A Decentralized, Open Source Solution for Digital Identity and Access Management

JOLOCOM · Whitepaper v2.1 · *December 2019*

Jolocom develops an open source protocol for decentralized identity management designed along the principles of Self-Sovereign Identity.

The Jolocom Protocol facilitates the generation and management of Decentralized Identifiers, Verifiable Credentials, and cryptographic signatures — the core building blocks of Jolocom identities. Jolocom identities are created entirely locally using hierarchical deterministic keys and are designed to enable management of multiple personas by individual users as well as preservation of pairwise anonymity in context-specific interactions.

The protocol logic encodes a granular, claim-based model of identity that is highly generalized and unrestrictive in scope in order to accommodate a multiplicity of potential use cases and broad range of subjects of identity (users), including individual persons as well non-person entities like organizations (e.g. companies, governmental bodies), IoT devices (e.g. hardware/software agents), and autonomous agents (e.g. DAOs).

01 Introduction

Early visions of the Web anticipated development of a cooperative space, a common virtual resource environment that would serve communities linked together via a non-proprietary network of distributed nodes. The idea was to create a universally accessible system of communication based on links between agents and resources.

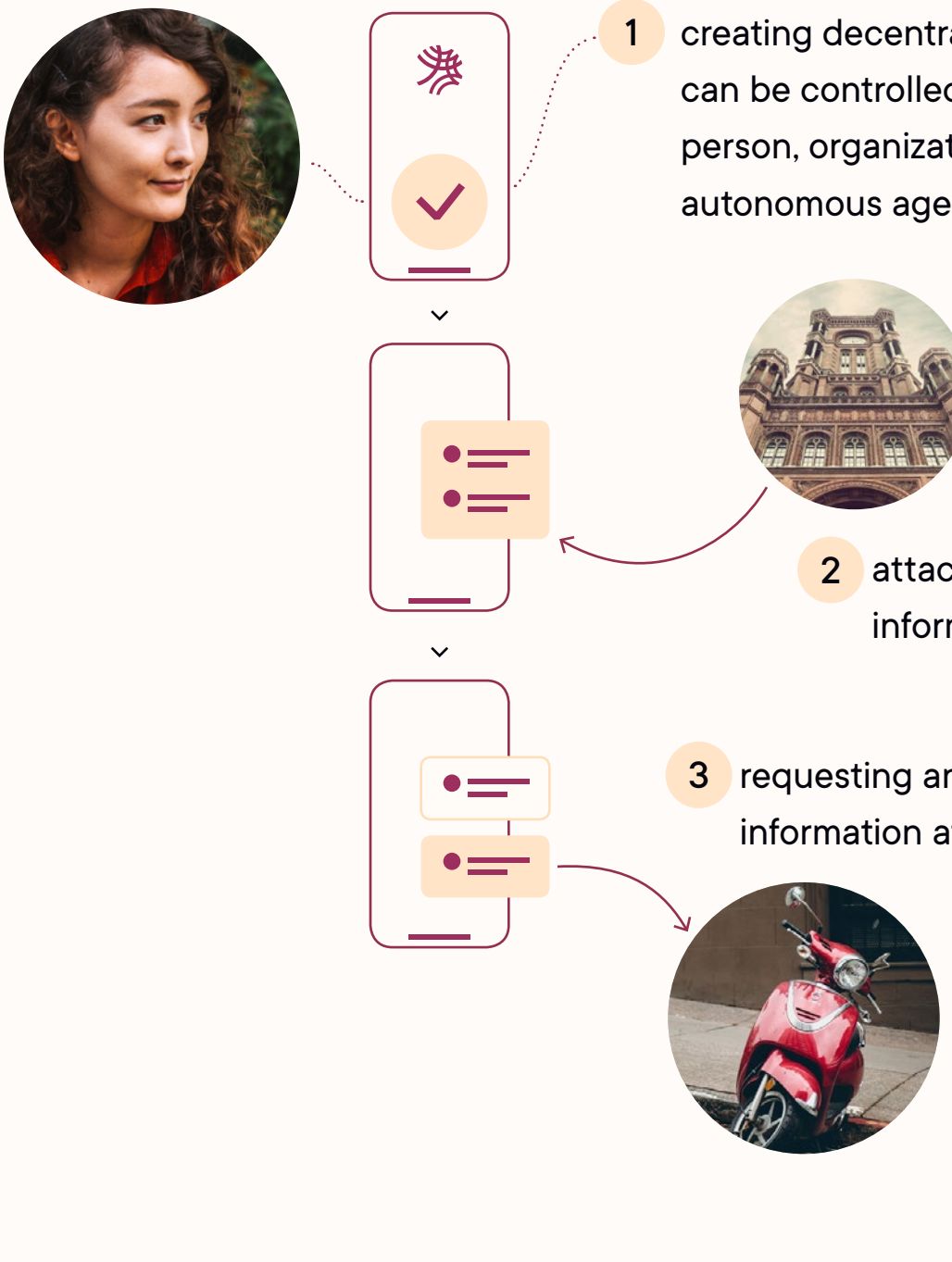
In the late 1980s Tim Berners Lee and his team at CERN initiated the development of HTTP [1] for inter-network data communication, work that still serves as a bedrock for today's Web. Around the same time a practical need to communicate electronic messages between networked computers running different systems led to SMTP [2], a crucial element of how email works today. General open protocols like these are incredibly powerful as they serve as the foundation of everything subsequently built on top.

In the case of the Web, no open protocol for users to generate, manage, and resolve identities on their own (individual) terms has gained sufficient adoption. In other words, no common standards exist for self-sovereign representation of Identity across the Web.

With this background in mind, Jolocom is actively developing an open source protocol, the "Jolocom Protocol", for the secure communication of identity

(information) that prioritizes user privacy. Our decentralized solution to identity management equips users with a decentralized identity based on highly secure cryptographic keys. Our software for sophisticated key management and reliable claim verification allows for independent user control over self-issued identities. The Jolocom Protocol thereby enables individual users (e.g. persons, organizations, IoT devices, autonomous agents) to securely generate, provision, and control the keys to their identities privately, i.e. without ever having to rely on a third party or intermediary service.

The protocol manifests the following general functionalities related to digital identity management:



1 creating decentralized identities that can be controlled by an individual person, organization, IoT device, or autonomous agent;

2 attaching verifiable information to identities;

3 requesting and consuming verifiable information attached to identities.

The architecture of the protocol revolves around three main concepts:

1 — **cryptographic keys**, which enable context-specific interactions and provide identifiers (as proxy for user identities) with signing and transaction capabilities;

2 — **“Decentralized Identifiers”** (DIDs) [3], which enable globally unique identifiers that are self-issued and can be configured to automatically resolve to DID Documents [4] containing more information about the identifier in question;

3 — **“Verifiable Credentials”** [5], which provide a way to relate an identifier to statements which are cryptographically verifiable.

Cryptographic keys and DIDs enable the existence of a decentralized identity. Keys and Verifiable Credentials provide the tools required to create complex data while simultaneously preserving simplicity at the core. This approach allows the protocol to remain generic while facilitating an unlimited number of specific use cases with varying levels of complexity.

The idea of using a public key infrastructure (PKI) to manage identities is not a new one (see WOT [6], DPKI [7], DKMS [8]). However, key management and recovery in a decentralized approach to identity has largely been inefficient and resource-intensive, with many approaches simply offloading complexity related to such issues onto the end user, which hinders mass adoption of privacy-friendly tech.

The Jolocom Library [9] aims to simplify, improve and enhance the management of identities and the data associated with them, unifying the underlying technologies into a single, developer-friendly API. Due to the rapidly changing decentralized technology landscape, the architecture is designed to not be bound to any particular technology, but instead to be able to adopt technologies as they arise that are most suitable to particular parts of the functionality the Library exposes.

Currently, it exposes the following core identity management functionalities:

1 — Generation of a unique, decentralized, and permanent global identity;

2 — Derivation of child identities from this master identity to accurately model different user personas and/or provision IoT devices;

3 — Creation of Verifiable Credentials associated with the identity which may be used in further interactions with services or other parties;

4 — Association of Verifiable Credentials from third parties with chosen user identity;

5 — Definition of a set of standard interaction tokens which can be used to model any identity or credential-related interactions.

Extending these capabilities to users is the first step to building an enabling environment for Self-Sovereign Identity.

02

The concept of SSI

‘Self-Sovereign Identity’ (“SSI”) is a relatively new term that signifies a particular way of thinking about identity. Despite more recent efforts emerging from a variety of working groups, organizations, and institutions concerned with issues of digital identity and online privacy, no formal consensus definition of SSI has emerged.

Any coherent account of SSI will necessarily rely on certain (fundamental) assumptions (which, in turn, rely on certain other assumptions). In the case of SSI, a consensus definition would logically require widespread agreement on a specific set of assumptions related to concepts like identity, governance, sovereignty, power, control, authority, agency, decentralization, and so on.

Placed under scrutiny, seemingly straightforward concepts like these tend to give rise to divergent or inconsistent definitions among interpretations. In other words, Self-Sovereign Identity is a type of concept that requires extensive definition given the competing interpretations available, and to reach consensus on a coherent definition of SSI in practice requires precise reasoning, extensive discussion, and examination of underlying assumptions. At this nascent stage, a common definition remains elusive.

