TABLE OF CONTENTS

LIST OF FIGURES

Page

INTRODUCTION

All individuals involved with digital information transfer are now aware of the implications of the lack of security in computer systems. A combination of early design decisions and commercial interests led to the release of a plethora of products flawed by poor security practises. A common approach to cyber threats has been to make it very cumbersome for users to access a computer system. This approach has taken various forms: long, complex and difficult to remember passwords; token-based security, where a trusted third party guarantees the authenticity of the parties and the integrity of the data being exchanged; geographical and IP-based access restriction, among others. The combination of these techniques makes it difficult for an attacker to access non-authorised information. However, this approach has also made it difficult for legitimate users to use resources they are entitled to. The academics, and a sector of the industry, have worked in the development of secured easy to use systems. However, many of the proposed systems are still highly complex for the regular user, requiring a level of user participation that is not easily achievable. In the last two decades, two technologies have gone through dynamic trends of development and investment:

- Biometric technologies. A way of identifying people by a combination of what they are and what they know (Jain *et al.*, 2016).
- SIP communications. A technology to standardise signalling between parties who need to exchange media information (Rosenberg *et al.*, 2002).

Biometric technologies have extensive security applications while SIP powers the development and enhancement of products, notably, those related to telephony over Internet Protocol networks.

Both of these fields have reached technological maturity. Biometric authentication is a proven and accepted way to recognise individuals and grant them access to systems of information

(Beranek, 2013). SIP is the accepted standard for establishing multimedia sessions (Sisalem *et al.*, 2013).

This thesis aims to contribute to the development of comprehensive, flexible, and secured systems, which are easy to implement and understand. This thesis argues that it is possible to leverage SIP properties to create a common session establishment protocol for sessions that require biometric authentication. By using an already trusted protocol, vendors can provide standard solutions to include biometric-based authentication, and access control, across network elements. They will know that their solution can be easily integrated with other elements of a system supporting the same standard. These solutions can be proven to be simpler for the end user. They benefit by accessing resources using their own biological properties as an authentication token. The approach could be as simple as associating a user's voice with a standard user identifier, which would require the user to speak in their normal voice or to repeat a simple predetermined phrase. An interaction of this kind is easier than typing a long and complicated password. An extension to the SIP protocol, called SIPBIO is proposed to handle the tasks of establishing a biometric session.

Chapter 1 reviews the history and development of using biometric methods in telecommunications networks. Several earlier proposals are identified and analysed in the context of their own time and for their contribution to most recent technological developments. Chapter 2 reviews the basic concepts of the SIP protocol in preparation for Chapter **??** which presents relevant SIP canonical extensions. The aim of this analysis is to reveal how SIP is extended in practice and how these previous extentions can be applied to the SIPBIO proposal. In Chapter 3 the evaluation methodology is explained in detail, starting with some test case scenarios. This information will lead to building SIPBIO requirements that serve as the base for the protocol construction later. Finally, the process of extending SIP is explained in detail.

Chapter 4 presents how SIPBIO can be used to handle different types of biometric processes, and the core of the protocol, its flows and SIP messages. Chapter 4 is the core of this proposal. The concepts are reinforced in Chapter 5 with a simulation of a protocol implementation.

Finally, the conclusions are presented along with suggestions for extended developments around the SIPBIO proposal.

This thesis is motivated by fifteen years of telecommunications experience in the field, working with customers in need of solutions, practical and easy to understand, for security and communication challenges.

CHAPTER 1

BIOMETRIC AUTHENTICATION IN TELECOMMUNICATION NETWORKS.

The idea of using biometric techniques to authenticate users in telecommunications has been around for many years as a theoretical, yet cumbersome to implement, possibility (Lapere and Johnson, 1997). Recent increases in computing power, data transmission speeds and the availability of affordable storage, now make viable the use of biometrics in enterprise and consumer-based telecommunications (Gafurov, 2010).

The definition of a *telecommunications network* derives from the concepts of *computer networks* and *distributed systems*. According to Tanenbaum and Wetherall (2011a), a *computer network* is a collection of autonomous computers (nodes) interconnected by a single technology and a *distributed system* is a collection of independent computers that appear to their users as a single coherent system. Consequently, a *telecommunications network* can be defined as a *distributed system* in which nodes are either computing entities or *computer networks* and offers services related to information sharing.

The above definition determines that the main objective of telecommunications is to share or allow access to information through or from any kind of voice or data network. Information can take any form, including documents, audio, voice and video. Some information is intended to be openly available, such as websites on the Internet. Other information is restricted to a single party like a personal bank account web site or phone line or to a specific group of people as with a corporate intranet. Different mechanisms have been developed to control access to shared resources, which use passive or active authentication by the user sharing the resources (Mallery, 2013).

When telephonic communications were controlled by public providers, security was maintained by their exclusive access to all hardware. As networks evolved to packet and mobile-based technologies access was controlled by a *personal identification number* (PIN) and, in the case of mobile telecommunications, a smart card in the form of a *Subscriber Identification*

Module (SIM). These solutions are convenient but lack security (Lapere and Johnson, 1997). Any telecommunications access control measure must meet the following requirements:

a. be simple to use yet effective enough to provide a noticeable level of security;

b. be measurable, recorded and quantifiable (Eur, 1997).

These are also characteristics of a biometric security system.

Biometric characteristics for authentication in telecommunications environments have been explored since the 1990s when the *European Telecommunications Standards Institute* (ETSI)[1], made it a priority to provide secure communications standards for UMTS. They stated, "without a reliable authentication service through the *Telecommunications Management Networks* (TMN), every other effort to secure the system is in vain" (Eur, 1997). In the 1990s, there were obstacles to use biometric authentication methods: sensors were costly, processing power was low, service provider charges for data transmission were very high, acceptance and use of the technology were challenging. In the particular case of voice biometrics, telephones already had audio capturing sensors and they were a familiar device, making them a viable and non-intrusive option.

1.1 Basic biometric architecture

Communication network biometric systems are known as *remote biometric authentication systems* (Syta *et al.*, 2015). A general architecture of a biometric system is shown in Figure 1.1.

The set of relevant elements shown in Figure 1.1 are described by Gafurov (2010):

- A **subject** from whom the biometric information is read.

[1] www.etsi.org

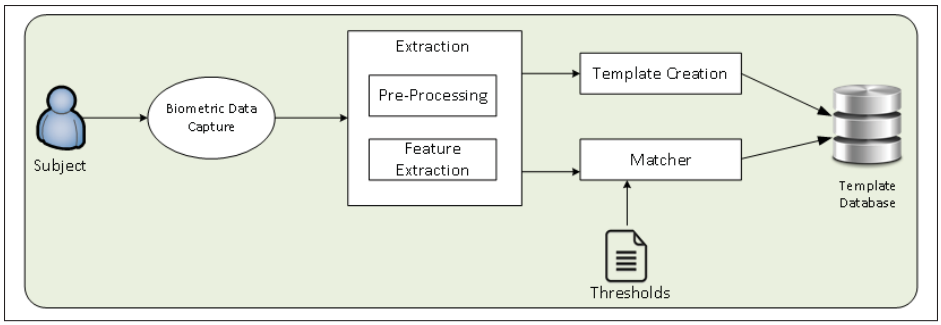


Figure 1.1 *Basic biometric process.*

- The **biometric data capturing** process. This is accomplished by one or more mechanical or electrical objects that capture raw data to be analysed. The usual objects are microphones, webcams, mobile phones (which have several capturing methods), fingerprint readers, keyboards, mouse devices. A capturing element does not have to be a device specifically designed for biometric operations. A *finger print reader* has been specially designed for biometric data capture whilst a telephone has not. In general, any device able to retrieve biometric data and retransmit it in a digital format can be considered a *capturing element*.

- ***Database:*** Once a template is created, it is stored in a database. During a verification process, the matcher retrieves the claimed user template from the database and compares it with the one obtained from the *feature extractor*.

- The ***extraction*** process of finding the digital representation biometric data. It is as a two-step process:

  - ***Pre-processing***: Before being digitized, the raw data set is pre-processed to assess its quality. It is then segmented and enhanced. A quality assurance process for the raw data set is necessary to determine if more sets of biometric data are to be collected. Segmentation is mainly the process of separating actual biometric information from the background model. Finally, the raw data set is enhanced to improve its quality and reduce signal noise.

- ***Feature extractor***: Pre-processed biometric information is digitized to create a biometric template. This template is expected to have unique individual information. This step is functionally merged with the **template creation** and **matcher** processes. For a new biometric speaker, the product of the *feature extractor* is used to build their biometric template. In the case of an existing speaker, the product is used to create a temporary digital representation of the captured biometric payload to be compared against their existing stored template.

1.2 Interpretation of biometric results

In telecommunications networks, as in biometric systems, the tradeoff between *False Accepts* (FA) and *False Rejects* (FR) is an ubiquitous problem. During an authentication process a stored biometric print is compared against the results of a live feature extraction process as shown in Figure 1.1. When an imposter is authorized, it is called a *False Accept*. When the authentic user is denied access, it is known as a *False Reject*. The probabilities of these events happening are, correspondingly, *False Accept Rate* (FAR) and *False Reject Rate* (FRR) (Reid, 2003a). These two rates should be as close to zero as possible. Figure 1.2 shows a distribution of authentication events for a general population. This representation is commonly known as an *Equal Error Rate* (EER) curve. Biometric implementations set thresholds to balance high security with convenience of use. The more secure the system, the higher the likelihood of a FR event. The more convenient the system, the higher the probability of a FA event. Note that the curves in Figure 1.2 are not *normal* distributions but simply a generalist representation of the expected number of reject and accept events.

Defining the following events:

- *Impostor*: a fraudster.
- *Authentic*: authorized user.

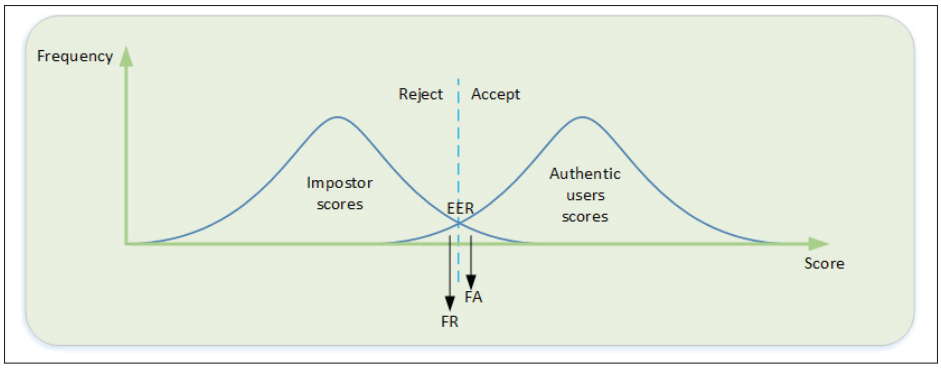


Figure 1.2 *EER curve*.

- *Positive Result*: a result of a biometric operation giving the provider of the bioprint as the authentic user.

Each event can be *true* or *false*. For instance, whether an *Impostor* is actually a *fraudster* or not. The result depends on several factors, mainly, the algorithm itself and the quality of the training set (data use to tune the system).

The EER distribution, inferred from a formulation of the Bayes' theorem (Lee, 2012), would take the form shown in Equation 1.1

$$P(Impostor|positiveResult)=\frac{P(positiveResult|Impostor)\times P(Impostor)}{P(positiveResult|Impostor)\times P(Impostor)+P(Authentic|!Impostor)\times P(!Impostor)} \quad (1.1)$$

where:

$$P(!Impostor)=1-P(Impostor)$$

and:

- *P(Impostor | Positive Result)*: the probability of the bioprint belonging to an impostor given that the result of the operation was positive.
- *P(Positive Result | Impostor)*: the probability of having a positive result in the operation given that the bioprint provider is an impostor.
- *P(Impostor)*: the probability of the bioprint provider being an impostor.

- *P(Authentic | !Impostor)*: the probability of the operation yielding a positive result given that the bioprint provider is not a fraudster (it is the true user).
- *P(!Impostor)*: the probability of a bioprint provider not being a fraudster (being the true user).

In general, Equation 1.1 can be interpreted as the probability of obtaining a positive result when the bioprint provider is an authorized user divided by the probability of obtaining a positive result in the operation regardless of the nature of the provided bioprint.

1.3 Literature review

Several identity management systems for secure authentication have been proposed. This section reviews not only some of those systems but also some previous studies that made them possible.

1.3.1 Interoperable framework for biometric communications

It was once unsafe and expensive to perform remote authentication on telecommunications networks, so the focus of early biometric authentication was on local authentication. Here, remote authentication refers to a system where the subject and the biometric capturing mechanisms are in a different local area network (LAN) to that of the rest of the biometric system as represented in Figure 1.1. This situation made biometric authentication technique impossible in practice. With the availability of higher bandwidth and the development of traffic encryption techniques, the possibility of using remotely distributed processing power and storage became a more acceptable option (Benavente and Piccio-Marchetti, 2005).

Biometric implementations must be inherently secured. Security of communication paths and data repositories must be mandatory. Benavente and Piccio-Marchetti (2005) proposed the encapsulation of the whole communication path during the biometric interaction, suggesting a *interoperable framework*. This framework has three components: the first is an API to access

biometric functions, the second is another API to access biometric information in the form of a token, which is as a pattern provided for comparisons, and the third component is a fusion layer that exposes a common interface to third-party applications using a given framework. The second API performs matching between the protocols. Each module uses encryption and authentication protocols in the form of *Extensible Authentication Protocol* (EAP) over a *Transport Layer Security* (TLS) tunnel. Further details are not discussed because more suitable approaches have been proposed since. However, this contribution was important in raising awareness about the security challenges faced by biometric authentication methods when used over communication networks.

1.3.2 IDM3G, identity management protocol

Dimitriadis and Polemi (2006) propose a detailed *identity management* protocol (IDM3G) for Internet applications over 3G mobile networks. This protocol combines the identity management principles of the former *Liberty Alliance* specification (currently Kantara Initiative)[2] with those of the *Organization for the Advancement of Structured Information Standards* (OASIS)[3] and the *3rd Generation Partnership Project* (3GPP) [4]. This proposal seeks to provide a lightweight identity management system. Its relevance is linked to three key concepts required for any authentication protocol: security assessment, performance and implementation complexity. Dimitriadis and Polemi (2006) use the same *Authentication Key Agreement* (AKA) mechanism of the *Universal Mobile Telecommunications Systems* (UMTS), which is the formal extension of the PIN/SIM combination previously mentioned. IDM3G specifies four entities: user (U), User's SIM (USIM), *Mobile Operator* (MO) and *Service Provider* (SP). The logic of the protocol is based on the transmission of a random token from U to SP. The same token is then forwarded by SP to MO, which will use it to determine which voice and data services to provide to U. IDM3G assumes that there is already a trust relationship between MO and SP in which the identity of U needs to be assessed using alternative methods (including but

[2] kantarainitiative.org

[3] www.oasis-open.org

[4] www.3gpp.org

not limited to biometric authentication). IDM3G also assumes that between the USIM and the MO there is a mutual UMTS-AKA mechanism to guarantee the identity of the parties. The technical details of the IDM3G protocol are beyond the scope of this document, however, it can be summarised as follows:

a. USIM request a service on behalf of U, who has been previously authenticated.

b. USIM provides proof of identity to MO.

c. SP requests MO to verify the USIM identity.

d. MO certifies the USIM identity based on matching of the two pieces of information, the one sent by SP and the one previously sent by USIM.

IDM3G was evaluated in terms of performance, implementation complexity and security (the third, security, against guidelines established by the *Internet Engineering Task Force* (IETF)[5]. SIPBIO uses similar guidelines to be explicated in Chapter 3.

1.3.3 Voice Interactive Personalized Security

Another interesting proposal for the use of biometrics for telephony environments was the *Voice Interactive Personalized Security* (VoIPSEC) protocol presented by Kopsidas *et al.* (2006). VoIPSEC attempts to provide end-to-end secured communications with the use of inbound key exchange and biometric verification. Through analysis of VoIP communication patterns over the Internet, the authors concluded that an end-to-end encryption between the parties was the only way to offer full communications confidentiality. VoIPSEC was intended to provide such a mechanism in three phases. The first phase consists of the generation of constituent components:

[5] www.ietf.org

a. A *private key*, a *public key* and a *User Session Signature* (USS) for each party. The USS has a particular property: it can be any binary object that the party prefers to use (e.g. an email address, a video, etc.).

b. A *symmetric session key* to be used during the communication between both parties.

In the second phase, the communication parties were to exchange the *symmetric session key* and their respective USS. In the third phase, the USS was to be biometrically verified. This biometric action could be of level 1 (only voice) or level 2 (voice and video). If any participant were not successfully verified, both communication parties were to be notified. It is up to the application using the protocol to define if it makes the authentication compulsory or if it gives the participant the option to continue opening the communication channel after an unsuccessful verification. The security of the protocol depends heavily on the USS and the symmetric session key exchanged by the participants before any information was to be transmitted. This behaviour is quite similar to a registration process. Research literature suggests that the VoIPSEC proposal has not been adopted. It may be because it appears rather complicated and no canonical implementation is freely available. However, the use of biometric data as part of an authentication process makes VoIPSEC an interesting alternative: it uses the biometric information to secure itself.

1.3.4 BIO3G protocol

Another attempt to create a biometric based authentication protocol was presented by Dimitriadis and Shaikh (2007) with their BIO3G protocol proposal for *Third Generation* (3G) mobile systems. This was an extension of what was commented on above and presented by Dimitriadis and Polemi (2006). Unlike VoIPSEC, BIO3G does not enroll users or transfer any data across the data network. The aim of BIO3G is to provide PIN-less authentication between an end user and a mobile operator using only biometric techniques, without the need for biometric enrolment. BIO3G still uses the same network access authentication mechanism of UTMS, UMTS-AKA, as its core authentication element.

The logic of the protocol is as follows: during the first interaction between the user and the mobile operator (e.g. when the mobile phone is turned on for the first time or when the mobile radios are enabled) the end user provides some sort of biometric material (possibly a voice sample) to the phone (specifically to the SIM) which calculates a key k; the key is sent to the mobile operator which produces a new key k based on the one received from the SIM; the mobile operator sends back k to the SIM. From that point forward, any time the user wants to make a call, they will provide a biometric sample that will be used by the SIM to locally generate k that will be used by UMTS-AKA to authenticate the SIM against the mobile operator (Dimitriadis and Shaikh, 2007).

BIO3G is an interesting approach that addresses some problems of the authentication biometric techniques: storage of biometric material (i.e. it does not store any). However, BIO3G also faces some difficulties. Variable quality of biometric material can yield a non-compatible key for subsequent user authentication, the limited processing power on some phones being the cause. However, due to the development of authentication techniques and the capacity of current mobile phones, those limitations are less relevant compared to when BIO3G was proposed more than ten years ago. Dimitriadis and Shaikh (2007) evaluate the compliance of BIO3G using formal process algebra *Communication Sequential Processes* (CSP) (Hoare, 1978) and *Rank Functions* (Schneider, 1998).

1.3.5 Securing biometric templates transmission

Biometric features for authentication over communication networks, such as an individual's biometric print (template), may be stolen either in transit or from the system database. Several solutions have been proposed to address this concern. BIO3G completely avoids both transmission and remote storage of biometric prints (Dimitriadis and Shaikh, 2007). Kikuchi *et al.* (2010) proposed the use of a *zero-knowledge proof* protocol, which allows a user to prove that they have some valid piece of biometric data, without actually revealing it. They proposed two protocols to achieve their purpose: The *Private-Cosine* and the *Private-Euclid* based on the *cosine correlation* and the *Euclidian distance* respectively. With reasonable computational

resources, both protocols allow remote authentication without revealing any private data. In terms of accuracy, the biometric results are not satisfactory, however, for the purpose of this literature survey we are only interested in the description of a feasible secure remote biometric authentication. Kuseler *et al.* (2010) proposed a layer authentication architecture, *eBiometrics*, to be implemented as an application on mobile phones. *eBiometrics* pretends to be an application that offers authentication services to other applications through the control of all matching sensors (e.g. camera, fingerprint reader and microphone). *eBiometrics* would pre-process, analyse and deliver results on raw biometric pieces of information. It can be either implemented as a local (on the mobile phone itself) self-contained system or as the client of a hosted application.

Johnson *et al.* (2014) presented a different approach to the issue of compromised biometric templates. Their proposal is based on the concept of *vault verification* introduced by Wilber *et al.* (2012), which consists in separating the biometric template into several pieces, scrambling them with fake pieces of biometric data and then putting them back together. The objective is to obfuscate the real information making it very difficult for an attacker to identify the authentic data in the biometric template. Subsequently, in similar papers, Wilber and Boult (2012) and Johnson *et al.* (2014) argued that in the context of voice authentication over remote networks, even if the biometric template is stolen, the risk of a successful attack can be minimised by using random authentication with short phrases as a complement of the main authentication process. They also proposed to add random pass phrases to the real template in order to use them to confirm the identity of the true speaker. This technique has the added value of mixing text dependent (a pass phrase) and text independent (a random phrase) audio responses that may mitigate reported vulnerabilities of both text dependent and independent approaches.

1.3.6 Related projects

Finally, to close this review, several authors developed approaches to solve different aspects of the problem of secured remote biometric authentication over communication networks. These

proposals were not reviewed extensively to avoid making this chapter overly long. The reader is invited to explore the original articles for further details.

- Li and Hwang (2010) suggestd the use of one-way hash functions, biometric verification and smart cards.
- Agbinya *et al.* (2011) proposed a *Multimodal Identity Management System* that fuses fingerprints and face recognition using neural networks based biometric techniques.
- Xi *et al.* (2011) proposed a client server biometric authentication protocol oriented to mobile environment; it uses encrypted fingerprints (based on Elliptic Curve Cryptography) and a *Public Key Infrastructure* (PKI) to protect biometric authentication sessions through insecure mobile networks.
- To avoid user information to be extrapolated from biometric data (e.g. genetic information or diseases), Abidin and Mitrokotsa (2014) proposed an enhancement to the *privacy preserving biometric authentication protocol* (PPBA) proposed by Bringer *et al.* (2007) using the *Goldwasser-Micali* cryptosystem based on homomorphic encryption. They improved the PPBA security by using two secret keys against the system biometric sensor during the verification stage.
- Using homomorphic encryption and a similarity scored based on Squared Euclidian Distance between the query vector (the analysed biometric verification sample) and the biometric print, Wong and Kim (2012) claimed that a biometric verification can be completed without exposing the original data.
- Traore *et al.* (2014) proposed Behavioural biometrics. A Bayesian network model is applied to remote keyboard and mouse events to create their particular usage characteristics of a specific user. Their experimentation yields a EER of 8.21% on a limited subset.
- Nomura *et al.* (2015) took the heartbeat waveform as their biometric measure; they analyse the properties of an electrocardiogram (ECG) to identify a subject with the dynamic variability of the ECG which, reportedly, keeps unique the features of the subject. The

reported positive authentication rate is not that encouraging at around 80%, however, since wearable devices could be adapted to read ECG signals providing a very unique biometric identification, a great potential is seen in this type of approach.

- Saevanee *et al.* (2015) used multimodal biometrics over mobile networks: linguistic analysis, key strokes dynamics and behavioural profiling to reach a reduction of 91% in the rate of spoofed authentication. The key element of this study is the demonstration that, even though one single biometric entity may not be trustworthy, the combination of several biometric measures can leverage positive results; furthermore, it opens the door to the possibility of using continuous data entries for an equally continuous authentication through the length of a user-to-user interaction.

All the works mentioned in this chapter focus on characteristics of the biometric process and the process to provide biometric payloads and collect the results. None of them has leveraged an already accepted communication mechanism to facilitate these processes. Next chapter introduces such mechanism to, subsequently, explore its use in biometrics.

CHAPTER 2

SIP REVIEW

Session Initiation Protocol (SIP) is an application layer telecommunications control protocol. It has been designed to handle signalling for multimedia sessions: establishment, modification and termination. SIP is used with other protocols, notably the Session Description Protocol (SDP), to transport multimedia content (including VoIP, images, video, etc.). SIP can be considered as a framework for the deployment and development of communication services (Rosenberg and Schulzrinne, 2006). SIP provides all actors of a communication exchange with a set of rules for establishing a session. SIP is text-based encoded. In practice this means that SIP is easy to read and understand. Any protocol that SIP uses (e.g. SDP) is expected to have the following features (Martinez, 2008a):

- Clear indication of supported media.
- The availability of the media through the session.
- Transport information for the media itself (IP and port to which the media packets should be sent).

In its basic form, SIP can be used as a peer-to-peer protocol where endpoint devices (called *User Agents* (UAs)) have a high level of autonomy. A complete exchange of data can be entirely processed between two UAs without the need for any third-party component. Figure 2.1 illustrates the basic building blocks of a SIP communication exchange: each participant is called a *User Agent (UA)*, each UA can take the logical role of a *client* (UAC)or a *server* (UAS), the client is the one initiating the conversation *requesting something* and the server is in charge of *replying* to that client's requests (Martinez, 2008a).

As extrapolated from Figure 2.1, UAs are flexible in nature and can take different roles. It is customary to define the SIP functions in two main groups: UA clients (UAC) and UA servers (UAS). An endpoint usually supports both groups of functions.

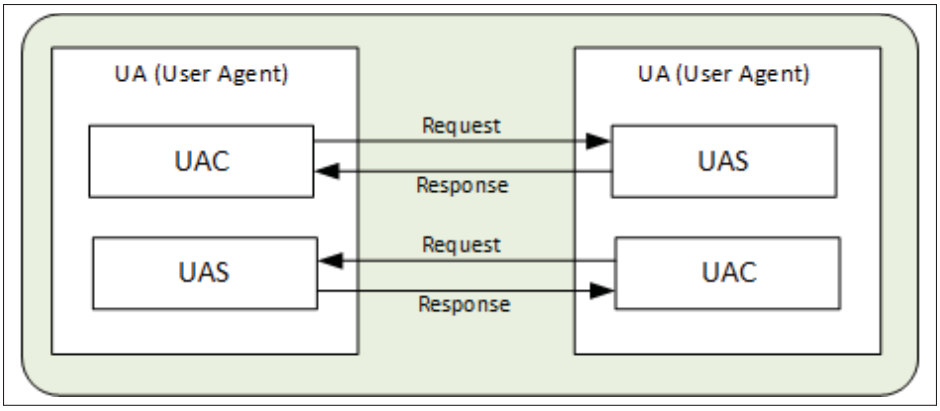


Figure 2.1 *Basic SIP Communication*.

In practice UA to UA communications are processed by SIP-aware devices such as proxy servers, soft-switches, SBCs, etc. All intermediate components facilitate services and enable UAs to communicate over different IP networks. In general, UAs initiate and control dialogs while intermediate devices provide routing and offer extra services. Under certain conditions, intermediate devices can also act as UAs. When required, potential confusions are to be avoided by referring to the UAs as SIP clients or servers, where the client is the entity originating the request and the server is the entity receiving and processing the request (Martinez, 2008a).

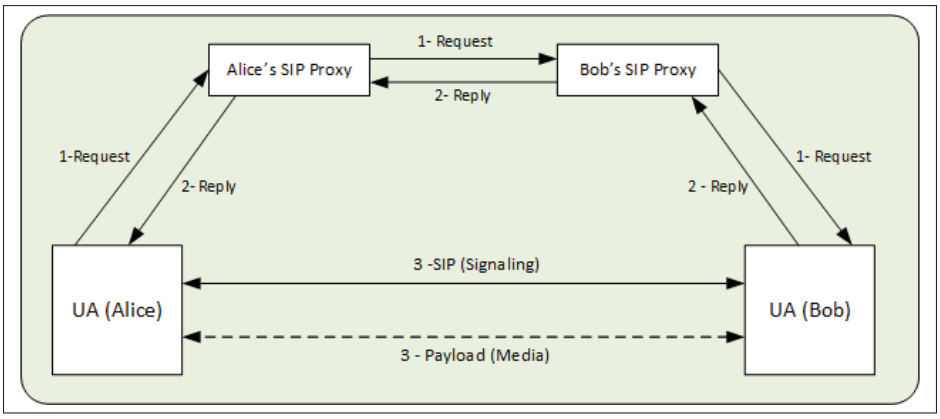


Figure 2.2 *SIP Basic Trapezoid*.

Figure 2.2 illustrates the most basic scenario of a SIP communication establishment between two UAs located in different networks. The initial request is sent by *Alice's* UA through her local SIP proxy; this proxy sends the request to Bob's SIP Proxy (either directly or through intermediate SIP proxies); eventually the request reaches Bob's UA which reply follows the same logical path backwards to reach *Alice's* UA. If everything works as expected, a direct communication is established between both UAs. This scenario is the simplest of the cases. It can very well happen that the proxies are kept in the path of both the signalling and media sessions, or only the signalling goes through the SIP proxies whilst the media establishes direct communication or goes through different proxies specialised in handling the specific payload type (Steffen *et al.*, 2004).

UAs are identified by a SIP *Universal Resource Identifier* (URI). A SIP URI must conform to the rules established by RFC 3986 and must have enough information to establish and keep a communication session with the UA (Berners-Lee *et al.*, 2005). An example of a SIP URI would be: *sip:wilmar.perez@ens.etsmtl.ca.*

A SIP URI can be virtually identical to a regular email address. A more generalised description of the SIP URI format would be *sip:userID@host:port*, where the *user* ID is an optional component which uniquely identifies a resource (a user, a group of users, an extension, a service, etc.) and the host is either the *Fully Qualified Domain Name* (FQDN) or the IP the resource is associated with. Finally, the port is only used when a non-default port is used for the communication establishment. SIP URIs that provide identification to resources to be contacted over a secured channel (e.g. *Transport Layer Security* (TLS)) change their format to SIPS URI, note the S after the SIP identifier. SIP URIs have other interesting properties such as the generic use of URIs to identify secondary resources for a given already identified resource. For instance, starting from a single URI, a user could be linked to his email, phone number, web site, etc. (Schulzrinne, 2001).

Figure 2.2 shows how a proxy server entity is used to facilitate the communication between two UAs. SIP scenarios include the use of several facilitating entities or servers. These servers are usually named based on their function:

- **Proxies**: *SIP Proxy Servers* mainly determine where to send signalling messages (i.e. route SIP messages). Given their location in the communication path, SIP Proxies also are often given the task of performing authentication and authorisation. A *SIP Proxy* can participate during call establishment or for the entire length of the call, which would depend on the level of call control desired and the topology of the network. For network to network communication the *SIP proxy* is usually required during the entire communication. SIP proxies can manipulate SIP headers to redirect or modify the characteristics of an interaction. Each UA needs to be notified of its corresponding *SIP proxy or proxies*. This notification is usually accomplished by direct configuration or by using network configuration protocols (e.g. *Dynamic Host Control Protocol* - DHCP) (Subramanian and Dutta, 2013).

