a4wide verbatim [toc,page]appendix [withpage]acronym amsthm

xcolor LightBluergb0.55,0.55,1 DarkBluergb0.2,0.2,0.5 DarkRedrgb0.7725490196,0.05490196078,0.12549019607 Blackrgb0,0,0

geometry inner=3cm, outer=2cm, bottom=4cm

style/noindent

[hyphens]url hyphenat

mathpazo [scaled=.95]helvet courier

textcomp

tabularx booktabs multirow longtable

amsmath amssymb setspace

[english]babel

english

[T1]fontenc [utf8]inputenc

[babel,english=american]csquotes

units

appendix

listings

violetcmyk0.45,0.97,0.27,0.21 lstbluecmyk1,0.80,0,0 lstgreencmyk0.71,0.21,0.65,0.22 bluegreycmyk0.56,0.24,0.11,0.05 javadoccmyk0.88,0.59,0,0 lstgreycmyk0.55,0.44,0.42,0.32

SQL keywords=, keywordstyle=bluegrey**, morekeywords=[2]CREATE,TABLE,IF,NOT,EXISTS,NUL** $[2]violet, otherkeywords = int, varchar, double, text, tinyint, sensitive = false, morecomment = [l][lstgreen]//, morecomment = [s][lstgreen]/**/, morecomment = [s][javadoc]/***/, morestring = [b]', morestring = [b]"PHPkeywords = , keywordstyle = bluegrey, morekeywords = [2]static, function, if, retu$

json basicstyle=, `numbers=left, numberstyle=, stepnumber=1, numbersep=8pt, showstringspaces=false, breaklines=true, frame=lines, backgroundcolor=background, literate= *0numb01 1numb11 2numb21 3numb31 4numb41 5numb51 6numb61 7numb71 8numb81 9numb91 :punct:1 ,punct,1 {delim{1 }delim}1`

Java keywords=, keywordstyle=bluegrey, morekeywords=[2]abstract,boolean,break,byte,case,cat const,continue,default,do,double,else,extends,false,final, finally,float,for,goto,if,implements,import,i interface,label,long,native,new,null,package,private,protected, public,return,short,static,super,swit throws,transient,true,try,void,volatile,while, keywordstyle=[2]violet, morekeywords=[3]@SuppressWarnings, @Capability, @Override, keywordstyle=[3]lstgrey, otherkeywords=@param, @return, @returns, @author, @link, @see, sensitive, morecomment=[l]//, morecomment=[s]//, morecomment=[s][javadoc]/***/, morestring=[b]", morestring=[b]', [keywords,comments,strings]**

HTML5 morekeywords=a, abbr, address, area, article, aside, audio, b, base, bb, bdo, blockquote, body, br, button, canvas, caption, cite, code, col, colgroup, command, datagrid, datalist, dd, del, details, dialog, dfn, div, dl, dt, em, embed, eventsource, fieldset, figure, footer, form, h1, h2, h3, h4, h5,

h6, head, header, hr, html, i, iframe, img, input, ins, kbd, label, legend, li, link, mark, map, menu, meta, meter, nav, noscript, object, ol, optgroup, option, output, p, param, pre, progress, q, ruby, rp, rt, samp, script, section, select, small, source, span, strong, style, sub, sup, table, tbody, td, textarea, tfoot, th, thead, time, title, tr, ul, var, video, sensitive=false, morecomment=[s]¡!––¿, morestring=[b]", morestring=[d]'

CSS morekeywords=azimuth, background-attachment, background-color, background-image, background-position, background-repeat, background, border-collapse, border-color, border-spacing, border-style, border-top, border-right, border-bottom, border-left, border-top-color, border-right-color, border-bottom-color, border-left-color, border-top-style, border-right-style, border-bottom-style, border-left-style, border-top-width, border-right-width, border-bottom-width, border-left-width, border-width, border, bottom, caption-side, clear, clip, color, content, counter-increment, counter-reset, cue-after, cue-before, cue, cursor, direction, display, elevation, empty-cells, float, font-family, font-size, font-style, font-variant, font-weight, font, height, left, letter-spacing, line-height, list-style-image, list-style-position, list-style-type, list-style, margin-right, margin-left, margin-top, margin-bottom, margin, max-height, max-width, min-height, min-width, orphans, outline-color, outline-style, outline-width, outline, overflow, padding-top, padding-right, padding-bottom, padding-left, padding, page-break-after, page-break-before, page-break-inside, pause-after, pause-before, pause, pitch-range, pitch, play-during, position, quotes, richness, right, speak-header, speak-numeral, speak-punctuation, speak, speech-rate, stress, table-layout, text-align, text-decoration, text-indent, text-transform, top, unicode-bidi, vertical-align, visibility, voice-family, volume, white-space, widows, width, word-spacing, z-index, sensitive=false, morecomment=[s]/**/, morestring=[b]", morestring=[d]'

JavaFX morekeywords=abstract, after, and, as, assert, at, attribute, before, bind, bound, break, catch, class, continue, def, delete, else, exclusive, extends, false, finally, first, for, from, function, if, import, indexof, in, init, insert, instanceof, into, inverse, last, lazy, mixin, mod, new, not, null, on, or, override, package, postinit, private, protected, public-init, public, public-read, replace, return, reverse, sizeof, static, step, super, then, this, throw, trigger, true, try, tween, typeof, var, where, while, with , sensitive=false, morecomment=[l]//, morecomment=[s]/**/, morestring=[b]", morestring=[d]'

MXML morekeywords=mx:Accordion, mx:Box, mx:Canvas, mx:ControlBar, mx:DividedBox, mx:Form, mx:FormHeading, mx:FormItem, mx:Grid, mx:GridItem, mx:GridRow, mx:HBox, mx:HDividedBox, mx:LinkBar, mx:Panel, mx:TabBar, mx:TabNavigator, mx:Tile, mx:TitleWindow, mx:VBox, mx:VDividedBox, mx:ViewStack, mx:Button, mx:CheckBox, mx:ComboBase, mx:ComboBox, mx:DataGrid, mx:DateChooser, mx:DateField, mx:HRule, mx:Image, mx:Label, mx:Link, mx:List, mx:Loader, mx:MediaController, mx:MediaDisplay, mx:MediaPlayba mx:MenuBar, mx:NumericStepper, mx:ProgressBar, mx:RadioButton, mx:RadioButton

