swings. These costs cannot be managed by any consensus organization, and indeed the cost may collapse entirely, completely destroying the benefit of the system.

Finally, Bitcoin is too slow. It, and similar cryptocurrencies, rely on ledgers which post infrequently, causing transactions to take minutes or even hours to eventually post and be verified. This system requires a transaction - the spending of a coin to place a

call - to happen fast enough that it can be spent by the caller, and verified by the callee, within one to two seconds.

6. Applicability of Challenge Based Solutions

The second question to ask is - why not just have the callee challenge the caller to perform a computational puzzle at time of call setup, and the caller returns the results?

The primary problem with this class of solution is the time it takes to perform enough computation to serve as an economic disincentive for placing spam calls. To get a general feel for the costs using modern compute, consider Amazon EC2 on demand pricing. For a middle of the road compute optimized node - say - the c4.large instance - as of February 25, 2018, Amazon is charging USD 10 cents per hour (.0027 cents per second) of computation for an instance in US East. We can imagine that our goal for disincentivizing an attacker is somewhere between a .1 cent per call, and perhaps as high as a 10 cents per call, this would require computation on this particular instance type of between 37 seconds (for .1 cent of cost) and 1.01 hours (for one dollar).

Of course, modern Bitcoin mining no longer uses CPUs or even GPUs for that matter, but rather ASICs. Though these can perform far more computation per unit time interval than a CPU for specialized hashing. However, the raw cost per hour of operation - regardless of the amount of computation that can be performed - is the question at hand for analyzing the viability of a challenge/response approach. ASIC and GPU based systems are higher cost per hour to operate due largely to their scarcity. [[OPEN ISSUE: hmm, not sure this argument works owing to asymmetry issues]]

37 seconds - and certainly one hour - is far too long to wait before a call can be forwarded to the called party. For this reason, this class of technique does not work. The solution requires the performance of the computation ahead of the call.

[[TODO: go through all EC2 instance types, price out a more normalized compute cost - dollars per Ghz per hour. Such a metric normalizes against number of CPUs as well as variations in the performance of the CPUs.]]

7. Overview of SIPCoin

This section provides an overview of SIPCoin, a new cryptocurrency used for placing SIP calls over the global SIP network.

SIPCoin differs from Bitcoin significantly in that it does not rely on completely decentralized trust. Rather, it bootstraps itself on the existing certification authorities which power the modern web. As such, the system has two distinct actors - clients, and servers. Clients are entities which perform computation in order to create SIPCoins, and then "burn" those coins in order to place a call. Consequently, SIPCoin supports only two types of transactions - a "create" transaction which creates a Bitcoin through the solution of computational puzzles, and then a "burn" transaction which destroys a coin by binding it to a particular SIP call. Since the create and burn transactions are localized - they affect only the client itself - there is never a need for sharing of the ledger. Consequently, clients actually maintain their own ledgers for these transactions, as described below. A client needs to provide proof that it has burned a token; that proof is performed with a different object - a Burn Receipt - constructed by the client using data returned from the server.

7.1. SIPCoin Roles

Clients are uniquely identified by their public key. There is no need for a certificate to be associated with the public/private key pair. Indeed, typically a single administrative entity - such as a telco operator - would have hundreds or thousands of clients, each with its unique public/private keypair. An administrative entity can create and destroy client instances at will, without any centralized configuration or provisioning.

Servers - typically run by, or co-resident with certificate authorities - are responsible for verification of ledger pages created by clients, and issuing of data needed by clients to construct burn receipts for coins that are verifiably burned on the ledger. The protocol puts the burden of storage of all ledger information entirely in the hands of clients, such that servers require a tiny amount of storage per client. Since servers are run by certificate authorities, their verification of ledger pages and issuance of data to construct burn receipts relies on their private keying material, trusted by all other actors.

7.2. Creation and Maintenance of the Self Ledger

Each client is responsible for maintenance of a ledger of its own create and burn transactions, the only two types of transactions permitted in the system. The ledger is broken into a series of pages. The client posts transactions into the current page of the ledger, called the active page. Each page starts with a page key, which is a hash of the prior page, forming a chain. Following the hash are a series of transactions. The pages prior to the active one

will all - through the LDP protocol - be signed by the server. These pages are called closed pages, and the server's signature over the page forms the final element in a closed page. The client is responsible for storing the prior pages in the ledger persistently.