- **Location**: *Location Servers* keep track of the current location of all UAs in the system. They usually keep records of registered clients (i.e. UAs) in a local database. Each record holds an UA ID and its last known network location. A Location Server provides UA location information for other UAs. Note that a location service is not a SIP entity, it is a general service that, as mentioned, keeps records of the UA location without any impact on the SIP signalling or any associated payload. (Ott, 2001)

- **Registrar**: *SIP Registrar Servers* are entities that handle the registration of UAs. When a Registrar server allows a UA request to be registered, it notifies an associated location server that keeps a log of all UA whereabouts. When an UA changes location, it registers through the registrar in the corresponding zone which updates the record in the respective location server. UAs periodically send register messages to update their location (Schulzrinne, 2001).

- **Redirect**: *SIP Redirect Servers* provide alternative URI information to an UA sending a request. For instance, they may force traffic going to a specific domain to be handled by an

auxiliary server in the case of system failure. *SIP Redirect Servers* do not initiate requests or accept calls which makes them very efficient for handling high loads. However, their use is limited to a very specific set of functions (Osterhout, 2003).

Among the basic SIP entities there is another widely used SIP element known as a *Back-to-Back User Agent (B2BUA)*. A *B2BUA* is a special type of SIP entity able to act as a different UA (UAC or UAS) for both ends of a SIP call. A *B2BUA* takes a SIP request, processes it following an internal programmatic logic and then creates a new SIP request based on the original one plus relevant programming rules. A *B2BAUA* can be thought as a pair of UAs linked by a programmed logic. *B2BUAs* are also known as *SIP application servers*. *B2BUAs* are primarily used to modify signalling properties and routing calls based on high-level logic (Zave *et al.*, 2009). Figure 2.3 illustrates the design principle behind a *B2BUA*.

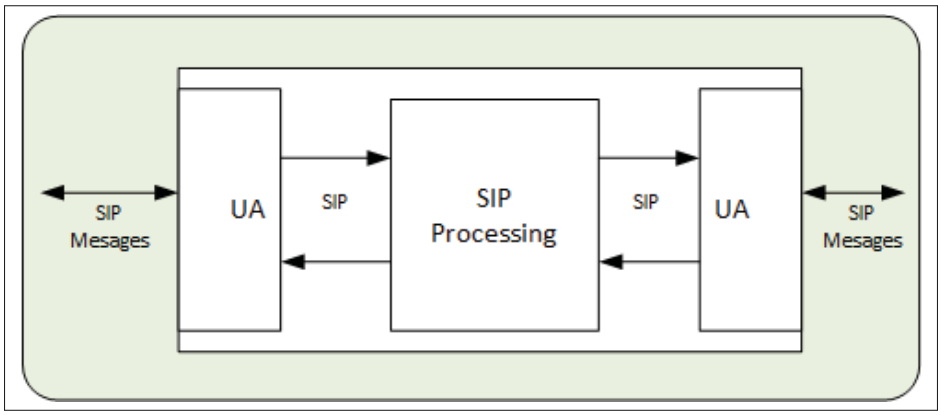


Figure 2.3 *B2BUA*.

SIP has become very popular in part because of the formatting of its messages. Unlike other communication protocols, SIP messages are text-based (they use the *UTF-8* charset and follow the Internet Message Standard as defined in RFC 2822 (Resnick, 2001)). SIP Messages are classified as requests *(UAC to UAS*) and responses (*UAS to UAC*). SIP messages are, in form, similar to the requests and responses of the *Hyper Text Transfer Protocol* (HTTP) 1.1. as described in RFC2616 (Fielding *et al.*, 1999). However, SIP is not an extension of HTTP; SIP and HTTP simply share the same type of formatting and the same *working philosophy* of

simplified requests and responses. This proposal is SIP-dependent so to understand what is expected of SIP messages a summary of their main characteristics follows.

2.1 SIP Requests

SIP requests always have three elements: a method name (REGISTER, INVITE, ACK, CANCEL, BYE and OPTIONS), a request URI, and the protocol version. In general, the format of a SIP request is **Method[ ]Request URI[ ]Protocol version**.

For instance, *INVITE sip: wilmar.perez@ens.etsmtl.ca SIP 2.0* is a valid SIP request.

The behaviour of every SIP request depends of the method being used. Most relevant characteristics of SIP Requests Methods are shown next.

REGISTER. It is used by the UA to associate its public identity to its current location (*contact address*) through a *registrar server*. The request must include the public identity (e.g. *wilmar.perez@ens.etsmtl.ca*) and its current location (e.g. *wilmar.perez@172.25.34.10*). When the registration process is successful, an entry with the appropriate information is added to the *location server*. The UA needs to send a new registration request before the time-to-live of the registered entry expires.

INVITE. Due to its flexibility, the INVITE has become the most used SIP request method. In its more basic form, a SIP INVITE is used when a UAC initiates a session. An INVITE request contains the UAC public identity. After an INVITE the UAC expects either a success or a provisional response. The message exchange initiated by an INVITE is known as a *SIP DIALOG*. The INVITE is also used for the parties to agree on the codecs to use and the IP and ports where media (e.g. RTP) traffic will be exchanged. SIP itself is not intended to be used for media characteristics definition. However, it can carry SDP information as body content. SDP, on the other hand, is specifically designed to describe the media properties of a communication session. (Handley *et al.*, 2006)

A particularly useful SIP characteristic is the possibility of modifying many of the initial dialogue parameters. Within an existing SIP dialogue, a new INVITE can be exchanged between the parties to modify session characteristics (e.g. change media destination). This type of request is known as a *Re-INVITE*.

Arguably, the INVITE is the most important type of SIP message. It can start a communication request and it can be used to modify the parameters of a communication in progress: change routing, change types of media, add parties to the conversation, etc. For the first communication establishment, an INVITE relies on a SIP proxy service to find the right party and its respective location. A SIP resource can register itself from multiple locations against a given SIP Proxy. A user, for instance, can register a SIP phone on a mobile and a computer at the same time, notifications are sent to both registered endpoints.

ACK. It is a special type of SIP request used by the three-way handshake implemented in the INVITE method. An UA generates an ACK when it receives the final response corresponding to an INVITE.

CANCEL. This request is used to interrupt a pending transaction. Upon reception of a CANCEL request the UAS simply acknowledges it with a 200 OK and cancels any pending transaction from the corresponding dialogue. The UAS also notifies the UAC about the cancellation of the original transaction with a 487 response.

BYE. It is mostly used to simply terminate an existing session.

OPTIONS. This request allows a UAC to query a UAS to know of its capabilities (e.g. supported methods, codecs, etc.). The main objective of an OPTIONS request is for the UAC to know how the build a subsequent INVITE based on the actual UAS capabilities.

2.2 SIP Responses

There can be more than one SIP to a single request: one final response and, alternatively, several provisional ones. Responses are identified by a three-digit status code that indicates

the result of a request. The three-digit code is intended to be used as a programmatic guide for SIP-based applications. Status codes are divided in six groups:

- 1xx: Provisional (task in progress.)
- 2xx: Success.
- 3xx: Redirection (a different set of actions need to be done to complete the request.)
- 4xx: Client error (the request is not valid or is not supported by the server.)
- 5xx: Server error (the request seems to be valid but the server was not able process it.)
- 6xx: Global failure

The format of a Response is as follows:

Protocol version /[ ] Status-Code /[ ] Reason phrase -> e.g. : SIP 2.0 200 OK

Header Fields:

Header Fields provide detailed information of requests and responses and their respective body contents. Each Header Field has a field name, a field value (which may contain an optional display name for visualization purposes) and, optionally, a parameter (*field name: Display Name <field value>; parameter name = parameter value*). The field value is any of the previously seen requests and responses. A simple example would be:

From: Wilmar Perez <sip:3738@172.25.34.10>; tag=074e6845296ae42ba

A short description of the main*header fields* is shown below:

FROM. It indicates the AOR (logical identity) of the UA initiating the request.

TO. It indicates the AOR (logical identity) of the recipient. Notably, the To Header Field is not modified by any intermediate proxy.

CALL-ID. It is an unique automatic identifier to link messages within the same SIP Dialogue.

VIA. It is added by each SIP proxy in the communication path. It indicates the routing path a response needs to follow. It must include the transport protocol used to send the message and the sender network information. It may also have two optional fields: *branch* to identify messages within the same transaction and *received* which shows the true origin of a request.

CONTACT. An UA can provide a SIP URI that can be used by other parties to establish future contacts. Since VIA headers can potentially be stripped off when a specific SIP proxy is no longer required in the path, a CONTACT header is a reliable way of keeping the UA information during the whole SIP Dialog.

RECORD-ROUTE and ROUTE. These two headers are used together to force responses to go through a specific route. A SIP Proxy can introduce a RECORD-ROUTE to force future requests in the same SIP dialogue to go through a specific route. When the requesting UA receives a response with a RECORD-ROUTE request header, it inserts the received value inside a ROUTE header to force subsequent responses to go through a path of specified proxies.

CSEQ (Command Sequence). This header consists of a sequence number and method. The number is used to keep a sequence of end-to-end requests in a SIP dialogue. The method is used to keep a correlation of requests and responses within the same transaction.

MAX-FORWARDS. The *Max-Forward Header* defines the maximum number of proxies that a request can traverse to get to its destination.

There are additional relevant headers and characteristics of the SIP protocol that are not central to the functioning of the communication. Further details and information can be found in RFC 3621. (Rosenberg *et al.*, 2002).

To understand the elements involved in a SIP exchange it is useful to illustrate the concept with a classical communication example: Alice and Bob. Alice wants to establish a voice communication with Bob whose endpoints are in at least one foreign network. As explained above, SIP uses a *three-way handshake* protocol to complete the session establishment. When Alice wants to establish a session, she sends an INVITE with the desired characteristics of the session, including the media and media transmission properties. Bob's endpoint immediately replies with a *100 Trying* to let Alice know that it is an actual entity capable of SIP communication. Bob's endpoint also sends a *180 Ringing* to indicate that there has been a contact and that action from Bob is expected to complete the establishment. Once Bob answers, a *200 OK* message is sent to Alice to indicate the request has been accepted, to which she would reply with ACK. This ACK request does not require a response. Media transmission starts. The set of events is illustrated in Figure 2.4.

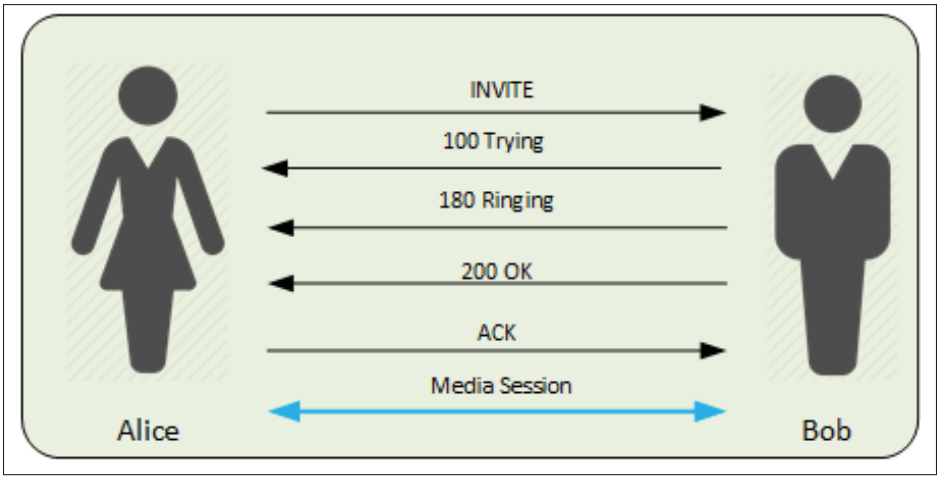


Figure 2.4 *Three-way handshake*.

Providing the communication establishment is successful, a simplified message exchange is illustrated in Figure 2.5.

A frequently used tool is the SIP trapezoid. It can be used to illustrate the message exchange between all SIP entities in the communication path. A basic example of a SIP trapezoid is

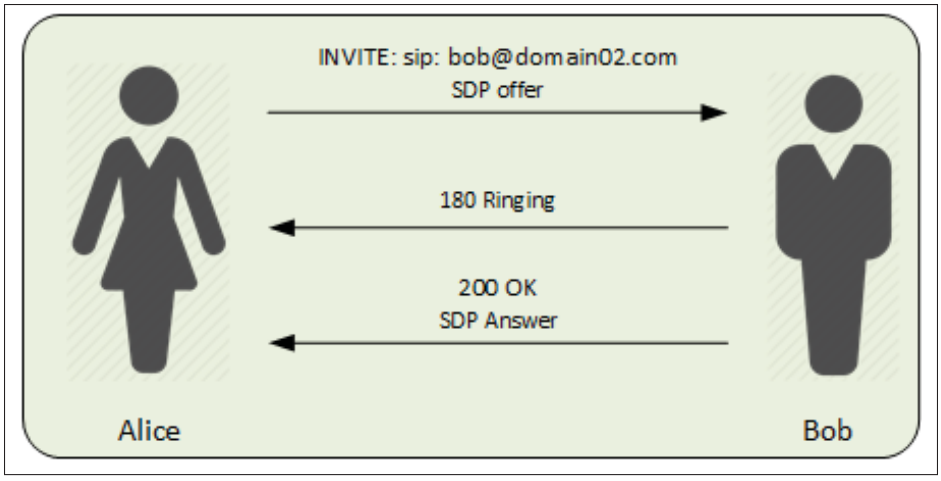


Figure 2.5 *Basic SIP Message Exchange*.

shown in Figure 2.6, where a successful basic session establishment and media exchange are represented.

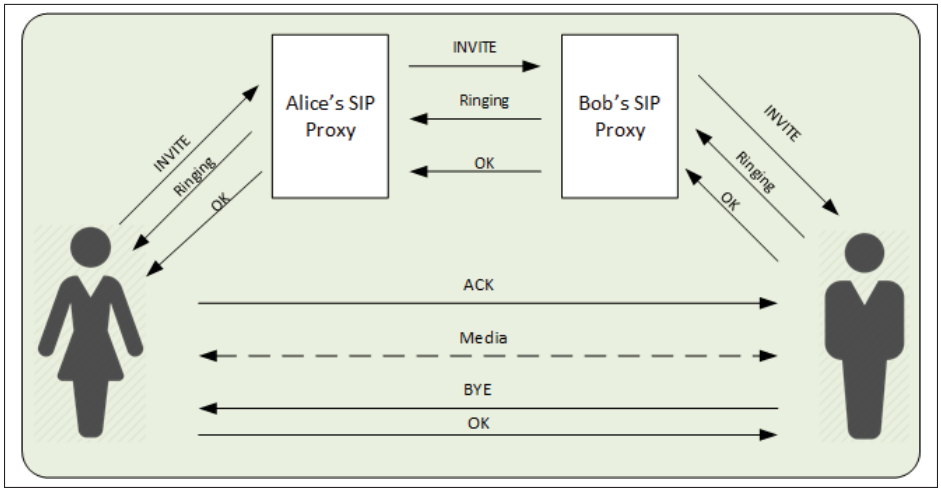


Figure 2.6 *Basic SIP Trapezoid*.

Even though Figure 2.6 shows initial communication establishment being routed through the proxies whilst subsequent SIP messages and media are exchanged directly, it could also happen that all communications flow exclusively through the proxies.

Understanding SIP trapezoids allows for easier understanding of SIP traces. In Figures 2.4 and 2.5 Alice and Bob are part of the same SIP domain and no proxies are required. On the other hand, Figure 2.6 includes the concept of proxies, which implies that Alice and Bob are on different SIP domains.

A testing scenario was configured to recreate and test SIP communication for this proposal. In the test environment Alice and Bob are in different locations as shown in Figure 2.7. Bob is at a remote location, connected through the Internet and establishing a VPN tunnel to the location where the softswitch is; Alice is collocated to the telephony switch. The tests on this proposal use FreeSwitch[1] as the softswitch, Linphone[2] as the endpoints and Cisco Any Connect[3] as the VPN client.

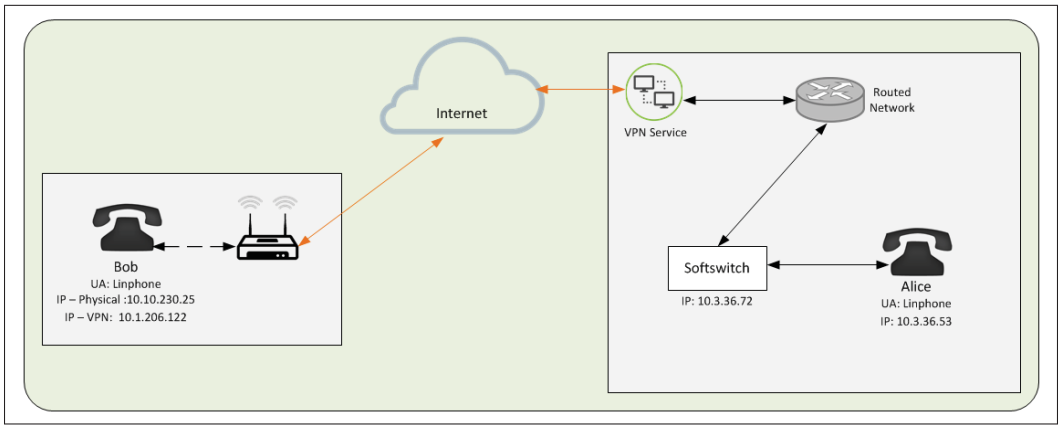


Figure 2.7 *Physical Diagram (Bob and Alice).*

Figures 2.8 shows a SIP trace of the message exchange between Alice and Bob as seen from Bob's side. To properly understand a SIP exchange, a dissection of the SIP messages follows.

---

1 freeswitch.org

2 www.linphone.org

3 goo.gl/rXRu9r

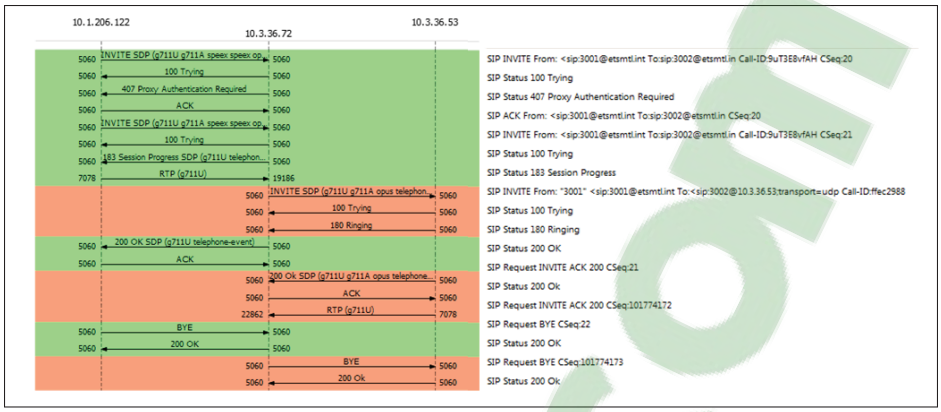


Figure 2.8 *SIP Flowfor Alice and Bob (Bob Side).*

The first SIP INVITE is sent by Bob's UA. Since Bob does not know how to contact Alice, he sends the request to its own register server that takes the role of a SIP proxy for the communication.

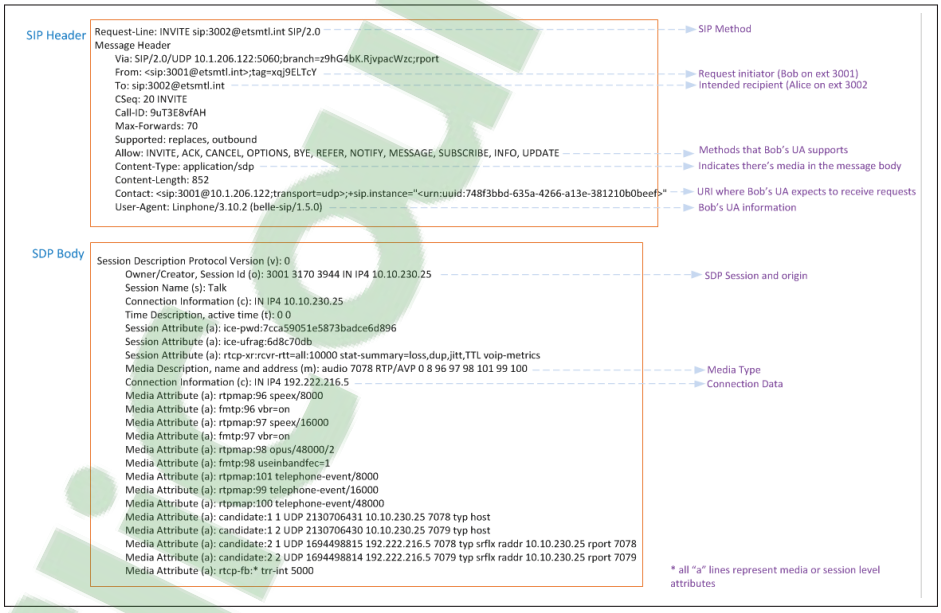


Figure 2.9 *SIP INVITE (Bob to Proxy).*

As shown in Figure 2.10, the SIP proxy lets Bob know that it is trying to complete the requested event.

Figure 2.10 *SIP Trying (Proxy to Bob).*

The SIP Proxy then realises that authentication is needed before serving any request from Bob. It sends an authentication request to Bob's UA. The SIP proxy response includes a challenge along with basic information to build an appropriate SIP INVITE request as shown in Figure 2.11.

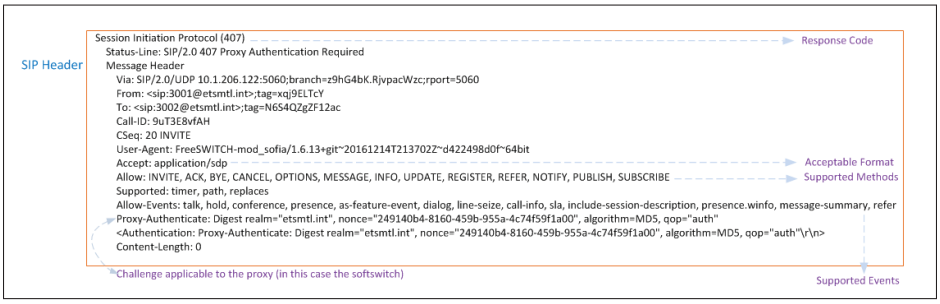


Figure 2.11 *Authentication Request (Proxy to Bob).*

Bob acknowledges the request.

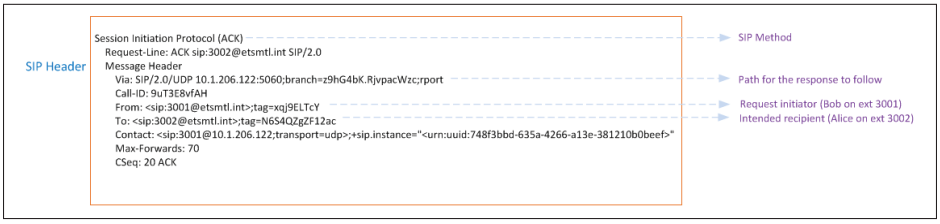


Figure 2.12 *Request Acknowledgement (Bob to Proxy).*

Bob sends a new SIP INVITE with the required authentication information. Figure 2.13 shows the new INVITE.

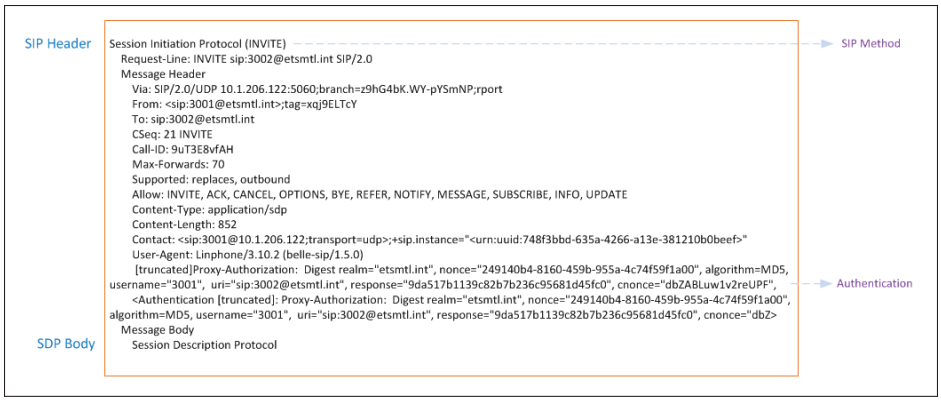


Figure 2.13 *SIP Invite with Authentication (Bob to Proxy).*

Once the use of the service has been granted, the proxy sends Bob a new indication that the process is going ahead.

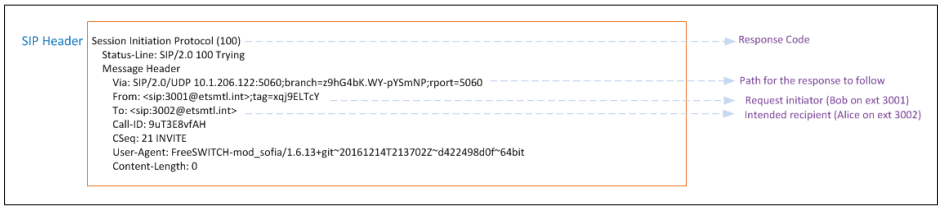


Figure 2.14 *SIP Trying response (Proxy to Bob).*

The proxy notifies Bob's UA that a session is in progress. Since Alice's UA has not been yet contacted, the telephony provider (which happens to be the same SIP proxy in this particular case) adds a SDP body indicating an upcoming artificial ring back tone or announcement. This is done in a 183 message as shown in Figure 2.15.

An INVITE, shown in Figure 2.16 is finally dispatched to Alice. Comparing against the ones in figures 2.9 and 2.13, some characteristics are worth highlighting:

- *From* and *To* headers are no different from the original INVITE.
- The *Contact* header is modified. It points at the softswitch SIP module handling the event.

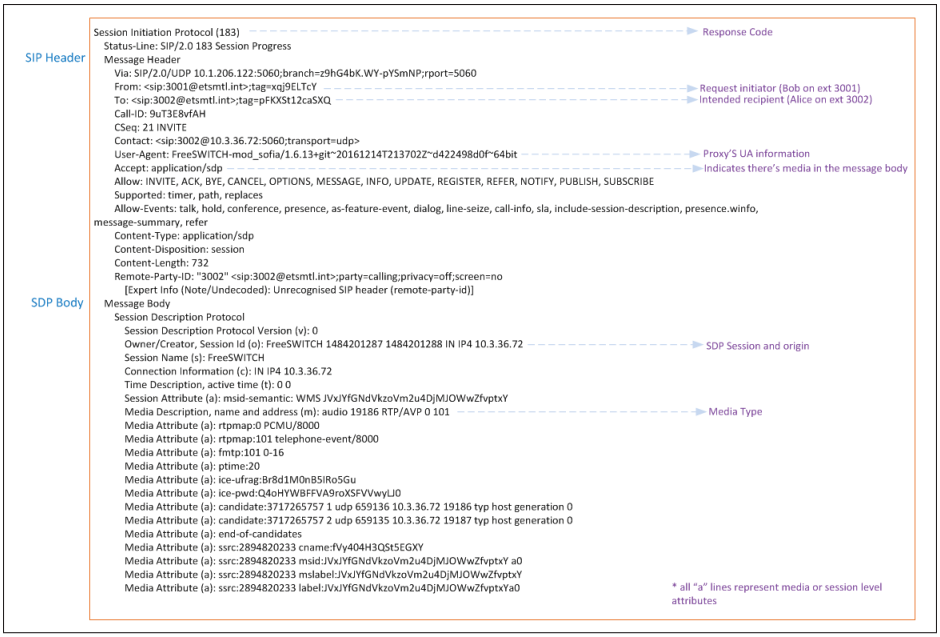


Figure 2.15 *183 Session in Progress (Proxy to Bob).*

- Media attributes in the SDP are limited to PCMU, PCMA and Opus as supported by all parties.

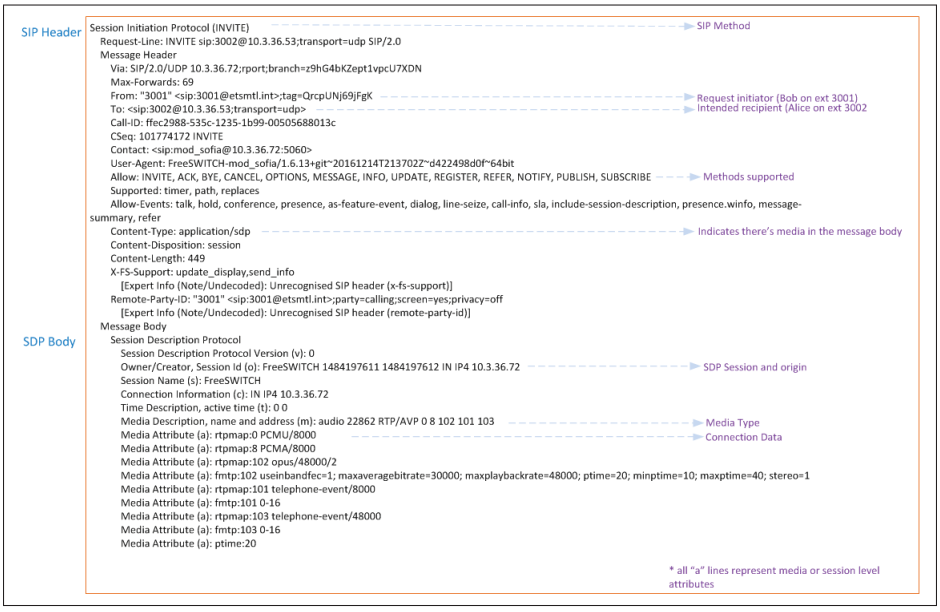


Figure 2.16 *SIP INVITE (Proxy to Alice).*

Alice's UA dispatches response codes signalling that it is, first, trying to complete the request and then informing that the user is being notified. Figures 2.17 and 2.18 show the respective SIP Trying (100) and SIP Ringing (100) messages.

Figure 2.17 *SIP Trying response (Alice to Proxy).*

Figure 2.18 *SIP Ringing (Alice to Proxy).*

The proxy notifies Bob of the successful reception of the message. Figure 2.19 shows the corresponding 200 OK message.

