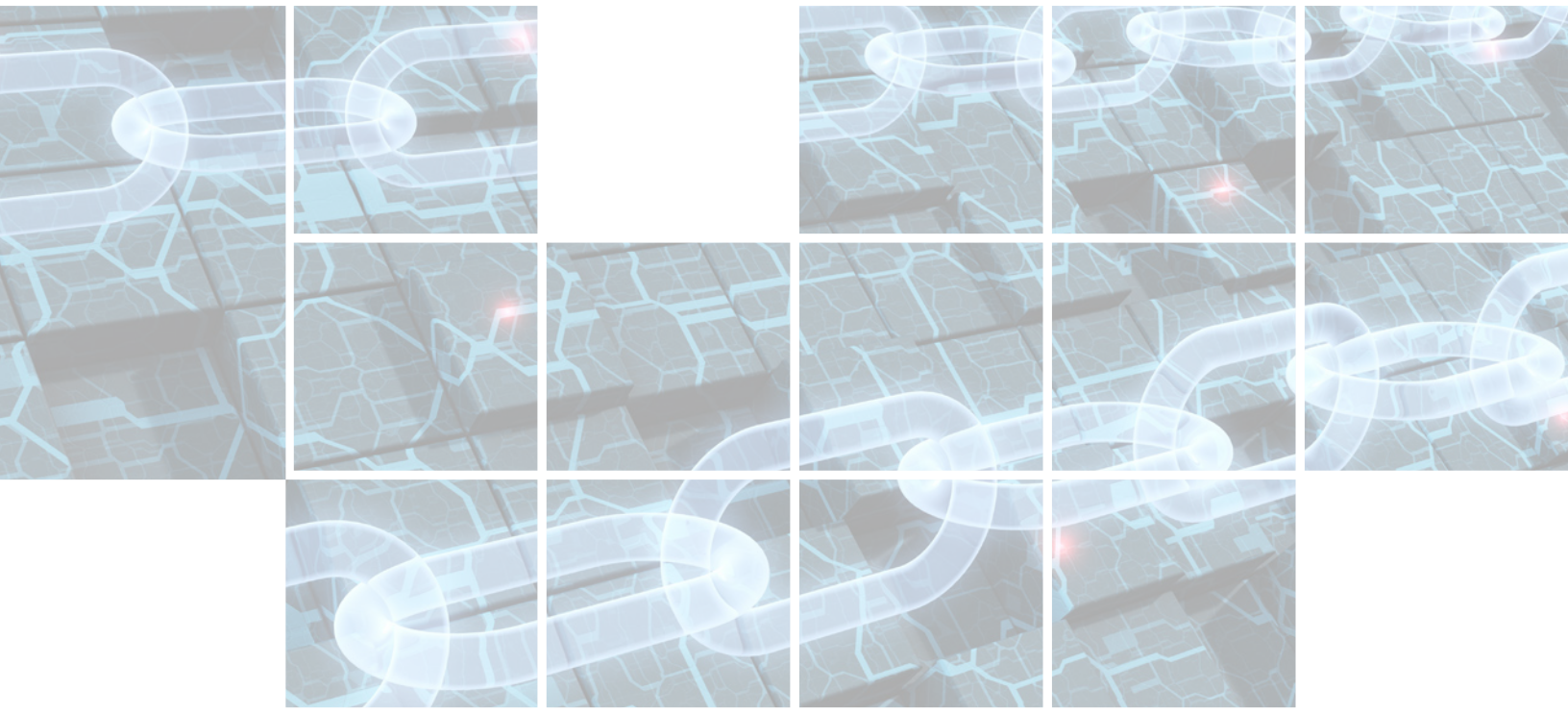