Clients do not need to maintain prior pages indefinitely. Recall that each page is composed of a series of create and burn transactions. For a particular page, a client can delete a page from storage when all of the following conditions are met:

1. All the prior pages have been deleted

2. All of the create transactions in the page have been burnt in a subsequent page which has been closed

3. All of the Create transactions in the page have a subsequent Create transaction in a page which has been closed

In essence, the client maintains a sliding window of pages, with the tail being the current active page, and the head being the newest page that still contains an unburnt coin or Create transaction that formed the seed of the hash for the current, in-progress one.

The client is required to maintain these pages because they will need to be presented to the server to sign the current page, transitioning it from active to closed.

If a client should lose its pages, it forfeits any coin which it may have created. This is a significant difference compared to traditional Bitcoin, which uses a distributed storage system to provide a global ledger based on consensus, shared by all participants. In SIPCoin, there are many parallel ledgers, and each is stored locally only to that participant. This also means that all partiicpants in SIPCoin can mine coins; it is not a competition. Competitive mining favors the largest and most invested players, preventing others from being able to mine at all, in some cases. Since it is not possible to transfer SIPCoin, such a situation would mean that a SIP entity might not be able to place a call since it never won a lottery.

When a new client is created by an administrative entity, it needs to begin a new ledger. Each ledger and ledger page must be unique, ensureing that the proof of work transactions on one ledger cannot be copied into any other ledger. To create a new ledger, the client transacts with the server to obtain a first page. The first page is signed by the server - like all other pages. However, unlike subsequent pages, it contains no transcations - just a page key. The

server will choose a crypto-random value for the page key, ensuring that no two ledger pages start with the same value.

7.3. Transaction Types

SIPCoin supports only two types of transactions that can be placed into the ledger. These are the create transaction and the burn transaction.

7.3.1. Create Transaction

The create transaction is composed of the following elements:

1. The challenge. This is a number that forms the seed of the hashing. For the first transaction in a page, the challenge is equal to the page key. For all subsequent create transactions, the challenge is a hash of the prior Create transaction in the ledger.

2. The solution. This is a number which demonstrates that the proof of work has been done. Each proof of work is a hash function Ht() which takes as input two numbers, and returns a hashed result. The proof is demonstrated by providing a value S for the solution which, when hashed with the challenge C, forms a result H(S,C) which has N_Zero consecutive zeroes in the result. N_Zero is a global configuration parameter, and is discussed in more detail later on. Its adjustment is a principle part of the governance of the operation of SIPCoin.

3. The Coin ID: This is computed by the client as a hash over its public key, the challenge, and the solution. It serves as a unique identifier for the Coin produced by this create transaction.

7.3.2. Burn Transaction

The Burn transaction is created by the client when it wishes to place a SIP call. Consequently, each burn transaction is bound with a SIP INVITE. To perform this linkage, the burn transaction is composed of the Coin ID (obtained from a prior create transaction for an unspent coin) along with a hash over several fields of the SIP INVITE. The fields incude the From, To, Call-ID and fields from the SDP, such as media encryption keys. The hash also includes the timestamp for the burn transaction.

Beacuse the burn transaction is a hash over these various parameters, when it is sent to the server for signature, the server has no way to invert the hash. Consequently, the server learns nothing about the

originator of the call, the recipient of the call, the type of media in the call, or anything else. All that the server learns is that a call was placed, and that it was placed by the administrative entity that has a relationship with the server. This does mean that servers, through the observation of burn transaction rates, will know the call volume being emitted by the entity, but thats it.

The SIP agent running the client will not be able to send the SIP INVITE until it has received a burn receipt from the server. In essence, it needs to hold the INVITE until the ledger page is complete. For this reason, in SIPCoin, ledger pages close very fast. A client can post a ledger page for closure at a frequency on the order of one every 250ms to 500ms.

