TESIS DE LA UNIVERSIDAD DE ZARAGOZA
2021
221

Jorge Sancho Larraz

Desing and evaluation of novel authentication, authorization and border protection mechanisms for modern information security architectures

Director/es
García Moros, José
Alesanco Iglesias, Álvaro

ISSN 2254-7606

Prensas de la Universidad
Universidad Zaragoza

ISSN 2254-7606

Universidad Zaragoza
1542

Tesis Doctoral

DESING AND EVALUATION OF NOVEL AUTHENTICATION, AUTHORIZATION AND BORDER PROTECTION MECHANISMS FOR MODERN INFORMATION SECURITY ARCHITECTURES

Autor

Jorge Sancho Larraz

Director/es

García Moros, José
Alesanco Iglesias, Álvaro

UNIVERSIDAD DE ZARAGOZA
Escuela de Doctorado

2021

Universidad Zaragoza
1542

Tesis Doctoral

DESING AND EVALUATION OF NOVEL AUTHENTICATION, AUTHORIZATION AND BORDER PROTECTION MECHANISMS FOR MODERN INFORMATION SECURITY ARCHITECTURES

Autor

Jorge Sancho Larraz

Directores

Álvaro Alesanco Iglesias
José García Moros

Escuela de Ingeniería y Arquitectura

2021

A todas las personas que me han ayudado a llegar hasta aquí,
Gracias

Resumen

En los últimos años, las vidas real y digital de las personas están más entrelazadas que nunca, lo que ha dado lugar a que la información de los usuarios haya adquirido un valor incalculable tanto para las empresas como para los atacantes. Mientras tanto, las consecuencias derivadas del uso inadecuado de dicha información son cada vez más preocupantes. El número de brechas de seguridad sigue aumentando cada día y las arquitecturas de seguridad de la información, si se diseñan correctamente, son la apuesta más segura para romper esta tendencia ascendente.

Esta tesis contribuye en tres de los pilares fundamentales de cualquier arquitectura de seguridad de la información —**autenticación, autorización y seguridad de los datos en tránsito**— mejorando la seguridad y privacidad provista a la información involucrada. En primer lugar, la **autenticación** tiene como objetivo verificar que el usuario es quien dice ser. Del mismo modo que otras tareas que requieren de interacción por parte del usuario, en la autenticación es fundamental mantener el balance entre seguridad y usabilidad. Por ello, hemos diseñado una metodología de autenticación basada en el fotopletismograma (PPG). En la metodología propuesta, el modelo de cada usuario contiene un conjunto de ciclos aislados de su señal PPG, mientras que la distancia de Manhattan se utiliza para calcular la distancia entre modelos. Dicha metodología se ha evaluado prestando especial atención a los resultados a largo plazo. Los resultados obtenidos muestran que los impresionantes valores de error que se pueden obtener a corto plazo (valores de EER por debajo del 1%) crecen rápidamente cuando el tiempo entre la creación del modelo y la evaluación aumenta (el EER aumenta hasta el 20% durante las primeras 24 horas, valor que permanece estable desde ese momento). Aunque los valores de error encontrados en el largo plazo pueden ser demasiado altos para permitir que el PPG sea utilizado como una alternativa de autenticación confiable por si mismo, este puede ser utilizado de forma complementaria (e.g. como segundo factor de autenticación) junto a otras alternativas de autenticación, mejorándolas con interesantes propiedades, como la prueba de vida.

Después de una correcta autenticación, el proceso de **autorización** determina si la acción solicitada al sistema debería permitirse o no. Como indican las nuevas leyes de protección de datos, los usuarios son los dueños reales de su información, y por ello deberían contar con los métodos necesarios para gestionar su información digital de forma efectiva. El framework

OAuth, que permite a los usuarios autorizar a una aplicación de terceros a acceder a sus recursos protegidos, puede considerarse la primera solución en esta línea. En este framework, la autorización del usuario se encarna en un token de acceso que la tercera parte debe presentar cada vez que desee acceder a un recurso del usuario. Para desatar todo su potencial, hemos extendido dicho framework desde tres perspectivas diferentes. En primer lugar, hemos propuesto un protocolo que permite al servidor de autorización verificar que el usuario se encuentra presente cada vez que la aplicación de terceros solicita acceso a uno de sus recursos. Esta comprobación se realiza mediante una autenticación transparente basada en las señales biométricas adquiridas por los relojes inteligentes y/o las pulseras de actividad y puede mitigar las graves consecuencias de la exfiltración de tokens de acceso en muchas situaciones. En segundo lugar, hemos desarrollado un nuevo protocolo para autorizar a aplicaciones de terceros a acceder a los datos del usuario cuando estas aplicaciones no son aplicaciones web, sino que se sirven a través de plataformas de mensajería. El protocolo propuesto no lidia únicamente con los aspectos relacionados con la usabilidad (permitiendo realizar el proceso de autorización mediante el mismo interfaz que el usuario estaba utilizando para consumir el servicio, i.e. la plataforma de mensajería) sino que también aborda los problemas de seguridad que surgen derivados de este nuevo escenario. Finalmente, hemos mostrado un protocolo donde el usuario que requiere de acceso a los recursos protegidos no es el dueño de estos. Este nuevo mecanismo se basa en un nuevo tipo de concesión OAuth (grant type) para la interacción entre el servidor de autorización y ambos usuarios, y un perfil de OPA para la definición y evaluación de políticas de acceso. En un intento de acceso a los recursos, el dueño de estos podría ser consultado interactivamente para aprobar el acceso, habilitando de esta forma la delegación usuario a usuario.

Después de unas autenticación y autorización exitosas, el usuario consigue acceso al recurso protegido. La **seguridad de los datos en tránsito** se encarga de proteger la información mientras es transmitida del dispositivo del usuario al servidor de recursos y viceversa. El cifrado, al tiempo que mantiene la información a salvo de los curiosos, también evita que los dispositivos de seguridad, como los firewalls, puedan inspeccionar la información, detectando posibles amenazas. Sin embargo, mostrar la información de los usuarios a dichos dispositivos podría suponer un problema de privacidad en ciertos escenarios. Por ello, hemos propuesto un método basado en Computación Segura Multiparte (SMC) que permite realizar las funciones

de red sin comprometer la privacidad del tráfico. Esta aproximación aprovecha el paralelismo intrínseco a los escenarios de red, así como el uso adaptativo de diferentes representaciones de la función de red para adecuar la ejecución al estado de la red en cada momento. En nuestras pruebas hemos analizado el desencriptado seguro del tráfico utilizando el algoritmo Chacha20, mostrando que somos capaces de evaluar el tráfico introduciendo latencias realmente bajas (menores de 3ms) cuando la carga de la red permanece suficientemente baja, mientras que podemos procesar hasta 1.89 Gbps incrementando la latencia introducida. Teniendo en cuenta todo esto, a pesar de la penalización de rendimiento que se ha asociado tradicionalmente a las aplicaciones de Computación Segura, hemos presentado un método eficiente y flexible que podría lanzar la evaluación segura de las funciones de red a escenarios reales.

Abstract

People's real and digital lives are now closer than ever before. As a result, user information has become invaluable for companies and attackers, while the repercussions from misusing it are causing increasing concern. Although the number of data breaches has consistently climbed year after year, information security architectures are the safest way to break this rising trend.

This Thesis contributes to three Information Security Architectures fundamental pillars — **authentication**, **authorization** and **data in-transit security**— with the aim of enhancing the security and privacy afforded to the information in question. The first stage is user authentication, to ascertain whether users are who they claim to be. As is the case with other tasks requiring user interaction, striking the right balance between security and usability has proved critical in authentication tasks. We have designed a novel authentication method to verify user identity based on photoplethysmogram (PPG)—a biomedical signal estimating volumetric blood flow changes in peripheral circulation by means of infrared light absorption. The process consists in generating users' templates as a set of their independent PPG cycles, while Manhattan distance is used to perform the matching. The proposed method has been evaluated by focusing on long-term results. We have shown that the excellent error values obtained in the short term (EER values were lower than 1% depending on the dataset) quickly rise as the time between user enrollment and the actual authentication increases (EER values rise up to 20% within the first 24 hours and remain stable from then on). Although error values found in the long term may prevent the PPG from being used as a reliable user authentication alternative on its own, it may supplement other authentication alternatives (as a second factor of authentication) enhancing them with further interesting properties, such as a proof of live.

After successful authentication, subsequent **authorization** is required to determine whether an entity should be allowed to perform an action. As new data protection regulation states, users are the real owners of their information and, therefore, they must be provided with appropriate methods to effectively manage their digital information. The first solution along these lines was the OAuth framework, which enabled users to authorize a third-party application to gain access to their protected resources. This authorization takes the form of an access token that the third-party application must hold in order to retrieve any user's resources. We have enhanced this

framework in three ways. First, we proposed a protocol allowing the authorization server to seamlessly verify that users are present on each access attempt by using biometric signals gathered from their wearable devices, which can mitigate the major consequences of access token leakage in many scenarios. Second, we developed a new protocol to authorize third-party applications to access user information when they are not web applications but are services provided through messaging platforms. The proposed protocol not only enhances the usability aspects, allowing the authorization to be performed through the same interface the user is utilizing to interact with the third party —i.e., the messaging platform— but also addresses the security issues raised in this new scenario. Finally, we put forward a protocol whereby the user requesting access to the resources is not the resource owner. This new mechanism, based on a new OAuth grant type for the interaction between the authorization server and both users and an OPA profile to deal with policy definition and evaluation, allows that the resource owner may be interactively asked to approve a given access attempt, thus enabling user-to-user delegation.

After correct authentication and authorization, users gain access to the protected resource. **Data in-transit security** protects the information while it is transmitted from the user's device to the resource server and the way back. Encryption—hiding information from curious eyes—also prevents border protection devices, such as firewalls, from providing their functionality. However, information privacy may be hampered by making it available to these devices. We have proposed a method based on secure multiparty computation (SMC) that allows middleboxes to provide their functionality while preserving information privacy; this is known as secure network functions (SNF). The proposed approach takes advantage of the parallelism in network scenarios and uses varying SNF representations to adapt the evaluation to the network state every time. We have analyzed the secure traffic decryption with the ChaCha20 algorithm in our tests, which demonstrates that we can process the traffic by introducing low latencies (less than 3 ms) when the network load remains low enough, while we can process up to 1.89 Gbps at the cost of increasing the latency introduced. All in all, despite the overheads traditionally associated with SMC applications, we have presented an efficient and flexible method that may push SNF to real-world scenarios.

Scientific Contributions

Next peer-reviewed scientific publications have been derived from the research of this thesis.

Publications in International Journals (JCR indexed).

Sancho, J., Alesanco, Á. and García, J., 2018. Biometric authentication using the PPG: a long-term feasibility study. *Sensors*, *18*(5), p.1525.

Sancho, J., García, J. and Alesanco, A., 2020. Oblivious Inspection: On the Confrontation between System Security and Data Privacy at Domain Boundaries. Security and Communication Networks, 2020.

Submitted Publications in International Journals (JCR indexed).

Sancho, J., García, J. and Alesanco, A., 2021. Authorizing third-party applications served through messaging platforms. Computers & Security.

Sancho, J., García, J. and Alesanco, A., 2021. Can Messaging platforms align authorization requirements with users' habits?. Computers & Security.

Publications in International Conferences and Proceedings.

Sancho, J., Alesanco, A. and Garca, J., 2017. Photoplethysmographic authentication in long-term scenarios: A preliminary assessment. In *EMBEC & NBC 2017* (pp. 1085-1088). Springer, Singapore.

Sancho, J., Mikkelsen, G.L., Lindstrøm, J., García, J. and Alesanco, Á., 2019, September. On the Privacy Enhancement of In-Transit Health Data Inspection: A Preliminary Study. In *Mediterranean Conference on Medical and Biological Engineering and Computing* (pp. 855-860). Springer, Cham.

Sancho, J., García, J. and Alesanco, Á., 2020, November. Patient Identification Workflow for Seamless EHR Access During Patient Follow-Up. In *European Medical and Biological Engineering Conference* (pp. 962-967). Springer, Cham.

Sancho, J., García, J. and Alesanco, Á., 2020, November. Distributed Access Control for Cross-Organizational Healthcare Data Sharing Scenarios. In *European Medical and Biological Engineering Conference* (pp. 407-413). Springer, Cham.

Related Publications in International Journals (JCR indexed).

Hernando, D., Roca, S., **Sancho, J**., Alesanco, Á. and Bailón, R., 2018. Validation of the apple watch for heart rate variability measurements during relax and mental stress in healthy subjects. *Sensors*, *18*(8), p.2619.

Roca, S., **Sancho, J.**, García, J. and Alesanco, A., 2020. Microservice chatbot architecture for chronic patient support. *Journal of Biomedical Informatics*, *102*, p.103305.

Related Publications in International Conferences and Proceedings.

Alesanco, Á., **Sancho, J.**, Gilaberte, Y., Abarca, E. and García, J., 2017. Bots in messaging platforms, a new paradigm in healthcare delivery: application to custom prescription in dermatology. In EMBEC & NBC 2017 (pp. 185-188). Springer, Singapore.

Roca, S., Hernández, M., **Sancho, J.**, García, J. and Alesanco, Á., 2019, September. Virtual Assistant Prototype for Managing Medication Using Messaging Platforms. In *Mediterranean Conference on Medical and Biological Engineering and Computing* (pp. 954-961). Springer, Cham.

Contents

List of Figures

List of Tables

Acronyms

1DMRLBP	One-Dimensional Multi-Resolution Local Binary Patterns
ABAC	Attribute-Based Access Control
API	Application Programing Interface
ATM	Automatic Teller Machine
CIA	Confidenciality, Integrity and Availability
CPU	Central Processing Unit
CWT	Continuous Wavelet Transform
DAC	Discretionary Access Control
DDBB	Data Bases
DLDA	Direct Linear Discriminant Analysis
DTN	Delay Tolerant Networks
DWT	Discrete Wavelet Transform
ECG	Electrocardiogram
EEG	Electroencephalogram
EER	Equal Error Rate
EHR	Electronic Health Record
EMG	Electromyogram
ES	Evaluation Set
FIDO	Fast IDentity Online
FIDO2	Fast IDentity Online 2
FMR	False Match Rate
FNMR	False Non-Match Rate
GDPR	General Data Protection Regulation
GMW	Goldreich-Micali-Wigderson
GPU	Graphics Processing Unit
HTML	HyperText Markup Language
HTTP2	HyperText Markup Language 2
HTTPS	HyperText Markup Language Secure
ICT	Information and Communication Technology
IDS	Intrusion Detection System

IKE	Internet Key Exchange
IO	Input/Output
IP	Internet Protocol
JSON	JavaScript Object Notation
JWE	JSON Web Encryption
JWS	JSON Web Signature
JWT	JSON Web Token
KLT	Karhunen-Loève Transform
k-NN	k-Nearest Neighbors
KPCA	Kernel Principal Component Analysis
KS	Kolmogorov–Smirnov Test
LDA	Linear Discriminant Analysis
MAuth	Messaging platform application Authorization
MAC	Mandatory Access Control
MCE	Maximum Concurrent Evaluations
MIMIC2	Multiparameter Intelligent Monitoring in Intensive Care 2
NAT	Network Address Traslation
NFC	Near-Field Communication
OAuth	Open Authorization
OPA	Open Policy Agent
OT	Oblivious Transfer
OTP	One Time Password
PD	Pearson Distance
PIN	Personal Identification Number
PKCE	Proof Key for Code Exchange
PPG	Photoplethysmogram
PRRB	Photoplethysmography Respiratory Rate Benchmark
RAM	Random Access Memory
RBAC	Role Based Access Control
RFC	Request For Comments
SCD	Simple Circuit Description
SeamAuth	Seamless Authorization
SIMD	Single Instruction, Multiple Data

SMC	Secure Multi-Party Computation
SSL	Secure Sockets Layer
SSO	Single Sign-On
TCP	Transmission Control Protocol
TCS	Temporal Cycles Set
TLS	Transport Layer Security
U2UAuth	User-to-user Authorization
UMA	User-Managed Access
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
USB	Universal Serial Bus
UTF	Unicode Transformation Format
VPN	Virtual Private Network
XACML	eXtensible Access Control Markup Language

Chapter 1: Introduction

People's real and digital lives are now closer than ever before. As a result, user information has become invaluable for companies, while the repercussions from misusing it are causing increasing concern. Many countries have adopted comprehensive data protection laws—such as the General Data Protection Regulation (GDPR) in Europe—to protect this asset and ensure users can securely manage their digital counterpart, thereby preventing abusive behavior by companies. Within the framework of these regulations, users, who are the real owners of their information, must be able to manage the entire information lifecycle, from generation to destruction, including aspects such as who can access their information and for which purpose. However, compliance with these regulations is not straightforward in most cases, and adopting them poses numerous challenges.

1.1 Information Security

In recent years, many companies have shifted their focus to user information. Fitness device manufacturers, such as Polar or Garmin, are good examples of this trend. With the arrival of globalization, many foreign companies from emerging markets now compete with aggressive pricing. Local companies, in a comfortable position for years, realized they could no longer limit themselves to producing the hardware; if they wanted to keep their market share, they needed to provide their customers with differentiating value. In most cases, this value took the form of services, such as custom workout programs or performance reports based on the information collated during training sessions. However, this business model had to be adopted carefully to avoid new risks and threats arising from this new connected scenario. As user information is so vital for companies, security architectures, whose objective is to protect resources that are important for the business, needed to evolve to protect it [1]. Besides elements intended to provide security to underlying layers (network, infrastructure and application), new areas must be addressed at an information level to guarantee information confidentiality, integrity and availability [2] (the CIA triad).

1.1.1 Information Security Architecture

Information Security Architectures (see Figure 1.1) are used for the authentication of involved parties, authorization of access attempts, audit logs, data-in-transit and at-rest security.

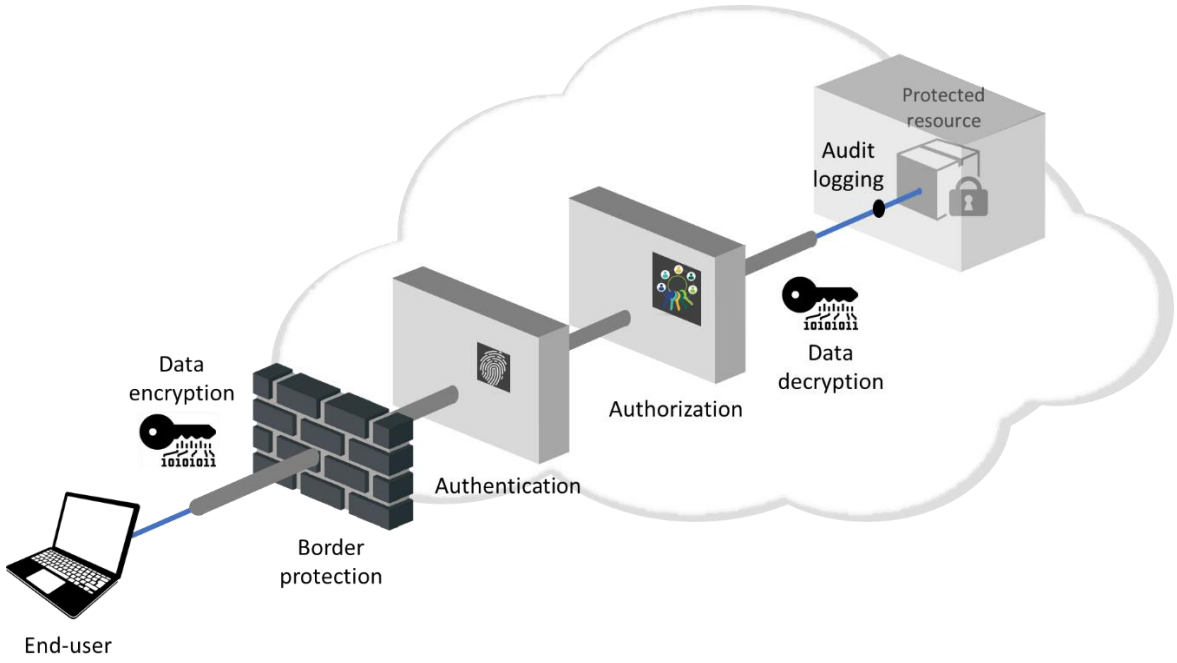

Figure 1.1 Information Security Architecture.

Authentication

Authentication is the first step to gain access to an information system. The process consists in ascertaining that an entity is who it claims to be. In general, it is widely accepted that there are three approaches to user authentication: something only the user knows (e.g., passwords, PINs, etc.); something only the user has (e.g., smart cards, digital certificates, etc.); and something only the user is (biometrics) [3]. No single approach is better than the others per se; choosing a particular method depends on a thorough study of the application scenario. Combining two or more approaches from different classes is known as two-factor (or multi-factor) authentication [4]. The use of several authentication factors might improve the overall authentication security by providing the best of each approach.

Independently of the credential type used to prove users identity, authentication can be direct or brokered [1]. In direct authentication, the entity wanting to prove its identity provides its credentials to the other party; for instance, when users access a webpage directly by providing username and password. Brokered authentication, however, involves a trusted third party, usually referred to as an identity provider; in this case, the entity wanting to prove its identity presents its credentials to the identity provider, which issues an assertion about the entity's identity that the other party trusts. Social login, which allows using Google or Facebook accounts to access a variety of services, is an example of brokered authentication.

Authorization

After successful authentication, subsequent authorization is required to determine what the user is allowed to do. The mechanism to assign access rights is known as delegation, and two parties intervene in it: the delegator (a.k.a. grantor), who gives the access rights, and the delegate, who receives them [5], [6]. In general outlines, the delegate could be software or a device acting on the user's behalf (e.g., an ATM transaction) or another user needing to access protected resources for any reason (e.g., a patient's treatment process, which may require the collaborative efforts of multiple parties [7]). There are two types of delegation: administrative delegation and user delegation [8]. In administrative delegation, administrative users grant other users access rights that the administrative users themselves may or may not have, while in user delegation, users assign other users a subset of their rights. As the authors of [8] state, administrative delegation is more permanent, while user delegation is usually intended for a specific short-lived purpose. Administrative delegation has traditionally been implemented by writing access policies under the restrictions of a specific access control model. These policies are evaluated on each access attempt to decide whether access is granted. The OAuth 2.0 framework has recently enabled user delegation when the delegate is a third-party application. However, user-to-user delegation, in which both the delegator and the delegate are users, has not been successfully adopted other than in a few isolated scenarios.

Audit logging

Auditing consists of keeping track of what is happening in the system. Thus, any access attempt must be tracked irrespective of the authentication and authorization result. Tracking illegal access attempts is required to identify possible threats, while tracking legitimate access attempts facilitates nonrepudiation [9]. Even with all precautions, something will happen at a certain time; whether due to some misconfigured permission or a new vulnerability on the underlying layers, something will happen at some point. Audit trials are useful to detect that something is wrong, how it happened and, more importantly, the extent of the threat. There are also cases in which some entities have rights to access a given resource, but only under certain conditions. For example, an emergency physician can access any patient information under the break-the-glass assumption [10]. However, if someone accesses patients' information frequently using this situational empowerment, audit trials would help to detect improper behavior.

Data in-transit security

After successful authentication and authorization, users can access the protected resource. Data-in-transit security protects the information while it is being transmitted from one device to another. The most widespread means of dealing with data-in-transit security is encryption, which can be applied at different layers of the network model. In a TCP/IP network, we can provide information security at the IP layer using IPsec [11], at a transport layer using transport layer security (TLS) [12] or at an application layer with solutions such as JSON web encryption (JWE) [13] and JSON web signature (JWS) [14]. Currently, the prevailing solution to address data-in-transit security is TLS; it allows a secure communication channel that protects the security of all the information flowing through it. TLS is not limited to providing confidentiality, since communication peer authentication, secure encryption key generation and integrity checks are also provided.

Besides encryption, best practices for robust data-in-transit protection include robust network security controls [15], such as border protection filtering incoming and outgoing traffic to prevent malware reaching the company network and protected information leaks.

Data at rest security

Data at-rest security protects information while it remains in one place for any length of time. Therefore, it should not be limited to protecting the information while it is stored on the remote server but also on the user's device. If not treated properly, data at rest implies a vulnerability that has gone unnoticed for decades. Anyone gaining physical access to the device would be able to access this valuable information. In recent years, with the increase in mobility and "bring your own device" policies, this problem has become critical. Companies private information is on employees' devices, which are certainly exposed to theft [16].

As happens for data-in-transit security, the most widespread method to deal with data-at-rest security is the use of encryption. Encryption has been viewed as the panacea; however, it is associated with some drawbacks. Using it involves a performance penalization that must be considered when the system is dimensioned. Encryption may be classified as follows, based on the layer it is applied to: preboot encryption, file/folder encryption, database encryption, and application encryption [17]. It is not necessarily limited to just one of these layers as any combination of them is also possible, depending on how sensitive the information involved is and the requirements of each specific scenario.

1.1.2 Challenges

A notorious change has recently taken place in the information-security paradigm. In this new model, applications and services from companies usually require sharing user information between them to provide improved functionality [18], [19] (always on users' behalf and with their consent)—for instance, Google Fit can obtain information from fitness device manufacturers cloud services, thus providing users with an information hub. Several security mechanisms and frameworks already exist to cover the areas defined by the information security architecture under various assumptions. However, as technology evolves, so do these assumptions, and security mechanisms and frameworks must be reviewed to keep pace with changes introduced in the scenario. This thesis contributes to some information security architecture areas (specifically, authentication, authorization and data-in-transit security) from several perspectives.

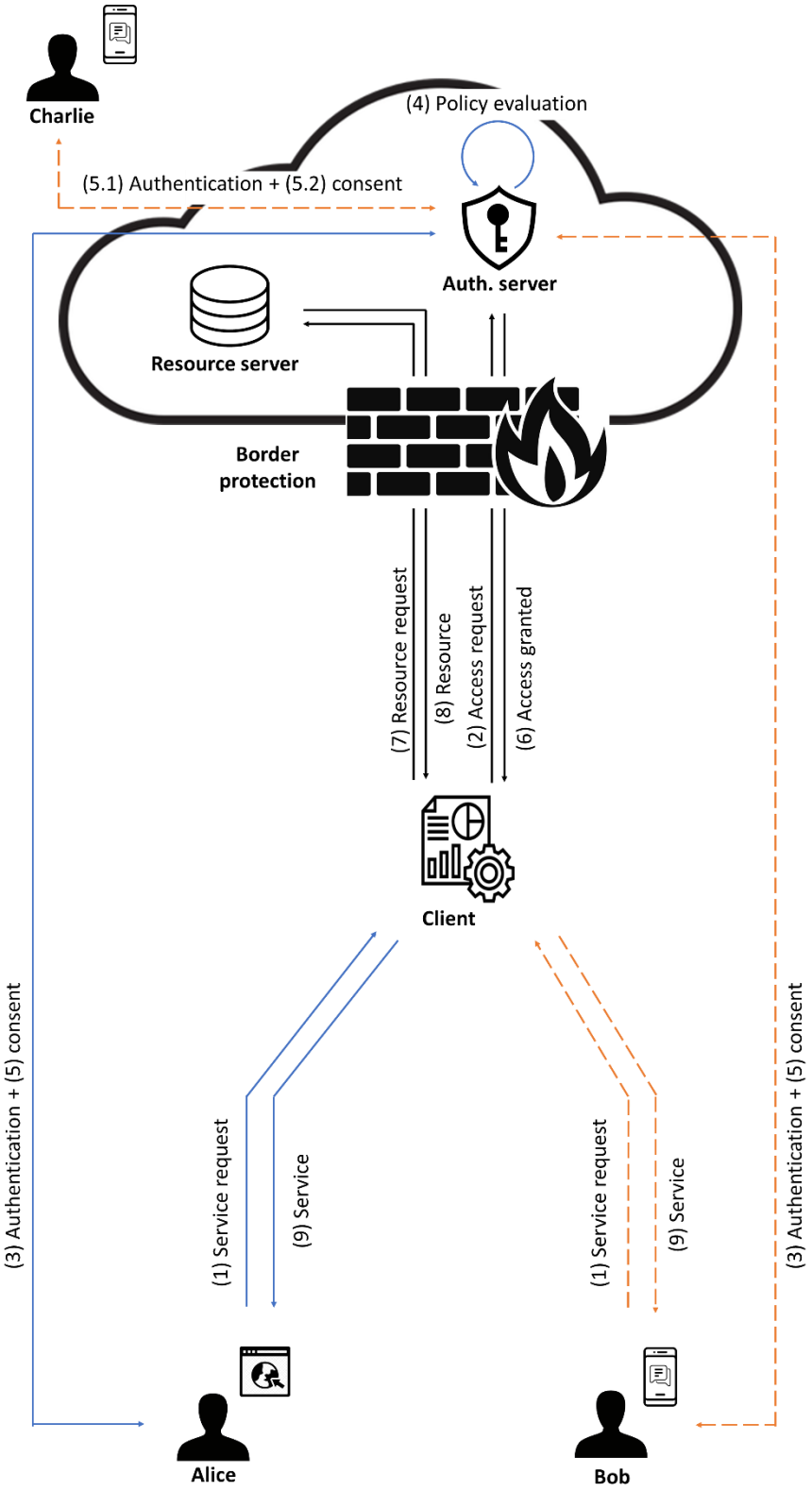

Figure 1.2 General scenario.

All contributions made by this thesis are part of the scenario shown in Figure 1.2, which presents a user, Alice, who wants to obtain a service provided by a web application, the client (1). In turn, to provide the requested service, the client needs access to some of Alice's protected resources held by the resource server (2). To approve the client's request, the authorization server firstly prompts the user to authenticate herself (3). Passwords have been the widespread authentication method for decades. As services move online, however, the attack surface has increased and more complex password policies are required to prevent brute force attacks. Complex passwords are hard for any user to remember, and they, therefore, write them down—on post-it notes, for example—thus negating all the significance of the strong password policy. The proliferation of smartwatches and wristbands, which already acquire signals, such as the photoplethysmogram (PPG) and the electrocardiogram (ECG) for health and fitness purposes [20]–[24], makes these signals perfect candidates to supplement or substitute passwords for seamless user authentication.

After successful user authentication, the access request needs to be authorized. The authorization server evaluates access policies to determine whether Alice is granted access to the requested resource (4). Given that she is the owner of the requested resource, the policy evaluation result would be access permission. However, as the web application making the access request (on Alice's behalf) does not belong to the same company holding the protected resource, the authorization server must also obtain Alice's consent to grant the client access to her protected resource (5). If everything goes as expected, access is granted (6) and the client can retrieve the protected resource from the resource server (7 and 8), thus providing Alice with the requested service (9).

Similarly, another user, Bob, wants to access the service provided by the client but using a different communication channel, a messaging platform. Figure 1.2 shows communications through the messaging platform using dashed orange lines. These communications are routed through the messaging platform server and end-to-end encrypted between communication peers. Although the problem to solve is essentially the same as in Alice's access attempt (authenticate the user, determine that the user has the required rights to access the requested resource and obtain the user's consent to allow the client to retrieve the protected resource on their behalf) issues arising from this new communication pattern must be carefully addressed.

In some situations, providing the service requested by Alice or Bob may require the client to access a protected resource that belongs to another user, Charlie; this scenario would fall under the user-to-user delegation. To support this delegation type, the policy definition and evaluation must be flexible enough to allow users to define their own policies; however, other delegation mechanisms should also be considered to fit in a wider range of scenarios. For example, there might be situations in which Alice or Bob do not know about their need to access the protected resource or they are not even registered with the authorization server when Charlie updates his access policies. In these instances, the policy evaluation result (4) would be to decline access; however, they might still be allowed to interactively request Charlie to approve the access attempt (5.1 and 5.2). Messaging platforms may be an appropriate way to establish this communication with the resource owner, aligning authorization tasks with users' daily routines.