Bob UA acknowledges the notification. Authentication information is included along with every SIP message from Bob as shown in Figure 2.20

Alice's UA, notifies of the successful message delivery as illustrated in Figure 2.21.

The proxy then responds with an acknowledgement to Alice as shown in Figure 2.22.

At this point the communication path has been fully established and regular media (RTP in this example) flow starts or continues (early media could have been already sent) between the UA. In this particular testing layout the softswitch acts as the media gateway as well as the router (i.e. all media traffic necessarily traverses the softswitch).

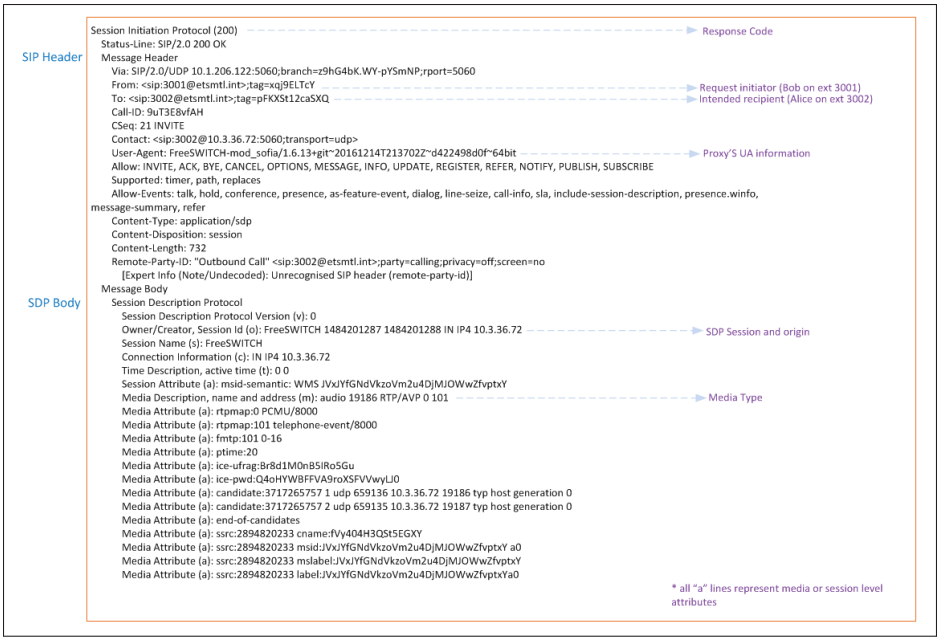


Figure 2.19 *SIP OK (Proxy to Bob).*

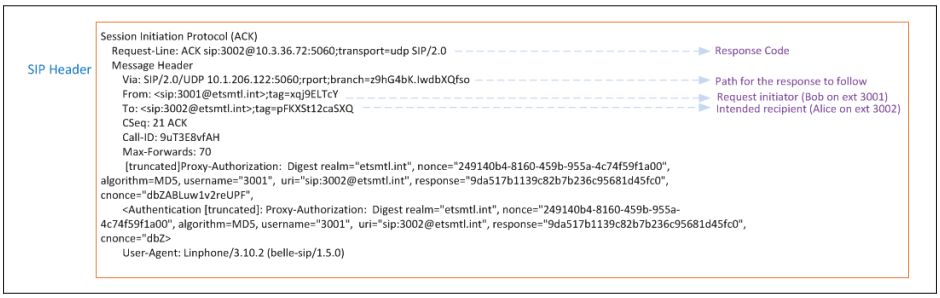


Figure 2.20 *ACK (Bob to Proxy).*

There are many possible subsequent SIP messages: one of the parties may be put on hold, an additional media channel could be added, another party may join the call, etc. In the sample interaction Bob simply decides to terminate the call. Bob's UA notifies the proxy that replies with a 200 OK and, in turn, notifies Alice, who also replies with a 200 OK. A summary of these four messages is shown in Figure 2.23. This concludes the message analysis exercise.

SIP as a signalling protocol which is proven to be clear and rigorous to establish trustworthy communication paths between parties. Although the process seems lengthy and cumbersome, in practice most of the heavy lifting is done by any chosen SIP framework. The base flow of

Figure 2.21 *OK (Alice to Proxy).*

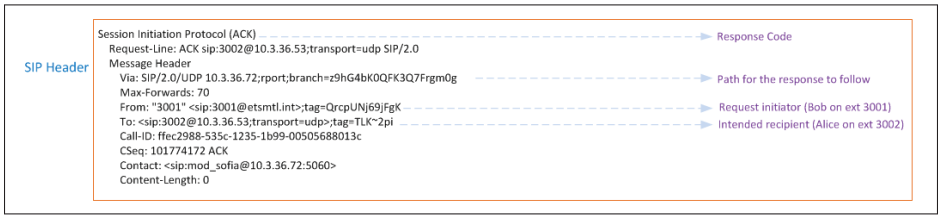


Figure 2.22 *ACK (Proxy to Alice).*

the Alice and Bob message exchange is used to understand SIPBIO messaging flow later in this document.

Figure 2.23 *SIP Dialog termination (Bob to Proxy, Proxy to Bob, Proxy to Alice, Alice to Proxy).*

It is important to remember that SIP is a layered protocol that suits independent development of sections or functionalities. Notably, not every UA implements all layers. SIP layers can be summarized as follows:

Syntax and encoding. Defines how requests and responses are written.

Transport. Defines how UAs send and receive requests and responses.

Transaction. It can be thought as the set of client / server message exchanges needed to handle a specific request.

Transaction user. It is basically any SIP entity except for a stateless proxy (a message forwarder that does not actively participate in the communication).

So far, the concepts of SIP *entities services* have been presented without much explanation of their true nature. They are logical components, the former responds and acts to the logic of *requests* while the later delivers extra enhanced functionality for the UA. SIP services are most commonly implemented on a B2BUA due to their flexibility to modify signalling and routing as required.

This chapter has presented how a basic SIP implementation can manage the requirements of most multimedia sessions: establishment, maintenance, and finalisation. However, there are cases deemed very difficult to handle with the default SIP implementation. SIP can be adapted to such cases through *extensions* as defined in RFC 4485 (Rosenberg and Schulzrinne, 2006). The process of evaluating the need or suitability of a new SIP extension is discussed in the next chapter with some examples selected for their similarity (or complementarity) to SIPBIO.

2.3 SIP extensions

Previous sections displayed the power and flexibility of SIP to establish interactive communications between endpoints across networks. SIP flexibility comes, in part, from the ability to extend core functionalities through extensions. These extensions are used to define new methods, header fields, body types and parameters. Many have been proposed with some being widely used whilst others have never been adopted. This section introduces the guidelines to create SIP extensions and reviews some of those extensions from which SIPBIO borrows logic and functionalities.

2.3.1 SIP extensions guidelines

In RFC 4485, Rosenberg and Schulzrinne (2006) proposed a set of guidelines to author them. The aim of these guidelines is to be used as a reference for extension developers on SIP architectural concepts. Clarity on the SIP architecture helps developers to evaluate the viability of a proposed extension. An overview of the proposed guidelines is shown next.

SIP Solution Space. SIP is a protocol for initiating, modifying, and terminating interactive sessions. This implies that SIP excels at finding remote parties to communicate with. SIP locates those remote parties through discovery. Subsequently, SIP can register sessions and fork communications. A key factor is the fact that SI is independent of the media session that it establishes.

SIP Architectural Model. Every proposed SIP extension must not violate any of the base protocol architectural concepts. Some of them are:

- *Session Independence.* The details of the media session are independent of the session establishment: path independence.
- *Multi-provider and multi-hop.* SIP expects messages to traverse through different networks.
- *Transactional.* All messages have a request / response model.

- *Proxies can ignore bodies.* An extension cannot rely on a SIP Proxy analysing and acting based on the body of a message. SIP Proxies are to ignore the body content.
- *Proxies do not need to understand the method.* No extension can rely on new methods that need to be understood by proxies.
- An INVITE must be self-compliant. Any SIP message must be susceptible of being processed. Behaviour based on collecting information across several INVITES and RE-INVITES must not be used.
- *Generality is preferred over efficiency.* It is preferable to offer capabilities covering a large spectrum of cases rather than having specialized capabilities for a small subset of scenarios.
- *The Request URI is the primary key for forwarding.* All forwarding operations must be guided by the Request URI which indicates the desired recipient of the message.
- *Heterogeneity.* No extension should have the constraint of only working if all devices support it. The extension must handle the cases when there are non-compliant UAs.
- *Other general requirements.* Any proposed SIP extension must be backward compatible with base SIP implementations. Extensions should prefer default SIP security mechanisms. The definitions of each extension should comply with SIP terminology, syntactic, semantic and document formalities as stated in RFC 4485.

One of the restrictions previously presented (proxy agnostic or independence) comes as an advantage. Since proxies do not need to understand extensions behaviour, the design of the extension needs only to account for UA requests and responses. A de facto rule for any SIP implementation is that when sending requests, the protocol must be very strict, however, when receiving them the implementation must be flexible to deal with situations when there is incompatibility. (Martinez, 2008b)

SIP can be extended in different ways: new headers, new methods, new content types. One of the most important things to consider during the design is how to handle the cases of in-

compatibility. Some of the recommended ways to deal with incompatibilities on different SIP extension types are shown next.

New headers. The easiest approach to prevent incompatibility issues on new headers is to avoid using them altogether if the requested party does not support the new definition. An UAC can be proactive and enquire the UAS for what headers it supports. This can be done by sending an OPTIONS request before any initial INVITE. This allows for the construction of the INVITE with only the supported headers or the cancellation of the request.

An UAC can also tell the UAS that a specific header is supported. When using this strategy, the UAS reply includes the supported header. If, then, the UAC replies without header support information (i.e. a regular plain request), the UAS may try replying including the header it wants to use. If the UAC replies with a *non-supported* error message, the UAS can rebuild a standard request without including the header. Instead, if the UAS does require the use of a specific header to force an extension, it can reply with an *extension required* (421) message. This behaviour is implemented using *Require* and *Proxy-Require*. The UAC use these headers to require extension support. If the UAS (or the proxy) does not support any of the required extensions, they reply with a *bad extension* (420) message. The UAC needs to handle the situation by either defaulting to plain SIP or stopping the dialogue.

Header extensions are defined through unique identifiers know as *option tags*. They are commonly used to define nonstandard extensions. An extension that has not been approved for the SIP *standard track* can, nevertheless, be invoked by using the private headers session *P-Headers.* This approach is primarily employed for in house or private applications where general compatibility is not needed and the behaviour of the protocol can be easily controlled.

New methods. The use of new methods is less flexible than the use of new headers. The UAC and the UAS must agree on supporting any new method. Agreement can be achieved by

having the UAC checking against the UAS before sending the first request. As previously explicated for headers, either an OPTIONS or the publishing of a list supported methods in the initial request, are mechanisms an UAC can employ. In the latter, an ALLOW header is appended to the request.

New content. New content is handled similarly to new methods. An UAC only uses a new content type once it confirms that the UAS supports it. An UAC may send an OPTIONS message, then the UAS response might include *Accept*, *Accept-Encoding* or *Accept-Language* headers to represent that a content type, encoding or language are supported. An UAC may also send a request to indicate its own supported content types to which the corresponding UAS should reply reporting which one, if any, it supports or prefers. Optionally an UAC request may also include a *Content-Disposition* header to label the content type as optional (Martinez, 2008b).

2.3.2 Representative SIP extensions

After a revision on the basics of extending SIP, a review of some representative SIP extensions follows. These SIP extensions have been chosen because they are currently being used in commercial applications and are relevant to the SIPBIO protocol.

2.3.2.1 SIPREC

SIPREC is one of the most successful SIP extensions. It aims at enabling recording of media communications. SIPREC was developed as a joint effort of several telecommunications companies: Cisco[4], Nice Systems[5], Genesys[6], Avaya[7] and Unify[8]. SIPREC is defined in RFC 7866 (Portman *et al.*, 2016).

[4] www.cisco.com

[5] www.nice.com

[6] www.genesys.com

[7] www.avaya.com

[8] www.unify.com

SIPREC fits especially well within the context of operation of Session Border Controllers (SBC). An SBC is a highly specialized firewall device that allows proper routing and transcoding of SIP communication between different networks. They usually sit at the border between an ISP or Telephony Service Provider and a corporate data / telephony network. Since all communications necessarily pass through them, they are in a privileged position to intercept and manipulate packets as required (Rehor *et al.*, 2011).

A review of the SIPREC architecture as described in RFC7245 by Hutton *et al.* (2014) follows. The basic component block of any recording system is the ability to obtain a copy of the media to be recorded without disrupting the original intended communication. Recording of VoIP calls usually requires obtaining a copy of the original RTP stream associated with a metadata set. SIPREC defines two UAs: logical functions Session Recording Client (SRC) and Session Recording Server (SRS). The SRC purposefully sends media packets to the SRS. The SRS should be able to concurrently receive media and identify its metadata from multiple sources. The role of SRC can be taken by any device capable of acting as a UA: SIP Phone, SBC, SIP Media gateway. Any UA must be able to deliver media and metadata in real time to the SRS. The recording session should be independent of the original communication session being recorded (Hutton *et al.*, 2014). SIPREC can provide a recording indication in the SIP requests and responses of the recorded communication session (Kyzivat *et al.*, 2016).

There are several possible topological distributions for SIPREC enabled systems. Only the two most widely used distributions are discussed in this document. In the first topological distribution (shown in Figure 2.24) a Back-to-Back User Agent (B2BUA) acts as the SRC. This is usually the case of the SBC-based recording systems. Either the SRC or the SRS may decide to start a recording session by sending an INVITE to the other party. Such an INVITE must contain enough information for the receiving party to understand that the session will be initiated with recordability. The sender must also prevent the session from being sent to an unintended UA. The SRC may notify UA A and B that their session is being recorded.

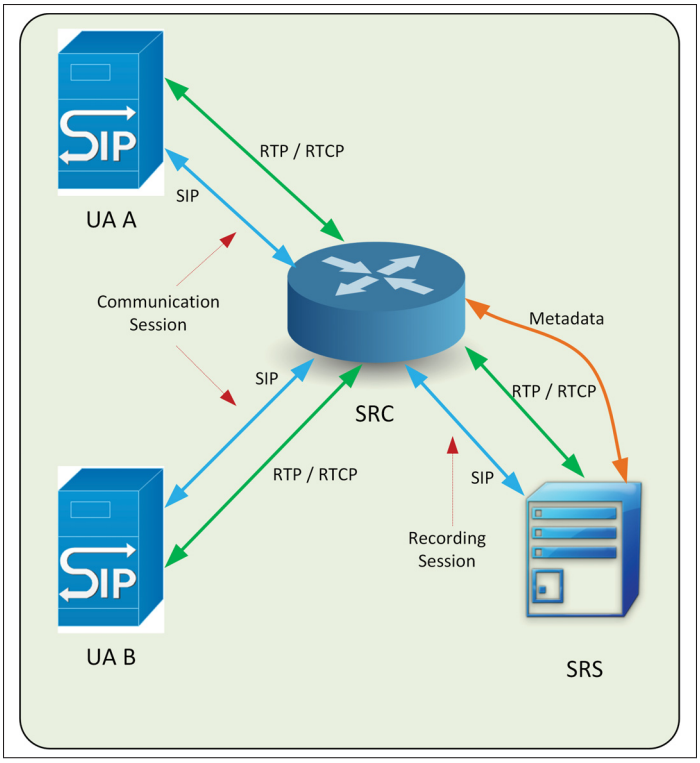


Figure 2.24 *B2BUA as SRC.*

A second SIPREC topology distribution is built when one of the UA takes the SRC role as illustrated in Figure 2.25. It is the responsibility of the SIPREC-enabled UA to send both media and metadata to the SRS. The recording session can be initiated by the UA or by the SRS, depending on the solution design (Kyzivat *et al.*, 2016).

Recording session establishment is completed in a similar way in either topology. In both cases media characteristics are negotiated through a regular SIP process. As noted, the SRC or the SRS can initiate the recording session. The process of communication establishment is very similar (and surprisingly simple) in either case. It is outlined in figures 2.26 and 2.27 (Ravindran and Kyzivat, 2016).

When the media is replicated, it can be mixed or separated into two media streams: one for the caller and the other for the receiver (Ravindran and Kyzivat, 2016). This ability is key for voice biometrics where each party RTP stream needs to be clearly identified.

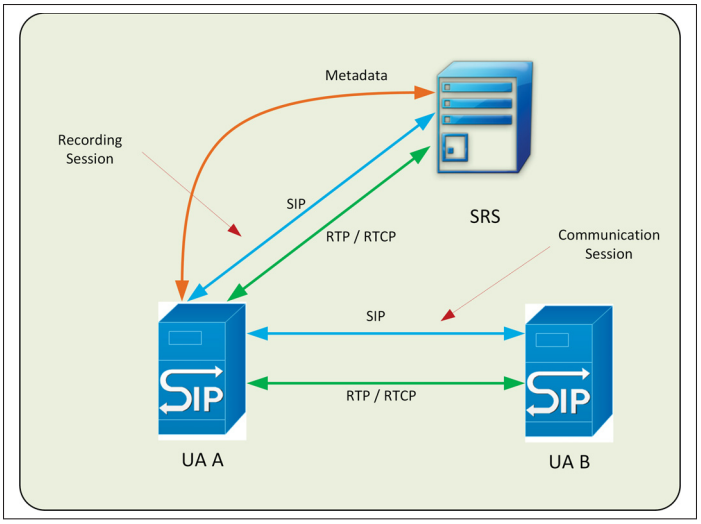


Figure 2.25 *UA as SRC*.

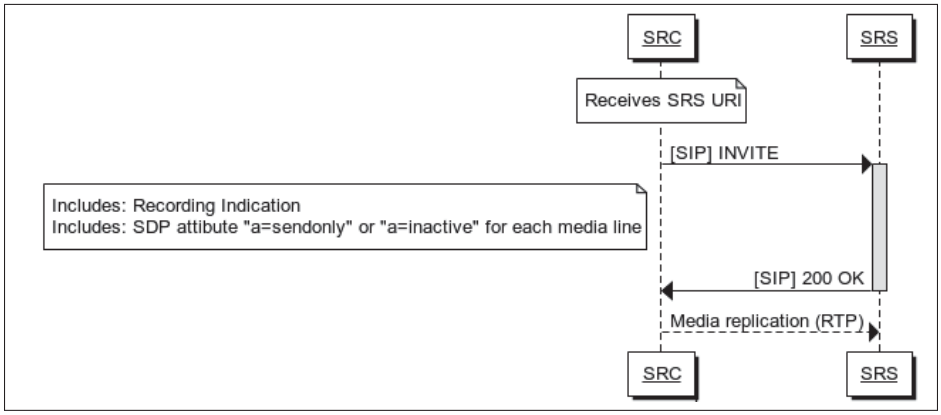


Figure 2.26 *SRC initiated recording*.

Besides providing media content, the SRC is also responsible for providing metadata to the SRS. This is usually achieved through a series of metadata snapshots (similar to call states). The first snapshot is sent on the initial INVITE. All subsequent updates are sent as Re-INVITE or UPDATE messages. All metadata is transported in the body of the messages. A special case would be the media stream attributes present in the SDP of the recording session. Metadata is generated by a predefined set of constituent block classes that serve as the constructor of the protocol objects. These blocks constitute the base of the recording call flows as defined by Kyzivat *et al.* (2016).

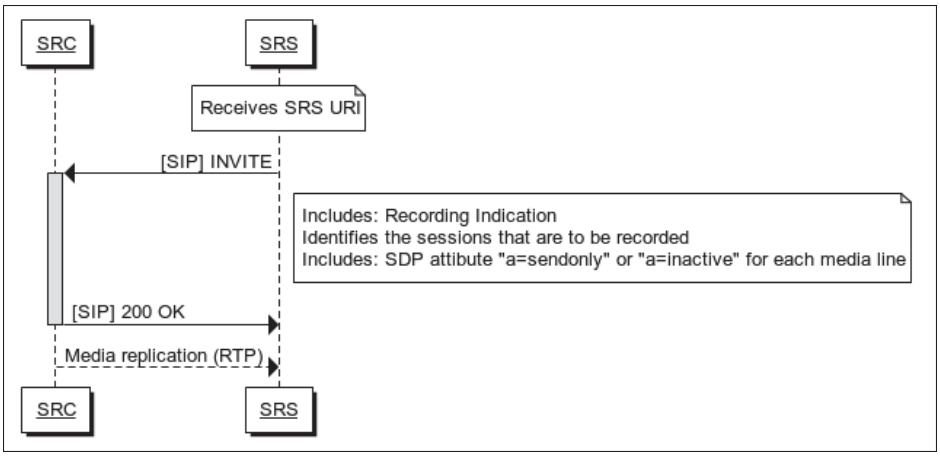


Figure 2.27 *SRS initiated recording*.

SIPREC has already inspired the development of related protocols. The one that has gained most recognition is the one proposing the recording of *Message Session Relay Protocol* (MSRP) sessions. This Internet draft proposes using SIPREC to enable recording of text messages on MSRP. The recording is achieved by defining a new MIME type (media type) to wrap messages in a MSRP communication session. However, the latest reported state of the document proposal is an early draught of unknown viability (Yan and Kyzivat, 2015).

2.3.2.2 SIP Extension for payments support

With the increased popularity of streaming technologies, there is a pressing need for an easy way to charge customers for the use of multimedia services. The most popular mode of accessing multimedia streaming services is through subscriptions to a company that provides media channels or specific pieces of multimedia content in a pay-per-use fashion. Subscribers pay a monthly fee or a flat price. Access control is based on the number of concurrent sessions, number of downloads, streaming times and, possibly, geo-localisation. In all cases, subscribers are required to have a different account per multimedia provider (Friedlander, 2015).

A SIP extension to support payments is analysed in this section. Familiarity with this proposal helps clarifying some of the requirements of a SIP extension to support biometric authentication.

Ruiz-Martinez *et al.* (2016) propose a SIP extension to handle payments in a generic way. They define new headers, tags and contents for the body part of the message. Notably, this SIP extension proposal meets requirements for Digital Rights Management (DRM), supports negotiation through the offer / answer model and allows price-to-quality differences in the initial offer and on the additional payments whilst the session is still in progress. SIP payments must comply with, at least, the requirements listed next.

- Association of payment information with multimedia session description.
- Price offering based on quality.
- Support of different payment methods and price differentiation per method.
- Price negotiation.
- Receipt processing and handling.
- Loyalty information.
- Keep number of connections at a minimum.

As it is usual in SIP, the offer can start by the UAC sending an INVITE with a multimedia description of the supported payment options. The UAC may send an OPTIONS request to the UAS to find out what methods are supported and, consequently, prepare the initial INVITE. The UAS replies to the first INVITE with a 200 OK containing a description of all supported payment methods. The UAC completes the three-way handshake with an ACK indicating the payment. All payment information is contained inside an XML structure. To support price negotiation and processing of loyalty information, intermediate steps between the first SIP INVITE and the 200 OK can be inserted. These intermediate steps are implemented with the use of 183 PRACK and OK(PRACK) messages. The UAC can easily indicate to the UAS if it wants to start a price negotiation by using the optional tag negotiation in the INVITE. Provisional responses PRACK and OK (PRACK) have been also extended to support the transport

of the negotiation details in the also extended SDP body. Support for additional payments is also included by using a new SIP MESSAGE method that contains the time left for the current session as well as the information to start a new payment process. Figure 2.28 shows the exchange of SIP messages for a typical use case.

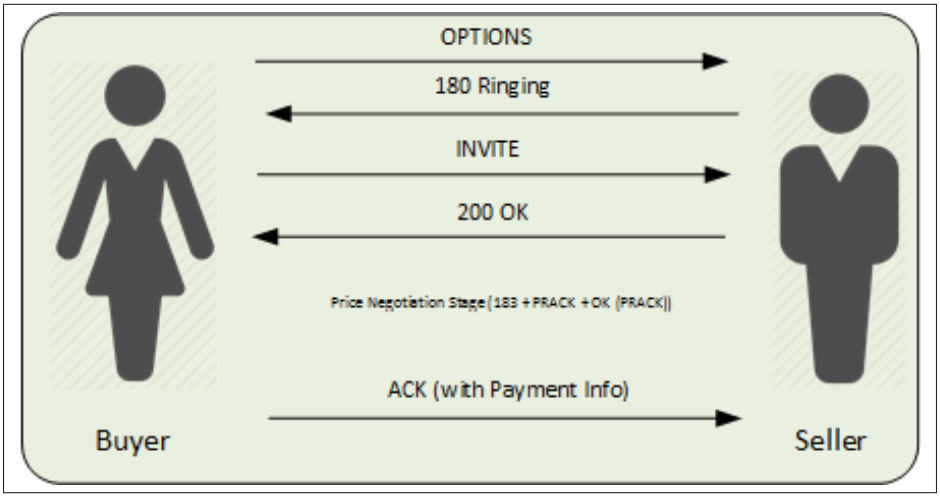


Figure 2.28 *Simple Payment Process.*

In terms of security, the proposal does not include any specific capability relying instead on the security of the payment protocol and the security mechanism of the underlying communication: SIP authentication, IPsec, SIPS, TLS, etc.

The elements of the proposed extension can be summarized as follows:

SDP extension. SDP is used to specify the characteristics of the content to be transferred. A new attribute, *payment-info*, was defined to hold the information required during the message exchange. The actual payment information is contained in the form of an XML element with all the transaction details.

Option-tags. Standard SIP uses *option-tags* to inform about pieces of information that may be needed or required during the multimedia session. There are three related headers: *Required*, for those features that are mandatory to be supported; *Supported*, for

those additional features that the UA also supports but are not mandatory; and *Accept*, which indicates which headers are accepted. The payment proposal defines the following option-tags:

- *payment.* It is mandatory for the Required header. If the UAC does not specify it, the UAS will return a 402 Payment Required message.
- *additionalpayments.* It is an optional *Supported* header used for the UAC to indicate that it wants to make additional payments at any point during the session.
- *negotiation.* It is an optional tag for both, the *Required* and the *Supported* headers. It is used for the UAC to indicate that it wants to negotiate the price and / or the quality of the media. For instance, in the case of music or video streams, the UAC may be willing to sacrifice quality to save on price.
- *application/sippayment.* It is an optional tag in the *Accept* header used for a UA to indicate that it supports exchange of payment information.

Application/sippayment content. It is a new proposed content type which is used to exchange payment messages. It is defined as an XML structure called *PaymentInformation* that represents an element of the *PaymentInformationType* type (defined as an XML schema). These elements are included either as an attribute of the SDP content (*payment-info*) or as the content itself (*application/sippayment*) of a SIP message in the payment exchange. All elements in the XML schema are optional according to the need of the transaction.

By extending SIP and SDP, plus the definition of a new XML structure, this proposal constitutes a solid alternative for a payment exchange system (Ruiz-Martinez *et al.*, 2016).

SIPBIO leverages many of the concepts of this proposal, building upon it for the purpose of biometrics.

2.3.2.3 Other SIP extensions

The architectures of two SIP extensions: *SIPREC* and *SIP Payment Support* have been analysed. The way the SIP integration proposal works has been described above. These extensions have been chosen because they reflect a similar theory of operation to that of SIPBIO. Due to space constraints additional SIP extensions are not included in this review. However, a brief mention of two more SIP extensions is presented to reinforce some of the concepts this proposal relies on.

A proposal by Gurbani and Sun (2004), although already superseded by newer developments in telecommunications, constitutes an interesting attempt to integrate traditional telephony and digital services. Gurbani and Sun (2004), proposed an extension of the SIP protocol to support telecommunications services. Their objective was to enhance cellular telephony to offer a richer set of services. In 2004 those services were being offered by Internet-based operators. At that time, mobile offers were mainly based on 2G and 2.5G (in some markets there was an early introduction of 3G). It was generally thought that it was more reliable to offer services based on the architecture of the mobile network. The offering of Internet-based services was limited to the constraints of event-handling on the mobile section of the communication path. An advantage of the approach was that no Internet connection was needed to access any covered service. The proposed architecture achieves this level of isolation by separating Internet and mobile-based services with an *Event Manager* that works like a proxy gateway service. The system leverages entities called *Detection Points* to which Internet hosts subscribe in order to receive events generated at the mobile side. Whenever an event occurred (e.g. a new call) the Detection Point would be notified and the *Event Manager* (which is aware of all events) would notify the Internet Host in its subscribers list (Gurbani and Sun, 2004).

To achieve a SIP compatible event management, this SIP extension proposes two new SIP event packages and a new MIME type payload. The resulting extended protocol leverages SIP asynchronous event notification capabilities between entities: *subscribers* and *notifiers.* The message exchange between *subscribers* and *notifiers* follows RFC 3265 (Roach, 2002). The

newly defined SIP event packages are labelled *spirits.INDPs*, which correspond to call events, and *spirits.user-proof* which corresponds to all other types of events. The characteristics of the events are passed as a payload in an XML formatted extended MIME type (*application/spirits-events+xml*). It also defines a XML schema that represents the Detection Point and its associated parameters. Every time there is a message exchange (i.e. subscription, event notification, etc.) a XML payload is created holding all required information. XML has the advantage of being, by design, easily extensible.

No further review is given of this proposal, the objective was to illustrate how SIP extensions have been used to add additional functionality to SIP with the specific aim of replacing or complementing existing communication protocols on carrier infrastructures.

The last SIP extension analysed in this review is the *SIP Device Discovery in Future Service Platforms.* This proposal, intends to deliver customised media services using SIP for signaling and service selection. The architecture is designed around a *Service Delivery Platform*. This platform is tasked with providing requested services for UA. The proposal uses the possibility of indicating UA capabilities in SIP as Huston *et al.* (2004) proposes in RFC 3840. Every UA sends information as an XML payload. The proposal extends the OPTIONS method with the ability to query the capabilities of a registered UA. It also extends the MESSAGE method to carry the device and media descriptions. Finally, SIP headers are extended to distinguish between extended and default methods. The description header is extended with a new field for mixed purposes as enumerated next (Chen *et al.*, 2012).

- *Complete device description upload.* After the standard SIP registration, the device sends a MESSAGE request to a media registration server, with XML content carrying all required information. It can be compared to a second registration against that of a service provider.
- *Request / Response for list of registered UA.* A UA may query the media registration server for a list of available, previously registered, devices. Further messaging can be requested out of the resulting list to mine further information on any device. All subsequent requests use the OPTIONS / MESSAGE combination.

- *Multimedia streaming notification.* A MESSAGE is used to notify on a media stream. The header description would denote that a streaming session is in progress.