mx:Spacer, mx:Text, mx:TextArea, mx:TextInput, mx:Tree, mx:VRule, mx:VScrollBar, mx:Application, mx:Repeater, mx:UIComponent, mx:UIObject, mx:View, mx:FlexExtension, mx:UIComponentExtension, mx:UIObjectExtension, mx:Fade, mx:Move, mx:Parallel, mx:Pause, mx:Resize, mx:Sequence, mx:WipeDown, mx:WipeLeft, mx:WipeRight, mx:WipeUp, mx:Zoom, mx:EventDispatcher, mx:LowLevelEvents, mx:UIEventDispatcher, mx:CurrencyFormatter, mx:DateFormatter, mx:NumberFormatter, mx:PhoneFormatter, mx:ZipCodeFormatter, mx:CursorManager, mx:DepthManager, mx:DragManager, mx:FocusManager, mx:HistoryManager, mx:LayoutManager, mx:OverlappedWindows, mx:PopUpManager, mx:SystemManager, mx:TooltipManager, mx:CreditCardValidator, mx:DateValidator, mx:EmailValidator, mx:NumberValidator, mx:PhoneNumberValidator, mx:SocialSecurityValidator, mx:StringValidator, mx:ZipCodeValidator, mx:DownloadProgressBar, mx:ArrayUtil, mx:ClassUtil, mx:Delegate, mx:ObjectCopy, mx:URLUtil, mx:XMLUtil, mx:CSSSetStyle, mx:CSSStyleDeclaration, mx:CSSTextStyles, mx:StyleManager, mx:HTTPService, mx:RemoteObject, mx:Service, sensitive=false, morecomment=[s]¡!—¿, morestring=[b]", morestring=[d]'

LZX morekeywords=a, alert, animator, animatorgroup , attribute, audio , axis, axisstyle , b, barchart, basebutton , basebuttonrepeater , basecombobox , basecomponent , basedatacombobox , basedatepicker , basedatepickerday , basedatepickerweek , basefloatinglist , basefocusview , baseform , baseformitem , basegrid , basegridcolumn , baselist , baselistitem , basescrollarrow , basescrollbar , basescrollthumb , basescrolltrack , baseslider , basestyle , basetab , basetabelement , basetabpane , basetabs , basetabsbar , basetabscontent , basetabslider , basetrackgroup , basetree , basevaluecomponent , basewindow , br , button , canvas , chart , chartbgstyle , chartstyle , checkbox , class , columnchart , combobox , command , connection , connectiondatasource , constantboundslayout , constantlayout , datacolumn , datacombobox , datalabel , datamarker , datapath , datapointer , dataselectionmanager , dataseries , dataset , datasource , datastyle , datastylelist , datatip , datepicker , debug , dragstate , drawview , edittext , event , face , floatinglist , font , font , form , frame , grid , gridcolumn , gridtext , handler , hbox , horizontalaxis , hscrollbar , i , image , img , import , include , inputtext , javarpc , label , labelstyle , layout , legend , library , linechart , linestyle , list , listitem , LzTextFormat , menu , menubar , menuitem , menuseparator , method , modaldialog , multistatebutton , node , p , param , piechart , piechartplotarea , plainfloatinglist , plotstyle , pointstyle , pre , radiobutton , radiogroup , rectangularchart , regionstyle , remotecall , resizelayout , resizestate , resource , reverselayout , richinputtext , rpc , script , scrollbar , security , selectionmanager , sessionrpc , simpleboundslayout , simpleinputtext , simplelayout , slider , soap , splash , stableborderlayout , state , statictext , style , submit , swatchview , SyncTester , tab , tabelement , tabpane , tabs , tabsbar , tabscontent , tabslider , Test , TestCase , TestResult , TestSuite , text , textlistitem , tickstyle , tree , u , valueline , valuelinestyle

, valuepoints , valuepointstyle , valueregion , valueregionstyle , vbox , verticalaxis , view , view , vscrollbar , webapprpc , window , windowpanel , wrappinglayout , XMLHttpRequest , xmlrpc , zoomarea, sensitive=false, morecomment=[s]¡!—¿, morestring=[b]", morestring=[d]'

numbers=left, numberstyle=, numbersep=5pt, breaklines=true, stepnumber=1, tabsize=2, basicstyle=, frame=none, numberfirstline=true, firstnumber=1, keywordstyle=violet, ndkeywordstyle=identifierstyle=black, commentstyle=lstgreen, stringstyle=lstblue, showstringspaces=false

enumitem font=Black

graphicx images/

eso-pic chngpage tikz

[hang,small,sf]caption

float placeins

[it,bf,tight,hang,raggedright]subfigure

calc fancyhdr [RO,LE]0.1cm [RE,LO]0.1cm [RE] [LO] [RO,LE]4

[LE] [width=0.6]images/snet/snet$_{footer.png}$

* [width=height=keepaspectratio=false]images/snet/titlepage.pdf

.9 DarkRedIntegrating Didcomm Messaging to ActivityPub-based Social Networks

by

Adrian Isaias Sanchez Figueroa

Matriculation Number 397327

A thesis submitted to

Technische Universität Berlin
School IV - Electrical Engineering and Computer Science
Department of Telecommunication Systems
Service-centric Networking

Bachelor's Thesis

July 4, 2022

Supervised by:
Prof. Dr. Axel Küpper

Assistant supervisor:
Dr. Sebastian Göndör

BlackEidestattliche Erklärung / Statutory Declaration

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und eigenhändig sowie ohne unerlaubte fremde Hilfe und ausschließlich unter Verwendung der aufgeführten Quellen und Hilfsmittel angefertigt habe.

I hereby declare that I have created this work completely on my own and used no other sources or tools than the ones listed.

Berlin, July 4, 2022 **Adrian Sanchez**

BlackAbstract

In this thesis, we show that lorem ipsum dolor sit amet.

BlackZusammenfassung

Hier kommt das deutsche Abstract hin. Wie das geht, kann man wie immer auf Wikipedia nachlesen `http://de.wikipedia.org/wiki/Abstract`**...**

BlackContents

Black1 Introduction

This section gives an introduction into the general field in which you are writing your thesis. It further describes the situation today, and the problems that are solved and not.

One of the most accepted and used definitions of OSN services was given by Boyd and Ellison in [11], who defined a Social Network Site (SNS) as a web based service that allows "[...] individuals to

Black1.1 Motivation

Why choose this topic? Because identities are important. Web3 brings decentralization, and identities should stay behind. W3C provides us with new standards for interoperability, and we should take this goals in mind.

Black1.2 Digital Identities

The evolution of the digital identity

Almost every person who uses computers or accesses the Internet today has some form of digital identity.

An Identity is, at its most basic level, a collection of claims about a person, place or thing. These claims are issued by centralized entities and then stored in centralized databases. Physical forms of identification have many downsides. For instance, they are not widely available to every human. According to the ID4D Initiative [https://id4d.worldbank.org/global-dataset], around 1.1 billion people worldwide do not have a way to claim ownership of their identity, leaving them incapable to perform actions where identity uniqueness is important. For example voting in elections, accessing financial systems or governmental services.

Identities are integral to a functioning society and economy. Being able to identify ourselves and our possessions enable us to create thriving societies and global markets.