Finally, all communications must be encrypted end-to-end as part of data-in-transit security while a border protection device filters incoming and outgoing traffic to prevent malware reaching the company network or protected information leakage. Encryption—hiding information from curious eyes—also prevents border protection devices from providing their functionality. Since the information is encrypted, the data the border protection device can use to decide whether forwarding the packet poses a risk or not is limited to the network and transport headers. However, making all the information available to the border protection device might involve a privacy concern that cannot be risked in every situation. This leads to a dilemma: the more information used to decide whether a packet might pose a security concern, the more reliable the decision is (security), but the more information available, the more data flow confidentiality is compromised (privacy). Secure computation protocols have lately attracted much attention due to applications such as secure auctions [25], [26], in which parties' bids are never revealed to any other stakeholder in the scenario. This powerful cryptographic tool has the required potential to align security and privacy at domain boundaries, thus allowing border protection devices to reliably provide their functionality without compromising the privacy of the information involved.

1.1.3 Working scenarios

Even though the techniques proposed in this thesis are generic enough for most current information security architectures, they fit especially well into eHealth scenarios. In these instances, signals proposed for use in user authentications (PPG and ECG) are typically collected during patient follow-up, making them generally available for most users.

Furthermore, healthcare is increasingly provided by specialists from several organizations— as is the case with patients who look for a second opinion on their diagnosis away from their main healthcare institution. The interactive user-to-user delegation would be very interesting in this case, since it enables patients to grant specialists access to their medical tests, thus avoiding duplicate clinical procedures and improving patient care efficiency, and, as a consequence, a smoother patient lifecycle, better results and lower costs.

Finally, all the cryptographic machinery needed to improve data privacy levels is related to considerable resource overheads (CPU, RAM, bandwidth, etc.) that might not be feasible in all scenarios. However, they would surely be welcome in healthcare given how sensitive the information involved is.

1.2 Thesis approach, hypothesis and objectives

The general approach of this thesis is to research and make contributions to the field of usable security and applied cryptography, with a specific focus on information security and privacy. In recent decades, many information systems and services have moved online. However, the advantages related to the connectivity characterizing this world-wide network are frequently accompanied by several risks and threats, such as data breaches. Therefore, research in this area is highly relevant to prevent unauthorized information disclosures, benefiting not only information owners but also system owners, given the legal implications related to the misuse of third-party information, especially in scenarios involving sensitive data.

The main hypothesis of this thesis is that by making contributions in information security architecture areas—specifically in user authentication, authorization and data-in-transit security—we can improve the overall security and privacy of user information. The following objectives, subdivided into the main topics of the thesis (i.e., PPG biometrics, API security, and Border protection) are used to sustain this hypothesis. **PPG biometrics:**

- To conduct reviews on the state of the art in general aspects of usable user authentication mechanisms. Specifically, to review literature on emerging biometric authentication approaches.

- To design and evaluate biometric authentication systems that use PPG for user authentication in both the short- and the long-term.

API security:

- To conduct reviews of the state of the art in general aspects of information access control systems, including topics such as access delegation management and policy definition and evaluation. Specifically, to review literature on web APIs security.

- To design and evaluate authentication and authorization protocols that improve web APIs access control by providing secure seamless client re-authorization, secure access to services provided over messaging platforms, and user-to-user delegation.

Border protection:

- To conduct reviews of the state of the art in general aspects of secure network functions (SNF). Specifically, to review the literature on systems allowing the firewalling functionality to be applied on encrypted traffic without compromising its privacy.

- To design and evaluate new approaches to efficiently enable SNFs based on secure multiparty computation (SMC).

1.3 Research context

This thesis, entitled “Design and evaluation of novel authentication, authorization and border protection mechanisms for modern Information Security Architectures” and supervised by Álvaro Alesanco Iglesias and José García Moros, has been completed within the framework of the Communication Networks and Information Technologies (CeNIT) Group of the Aragón Institute of Engineering Research (I3A), within the Mobile Network Information and Communication Technologies doctoral program of the University of Zaragoza, Spain.

The research context was mostly wider projects on the secure and private management of sensitive information, such as: “Microservices and ontologies in building an architecture for secure and private information management in the personalized follow-up of psoriasis patients” (TIN2016-76770-R) and a doctoral grant to the author (FPU15/04841).

1.4 Thesis outline

The remainder of this Thesis is structured as follows:

•Chapter 2 explores the feasibility of using the PPG for biometrical purposes. It firstly presents the four PPG databases that would be used to obtain the results for both the short- and the long-term. It also contributes several methodologies, whose performances are compared with each other and with others in the state of the art using the previously presented databases.

•Chapter 3 describes and evaluates extensions to enhance the OAuth 2.0 framework in three ways. These extensions allow the authorization server to seamlessly verify that users are present on each access attempt, they enable access delegation to a third-party application when this application is being accessed through a messaging platform and they allow to interactively request a resource owner to approve a transaction when users request access to a protected resource they have not been previously authorized to access.

•Chapter 4 introduces the security vs. privacy dilemma at domain boundaries and proposes a solution based on SMC to enable SNF, shortcutting this limitation. The proposed solution relies on multi-circuit function representation to balance the delay introduced by the function evaluation and the total throughput achievable depending on the network state.

•Chapter 5 presents the research objectives achieved, contributions and accomplished results of this thesis, and future lines of research.

Chapter 2: PPG Biometrics

Authentication is the first step to gain access to an information system and it is aimed at determining whether an entity is who it claims to be. The proliferation of smartwatches and wristbands, which already acquire the PPG for health and fitness purposes, has drawn attention to the potential of this signal for seamless user authentication. This chapter analyzes the feasibility of using the PPG for biometric purposes in the long term.

2.1 Background

During the past few years, biometric authentication [27], [28], which uses measurable and distinctive features from one person to perform robust authentication, has been gaining increasing attention due to its intrinsic characteristics: it cannot be forgotten (like authentication based on something we know), it cannot be stolen (like authentication based on something we possess), and it is hard to reproduce, modify, or hide, offering a great non-repudiation capability [29]–[31]. Biometric authentication can be run into two different operational modes: authentication (verification) and identification [3], [32]. In the authentication mode users claim an identity and show their credentials. The system checks whether those credentials belong to the claimed identity. On the other hand, in the identification mode no identity is claimed, users only show their credentials and the system decides whether they belongs to one of the previously enrolled users. In order to evaluate the performance of a biometric system, results are usually provided with the system working in authentication mode [33].

Unlike authentication approaches using something we know, or we have (e.g., passwords or tokens, respectively) where the result is Boolean, i.e. either there is a complete match or a complete failure, in biometric authentication the result is a confidence measure. Biometric authentication is essentially a pattern recognition problem. As such, it involves two main stages: enrollment, aimed at producing a template (also called a model) with the most representative characteristics from the user, and testing, where the user is contrasted against their model to perform the authentication by obtaining the confidence measure. This measure is compared to a threshold and the system validates (or not) the user identity. Hence, two error measurements

model the system behavior, the false match rate (FMR) and the false non-match rate (FNMR) [30], [34]. The FMR is the probability that the system incorrectly matches the input pattern to a non-matching template in the database (i.e., the probability of an intruder has been treated as genuine). On the other hand, the FNMR is the probability that the system not correctly matched the input pattern to a matching template in the database (i.e., the probability that a genuine user has been treated as an intruder). The system FMRs and FNMRs are not static points but values depending on the threshold selection. The equal error rate (EER), the value where FMR and FNMR are equal, has been traditionally used as the performance point for biometric authentication systems [33].

Biometric research has recently shifted its attention from using classical approaches e.g., fingerprints or iris patterns, to employing physiological signals like the electrocardiogram [33], [35]–[39] (ECG), electromyogram [40] (EMG) or the electroencephalogram [41], [42] (EEG) to perform user authentication. These signals intrinsically provide a proof of life (fingerprints and iris patterns do not) and are hard to fake. Nevertheless, they present two main handicaps that hamper their widespread adoption: recording is complex (e.g., multilead ECG acquisition requires the use of electrodes distributed along the body and for EEG recording, an electrode hat should be used) and the required devices can be expensive. On the contrary, the PPG [43] can be easily used with one low-cost sensor placed on the fingertip. The PPG is a biomedical signal that estimates volumetric blood flow changes in peripheral circulation by means of infrared light absorption. Figure 2.1 shows a PPG cycle along it most relevant characteristics.

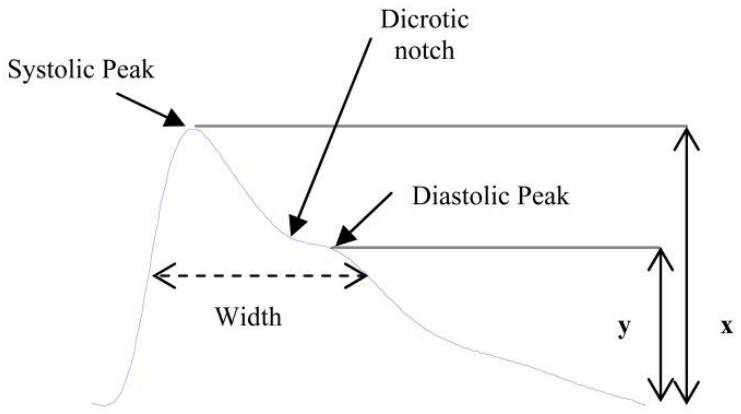

Figure 2.1 PPG cycle morphology and relevant characteristics

Some studies have already evaluated the use of the PPG for authentication purposes. Since biometric authentication can be considered a pattern-recognition problem, these studies differ in the feature extraction approach applied for enrollment, the metric used for testing and the time interval between signals used for these two phases. In [44], the authors propose using a linear discriminant analysis (LDA) to generate user templates and k-nearest neighbors (k-NN) to classify them. They test their method on two PPG datasets (OpenSignal and BioSec) and obtain disparate results. The time interval between the signals used for enrollment and testing is not stated in the paper. In [45], user templates comprise a set of PPG cycles, and the similarity between templates is measured in terms of the cross-correlation between cycles. Just one PPG dataset is used but several tests are run on this dataset to check the system's performance when the length of the signal chunk used for enrollment and testing varies. PPG signals used for enrollment and testing were recorded in the same session. In [46], the authors propose a feature extraction based on discrete wavelet transform (DWT) accompanied by the Kolmogorov–Smirnov test (KS-test) and the kernel principal component analysis (KPCA) to reduce template dimensionality. They suggest using the k-NN for template classification in the testing stage and state that non-fiducial feature extraction provides better results than fiducial feature extraction; they present their results using the photoplethysmography respiratory rate benchmark (PRRB) dataset. Finally, in [47], the authors advocate using the continuous wavelet transform (CWT) accompanied by the direct linear discriminant analysis (DLDA) in the feature extraction stage and the Pearson distance in the testing one. An extensive performance evaluation of PPG biometrics against single-session data, different emotions, physical exercise, and time lapse was performed.

A careful inspection of all these papers results seems to indicate that PPG signals could present the potential to be used in authentication, but results are disparate depending on the databases, ranging from EER values of 0.5% up to 25%. Although this variance is problematic for deciding whether the PPG is a good candidate for authentication or not, the main handicap of these studies is that they do not face the natural evolution of the PPG signal with time since most of them (apart from [47]) use signals from the same recording session for both enrollment and testing. The baseline and the amplitude of the PPG signal present fluctuations in low and high frequencies, which are mediated by the autonomic (sympathetic and parasympathetic) nervous systems [48]. Sympathetic nervous system arousal can be triggered, for example, by anxiety,

exercise, or health status, provoking fluctuations in the PPG waveform [49]. Hence, there is still a clear need for an in-depth study of the evolution of EER value depending on the time distance between the model generation (enrollment) and the PPG acquisition for user authentication (testing). If the EER value increases with time over a certain threshold, the system will become inoperative.

2.2 Methodology

The operation of a typical biometric authentication system involves two stages: enrollment and testing. Enrollment is devoted to creating a database of templates that characterizes the users (one template per user). Testing is the stage where a subject that wants to be authenticated into the system is parametrized (i.e., the system generates a template with its characteristics) and compared with the known-users' templates. Enrollment and testing are divided in turn into three steps. Since they share the same initial goal i.e., create a template of the user, they share the first two stages: preprocessing, aimed to adapt the signal to reduce quality problems generally related with the acquisition, and feature extraction, looking for the most representative characteristics of the signal (features subset) to create a template of the subject. The third step for enrollment is database generation, where templates of the users are stored. For testing, the third step is matching (verification), where users are authenticated or not depending on the similarity of their templates to the templates in the database. Figure 2.2 illustrates all these stages and steps.

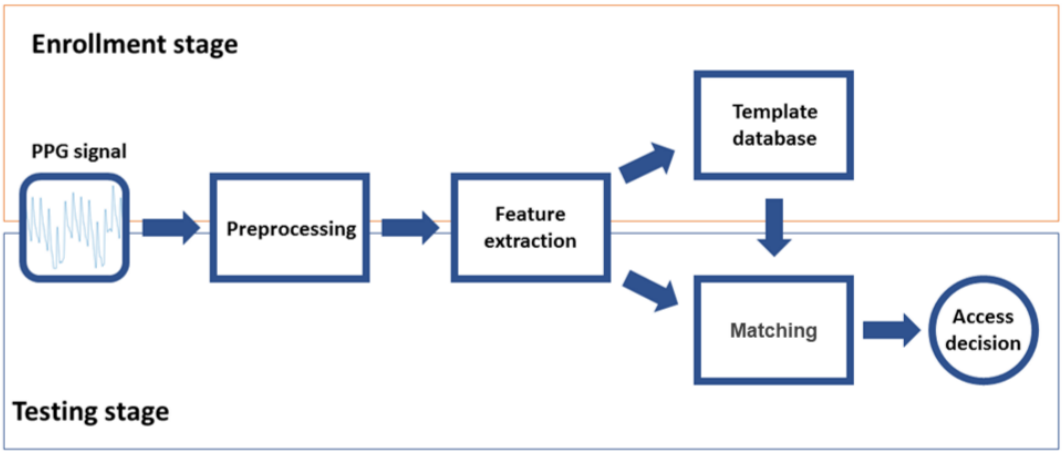

Figure 2.2 Biometric system workflow. The enrollment stage includes preprocessing, feature extraction and template storage. Testing stage includes preprocessing, feature extraction and matching.

2.2.1 Preprocessing

PPG signal quality depends not only on physiological characteristics (e.g., skin properties) but also on external conditions such as power line interferences or motion artifacts. Signal preprocessing allows the minimization of the negative effects of these noisy components on the system performance as well as the extraction of the PPG cycles, which are the base materials to work with in order to create a user template. In this work, preprocessing is divided into three tasks.

Filtering

High-pass filtering has been applied to remove the low-frequency baseline that is present in PPG signals due to sensor motion during acquisition. A Butterworth filter is used due to its maximally flat response in the passband. In our analysis we used a third-order filter with a cutoff frequency of 0.5 Hz. Forward-backward implementation has been used to avoid phase distortion [50].

PPG Cycle Detection

The modelling unit is the PPG cycle. This step aims to delimit and extract these cycles from the continuous PPG signal. PPG cycle boundaries are placed where a local minimum is followed by a local maximum and the amplitude difference between those inflection points is greater than the threshold, $th = \frac{A_5 - A_{95}}{2}$, where A is the signal amplitude vector sorted in decreasing order, and A_x is the value at the x percentile of A. This threshold ensures that a dicrotic notch is not confused with a delimitation point, and that the real boundaries can be detected despite the noise and the small variations in the amplitude of each pulse.

Cycle Normalization and Alignment

Cycles are normalized by dividing the amplitude of the cycle by the amplitude of the systolic peak. The PPG cycles are also scaled on the temporal axis by using cubic spline interpolation, so that each cycle is formed by 128 samples, allowing the system to work independent of the users heart rate at the time of acquisition and the sample rate of the device utilized for the acquisition. Finally, PPG cycles are aligned by allocating its systolic peaks at the same point.

2.2.2 Feature Extraction

After the preprocessing step, when a predefined number of PPG cycles (N cycles) are available to work with, a feature extraction process is carried out, producing a template. Note that other approaches in the literature do not use a predefined number of cycles to produce the template but they use the cycles found in a predefined PPG block length instead (measured in seconds) [44], [45], which depends on the subject's heart rate. In this work a constant number of cycles is preferred since it homogenizes the calculation of the templates. In the enrollment stage this template is stored in a database of authorized users. In the testing stage, this template feeds the classifier where it is compared with the stored template of the user requiring access to proceed with its verification. Several approaches have been analyzed as feature extractors.

Cycles Average

This approach works on the time domain. The model of each user is defined as the mean of their N aligned cycles (see Equation (2.1)). Hence, each template in this method is composed by a vector of 128 values to be stored (float precision is enough).

$$t = \frac{1}{N}\sum_{n=1}^{N} C_n \tag{2.1}$$

KLT Average

This approach is based on the Karhunen-Loève transform (KLT) [51]. The KL projection base is calculated using the cycles from all users. The model of each user is the projection of their mean cycles over the transformation matrix (see Equation (2.2)). Hence, each template is a vector of 128 values plus the transformation matrix (128 x 128 values).

$$t = KL\left[\frac{1}{N}\sum_{n=1}^{N} C_n\right] \tag{2.2}$$

Multi-Cycles

The template of each user is composed by the N cycles in the temporal domain (see Equation (2.3)). In this case, the storage would take 128 x N values, where N is the number of cycles considered.

$$t = \{C_n\} \qquad n = 1..N \tag{2.3}$$

KLT multi-cycles

This method is also based on the KL domain, but in this case the model is composed of the projection of their cycles in the transformed domain (see Equation (2.4)). The KL projection base is calculated using the cycles from all users. The storage would take 128 x N values, where N is the number of cycles considered plus the transformation matrix (128 x 128 values).

$$t = \{KL[C_n]\} \qquad n = 1..N \tag{2.4}$$

2.2.3 Matching

In order to undertake the matching, two templates are compared: the authorized user template generated and stored during the enrollment stage, and the template coming from the feature extractor in the test stage. A matrix with the distances from each cycle in the testing template to each cycle in the enrollment template is calculated. The distance between templates is selected as the minimum value in the distance matrix (see Equation (2.5)).

$$d(T_e, T_t) = \min(\|c_t - c_e\|_x) \qquad \forall\, c_t \in T_t,\ \ c_e \in T_e \tag{2.5}$$

The distance between cycles ($\|c_t - c_e\|_x$) can be either the Manhattan (x = 1) or the Euclidean distance (x = 2). If distance, *d*, is lower than a threshold, *th*, the subject is classified as being the user they claim to be i.e., the subject is authenticated. Otherwise, the subject would not be recognized, and the authentication would fail. Note that for feature extraction methods based on means (feature extractors 1 and 2), the template contains only one cycle. On the contrary, for feature extraction methods based on independent cycles (feature extractors 3 and 4), templates are composed by N cycles.

2.3 Signal Databases

Assessing the feasibility of PPG-based authentication in the long term has to be grounded into representative and extensive PPG databases. Using public PPG databases, reproducibility is allowed. Thus, in this work four publicly available PPG databases presenting different characteristics have been used. The first is the PRRB [52]. The number of subjects (PPG signals) in the database is high, and they present good quality. The second dataset is a subset of the public "MIMIC II Waveform Database" [53]. This specific subset can be found at the ehealthz github site [54]. Two more datasets have been locally acquired by authors for this work using a Nonin WristOx2 pulse oximeter [55] and a Berry pulse oximeter [56], respectively. Both datasets contain signals from the same 24 subjects acquired on three different days. Signals are publicly available at ehealthz github site (a completed request form is needed to allow access). The two sets of data have been acquired under realistic conditions. We asked the users to sit down and then we started to record the PPG signals without a great period of relaxation. None of the subjects had health problems related to the circulatory or respiratory system. Table 2.1 summarizes the characteristics of these databases and Figure 2.3 shows an extract from one PPG signal of each database. Note that these are raw signals as they were acquired before the preprocessing step.

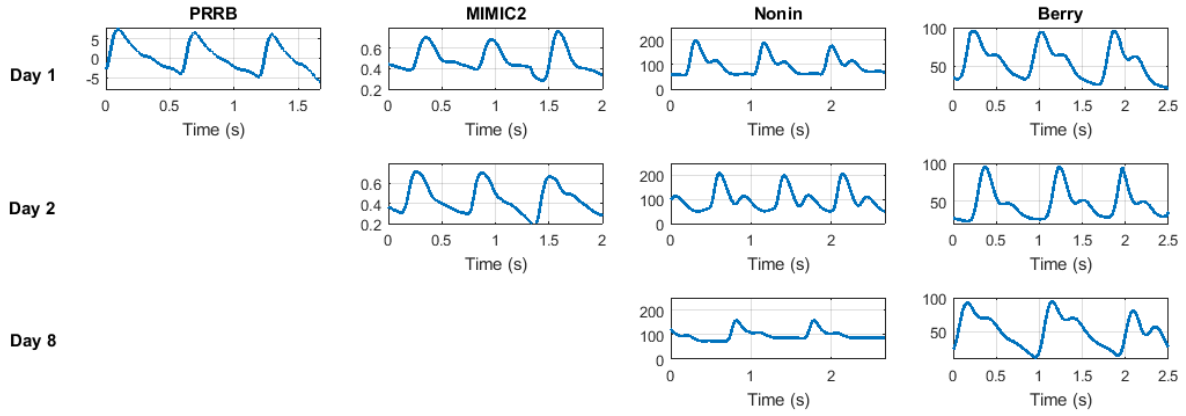

Figure 2.3 Representative photoplethysmogram (PPG) signals extracted from several databases. For each database, segments belong to the same subject in a different session (days 1, 2, and 8 when available).

Table 2.1 The relevant parameters of the databases. PRRB: Photoplethysmography Respiratory Rate Benchmark.

DDBB	Subjects	Fs (Hz)	Resolution (Bits)	Sessions	Length	Time interval (days)
PRRB	42	300	n.a.	1	8 m	0
MIMIC2	56	125	10	2	60 s	1
Nonin	24	75	8	3	60 s	1 & 7
Berry	24	100	7	3	60 s	1 & 7

2.4 Results

This section is divided in four subsections. First, in Section 2.4.1 we explain how the results have been calculated. Then, in order to go from the broad authentication methodology performance to the specific detailed findings in the long term smoothly, results are provided following a three-step approach. Firstly, in Section 2.4.2, EER values are used so as to select the best authentication method among all the possible combinations both in the short- and long-term scenarios. Secondly, in Section 2.4.3, a deeper study of the selected method is carried out in order to assess PPG authentication feasibility in the long term. Finally, in Section 2.4.4 we compare the results obtained using the method selected in the previous section with the state-of-the-art methodologies.

2.4.1 Evaluation Procedure

In order to test the authentication methods, FMR and FNMR curves are calculated as:

$$FMR(th) = \frac{\sum_{i=1}^{L}\sum_{j=1,j\neq i}^{L} sgn(d((T_e)_i,(T_t)_j) - th)}{L\cdot(L-1)} \quad \textbf{(2.6)}$$

$$FNMR(th) = \frac{\sum_{i=1}^{L} sgn(th - d((T_e)_i,(T_t)_i))}{L} \quad \textbf{(2.7)}$$

$$sgn(x) = \begin{cases} 1 & if\ x \leq 0 \\ 0 & if\ x > 0 \end{cases} \quad \textbf{(2.8)}$$

L is the number of subjects used for the results calculation and *th* is the acceptance threshold. The EER is the value where FMR = FNMR. Since signals in the datasets contain a number of cycles much higher than N, not only one but many different templates can be produced/generated for obtaining the FMR and FNMR results. If S_i is the signal used for enrollment or testing for subject i and contains a total of M cycles, different templates for that subject can be generated (see Equation (2.9)).

$$T_i(n) = S_i(n, n+N) \quad \forall\, n < M - N \tag{2.9}$$

Thus, in order to obtain more comprehensive results, several templates (both for enrollment and testing) can be used to test the authentication methods so as to take into account the complete M cycles of the signal, not only N cycles of one template. Hence, in this study, the final FMR and FNMR curves used to present the results are a combination up to K templates derived from the same signal (K = 9 for all the databases, except for the Nonin and Berry databases when the time interval is 0, where K = 6 due to the signals duration), as expressed in Equations (2.10) and (2.11).

$$FMR(th) = \frac{\sum_{k=1}^{K} FMR_k(th)}{K} \tag{2.10}$$

$$FNMR(th) = \frac{\sum_{k=1}^{K} FNMR_k(th)}{K} \tag{2.11}$$

2.4.2 Method Selection

The EER was calculated for each authentication method (all possible combinations of feature extraction techniques and comparison metrics), using three different length values of N (10, 20 and 30 PPG cycles) for both the enrollment and testing templates. EER values have been obtained for all databases (except for PRRB which do not include long-term signals), and merged (averaged) in two sets, the short term, where the time interval between the enrollment and testing signal cycles is set to 0 s i.e., they are consecutive, and the long term, where the time interval between the enrollment and testing signal cycles is at least one day. The mean and the standard deviation (std) of the EER values obtained for all the databases that match in one of the sets (short term or long term) is shown in Table 2.2.

Table 2.2 Authentication methods and template length comparison for the short term and the long term. KLT: Karhunen-Loève transform.

Feature extractor	**N**	**EER (Mean/Std)**			
		Short Term		Long Term	
		Manhattan	Euclidean	Manhattan	Euclidean
Cycles average	10	12.6 / 1.8	12.5 / 0.7	26.3 / 2.1	26.7 / 1.8
	20	10.8 / 2.5	10.2 / 2.1	24.6 / 1.5	24.6 / 1.6
	30	8.2 / 3.0	7.9 / 2.4	24.3 / 1.8	24.0 / 1.7
KLT average	10	14.6 / 1.6	12.5 / 0.7	28.5 / 2.2	26.7 / 1.8
	20	11.0 / 1.5	10.2 / 2.1	25.4 / 1.4	24.6 / 1.6
	30	8.8 / 2.1	7.9 / 2.4	23.3 / 1.6	24.0 / 1.7
Multi-cycles	10	9.9 / 1.1	10.3 / 0.9	24.1 /2.0	24.7 / 2.5
	20	9.2 / 0.9	9.1 / 1.2	21.7 /1.3	22.4 / 2.2
	30	**6.9 / 1.0**	7.3 / 1.3	**20.8 / 1.6**	22.1 / 2.2
KLT multi-cycles	10	11.2 / 1.6	10.3 / 0.9	24.9 / 2.5	24.7 / 2.5
	20	8.9 / 1.8	9.1 / 1.2	21.6 / 2.6	22.4 / 2.2
	30	9.0 / 2.0	7.3 / 1.3	21.6 /1.7	22.1 / 2.2

Feature extraction methods that use multi-cycle templates obtain better performance than methods using cycle-average templates, especially in the long-term results (see Table 2.2). Analysing the average methods group, use of the time domain (cycles average) or the transform domain (KLT average) to calculate the EER value did not significantly affect the results. The type of distance used (Manhattan or Euclidean) also did not significantly affect the results. The same can be said for the multi-cycle methods group, as shown in Table 2.2. Regarding the N value in the EER results (see Table 2.2), an improvement in mean values can be observed in most cases when the number of cycles used for the enrollment and testing templates increases. As Figure 2.4 shows, the execution time of multi-cycle methods increases quadratically with the number of cycles in the templates (N). However, this time is not significant in comparison with the acquisition time, which really determines the time required to authenticate a user because in an operative real scenario, all PPG cycles (N) would be recorded before proceeding in the authentication process.

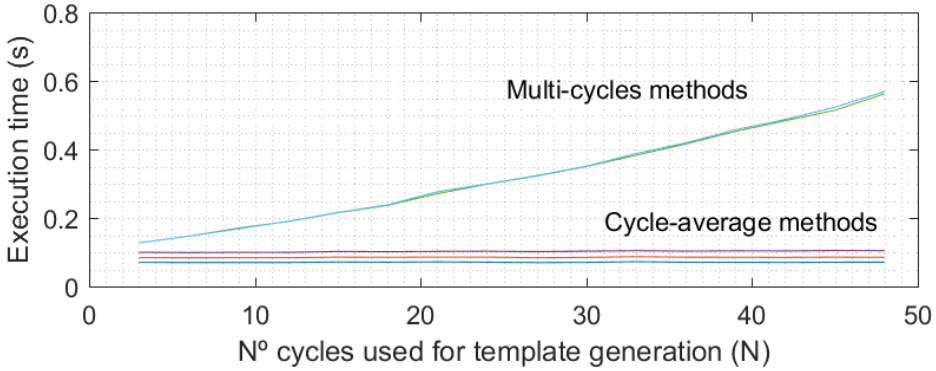

Figure 2.4 Execution times for multi-cycles methods and average-based methods. Times were obtained for the execution of the complete authentication method on a system with an Intel i7 6700 processor and 16 GB of RAM.

Hence, there is a trade-off between authentication speed (low N values) and final EER results (large N values). However, it is interesting to observe how EER values do not improve significantly when changing from 20 to 30 cycles (around 1% for the selected method), and as such, not much more improvement could be expected when increasing the number of cycles. Hence, for this study an N value of 30 cycles for both the enrollment and testing stages has been selected. However, in a real scenario the required number of cycles could be changed to reduce the time the user must remain with the pulse oximeter connected (with an accuracy cost), which is very important for guaranteeing the usability of the biometric system.

Finally, there is a clear performance degradation between the short- and long-term results regardless of the authentication method considered (see Table 2.2). Instead of expanding all the methods in order to go deep in the analysis of these results, only the best method will be further analyzed in the long-term detailed scenario. According to the previous discussion, the multi-cycles-based method is the best choice. Among them, the multi-cycles approach using the Manhattan distance is selected as it offers the best results.

2.4.3 FMR and FNMR Time Stability

To assess the potential of using PPG signals as biometric authentication in the long term, the stability of the FMR and FNMR curves as the time increases between the acquisition of signals used for enrollment and the acquisition of signals used for testing has to be deeply evaluated. Time stability implies that the values of FMR and FNMR do not vary for a selected threshold as the time passes i.e., FMR and FNMR curves converge to static values. Otherwise, a selected

threshold (working point) for the authentication system would produce different performance results with time. Note that PPG cycles could slightly change over time, so the stability is not guaranteed a priori (this is not the case of e.g., the fingerprint since it is stable over time).

The general results of the proposed method when run over the different datasets organized for the increasing time interval between modelling and testing stages are shown in Table 2.3. First, we can see the EER value obtained using the PRRB database is significantly lower than for the other three databases. This is due to the fact that the PRRB database was recorded during elective surgery and routine anaesthesia, which involve very controlled conditions and professional equipment, obtaining high signal quality. For the other three databases the recording conditions were not so controlled and are closer to real biometric applications where the subject will not spend 5 minutes at rest every time he wants to be authenticated. On the other hand, time interval is a key factor for system performance. EER values dramatically increase from a time interval of 0 to 1 day. Nevertheless, EER results obtained from 1-day and 7-day time intervals seem to remain stable, indicating a stable behavior of the EER value in the long term (slight variations on specific EER values could be observed depending on the dataset and the quality of its signals, but it is clear that the performance is degraded in a short period of time and remains stable after that). On the other hand, it is interesting to observe that results are coherent among all the four databases, yielding similar EER results for time interval values (around 7% for 0-day time intervals and 20% in the around 1-day and 7-day time intervals).

Table 2.3 EER results

Dataset	Time Interval	EER value
PRRB	0	1.0
Nonin	0	6.6
Berry	0	6.0
MIMIC2	0	8.0
Nonin	1 d	19.8
Berry	1 d	20.5
MIMIC2	1 d	21.5
Nonin	7 d	23.2
Berry	7 d	19.1

FMR and FNMR curves obtained for all datasets are shown in Figure 2.5. The long-term stability of these curves has been analyzed using the Nonin (Figure 2.5c) and Berry (Figure 2.5d) datasets as they are the only ones available with signals from at least three different days. It can be seen that FMR is very stable for each time interval value (0, 1, and 7 days), indicating that the difference between a user PPG model and the rest of PPG models for other users is time invariant. On the other hand, FNMR curves show a converging behavior as the time interval increases, stabilizing for a 1-day time interval value (values for 1 day and 7 days are close).