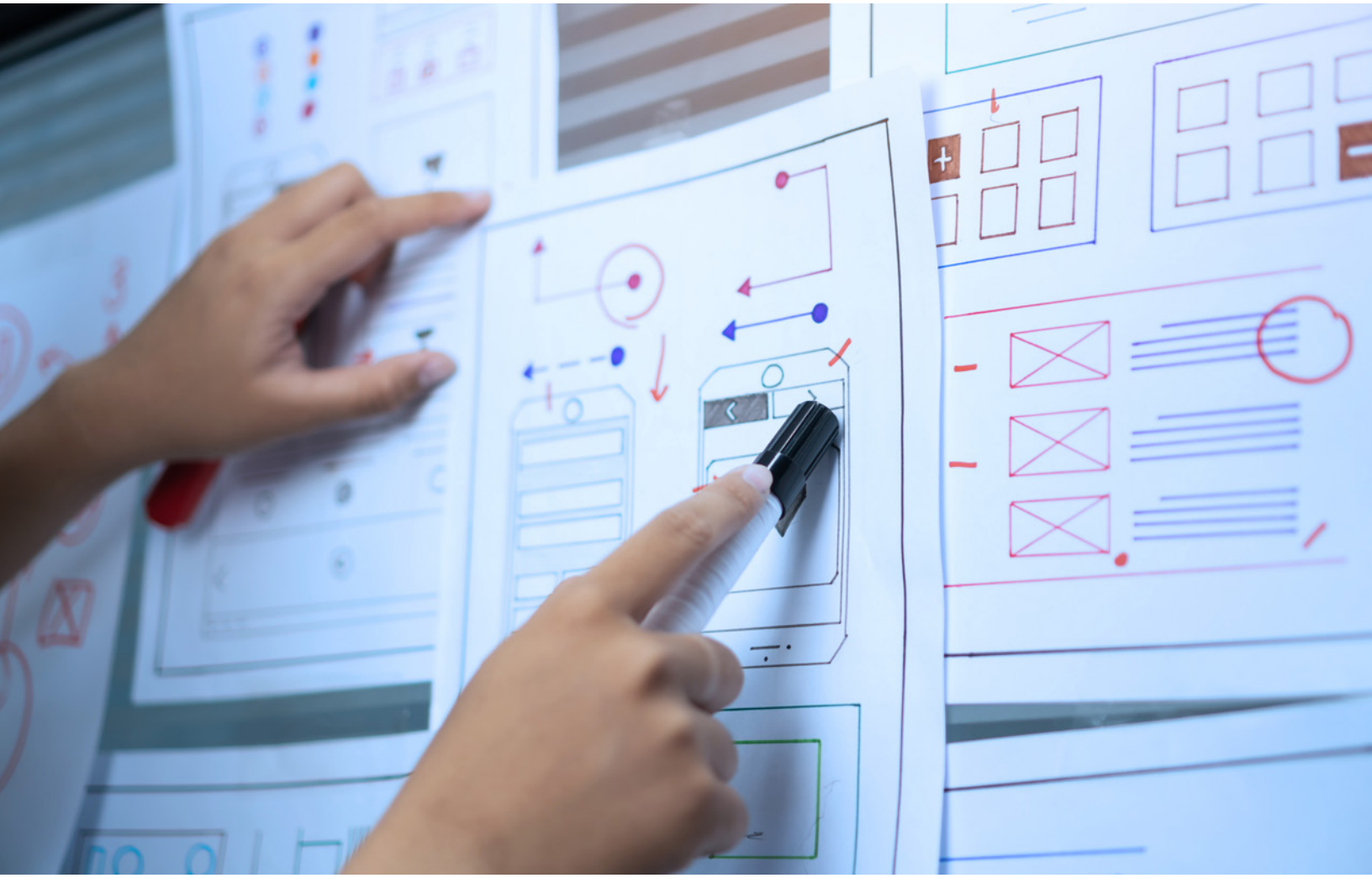
ANONYOME LABS