Historically, identity management has never been User-centric. The user has always had to give control of its own identity. How many times have we created an account to use a service online? Right now my password manager,

for which I have to pay, has a count of 464 different log in details.

Identity management comprises a series of different processes to identity, authenticate and authorize users to access systems or services.

Human or object identities are stored in multiple centralised or federated systems such as government, ERP, IoT, or manufacturing systems. From the standpoint of cryptographic trust verifcation, each of these centralised authorities serves as its own root of trust. An entity trailing along a value chain is interacting with multiple systems. Consequently, a new actor in any given value chain has no method to independently verify credentials of a human or attributes of either a physical object or data item (provenance, audit trail). This results in the existence of complex validation, quality inspection, and paper trail processes, and enormous hidden trust overhead costs are added to all value chains and services (DID for everything).

Identity Management Identity management, sometimes called identity and access management, is composed of all the different ways to identify, authenticate and authorize someone to access systems or services within an organization or associated organizations.

There are several problems with our current identity management systems:

A paper-based identification such as a passport, birth certificate or driver's license is easy to lose, copy or be lost to theft. The bureaucracy behind this type of identity management is typically slow and hard to organize. The current identity and access management systems are storing your data on a centralized server along with everybody else's. This puts your digital property in danger as centralized systems are huge targets for hackers. Since 2019 alone, over 16 billion records have been leaked due to hacks and data breaches. This includes credit card numbers, addresses, phone numbers and other highly sensitive personal data. Current identities are not easily portable or verifiable. Blockchain identity management solves these issues.

Black1.3 Missing factors

It was this gap in technology that prevented digital currencies from being created or adopted in the past. Blockchain fills the gap for currencies, but it also fills the gap for decentralized identity systems, opening up a whole new world of possibilities for data ownership.

Your Blockchain-Based Digital Identity At least 1 billion people worldwide are unable to claim ownership over their identities. This is one of the huge drawbacks of physical identity documentation. It's not widely available in every country. This leaves 1/7th of the entire planet unable to vote, open a bank account, or, in some cases, find a job.

Our current identity management systems are unfair and outdated, but

there is a blockchain ID solution: a decentralized identity system that will revolutionize digital freedom. This is more critical now than ever before, with centralized companies left, right and center hoping to create the metaverse.

Your digital identity will be portable and verifiable all over the world, at any time of day. In addition, a blockchain-based decentralized identity is both private and secure. With verifiable credentials, your decentralized identity will empower you to interact with the SmartWeb without restrictions.

Your unique digital identity.

Imagine being able to verify your education qualifications or your date of birth without needing to actually show them. For example, a university degree could theoretically be on the blockchain and you could certify the credentials by checking the university or other issuing authority.

Similarly, you wouldn't have to show your physical ID to verify your date of birth. The authority that wants to know your age could instead use decentralized identifiers and verifiable credentials to find out if you're of the required age or not.

Authorization can be conducted in a trustless manner in which the digital identity in question is verified by an external source, and the person or organization checking can in turn verify the integrity of said source.

The verification of a proof is established by the verifier's judgment of the trustworthiness of the testimony.

This is known as a zero-knowledge proof.

Everyone in a distributed network has the same source of truth. This guarantees the authenticity of data without having to store it on the blockchain.

Blockchain technology has made it possible, for the first time ever, to have a trustlessly verifiable self-sovereign identity.

Blockchain Identity Management Blockchain identity solutions, such as Elastos DIDs, integrate state-of-the-art cryptography technology to ensure that your data is protected and private. By using decentralized identifiers, we can rebuild the structure of several flailing industries.

All of the following have poor identity management and could do with being brought up to date.

Government Healthcare Education Banking Business There are tons more that could be added to this list, but broadly speaking, these five categories are crucial to society as we know it. By incorporating decentralized identifiers into these fields, we can minimize bureaucracy-related inefficiencies, protect data privacy and let users own their own data.

In DeFi (decentralized finance), for example, imagine being able to take out a loan without collateral. It sounds absurd right now. To take out loans

from a bank, we need to provide a credit rating to show that we can be trusted to pay it back. By owning our own data through a decentralized identity, we can port our credit score onto the blockchain.

Effectively, we can use verifiable credentials to prove our trust and take out a loan on the blockchain without needing to deposit any collateral. CreDA, a credit oracle founded by Feng Han, the co-founder of Elastos, is currently working towards this goal. By combining the power of Elastos DIDs with a unique NFT (non-fungible token) concept, CreDA will enable you to mint your data as an NFT.

Learn more about this groundbreaking innovation here.

Decentralized Identifiers: 3 Reasons To Avoid Putting Personal Data Directly on the Blockchain While DIDs enable you to control your data, it's important to note that, for security reasons, it's best to not actually put sensitive data directly onto the blockchain. This is why we have zero-knowledge proofs to verify credentials.

The 3 Main Benefits of Having a Decentralized Digital Identity 1. Accessibility As mentioned earlier, a large portion of the world lives without physical identities, let alone digital identities. By utilizing the power of a distributed ledger, a decentralized identity becomes available to everybody. Your data is yours to claim for yourself.

2. Data Security And Protection Current centralization makes the storing of personal data dangerous. It's highly likely to be leaked or hacked at some point. With a decentralized identity, you own your data as digital property. Its value is determined by you and you alone.

It will also be secured by a private key, just like your cryptocurrencies. This means that the process of verification or authentication is only possible if you know the key. Not only does this increase security, it helps you control your data. Check out data security tips here.

3. Superior User Experience With one universal login—using the verifiable credentials on your digital identity—you can access the entire web without needing to create new accounts and remember hundreds of passwords. One day, we'll look back on the amount of accounts and passwords we were required to have for authentication and laugh like we do at dial-up broadband and the sheer size of 80s computers. (https://elastos.info/decentralized-identity-dids/)

Black1.4 Problem statement

This section explains why this is important, why it is a problem, and why this hasn't been solved already yet. Centralized Identities, secure and open communication protocols. ActivityPub implementations at the present moment

rely on HTTPS as their transport, which in turn relies on two centralized systems: DNS and SSL certificate authorities. Is there any way to bring self-sovereignty to the federated social web? [?]

ActivityPub security concerns. Encryption, non-repudiation, confidentiality. . . .. ? → No agreed-upon security mechanisms for ActivityPub. → No encryption in scope of ActivityPub. Research Questions What are the implications of introducing DIDs to Mastodon and ActivityPub in terms of usability, discovery and human-readability? Can a DID-compliant ActivityPub protocol use DIDComm for its standard communication? Can DIDs allow ActivityPub to stop relying on the DNS for its server-to-server discovery? Can DIDComm allow ActivityPub to stop relying on transport-level security for its communication?