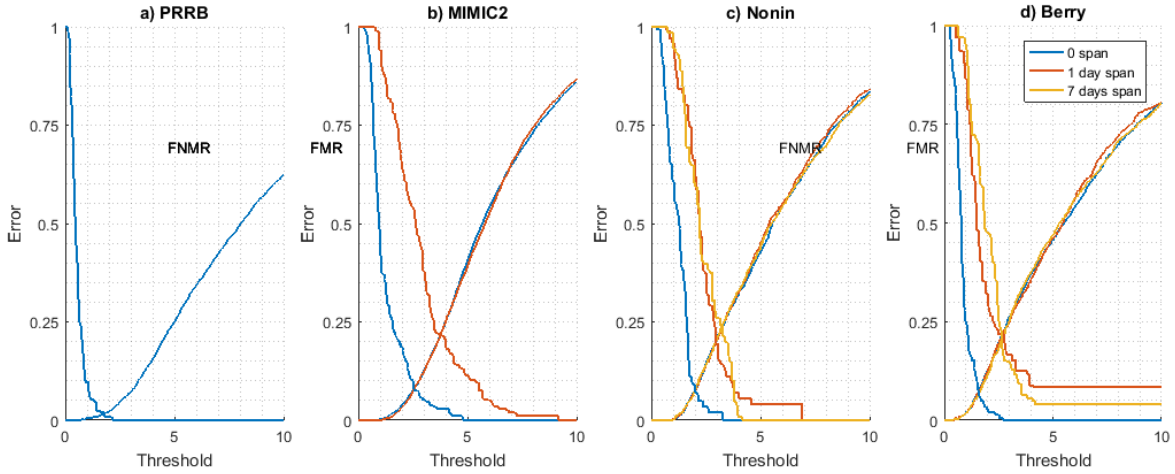

Figure 2.5 False match rate (FMR) and false non-match rate (FNMR) curves for all databases with time intervals of 0, 1, and 7 days when available.

This result seems to indicate that the evolution of the PPG signal (and thus the user signal model used for authentication) is not produced in a long time interval but it is a situation produced in a relatively short period of time. All in all, results seem to prove the time stability of FMR and FNMR curves in the long term. This is an essential condition since the threshold (working point of the system) is a pre-configured parameter of the system, so if these curves vary with time, the system will be inconsistent and useless for biometric applications.

2.4.4 Results Comparison

Results provided by the selected methodology (the multi-cycles model using the Manhattan distance) [57] has been compared with methods referenced in Section 2.1. On the one hand, performance results have been compared when using the PRRB database, which has been used by the authors of [46], [47] to present their results and is publicly available. Our method reports

an EER value of 1.0% running on authentication mode (see Table 2.3). In [46], [47], EER values close to 1% are obtained in similar conditions using the PRRB database for both (see Table 2.4). To compare the long-term results using the same databases, the methodology presented in [47] has been implemented. In this paper, authors present a methodology that obtains good results (EER value of 5.88%) in the long term using an in-home PPG database (BioSec). To validate our implementation of their method, results were obtained for PRRB database and compared with those reported in their study with the same database. For our implementation of their method, an EER value of 0.87% using the first 45 s for training stage and the next 30 s for the testing stage is obtained. In [47], the EER value reported is 0.85% for similar conditions. As can be seen, these results are similar, which seems to indicate that our implementation is accurate. On the other hand, results obtained with our implementation of the method in [47] when tested with the Berry, Nonim, and MIMIC2 databases are poorer than those obtained with our method. More precisely, EER values of 13.23% for the short term and 25.68% were obtained for the long term using method in [47], and values of 6.9% for the short term and 20.8% for the long term were obtained using our method. These results have been calculated in the same conditions explained in Section 2.4.2 (basically averaging the results for different databases in long-term and short-term scenarios).

Table 2.4 Related works analysis. LDA: linear discriminant analysis; TCS: temporal cycles set; KS-test: Kolmogorov–Smirnov test; KPCA: kernel principal component analysis; DLDA: direct LDA; k-NN: k-nearest neighbours; CC: cross-correlation; PD: Pearson's distance; EER: equal error rate.

Work	Dataset	Subjects	Enrollment stage		Testing stage		Time interval	EER
			Method	Length	Method	Length		
[44]	OpenSignal	14	LDA	50 %	k-NN	50 %	0 s	0.5%
	BioSec	15	LDA	50 %	k-NN	50 %	0 s	25%
[45]	Dataset1	44	TCS	20 s	CC	20 s	0 s	10.1%
	Dataset1	44	TCS	30 s	CC	30 s	0 s	8.3%
	Dataset1	44	TCS	40 s	CC	40 s	0 s	5.3%
[46]	PRRB	42	Wavelet + KS-test +KPCA	-	k-NN	-	0 s	1.31%
[47]	PRRB	42	Wavelet + DLDA	45 s	PD	435 s	0 s	0.46%
	BioSec	34	Wavelet + DLDA	45 s	PD	135 s	0 s	0.86%
	BioSec	34	Wavelet + DLDA	45 s	PD	135 s	14 days	5.88%

These results seem to indicate that long-term results are highly influenced by the nature of the database. For the Biosec PPG database, EER values are low (around 5%) and for the Berry, Nonim, and MIMIC2 databases the EER values are above 20%. Thus, the viability of the PPG as a feasible biometric trait may depend on the quality of the device used for the signal recording, the environment conditions during the acquisition, and even the concrete population using the system.

2.5 Conclusion

In this chapter, a comprehensive study on the viability of using the PPG as a reliable user authentication approach has been carried out. Different alternatives of authentication methods have been tested on four PPG databases. Despite of the promising results obtained in previous works, this study shows how for the same methodology the EER value may increase depending on the recording conditions of the PPG signals and the population from which the PPG is recorded (from 1% up to 8%). Moreover, although it presents stability in the FMR and FNMR curves (threshold selection coherence), EER values seem to rise quickly (up to 23.2%) when the time interval between the enrollment and testing stages increases. This performance degradation may hamper the use of PPG as a reliable biometric alternative in certain scenarios and prior system testing may be needed in the specific scenario to determine whether the PPG can be used to authenticate subjects or not.

Chapter 3: API security

After successful authentication, subsequent authorization is required to determine whether an access attempt should be permitted. OAuth 2.0, which is presented in Section 3.1, is currently the most widespread solution to deal with authorization tasks in attempts to access information or services exposed through Web-APIs. In this chapter we propose some extensions to the OAuth framework to improve it in several ways. Section 3.2 describes an extension to provide seamless client re-authorization using signals recorded by users' wearable devices. Section 3.3 proposes an OAuth grant type that improves both security and usability for services provided through messaging platforms. Finally, Section 3.4 extends the OAuth framework to support user-to-user authorization with minimal modifications.

3.1 Background

Nowadays, users' information is spread across multiple service providers and applications from different companies requires access to this information to provide the user with some enhanced functionalities. However, information sharing must always have the user's approval, as he is the real owner of the information. OAuth 2.0 [58], which is a standard framework for authorization, enables this kind of access delegation to third-party applications. It defines a base scenario composed by four actors: the resource owner, the resource server, the client and the authorization server. The resource owner, who acts as delegator, is an entity capable of granting access to a protected resource that is stored in the resource server. The client, who acts as delegate, is an application that requires access to the protected resource to perform any action on behalf the end-user and with his consent. The authorization server arbitrates the access to the protected resource by means of two endpoints: the authorize endpoint and the token endpoint.

The OAuth 2.0 framework defines different grant types (protocols) that can be used on different scenarios to provide different security guaranties. The Authorization Code grant type (and its extensions), which has been the most widely used grant type for the last decade, runs as follows (see Figure 3.1); when the client requires access (on behalf of the end-user) to the protected

resource, it redirects the user to the authorize endpoint including the authorization request parameters as URL query parameters (A), so that the authorization server would directly interact with the end-user. At this point, the authorization server would authenticate the user and determine whether he has the required rights to access the requested resource. If the end-user has the required access rights (i.e. he is the resource owner), he would be asked to consent the client application to access the requested resource on his behalf. On user's approval, the authorization server would redirect the user back to the client including the authorization response parameters as URL query parameters (B). The client would exchange the code obtained for an access token at the token endpoint of the authorization server, sending the code in the token request (C) and receiving the access token in the body of the token response (D). Requests to the token endpoint must include the client credentials whenever client authentication would be required. Finally, the client can use the obtained access token to access the protected resource at the resource server.

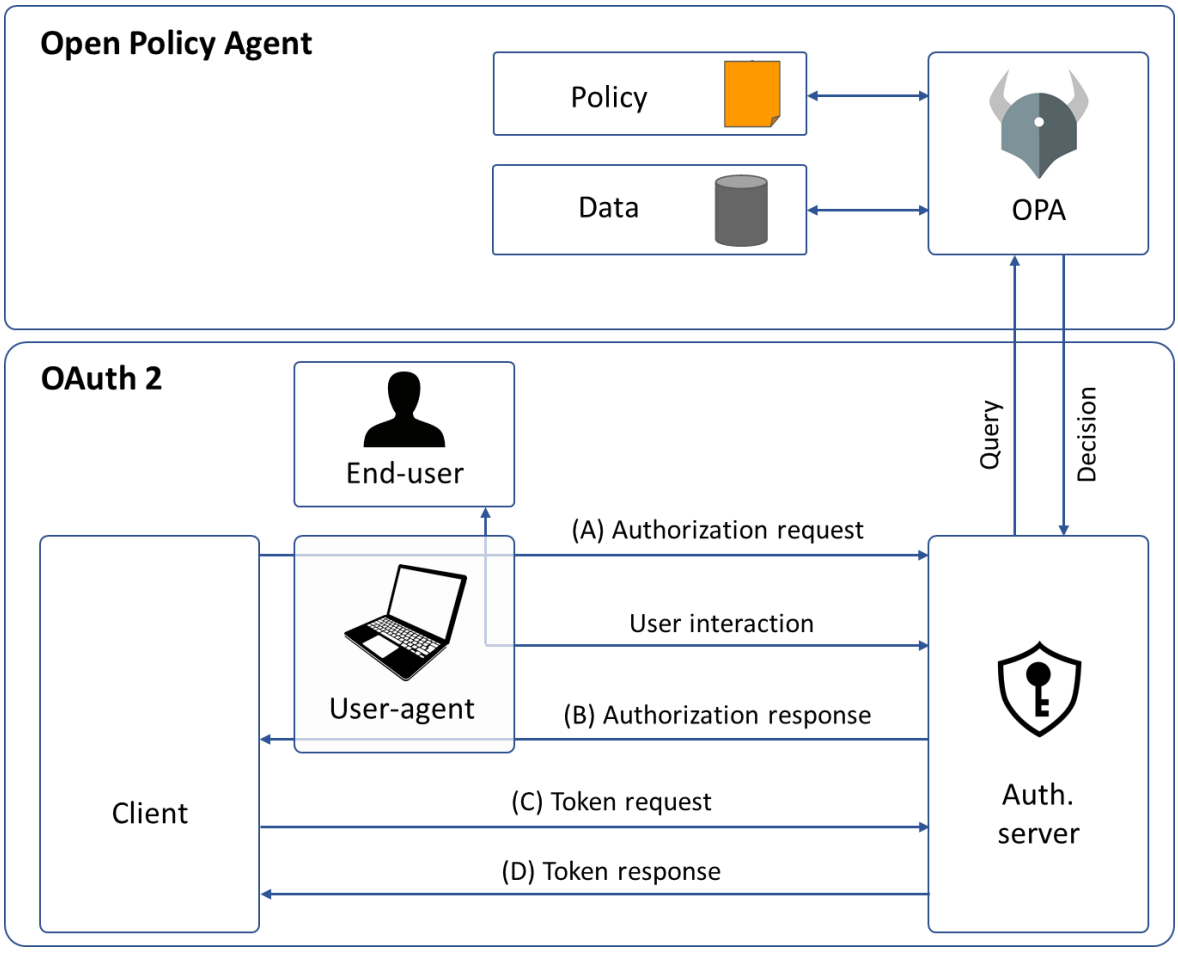

Figure 3.1 OAuth Authorization grant type and Open Policy Agent interaction

How access rights are assigned and evaluated is out of the scope of the OAuth 2.0 specification. However, it is generally done by writing access policies, that would be later evaluated by a policy engine to enforce the rules that would govern the system. Several access control schemes, such the Discretionary Access Control (DAC), the Mandatory Access Control (MAC), the Role-Based Access Control (RBAC) or the Attribute-Based Access Control (ABAC), have been widely used in the last decade in a huge range of applications [59]. ABAC schemes, which allow to perform a fine-grained access control, are especially appropriated to reflect the complex relationships that exist in the real world. In this scheme policies are defined in terms of attributes, using an expressive language to define conditions related with these attributes. Attributes are sets of labels or properties that describe the actors involved in the authorization process (i.e. the user requesting access, the resource being requested, the environment, etc.).

Open Policy Agent (OPA) [60] is a policy engine that has lately gained strength over existent approaches, like XACML [61], for its simplicity and flexibility, becoming the preferred option for many research and industry applications. To make access decisions, OPA relies on two main elements, policies and data. Data is composed by the attributes that describe the actors involved in the authorization process encoded in JSON format, while policies describe the relation and restrictions on these attributes. Policies in OPA are defined using the Rego language [62] and are organized in modules. Each policy module is identified by a unique value, which is only used with management purposes, and contains a set of rules. These rules are used to conditionally assign values to variables, which would be returned as output of the policy evaluation. Each policy module belongs to a package and packages are organized hierarchically and determines how policies are evaluated. When an evaluation request referencing a given package is received, all rules in the referenced package and all its sub packages are evaluated. The result of a policy evaluation in OPA is not limited to simple yes/no or allow/deny answers. Policies can generate arbitrary structured data as result of the evaluation of the rules.

Figure 3.1 shows the OPA architecture on its top. Interaction with OPA is performed through its Restful API that can be used to manage the policy modules lifecycle, to provide the system with additional data required for the policy evaluation and to start a new policy evaluation when required. Access to this API can be secured by means of access tokens, so that the use of OAuth is also recommended.

On its first days, OAuth 2.0 was used by companies such as Google or Facebook to provide its social login service. In this scenario, the protected resource requested by the third-party application was an assertion about the end-user identity issued by the trusted company (Google or Facebook). This way, application developers have not to care about the credential's management. However, OAuth was not designed to provide federated identity but to deal with authorization tasks, so that, as this use became more popular a new specification was created to separate these usages. This new specification was OpenID Connect [63], which is a federated identity protocol build on top of the OAuth 2.0. It is used by client applications to obtain some information about the end-user, such as his identity, his age, or his postal address. In the scenario proposed by this new protocol the authorization server and the resource server defined in the OAuth standard are substituted by the identity provider. This new actor accomplishes both functions, hold the information of interest about the user and authenticate him and obtain his consent to share his information with the client. When the client application needs to obtain the information about the user, it redirects him to the authorize endpoint of the Identity Provider, as done in normal OAuth 2.0 interactions. In this case, the "openid" value must be included in the scope field of the authorization request. After a successful user authentication, the identity provider sends an authorization code to the client application that would be exchanged for an access token and and identity token at the token endpoint of the Identity Provider. The identity token is a JSON Web Token (JWT) [64] signed by the Identity Provider ascertaining the end-user identity. On the other hand, the access token is used to access the Userinfo endpoint of the Identity Provider to obtain further information about the user more than his identity.

3.2 SeamAuth: Seamless authorization

In the OAuth 2.0 context, access tokens represent the authorization of a specific application to access specific parts of a user's data. Thus, access token leakage may imply several security issues as they can lead to personal data leakage or, even worse, account takeover. The threat of leakage is covered in OAuth-related RFCs [65], [66] and this risk can be significantly reduced simply by following the standard. However, there are always factors beyond the reach of OAuth server and client developers that introduce new vectors for token leakage. For example, a bug in Chromium-based browsers led to the leakage via the Referer header and browser history [67].

Issuing tokens to last only for a certain time helps minimizing the potential impact of access token leakage [68]. Although most application developers prefer using non-expiring access tokens due to their simplicity, they are not recommended in any scenario involving any kind of sensitive information. Currently, the most widespread solution uses short-lived access tokens lasting anywhere from several hours to a couple weeks, and long-lived refresh tokens, which can be used to obtain a new access token after the previous one expires in a convenient and user-friendly way (i.e., this can occur behind the scenes, without the user's involvement, so that the process is seamless for the user). However, although this limits the risk of access token leakage, to a certain extent it only shifts the problem from one kind of token to the other. Refresh tokens are usually more protected than access tokens, but, in the end, they could also be leaked (especially from non-confidential clients). Finally, the most secure option for limiting the potential damage resulting from access token leakage consists in using short-lived access tokens without refresh tokens. When using this method, the client must make the user sign in again as soon as the access token expires, so that the authorization server knows the user is continually involved in re-authorizing it. This makes it impossible for applications to use access tokens on an ongoing basis without the user being in front of the screen. Therefore, this method would not be suitable for clients needing access to continually sync data, and, from the user's perspective, this is the option most likely to frustrate people, since it will look as if the user has to continually re-authorize the client.

In this section, we propose a novel approach involving two subprotocols: interactive pre-authorization, in which users interact with the authorization server to provide the client with a preauth token, and just-in-time authorization, in which fresh biometrics are jointly used with the previously obtained preauth token to seamlessly authorize the client providing it with a one-time access token. This approach is suitable for the same scenarios as configuration with short-lived access tokens without a refresh token, where high security is required and clients should not have offline access to user data. Unlike the previous approach requiring user interaction to re-authorize the client each time the access token expires, the proposed approach relies on biometric signals that can be gathered from users' wearable devices, such as the PPG or ECG, to seamlessly verify user presence.

3.2.1 Interactive pre-authorization

This subprotocol is executed the first time the user consents to the client application accessing their protected data and ends with the client obtaining a preauth token from the authorization server. All interactions between the user and the authorization server take place in this subprotocol, thus enabling the subsequent access attempts to be completely seamless for the user. These interactions include verifying user identity with a strong credential (i.e., password, certificates, FIDO2 [69]) and obtaining their consent to the client application accessing their protected resources in subsequent access attempts using the just-in-time authorization subprotocol (see Section 3.2.2). Since this step would be required only once, authentication could occur using any kind of strong credential without hampering user experience. If the authentication method used in the just-in-time authorization subprotocol requires interaction with an external device (i.e., Bluetooth or NFC) the pairing process would be performed at this time and permission would be granted to the authorization server to interact directly with the device through the browser (i.e., using the Web Bluetooth API).

This subprotocol has been implemented as an extension to the OAuth authorization code grant type (see Figure 3.2). Unlike traditional OAuth setups, this subprotocol's objective is not to provide the client with an access token giving full access to protected resources but rather to sign a contract allowing the client to access these protected resources any time the authorization server can verify that the resource owner is interacting with the client. This contract is embodied in a new type of token, called a preauth token, which the client obtains as a result of executing this subprotocol and provides as an input to the just-in-time authorization. Therefore, preauth tokens would not only bind the client to whom it was issued but also the user allowing it to be issued.

If many, rather than one, users share the device where the client application runs, this step would be performed each time a new user utilizes it for the first time and a preauth token would be stored in the device for each of them (directly when the client is a desktop or mobile application or using cookies when is a webpage). The main subprotocol messages and interactions are detailed below.

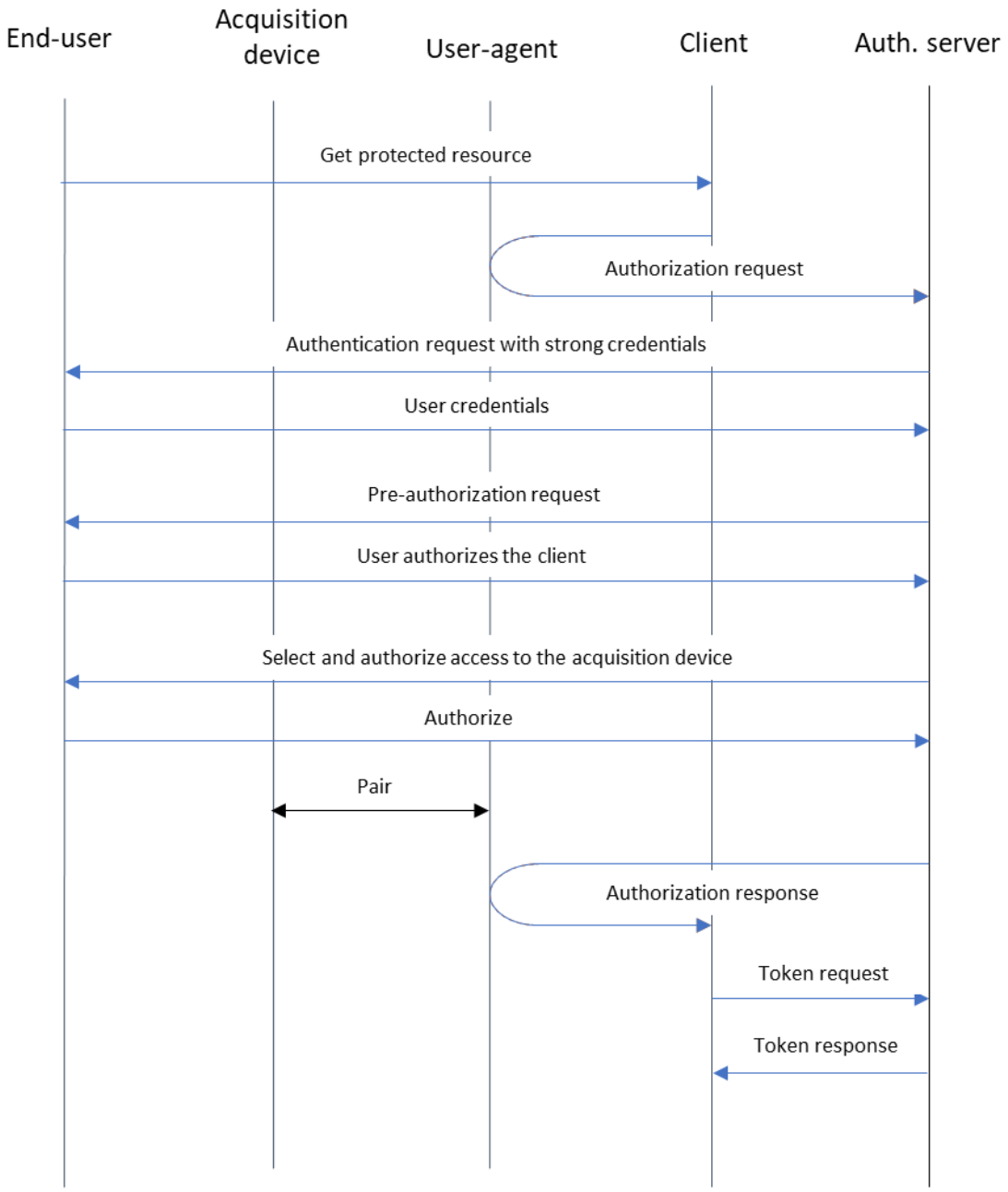

Figure 3.2 Interactive pre-authorization flow

Authorization request

To start the pre-authorization protocol, the client application redirects the user to the authorization server's authorize endpoint. This request would include the authorization request parameters (see Table 3.1) as URL query parameters.

Table 3.1 Interactive pre-authorization: authorization request parameters

Parameter	Compulsoriness	Description
scope	REQUIRED	Set of protected resources the client is requesting permission to access.
preauth_scope	REQUIRED	Set of protected resources the client is requesting permission to access in further access attempts using the proposed just-in-time authorization subprotocol.
jit_auth_method	REQUIRED	Supported authentication methods for the just-in-time authorization.
response_type	REQUIRED	Value MUST be set to "code".
client_id	REQUIRED	Unique client identifier issued by the authorization server at registration time.
redirect_uri	OPTIONAL	Endpoint at which the client would be waiting for the end-user to be redirected back after a once the interaction with the authorization server would be completed.
state	RECOMMENDED	An opaque value used by the client to maintain state between the request and callback.

The *response_type*, *client_id*, *redirect_uri* and *state* parameters are used as defined in the standard authorization code grant type [65]. The *scope* parameter also has the same meaning; however, it must contain the "seamless_auth" scope value. If the "seamless_auth" scope value is not present, the behavior is entirely unspecified. Other scope values may be present; however, scope values that are not understood by the implementation should be ignored. Finally, two new parameters, *preauth_scope* and *jit_auth_method*, have been defined. The *preauth_scope* is a list of space-delimited, case-sensitive strings indicating scopes the client requests permission to access in subsequent access attempts using the just-in-time authorization subprotocol. The *jit_auth_method* parameter includes authentication methods expected to be used to seamlessly authenticate the user during the just-in-time authorization subprotocol.

End-user interaction

The authorization server validates the authorization request and verifies that all required parameters are present and have valid values. If this is the case, the authorization server first validates users' identity using any kind of strong credential, such as a secure password or a FIDO device (see Figure 3.3). Once users' identity has been verified, they are asked for their consent to allow the client application to access their protected resources (encompassed by the scopes included in the *preauth_scope* parameter) using a method from *jit_auth_methods* to seamlessly verify their identity in subsequent access attempts (see Figure 3.4). Any special requirement associated with the requested authentication methods listed by *jit_auth_methods* must be addressed at this point. For example, if the PPG signal is used to perform user authentication in the just-in-time authorization subprotocol, the user must allow the authorization server to access the device that would be used for acquiring the PPG signal through the web browser (i.e., using the Web Bluetooth API[1] as shown in Figure 3.5).

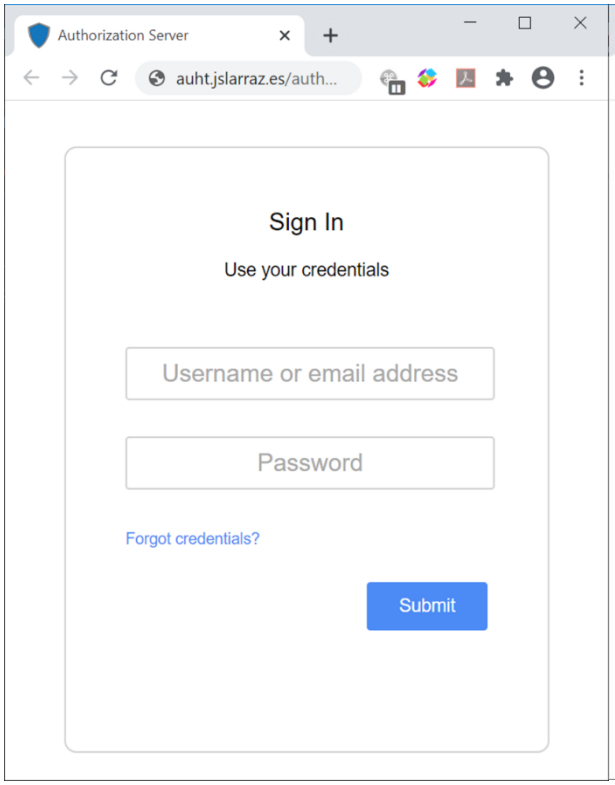

Figure 3.3 User is required to authenticate himself.

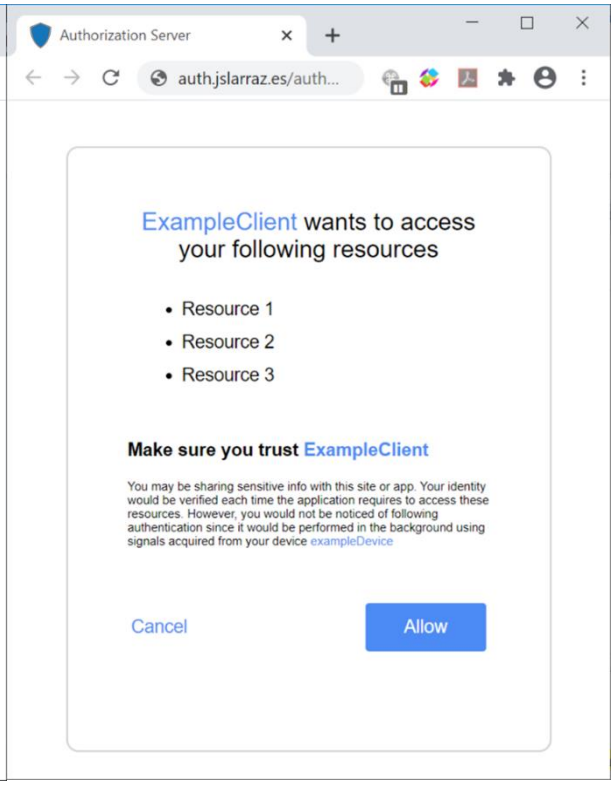

Figure 3.4 User consent is required to grant ExampeClient access to user resources.

[1] Note that the Web Bluetooth API is under active development and the path from [97] is required at time of writing.

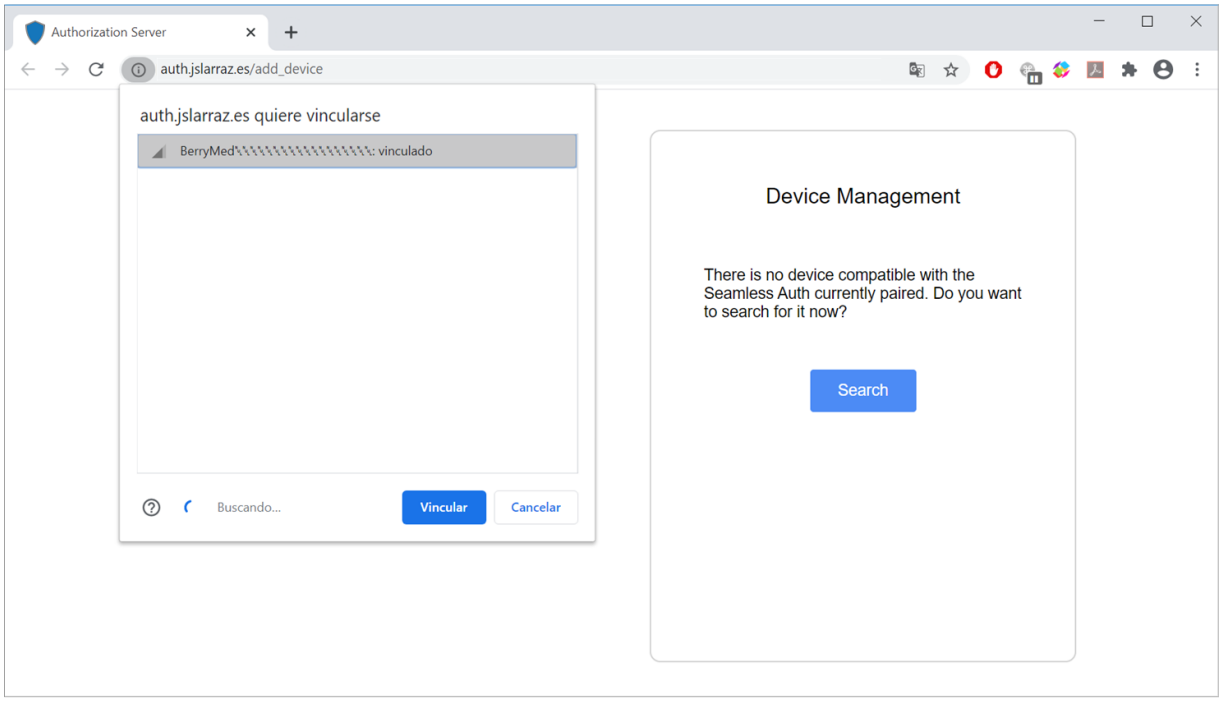

Figure 3.5 User is prompted to grant the authorization server access to the BerryMed Bluetooth device over the web browser.

Authorization response

When a decision is made, the authorization server redirects the user-agent to the provided client redirection URI using an HTTP redirection response, or another means available to it via the user-agent. If everything goes as expected (the user identity is correctly verified, the user is the owner of the resources for which access is being requested, and the user consents to the client application accessing their protected resources), the authorization server passes the authorization response parameters (*code* and *state* as defined in the standard authorization code grant type, see Table A.2) to the client, usually added as query parameters to the client redirection URI. Otherwise, the error response parameters (*error*, *error_description*, *error_uri* and *state* as defined in the standard authorization code grant type, see Table A.3) are included in the response instead.