In light of this, we offer a concise definition of SSI that reflects our collective understanding at Jolocom in §2.1, and expand on this definition with reference to a common set of principles for SSI as each relate to our overall design approach and implementation.

02.1

A concise definition of SSI

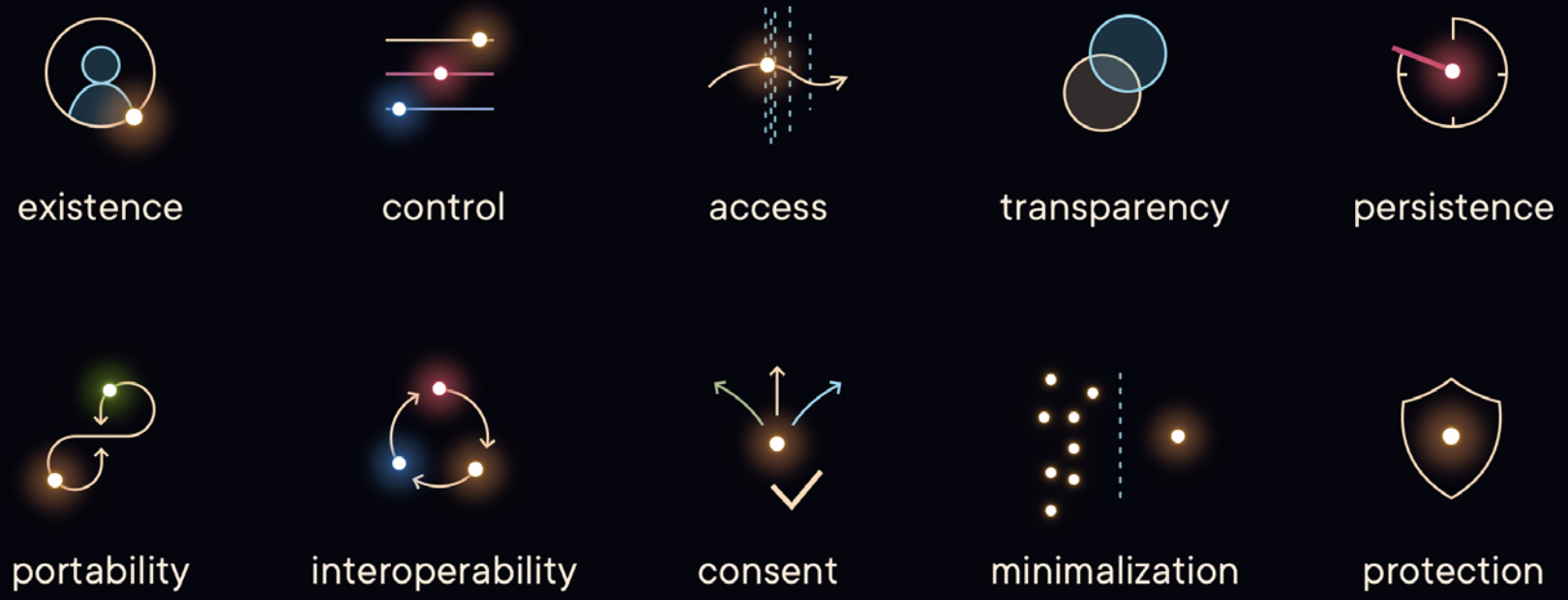
Self-Sovereign Identity refers to a particular model of identity in which subjects of identity are able to express their identities autonomously and to control their identities on their own terms when interacting & communicating with other subjects irrespective of context.

SSI - ***Self-Sovereign Identity*** / *n* - a model of digital identity where individuals and entities alike are in full control over: **1.** central aspects of their digital identity, including their underlying encryption keys; **2.** creation, registration, and use of their decentralized identifiers or DIDs; and **3.** control over how their Credentials and related personal data is shared/used.

02.2

Designing for Self-Sovereign Identity

In order to design an identity management system in a way that supports Self-Sovereign Identity among users, certain conditions must be upheld by the system architecture across implementations. In a popular 2016 blog post discussing digital identity, trust, and privacy, Christopher Allen outlines his “Ten Principles of Self-Sovereign Identity”, a series of guiding principles “that attempt to ensure the user control that’s at the heart of self-sovereign identity” [10].



In §2.2.1—§2.2.4 we expand on our definition of Self-Sovereign Identity presented in §2.1 and proceed to examine our approach to digital identity management according to clusters of the principles outlined in Allen’s blog post (which often serves as a common reference point for analyzing and dissecting SSI solutions). Each section begins with a few remarks from our own perspective and goes on to detail how Jolocom's system design and implementation uphold and embody the principles of SSI.

Existence. Persistence.

Our vision of Self-Sovereign Identity recognizes the entity referenced by any given identity as the ultimate and independent authority responsible for that identity. Identities should be shaped, conditioned, and regulated by the person or non-person entity (e.g. public body, company/corporation, IoT device, autonomous agent) which they describe. Indeed, the very existence of an identity (i.e. whether an identity persists or ceases to exist) should be determined at the exclusive discretion of the entity to which the identity refers.

Our solution models subjects of identities as “Holders” in possession of trustable, claim-based digital identities. Holders are provided persistent virtual ownership over the “root” of their identity in the form of local private key generation and storage. Jolocom users maintain exclusive control of their private keys by default.

The notion of persistence is further reflected in our agnostic approach to identity management. The system architecture is designed so as to accommodate a variety of different technologies and stacks. In terms of storing public claims, for example, a Holder is not beholden to any particular backend, service, or technology. This technological flexibility mitigates the dependency of system identities on particular technological integrations, thus enabling greater longevity of identities.

— *The protocol does not currently discriminate between “subjects” of identity & “holders” of identity, but the protocol logic is nonetheless designed to support fiduciary relationships.*

Control. Consent. Minimalization.

The Holder of an identity should be capable of manifesting intent related to the identity, especially consent related to disclosure. That is, users should generally have the capacity and means necessary to influence and have an effect on identities they hold. By purposefully equipping Holders with exclusive access to their private keys by default, the Jolocom Protocol supports novels forms of user agency by means of powerful methods for manifesting intent related to digital identities.

Jolocom does not collect any user data: all user data is stored privately (locally) by default. The user can of course choose to make certain identity information publicly accessible.[1] In other words, by default, only the Holder of a given identity is capable of instrumentalizing that identity to manifest intent. Furthermore, the Jolocom Protocol will support selective disclosure of claims such that Holders are capable of granular control over their claim-based digital identities.

[1] *For example, a user may publish a public profile to IPFS, and the DID-to-DID-Document mapping on the Ethereum Rinkeby Testnet. As detailed in §3.6, this reflects our current reference implementation of the Jolocom Protocol.*

Access. Portability. Interoperability.

The Holder of an identity should always have the identity available at the Holder's disposal, i.e. ready for the Holder to use at any moment in any situation. An identity should therefore be accessible to its Holder across any given context or interaction. The Holder of an identity should be able to deploy the identity at the Holder's discretion, which requires the Holder have sufficient access to the identity necessary for whatever purposes of deployment.

All identity attributes created using the Jolocom Protocol , including private keys, are stored client-side on a user's device by default. These data can only be controlled by the device owner. A Holder can attach a public profile to a DID Document containing human-readable information about the Holder's identity to make certain aspects selectively accessible.

Additionally, an identity should be able to accompany its Holder throughout any given context or interaction. Holders should therefore be generally capable of transporting their identities fully intact across different situations and environments. Identities created using the Jolocom Protocol are interoperable in that they can be used by other identity systems implementing the same (open) standards. Furthermore, the protocol will enable portability of identities so that Holders can access their identities from different devices and applications.

Transparency. Protection.

Transparency is integral to rooting trust in a system by its users. Being open about what decisions are made and making explicit about how decision-making works ensures that all governance of or on a platform, network, or system are inspectable and understandable by all stakeholders. For Holders, protection is essential to maintaining individual sovereignty over their digital identities. Holders must be able to (inter)act in a climate of confidence, certain that their digital identities are insulated from circumstantial external pressures (e.g. network failure, system abuse, ambiguous policy changes). The principle of protection revolves around the notion of reslience — an identity management system must shield (the rights of) individual users against arbitrary or inscrutable changes in (whatever) environment their data lives and operates.

All the code behind the Jolocom protocol is open source, published and maintained in a public, easily accessible online repository. As such, our solution codebase is fully auditable. Furthermore, in light of its technology-agnostic design, the protocol affords Holders an unprecedented ground for assurance that the integrity of their digital identities and related data will endure beyond a given (network) environment.

03

The Jolocom Protocol

Our decentralized, open source protocol for digital identity and access management allows people, organizations, IoT devices, and autonomous agents (e.g. DAOs) to operate digital identities that they create independent of any issuing authority or service provider. Any identity holder can produce, share, and consume trustable identity data by implementing and (inter)acting according to the protocol. Beyond data management capabilities, the protocol enables users to own the digital identity information they create. That means Holders are equipped with an unprecedented degree of control over their digital resources and online interactions involving those resources.

Our solution models subjects of identities as "Holders" in possession of trustable, claim-based digital identities. The protocol does not currently discriminate between "subjects" of identity & "holders" of identity, but the protocol logic is nonetheless designed to support fiduciary relationships.