The key concept of this SIP extension proposal is the possibility of having a bank of registered devices with different media capabilities to choose from. Usually a single SIP implementation allows for a closed set of available media properties. This SIP extension allows an UAC to select any UAS based on its published capabilities without needing to query it (Chen *et al.*, 2012).

Chapters 2 introduced SIP and its extensibility. Chapter 3 presents the evaluation methodology proposed for the SIPBIO extension.

CHAPTER 3

REQUIREMENTS AND USE CASES

In this chapter the basis of the SIPBIO proposal is presented. A detailed description of a biometric interaction is followed by a list of the use case scenarios covered by the proposal. This is followed by a compilation of SIPBIO requirements and then its explanation in the context of the SIP protocol stack.

3.1 Canonical biometric process

The industry has not defined a standard biometric process. However, general algorithms for the most basic functions of *enrolment* and *authentication* are broadly accepted. This section introduces these algorithms in their most common form. The main parties in a biometric interaction are known as the *biometric client* and the *biometric server*. They are referred to here as the bio-client and bio-server.

Enrolment. Before a user can be authenticated, they have to be registered through an enrolment process. The steps followed during this procedure are shown next.

1. The bio-client requests a user's status from the bio-server who replies with a negative result. This step is not part of the enrolment process but is included for clarity.
2. The bio-client sends a payload collection event to the bio-server that replies with the result of the collection.
3. If the result of the collection is successful, the biometric print is created and the registration process is completed.

Verification. A registered user can be verified through the analysis of a fresh biometric payload. The results are to be compared against the one on record.

1. The bio-client requests a user status from the UAS. The bio-server replies with a positive result.

2. The bio-client sends a *payload collection event* to the bio-server that replies with the result of the collection.
3. If the result of the collection is successful, the biometric verification is completed, and the bio-server sends the result back to the bio-client.

These general biometric algorithms are not complex. For biometric implementations, the difficulties lie on the challenges associated with the implementation: data collections, payload quality and security. A biometric implementation should also include intermediate steps to deal with other outcomes such as uncertain results or unexpected payloads, etc. Figure 3.1 shows a schema of the general process. Figure 3.2 shows the corresponding sequence diagram.

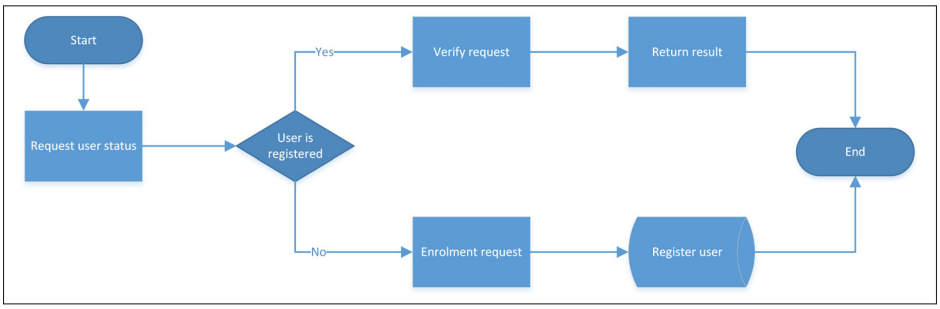


Figure 3.1 Biometric process flow

At this point no relationship is intended between the biometric algorithms and the SIP. The terms bio-client and bio-server are used only to establish a parallel between them and the corresponding UAC and UAS entities. Even if repetitive, it is important to clarify the role of each entity in the biometric exchange.

bio-client: UAC. This is the client application. It features either a biometric payload reader or close relationship with a separate device holding one. Depending on the implementation, it can be a light application to interface customer data, send requests and display retrieved

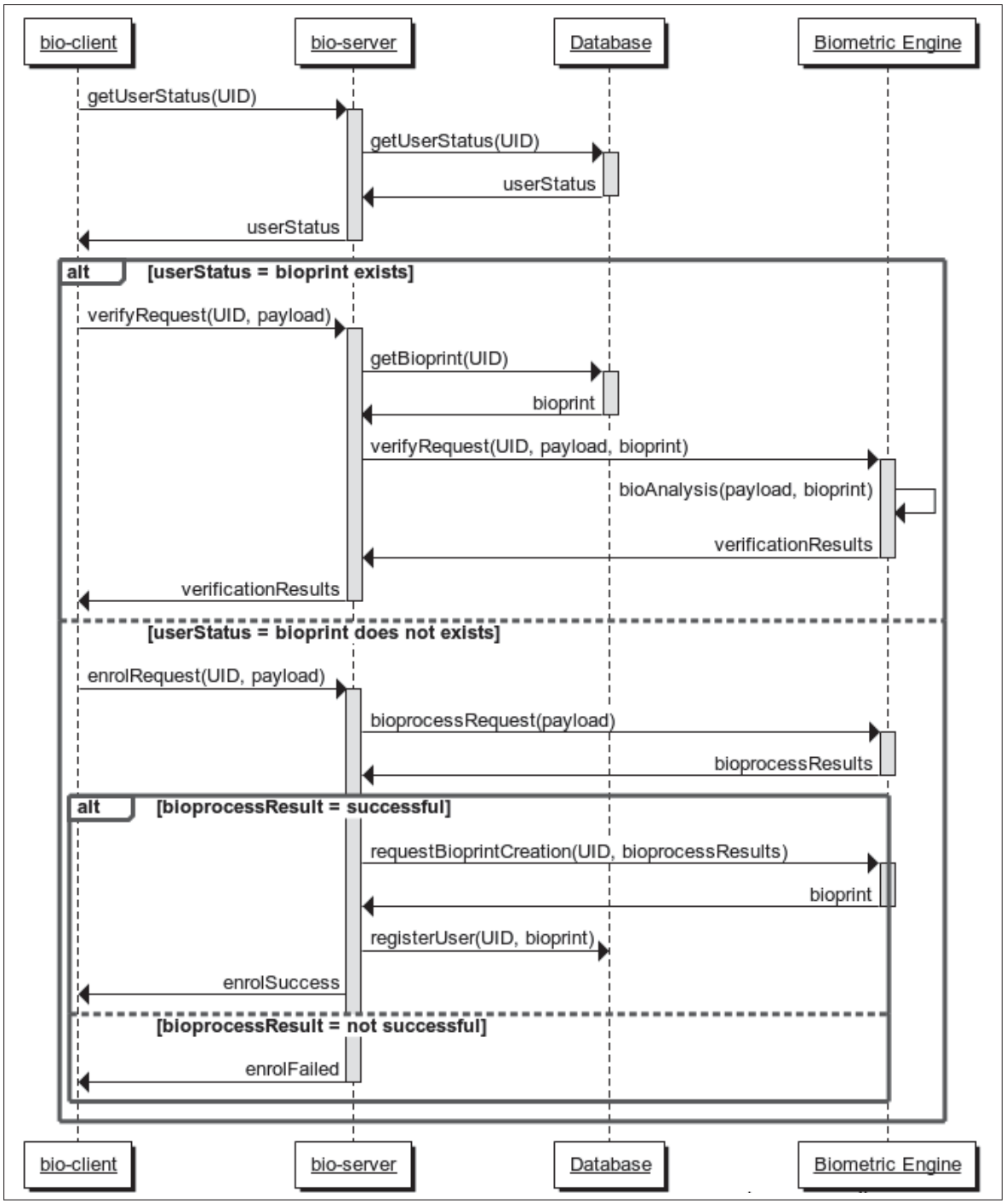


Figure 3.2 Biometric process sequence diagram

results; or it can be a full application server where biometric interfacing is only one of its functions.

bio-server: UAS. This is the application server. It receives biometric requests that it needs to properly route to a biometric engine. It needs to be "intelligent" enough to build requests based on requested information, complemented with data gathered from other sources (e.g. a database).

Database (DB). This is the entity that holds bioprints. Most implementations use it to hold configuration and application states.

Biometric Engine. This is the element that performs biometric analysis. It should be able to receive a digital representation of a biometric payload, analyse it and return either a bioprint or a score.

The sequence shown in Figure 3.2 includes not only the communication between the client and server entities but also depicts active process components: the database and biometric engine from an event and response perspective. This sequence allows the expansion of the simplified enrolment and verification processes into a full implementable algorithm. Once again, no direct SIP representation is intended at this point. Communication between the actors can be completed through any available channel like a TCP socket or an HTTP.

1. The bio-client sends a request to a bio-server to inquire about the status of a user identified with a certain user ID: the claimed ID. Note that the bio-client itself should not get this information directly from the database. This job should be left to the bio-server.

2. Since the bio-server does not hold user information it queries the database for the status of the ID.

3. The database replies to the bio-server with the user status.

4. The bio-server receives the status and informs the bio-client accordingly.

5. If the ID is already associated with a bioprint:

a. the bio-client sends a verify identity request for the claimed ID along with the payload to be analysed. It depends on the biometric application what the payload actually is. It could be a reference file (usually in XML format), or the actual digital representation of the payload.

b. the bio-server retrieves from the database the bioprint associated with the claimed ID.

c. the bio-server builds a verify request for the biometric engine to process. The request is built in the format expected by the biometric engine (HTTP API, SIP, TCP socket, etc.). The request ought to contain the claimed ID along with its correspondent bioprint, and the digital representation of the payload to be analysed. This particular process is only one of the possible implementation alternatives.

d. the biometric engine sends the results to the bio-server.

e. the bio-server forwards the results to the bio-client. The bio-client, the bio-server, or both should implement the logic to process different outcomes. Most biometric engines provide a certitude rate to support the implementation.

6. If the ID is not associated with a bioprint:

a. the bio-client sends an enrol request to the bio-server for the claimed ID along with its associate payload.

b. the bio-client asks the biometric engine to analyse the biometric payload. The analysis often includes assessment of not biometric features: size, quality, format, etc. The biometric engine sends the result back to the bio-server.

c. If the biometric payload complies with all required characteristics:

i. the bio-server requests a bioprint creation to the biometric engine. The bioprint is a binary representation of the biometric characteristics in the payload.

ii. the biometric engine sends the bioprint to the bio-server.

iii. the bio-server registers the bioprint on the database. A user ID may have more than one associated bioprint.

iv. finally, the bio-server reports a successful enrolment to the bio-client. An enrolment is considered successful when the user registration is completed, and its biometric print is saved in the database.

d. If the result of the biometric analysis is not successful, the bio-server sends an enrolment error to the bio-client who should have the necessary logic to handle the situation.

3.2 Use case scenarios

The process shown in Figure 3.2 is the general representation of a biometric event. This section shows cases as seen in informatics security.

Biometric solutions are classified into three types: active, passive and off-line.

- **Active.** It refers those cases where the user intending to access a service actively interacts with the system collecting the biometric payload. Some classic examples of active biometric systems are:
  - A finger print reader.
  - A person interacting with an IVR in a call centre.
  - An iris scanner.

  In all these cases user are aware of providing their information to a system.

- **Passive.** It refers to cases where biometric systems collect information without the user actively participating in the action. Users may or may not be aware of the biometric payload being collected but they do not need to change their behaviour or execute a particular action to provide the information. Typical cases of passive biometric systems are:
  - Security cameras in a public space.
  - A call recording system.

  - A behavioural typing system.

- **Off-line.** Off-line analysis is the exercise of analysing biometric payloads that have been previously collected. It is mainly used for fraud detection and identification purposes. For instance, when a biometric payload is required to be compared against a collection of existing identified payloads to find a match. This use case is frequently found in the work of public safety investigation agencies.

Some key differences between the biometric types are:

- the way the biometric payload is collected. In active systems the biometric payload is directly sent from a provider device to the biometric component able to process it. Passive and off-line biometrics require an architectural design able to collect the payload from third-party systems.
- the way the biometric payload is sent to the biometric analysis server. Active systems gather the payload and send it along with the biometric processing request to the biometric analysis engine or its front end. Passive systems usually provide indications of where the payload is located and how it is going to be provided. In other words, additional components are required to access, classify and provision the payload for the biometric engine to analyse.
- the way the user identification is included in the process. In active and off-line systems, the claimed ID is collected from a third party system automatically as part of the interaction process. In passive systems, there is usually a trusted party (a person / customer representative) required to guarantee the claimed ID matches the claimed identity. For instance, if the claimed ID belongs to a woman but the person making the claim is a man, there would be a contradiction that the trusted party should be able to identify (Reid, 2003b; Martin, 2013).

The next following subsections describe some common scenarios of biometric interactions to illustrate use case scenarios in detail.

3.2.1 Scenario 1: One-time Active Voice Biometrics authentication (OTAVB)

Active Voice Biometrics is the process of a speaker (user) consciously interacting with a system that guides them through a set of stages to complete a given process. For instance, a user calls a support centre, his call is answered by an IVR system that proceeds to direct them through a series of prompts to achieve authentication. Providing the user has already enrolled with the system, they are asked to repeat a passphrase. The collected spoken audio is processed and the biometric result is compared against a stored voiceprint. Since the biometric process is of a statistical nature, the result is never expected to be 100% match with the original voiceprint. It is up to the company using the biometric system to define which *False Acceptance* (FA) and *False Reject* (FR) rates are acceptable. If the verification result of the operation falls within the accepted margins the authentication is considered successful (Jain *et al.*, 2006).

An important aspect of the active authentication process is the passphrase the user is expected to repeat with the same biometric properties that the passphrase used during the enrolment process. In some instances, the user may be asked to repeat some random but predetermined phrases to rule out the possibility of a playback attack. Active biometrics is also characterised by the user being fully aware of the procedure and actively participating in the steps to complete the biometric process. For instance, a digital finger print requires the active participation of the user. The user needs to purposefully place a finger or thumb onto a fingerprint reader. In this active process the user is guided through the steps. In the specific case of voice biometrics, an IVR guides the customer to repeat a passphrase leading to either their enrolment or authentication. Usually, in active biometrics, the gathering of the biometric payload is simple: the user is requested to provide it through a specific technical mechanism that collects and sends it to the *bio-server* in the required format for its analysis (Sui *et al.*, 2012).

The authentication process is executed at the beginning of the voice interaction and the result will determine if the user is given direct access to the offered service or if a further authentication stage is needed (e.g. forward the call to a customer advisor for further authentication). Figure 3.3 shows a verification process and its corresponding enrolment steps. Note that the

sequence is based in the case of voice biometrics where an IVR directs biometric requests to the biometric system.

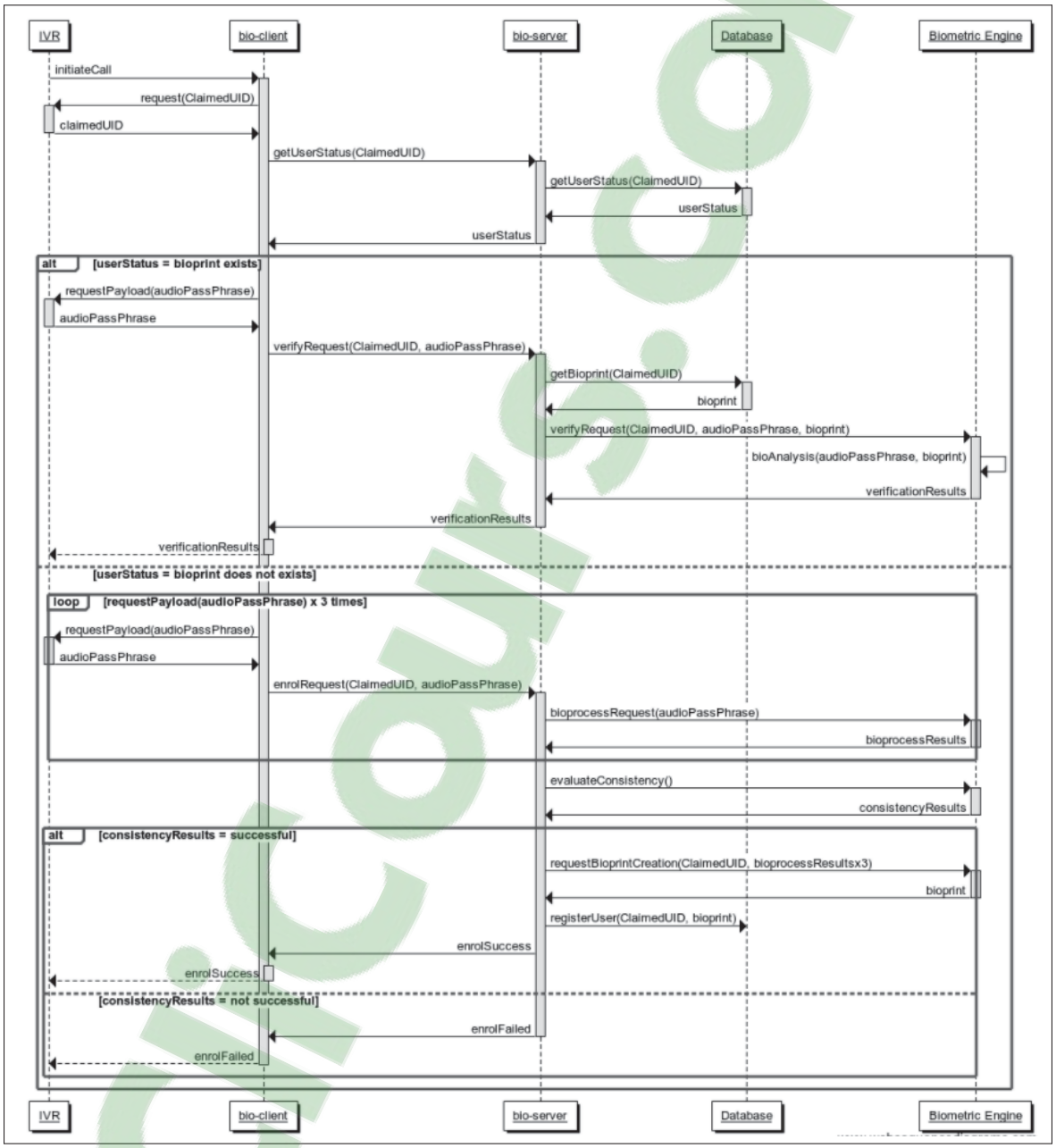


Figure 3.3 *Classic enrolment / verification sequence - active biometrics.*

3.2.2 Scenario 2: One-time Passive Voice Biometrics authentication (OTPVB)

Passive Voice Biometrics is the process of analysing the voice of a speaker without direct user interaction with the biometric system. The user payload is collected and analysed in the background whilst the user interacts freely with another person (e.g. a customer calling a call centre would have his voice captured for voice biometrics analysis). Contrary to the active biometrics process, the user is not expected to follow a specific matching key sequence (like a passphrase in a voice biometrics scenario). The system instead analyses the biometric characteristics of the provided payload while the user participates in a regular interaction. The associated enrolment process for passive biometrics requires more data in the payload to complete. During the authentication process, the payload collected is evaluated and compared with the stored bioprint (Desai, 2016).

The rest of the process is similar to the active biometrics previously described: if the biometric results are considered successful per the predetermined FA / FR rates, the user is given access to the service, otherwise the user is guided to another system for further authentication. For the specific case of passive voice biometrics, an abstraction of the flow is shown in Figure 3.4. As with all other mentioned biometric processes, each implementation may introduce variations to that flow.

3.2.3 Scenario 3: Discrete Intervals Passive Voice Biometrics authentication (DIPVB)

The DIPVB process differs from OTPVB in one way: the authentication is repeated at discrete intervals to guarantee that the same person is using the system through the entire length of the interaction. OTPVB can be thought of as a special case of DIPVB where the number of discrete intervals is one. This method is seldom used since it requires more system resources and it is usually safe to assume that, at least for the case of voice biometrics, a customer advisor should be able to recognize if there is a sudden change of customer voice on the line. This method is used in high security environments or when the user is not interacting with a trusted party

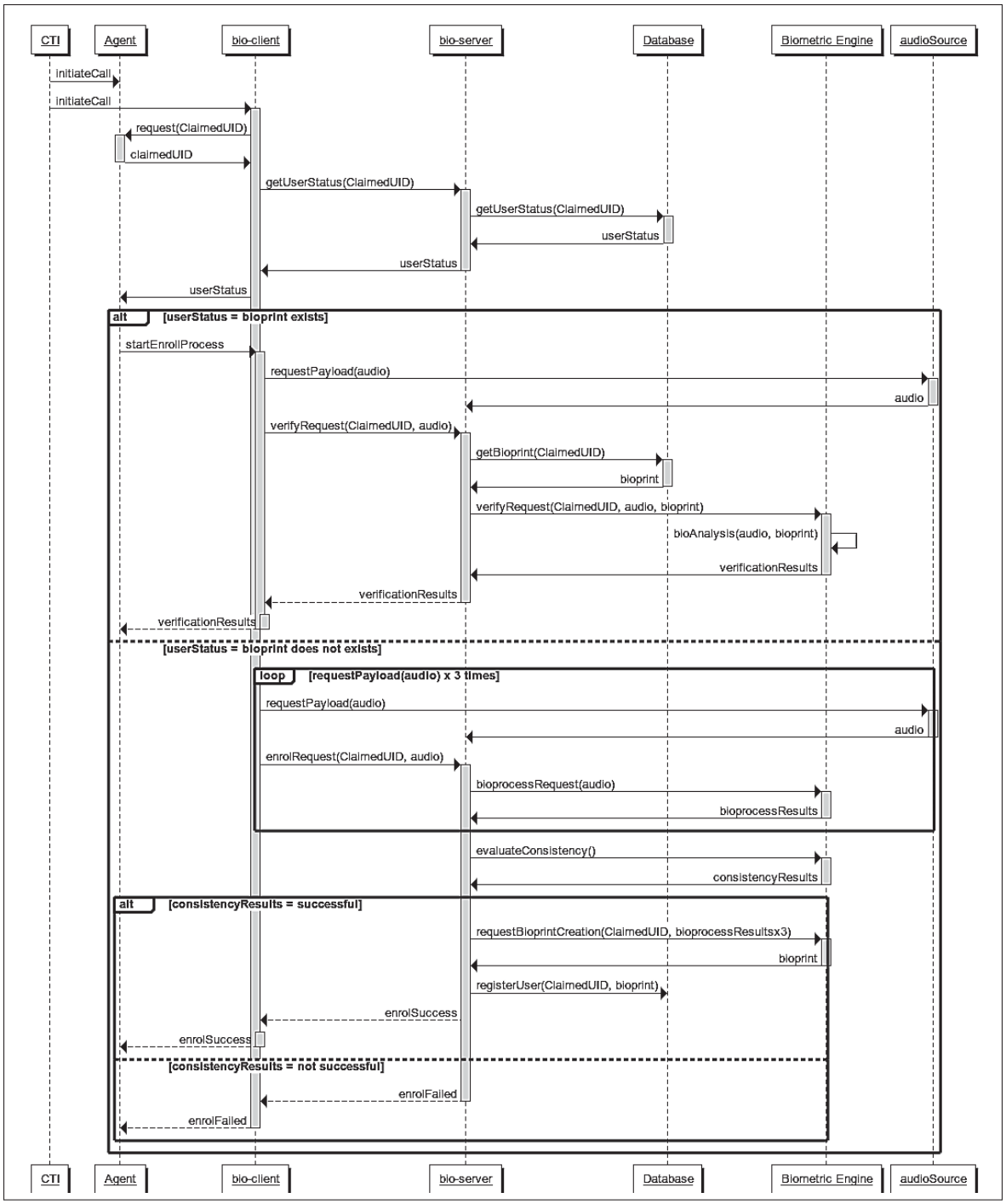


Figure 3.4 *Classic enrolment / verification sequence - passive biometrics.*

so is unable to confirm that the identified individual remains the same during the interaction (Saevanee *et al.*, 2015).

3.3 Simplified biometric distribution

Until now all components have been described in real-time systems that use a biometric interaction. This section simplifies the description to make it approachable from a SIP design perspective. It has been mentioned that SIPBIO aims at standardising the interactions between components in a biometric interaction. Although any component can be upgraded to support SIPBIO, some systems have established communication protocols. It is the work of the SIPBIO designer to blend seamlessly into any environment. The sequence shown in Figure 3.1 can be condensed into the one shown in Figure 3.5 where only two entities the UAC (bio-client) and UAS (bio-server) are pictured. All intermediate interactions are integrated in the SIP side of the design.

A condensed representation allows a simplification of the model. The sequence in Figure 3.5 shows how some of the interactions are abstracted behind the concept of the *bio-server* previously defined, but which takes a lead character in a condensed flow. An implementation must allow for the *internal* tasks required to complete bio-client requests. This could be a biometric engine that exposes a set of HTTP-based APIs. In such case, a segment of the focus interaction may appear as the flow extract shown in Figure 3.6. Note that bio-client to bio-server communication is using SIP, as proposed by SIPBIO, while requests to the biometric engine use HTTP. This is one of multiple options.

The concept of biometric engine interaction and SIPBIO endpoint separation is key to avoid confusing the responsibilities and tasks assigned to each of them. SIPBIO endpoints are used is to request biometric operations or respond with the results of such operations. However, SIPBIO endpoints do not process the biometric payload, their role is messaging, not the biometric analysis itself. The endpoint holding a SIPBIO endpoint must also have a component that interacts with the biometric engine. Those two operations are to be kept separate to avoid one depending on the other. Biometric engines evolve themselves but the basic operations of analysing a payload and provide biometric results is universal and will not be altered. The

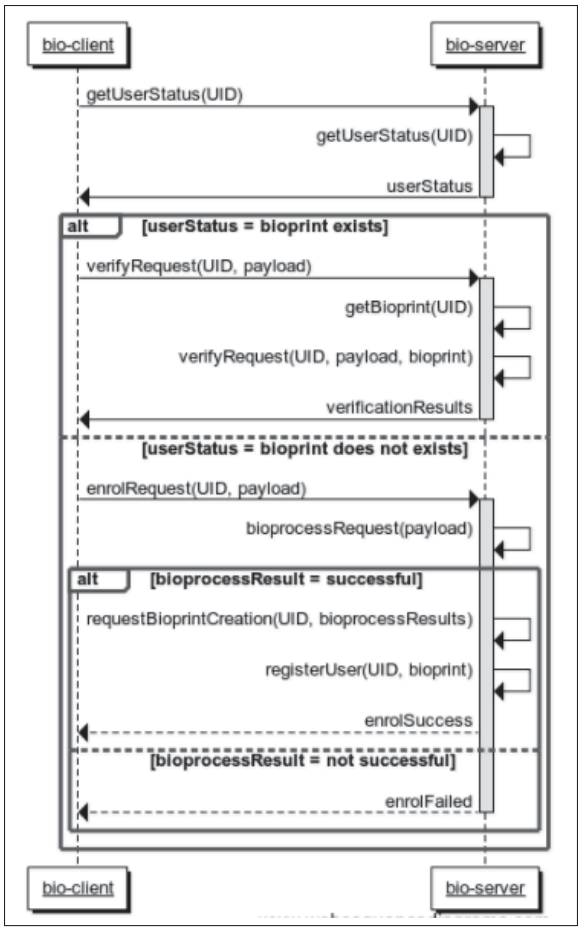


Figure 3.5 *Biometric flow condensed.*

designer must make sure there is communication between the two sides (components) while providing them process independence.

This abstraction subsequently allows for a clear transition in the naming conventions to UAC to UAS as shown in Figure 3.7.

In any case, it is key to remember that UAC and UAS are naming conventions to represent two actors conversing by SIP, one being the initiator and the other the responder.

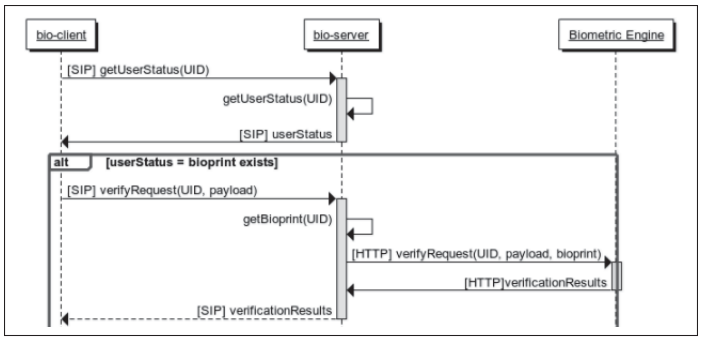


Figure 3.6 *Biometric flow with associated HTTP APIs.*

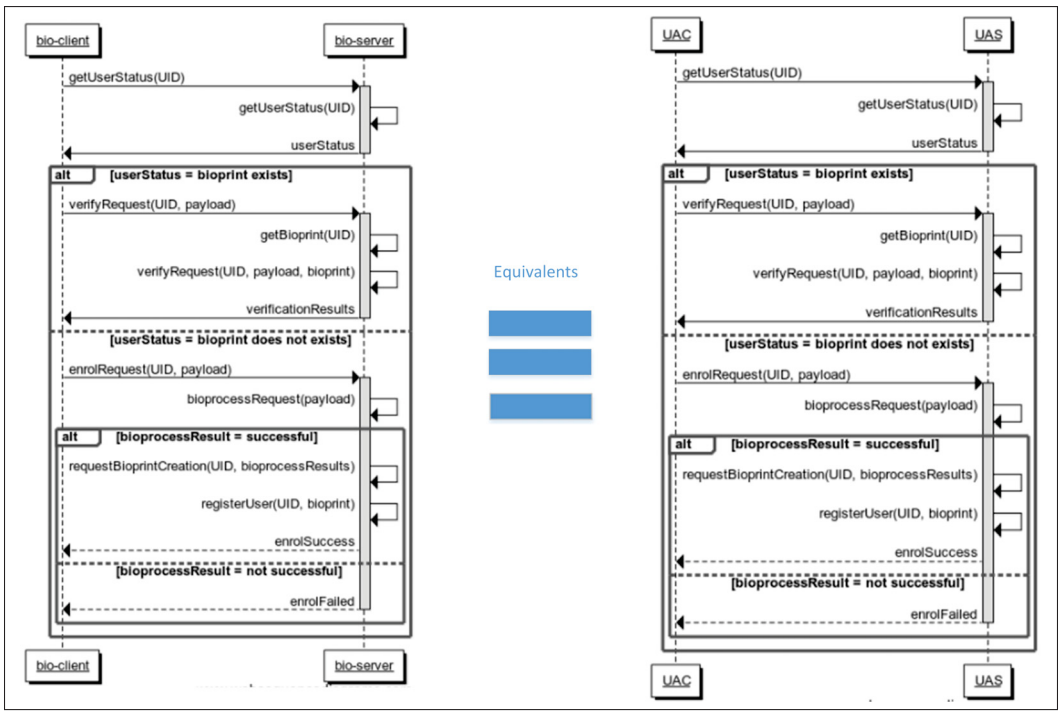


Figure 3.7 *Naming equivalence for biometric actors.*

3.4 Extended distribution

Section 3.3 is an exercise to understand the concept of entities assuming a SIP enabled role in the communication establishment process. This section intends to extend the concepts to real life scenarios in order to better acknowledge the upcoming SIPBIO protocol requirements.

