D4.1 Distributed Trust Management Framework - Intermediate version

Document Identification			
Status	Final	**Due Date**	31/01/2024
Version	1.0	**Submission Date**	31/01/2024

Related WP	WP4	**Document Reference**	D4.1
Related Deliverable(s)	D2.1, D2.2, D2.3	**Dissemination Level (*)**	PU
Lead Participant	VTT	**Lead Author**	Sami Lehtonen
Contributors	ATOS, UMU, CEA, QBE, UTH	**Reviewers**	Manos Panaousis, UOG
			Sofiane Lagraa, FUJ_LU

Keywords:
SSI, Seamless onboarding, behavioural authentication, side-channel attack

Document Information

List of Contributors	
Name	Partner
Valtteri Lipiäinen	VTT
Anni Karinsalo	VTT
Ross Little Armitt	ATOS
Nicolas Belleville	CEA
Niklas Palaghias	QBE
María Hernández	UMU
Jesús García	UMU
Apostolos Apostolaras	UTH
Ilias Syrigos	UTH

Document History			
Version	Date	Change editors	Changes
0.1	3/11/2023	VTT	ToC draft
0.2	22/12/2023	VTT, UMU, ATOS	Initial draft
0.2.1	05/01/2024	CEA	Draft
0.3	16/01/2024	VTT, UTH	Compiled and edited several contributions in chapter 2
0.4	19/01/2024	VTT	Internal review
1.0	31/01/2024	VTT	FINAL VERSION TO BE SUBMITTED

Quality Control		
Role	Who (Partner short name)	Approval Date
Deliverable leader	Sami Lehtonen (VTT)	30/01/2024
Quality manager	Jürgen Neises (FUJ_GE)	30/01/2024
Project Coordinator	Tomás Pariente Lobo (ATOS)	31/01/2024

Table of Contents

List of Tables

List of Figures

List of Acronyms

Abbreviation / acronym	Description
ABC	Attribute-Based Cryptography
DID	Decentralized Identity
Dx.y	Deliverable number y belonging to WP x
DSBA	Data Spaces Business Alliance
EBSI	European Blockchain Service Infrastructure
EC	European Commission
eIDAS	EU regulation for electronic IDentification, Authentication and trust Services
FE	Front End
GAIA-X	Federated and secure Data Infrastructure
IDSA	International Data Spaces Association
IoT	Internet of Things
JWT	JSON Web Tokens
MQTT	Message Queueing Telemetry Transport
NFC	Near Field Communication
OID4VP	OpenID Verifiable Presentation
OID4VCI	OpenID Verifiable Credential Issuance
OIDC	OpenID Connect
PII	Person Identifiable Information
SD	Selective Disclosure
SDK	Software Development Kit
SIOP	Self-Issued OpenID Provider
SSI	Self-Sovereign Identity
UI	User Inteface
VC	Verifiable Credential
VDR	Verifiable Data Registry
W3C	The World Wide Web Consortium
WP	Work Package
ZKP	Zero-Knowledge Proof

Executive Summary

This report describes the TANGO distributed trust management framework and its components developed in WP4. This is an intermediate version describing ongoing work carried out in different tasks of the aforementioned work package. Each task describes its component(s) and the current state achieved in a form of a demonstration.

First chapter introduces the work conducted in WP4 and the contents of this deliverable. Chapter two covers all demonstrators created in WP4 and its five subtasks. All subchapters follow common structure: component(s) description, demonstration description, support for pilots, and future work. First subchapter covers Self Sovereign Identity (Issuer, Verifier, Wallet), second subchapter covers Seamless Onboarding. Third and fourth subchapter cover Human Behavioural Authentication and Device Behavioural Authentication respectively. Last subchapter handles Hardening against Side-Channel Attacks.

Third chapter covers conclusions drawn from the current results of these demonstrations and next steps in the development of this trust management framework for TANGO. WP4 is still in progress and its components are pending integration into a common platform.

1 Introduction

1.1 Purpose of the document

The purpose of this document is to wrap describe all tools and demonstrations created in WP4 that together form the TANGO distributed trust framework. This work is ongoing and the presented version here is an intermediate milestone of the project. The document also covers features identified as missing and integration work.

1.2 Relation to other project work

These components are based on previous design and specification work conducted in WP2 and presented in the three deliverables D2.1, D2.2, and D2.3. They are the basis for the work in this work package.

Important input to this deliverable for shaping these solutions is based on:

- D2.2 User Needs and Requirements & Use Case Scenarios.
- D2.3 System Requirements and Specifications, Platform Architecture, and Privacy, Ethical, Social and Legal Impact Assessment.
- Architecture discussions to align with GAIA-X and IDSA and therefore analysis of different IDSA Connectors and their fit with TANGO.

The necessary components developed in this WP4, will in turn provide necessary functionality and services for the TANGO platform and the pilots later in the project.

1.3 Structure of the document

The second chapter goes through each task of the WP4 and components they provide for the framework. It covers all demonstrators created in WP4 in its five subtasks respectively. All subchapters follow common structure: component(s) description, demonstration description, support for pilots, and future work.

The third chapter covers conclusions drawn from the current results of these demonstrations and next steps in the development of this trust management framework for TANGO.

2 Demonstrations

This section covers the demonstrations from each of the tasks within WP4. These demonstrations are planned to be presented in video format although they are based on dockerised components and therefore they could also be demonstrated live.

2.1 Self-Sovereign Identity [T4.1]

Previously in D2.3, it was analysed the possible integration using existing software such as Hyperledger Aries, supported by released GAIA-X components and also partner´s assets; using the Eclipse Dataspaces Connector; and/or developing new SSI components based on OpenID standards and open-source technologies to support Verifiable Credentials.

Subsequently, the decision was to choose the latter and support TANGO with new SSI components based on OpenID standards, following EBSI specifications[1].

Additionally, the architecture discussions have concluded in selecting the FIWARE Data Space Connector[2] as the basis for providing the TANGO Connector components as it is in greater alignment with TANGO objectives being in alignment with the DSBA Technical Convergence document[3]. Note the FIWARE Data Space Connector SSI components will be on the Provider Connector whereas on the Consumer Connector side the SSI components will be provided by TANGO implementation described in this section. The resultant credentials and interwork will ultimately prove the different SSI Solutions from TANGO and FIWARE with collaboration being undertaken with FIWARE on any interoperability issues that may arise.

It is finally noted that D2.3 focused on SSI support of persons with wallets, and devices supported by DIDs with no support of VCs. However, with the decision to integrate with the FIWARE Data Space Connector, then it is being re-visited to support IoT devices with SSI based on EBSI and the DSBA Technical Convergence for IoT devices that are not constrained on processing and memory, as is the case for the TANGO pilots. As this is a new implementation for the partners and only recently has the decision been taken to integrate the FIWARE Data Space Connector the scope of the first SSI solution is focused on supporting legal and natural persons with their wallets, verifier and issuer components on the consumer side. In the final deliverable D4.2 it will be analysed the support for IoT Devices, as well as the final solution supporting persons and integrated with the FIWARE Data Spaces Connector.

2.1.1 Component description - SSI Agent for Issuer and Verifier

The SSI Agent implementation considers the scope as outlined in section 2.1 and is focused on persons for this deliverable. The following sub-sections will go over the specification and design of the SSI Agent supporting the issuer and verifier capabilities.

2.1.1.1 Standards

The SSI Agent will follow the following specifications for implementing issuer and verify components:

- W3C Decentralized Identifiers V1.0[4]
- W3C Verifiable Credentials data Model V2.0[5]
- Self-Issued OpenID Provider (SIOP) v2[6]

[1] https://hub.ebsi.eu/get-started
[2] https://github.com/FIWARE/data-space-connector
[3] https://data-spaces-business-alliance.eu/wp-content/uploads/dlm_uploads/Data-Spaces-Business-Alliance-Technical-Convergence-V2.pdf
[4] https://www.w3.org/TR/did-core/
[5] https://www.w3.org/TR/vc-data-model-2.0/
[6] https://openid.net/specs/openid-connect-self-issued-v2-1_0.html#name-self-issued-openid-provider-r

- OpenID for Verifiable Presentations (OID4VP)[7]
- Verifiable Credential Issuance (OID4VCI)[8]
- JWT VC Presentation profile[9]
- EBSI Guideline specifications[10]

2.1.1.2 Verifiable Credentials

In alignment with EBSI and DSBA it supports natural and legal persons and representatives of the legal person organizations and also organization member representatives and end users of the organization´s services.

The format of the Verifiable Credentials (VCs) for natural and legal persons is specified by EBSI[11].

The onboarding of natural persons is simulated in TANGO by the onboarding component described in section 2.2 and will interwork with the SSI Agent component to support the issuing of a natural person VC.

Onboarding Natural Person integration point with T4.2 noted.

The onboarding of legal persons will be handled by a frontend User Interface supported by the issuing organization with interwork with the SSI Agent component to support the issuing of a legal person VC. It will either be simulated or registered as a Legal Person on the EBSI registry[12].

Onboarding Legal Person Integration point noted with Service Provider UI/FE & EBSI Registry.

Additionally, support of end users and organization members credentials is supported by the Service Provider issuing these credentials to users that previously presented their natural person eID VC to the SP.

2.1.1.3 Internal architecture

The self-sovereign identity (SSI) Agent is composed of several submodules to handle the issuing, verification and presentation of Verifiable Credentials, such as Decentralized Identifiers (DID), cryptography proofs, and presentation and issuance protocols following the OID4VP and OID4VCI standards. The following figure provides a high-level overview of the SSI Agent and its sub-modules components.

[7] https://openid.net/specs/openid-4-verifiable-presentations-1_0.html
[8] https://openid.net/specs/openid-4-verifiable-credential-issuance-1_0.html
[9] https://identity.foundation/jwt-vc-presentation-profile/
[10] https://hub.ebsi.eu/conformance
[11] https://www.w3.org/TR/vc-data-model
[12] Trusted Issuers Registry API v4 | EBSI hub

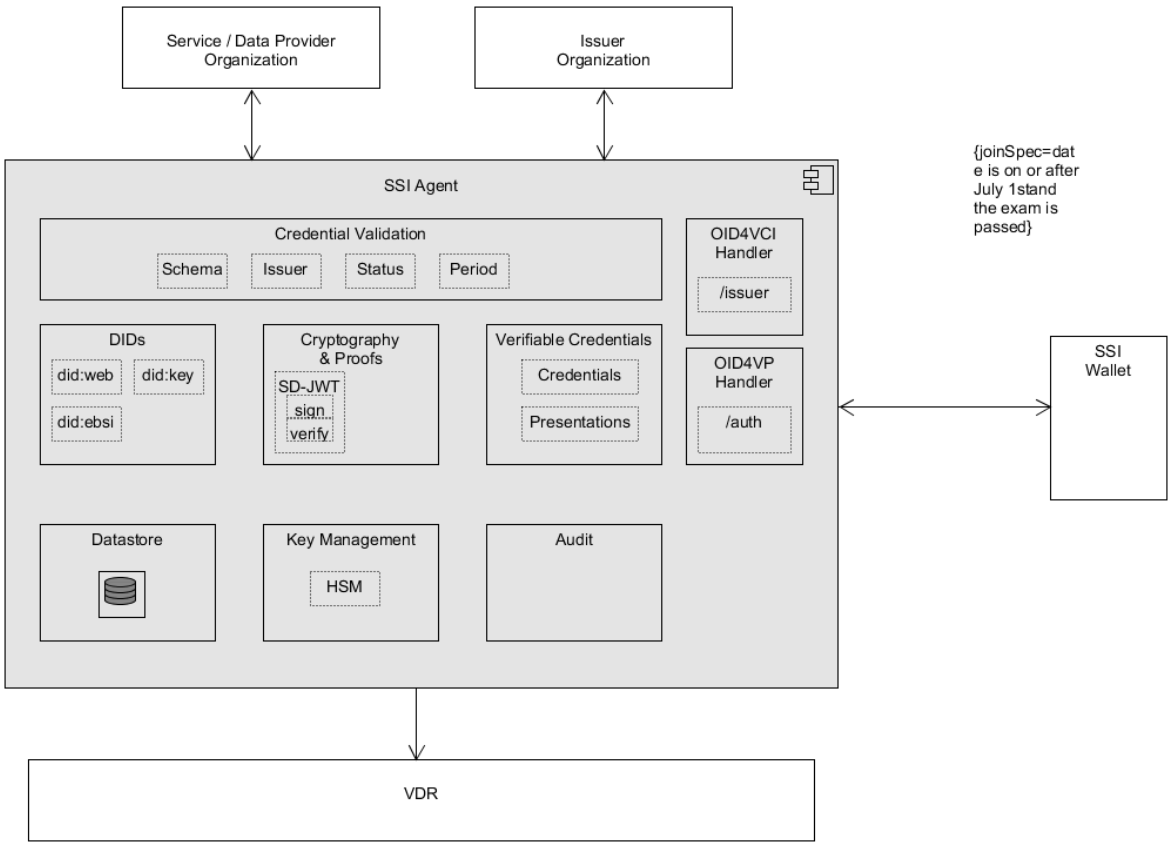


Figure 1. SSI Agent Component Architecture

The main functionality provided by the different submodules of the SSI Agent are described below:

DIDs:

- DID Generator: Creates new DIDs for users and entities within the SSI ecosystem.
- DID Resolver: Responsible for resolving and retrieving DID documents published on Verifiable Data Registries and other DID methods.

Verifiable Credentials

- Presentation Generator: Constructs verifiable presentations that include selected credentials for presentation to relying parties.
- Presentation Verifier: Validates verifiable presentations received from other parties to ensure the integrity and authenticity of the presented credentials and is supported by the Credential Validation for further checks.
- Manages the issuing, storage and presentation of verifiable credentials.
- Generates and signs verifiable credentials, attesting to certain claims or attributes about a subject.

Datastore

- Repository for Verifiable Credential templates
- Stores and manages issued verifiable credentials for easy retrieval and verification.

Cryptography & Proofs

- Implements cryptographic functions such as digital signatures, hash functions, and encryption required for secure interactions and validation.
- Creates cryptographic proofs (e.g., zero-knowledge proofs) to demonstrate the validity of claims without revealing sensitive information.

Key Management

- Manages the generation, storage, and usage of cryptographic keys associated with DIDs and verifiable credentials.

OID4VP and OID4VCI Handlers:

- Implements the OpenID protocols for Verifiable Presentation and Verifiable Credentials, ensuring compatibility with the specified standards.
- Data Model Adapter: Converts and maps data structures between the internal representation used by the SSI agent and the formats defined by the OID4VP and OID4VC standards.

Audit

- Provides records and logs all relevant actions and interactions for auditing and accountability purposes.

2.1.1.4 Implemented features

Further to the core functionality that has been provided in the previous section as supported by the SSI Agents internal architecture, this section will elaborate on key features that are supported in this first version which are important to ensure interoperability in the TANGO solution and considering integration with the FIWARE Data Space Connector.

Verifiable Credentials interoperability

- The SSI Agent will support SD-JWT for issuing and verifying credentials with Selective Disclosure (SD)
- Further to the crypto algorithms supported by the JWT VC presentation Profile the SSI Agent will support the ZKP algorithm provided by the dp-ABC module as described in section 2.1.2.2.

Issuer of Verifiable Credentials

To ensure interoperability with issuer organizations and wallets for issuing verifiable credentials the interwork flow will follow the EBSI specification guidelines on Issue Verifiable Credentials[13]. The flow is shown below, and note is still subject to some changes as EBSI implementation guidelines are subject to change and the development and testing is still ongoing.

[13] https://hub.ebsi.eu/conformance/learn/verifiable-credential-issuance

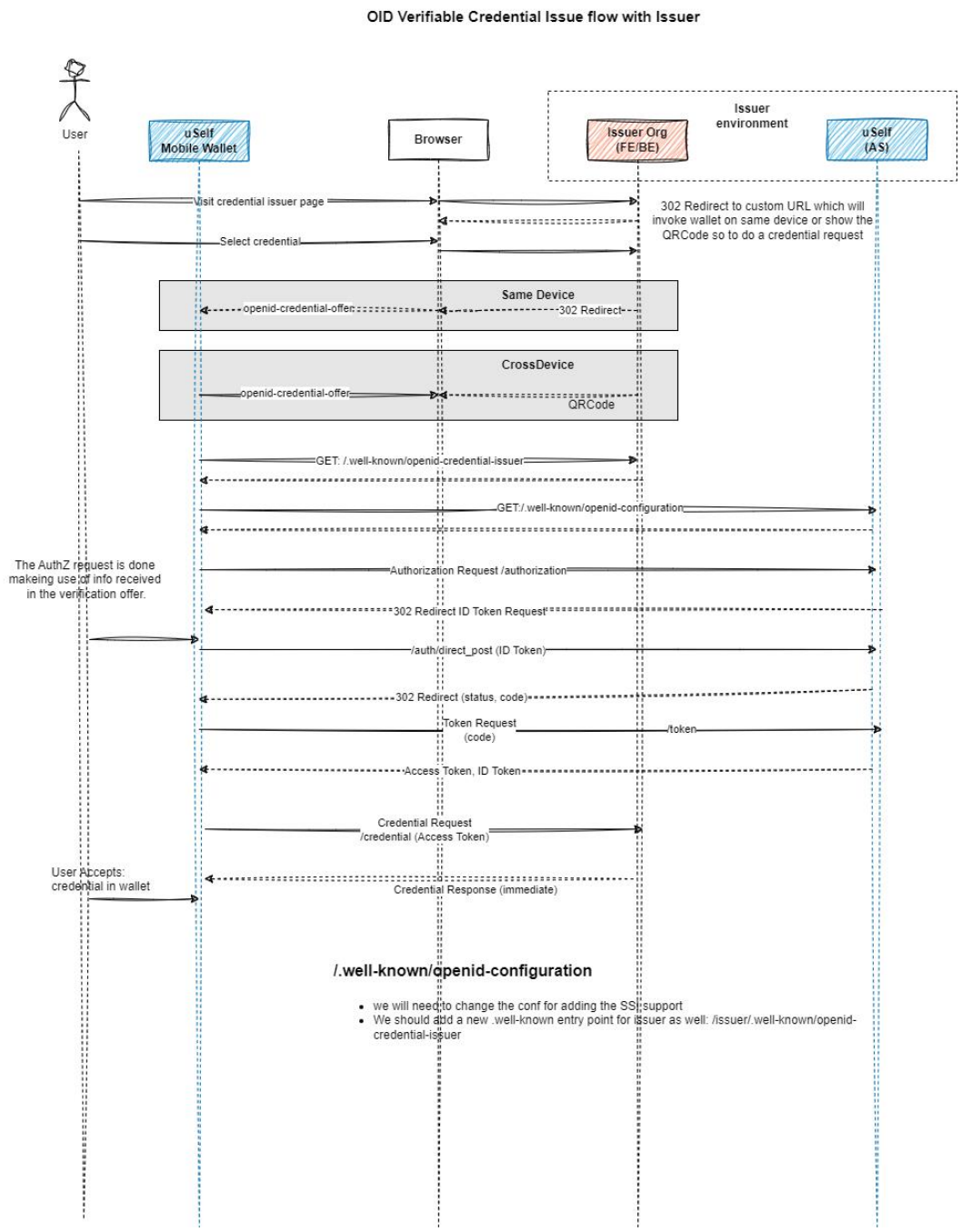


Figure 2. Issue Verifiable Credential flow diagram

Verifiable Presentation Exchange

To ensure interoperability with other wallets and relying parties and wallets the interwork flow will follow the EBSI specification for Verifiable Presentation Exchange[14]. The flow is shown below, and note is still subject to some changes as EBSI implementation guidelines are subject to change and the development and testing is still ongoing.

