Workgroup: Mimi
Internet-Draft: draft-rosenberg-mimi-spin-00
Published: 24 October 2022
Intended Status: Standards Track
Expires: 27 April 2023
Authors: J. Rosenberg C. Jennings A. Cooper J. Peterson
Five9 Cisco Cisco Neustar

Simple Protocol for Inviting Numbers (SPIN)

Abstract

This document introduces a framework and a protocol for facilitating voice, video and messaging interoperability between application providers. This work is motivated by the recent passage of regulation in the European Union - the Digital Markets Act (DMA) - which, amongst many other provisions, requires that vendors of applications with a large number of users enable interoperability with applications made by other vendors. While such interoperability is broadly present within the public switched telephone network, it is not yet commonplace between over-the-top applications, such as Facetime, WhatsApp, and Facebook Messenger. This document specifically defines the Simple Protocol for Inviting Numbers (SPIN) which is used to deliver invitations to mobile phone numbers that can bootstrap subsequent communications over the Internet.

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 27 April 2023.

Table of Contents

1. Introduction

Voice, video and messaging today is commonplace on the Internet, enabled by two distinct classes of software. The first are those provided by telecommunications carriers that make heavy use of standards, such as the Session Initiation Protocol (SIP) [RFC3261]. In this approach - which we call the telco model - there is interoperability between different telcos, but the set of features and functionality is limited by the rate of definition and adoption of standards, often measured in years or decades. The second model - the app model - allows a single entity to offer an application, delivering both the server side software and its corresponding client-side software. The client-side software is delivered either as a web application, or as a mobile application through a mobile operating system app store. The app model has proven incredibly successful by any measure. It trades off interoperability for innovation and velocity.

The downside of the loss of interoperability is that entry into the market place by new providers is difficult. Applications like WhatsApp, Facebook Messenger, and Facetime, have user bases numbering in the hundreds of millions to billions of users. Any new application cannot connect with these user bases, requiring the vendor of the new app to bootstrap its own network effects.

This situation has recently drawn the attention of regulators, and was one of the motivations behind the Digital Markets Act (DMA) in the European Union. Amongst its many provisions, it requires vendors of large communications platforms to enable interoperability with third party vendors. It does not, of course, specify an actual set of protocols or technologies for enabling that interoperability.

This document seeks to fill that void, by defining a framework - the SPIN Framework - for such interoperability. This framework seeks to strike a balance between innovation and standardization, by identifying only those portions of the protocol stack that must be standardized in order to achieve end-to-end security for a minimum feature set between providers, and leaving everything else to APIs and protocols which each vendor can define on it's own.

This framework identifies the need for a new protocol to solve the identity mapping problem - the SPIN Protocol. Specifically, how does an originating user using one application identify a target user in a different application with which they wish to communicate, and then obtain an identifier for the target user in the target application that is utilized by that target user? Consider the following example. User Alice is a user of Facebook Messenger, and wishes to send a 1-1 chat message to her friend Bob. Bob is a user of a different application for messaging - Signal for example - but this fact is not known to Alice. Alice needs to somehow obtain a URI that can be used to send messages to the Signal application targeted at Bob. This is the identity mapping problem, and is addressed by the SPIN protocol defined here.

2. Implications of no Standards Action

In theory the application interoperability envisioned in the DMA could be achieved entirely through the publication of vendor-specific APIs and without standardization. However, this would yield a suboptimal outcome for both users and app developers, as supporting the matrix of pairwise communication flows between all of the affected voice, video, and messaging applications in the market via vendor-specific APIs will create a patchwork of inconsistent user experiences and likely lead to buggy implementations. Using a minimal standardized framework to bootstrap cross-app commmunications will provide more consistency while leaving app developers freedom to continue to make their own design choices.

Furthermore, the usage of a standards-based solution ensures that end-to-end messaging, voice, and video can happen between providers. Without a standard, each vendor subject to the DMA will publish APIs for access to their services. These APIs have traditionally provided access to messages, voice and video that are not protected by e2e crypto. While it is possible, in theory, that each application

provider could amend their APIs to provide access to e2e encrypted content, doing so without an agreed-upon standard will almost certainly lead to third parties decrypting in the cloud to avoid implementing N variations in each client, one for each provider they interop with.

3. Affected Actors

The solution defined by the SPIN framework requires participation from multiple actors, and thus requires coordination through standards. These actors are:

* Mobile OS Vendors: Most notable Apple and Google. It requires them to implement new APIs in their operating systems, new user preference capabilities, and support for user identity through certificates.