Figure 3.8 depicts how a SIPBIO can be introduced to a customer self-service system in a call centre or mobile authentication environment. Note that the inter-network separation element can either be telephony or network oriented: an Internet router, a firewall, an SBC, etc. The authentication payload element can either be a voice, face, digital print, etc. In general, any payload for which a capturing device and a digitalization method exist can be accurately represented by Figure 3.8 schema. Although every chosen interaction between two different elements can be set up through SIP, interactions are represented under the most commonly used protocol to date as per thread type (e.g. HTTP, SIP, etc.). Note how the self-service system is the coordinator of transactions between different components with the bio-client application server being the active element facilitating interactions. This model would allow the introduction of a universal session identifier for all SIP-based communications as proposed in RFC 7329 (Kaplan, 2014). The first active element receiving a SIP interaction would introduce a universal SIP identifier that would be consequently handled by all other elements in the path. The bio-client should be able to understand and keep the identifier with the SIPBIO session and all other coordinated SIP sessions. Every element would be part of the same interaction.

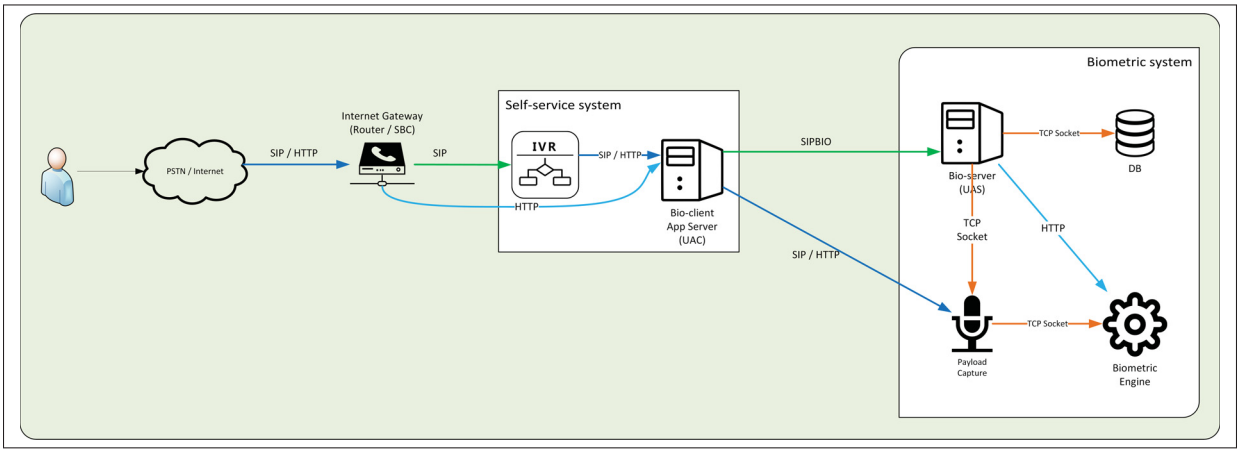


Figure 3.8 *Extended distribution active biometrics example*.

This section does not intend to repeat the flow diagrams already presented. However, to make it clear the interactions in 3.8 are itemized next:

1. A user initiates a contact to a service provider. If a voice contact, the communication is initiated through PSTN SIP trunks. If not a voice contact, the communication is likely to be HTTP. Should the client be RFC 7329 compliant, it should include a universal SIP ID.

2. Depending on the nature of the incoming request, the border element (Internet router or SBC) forwards the request to a self-service system via SIP or HTTP. If this active element is RFC 7329 compliant, it would either handle a received universal SIP ID or insert one into the path if relevant.

3. The self-service system includes the bio-client application server and may also include an IVR. Communication between the elements in the self-service system is accomplished through SIP or HTTP. If none of the previous active elements has introduced a universal SIP ID, the bio-client should add it to the communication path.

4. The bio-client orchestrates the user interaction. For instance, it establishes a SIPBIO session to a biometric system. It can also coordinate the delivery of payload to the appropriate capture device. Depending on the payload type, the source of data, as well as its transport method, may vary.

5. Being the front-end of the biometric system, the bio-server receives requests and coordinates appropriate responses or actions with other internal components (e.g. databases, biometric engines, etc.) Internal communication is currently usually done through TCP sockets; however, nothing impedes the development of SIP/SIPBIO interfaces when commercially viable.

Another case of an extended distribution would be a passive biometric system in a call centre environment. If a customer enquiry could not have been resolved at the IVR level, the call would be transferred to a customer representative. A simplification of such a system is shown in Figure 3.9. It must be noted that there is not a universal configuration for a call centre. A viable alternative is presented based on the most commonly used technology, communication protocols and the proposed SIPBIO alternative.

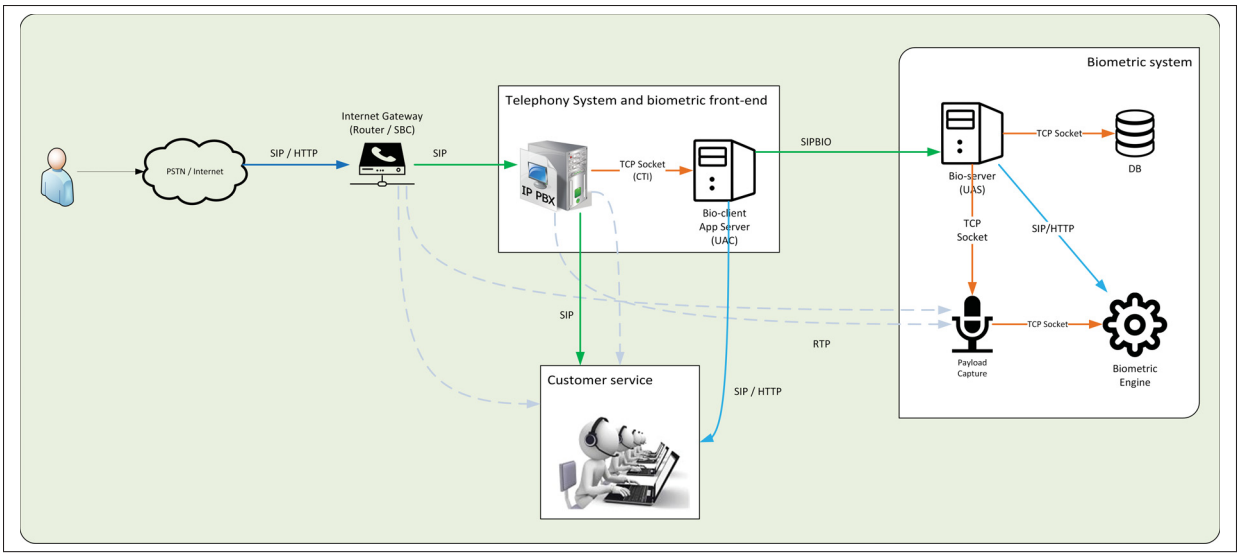


Figure 3.9 *Extended distribution passive biometrics example.*

While many steps are involved in completing a call centre request, the following is a highlight of those interactions of interest for the purposes of this proposal.

1. A user initiates a contact to a service provider. The communication is likely to be initiated through PSTN SIP trunks or through a SIP phone over the Internet. Should the client be RFC 7329 compliant, it should include a universal SIP ID.

2. If the self-service system was not able to fulfil the user's request, the call is forwarded to a queue where an agent with the appropriate skills eventually receives it. Notably two events are relevant: if the agent phone is SIP, a SIP communication establishment would be completed between the telephony system and the agent extension; and, a CTI event is generated from the telephony system to the bio-client application server. A universal SIP ID can be passed (or introduced) in both interactions. Note that most CTI protocols are vendor proprietary. It does not mean CTI could not be passed through an SIP-like protocol, it is just not available in the market.

3. The bio-client orchestrates the agent interaction with the biometric system. For instance, it establishes an SIPBIO session to a biometric solution. It can provide the agent with

visualisation highlights through the enrolment or verification process. If required, it can also coordinate the delivery of payload to the appropriate capture device.

The descriptions of several real-time scenarios have been introduced to display some of the possibilities SIPBIO opens for easing biometric authentication across elements in a communication chain. The next section shows the standard requirements of the protocol to facilitate its design in the upcoming chapter.

3.5 Requirements

This section provides a description of the requirements of a SIP extension to handle biometric authentication. Requirements are developed based on the use case scenarios described in the previous section.

Association of biometric information with multimedia session description. The UA receiving the biometric information should be able to associate it with the corresponding multimedia information (audio containing the voice in the case of voice biometrics). Usually this information is transmitted in the body of SIP messages as SDP content. The biometric information should be inside the SDP characteristics.

Support Active or Passive Biometrics. Both, UAC and UAS should be able to select and handle both types of authentication. If one of the biometric types is not available, the protocol should be able to readily switch to the supported type or cancel the request.

Support frequency of authentication. In any authentication type, it should be possible to define whether it will be executed only once or at different intervals during the duration of the interaction.

Support exchange of metadata of information with third party systems. To search for matches between biometric identification and identification, UAs need to receive a claimed ID from a back-end system. Receiving information, to use it in the matching process of the biometric print, must be supported.

Avoiding additional connections. The authentication process should be completed within one single connection.

Format negotiation. Every environment susceptible of being *biometric enabled* handles media in different media formats. For instance, in telephony environments every VoIP solution uses one or more compression codecs: G.711, G.729, G.723, etc. Support for different media formats should be allowed.

Progress information. The ability to report partial progress of the biometric analysis should be supported.

3.6 Extending SIP

As discussed above, when extending the base SIP protocol, guidelines exposed in RFC4485 need to be followed. A recapitulation of those guidelines with focus on the key design topic suggested by them is useful before proceeding to the proposal presentation.

Criterion 1: SIP Solution Space. This proposal leverages the capabilities of SIP to initiate, modify, and terminate interactive sessions. Sessions are to be established with the purpose of performing a biometric operation. For this purpose, SIP will work in combination with SDP for the definition of the media characteristics of biometric payloads. The path of the media exchange is completely independent of the path followed by SIP session messages. Communication parties (UAs) are to be aware of the biometric nature of the sessions they will be establishing.

Criterion 2: SIP Architecture Model. SIP architecture is a vast topic, those of interest for the purpose of this proposal are outlined next.

1. *SIP independence.* The nature of the media content exchange between the biometric aware parties is independent of the SIP session itself. Any type of biometric payload can be exchanged through the SIP session.

2. *SIP and Session path independence.* SIP messages traverse an independent path of the one traversed by the media content of the correspondent sessions.
3. *Multi-provider and multi-hop.* It is assumed that the parties in the biometric exchange are in different networks separated by n hops where $n > 1$ up to the maximum number specified by the base SIP protocol ($max - forward = 70$ by default).
4. *Transactional.* All proposed SIP message exchanges are defined under the assumption of a request / response model.
5. *Proxies can ignore bodies.* None of the proposed SIP extensions to the body (payload) influence the routing policies.
6. *Proxies do not need to understand the method.* No new methods are defined in this proposal.
7. *INVITE message carries full state.* Any SIP INVITE or subsequent *Re-INVITE* carry all signaling information to proceed with the transaction. There is no need to collect information from several INVITEs to process a single response.
8. *Generality over efficiency.* SIPBIO is designed to handle any payload and treat biometric processes in a general way. For illustration purposes the functionality is explained as a function of voice biometrics.
9. *The Request URI is the primary key for forwarding.* The Request URI indicates a resource that resolves to the desired recipient. The semantics of the Request URI is not modified.
10. *Heterogeneity as a norm.* Any device able to provide the required information for the protocol functioning should be able to make use of the extension. There is no focus on a specific type of device.

Criterion 3: attention to a set of known SIP issues. There are known issues that have been previously identified as being caused by badly defined extensions. The extension proposal must guarantee that these known situations are properly handled.

1. *Backward compatibility.* Since the purpose of SIPBIO is to provide a protocol to handle biometric transactions over SIP, it is necessary that the participant UAs correctly understand all operations. At the very minimum, a UAS should be able to report unavailability of the service. If the biometric processing service is unavailable, the transaction must end, and the UAC must be properly informed of the situation. If a UAS does not understand the extension components the request fails, and the service cannot be provided; however, an appropriate reply must be sent to the UAC to allow it to provide a clean treatment of the request (e.g. instruct the application to handle the authentication in a different way). Due to the nature of SIPBIO security service unavailability must be cleanly handled. To avoid proxy failures, the *Proxy-Required* field is never used.

2. *Security.* SIPBIO does not require any new security specification. All default SIP security mechanisms are deemed sufficient. Owing to the security nature of the information it is strongly suggested that the signaling and the media sessions are established through a secure channel. However, it is not a requirement for SIPBIO to work.

3. *Terminology.* All the terminology used in this proposal follows the guidelines defined in RFC 2119 and BCP 14.

4. *Syntactic.* All formal naming used in this proposal follows the recommendations of RFC 4485.

5. *Semantics.* This proposal follows the recommendations of RF4485 for semantics.

6. *Examples section.* To better understand the proposal, an exampled section with most commonly used cases is included. Examples follow the format and presentation recommended in RFC 3665 and BCP 75.

7. *Other general recommendations.* Other recommendations are also considered: overview section, IANA considerations, document naming conventions and additional considerations for each extended entity.

3.7 Protocol compliance

A communication protocol is an agreement between two or more entities (peers) on how to proceed with a communication. The protocol defines details such as syntax, semantics and error handling (Tanenbaum and Wetherall, 2011b). An implementation does not necessarily need to comply with the protocol. For instance, Adamczyk *et al.* (2008) found that only a few websites implement the standard HTTP protocol completely.

The implication of developing a non compliance implementation of the protocol is that the interoperability of the system uses the protocol is handicapped. If another system requires integrating with an non-compliant entity, it would require custom development which is one of the situations that SIPBIO is trying to avoid.

Sometimes a vendor finds that their use cases require to go beyond the protocol standard or they do not have the resources to fully implement all the protocol requirements. In those cases, the vendor must report the situation and, if possible, provide instructions on how to achieve protocol compliance. An example of this case is the base SIP implementation of Cisco Systems[1]. They provide instructions on how to achieve RFC compliance of their protocol implementation which is not 100% compliant by default (Cisco, 2018).

SIPBIO cannot enforce implementation compliance. SIPBIO proposal aims at being clear, general and complete as to fulfil the requirements of biometric vendors and biometric enabled applications.

[1] https://www.cisco.com

CHAPTER 4

SIPBIO, EXTENDING SIP AND SDP TO SUPPORT BIOMETRIC AUTHENTICATION

The purpose of this chapter is to describe SIPBIO, the extension to the base protocol, and working scenarios. The goal of SIPBIO is to provide a signaling mechanism to support a biometric process. SIPBIO is designed to comply with the requirements defined in the previous chapter. Different biometric scenarios and their processes have been described in Chapter 3. This chapter now focuses on the extended content of SIP messages used in the protocol exchange.

Even though this chapter does not explore in detail all possible use case scenarios, the theoretical layout can be applied to any biometric interaction. It is up to the developer or software architect to define the cases covered by an application.

4.1 SIP option-tags

SIPBIO makes use of the flexibility of the SIP protocol to include information required during the session as part of the protocol header. SIP headers *Required*, *Supported* and *Accepted* indicate those features that are, respectively, mandatory, supported or accepted by the UA. SIPBIO defines a set of option tags to be used as described next.

***bioanalysis*.** This tag indicates that a UA must support the biometric information exchange as proposed by SIPBIO. It is set as mandatory by using it in the *Required* header. To avoid further processing and to allow the UAC to implement the required logic to handle the situation. If the feature is not supported, the UAS returns a *420 Bad Extension* response. If the UAC does not implement this tag, the UAS should respond with a *421 Extension Required* message. The UAC can either retry the request by adding the tag in the *Required* or *Supported* headers, or by sending the request to a different UAS that might accept it with the original format.

multibiopayload. The use of this tag is optional. It is used in the *Supported* header. It is used to indicate that the UAC supports more than one biometric method. This is useful to perform multi-factor authentication. For instance, a first factor could be based on the voice and the second on the iris recognition. By including this tag, the UAC also indicates that it supports requests for several biometric payloads for the same session. Since only one payload type is supported per biometric transaction, several biometric payloads are supported in a serial fashion.

multibiotype. The use of this tag is also optional. It is used in the *Supported* header. This tag is used to indicate that the UAC supports different biometric types. For instance, it may support *active biometrics*, *passive biometrics* or *offline analysis*.

application/sipbio. This tag is used to indicate the acceptable Internet media type in the message body. In this case, the type is *application* and the sub-type is *sipbio*. This tag is used in the *Accept* header. In particular this tag indicates that the UAC supports exchange of information of the specified *sipbio* content.

4.2 SIPBIO application content

The *application/sipbio* tag defines a new content type for the body of a SIPBIO message. It is used to define the characteristics of a biometric session negotiated by the enabled UAC and UAS pairs. Its definitions are based on an XML structure defined as *SIPBIOInformation*.

The elements of the *SIPBIOInformation* XML structure are defined to support real time transactions as well as to serve as a medium of information exchange by biometric peers. For real time transactions, the element has to be referenced in the media description of the SIP exchange that is usually handled by making use of the SDP protocol. Otherwise the XML structure can be directly transmitted as the body of the exchanged SIP messages. The top-level members of *SIPBIOInformation* are described next. All elements of the XML entity are optional depending on the information exchange. It is up to the implementation developer to choose the right ones to cover the intended purpose.

- *biotype.* It defines the type of biometric interaction in terms of it being *active*, *passive* or *off-line*. No constraint on the categories is defined. It is suggested, however, to implement based on those industry recognized categories.

- *vendor.* It can be used to describe the specific technology developer. At the time this proposal was written, no standard has been adopted for biometric solutions. This XML structure can be used by developers implementing a particular solution to process based on logic required for each vendor offering.

- *biopayload.* It is used to define the type of payload for the biometric operation. For instance, it can be a *voice*, a *finger-reading*, an *iris* sample, etc.

- *claimedIdentity.* This XML high level structure serves as a placeholder for details about the claimed identity. It can be expanded based on the implementation requirements. It can include lower level XML declarations referring to customer ID, names, national identification numbers, etc.

- *groupInfo.* It represents an array of groups the claimed identity should belong to. It can be used to speed up a biometric identification process.

- *payloadformat.* It is used to define the format of the biometric payload. Lower levels can define technical characteristics such as resolution, file type, sampling, codecs, etc.

- *payloadSource.* It defines the payload source. Possible payload sources include, but are not limited to: an URI pointing at the location of a file, network parameters (IP/port, socket) of a payload source (for instance a media gateway or an IP phone for voice biometrics).

- *biometricResults.* This high-level structure is used to provide details of the result of a biometric transaction.

- *metadata.* It is used for adding metadata to the transaction. In production environments, this metadata is sometimes called *business data.*

- *validity.* Biometric authentications can have a validity period to avoid impostors taking over an operation after the initial verifications. If not specifically set, the validity of the operation is assumed to be valid for the entire length of the transaction.
- *extraInformation.* It can be used to include information not covered by any other XML label.

Note that further ahead in the description of the message exchange the *SIPBIOInformation* XML structure is used extensively under different names just to represent the intention of the exchange. To avoid confusion, a description follows of the naming convention used in SIPBIO to designate a particular version of the structure.

- *sipbioInfo.* It represents an offer with a set of options related or required for a biometric exchange to be completed. Typically, when it comes from the UA starting the conversation (the UAC), it represents the set of options supported and offered. When the receiving UA (the UAS) sends a *sipbioinfo*, it represents the set options that are being accepted to finalize the channel establishment and start the media exchange.
- *sipbioInfoRequest.* It is the representation of a set of options requested and supported by a biometric system to complete at least one biometric operation. The messenger of these options is an UAC.
- *sipbioResults.* It is the set of results given by a biometric operation. It is sent by the UAS which, in turn, needs to get them from the biometric engine.
- *siobioStatus.* It is a partial result of a biometric operation. Once the status reaches 100%, it is replaced by *sipbioResults*.

4.3 Extending SDP

SDP is the default protocol of choice for SIP transactions to specify the characteristics of the multimedia content to be used during a session. Default SDP content includes media informa-

tion independent of the transaction. Due to the security purpose of biometric transactions, it is important to establish a direct association between the biometric information and the multimedia content to guarantee the accuracy of the results. Biometric information can be included as part of the multimedia content. By doing this association, a direct relationship between the payload provided during a biometric transaction can be associated with the signaling in the SIP exchange. Note that isolation between the two elements may be desirable precisely in order to obfuscate their relation.

When a single biometric transaction is executed as part of, or previous to, an intended information exchange, the SDP to SIP association should be completed at the stream level. Stream processing should be concurrent for multiple authentication methods (e.g. voice and finger print reader being performed at the same time.) By allowing stream level flexibility, two streams on the same biometric transactions can transport media in different formats. The SIPBIO proposal defines an SDP attribute called *biometric-info*. In general, the attribute looks like:

$$a = biometric-info : codec : schema : SIPBIOInformation \tag{4.1}$$

or

$$a = biometric-info : codec : uri : SIPBIOInformation \tag{4.2}$$

As per RFC4566, an attribute *a* is the recommended way to extend SDP. SIPBIO makes use of value attributes to be able to convey all required information in one single attribute (Handley *et al.*, 2006). The following is the breakdown and purpose of each element in the attribute definition.

- *a.* It indicates an attribute in the SDP definition in the form of *a*=<*attibute*> or *a*=<*attribute*>*:*<*value*>.
- *biometric-info.* It is the name of the attribute. SIPBIO introduces an attribute to represent media features are required for a biometric process.

- *codec.* It references the codec to be used for the biometric analysis. The codec should be taken from one of the *m* attributes defined. For instance, in a media exchange including audio, video and text, the audio channel can be selected for the biometric operation.

- *schema / uri.* This tag references whether the biometric information is sent along with the message itself (in which case it is tagged as *schema*) or it is contained in an external document reachable through a URL (tagged as *uri*).

- *SIPBIOInformation* It is an XML element with information required for the biometric analysis as described in Section 4.2.

4.4 SIPBIO process

SIPBIO expands the content of a default SIP request to establish the dialogues already described. This section shows the expanded content of each type of SIP request. For simplicity, only the elements required to understand each message are shown.

4.4.1 Pre-session establishment

A proper session can be preceded by an optional *OPTIONS* SIP message. It can be used before establishing the biometric dialogue. A SIP OPTIONS message is used by the UAC to request from the UAS a list of supported capabilities. A SIPBIO UAC is expected to establish a dialogue with the purpose of a biometric exchange. Thereafter, the UAC needs to determine the biometric processing characteristics of the UAS. This message is optional. In a non-dynamic environment, it is possible to define SIP dialogues starting directly with an INVITE as is explained later. The use of OPTIONS is suggested to properly construct the INVITE based on the available UAS features.

Minimally, the OPTIONS message *Accept* header must have the option tag *application/sipbio* as shown in listing 4.1. An *Accept* header indicates which media types are accepted in the dialogue.

Listing 4.1: SIP OPTIONS, no body content

```
Request-Line: OPTIONS sip:3002@etsmtl.int SIP/2.0
  <Message header>
  [...]
  Accept: application/sipbio
  [...]
  Content-Length: 0
```

Optionally, a UAC can send all the biometric options supported in the body. A *sipbioOptions* structure of content type *application/sipbio* can be used. In that case the OPTIONS message would look similar to what is shown in listing 4.2. An XML sample for this case is shown in listing 4.3.

Listing 4.2: SIP OPTIONS, with body content

```
 OPTIONS sip:3002@etsmtl.int SIP/2.0
  <Message header>
  [...]
  Content-Type: application/sipbio
  [...]
  <Message body>
  <?xml version="1.0" encoding="UTF-8" ?>
    <sipbio xmlns="urn:ietf:params:xml:ns:sipbio">
    [...]
```

Listing 4.3: XML sample for SIP OPTIONS

```
<?xml version="1.0" encoding="UTF-8" ?>
  <sipbio xmlns="urn:ietf:params:xml:ns:sipbio">
    <biometricTypes>
      <entry biometricType="active" />
      <entry biometricType="passive" />
```

```
      <entry biometricType="offline" />
    </biometricTypes>
    <payloadTypes>
      <entry payloadType="voice" />
      <entry payloadType="face" />
      <entry payloadType="fingerprint" />
    </payloadTypes>
    <payloadSources>
      <entry payloadSource="uri" />
      <entry payloadSource="stream" />
    </payloadSources>
  </sipbio>
</xml>
```

A SIP OPTIONS message triggers a response from the UAS. Providing there is no error in the request and the UAS is able to understand it, a *200 OK* SIP response is sent back to the UAC. Otherwise the UAS should respond as helpfully as possible. For instance, a *406* error message can be sent when the characteristics required cannot be met by the UAS. An acceptable *200 OK* response is shown in listing 4.4. Its corresponding XML body content is shown in listing 4.5. As expected, the XML body is a subset of the characteristics requested in the OPTIONS requests. A UAS must only reply with the options it supports. For this sample, the UAS (and the biometric system behind) support *active* and *passive* voice biometrics analysis. Also, the UAS is able to process or coordinate payloads referenced through an URI or included in the media definition protocol.

Listing 4.4: 200 OK for SIP OPTIONS

```
Session Initiation Protocol (200)
Status-Line: SIP/2.0 200 Ok
   <Message header>
      [...]
```

```
      CSeq: 101774172 OPTIONS
      User-Agent: BioProcessingServer/10.0.0.2 (Build/2.0.3.)
      Allow: INVITE, ACK, CANCEL, OPTIONS, BYE
      [...]
      Content-Type: application/sipbio+xml
      [...]
   <Message body>
  <?xml version="1.0" encoding="UTF-8" ?>
    <sipbio xmlns="urn:ietf:params:xml:ns:sipbio">
    [...]
```

Listing 4.5: XML body 200 OK for SIP OPTION

```
<?xml version="1.0" encoding="UTF-8" ?>
  <sipbio xmlns="urn:ietf:params:xml:ns:sipbio">
    <biometricTypes>
      <entry biometricType="active" />
      <entry biometricType="passive" />
    </biometricTypes>
    <payloadTypes>
      <entry payloadType="voice" />
    </payloadTypes>
    <payloadSources>
      <entry payloadSource="uri" />
      <entry payloadSource="stream" />
    </payloadSources>
  </sipbio>
</xml>
```

4.4.2 Session initiation

The first SIP options message is useful for the UAS to build a properly formatted SIP INVITE. An UAC should have the logic to construct an INVITE based upon the *guidance* received in the *200 OK* SIP response to the OPTIONS request. The exact implementation of the protocol may vary depending on the application. However, in general, a SIBIO INVITE should include the *bioanalysis* tag as a *Required* header (view tag description in Section 4.1). Tags *multipayload* and *multibiotype* can also be included as *Supported*. All details of the biometric transaction are to be included as part of the *SIPBIOInformation* XML structure. If an OPTIONS message interchange was previously used, the INVITE should only include what was reported as accepted by the *200 OK* response to the OPTIONS request. A SIPBIO implementation must support MIME format as per RFC2045. Meeting this requirement allows SIPBIO messages to include more than one body part. For instance, a SIPBIO INVITE includes an SDP element containing an XML payload for the *SIPBIOInformation* structure. Listing 4.6 displays the main components of the header and the multi-part body.

Listing 4.6: SIP INVITE

```
Request-Line: INVITE sip:3002@etsmtl.int SIP/2.0
<Message header>
  [...]
      Require: bioanalysis
      Supported: multipayload, multibiotype
      Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, NOTIFY,
         MESSAGE, SUBSCRIBE, INFO, UPDATE
      Content-Type: multipart/alternative;boundary=boundary
   [...]
  <Message body>
  [...]
  --boundary
  Content-Type: application/sdp
  Session Description Protocol
```

```
 Session Description Protocol Version (v): 0
[...]
Media Description, name and address (m): audio 6000 RTP/AVP 0
Media Attribute (a): rtpmap:0 PCMU/8000
Media Attribute (a): biometric-info:0:schema:SIPBIOInformation
[...]
--boundary
[...]
--boundary
Content-Type: application/sipbio+xml
<?xml version="1.0" encoding="UTF-8" ?>
  <sipbio xmlns="urn:ietf:params:xml:ns:sipbio">
[...]
```

At this point of the exchange, two different circumstances may happen: the UAS agrees with the media session parameters received in the INVITE or the UAS responds with an alternative offer to establish the session. The latter situation would be avoided if an OPTION / 200 OK exchange were properly completed before the initial INVITE.

The simplest case is the one in which there is an immediate agreement on the *sipbio* content type as represented by the *SIPBIOInformation* structure in either of the alternative body contents of the INVITE.

The *200 OK* message is likely to look as the response shown in listing 4.7. Note that even though the SIP INVITE supports MIME, that is, multiple bodies may be defined (as shown in listing 4.6), a 200 OK response must choose one single body type. To keep the information exchange going, the body part contains an *biometricInfo* attribute that is formed as defined by the *SIPBIOInformation* structure.

Listing 4.7: 200 OK for SIP INVITE

```
Session Initiation Protocol (200)
   Status-Line: SIP/2.0 200 OK
   <Message header>
     [...]
      CSeq: 101774189 INVITE
      Contact: <sip:test@10.0.2.15:5060;transport=udp>
      Accept: application/sdp
      Allow: INVITE, ACK, BYE, CANCEL, OPTIONS, MESSAGE, INFO,
         UPDATE, REGISTER, REFER, NOTIFY, PUBLISH, SUBSCRIBE
      Supported: multipayload, multibiotype
      Content-Type: application/sdp
     [...]
   <Message body>
      Session Description Protocol
         Session Description Protocol Version (v): 0
       [...]
       Media Description, name and address (m): audio 27942 RTP/AVP 0
       Media Attribute (a): rtpmap:0 PCMU/8000
       Media Attribute (a): biometric-info:0:schema:SIPBIOInformation
       [...]
```

Should the INVITE not be compliant with the expected format, an UAS should respond with a *405 Method not allowed* message.