7.4. Closing Ledger Pages

A client closes the active ledger page when one of two conditions is met:

1. The ledger page contains N_trans transactions in it

2. The client requires a burn receipt for a burn transaction on the page, and it has not posted a ledger to the server within the last T_min seconds

A client is not required to close a ledger every T_min seconds; if it has no pending burn transactions in the ledger (only creates), it can wait. T_min specifies the minimum interval, and it is nominally enforced on the server to ensure the server is not overloaded.

To actually close the page, the client signs the active page with its public key, and then transmits the active page to the server, along with the public key. The first time it closes a page, it will also need to post all closed pages to the server. The server will validate the transactions in the current page, including insuring that the client has not double burnt the same coin. That particular check requires the server to have all active pages for the client, which is why they must be sent.

Once the server performs its checks, it will send back a signed version of the page, closing it. This enables the client to start a new active page in the ledger. The server also returns a signature over the now-closed page, using its trusted certificate.

The server also returns a signed hash, described below, that allows the client to compute burn receipts for each SIPCoin that was burned.

7.5. Server Validation

The server follows a standardized process for validating the page submitted by the client. At a high level, it composes the following steps:

1. The server authenticates the client; typically this is done using an administrative credential for the administrative entity responsible for the client. [[NOTE: Use ACME techniques for this??]]. LVP technically speaking does not require the server to actually authenticate the client if it chooses not to.

2. The server checks the signature on all pages sent by the client to ensure that they have been signed by itself.

3. The server validates that the pages form a sequential chain. It starts at the first page, computes it hash, and ensures that the result matches the page key of the subsequent page.

4. The server keeps stored, for each unique client (as indexed by public key), the hash of the most recently signed active page from that client, thus closing it. It checks that the active page that is to be signed is the successor, by comparing the page key in the active page to the stored value. This prevents malicious clients from forking the ledger and placing the same burn transaction, but for different INVITEs, into each fork.

5. The server examines every burn transaction in all pages sent by the server, and makes sure it matches exactly one create transaction. This ensures that the server has received all pages from the client (omission of a page from the client would enable it to double burn).

6. The server processes the transactions in order in the active page which is to be signed. If a transaction is ia create transaction, it verifies that the challenge is either the page key (for the first ever Create transaction) or the hash of the prior Create transaction in the ledger otherwise. The server stores, indexed by the public key of the client, the hash of the most recent Create transaction. It verifies this Create transaction has used that value as the challenge. It then takes H(), and uses it with the challenge and solution values. It verifies that the result has N_zero consecutive zeros. It then hashes the client public key with the challenge and solution, and makes sure it matches the Coin ID. If the transaction is a burn transaction, the server takes the CoinID and searches through all burn transactions in all pages sent by the client, and makes sure it doesnt match the Coin ID in any other burn transaction.

Once these validation steps pass, the server generates a signature over the active page using its certificate. It then stores the hash of this closed page to enable it to validate the next one, and stores the hash of the last Create transaction in the page to validate the next Create transaction.

To enable the client to create and send burn receipts, the server computes a balanced binary merkle tree, where the leaf nodes in the tree represent the Burn transactions from the page which was just closed. The head of the merkle tree is the signed by the CA with its private key. The signed head is returned to the client, along with the signed page that was just closed.

For purposes of performance optimization, the server can elect the cache the inactive pages, avoiding the need for the client to resend them each time. To do that, the server stores the pages and generates a cache key, which is an opaque parameter chosen by the server. The client, in subsequent validation requests, can include this key. It can then be used by the server to route those requests to the server instance which is holding the cache, and then used to extract the cached pages indexed by that key. If the server has a cache miss, it can reject the request and force the client to resubmit all its inactive pages.

7.6. Constructing Burn Receipts

To construct burn receipts, the client computes the merkle tree identically to the algorithm used by the server. It then verifes the signature over the head. This will normally be valid, since the CA is trusted in this architecture. The burn receipt for a SIPCoin is a digital object composed of:

1. All of the nodes in the merkle tree, starting at the leaf for the burn transaction for the coin in question, to the head of the tree.