03.1

Our philosophy of decentralized design

As a general design rule, we endeavor to build decentralization into the protocol in a rigorous effort to implement the principles of SSI (outlined in §2.2.1–§2.2.4). There are many ways for a system to be decentralized. Indeed, different components within the same system may exhibit varying degrees of decentralization, or centralization for that matter. We find it useful to characterize the terms “decentralized” and “centralized” as extremes at either end of a continuous spectrum, rather than absolute categories or mutually exclusive states of a system.

Decentralization or centralization can occur over multiple layers of a system’s architecture or operations. For instance, a system might have all of its core logic delegated to smart contracts running on a public permissionless blockchain, its underlying data storage provided by a distributed peer-to-peer network, but the registration and allocation of unique identifiers for new users regulated by one central authority. In this example, the system architecture features both decentralized and centralized elements. This is not necessarily a bad design and might prove sufficient depending on the requirements and use cases.

Furthermore, a given system may appear centralized from one perspective and decentralized from another frame of reference: for example, a system can be decentralized from a technical point of view and centralized from a legal point of view. Depending on the specifics of a given legal framework (especially as concern licensing), a system can be fully “dependent” on an external legal entity such that failure of the responsible legal entity to support and maintain the system would indeed compromise the supervening system

to the point of collapse. In this scenario the system's technical degree of decentralization is a moot topic given the system's overarching dependence on a (central) external entity, often a corporation or foundation.

A final dimension worth keeping in mind in terms of developing a coherent framework for thinking about decentralization is the basic fact of reality that different people, communities, organizations, and institutions consider different levels of decentralization acceptable. A company might be building a proprietary (i.e. closed source) identity platform that relies on a permissioned blockchain (that only the company itself or its partners are allowed to append data to), and use proprietary protocols, but still claim on the basis of certain interpretive frameworks that the company is building a "decentralized" solution. There is no exhaustive technical or legal definition of 'decentralization' and indeed an exact definition remains a point of debate.

At Jolocom, we envision a dynamic ecosystem for self-sovereign identity featuring a diverse array of modular, interoperable identity stacks built according to open standards and designed to enable Holders to securely own and communicate the data that defines them. This vision largely encapsulates our understanding of "decentralization" and its applicable merits within the scope of modern digital identity management.

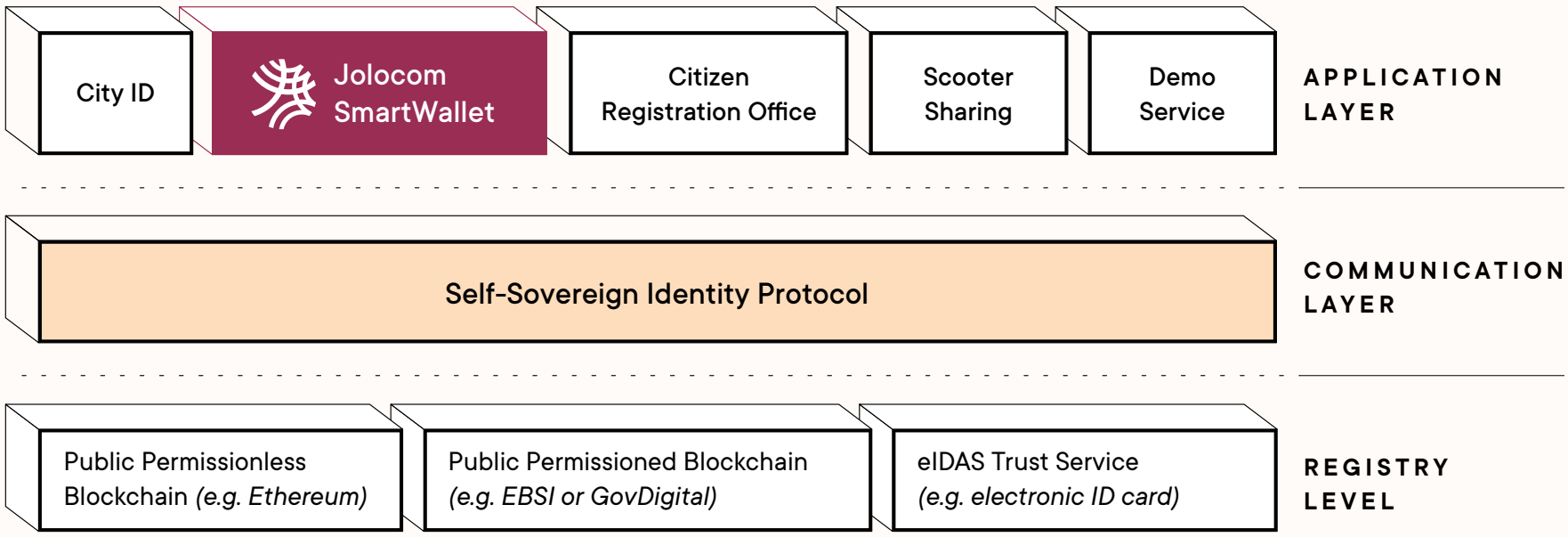
03.2

Core assumptions

The main goal is to develop a system in which various entities can reliably model their real world identities in a way that provides end-users with sovereign authority over their data. In general a Holder should be able to “act” on behalf of the identity (e.g. authentication, exchanging data). Furthermore, Holders should have full control over the existence of digital identities in their possession (e.g. Holders should be able to decide if and when their digital identities becomes “resolvable” or “knowable” to the rest of the system, including when a given identity should no longer “exist” within the system).

In addition to these primary capabilities, enabling interaction among Holders further requires support for non-repudiable data exchange. That is, Holders must be able to send and receive messages to each other in a trustable manner. This basis of trust serves as a foundation for enriching the protocol with more complex interactions flows.

§3.2.1 and §3.2.2 describe in detail how the Jolocom Protocol is designed to handle these conditions and support identity management without an overarching (central) authority. We illustrate in §3.6 how the protocol can be deployed to satisfy these conditions in practice with reference to our current implementation.



03.2.1

The building blocks of trustable identities

Centralized identity management systems characteristically feature a central system authority to underpin (i.e. support and oversee) all user actions and interactions. Central authorities maintain a record of provenance for data within a given system, including user data, which effectively serves as a basis of trust for users when interacting with the system. The absence of a centralized, “omniscient” authority to facilitate interactions between system identities introduces a number of technical challenges.

First and foremost, absent a central oversight authority, there must be a way to ensure that any given identifier is unique and will remain unique for the lifetime of the identifier. This includes identifiers that Holders may generate for themselves. Without this assurance, users have no basis to trust that a given identifier encountered in different contexts or interactions refers to the same (identical) subject across each encounter. On a technical level, in order for identifier generation to produce unique outputs (identifiers), the generation process must be collision resistant, i.e. there must be a way to

ensure identifier generation always produces unique values. In practical terms, users require a way to acquire identifiers for themselves that they can trust are unique.

A further challenge introduced by the need for collision resistance in identifier generation concerns proving ownership over identifiers. There must be a way for users to prove a given identifier is held by a specific Holder across interactions with that identifier (whether for the purposes of proving a user's own identity or the identity of another Holder). In other words, users must be able to be confident that any identifier they interact with represents the same Holder in each (subsequent) interaction. After interacting with an identifier for the first time, a user must be able to discern throughout (all) subsequent interactions with that identifier that the Holder of the identifier is indeed the same identity holder.

Accordingly, a way for users to generate a unique identifier so that they can "be known" by the rest of the network must be provided. Furthermore, a resolution process that other entities on the network can use to dereference the identifier to certain metadata must also be defined. Lastly, we must define protocols for authentication (i.e. establish methods according to which a Holder can prove control over a particular identifier) and for key management (i.e. specify what cryptographic material is required, methods for key recovery). Given these building blocks, more complex and intricate interaction protocols can be developed.

The importance of developing the aforementioned building blocks with interoperability in mind has been recognized early on in the form of standardization efforts. One of the many outcomes of these ongoing efforts is the set of specifications for Decentralized Identifiers (DIDs) actively developed by the W3C. The aim is to standardize the general structure of unique identifiers and the structure of associable metadata documents

(DID Documents), as well as specify a high-level protocol for dereferencing identifiers.

The aforementioned DID specification is used within the Jolocom Protocol to model a number of different types of real world identities (i.e. identities belonging to humans, organizations, IoT devices, or autonomous agents like DAOs). We have developed a DID method implementation ("Jolocom DID Method" [11]) for anchoring DIDs using the Ethereum Rinkeby Testnet, and persisting DID Documents on IPFS. We have also developed a typescript library — the Jolocom Library [12] — which encapsulates a number of useful data structures for working with DID Documents and Verifiable Credentials, and provides simple APIs for:

- creating, anchoring, and resolving identities;
- creating and validating Verifiable Credentials;
- DID authentication and exchanging Verifiable Credentials.

Though our current reference implementation uses Ethereum and IPFS, the Jolocom Protocol itself is ledger agnostic, so connectors to other networks (e.g. Bitcoin [13], BigchainDB [14], Quorum [15]) can be easily integrated with the Jolocom Library. This approach builds flexibility into the protocol by allowing for iteration on the Jolocom DID Method as the ecosystem matures. We are currently exploring emerging DID methods which afford better scalability through transaction aggregation (e.g. Sidetree [16]), and better privacy by removing the usage of the logically centralized registry on the ledger in favor of peer-to-peer message passing (e.g. using Peer DIDs [17]).