[14] https://hub.ebsi.eu/conformance/learn/verifiable-presentation-exchange

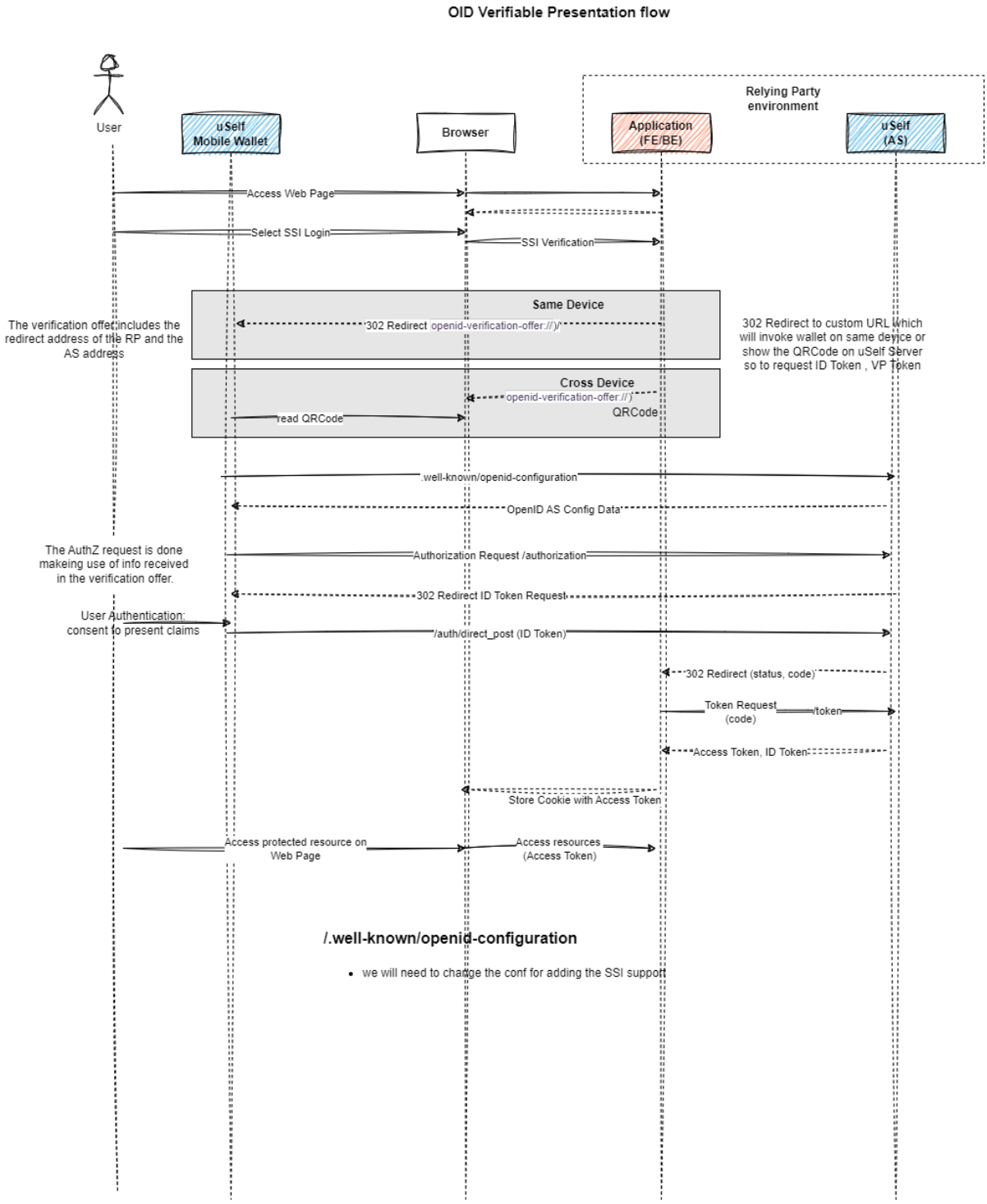


Figure 3. Verifiable Presentation flow diagram

2.1.2 Component description - Wallet

TANGO Wallet is based on the Walt.id implementation. The current version can be used (in conjunction with an issuer and verifier) to receive issued credentials and present credentials for verification.

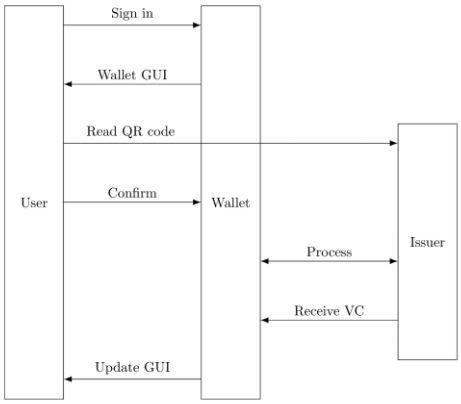


Figure 4. Issuing a Verifiable Credential (VC)

Figure 4 is a message flow chart about issuing a verifiable credential to the Walt.id Wallet. Next Figure 5. Shows a message flow chart depicting verifying a VC with Walt.id Wallet.

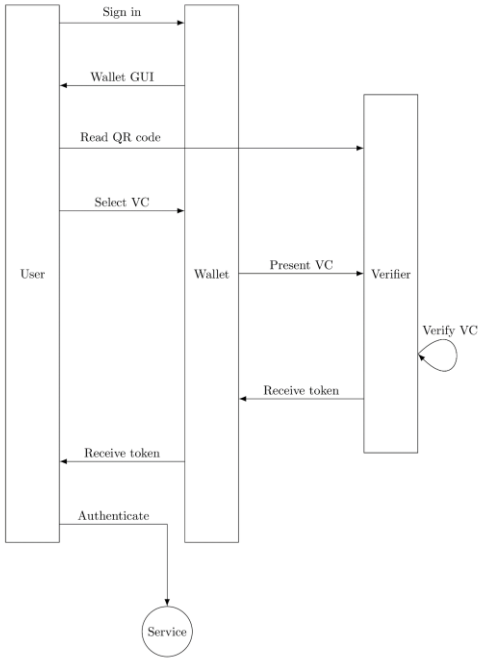


Figure 5. Verifying Verifiable Credential

2.1.2.1 Internal architecture

Walt.id Wallet architecture is depicted in Figure 6.

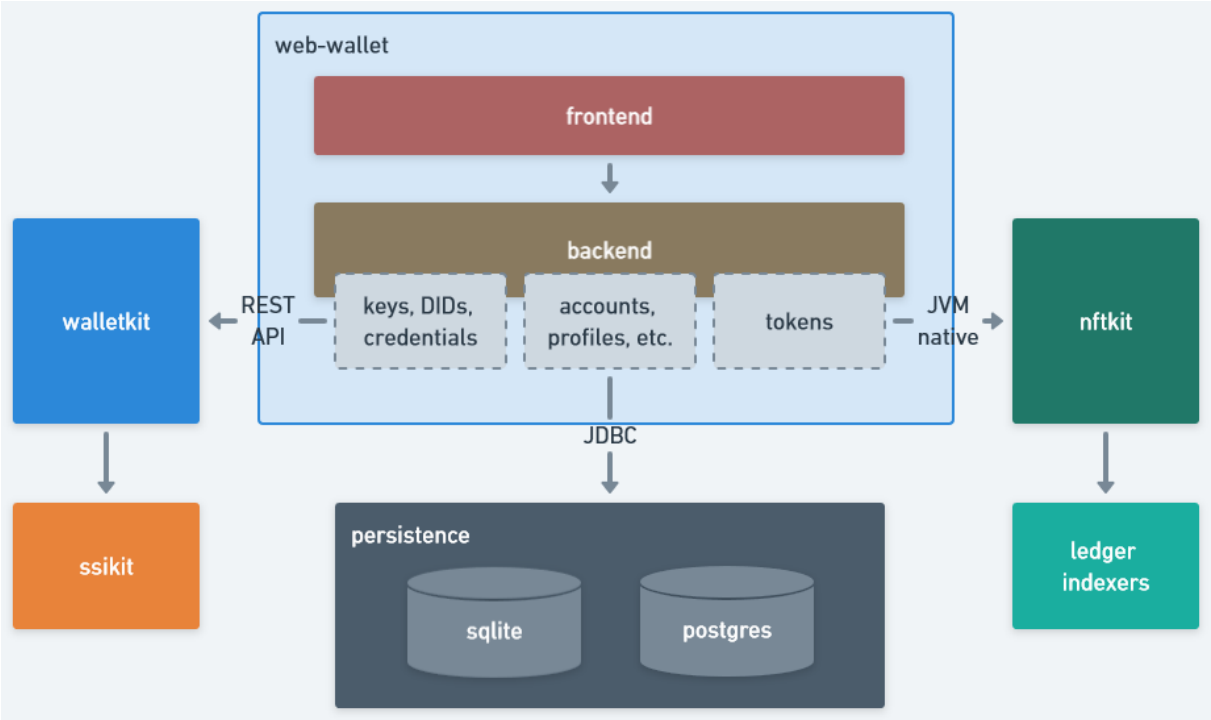


Figure 6. walt.id Wallet architecture (source walt.id documentation[15])

2.1.2.2 Implemented features

Privacy-preserving attribute-based credential scheme based on multi-signatures.

This deliverable, and the task it is related to, deals with the privacy-preserving identity management of TANGO architecture supported by distributed and cryptographic techniques.

The identity management solution of the TANGO project, following a self-sovereign approach, addresses the challenge of 1) secure authentication and identification of devices, while also tackling 2) privacy concerns.

Traditional identity systems that rely on personally identifiable information (PII) or centralized authorities disclose comprehensive identity details. In contrast, by using Privacy Attribute-Based credentials users can present credentials based on specific attributes, disclosing attributes selectively (Selective Disclosure, SD), granting access to services and resources without exposing unnecessary sensitive information.

For this reason, one of the key components integrated within the architecture is the cryptographic p-ABC module, enabling the use of distributed privacy-preserving Attribute-Based Credentials based on Pointcheval-Sanders multi-signatures[16] (dp-ABC). The dp-ABC module will be integrated in the SSI management components so that the enabled operations are used in the TANGO identity management flows.

Thanks to this cryptographic module, the security and privacy of the Self-Sovereign Identity (SSI) mechanisms are strengthened. Nevertheless, it also leads to the usual limitations: p-ABC are more expensive than plain signatures in terms of computation resources, and they are not a widely implemented solution. However, these limitations are palliated by the work in this project. The cryptographic primitive is integrated into W3C's Verifiable Credentials (VC)17, which is an emerging

[15] https://docs.walt.id/v/apps/solutions/web-wallet/architecture

[16] Camenisch, J., Drijvers, M., Lehmann, A., Neven, G., & Towa, P. (2020, September). Short threshold dynamic group signatures. In Security and Cryptography for Networks: 12th International Conference, SCN 2020, Amalfi, Italy, September 14–16, 2020, Proceedings (pp. 401-423). Cham: Springer International Publishing.

[17] https://www.w3.org/TR/vc-data-model

standard gaining lots of traction in identity management solutions, especially those based on SSI. The standard establishes a model for representing digital credentials in an interoperable and machine-verifiable way, with the key property of being cryptographically secure. Verifiable Credentials play an equivalent role to physical identity credentials, consisting of information related to the subject of the credential (identity attributes such as name), information that identifies the issuing authority, and other metadata (expiration dates, type of credential…). Verifiable Credentials will be generated and signed by an issuer, and the holder will have complete control over them from that point. Holders will carry out authentication processes directly against verifiers. For that, they can use their VCs to derive Verifiable Presentations, which are a tamper-evident way to gather and share identity information from the credentials for a presentation process.

In our case, the p-ABC technology will be used to modify the signed VC so that only part of the information is revealed, while keeping the formal authenticity guarantees. Thus, the Verifiable Presentation will contain the derived credential, increasing the privacy of the holder. Thanks to the achieved integration, the interoperability and ease of adoption of the p-ABC solution improves greatly. Apart from the integration into VCs, we integrated the results into the open-source wallet of walt.id[18], which intends to create a shared, reusable, interoperable tool kit designed for initiatives and solutions focused on creating, transmitting and storing verifiable digital credential, following SIOP[19] and OIDC4VP[20] specifications.

Implementation design

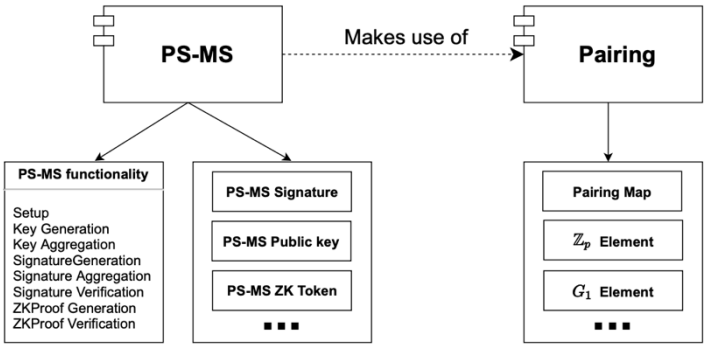


Figure 7. Abstract overview[21] of the components inside the p-ABC solution

Functionalities of the solution are separated in two differentiated modules:

- PS-MS: comprises all necessary operations like serialization/deserialization, hashing, etc.
- Pairing-friendly elliptic curves: encapsulates all pairing-related functionalities implemented through MIRACL Core Cryptographic Library[22].
- The implementation must support the functionalities provided by the scheme, depicted in the following methods and processes:
  I. Key generation: used to obtain a private key and the corresponding public key.
  II. Key aggregation: combines n public keys into one that corresponds to the group of n signers
  III. Signature Generation: uses a private key to sign a message (a set of attributes) so the signature is valid for the corresponding public key.

[18] https://github.com/walt-id
[19] https://openid.net/specs/openid-connect-self-issued-v2-1_0.html
[20] https://openid.net/specs/openid-connect-4-verifiable-presentations-1_0-07.html
[21] https://www.sciencedirect.com/science/article/pii/S2214212621001824
[22] https://github.com/miracl/core

IV. Signature Aggregation: aggregates a set of n valid signatures that were created with n private keys for the same message. The result is a single PS-MS signature that is valid for the aggregated public key of n signers.
V. Signature Verification: performs a verification of a signature with respect to a specific public key (whether they are the result of aggregation or not).
VI. ZKProof Generation: takes a signature sigma over a set of attributes and generates a zero-knowledge proof of possession of sigma and the attributes, revealing only a subset of them
VII. ZKProof Verification: verifies a zero-knowledge proof generated by the previous method for a specific public key. If the verification is successful, the verifier can be sure that the prover has a signature sigma valid for the public key, and the revealed attributes are a subset of the attributes used for generating sigma.

The followed implementation approach for integrating the dp-ABC schema into the TANGO SSI model consists in the integration inside the cryptographic library that walt.id is currently using as the solution[23]. For this reason, two main steps are required:

I. Adapting source cryptographic libraries (creating new signature suites) to implement ZKP functionalities.
II. Adapting the three main entities of the SSI model so that the needed functionalities for the dp-ABC flows are implemented:
III. User's credential manager (PCM, SSI wallet): functionality of combining the shares into a complete signature (verifying its validity) and generate ZK proofs from the signatures to create presentations tokens.
IV. Verification module (Verifier, SSI Agent): functionality of checking that a ZKP proof is valid for the master public key.
V. Distributed credential module (Issuer, SSI Agent): functionality for generating the issuer public and private key, and for signing a set of attributes to generate a credential share.

Granular Disclosure. To allow users to protect their digital identity, the wallet includes granular disclosure functionality. The current version keeps track of all previous disclosures of a credential, and presents this information to the user when making a new presentation. In this way users can keep track of whether they are comfortable with the amount of identity data they have shared. More advanced functionality meeting this aim is envisioned as future work.

2.1.2.3 Software artifacts

SSI Agent:

- A docker image
- Readme with deployment & test guide

Wallet:

- A guide to using the wallet in the context of Tango including installation instructions.
- A description of the interface of the wallet (for credential verification and presentation of authentication tokens).
- Instructions for deploying the wallet in a docker environment.

2.1.3 Demonstration description

2.1.3.1 Docker/API description

SSI Agent

- The SSI Agent is provided as a docker with the issuer and verifier functions, as described previously, for supporting a demonstration with the wallet.

[23] https://github.com/WebOfTrustInfo/ld-signatures-java/tree/main, https://github.com/danubetech/key-formats-java/tree/main

- Test logs demonstrating the interwork flow have been supplied for demonstrating the integration with the wallet.

The wallet is composed of two dockerized applications:

- **Wallet back-end**. The back-end handles logic around storage, users and presentation of verifiable credentials. It interfaces with the front-end and issuers and verifiers.
- **Wallet front-end**. The front-end presents the user with an easy-to-use interface for receiving and presenting verifiable credentials, as well as logic for using authentication tokens in Tango services.

2.1.4 Support for pilots

All the use cases in pilots use Self-sovereign identity in some form, at least for the purpose of identifying users. Some of the pilots utilize also other functionalities provided by SSI as follows:

- **Smart hospitality.** SSI provides user login, credentials, and user key data. The user is issued with a eID (simulation national eID). Service providers are able to verify Verifiable Credentials from users SSI wallet.
- **Autonomous Vehicles.** A user is issued with an eID.
- **Smart Manufacturing (case FMAKE).** Issue Service Provider staff with Organization Member credentials and verify Verifiable Credentials submitted from a user SSI Wallet.
- **Smart Manufacturing (case RIAS).** Issue Service Provider staff with Organization Member credentials and verify Verifiable Credentials submitted from a user SSI Wallet.
- **Public Administration (case VISAR).** Issue Service Provider staff with Organization Member credentials and verify Verifiable Credentials submitted from a user SSI Wallet.
- **Retail.** Issue Service Provider staff with Organization Member credentials and verify Verifiable Credentials submitted from a user SSI Wallet.

2.1.5 Future work on this component

The work in this task is still ongoing and further work as part of TANGO on these components consists of analysing support for:

- Interoperability testing with FIWARE Data Space Connector
- Issuing IoT Devices with Verifiable Credentials
- Verifying presentations from IoT Devices
- Support IoT Devices with SSI Agent (acting as a wallet) for receiving, storing and presenting Verifiable Credentials

2.2 Seamless onboarding [T4.2]

Criminals will always invent new ways of committing fraud so identity theft will always carry challenges for consumers and organizations. In parallel, COVID – 19 forced public and private organizations to speed up digital transformation in order to shift most of their services online. Experian reports that during COVID – 19 lockdown there was a 33% increase in fraud rates. In 2022[24], the Federal Trade Commission received a 46 percent in identity theft complaints, with a total of 2.4M reported cases. The most reliable mechanism for physical identification and authentication these days is still the request of a photo ID document and the match between this document with attributes of our physical appearance[25]. This method is still bound to frauds as the ID verifier rarely has the tools to verify the legitimacy of the physical ID document and in very few places a biometric check can be successfully used. While virtual identification solutions have emerged, their substantial costs (averaging $0.1-5 per

[24] https://www.ftc.gov/system/files/ftc_gov/pdf/CSN-Data-Book-2022.pdf
[25] https://www.forrester.com/report/the-forrester-wave-tm-identity-verification-solutions-q4-2022/RES176428

transaction[26]) pose a challenge. Additionally, citizens lack ownership of their identity data, and there are concerns regarding the reliability of the security measures employed for its management and storage.