Another possibility is when the INVITE contains several mutually exclusive biometric options that are all accepted by the UAS. This case is handled by the UAS sending a *183 (session in progress)* message. This message carries a body content with the *biometricInfo* attribute. This message can be considered a biometric information offer. The UAS responds with the subset of supported biometric parameters to which the UAC must subsequently respond according to

the offered format. A *183 session in progress* attribute indicates that information about the session state is included in the body of the message. As stated, this quality is used to establish a dialogue with the UAC with the aim of agreeing on the parameters that allow a biometric exchange of information. Since *183* is a provisional response, the protocol indicates that a *200 OK PRACK (provisional acknowledge)* is due. This *200 OK PRACK* message allows the UAC to report the reception of the previous communication and, at the same time, make an offer on the set of properties required by the UAS to allow the initiation of the biometric exchange. As shown previously, this offer is contained in the *biometricInfo* attribute in the body of the message. This exchange could potentially follow several interactions of *183* vs *200 OK PRACK*, until the UA runs out of options or as long as the implementation allows. In a system able to receive multiple biometric payloads this exchange can help to narrow down the available payloads. For instance, a Website or mobile application could allow biometric authentication through the on-board camera (face, retina), a fingerprint, voice, typing behaviour, etc.

4.4.3 Three way handshake completion

Only at this point is the UAC is able to fully acknowledge the initial *INVITE*. As the previous subsection explained, the UAS is to send a *200 OK* message with the accepted parameters to enable the media session exchange of the biometric payload required for a biometric interaction. At that point the UAC sends an *ACK* message that marks the final response to the *INVITE*. To keep consistency, the *CSeq* number is not increased for the *ACK* but the *CSeq* method is changed to *ACK*. Since the initial *INVITE* in the process contained a message body, the *ACK* may not contain a message body. However, if the implementation requires a message to be carried in the body, it may do so by taking advantage of the fact that an *ACK* to a *200 OK* is end-to-end instead of hop by hop.

4.4.4 Media session and termination

Once the communication is agreed upon, media can be exchanged either between UAC and UAS or between the corresponding nodes as established by the signaling (as previously de-

scribed). The media session is the transmission of the biometric payload from the biometric reader-enabled device (e.g. a phone for voice biometrics) to the biometric payload reader. The reader is in charge of collecting the payload, applying any required transformation to be finally sent to the biometric engine for analysis. A biometric payload reading process, along with the biometric analysis, is independent of the signaling process facilitating the operation. An implementation of the protocol must take into account all realms: signal establishment, media exchange and biometric analysis.

In a simple interaction, once the media exchanged is completed the implementation should handle the logic to complete the biometric operations and tear down the SIPBIO session. A *BYE* is sent by the UAS to the UAC to indicate the finalization of the SIP session. Note that under this basic scenario SIPBIO transports the results in the *BYE* message that triggers the termination of the dialogue. Even though this is a rather unusual approach, it is not forbidden by the SIP based protocol. The implementation must ensure not only that it is the UAS that finalises the dialogue but also that the UAC is able to read and process *sipbioResults*. As mentioned, this approach is not conventional but it is supported as a simple alternative for implementations that require light and simple code.

Figure 4.1 illustrates the process described in the latest subsections. It expresses the simplest information exchange in a SIPBIO operation. Out of simplicity, it is assumed that the UAC is also the biometric payload reader while the UAS is the biometric payload receiver. Figure 4.2 shows a diagram with an alternative approach where these two entities are completely separated from those that establish the media exchange channel.

4.4.5 Multiple alternative payloads

Subsection 4.4.2 illustrates a UAC generated INVITE including concurrent biometric alternatives. A UAS can negotiate a preferred biometric exchange type or, alternatively, it can reply with clear information for the UAC to handle the payload for each biometric type. To refresh the understanding of the former case, UAS replies to an *INVITE* with a *183 Session in progress*

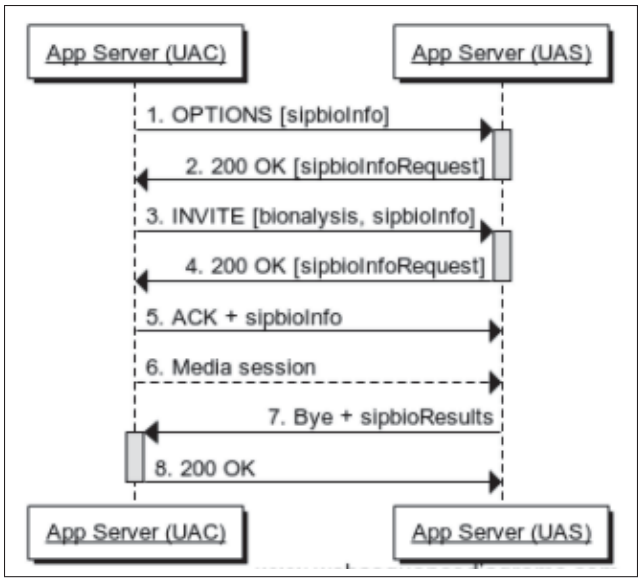


Figure 4.1 *SIPBIO simple session.*

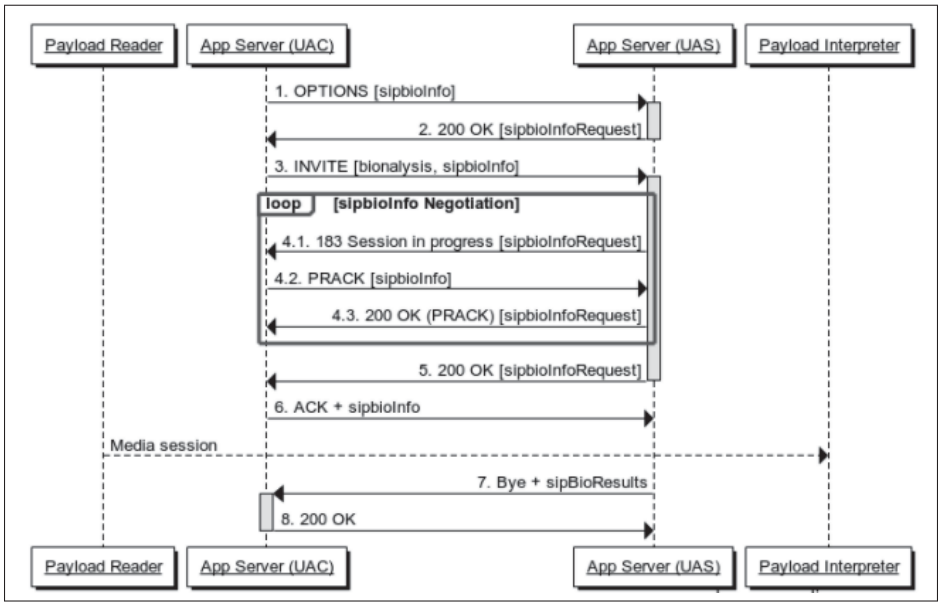


Figure 4.2 *SIPBIO simple session (off band media).*

with the suggested features in its body either as *content/sdp* or *content/sipbio*. In both cases, supported biometric fields are transmitted in the form of a *biometricInfo* XML structure. The implementation must be as specific as possible unless it is imperative that the UAC choose the desired format. For instance, if the UAS supports handling of session establishment for voice

and face biometrics but consent from the user is required for either case, the UAC must implement the logic to let the user decide, instead of the UAS imposing a method. To a *183* SIP message, a UAC must reply with a *200 OK PRACK*, which it can use to provide its biometric offer.

In theory, this exchange could go on for a number of loops equal to the number of biometric handling options. In practice, an implementation is likely to favour an option based on a particular scenario in order to keep SIP message exchange at a minimum. A visual representation of this process is shown in Figure 4.3. To ease visualisation, all sequences assume the UAC to be a payload reader for any biometric type and the UAS to be a payload receiver for any type of media. Note that the final loop (sipbioinfo negotiation) does not have a *200 OK PRACK*, instead the UAS simply responds with a *200 OK* containing the *sipbio* info parameters agreed upon in the loop.

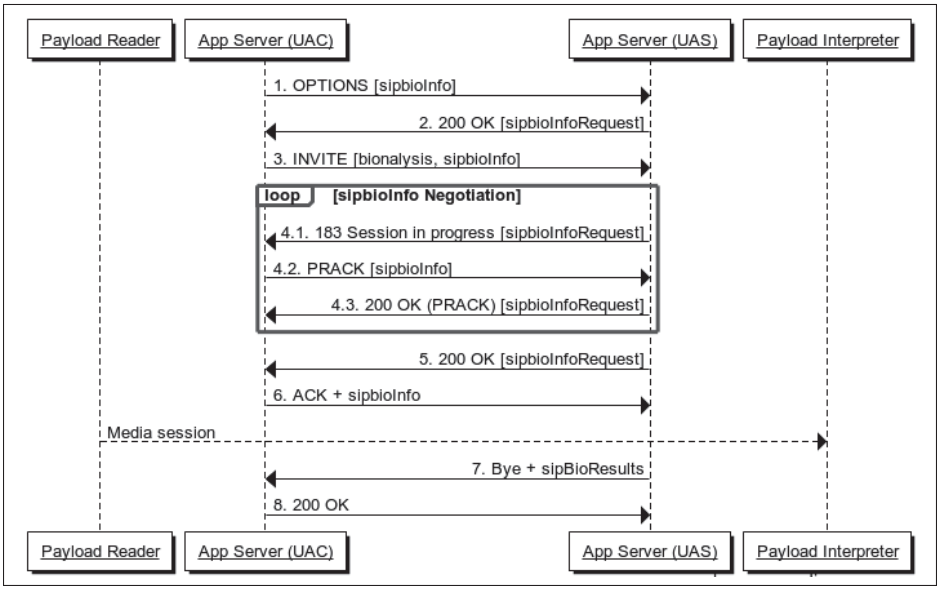


Figure 4.3 *SIPBIO multiple choices / one stream.*

The treatment of the SIPBIO invite with multipart body assumes a *multipart / alternative* directive is being used: each alternative body carries the same information but in different formats. Since it may be the case that a particular environment is able to handle multiple biometric pay-

loads, a SIPBIO UAC implementation must be able to set media paths for multiple payloads. Should a UAS need to provide information to handle this case, it can be done by constructing a *sibioInfo* XML response, based on the *SIPBIOInformation* XML structure, or building a SDP response body specifying several media channels, one for each payload type.

4.4.6 Multiple concurrent payloads

In Subsection 4.4.5, an UAC offers several alternative descriptions of the same payload type for the UAS to choose from. A negotiation follows the offer to agree on a specific set of options to establish a media session. However, it is very possible that the objective is to transmit multiple payloads at once, like when two or more biometric operations need to be performed in parallel: voice, face and fingerprint through a mobile phone. RFC 3388 and 8108 outline recommended approaches for different scenarios ((Camarillo *et al.*, 2002), (Lennox *et al.*, 2017)). Implementers should reference the specific RFC for design guidelines. In the context of SIPBIO, the objective would be to achieve the situation shown in Figure 4.4 where a single SIP negotiation leads to *n* different payloads being sent in parallel from one or more payload readers to one or more payload interpreters or receivers. This behaviour is ideal but potentially complex to implement under the constraints of current protocol standards so implementers may choose to either transmit payloads sequentially by modifying SIPBIO sessions characteristics through *RE-INVITES* or have separate concurrent SIPBIO sessions, one per payload. Either approach reduces complexity but introduces delays and a number of required signalling messages.

4.4.7 Status updates

Most biometric operations take a short time to complete, usually between one to three seconds. However, some of them can take several minutes to complete, as with voice biometrics enrolments. An UAC requesting a biometric result is likely to want feedback on the status of a request to know if it failed or if it is still in process. SIPBIO makes use of the SUBSCRIBE and SIP NOTIFY methods to allow a UAC to receive updates from an UAS. Each subscription establishes a new dialogue between the parties.

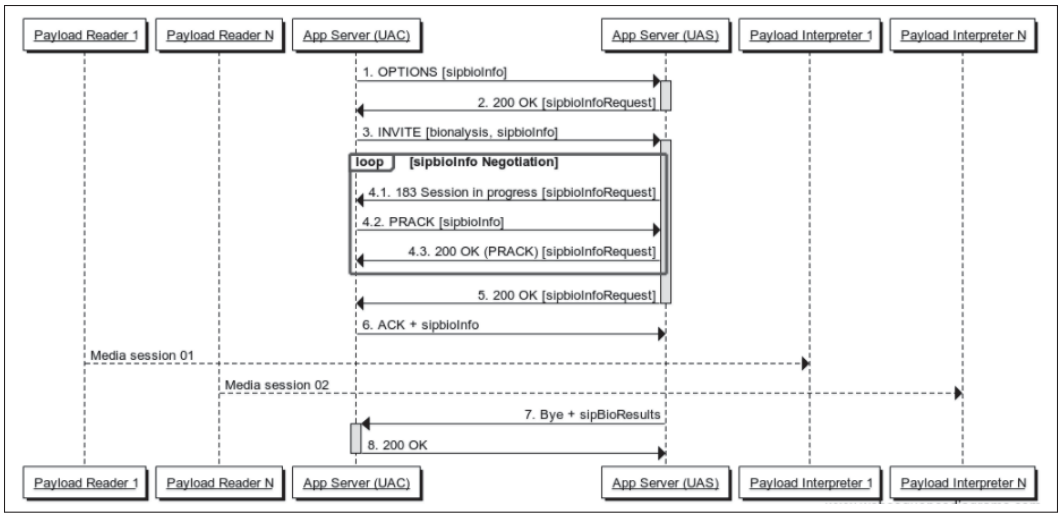


Figure 4.4 *SIPBIO multiple choices / multiple streams.*

So far, biometric results have been sent as the body of the SIPBIO BYE used to terminate the biometric session dialogue. Although this is a technically viable approach, it is not a common either which may lead to SIPBIO not being adopted. A more standard approach would be to add a subscription to biometric events to receive the results of the analysis. A viable call flow with updates on the result of the analysis is shown in Figure 4.5. In this case, the implementer decides to subscribe for a long time (the definition of a *long time* depends on the application), once media starts to flow, the UAC is able to send NOTIFY events with updates on the current states of the biometric analysis. When the analysis finishes, the UAC sends a re-SUBSCRIBE on the same session with an *expires* parameter of *zero* to effectively terminate the subscription dialogue. Implementing updates with the SUBSCRIBE / NOTIFY methods provides the sort of flexibility required to accommodate different types of biometric analysis. The implementer can choose to subscribe for a short interval and only renew the subscription if required (e.g. the biometric analysis has not finalised). Also, the subscription could be terminated by the UAS if required.

Notably, the last NOTIFY would carry the result, allowing the SIPBIO BYE to act in the standard way of simply being a termination signal to destroy the communication path and the SIP dialogue.

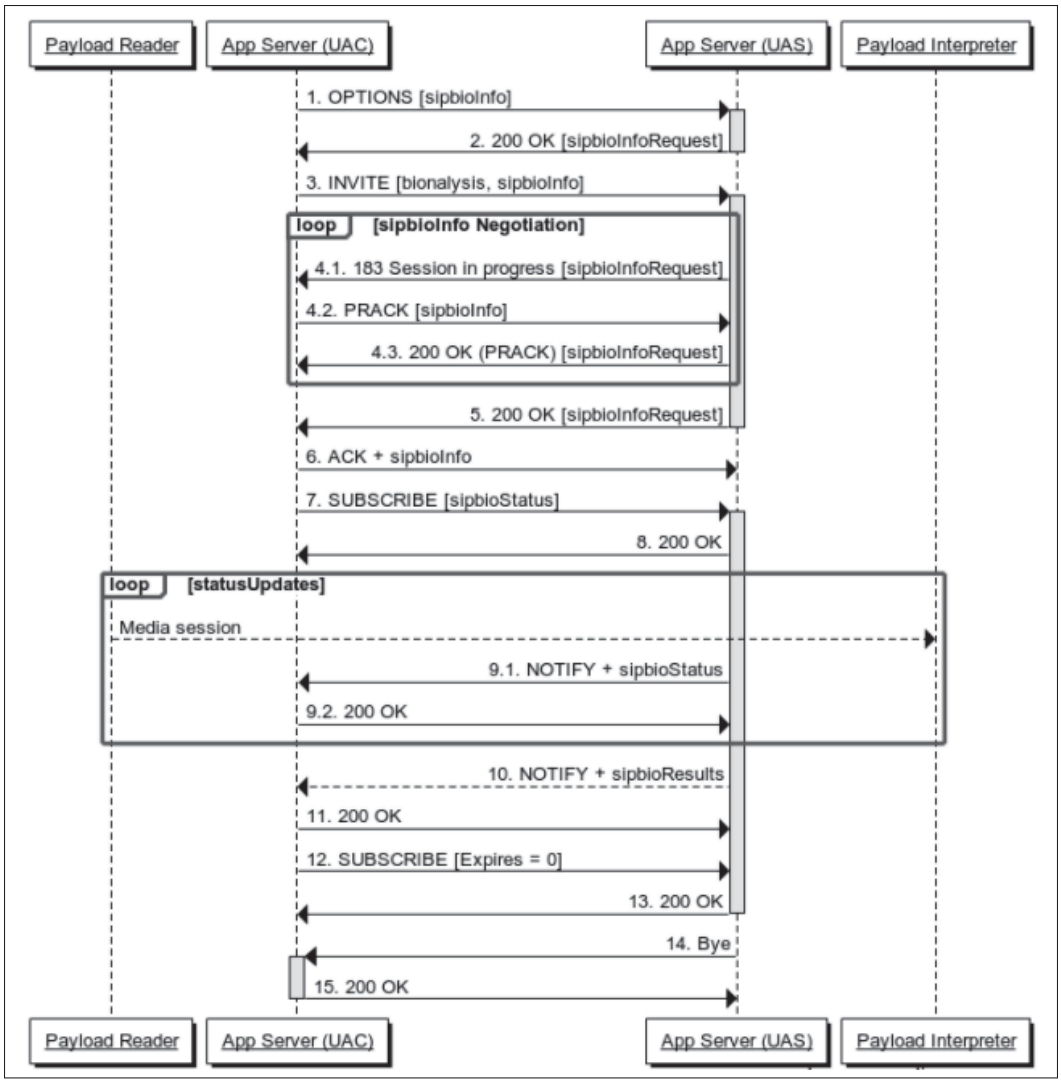


Figure 4.5 *SIPBIO with updates*.

4.5 Mandatory *methods* header fields

Methods serve to request a specific action to be taken by a UA. Previous sections introduced the functioning of SIPBIO and some common scenarios. This section is intended as a reference guide for developers and solution architects planning to use SIPBIO. Due to the nature of the SIPBIO protocol and its inherited SIP flexibility, every method and response message in SIP-BIO are customisable to meet implementation requirements. Designers and developers must take into account that there are mandatory fields in order to keep the consistency of the protocol

and to create a logical dialogue. Those fields are listed next for all relevant methods. Those unmentioned methods are either not used by SIPBIO or do not have any possible modification compared to SIP. Messages and their purpose are not explained. For an introduction please refer to Chapter 2 or the referenced literature. Some fields are shown with values to provide greater clarity on their role in SIPBIO.

INVITE. As per its role in starting the communication channel, the INVITE must be able to inform or establish the following: who to contact, who is trying to establish the connection, how to reach the destination, initiate the SIP sequencing and request the initiation of a biometric analysis operation. Mandatory header fields are listed next.

- Via
- To
- From
- Call-ID
- CSeq
- Contact
- Max-Forwards
- Require: bioanalysis; UAS needs to communicate the requirement to the respective UAS.

BYE. In the context of simplified sequences, a BYE message has not only the task of closing down the communication channel, but it is also in charge of sending the results of a biometric operation to the UAC. For SIPBIO communication establishment, mandatory header fields are shown next.

- Via
- To
- From

- Call-ID
- CSeq
- Max-Forwards
- Content-Length > 0; For SIPBIO short form transactions.

ACK. As a method used to acknowledge a final response to an *INVITE* request, the CSeq field is not increased. Even though, in general, when the initial *INVITE* carries media information, the corresponding *ACK* usually does not. SIPBIO leverages this capability to communicate parameters decided upon.

- Via
- To
- From
- Call-ID
- CSeq
- Max-Forwards
- Content-Type; UAS needs to know what type of content is expected to process SIP-BIO properties accordingly.

OPTIONS. As a method used to acknowledge a final response to an *INVITE* request, the CSeq field is not increased. Even though, in general, when the initial *INVITE* carries media information the corresponding *ACK* usually does not carry any, SIPBIO leverages this capability to communicate parameters decided upon.

- Via
- To
- From
- Call-ID
- CSeq

- Max-Forwards
- Content-Type; UAS needs to know what type of content is expected to process SIP-BIO properties accordingly.

SUBSCRIBE. This method is used in the context of operations explained in Subsection 4.4.7.

- Via
- To
- From
- Call-ID
- CSeq
- Max-Forwards
- Contact
- Event
- Allow-Events: dialog; alternative SIP Event Packages for biometrics could be developed but that topic is not considered here. For the time being the *dialog* event package, as of *RFC 4235* is considered to be sufficient as long as content type *application/dialog-info+xml* is supported. The use of a SIPBIO *SIPBIOInformation* XML structure is consequently compatible.

NOTIFY. This method is used in the context of operations explained in Subsection 4.4.7. It must only be used when a subscription status has been agreed upon.

- Via
- To
- From
- Call-ID
- CSeq
- Max-Forwards

- Contact
- Event
- Allow-Events: dialog; (see note on SUBSCRIBE message.)
- Subscription-State

PRACK. This method is used in the context of operations explained in Subsection 4.4.7. It must only be used when a subscription status has been agreed upon.

- Via
- To
- From
- Call-ID
- CSeq
- Max-Forwards
- RAck
- Content-Type; UAS needs to know what type of content is expected and / or supported to process SIPBIO responses accordingly.

4.6 SIPBIO limitations

Being a signaling text based encoded protocol, SIPBIO has some known limitations:

- SIPBIO does not have any security mechanism beyond the regular digest authentication that SIP provides. For a secured implementation, SIPBIO needs to be deployed over a secured transport protocol.
- When no secured, SIPBIO can be subject to eavesdropping and man in the middle attacks.

- As of today, there is not an available SIPBIO SDK. Developers need to start from a known SIP SDK and create their own models for SIPBIO to work properly. It risks implementations to follow non-compliant patterns for biometric scenarios.

As SIPBIO usage becomes more popular, other issues are likely to arise. A proper version management system is required to allow protocol improvements to be track and made available to de the developers community.

4.7 Closing summary

This concludes the presentation of SIPBIO, a SIP-based protocol to establish, maintain and terminate communication channels that allow parties to exchange biometric information. New option tags were introduced to serve as flags to guide session establishment. An XML structure, to transport information required for biometric sessions, was presented. An alternative mechanism to transmit biometric payload information on an SDP body was explained. The operation process of session handling was introduced for most common scenarios and, finally, a guide for required method headers was lined up.

CHAPTER 5

SIMULATION AND TESTS

SIPBIO is a flexible protocol conceived to be implemented in different situations and environments to supply solutions to any biometric scenario. This chapter presents the simulation of a single scenario to prove the feasibility of the protocol to cover the use cases and demonstrate to clarity of a potential implementation.

5.1 Preparation

This section illustrates the criteria used to define the strategy to probe the viability of SIPBIO. It explains the reasons to selecting a simulation versus an implementation of the protocol and introduces the technology used to complete a relevant representation of a use case.

5.1.1 Selection criteria

A biometric scenario to test has been selected based on the following criteria:

- Representative of the main messages (events) required to complete a biometric operation.
- Simple enough to clearly represent the concepts without needing too many messages to establish a communication.

Then, taking into account that:

- Active biometrics interactions are simpler than passive ones since they do not have to deal with the complexity of capturing a payload while it is being produced (active interactions expect a biometric payload to be provided after it has been collected).
- A verification is simpler than an enrolment since given a known claimed ID, the biometric payload has to be analysed against an expected biometric print rather than performing a

registration for a new user. However, a verification requires all basic biometric operations to be completed which makes it suitable for the required testing process.

The scenario that has been chosen is the *One-time Active Voice Biometrics authentication* (OTAVB) described in Subsection 3.2.1. The sequence shown in Figure 3.7 is used as a reference for the simulation.

It is important to note that non valid scenarios were also tested. Non valid scenarios in the concept of SIPBIO refers to those in which the protocol is malformed or the message sequencing does not follow a valid SIP flow (for instance a response to a request never sent.) Non valid biometric scenarios do not necessarily mean non valid SIPBIO scenarios. Biometrics payloads and the result of their analysis are independent of the communication between the SIPBIO endpoints.

5.1.2 Environment

The testing environment has been built to allow two SIPBIO enabled UAs to establish a session between them and to complete a biometric operation on a local network.

Instead of developing a partial or full implementation of the protocol, it was decided to simulate a chosen scenario. A simulation provides the advantage of being highly flexible and easy to set up. By simulating scenarios, the syntaxes and request / response models can be proven before investing time and resources in an implementation.

A simulation is also useful to detect eventual problems in the protocol design, provide alternatives and, in general, foresee the functioning of a solution that follows the proposal. A simulation is not run with load test objectives. Load testing would be performed by a future implementation of the protocol.

While a simulation makes a simplification of real scenarios, it provides controlled experimentation without much overhead. A further step in the testing could be an emulation which mimics a more realistic behaviour by combining the simulation with a realistic testbed (Fall, 1999).

For instance, to emulate SIPBIO, SIPBIO endpoints would be required to actually communicate with biometric engines and provide real results. In the chosen simulations, biometric results are assumed.

To accomplish the simulation and its posterior visualization and analysis, the following tools are used:

SIPp, SIP simulation engine. SIPp[1], is an open source test tool / traffic generator for the SIP protocol (Gayraud and Jacques, 2014). SIPp is commonly used in academic works to test functionality and performance. Some recent examples of SIPp utilization to simulate SIP proposals are the works by Stanek and Kencl (2011); Dassouki *et al.* (2014); Ryu *et al.* (2012).

SIPp allows the testing of the SIPBIO messaging exchange without requiring the programming of a whole SIP endpoint from scratch, with the added complexity that it implies.

Traffic capturing. Network traffic is captured on the servers with TCPDump[2] 4.9.0. TCPDump constitutes a simple yet powerful tool to *listen* to newtwork traffic at any communication point. A good introduction to the use of TCPDump has been written by Datt (2016). TCPDump is a common tool in computer networks research: for instance, Middleton and Modafferi (2016) use it to determine traffic levels on which to apply a QoS classification.

Taffic analysis and visualization. Wireshark[3] 2.4.2. is used to read, verify, and analyse the captured traffic. Wireshark is a popular tool among network engineers and computer scientists. Ample literature is available for the use of Wireshark, a recent good reference being by Chappel and Combs (2017). Wireshark use is common in academic research.

[1] sipp.sourceforge.net

[2] www.tcpdump.org

[3] www.wireshark.org

Examples directly related to SIP traffic analysis can be found in Devlic (2010) and Aimilia *et al.* (2016).

5.1.3 Test environment layout

Figure 5.1 illustrates the simplicity of the test environment. To keep things simple all tests are run on two virtual machines connected to the same network segment. Table 5.1 shows relevant information for the test servers.

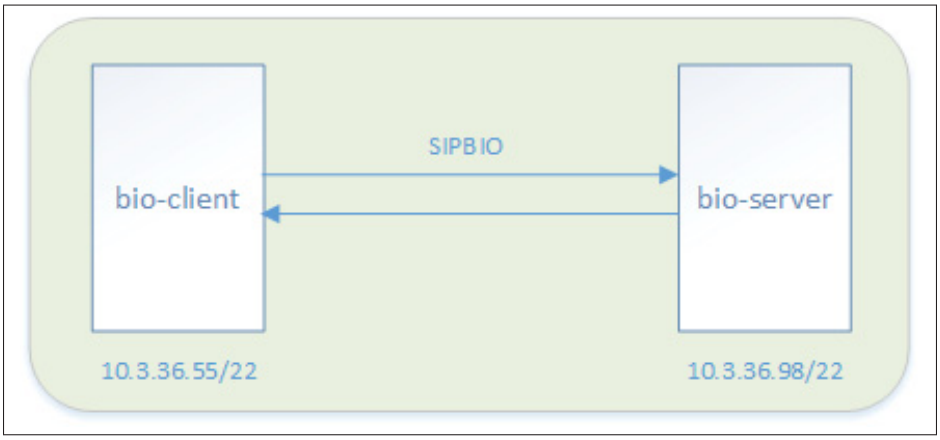


Figure 5.1 *Test layout*.

Table 5.1 Unique server properties.

Property / Server	**Server 01**	**Server 02**
General role	UAC	UAS
Biometric role	bio-client	bio-server
IP address	10.3.36.55/22	10.3.36.98/22

5.1.4 SIPp theory and usage

SIPp is an open source traffic generator and simulation tool written in C++. It is designed to test SIP scenarios, flows and variations. SIPp is used with different purposes:

- Verify *grammatical correctness* of SIP messages.
- Load test. Two or more endpoints can simulate hundreds or thousands of simultaneous calls.

Testing of the SIPBIO messaging on SIPp is a key preamble in the implementation of the protocol. An implementation of the protocol is beyond the scope of this thesis. However, by simulating SIPBIO with SIPp its messaging consistency and workability are proven.

To use SIPp, XML configuration files, called scenarios, must be created to hold the SIP messaging in segments known as *macros*. These macros are placed between *actions* that instruct the endpoint on how to handle the messaging. For instance, listing 5.1 shows the initial segment of the scenario of the *client* SIPBIO endpoint while listing 5.2 shows the counterpart for the *server* SIPBIO endpoint.