The process of creating an identity is essentially reducible to generating a public-private key pair, and then further generating a DID and a corresponding

See §3.6 for an overview of our current reference implementation.

DID Document (which contains metadata such as public keys used for authentication, encryption, and recovery). Given that the algorithm for generating a DID must be collision-resistant, in many cases the DID is either the output of a cryptographic hash function (using the user's public key as input), the public key itself, or the cryptographic hash of a templated DID Document [18]. It's worth noting that the identity is already usable at this stage although the DID Document has not yet been published. For example, Holders can interact with other entities directly by including their valid DID Document with every message, though the DID is not resolvable for any entity outside of that interaction.

The final step of the identity creation process is anchoring the identity on a ledger. Anchoring enables other users on the network to dereference the DID using the resolution process defined in the DID method specification. Having a publicly resolvable DID Document can be beneficial for including public credentials (e.g. certifications), establishing chains of trust, advertising endpoints for further, off-chain interactions, and maintaining a credential revocation list.

These building blocks can be used to recreate conventional trust hierarchies and change the way service providers are discovered. While many of the aforementioned features are required for modeling categorically public entities (e.g. service providers, governmental institutions), they can in most instances be omitted from the user's DID Document. We are currently exploring DID method specifications (like those mentioned above) which side-step the need for a ledger and result in a more scalable and privacy-preserving solution when used for modeling individual user and device identities.

03.2.2 ·····

Enabling the exchange of verifiable data

Building on the assumptions presented in the previous section (namely the presence of unique identifiers and a protocol for resolving them to metadata documents), more complex interactions can be modeled. As will be explored in §3.3, all supported interaction protocols are either related to exchanging Verifiable Credentials or rely on passing signed interaction tokens. Therefore, the protocol should allow for a way for entities to generate and broadcast non-repudiable, cryptographically signed messages.

In our reference implementation, we use the W3C Recommendation [19] to model attestations (also referred to as credentials) issued between identities. When combined with DIDs as specified in the W3C Working Draft on data models and syntaxes for decentralized identifiers, Verifiable Credentials allow for complex interaction flows to be modeled intuitively. In the Jolocom Protocol, all identities listed in a Verifiable Credential (i.e. Holder, Issuer, Signer) are DIDs, which allows for simple signature verification (even across different DID methods), and gives Verifiers access to the public profiles of the Issuer and Signers, including a reference to a revocation registry. Consumers of credentials can define and encode their own criteria for trust (e.g. “the credential must be issued by a specific identity”, “the Issuer must have a specific credential listed in their public profile”) as constraints broadcasted as part of the credential exchange protocol.

In addition to Verifiable Credentials, identities can generate a number of other signed messages for use in various interaction protocols. These messages are frequently ephemeral, discarded after an interaction has concluded. In light of this fact, instead of modeling these messages as Verifiable Credentials, we have opted to utilize the more lightweight JSON Web Signature (JWS)

specification [20]; this allows the protocol to reasonably accommodate use cases involving restricted bandwidth and interaction with embedded devices. Both specifications were chosen with interoperability in mind, i.e. to eventually enable interactions across different DID methods. The next sections discuss these topics in further detail.

Finally, as shown in the next section, the ability to generate auditable Verifiable Credentials and exchange non-repudiable messages serve as the foundational pillars of all interaction protocols supported by the Jolocom Protocol. Given the importance of digital signatures, and cryptographic keys by extension, a suitable mechanism for creating, revoking, and recovering key pairs must be provided as well. §3.4 introduces our current approach to these challenges in greater detail and explains how certain building blocks described in the previous sections can be used to implement protocols for key revocation and recovery.

03.3

Interaction flows

Decentralized Identifiers and Verifiable Credentials are powerful and secure models of identity and attestation. However, without a standardized way to communicate credentials between identities, implementations of these models would be very limited. For this reason, the Jolocom Protocol defines a mechanism for interaction between identities.

Interactions between identities under the Jolocom Protocol are mediated by a concise set of interaction tokens designed to model any credential exchange in an intuitive, modular way. These interaction tokens can be incorporated by services into well-defined interaction flows to enable secure and arbitrarily complex models of authorization, issuance, and verification.

03.3.1

A note on interoperability

The intent of these interaction tokens is to enable the exchange of Verifiable Credentials between decentralized identities. Since these identities will not necessarily exist within the Jolocom Protocol, the format used for interactions between identities should have the following properties:

- widely adopted;
- easy to implement;
- established, open, and extensible standards for cryptographic operations;
- minimal distance to the data formats of its intended payloads (DID Documents and Verifiable Credentials).

For these reasons, the standard JSON document structure [21] is used to format these tokens: standard JSON documents are encoded as JSON Web Tokens (“JWTs”) [22] with JSON Web Signatures for verifiability/non-repudiation and with optional JSON Web Encryption (“JWE”) [23] for privacy. These standards are some of the most widespread and extensively used on the internet, and appear in a W3C Working Draft on the core data model and syntaxes for decentralized identifiers (specifically in the DID Document data model [24]) as well as the W3C Recommendation for a Verifiable Credential DAta Model. While the interaction flows used specifically in the Jolocom Protocol may not yet be widespread, they are as standardized and as simple as is currently possible.

03.3.2 ·····

Authentication

The intent of the authentication interaction token is to simply request and/or provide proof of control over a DID. For this reason, the payload consists solely of a DID and a signature created with the key pair listed on that DID’s public profile. It is important to note that authentication and authorization are distinct activities: authentication deals with proof of identity, whereas authorization does not.

03.3.3 ·····

Credential exchange

Credential exchange is the primary way that identities interact and there are five different message types which are covered by the Jolocom interaction tokens. As all of these are encoded as JWTs with the associated verifiability and non-repudiability described above, each interaction is secure, trustless, verifiable, and non-repudiable. The credential exchange message types are as follows:

— **Credential Request**

A credential request is a simple list of requirements to be satisfied by the credential(s) contained in a corresponding credential response. It allows for first-order logic operations to be defined by the requirements in order to enable complex criteria definitions.

— **Credential Response**

A credential response is a list of credentials corresponding to the criteria listed in a credential request, with the intent of fulfilling the requirements.

— **Credential Offer Request**

A credential offer request is a list of credential types offered by an Issuer with associated metadata. The metadata can include anything from defined application requirements (similar to the Credential Request) to information for rendering credentials (e.g. for graphical representation).

- **Credential Offer Response**

  A credential offer response is a list of selected credential types corresponding to some or all of the credentials listed in the credential offer request. If application requirements are listed in the metadata for the offer request, the response should also include the required information.

- **Credential Receive**

  A credential receive is a simple list of issued credentials corresponding to the valid selection(s) made in a credential offer response. This message type is generally a subset of the Credential Response type, which always specifies the subject of a credential as the recipient.

The most basic flows enabled by these tokens might contain only two interactions (e.g. a simple request and response), while a more complex flow may contain several nested interactions (e.g. authentication followed by a proof of credential exchange followed by a credential offer request, response, and receipt).

It is important to note that the interaction tokens are merely a transportation format providing security and unambiguous intent. The context of a flow as a whole is determined by its particular implementation within a given service: a flow can be as programmatically complex as the service desires.

03.3.4 ·····

Security

Security in an interaction flow between two parties is comprised of three principal areas:

1 — Preventing interference by third parties (man-in-the-middle attacks, data interception and theft, etc.);

2 — Ensuring honest behavior of the second party (preventing replay attacks, non-repudiation, verifiability, etc.);

3 — Preventing the compromise or loss of the first party's private keys or equivalent sensitive information.

Third-party security primarily concerns interception and subsequent manipulation of transmitted data. Manipulation is easily detectable using cryptographic signatures (where the JWS standard is used), and interception can be prevented using encryption (by using the JWE standard). The only condition necessary for either of these safeguards to work effectively is the continued privacy of the private keys belonging to the primary parties respectively, which is a common assumption for any cryptographically secure communication.

Second-party security depends primarily on establishing a trustless verification method in order to ensure that any dishonest behavior can be detected. Once again, cryptographic signatures make detection of dishonest behavior generally possible by providing a mechanism for both verification and non-repudiation. Additionally, the nonces and expiry times of JWTs provide information which can help prevent replay or duplication attacks. For second-party security, no assumptions must be made about the state of any individual interaction token; however, this does not discount the potential for vulnerability in the design of an interaction flow of a service. First party security is a broad topic and is covered in the next section.

— *Exploits in the design of interaction flows lay outside the scope of this paper.*

03.4

Identity and key management

In light of extended usage of cryptographic signatures, suitable mechanisms for generating, managing, and recovering private keys must be provided for enabling trustable data and complex interaction protocols. It is worth pointing out that although numerous practical solutions are available today, a general comprehensive solution has not yet been adopted. Given the paramount function of private keys, and the risks associated with compromise, key persistence and key management warrant special consideration and attention.

All operations involving private keys (e.g. generating signatures for Verifiable Credentials and interaction messages, deriving identities for pairwise interactions, decrypting messages) should ideally be delegated to a secure hardware module which exposes a very limited API. In the best case scenario, private keys generated by such modules cannot be retrieved, effectively preventing exfiltration by an attacker. Dedicated hardware modules can be used to support generation and management of trustable digital identities belonging to IoT devices and backend systems, which in turn can lead to more auditable, cryptographically secure interaction protocols.