2. For each node in the list above, the sibling of that node.

3. The signature over the head, as provided by the server.

This object is readily verified by having the receiving call agent hash upwards through the merkle tree and compare the result against the signature on the head. This burn receipt is included in the SIP INVITE. The usage of a merkle tree reduces the number of signing operations at the CA and also reduces the amount of data that must be transferred back to the client.

8. Usage of SIPCoin with SIP

The usage of SIPCOin with SIP is relatively straightforward. We say that a "SIPCoin is included in the INVITE" when the INVITE includes a Burn receipt for that coin; in this architecture coins are not actually transfer, only proof of their destruction. SIPCoins can be included in a SIP INVITE proactively with a Burn receipt, or they can be inserted reactively at request of the receiving agent. Its easiest to understand through the reactive flow.

The generating agent sends an INVITE normally, without any SIPCoin in it. This arrives at the receiving agent. Ideally, the receiving agent will verify the caller ID (see [draft-rosenberg-stir-callback] for a solution to enable this to occur). Once verified, the receiving agent checks whether the caller is known to be acceptable to the called party. The definition of acceptable is a matter of local policy and depends on the physical entities performing the receiving agent role, as discussed below.

If the caller is acceptable, the call is passed to the called party. If the nature of the caller is unknown (which is again a matter of local policy), the receiving agent rejects the INVITE with a response code 4xx which challenges for SIPCoin in order to accept the call.

When this is received at the generating agent, it constructs a new INVITE, burns a coin, constructs the burn receipt, and places those into the INVITE. This passes to the same receiving agent. If the caller ID is verified (whcih would have been done from the prior step) and it continues to be unknown, the receiving agent validates the burn receipt.

To validate it, the receiving agent performs the hashing through the merkle tree and verifies the signature on the hash at the top. The certificate verification requires the generating and calling agents to share a common trust anchor. This specification mandates that all agents trust the same set of CAs present in the Mozilla Firefox browser. This allows SIPCoin to be rooted in a well vetted, continuously maintained set of trust anchors which is proven to work globally.

If the signature is valid, the receiving agent considers the burnt coin as a sufficient proof of work to allow the call to proceed to the called party.

In the proactive model, which can be used by the caller to speed up call setup if they desire, they burn a SIPCoin prior to the challenge and include it in the INVITE straight away.

9. Deployment Considerations

There are many ways in which SIPCoin can be used. And in fact, the hardest part of rolling out a solution like SIPCoin is handling the intermediate states where it is only partially deployed on the Internet. This document proposes a phased rollout where each step is motivated by economic benefit to the parties at hand.

9.1. Enterprise SIP Trunks

The easiest deployment topology, and the best way to start, is on SIP trunks between a customer and their provider. In this model, the generating agent is that of the administrative entity which is using the SIP trunk, and the receiving agent is that of the provider. These are adjacent agents connected by a single SIP hop. As an example, the generating agent could be an enterprise, and the receiving agent would be a traditional telco offering enterprise SIP trunks. This would also be combined with the reverse role, where the service provider also runs a generating agent and the enterprise runs a receiving agent.

This arrangement provides a value proposition for the enterprise to protect itself from inbound spam calls which are received through their SIP trunk provider. If the spammer is another enterprise customer of the same provider, that enterprise becomes disincented from spamming due to costs. If the spammer is farther away - and in this phase they are most likely to be - the SP eats the cost and genreates the SIPCoin.

In such a service model, the service provider would - through its bilateral relationships with its customers, insist its customers implement the Outbound SIP Trunk role. As a result, the service provider itself would not need to generate SIPCoin for intra-provider calls. However, it would genreate them for inter-provider calls. This provides a benefit to the enterprise, who are now protected from spammers connected to the same SP, and the fact that the SP creates and burns calls for transit calls means that the enterprise gets the benefit of only ever accepting inbound calls which have SIPCoins burned.

In this model, the SP can save itself money in one of two ways. Firstly is through whitelisting. As part of the SIP trunk specification, enterprises on the receiving side should maintain a database of callers they 'trust'. A caller ID is trusted if the caller ID has been verified [draft-rosenberg-stir-callback], and the enterprise had previously, in the last few weeks, placed multiple calls to that number, those calls having connected and had a duration of at least a few minutes. This provides a simple model of: I'll

trust your inbound call if I've called you previously. The enterprise PBX can also use contact lists from employees contianing phone numbers to populate this list.