Listing 5.1: Fragment of SIPBIO client scenario

```
<scenario name="Basic Bio-Client">
  <send>
   <![CDATA[
    OPTIONS sip:[service]@[remote_ip]:[remote_port] SIP/2.0
     Via: SIP/2.0/[transport] [local_ip]:[local_port];branch=[branch]
     From: bio-client <sip:bio-client@[local_ip]:[local_port]>;
      tag=[pid]SIPpTag00[call_number]
     To: [service] <sip:[service]@[remote_ip]:[remote_port]>
     Call-ID: [call_id]
     CSeq: 1 OPTIONS
     Contact: sip:bio-client@[local_ip]:[local_port]
     Max-Forwards: 70
     Accept: application/sipbio
     Content-Length: [len]
    ]]>
  </send>
  <recv response="200"></recv>
  <send>
   <![CDATA[
    INVITE sip:[service]@[remote_ip]:[remote_port] SIP/2.0
    Via: SIP/2.0/[transport] [local_ip]:[local_port];branch=[branch]
    From: bio-client <sip:bio-client@[local_ip]:[local_port]>;
      tag=[pid]SIPBIOtag00[call_number]
    To: [service] <sip:[service]@[remote_ip]:[remote_port]>
    Call-ID: [call_id]
    CSeq: 1 INVITE
    Contact: sip:bio-client@[local_ip]:[local_port]
    Max-Forwards: 70
    Subject: Biometric operation
    Require: bioanalysis
```

```
   Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, NOTIFY,
      MESSAGE, SUBSCRIBE, INFO, UPDATE
   Content-Type: application/sipbio+xml
   Content-Length: [len]

    [...]
   ]]>
 </send>
```

Listing 5.2: Fragment of SIPBIO server scenario

```
<scenario name="Basic Bio-Server">
<recv request="OPTIONS" crlf="true"></recv>
 <send>
   <![CDATA[
     SIP/2.0 200 OK
     [last_Via:]
     [last_From:]
     [last_To:]
     [last_Call-ID:]
     [last_CSeq:]
     Contact: <sip:[local_ip]:[local_port];transport=[transport]>
     Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, CANCEL, REFER
     Supported: bioanalysis, multibiopayload, multibiotype
     Content-Type: application/sipbio+xml
     Content-Disposition: properties-list
     Content-Length: [len]

     <?xml version="1.0" encoding="UTF-8" ?>
     <sipbio xmlns="urn:ietf:params:xml:ns:sipbio">
     <biotype>active passive off-line</biotype>
```

```
    <vendor>nuance microsoft google</vendor>
     <biopayload>voice face fingerprint</biopayload>
     <payloadSource>uri stream</payloadSource>
     </sipbio>
   ]]>
 </send>
 <recv request="INVITE" crlf="true"></recv>
```

In both, client and server scenarios, the same pattern can be identified:

- A scenario starts with the *scenario* keyword. Some keywords have properties, for instance, the *scenario* keyword has a property called *name* to label the scenario.
- An action is triggered by a keyword. In the case of listing 5.1, the first keyword is *send*. It indicates to the SIPp endpoint that the message inside the keyword tags is to be sent across to the other SIPp endpoint in the conversation.
- Meanwhile, at listing 5.2, the pair endpoint first action keyword (after its own definition) is *recv*. Note also that it just does not expect to receive anything, it expects to receive a SIP request of type *OPTIONS*. If, and only if, a SIP OPTIONS request is received, the endpoint carries on with the next action. In this case the next action is a *send* of a 200 OK SIP response.
- Consequently, the initial endpoint is waiting (*recv*) for a response of type 200.
- The same logic goes on until the end of the message which, as usual, is expected to be a BYE requests followed by a 200 OK response.

Once the scenarios are defined the endpoints, SIPBIO endpoints in this case, are made to run as such. For instance for the test layout shown in figure 5.1, the endpoints can be started as shown next.

UAS:> # sipp -sn basicBioServer

UAC:> # sipp -sn basicBioClient 10.3.36.98

Where *basicBioServer* and *basicBioClient* are the respective names of the server and client XML files, with the configuration of the SIPBIO scenarios to test. Both commands shown above are to be started from the command console of the incumbent servers.

5.2 Test scenarios

This section illustrates two basic test scenarios to illustrate the consistency and viability of SIPBIO.

Due to the nature of the SIPBIO XML body content, the maximum transmission unit (MTU) authorized in the network (usually 1500 bytes) may be easily be reached. This would result in IP fragmentation. The datagram is then split across multiple IP packets. When using UDP as a transport protocol, if a fragment is lost, the entire datagram is lost. To avoid this situation TCP was chosen as the transport protocol for the tests. RFC 3261 specifically acknowledges this situation when using UDP (Rosenberg *et al.*, 2002).

5.2.1 Base test scenario

The first scenario to test is the ideal scenario with no errors. To achieve client and server behaviour, templates with idealised event responses are created. The resulting SIPBIO events of the communication are shown in 5.2. This SIP trace is the reflection of the flow diagram shown in Figure 4.1. The corresponding trapezoid is shown in Figure 5.3, although the packet dissector of he analysis tool (Wireshark) does not take the initial OPTIONS message as part of the communication flow. A dissection of the individual messages follows. Since the objective of each SIP message has been previously explained, only the key features are commented on.

	Time	Source	Destination	Protocol	Length	Info
4	22:21:43.150681	10.3.36.55	10.3.36.98	SIP	448	Request: OPTIONS sip:bio-server@10.3.36.98:5060 |
6	22:21:43.151114	10.3.36.98	10.3.36.55	SIP/XML	845	Status: 200 OK |
8	22:21:43.151428	10.3.36.55	10.3.36.98	SIP/XML	1266	Request: INVITE sip:bio-server@10.3.36.98:5060 |
9	22:21:43.151544	10.3.36.98	10.3.36.55	SIP/XML	1237	Status: 200 OK |
10	22:21:43.151713	10.3.36.55	10.3.36.98	SIP/XML	1056	Request: ACK sip:bio-server@10.3.36.98:5060 |
11	22:21:43.155028	10.3.36.98	10.3.36.55	SIP/XML	715	Request: BYE sip:bio-server@:5060 |
12	22:21:43.155185	10.3.36.55	10.3.36.98	SIP	389	Status: 200 OK |

Figure 5.2 *Test scenario 1.*

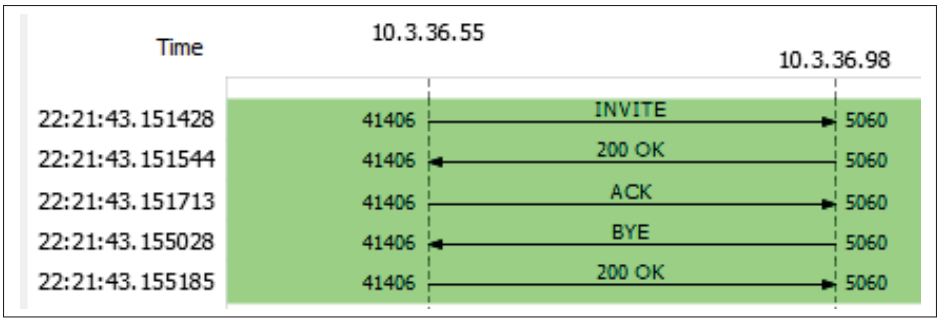


Figure 5.3 *Test scenario 1, associated trapezoid.*

OPTIONS The first message is the request from bio-client to bio-server to determine what features are available as shown in Figure 5.4. Nothing is specific to SIPBIO in the OPTIONS message.

```
▲ Session Initiation Protocol (OPTIONS)
  ▷ Request-Line: OPTIONS sip:bio-server@10.3.36.98:5060 SIP/2.0
  ▲ Message Header
    ▷ Via: SIP/2.0/TCP 10.3.36.55:5060;branch=z9hG4bK-7102-1-0
    ▷ From: bio-client <sip:bio-client@10.3.36.55:5060>;tag=7102SIPpTag001
    ▷ To: bio-server <sip:bio-server@10.3.36.98:5060>
      Call-ID: 1-7102@10.3.36.55
    ▷ CSeq: 1 OPTIONS
    ▷ Contact: sip:bio-client@10.3.36.55:5060
      Max-Forwards: 70
      Accept: application/sipbio
      Content-Length:     0
```

Figure 5.4 *Test scenario 1, OPTIONS.*

200 OK (to OPTIONS) This message is the beginning of the biometric-related content. The bio-server replies with a minimum set of supported biometric features for the bio-client to properly initiate the conversation. Figure 5.5 shows the relevant sections in the SIP

message. In the *Supported* header the bio-server notifies that it supports biometric analysis (*bioanalysis*) and that it is also able to support different types of biometric payloads as well as being able to support several types of interactions (*mutipayload* and *multibiotype*, respectively).

The message body, an XML structure, contains the specific types of payloads and biometric operations along with other key pieces of information: the bio-server informs that it supports reception of the payload by either a direct URL link (*uri*) or through a direct stream, where it requires a complementary protocol (e.g. SDP or MRCP). This specific bio-server is ready to support biometric parameters as handled by Nuance, Microsoft or Google. This *vendor* field is an example of different features that can be specified inside the XML structure. A bio-client can choose to only use the parameters provided by the bio-server, or fewer additional ones. The only ones sure to be accepted are the ones carried by this message.

INVITE This is the first message in the SIP communication exchange and the pillar of the SIPBIO interaction. A bio-client proposes to start a communication channel with the objective of transmitting biometric information and to carry related information. Figure 5.6 shows a sample in detail. Notably, the message header explicitly expresses that *bionalysis* must be supported, otherwise the exchange cannot be completed. In this example only one payload and one biometric type are to be chosen, this is why no explicit requirement for multi-feature support is required.

The body content includes the types explicitly expressed in the previous OPTIONS message but also expands on more detailed characteristics of the payload format (sampling frequency, file types available, encoding types and audio sides). It also introduces new content, namely, the claimed identity and grouping attribute. The bio-client is coded to probe with this variation, however, even if the bio-server does not explicitly express its support, this does not mean it is not capable of processing them.

200 OK (to INVITE) This message is the response to the INVITE. It starts, or in some cases settles, the negotiation. To the requests of a bio-client, the bio-server responds with what

Figure 5.5 *Test scenario 1, 200 OK to OPTIONS.*

it actually is able to support. For instance, in Figure 5.6 it is shown that the client can process active, passive and off-line biometric operations; the bio-server, however, replies that it only supports the two first types. Figure 5.7 shows the entire response.

ACK This is the message that establishes the final values to be used as a base for the media transmission, along with those to be passed to the biometric engine. It completes the customary three-way handshake for SIP communications. At this stage the bio-client client selects values for each component in the chosen XML structure. Figure 5.8 shows the resulting configuration in detail. This time the bio-client decided on an active voice biometric authentication for a Nuance-provided solution. The voice (payload) to be analysed is being supplied as a WAV file, reachable through a URL. The audio file contains

```
Session Initiation Protocol (INVITE)
  Request-Line: INVITE sip:bio-server@10.3.36.98:5060 SIP/2.0
  Message Header
    Via: SIP/2.0/TCP 10.3.36.55:5060;branch=z9hG4bK-17399-1-2
    From: bio-client <sip:bio-client@10.3.36.55:5060>;tag=17399SIPBIOtag001
    To: bio-server <sip:bio-server@10.3.36.98:5060>
    Call-ID: 1-17399@10.3.36.55
    CSeq: 1 INVITE
    Contact: sip:bio-client@10.3.36.55:5060
    Max-Forwards: 70
    Subject: Biometric operation
    Require: bioanalysis
    Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, NOTIFY, MESSAGE, SUBSCRIBE, INFO, UPDATE
    Content-Type: application/sipbio+xml
    Content-Length:   665
  Message Body
    eXtensible Markup Language
      <?xml
      <sipbio
          xmlns="urn:ietf:params:xml:ns:sipbio">
        <biotype>
            active passive off-line
            </biotype>
        <vendor>
            nuance microsoft google
            </vendor>
        <biopayload>
            voice face fingerprint
            </biopayload>
        <payloadSource>
            uri stream
            </payloadSource>
        <payloadFormat>
          <fileType>
              wav ogg mp3 m4a raw
              </fileType>
          <sampligFrequency>
              8000 16000 32000 44100 96000 192000
              </sampligFrequency>
          <encoding>
              16bit_linear_PCM 8bit_linear_PCM 8bit_logarithmic_A-law 8bit_logarithmic_mu-law
              </encoding>
          <audioSide>
              mono stereo
              </audioSide>
            </payloadFormat>
        <claimedIdentity>
            ID SSN acountID
            </claimedIdentity>
        <groupInfo>
            customer employee
            </groupInfo>
          </sipbio>
```

Figure 5.6 *Test scenario 1, INVITE.*

one single audio channel (mono) with encoding / acoustic characteristics of 8KHz, 8-bit A-Law. The audio is claimed to contain the voice of a customer identified with ID 71777287. The XML structure also contains some additional information, classified as metadata that may also be of interest, if not for the biometric transaction, then for reporting or extra actions coordinated by the bio-server.

At this point of the communication exchange the bio-client and bio-server, or the parties negotiated by them, exchange the biometric payload. In the example shown, an HTTP re-

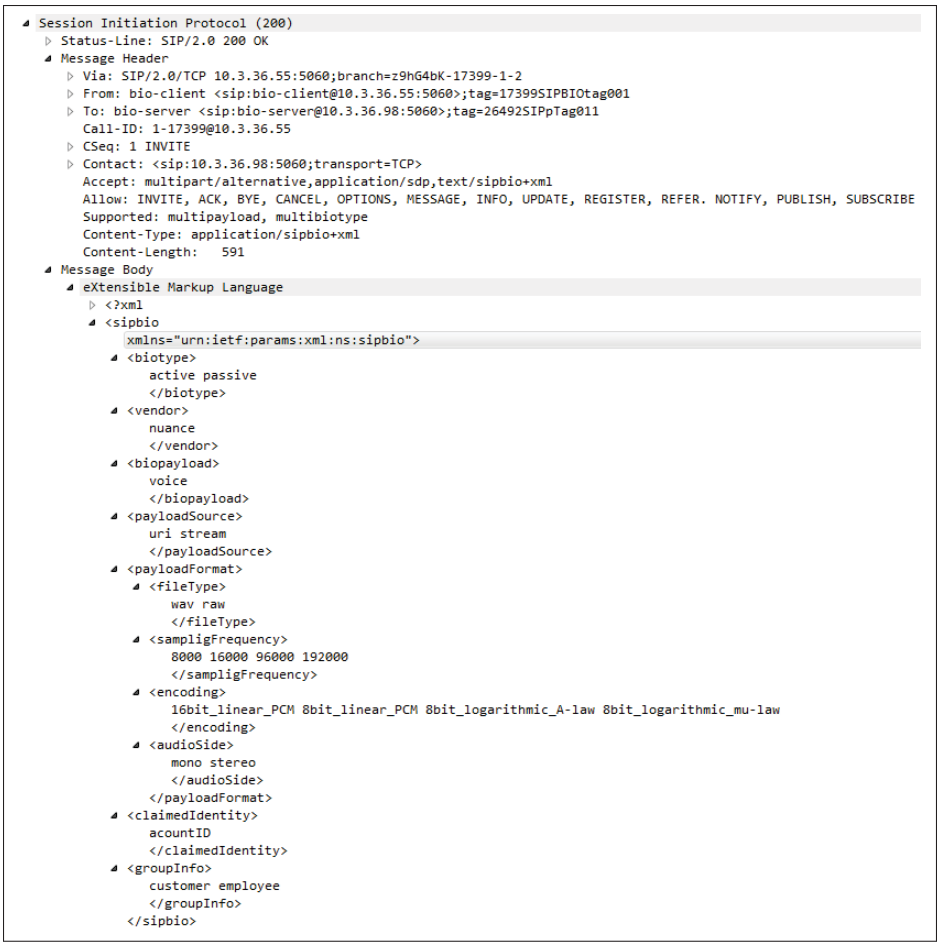


Figure 5.7 *Test scenario 1, 200 OK to INVITE.*

quest is likely to be made by the bio-server to the media server reachable by *mediarepository.corp*. The HTTP request aims at obtaining the wave audio file 201718111551003.

Once the media exchange is completed and the biometric results are available, the rest of the SIPBIO messaging exchange is completed as shown next.

BYE In most regular SIP operations, the BYE message is simply an indication of the wish of one of the parties to end the session. SIPBIO enhances the BYE message with message body content carrying the results of the biometric operation along with any other information suitable to the XML structure and usable by the bio-client. Figure 5.9 shows the content for the sample operation where the bio-server informs that customer 71777287

```
Session Initiation Protocol (ACK)
  Request-Line: ACK sip:bio-server@10.3.36.98:5060 SIP/2.0
  Message Header
    Via: SIP/2.0/TCP 10.3.36.55:5060;branch=z9hG4bK-17399-1-6
    From: bio-client <sip:bio-client@10.3.36.55:5060>;tag=17399SIPBIOtag001
    To: bio-server <sip:bio-server@10.3.36.98:5060>;tag=26492SIPpTag011
    Call-ID: 1-17399@10.3.36.55
    CSeq: 1 ACK
    Contact: sip:bio-client@10.3.36.55:5060
    Max-Forwards: 70
    Subject: Biometric operation
    Content-Length:   592
  Message Body
    eXtensible Markup Language
      <?xml
      <sipbio
          xmlns="urn:ietf:params:xml:ns:sipbio">
        <biotype>
            active
            </biotype>
        <vendor>
            nuance
            </vendor>
        <biopayload>
            voice
            </biopayload>
        <payloadSource>
            "http://mediarepository.corp/voice/201718111551003.wav"
            </payloadSource>
        <payloadFormat>
          <fileType>
              wav
              </fileType>
          <sampligFrequency>
              8000
              </sampligFrequency>
          <encoding>
              8bit_logarithmic_A-law
              </encoding>
          <audioSide>
              mono
              </audioSide>
          </payloadFormat>
        <claimedIdentity>
            71777287
            </claimedIdentity>
        <groupInfo>
            customer
            </groupInfo>
        <metadata>
            15142345654 financial visa silver
            </metadata>
        </sipbio>
```

Figure 5.8 *Test scenario 1, ACK.*

was previously registered and the provided audio file corresponds to their biometric characteristics with a confidence of 98.5%.

```
Session Initiation Protocol (BYE)
  Request-Line: BYE sip:bio-server@:5060 SIP/2.0
  Message Header
    Via: SIP/2.0/TCP 10.3.36.55:5060;branch=z9hG4bK-17399-1-6
    From: bio-client <sip:bio-client@10.3.36.55:5060>;tag=17399SIPBIOtag001
    To: bio-server <sip:bio-server@10.3.36.98:5060>;tag=26492SIPpTag011
    Call-ID: 1-17399@10.3.36.55
    CSeq: 2 BYE
    Contact: sip:bio-server@10.3.36.98:5060
    Max-Forwards: 70
    Subject: Biometric operation
    Content-Type: application/sipbio+xml
    Content-Length:   223
  Message Body
    eXtensible Markup Language
      <?xml
      <sipbio
          xmlns="urn:ietf:params:xml:ns:sipbio">
        <biometricResults>
          <entry
              registered="true"/>
          <entry
              match="true"/>
          <entry
              certainty="0.985"/>
        </biometricResults>
      </sipbio>
```

Figure 5.9 *Test scenario 1, BYE.*

200 OK (to BYE) The final message of the exchange that acknowledges its termination is the 200 OK to the BYE. This is a standard SIP message and it is only mentioned here for completeness.

```
Session Initiation Protocol (200)
  Status-Line: SIP/2.0 200 OK
  Message Header
    Via: SIP/2.0/TCP 10.3.36.55:5060;branch=z9hG4bK-17399-1-6
    From: bio-client <sip:bio-client@10.3.36.55:5060>;tag=17399SIPBIOtag001
    To: bio-server <sip:bio-server@10.3.36.98:5060>;tag=26492SIPpTag011
    Call-ID: 1-17399@10.3.36.55
    CSeq: 2 BYE
    Contact: <sip:10.3.36.55:5060;transport=TCP>
    Content-Length: 0
```

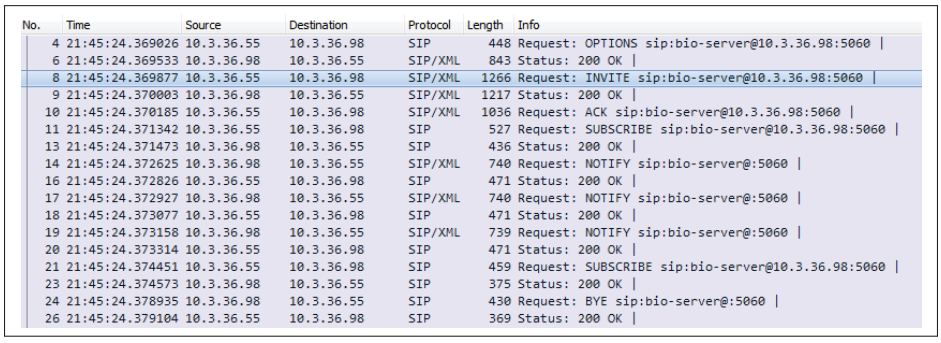
Figure 5.10 *Test scenario 1, 200 OK to BYE.*

5.2.2 Test scenario with partial results notifications

Previous sections focused on the simple biometric session where once provided with the payload information, sole feedback from the bio-server is the result. When the biometric operation takes a long time (for instance, more than two seconds) the bio-client requires partial results to apply programming logic or, simply, to notify other applications. This type of scenario corresponds to the one discussed in Subsection 4.4.7 and represented in Figure 4.5.

This test scenario is reproduced in the same conditions as the one discussed in the previous section. Since the initial session establishment are the same in both scenarios, only those messages that are new or different are dissected in the analysis.

A SIP trace of this scenario and its corresponding trapezoid are shown in figures 5.11 and 5.12 respectively.



No.	Time	Source	Destination	Protocol	Length	Info
4	21:45:24.369026	10.3.36.55	10.3.36.98	SIP	448	Request: OPTIONS sip:bio-server@10.3.36.98:5060 |
6	21:45:24.369533	10.3.36.98	10.3.36.55	SIP/XML	843	Status: 200 OK |
8	21:45:24.369877	10.3.36.55	10.3.36.98	SIP/XML	1266	Request: INVITE sip:bio-server@10.3.36.98:5060 |
9	21:45:24.370003	10.3.36.98	10.3.36.55	SIP/XML	1217	Status: 200 OK |
10	21:45:24.370185	10.3.36.55	10.3.36.98	SIP/XML	1036	Request: ACK sip:bio-server@10.3.36.98:5060 |
11	21:45:24.371342	10.3.36.55	10.3.36.98	SIP	527	Request: SUBSCRIBE sip:bio-server@10.3.36.98:5060 |
13	21:45:24.371473	10.3.36.98	10.3.36.55	SIP	436	Status: 200 OK |
14	21:45:24.372625	10.3.36.98	10.3.36.55	SIP/XML	740	Request: NOTIFY sip:bio-server@:5060 |
16	21:45:24.372826	10.3.36.55	10.3.36.98	SIP	471	Status: 200 OK |
17	21:45:24.372927	10.3.36.98	10.3.36.55	SIP/XML	740	Request: NOTIFY sip:bio-server@:5060 |
18	21:45:24.373077	10.3.36.55	10.3.36.98	SIP	471	Status: 200 OK |
19	21:45:24.373158	10.3.36.98	10.3.36.55	SIP/XML	739	Request: NOTIFY sip:bio-server@:5060 |
20	21:45:24.373314	10.3.36.55	10.3.36.98	SIP	471	Status: 200 OK |
21	21:45:24.374451	10.3.36.55	10.3.36.98	SIP	459	Request: SUBSCRIBE sip:bio-server@10.3.36.98:5060 |
23	21:45:24.374573	10.3.36.98	10.3.36.55	SIP	375	Status: 200 OK |
24	21:45:24.378935	10.3.36.98	10.3.36.55	SIP	430	Request: BYE sip:bio-server@:5060 |
26	21:45:24.379104	10.3.36.55	10.3.36.98	SIP	369	Status: 200 OK |

Figure 5.11 *Test scenario 2*.

Besides the obvious number of additional messages, there is one key difference in this scenario compared with the one in Section 5.2.1: biometric results are given as part of a NOTIFY and BYE messages do not carry any payload. These differences make this scenario more in line with regular usage of other SIP applications as explained in Subsection 4.4.7. In detail message information is shown next.

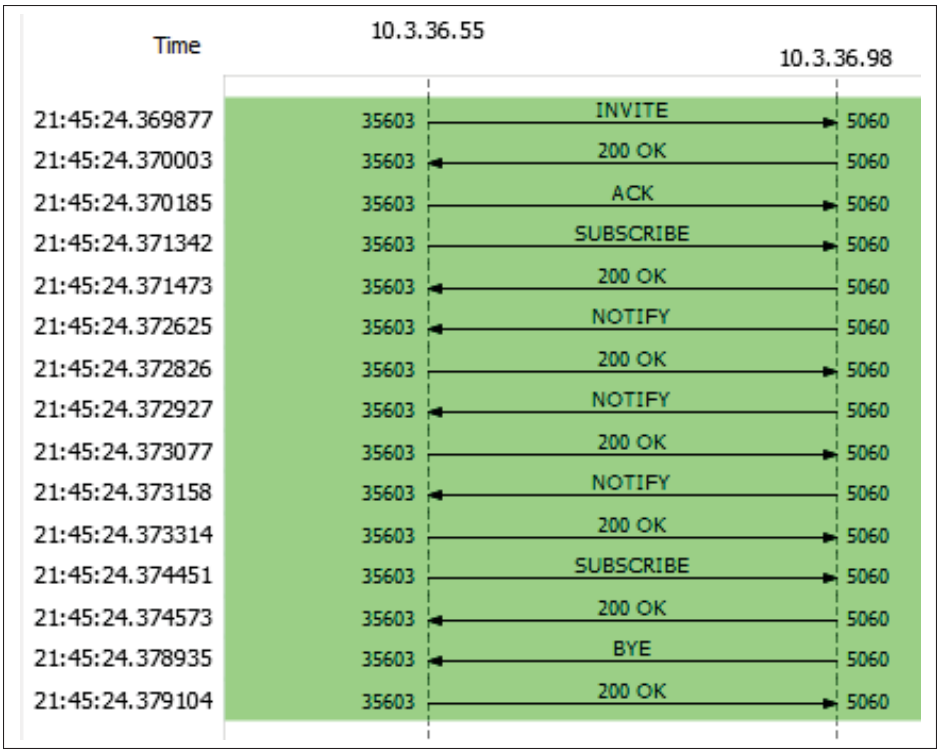


Figure 5.12 *Test scenario 2, associated trapezoid.*

SUBSCRIBE Once the three-way handshake is completed, the bio-client proceeds to sign up for bio-server events. The SUBSCRIBE message notably specifies the type of events being supported: *dialog* and *sipbioStatus* of type sipbio-xml. This message is followed by a 200 OK from the bio-server confirming the acceptance of the request.

```
Session Initiation Protocol (SUBSCRIBE)
  Request-Line: SUBSCRIBE sip:bio-server@10.3.36.98:5060 SIP/2.0
  Message Header
    Via: SIP/2.0/TCP 10.3.36.55:5060;branch=z9hG4bK-9390-1-2
    From: bio-client <sip:bio-client@10.3.36.55:5060>;tag=9390SIPBIOtag001
    To: bio-server <sip:bio-server@10.3.36.98:5060>
    Call-ID: 1-9390@10.3.36.55
    CSeq: 3 SUBSCRIBE
    Contact: sip:bio-client@10.3.36.55:5060
    Max-Forwards: 70
    Event: Biometric status
    Expires: 70
    Accept: application/sibio+xml
    Allow-Events: dialog, sipbioStatus
    Content-Length: 0
```

Figure 5.13 *Test scenario 2, SUBSCRIBE.*

NOTIFY A bio-server should notify its subscriber according to logic sensible for the application. It is up to the implementation to set the options properly. Information is provided to the bio-client as a *sipbioStatus* XML structure.

For the scenario that is being tested, the first NOTIFY is sent once a percentage of the payload has been analysed. In the current simulation a valuer of 40% payload analysis is shown for illustration purposes. The actual value would depend on the frequency of the notifications and the speed of the biometric engine. In cases where the payload size is not previously known, time based notifications, or any suitable parameter, can be used to trigger the message. Figure 5.14 shows the details. The XML content clearly states that it is a partial result for which no biometric decision has been already made based on an analysis certainty of 0%.

This interaction belongs to a verification, as already stated on previous messages. The implementation is free to choose parameters and interpretations of content. For the test case it is assumed that there is no match (positive biometric authentication) unless a threshold of certainty is reached.

Figure 5.15 shows the second set of results where there is still a high level of uncertainty, which leads to a mismatch conclusion albeit 80% of the payload has been already analysed. This partial result allows the bio-client to implement logic to inform third party dependent components (or end users) aware of the operation status.

The final set of results is shown in Figure 5.16. A certainty of 98.5% after having analysed 100% of the payload, allows to determine a match. Even though the represented interaction is a *toy* example of a real implementation, it exposes the work essential of the messages. A *sipbioStatus* structure can be expanded with as many as available pieces of information. Biometric engines provide different levels of results granularity that can be used to enrich the NOTIFY action when relevant.

Un-SUBSCRIBE Once the bio-client receives al information it is expecting, it should proceed to un-subscribe from notifications. This is accomplished by sending a new SUBSCRIBE with a *Expires* header field equals to zero as displayed in Figure 5.17. At this point the

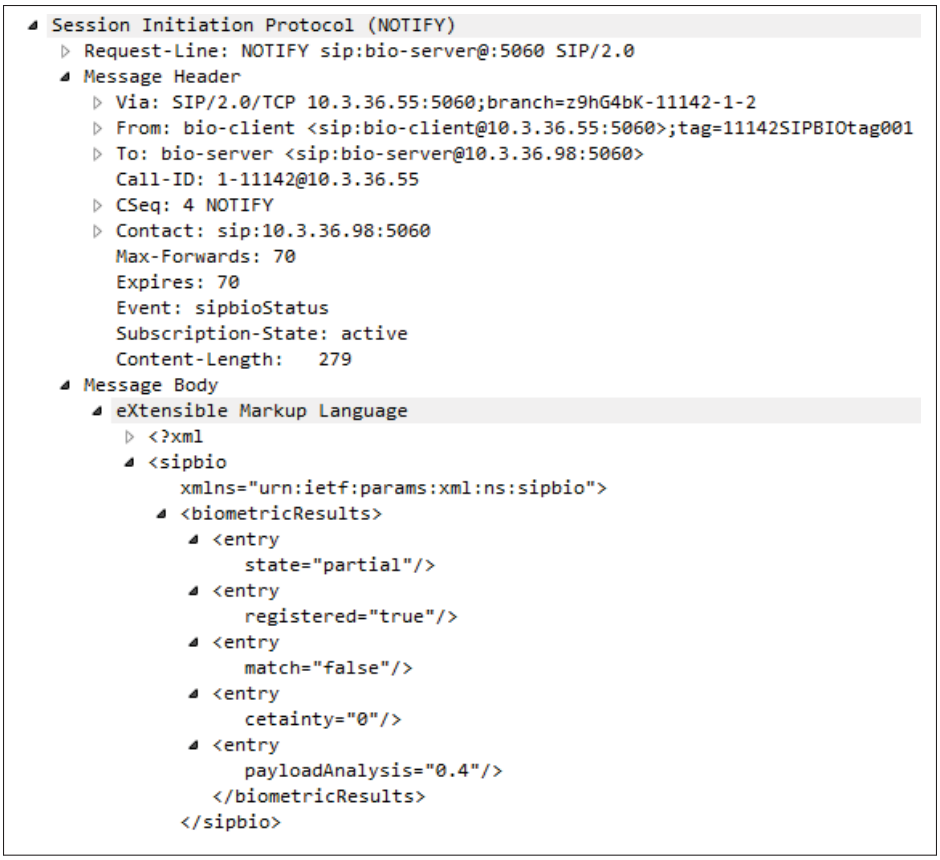


Figure 5.14 *Test scenario 2, NOTIFY (partial result 1).*

biometric process is completed. The implementation can either enhance the process with new biometric operations over the same session or wrap it up.