# Securing

## Private Communications with DIDComm

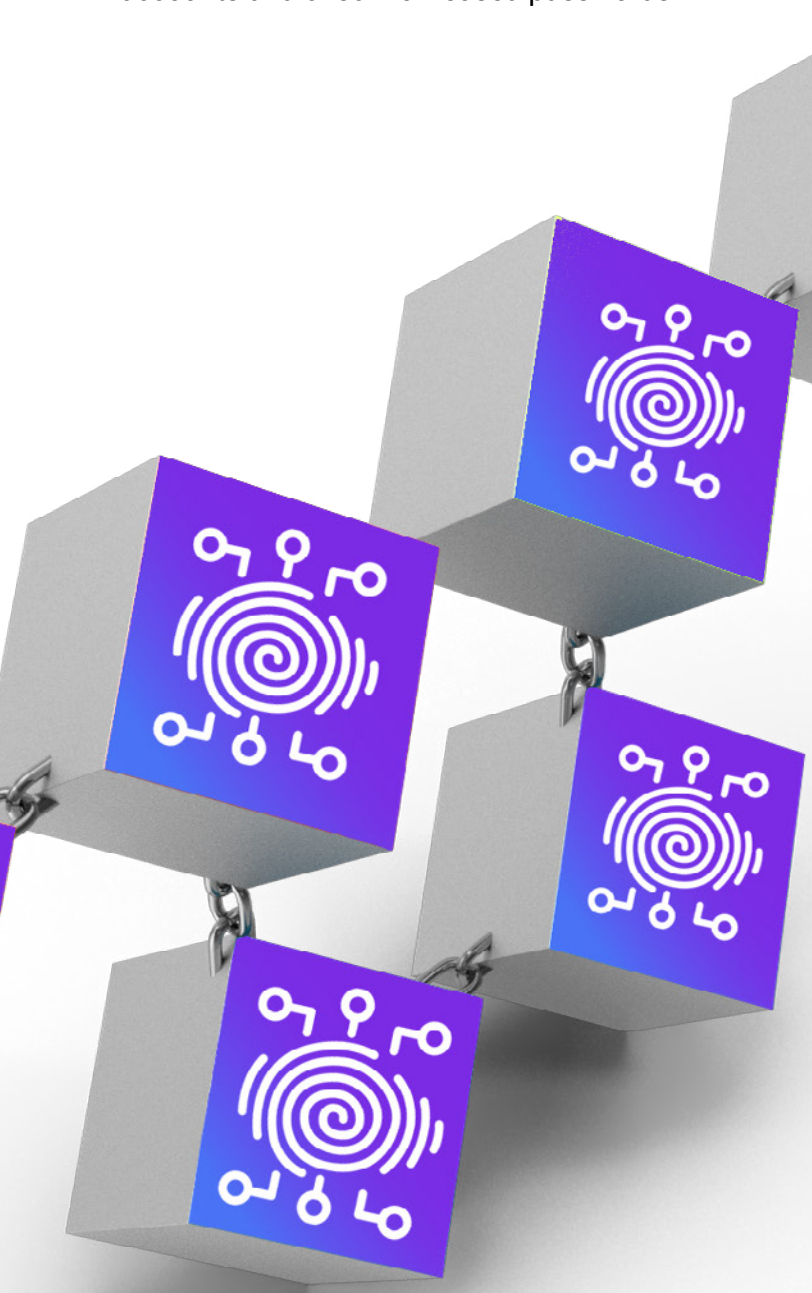


**Building an app?** Then you will likely be adding digital communications—and will need to secure them. Secure digital communications are used in everything (or should be) across all sorts of areas, such as healthcare, employment, socialization, shopping, education, government, gaming, and Internet of Things (IoT). These systems enhance our lives in countless ways—but how good are they at protecting our private information?

During the first half of 2022, over 800 reported compromises of personal data occurred. In 2021, nearly six billion accounts were breached. These breaches resulted from account compromises at mainstream organizations, such as Facebook, LinkedIn, and T-Mobile, as well as many financial and government organizations. Incident after action reports often recommend consumer-side solutions, such as changing passwords, using unique passwords, adding two-factor authentication, increasing to multi-factor authentication, or accepting a settlement of a few months of credit monitoring, etc. Given the many accounts the average consumer has, this is a significant burden on non-technical consumers.

# Your online identity: what is it really?

On today's internet, your identity is usually just a reference to an anchor account, such as your email or social media. Originally, an email address was for communication, but now it's used as your main website login name, because it's unique. This works great—right up until you want to change email providers or hackers want to correlate your accounts and check for reused passwords.



# A new type of secure identity is here

In recent years, a new type of secure identity has emerged and is built on very strong cryptographic processes. Known as decentralized identity (DI) or self-sovereign identity (SSI), this new identity overcomes many of the security problems associated with login name and password combinations. Also, despite its complex cryptographic foundations, DI is even easier for users to operate than current login methods. DI also goes beyond website logins and facilitates other modern digital features, such as verifiable credentials and transport agnostic encrypted communication protocols.