Token request

After obtaining a valid authorization code, the client exchanges it for a preauth token in the authorization server's token endpoint. If the client type is confidential or the client was issued client credentials, the client must authenticate with the authorization. The token request is sent using an HTTP POST request with the token request parameters (see Table 3.2) encoded using the "application/x-www-form-urlencoded" format.

Table 3.2 Interactive pre-authorization: token request parameters

Parameter	Compulsoriness	Description
grant_type	REQUIRED	Value MUST be set to "authorization_code".
code	REQUIRED	The authorization code received from the authorization server.
redirect_uri	-	REQUIRED, if the “redirect_uri” parameter is included in the authorization request; in this case, their values MUST be identical.
client_id	-	REQUIRED, if the client is not authenticating with the Authorization server by other means.

Token response

When the authorization server receives a token request, it firstly verifies that all required parameters are present and have valid values. Next, it checks the client’s identity when required and confirms that the code included in the request is valid and has not yet expired. If everything goes as expected, the authorization server responds to the client and includes the token response parameters (see Table 3.3) in its response.

Table 3.3 Interactive pre-authorization: token response parameters

Parameter	Compulsoriness	Description
preauth_token	REQUIRED	The preauth token generated as a result of the interactive, pre-authorization subprotocol
token_type	REQUIRED	The type of the token issued.
expires_in	RECOMMENDED	The lifetime in seconds of the preauth token.
preauth_scope	-	OPTIONAL, if identical to the preauth_scope requested by the client; otherwise, REQUIRED.

The preauth_token includes the actual token issued by the authorization server; it embodies the contract signed by the users to grant the client access to their protected resources on their behalf as long as the just-in-time authorization subprotocol can be used to authenticate the users. The *token_type* indicates the kind of token; supporting “bearer” and “jwt” tokens is mandatory. The *expires_in* parameter indicates the token’s lifetime, and typically it would be valid until the user explicitly revokes it. The *preauth_scope* indicates which of the requested scopes the user has consented to. If the request failed client authentication or is invalid, the authorization server returns an error response including parameters defined in the access token error response (*error, error_description* and *error_uri* as defined in the standard authorization code grant type, see Table A.6).

3.2.2 Just-in-time authorization

The just-in-time authorization is performed each time the client application requests access to the protected resources and ends when the client application obtains a one-time access token that explicitly authorizes the operation. The just-in-time authorization subprotocol has been defined as a new OAuth grant type (see Figure 3.6) combining the previously obtained preauth tokens with the identifying properties of some signals recorded by the user's wearable device, such as the PPG [57] or ECG [70], to seamlessly obtain an access token authorizing access to the resource server. If everything goes as expected, the authorization server issues an access token authorizing the client application to access the protected resource. The main subprotocol messages are detailed below.

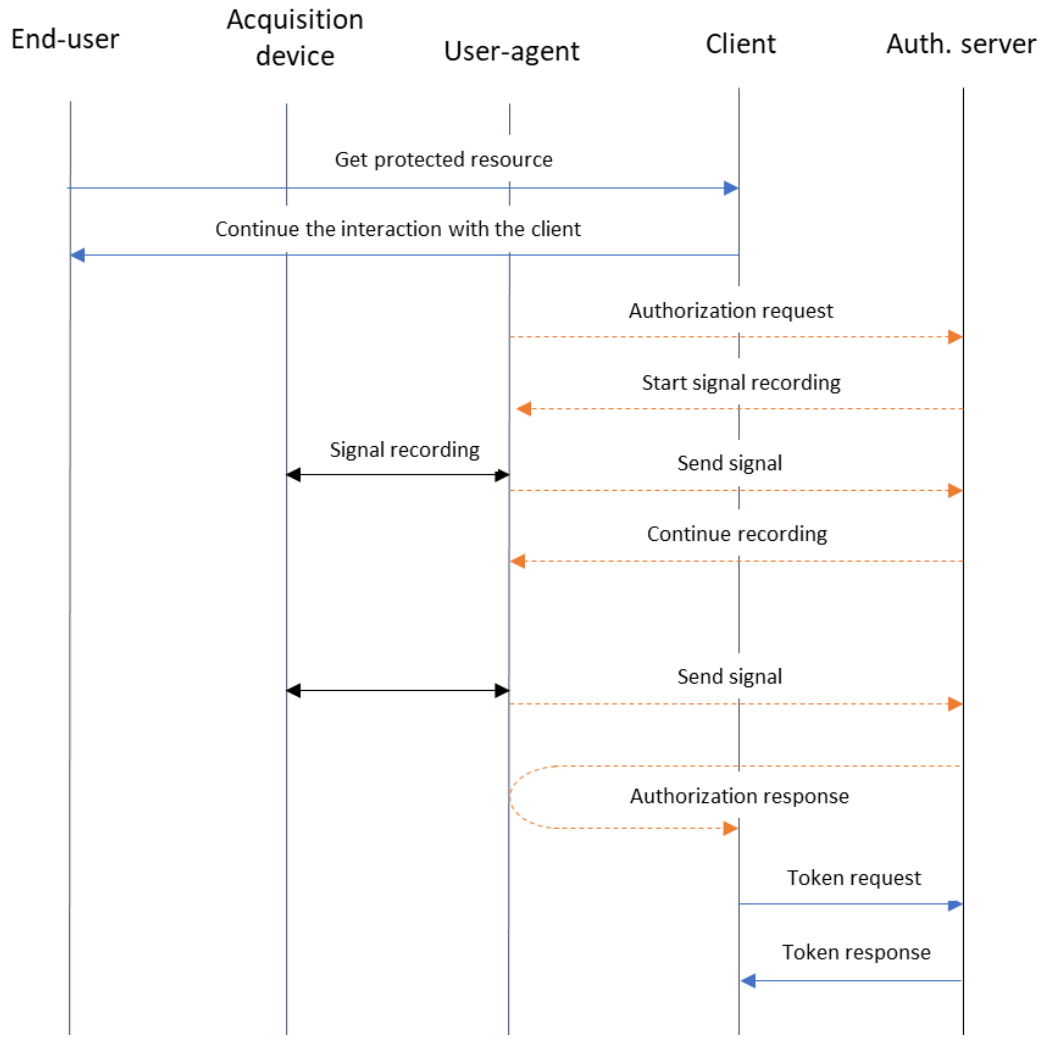

Figure 3.6 Just-in-time Authorization flow

Authorization request

The just-in-time authorization subprotocol begins with a user starting a new interaction with the client application using a device in which the interactive pre-authorization has previously been completed. The client application includes a hidden iframe in its response page, which makes the user-agent send the authorization request to a new authorization server's endpoint, the seamless_authorize endpoint. An iframe is a HTML element used to embed another document (such as a new web page) within the current HTML document. Dashed orange lines have been used in Figure 3.6 to represent communications realized through the iframe. This request would include the authorization request parameters (see Table 3.4) as URL query parameters.

Table 3.4 Just-in-time authorization: authorization request parameters.

Parameter	Compulsoriness	Description
preauth_tokens	REQUIRED	List of preauth tokens obtained from the interactive pre-authorization.
client_id	REQUIRED	A string uniquely identifying the client.
redirect_uri	OPTIONAL	A URI to redirect the user back after the authorization.
scope	OPTIONAL	A string describing the access rights requested by the client.
state	RECOMMENDED	An opaque value used by the client to maintain state between the request and callback.

The *client_id*, *redirect_uri*, *scope* and *state* parameters have the same meaning as defined in the standard authorization code grant type. One new parameter is included in the request, *preauth_tokens*, to enable seamless authorization. This parameter would include a list with all preauth tokens previously obtained by the interactive pre-authorization subprotocol.

Authorization response

The authorization server verifies that all required fields are present in the request and valid; if this is the case, the seamless authorization page is loaded in the iframe. It searches for a compatible device for which it already has obtained permissions to connect to and is capable of recording some biometric signals supported by the authorization server to perform user authentication. If the connection can be established with such a device, the biometric signal begins to be acquired and streamed to the authorization server. Once the authorization server has enough signal chunks to decide whether the signal belongs to a user for which a preauth token has been included in the authorization request or not, the authorization server redirects

the iframe to the client redirection URI. If the user can be successfully authenticated, the authorization response parameters (*code* and *state* as defined by the standard authentication code grant type, see Table A.2) are added as query parameters to the client redirection URI. Otherwise, the error response parameters (*error*, *error_description*, *error_uri* and *state*, as defined by the standard, see Table A.3) would be included in the response instead. A special error code, "no_device_reachable", has been defined to inform the client that the connection cannot be established with any compatible device. While the users are authenticated in the background (through the hidden iframe), they can continue interacting with the client application, keeping this entire authentication process completely seamless for them.

Token request

If the client has obtained a valid authorization code, it will exchange that code for an access token in the authorization server's token endpoint, authenticating the client when required, as described in the standard's section 3.2.1 [65]. The token request is sent using an HTTP POST request with the token request parameters (*grant_type*, *code*, *redirect_uri* and *client_id*, see Table A.4) encoded using the "application/x-www-form-urlencoded" format, as defined by the standard authorization code grant type.

Token response

The authorization server verifies that all required parameters are included in the token request and valid. In that case, the authorization server issues an access token, whose scope is the intersection of scopes requested in the authorization request and scopes included in the preauth token's *preauth_scope* parameter. The *access_token*, *token_type*, *expires_in* and *scope* parameters are included in the entity-body of the HTTP response with a 200 (OK) status code using the "application/json" media type as defined in the standard's section 5.1 (see Table A.5). In contrast with the behavior defined in the standard, no refresh token can be included in this response, and the user must be reauthenticated using the just-in-time authorization subprotocol once the access token has expired. If the token request turns out to be invalid, the authorization server responds with an HTTP 400 (Bad Request) status code (unless specified otherwise) and includes the error response parameters (*error*, *error_description* and *error_uri*) in the entity-body of the HTTP response using the "application/json" media type as defined in the standard's section 5.2 (see Table A.6).

3.2.3 Security analysis

In this subsection we analyze the security provided by the proposed authorization flow. We first discuss how known attacks versus the OAuth 2.0 described in [89] affect the proposed solution or if they affect it at all, considering the five attacker models defined in [89]. As the proposed authorization flow is based on the authorization code grant type and uses the redirections mechanism defined in the standard, it would be affected by most attacks defined in [89] and countermeasures should be applied to prevent them as described in the original document. Having said this, it is worth noting that (seamlessly) authenticating the user on each access attempt combined with the short-lived nature of access tokens issued as a result of the just-in-time authorization subprotocol drastically limits the risks arising from any possible access token leakage.

The clickjacking attack defined in the standard's section 10.13 [82] deserves special attention. The OAuth 2.0 specification discourages allowing the authorization server to accept requests made from iframes. For instance, if a user lands on a malicious site that sends an authorization request from a transparent iframe, the web page on which the authorization server tries to gather the user's consent to grant the malicious site access to the user's resources would be loaded in the iframe. This transparent iframe may be overlaid on top of a set of dummy buttons placed directly under important buttons on the target site. When users click on a visible button, they are actually clicking a button (such as an "Authorize" button) on the hidden page. The proposed method does not incur in this problem, given that no interaction with the seamless_authorize endpoint is required. However, the seamless_authorize endpoint must be the only authorization server endpoint accepting requests from an iframe; this behavior must be prevented in the other endpoints by using the X-Frame-Options header.

Finally, in the proposed protocol, the biometric material required for user authentication is only exposed to the authorization server, which interacts directly with the user's device to collect it.

3.2.4 Conclusion

We have presented a novel authentication and authorization workflow that smartly combines state-of-the-art standards with novel biometric authentication methods to enable users to seamlessly gain access to their protected resources with high security guarantees. To that end, a protocol based on two new OAuth grant types has been defined. It combines long-term authorization information embodied as a preauth token with biometric authentication using fresh biomedical signals. This workflow could be easily adopted in already deployed scenarios given that it relies on state-of-the-art standards.

Although the user has to be reauthenticated each time the client application requires access to a protected resource, usability is not hampered, since the authentication is performed using signals that can be acquired seamlessly. Moreover, for the multi-user scenario in which more than one user shares the device the client application runs on, the proposed approach remains completely seamless, unlike existing approaches, which require some user interaction.

3.3 MAuth: Messaging platforms application authorization

The way users interact among themselves and with their environment is constantly changing, and the delivery of digital services continuously evolves to keep pace with these changes. The use of virtual assistants has recently revolutionized the service delivery paradigm. These assistants interact with users through multiple interfaces (smart speakers, messaging platforms, etc.) providing different kinds of services ranging from simple information enquiries (e.g. the weather forecast) to complex patient follow-up management [71]. As occurred with traditional web applications, virtual assistants might eventually need access to some users' private information held by an external organization to perform some operations on the user's behalf. For example, when a user asks his virtual assistant when his next appointment with the physician is scheduled, the virtual assistant might need to retrieve this information from the healthcare institution (on behalf of the user and with his consent) to answer this question.

Access to this information is usually provided through Web-APIs secured by the OAuth 2.0 framework. However, this framework was designed thinking on a concrete type of client profiles, web-applications, that are served by a web server and accessed by end-users using a web-browser as user-agent. Thus, grant types were conveniently designed so that the interaction between the resource owner and the authorization server is performed through the same interface (user-agent), that he was already using to access the client, i.e. a web browser. When communication channels other than web-browsers are used for the interaction between the end-user and the client application, such as messaging platforms, the underlying authorization problem is essentially the same to (provide the end-user with a reliable method to express his intention of authorizing the client to act on his behalf), but some considerations must be taken into account. These considerations include two aspects, to deal with security issues and usability aspects derived from changes in the communication channel.

3.3.1 Problem definition

In this new scenario (see Figure 3.7), if the client needs to access a resource stored at the resource server, which access has been secured using the OAuth framework, the client best attempt to obtain the user consent consists on sending a link to the user as a normal message in the conversation pointing to the authorize endpoint of the authorization server with the required parameters (the same URL to which the user would be redirected if the communication between the user and client would be through a web-browser). The user must follow this link and continue the interaction through the web-browser to authorize the client to access his protected resource. Once the client obtains the required access token, the client will again interact with the user through the messaging platform, so that he must return from the web-browser to this platform. Switching from the messaging platform to the web-browser and back may hamper the usability of the system, making users reluctant to use new communication channels to consume services.

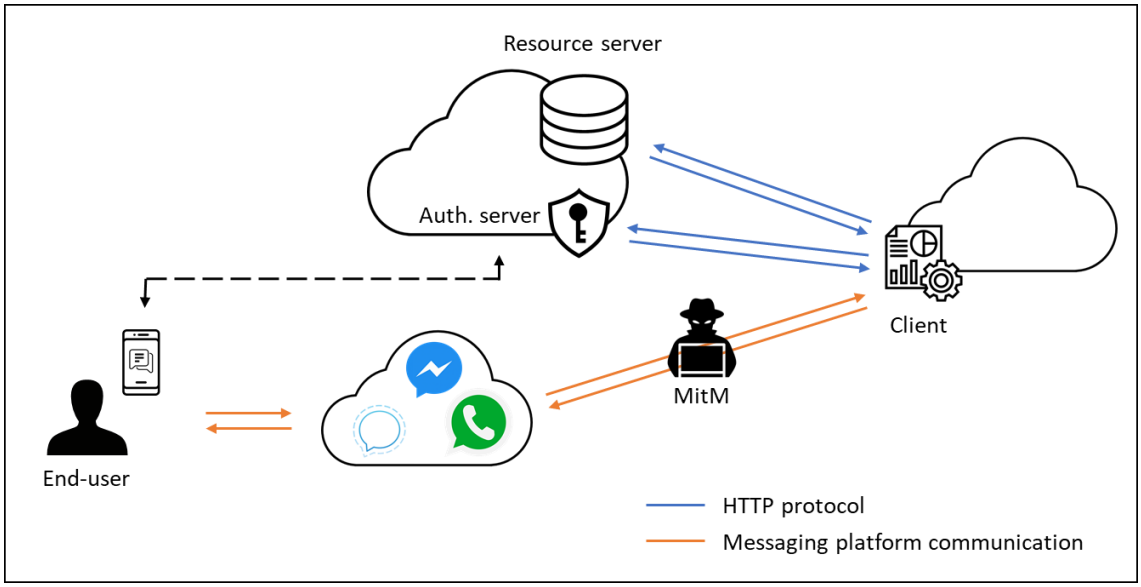

Figure 3.7 Reference scenario. End-user authorizes the client to access his resources at the resource server on his behalf.

Most messaging platforms, such as Signal, WhatsApp, Facebook Messenger or Skype, rely on the TextSecure protocol [72] to provide security to users' conversations. This protocol provides end-to-end encryption between communication peers, so that no eavesdropper is able to see the content of exchanged messages. To that end, it relies on a public key scheme where each user-agent (i.e. application installed in the user's device) generates its own key pair (public and private keys) at the time of installation. This key pair is further used to generate session keys that are used to encrypt the conversation. However, there is no way for a party to be sure a priori that a given public key belongs to his communication peer. A MitM attacker would be able to modify messages exchanged at the start of a conversation, tricking parties into believing that his own public key (the attacker one) belongs to the other communication peer, developing the attack. This is addressed by the so-called authentication ceremony, which consist in comparing parties' public keys using an out-of-band channel, thus preventing this risk. In current messaging platforms it can be done by comparing a safety number (which is really a concatenation of both users' public key fingerprint) or scanning a QR code. However, users must complete the authentication ceremony with every single client they want to interact with, which would hamper the user-experience. Studies point out that most users do not complete the authentication ceremony even to exchange sensitive information such as credit cards numbers [73]–[75], and the worst part is that the authorization server, which is the custodian of the users' data privacy, has no way of ensuring that the authentication ceremony has been completed between them before issuing an access token.

3.3.2 Protocol definition

In this section, we propose a new OAuth 2.0 grant type (see Figure 3.8) to be used with client profiles that use messaging platforms as user-agents to deliver services. In this new grant type, the authorization server is able to interact with the end-user using the same interface (user-agent) that he is already using to interact with the client, i.e. the messaging platform, integrating the security related aspects as a part of the protocol. The authorization server is also allowed to complete the authorization process securely without relying on the authentication ceremony between the user and the client while it can be sure that no MitM risk exists before issuing the access token.

Prerequisites

Since the resource owner will interact with the authorization server using a messaging platform, some prerequisites are needed before the proposed grant type can be used. There are two main prerequisites: obtaining the user identifier (id) on a specific platform and completing the authentication ceremony on this platform. This must be done using an out-of-band channel and would typically be performed at the time the user registers himself at the authorization server using its web interface. Note that an authorization server may support several messaging platforms. In that case, the user would firstly be asked to select which platform is going to be registered. On the other hand, the same user may register himself on several platforms.

Most messaging platforms currently support two ways of completing the authentication ceremony, by comparing a safety number or by scanning a QR code. We present three different ways of doing this. The first way is asking the user to manually write the safety number in a text box. The second way, which is appropriate when the registration is being done from a different device from that where the messaging application is installed (i.e. a laptop), is asking the user to show the pairing QR code to the webcam of the device where the registration is taking place. The third way, which is appropriate when the registration is being done from the same device that has the messaging application installed (i.e. a smartphone), is asking the user to take a snapshot of the pairing QR code and upload it to the authorization server.

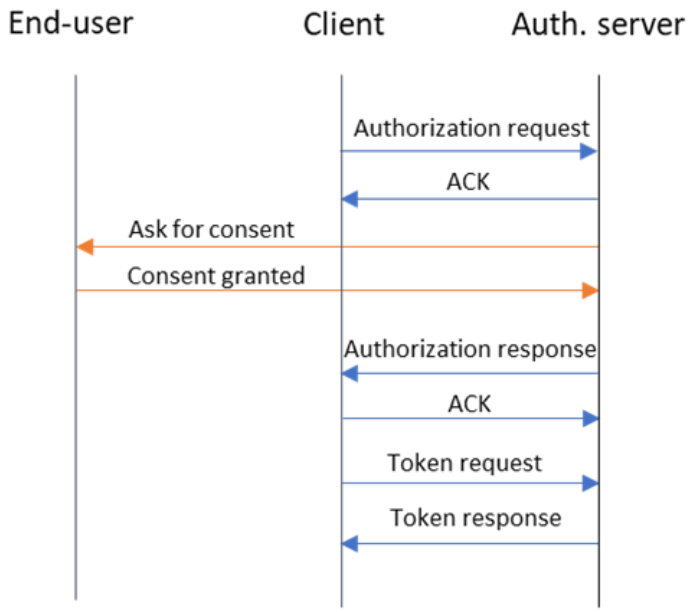

Figure 3.8 Proposed grant type to authorize third-party applications served through messaging platforms.

Authorization request

When the client application requires access to a protected resource, it would send a request to the authorize endpoint of the authorization server, including the authorization request parameters (see Table 3.5) using the "application/x-www-form-urlencoded" format with a character encoding of UTF-8 in the entity-body of the HTTP POST request:

Table 3.5 MAuth: Authorization request

Parameter	Compulsoriness	Description
platform	REQUIRED	The concrete messaging platform the user is interacting with (e.g. WhatsApp, Signal).
id	REQUIRED	The user identifier in the messaging platform (e.g. phone number)..
key_fingerprint	REQUIRED	Fingerprint of the long-term identity public. key
client_id	REQUIRED	Unique client identifier issued by the authorization server at registration time.
redirect_uri	OPTIONAL	Endpoint at which the client would be waiting for the end-user to be redirected back after a once the interaction with the authorization server would be completed.
scope	OPTIONAL	A string describing the access rights requested by the client.
state	RECOMMENDED	An opaque value used by the client to maintain state between the request and callback.

Among these parameters, there are some defined in the standard (*client_id*, *redirect_uri*, *scope* and *state*) and others defined specifically for this grant type (*platform*, *id* and *key_fingerprint*). The *platform* parameter is used to determine which specific messaging platform the resource owner is using to interact with the client (e.g. WhatsApp, Signal) and the *id* is the identifier of the resource owner at this platform (e.g. the user's phone number). These parameters are required to allow the authorization server to contact the resource owner since his identity is a priori unknown. The *key_fingerprint* is the fingerprint of what the client believes to be the end-user's public key. It is included to allow the authorization server to ensure that no MitM attack between the client and the end-user is taking place before issuing the access token, even if the authentication ceremony between them has not been completed.

Authorization request processing

When the authorization server receives this request, it is validated ensuring that all the required parameters are present and valid. Then, the authorization server verifies that the received combination of *platform* and *id* has previously been registered for any user. If so, the *key_fingerprint* received from the client is verified to ensure that it is the same as that shown to the authorization server (the fingerprint of what is currently being shown to the authorization server as the end-user's public key) and that it is also the same as that which was stored as a result of the authentication ceremony performed as explained in the "prerequisites" section (the fingerprint of the actual end-user's public key). If everything goes as expected, the authorization server acknowledges the received request sending a 200 OK response to the client. Otherwise, the authorization server quickly rejects the authorization sending a 400 Bad Requests message, including the error response parameters (see Table A.3) in the response body with the error parameter set with some of the values defined in the standard. If the provided combination of *id* and *platform* has not been registered previously by any user, the error parameter is set to "access_denied". If the public key fingerprint sent by the client does not match the one registered in the authorization server for that combination of *id* and *platform*, the authorization server returns the error parameter set as "public_key_not_match". It can be used by the client to inform the resource owner about a possible MitM attack and abort the communication when considered necessary.

At this point the authorization server can already contact the resource owner at the *platform* and *id* specified by the client. In this interaction, the authorization server may ask the user for some extra authentication information, like a one-time password (OTP) or some voice biometrics, or simply rely on the possession of the device where the messaging application is running (i.e. the possession of the complementary private key of the public key that was associated to a specific user during the authentication ceremony at the time of registration). Once the user identity has been verified, the authorization server evaluates the access control policies as it normally does (the specific policy evaluation method lies outside the scope of the OAuth standard). If everything goes as expected, the authorization server contacts the resource owner again to inform him the client application is requesting access to a resource on his behalf and asks him to authorize the client to complete this operation.

Authorization response

Once the authorization server has obtained the resource owner consent, it sends the authorization response to the client endpoint specified at the time of client registration or in the authorization request by the *redirect_uri* parameter. This response includes the authorization parameters (*code* and *state* with the meaning defined in the standard Authorization Code grant type, see Table A.2) in the entity-body of the HTTP POST request using the "application/x-www-form-urlencoded" format with a character encoding of UTF-8.

If the resource owner consent cannot be obtained the parameters included in the response would be those defined in the error response (*error*, *error_description*, *error_uri* and *state*, see Table A.3) setting the error parameter as “access_denied”. Independently of the result of the authorization and the parameters included in the authorization response, the client acknowledges the reception of the authorization response sending a 200 Ok response.

Token request and response

Finally, if the client has obtained a valid authorization code, it would be exchanged for an access token in the token endpoint of the authorization server using the token request and token response defined in the standard, authenticating the client when required. Once the client application has obtained the access token, it can obtain the required resource from the resource server authorizing the operation with the obtained access token.

3.3.3 Security analysis

In this subsection we analyze the security provided by the proposed authorization flow. First, we show how this method allows the authorization server to prevent the existence of a MitM between the end-user and any client, only requiring that the authentication ceremony between the end-user and the authorization server has previously been completed. Secondly, we analyze the degree of security of the proposed method in the face of known attacks against OAuth.

MitM attacker

All the trust in the TextSecure protocol is based on asymmetric cryptography. Each participant generates its own key pair (a public key, K_x^+, and a private key, K_x^-) that is used in the generation of all the subsequent cryptographic material required to encrypt and sign all messages exchanged during the conversation. Thus, a MitM attacker has to cheat both communication ends to make one communication end think that the attacker's public key (K_e^+) actually belongs to the other communication end [76]. In the scenario shown in Figure 3.9, the user has been cheated into thinking that the attacker's public key (K_e^+) really belongs to the client. In the same way, the client has been cheated into thinking that the attacker's public key (K_e^+) really belongs to the user. Finally, the user can be sure that what he believes to be the authorization server's public key (K_{as}^+) belongs to the authorization server and, in turn, the authorization server can be sure that what it believes to be the user's public key (K_u^+) really belongs to the user, thanks to having completed the authentication ceremony (which is a prerequisite of the proposed protocol).

In such a scenario, the MitM would be able to see and modify messages exchanged between the end-user and the client, while the authorization server has no way of detecting his presence if no additional measures are applied. The use of the *key_fingerprint* parameter in the authorization request of the proposed protocol is intended to sort out this situation. When the client starts the authorization process, the fingerprint of the attacker's public key (K_e^+) is included in the request, since the client has been cheated into thinking that this public key actually belongs to the end-user. When the authorization server receives the authorization request, it is able to compare the fingerprint of the received public key (K_e^+) with the fingerprint of the public key that it has stored for the end-user (K_u^+) as a result of the authentication

ceremony. Any time that a MitM appears between the end-user and the client, the public key fingerprints will not match, and the authorization server would be able to detect its presence before issuing any access token.

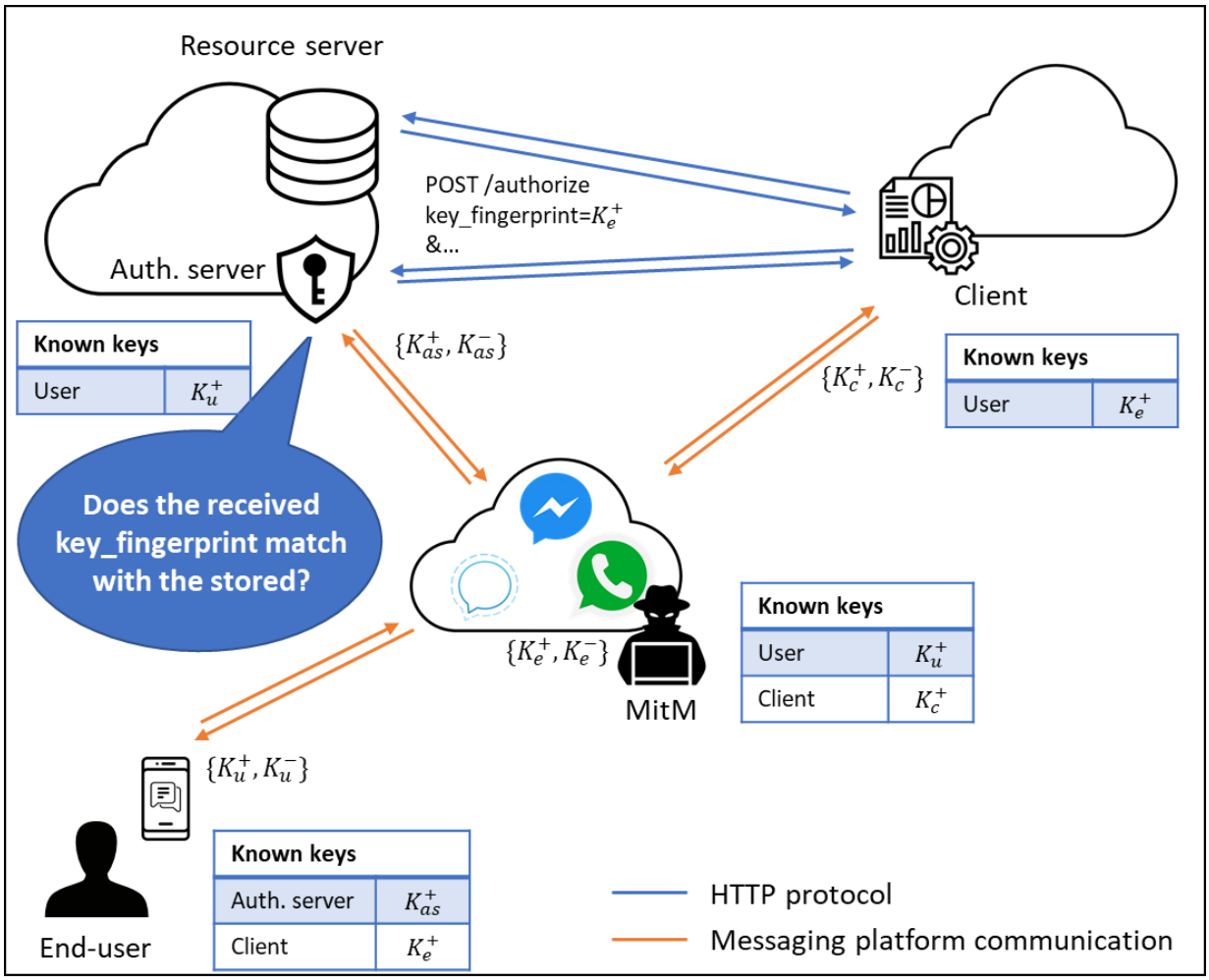

Figure 3.9 Man-in-the-Middle prevention showcase using the proposed grant type

Security against known attacks

In this section, we compare the security provided by the standard Authorization Code grant type through a web-browser as suggested in Section 3.3.1 (method A hereafter) with the new grant type proposed in Section 3.3.2 (method B hereafter). To that end, we analyze how attacks described in [77] affect these solutions or not, considering the 5 attacker models defined in [77] and the MitM attacker presented in the previous subsection. We classify these attacks in three groups. The first group includes attacks that do not depend on the grant type flow but are related with other architectural aspects of the OAuth framework. These attacks include Access Token

Leakage at the Resource Server, TLS Terminating Reverse Proxies, Refresh Token Protection and Client Impersonating Resource Owner. Attacks in this group affect both methods as they are independent of the specific grant type, and its countermeasures should always be applied. The second group includes those attacks that are tightly coupled to the redirection mechanism used by the Authorization Code grant type or to the use of web browsers as user-agents. Thus, they would affect method A but would not affect method B, at least in its current form. This group includes Credential Leakage via Referer Headers, Credential Leakage via Browser History, Authorization Code Injection, Access Token Injection, Cross Site Request Forgery, Open Redirection and Clickjacking. Finally, attacks in the third group, which are the most interesting for us, affect both methods in different ways and are analyzed more carefully below. These attacks are Insufficient Redirect URI Validation and Mix-Up Attacks. Table 3.6 summarizes this information.