Despite the aforementioned benefits, such modules have not yet gained widespread popularity. Given how often a typical user is expected to use a digital identity (e.g. to exchange Verifiable Credentials or authenticate against services) and the rather brief duration of a typical (inter)action, the necessity to maintain key availability at all times proves to be a burden.

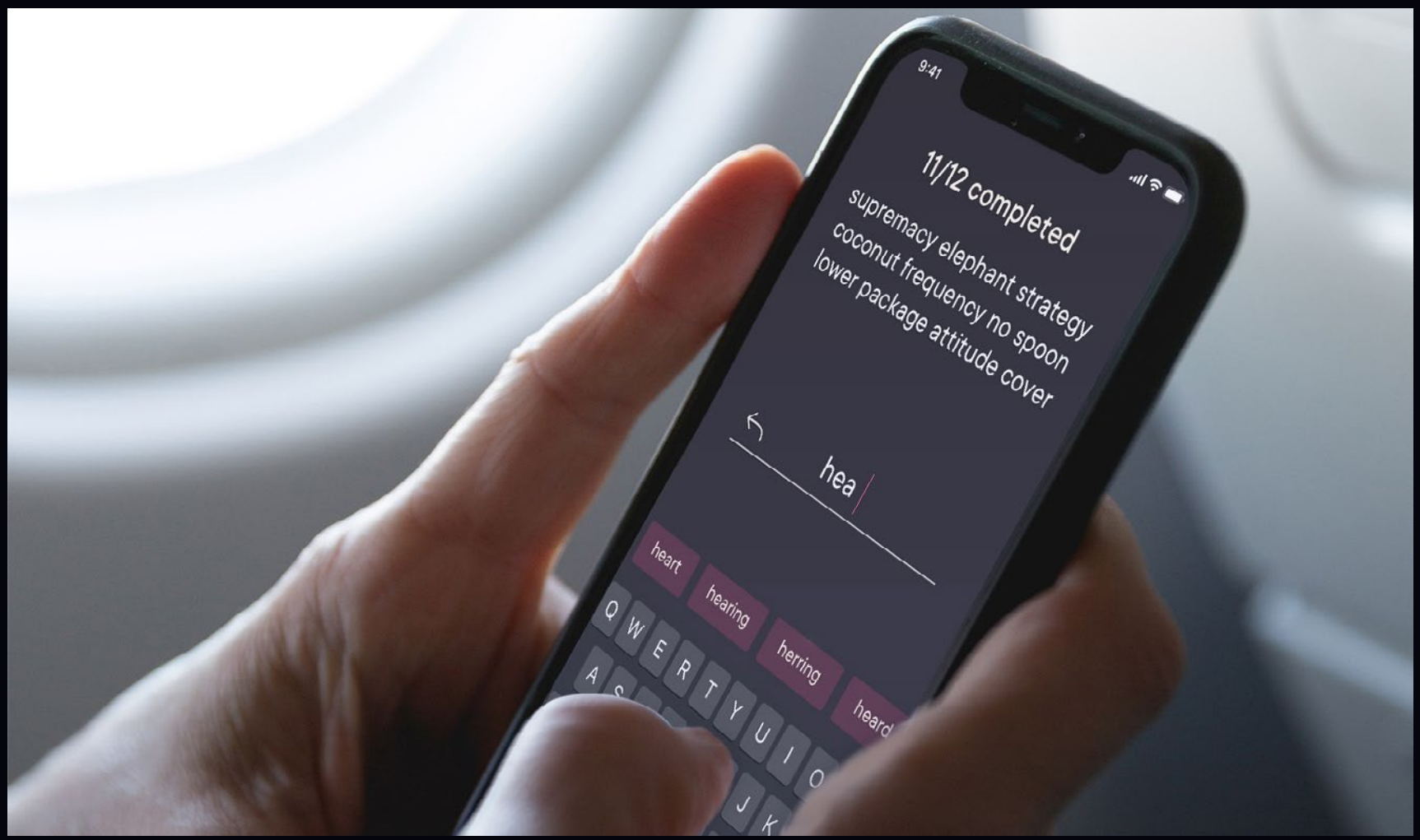
The next sections provide a brief overview of the individual aspects of key management and outline some of the main challenges associated with each. Our current approach to solving the aforementioned problems will be introduced as well.

03.5

Key recovery

A decent compromise between usability and security entails instrumentalizing a user's smartphone as a secure enclave for storing and managing the user's private keys. By securely storing a Holder's keys on a personal device that the Holder owns (e.g. by using the credential manager provided by the underlying operating system, or an encrypted database), all operations requiring the Holder's approval (in the form of a digital signature) can be delegated to the device, where a dedicated application (referred to as an "Identity Wallet") can parse the requests and initiate the appropriate user interactions.

However, in order for this compromise to work in practice, Holders must be entrusted with exclusive possession of their private keys (stored locally — meaning no copy of the keys exists elsewhere). This premise raises certain practical questions related to key management — like how might the process of key recovery generally unfold when a Holder's smartphone (along with the Holder's private keys, locally stored identity information, Identity Wallet, etc.) are no longer in the Holder's possession.



In this section we concentrate on one particular scenario, namely when a Holder can be certain that the device on which the Holder's private keys are stored is no longer usable and that the keys cannot be compromised. The alternative scenario, wherein a Holder's keys have been compromised, is outlined in §3.5.1.

The process of recovering a Holder's identity from a broken or defective device (e.g. a smartphone) in the first scenario revolves around the notion of a reliable and secure way to regenerate the cryptographic material required to derive all private keys used to participate in previous interactions. Given a simple implementation wherein a single key pair is used to sign all transactions and interaction messages, the only material in need of recovery would be the aforementioned (32-byte) private key. However, if multiple private keys are used on a regular basis, recovery would require a random seed (ranging from 16 to 32 bytes) as well as the individual derivation paths.

The simplest way to handle key recovery is to delegate certain functions to the user. This can be as simple as instructing and entrusting users to make a copy (e.g. encoded in hex) of their private keys and store the physical backup securely over time. Given the user must effectively copy and store a hexadecimal string of 64 characters in length for an indeterminate amount of time, this solution is largely error prone, suffers from bad UX, and substantially increases the risk of key compromise.

A more Holder-friendly method to implementing the recovery mechanism so far outlined involves use of a “seed phrase” (also referred to as “mnemonic phrase” [25]) to encode the aforementioned “seed". This alternative approach places less of a burden on users, e.g. by requiring Holders note down only 12 simple words in a particular order from a predefined dictionary (instead of having to persist 64 random characters). If the user is able to reintroduce the aforementioned twelve words in the correct order, the original seed will be recovered.

However, this method does admit significant vulnerability in terms of key compromise insofar as the persisted seed phrase exposes a single point of failure. One approach to mitigating the risks of seed phrase compromise caused by the need for backup persistence involves distributing the risk by sharding a seed phrase into a number of separate parts that can be persisted separately and independently. An attacker would need to find (access) K shards to recreate the seed.

A further alternative is to encrypt the sensitive data (the seed and optionally the derivation paths) using a secure encryption algorithm (e.g. the one-time-pad (OTP) as proposed in some designs [26]) and persist

This approach can be further modified so that N out of K shards are needed to recreate the original seed, which allows for some degree of fault tolerance.

the encrypted material publicly (e.g. on IPFS). Instead of a sensitive seed, a user need only persist a sensitive encryption password. Since users have a significant degree of freedom to choose the OTP used to encrypt their random seed, a number of novel sources for OTPs can be employed (e.g. the contents of a book given a specific page and paragraph, or the coordinates of a number of geographical locations).

03.5.1 ·····

Key rotation

In the event that an attacker compromises a Holder's private key, the keys associated with that identity must be replaced with a non-compromised pair in order for the Holder to regain (exclusive) control over the linked identity. A process of key rotation is used to re-secure the Holder's identity. Multiple methods for performing key rotation are available using DID Documents.

Native key rotation (in the Jolocom system) would involve a simple DID Document update with a new public key in place of the old. This is an effective measure for addressing the compromise of private key information for that particular key pair; however, if the entropy from which an entire identity is derived is compromised, there is no recourse for the identity holder. Indeed, prior to detection of compromise, an attacker could commit a new key and thereby permanently compromise the (former) Holder's control over the identity.

A more nuanced approach calls for collecting not one, but two separate pieces of entropy, and using only one piece for identity creation and control. The other would be used to derive keys which control the key rotation mechanism and allow for a single-use rotation that, when carried out,

would gather new entropy such that the process could repeat from the start with the new key sets. This method can provide an additional layer of security given that the entropy used for identity control (further stored on device) can be compromised without permanent consequences for the identity or Holder since the secondary entropy is kept safely by the identity holder (e.g. in cold storage).

Though this protection shares an analogous vulnerability to the native approach (namely, acquisition of a single private secret), separating the rotation key from the control key does allow the former to be stored in ways that would be impractical for a resource which must be accessed routinely. In particular, the key can be broken up and distributed to M (more or less trusted) parties using the Shamir-Blakely splitting algorithm, where N < M pieces must be combined to reconstitute the original secret.

This method can be further hardened by first encrypting the secret prior to splitting and distributing it. This adds an extra layer of protection for the rotation key by preventing collusion among a user's trusted parties to and potential reconstruction of the rotation key without the user's permission.

03.6

Deploying the protocol

Our reference implementation of the Jolocom Protocol makes use of the following components and technologies to support a flexible, yet robust decentralized identity management system.