Black1.5 Expected Outcome

The goal of this thesis is to have a fully functioning Mastodon instance that is DID-compliant and that implements ActivityPub using DIDComm as its communication protocol. Replacing the existing centralized identity management with a Self-Sovereign Identity approach through DIDs, and enabling a communication protocol that allows confidentiality, non-repudiation, authenticity, and integrity without being bound to a platform-specific security mechanism.

Black1.6 Outline

Overview of your thesis structure in this chapter.

Black2 Related Work

The following chapter covers the most important and significant concepts that are required to fully comprehend the approach of this thesis. This includes a closer look at decentralized communication protocols and identifier standards, as well as social networks that implement them. The revision and structuring of these concepts allows us to understand, build upon, and apply them in order to address our identified research questions.

Black2.1 ActivityPub

PResent first activityStreams to then go ahead and explain how ActivityPub works:

In 2018, Mastodon started using ActivityPub. A decentralized social network protocol based upon the Activity Streams 2.0 data format, which provides a data model for representing an "Activity" using a JSON-based syntax [sne17]. ActivityStreams also provides a vocabulary that includes all the common terms you need to represent all the activities and content flowing around a social network. ActivityPub can be described as decentralized, because it provides two different APIs. The first one provides endpoints to connect clients to servers, and the second one, used in federated implementations, includes a server-to-server API [LWTS+18]. After it was published as a Recommendation by the Social Web Working Group of the W3C in January 2018, Mastodon switched and pioneered the use of ActivityPub on a large scale. Promoting this way, its adoption.

Security and Privacy

Black2.2 Mastodon

In today's most popular social networks like Facebook, Twitter, or Youtube exists a centralized architecture that keeps millions of users in one single platform. Here, the control, decision-making, user data, and censorship depend on a single profit-driven organization. On the contrary, Mastodon is a decentralized microblogging social network created with the idea of bringing social networking back into the hands of its users. In the time when switching apps, services every few years and having friends on a dozen

different chat applications was a standard in terms of social networking, the German founder envisioned something that could put an end to this, and last forever [https://medium.com/we-distribute/one-mammoth-of-a-job-an-interview-with-eugen-rochko-of-mastodon-23b159d6796a]. Today, Mastodon is a network of more than 3,500 communities, each operated by different individuals and organizations that implement their own policies and code of conduct. By means of this, the user has the opportunity to choose whichever instance he finds most appropriate. Mastodon takes a big share in the Fediverse. A single interoperable ecosystem of different social networks that can communicate with each other. In other words, it is a collection of federated social networks running on free open software on thousands of servers across the world, that implement the same protocol in order to be able to interact with each other. The Fediverse is developed by a community of people around the globe that is independent of any corporation or official institution. It is not profit-driven, and it gives you the freedom of choosing which service and instance of the Fediverse to register to. On top of that, you are free to run your own instance with your own policies and allow other users to join [Los22]. The range of services that can be found inside the Fediverse includes blogging, microblogging, video streaming, photo, music sharing as well as file hosting. Since its standardization, ActivityPub has been widely adopted in the Fediverse, being today the dominant protocol.

Black2.2.1 Well-known Endpoints

[https://datatracker.ietf.org/doc/html/rfc8615] As per documentation, Mastodon implements the following 4 different well-known endpoints:

Black2.2.1.1 .wel-knowwn/change-password

This endpoint origins form the proposal of the Web Application Security Working Group (WebAppSec) from the W3C, whose main mission is to provide security and policy frameworks to increase web application security and enable safe cross-site communication [https://www.w3.org/groups/wg/]. It defines a well-known URL that sites can use to make their change password forms discoverable by tools. This URL would enable software, specially password managers, to easily redirect users to the right link for them to change their password.

Black2.2.1.2 .well-known/nodeinfo

NodeInfo is an initiative to standardize the presentation of metadata about a server operating one of the distributed social networks. The two main aims are to get greater insights into the distributed social networking user base and

to provide tools that allow users to select the most suited software and server for their requirements. Mastodon is one of the implementers of this protocol, along with other federated social networks such as Diaspora, Peertube and Wordpress [http://nodeinfo.diaspora.software/]. NodeInfo specifies that servers must provide the well-known path /.well-known/nodeinfo and provide a JRD document referencing the supported documents via Link elements, as shown in fig. (number). [http://nodeinfo.diaspora.software/protocol.html] Accessing the hypertext reference from the JRD response will give a schematized series of metadata of the instance running the endpoint, such as NodeInfo schema version, software, protocols supported by the server, statistics and even a list of third-party services that can interact with the server via an API. Figure (number) shows the NodeInfo 2.0 schema of the Mastodon instance mastodon.social.

'.well-known/host-meta': Host-Meta is lightweight metadata document format that allows for the identification of host policy or information, where "host" refers to the entity in charge of a collection of resources defined by URIs with a common URI host. It employs the XRD 1.0 document format, which offers a basic and flexible XML-based schema for resource description. Moreover, it provides two mechanisms for providing resource-specific information, specifically, Link templates and Link-based Resource Descriptor Documents (LRDD). On the one hand, link templates require a URI in order to be a functioning link, thus avoiding the use of fixed URIs. On the other hand, the "lrdd" relation type is used to relate LRDD documents to resources or host-meta documents [https://datatracker.ietf.org/doc/html/rfc6415]. In the specific case of the Mastodon implementation, as shown in fig. (number), accessing the host-meta endpoint will give us back the "lrdd" link to the Webfinger endpoint. In which specific resource information can be found.

GET /.well-known/host-meta HTTP/1.1 Host: mastodon.social Accept: application/xrd+xml

[language=XML, caption=Python example]

¡?xml version="1.0" encoding="UTF-8"?¿ ¡XRD xmlns="http://docs.oasis-open.org/ns/xri/xrd-1.0"¿ ¡Link rel="lrdd" template="https://mastodon.social/.well-known/webfinger?resource=uri"/¿ ¡/XRD¿

'.well-known/webfinger': Finally, Webfinger is the protocol in which Mastodon heavily relies for the discovery process and for its normal functioning [https://docs.joinmastodon.org]. Webfinger allows for discovering information about persons or other entities on the Internet using HTTP that would not otherwise be useful as a locator, such as a personal profile address, identity service, telephone number or an email. Performing a query to a WebFinger endpoint requires a query component with a resource parameter, which is the URI that identifies the identity that is being looked up. Mastodon employs the "acct" URI format [https://datatracker.ietf.org/doc/html/rfc7565 fn], which aims to offer a scheme

that generically identifies a user's account with a service provider without requiring a specific protocol to be used when interacting with the account. In the same way NodeInfo works, it returns a JRD Document describing the entity [https://datatracker.ietf.org/doc/html/rfc7033]. Fig. (number) shows an example of the returned JRD that is being provided by the WebFinger endpoint in the mastodon social instance when querying Bob's account "acct:bob@masto

[caption=HTTP request to Webfinger endpoint] GET /.well-known/webfinger?resourc HTTP/1.1 Host: mastodon.social Accept: application/xrd+xml