Table 3.6 Exposure of methods A and B to attacks defined in [77]. Attacks in the first, second and third groups are written in yellow, green and red respectively.

Attack vector	A	B
4.1. Insufficient Redirect URI Validation	Yes	Yes*
4.2. Credential Leakage via Referer Headers	Yes	No
4.3. Credential Leakage via Browser History	Yes	No
4.4. Mix-Up Attacks	Yes	Yes*
4.5. Authorization Code Injection	Yes	No
4.6. Access Token Injection	Yes	No
4.7. Cross Site Request Forgery	Yes	No
4.8. Access Token Leakage at the Resource Server	Yes	Yes
4.9. Open Redirection	Yes	No
4.10. 307 Redirect	Yes	No
4.11. TLS Terminating Reverse Proxies	Yes	Yes
4.12. Refresh Token Protection	Yes	Yes
4.13. Client Impersonating Resource Owner	Yes	Yes
4.14. Clickjacking	Yes	No

The Insufficient Redirect URI Validation attack, as described in [77], is conducted as follows. First, the attacker needs to trick the user into opening a tampered URL in his browser that launches a page under the attacker's control. This URL initiates an authorization request with the client ID of a legitimate client to the authorization endpoint including a redirect URL under the attacker’s control and matching the registered redirect URL pattern for the legitimate client.

The authorization request is processed and presented to the user. If the user does not see the redirect URI or does not recognize the attack, the code is issued and immediately sent to the attacker's domain. When using method A, the MitM only needs to tamper with any legitimate authorization link sent by the client, so that the *redirect_uri* points to a domain under his control. In this case, it would be difficult for the user to notice that he is being attacked, given that starting the authorization by opening the link is part of the legitimate authorization using method A. On the other hand, when using method B, the attacker's best attempt would be to trick the user into believing that a client under his control is actually a legitimate client (e.g. initiating a new conversation with the user and stating that it is a known client application whose phone number has changed recently) and to obtain the user's consent to access his protected resources. Independently of the method used, this could be shortcut by strictly validating redirect_uris (i.e. performing strict string matching instead of supporting regular expressions) at authorization server.

Mix-Up attacks require the client to try to obtain authorization from the user using an authorization server under the attacker's control. When using method A, the MitM can simply modify the user messages to trick the client into thinking that the user has selected the authorization server under the attacker's control, when he actually has not. On the other hand, when using method B, this attack would only be possible if the user intentionally selects the authorization server under the attacker's control for any reason. This attack could be prevented by the client using distinct redirect URIs for each authorization server.

Finally, there is the passive MitM attack, where the MitM is placed between the client and the user without modifying any message with the sole purpose of eavesdropping on the user's private information exchanged from his routine use of the client. When using method A, this kind of attack can be prevented by completing the authentication ceremony between each user and each client. However, as already stated, the authorization server, which is responsible for the security of the user's resources, has no way of being sure that this authentication ceremony has been performed before issuing an access token. Method B prevents this attack as explained in the previous subsection "MitM attacker".

3.3.4 Usability study

We have conducted a study to better understand how users perceive the proposed authorization method and what might make them reluctant to use it. This study consists of two tests, test A and test B. In both tests the subject is required to interact with a virtual assistant, Alfred, using the Signal secure messaging application [78]. At a certain point of the conversation, Alfred informs the user that he needs his authorization to access some protected resource on his behalf. In test A subjects are requested to complete the authorization process using the standard Authorization Code grant type through a web-browser, as suggested in Section 3.3.1 (see Figure 3.10). In test B they are requested to authorize Alfred using the new grant type proposed in Section 3.3.2 (see Figure 3.11). After completing both tests, subjects are required to fill in a small questionnaire including some demographic information and their impressions about both authorization methods.

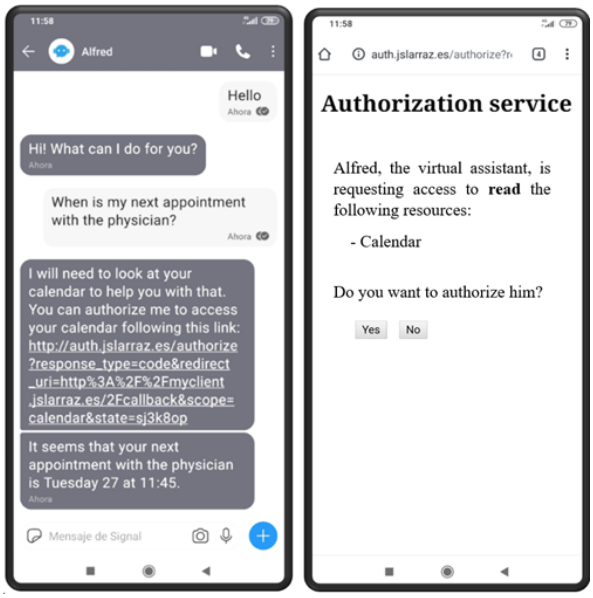

Figure 3.10 Example interaction in test A

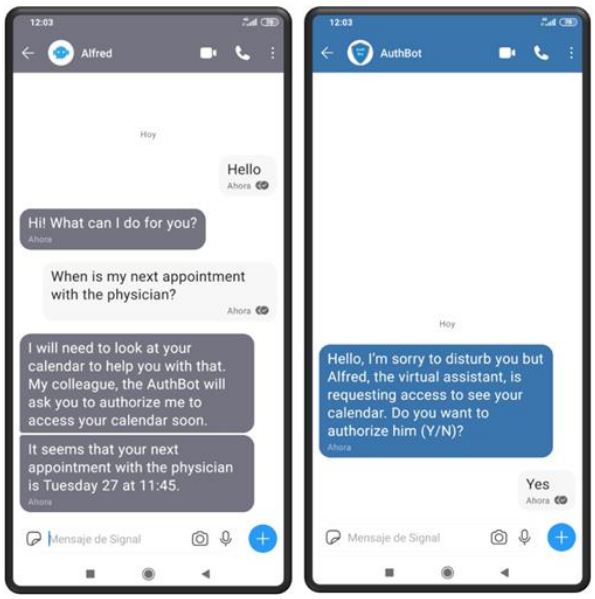

Figure 3.11 Example interaction in test B

Study recruitment, design and realization

The study participants were recruited from our campus and from our circle of acquaintances in equal parts. We ensured that none of them previously knew what our work consists of or the objective of the study, to avoid biased results. We designed the study so that each subject would be provided with two similar smartphones (one for each test) with a preregistered virtual assistant contact. The reason behind using different smartphones for each test was that it eases

the subjects' understanding of what they are doing (authorizing the virtual assistant by two different means), as we saw during the study design, providing a more reliable feedback.

When the participants arrived, they were firstly asked to read and sign the informed consent. After that, we briefly explained the basis of the study and informed them that a study coordinator would be observing their interaction and would answer any possible question they might have. At this point, the study coordinator handed out the smartphone prepared for test A and provided the following context information:

Suppose that you are using the Signal app to normally interact with your virtual assistant, Alfred. You ask him when the following appointment with the physician is. The objective is to complete the required steps to obtain this information from Alfred.

After successfully completing the first test, the subject was provided with the other smartphone (prepared for test B) and instructed to repeat the task, after being warned that some steps in the process would be different. After completing the task, the subject was required to briefly explain to the study coordinator what he/she had done to check the subject's understanding of the technology (up to a certain point). The subject was then required to complete a small summary containing some demographic questions and some related with his/her impressions of both tests. The demographic questions include the subject's gender and age. The subjects also had to select one of three options describing their degree of familiarity with the technology. The three levels were "Occasional user", "Habitual user" and "Advanced user", defined as follows:

- Occasional user: your main use of computers is to occasionally navigate the web, send/read emails or see some videos in YouTube.

- Habitual user: you usually rely on a computer for many tasks daily and/or part of your work depends on it as a user.

- Advanced user: you are a computer enthusiast and/or your work involves deep level of computer understanding (programmer, computer sciences, etc.).

For each test, the question “What has been your impression on the usability of method X?” was asked to rate the usability of the authorization method. Possible answers to this question were integers from 1 to 10, where the higher score was the better. The question “Do you believe the method X to be secure? Why?” was also included for each test, where possible answers were “yes” and “no” along with a space to justify their answer. Finally, the participants were requested to answer the question “Which method would you prefer to use?” considering their overall experience and taking into account both usability and security. A text box was also required to be filled in including some “Specific comments that motivate your previous responses”. After completing the questionnaire, the study coordinator announced that the study had finished.

Demographics

A total of 24 participants took part in the study. One of them was a priori excluded from the study given that he affirmed that he did not understand what he had done after completing the test. The remaining 23 participants were categorized in accordance with three parameters, their gender, their age and their degree of experience of interacting with computers. 10 participants out of the 23 were male (43% of the total). The participants were categorized in three age groups: under 25, between 25 and 48, and 49 and over. The first group had 8 participants (35% of the total), the second group 10 (43%) and the last group 5 (22%). The level of familiarity with computers was categorized in the three levels defined previously: “Occasional user”, “Habitual user” and “Advanced user”. The first group had 10 participants (43%), the second group 4 (17%) while the last group had 9 (39 %). All this information is summarized in Table 3.7.

We can see that the population is reasonably well-balanced as regards the gender of the participants. Looking at their ages, the younger and mid-range groups are also well balanced while the oldest subjects’ group has less members than the others. Finally, the participants experience with the use of computers is skewed since only a few are habitual users. This is due to the recruitment procedure. Most of the participants from our circle of acquaintances are occasional users, while participants from our campus are advanced users. However, the occasional users and advanced users’ groups are well balanced.

Table 3.7. Participants demographics

Individual characteristics	N	%
Gender		
Male	10	43
Female	13	57
Age		
0-24	8	35
25-48	10	43
49+	5	22
Tech. Level		
Occasional user	10	43
Habitual user	4	17
Advanced user	9	39

Results

In this section we present the results obtained from the usability study both qualitatively and quantitatively. The participants' responses to the questionnaire are summarized in Table 3.8. The first row in the table includes the mean number of points with which participants rated the usability of both methods out of a maximum of 10. The second row shows the number of participants who believe each method to be secure while the last row shows how many participants preferred one method over the other.

Table 3.8. Overall results

Question (overall)	**Test A**	**Test B**
Usability	7.74	8.52
Security	13	23
Pref. method	4	19

There is no significant difference between the usability rates obtained by both methods. As many participants stated, "both methods are very simple to use". However, the method proposed in this work obtained a slightly better result. In Table 3.9 we can observe participants comments that justify this. From the usability point-of-view, most participants who preferred method B said that it is simpler because they do not have to leave the application to complete the process (15 participants). A smaller set of participants stated that they prefer method A since the interaction is more similar to that which they currently use for authorization tasks (just 3

participants). From the security perspective, 13 out of the 23 participants believed the method used in test A to be secure. Many participants (16 out of 23) expressed their concern that they do not feel comfortable clicking the link provided by the client. However, some of them (6) still considered this method to be secure. On the other hand, all the participants involved in the study believe the method used in test B to be secure. Just one participant pointed out that he obtained a better security impression with the method for the test A stating that "seeing the link makes me more comfortable as I get a deeper understanding about how the system works". The conjunction of all these facts explains that most participants (19 out of 23) prefer to use the method proposed in this work.

Table 3.9. Participants' commentaries

Participant comments	#
Test A	
It is more familiar	3
The interaction is simpler	2
Seeing the link gives me more security	1
Test B	
The interaction is simpler	6
It is more secure	14
Do not have to leave the app	15
I do not feel comfortable clicking a link	16

Figure 3.12 details the results showing differences between the population groups considered in this work and previously detailed. Figure 3.12a shows results distributed by gender, Figure 3.12b shows them distribution by ages while Figure 3.12c does the same with the level of familiarity with the technology. In all the subfigures, the bars are grouped in three categories: the usability rate, security and the preferred method. The first group of bars represents the usability rate assigned to each test (over a maximum of 100 points), the second group shows the percentage of participants that believe the method to be secure while the third group shows the participants' preferences of one method over the other. The bars for the same demographic group share the same colour between categories, while the solid bars represent results for Test A and the hollow bars for Test B.

Figure 3.12a shows no significant differences in how participants of different genders rated the usability, although male participants believe method A to be more secure than female participants, which is also reflected in the preferred method for each of them. Figure 3.12b, which shows the results split by the age of the participants, indicates that there are no significant differences among users in the 18 to 24 age group and users in the 25 to 48 age group in any of the three categories. However, older participants (49 and over) rated the usability of method A as being worse than method B and had less confidence in the security of method A. This is reflected in the fact that no user in this group preferred method A over method B.

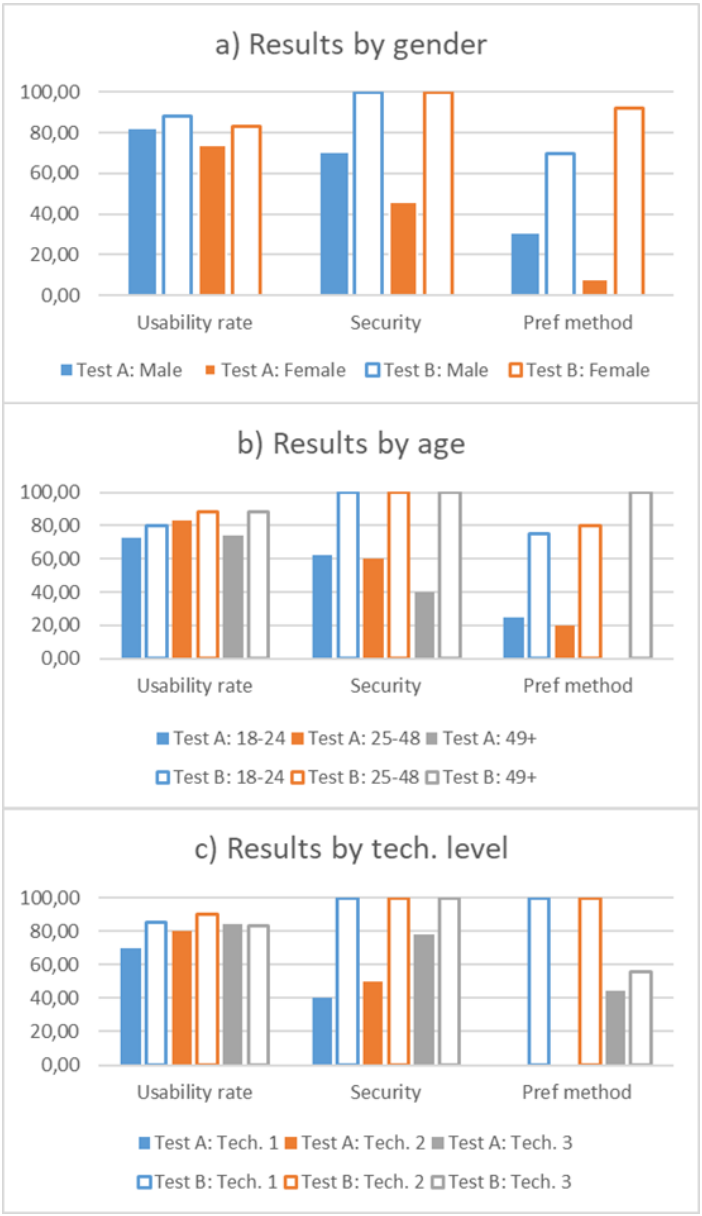

Figure 3.12 Usability study detailed results

The most interesting results can be seen in Figure 3.12c, which shows the results depending on the technical abilities of the participants. In this figure, occasional users are labelled as tech. 1, habitual users as tech. 2 and advanced users as tech. 3. The more experienced participants rated the usability of method A more highly than others with less experience. A greater number of experienced participants also trusted method A to be secure compared with participants in other groups. Finally, all the participants that preferred method A over method B were advanced users. However, none of the participants involved in the study noticed that the link sent by the client points to a HTTP service (see Figure 3.10), which is not using TLS to secure the connection (i.e. using HTTPS instead). In a real scenario this link might be sent by a MitM (trying to cheat the user) if the authentication ceremony has not been completed between the user and the client. This demonstrates that most users (including some graduates in computer sciences) are far from understanding all the security implications of their decisions and actions. Thus, security methods should be designed to protect users' security on their behalf, reducing their exposure to possible threats derived from their actions.

In this context, minimizing the number of required authentication ceremonies would improve not only the system usability but also the security of the communication. Using the flow proposed in this work with the OpenID Connect [63] protocol to provide federated identity would help with this problem. As it was presented in Section 3.1, in the OpenID Connect protocol, the client application wants to obtain some information about the end-user (such as his/her identity) from an identity provider. One of the OAuth grant types is used to allow the identity provider to authenticate the user and obtain his/her consent to share the information with the client. A client application that uses the OpenID Connect protocol with the proposed grant type to deal with users' identities would not need to worry about the presence of a possible MitM attacker even without completing the authentication ceremony. In this case, the identity provider would seamlessly verify that the public key fingerprint included in the authorization request really belongs to the user (just in the same way that an authorization server does as part of a normal authorization flow) ensuring that no MitM risk exists. Thus, the user is only required to complete the authentication ceremony with the identity provider instead of doing so with each client.

In the same way, an authorization server may rely on an identity provider to deal with a user's identity. In such a situation the authorization server would act as the client of the identity provider, so that only the authentication ceremony with the identity provider would newly be required. This is especially interesting for those scenarios where users' resources are spread across several resource servers protected by different authorization servers.

3.3.5 Conclusions

In this section we propose a new protocol to allow users to authorize third-party applications when the interaction with these applications is taking place through a messaging application. This protocol has been designed as a new OAuth grant type to take advantage of all the elements already defined in this framework, and to provide direct access to all existent APIs which are already secured using it. The proposed grant type allows the authorization server to interact with the resource owner directly through the same messaging platform already being used for interaction with the client. It also allows the authorization server to be sure that there is no risk of an MitM between the client and the user before issuing an access token.

Aligning the way that authentication and authorization tasks are handled with how users interact to obtain the service that requires these tasks improves the overall system usability. In the usability test, we have seen that most users found the proposed method usable for authorizing clients through messaging platforms and preferred it to using a web-browser with this same purpose. This is especially true for those users less experienced with computers, who rated our proposed approach highly for both usability and security. This is very important since users with less technical skills are the main target of the new service delivery paradigm aiming to reach more population sectors.

3.4 U2UAuth: User-to-user delegation

During the past few years, many data protection laws, such as the GDPR in Europe, have been adopted to highlight users' rights to manage their information lifecycle, deciding among other things who and how is going to use it. Currently there is not a widespread way to enable users to delegate access rights. Most extended way involves users writing access policies under the restrictions of a concrete access control model. Then, these policies are evaluated on each access attempt to decide whether it should be permitted or not. However, the closer to the access attempt users can provide their decision, the more information would be available to made it. Querying users to approve an access attempt has traditionally been related with a high delay to obtain an authorization decision. Thus, it has not been addressed except for a few professional scenarios, such as healthcare urgencies, where a resource custodian must be on-call waiting to approve some required access. With the widespread adoption of smartphones and wireless networks, users are continuously interacting using messaging platforms (e.g. WhatsApp) [79]. Thus, this new communication channels may be used to align new authorization requirements with everyday user's interactions, returning users the complete control of his/her data.

Most extended way to provide access to protected resources is through a Web-API secured using the OAuth 2.0 framework. However, it assumes that the requesting party and the resource owner are the same entity, so that a concrete access control decision (permit or deny) would be always obtained from the policy evaluation. If the role of resource owner is decupled from the requesting party, the authorization server can interactively query the resource owner at the time the requesting party is attempting to access the protected resource to approve it. To avoid that the resource owner gets flooded by illegitimate requests of this kind, he would also be able to decide which requesting parties may ask for his consent and to name some resource custodians that could be asked to approve the transaction on his behalf under certain conditions. Two implications are derived from this situation. First, given that the policy decision might not be limited to simple permit or deny but complex decisions including which resource custodians may be queried, both policy engine and OAuth ecosystem must speak a common language to exchange this information. Second, following the OAuth protocol as defined in the standard when the requesting party has not the required access rights, and querying the resource owner (or a resource custodian on his behalf) to approve the transaction is required, the requesting

party would be held at the authorization server until the resource owner sends his response, moment when the client might had already discarded the data related with this access request.

3.4.1 Protocol definition

In this section we propose an OAuth authorization code grant type extension to support those cases where the requesting party has not access rights to get the requested resource, but he is still allowed to ask some resource custodian to approve the transaction. This extension has two variants (see Figure 3.13) to be used depending on if the client redirection endpoint is reachable from internet or it is not. We also propose a data format to express the result of the policy evaluation, including information about the custodians of the resource and the order they should be asked, so that any policy engine that could generate this kind of response could be used jointly with the proposed OAuth extension.

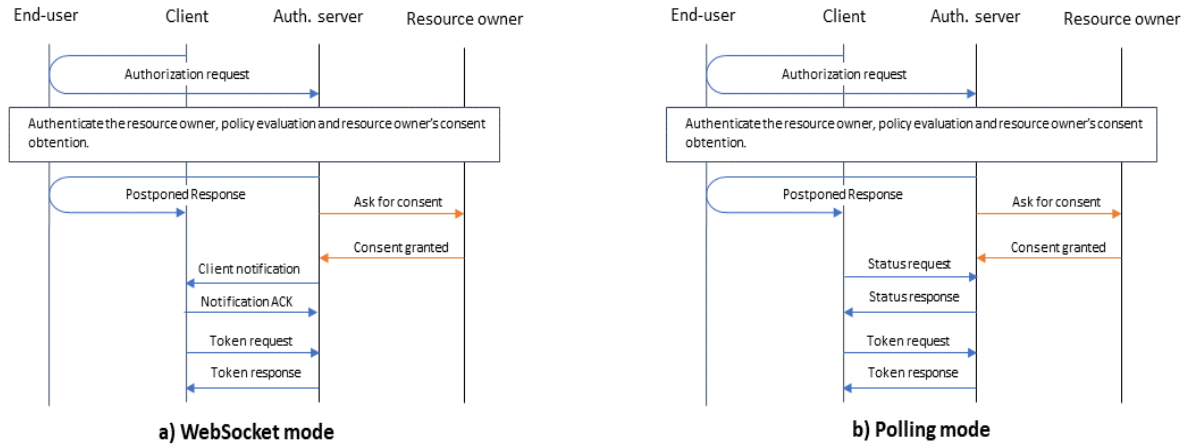

Figure 3.13 Authorization code grant type extensions

Prerequisites

When the resource custodian would be contacted by the authorization server using a messaging platform, some pre-requisites would be needed. There are two main prerequisites: obtaining the user identifier (id) on a specific platform and completing the authentication ceremony on this platform. What the authentication ceremony really does is to verify the fingerprint of the other party's public key, and as already explained in Section 3.3 it is essential to avoid man in the middle attacks when using messaging platforms.

Authorization request

When the client requires access to the protected resource, it would redirect the requesting party to the authorize endpoint of the authorization server, as normally done in the OAuth 2.0 Authorization Code grant type. This request would include the authorization request parameters (see Table 3.10) using the "application/x-www-form-urlencoded" format with a character encoding of UTF-8 as query parameters.

Table 3.10 U2UAuth: Authorization request

Parameter	Compulsoriness	Description
interaction	REQUIRED	This parameter indicates how further interaction would be performed. Possible values defined in this extension are “polling” and “websocket”.
client_id	REQUIRED	Unique client identifier issued by the authorization server at the registration
redirect_uri	OPTIONAL	Endpoint at which the client would be waiting for the end-user to be redirected back after a once the interaction with the authorization server would be completed.
scope	OPTIONAL	A string describing the access rights requested by the client.
state	RECOMMENDED	An opaque value used by the client to maintain state between the request and callback.

Most of these parameters (*client_id*, *redirection_uri*, *scope* and *state*) are already defined in the standard Authorization Code grant type and are used with the same meaning. A new parameter, *interaction*, is required with this extension and currently can take values “polling” or “websocket” indicating how further interactions between the client and the authorization server would be performed.

Policies evaluation

When the authorization server receives the authorization request, it verifies that all required parameters are present and valid. If the request is valid, the authorization server authenticates the requesting party and evaluates the access policies. Whenever the policy evaluation decision is to permit or to deny unconditionally, the protocol continues as specified in the standard. Otherwise, if the requesting party has not access rights but is allowed to ask some resource custodian, the output of the policy evaluation would provide information about which custodians should be asked, in which order and how long the authorization server should wait for their responses. The OAuth protocol is policy engine agnostic and how policies are defined and evaluated is out of the scope of the specification. In this line, any policy engine that can provide this information on its output could be used with this OAuth extension.

When interaction with custodians is required, the authorization server would inform the requesting party about this fact and would gather his consent to interact with the resource custodian. If the requesting party approves it, the authorization server would look for the default way to contact the resource custodian (on which platform and at which identifier) that he specified at the registration time. Finally, the authorization server would verify that the public key has not changed and would ask the resource custodian to approve the operation.

Authorization response

The authorization server redirects the requesting party to the client endpoint (provided during the client registration or/and as parameter in the authorization request). The required parameters are included as query parameters in the URL. If the policy evaluation result was to permit or to deny, the authorization response parameters (*code* and state, see Table A.2) or the error response parameters (*error, error_description, error_uri* and *state*, Table A.3) would be respectively included, exactly matching the behavior defined in the standard Authorization Code grant type. Otherwise, the postponed response parameters would be included in the response instead:

Table 3.11 U2UAuth: Authorization response

Parameter	Compulsoriness	Description
status	REQUIRED	Value MUST be set to "decision_postponed" indicating that the access decision has not been taken yet
expires_in	REQUIRED	Indicates how long the client is required to keep the information related with this access request if no access decision is taken before.
state	REQUIRED	An opaque value used by the client to maintain state between the request and callback.

These parameters include the *status, expires_in* and *state*. The *status* parameter must be set to "decision_postponed" indicating that the access decision has not been taken yet. The *expires_in* parameter indicates how long the client is required to keep the information related with this access request if no access decision is taken before and would be set at least as the sum of timeouts specified for all custodians in the policy evaluation output. The *state* parameter would be used to keep state between requests and responses. After redirecting the requesting party to the client, it would be able to continue using the client as usual (although he does not have access to the protected resource yet).

Depending on whether the value of the *interaction* parameter in the authorization request was "websocket" or "polling", one of the two defined protocol variants would be followed, the websocket mode or the polling mode respectively.

WebSocket mode

If the *interaction* parameter was set to "websocket" in the authorization request, the WebSocket mode would be used (see Figure 3.13a). In this mode, if a response from the resource custodian is obtained prior than the timeout expires, the authorization server would notify with the client, sending a HTTP GET request to the client end-point including the following query parameters in the URL. If the resource owner approves the resource access, the authorization server would include the authorization response parameters (*code* and *state*). If the resource owner response denies the protected resource access, the authorization server would include the error response (*error*, *error_description*, *error_uri* and *state*) parameters, setting the *error* as "access_denied". The client would acknowledge the notification with a 200 OK. If by the moment that the timeout expires, no response from the resource owner has been obtained, both client and authorization server should erase all the information related with this authorization request. The websocket mode is appropriate for those clients' which endpoint indicated by the *redirect_uri* can be accessed from the internet. This group generally includes web-applications.

Polling mode

If the *interaction* parameter was set to "polling", the polling working mode would be used (see Figure 3.13b). When using this mode, the client would poll the authorization server periodically to check whether it has already emitted a final decision. It would be done by sending a probe message containing the state parameter as URL query parameter to a new authorization server endpoint, the status endpoint. If the resource custodian has not responded yet, the authorization server would respond with a 400 Bad Request including the postponed authorization response parameters (*status, expires_in* and *state*) url-encoded in the HTTP body. The timeout parameter would be updated to the remaining time. If any resource custodian has already stated his intention of granting access the authorization server would respond a 200 OK message including the authorization response parameters (*code* and *state*) in the body of the HTTP response. If the resource custodian provides a negative answer, the authorization server would response a 400 Bad Requests including the error response parameters (*error, error_description,*

error_uri and *state*) url-encoded in the HTTP response body. The error parameter would be set as "access_denied". The polling mode is appropriate for those clients' which endpoint indicated by *redirect_uri* is not accessible from the internet. This group generally includes user-agent based application and native applications.

Token request and response

Finally, if the client has obtained a valid authorization code, it would be exchanged for an access token in the token endpoint of the authorization server using the token request and token response defined in the standard (see Table A.4 and Table A.5), authenticating the client when required. Once the client application has obtained the access token, it can notify the requesting party about it using web push notifications, sending an email or a text message, depending on the information available.

Compatibility

Server implementations of this specification may accept OAuth2.0 clients that do not implement this extension. If the *interaction* parameter is not present in the authorization request, the authorization server would assume that the client does not support this extension and would fall back to the protocol without it. This implies that whenever the result of a policy evaluation would include the *custodians* field, the authorization server would consider it as a deny to all effects. Thus, it would immediately return an error authorization response to the client with the error parameter set as "access_denied", without contacting the resource owner. On the other hand, if a server that does not support this extension receives an authorization request using it, the authorization server would simply ignore the *interaction* parameter and continue with the standard behavior.

3.4.2 OPA profile

We have defined an OPA profile to easily enable users to define their own policies. This profile supports both administrative and user delegation considering three kind of rights: access rights, ask rights and delegation rights. Access rights are used to explicitly allow or deny users to access a given resource. Ask rights allow users that need access to a protected resource and do not have the required access rights to ask some resource custodian to approve the access.

Finally, the delegation rights are used to enable user delegation. A user provided with delegation rights would be able to assign access and ask rights to other users in the system.

This profile mainly consists of two policy modules and a series of directives. The first policy module, identified as "authz", is placed at the package "<system_name>.authz" and provides the functionality of this profile. The other policy module, identified as "internal", is placed at the package "<system_name>.internal" and cares about the security of the policy management.

Policies definition

Application related policies are classified in two categories: administrative policies and user policies. Administrative policies are defined by administrative personal to perform administrative delegation. Policy module identifiers of this kind of policies must be prefixed by the "admin" keyword and placed at the package "<system_name>.policies.admin". These policies are the root of the rights assignment and establish the base behavior of the system. Rules in these policies can be classified in three groups depending on the rights that it is related to. Rules that assign access rights must write its decision to the *admin_allow* variable while rules that assign ask rights must add the custodian's information to the *admin_custodians* set. Finally, rules that assign delegation rights would include users with those rights in the delegators set. Additionally, to the user identifier, a priority indicator that would be used during the policy evaluation would be added to this set for each user with delegation rights.

On the other hand, user policies are written by users that have delegation rights to perform user delegation. Policy module identifiers of this kind of policies must be prefixed by the user identifier and placed at the package "<system_name>.policies.user.<user_id>". Rules in user policies can be classified in two groups depending on the rights that it assigns, access rights or ask rights. Rules that assign access rights must write its decision to the *user_allow* variable while rules that assign ask rights must add the custodian's information to the *user_custodians* set.

Policies at the "internal" policy module defined in the proposed profile regulate the access to the policy management API, ensuring that users are only able to manage policies correctly prefixed and placed at the packages as explained before.