# Decentralized identifiers or DIDs make it possible

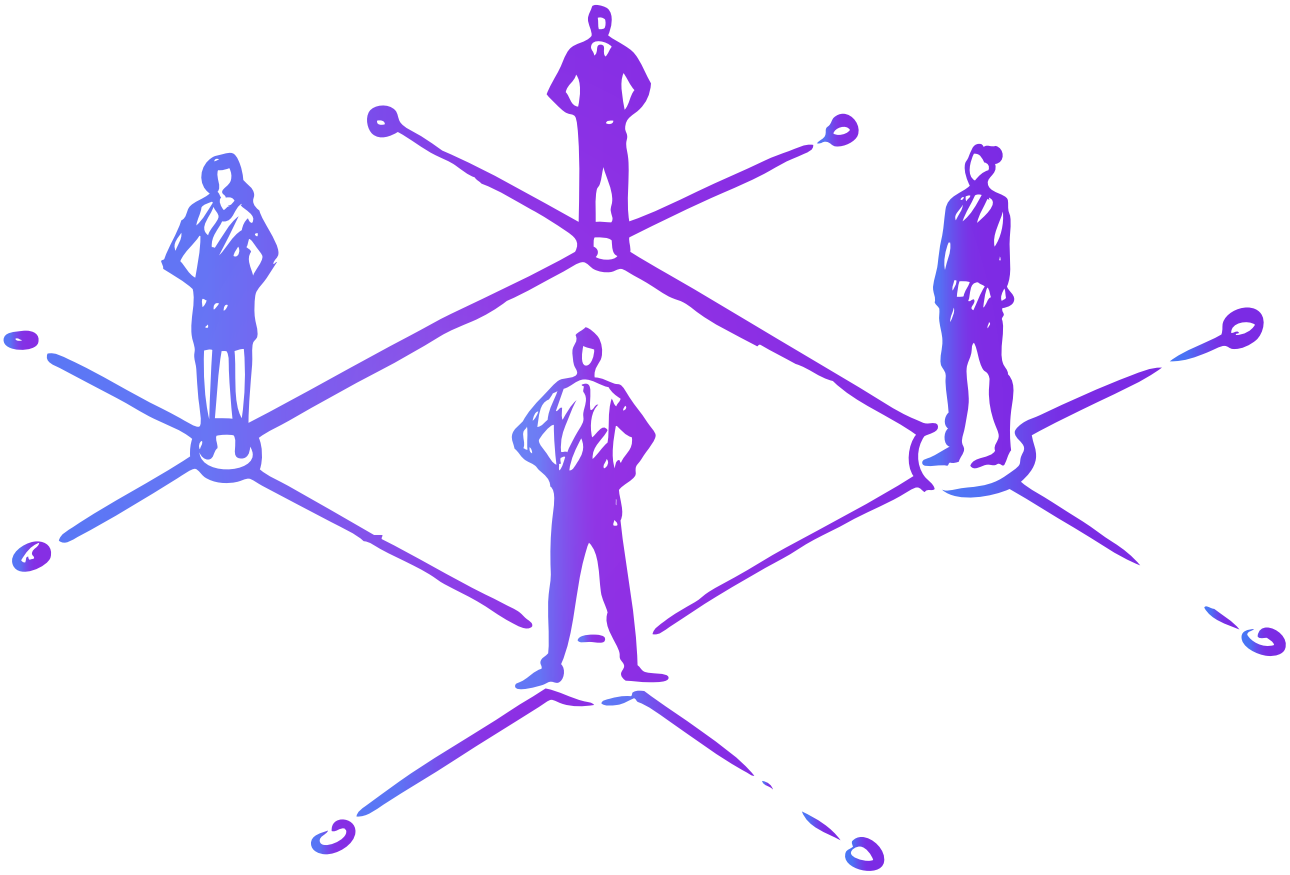
The basis of DI and SSI is a decentralized identifier or DID. A DID is very similar in design to a website's URL and looks like this:


**Figure 1: DID structure**

**In Figure 1**, both DIDs and URLs have a scheme element that shows the communication protocol. While web URLs have a website reference that describes where the webpage is hosted, DIDs have a DID method that is an online source that hosts information about the DID. Finally, the webpage and DID ID string each describe the specific object being referenced. The web URL tells a browser what protocol to use, what server to talk to, and what webpage to ask for. Similarly, a DID tells agent applications to use the DID scheme, which DID method (e.g., server) to talk to, and what DID ID string to request.