Seamless onboarding for users will provide the first open-source solution that allows strong identity verification with High Level of Assurance, allowing users to transform their physical identity into Verifiable Credentials. This section in focused on providing a component description, the overall architecture of the system including the various components that are involved in the process, the description of the demonstration, the support for the pilots and the future work to be implemented for the component.

2.2.1 Component description

Seamless Onboarding for User is one of the first open-source components that enable end-users to easily and without friction to be able to perform remote identity verification based on an existing identity document, without the intervention of a human person, and maintaining High Level of Assurance based on the eIDAS regulation. The component can be leveraged by different organisations that would like to perform strong identity verification of their end-users, while interconnecting with the rest of the identity management system, which will leverage the digitalised and verified identity of the end-user. In the context of the TANGO platform, the component will be used in order to verify the identity of the end-users before they become part of the TANGO identity management based on SSI. It will constitute the initial step that will allow the users to verify their identity, create their digital identity on the SSI component and then create verifiable credentials that will be used for the secure communication and authentication with other components of the TANGO platform.

The component presents to the end-users a user-friendly wizard that guides the end-user throughout the process of onboarding their identity and digitalising their identity information. During the process of the onboarding, different types of security measures are applied in order to mitigate potential attacks and attempts to falsify the identity of the user. Once the end-user has gone through all the steps of the onboarding process, the information is cross-check in order to detect any inconsistences, and in case the cross-check is successful the data are sent to the SSI components that are responsible for the generation of the verifiable credential at the Issuer side of the SSI component. Then the SSI component is managing appropriately the rest of the interaction with the Verifier as well as the rest of the TANGO components in order to enable the secure authentication of the user. Upon confirming the person's identity at the device level with a high level of confidence, the Issuer can examine the verified identity components, cross-referencing them with national databases (if accessible). Upon successful validation, Verifiable Credentials are subsequently issued.

For the device, each of the device will provide a unique identifier to the SSI component, generating a distributed identifier, which will enable then the Issuer to create the verifiable credentials that will be used by the Verifier to perform the authentication process of the devices. Onboarding of IoT Devices is not in scope for this first version of the deliverable and will be in the final version. With the integration of the FIWARE Data Space Connector now agreed in TANGO, the previous solution based on DID Auth Challenge is seen to be no longer feasible and it will now analyse support on the consumer side for an IoT Device integrated with an SSI Agent (acting as a wallet), supporting the full SSI stack capability.

The Seamless Onboarding component is a technology focused on mobile platforms considering the increased security available of those devices. The component will allow developers to easily and with one line of code, to integrate a seamless onboarding of users, to their apps, enabling the end-users to verify their identity in a user-friendly manner. The end-users will securely and seamlessly onboard their identity to the TANGO platform, by leveraging an existing identity document such as a passport. The end-users will be asked by the target app, in this case the TANGO wallet, to perform identity verification

26 https://documents1.worldbank.org/curated/en/945201555946417898/pdf/Identity-Authentication-and-Verification-Fees-Overview-of-Current-Practices.pdf

by showcasing a wizard that guides the user through the process, regarding the steps that need to be followed. These steps are described below:

- **Choosing the identity document**. The type of verification that will be prompted will be depended on the type of document that will be used. This screen will inform the users, that they need to choose what type of identity document they will use for the identity verification process and from which country the document was issued. The initial version of the system will encompass the validation of a European passport that features an integrated NFC chip. The use of NFC chip allows the system to achieve the High Level of Assurance (eIDAS) and if not used then only a Moderate Level of Assurance can be achieved. In such cases, where the organisations would like to support additional documents apart from the documents that incorporate an NFC-chip, there is a need to setup an additional procedure internally at the organisation that requires human intervention done through a video-based verification. The process for choosing the country from which the document origins, allows the future development of personalised models for each country. This means that the current version of the document scanning process will incorporate a generalised models that allows the scanning of all documents available. Further robustness could be achieved as well as additional security mechanisms could be incorporated by developing specific models for each of the supported identity documents, depending on the country selected by the end-user.
- **Taking a photo of the document**. During this stage the end-user is informed about the process to be followed in order to take a photo of the information shown on the identity document. For the case of the passport the end-user will need to open the passport document and take an image from the information shown. It is important that the camera is able to read the machine-readable zone of the passport document which allows the conversion of the information shown on the passport document, into digital information. A camera screen is displayed to the user for capturing a clear image of the information on the identity document. The user will be guided to capture a clear image of the document including the necessary information essential for the verification process. Using AI-based Optical Character Recognition technology, the identity information from the document can then be extracted. In order to avoid taking a blurry image, the user is required to hold steady the smartphone camera related to the identity document. Once the software detects a steady image including all the appropriate information which are visible on the specific image, the component captures the image of the identity document and cross-validates with the user if the image is blurry. Following the confirmation from the user, the component moves to the next step of the verification process.
- **Scanning the identity document through NFC**. The majority of European passports currently incorporate an Near Field Communication chip for security purposes. This is an encrypted chip that has been placed by the relevant Authorities which issued the identity document, and incorporated the identity information inside the chip, make it difficult for the fraudsters to falsify the identity document. The information stored in the chip are encrypted and cannot be changed, as the available actions are only to read the identity information. The chip stores information such as the visible identity information including name, surname, date of birth, passport number, additional information which are not visible on the passport, as well as a digital version of the facial image of the passport owner. This high resolution image allows the Authorities to use the particular image to perform facial matching between the person present on the passport document and the person that claims that is the owner of the identity document. During the onboarding process, when the end-users reaches the particular stage, the user will be prompted with a screen guiding the user on what action should be performed. In particular the end-user will have to place the smartphone above the NFC-enabled identity document. Having enabled the capability of the smartphone to read NFC chips, the component will have the capability to retrieve information stored on the chip including the identity information of the end-user as well as the image stored on the chip. This process will enable the comparison of the identity data of the NFC chip and the one displayed on the document itself, preventing any fraud cases or tampering of the identity data.

- **Taking a selfie from the user**. The next step to be followed in the identity verification process is the live acquisition of the end-user's facial image by leveraging the front camera of the smartphone. The majority of today's smartphone incorporate a frontal and back camera enabling the end-users take images. The frontal camera is primarily used in order to allow end-users to take selfie images, while showcasing their own face on the smartphone's screen. The user will be guided through a screen to take a selfie through the frontal on-device camera of the smartphone. Once the screen is visible a face detection algorithm will start the process of locating the face of a person. This step is added in order to avoid performing unnecessary processing and potentially introduced error into the face matching process. After identifying a face, the liveness detection process is activated to determine if the face visible on the screen corresponds to a real person and to confirm the absence of any presentation attacks. Presentation attack is the process of a malicious user showing in the camera a fake image or a video. If the liveness detection is successful then the user is permitted to take a selfie. Blurriness detection is also performed in order to avoid cases where the image taken by the user is blurry due to potential movement of the smartphone device when the user was taking the image i.e. movement blurriness. The blurriness detection allows also the reduction of potential error that could propagate into the face detection, liveness and face matching mechanisms. The face matching algorithm is employed to compare the selfie image with the images obtained from the passport through NFC and those displayed on the document.
- **Final validation of the identity and facial information**. The previous steps focused primarily on collecting the appropriate information about the end-users' identity and the facial data required in order to cross-check them, perform the appropriate validations, resulting in the successful identity verification process. There are primarily two types of data that are leveraged in the validation process, the identity textual data and the facial data. The identity data have been retrieved through optical character recognition when the user was asked to take a photo of the internal part of the document and secondly when the user was asked to place the smartphone on the identity document with enabled the NFC-chip. The facial data have been retrieved in three ways, a) from the image taken from the actual identity document that contains the image of the end-user next to the identity data, b) the high resolution image of the user that is stored on the NFC-chip of the passport, c) the high resolution image taken from the selfie of the user, where the different defence and pre-processing steps are applied. During the final stage of the verification process, cross-validation occurs, involving the verification of both the identity details presented on the document and the image of the individual performing the identity verification. The textual are data are compared one-by-one, while the images are compared with each other in order to be able to understand if it is the correct person performing the onboarding process comparing to the actual document. If the validation is successful then, the information is passed on to the Issuer not only for further verification of the document but also for the issuance of the Verifiable Credentials.

2.2.1.1 Internal architecture

The component has been built on mobile multi-platform framework, the Kotlin Multi-platform framework. This framework enables the development of the majority of the code of the component on one framework and then through user interface additions on platform specific code, the component is able to compile into both Android and iOS related libraries that can be incorporated by native Android and iOS applications. In this way, the framework provides the flexibility to the developers allowing them to support the required mobile operating platform that the specific host app is about to support. Through the Kotlin Multi-platform the core elements of the component are built providing the guidelines to the user on how to go through the onboarding process, the functionality for collecting the appropriate identity data required, and applying all the validation mechanisms in order to verify the identity of the user based on an existing identity document. Following the Model View Controller approach, the functionality of the component has been split according to the different screens required in order to retrieve the relevant identity information from the user. Each of the following screens is defined based

on the functionality they need to implement for the component, accompanied by an introductory screen that describes to the user what the actual step requires from the user to perform in terms of guidelines.

- Identity document selection screen. The user needs to select what type of document will be used for the identity verification and from which country was the document issued.
- Document scanning screen. Optical character recognition scans the Machine Readable Zone in order to retrieve the identity information visible on the document.
- NFC scanning screen. Reading of the identity information stored for security reasons on the identity document NFC chip, is performed, including the greyscale image stored on the NFC chip.
- Selfie screen. A camera preview allows the user to take a selfie from the frontal camera, while face detection and liveness detection assist the component in taking a suitable image for the facial matching among the various version of the user's facial image.
- Validation screen. The final screen performs a validation of the identity information collected and if successful it showcases the summary of the identity information that have been collected and validated.

2.2.1.2 Implemented features

This section presents the key implemented features of the Seamless onboarding component, with respect to the technology described in the Description of Work as well as the description provided above about the actual component. Each of the features is analysed with respect to the benefit and value it brings to the overall component as well as in terms of how the feature was developed and incorporated into the overall component.

- **Optical Character Recognition**. This feature will provide the capability to the component to digitalise the identity information of the document. In essence the Optical Character Recognition allows developers to convert a text visible on an image to be read into a machine readable format, which can then be further process and be incorporate in a higher level procedure such as the identity verification process. There are different methods that have been applied that allow the reading of the visible text. This component builds on machine learning mechanisms that allow the automated reading of the text visible on an image. The technology allows the component to read the identity information of the document visible through the Machine Readable Zone that incorporates the majority information of the identity document. The information is displayed through a specific format following a particular sequence on the machine readable zone. A detector allows the component to detect the specific zone visible on the camera frame, ensuring that the whole zone is included in the image before taking the actual image and reading the information of the identity document. The technology is based on Tesseract which is an open source library provided by Google, incorporating all the appropriate mechanisms to read and convert the text from an image. The component performs continuous scanning of the image frames visible on the streaming video of the camera, and once the machine readable zone is detected as a whole, and the image is not blurry the component captures the specific image. In order to ensure that the image is not blurry, the component will ask from the end-users to validate that the image is not blurry.
- **NFC chip reading**: This feature will provide an additional layer of security by ensuring that the identity details on the chip align precisely with the information presented on the physical document. Generally, the NFC chip in identities & passports holds encrypted information of the users' personal details, making it extremely challenging for fraudulent activities. Through this application users will be prompted to place their smartphones above the NFC-enabled identity document. The component, with its NFC reading capabilities, will retrieve encrypted information from the chip, including identity details (name, surname, date of birth, passport number, and any existing additional non-visible information) and the stored image of the user. This process will ensure a robust comparison between the NFC chip's identity data and the visible information on the document, effectively preventing any potential fraud or tampering incidents. Additionally, encryption algorithms and secure data transmission channels will

guarantee confidentiality of the users' sensitive information after the successful completion of the comparison process. NFC-chip reading is a security mechanism that is enforced by the eIDAS regulation in order to avoid tampering of the identity document and the information visible on the actual document. For that reason, it constitutes an essential part of the identity verification process and without this step the level of assurance of the verification process would fall to moderate. Thus, the retrieval of the identity information is performed through a secure communication established by the mobile operating system, allowing the component to read the identity information.

- **Blurriness detection**. This feature constitutes a pre-processing step that is employed in various steps of the facial verification process of the user, and in particular for the facial data provided by the user during the selfie step. In essence, it constitutes a step that removes potential noise that could be introduced accidentally by the end-users when in the process of taking the selfie image. Blurriness can be introduced in various ways such as lack of camera focus and user movement either of the device or of the actual user. The camera focus has been configured by the component to automatically focus on the face of the person, and considering the short distance between the face of the person and the actual smartphone/camera, this type of blurriness will take place very seldom. The initial step for the component was to configure the camera appropriately to be able to focus on the face of the person and to try to detect a face of a certain size, by discarding other potentially smaller faces that could be visible in the camera frame. Following experimentation that took place, the most common blurriness factor that was observed is the blurriness due to user movement, primarily of the device e.g. shaky hands. For that reason, a blurriness detection mechanism was introduced to be able to detect such frames and discard them from the onboarding process. Gaussian filters were applied on the images and based on empirical measurements, certain thresholds were defined to be able to detect such frames.
- **Liveness detection**. This feature will provide the liveness detection of the user and ensure its authenticity. Liveness detection is imperative for ensuring that the person that is in the process of onboarding is not applying any attack to the face matching mechanisms and thus ensuring the authenticity of the users' image during the verification process. This process will not require any action from the user. It will be based exclusively on a single frame image which will be the same image that will be used for facial biometric authentication. Liveness detection will provide security across the several types of liveness detection attacks that scammers might use like video replaying of the user's image (Video Replay attacks), printed photo attack where scammers present to the app a printed photo of the user, or Mask attack (printed or 3D where scammers create a mask with the user's characteristics to try to deceive the app by creating a fake identity. The Liveness Detection works on the hypothesis that image capture employs the front-facing camera of a mobile device, resembling the commonly used process for taking selfie images. Following this, the captured image is then processed in order to bring the image into the required format and applying the appropriate filter to enable the classification process to begin. Using powerful Liveness Detection mechanisms, the component will provide an extra layer of protection against the scammer attacks that were previously described ensuring the authenticity of facial matching during identity verification processes.
- **Face matching**. This feature provides the ability to the component to be able to compare two images and understand their similarity i.e. if they belong to the same person or not. During the onboarding process, the component retrieves the facial image from the person whoms identity is about to be verified and compares the facial images retrieved from the frontal camera, the image of the identity document and the NFC chip. The face matching algorithm is built on the basis of Google's FaceNet, a state-of-the-art deep learning model for face recognition. An enhanced version of the Google's FaceNet with additional components has been further developed to ensure robust and accurate face matching. The face matching procedure consists of various stages, each strategically created to address all different challenges and boost overall performance. The component integrates two tiers of face detection, precisely pinpointing and extracting facial regions from input images. The face matching technology checks if the images

provide by the user are in the right size, resolution, and format. This process makes sure the images work well for accurate facial analysis and comparison. By making the faces look similar in different images, the face matching makes recognition more accurate and reliable. The "heart" of this technology is relying on the powerful FaceNet which utilizes deep neural networks to extract facial features and depict them in a high-dimensional embedding space. Then, the face detection, blurriness and liveness detection as well as certain pre-processing steps of the images help the component match and identify faces reliably. The component then converts the images from the image of the passport, the image from the NFC chip and the selfie taken in the previous steps, and then compares these images to identify if it is the same person across all images.

2.2.1.3 Software artifacts

The Seamless onboarding component constitutes of primarily one software artifact that incorporates all the relevant steps required to perform identity verification with High Level of Assurance based on the eIDAS regulation. This software artifact is responsible for kickstarting the process for performing identity verification on the device of the end-user. Once the identity verification of the end-user has been successfully completed, the identity information of the end-user is forwarded to the SSI component in order to enable the Issuer to create Verifiable Credentials that will allow the end-user to authenticate to the different components and services developed and exposed through the TANGO platform.

The Seamless onboarding component consists of a software development kit that integrates directly with the TANGO wallet app and any other mobile app that envisions and requires strong identity verification leveraging an open-source technology. The software artifact integrates with the host app, by simply incorporating the library itself and incorporating one line of code, to kick-start the process of identity verification. Through this software artifact, the component informs the end-user about the data collection process, guides the user throughout the identity verification process, while provides the appropriate technological elements that will enable the strong identity verification of the user based on an identity document without requiring any human intervention.

The software artifact incorporates:

- Optical Character Recognition in order to be able to retrieve and convert into a digital form, the identity data available on the identity document.
- NFC-chip reading enables the retrieval of the identity information including the facial image of the identity owner stored on the NFC-chip available on the European passports.
- Blurriness detection that enables the detection and discarding of potentially blurry images retrieved at the step, when the user is taking a selfie to prove, that they are the owner of the identity document.
- Liveness detection prevents malicious users to try to utilise an identity document that does not below to them, during the identity verification process, by presenting an image, a video or a mask at the camera when the user is about to take the required selfie.
- Face matching performs a facial verification among the facial images retrieved from the camera of the smartphone, from the NFC-chip and from the image of the identity document.

2.2.2 Demonstration description

For the demonstration of the component, a demo Android app has been developed. It constitutes a simple Android app that integrates the component's software artifact and calls the appropriate functionality that initiates the wizard for the identity verification process. Through the demo app, the end-user is able to go through the whole process of identity verification including the screens of user guidelines as well as the screens that allow the user to retrieve the identity information. Finally, the identity information as well as the image of the passport are shown in a scrollable screen. In addition, a video has been recorded with a user that goes through the identity verification process.

2.2.2.1 Docker/API description

This section provides the description for the API that is exposed through the Seamless onboarding SDK. The SDK exposes the required functionality of the component to the developer that will integrate the component with a host app, in the context of TANGO this app will be the TANGO wallet. The SDK has a simple integration with one line of code that initiates the wizard that will retrieve the appropriate identity and facial information of the end-user, apply the relevant security and pre-processing measures in order to reach to a final conclusion if the person that is performing the identity verification process is the same with the person that owns the identity document. Below there are three tables including the relevant interfaces that are exposed through the Seamless onboarding SDK.

Table 1. Interfaces

Methods	Parameters	Description
startPassportAdditionFlow	No parameters	Opens the screen which will guide a user to add a document
Fields	**Values**	**Description**
documentAdditionResult	A Flow that emits "A2" entries	This fields is needed in order to listen about the state of the document addition. The emission happens once and only for the subscribed collectors. New collectors won't get previous results.
Enums	**Values**	
DocumentAdditionResult	**SUCCEED :** Indicates that a document was added successfully.	
	FAILED : Indicates that a failure occurred and a document could not be added.	
	CANCELED : Indicates that document addition process was cancelled by the user.	