Policies evaluation

Policy evaluation is performed querying the "<system_name>.authz" package that is part of the profile. Inputs provided to the evaluation would depend on the concrete application, but it usually includes the requesting party identity, the client application that is acting on his behalf, the resource being access or the action requested to be performed on this resource. Figure 3.14 shows a JSON schema corresponding to the expected policy evaluation output. The decision to permit or deny the access is carried by the field *allow*. When the requesting party has not access rights but ask rights to the requested resource, the *allow* field would be set to true but the custodians field would be also included in the response. Custodians parameter is a list of the users that may be asked to approve a given access request. Each custodian element includes three fields, *prio*, *id* and *timeout*. The *prio* field indicates the order in which custodians would be asked in case that previous do not provide an answer in a defined period. The *id* indicates the custodian unique identifier in the server while *timeout* indicates the period after which next custodian would be asked.

```
{
  "title": "auth_output",
  "type": "object",
  "properties": {
    "allow": {
      "type": "bool",
      "description": "Indicates whether the access should be granted or not."
    },
    "custodians": {
      "type": "array",
      "description": "Users that should be requested to approve the access",
      "items": {
        "type": "object",
        "properties": {
          "prio": {
            "type": "number",
            "description": "place in which this custodian would be asked"
          },
          "id": {
            "type": "string",
            "description": "unique id of this custodian in the system"
          },
          "timeout": {
            "type": "number",
            "description": "time after which the request would automatically be
considered as rejected by this custodian".
          },
        }
      }
    }
  }
}
```

Figure 3.14 Policy evaluation output format

The “authz” policy module contains the combination rules to generate the evaluation output mixing both administrative and users defined rules. These combination rules work as follows. *Admin_allow* rules are firstly evaluated. If the evaluation of these rules explicitly permits or denies the access, this action would be assigned to the allow output parameter and would be immediately returned as evaluation output. If no decision is available at this point, delegation rules would be evaluated to obtain the delegators set, which includes users that can perform user delegation for this concrete request. *User_allow* rules for users in the delegators set would be evaluated at this point. If these rules for any user provides explicitly provides the decision of permitting or denying the access it would be written to the allow output parameter. Since rules written by different users may provide contradictory results, the allow would take the action indicated by the user with higher priority from the delegators set. If users with the same priority value indicates contradictory results the parameter allow would be set to false, denying the access. The policy evaluation would end including exclusively the allow parameter in its output.

If no access rights are assigned (or rejected) neither by administrative policies nor by users’ policies for any user with delegation rights, *admin_custodians* and *user_custodians* rules would be now evaluated. All custodians indicated by both administrative and user defined rules are included in the custodians output parameter and the policy evaluation ends. If there is at least one custodian defined for the current request the result of the policy evaluation would be the allow parameter sets as true and all custodians in the *custodians* parameter. Otherwise, the *allow* parameter would be set as false, denying the access.

3.4.3 Security analysis

First, the proposed protocol is subject to suffer attacks described in [77] and countermeasure should be applied as explained in the original document. This extension is directly compatible with the PKCE extension proposed in [80] and should be used whenever possible as recommended in [77]. No additional security issues derived from postponing the access decision have been detected.

3.4.4 Elapsed authorization times

In the proposed scenario, when the requesting party has not access rights but ask rights, it would not be able to access the requested resource until a resource custodian approves the transaction. Thus, the lower times required to obtain the custodian's response the more scenarios the proposed method would fit. We have conducted a study with the objective of measuring times taken by users to respond messages received over messaging platforms. To that end, we have developed a bot that randomly pulled study participants over the Signal secure messaging platform, asking them to send their response as soon possible (see Figure 3.15). Messages were sent to users for 10 days following a Poisson distribution with a mean occurrence of 1 message each 6 hours, from 8:00 to 00:00 to avoid sleeping periods.

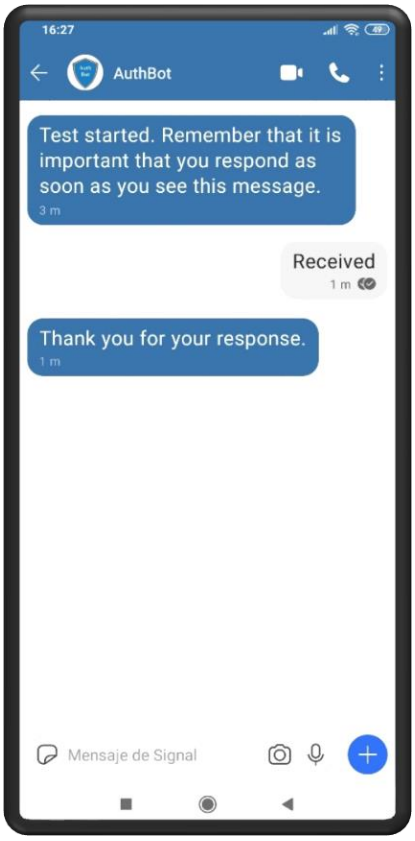

Figure 3.15 Interaction during the study

Study participants are from our campus and from our circle of acquaintances in equal parts. We have ensured that none of them previously knows about what our work consists of nor the objective of the study to avoid biasing the results. The study was performed by a population of 14 subjects. This population was skewed in terms of age given that most of the subjects (12 out of 14) were between 19 and 28 ages, while the number of male participants (6) is roughly equal to the females (8). A total of 398 events (i.e. a request and its response) were generated during the 10 days period that the study lasted.

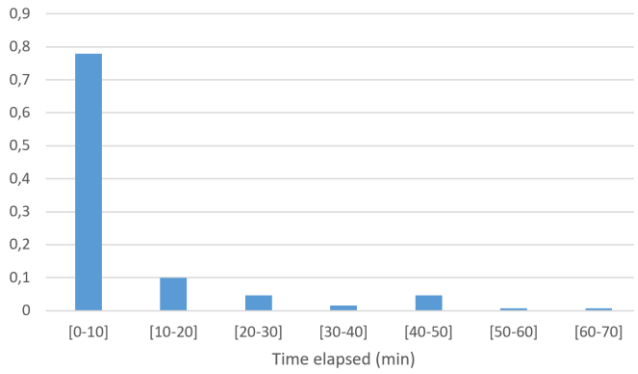

Figure 3.16 Times elapsed to obtain user's responses.

Histogram of obtained times are shown in Figure 3.16. These times are presented in 10 minutes intervals. We can observe that almost the 80% of these requests are responded within the first 10 minutes and more than the 92% within the first 30 minutes.

3.4.5 Conclusion

In this work we propose a set of tools, in the form of an OPA profile and an extension for the OAuth authorization code grant type, to enable user-to-user delegation. Providing this functionality as an extension of an existing OAuth grant type allows us to take advantage of all the elements already defined in this framework, and directly provide access to all existent APIs which are already secured using it.

Although user-to-user delegation could be already performed using UMA, it is a federated authorization framework that tastes more like a new protocol that uses OAuth than an OAuth extension. This extra complexity introduced compared with the OAuth protocol itself has hampered its adoption in those scenarios that only requires for user-to-user delegation rather than federated authorization, especially in all those scenarios where OAuth is already being used to protect user resources. In contrast to UMA, the extension proposed in this section has been designed to minimize the complexity added to the Authorization Code Grant type defined in the standard and to be compatible with it and with most relevant extensions, such as PKCE, as previously discussed. These considerations make fairly simple to adopt this proposed method both, from scratch and in already deployed scenarios, which is expected to bring user to user authorization closer to a wide spectrum of scenarios.

This kind of systems, that allows to query a resource custodian to approve a given access attempt, opens a new dimension in authorization scenarios, the multi-custodian authorization, where the approbation of n-out-of-N resource custodians is required to grant the requesting party access to the protected resource. It is especially interesting for those scenarios where more than one single party intervene in the life cycle of a resource, such as healthcare ones.

Finally, looking at the times elapsed to obtain users' responses provided in section 3.4.4, messaging platforms seem to be a suitable choice to asynchronously contact users when a quick interaction is required. Although the presented results have been obtained during hours when users are typically awake, it is the most plausible scenario given that authorization requests are usually triggered as part of the requesting party interaction. Thus, it would be very likely that the resource custodian would be awake when the requesting party needs the access, at least when both are located at the same time zone. By using messaging platforms, we would be able to resolve most access attempts in just a few minutes. Moreover, almost all of them require less than an hour to obtain a response. These times are likely to be acceptable for most applications. However, a deeper analysis would be required on the requirements of every concrete scenario. Although an hour could be acceptable for obtaining access to your friend fitness data, a few minutes might be required in healthcare emergencies.

Chapter 4: Border protection

After correct authentication and authorization, the user gains access to the protected resource. Data-in-transit security protects the information while it is transmitted from the user device to the resource server and back. Using encryption—hiding information from curious eyes—also prevents border protection devices, such as firewalls, from providing their functionality. This chapter analyzes the feasibility of using secure computation protocols to enable border protection devices on encrypted data without compromising data privacy.

4.1 Background

Data leakage and exfiltration is one of the top security concerns in modern information systems. The consequences of data exfiltration are huge for companies; information is the most valuable resource a company has. But these consequences escalate to disastrous when sensible data, such as health-related one, is involved due to the consequent legal implications. For decades, companies have addressed this problem by holding this sensible information within their boundaries. However, in the last decade, their doors have been opened to take advantage of all the advances that are boosting the ICT industry: information exchange between partners has been proved to provide an important strategic advantage (e.g. improvements in global supply chain). This unstoppable new connected environment makes it impossible for organizations to keep as isolated silos, experimenting the need to open their boundaries to take advantage of all these changes. However, with these new opportunities also come new risks and threats and the adoption of this connected scenario must be done carefully, without falling into security concerns.

The most extended strategy to deal with data security and privacy is the use of encryption. However, encryption, keeping the private information away from prying eyes, can also be used by attackers to mask their activities, allowing some threats go unnoticed for system defenses. For example, form the network perspective, malware is increasingly using TLS to hide itself from the Firewalls and Intrusion Detections Systems (IDS), that inspect the traffic that flows along the system boundaries to detect this kind of threats. Some existent approaches, like SSL

inspection, propose to use a man-in-the-middle like controlled (by the system administrator) attack [81] allowing the firewall to decrypt the whole packet and inspect its content. However, this approach presents several drawbacks, being the most important the compromise of data privacy, making it unacceptable for sensitive scenarios. Hence the systems boundary security vs privacy dilemma arises: if more information is used to decide whether a packet might suppose a security concern, more reliable would be the decision (security) but more information available compromises the confidentiality of the data flow (privacy).

Data privacy has lately attracted much attention and Secure Computation methods, which allows to perform some computation on data while keeping that data secret, evolved from lab toys to enable some relevant privacy-preserving applications in the real-world. Secure Multiparty Computation (SMC) [82], which is a subgroup of Secure Computation, allows to two or more parties to jointly compute a function over their private inputs, guaranteeing that no party would learn anything about other parties' inputs or intermediate results and only the output of the function would be revealed. In SMC, some parties might be corrupted, and they would try to extract information related with the inputs of the other parties. This behavior can be classified within two adversaries' models, each with its own security concerns. The Semi-Honest adversary model, which provides passive security, assumes that the adversary will cooperate to gather all information leaked from the protocol execution without deriving from the agreed protocol. In the malicious adversary model, which provides active security, the adversary may arbitrarily deviate from the protocol execution in its attempt to cheat. The only thing that an adversary can do in the case of dishonest majority is to cause the honest parties to abort having detected cheating.

One of the most basic SMC protocols is known as Oblivious Transfer (OT). The standard 1-out-of-k OT involves two parties, the sender, S, that holds k secrets, S_i with $0 \leq i \leq k-1$, and the receiver, R, that hold a choice selector, j, from [0, .., k-1]. At the end of the protocol the receiver obtains the secret associated with his choice selector, S_j, and anything about the rest of secrets. The sender does not learn anything about the choice selector. During the past few years, some approaches have been proposed to perform many OT in a bulk very efficiently [83], [84]. This performance enhancement has enabled the use of SMC protocols in many real applications.

OT is used as building block to enable other SMC protocols, such as the Goldreich-Micali-Wigderson (GMW) protocol [85]. GMW is a general-purpose SMC protocol that allows two or more parties to securely evaluate any function that can be defined as a Boolean or arithmetic circuit. Privacy in GMW is founded on an additive sharing scheme, where the real value of any wire in the circuit, W, is divided in shares, Wi, spread across the N parties evaluating the circuit. In the two-party Boolean-circuit case, inputs are provided as follows; parties generate a random masking bit, R_i, for each bit in their own inputs, X_i. Then, they send their generated masking bits to the other party and calculates $X_i + R_i$, so that R_i and $X_i + R_i$ are additive shares of each input bit, X_i, spread across the parties. Once both parties hold shares of all bits in the circuit inputs, both parties start to evaluate all gates in the circuit. NOT gates are evaluated by flipping the share of just one party and XOR gates are evaluated by each party performing the XOR of their own gate inputs' shares. Thus, evaluation of NOT and XOR gates only requires of local computation. On the other hand, evaluation of AND gates require of communication between parties and each gate can be evaluated performing a 1-out-of-4 OT. Once the complete circuit has been evaluated, both parties can open their shares of the output, so that they both will learn the output by adding their shares.

To improve the efficiency of the circuit evaluation some protocols can be evaluated in the preprocessing model. In this model the circuit evaluation is split in two phases, the preprocessing phase and the online phase. All the heavy cryptographic machinery is move to the pre-processing phase where some correlated randomness is generated. This correlated randomness is independent of the circuit inputs, so that the preprocessing phase can be run even before circuit inputs are known by parties. Then, in the online phase, previously generated correlated randomness is consumed to evaluate the circuit efficiently. The GMW protocol can be run in the preprocessing model using Beaver Triples [86] to efficiently evaluate AND gates. A Beaver Triple is a triple (a, b, c), where each party hold an additive share of a, b and c, so that $[c] = [a] \cdot [b]$. These triples can be generated from two 1-out-of-2 Random OTs per triple. It is interesting to note that as the Beaver triples do not depend on any circuit value, they can be generated in a bulk, taking advantage of existent OT extensions that could not be applied in the vanilla GMW protocol and provide remarkable performance improvements.

In the past few years some works already explore the possibility of using Secure Computation methods to perform Network Functions (i.e. routing, firewalling…) without compromising the privacy of the traffic being evaluated, which has been named as Secure Network Function (SNF). First approaches to enable SNF could be found in [87], [88]. These works use Searchable Encryption to check whether some patterns of interest are present or not in the ciphered payload. To that end, the packet payload is first tokenized and then these tokens are matched against the rule patterns. Those studies do not show interest only on the confidentiality of the ciphered data but also on the rules. The rules have to be protected given that in some cases can be intellectual property (e.g. commercial IDS rules). In [87] only the content of the rules is protected while some metadata (inspection fields, offsets and number of patterns) remains in clear. This could leak some confidential information and is addressed in [88]. Although those works pointed into a promising research direction, functionality achievable with those methods is quite limited since they only allow to look for patterns (i.e. do not support ranges matching nor regular expressions) inside the ciphered payload.

In [89], [90] authors propose novel methods to provide new functionalities to middleboxes. These improvements enable them to match encrypted values against encrypted prefixes and ranges (e.g. check if a port belongs to a range). This can be used to perform tasks such as load balancing, NAT or traffic classification. In [89], authors propose a new encryption scheme called PrefixMatch and use it to perform the secure range matching. Some metadata about the packet headers is leaked in this system. In [90] a different approach based on Secret Sharing Secure Multiparty Computation is taken. This approach is limited to substring comparison, and leaks information about which bits of the incoming packet do not match.

The importance of privacy-preserving packet filtering for Internet of Things (IoT) scenarios is discussed in [91]. In their work, authors state that message content filtering would avoid duplicate packet transmission which would reduce both computational and communication cost. To that end, they proposed the use of PReFilter [92], an efficient privacy-preserving relay filtering scheme for Delay Tolerant Networks (DTNs) in vehicular IoT communication.

4.2 SNF principles, requirements and threat models

Some approaches have recently enabled secure network function (SNF), which consists in performing a network function (routing, firewalling, etc.) without compromising traffic privacy. Secure multiparty computation (SMC) is a cryptographic primitive that allows to two or more parties to jointly compute a function over their private inputs, thus guaranteeing that no party learns anything about other parties' inputs nor intermediate results; only the function's output is revealed. In this model, SNFs are evaluated jointly by two parties, the middlebox and one of the communication peers, either the client (in Figure 4.1 represented by the user, social networks and data analytics) or the server—depending on the scenario—hereafter referred to as "host". The middlebox takes an action depending on the result of the network function evaluation (e.g., forwarding or dropping a packet), while the host holds the encryption keys and knows the packet content.

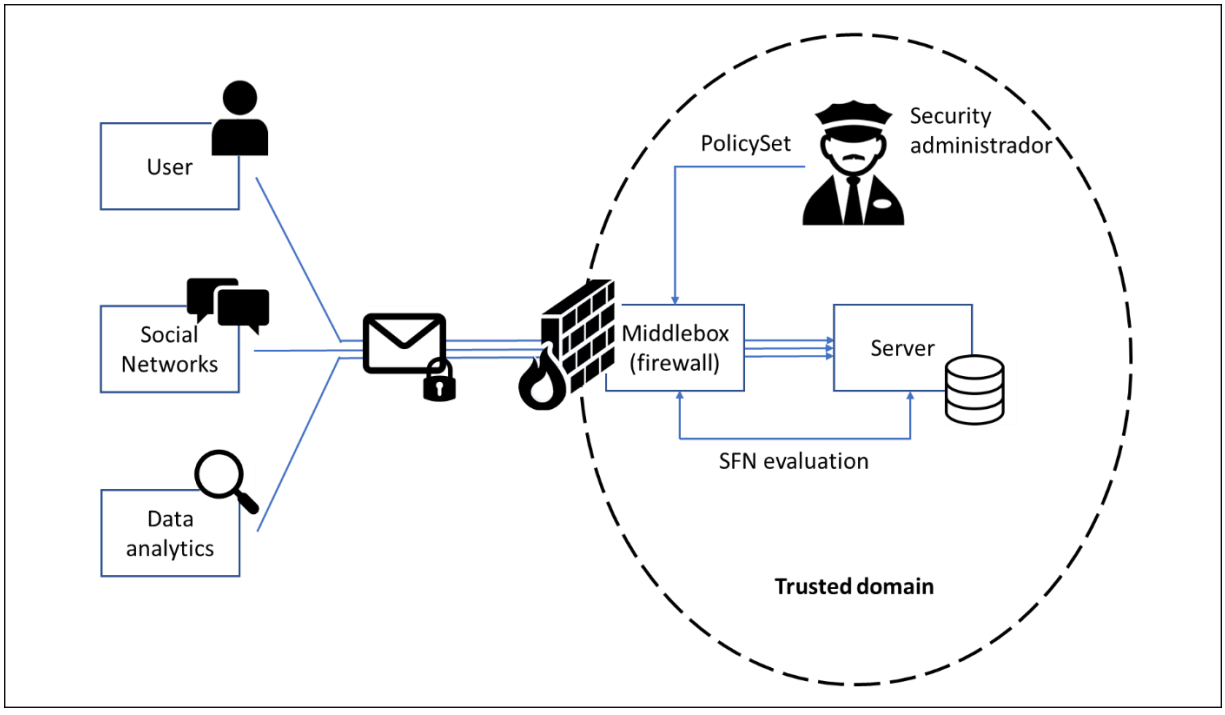

Figure 4.1 Reference scenario

General-purpose SMC protocols are usually designed to evaluate Boolean or arithmetic circuits. Thus, the target function to be evaluated (i.e., the network function) must be firstly represented as a circuit of this kind. Circuit complexity can be characterized by two parameters: size, which

is the total number of (non-linear) gates; and depth, which is the longest path from any input to any output. One function can be represented by several circuits, each with its own size and depth parameters; the preferred representation will depend on the SMC protocol used to evaluate the circuit and the scenario. The main SMC protocols can be categorized in two settings: computational and information-theoretic. Protocols in the computational setting, such as garbled circuits, present high computation and communication complexities that depend on the circuit size. In this setting, circuits can generally be evaluated in a constant number of communication rounds (that only depend on the protocol itself and are independent from the evaluated circuit), so that the time required to complete the evaluation depends only on the circuit size. In contrast, protocols in the information-theoretic setting present lower computation and communication complexities that also depend on the circuit size. However, they cannot be evaluated in a constant number of rounds (they usually depend on the circuit depth), requiring all parties to communicate in each round. Therefore, for most scenarios, the time required to evaluate a circuit would mainly depend on its depth.

Unlike most current SMC applications, which only require one protocol evaluation at a time, SNF needs one evaluation per packet and all these evaluations can be performed concurrently. Consequently, protocols in the information-theoretic setting are especially interesting for enabling SNF as their communication and computation requirements are lower than others in the computational setting, and parallelizing all those required evaluations could provide a better amortized throughput, thus maximizing the network load that can be processed with a constrained number of resources.

4.2.1 Requirements

Designing methods to perform SNF presents a series of challenges and the relevance of a given method can be evaluated by the extent to which it achieves the following requirements [93].

Data privacy

Data privacy is the main reason why SNF exists. It is intended to reduce (or eliminate) the exposure of sensitive data during the network function evaluation. This sensitive data usually includes, but is not limited to, the packet content. Depending on the specific network function

performed, other additional data might also be protected, such as the intrusion detection system (IDS) signatures or the firewall rules.

Ease of deployment

The proposed solutions should be designed to require minimal modifications in the existing infrastructure. For typical service access over the Internet, clients should not notice the use of SNF; therefore, the proposed changes should not involve additional interactions with them and the protocols used for the communication should also remain unaltered (i.e., IP/TCP/TLS).

Functionality

Required functionalities would be determined by the specific network function to be securely evaluated. Guaranteeing that each SNF can provide a similar functionality to its non-secure counterpart would be necessary. There are some challenging functionalities that hold particular interest but have not typically been addressed; this is the case of structured data interpretation, since many protocols rely on the JSON format for information exchange between parties, and stateful evaluations, since modern protocols such as HTTP2 rely on this state information to minimize the redundant information sent over the network.

Performance

Unlike other SMC applications in which the business is built around them (e.g., private auctions and private voting, among others), using SNF consumes some resources that otherwise would be destined to the business application. For example, in the scenario shown in Figure 4.1, the SNF provides firewalling and the network function is securely evaluated between the firewall and the resource server. In this scenario, evaluating the SNF requires consuming bandwidth between these two parties and computation resources at the resource server that otherwise could be used to serve more clients' requests. Thus, performance is especially critical for SNF applications.

Security

The middlebox should be able to provide its functionality as usual without making any additional assumptions. Most current SNF approaches require the assumption that, at least, one of the communication end points is honest, as discussed in the following subsection.

4.2.2 Threat models

The specific circuit to be evaluated depends on the desired functionality. However, the way inputs are provided to the circuit would depend on the threat model assumed. This work presents two threat models to fit in various scenarios, each with its own security assumptions and performance restrictions. To illustrate this, we propose a very simple scenario in which many clients using a VPN service for their external connections. A firewall placed at the boundary of the clients' network checks whether connections are allowed by inspecting the network parameters (IP addresses and ports) that are encapsulated within the encrypted VPN payload without compromising its privacy.

In both threat models, the middlebox is assumed to be honest-but-curious. It will perform its function as agreed beforehand, but it will try to obtain as much information as possible from the processed traffic. This is the most typical attacker model for middleboxes, in which curious system administrators try to gain access to their users' private information. In the first threat model, the packet sender would provide the information required to perform the function evaluation directly as protocol input. Following the previous example, the sender would provide the IP addresses and ports encapsulated within the encrypted VPN payload as inputs. Thus, the network function would be as simple as matching these parameters with the defined rules. A malicious sender can cheat the middlebox by providing other parameters than those actually present in the evaluated packet as inputs. In this situation, the receiver must be able to detect the cheat attempt and abort the communication. Thus, at least one host must be honest in this model. This threat model is assumed in most recent works on SNF; however, it might not fit in many real-world scenarios. Attackers that compromise the server of a publicly accessible service can trivially start a communication from the outside, thus controlling both end points.

In the second threat model, inputs are provided differently. The middlebox provides the encrypted packet as a protocol input and the host provides the encryption key. In this case, before the network parameters (IP addresses and ports) can be matched with the defined rules as in the previous threat model, parameters must be extracted from the encrypted payload. Thus, the function to be evaluated must firstly include the secure decryption of the packet and the extraction of the relevant information. In this threat model, the middlebox does not need to trust

any other party to provide a reliable decision. Although a malicious host can still provide erroneous input (i.e., another encryption key), it would be statistically improbable to find a key that, used to decrypt the packet, meets all the middlebox rules.

4.3 SNF evaluation

Some configurations are required before any packet evaluation. Since the evaluation is performed between two parties (the middlebox and the host), they both must agree on which SNF circuit representations to use. The middlebox, which would start the evaluation, must also know how to notify its peer that a new evaluation is to start. To that end, the host would be waiting for evaluation start requests on a specific port and the appropriate configuration (IP address and port) is provided to the middlebox beforehand.

In the normal operation mode (see Figure 4.2), each time a packet reaches the middlebox, it checks whether the packet source or destination addresses belong to a registered peer; if not, the middlebox drops the packet and sends an RST packet to its source. Otherwise, the packet is included in a buffer of packets to be evaluated with this specific peer. If there are no pending evaluations with this peer, the buffer is created, and a timeout is set. Once the buffer is full or the timeout expires, the middlebox sends the evaluation start request to the host (at the previously configured address and port) over a new session including a manifest with IP addresses and ports (both source and destination) of all packets in the buffer. This information is used by the host to identify the flow to which each packet belongs and obtain the session keys used for their encryption. The host responds to the evaluation start request with an evaluation start response, including its own manifest indicating whether each packet is recognized as belonging to an active flow or not. When the middlebox receives the evaluation start response, any packet that has not been acknowledged by the host is immediately dropped and the middlebox sends an RST message to its source. All the recognized packets form an evaluation set, and they are now evaluated using the GMW protocol with multiplication triples.

Circuit evaluation when using the GMW protocol requires a number of communication rounds that depends on the circuit depth (as with other protocols in the information-theoretic setting). The middlebox starts the evaluation (round 0) and sends a packet including the circuit identifier (Ci), the evaluation set size (ESs), an evaluation set identifier (ESi) and shares for its own inputs for each packet in the evaluation set. The circuit identifier points which SNF circuit representation would be used for the evaluation and it is included to ensure both parties use the same representation. The ESs is the number of packets included in the current evaluation set; it depends on the buffer size and may change over time depending on the network state. Larger ESs will introduce higher latencies in the evaluation process. However, greater throughput would be achievable, since the cost of the IO operations would be shared between packets in the same evaluation set and larger bandwidths can be achieved by larger packet sizes in most network technologies. The evaluation set identifier uniquely identifies each evaluation set that is currently under evaluation and would be included in any further packet related to it. This is necessary because more than one set can be evaluated concurrently, thus overlapping communication and computation times. The number of evaluation sets that can be evaluated concurrently is decided by the middlebox and depends on the latency between the middlebox and the host.

When the host receives this message from the middlebox, it is ready to begin evaluating the first layer in the circuit. Each layer, n, is evaluated in two phases, n_a and n_b. First, in phase n_a, all linear gates (XOR and NOT) in the layer are evaluated and each party calculates the public values (d and e) of multiplication triples, PV_n, for each AND gate in the layer and sends them to its peer. In the second phase, n_b, the public parameters received from the other party are used to evaluate all AND gates in the layer, thus obtaining a share of each gate output. Evaluating a circuit's gates only requires 1-bit operations, which are very inefficient in modern processor architectures. The resources required to perform 1-bit and word-width operations are the same. Thus, gate evaluation runs in a SIMD fashion, evaluating packets in the evaluation set in groups of the same size as the system's word-width, increasing performance up to 64 times in current 64-bit architectures.

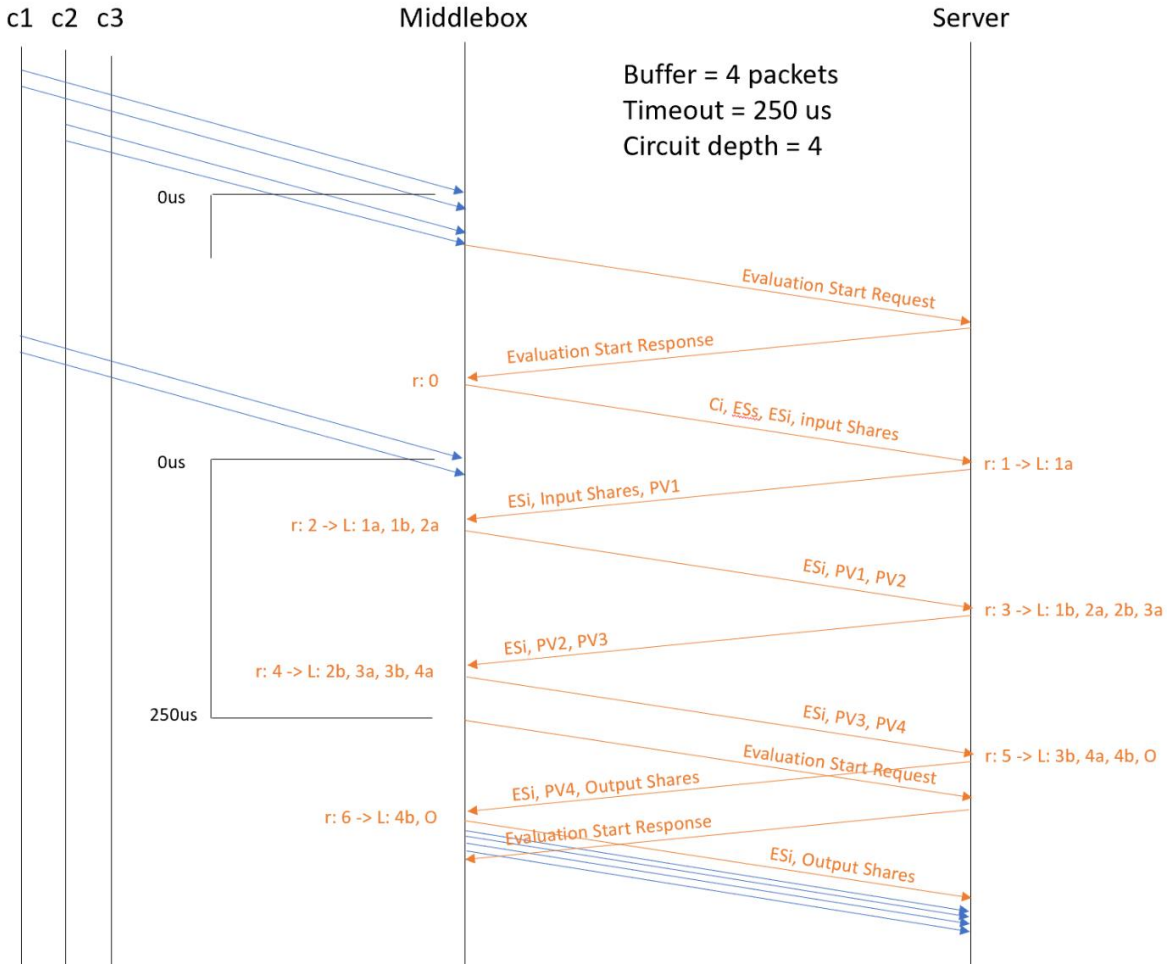

Figure 4.2 Function evaluation procedure

As a result of the round 1, the host sends shares of its own inputs and public values for each AND gate in the first layer of the circuit (PV_1) to the middlebox. In round 2, the middlebox completes the evaluation of layer 1 using the public parameters received from the host and sends back all public parameters (d and e) for each AND gate in layers 1 and 2 of the circuit (PV_1 and PV_2). The evaluation continues, so that, in each round n, one party (middlebox in even rounds and host in odd rounds) completes the evaluation of the n-2 and n-1 layers using the public parameters received from the other party and sends the public parameters for each AND gate in the n-1 and n layers. Once all layers in the circuit have been evaluated, both parties communicate their own shares of the circuit output, which are then used to reconstruct the output. After the evaluation, both parties learn the function output for each packet, and the middlebox takes the required action.

4.4 Adaptative circuit representation

Most functions can be synthetized in several circuit representations, each with its own depth and size parameters. For example, a 16-bit adder can be synthetized using the 16-bit ripple-carry adder design (see Figure 4.3a), which has a depth of 15 layers and a size of 15 (non-linear) gates. However, it may also be synthetized using the 16-bit Sklansky adder design (see Figure 4.3b), which provides the same functionality and reduces the circuit depth to four layers at the cost of increasing the circuit size up to 49 (non-linear) gates.