[language=Json, caption=Webfinger response]

"subject": "acct:bob@mastodon.social", "aliases": ["https://mastodon.social/@bob" "https://mastodon.social/users/bob"], "links": ["rel": "http://webfinger.net/rel/prof page", "type": "text/html", "href": "https://mastodon.social/@bob" , "rel": "self", "type": "application/activity+json", "href": "https://mastodon.social/users/bo , "rel": "http://ostatus.org/schema/1.0/subscribe", "template": "https://mastodon.so *uri*"]

Black2.2.2 Security

HTTP Signatures

JSON LD Signatures

Black2.3 Decentralized Identifiers

Globally unique identifiers are used by individuals, organizations, abstract entities, and even internet of things (IoT) devices for all kinds of different contexts. Nonetheless, the large majority of these globally unique identifiers are not under the entity's control. We rely on external authorities to issue them, allowing them to decide who or what they refer to and when they can be revoked. Their existence, validity scope and even the security mechanisms that protect them are all dependent on these external authorities. Leaving their actual owners helpless against any kind of threat or misuse. [?] In order to address this lack of control over an entity's identity, the W3C DID Working Group conceptualized the Decentralized Identifiers (DIDs).

DIDs are a new type of globally unique identifier that enables individuals and organizations to create their own identifiers using trustworthy systems. By means of this, entities are able to prove control over them by authenticating using cryptographic proofs. Furthermore, given that the generation and assertion of DIDs are entity-controlled, an entity can create any number of DIDs that can be tailored and confined to specific contexts. This would enable interaction with other systems, institutions, or entities that require authentication while also limiting the amount of personal or private data to

be revealed. All of this without the need to rely on a central authority [?].

To better illustrate how DIDs work, figure (number) provides a basic overview of the major components of the Decentralized Identifier architecture. Let's explain each component separately:

DID The DID itself is a URI, as defined by [RFC3986], that consists of 3 different parts:

did URI scheme identifier Identifier for the DID method. DID method-specific identifier

DID URL A DID can include path, query and fragment in order to be able to locate a specific resource inside a DID document, as shown in fig (number)

Example

DID-Subject Refers to the entity being identified by the DID. According to the specification, any person, group, organization, physical thing, digital thing or logical thing can be a DID-Subject.

DID-Controller The DID-controller is the entity that has the capability to make changes to the DID-Document. This entity is not necessarily the DID-Subject itself.

DID-Document They contain information associated with a DID. They usually describe verification methods, such as cryptographic public keys, as well as services that are relevant to interactions with the DID-subject. Figure (number) shows an example of a simple did-document.

"@context": "https://w3id.org/did/v1", "id": "did:example:123456789abcdefghi", "publicKey": ["id": "did:example:123456789abcdefghikeys-1", "type": "RsaVerificationKey2018", "owner": "did:example:123456789abcdefghi", "publicKeyPem": ""], "authentication": [// this key can be used to authenticate as DID ...9938 "type": "RsaSignatureAuthentication2018", "publicKey": "did:example:123456789abcdefghikeys-1"], "service": ["type": "ExampleService", "serviceEndpoint": "https://example.com/endpoint/8377464"]

Attributes The only required attribute for a DID-Document is the ID. Next, other optional attributes include a controller, which specifies the DID-controller; verification methods, which may be used to authenticate or authorize interactions with the DID subject or associated parties, such as authentication, assertionMethod and keyAgreement; and services, which can be any type of service the DID subject wants to advertise, including decentralized identity management services for further discovery, authentication, authorization, or interaction.

DID-methods

DID-methods describe the processes for CRUD operations for DIDs and DID documents, based on a specific type of verifiable data registry. In addition to implementation specifications, DID methods must describe security

and privacy considerations. According to the W3C registry, there are around 130 registered DID-Methods. Based on their characteristics and patterns, the following a 4-way classification of DIDs has arised.

Ledger-based DIDs This includes all the DIDs that store DIDs in a blockchain or other Distributed Ledger Technologies (DLTs). Examples include did:btcr, did:ethr and did:trx, whose DIDs are stored correspondingly in the Bitcoin, Ethereum and Tron network [https://ssimeetup.org/decentralized-identifiers-dids-fundamentals-identitybook-info-drummond-reed-markus-sabadello-webinar-46/].

Ledger Middleware ("Layer 2") DIDs

Broadly speaking, Layer 2 refers to a framework or protocol that is built on top of an existing blockchain system that takes the transactional burden away from layer 1 and post finalized proofs back to layer 1. By removing this transaction load from layer 1, the base layer becomes less congested, and everything becomes more scalable [https://ethereum.org/en/layer-2/]. The Lightning Network (LN) is an example of a layer 2 Bitcoin protocol that offers users a fast micro-payment platform. Furthermore, regarding Layer 2 DID methods, the DIF has developed Sidetree [https://github.com/decentralized-identity/sidetree]. A protocol for creating scalable blockchain-agnostic Decentralized Identifier networks that enables users to create globally unique, user-controlled identifiers and manage their associated PKI metadata, without the need for centralized authorities or trusted third parties. [https://identity.foundation/sid] Examples include: did:ion, did: elem.

Peer DIDs DIDs have the required ability to be resolvable, however not all of them have to be globally resolvable. The DIDs in this category do not exist on a global source of truth, but in a context of relationships between peers in a limited number of participants. Nonetheless, their validity is not affected due to the fact that they comply with the core properties and functionalities that a DID has to provide.

Static DIDs This type of DIDs are limited in the kind of operations that can be performed to them. These DIDs are not stored in any registry, consequently, it is not possible to update, deactivate or rotate them. Using the did:key method as an example, the DID-method-specific part of the DID is encoded in a way that the DID document can be deterministically extracted from the DID itself. [https://w3c-ccg.github.io/did-method-key/]

Below there are some examples listed, however the full list can be found here (https://w3c.github.io/did-spec-registries/did-methods). did:ethr: This DID-method was registered by Uport, an Ethereum-based system for self sovereign identity (https://whitepaper.uport.me/uPort$_{w}hitepaper_{D}RAFT20170221.pdf).Itd$
$//github.com/decentralized-identity/ethr-did-resolver/blob/master/doc/did-method-spec.md).uPortisasmart-contract-basedsystemthatabstractsuseraccountsusingproxyco$

*//ieeexplore.ieee.org/document/*8783188/]

did:ens: The Ethereum Name Service (ENS) is a distributed, open, and extensible naming system based on the Ethereum blockchain. Its purpose is to map names like 'bob.eth' into machine-readable identifiers like Ethereum addresses, other cryptocurrency addresses, content hashes, and metadata. In the same way the Domain Name System (DNS) maps human-readable domains like 'google.com' to IP addresses. However, their architecture differs due to the capabilities and constraints of the Ethereum blockchain. (https://github.com/veramolab ens-spec) This DID-Method was registered by Consensys, a blockchain venture studio founded by Ethereum's cofounder Joseph Lubin that supports Ethereum-based projects. Some of these projects include uPort, Metamask, Civil, Truffle and Gnosis. [https://decrypt.co/resources/consensys].