2.2.3 Support for pilots

This section provides a brief description of the support and the value introduced by the Seamless onboarding component. For the seamless onboarding of users, the component will be integrated with the TANGO wallet, from which each of the end-users will be able to onboard to the TANGO platform and in particular on the SSI identity management component as well as to manage their identity in the various relevant use cases. More details regarding the support and value provided by the seamless onboarding component is detailed below in each of the relevant TANGO use cases:

- **Smart hospitality**. This component poses significant benefits not only for the guests but also for hotel administration. Guests will be able to save time as they will be able to check in before they arrive at the hotel. On the other side hotel owners will be able to reduce productivity costs related to check in and offer an improved experience to their customers. Moreover, they will be able to offer better and more secure access control management for the employees of the hotels.
- **Manufacturing (Use case 1 and Use case 2)**. This component will improve security to manufacturing organisations and reduce the risk of unauthorized access to their equipment or infrastructure as well as reduce the risk of potential malicious users and cyber-attacks, as well as potential accidents due to unauthorised access. Moreover, organisation will be able to better monitor their employees if needed. It will also lead to more efficient and tailored audits regarding the way employees, workers and contractors are onboarded to the organisation. HR departments will have an easier and frictionless way of onboarding and managing new and

existing employees, workers and contractors, through the help of SSI to be able to revoke access in case an employee is leaving the organisation.

- **Autonomous Vehicles**. The component in autonomous vehicles will be able not only to provide secure access of passengers to vehicles but also provide authentication of passengers to the service provider. In essence, the driver and the passengers will be able to easily onboard to the TANGO platform and thus access and manage the potential autonomous vehicle, without friction. Through the SSI based access control, automated access control could be applicable, as well as identity attributes could be automatically checked such as the drivers' license. Additionally, the service provider could be able to provide a ride tailored to the preferences of each passenger depending on different profiles.
- **Public Administration**. The component in public administration will be able to reduce the costs on the physical resources like paper and be able to better allocate their human resources to other tasks. The most important benefit though is the improved security that this solution will provide to control checks for visa applicants. Visa applicants will be able to perform identity verification remotely without having to go to the consulate or the embassy or other relevant organisation, in order to apply for a visa to travel to a specific country. Through the SSI visa applicants will be able to onboard and easily manage their identity including identity document, drivers license and other relevant documentation required to apply for the visa. For the employees of the public sector organisation, a stronger access control and onboarding mechanism will be applicable.

2.2.4 Future work on this component

As future work on this component, the focus will be on improving the reliability of the document screening and the validation of the facial image. Depending on the different lightning conditions, the steps of optical character recognition and the selfie retrieval could affect the data collection and inference process. For that reason, the focus will be on testing the component in various conditions in order to create edge cases that could occur in real-world environments when the users are performing onboarding to the TANGO platform, in order to understand potential errors that could occur and introduce false rejections of legit users trying to verify their identity. Through further testing of the component in various conditions, in order to understand the tolerance of the false rejections and false acceptance metrics, tuning of the thresholds for the liveness and face matching mechanisms will be performed.

2.3 User behavioural authentication [T4.3]

Cybersecurity ventures estimate that global identity fraud already costs $5tn annually[27]. They also expect global cybercrime costs to grow by 15% per year over the next five years, reaching $10.5 tn annually by 2025[28]. As Verizon stated in 2019 phishing remained "the number one cause of data breaches globally" as victims are deceived to reveal sensitive information like login and banking information[29]. In the contrary, IoT devices are increasing the presence in people's daily lives, and it is estimated to grow at $75bn in 2025[30]. This broad range of devices, often designed for cost efficiency by manufacturers, is commonly shipped with insufficient security measures. A10 Networks, a leading manufacturer of application delivery controllers, highlights the security shortcomings found in IoT devices, such as the prevalence of non-existent or default passwords[31]. These vulnerabilities are

[27] https://www.veriff.com/fraud/learn/the-cost-of-fraud

[28] https://www.evolvesecurity.com/blog-posts/actual-cost-of-cybercrime#:~:text=In%202021%2C%20it%20caused%20global,(Source%3A%20Cybersecurity%20Ventures).

[29] https://www.vadesecure.com/en/blog/verizon-data-breach-report-2023#:~:text=Phishing%20represented%2044%25%20of%20all,the%20human%20element%20in%20cybersecurity

[30] https://www.verdict.co.uk/smart-homes-inside-a-new-fast-growing-market-set-to-be-worth-75bn-by-2025/

[31] https://www.a10networks.com/wp-content/uploads/A10-DG-16170-EN.pdf

exploited by automated attacks designed by cybercriminals and allowing them subsequently to quickly assemble botnets for launching Distributed Denial of Service (DDoS) attacks.

The traditional authentication methods that are used for cybersecurity pose several limitations and vulnerabilities. These vulnerabilities might be password weaknesses and increased cyber threats and the limitations might be less adaptive security, one time verification at the point of login or the advancements in technology and the need for stricter regulations.

All the above introduce the need of the user behavioural authentication method which is a method that provides solutions to this need of the modern organisations and their services' users.

User behavioural authentication will provide a novel component that will allow users to authenticate to the TANGO platform, by leveraging their behavioural patterns, enabling continuous assessment of the authenticity of the users. This section describes the technology of the component introduced including the internal architecture and the incorporated features, followed by the demonstration description, the support for the pilots and the next steps as future work applicable on the component.

2.3.1 Component description

User behavioural authentication offers a continuous user authentication tool without requiring user input in order to prevent unauthorised access as well as to improve the user-experience when logging in and when requiring multi-factor authentication. This component achieves authentication by learning and analysing users' behavioural patterns over time, integrating various behavioural observations to establish a user's identity and continuously assess the user when accessing a particular online service. Considering that each of the behavioural elements have a certain amount of error, the proposed component has the ability to combine those behavioural observations while minimising the errors of each individual mechanism, enhancing the overall identification accuracy. Unlike current methods relying on a single biometric (e.g., fingerprint, iris, voice, or face recognition), this approach incorporates multiple biometrics and behavioural patterns from daily activities such as location, interaction, speed, movement, walking, smartphone usage, and more. The integration of diverse behavioural signals contributes to a more robust identification of individuals, considering that it is able to compensate the error of individual behavioural elements, and derive a more reliable and resilient risk score regarding the particular user that is accessing the online service.

The user behavioural authentication component is designed for user authentication on third-party platforms. The authentication process revolves around grasping and learning user behavioural patterns, utilizing data extracted from smartphones, including sensor and device usage data. The components of this authentication system encompass the platform where processing and the authentication procedure take place, and the data collection SDK is responsible for collecting data to be sent to the backend. Data is gathered through a host smartphone app that has integrated the data collection SDK and then transmitted to the backend, which subsequently relays the collected data, accompanied by a pseudo-ID. During the initial phase, the User behavioural authentication component generates a specific user profile based on a pseudo-ID that has been provided by the TANGO platform. Based on the created user profile, the backend stores the received data and starts the data processing and the creation of behavioural profiles for each of the individual behavioural patterns based on the available data received. Once the behavioural models have assimilated knowledge of user patterns, a similar approach is applied to biometric data, involving an authentication service that undergoes an initial enrolment process followed by the biometric profile creation. The overall design of the User Behavioural Authentication component addresses scenarios where the TANGO and any other third-party platform necessitates an authentication mechanism based on behavioural patterns. The TANGO wallet app seamlessly integrates the User Behavioural Authentication component and in particular the data collection SDK, which gathers behavioural data from the smartphone, encrypts it, and transmits it through a secure communication channel to the backend component. The backend component is responsible for the processing and the authentication result inference, which will then be forwarded either back to the data collection SDK or retrieved by TANGO or any other platform through API This facilitates the transmission of behavioural

data to the User Behavioural Authentication backend for authentication, confirming the user's legitimacy and thwarting potential scammer.

2.3.1.1 Internal architecture

The internal architecture of the component comprises to main subcomponents: a) the data collection SDK that is responsible for the collection of the appropriate biometric and behavioural data and b) the backend that is responsible for the data storage, processing and authentication inference. Each of the two components follows a specific architecture that is relevant to the best practices of each of the respective subcomponents based on the deployment target. The subcomponents have been designed following the privacy by design and by use principles to protect the end-users privacy, but also to be able to cope with high demand of users offering geometrical scalability of the component, reliability in terms of enabling updates resilience to potential errors without compromising the available uptime. An overview of the architecture of each subcomponent is provided below.

- **Data collection SDK**. The particular subcomponent differentiates the internal functionality into specific groups based on the focus of the particular implementations. Various data collectors have been implemented for reach of the various data sources i.e. sensors, screen inputs, which potentially incorporate certain data pre-processing procedures that are required to remove potential noise in the data or to convert the data into the suitable format. The functionality for storing each of the data points into temporary databases on the smartphones have been also implemented, in order to allow the developer to temporary cache the data in case of lack of internet connectivity. A separate package has been created that allows the secure communication of the data collection SDK with the backend subcomponent. Additional utilities package have been implemented to facilitate the data transformation but also to implement potential encryption applied to the data stored on the device. In case the facial image acquisition is required to be engaged by the SDK, a package has been included that facilitates the developer to acquire in a secure and user-friendly manner the facial image of the user in order to setup the initial profile of the user.

- **Backend**. The specific subcomponent is deployed on the server side, following a containerised approach through Kubernetes and incorporating various Docker images to ensure scalability and reliability through a microservices architecture. An API gateway is responsible for the communication of the subcomponent with the data collection SDK in order to retrieve the biometric and behavioural data from the particular device. The specific gateway enforces all security measures in order to avoid any potential attacks to the backend, meaning that it implements various mechanisms such as authentication of the calls received, and various security mechanisms such as prevention of SQL injection attacks. The API gateway, once it receives an API call including the behavioural or biometric data, after applying the security mechanisms, it forwards the data to the relevant microservice, for further processing of the specific data points received. Once the data are processed by the particular microservice, the inference result / confidence generated by the particular behavioural biometric microservice, they are forwarded to the fusion mechanism that will create the overall risk score. The risk score will be logged in specific databases allowing the developer to retrieve the risk score of a user during a particular period, but also maintain the ability for future audits. Finally, the result of the inference process can be forwarded to either another backend platform for further processing or could be directed back to the data collection SDK.

2.3.1.2 Implemented features

This section details the implemented features of the user continuous behavioural authentication component with respect to the most common use cases that the component could be utilised in, and the way these features could be utilised by the end-users. The features showcase also the uniqueness of the component with respect to the state-of-the-art mechanisms in the area of user continuous behavioural authentication, indicating the progress beyond state-of-the-art and the value provided to the organisation

that will incorporate the component, as well as for the end-user that will use the component in order to access the TANGO services.

- **Continuous behavioural authentication** that **does not require any user input**. The component provides the ability to the host app to continuously assess who is accessing the app during the whole session. In this way a continuous secure channel is created between the user and the app, during the whole session when the user is accessing and managing the target app. The component collects behavioural data from different types of sensors, device usage events and other sources, and combines the data in order to continuously generate a risk score regarding the particular user that is using the host app at the specific point in time. The collected data are generated primarily based on different types of events that the user creates implicitly through their interaction with the device. Device sensor data such as from gyroscope, accelerometer, magnetic field and GPS are periodically created based on the initially defined data collection frequency. Other types of data are generated based on historical data and data that are generated on the fly, for example when someone is performing a financial transaction. Hence, the component retrieves data that are both periodically generated as well as when certain types of events are generated, thus creating a continuous generation of behavioural data. Once a batch of data is forwarded to the backend platform, the corresponding behavioural microservice performs the appropriate computations in order to generate and propagate the estimated risk score for the particular user. The risk computation is performed every time a batch of data is received, creating a continuous estimation of user authenticity.
- Authentication that combines multiple behavioural traits such as **biometrics, human, device and transactional behavioural patterns**. The component leverages multiple sources of behavioural data including different type of sensor and device usage data. Each of the different types of sensor and device usage data points enable the detection of a particular behavioural pattern that will generate a certain confidence score for the authenticity of the user based on the specific behavioural pattern. Each of the behavioural patterns is able to provide a certain confidence about the authenticity of the user, with a measured level of error, meaning that each behavioural pattern has a certain amount of accuracy and error. None of the existing methods is able to provide 100% accuracy. For that reason, the component builds on the specific assumption is focuses on taking into account the error introduced by each of the behavioural patterns, and by trying to fuse the various behavioural patterns, it targets on compensating the error of each individual behavioural pattern, thus striving to achieve an even higher accuracy of the authentication score. The component leverages the available behavioural patterns and biometrics based on the device's sensing capabilities. For the biometrics part, the component incorporates facial recognition that operates independent of the device, while collects various behavioural data that enable the understanding of the way the user uses the device and other behavioural patterns.
- Setup with **only one selfie photo or a passport**. The components requires an image of the person for which the behavioural and biometric profile is being set up, when the service is running for the first time. The image of the person needs to be provided to the component in a trustworthy manner. For this reason, the component incorporates a secure facial image acquisition mechanism that includes the appropriate security and pre-processing mechanisms to prevent any falsification of the image as well as to retrieve a high-quality image of the user. The mechanisms employed at the specific setup process of the user continuous behavioural authentication mechanism as similar to the mechanisms defined in the User onboarding mechanism of the user (See previous component) when the user is about to take a selfie. In particular the step incorporates a robust face detection mechanism, blurriness detection, liveness detection and a biometric profile extraction mechanism. Depending on the use case, the component could be interconnected with the a trustworthy backend service that could provide a trusted facial image of the user, and perform the profile setup based on that image, while skipping the initial profile setup where the aforementioned user interface is prompted to the user.

- **Primary standalone continuous authentication** that fully discards passwords. The component is able to operate as standalone user continuous authentication. This means that the component can substitute any existing old-fashioned username/password-based authentication mechanism. Through a unique pseudo-id that is provided by the host platform, in this case the TANGO platform (SSI component), the component creates a unique profile of the user at the component backend. This unique id constitutes the identifier that enables the differentiation among the different profiles. This unique id is then couple with the behavioural and biometric data. Every time a new user signs up for the component, the process with the selfie described above is performed. When the user is about to login, depending on the use case, the most common approach is that the user provides the unique id to the component and then the biometric and behavioural elements are engaged, and initiate the 1-1 comparison of the biometric and behavioural data retrieved at that point of time with the existing profile that has been created for the user. Once the authentication is completed, the user is able to access the functionalities of the host app, while continuous assessment of the user is taking place throughout the whole session, until the user logs out or closes the session/app.
- **Secondary authentication** that replaces multi-factor authentication such One-time-passwords and provides **PSD2 compliance**. The component is able to operate both as primary and as secondary authentication for the user. As a secondary authentication mechanism, the host app that integrates the component maintain the existing way of performing authentication, for example using username and password, and then the user continuous behavioural authentication component perform continuous assessment of the user from the point the login has been successfully completed, until the session of the user ends or the user logs out. Instead of having a single one-off authentication mechanism, through the user continuous behavioural authentication component, the user experiences a secure communication channel with the host app throughout the whole duration of the session. The component leverages various behavioural and biometric elements that have been recognized by the European Banking Association as compatible with the Strong Customer Authentication Directive of PSD2. Thus, when the component is leveraged in a banking use case, the risk and authentication score provided by the component could be used by the host app in order to approve particular transactions that require the enforcement of the Strong Customer Authentication requirements, replacing existing vulnerable and non-user friendly mechanisms such as one-time passwords.

2.3.1.3 Software artifacts

The User continuous behavioural authentication component constitutes primarily of two main software artifacts: a) the data collection SDK that is responsible for the collection of the behavioural data and b) the backend that performs all the processing and inference of the behavioural elements in order to generate an overall risk score for a particular user at a specific time period. Each of the two software artifacts are detailed below:

- **Data collection SDK.** This software artifact is responsible for standardising the data collection process from the device and then communicate the data to the backend where the actual processing and inference of the user behaviour will take place. The user behavioural authentication component requires various types of a data in order to be able to infer the various types of behaviours and fuse them into an overall risk score than will enable the authentication of the user. The data collection SDK is responsible for the collection of the relevant behavioural data that include sensor data (such as accelerometer, gyroscope, magnetic field, geolocation, Bluetooth and WiFi traces etc.) and device usage (such as data generated through the interaction of the user with the screen of the device such as touch events, gestures, swipes, typing keystrokes etc.). The data are pre-processed and converted into the appropriate format. Following the conversion of the data to the suitable format, the software artifact incorporate a secure communication module that is responsible for establishing a secure communication with the backend and transmitting the behavioural data whenever the data are generated by the module.

- **Backend.** This software artifact provides all the processing and inference capabilities of the user behavioural authentication component. Through the secure communication channel established with the data collection SDK, the backend component retrieves the appropriate information from the client in order to feed the behavioural models towards inferring about the behavioural traits of the user, and fusing the information into an overall risk score. The component has been developed through a microservices architecture. An API gateway is responsible for the communication of the backend with the internet. Behind the API gateway there are various microservices that process specific types of data. Once a bunch of data points arrive at the backend, the API gateway is responsible for applying the appropriate security measures including authentication between the data collection SDK and the backend. Once the security checks are passed, the API gateway selects the appropriate microservice that is responsible to process the specific types of data. The data are then forward to the specific microservice and are processed in order to generate a confidence regarding the particular batch of data, which are in essence compared with the profile that has been created about the specific user. The microservice will generate a confidence about the specific user, which is the forwarded to the fusion mechanism. The fusion mechanism is responsible for collecting all the confidences generated by each of the microservices that process behavioural data, and then fusing the confidence scores created into an overall risk score regarding if the user the same with the user that initially created the profile. This risk score is the forwarded either to the data collection SDK or to another backed of the organisation that would like to process that score. Where the actual score will be forwarded depends highly on the actual use case and how the organisation would like to utilise the particular score.

2.3.2 Demonstration description

In order to demonstrate the user continuous behavioural authentication, a demo mobile banking app has been developed. The demo banking app emulates the environment of a hosting app that would be integrated with the user continuous behavioural software artifact and then would leverage the continuous security. The integration of the software artifact has been performed in such manner that the user continuous behavioural authentication can operate both as primary and as secondary authentication. This means that the initial login of the user is performed through the particular software artifact without requiring from the user to apply another authentication mechanism. Furthermore, once the user logs in the component performs continuous assessment of the user without requiring any active engagement, while the user is able utilise the app as usual, performing transactions, viewing previous transactions etc. Once an imposter is accessing the demo mobile banking app, then a lock screen will pop up and inform the user that they were not able to be authenticated, and that the legitimate user should use the device. The demo banking app has been developed for demonstration purposes in order to showcase the functionalities of the user continuous behavioural authentication, how the integration takes place, what is the user interaction and experience when utilising such technology and furthermore for testing purposes. A demonstration video has been recorded of the interaction with the demo banking app that integrates the user continuous behavioural authentication.

2.3.2.1 Docker/API description

This section provides an overview of the API description provided by the User continuous behavioural authentication component and in particular of the data collection SDK that integrates with the host app, in this case the TANGO wallet. The description of the API incorporates the exposed classes and methods that are provided to the developer in order to facilitate the integration process and how the actual implementation of the data collection SDK inside the TANGO wallet will take place.

Table 2. API Calls

Class: BehavAuth		
Method name	**Parameters**	**Description**

Class: BehavAuth		
getInstance	-	Returns the singleton instance of the BehavAuth class
Install	- android.app.application - String (apiSecret) - int (mode)	Setup the BehavAuth library, to start the data collection and processing. For 'mode' see constants below.
Start	- RegistrationCallback - String (externalID) - Long (expiresAt) - String (verificationSecret)	Start the BehavAuth continuous authentication. If it is the first time it is launched, then it will perform a registration and initialisation process.
Stop	-	Stop the continuous authentication when the app is in the background or when it is closed/killed.
setCallback	- AuthenticationCallback	Set the callback to inform the application when the user is legitimate and when the user is not authorized.
deleteProfile	- DeletionCallback	This deletes the profile of the user and also deletes any data and knowledge extracted until that point.