When DID methods are presented with a DID ID string request, they will return some information about the DID, which is contained in a DID document or DIDDoc. The DIDDoc is normally a JSON formatted textual file that contains data elements like public encryption keys, a service address (where to talk to the DID's owner), and any necessary algorithms for communicating with the DID owner's agent services.

One very compelling thing about DIDs is that they aren't created by any website or service that can revoke them. Rather, DIDs are created by individuals on their own computers and stored in their own wallet apps. If a user decides to swap their current wallet app for a new one, then they get to export their DIDs and other cryptographic information and import them into a new wallet. This key feature keeps the user in control of their identity information.



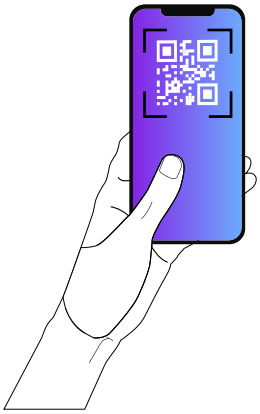
# Every DI connection is a unique relationship

Today, when creating a traditional website login account, users typically provide a username (email address) and a password (or hash value). Reusing the username provides a correlatable identifier that leads to cross-site user tracking. Also, reusing the password can give hackers a full login credential that they can use to attempt logins on other websites, which leads to greater account and identity compromises. Why do people reuse their login info? Because, simply put, remembering lots of different passwords is hard. On the other hand, DI provides users with much more secure account creation and login processes that are even easier to use.

# How are unique DI relationships created?

**When two parties wish to connect and communicate using DI methods, they start by exchanging DIDs using the Aries DID Exchange protocol. This process helps them generate unique peer DIDs (containing unique identifiers and public keys), which only they know. This ensures that each secure connection is protected separately from every other unique connection created between other parties.**

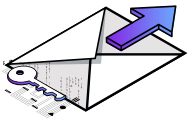
**The DID exchange process is accomplished by exchanging just a few connection messages between the connecting parties, like this:**



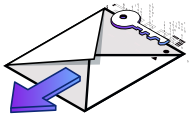
1 **Get the remote responder's DID**: There are several ways to do this, but here are a couple:

a. Scan a QR code with a cellphone: One user can display a QR code (representing their DID) on their phone and the other can scan it with their phone (see Aries RFC 0434: Out-of-Band Protocol). This can be a public DID or even a peer DID that is only used with one party.

b. Public DID: Many businesses will publish their DID on their website, business cards, etc. and these can be verified on a public ledger.



2 **Send a request message**: This is used to send the requestor's new peer DID and DIDDoc (with public keys) to the responder.