BYE In contrast to the simple scenario described in Subsection 5.2.1, this time the BYE message does not constitute part of the biometric operation and it is simply used as a regular SIP message to trigger the end of the dialogue. Comparably, this is a more orthodox usage of a BYE. Developers can use this approach to maintain higher compatibility with other SIP implementations.

These two sections are merely an introduction to feasible implementation scenarios for SIPBIO. They were prepared in order to illustrate the validity of SIPBIO as viable communication protocol under the conditions of operations of biometric transactions. The groundwork has been laid for customisation of the protocol if required. All XML structures are flexible to al-

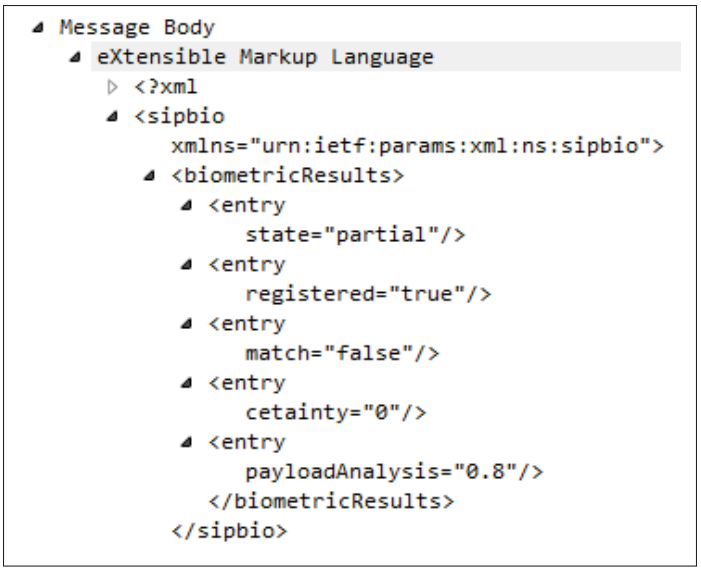


Figure 5.15 *Test scenario 2, NOTIFY (partial result 2).*

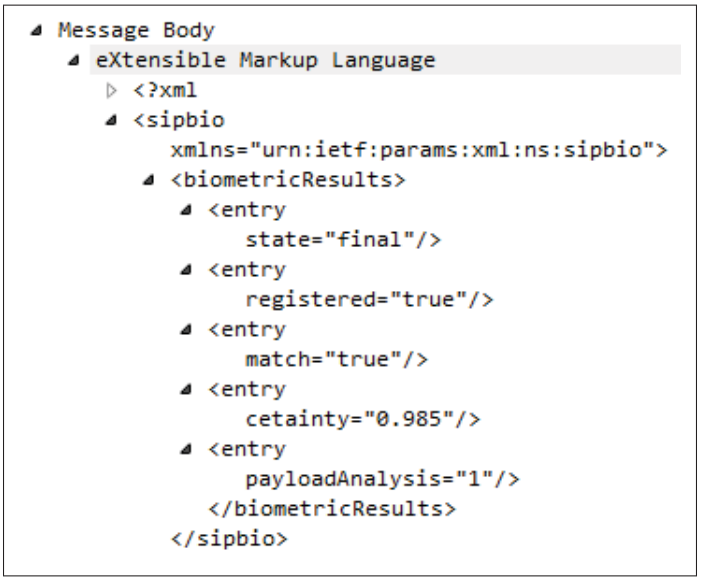


Figure 5.16 *Test scenario 2, NOTIFY (Final result).*

low for any type of implementation requirement and the construction of any other operational scenario.

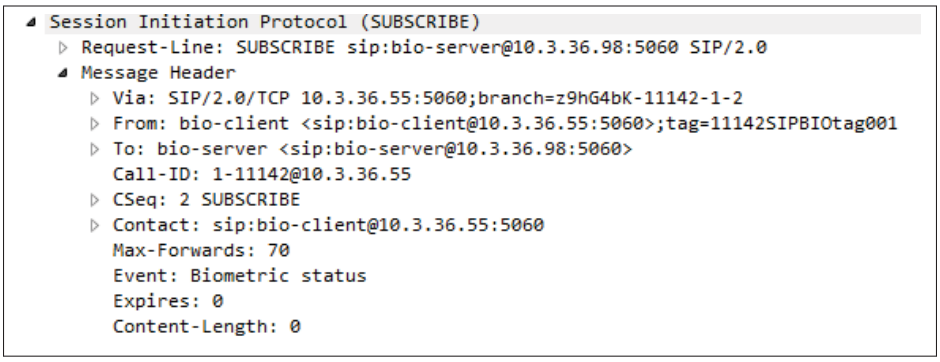


Figure 5.17 *Test scenario 2, Un-SUBSCRIBE.*

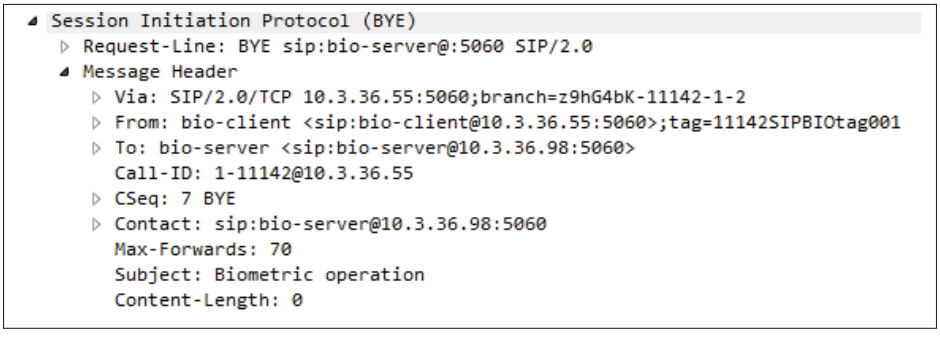


Figure 5.18 *Test scenario 2, BYE.*

CONCLUSION AND RECOMMENDATIONS

As biometric technologies become omnipresent in everyday life, there is a need for standard integration methodologies. This thesis proposes to satisfy this need by creating the logical framework for SIPBIO. The question of this research is if it is possible to extend SIP to cover cases of biometric interactions.

This work began by presenting SIP as a general-use protocol with applicability to a limitless number of situations. The basics of biometric technologies and their use in telecommunications networks were examined, leading to the exploration of expanding the base SIP protocol to support alternative requirements and scenarios. The foundation of the proposal, through scenarios covered by SIPBIO, were explained and presented; this was followed by an outline of the concepts with a detailed expansion of the proposed protocol. Finally, a simulation of a potential implementation was presented to guide prospective SIPBIO developers.

This thesis starts from the basic SIP protocol as defined in RFC 3261 by Rosenberg *et al.* (2002) and follow the guidelines to author SIP extensions as proposed by Rosenberg and Schulzrinne (2006). It develops a new protocol, SIPBIO, to support session establishment for executing biometric transactions. The proposal was proven to be feasible.

Previous works led the way to identify a mechanism to design protocol extensions. Of singular importance for the development of a working design were the proposal to extend SIP to support payments and micropayments developed by Ruiz-Martinez *et al.* (2016); Ruiz-Martínez *et al.* (2007); SIPBIO borrows the same algorithm of communication establishment and the same model of data transmission but applies them to biometric interactions.

Two key characteristics of biometric operations were found amenable to the concept of SIP oriented operations:

- A biometric interaction requires a stage of session establishment where general characteristics are negotiated.
- A biometric operation aims to exchange media content between parties.

This proposal explains how these two characteristics led to a set of clear standardization policies and a corroborated conclusion that a protocol to drive biometric exchanges could be created through an extension to the base SIP protocol. SIPBIO demonstrates that it is possible to expand a well-known - widely adopted protocol, SIP, to facilitate biometric interactions.

Through the development of SIPBIO, the assumption that biometric transactions could be represented as groups of SIP messages, was corroborated. SIPBIO follows the path created by previous protocols like SIPREC, which extended SIP to create a solution to problems faced in the industry. In the case of SIPREC, it is call recording, while for SIPBIO it is biometric transactions. This trend is a validation of the capacity of SIP to mutate depending on the requirement whist still maintaining its core functionality. It has been attested during the development of the protocol that SIP indeed allows for flexibility without compromising structure.

Being a SIP-based protocol, SIPBIO is easily comprehended by developers and implementers. It can be simulated using standard tools such as SIPp, largely easing the actual development of a solution based on proven scenarios. Applications can be developed using well known, industry standard, IDE such as Eclipse[4] or IntellyJ[5].

SIPBIO does not pretend to handle data security. It relies on SIP basic mechanisms of authentication and the capabilities of lower layers to keep information and communication channels private and secured. Due to the nature of the information handled by SIPBIO, it is not advisable to implement it without any appropriate security measures to protect data and associated meta-

[4] www.eclipse.org

[5] www.jetbrains.com/idea/

data. Try to have it handle security as well would not add value to SIPBIO. It is recommended to leave that role to the appropriate layers.

Although SIPBIO has the theoretical basics required to implement any biometric operation in a variety of environments, further work needs to be completed in order to provide better integration with industry vendor CTI solutions. Unfortunately, there is not a CTI communication standard, every vendor exposes a different set of APIs. This leads to costly bespoke solutions. As CTI data is required in most passive biometric integrations, a middleware is a valid approach to bridge CTI and SIPBIO enabled systems. It would make an interesting project to propose a SIPBIO branch to handle CTI information. Being SIP, by design, a signalling protocol a new SIPBIO-based protocol could be proposed to primarily handle CTI messages.

This research demonstrates the feasibility of the protocol by playing scenarios that can be used later by developers to parametrise their designs. A practical project derived from this thesis could be the creation of a SIPBIO SDK implementation. Any existing SIP SDK should be the starting point to build one for SIPBIO. It is not advisable to create an entire implementation from scratch.

An interesting practical project derived from this thesis would be a stress test of the protocol. This could be done by placing several scenarios running in parallel between the same actors with slight differences in the requested the biometric operation. Requests must be generated in the same volume as in commercial systems. This type of test would expose any flow in the protocol logic under real time scenarios, including but not limited to performance and race conditions.

During the development of the protocol proposal, SIPBIO was presented to application developers and system architects with experience in telecommunications. The reception was not always warm. While architects appreciated the effort to provide a standard platform for

exchanging biometric transactions, developers were not as interested. The latter group had difficulty seeing any utility at all in the proposal because they saw the problem as having been already solved by classical programming techniques where application transactions are usually transported through HTTP or other socket communications. It is understood that the SIPBIO niche must start in the telecommunications industry with vendors able to provide API and SDK featuring the capabilities of the protocol. With the rise of biometric applications in mobile environments as well as in the financial industry, the call for easier, faster and standardised methods to implement solutions opens a plethora of opportunities for SIPBIO to be used.

APPENDIX I

LIST OF ABBREVIATIONS AND ACRONYMS

- **3G** Third Generation
- **3GPP** 3rd Generation Partnership Project
- **AOR** Address-of-Record
- **API** Application Program Interface
- **API** Application Program Interface
- **B2BUA** Back-to-Back User Agent
- **CSP** Communication Sequential Processes
- **DB** Database
- **DHCP** Dynamic Host Control Protocol
- **DRM** Digital Rights Management
- **EAP** Extensible Authentication Protocol
- **ECG** Electrocardiogram
- **EER** Equal Error Rate
- **ETS** École de Technologie Supérieure
- **ETSI** European Telecommunications Standards Institute
- **FA** False Accept
- **FAR** False Acceptance Rate
- **FQDN** Fully Qualified Domain Name

- **FR** False Reject
- **FRR** False Rejection Rate
- **HTTP** Hyper Text Transfer Protocol
- **HTTP** Hypertext Transfer Protocol
- **HTTPS** Secured Hypertext Transfer Protocol
- **IDE** Integrated Development Environment
- **IDM3G** Identity Management Protocol
- **IETF** Internet Engineering Task Force
- **ISP** Internet Service Provider
- **IVR** Interactive Voice Response
- **LAN** Local Area Network
- **MO** Mobile Operator
- **MSRP** Message Session Relay Protocol
- **MTU** Maximum transmission unit
- **OASIS** Organization for the Advancement of Structured Information Standards
- **OS** Operative System
- **OTAVB** One-time Active Voice Biometrics authentication
- **PIN** Personal Identification Number
- **PKI** Public Key Infrastructure
- **PPBA** privacy preserving biometric authentication protocol

- **QoS** Quality of Service
- **RFC** Request for Comments
- **RTP** Real-time Transport Protocol
- **SBC** Session Border Controller
- **SDK** Software Development Kit
- **SDN** Software Defined Networking
- **SDP** Session Description Protocol
- **SIM** Subscriber Identification Module
- **SIP** Session Initiation Protocol
- **SIPREC** Session Initiation Protocol Recording
- **SNMP** Simple Network Management Protocol
- **SP** Service Provider
- **SRC** Session Recording Client
- **SRS** Session Recording Server
- **TCP** Transmission Control Protocol
- **TDM** Time-division multiplexing
- **TLS** Transport Layer Security
- **TMN** Telecommunications Management Networks
- **UA** User Agent
- **UAC** UA Client

- **UAS** UA Server
- **UDP** User Datagram Protocol
- **UID** Unique Identification
- **UMTS** Universal Mobile Telecommunications Systems
- **URI** Universal Resource Identifier
- **USIM** user's SIM or Universal Subscriber Identity Module
- **USS** User Session Signature
- **VPN** Virtual Private Network
- **VoIPSEC** Voice Interactive Personalized Security protocol

BIBLIOGRAPHY

Abidin, A. and Mitrokotsa, A. (2014). Security of a privacy-preserving biometric authentication protocol revisited. *Proc. of the Yet Another Conference on Cryptography (YACC)*, pp. 1-16.

Adamczyk, P., Hafiz, M. and Johnson, R. (2008). *Non-compliant and Proud: A Case Study of HTTP Compliance*. Consulted at http://hdl.handle.net/2142/11424.

Agbinya, J. I., Mastali, N., Islam, R. and Phiri, J. (2011, Nov). Design and implementation of multimodal digital identity management system using fingerprint matching and face recognition. *Proc. of the 7th International Conference on Broadband Communications and Biomedical Applications*, pp. 272-278.

Aimilia, T., Efraimidis, P., Yannis, S., Lilian, M. and Vasilios, K. (2016). Privacy-preserving, user-centric VoIP CAPTCHA challenges: An integrated solution in the SIP environment. *Information and computer security*, 24(1), 2-19.

Benavente, O. S. and Piccio-Marchetti, R. (2005, Oct). Authentication services and biometrics: network security issues. *Proc. of the 39th Annual International Carnahan Conference on Security Technology*, pp. 333-336.

Beranek, B. (2013). Voice biometrics: Success stories, success factors and what's next. *Biometric technology today*, 2013(7), 9-11.

Berners-Lee, T., Fielding, R. T. and Masinter, L. (2005). *Uniform Resource Identifier (URI): Generic Syntax* (Report n°66). RFC Editor. Consulted at http://www.rfc-editor.org/rfc/rfc3986.txt.

Bringer, J., Chabanne, H., Izabachène, M., Pointcheval, D., Tang, Q. and Zimmer, S. (2007). An application of the goldwasser-micali cryptosystem to biometric authentication. In *Information Security and Privacy: 12th Australasian Conference - ACISP 2007* (pp. 96-106).

Camarillo, G., Eriksson, G., Holler, J. and Schulzrinne, H. (2002). *Grouping of Media Lines in the Session Description Protocol (SDP)* (Report n°3388). RFC Editor. Consulted at https://tools.ietf.org/rfc/rfc3388.txt.

Chappel, L. and Combs, G. (2017). *Wireshark 101: Essential skills for network analysis*. Reno, NV: Protocol Analysis Intitute.

Chen, Y., De, S., Kernchen, R. and Moessner, K. (2012). Device discovery in future service platforms through SIP. *Proc. of the 2012 IEEE Vehicular Technology Conference (VTC Fall)*, pp. 1-5.

Cisco. (2018). Achieving SIP RFC Compliance. In Cisco (Ed.), *SIP Configuration Guide* (pp. 57-108). Cisco Systems Inc.

Dassouki, K., Safa, H. and Hijazi, A. (2014). End to end mechanism to protect sip from signaling attacks. *Proc. of the 6th International Conference on New Technologies - Mobility and Security (NTMS)*, pp. 1-5.

Datt, S. (2016). Collecting network traffic using tcpdump. In *Learning Network Forensics* (pp. 38-44). Birmingham, UK: Packt Publishing.

Desai, A. (2016). Contact centres: how to escape the games of 'guess who?'. *Biometric technology today*, 2016(4), 8-11.

Devlic, A. (2010). SIP-based context distribution: does aggregation pay off? *SIGCOMM Comput. Commun. Rev.*, 40(5), 35-46.

Dimitriadis, C. and Polemi, D. (2006). An identity management protocol for internet applications over 3g mobile networks. *Computers and security*, 25(1), 45-51.

Dimitriadis, C. and Shaikh, S. (2007). A Biometric Authentication Protocol for 3G Mobile Systems: Modelled and Validated Using CPS and Rank Functions. *International journal of network security*, 5(1), 99-111.

European Telecommunications Standards Institute ETSI. (1997). Introduction to standardizing security for TMN. ETR 336. Valbonne, FR: European Telecommunications Standards Institute ETSI.

Fall, K. (1999, July). Network emulation in the vint/ns simulator. *Proceedings IEEE International Symposium on Computers and Communications (Cat. No.PR00250)*, pp. 244-250.

Fielding, R. T., Gettys, J., Mogul, J. C., Nielsen, H. F., Masinter, L., Leach, P. J. and Berners-Lee, T. (1999). *Hypertext Transfer Protocol – HTTP/1.1* (Report n°2616). RFC Editor. Consulted at http://www.rfc-editor.org/rfc/rfc2616.txt.

Friedlander, J. (2015, march). News and Notes on RIAA Shipment and Revenue Statistics. Consulted at https://www.riaa.com/wp-content/uploads/2016/03/RIAA-2015-Year-End-shipments-memo.pdf.

Gafurov, D. (2010). Emerging biometric modalities: Challenges and opportunities. In *Security Technology, Disaster Recovery and Business Continuity: International Conferences, SecTech and DRBC 2010* (pp. 29-38).

Gayraud, R. and Jacques, O. (2014). SIPp (version 3.4.) [Software]. Consulted at http://sipp.sourceforge.net.

Gurbani, V. and Sun, X. (2004). Extensions to an internet signaling protocol to support telecommunication services. *Proc. of the IEEE Global Telecommunications Conference - GLOBECOM'04*, 3, 1640-1644.

Handley, M., Jacobson, V. and Perkins, C. (2006). *SDP: Session Description Protocol* (Report n°4566). RFC Editor. Consulted at http://www.rfc-editor.org/rfc/rfc4566.txt.

Hoare, C. A. R. (1978). Communicating sequential processes. *Communications of the acm*, 21(8), 666–677.

Huston, G., Lord, A. and Smith, P. (2004). *IPv6 Address Prefix Reserved for Documentation* (Report n°3849). RFC Editor. Consulted at https://tools.ietf.org/rfc/rfc3840.txt.

Hutton, A., Portman, L., Jain, R. and Rehor, K. (2014). *An architecture for media recording using the session initiation protocol* (Report n°7245). RFC Editor. Consulted at https://trac.tools.ietf.org/rfc/rfc7245.txt.

Jain, A., Bolle, R. and Pankanti, S. (2006). Introduction to biometrics. In Jain, A., Bolle, R. and Pankanti, S. (Eds.), *Biometrics, Personal Identification in Networked Society* (ed. 1, pp. 1-38). New York, NY: Springer.

Jain, A., Nandakumar, K. and Ross, A. (2016). 50 years of biometric research: Accomplishments, challenges, and opportunities. *Pattern recognition letters*, 79, 80-105.

Johnson, R., Boult, T. and Scheirer, W. (2014). Voice authentication using short phrases: Examining accuracy, security and privacy issues. *Proc. of the Sixth IEEE International Conference on Biometrics: Theory, Applications and Systems (BTAS)*, pp. 1-8.

Kaplan, H. (2014). *A Session Identifier for the Session Initiation Protocol (SIP)* (Report n°7329). RFC Editor. Consulted at https://tools.ietf.org/rfc/rfc7329.txt.

Kikuchi, H., Nagai, K., Ogata, W. and Nishigaki, M. (2010). Privacy-preserving similarity evaluation and application to remote biometrics authentication. *Soft computing*, 14(5), 529-536.

Kopsidas, S., Zisiadis, D. and Tassiulas, L. (2006). Voice Interactive Personalized Security (VoIPSEC) protocol: Fortify Internet telephony by providing end-to-end security through inbound key exchange and biometric verification. *Proc. of the First IEEE Workshop on Hot Topics in Web Systems and Technologies*, pp. 1-10.

Kuseler, T., Lami, I., Jassim, S. and Sellahewa, H. (2010). eBiometrics: An enhanced multi-biometrics authentication technique for real-time remote applications on mobile devices. *Proc. of the Mobile Multimedia/Image Processing, Security, and Applications*, pp. 7708 - 7709.

Kyzivat, P., Ravindranath, R. and Ravindran, P. (2016). *Session Initiation Protocol (SIP) Recording Metadata* (Report n°draft-ietf-siprec-metadata-22). Consulted at http://www.ietf.org/internet-drafts/draft-ietf-siprec-metadata-22.txt.

Lapere, M. and Johnson, E. (1997). User authentication in mobile telecommunication environments using voice biometrics and smartcards. *Proc. of the 4th International Conference on Intelligence in Services and Networks*, pp. 437-443.

Lee, P. M. (2012). Probability and bayes' theorem. In Lee, P. M. (Ed.), *Bayesian Statistics: An Introduction* (ed. 4, pp. 1-10). John Wiley and Sons Ltd.

Lennox, J., Westerlund, M., Wu, Q. and Perkins, C. (2017). *Sending Multiple RTP Streams in a Single RTP Session* (Report n°8108). RFC Editor. Consulted at https://tools.ietf.org/rfc/rfc8108.txt.

Li, C.-T. and Hwang, M.-S. (2010). An efficient biometrics-based remote user authentication scheme using smart cards. *Journal of network and computer applications*, 33(1), 1-5.

Mallery, J. (2013). Building a secure organization. In Vacca, J. R. (Ed.), *Computer and Information Security Handbook* (ed. 2, pp. 1-24). Waltham, MA: Morgan Kaufmann.

Martin, L. (2013). Biometrics. In Vacca, J. (Ed.), *Computer and Information Security Handbook* (ed. 2, pp. 157-972). Waltham, MA: Morgan Kaufmann.

Martinez, R. (2008a). SIP overview. In Martinez, R. (Ed.), *Internet Multimedia Communications Using SIP: A Modern Approach Including Java® Practice* (ed. 1, pp. 43-58). Burlington, MA: Morgan Kaufmann.

Martinez, R. (2008b). SIP Protocol Operation. In Martinez, R. (Ed.), *Internet Multimedia Communications Using SIP: A Modern Approach Including Java® Practice* (ed. 1, pp. 73, 75-112). Burlington, MA: Morgan Kaufmann.

Middleton, S. E. and Modafferi, S. (2016). Scalable classification of QoS for real-time interactive applications from IP traffic measurements. *Computer networks*, 107, 121-132.

Nomura, R., Ishikawa, Y., Umeda, T., Takata, M., Kamo, H. and Joe, K. (2015). Biometrics authentication based on chaotic heartbeat waveform. *7th biomedical engineering international conference 2014*, pp. 1-5.

Osterhout, G. (2003). Comparison of SIP Proxy and Redirect Servers [Power Point Presentation]. Consulted at www.cs.columbia.edu/sip/talks/von9909_nortel.ppt.

Ott, J. (2001). Location Servers [Power Point Presentation]. Consulted at http://www.cs.columbia.edu/sip/talks/von2001-sip-location-servers.pdf.

Portman, L., Lum, H., Eckel, C., Johnston, A. and Hutton, A. (2016). *Session recording protocol* (Report n°7866). RFC Editor. Consulted at https://datatracker.ietf.org/doc/rfc7866/.

Ravindran, P. and Kyzivat, P. (2016). *Session Initiation Protocol (SIP) Recording Call Flows* (Report n°draft-ietf-siprec-callflows-06). Consulted at http://www.ietf.org/internet-drafts/draft-ietf-siprec-callflows-06.txt.

Rehor, K., Portman, L., Hutton, A. and Jain, R. (2011). *Use Cases and Requirements for SIP-Based Media Recording (SIPREC)* (Report n°6341). RFC Editor. Consulted at https://www.rfc-editor.org/rfc/pdfrfc/rfc6341.txt.

Reid, P. (2003a). An introduction to statistical measures of biometrics. In Reid, P. (Ed.), *Biometrics for Network Security* (ed. 1, pp. 141-154). Upper Saddle River, NJ: Prentice Hall PTR.

Reid, P. (2003b). Authentication technologies. In Reid, P. (Ed.), *Biometrics for Network Security* (ed. 1, pp. 9-22). Upper Saddle River, NJ: Prentice Hall PTR.

Resnick, P. (2001). *Internet message format* (Report n°2822). RFC Editor. Consulted at https://www.ietf.org/rfc/rfc2822.txt.

Roach, A. B. (2002). *Session Initiation Protocol (SIP)-Specific Event Notification* (Report n°3265). RFC Editor. Consulted at https://www.ietf.org/rfc/rfc3265.txt.

Rosenberg, J. and Schulzrinne, H. (2006). *Guidelines for Authors of Extensions to the Session Initiation Protocol (SIP)* (Report n°4485). RFC Editor. Consulted at https://tools.ietf.org/rfc/rfc4485.txt.

Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M. and Schooler, E. (2002). *SIP: Session Initiation Protocol* (Report n°3261). RFC Editor. Consulted at http://www.rfc-editor.org/rfc/rfc3261.txt.

Ruiz-Martinez, A., Sanchez-Laguna, J. A. and Skarmeta, A. F. (2016). Extending SIP to support payments in a generic way. *Computer standards and interfaces*, 46, 23-36.

Ruiz-Martínez, A., Sánchez-Laguna, J. A. and Gómez-Skarmeta, A. F. (2007). SIP extensions to support (micro)payments. *21st International Conference on Advanced Information Networking and Applications, AINA 2007*, pp. 289-296.

Ryu, J. T., Ron, B.-H., Ryu, K. Y. and Yoon, M. (2012). Detection and countermeasure scheme for call-disruption attacks on SIP-based voIp services. *KSII Transactions on Internet and Information Systems*, 6(7), 1854-1873.

Saevanee, H., Clarke, N., Furnell, S. and Biscione, V. (2015). Continuous user authentication using multi-modal biometrics. *Computers and security*, 53, 234-246.

Schneider, S. (1998). Verifying authentication protocols in CSP. *IEEE Transactions on Software Engineering*, 24(9), 741-758.

Schulzrinne, H. (2001). The Session Initiation Protocol [Power Point Presentation]. Consulted at http://www.cs.columbia.edu/~hgs/teaching/ais/slides/2003/sip_long.pdf.

Sisalem, D., Kuthan, J. and Ott, J. (2013). A short history of VoIP services. In *Evolution of Telecommunication Services: The Convergence of Telecom and Internet: Technologies and Ecosystems* (pp. 90-110).

Stanek, J. and Kencl, L. (2011). SIPp-DD: SIP DDoS Flood-attack Simulation Tool. *Proc. of the 20th International Conference on Computer Communications and Networks*, pp. 7.

Steffen, A., Kaufmann, D. and Stricker, A. (2004). SIP Security. *E-science und grid, ad-hoc-netze, medienintegration*, 18. DFN-Arbeitstagung über Kommunikationsnetze, Düsseldorf(GI-Edition), 397-410.

Subramanian, S. and Dutta, R. (2013). PSTN and VoIP Services Context. In Subramanian, S. and Dutta, R. (Eds.), *Measuring SIP Proxy Server Performance* (pp. 5-13). Springer.

Sui, Y., Zou, X., Du, E. and Feng, L. (2012, Oct). Secure and privacy-preserving biometrics based active authentication. *Proc. of the IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 1291-1296.

Syta, E., Fischer, M., Wolinsky, D., Silberschatz, A., Gallegos-Garcia, G. and Ford, B. (2015). Private eyes: Secure remote biometric authentication. *Proc. of the 12th International Conference on Security and Cryptography*, pp. 243-250.

Tanenbaum, A. and Wetherall, D. (2011a). Introduction. In Tanenbaum, A. and Wetherall, D. (Eds.), *Computer Networks* (ed. 5, pp. 23-24). Prentice Hall.

Tanenbaum, A. and Wetherall, D. (2011b). Introduction. In Tanenbaum, A. and Wetherall, D. (Eds.), *Computer Networks* (ed. 5, pp. 51-52). Prentice Hall.

Traore, I., Woungang, I., Obaidat, M. S., Nakkabi, Y. and Lai, I. (2014). Online risk-based authentication using behavioral biometrics. 71(Multimedia Tools and Applications), 575-605.

Wilber, M. and Boult, T. (2012). Secure remote matching with privacy: Scrambled support vector vaulted verification. *Proc. of the Workshop on Applications of Computer Vision (WACV)*, pp. 169-176.

Wilber, M., Scheirer, W. and Boult, T. (2012). PRIVV: Private remote iris-authentication with Vaulted Verification. *Proc. of IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pp. 97-104.

Wong, K.-S. and Kim, M. H. (2012). A privacy-preserving biometric authentication protocol. *Advanced science letters,*, 9(1), 683-688.

Xi, K., Ahmad, T., Han, F. and Hu, J. (2011). A fingerprint based bio-cryptographic security protocol designed for client/server authentication in mobile computing environment. *Security and communication networks*, 4(5), 487-499.

Yan, M. and Kyzivat, P. (2015). *Overview for MSRP Recording based on SIPREC* (Report n°draft-yan-siprec-msrp-recording-04). Consulted at http://www.ietf.org/internet-drafts/draft-yan-siprec-msrp-recording-04.txt.

Zave, P., Cheung, E., Bond, G. W. and Smith, T. M. (2009). Abstractions for programming SIP back-to-back user agents. *Proc. of the 3rd International Conference on Principles - Systems and Applications of IP Telecommunications*, pp. 1-12.