Class: BehavAuth

Constants

Name	Value	Description
HIGH_ACCURACY	0	This mode provides the highest accuracy and requires that the user uses and interacts with the phone. Actions such as leaving the phone on the table will immediately trigger an authenticationFailed() event.
USER_EXPERIENCE	1	This mode is similar to previous one but is more tolerant. Actions such as leaving the phone on the table will trigger an authenticationFailed() event after 8 seconds.
PASSIVE	2	This mode will provide events only when an imposter has taken the device. Actions such as leaving the phone on the table will not trigger any event.

Class: RegistrationCallback		
Method name	**Parameters**	**Description**
onRegistrationSuccess	-	Called when the registration was successfully completed. In case it was interrupted, the registration will continue from the previous point.
onRegistrationFailure	-	Called when the registration failed

Class: PredictionCallback		
Method name	**Parameters**	**Description**

onSuccess	- BehavAuthPrediction (prediction)	Called when the prediction was successfully derived.
onFailure	- String (reason)	Called when the prediction failed due to various reasons e.g. error in communication

Class: AuthenticationCallback		
Method name	**Parameters**	**Description**
onAuthenticationSucceeded	-	Called when the authentication was successfully completed.
onAuthenticationFailed	-	Called when the authentication failed. This means that either a non authorised user has access to the device or the user is not using the device.

Class: DeletionCallback		
Method name	**Parameters**	**Description**
onDeletionSuccess	-	Called when the user was deleted successfully.
onDeletionFailure	-	Called when the user was not deleted successfully.

Support for pilots

The User Behavioural Authentication component, via TANGO will exhibit its capabilities through diverse industry sectors, showing innovation and enhanced efficiency. In each use case TANGO will be committed to innovate, enhance security and privacy and bring transformative solutions to all the diverse industries that are listed below:

- **Smart hospitality.** This component will enhance the customer experience by providing continuous authentication of the guests and the employees of the hotels, through continuous assessment of their interaction with online services including the remote identity verification service, while for employees' continuous assessment with the hotel management software. TANGO will be facilitating a smooth, comfortable and secured check – in process for hotel visitors using just their smart devices Guests will be able to bypass the time – consuming reception check – in. Until now regulations and technology limitations do not allow the direct to room check – in process as guests must ensure legitimate access to the hotel and provide their essential personal data to reception face to face. In TANGO use case scenario this will be achieved with just a tablet at the hotel reception. Hotel visitors will input their personal preferences and other information in the tablet. This data will be combined with information from in – room sensors and will configure the room conditions as well as offering recommendations of hotel activities or tailored menu lists. Guests will be able to observe their impact on energy savings, organic waste management and sustainability. All personal information data will be following GDPR regulations and will be applied to guests' rooms without compromising data breach. In this use case scenario, there will be tested also attempts to impersonate another guest or exchanges of smartphones with different user data. Finally, participants will also provide feedback on user satisfaction and their overall experience within this framework.
- **Autonomous vehicles.** This component will allow the combination of distributed identity and trust management in the process of hiring an autonomous vehicle. Autonomous vehicles share

data with other cars and users as well as companies. So, the need to ensure the authenticity of the user that is accessing a particular autonomous vehicle, and to secure this data from any leak and possible modifications is great. This TANGO use case scenario will provide solutions to guarantee that data sharing inside the autonomous vehicle network will be protected. The user continuous behavioural authentication will perform continuous assessment of the user that is accessing a particular vehicle through the TANGO wallet app, in order to provide access to a specific vehicle. In this way, the autonomous vehicle owner will be reassured regarding the authenticity of the person accessing the specific vehicle.

- **Smart Manufacturing.** This component will ensure robust security across aspects such as data handling, sharing, storage, as well as authentication, access rights and management of IoT devices and 3D printers. In essence, the user continuous behavioural authentication component will provide the ability to factories, to continuously assess the authenticity of the workers, employees and contractors that will use their TANGO wallet app in order to access particular infrastructure and data. A similar approach will be followed in both use case scenarios defined in the TANGO project. Access control is a major and important aspect of smart factories that requires strict rules about the access that different stakeholders have to different types of data and infrastructure. Lack of continuous assessment of these key stakeholders could lead to security holes and potentially to data breaches, as well as to unauthorised access to equipment and areas that may lead to potential accidents due to lack of training.
- **Public Administration.** This component will focus on enhancing the security of the TANGO platform when assisting in the process of visa applications, in particular when visa applicants are about to have access to their personal data, while administrators from the public organisation are about to have access to the visa applicant's data, continuous assessment of the user will be enforced. In this way, the user behavioural authentication component will reduce the bureaucratic & high-risk processes for visa applications. This use case will enhance the privacy, transparency as well as the security of data gathering and processing when a citizen applies for a visa. Continuous assessment of the user when accessing the personal data stored for the public administration including visa application scenarios is essential, taking into account the amount of data breaches taking place as well as the continuous increase in the number of identity fraud cases. Furthermore, personnel from public organisation will have access to personal data and thus need to enforce particular measures for securing the access control.
- **Retail.** This component will focus on providing continuous behavioural authentication to the TANGO platform by assessing various behavioural patterns of the user, in order to ensure that only the correct user has access to the particular data of the customers of the retailers, in order to enable the TANGO platform to secure and private exchange of data for personalised shopping lists and recommendations to consumers and retailers. Regarding the TANGO platform, historical anonymized data that have been collected the past two years will be used in order to train the AI algorithms on shopping preferences of consumers and retailers. The most important aspect of this use case is to ensure efficient data exchange mechanisms in the retail sector while being complied with all relevant regulations. This use case will also facilitate interconnection between different corporate systems and implement also federated learning mechanisms along with tokenization to create personalized recommendations in a privacy preserving manner for all stakeholders. Regarding the user continuous behavioural authentication, it will ensure strong access control to the data collected by the TANGO platform.
- **Banking.** This component will enable the involved banks to incorporate in the federated learning mechanism, a robust transaction fraud detection model. The model will be trained leveraging the data across different banks involved in the use case scenario. An initial model will be trained through a client across each of the banks. Then the weights that have been trained by each of the banks, will be retrieved and through an averaging mechanism, a set of optimal weights will be derived, evaluated and then distributed to the banks in order for them to receive the optimal set of weights to be used in the transaction fraud detection mechanism. In this way, a federated learning approach will be introduced, where the transaction fraud detection mechanism will be trained leveraging knowledge from not only one bank, while fulfilling the

banks' requirement of not sharing any data outside the bank itself. Following this approach, a more generic and robust transaction fraud detection mechanism will be introduced to the banks, enabling them to tackle the continuous problem of financial fraud.

2.3.3 Future work on the component

Following the initial implementation of the user continuous behavioural authentication component, the focus of the future work on further development of the component will be on improving the reliability of the inference results in various conditions and use cases. As the pilots incorporate the user behavioural authentication, additional behavioural data will be collected accompanied by the feedback of the users regarding the results produced by the user behavioural authentication. Through the feedback, the component will further tuned in order to fit the particular pilots and use cases considering how strict and how user friendly the component should be depending on the requirements of the each of the use case scenarios. Improved models will be trained for the device usage patterns such as the typing and the swipes of the users. The collected data will be explored in order to understand the abilities to prevent fraud in the onboarding based on the behavioural patterns of the user and prevent malicious users from creating malicious profiles on the TANGO platform.

2.4 Device behavioural authentication [T4.4]

2.4.1 Component description

The aim of the device behavioural authentication mechanism is to provide continuous assessment and identification of a device identity in real time when the device operates. The continuous authentication mechanism relies on a risk assessment engine that performs continuous data processing and analysis and confirms the authentication of the user's device while it operates during the whole session. The difference between device vs. user continuous behavioural authentication is the need to distinguish from physiological and user behavioural metrics and leverage data and information stemming from the device operation and not directly from the users. Device continuous behavioural authentication can be complementary to user behavioural authentication. The following diagram presents the flowchart of the continuous authentication processes. Initially, we assume that a user is already authorized and uses his/her device. The continuous data acquisition process collects the device system logs. Then those data are pre-processed for the extraction of specific features of the device and its operational behaviour. In sequence, based on those features the authentication algorithm conducts a risk assessment and decides whether the retrieved data from the system logs match the device's normal behavioural operation. A feedback loop is used to update information about feature extraction and finally the authorization session is retained or not. This implies that once the users are logged in the application, the authentication undergoes continuous monitoring. The application continuously captures the Android Log Data to observe the device's behaviour, determining whether the current behaviour aligns with the devices' expected behaviour. Based on this assessment, the system decides whether to retain the user's logged-in state or prompt for additional credentials, such as device's PIN.

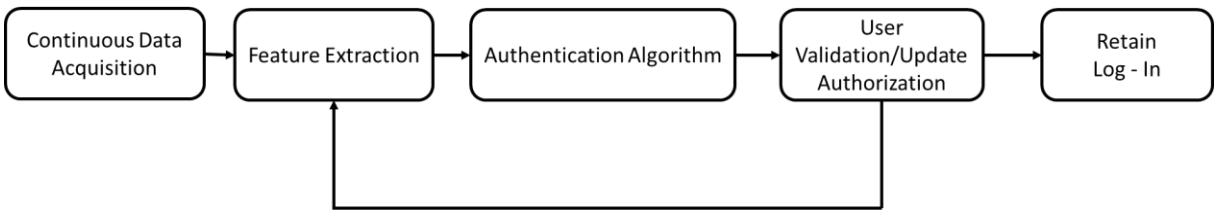


Figure 8. Continuous Authentication Process

The implementation consists of two components:

1. The Backend Server: The placement of the server deployment is located in the cloud. The server is developed in JavaScript, using the Node.js JavaScript's runtime environment and manages

the user authentication, registration, login & logout requests. It is responsible to capture continuously and store the Android System Logs of the Device, associated with the user who is logged in the application.

2. The application 'Authenticator': the client-side component of the project. It's an Android application developed in Java, using Android Studio. It is currently installed on an Android 10.0 device (Lenovo Tablet). This application is responsible for user interactions, such as login, registration, and logout, as well as running foreground services for capturing Android System Logs.

Important Note: The SSI concept, at the time it will be available, will be used for the authentication. For the development and testing purposes of the device behavioural authentication mechanism, a simple custom authentication mechanism was implemented, which will be replaced by the SSI mechanism.

2.4.1.1 Internal architecture

Current Issues and Assumptions

In order to collect the Android System Logs of the device, a specific setup is required. Due to restrictions on accessing the Android Log's folder in non-rooted devices, attempting to execute the 'adb logcat' command within the application results in the error: *'Cannot run program "adb": error=13, Permission denied'*. Android does not allow the execution of adb commands and the collection of the Logs directly from the device. This limitation is part of Google's efforts to enhance user privacy and security, so applications running on Android 9.0 or higher, will encounter permission errors and issues, except System Apps, which are the by-default installed apps in the System. Also, the attempt to execute the 'logcat' command and save its output in the external storage of the device revealed that this command provides logs exclusively related only to the application's functionalities. So, the current method of capturing and collecting these logs, consists of some steps, which are also presented within the application to guide users in preparing the entire setup:

- Navigate to 'Settings' -> 'About Phone' -> Tap 7 times -> Go Back and find 'Developer Options' -> Locate 'USB Debugging' and enable it.
- On the Server side, the command 'adb tcpip 5555' is executed in order to switch the ADB Connection to TCP mode, allowing the remote collection of Android logs. The setup requires both the server and the device to be connected to the same network.
- Server connects to the device, by executing: 'adb connect <Android_Device_IP>: 5555'

Internal Architecture and Functionalities

The application communicates with the server side, by making the proper http requests on the server's URL Endpoints. Our implementation performs functionalities, such:

1. User **Login**
2. User **Register**
3. User **Logout**
4. **Foreground services**

To continuously capture and collect the Android System Logs of the user's device, it is necessary to create foreground services that will run continuously, even if the application is terminated or even if the user does not interact anymore with the application, but still remains logged in.

The primary objective is to consistently identify users, based on the various behaviours of the device. For user authentication, a public-key cryptography is employed along with password. As soon as the user is considered authenticated, server continuously captures and stores the Android system logs, offering valuable insights into the device's behaviour, including Network and Bluetooth connections, along with Battery statistics. Since information about the Location of the device can't be provided by the Android System Logs, the user's permission is granted in order to collect data about latitude and longitude. The future implementation will utilize *https* protocol, incorporating SSL (Secure Sockets Layer) and TLS (Transport Layer Security) protocols, to encrypt the data transmission between users and server.

2.4.1.2 Implemented features

A closer examination of the implemented features consists of:

1. The **Registration** process: User attempts to create an account.
2. The **Login** process: User attempts to be authenticated.
3. The **Logout** process: User attempts to logout.
4. The **Android System Log Capture** process.

REGISTRATION

- The user enters credentials (**Username**, **Password**) in order to register in the application.
- If the credentials are valid:
  - Server generates and sends a Register Challenge to the user.
  - User creates a pair of public-private key and signs the Register Challenge using the private-key, creating the Register Signature.
  - This Register Signature, along with the corresponding Public Key and the original Register Challenge, is sent back to the server.
- If the Signature is valid, the register request is accepted. Along with the name of the user, the Public Key, the password (encrypted) and the user's device are also stored.

For example, when a user with username 'Jason' is registered, his credentials are stored in JSON format:


```
"type": "AuthGuard-CLIENT",
"credential": {
  "rp": {
    "name": "NitLabAuthGuard-Server",
    "id": "http://10.64.45.235:3000"
  },
  "user": "Jason",
  "challenge": "r2bWYo0K9c76UTTkLQHDGsQcAtdPgmVzpAJtP/Z3UjY=",
  "pubKeyCredParams": [
    {
      "type": "public-key",
      "alg": -7
    }
  ],
  "timeout": 60000,
  "attestation": "direct",
  "PublicKey": "-----BEGIN PUBLIC KEY-----\nMFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAE3IvBGMMX2P/Iw1Tglqd9fhRtL6mqmxuBXp
  "password": "$2b$10$Kyj2g1rn8Kn0rzzzbq3CBOus/.Q1nqf9WXRVqMek85C7ocQw3M.12",
  "ConnectedDevice": "Lenovo TB-8505F"
```


Figure 9. User's Credentials stored in server

In the above figure, details about the relying party, user's username, Public-Key (in PEM format), encrypted password and user's device name are illustrated.

A simple scheme that describes the whole Registration is illustrated in the figure below:

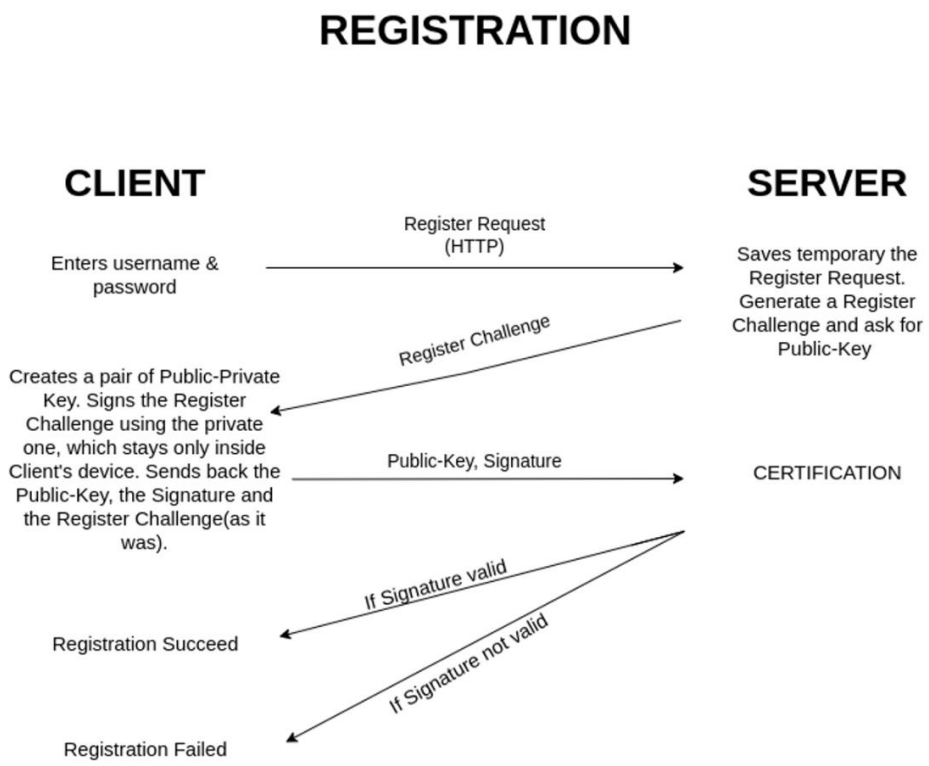


Figure 10. Registration Procedure

LOGIN

- User enters **Username**, **Password** in order to login.
- If both of them are valid and correct:
  - Server generates a random Login Challenge and sends it back to the client, in order to sign it with the private key that resides only within the user's device.
  - User retrieves the corresponding private-key and signs the challenge, producing a Login Signature, which is sent back for certification.
- Server searches in the database for the corresponding public-key that is associated with the user's username, retrieves it, signs the original challenge and then compares the two signatures together. If the client's Signature is valid, user is authenticated successfully, else the authentication process has been failed.