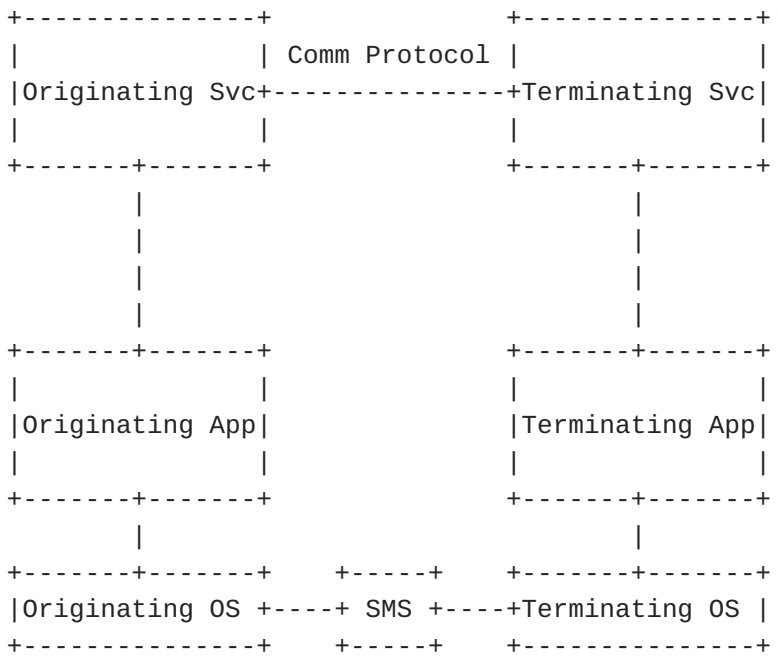
* App Developers: App developers, such as a Signal or Facebook Messenger, are required to change. They are required to utilize the APIs exposed by the mobile OSs, and also implement the voice, video and/or chat protocols specified by the SPIN Framework.

* STIR/SHAKEN PA/CA: The SPIN framework suggests that it be possible for the mobile OS vendors to generate STIR certificates for the device. This requires that these vendors be supported as valid CAs for STIR.

Note that the SPIN Framework described here does not require any support or changes from the carriers themselves (Note however, the open issue discussed below where we discuss an alternative certification model where the telcos perform delegation to the mobile OS vendors to install a cert on the phone).

4. SPIN Framework

The framework for SPIN is shown in the figure below:



In the framework, we have two users - the originating and terminating. The originating user wishes to send a message, make a video call, or make a voice call, to the terminating user. A fundamental assumption of SPIN is that the originating and terminating users are both identifiable by telephone numbers on the Public Switched Telephone Network (PSTN), and that the terminating user can be reached via SMS. The originating user knows the telephone number for the terminating user. The originating user is using an app running on an operating system. The operating system can be a mobile OS, such as iOS or Android. The originating OS exposes APIs towards the application, which allow the originating app to request communication to a user with the specified number. The originating app is associated with a service running on the Internet, and can connect to it for communications services. There is a similar setup on the terminating side - the user has an application running on an operating system which can receive SMS messages, and their app is associated with a service reachable over the Internet.

The role of the operating systems in this framework is to act as a trust anchor. The OS is responsible for authenticating the applications and vetting their behaviors, as they normally do on mobile OSs.

The goal of the SPIN protocol is to allow a user of the originating app to select a service (voice, video or messaging), and select a phone number to which they communicate, and then receive a URI which corresponds to the terminating service which can be used to perform