This means the SP cost is reduced for trusted callers, and not for others. To further reduce costs, the SPs are incented therefore to establish bilateral peering with each other over Inter-carrier trunks.

9.2. Inter-Carrier Trunks

These work identically to the enterprise SIP trunks; the carriers on each side of an inter-carrier peering link implement both the generating and terminating roles of the call agents. When a terminating enterprise challenges its SP for a coin, if the call arrived via an inbound trunk from another carrier, the SP can propagate the request for a coin upstream to save itself costs. If the upstream provider doesnt support SIPCoin, the SP must burn the coin itself, creating costs, and thus incentive for each side to insist on implementation to reduce costs.

In this way, SIPCoin implementations propagate outwrads, ultimately reaching the originating carriers for consumer services and enterprises. This brings us to the final phases.

9.3. Consumer provider to Mobile Phone

This specification recommends that the terminating role be implemented in smartphones implementing the IMS specifications. Consider now an enterprise which placed a call towards a consumer mobile phone. This call is received at the terminating mobile provider. Since it knows that the mobile callee SIP UA supports SIPCoin (from the SUpported header field in the REGISTER), it propagates the INVITE towards the called phone after verifying the caller ID. The callee, seeing that the caller ID is verified, checks its local contact list. If the caller is on the contact list, it doesnt challenge for coin. If it isnt, it challenges for the coin. This propagates all the way back to the originating enterprise, which burns a coin to place the call, which is then accepted by the callee.

The generting role is not appropriate for implementation on mobile phones, and as such the consumer mobile operator cannot pass its costs upstream. However, as part of bilateral peering arrangements and standards coordination, the SP can insist that each other require their mobile phones to comply with the specs that mandate implementation of the terminating role. That will save each other money in proportion to the balance of their inbound to outbound calls.

This then provides the final economic incentive to achieve the target architectural model.

9.4. Target Model

In the idealized model, the terminating role is implemented by the receiving phones, and the generating role implemented by the call agents operating on their behalf. The entire SIP core network supports these roles, but as this target deployment architecture is reached, they never need to generate or verify SIPCoin since it is fully handled e2e. This minimize cost for all parties and concentrates it on the entites generating calls to numbers which are never called back, and not on the contact lists of mobile phones.

10. Governance

In order for SIPCoin to be an effective tool against spammers, it requires ongoing governance. This governance takes three forms:

1. Updating of this specification

2. Periodic adjustment of the value of N_Zero

The first of these is fairly routine for the IETF, but new for cryptocurrencies, which rely on distribued consensus amongst majority implementations. SIPCoin is more managed than those networks, and as such we propose the IETF, in essence, manage the behavior of the system through the published RFC.

The second of these is more interesting. In order to deal with changes in the cost of computation over time, it is necessary to adjust the value of N_Zero periodically. This specification suggests that the IETF consensus process be used for this purpose. To speed up implementation, the value of N_Zero must be loaded dynamically by all clients and servers from an IETF maintained and verified website. This allows IETF governance to decide on a new value, and for that new value to be used instantly across the entirety of the SIP based telephone network.

11. Economic Analysis and Parameter Tuning

11.1. Cost Targets

The goal of SIPCoin is to incur cost to callers, in such a way that it erodes the profitability of the spammers to the point of making it no longer viable, and, at the same time, representing only a small increase in the cost to legitimate callers. This represents an operating window in which the system needs to operate.

Let us first consider the tolerable costs to legitimate callers. In most cases we anticipate the costs to be borne by the service providers, and then passed on to consumers or perhaps absorbed if the costs do not merit it. Its important to point out that the cost of SIPCoin is metered per call regardless of destination or duration of call. This tends to penalize entities that make many short calls (as telemarketers do) while benefit those who make fewer, long, international calls (which is more typical of users paying high costs today to call friends and family abroad).