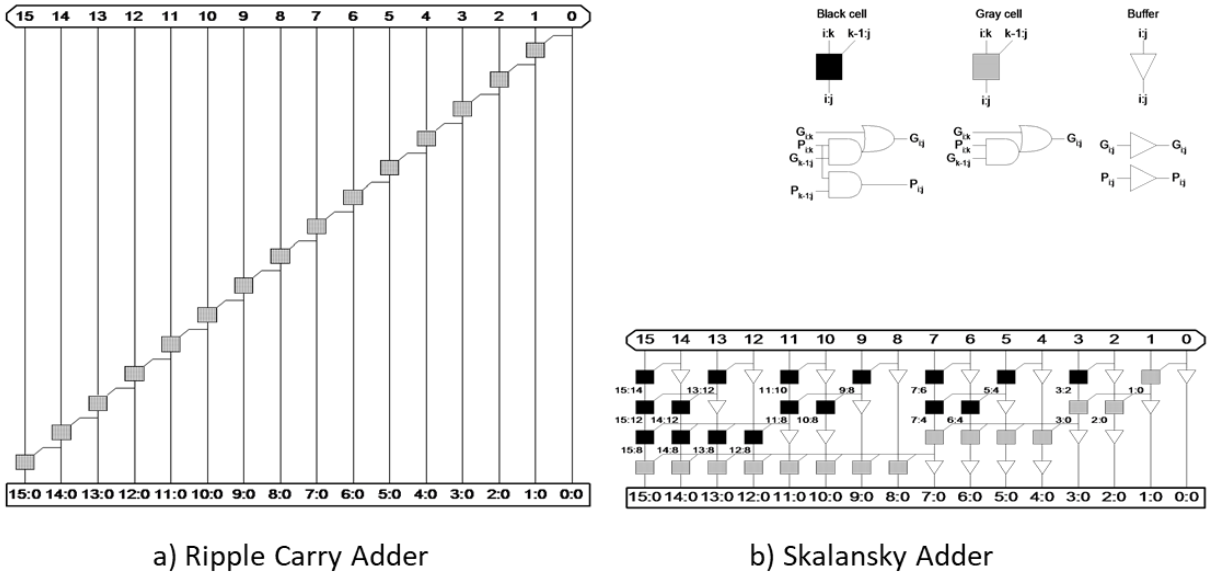

Figure 4.3 Topologies of Ripple Carry Adder and Sklansky 16-bit adders.

For a protocol in the information-theoretic setting, the circuit depth is related to the latency introduced by protocol execution (i.e., the time required to evaluate the circuit), while the circuit size is translated into the amount of communication and computation resources required to complete each single evaluation. As more than one concurrent evaluation can be performed in SNF applications, lower size circuit representations allow higher network loads to be processed at the cost of increasing the latency introduced. Since the network load is not a static parameter but varies over time, circuit representation (with varying depth and size parameters) may fit under various network conditions, minimizing the latency introduced by the SNF while ensuring that all the current network load can still be processed. In this section we present PyFrocen, a framework designed to ease the definition of the function to be securely evaluated and synthesize it into different circuit representations.

4.4.1Function design

First, the function to be securely evaluated (i.e., the network function) is designed. To that end, the PyFrocen framework provides an extended python syntax to define it. Figure 4.4 and Figure 4.5 show the definition of the ChaCha20 encryption/decryption algorithm [94] that is also used in this work. The “frocen_function” decorator is used to identify the entry function and provide the required information about circuit input and output. The frocen_function decorator receives one argument per function input and output. In the ChaCha20 algorithm example definition, the first three arguments of the frocen_function decorator correspond to function inputs (counter, key and nonce, respectively), while the fourth argument corresponds to the function output. Each argument is a triplet containing relevant information about the input/output. The first triplet element is an integer (from 0 to 2) containing the IO configuration (which party provides the input or which party receives the output).

Input:

- 0: public parameter
- 1: first party input
- 2: second party input

Output:

- 0: output to both parties
- 1: output to first party
- 2: output to second party

The second triplet element is an integer indicating the size of the parameter when it is a list, or “-1” when it is a scalar value. Finally, the last element in the triplet indicates the bit wide of the parameter or of each element in the list. The counter parameter in the ChaCha20 algorithm example definition is a 32-bit scalar value known by both parties (i.e., public value), while the key parameter is an array of eight 32-bit elements provided by the first party.

```
@frocen_function((0, -1, 32), (1, 8, 32), (1, 3, 32), (0, 16, 32))
def chacha20(counter, key, nonce):


    # Create state from inputs
    const = ["61707865", "3320646e", "79622d32", "6b206574"]

    state = []
    for c in const:
        state.append(c)

    for n in key:
        state.append(n)

    state.append(counter)

    for n in nonce:
        state.append(n)


    # Run the chacha block
    x = deepcopy(state)

    for i in range(10):

        # Odd round
        state[0], state[4], state[8], state[12] = QR(state[0], state[4],
state[8], state[12])  # column 0
        state[1], state[5], state[9], state[13] = QR(state[1], state[5],
state[9], state[13])  # column 1
        state[2], state[6], state[10], state[14] = QR(state[2], state[6],
state[10], state[14])  # column 2
        state[3], state[7], state[11], state[15] = QR(state[3], state[7],
state[11], state[15])  # column 3

        # Even round
        state[0], state[5], state[10], state[15] = QR(state[0], state[5],
state[10], state[15])  # diagonal 1 (main diagonal)
        state[1], state[6], state[11], state[12] = QR(state[1], state[6],
state[11], state[12])  # diagonal 2
        state[2], state[7], state[8], state[13] = QR(state[2], state[7],
state[8], state[13])  # diagonal 3
        state[3], state[4], state[9], state[14] = QR(state[3], state[4],
state[9], state[14])  # diagonal 4


    output = []
    for i in range(16):
        output[i] = state[i] + x[i]

    return output
```

Figure 4.4 Example definition of ChaCha20 encryption/decryption algorithm using the PyFrocen Framework.

```
def ROTL(a, b):

    aux = (a << b) | (a >> (32 - b))
    return aux[0:32]


def QR(a, b, c, d):

    a = a + b
    d = d ^ a
    d = ROTL(d, 16)

    c = c + d
    b = b ^ c
    b = ROTL(b, 12)

    a = a + b
    d = d ^ a
    d = ROTL(d, 8)

    c = c + d
    b = b ^ c
    b = ROTL(b, 7)

    return a, b, c, d
```

Figure 4.5 Auxiliar functions used to perform the Chacha20 encryption/decyption

Although the PyFrocen framework eases the function definition, common SMC design patterns must still be considered. For example, since private data would not be known even in runtime, loop limits cannot depend on these data and a conservative upper limit must be used instead.

4.4.2 Circuit synthesis

The target function is then synthetized in a set of Boolean circuits, each with different size and depth parameters. Figure 4.6 shows how to perform this process using the PyFrocen framework. The first two lines are for importing the framework itself and the target function that is to be synthetized (i.e., the function shown in Figure 4.4). Then the framework is initialized by instantiating the “frozen” class and the function to be evaluated is selected using the “add_function” subroutine. Next, the synthesis process starts by calling the “build” function, which initiates a circuit object that contains all the logical gates in the circuit and calls to the target function after that. The target function is decorated by the “frocen_function” decorator, which runs as an entry point. It also generates an instance of the new defined class, Num, for

each input parameter in the target function using the information (size and bit wide) specified in the decorator definition. This new class internally contains references to each circuit wire representing this variable. The information on which wires represent each input and which party would provide each of them is registered in the circuit object. Next, the evaluation of the target function continues using these Num instances provided by the decorator as inputs.

```
from frocen import *
from chacha20 import chacha20

frocen = Frocen()
frocen.add_function(chacha20)
frocen.build(3)
frocen.save_circuit("chacha20.scd")
```

Figure 4.6 Main code used to synthetize a previously defined function into a set of Boolean circuits in PyFrocen

As the operators in the Num class have been overloaded, executing an operation between an instance of this Num class and a primitive type or two Num instances adds the required logical gates to the circuit object, and returns a new Num instance pointing to output wires. When the evaluation of the function ends, the decorator captures the output, which is a Num instance, and registers the information about the output (which circuit wires form the output and which parties should receive it) in the circuit object. Finally, information in the circuit object is serialized following the TinyGarble simple circuit description (SCD) format [95].

The whole build process is repeated to generate several circuit representations. These representations can be obtained by using a variety of building blocks to overload the Num class operators. For example, in the case of the sum operator, the addition can be performed using the previously presented ripple-carry or Sklansky adder topologies, thus resulting in circuits with different depth and size parameters. For the above-shown ChaCha20 decryption algorithm, we have synthesized three circuit representations.

4.5 Model generation

As already seen, any function can be synthetized in circuit representations, each with their own size and depth parameters. Low-depth circuits will allow the function to be evaluated quickly, introducing very low latency, while lower size circuits will allow higher network loads to be

processed, at the cost of increasing the latency introduced. Thus, a model can be generated to assist the middlebox in selecting the most suited circuit representation to perform each evaluation depending on the network state. The maximum network load (BW_{useful}) that could be processed using a given circuit representation, Cx, would be limited by the communication and computation resources required to perform the associated evaluations ($BW_{SNF}\ and\ CPU_{SNF}$) and the resources available in the specific scenario ($BW_{available}\ and\ CPU_{available}$) as they must meet Equations (4.1) and (4.2).

$$BW_{SNF}(BW_{useful}, C_x) < BW_{available} \quad \textbf{(4.1)}$$

$$CPU_{SNF}(BW_{useful}, C_x) < CPU_{available} \quad \textbf{(4.2)}$$

To illustrate this, we can calculate a simplistic model for the previously shown ChaCha20 decryption algorithm and its three circuit representations considering only communication resources. For each circuit representation, we must firstly determine the overhead factor, representing the ratio between the bandwidth consumed for any single SNF evaluation (BW_{SNF}) and the bandwidth processed by the evaluation (BW_{useful}). Obtaining it is quite simple for the ChaCha20 decryption algorithm. Each circuit evaluation would provide a 512-bit output, which would be the BW_{util}. However, resources required to perform the circuit evaluation would depend on the specific SMC protocol used to evaluate the circuit. When using the GMW protocol with multiplication triples, each party only has to communicate shares for its own inputs, shares for the circuit output and an average of two bits per non-linear gate as stated by the authors of [96] if using their proposed OT extension. Table 4.1 shows the size, depth, input size and output size parameters for the three above-described circuit representations. This table also shows the BW_{useful} and BW_{SNF} in terms of bits per evaluation and the overhead ratio (BW_{SNF}/BW_{useful}).

Table 4.1 Characteristic parameters of three Chacha20 circuit representations

Circuit	Size	Depth	Input	Output	BW_{useful}	BW_{SNF}	BW_{SNF}/BW_{useful}
Sklansky	53724	466	384	512	512	108344	211,61
Mixed	24769	1131	384	512	512	50434	98,5
Ripple carry	10736	1731	384	512	512	22368	43,68

Figure 4.7 shows the required bandwidth to perform the SNF depending on the network load (bandwidth useful) for the three previously defined circuits. Both these parameters are directly proportional, as they are related by the overhead ratio. The maximum load processable with each circuit representation can be found where the bandwidth required to perform the SNF is equal to the resources available (black line).

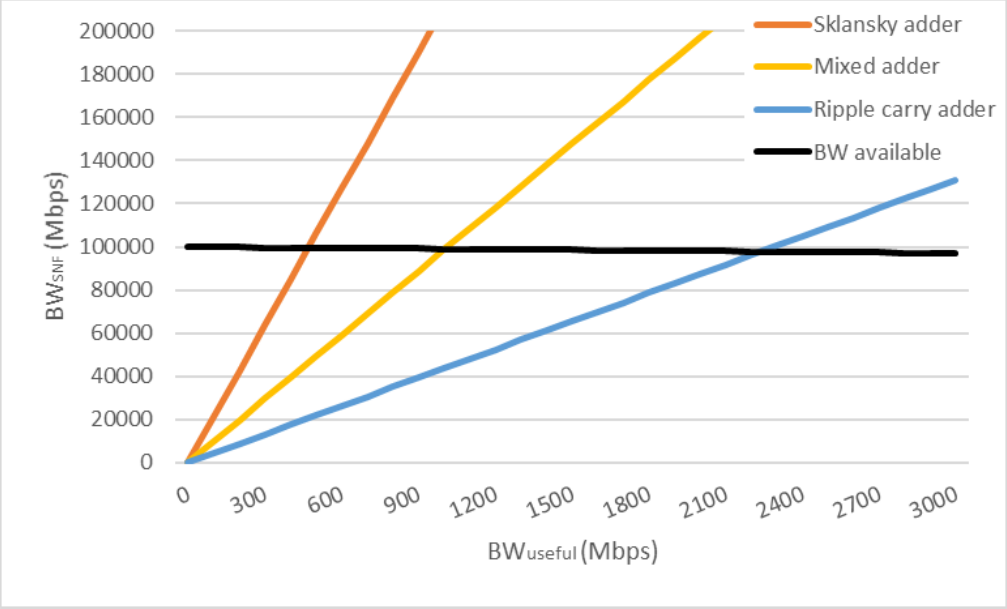

Figure 4.7 Bandwidth consumption by the SNF.

In a scenario in which a bandwidth of 100 Gbps is available, the maximum network load processable using the Sklansky, mixed and ripple-carry circuits would be 420 Mbps, 1Gbps and 2.2 Gbps, respectively. Thus, to provide optimal latency, the Sklansky adder-based circuit would be used when the network load remains under 473 Mbps. If the network load exceeds this value, the mixed-adder circuits would be used until the load reached 1 Gbps, which is when the ripple-carry adder-based circuits would be used instead. If the network load surpasses 2.2 Gbps, some packets would start to be dropped as there would not be enough resources to process them.

This model is quite ideal as it only considers communication resources and does not encompass computational ones. Generating a more complete and real model is not as straightforward as the previous one, which only considers communication resources, as performance would be

determined by several factors (CPU architecture, size of cache memory, latency between parties, etc.). Moreover, the latency introduced by the SNF execution would not only depend on the circuit used to perform the evaluation, but also on other configuration parameters, such as the ESs or maximum concurrent evaluations. Thus, the real model would be generated empirically by performing several circuit evaluations using synthetic inputs and various configurations (i.e., circuit representation, evaluation set size, maximum concurrent evaluation sets). Each configuration's performance is measured in terms of the maximum network load that can be evaluated and the latency introduced by the SNF. This model would be used by the middlebox at evaluation time to decide the optimal configuration that should be used to evaluate each evaluation set depending on the network state, so that the configuration minimizing the latency is selected for each network load value.

4.6 Results and discussion

To proposed method has been tested by the above-shown ChaCha20 encryption algorithm along its three circuit representations. This is interesting since secure decryption would be the first step in any middlebox for applying its functionality over encrypted traffic, and ChaCha20 is currently the most extended encryption scheme as it is the preferred one in TLS 1.3. Performance tests have been run on Azure using one HC-series virtual machine for each party. These instances have 44 vCPU each, 352 GB of system memory and 100 Gbps of network bandwidth through an InfiniBand adapter.

The following results have been obtained by running the online phase of each evaluation using synthetic inputs and varying configuration parameters. The evaluation set size (ESs) has been varied from 1 to 40, the maximum concurrent evaluations (MCE) from 1 to 10 and the number of active cores (CPUs) from 1 to 40 in different runs. The achievable bandwidth and the latency introduced by the evaluation have been obtained for each configuration, performing a total of 20000 circuit evaluations per configuration to ensure the reliability of the results.

The minimum latency introduced by the SNF based on network load has been obtained for the three above-presented circuits and it is shown in Figure 4.8. These values are obtained for optimal configuration in terms of ESs, MCE and CPUs. We can achieve very low latency values

(even lower than 3 ms) using the Sklansky adder-based circuit representation when the network load remains low enough. At the other end, we can process up 1.89 Gbps using the ripple-carry adder-based circuit at the cost of increasing the latency introduced. The latency introduced by the SNF is highly dependent on the depth circuit representation used, given that a communication round between parties is required on each circuit layer. Thus, the difference between latencies introduced by each circuit representation would be more evident as the network latency between parties (middlebox and host) increases.

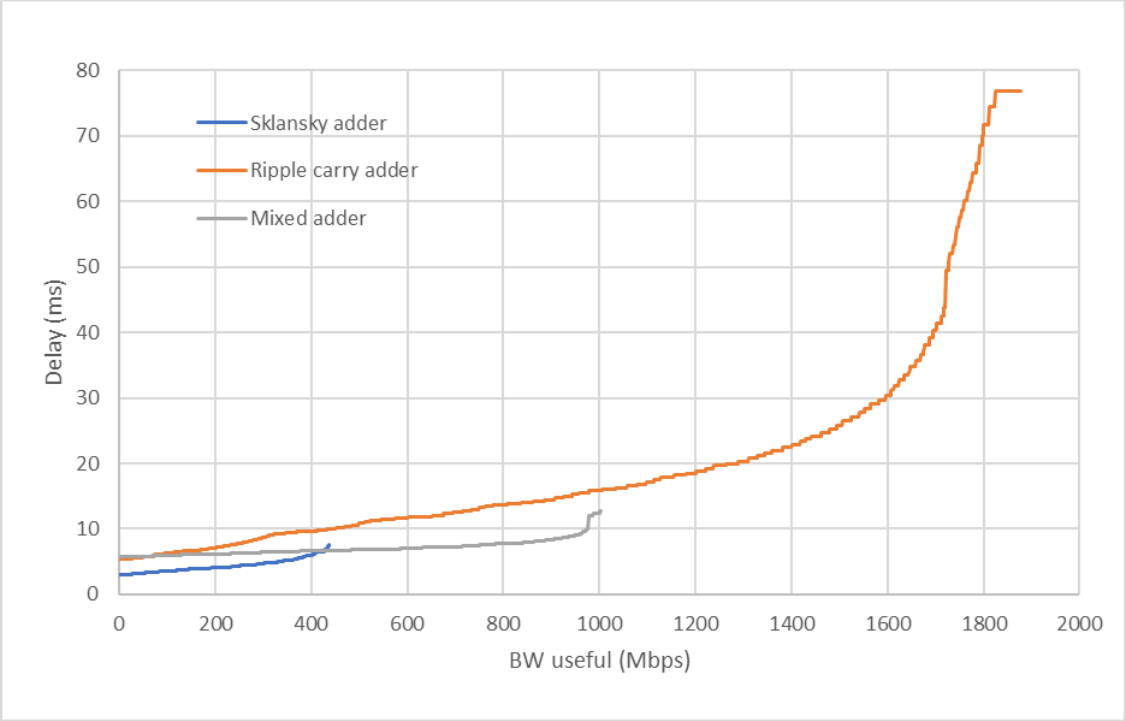

Figure 4.8 Latency introduced by the SNF.

As we can see in Figure 4.8, the maximum network load processable using the Sklansky and mixed circuits is very close to the load predicted by the model in the previous section (i.e., 438 Mbps and 1 Gbps, respectively). It means the computational resources are not a limitation here and the communication ones, considered by this previous model, restrict the maximum processable network load. However, for the ripple-carry adder circuit, the maximum processable load is slightly under the load predicted by the previous model (achievable 1.89 Gbps compared with the predicted 2.2 Gbps). That means computational resources may be limiting performance.

The above assumption can be confirmed by Figure 4.9, which shows the maximum network load that can be processed for each circuit representation depending on the number of active cores. As the figure reveals, the maximum processable network using Sklansky and mixed adders remains stable from 15 and 25 cores, respectively, when the communication resources start to be exhausted. For the ripple-carry circuit, however, the maximum processable network load continues to increase even using 40 cores.

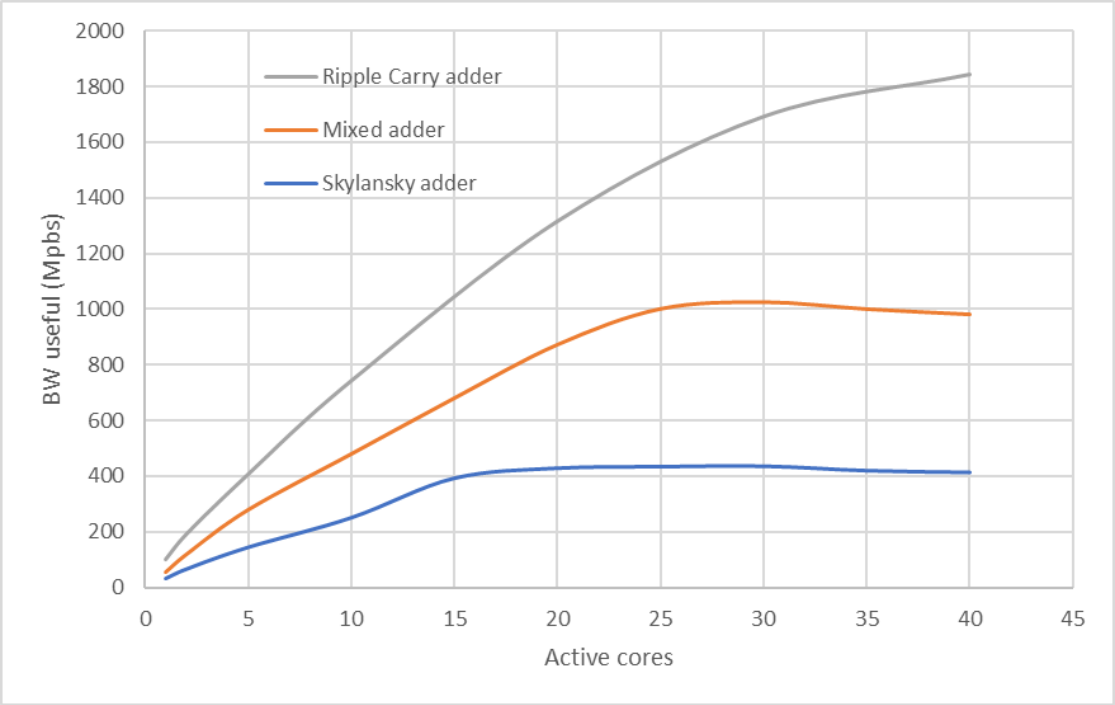

Figure 4.9 Maximum processable bandwidth using different number of CPUs

Figure 4.10 shows the ChaCha20 model, which comprises the latency introduced by the SNF depending on the network load and the configuration parameters (circuit representation, cores, ESs and MCE) that should be used to obtain these optimal latency values. As the network load increases, computational resources start to be exhausted and the only way to keep expanding the bandwidth is to increase the size of evaluation sets, causing latency to increase dramatically. As a consequence, there is a region where latency quickly rises to obtain a marginal improvement in the amount of processable traffic (e.g., changing from processing 1.73 to 1.89 Gbps causes the latency introduced to increase from 40 to 80 ms). These working points would not be desirable for most applications; however, a comprehensive study of each scenario would be required to determine the extent to which the trade-off (bandwidth-latency) is still worth it.

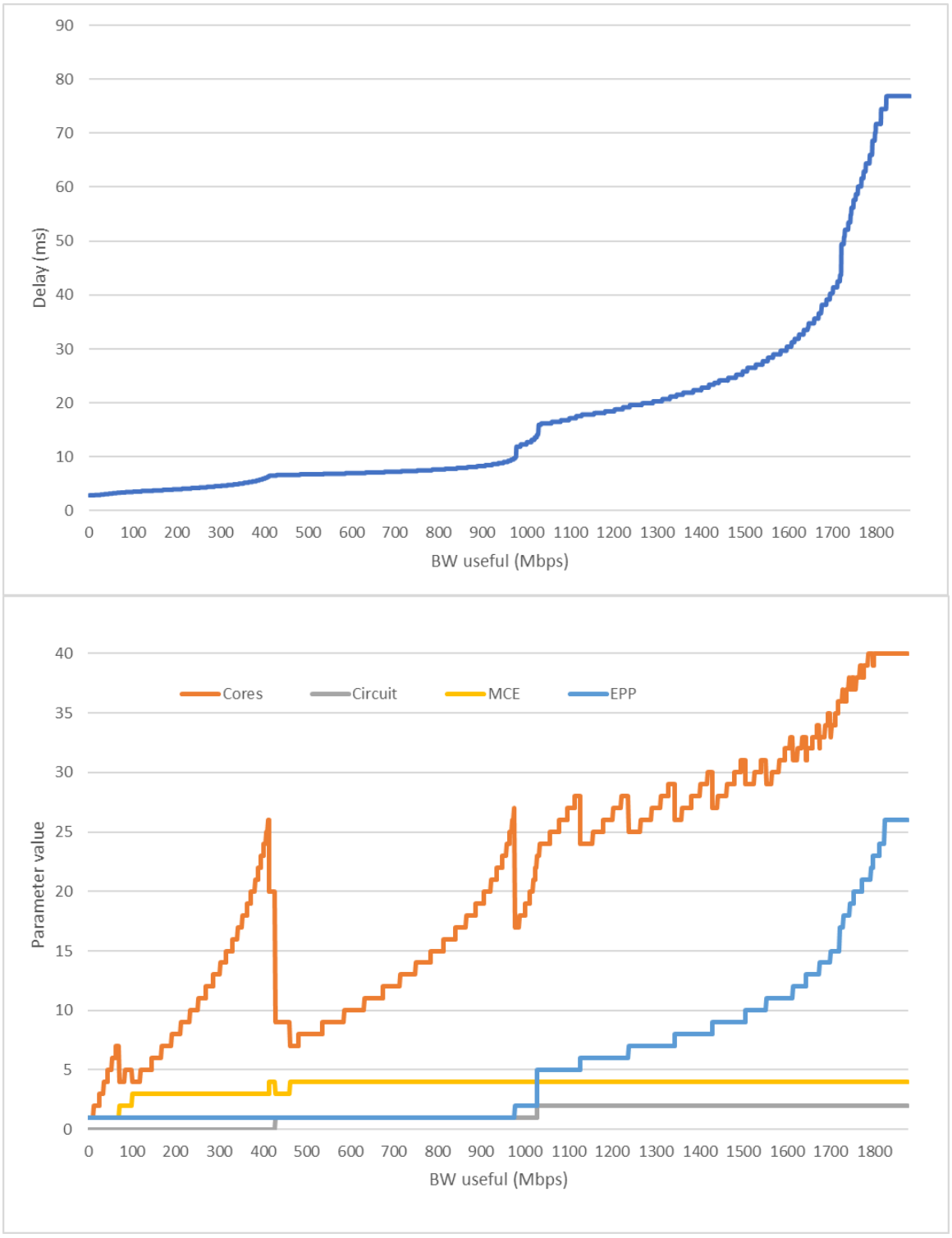

Figure 4.10 Chacha20 model for the test scenario

Comparisons with other works may prove complicated. However, this approach clearly outperforms a preliminary study we presented [96] in which garbled circuits are used to provide privacy to the packet inspection. Garbled circuits is a protocol in the computational setting and,

as discussed in the introduction, protocols in this setting may not be suitable for taking advantage of the intrinsic characteristics of network scenarios. First, using garbled circuits introduces a far greater bandwidth overhead than the one introduced by the GMW protocol. In the case of the presented ChaCha20 decryption, the bandwidth overhead factor introduced by garbled circuits would be around 5500 assuming a security parameter of 128 and the use of the ripple-carry adder circuit (which is size optimal). This overhead is much greater than the worst case for the presented results, in which the Sklansky circuit introduces an overhead factor of 256. The proposed version is also more efficient in terms of computational resources. The approach shown in [96] can decrypt up to 160 Kbps using a single CPU core, while we can decrypt up to 90 Mbps using a single core with this new approach in our tests. Finally, the proposed approach is very flexible as various configurations can be used to accommodate the evaluations of the network state, thus providing the best possible latencies. It could also prove interesting to treat various traffic types differently, guaranteeing the agreed quality of service (QoS) to each flow.

Although resources required to enable SNF might still be unaffordable for the general public, its use could be justified by the sensitivity of the information involved in many scenarios and applications. Moreover, some applications can be evaluated very efficiently as only some packet fragments need to be inspected to make the forwarding decision. This would be the case of an institution that only grants access to its protected resources to clients connected to the intranet using their VPN service. The VPN encapsulates the entire original packet so that no information is available for the firewall to determine whether forwarding the packet may suppose a security risk or not. To enable the firewall to perform its function, understanding it as classic packet filtering depending on the original packet's headers (which are encapsulated as the VPN payload), only some packet chunks must be decrypted and inspected (see Figure 4.11).

Figure 4.11 Example VPN packet

4.7 Conclusion

This chapter shows how SMC can be used to address the confrontation between system security and data privacy at domain boundaries. The work presented here offers a novel approach based on an adaptative circuit representation to follow the changes in the network state, thus providing the best performance at any given time. To that end, an optimized implementation of the GMW protocol has been developed to take advantage of the concurrency that is intrinsic to SNF applications.

The proposed inspection method is associated with some key benefits. First, it can work over a standard implementation of TLS with minimal modifications on the server side being completely seamless for clients, which eases its deployment in real-world scenarios. Second, as it is built on top of a general-purpose SMC protocol, it could be used to provide any functionality (at the cost of performance); therefore, it could be used to inspect not only plain text, but also structured data, thus enabling a wide spectrum of applications. It could also keep track of the connection state at an application level, which is crucial for inspecting new-generation protocols such as HTTP/2, which, in turn, is already replacing the previous version due to its performance improvement. Finally, with the proposed method, the middlebox can ascertain whether a given packet matches the defined rules before forwarding it without relying on any other party.

This chapter has presented an implementation of the ChaCha20 encryption/decryption algorithm and its performance results. The authors, based on these performance results, conclude that these promising privacy-preserving methods are close to debuting in real-world applications and thorough analyzes of each specific application are required (i.e., bandwidth and latency between parties, functionality required, etc.). Research on SMC-friendly cryptographic algorithms and information serialization formats is still needed to further improve this performance. These measures are expected to reduce performance degradation significantly, thus leading to the widespread adoption of these solutions.

In general, this work shows how multiparty computation could be used to address the confrontation between system security and data privacy at domain boundaries. The proposed inspection method would help keep private information away from the eyes of curious network administrators or from intruders by taking control of the firewall (which is usually facing the Internet) without preventing the firewall from providing its crucial functionality of keeping systems secure.

Chapter 5: Discussion and Conclusions

This last chapter draws conclusions of the thesis and outlines future work. The underlying hypotheses and research objectives presented in the first chapter have been raised and discussed throughout this thesis. The most challenging key factors in information security architectures have been addressed; these include usable user authentication mechanisms, usable and private authorization protocols, and privacy-preserving border protection functions. In this last chapter, the main outcomes are discussed, and the final conclusions presented to establish this work's contributions. Finally, future challenges are listed as this is a growing research field with new challenges arising as society and technology evolve.

5.1 Research Objectives Achieved

Chapter 1 describes the reasons and evidence for the relevance given to appropriate information management and use for both companies and users. Section 1.1 introduces the security architecture concepts, describing its core components. The objectives of this thesis are established in Section 1.2, particularly, research on information security architectures from usability and privacy perspectives, which is the main objective of this work. All the objectives outlined in Chapter 1 are addressed throughout the remaining chapters.

- The design and evaluation of biometric authentication approaches using the PPG are presented in Chapter 2.

- The design and evaluation of authorization protocols along with policy evaluation mechanisms to improve the functionality, usability, and privacy of the API security panorama from several perspectives are addressed in Chapter 3.

- The design and evaluation of protocols to efficiently enable SNF, improving privacy in border protection devices is addressed in Chapter 4.

The detailed conclusions and objectives achieved in each section and chapter are listed in Sections 2.5, 3.2.4, 3.3.5, 3.4.5 and 4.7.

5.2 Contributions and accomplished results

The major contribution of this thesis is to provide information security architectures with novel authentication, authorization and border protection mechanisms. The most challenging issues have been addressed and efficient solutions have been proposed. The work involved in achieving this main objective has also resulted in several minor contributions. They are presented below, subdivided into the main topics of this research. First, in the field of **PPG biometrics**:

- A methodology to perform and evaluate biometric authentication using PPG signals. Several approaches have been evaluated to perform preprocessing, feature extraction and matching steps. In contrast with previous literature, which uses signals from the same recording session for both enrollment and testing stages, the proposed evaluation method is the first to consider the long-term results, in which a time gap exists between signals used for enrollment and testing stages.