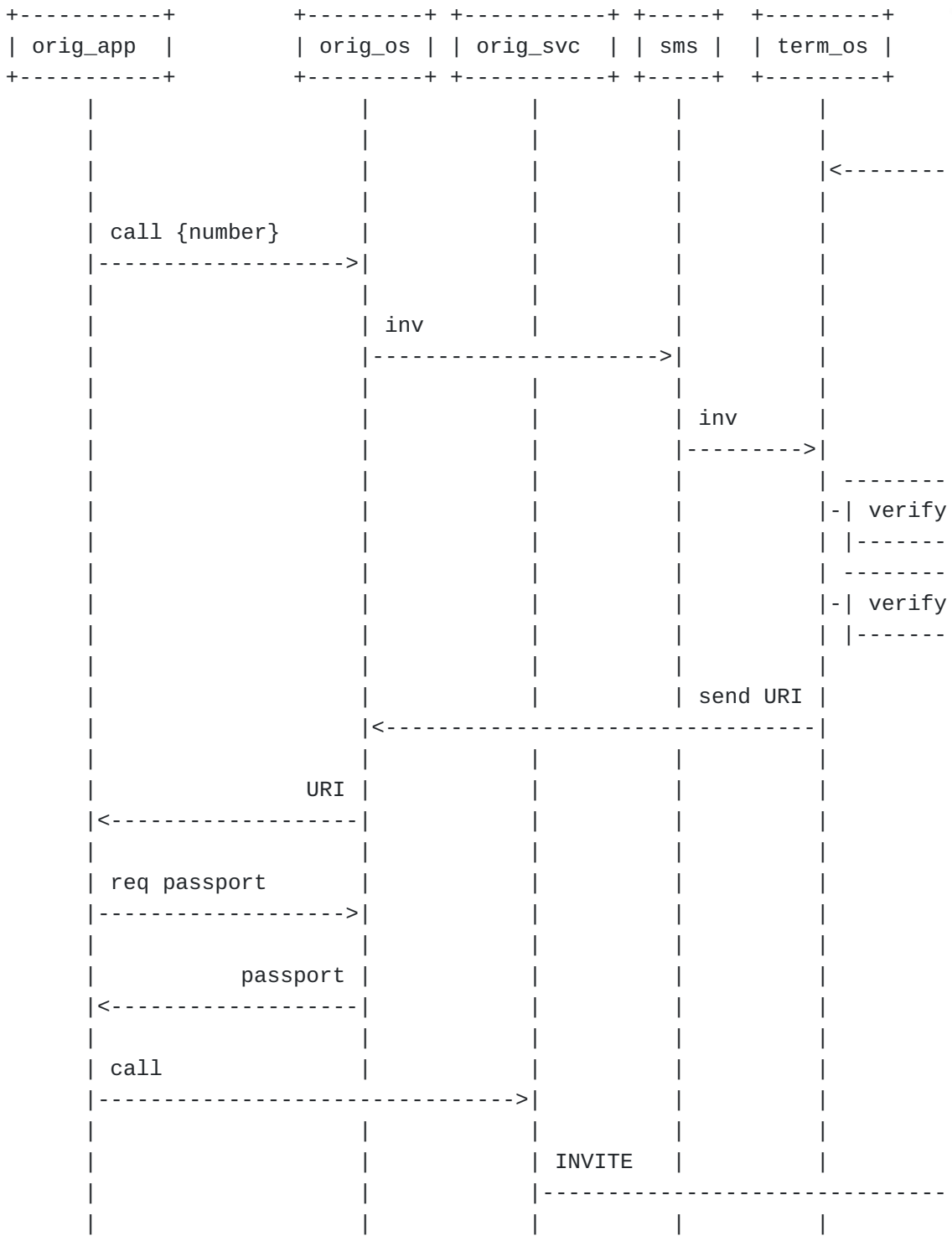
that communication. The URIs of course correspond to protocols for that form of communication.

Once the SPIN Protocol has run, the originating service now has a protocol URI for the particular media type - voice, video or chat, and can initiate it towards the terminating service. The SPIN Framework recommends specific protocols for voice, video and chat. For voice and video, the SPIN Framework suggests SIP [RFC3261], with [I-D.rosenberg-dispatch-cloudsip], [RFC8224] and the webRTC media stack. For messaging, it suggests creation of a new REST-based protocol for 1-1 messaging, including e2e encryption using STIR-based certificates, and features such as delivery and read receipts, emojis, stickers, reactions, threads, images, URLs, contacts, and so on, forming a baseline set of minimum viable 1-1 messaging. For the initial phase of SPIN, group communications would be out of scope.

Though the framework is expressed in terms that align with mobile operating systems, the same framework can apply in other cases. For example, the terminating service, app and OS can logically be a single entity. As an example, the terminating service, app and OS could be associated with a Contact Center as a Service (CCaaS) provider. In that setup, the SMS messages are delivered directly to the CCaaS provider, and there is not a mobile operating system involved to receive them.