This DID-method was created with the following objectives, firstly to integrate ENS names to be interoperable with applications that implement DIDs, and secondly, to add DID-capabilities to ENS names such as services and verification methods [https://github.com/veramolabs/did-ens-spec].

did:ion: This DID-method works by registering DIDs to the Bitcoin network. Launched to the public in March 2021, Identity Overlay Network (ION) is Microsoft's DID-network built on top of the Bitcoin Network implemented using Sidetree (https://techcommunity.microsoft.com/t5/identity-standards-blog/ion-we-have-liftoff/ba-p/1441555). ION is public, permissionless, it does not implement its own currency and its nodes do not require an additional consensus mechanism. (https://github.com/decentralized-identity/ion).

Verifiable Data Registry Essentially, any system that enables capturing DIDs and returning required data to generate DID documents. This can be distributed ledgers, decentralized file systems, any type of database, peer-to-peer networks, or other types of trustworthy data storage [?].

DID resolver A DID-resolver is able to implement the DID-resolution, which consists of taking a DID as an input, and giving a DID Document as an output [?]. As of the writing of this thesis, the Identifiers Discovery Working Group (ID WG) has implemented a non-production-ready Universal Resolver, which allows the resolution of DIDs for numerous DID-methods [https://github.com/decentralized-identity/universal-resolver], including all the examples mentioned above. In addition, this working group has also developed a Universal Registrar, which allows the creation, edition and deactivation of the DIDs across different DID-methods. Working instances of these projects can be found here. [https://uniresolver.io/ , https://uniregistrar.io/].

Comparing DIDs with other identifiers

UUID Universally Unique Identifiers: UUIDs are similar to DIDs, because they do not require a centralized registration authority. But they are not

resolvable or cryptographically-verifiable.

Almost all types of identifier systems can support DIDs. This would create an interoperability bridge between centralized, federated and decentralized worlds. Implementers can use any kind of computing infrastructure they trust, such as distributed ledgers, decentralized file systems, distributed databases or peer-to-peer networks.

Services Services are used to specify ways to communicate with the DID subject. Services can be decentralized identity management for further discovery, authentication, authorization or interaction.

DID Privacy

The 7 Principles of Privacy by design have been applied to DIDs. (https://iapp.org/me

Keep personal Data private: DID documents do not include personal information. Data can be transmitted using Verifiable Credentials, or service endpoints under control of the DID subject / controller. Putting a username in an URL is dangerous, because the username is likely to be human-meaningful in a way that can reveal information that the DID subject did not consent to sharing. (Using Mastodons username here is an argument). NO PERSONAL DATA IS TO BE WRITTEN TO AN IMMUTABLE DISTRIBUTED LEDGER. DID CORRELATION RISKS. This is relevant DID Subject Classification: It is dangerous to add properties to the DID document that can be used to indicate, explicitly or through inference, what type or nature of thing the DID subject is, particularly if the DID subject is a person. Service Privacy: The more service endpoints a DID Document has, the more risk in correlation it has. First, because service endpoints might be specified as URIs, they could unintentionally leak personal information because of the architecture of the service. For example, a service endpoint of http://example.com/MyFirstName is leaking the term MyFirstName to everyone who can access the DID document. When linking to legacy systems, this is an unavoidable risk, and care is expected to be taken in such cases. This specification encourages new, DID-aware endpoints to use nothing more than the DID itself for any identification necessary. For example, if a service description were to include http://example.com/did

A DID is unlikely to contain any information about the DID subject, so further information about the DID subject is only discoverable by resolving the DID to the DID document, obtaining a verifiable credential about the DID, or via some other description of the DID.

Black2.4 DIDComm Messaging

Definition The Hyperledger Foundation is an open-source collaborative effort intended to further develop blockchain technologies across industries [https://wiki.hyper

]. Started in 2016 by the Linux Foundation, it has given birth to numerous enterprise-grade software open-source projects that can be classified into DLTs, libraries, tools and labs [https://www.hyperledger.org/wp-content/uploads/2021/11/HL$_{Pap}$
$basedidentitymanagement, includingkeymanagement, credentialmanagement, andanencrypted, peer-to-peerDID-basedmessagingsystemthatisnowlabeledasDidcommv1.[https://wiki.hyperledger.org/display/A$
$Aries].BasedonDidcommv1, theCommunicationWorkingGroup(CWG)oftheDIFhasimplementedDIDcommv$
$DIDsinordertomakeallDIDsequalfromtheperspectiveoftheDIDCommspec.Furthermore, theCWGpursuesth$
$basedprojectsbuttocreateaninteroperablelayerthatwouldallowhigher-orderprotocolstobuilduponitssecurity, pri$
$//medium.com/decentralized-identity/understanding-didcomm-14da547ca36b][https://didcomm.org/book/v2/whatsnew][https://identity.foundation/didcomm-messaging/spec]$

From this point on, we are going to refer to DIDComm Messaging v2 only as DIDComm. Didcomm can be described as a communication protocol that promises a secure and private methodology that builds on top of the decentralized design of DIDs [https://identity.foundation/didcomm-messaging/spec/overview]. Moreover, Didcomm is a very versatile protocol, as it supports a wide range of features, such as security, privacy, decentralization, routable, interoperability, and the ability to be transport-agnostic.

To better understand how it works, let's look at how it would work in a scenario where Alice wants to send a private message to Bob:

[https://arxiv.org/pdf/2006.02456.pdf]

DIDComm differs from the current dominant web paradigm, where something as simple as an API call requires an almost immediate response through the same channel from the receiving end. This duplex request-response interaction is, however, not always possible as many agents may not have a constant network connection, or may interact only in larger time frames, or may even not listen over the same channel where the original message was sent. DIDComm's paradigm is asynchronous and simplex. Thus showing a bigger resemblance with the email paradigm. Furthermore, the web paradigm goes under the assumption that traditional methods for processes like authentication, session management, and end-to-end encryption are being used. Didcomm does not require certificates from external parties to establish trust, nor does it require constant connections for end-to-end transport-level encryption (TLS). Taking the security and privacy responsibility away from institutions and placing it with the agents. All of this without limiting the communication possibilities because of its ability to function as a base layer where opposed capabilities like sessions and synchronous interactions can be built upon. [https://identity.foundation/didcomm-messaging/spec/]

To achieve the encryption and signing processes of figure (Algorithm), Didcomm implements a family of the Internet Engineering Task Force (IETF) standards, collectively called JSON Object Signing and Encryption (JOSE), which will be further explained in the next section.