User Interface

The user interface exists in an independent repository from the Jolocom Library, but consumes the endpoints exposed by the Library. The Jolocom SmartWallet is the default user interface to both create and manage identities and the credentials associated with each identity, and allows for this management in a visual and user-friendly way.

Public Blockchain

This is the trusted storage layer for storing the mapping of each unique, decentralized, and permanent global identifier to associated documents containing metadata. We currently implement the Ethereum [27] blockchain (Rinkeby Testnet) as our trust layer in our reference implementation, but have successfully deployed to other chains as well for certain use cases.

See §5.2 for further information about the Jolocom SmartWallet.

- **Storage Backend**

  This refers generically to any storage backend, public or private; there are no artificial constraints on the particular technology used for the backend. We currently use InterPlanetary File System (IPFS) [28] as our default public storage backend for storing the DID Document, which contains documents containing content-addressed hashes for the specific retrieval of specific public Verifiable Credentials associated with an identity (identifier). The default storage option for private claims will be directly on the user's (personal) device.

- **Current implementation**

  Our current reference implementation of the protocol enables users to create and interact with self-sovereign digital identities. Users create a Jolocom identity using the Jolocom SmartWallet application. The application generates an Ethereum address using entropy created within the application, including user input. The public key is hashed to create a DID, and the DID and a public profile credential are both stored on IPFS. The user spends Ether to update the Jolocom registry smart contract, which maps DIDs to IPFS addresses.

Once a DID is registered, most activity involving the identity will occur locally on the user's device. The user can add self-signed credentials through the Jolocom application, and Verifiable Credentials can be received and managed as part of the local identity as JWTs. To add a credential, the user can scan a QR code containing a Verifiable Credential using the SmartWallet. The user can also generate credentials to issue to other individuals. Making changes to the user's DID file or entry in the Jolocom registry are necessary only to update the user's public profile.

04

Conclusion

Jolocom is developing a decentralized, open source infrastructure solution to digital identity and access management which places users in control of their data. The Jolocom Protocol provides users with a digital identity based on cryptographic keys that are locally generated, provisioned and controlled by the users themselves. An identity based on cryptographic keys is designed to facilitate the management of multiple 'personas' and preservation of pairwise anonymity in context-specific interactions, as well as recovery of all of these derived key pairs with a simple seed phrase.

Persons, organizations, IoT devices, and autonomous agents (e.g. DAOs) can use the Jolocom Protocol to create decentralized identities using Decentralized Identifiers, Verifiable Credentials, and cryptographic signatures — the core building blocks of Jolocom identities. The protocol logic encodes a granular, claim-based model of identity that is highly generalized and unrestrictive in scope in order to accommodate a multiplicity of potential use cases and broad range of subjects of identity (users). Our aim is to support a new status quo of greater user control and privacy that aligns with the concept of Self-Sovereign Identity.

For the most up-to-date information on our development, visit our website at jolocom.io.

05 Additional resources

05.1 Libraries and tools for development

Jolocom Library

A library for interacting with the identity solution provided by Jolocom.

Source code: github.com/jolocom/jolocom-lib

Documentation: jolocom-lib.readthedocs.io

Jolocom SmartWallet

A decentralized Self-Sovereign Identity solution developed by Jolocom.
See §5.2 for further information.

Source code: github.com/jolocom/smartwallet-app

Generic Backend

A generic backend implementation that makes use of the Jolocom Library for authentication, and for issuing credentials.

Source code: github.com/jolocom/generic-backend

Registry Contract

A basic smart contract on top of the Ethereum network that contains mappings between DIDs and references (hashes) to DID Documents persisted on IPFS.

Source code: github.com/jolocom/registry-contract

Jolocom CLI

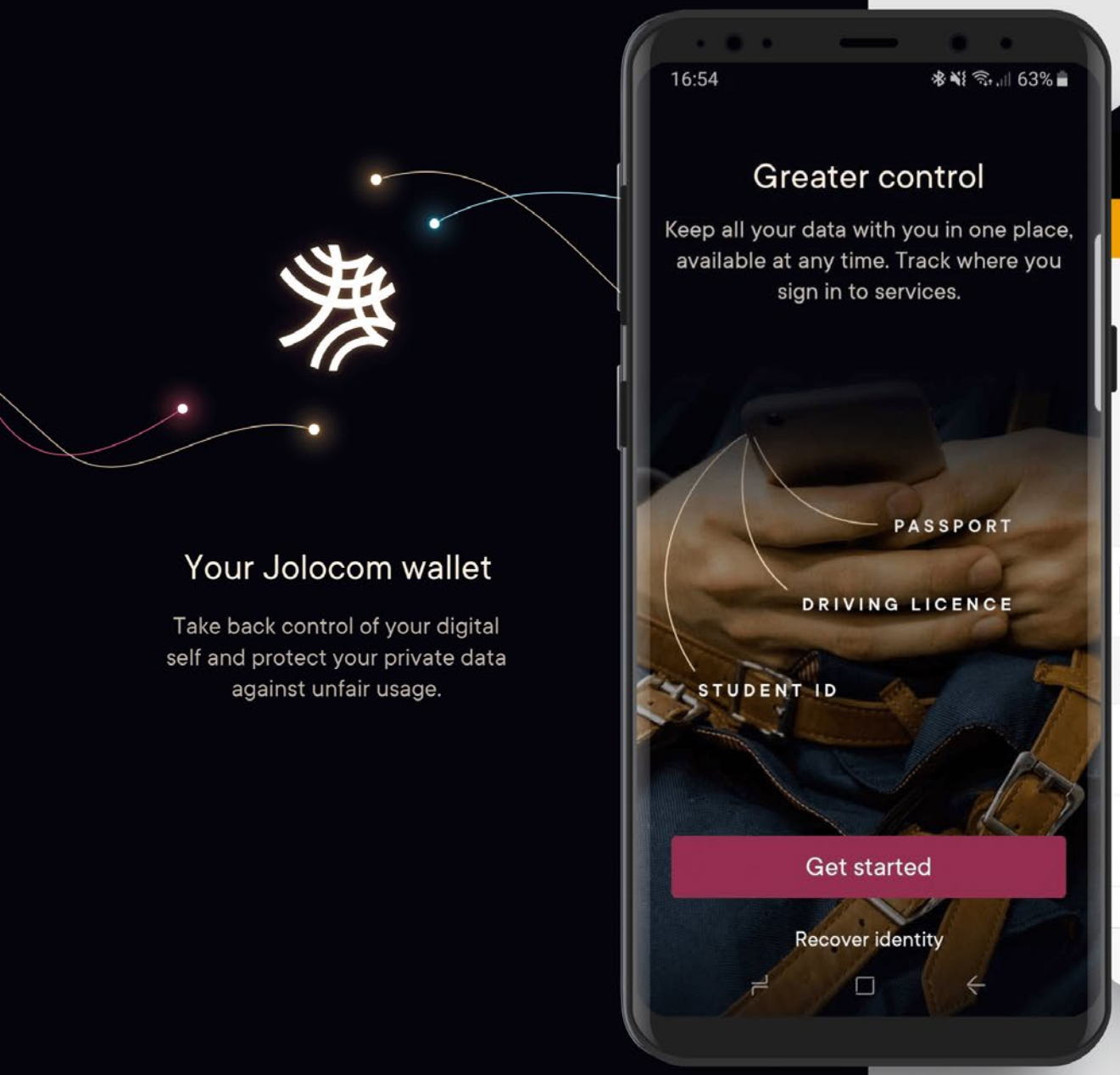
A simple tool for interacting with the Jolocom identity infrastructure from the command line.

Source code: github.com/jolocom/jolocom-cli

Jolocom Universal Resolver

A Universal Resolver DID Driver for the did:jolo identity space.

Source code: github.com/jolocom/jolocom-did-driver



Your Jolocom wallet

Take back control of your digital self and protect your private data against unfair usage.

Download on the App Store

GET IT ON Google Play

05.2

The Jolocom SmartWallet

Designed for use on personal, portable devices (such as smartphones and tablets), the Jolocom SmartWallet is a mobile application that allows individuals to easily, conveniently, and securely manage their digital identity credentials. The SmartWallet serves as an interface to the protocol and makes it easy to use the protocol's various functionalities for identity management. This focus on accessibility makes it possible for users of varying technical familiarities and proficiencies to take advantage of the app's features and capabilities.

The Jolocom SmartWallet is versioned for devices running Android (4.1 and up) as well as iOS (9.0 or later) and can be downloaded as follows:

GET IT ON Google Play

Download on the App Store

After downloading and installing the app, users can conveniently request and share identity data in an automated fashion. The SmartWallet enables users to easily generate data, secure data on local storage, attach meaningful information to digital attributes, and exchange claims — all without going through a third-party service.

05.3

Demos

The following sets of demo services are available for users of all levels of technical familiarity to try out in combination with the Jolocom SmartWallet.

05.3.1

‘SSI Building Blocks’

As technical partner during Phase 1 of Blockchain on the Move, we were eager to showcase how Self-Sovereign identity and decentralized technologies could practically benefit both Citizen & State by enabling individuals to self-manage and own their own data and identity. Three scenarios with well-defined use cases were selected as a basis for developing our technical building blocks into a full SSI solution.