A scheme describing the Login (Authentication) process, is shown in the image below:

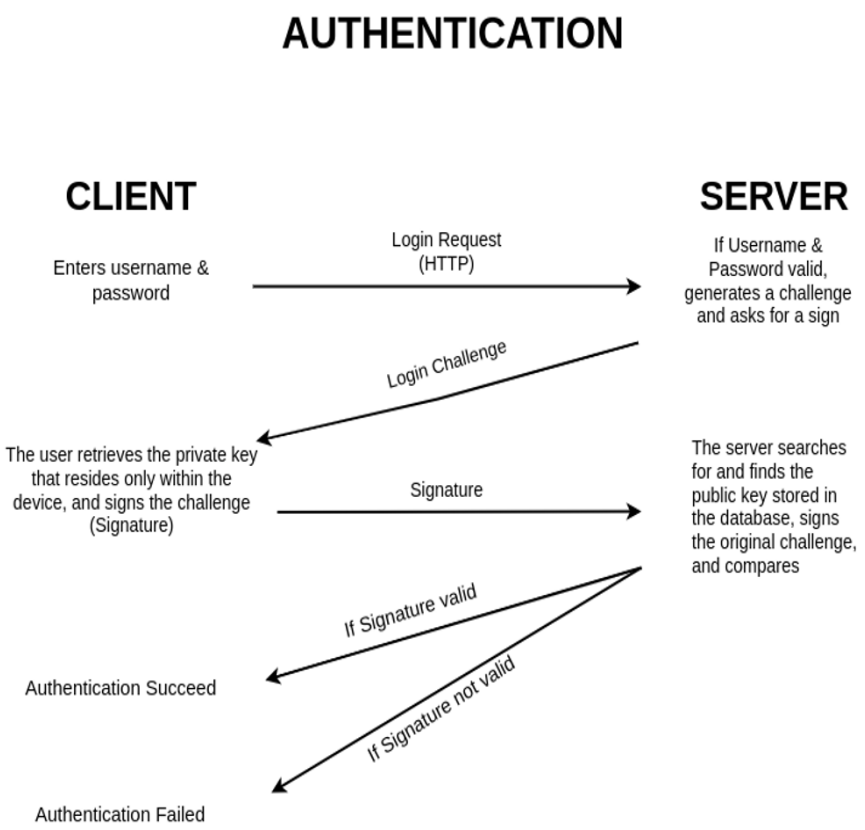


Figure 11. Authentication Process

LOGOUT

When a user is logged in, the server executes the necessary ADB commands, resulting in active PIDs on its system. So, for a user that is logged in, several active PIDs in the server side are stored:

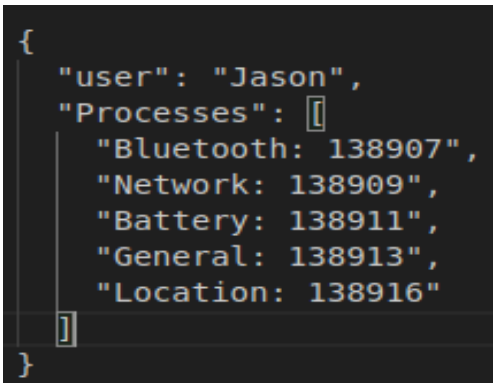


Figure 12. Active PIDs stored in the server

The above image illustrates that for the user 'Jason', there are currently five (5) PIDs on the server side, each responsible for executing a different ADB Command (Bluetooth, Network, Battery, Location, General). When the user decides to log out, an HTTP request is sent to the server's designated endpoint URL, containing the username. Once the server receives the request and parses it, it locates the user in the file containing active PIDs and terminates these PIDs that were capturing the Android System Device's Logs (Bluetooth, Network, Battery, Location, General). Then, server proceeds to log out the user.

ANDROID SYSTEM LOG CAPTURE

The files containing the device's logs are located within a directory named 'LOGS'. For each user, there are five (5) different text files, each one containing logs related to its interface. For example, the folder of a user ('Jason'), looks like this:

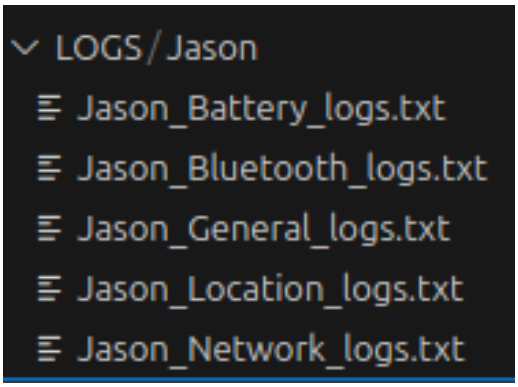


Figure 13. Android Logs Storage Directory

Where:

- **Username_General_logs.txt**: Contains all the Android System Logs.
  Command: adb –s DeviceName logcat
- **Username_Network_logs.txt**: Contains Logs related to Networking.
  Command: adb –s DeviceName logcat –s {Network Tags}
- **Username_Bluetooth_logs.txt**: Contains Logs that related to Bluetooth.
  Command: adb –s DeviceName logcat –s {Bluetooth Tags}
- **Username_Battery_logs.txt**: Contains Logs related to the Device's Battery.
  Command: adb –s DeviceName logcat –s {Battery Tags}
- **Username_Location_logs.txt**: Contains Logs related to the Device's Location.
  Command: adb –s DeviceName {Location Tags}

A table of the several tags in Android 10.0 that provide information about the different interfaces, are shown in the table below:

Table 3. Interfaces and their general Android Tags

NETWORK TAGS	BLUETOOTH TAGS	LOCATION TAGS	BATTERY TAGS
WifiNetworkSelector	BluetoothManagerService	LocationService	StatusBar
IpClient/wlan0	BluetoothAdapterService	LocationManagerService	LPPeService
wificond	BluetoothAdapter		UserExperience
MtkConnectivityService	CachedBluetoothDevice		
netd	BluetoothDatabase		
WifiService	BluetoothEventManager		
WifiCountryCode	BluetoothAdapterService		
WifiNative			
WifiCond	BluetoothDevice		
WifiNl80211Manager			
WifiScanRequestProxy			
WifiManager			
WifiScoringParams			
ConnectivityService			
WifiScoreReport			
NetworkMonitor/100			
WifiClientModeImpl			
8021q			
NetworkStatsObservers			
DhcpClient			

As the entire Android Logs are becoming more complicated during the collection, specific tags that are crucial and provide essential information about the device's behaviour are selectively retained by the server:

Table 4. Bluetooth Tags and related Functionalities

BLUETOOTH TAGS	FUNCTIONALITIES
BluetoothAdapter	Bluetooth is ON/OFF
CachedBluetoothDevice	Informs when there is a connection/disconnection to/from a Bluetooth device. Additionally, displays the devices that were connected via Bluetooth in the past
BluetoothEventManager	Informs about the nearby available Bluetooth devices
BluetoothActiveDeviceManager	Informs about the name of the currently connected Bluetooth device

Table 5. Battery Tags and related Functionalities

BATTERY TAGS	FUNCTIONALITIES
StatusBar	Provides information related to device's power connection status (whether it is connected to power or not). Also, displays the current percentage of the battery when it's plugged into power
LPPeService	Informs when the device's battery percentage changes (+/-), by displaying the current battery's level

Table 6. Location Tags and related Functionalities.

LOCATION TAGS	FUNCTIONALITIES
LocationService	A custom Log Tag that is generated by the '*Authenticator'* application, in order to collect details about the Longitude and Latitude of the device (requires user's necessary permissions)

Table 7. Network Tags and related Functionalities

NETWORK TAGS	FUNCTIONALITIES
WifiService	Wi-Fi is ON/OFF
WifiNetworkSelector	Provides information about the nearby available networks, filtered based on low signal strength if not connected to any specific one, or about the network that device is currently connected, each short time intervals
WifiClientModeImpl	Defines the WLAN MAC when device is connected to a network, by displaying default and the new one
MtkConnectivityService	Informs about the network's SSID and BSSID that device is about to be connected. When the connection is established, it provides information about the IP Addresses of the device within the Network, the DNS Addresses, and the gateways
DhcpClient	Informs about the ACK and OFFER packets when device is connected to a network. Also provides information about the gateways, DNS servers, and the device's IPs
IpClient/wlan0	Updates the device's IP Addresses (IPv4, IPv6 if exists) and removes them when the device is disconnected from the connected network

The Android System Logs of an Android 10.0 Device, are as follows:

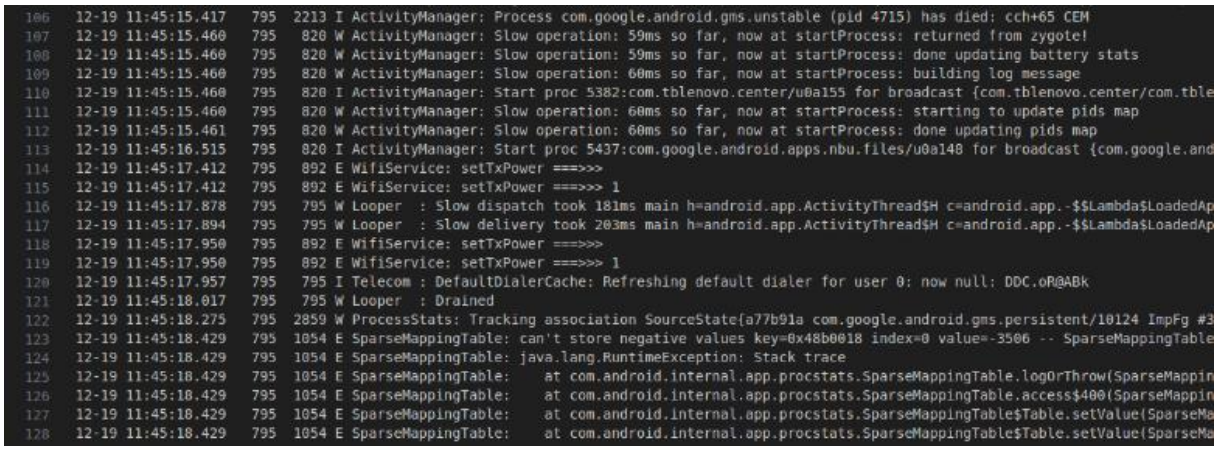


Figure 14. Raw Android System Logs of the Android Device

Then a **Python Script is executed**, that parses these raw Log Data and extracts the specific tags mentioned in the table above, enhancing the clarity of the logs. Now, the processed Log Data look like this:

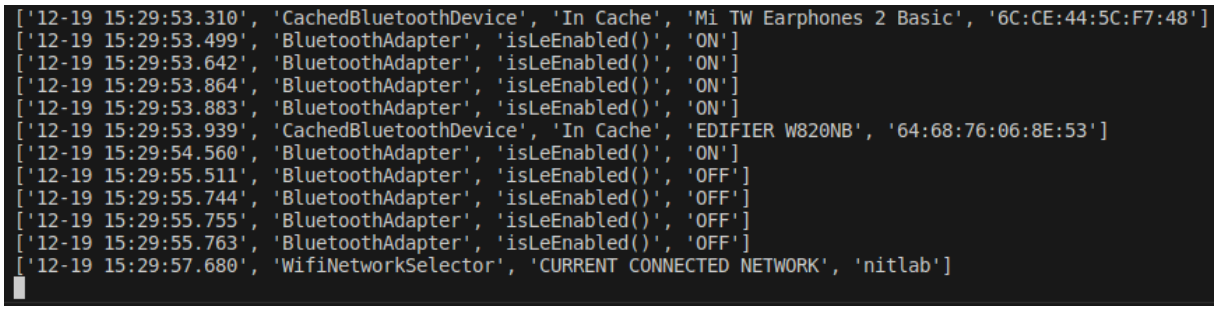


Figure 15. Processed Android System Logs of the Android Device

For each Android Log tag, its timestamp is included. In the above image, the 'CachedBluetoothDevice' displays information about devices that are already in cache, providing their MAC addresses. 'BluetoothAdapter' informs when the user turns ON/OFF the Bluetooth. Also, in this example, the device is already connected to the network 'nitlab', so the 'WifiNetworkSelector' displays the current connected network at regular intervals. This approach ensures a better understanding of the Android System Logs and their use-cases/functionalities, serving a necessary step in order to begin the classification process for the neural network's input, which is currently under development.

2.4.1.3 Software artifacts

In this section the analysis of server's endpoints and their functionalities takes place. For the better clarity, these endpoints will be categorized into the following groups, and the complete description is given in Annex B:

- **REGISTER** Endpoints
  - The registration endpoints allow for the registration of new users and couple devices to their identity.
- **LOGIN** Endpoints
  - The login endpoints represent the list of permission is requested. It can be a single scope or a list of scopes together. In the latter case, the scopes must be written as a space separated list of values.
- **Android System Logs Capture** Endpoints
  - The logging endpoints represent a list of the recording activities that occur in the system, monitoring the creation, update, and delete of activities e.g., the user or the service that conducted changes, the time when that happened, and what was changed.

- **LOGOUT** Endpoints
  - The logout endpoints redirect either to an authorized sign-out URL for your app client, or to the login endpoint.
- Other Endpoints
  - These endpoints are responsible for the setup's configuration and all the necessary operations in order to keep the user's device continuously connected to the Server.

2.4.2 Demonstration description

'Authenticator' - APPLICATION

The application complexity increases with approximately twenty (20) different classes. Some of them handle services, such as managing the Wi-Fi and USB-Debugging features, which, if they are disabled by the user, the Android System Logs capture by the Server, is prevented. There are also classes for the Login Page, for the Register and Logout. Additionally, user's permissions are requested to gain access on the Device's Longitude/Latitude at any time. Indicative, some images of the application can be shown below:

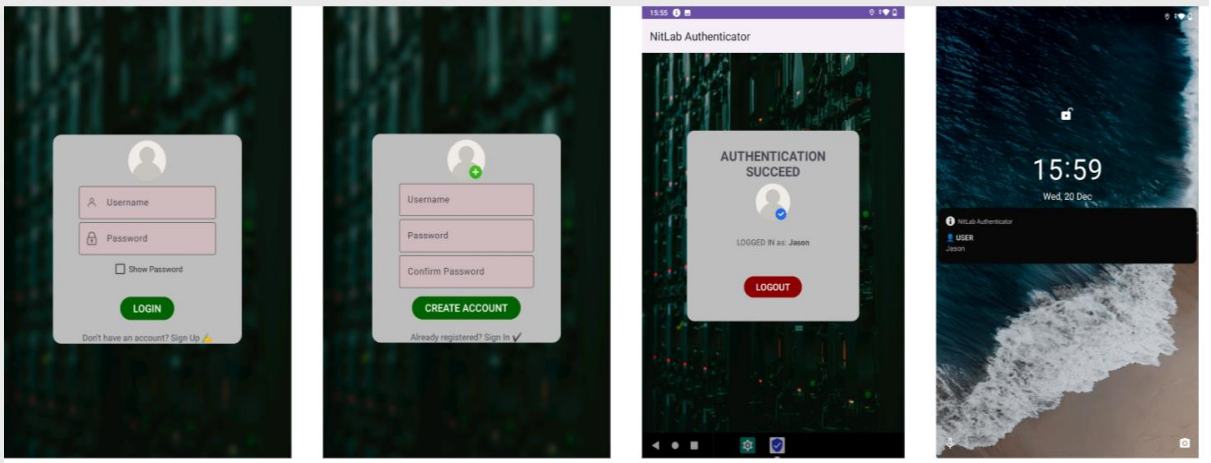


Figure 16. Screenshot of the application "Authenticator" in the device

The first image depicts the Login Page, while the subsequent shows the Register. The third one displays the username of the user who is currently authenticated and logged in. The final image illustrates one of the application's services. This service is responsible for establishing communication with the server when the user is logged in, even if the application is terminated or closed, allowing to continuously capture the device's Android System Logs.

2.4.2.1 Docker/API description

As mentioned earlier, the current deployment includes both the Server and the 'Authenticator' Android application. The application is designed to run natively on Android devices and doesn't require containerization. It can be installed using its .apk file, which is the standard installation file format for the Android operating systems. This file contains all the necessary components for installing an Android application on a device. Nevertheless, it can be deployed in a Docker Environment if the need arises.

Regarding the Server, as part of the ongoing efforts to enhance scalability, plans will be in progress to containerize it in a future implementation, if deemed necessary. This approach, ensures that server's deployment in Docker remains adaptable to varying conditions, allowing it to operate in isolated runtime environments as designed, facilitating easier deployment and maintenance.

2.4.3 Support for pilots

The Device Continuous Behavioral Authentication support the following pilots:

- **Autonomous vehicles.** The Device Continuous Behavioral Authentication (DCBA) will be used in this Pilot to provide continuous authentication services. The DCBA mechanism will

assess the behavioral performance of a device based on power consumption and network analytics --and will be able to detect deviations in comparison to their normal operation based on specific performance efficiency metrics (e.g. power consumption, RSSI, network traffic metrics) and infer whether a specific operational behavior is suspicious or not. So, the DCBA mechanism will be able to assess the continuous authentication procedure and enable the authentication of the trusted user while sharing cars.

- **Smart Manufacturing - Case 2 (RIA STONE).** The DCBA mechanism will assess the behavioral performance of a user's device. The device is used by the workers to access specific regions in the factory. Not all the workers are allowed to enter or have physical access to all the factory's places. For security reasons, workers according to their roles and responsibilities are restricted to specific factory areas. Therefore, using the DCBA mechanism will allow the continuous authentication of the user's device when the user access permitted areas of the factory. If the device is placed or found in place where the user is not allowed or permitted to have physical access, then an alert will be created to notify the responsible users/managers and the session will be terminated. Enter your text here.

2.4.4 Future work on this component

The classification method, which is under development, embraces a categorical nearest-neighbour approach. Thus far, the implementation has successfully concluded the Data Collection phase and is poised to complete the Data Preprocessing step. For instance, take this Android Log message:

['BluetoothAdapter', 'isEnabled()', 'OFF']

This log entry signifies that the user has disabled the Bluetooth functionality on his/her device. To facilitate the classification process, this log entry can be represented as follows:

Table 8. Log entry representation

WORD	CATEGORY
'BluetoothAdapter'	1
'isLeEnabled()'	2
'OFF'	3

Each element in the list corresponds to a category or a group. By assigning numerical labels to these categories, the above Android Log message now becomes '[1,2,3]'. This categorical encoding allows to apply distance measures between the unknown and known items, in the context of the categorical nearest-neighbour approach. In upcoming work, once the Preprocessing of the Android Log Data is finalized and ready for consumption by a machine learning model, the development will focus on the rest of the classification process. This includes the construction and training of the neural network, enabling it to recognize patterns and make predictions or decisions based on the several input Android Log data. Our future classification approach will adapt and build upon a similar approach that can be shown, in order to enhance the analysis of Android Log data.

2.5 Side-channel attack hardening [T4.5]

Side-channel attacks are critical for embedded systems and IoT devices. They exploit the link between physical quantities such as the electromagnetic emissions and the internal activity of a chip in order to reveal secret data.

Two classes of countermeasures have emerged in the state of the art: hiding countermeasures and masking countermeasures. Hiding countermeasures consist in lowering the signal to noise ratio, either by lowering the signal, either by increasing the noise. Masking countermeasures is built upon the principle of secret sharing; secret variables are split into several variables that are manipulated separately in order to force attacker to recombine measurements to find the secret.

Our current work within TANGO is mainly focused on hiding countermeasures. In particular, our work builds upon the code polymorphism countermeasure and tries to improve the security level by improving the countermeasure and combining it with other countermeasures.

2.5.1 Component description

Code polymorphism is applied using a compiler named Odo, that we extend within TANGO. The countermeasure application flow is presented in the Figure 17. The developer is in charge of indicating functions that need to be secured. Then, the compiler Odo transforms the C file into a new C file, replacing the annotated function by a wrapper and a specialised generator of polymorphic code (SGPC). The resulting C file is compiled into an elf file using the usual compiler the developer used. At runtime, the wrapper intercepts any call to the annotated function, and calls the SGPC to generate the function's code in memory. It then calls the generated code. Regular calls to the SGPC allow the code to become polymorphic, as the SGPC generates a different code every time thanks to assembly level code transformations.