5. SPIN Protocol Overview

The behavior of the SPIN Protocol is best understood through a high level sequence diagram:



On the terminating side, the terminating user at some point installs an application which is capable of handling communications for one or more media types (voice, video or messaging). The application will register with the terminating OS, using APIs exposed in the OS, that it is capable of acting as a SPIN handler. As part of the

registration, the application provides the OS with a URI for the service it provides of that media type. As discussed below, this can be a proprietary API, or can be a baseline standard protocol. In the case of voice, that baseline standard is SIP, and in particular, cloud SIP [I-D.rosenberg-dispatch-cloudsip].

Later on, a user in an originating application decides to place a call to a number. The originating application does not have a user with that number as part of its own service, so it knows it needs to use SPIN to route the call. It goes to the operating system on the mobile phone, and requests it to provide a URI for voice communications to the specified phone number. The originating OS then prepares an SPINvitation object. This is a JWT which contains several fields. THe fields include the phone number of the originating user (which must be known and verified by the mobile OS), and an HTTP URI that can be used by the terminating OS to send the results back, and the communications service that is requested. This HTTP URI will normally contain an embedded Authorization header field that contains a short-lived token, valid to send the results back. It then signs the JWT and sends an SMS (more likely, an MMS given the size of the signed object), to the target user's phone number. The terminating OS receives the SMS/MMS, and notices that it contains an SPINvitation object, and thus should not be rendered to the user. Should the terminating user and its OS not support this protocol, it will end up rendering the MMS. The MMS includes some plain text, which can be rendered to the user, indicating that the caller wishes to speak with them, so that the human user can take some action (like a return voice call over the PSTN).

Assuming the terminating OS supports this protocol, the MMS is absorbed and decoded. THe signature is verified and then the communications service is obtained. In this example use case, it is for a voice call. The terminating OS has an application that has registered itself as a handler for voice. Note that, the terminating user might have multiple applications on their OS which can act as handlers for voice. In such a case, the mobile OS would offer the user a configuration setting to choose one as a default.

The app had previously registered itself as a handler and provided a SIP URI for the receipt of calls, something like sip:{number}@provider.com. This URI is sent back to the originating OS. Rather than sending this back via SMS/MMS, IP communications are used. The invitation object contained an HTTP URI which can be used by the terminating OS to send the SIP URI. The SPIN protocol defines the exact syntax and semantics of this HTTP POST operation. This is received by the originating OS, which then informs the app that it was able to locate the user. The originating OS provides the communications URI (in this case, a SIP URI for voice calls).

Next - the originating app places a SIP call. Because we are now dealing with inter-domain and inter-provider calls, secure caller ID is required. SPIN requires that STIR passports [RFC8225] are included, sent using [RFC8224]. The originating OS is required to obtain a passport that is valid for the originating user. In this framework, this is done by virtue of the mobile OS having a certificate by which it can perform the signing operation directly.

There are two ways in which the originating OS can obtain such a certificate. In one approach, the mobile OS would perform SMS verification (again, invisibly, by absorbing the SMS it sends to itself), and add an additional check of comparing it agaisnt the mobile numnber the user claimed they owned during provisioning time of the device. The mobile OS vendor would be a valid CA, and then generte a certificate valid for that individual phone number. In an alternative model, the telco uses certificate delegation [RFC9060], and generates a certificate that is handed to the phone during device provisioning. The latter approach is more secure in some ways (as it would no longer depend on SMS forward routability for authentication of a user), but is much harder to deploy.

The originating app makes an API call into the OS to obtain the passport, which is then returned to the app. The app uses its own app-specific protocols to communicate with its servers, and will send the passport and the terminating user's phone number to its service. Its service will then send a SIP INVITE to the target number, including the passport in the SIP Identity header field. From there, the terminating service can alert its app using the mobile OS push techniques, and a call has been placed.

The SPIN framework therefore consists of the following:

1. A standardized syntax for the SPINvitation object that can be sent via MMS

2. A standardized HTTP-based protocol for providing URIs for communications - the actual SPIN on the wire protocol

3. Recommendations for mobile OS vendors on APIs they should provide to enable SPIN, without actually specifying any details of what those APIs look like

4. Specifications for communications protocols needed for voice, video and messaging between app providers

6. SPINvitation Object Syntax

This will be a JWT that contains:

*The desired media type, one of an enumerated set

*An HTTP URI for a callback

Details TBD.

7. SPIN Protocol for Providing URIs

To be filled in

8. Mobile OS vendor API Recommendations

To be filled in

9. Specifications of Communications Protocols

There are several ways in which the communications protocols could be specified. On one extreme, the standard could leave this entirely up to the terminating provider to define its protocol or API and document it publically. It would then be the responsibility of the originating service to implement each of these APIs for every terminating provider it wishes to speak to. On the other extreme, we can fully specify a protocol - most likely with reference to existing standards.