Using open source libraries that enable interaction with Jolocom’s protocol for decentralized identity management, we developed three separate services and web platforms to handle the types of interaction flows and exchange of data specific to each use case. Each service utilizes a different combination of library functionalities to manage the logic of interactions and flow of data within the given use case.

Municipal Service
Get a 360° look at the public sector benefits of SSI and digital credential issuance while you load up on some demo credentials during a pretend trip to the local municipal official.

botm-municipal-demo.jolocom.com

University Service

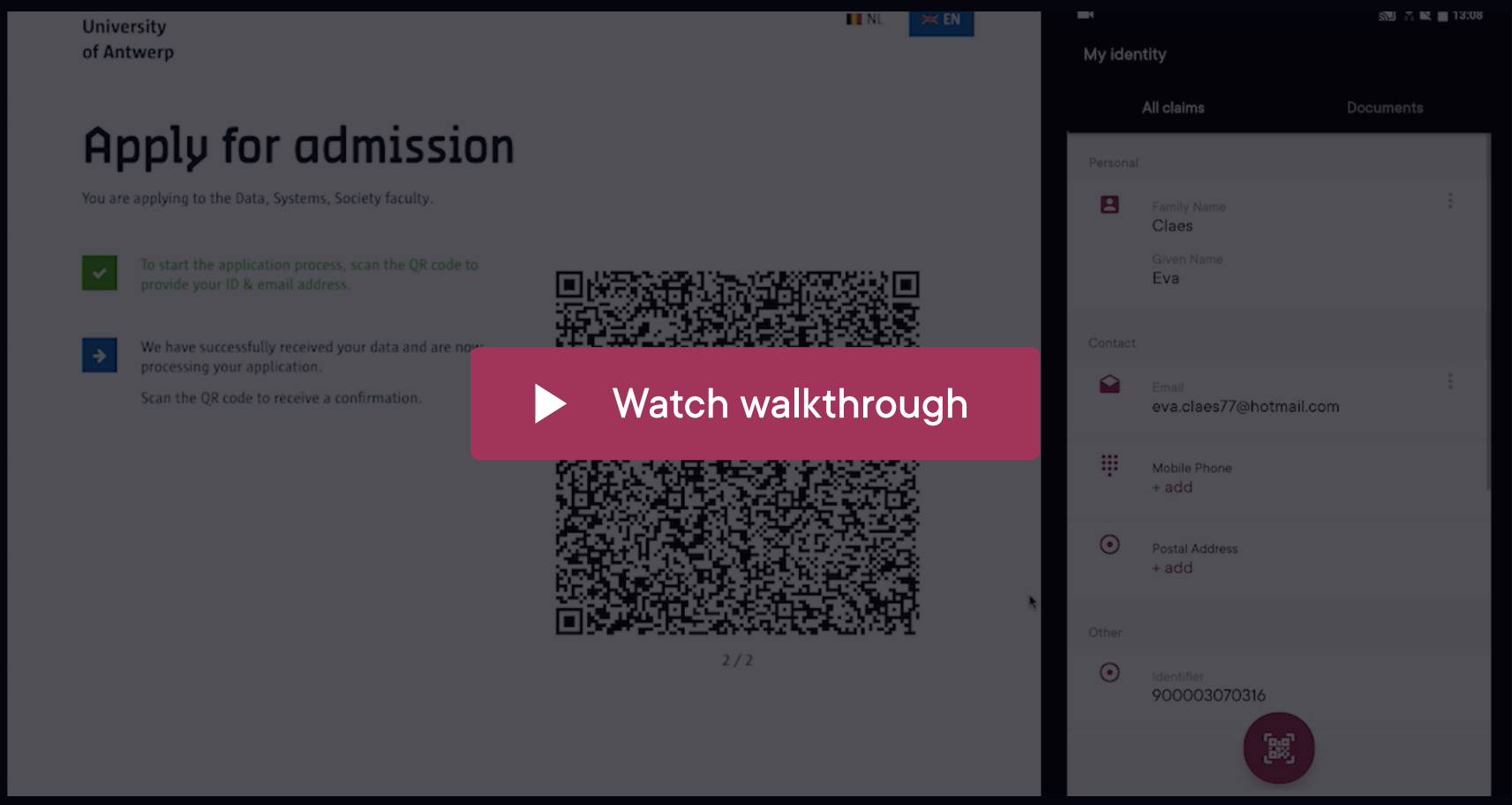
Use your demo digital ID card credential from the Municipal Service to begin the process of applying for enrollment in university admissions.

botm-university-demo.jolocom.com

Swimming Pool Service

Use your A-kaart voucher from the Municipal Service to get a discount on entry to the local swimming pool.

botm-pool-demo.jolocom.com

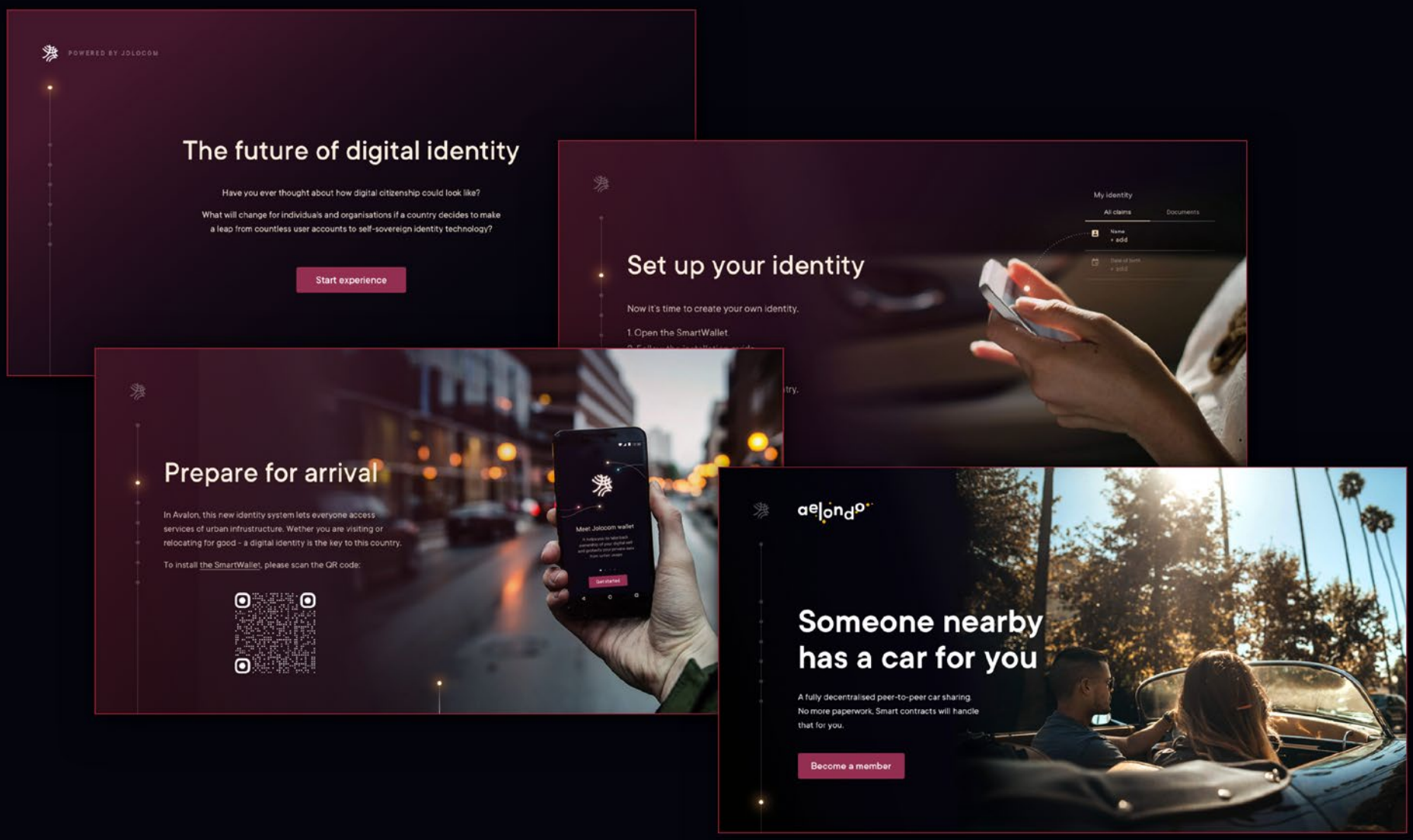


Watch walkthrough

Read the tech overview

05.3.2 .....

Avalon + Aelondo



Avalon

Register your arrival in the (fictional) city of Avalon.

avalon.jolocom.com

Aelondo

Rent a car through Aelondo, a demo ride sharing service.

aelondo.jolocom.com

Follow the user guide

05.4

Further reading

Self-sovereign Identity ↗

published 23 October 2018
by Blockchain Bundesverband e.V.

A position paper on decentralized identity written from the perspective of the German blockchain identity scene. It outlines the common vision on topics such as standardization, GDPR, and security requirements as regard enabling a universal identity infrastructure. Contributors include Kai Wagner (Jolocom).

Blockchain, data protection, and the GDPR ↗

published 25 May 2018
by Blockchain Bundesverband e.V.

A position paper on major improvements to the EU's General Data Protection Regulation going into effect in May 2018 and ways to deal with data protection on the blockchain effectively. Contributors include Kai Wagner (Jolocom).