- Two long-term PPG databases. Each database includes PPG signals for the same 24 users, acquired using two pulse oximeters: a high-end Nonin pulse oximeter and a low-cost BerryMed oximeter. Both databases include three records per user, acquired with a time lapse of one to seven days.

Second, concerning **API security**:

- A protocol to seamlessly re-authorize clients using biometric signals gathered from users' wearable devices (SeamAuth). The SeanAuth involves two subprotocols; the interactive pre-authorization where the user consents to the client application accessing their protected data and the just-in-time authorization that seamlessly verifies the user identity (using fresh biometric signals) on each access attempt to grant the client access.

- A protocol to authorize third-party applications when they are accessed by users through a messaging platform (MAuth). This novel protocol allows the authorization server to obtain users' consent directly through the same interface (i.e., messaging platform) they are already using to interact with the third-party application. This protocol has been designed under the umbrella of the OAuth 2.0 framework and, by design, specifically addresses the man-in-the-middle problem intrinsic to all messaging platforms relying on the TextSecure protocol.

- A framework to enable user-to-user delegation (U2UAuth). This includes a new OAuth 2.0 grant type for interaction between the authorization server and both users (requesting party and resource owner), and an OPA profile to deal with policy definition and evaluation. An interface for the communication between the proposed OPA profile and the OAuth framework has also been designed.

Third, as regards to **Border protection**:

- A network setup and a protocol to efficiently enable SNF relying on SMC protocols. The proposed protocol takes advantage of parallelisms present in this scenario, where one protocol evaluation per packet is required. This protocol allows several packets to be evaluated concurrently, and evaluations are adapted depending on the network state to optimize the performance achievable with available resources.

- A framework to ease the generation of several circuit representations of the SNF to be evaluated and a tool to generate the model used to determine the optimal configuration under various network conditions.

These contributions have been demonstrated in Information Security Architectures, leading to the following accomplished results:

- For biometric authentication using PPG signals, the results push the EER down to 1% in the short term when using signals from the PRRB database. It outperforms most existing methods in the literature, asserting the validity of the proposed method. In the

long term, FMR and FNMR curves present time stability and the threshold value at which the EER appears remains stable as time passes. However, these error values quickly rise (from ≈6% to ≈20%) when the time between enrollment and testing stages increases up to one day and stays constant from there.

- The SeamAuth protocol aligns security and usability in access delegation to third-party applications. This protocol allows to use short-living access tokens, which prevent the major consequences of access token leakage without hampering the user experience, as the re-authorization process does not require any kind of user interaction.

- The MAuth protocol improves both security and usability in the third-party authorization process when the application is accessed through a messaging platform. From a security perspective, it prevents the TextSecure MitM attacker out-of-the-box, as well as most issues related to the standard authorization code grant type (completely or partially). From a usability viewpoint, most users rated this new approach better than the alternative of launching the authorization by opening a web browser (scored on average with 8.52 and 7.74, respectively). Most users also perceive the MAuth protocol as more secure than its opponent.

- The U2UAuth framework provides support for user-to-user delegation directly on top of the OAuth 2.0 framework by decoupling the requesting party and the resource owner. Results show that, by using messaging platforms, the authorization server can receive the resource owner's authorization decision in under 10 minutes in roughly 80% of authorization attempts. Moreover, almost all access attempts can be resolved in the first hour. Aligning authentication and authorization tasks with users' daily routines improves overall access control usability and security while giving back users the control of their data.

- Regarding the SNF evaluation, we have shown that the proposed protocol can perform traffic decryption at wire rate using the ChaCha20 algorithm. The proposed adaptative circuit approach is flexible enough to decrypt traffic up to 1.89 Gbps using a circuit representation based on ripple-carry adder while it can introduce a latency of just a few

milliseconds (less than 3 ms) using the Sklansky adder-based circuit if the network load is low enough.

5.3 Future Work

Although the objectives have been fulfilled and very promising results have been obtained, more exhaustive experimentation would be welcome to contrast the results provided in this thesis.

- Evaluating the proposed PPG biometric methodology with a larger database. It should include a higher number of recording sessions with time gaps between them to further prove time stability and a larger population to generalize the results.

- Testing the SeamAuth, MAuth and U2UAuth protocols with some already deployed applications to obtain more realistic user impressions, by measuring the value of the proposed solutions in terms of whether users move to use them or not.

- Including OT generation (with silent preprocessing) in the proposed SNF evaluation to obtain more complete results. These required OTs can be efficiently bulk generated in GPUs given they are not coupled to a specific evaluation.

- Designing more network functions using the PyFrocen framework and evaluating its viability for use in real-world scenarios.

Bibliography

[1] N. Rerup and M. Aslaner, *Hands-On Cybersecurity for Architects: Plan and design robust security architectures*. Packt Publishing Ltd, 2018.

[2] M. E. Whitman, “Principles of Information Security Fourth Edition,” 2011.

[3] L. O’Gorman, “Comparing passwords, tokens, and biometrics for user authentication,” in *Proceedings of the IEEE*, 2003, vol. 91, no. 12, pp. 2021–2040, doi: 10.1109/JPROC.2003.819611.

[4] A. Teoh, B. Jin, D. Ngo, C. Ling, and A. Goh, “Biohashing: two factor authentication featuring fingerprint data and tokenised random number,” *Pattern Recognition*, vol. 37, pp. 2245–2255, 2004, doi: 10.1016/j.patcog.2004.04.011.

[5] S. Na and S. Cheon, “Role Delegation in Role-Based Access Control,” 2000.

[6] J. Wainer and A. Kumar, “A Fine-grained, Controllable, User-to-User Delegation Method in RBAC,” 2005.

[7] H. S. Gardiyawasam Pussewalage and V. A. Oleshchuk, “Attribute based access control scheme with controlled access delegation for collaborative E-health environments,” *Journal of Information Security and Applications*, vol. 37, pp. 50–64, 2017, doi: 10.1016/j.jisa.2017.10.004.

[8] J. Crampton and H. Khambhammettu, “Delegation in role-based access control,” *Int. J. Inf. Secur.*, vol. 7, pp. 123–136, 2008, doi: 10.1007/s10207-007-0044-8.

[9] C. N. Chong, Z. Peng, and P. H. Hartel, “Secure Audit Logging with Tamper-Resistant Hardware,” in *Security and Privacy in the Age of Uncertainty*, 2003, pp. 73–84.

[10] A. Adriansyah, B. F. van Dongen, and N. Zannone, “Controlling break-the-glass through alignment,” in *Proceedings - SocialCom/PASSAT/BigData/EconCom/BioMedCom 2013*, 2013, pp. 606–611, doi: 10.1109/SocialCom.2013.91.

[11] S. Frankel and S. Krishnan, “RFC6071: IP Security (IPsec) and Internet Key Exchange (IKE) Document Roadmap,” *Request for Comments, IETF*, 2011.

[12] "RFC 8446 - The Transport Layer Security (TLS) Protocol Version 1.3." https://tools.ietf.org/html/rfc8446 (accessed Jan. 07, 2021).

[13] "RFC 7516 - JSON Web Encryption (JWE)." https://tools.ietf.org/html/rfc7516 (accessed Jan. 07, 2021).

[14] "RFC 7515 - JSON Web Signature (JWS)." https://tools.ietf.org/html/rfc7515 (accessed Jan. 07, 2021).

[15] "Data Protection: Data In transit vs. Data At Rest | Digital Guardian." https://digitalguardian.com/blog/data-protection-data-in-transit-vs-data-at-rest (accessed Jan. 07, 2021).

[16] K. Scarfone, M. Souppaya, and M. Sexton, "Special Publication 800-111 Guide to Storage Encryption Technologies for End User Devices Recommendations of the National Institute of Standards and Technology." doi: 10.6028/NIST.SP.800-111.

[17] "Solutions Guide for Data-At-Rest SSIF Solutions Guide for Data-At-Rest-2-SSIF Guide to Data-At-Rest Solutions."

[18] "openBoM and the information sharing paradigm shift | by OpenBOM (openbom.com) | Medium." https://medium.com/@openbom/openbom-and-the-information-sharing-paradigm-shift-68b2b6eaef08 (accessed Jan. 07, 2021).

[19] "Beyond trust: Why we need a paradigm shift in data-sharing | World Economic Forum." https://www.weforum.org/agenda/2020/01/new-paradigm-data-sharing/ (accessed Jan. 07, 2021).

[20] N. Isakadze and S. S. Martin, "How useful is the smartwatch ECG?," *Trends in Cardiovascular Medicine*, vol. 30, no. 7. Elsevier Inc., pp. 442–448, Oct. 01, 2020, doi: 10.1016/j.tcm.2019.10.010.

[21] "Healthcare - Apple Watch - Apple." https://www.apple.com/healthcare/apple-watch/ (accessed Jan. 07, 2021).

[22] M. v. Perez *et al.*, "Large-Scale Assessment of a Smartwatch to Identify Atrial Fibrillation," *New England Journal of Medicine*, vol. 381, no. 20, pp. 1909–1917, Nov. 2019, doi: 10.1056/nejmoa1901183.

[23] "Using Polar technology in research | Polar Global." https://www.polar.com/en/science/research-tools (accessed Jan. 07, 2021).

[24] D. Hernando, S. Roca, J. Sancho, Á. Alesanco, and R. Bailón, "Validation of the apple watch for heart rate variability measurements during relax and mental stress in healthy subjects," *Sensors (Switzerland)*, vol. 18, no. 8, 2018, doi: 10.3390/s18082619.

[25] I. Damgård, M. Geisler, and M. Krøigaard, "Efficient and secure comparison for on-line auctions," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2007, vol. 4586 LNCS, pp. 416–430, doi: 10.1007/978-3-540-73458-1_30.

[26] P. Bogetoft, I. Damgård, T. Jakobsen, K. Nielsen, J. Pagter, and T. Toft, "A practical implementation of secure auctions based on multiparty integer computation," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2006, vol. 4107 LNCS, pp. 142–147, doi: 10.1007/11889663_10.

[27] A. K. Jain, A. Ross, and S. Pankanti, "Biometrics: A tool for information security," *IEEE Transactions on Information Forensics and Security*, vol. 1, no. 2, pp. 125–143, Jun. 2006, doi: 10.1109/TIFS.2006.873653.

[28] N. K. Ratha, J. H. Connell, and R. M. Bolle, "Enhancing security and privacy in biometrics-based authentication systems," *IBM Systems Journal*, vol. 40, no. 3, pp. 614–634, 2001, doi: 10.1147/sj.403.0614.

[29] A. K. Jain, S. Pankanti, S. Prabhakar, H. Lin, and A. Ross, "Biometrics: A grand challenge," in *Proceedings - International Conference on Pattern Recognition*, 2004, vol. 2, pp. 935–942, doi: 10.1109/ICPR.2004.1334413.

[30] P. Tuyls, A. H. M. Akkermans, T. A. M. Kevenaar, G. J. Schrijen, A. M. Bazen, and R. N. J. Veldhuis, "Practical biometric authentication with template protection," in *Lecture Notes in Computer Science*, 2005, vol. 3546, pp. 436–446, doi: 10.1007/11527923_45.

[31] S. Pankanti, R. M. Bolle, and A. Jain, "Biometrics: The future of identification," *Computer*, vol. 33, no. 2, pp. 46–49, Feb. 2000, doi: 10.1109/2.820038.

[32] A. K. Jain, A. Ross, and S. Prabhakar, "An Introduction to Biometric Recognition," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, no. 1, pp. 4–20, Jan. 2004, doi: 10.1109/TCSVT.2003.818349.

[33] I. Odinaka, P. H. Lai, A. D. Kaplan, J. A. O'Sullivan, E. J. Sirevaag, and J. W. Rohrbaugh, "ECG biometric recognition: A comparative analysis," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 6, pp. 1812–1824, 2012, doi: 10.1109/TIFS.2012.2215324.

[34] P. J. Phillips, A. Martin, C. L. Wilson, and M. Przybocki, “An Introduction to evaluating biometric systems,” *Computer*, vol. 33, no. 2, pp. 56–63, 2000, doi: 10.1109/2.820040.

[35] W. Louis, M. Komeili, and D. Hatzinakos, “Continuous authentication using One-Dimensional Multi-Resolution Local Binary Patterns (1DMRLBP) in ECG biometrics,” *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 12, pp. 2818–2832, Dec. 2016, doi: 10.1109/TIFS.2016.2599270.

[36] S. Peter, B. Pratap Reddy, F. Momtaz, and T. Givargis, “Design of Secure ECG-Based Biometric Authentication in Body Area Sensor Networks,” *Sensors*, vol. 16, no. 4, p. 570, Apr. 2016, doi: 10.3390/s16040570.

[37] I. Jekova, V. Krasteva, and R. Schmid, “Human Identification by Cross-Correlation and Pattern Matching of Personalized Heartbeat: Influence of ECG Leads and Reference Database Size,” *Sensors*, vol. 18, no. 2, p. 372, Jan. 2018, doi: 10.3390/s18020372.

[38] J. Pinto, J. Cardoso, A. Lourenço, and C. Carreiras, “Towards a Continuous Biometric System Based on ECG Signals Acquired on the Steering Wheel,” *Sensors*, vol. 17, no. 10, p. 2228, Sep. 2017, doi: 10.3390/s17102228.

[39] R. Tan and M. Perkowski, “Toward Improving Electrocardiogram (ECG) Biometric Verification using Mobile Sensors: A Two-Stage Classifier Approach,” *Sensors*, vol. 17, no. 2, p. 410, Feb. 2017, doi: 10.3390/s17020410.

[40] S. Venugopalan, F. Juefei-Xu, B. Cowley, and M. Savvides, “Electromyograph and Keystroke Dynamics for Spoof-Resistant Biometric Authentication,” 2015. Accessed: Oct. 26, 2020. [Online].

[41] R. B. Paranjape, J. Mahovsky, L. Benedicenti, and Z. Koles, “The electroencephalogram as a biometric,” *Canadian Conference on Electrical and Computer Engineering*, vol. 2, pp. 1363–1366, 2001, doi: 10.1109/CCECE.2001.933649.

[42] Q. Wu, Y. Zeng, C. Zhang, L. Tong, and B. Yan, “An EEG-Based Person Authentication System with Open-Set Capability Combining Eye Blinking Signals,” *Sensors*, vol. 18, no. 2, p. 335, Jan. 2018, doi: 10.3390/s18020335.

[43] M. Elgendi, “On the Analysis of Fingertip Photoplethysmogram Signals,” *Current Cardiology Reviews*, vol. 8, no. 1, pp. 14–25, Jun. 2012, doi: 10.2174/157340312801215782.

[44] P. Spachos, J. Gao, and D. Hatzinakos, “Feasibility study of photoplethysmographic signals for biometric identification,” 2011, doi: 10.1109/ICDSP.2011.6004938.

[45] A. Bonissi, R. D. Labati, L. Perico, R. Sassi, F. Scotti, and L. Sparagino, “A preliminary study on continuous authentication methods for photoplethysmographic biometrics,” in *2013 IEEE Workshop on Biometric Measurements and Systems for Security and Medical Applications, BioMS 2013 - Proceedings*, 2013, pp. 28–33, doi: 10.1109/BIOMS.2013.6656145.

[46] N. Karimian, M. Tehranipoor, and D. Forte, “Non-fiducial PPG-based authentication for healthcare application,” in *2017 IEEE EMBS International Conference on Biomedical and Health Informatics, BHI 2017*, Apr. 2017, pp. 429–432, doi: 10.1109/BHI.2017.7897297.

[47] U. Yadav, S. N. Abbas, and D. Hatzinakos, “Evaluation of PPG biometrics for authentication in different states,” in *Proceedings - 2018 International Conference on Biometrics, ICB 2018*, Jul. 2018, pp. 277–282, doi: 10.1109/ICB2018.2018.00049.

[48] M. Nitzan, A. Babchenko, B. Khanokh, and D. Landau, “The variability of the photoplethysmographic signal - A potential method for the evaluation of the autonomic nervous system,” *Physiological Measurement*, vol. 19, no. 1, pp. 93–102, 1998, doi: 10.1088/0967-3334/19/1/008.

[49] J. C. Richards and S. Bertram, “Anxiety sensitivity, state and trait anxiety, and perception of change in sympathetic nervous system arousal,” *Journal of Anxiety Disorders*, vol. 14, no. 4, pp. 413–427, Jul. 2000, doi: 10.1016/S0887-6185(00)00031-1.

[50] L. Sörnmo and P. Laguna, *Bioelectrical signal processing in cardiac and neurological applications*, vol. 8. Academic Press, 2005.

[51] K. R. Rao and P. C. Yip, *The transform and data compression handbook*. CRC press, 2018.

[52] “CapnoBase.org: Pulse Oximeter IEEE TBME Benchmark.” http://www.capnobase.org/database/pulse-oximeter-ieee-tbme-benchmark/ (accessed Oct. 27, 2020).

[53] “The MIMIC II Waveform Database Matched Subset.” https://physionet.org/physiobank/database/mimic2wdb/matched/ (accessed Oct. 27, 2020).

[54] “eHealthz Github Site.” https://github.com/ehealthz/ppg_signals (accessed Oct. 27, 2020).

[55] "WristOx2® Model 3150 with USB | Nonin." https://www.nonin.com/products/3150-usb/ (accessed Oct. 27, 2020).

[56] "Finger Pulse Oximeter | Berry." https://www.shberrymed.com/finger-pulse-bm1000b-altitude-oximeter-p00012p1.html (accessed Oct. 27, 2020).

[57] J. Sancho, Á. Alesanco, and J. García, "Biometric authentication using the PPG: A long-term feasibility study," *Sensors (Switzerland)*, vol. 18, no. 5, 2018, doi: 10.3390/s18051525.

[58] "OAuth 2.0 — OAuth." https://oauth.net/2/ (accessed Oct. 23, 2020).

[59] X. Jin, R. Krishnan, and R. Sandhu, "A unified attribute-based access control model covering DAC, MAC and RBAC," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2012, vol. 7371 LNCS, pp. 41–55, doi: 10.1007/978-3-642-31540-4_4.

[60] "Open Policy Agent." https://www.openpolicyagent.org/ (accessed Oct. 23, 2020).

[61] "eXtensible Access Control Markup Language (XACML) Version 3.0." http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html (accessed Oct. 23, 2020).

[62] "Open Policy Agent | Language Reference." https://www.openpolicyagent.org/docs/v0.12.2/language-reference/ (accessed Jan. 07, 2021).

[63] "OpenID Connect | OpenID." https://openid.net/connect/ (accessed Oct. 23, 2020).

[64] "RFC 7519 - JSON Web Token (JWT)." https://tools.ietf.org/html/rfc7519 (accessed Jan. 07, 2021).

[65] "RFC 6749 - The OAuth 2.0 Authorization Framework." https://tools.ietf.org/html/rfc6749 (accessed Oct. 28, 2020).

[66] "OAuth 2.0 Security Best Current Practice." https://tools.ietf.org/id/draft-ietf-oauth-security-topics-14.html (accessed Jan. 07, 2021).

[67] "What is going on with OAuth 2.0? And why you should not use it for authentication. | by Damian Rusinek | SecuRing | Medium." https://medium.com/securing/what-is-going-on-with-oauth-2-0-and-why-you-should-not-use-it-for-authentication-5f47597b2611 (accessed Jan. 07, 2021).

[68] "Access Token Lifetime - OAuth 2.0 Simplified." https://www.oauth.com/oauth2-servers/access-tokens/access-token-lifetime/ (accessed Jan. 07, 2021).

[69] "FIDO2 Homepage." https://fidoalliance.org/fido2/ (accessed Oct. 27, 2020).

[70] I. Odinaka *et al.*, "ECG biometrics: A robust short-time frequency analysis," 2010, doi: 10.1109/WIFS.2010.5711466.

[71] S. Roca, J. Sancho, J. García, and Á. Alesanco, "Microservice chatbot architecture for chronic patient support," *Journal of Biomedical Informatics*, vol. 102, 2020, doi: 10.1016/j.jbi.2019.103305.

[72] T. Frosch, C. Mainka, C. Bader, F. Bergsma, J. Schwenk, and T. Holz, "How secure is TextSecure?," in *Proceedings - 2016 IEEE European Symposium on Security and Privacy, EURO S and P 2016*, May 2016, pp. 457–472, doi: 10.1109/EuroSP.2016.41.

[73] E. Vaziripour *et al., Is that you, Alice? A Usability Study of the Authentication Ceremony of Secure Messaging Applications*. 2017.

[74] E. Vaziripour *et al.*, "Action Needed! Helping Users Find and Complete the Authentication Ceremony in Signal," in *Proceedings of the Fourteenth Symposium on Usable Privacy and Security* , 2018, pp. 47–62, Accessed: Oct. 28, 2020. [Online]. Available: https://www.usenix.org/conference/soups2018/presentation/vaziripour.

[75] J. Wu *et al., "Something isn't secure, but I'm not sure how that translates into a problem": Promoting autonomy by designing for understanding in Signal*. 2019.

[76] S. Schröder, M. Huber, D. Wind, and C. Rottermanner, "When SIGNAL hits the Fan: On the Usability and Security of State-of-the-Art Secure Mobile Messaging," 2016, doi: 10.14722/eurousec.2016.23012.

[77] "draft-ietf-oauth-security-topics-15 - OAuth 2.0 Security Best Current Practice." https://tools.ietf.org/html/draft-ietf-oauth-security-topics-15 (accessed Oct. 28, 2020).

[78] "Signal >> Home." https://signal.org/en/ (accessed Oct. 28, 2020).

[79] A. Rosenfeld, S. Sina, D. Sarne, O. Avidov, and S. Kraus, "A Study of WhatsApp Usage Patterns and Prediction Models without Message Content," Feb. 2018, Accessed: Oct. 29, 2020. [Online]. Available: http://arxiv.org/abs/1802.03393.

[80] "RFC 7636 - Proof Key for Code Exchange by OAuth Public Clients." https://tools.ietf.org/html/rfc7636 (accessed Oct. 29, 2020).

[81] T. Radivilova, L. Kirichenko, D. Ageyev, M. Tawalbeh, and V. Bulakh, “Decrypting SSL/TLS traffic for hidden threats detection,” in *Proceedings of 2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies, DESSERT 2018*, Jul. 2018, pp. 143–146, doi: 10.1109/DESSERT.2018.8409116.

[82] D. Evans, V. Kolesnikov, M. Rosulek, and B.- Delft, “Pragmatic Introduction to Secure Multi-Party Computation,” NOW Publishers, 2018.

[83] Y. Ishai, J. Kilian, K. Nissim, and E. Petrank, “Extending oblivious transfers efficiently,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 2729, pp. 145–161, 2003, doi: 10.1007/978-3-540-45146-4_9.

[84] E. Boyle *et al.*, “Efficient two-round OT extension and silent non-interactive secure computation,” in *Proceedings of the ACM Conference on Computer and Communications Security*, Nov. 2019, pp. 291–308, doi: 10.1145/3319535.3354255.

[85] O. Goldreich, S. Micali, and A. Wigderson, “How to Play any Mental Game or A Completeness Theorem for Protocols with Honest Majority,” in *19th Annual ACM Symposium on Theory of Computing*, 1987, pp. 218–229.

[86] D. Beaver, “Efficient multiparty protocols using circuit randomization,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 1992, vol. 576 LNCS, pp. 420–432, doi: 10.1007/3-540-46766-1_34.

[87] J. Sherry, C. Lan, R. A. Popa, and S. Ratnasamy, “BlindBox: Deep Packet Inspection over Encrypted Traffic,” *Computer Communication Review*, vol. 45, no. 4, pp. 213–226, Aug. 2015, doi: 10.1145/2785956.2787502.

[88] X. Yuan, X. Wang, J. Lin, and C. Wang, “Privacy-preserving deep packet inspection in outsourced middleboxes,” in *Proceedings - IEEE INFOCOM*, Jul. 2016, vol. 2016-July, doi: 10.1109/INFOCOM.2016.7524526.

[89] C. Lan, J. Sherry, R. Ada Popa, S. Ratnasamy, and C. Lan Justine Sherry Raluca Ada Popa Sylvia Ratnasamy Zhi Liu, “Embark: Securely Outsourcing Middleboxes to the Cloud ,” in *Proceedings of the 13th USENIX Symposium on Networked Systems Design and Implementation (NSDI ’16).* , 2016, p. 255, Accessed: Oct. 26, 2020. [Online]. Available: https://www.usenix.org/conference/nsdi16/technical-sessions/presentation/lan.

[90] H. J. Asghar, L. Melis, C. Soldani, E. de Cristofaro, M. A. Kaafar, and L. Mathy, “SplitBox: Toward efficient private network function virtualization,” in *HotMiddlebox 2016 - Proceedings of the 2016 ACM SIGCOMM Workshop on Hot Topics in*

Middleboxes and Network Function Virtualization, Part of SIGCOMM 2016, Aug. 2016, pp. 7–13, doi: 10.1145/2940147.2940150.

[91] J. Zhou, Z. Cao, X. Dong, and A. v. Vasilakos, "Security and Privacy for Cloud-Based IoT: Challenges," *IEEE Communications Magazine*, vol. 55, no. 1, pp. 26–33, Jan. 2017, doi: 10.1109/MCOM.2017.1600363CM.

[92] R. Lu *et al.*, "PReFilter: An efficient privacy-preserving relay filtering scheme for delay tolerant networks," in *Proceedings - IEEE INFOCOM*, 2012, pp. 1395–1403, doi: 10.1109/INFCOM.2012.6195504.

[93] C. Wang, X. Yuan, Y. Cui, and K. Ren, "Toward Secure Outsourced Middlebox Services: Practices, Challenges, and beyond," *IEEE Network*, vol. 32, no. 1, pp. 166–171, Jan. 2018, doi: 10.1109/MNET.2017.1700060.

[94] "RFC 7539 - ChaCha20 and Poly1305 for IETF Protocols." https://tools.ietf.org/html/rfc7539 (accessed Jan. 07, 2021).

[95] E. M. Songhori, S. U. Hussain, A. R. Sadeghi, T. Schneider, and F. Koushanfar, "TinyGarble: Highly compressed and scalable sequential Garbled Circuits," in *Proceedings - IEEE Symposium on Security and Privacy*, Jul. 2015, vol. 2015-July, pp. 411–428, doi: 10.1109/SP.2015.32.

[96] J. Sancho, J. García, and Á. Alesanco, "Oblivious Inspection: On the Confrontation between System Security and Data Privacy at Domain Boundaries," *Security and Communication Networks*, vol. 2020, 2020, doi: 10.1155/2020/8856379.

[97] "Add Web Bluetooth as a policy-controlled feature (I44767f3f) · Gerrit Code Review." https://chromium-review.googlesource.com/c/chromium/src/+/657572 (accessed Oct. 28, 2020).

Appendix A: Authorization Code grant type

In this appendix we present the parameters used in the authorization code grant type defined in section 4.1 of the OAuth 2.0 standard [65]. The authorization code grant type is used to obtain both access tokens and refresh tokens and is optimized for confidential clients. Since this is a redirection-based flow, the client must be capable of interacting with the resource owner's user-agent (typically a web browser) and capable of receiving incoming requests (via redirection) from the authorization server. The authorization code grant type flow includes the following steps:

Authorization request

The client initiates the flow by directing the resource owner's user-agent to the authorization endpoint. The client includes its client identifier, requested scope, local state, and a redirection URI to which the authorization server will send the user-agent back once access is granted or denied (see Table A.1).

Table A.1 Authorization Code: Authorization request parameters

Parameter	Compulsoriness	Description
response_type	REQUIRED	Value MUST be set to "code".
client_id	REQUIRED	A string that uniquely identifies the client.
redirect_uri	OPTIONAL	A URI to redirect the user back after the authorization.
scope	OPTIONAL	A string describing the access rights requested by the client.
state	RECOMMENDED	An opaque value used by the client to maintain state between the request and callback. The authorization server includes this value when redirecting the user-agent back to the client. The parameter SHOULD be used for preventing cross-site request forgery.

End-User interaction

The authorization server authenticates the resource owner (via the user-agent) and establishes whether the resource owner grants or denies the client's access request.

Authorization response

If the resource owner grants access, the authorization server redirects the user-agent back to the client using the redirection URI provided earlier (in the request or during client registration). The redirection URI includes an authorization code and any local state provided by the client earlier (see Table A.2).

Table A.2 Authorization Code: Authorization response parameters

Parameter	Compulsoriness	Description
code	REQUIRED	The authorization code generated by the authorization server.
state	-	REQUIRED, if the “state” parameter was included in the client authorization request. The exact value received from the client.

If the request fails due to a missing, invalid, or mismatching redirection URI, or if the client identifier is missing or invalid, the authorization server inform the resource owner of the error. If the resource owner denies the access request or if the request fails for reasons other than a missing or invalid redirection URI, the authorization server redirects the user-agent back to the client using the redirection URI provided earlier (in the request or during client registration). The redirection URI includes error parameters to inform the client and any local state provided by the client earlier (see Table A.3).

Table A.3 Authorization Code: Authorization response error parameters

Parameter	Compulsoriness	Description
error	REQUIRED	A single ASCII [USASCII] error code from a list of defined codes, specifying the reason that has occasioned the error.
error_description	OPTIONAL	Human-readable ASCII [USASCII] text providing additional information, used to assist the client developer in understanding the error that occurred. Values for the "error_description" parameter MUST NOT include characters out-side the set %x20-21 / %x23-5B / %x5D-7E.
error_uri	OPTIONAL	A URI identifying a human-readable web page with information about the error, used to provide the client developer with additional information about the error. Values for the "error_uri" parameter MUST conform to the URI-reference syntax and thus MUST NOT include characters outside the set %x21 / %x23-5B / %x5D-7E.
state	-	REQUIRED, if the “state” parameter was included in the client authorization request. The exact value received from the client.

Token request

The client requests an access token from the authorization server's token endpoint by including the authorization code received in the previous step. When making the request, the client authenticates with the authorization server. The client includes the redirection URI used to obtain the authorization code for verification (see Table A.4).

Table A.4 Authorization Code: Token request parameters

Parameter	Compulsoriness	Description
grant_type	REQUIRED	Value MUST be set to "authorization_code".
code	REQUIRED	The authorization code received from the authorization server.
redirect_uri	-	REQUIRED, if the "redirect_uri" parameter was included in the authorization request, and their values MUST be identical.
client_id	-	REQUIRED, if the client is not authenticating with the authorization server as described in Section 3.2.1 of the standard.

Token response

The authorization server authenticates the client, validates the authorization code, and ensures that the redirection URI received matches the URI used to redirect the client in the authorization restponse. If valid, the authorization server responds back with an access token and, optionally, a refresh token (see Table A.5).

Table A.5 Authorization Code: Token response parameters

Parameter	Compulsoriness	Description
access_token	REQUIRED	The access token issued by the authorization server.
token_type	REQUIRED	The type of the token issued. Value is case insensitive.
expires_in	RECOMMENDED	The lifetime in seconds of the access token. For example, the value "3600" denotes that the access token will expire in one hour from the time the response was generated. If omitted, the authorization server SHOULD provide the expiration time via other means or document the default value.
refresh_token	OPTIONAL	The refresh token, which can be used to obtain new access tokens using the refresh token grant type.
scope	-	OPTIONAL, if identical to the scope requested by the client; otherwise, REQUIRED. The scope of the access token.

If the request client authentication failed or is invalid, the authorization server returns an error response (see Table A.6)

Table A.6 Authorization Code: Token response error parameters

Parameter	Compulsoriness	Description
error	REQUIRED	A single ASCII [USASCII] error code from a list of defined codes, specifying the reason that has occasioned the error.
error_description	OPTIONAL	Human-readable ASCII [USASCII] text providing additional information, used to assist the client developer in understanding the error that occurred. Values for the "error_description" parameter MUST NOT include characters out-side the set %x20-21 / %x23-5B / %x5D-7E.
error_uri	OPTIONAL	A URI identifying a human-readable web page with information about the error, used to provide the client developer with additional information about the error. Values for the "error_uri" parameter MUST conform to the URI-reference syntax and thus MUST NOT include characters outside the set %x21 / %x23-5B / %x5D-7E.