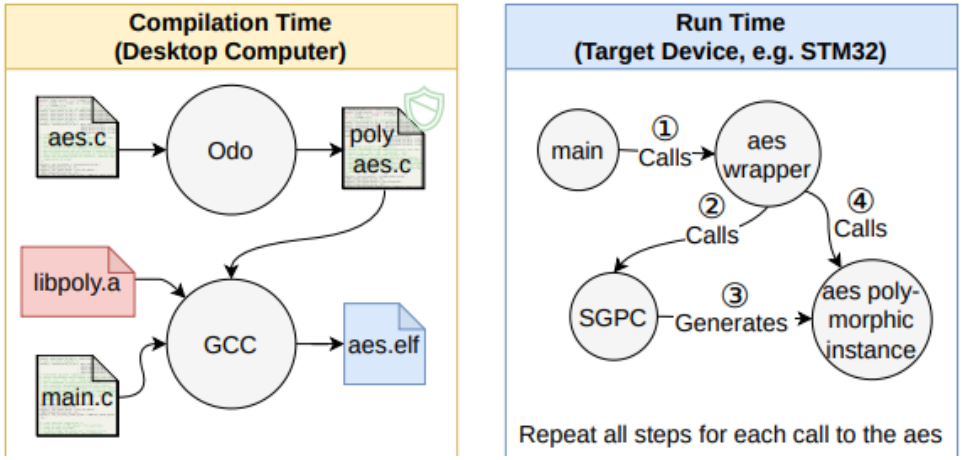


Figure 17: Application flow of code polymorphism

The SGPC has inherent knowledge of a reference assembly code, and generates variants of that code. It supports the following code transformations:

- Random register permutation: the SGPC draws a random permutation among the general purpose registers (except registers used for argument passing). It then uses the permuted register list when crafting instructions. For instance, the SGPC could emit an instruction using R11 even though the reference instruction used R5 instead. The change is global for all instructions, allowing to preserve the program semantics.
- Instruction shuffling: the SGPC randomly chooses the order of instructions that are independent from each other. – Semantic variants: for some instructions, the SGPC knows several semantically equivalent sequence of instructions. The SGPC randomly chooses one of them. As an example, a xor can be replaced by a sequence of an and, an or, and a xor: a $\oplus$ b = (a&b) $\oplus$ (a‖b).
- Noise instructions: the SGPC inserts a random number of useless instructions in between useful instructions. The noise instructions are randomly chosen among various frequently-used instructions.
- Dynamic noise: the SGPC sometimes inserts a sequence of contiguous noise instructions preceded by a branch instruction that will randomly jump inside the sequence during the execution of the polymorphic instance. A register is reserved to hold random data that is used at runtime to determine the branch offsets. This register value is updated throughout the execution by noise instructions, and is saved and restored between calls to make code execution different from one execution to the next. Dynamic noise's purpose is to partly decorrelate what happens during code generation and code execution, and to maintain code variability even in

between calls to the SGPC when the code generation is done less frequently for performance reasons.

The countermeasure can be configured by enabling or disabling any of these transformations. In addition, the insertion of noise instructions can be finely tuned: the probability distribution controlling the amount of noise instructions to insert in between useful instructions can be tuned.

Taken as-is, the code polymorphism countermeasure has room for improvements:

- Only dynamic noise can make loop iterations different from one another during an execution, as the code is only regenerated in between executions.
- 2 different implementations of AES secured with code polymorphism on a STM32F3 were shown very vulnerable to deep learning attacks: after training, about 20 traces were enough to find the secret key.

The loop shuffling countermeasure is another hiding countermeasure, that consist in executing loop iterations in a random order. It is frequently targeted by template attacks in the state of the art: the attacks manage to exploit the leakage of the variable controlling the iterations order.

During the TANGO project, we first aimed at studying the combination of both countermeasures, as well as at improving them. As code polymorphism and loop shuffling act at a different granularity (code polymorphism cannot shuffle large code sequence while loop shuffling can), one can expect to obtain a good security level from their combination.

We performed this study on a custom AES implementation, where we use the following principles:

- Merge all round functions to avoid as much as possible useless loads and stores to the AES state in memory.
- Store the SBox in RAM, as having the SBox in flash seems to make SBox accesses easily visible.
- Perform the mixColumns and addRoundKey on 32bits to speed up computation.

Compared to by-the-book or T-table implementations, our implementation presents an interesting trade-off between memory and execution time, as show in Table 9. Its execution time is significantly faster than the one of the by-the-book implementation, and its table size and code size remain low.

Table 9: Execution time, as measured on a STM32F7, and table size of different AES implementations.

Implementation	Execution time in clock cycles	Table size in bytes	Code size in bytes
8-bits (by the book)	6436	288	724
T-table (Mbed TLS)	1375	4288	960
Ours	2710	464	536

Compared to previous works on code polymorphism, we proposed the use dynamic variants: instead of generating a randomly selected variant, the SGPC generates a switch case between all variants. A register containing a random value allows to execute a different variant every time the code is executed. This transformation aims at introducing variability between loop iterations.

We checked the effect of this transformation by visually inspecting electromagnetic traces captured on a STM32F756ZG. Figure 18 shows electromagnetic traces captured for each considered configuration, averaged over 1000 executions without calls to the SGPC, and resetting the Pseudo-Random Number Generators, in order to improve visibility. A moving average of 3 samples is performed as well. The length of loop iterations in samples is indicated in red. The figure clearly shows that loop patterns are easily visible for all configurations, and that dynamic transformations fulfil their role as the loop iterations length vary significantly for the configuration where dynamic noise and dynamic variants are used. Larger variations could be obtained by using longer variants, to increase variants length variance.

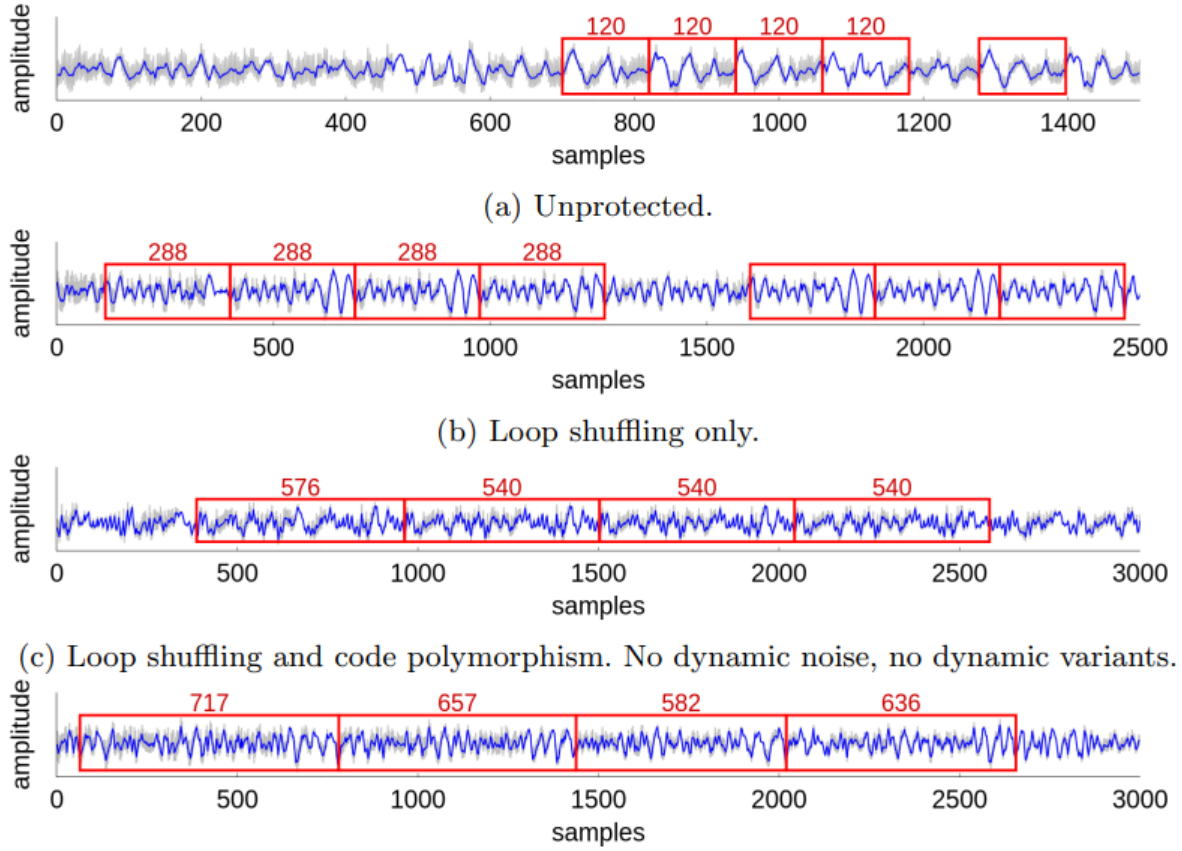
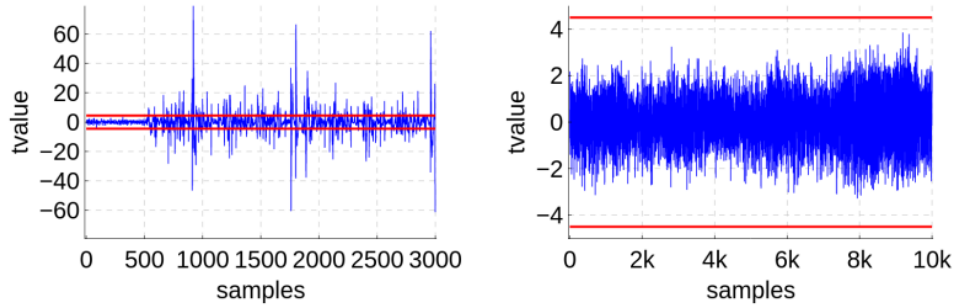


Figure 18: Loop patterns and their size (in samples), as observed on averaged electromagnetic traces.

We also showed how to implement shuffling without compromising on the number of permutations considered nor on execution time. For this purpose, we base our approach on the fact that, iterating on state column, our loop is only 4 iterations long. The 24 possible permutations can easily be stored in memory. We exploit fixed-point arithmetic to draw a random number between [0,24[, to choose a random permutation. In addition, to increase shuffling, not only the iterations are shuffled, but also the shiftRows and subBytes operations. Such operations are applied on 4 different bytes at each iterations. We shuffle at each iteration differently the processing of these 4 bytes. This makes the total number of possibilities significant: there are 4!^5 = 7962624 possibilities, without considering code polymorphism at all.

As mentioned, loop shuffling suffers usually from the leakage of its permutation variable, which greatly helps the attackers. We evaluated this leakage on our implementation, with and without code polymorphism by doing a fixed vs random ttest on the seed of the PRNG used for the permutation generation. More precisely, we gather 50k traces where all permutations choices are always the same across the 50k runs, and 50k traces where permutations are chosen randomly. The ttest compares the electromagnetic emissions measured in both classes. A tvalue larger than 4.5 in absolute indicates that the ttest detects a difference in the traces measured for both classes, i.e. a leakage. Figure 19 shows the result of this ttest. There is strong leakage of the permutation variable for the implementation with loop shuffling and without code polymorphism. However, code polymorphism effectively hides the leakage. As such, code polymorphism helps the loop shuffling countermeasure to keep its efficacy against attacker that try to exploit leakage of permutation variable.



(a) Loop shuffling only. (b) With code polymorphism.

Figure 19: fixed-vs-random ttest to assess the leakage of the permutation variable used for loop shuffling. Values above 4.5 or below -4.5 indicate leakage.

Finally, we conducted several attacks on the considered implementation. We start with a CPA with integration, which is a simple attack usually used in presence of desynchronization. Table 10 shows the result of the attack, with an integration window of 24 samples. The attack easily finds all key bytes for the unprotected implementation, and for the implementation with loop shuffling only. However, the attack fails for both implementations that feature the code polymorphism countermeasure.

Table 10: Results of CPA with integration. Red cross indicate attack failure within 500k traces.

Implementation	Target key byte															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Unprotected	3k	6k	1k	2k	3k	2k	4k	1k	2k	3k	2k	4k	4k	3k	4k	2k
Loop shuffling	28k	32k	35k	18k	38k	28k	42k	45k	20k	18k	35k	20k	35k	40k	50k	45k
Code polymorphism	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
All countermeasures	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗

We moved on with a deep learning attack, with a set of 100k traces. We used 80k traces for training, and 20k traces for validation. Such attack had shown great effectiveness against code polymorphism in the past, as already mentioned. Figure 20 shows the attack result, presenting the gaussian entropy for each of the key byte. The gaussian entropy is the averaged key rank. A low key rank makes key enumeration possible, and a key rank of 0 indicates that the attack succeed in finding the correct key hypothesis. The results are mixed: the attack succeeds in finding 12 out of 16 key bytes in less than 40 traces for the unprotected implementation, but it does not show convincing patterns for any of the protected implementations. This result is surprising, as the attack is not working even for the implementation protected with loop shuffling only, that was vulnerable to a simple CPA with integration. The reason why the deep learning attacks does not work well is still unclear.

As a conclusion of this study, code polymorphism helps loop shuffling to better resist attacks. The security gain from having both countermeasures compared to code polymorphism alone is still unclear though.

These results have been gathered in a paper and submitted to COSADE 2024.

In addition to this work, we worked on the automated application of loop shuffling, and we noticed that the countermeasure could be applied easily using simple C macros that we designed. Such approach makes irrelevant the planed work on automating the application of the countermeasure within the compiler, thus we decided to focus our effort on the study of countermeasures. We started working on Kyber post-quantum crypto scheme, in order to evaluate how to secure it efficiently. This work is still in a preliminary stage. We also started investigating how safe it is to combine code polymorphism with masking.

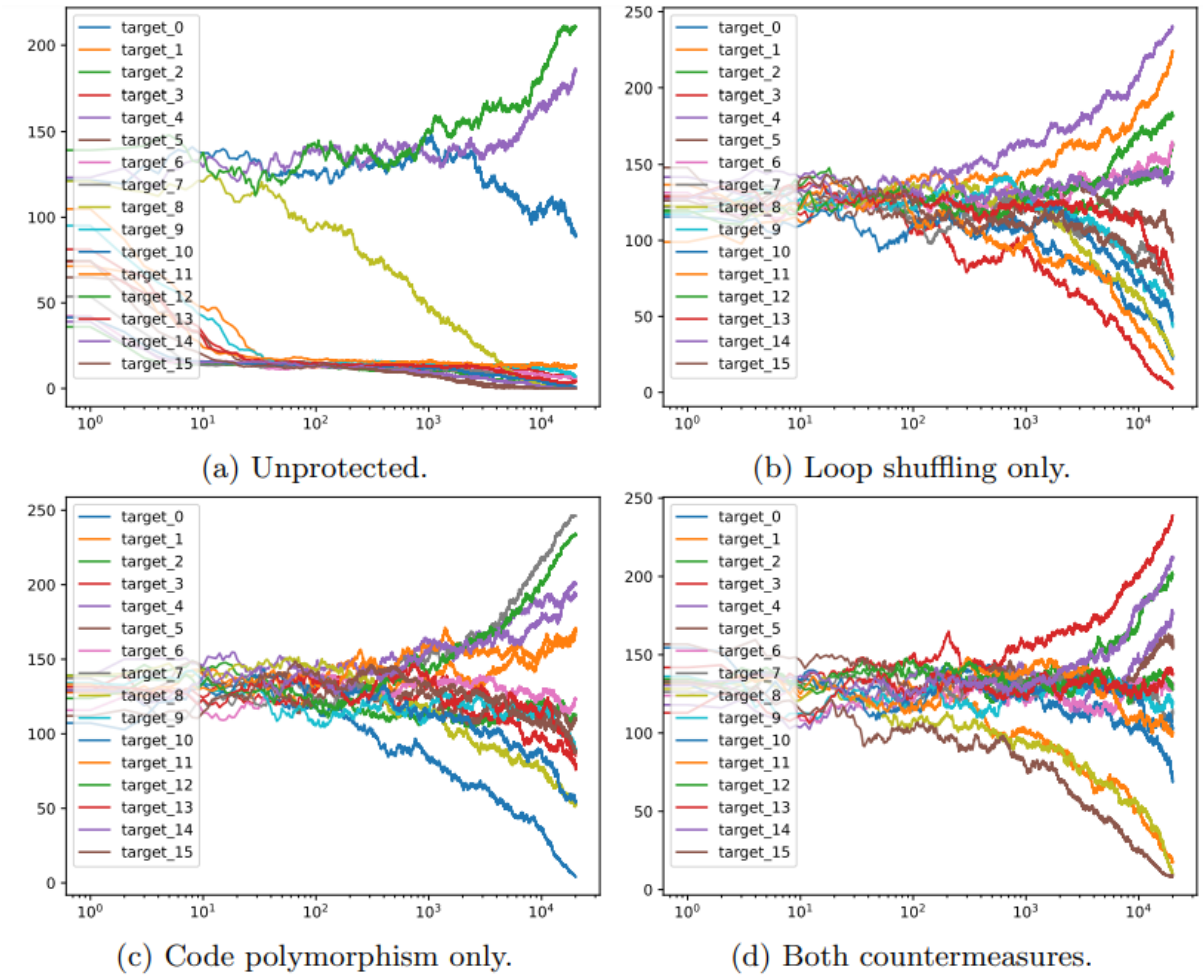


Figure 20: Result of deep learning attack. Figure shows the gaussian entropy, i.e. the average rank of the correct key hypothesis for each of the 16 key bytes. Lower is better.

2.5.2 Demonstration description

The demonstration starts by showing how to implement loop shuffling using our macro. We start with a simple program containing a for loop:

```
void loop()
{
  for(int i = 0; i < 16; i++)
  {
    printf("%d ", i); /* print current loop index */
  }
  printf("\n");
}
```

Figure 21. Simple looping program

The function “loop” is called 25 times. It iterates from 0 to 16 and prints the loop index at each iteration. The output is:

```
qemu-system-arm -machine lm3s6965evb -cpu cortex-m3 -nographic -monitor null -serial null -semihosting -kernel test.bin
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
make[1]: Leaving directory '/odo/tests/demo'
rm loop.bc
odo@20d8a6dbc411:/odo/tests/demo$
```

Figure 22. Output from the program

Then, we modify the code using our macro to enable loop shuffling:

```
void loop()
{
  //for(int i = 0; i < 16; i++)
  FORWALK(i, 0, 16)
  {
    printf("%d ", i); /* print current loop index */
  }
  printf("\n");
}
```

Figure 23. Code modified adding our macro

The use of this macro makes the loop iterate in a random order, as shown by the output of the 25 function calls:

```
qemu-system-arm -machine lm3s6965evb -cpu cortex-m3 -nographic -monitor null -serial null -semihosting -kernel test.bin
13 10 7 4 1 14 11 8 5 2 15 12 9 6 3 0
14 5 12 3 10 1 8 15 6 13 4 11 2 9 0 7
6 13 4 11 2 9 0 7 14 5 12 3 10 1 8 15
3 14 9 4 15 10 5 0 11 6 1 12 7 2 13 8
5 8 11 14 1 4 7 10 13 0 3 6 9 12 15 2
4 9 14 3 8 13 2 7 12 1 6 11 0 5 10 15
13 12 11 10 9 8 7 6 5 4 3 2 1 0 15 14
12 7 2 13 8 3 14 9 4 15 10 5 0 11 6 1
3 6 9 12 15 2 5 8 11 14 1 4 7 10 13 0
12 7 2 13 8 3 14 9 4 15 10 5 0 11 6 1
14 5 12 3 10 1 8 15 6 13 4 11 2 9 0 7
6 7 8 9 10 11 12 13 14 15 0 1 2 3 4 5
12 9 6 3 0 13 10 7 4 1 14 11 8 5 2 15
2 3 4 5 6 7 8 9 10 11 12 13 14 15 0 1
15 8 1 10 3 12 5 14 7 0 9 2 11 4 13 6
1 0 15 14 13 12 11 10 9 8 7 6 5 4 3 2
11 2 9 0 7 14 5 12 3 10 1 8 15 6 13 4
5 0 11 6 1 12 7 2 13 8 3 14 9 4 15 10
11 12 13 14 15 0 1 2 3 4 5 6 7 8 9 10
14 5 12 3 10 1 8 15 6 13 4 11 2 9 0 7
9 2 11 4 13 6 15 8 1 10 3 12 5 14 7 0
13 12 11 10 9 8 7 6 5 4 3 2 1 0 15 14
2 3 4 5 6 7 8 9 10 11 12 13 14 15 0 1
6 15 8 1 10 3 12 5 14 7 0 9 2 11 4 13
5 10 15 4 9 14 3 8 13 2 7 12 1 6 11 0
make[1]: Leaving directory '/odo/tests/demo'
rm loop.bc
odo@20d8a6dbc411:/odo/tests/demo$
```

Figure 24. Program output after modification

The demo moves on by enabling code polymorphism for the function “loop” as well:

```
odo@20d8a6dbc411:/odo/tests/demo$ LFLAGS="-polymorphic-function loop " make
```

Code polymorphism is then automatically applied. To see the effect of code polymorphism, we have to run gdb, to be able to disassemble the code generated in memory at runtime. We start by disassembling

the code of the "loop" function, and notice the call to "compilette_loop", which is the SGPC, i.e. the generator of polymorphic code, and the call "*blx r3*", that will call the polymorphic instance generated in memory.