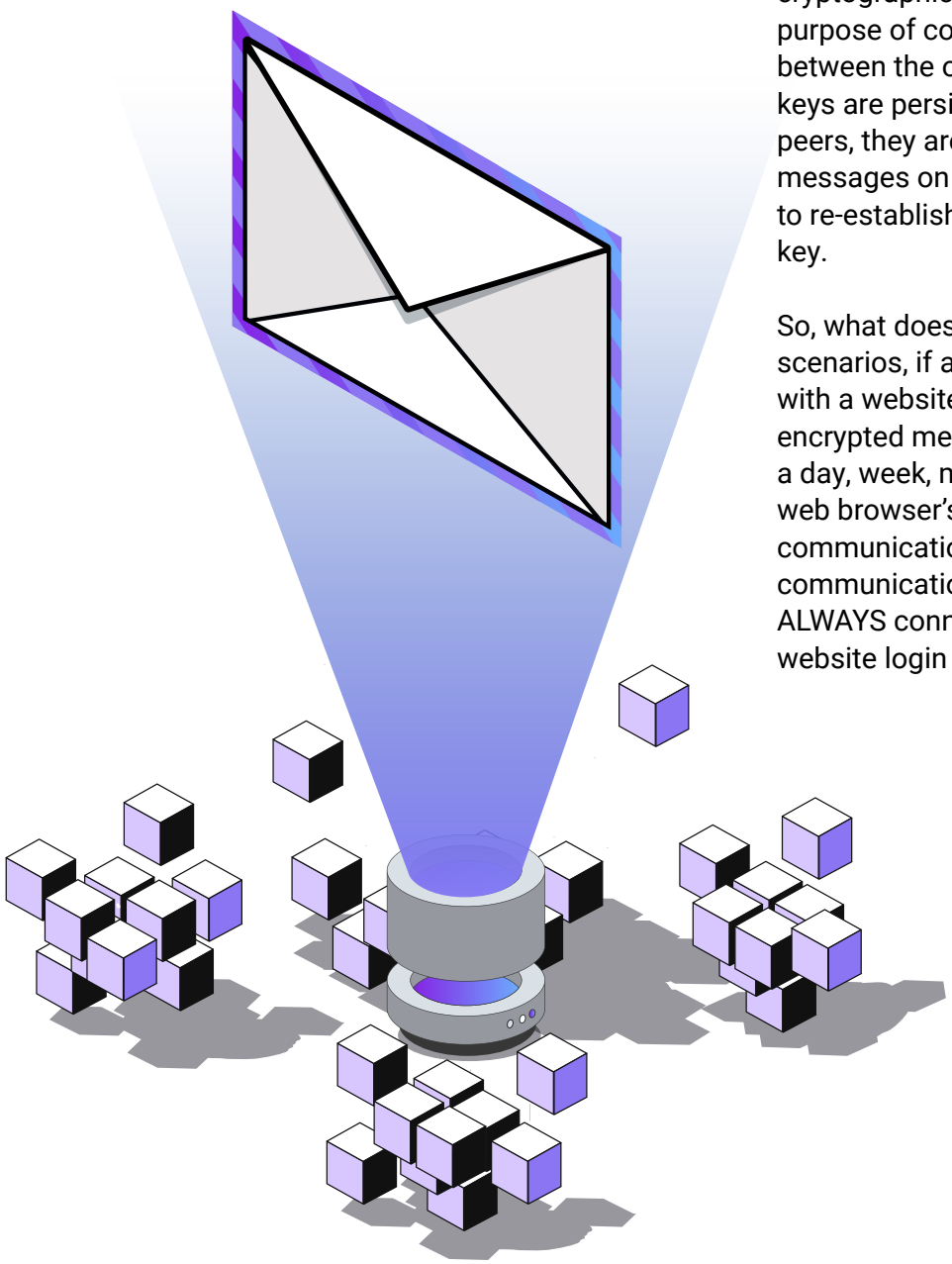


3 **Return a response message:** This message returns the responder's new peer DID and DIDDoc (with public keys) to the requestor.

4 **Return a response message:** This message returns the responder's new peer DID and DIDDoc (with public keys) to the requestor.

**The main benefit of this process is that it makes every connection with any party unique and protected by strong encryption. If any peer DID connection were to ever become compromised, then it would not affect the security of any other peer DID connection.**

# DI-based communication provides durable connections



Current connection methods and login processes use architectures that must transfer passwords, hashes, or keys to establish a secure encrypted connection. This requires every subsequent login or connection to repeat the cryptographic processes in order to re-establish a secure session despite having already performed those processes previously. This is why users must re-login when they return to websites. An exception to this is if a website stores the session credentials in a user's local web browser cache, which can persist until the web browser's cache is cleared.

Conversely, when a peer DID is created, unique cryptographic keys are created for the sole purpose of conveying secure communications between the owning peers. Since the peer DID's keys are persistent in the storage wallets of the peers, they are ready to send or receive encrypted messages on a moment's notice without any need to re-establish a session or create a new session key.

So, what does this all mean? In web-based scenarios, if a user makes a peer DID connection with a website, they can send and receive strongly encrypted messages at any time—even if that is a day, week, month, or more later—and even if the web browser's cache is cleared. Since DI-based communications effectively persist the encrypted communication channel indefinitely, they are ALWAYS connected (i.e., logged in) and the formal website login process can be eliminated!

# DIDComm is the standardized method for encrypted messaging

After exchanging DIDs, users can use those DIDs to establish secure, encrypted connections. While there are many ways to send encrypted data, the DI community has created a standardized methodology for exchanging encrypted messages, which is called DIDComm.

## The main goals of DIDComm are:

1. **Use DIDs**: Build an end-to-end communication protocol using DIDs as the foundation.

2. **Decentralized**: While other identity architectures require key registries, centralized identity providers, or certificate authorities, DIDComm leverages privacy-preserving decentralized identity architectures.

3. **Standardized cryptographic processes**: By requiring very specific cryptographic processes (e.g., encryption, digital signatures), it eliminates the guess work for application developers.