SPIN tries to take a middle ground. It allows terminating providers to choose whether their interface is proprietary, or, whether it follows a minimum baseline protocol specified here.

9.1. Voice and Video

Because the communications are between providers that may not have previously had an established bilateral relationship, we want the communications to be possible without any kind of manual configuration. For this reason, SPIN specifies that the default voice and video communications protocol is SIP [RFC3261], along with it's extension for cloud SIP [I-D.rosenberg-dispatch-cloudsip], and it utilizes the media protocols standardized by webRTC. The usage of cloud SIP allows scalable, reliable, inter-provider SIP over the Internet, and the usage of the webRTC media stack provides a well-defined baseline media stack that is already widely implemented. The SIP messaging MUST utilize [RFC8224] to ensure secure user identity. Media between the originating and terminating service will be DTLS-SRTP by virtue of using webRTC, and e2e media encryption is supported and bootstrapped using a certificate bound to the user's phone numbers. The mobile OS would hold the STIR certificate, and allow the application to request a signature over the keying material for driving DTLS-SRTP.

Details to be filled out.

9.2. Messaging

For messaging, 1-1 messaging will be supported in the initial specification. All messages will be e2e encrypted, using the STIR certificate as well. A specification will be produced that defines a REST-based protocol for basic 1-1 messaging features, including read receipts, delivery notifications, typing indicators, images, videos, contact cards, and so on. A baseline set of capabilities would be provided, along with an extensibility framework for future content that would allow users to pop out to a browser in cases where some new content is added, that is not yet supported.

Details TBD.

10. Security Considerations

The SPIN protocol defined here is meant to address the following threats:

* A malicious application that "steals" incoming calls or chats against user wishes. To prevent this, this protocol enlists the mobile operating system as a trusted third party that governs dispatch of communication requests to the right application based on user preferences.

* A malicious application that spams target users with requests for communication. This is mitigated by enlisting the aid of the mobile operating system on the terminating side to absorb SMS's conforming to this specification, and not presenting them to the user. Digital signatures are used over the content of the SMS messages, and the terminating OS can validate that it trusts the sender before taking further action.

* Intermediates that eavesdrop on communications between app providers. This is mitigated by using e2e encryption across messaging, voice and video, ensuring it can be retained when crossing provider boundaries.

11. Normative References

[I-D.ietf-mls-architecture] Beurdouche, B., Rescorla, E., Omara, E., Inguva, S., Kwon, A., and A. Duric, "The Messaging Layer Security (MLS) Architecture", Work in Progress, Internet-Draft, draft-ietf-mls-architecture-09, 19 August 2022, <https://www.ietf.org/archive/id/draft-ietf-mls-architecture-09.txt>.

[I-D.rosenberg-dispatch-cloudsip] Rosenberg, J., Jennings, C., and T. Asveren, "SIP Extensions for High Availability and

Load Balancing for Public Cloud", Work in Progress, Internet-Draft, draft-rosenberg-dispatch-cloudsip-00, 21 February 2021, <https://www.ietf.org/archive/id/draft-rosenberg-dispatch-cloudsip-00.txt>.

[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>.

[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>.

[RFC8224] Peterson, J., Jennings, C., Rescorla, E., and C. Wendt, "Authenticated Identity Management in the Session Initiation Protocol (SIP)", RFC 8224, DOI 10.17487/RFC8224, February 2018, <https://www.rfc-editor.org/info/rfc8224>.

[RFC8225] Wendt, C. and J. Peterson, "PASSporT: Personal Assertion Token", RFC 8225, DOI 10.17487/RFC8225, February 2018, <https://www.rfc-editor.org/info/rfc8225>.

[RFC9060] Peterson, J., "Secure Telephone Identity Revisited (STIR) Certificate Delegation", RFC 9060, DOI 10.17487/RFC9060, September 2021, <https://www.rfc-editor.org/info/rfc9060>.

Authors' Addresses

Jonathan Rosenberg
Five9

Email: jdrosen@jdrosen.net

Cullen Jennings
Cisco

Email: fluffy@cisco.com

Alissa Cooper
Cisco

Email: alissa@cooperw.in

Jon Peterson

Neustar

Email: jon.peterson@neustar.biz