As a back of the envelope analysis - the average phone bill in the U.S. is approximately $100 for a mobile phone each month. According to [PR Newswire][<https://www.prnewswire.com/news-releases/no-time-to-talk-americans-sendingreceiving-five-times-as-many-texts-compared-to-phone-calls-each-day-according-to-new-report-300056023.html>], the average American makes or answers six phone calls per day. Assuming this is symmetric, thats 3 placed calls per day, 90 per month. With a three percent increase in their bill as an upper bound, this means $3 per month, or 3 cents per call.

On the other side of the house - the spammers. Its hard to get precise data - but here is a back of the envelope. A recent [Boston Globe article][<https://www.bostonglobe.com/ideas/2017/05/11/the-onslaught-spam-calls-will-keep-getting-worse/2w1tyrSnzEj8NPO81hUUBK/story.html>] cites that in the US, 2.5B robocalls were placed in the US in April of 2017. Later in the article, it quotes a cost to Americans of $350 million between 2011 and 2013. If we assume this translates directly to the profits of the spammers, over that 36 month period thats $9.7M profit per month. If it took 2.5B robocalls per month to achieve that profit, that is a profit of 0.38 cents per call.

This means there is - on the surface - a viable operating point here. Assuming a 50% erosion in profit is enough to make a dent in telemarketing, our lower bound on the cost of SIPCoin is 0.19 cents per call, and our upper bound is 3 cents per call. This represents an order of magnitude spread. That is without consideration to the addition of whitelists.

When combined with the whitelist and verified caller ID, we can signicantly shift the cost to the spammers. As a back of the envelope, costs are incurred to non-spammers when a user makes a call to a number that the user has never received a call from nor is on the contact list of the callee. There are real use cases for this - a call to a contact center is one such case. Another is a call to a new contact number learned via business card or personal introduction. These are, relativey few. If we assume that, of the 100 or so calls made each month perhaps one is like that, this adds

another two order of magnitude to the spread, resulting in a three order of magnitude improvement. This means that, as long as we can keep the economics of calling such that it is not three times cheaper for a spammer than an SP to mine SIPCoin, the system can be effective.

11.2. Impact of Compute Variability

The hardest challenge in building a system that operates in the cost targets is dealing with the highly variable costs of computation. To give some perspective on this, a somewhat dated article on Bitcoin compute costs [<https://en.bitcoin.it/wiki/Non-specialized_hardware_comparison>] shows a spread of three orders of magnitude in hashing performance across a range of Intel CPUs (from 0.245 Mhash/s (million hashes per second), up to 140 million). It cites the performance of GPUs as sitting in a range from 1 MHash/s up to 2568 MHash/s, and quotes ASICs as being able to reach 1000 GHash/s (Billion hashes per second). The performance spread is therefore seven orders of magnitude. Though there is surely a spread in cost as well, it is assuredly not as large. This means that in SIPCoin, the spammers will be incentivized to buy high performance compute which is viable economically only at high scale.

However, considering the deployment architecture described above, the generating role is implemented by enterprises that have SIP trunks to their carriers, and the carriers. The low end computational devices - mobile phones - actually delegate their generating role to the call agent acting on their behalf. If we imagine that small home networks and small businesses would similarly delegate their generating role to their service provider, we end up in a model where trust relationships primarily put the burden of computation on larger entities, which can in general afford to just all use ASICs, which can eliminate the disparity between the spammers and the good guys.

In other words, if the spammer can afford some ASIC-based machines, Verizon can too.

11.3. Load Analysis on the CAs

This proposal introduces a new role to be played by a CA, in the verification of SIPCoin ledgers. This process is, fortunately, almost stateless, requiring a query for just two hash value indexed by a public key. There are no user records, payment systems, cryptographic storage (beyond what they already implement). However, it is extremely high volume.

Assume a large carrier is about 100 million subscribers. Assume that they do an average of about 10 calls attempts per day per user.

Assume volume at peak is 3x average (ignoring things like earthquakes in California). For calculation purposes, lets say we we are closing ledger ever 0.5 seconds. That gives us (100,000,000 * 10 * 3 / 246060 / 0.5) = 70,000 entries per close in busy case. Lets say our EC signature are 100 bytes and that a burn or create transaction fit in 256 bytes total and that a given page has about equal number of create and burn. This gives me that the CA, even it it only goes back a 2 pages, needs to look at 3 pages * 70,000 entries * 2 (for create and burn) * 256 bytes = 100 Mbyte each half second or about 1.6 Gbps.