Introduction to DID Auth ↗

published 31 July 2018
by Rebooting the Web of Trust

A white paper produced during a design workshop at Rebooting the Web of Trust VI: Santa Barbara (March 2018). A continuation of ongoing work by Rebooting the Web of Trust (RWOT) and other communities, the paper provides overview of the scope of DID Auth, supported protocols and flows, and the use of components of the DID Documents that are relevant to authentication. Contributors include Eugeniu Rusu (Jolocom).

Blockchain:
Opportunities and challenges of
a new digital infrastructure for Germany ↗

published 16 October 2017
by Blockchain Bundesverband e.V.

A position paper on blockchain and similar decentralized technologies based on cryptography for use as basic infrastructural innovations that further support a digital economy built on democratic structures. Co-authors include Joachim Lohkamp (Jolocom).

06

References

1 Hypertext Transfer Protocol. https://tools.ietf.org/html/rfc2616

2 Simple Mail Transfer Protocol. https://tools.ietf.org/html/rfc821

3 W3C. "Decentralized Identifiers (DIDs)" (section 4), in "Decentralized Identifiers (DIDs) v1.0," *Dec 2019. W3C Working Draft.* https://w3c-ccg.github.io/did-spec/#decentralized-identifiers-dids

4 W3C. "DID Documents" (section 5), in "Decentralized Identifiers (DIDs) v1.0," *Dec 2019. W3C Working Draft.* https://w3c-ccg.github.io/did-spec/#did-documents

5 W3C. "Verifiable Credentials Data Model 1.0," *Nov 2019. W3C Recommendation.* https://www.w3.org/TR/vc-data-model

6 Web of Trust. https://www.weboftrust.info

7 Allen, C., Brock A., Buterin, V., Callas, J., Dorje, D., Lundkvist, C., Kravchenko, P., Nelson, J., Reed, D., Sabadello, M., Slepak, G., Thorp, N., & Wood, H.T. "Decentralized Public Key Infrastructure," *Dec 2015. Rebooting the Web of Trust I; San Francisco (Nov 2015).* https://github.com/WebOfTrustInfo/rebooting-the-web-of-trust/blob/master/final-documents/dpki.pdf

8 Reed, D. "Decentralized Key Management System," *Apr 2017. Rebooting the Web of Trust IV: Paris (April 2017).* https://github.com/WebOfTrustInfo/rwot4-paris/blob/master/topics-and-advance-readings/dkms-decentralized-key-mgmt-system.md

9 Jolocom. “The Jolocom Protocol - Own Your Digital Self,” *2018. Read the Docs.* https://jolocom-lib.readthedocs.io/en/latest/index.html

10 Allen, C. “The Path to Self-Sovereign Identity,” *25 Apr 2016. Life With Alacrity.* http://www.lifewithalacrity.com/2016/04/the-path-to-self-soverereign-identity.html

11 Jolocom DID Method Specification. *Mar 2019.* https://github.com/jolocom/jolocom-did-driver/blob/master/jolocom-did-method-specification.md

12 The Jolocom Library. https://github.com/jolocom/jolocom-lib

13 Satoshi, N. “Bitcoin: A Peer-to-Peer Electronic Cash System,” *2009.* https://bitcoin.org/bitcoin.pdf

14 BigchainDB GmbH. “BigchainDB 2.0: The Blockchain Database,” *May 2018.* https://www.bigchaindb.com/whitepaper/bigchaindb-whitepaper.pdf

15 Quorum. https://github.com/jpmorganchase/quorum

16 Sidetree Protocol Specification. *Jun 2019.* https://github.com/decentralized-identity/sidetree/blob/master/docs/protocol.md

17 Peer DID Method Specification. *May 2019. W3C Editor's Draft.* https://dhh1128.github.io/peer-did-method-spec/index.html

18 "Hash doc method," *May 2019. Merged pull request (32) on "Peer DID Method Spec" (see [27]).* https://github.com/openssi/peer-did-method-spec/pull/32

19 See [5]

20 "JSON Web Signature (JWS)," *May 2015. Internet Engineering Task Force.* https://tools.ietf.org/html/rfc7515

21 JSON. https://json.org

22 “Introduction to JSON Web Tokens.” *Auth0.* https://jwt.io/introduction

23 “JSON Web Encryption (JWE),” *May 2015. Internet Engineering Task Force.* https://tools.ietf.org/html/rfc7516

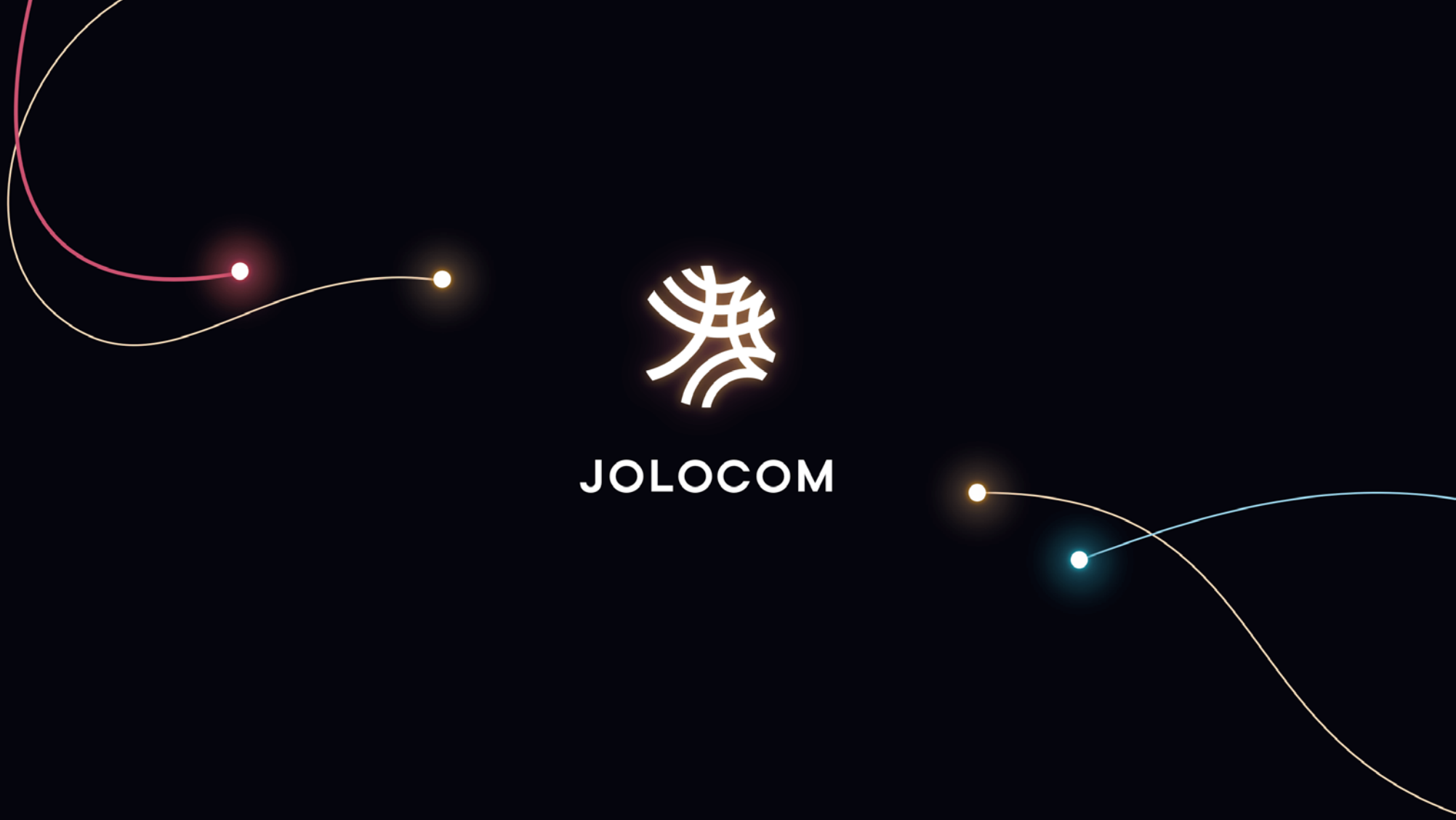
24 See [4]

25 Palatinus, M., Rusnak, P., Voisine, A., & Bowe, S. "Mnemonic code for generating deterministic keys," *2013. Bitcoin Improvement Proposal 39.* https://github.com/bitcoin/bips/blob/master/bip-0039.mediawiki

26 Smith, S. M., & Gupta, V. “Decentralized Autonomic Data (DAD) and the three R’s of Key Management,” *May 2018. Rebooting the Web of Trust VI: Santa Barbara (March 2018).* https://github.com/WebOfTrustInfo/rwot6-santabarbara/blob/master/final-documents/DecentralizedAutonomicData.pdf

27 Ethereum. “A Next-Generation Smart Contract and Decentralized Application Platform,” *Jun 2019.* https://github.com/ethereum/wiki/wiki/White-Paper

28 Benet, J. “IPFS - Content Addressed, Versioned, P2P File System (draft 3),” *2014.* https://ipfs.io/ipfs/QmV9tSDx9UiPeWExXEeH6aoDvmihvx6jD5eLb4jbTaKGps



Contact us

or via

hello@jolocom.io

jolocom.io

Get our monthly digest