JOSE Family When were they created? This RFC family includes both the

Json Web Signature (JWS) and the Json Web Encryption (JWE) standards that are subclasses of the Json Web Token (JWT), and Json Web Key (JWK).

Compare with other encryption methods. (TLS, Whatsapp end-to-end, Signal) (Use image below)

Black3 Concept and Design

The standards presented in Chapter 2 show the potential improvements that can be achieved in key components of ActiviyPub-based social networks such as identity management, discovery, and communication. In this section, we are going to go through the different steps that are required firstly to integrate DIDs into an ActivityPub-based social network, and finally to enable DIDComm Messaging v2 for its communication. The outline for this chapter is the following. First, individual concepts and definitions of the ActivityPub implementation are introduced and described. Then, an example of a simple use-case in a contemporary ActivityPub server is going to be illustrated and analyzed in order to be able to compare it with section 4.5, which presents the same use-case but with the proposed concept and design that includes DID integration and DIDComm enablement. Consequently, the reasons behind every decision made are going to be detailed in section 4.6, along with comparisons with other not-formalized proposals.

As explained in Chapter 2, Mastodon is the social network that pioneered the use of ActivityPub on a large scale, and it is also the social network with the most active users and presence inside the Fediverse. For this reason, the ideas presented in this section and the modus operandi of the ActivityPub server are going to be scoped to the actual implementation in Mastodon.

Definitions Mastodon has implemented its own ActivityPub server, and with it also its own terms to express different social network vocabulary. In order to prevent confusion or ambiguities, the used terms in this chapter are explained here.

Username The username in Mastodon consists of a unique local username and the domain of the instance. Ex. alice@example.com

Actor's object In this section, the term Actor's object refers solely to the ActivityPub's actor object explained in Chapter 2.

Toot In the user-facing part of Mastodon, a Toot is the äquivalent of a Tweet in Twitter. This is a small status update with a 500 character limit.

Status In the backend of Mastodon, the descriptor used for a Toot is a Status. Moreover, an account in Mastodon has a 1:n relationship with status.

In order to explain the current ActivityPub flow in Mastodon, let's describe what happens in the simple use case: Alice has an account in the Mastodon

instance mastodon.social and follows Bob, who has an account in the Mastodon instance mastodontech.de. Alice sends a direct message to Bob with the text: "Mastodon is amazing, isn't it?"

Black3.1 Today's implementation

[BPMN with normal Flow]

Object creation The first thing that happens when Alice presses the send button is the creation of an ActivityStreams object. In this case, the object is of type "Note", and will be created by the ActivityPub server inside the Mastodon instance, as shown in fig (number). Then, following the ActivityPub pattern of "some activity by some actor being taken on some object", the server recognizes that this is a newly created object and wraps it in a "Create" activity that looks like fig. (number). Now that the actor, the activity, and the object are well defined and wrapped, it is time to shift our focus to the recipients of this note object. The ActivityPub server will now look at all the fields of the ActivityStreams Audience, which includes: to, bto, cc, bcc, and audience. From these fields, all the addresses of the recipients can be retrieved. Afterwards, depending on where the recipient's account lives, the ActivityPub server may take one of two options. Even though the use-case explicitly dictates that Bob's account resides in a different Mastodon instance, both cases are still to be explained.

Same-server delivery If the recipient's account is on the same server, there is then no explicit discovery process. A simple query in the ActivityPub's server would find the right account and save the status within the account's statuses.

Discovery On the contrary, when the recipient's account is not in the same server, then a discovery process must be started. Discovery is the fundamental part of the federated side of Mastodon. Without it, users within different instances would not be able to interact, as the instance itself does not know where to find the actor object with all required endpoints to send or receive activities from and to external accounts. For this reason, the current way to look up for other accounts is through the DNS. In the same way Email works, the domain part of the username in Mastodon points to the domain of the instance where the account lives.

The purpose of the discovery in this specific use-case is to find the inbox url of Bob, which can can be found in Bob's actor object. As explained in chapter 2, Mastodon includes a series of well-known endpoints that are used to retrieve information about the host itself, resources or entities that are managed under the same host. As well as other operations, such a change-password endpoint. By default, when the account is not found on the same server, it will trigger a job that will run the ResolveAccountService

class. This class will then look for the specific account using the Webfinger endpoint of the domain part of the username. In our case, Bob's account lives inside the mastodontech.de server. The request shown in fig. (Webfinger request) will return a JRD Document, as shown in fig. (JRD document from mastodontech.de). Based on this document, the Mastodon instance retrieves the link with the rel: 'self' that includes the type and the href to which the Bob's actor object can be retrieved. If the Webfinger request returns a 404 code, it will then try, as a fallback, using the Host-Meta endpoint. The request and response are displayed in fig. (hostmeta request, response). If successful, it will then proceed to take the Link template provided and try the Webfinger request once more before throwing an error. After retrieving the needed URL, a subsequent HTTP GET request to this endpoint with the specific "application/activity+json" header will resolve Bob's actor object.

Delivery Succeeding the retrieval of Bob's actor object and therefore the needed inbox URL, a delivery can now take place. In this case, an HTTP POST request with the previously generated activity object. In order to provide end-to-end message integrity and to authenticate Alice in Bob's server, the request is signed by Alice's ActivityPub server using the HTTP Signature specification. Finally, upon receiving the POST request to Bob's inbox URL, Bob's server verifies the validity of the signature using Alice's public key. After a successful validation, it saves the Note object in Bob's statuses.

As indicated in chapter 2, HTTP signatures are not part of the ActivityPub protocol standard. These integrity and authentication features are within the Mastodon implementation of an ActivityPub server..

Black3.2 Proposed implementation

Proposed implementation Having seen the current flow of our use case mentioned above in a working ActivityPub-based social network, this section will now address the first research question of this bachelor thesis. That is, the implications of integrating DIDs to a Mastodon instance, and therefore to ActivityPub. Implications of integrating DIDs As mentioned in the definitions section, Mastodon's full username includes the domain of the server, and a locally-unique username. This type of username accomplishes the goals of human-readability and uniqueness. In addition, they are resolvable using DNS and the discovery methods previously mentioned in section (number). An ActivityPub actor object using such a username is shown in .

Even though there are cases when very similar domains may lead to confusion and may present some security issues. Examples of this can be found simply by comparing the most used Mastodon instance "mastodon.social" with another big-in-user-count instance such as "mstdn.social". This type of

username allows the following scenarios to play out, where differences are minimal and easily missed out, such as alice@mastodon.social and alice@mstdn.social. In addition, some users have commented that when the username is too long, the domain is not longer visible. This can be exploited by malicious actors, who may open an account in a different server with the same username as their target. Clone their profile, and try to social engineer the target's followers.