Is this too much? Its a lot. But not out of the realm of reasonableness.

12. Alternative Consensus Techniques

The proposal here uses the CAs as trusted third parties to verify the ledger. This is owing to the challenges in achieving rapid consensus in large scale distributed blockchains. However, a variant on the proposal here is to elect randomly a small subset of the entities participating in bitcoin and require consensus only amongst a subset. The size of the subset needs to only be larger than twice the number of malicious entities we wish to tolerate. One can argue that the incentives for being malicious in SIPCoin are smaller (just spammers), perhaps they only represent 5% of call agents in the network (whcih would be a lot!). So we only need 10% of the nodes for consensus.

If the set of elected nodes can be small, and they are very well connected to each other, we can run full-mesh consensus protocols which are potenitally fast enough to achieve consensus and sign results and then distribute them at a speed which meets the requirements here. These elected agents would exactly implement the server side role of LVP, and validation is by looking at consensus view rather than verifying signatures.

13. Security Considerations

There are many attacks possible in this system. THe primary ones to prevent are the clients acting maliciously in order to either create additional SIPCoin without doing the hashing work, or use the same SIPCoin for multiple SIP INVITEs. We consider both forms of attack.

13.1. Creating Additional SIPCoin

A client might maliciously obtain a SIPCoin from another client in some way (perhaps eavesdropping or theft of databaase), and then use it for itself. However, it cannot do that. Since the challenge in

the SIPCoin is bound to the ledger in which it lies, by using the page key, and then the page key is linked to the entire ledger chain for the same client, it is not possible to insert SIPCoins into different ledgers.

A client might try and perform the hashing and then insert the same SIPCoin twice into the same ledger page. However, this is not possible because the server will confirm each Create transaction derives from a unique predecssor. In a similar way, a client might try to insert the same create transaction into two different ledgers. Since the server maintains an index of the most recent Create transaction, it would detect this.

13.2. Burning a SIPCoin Multiple Times

One way in which a client might try and burn the same coin twice is to literally have the same burn transaction reference the same coin in its sequential ledger chain. This is prevented through the core validation steps performed by server, which looks for such duplicates.

Another way in which a client might try and burn the same coin twice is to fork the ledger, and put the same Burn event in different pages. This is prevented because the server will verify and then sign the first such forked page presented to it. When it does, the server basically advances the pointer it maintains to the most recently closed page in the ledger. When the client tries to fool the server into verifying the second fork, the server will reject it because the currently active page is not the direct descendant of the previously closed page. Thus, the client can only maintain a single, sequential ledger.

THe client might try and use the same Burn Receipt in two different SIP transactions. This is not possible, because the Burn receipt includes a hash over the fields in the INVITE which cannot be duplicated by the call agent without for differnt calls - the called party and timestamp. Narrow timestamp windows (say, 2 seconds), prevent even calls to the same number with the same Call-ID within that window.

A client might try and take burn receipts from INVITEs it reuses, and replay them in different INVITEs. The binding of the burn receipt to the called user prevents this.

[[TODO: lot more rigor needed here]]

14. IANA Considerations

TODO

15. Acknowledgments

Many thanks to Ram Jagadeesan and Richard Barnes for their input.

16. References

16.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

16.2. Informative References

[draft-rosenberg-stir-callback] Rosenberg, J. and C. Jennings, "Bootstrapping STIR Deployments with Self-Signed Certs and Callbacks", March 2018, <https://tools.ietf.org/html/draft-rosenberg-stir-callback-00>.

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, DOI 10.17487/RFC3261, June 2002, <https://www.rfc-editor.org/info/rfc3261>.

[RFC5039] Rosenberg, J. and C. Jennings, "The Session Initiation Protocol (SIP) and Spam", RFC 5039, DOI 10.17487/RFC5039, January 2008, <https://www.rfc-editor.org/info/rfc5039>.

Authors' Addresses

Jonathan Rosenberg
Cisco Systems

Email: jdrosen@jdrosen.net

Cullen Jennings
Cisco Systems

Email: fluffy@iii.ca