dm3 - encrypted and decentralized web3 messaging

Abstract

...

Motivation

Messaging (such as instant massages, chats, email, etc.) has become an integral part of most people's lives. Mobile devices (such as smartphones, tablets, laptops, etc.) with instant access to the Internet make it possible to be in touch with family, friends, as well as work colleagues and customers at any time.

Email

In the early days of the Internet, email became a critical killer application. Communication was (then as unfortunately still often today) unencrypted. But the email protocols were designed in such a way that sufficient decentralization was possible, since each user could decide for himself which provider to use or even whether to host the service himself.

Besides the weak point of missing encryption (which even PGP could not solve due to lack of application), spam became more and more a big problem. Today, spam accounts for the largest share of email traffic (60-90%, source: www.verbraucherzentrale.de). This has led to spam filters on the server side (at the service provider) as well as on the client side to filter out a large part of the spam or to prevent it from being transmitted in the first place.

Another consequence of this is that large email providers only accept emails from a few other large providers, which means that it is now almost impossible to host an email service yourself, as the mails sent in this way are not accepted by the spam filters of the large providers (see https://lmy.de/GgB0J).

SMS

After 1990 SMS was developed, which made it possible to send short text messages between especially cell phones (device-2-device communication). Despite the rather high price and the drastic limitation of the length (160 characters), the short messages became very popular. Short messages are transmitted unencrypted from the sender's device to the service provider and from there to the recipient. End-to-end encryption is not provided.

Web2 Messengers

In the late 1990s, instant messaging services emerged that made it possible to send direct messages from computer to computer. With the breakthrough of the iPhone and other smartphones at the end of the 2000s and messagers such as WhatsApp, WeChat, Facebook Messenger, Signal, Telegram and many others, these largely replaced SMS, not least because it was possible to send short messages free of charge.

Even though today most messengers promise end-to-end encryption for message transmission, in some cases the encryption may be lifted, e.g. when messages are to be checked for offensive content based on a report of suspicion. Although most messengers today have a very similar feature set, cross-platform

communication is not possible because each messenger maintains its own community and keeps data in its own silos. Interoperability is currently not possible, although this would be very desirable from the user's point of view. Legislative initiatives such as the planned "Digital Markets Act" of the EU (see https://lmy.de/XGqOR) as well as various national drafts aim to make such interactions between users of the different platforms possible in the future. The challenge, however, is that the various solutions have each developed their own protocols (with a different focus on security and privacy, for example) and store user data in their individual data centers.

What these solutions also have in common is that they are centralized services. Communication always takes place over the servers of the respective service, just as the data (e.g., chat histories, but also metadata such as connection data) is managed by the service providers. In some cases, users have to make far-reaching concessions regarding their sovereignty (e.g., sharing of contact information, ...).

Often a certain messenger has become established as a quasi-standard in a certain region or user group (e.g. WhatsApp in Europe, WeChat in China, Facebook Messenger in U.S., ...). This then leads to the fact that one must also be registered with this service in order to communicate with friends and acquaintances, since they often use the most common messenger and rarely select the messenger based on its technical performance but rather on the network effect in their circle of acquaintances.

`[PIC DATA-SILOS]`

User administration is carried out as usual with username and password (sometimes including 2FA), whereby the telephone number is often used as user identification. However, this also means that the user profiles are under the control of the service providers.

Web3 Approaches

dm3 Messageing

The aim of dm3 is to define a messaging protocol which, based on web3 technologies, enables communication to be fully end-to-end encrypted, sufficiently decentralized, and self-determined under the control of the users.

The user must have full control over his data at any time (chat histories, including media files, contact information, ...). He/she should also be free to decide which delivery service to use.

Interoparability is a central goal of the protocol. Users of different messaging applications can communicate with each other, regardless of which application they use, as long as the dm3 protocol is implemented.

`[PIC INTEROPERABILITY]`

Registry

In order to implement end-to-end encryption, the sender of the message must know the recipient's public key to encrypt the message. A registry accessible by all users of the protocol is needed to make the public keys as well as the URL of the delivery service to be used accessible.

With ENS (Ethereum Name Service, see https://ens.domains), a decentralized service is available that can manage other records in addition to the assignment of a name (such as dm3.eth) to an address (e.g.

Ethereum address). The dm3 protocoll uses a text record "eth.dm3.profile" as reference to the dm3 profile of an ENS domain.

The information can be referenced to as

- an IPFS address or
- a link to a cloud service plus hash of the content.

In case of using an IPFS address the Delivery Service itself is responsible for pinning the record.

The referenced profile contains the entries:

- **publicMessagingKey** - public key to encrypt the message.
- **publicSigningKey** - public key for signature verification.
- **deliveryService** - the ENS domain of the delivery service where the user's messages will be sent to. The delivery service's ENS domain has a text record "eth.dm3.deliveryservice" defined which contains the delivery URL or optionally a list of multiple URLs with fallbacks, if the primary delivery service is not reachable. Also, the optional URL to read the mutable configuration settings (like spam protection settings, ...) is given.
- the **signature** (with the signingKey) for the content above.

Since the setting of this text record is done on-chain, a layer-1 transaction is required for this. Furthermore, in this case it is assumed that the recipient has its own ENS domain to which a dm3 profile is assigned. However, not everyone has his/her own ENS domain or is willing to pay for this transaction. Then it is possible that he/she registers a subdomain of dm3 (e.g. *unique_name.dm3.eth*). The administration of the subdomains is done on layer-2.

Similarly, other messaging applications that support the dm3 protocol can easily implement the registry for their users (if they are not using a direct ENS profile) by resolving them through subdomains of their ENS domain (e.g. *name_or_telephonenumber.messengerXY.eth*).

But even users who do not want to have an ENS domain at all can be included via a special subdomain: *ethaddress.dm3.eth*)

Delivery Service

Communication in dm3 is strictly peer-2-peer. A message is sent directly from the sender to the delivery service of the recipient. The content of the message is already encrypted on the client side at the sender with the recipient's public key. The entire message packet (including meta data) is additionally encrypted with the public key of the delivery service. The delivery service stores the message until the recipient picks it up or the maximum storage time has expired. Since the message is encrypted and signed by the sender, the delivery service cannot evaluate or modify the content at any time.

`[PIC FLOW/ENCRYPTION]`

As soon as the recipient has picked up a message and confirmed its receipt and saving in the storage, the delivery service deletes it.

Each delivery service also has an ENS domain. With the text record "eth.dm3.deliveryService" the reference to the profile of the delivery service is provided.

The information can be referenced to as

- an IPFS address or
- a link to a cloud service plus hash of the content.

In case of using an IPFS address the Delivery Service itself is responsible for pinning the record.

The referenced delivery service profile contains the entries:

- **deliveryServiceURL** - the URL to the delivery service
- **publicKey** - the public encryption key of the delivery service. All packages sent to the delivery service are encrypted with this key.

Peer-2-Peer Messaging

Group-Chat

dm3 Storage

Local File Store

Private Cloud

Cloud

IPFS/Web3Storage

Keys

Derivation

Message Encryption

Storage Encryption

Signatures

dm3 Messenger

Message Flow

Notification

Spam protection

User Interface

Web Application

Mobile App

Widget

Interoperatibility

Web3 Messaging Multiverse

Potential for Web2 Messengers

dm3 Token