[language=json, caption=Alice's actor objec, label=actor object]

**"@context": ["https://www.w3.org/ns/activitystreams", "https://w3id.org/security
], "id": "http://lisztos.com/users/alice", "type": "Person", "following":
"http://lisztos.com/users/alice/following", "followers": "http://lisztos.com/users/alice
"inbox": "http://lisztos.com/users/alice/inbox", "outbox": "http://lisztos.com/users/
"featured": "http://lisztos.com/users/alice/collections/featured", "featuredTags":
"http://lisztos.com/users/alice/collections/tags", "preferredUsername": "alice",
"name": "", "summary": "", "url": "http://lisztos.com/@alice", "manuallyAp-
provesFollowers": false, "discoverable": false, "published": "2022-06-14T00:00:00Z",**

The first question that needs to be addressed when approaching the integration of DIDs to Mastodon and therefore ActivityPub is, what are the implications of switching from standard mastodon usernames to DIDs. Integrating DID to ActivityPub points immediately to the actor's object. Making the switch would mean that the DID has to be included. Currently, most of the interactions of the ActivityPub server inside Mastodon require the ID attribute to resolve to the Actor's profile and thus the Actor's object. Following a simple strategie, we could simply replace the username with the DID. Thus having an ID attribute that may look like this:

id: www.instance$_m a.com/users/did : example : 123456789abcdefghi$

However, there is another alternative that might work. Following the ActivityPub's specification, the ID attribute must be a publicly dereferencable URI, whose authority belongs to the originating server [?]. As explained in section(DID number), a DID is a URI and it is publicly dereferencable by nature. This allows different possibilities, in which the DID can be added. Take for example, using a stand-alone DID as an ID attribute, in order to take advantage of the discoverability of DIDs. This scenario would have the following implications. If another ActivityPub Server wanted to simply get the actor's profile URL, it would require resolving the DID to its respective DID-Document, adding the need to parse it and to methodically do a search until the endpoint is found and retrieved. This additionally requires that the DID-Document includes the actor's profile in the services section. This modification in the DID Document will be further discussed in the Discovery section. Moreover, another option would be to add a DID-URL with a query that points directly to the service endpoint that contains

the actor's profile URL. Simplifying in this manner the work that the ActivityPub server must do, although still requiring a DID resolution as an intermediate step, as well as adding the service endpoint to the DID-Document. Both cases are possible, because ActivityPub has the URL attribute that requires the actor's profile URL in case that it is not in the ID attribute [?]. Although plausible, for the purpose of this Thesis we will keep a simple replacing strategie and keep the ID attribute as the profile's URL with the username replaced with the DID.

DID URLs provide a lot of freedom of usability for DIDs. In addition to the ID, the actor's object must provide a supplementary set of URLs that point to different collections related to the Actor. These include mandatory attributes, like the outbox and the inbox, and other optional attributes such as the followers and following attributes. What if, instead of using the actual URL of these collections, we specify DID URLs that then point to the correct endpoint inside the DID-Document. This example is illustrated in fig. (Activitypub actor with all DID URLS).

This approach leads us to the question, isn't it simpler just to shift the whole actor's object endpoints directly to the DID-Document? Therefore removing the need for an ActivityPub actor's object. This idea was briefly suggested by [?] in a paper prepared for the 2017 Rebooting Web of Trust summit. Such an idea would look like (fig. DID-Document with all ActivityPub endpoints). Furthermore, the authors went a step further and proposed cutting all dependency from the DNS by using onion websites. The authors however failed to offer an adequate explanation and consideration of the ActivityPub protocol in existing implementers. Moreover, there are some security privacy concerns regarding the use of service endpoints in the DID Document. The DID specification stipulates "revealing public information through services, such as social media accounts, personal websites, and email addresses, is discouraged" [?]. DID Documents are stored in a publicly available verifiable data registry, therefore any personal information revealed here is for everyone to see. The usage of URLs in service endpoints might lead to involuntary leakage of personal information, or correlation between the URL and the DID subject. Looking at Fig(fig. DID-Document with all ActivityPub endpoints), the amount of personal information displayed in the DID Document, which would not be otherwise inferable, already poses a privacy issue for the DID Subject.

For this reason, in this thesis we differ from removing the actor's object from the ActivityPub protocol itself. This would also allow us to use freely all the other attributes in the actor's object to further describe its owner, such as name, preferredUsername, or summary without making this information forever public in an immutable ledger. Fig(final Actor'S object) illustrates the final design for the DID-compliant actor object.

Regarding Mastodon, replacing the standard username with a DID does not imply huge complications. When creating a username there are some validations made to it, such as length, regex and uniqueness. All of this is contained inside the Account class, which is the object that abstracts all a user's account. An example of the Mastodon instance that uses DIDs instead of standard usernames is illustrated in fig. (Image of Mastodon with DIDs).

Discovery As stated in the previous section, Mastodon starts the discovery process based on the username of the user. By replacing the standard username with a DID, the current discovery flow gets disrupted, as there is no domain and thus no well-known endpoints to send discovery requests to. Nonetheless, here is where the discoverability and always resolvability of the DIDs comes into play. The proposed flow of discovery takes the following steps. Firstly, the username, which now is a DID, can be resolved to its DID-Document using any kind of DID resolver. An example can be the Universal Resolver from the DIF mentioned in Chapter 2. The DID Document must now contain a service section with the type ActivityPub and a URL, where the actor's object can be retrieved. This gives us 2 possibilities. One the one hand, we could add in the well-known Webfinger endpoint, which then provides us with the profile URL from the user. On the other hand, we could skip the Webfinger request and provide the profile URL directly in the DID-Document. The latter looks like the most meaningful path to take. Especially when we refer to the purpose of Webfinger, which was to enable discoverability of entities represented by URIs [https://webfinger.net/]. Webfinger's purpose shares a lot of ground with the discoverable design of DIDs, nevertheless the DID design provides a less limited structure of discovery, as it does not rely on DNS and HTTP for its functioning. For this reason, the proposed workflow will completely remove the use of the discovery protocol Webfinger used in Mastodon and include the URL of the actor's profile in the ActivityPub service section in the DID Document.

Enabling DIDComm Messaging Considering the fact that DIDs are now implemented in our ActivityPub prototype, it is now possible to enable the DIDComm messaging protocol.

Extra maybe useful information

The proposed way to share personal information, according to the DID specification, is to either use service endpoints that are in control of the DID controller or the DID subject, or verifiable credentials.

Black4 Implementation

Black5 Evaluation

The evaluation of the thesis should be described in this chapter

Black6 Conclusion

Describe what you did here

BlackList of Tables

BlackList of Figures

[LE,RO]0cm [LO,RE]0cm [RE]2cm inner=2cm, outer=2cm, bottom=4cm

BlackAppendix 1

language=PHP **for(**$i = 1$;**i¡123;** $i++$)$echo$"$workharder$!;)";