Figure 25. GDB Breakpoint

Then, we put a breakpoint to the *blx* instruction, in order to stop execution and to disassemble the polymorphic instance.

We disassemble 2 successive polymorphic instances to show that the generator of polymorphic code generates every time a different code. The 2 polymorphic instances are shown side-by-side here:

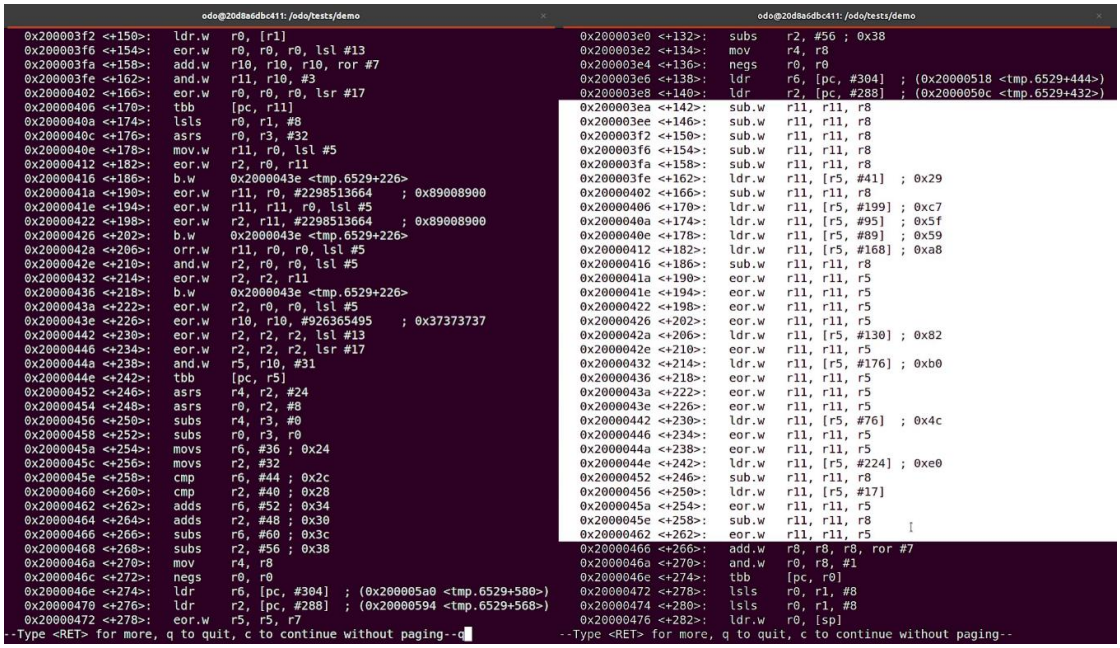


Figure 26. Two polymorphic instances

Finally, we explain the new code transformation we added to code polymorphism, namely dynamic variant, that makes the generator generates random switch cases, implemented with table-based branch (*tbb*) instruction:

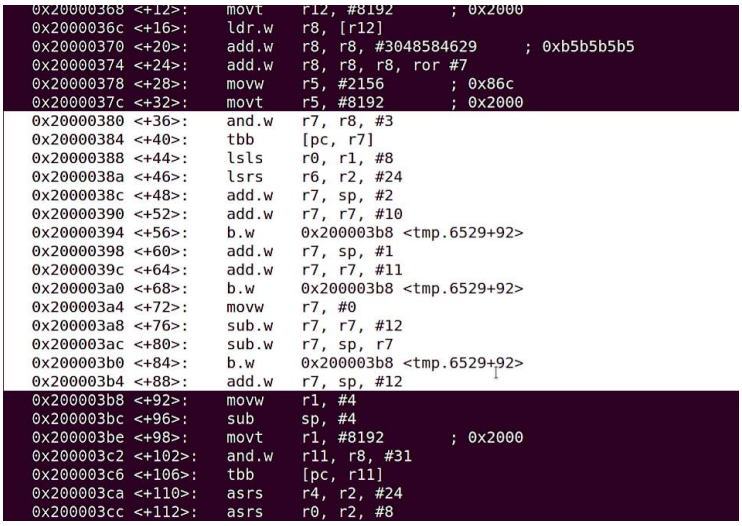


Figure 27. Table-based branch (tbb) instruction

This sequence contains 4 variants, that all compute an addition between *sp* and 12. R7 contains a random number between 0 and 3, and the *tbb* will thus randomly jump either at 0x2000038c, 0x20000398, 0x200003a4, or 0x200003b4. In all cases, instructions that result in doing r7=sp+12 are computed, and the execution continues at 0x200003b8. The dynamic variants transformation allows to have variability when we execute several times the same code, for instance during several loop iterations.

2.5.3 Support for pilots

We will choose target function and countermeasures depending on the pilots need. For instance, if the pilot requires the need of a particular block cipher, we will harden this block cipher. Depending on the platform constraints and the attacker model, we will adapt the chosen countermeasures. As of today, we plan to harden the AES implementation used for the MQTT connexion between the sensor gateway and the nadia platform of the smart hospitality use-case.

3 Conclusions

All five tasks have come up with the necessary components to form a trust management framework supporting the forthcoming pilots in the TANGO project. However, most of these components have not yet been integrated together and in some cases this integration work is still to be done within the task itself. Some of the components lack the required functionalities that the pilots need. These have been identified and the next steps in the implementation these functionalities and integration work are clear.

Implementation work within this WP4 is ongoing. Architectural design work (D2.x deliverables) supports this work and these initial results show progress towards project's goals. Further work will consist of development work concerning WP4 components that will be integrated in TANGO platform. Furthermore, new features required in pilots will be implemented. This implementation and integration should be largely ready by mid-2024 just before piloting phase starts. At that point WP4 continues supporting pilot work and is prepared to make additional implementation and integration work if needed by the pilots.

This report and related demonstrations will see another iteration round in the end of this project with final version of TANGO distributed trust framework used in the project pilots.

Annex A

ZKP generation (section 2.1.2.2)

As a first process in the signature suite, the verifiable credential is signed, for a which a private key and the non-signed credential are needed. The respective verification is done using the public key and the signed VC itself. The signed VC has the following form:


```
{
  "@context": [
    "https://www.w3.org/2018/credentials/v1",
    "https://w3id.org/citizenship/v1",
    "https://ssiproject.inf.um.es/security/psms/v1"
  ],
  "credentialSubject": {
    "birthCountry": "Bahamas",
    "birthDate": "1958-07-17",
    "commuterClassification": "C1",
    "familyName": "SMITH",
    "gender": "Male",
    "givenName": "JOHN",
    "id": "did:example:b34ca6cd37bbf23",
    "image": "",
    "lprCategory": "C09",
    "lprNumber": "999-999-999",
    "residentSince": "2015-01-01",
    "type": [
      "PermanentResident",
      "Person"
    ]
  },
  "description": "Government of Example Permanent Resident Card.",
  "expirationDate": "2029-12-03T12:19:52Z",
  "id": "https://issuer.oidp.uscis.gov/credentials/83627465",
  "issuanceDate": "2019-12-03T12:19:52Z",
  "issuer": "did:key:....",
  "name": "Permanent Resident Card",
  "type": [
    "VerifiableCredential",
    "PermanentResidentCard"
  ],
  "proof": {
    "type": "PsmsBlsSignature2022",
    "proofValue": "zQuxFYVEgHuWfZ7yLoGJXLmFhWGUrYxLxg51EhUP7WqpvWbjgyPfUkS7oGdtLX2EF2rzxpLqiAw1PQyMjhoNa2wsLv
      AdwKWk8thfpEXfh3RzqABcGDZYZM24qy1vzdiHpJyFcdnFnKjYwF6xp1poT41RDFGW33ugUMRQqQA7cCE4AmqYMTZb1JtZGaFF7ZtvN
      Cfx3xXXu33W94gCZrJ3a68cjVgqK1yP3DdXy1jmUfLrHKUdw42Esk9pgg67spsFE9dKgzNK8pw38wtFh79kUoaTFbmKYnZmWk6h7Muw
      fhVDPVhLgfaMJ9GoQgxigtVNU4YKA45Wxp1XTQfZPoCmwrkurqcVkCi5i7JRu..."
  }
}
```


Figure 28. The Signed VC

For the generation of a zero-knowledge proof presentation, the following elements are needed:

1. Public key
2. signed Verifiable Credential
3. Nonce value (to be used in the later verification)
4. Frame: includes the desired attributes to be included in the presentation


```
{
  "@context": [
    "https://www.w3.org/2018/credentials/v1",
    "https://w3id.org/citizenship/v1",
    "https://ssiproject.inf.um.es/security/psms/v1"
  ],
  "@type": [
    "VerifiableCredential",
    "PermanentResidentCard"
  ],
  "credentialSubject": {
    "@type": "PermanentResident",
    "@explicit": true,
    "birthCountry": {},
    "gender": {},
    "givenName": {},
    "image": {},
    "lprNumber": {}
  }
}
```


Figure 29. Zero knowledge present

The expected output has the following form (signed presentation containing ZKP):

```
{
  "id": "https://issuer.oidp.uscis.gov/credentials/83627465",
  "type": [
    "VerifiableCredential",
    "PermanentResidentCard"
  ],
  "description": "Government of Example Permanent Resident Card.",
  "name": "Permanent Resident Card",
  "sec:proof": {},
  "credentialSubject": {
    "id": "did:example:b34ca6cd37bbf23",
    "type": [
      "PermanentResident",
      "Person"
    ],
    "gender": "Male",
    "givenName": "JOHN",
    "image": "",
    "birthCountry": "Bahamas",
    "lprNumber": "999-999-999"
  },
  "expirationDate": "2029-12-03T12:19:52Z",
  "issuanceDate": "2019-12-03T12:19:52Z",
  "issuer": "did:key:....",
  "@context": [
    "https://www.w3.org/2018/credentials/v1",
    "https://w3id.org/citizenship/v1",
    "https://ssiproject.inf.um.es/security/psms/v1"
  ],
  "proof": {
    "type": "PsmsBlsSignatureProof2022",
    "proofValue": "zANMg2ax8reKrRYkrbns83mR8h2PVz5ojcnsjrJmS1ogUMhKanq4XUwt9jV6iQmU1gazLLvJNiGSns4TF3R6M9761hgf
      wNnvtYQe8m84mcrkqcWL6QuJcju1aC7tdmqckKzbnd8MRQFkUBQBUQFWGrH7iVwZ7tpEFTaHsW31oyFBah69uAso4QrCYLEHXZXzrK9eD
      9BLai2jxN8anryBkhLMMEzfThNzx1HJdCvEXdq1VjZGDbUg4mEoL2SMtUtPyexrsa2KV9CjbxFx564DNSgTvFZz92kzwK62r7EtUaDr52
      AVHn73gYaVQoLqHcBH9u1vqB1LYd2yT3xrN47fb5nLHWuN5
      ....E4UYjdZzp3JTVJsek1kE1jijbSTVkVPkYKuoZhnMQwk3XYbeuMHrkskBu8sdfhbVdZ8jEb5a9t7c8ZjjsY4iwWsTo7rVCm1v5ztDd
      nmgEZJz22ALirrKRJQ7qSSNTYbs9V9jBjmMfomNZfokaFBwLANdgpgDNNSPaqUM8XLJ1hSFcK19ZeswekYwqLqeUSaervyTqiVNFqtAg5
      vgYQaTrcqEdvJaMANW"
  }
}
```


Figure 30. Signed presentation containing ZKP

Once the ZKP presentation is generated, the verification process needs:

1. Nonce value: coming from VP generation, it makes sure in the verification process that the presentation belongs (as it is the same as used in the generation).
2. Public key
3. ZKP VP

Annex B

Device Continuous Behavioral Authentication Endpoints

REGISTER ENDPOINTS

1. /users/register/request:
   a. Client uploads the credentials (username, password).
   b. Server searches for the username in database.
   c. If the username does not exist, the username and the encrypted password are temporarily stored. Server generates a random challenge and sends it back to client in order to sign it, along with some Server Information, such as the relying party etc, in order to receive it back from the client's response.
2. /users/register/request/upload/signature:
   a. Client uploads the username, the Public-Key, the original challenge that server had been sent in order to get signed, the Register Signature which is the Signed Register Challenge by the client, the client's device name, and the Server Information that received from the server when user made the HTTP Register Request.
   b. Server verifies the signature by signing it with the corresponding Public-Key and comparing with the client's Register Signature.
   c. If the client's signature is not valid, register request is rejected by the server. In the opposite scenario, server retrieves the client's username and password which were temporarily stored when the client made the request and adds the user. At this stage, server creates the user's folder, where the Android Device's System Logs will be stored.

LOGIN ENDPOINTS

1. /users/login/request:
   a. Client enters username & password in order to login.
   b. Server searches in the database for the username, and if exists, then it compares the password that user uploaded with the password that is stored in the database, associated with this username.
   c. If these two (2) passwords match, then, as server did in the Registration, generates a random challenge in order to send it back for sign, along with some Server Information as before.
2. /users/login/request/upload/signature:
   a. Once client receives the Login Challenge and signs it with the corresponding private key, produces the Login Signature which is sent back to the server, along with the device's name, the server information and the username.
   b. Server verifies the client's signature, by signing the original Login Challenge with the corresponding public-key and compare these two signatures together.
   c. If they match, user is successfully logged in and considered authenticated, while server temporarily stores the username in a text file named 'currentUser.txt', as long as the user remains logged in.

ANDROID SYSTEM LOGS CAPTURE ENDPOINTS

The capture of the Android Logs begins when a user is logged in. The endpoint that is responsible for this:

1. /users/android/logs/capture/status:
   a. User is logged in and makes an HTTP request to this endpoint, uploading the username.
   b. Server checks if this username already have PIDs active and running on the system. This check is necessary, since it is possible that user disabled the Wi-Fi while capture was in operation, resulting the device to disconnect from the server. So, if active PIDs exist and also associated with this username, server destroys them, and generates new ones. If not, server executes the ADB Commands (generating new PIDs that capture the

Logs of the device) without destroying anything. The file where the PIDs are temporary stored is named 'Capture_Log_Processes.txt', and contains data like the image below:

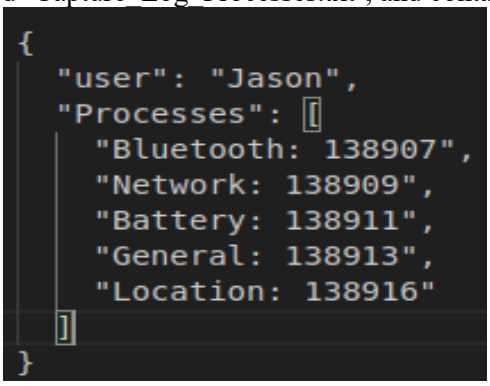


Figure 31. Active PIDs associated with the logged-in user.

LOGOUT ENDPOINTS

Since there are still some permission issues to enable the application to send by itself the Android Log data to the server remotely, server collects and captures the Android System Logs of the device on his own, by executing the proper ADB commands for the different interfaces. So, when a user logs out, server tracks these PIDs, destroys them and deletes the user's username from the file 'currentUser.txt'. Two endpoints responsible for operating this functionality:

1. **/users/stop/android/logs/capture**:
    a. When the user initiates the logout process by clicking the logout button, an HTTP request is made to this endpoint, containing the username and a Boolean value. The Boolean's value use-case is to examine the possibility that during the capture of the Logs, user disabled the 'USB Debugging' functionality we mentioned earlier. In this scenario, the PIDs are automatically terminated, but they still exist in the 'Capture_Log_Processes.txt' file as data. So, server searches in this file for these PIDs that were active and associated with this user and deletes them. In the other scenario, server also terminates the PIDs in the System, by killing them.
2. **/users/logout**:
    a. When the Logout button is pressed and the PIDs terminated successfully, server searches in the file 'currentUser.txt' and deletes the user.
    b. User logs out.

OTHER ENDPOINTS

Three (3) endpoints, responsible for the Setup's configuration and all the necessary operations in order to keep the user's device continuously connected to the Server:

1. **/connect/device/setup**:
    a. An HTTP request is initiated directly from the application at this endpoint, uploading the IP Address of the device along with the device name.
    b. Server waits until the device appears online, and executes these two commands:
        i. '**adb -s {DeviceName} tcpip 5555**', switching the Android Debug Bridge (ADB) communication to TCP on port 5555, allowing to collect all the Logs remotely.
        ii. '**adb connect {DeviceIP}:5555**', connects the device given its IP, to the server.
        iii. Then, inside the file 'devices.txt', server stores in JSON format the device's name along with its IP address.
2. **device/past/activity**:
    this endpoint communicates with the main Class of the 'Authenticator' app, determining if the device that initiate the HTTP request to this endpoint, was connected in the past or not. If yes, then it is not necessary to rerun the setup, instead, reconnect the device.
3. **/reconnect/device**:

In some cases, the device needs to be reconnected, such as when user disables the 'USB Debugging' functionality. So, in these cases:

a. User makes an HTTP request, containing the IP Address of the device and its name.
b. Server searches in the file 'devices.txt' which contains information about all devices that are or were connected to the server, locates this specific device, update its IP Address -if changed- and executes the connection ADB command.

LIBRARIES AND FRAMEWORKS USED

SERVER's SIDE:

- **Express.js**: Web application framework for Node.js.
- **Bcrypt**: A library for hashing passwords.
- **Crypto**: A Node.js module, used by the server to generate random challenges (Login – Register Challenge), intended for signing by the user, using the private key.

APPLICATION's SIDE (Developed in Android Studio):

- **androidx.appcompat:appcompat:1.6.1**: Provides implementations of some Android framework components, in order to perform some tasks easier.
- **com.google.android.material:material:1.9.0**: Material Components for an Android System, such as design support.
- **androidx.core:core:1.7.0**: Offers several functionalities for working more efficiently with an Android system.
- **com.google.android.gms:play-services-location:18.0.0**: A library provided by Google Play, in order to access services related to device's location.
- **junit:junit:4.13.2**: A testing framework for Java.
- **androidx.test.espresso:espresso-core:3.5.**1: Another testing framework, designed for writing UI tests in Android.Enter your text here.