4. **Interoperability**: Ensure that messages transmitted from one agent can be processed by any other DIDComm-compatible agent. DIDComm specifically enables agents from different vendors to exchange end-to-end encrypted messages.

5. **Transport agnostic**: While some protocols are tied to a specific transport medium, DIDComm messages can be sent over any transport, such as TCP/IP, bluetooth, NFC, HTTP, QR code scanning, or even something like email.

6. **Extensibility**: While DIDComm defines a basic message structure and related protocol usage, additional protocols can be layered on top of it and transmitted in the DIDComm message payload. When a DIDComm agent receives a DIDComm message containing a layered protocol, it can either process it or reject it depending on whether the agent supports that protocol or not.

DIDComm Messaging v2 was recently released as a DIF Ratified Standard by the Decentralized Identity Foundation after a strong international standard creation effort. The DIDComm standard is published as an open source standard and may be freely used by open source and commercial projects.

# DIDComm facilitates seven message types

The simplest DIDComm message is a plaintext formatted message that is based on the IETF JSON Web Message standard. The following is a sample plaintext message:

```
{
  "id": "1234567890",
  "type": "application/didcomm-plain+json",
  "from": "did:example:alice",
  "to": ["did:example:bob"],
  "created_time": 1516269022,
  "expires_time": 1516385931,
  "body": {
    "attribute_1": "value",
    "attribute_2": "value"
  }
}
```

**Figure 2: DIDComm plaintext message**

Each message has a required id field that is unique across all messages that a sender sends. The required type field must be "application/didcomm-plain+json", which specifies a plaintext message. The optional to and from fields each represent the DIDs of the recipient and sender, respectively. The optional created and expires time fields are specified as integer values representing UTC Epoch Seconds (seconds since 1970-01-01T00:00:00Z). The required field is the body, which contains all of the data required by individual message types. Additional optional fields (not shown) include a thread identifier (thid) and a parent thread identifier (pthid), which can help sender and recipient manage message flow sequences. The final optional field (not shown) is attachments which is used to attach arbitrary files to a DIDComm message.

Building on the DIDComm Plaintext Message is the DIDComm Signed Message, which adds non-repudiable JWM signatures as an envelope around the plaintext message. The header for a signed message is represented as follows:

```
{
"typ":"JWM",
"kid":"Ef1sFuyOozYm3CEY4iCdwqxiSyXZ5Br-eUDdQXk6jaQ",
"alg":"ES256"
}
```

**Figure 3: DIDComm signed message header**

In Figure 3, the typ field denotes the message is a JWM that has been signed with ECDSA using curve p-256. The alg field denotes that the hashing algorithm used is SHA-256. The kid field is used by the sender to reference the key value used in the signature.

When a plaintext message is signed, the original plaintext message's type field must be denoted by "application/didcomm-signed+json". Assembling the DIDComm Signed Message starts with the Signed Message Header prepended to the DIDComm Plaintext Message which has the signature bytes appended.

Securing the message contents requires a DIDComm Encrypted Message, which is created as an encrypted JWM. This message uses encryption to hide the message contents to all but the intended recipient. With encrypted messages, the typ field must be "application/didcomm-encrypted+json".

Using the above message formatting, digital signature, and encryption specifications, seven permutations of messages are facilitated in DIDComm:

- **Plaintext**: basic message content without security protections
- **Signed**: a plaintext message wrapped with a digital signature
- **Anoncrypt(plaintext)**: confidential, hides the sender's identity
- **Authcrypt(plaintext)**: confidential, proves the sender's identity
- **Anoncrypt(signed(plaintext))**: confidential, hides the signature from network observers
- **Authcrypt(signed(plaintext))**: similar to previous, but not used in practice
- **Anoncrypt(authcrypt(plaintext))**: hides the header of the authcrypt envelope, used for onion routing with mediators and relays

# Routing protocol 2.0 preserves privacy

The purpose of the routing protocol is to help senders and recipients know how to exchange DIDComm messages in a cooperative privacy-preserving fashion. In the most basic message delivery scenario, a sender can transmit a message directly to a recipient. This direct scenario might be used when both sender and recipient include application processes that would be awaiting messages at any time and would appear like this:

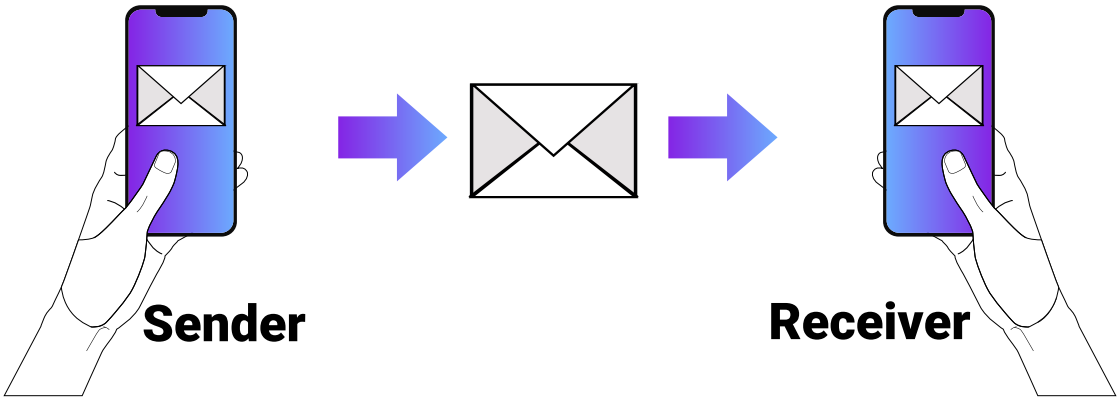



**Figure 4: Agent-to-agent message passing**

For more general internet-based cases, the routing protocol has three roles: sender, mediator, and recipient. From the DIDComm specification, Figure 5 illustrates this relationship:

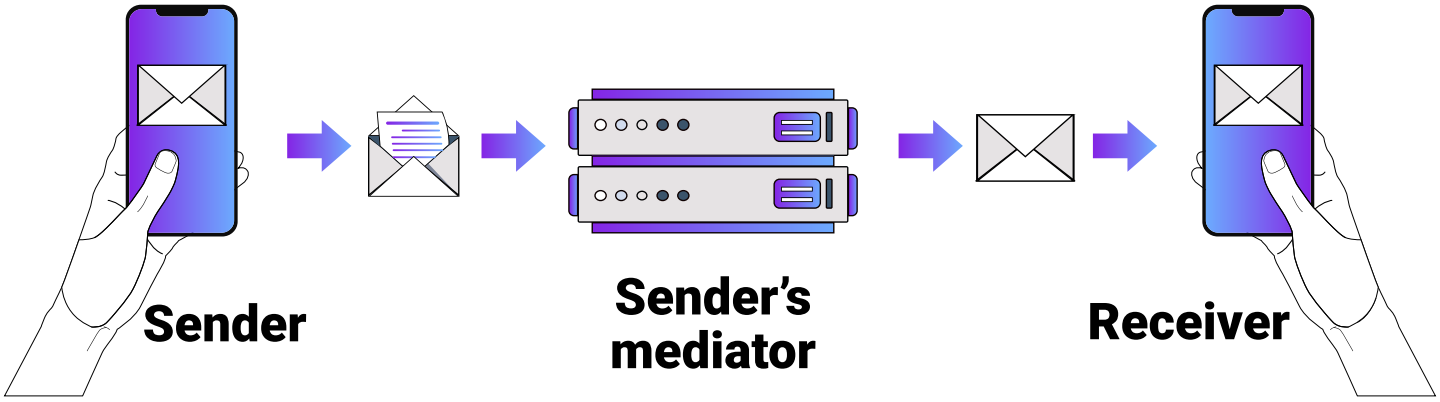



**Figure 5: Routing protocol roles**

While there are many ways for DIDComm agents (applications) to exchange messages, the routing protocol facilitates the sender and recipient never needing to directly interact with each other. This has several benefits. One benefit of using a mediator is that the sender of a message can be in a transient (not always online) device such as a cell phone. The mediator eliminates the need for sender and recipient to be online simultaneously to send a message. The paradigm in the routing protocol is similar to that used with email servers. In email transmission, a sender will pass an email to their email server and their email server will transmit it to the recipient's email server. Then, when the recipient is online, it can request any inbound messages from their email server. The routing protocol is modeled after this process and substitutes a mediator in place of an email server.

Another feature of the routing protocol is that messages can be rewrapped multiple times and this rewrapping keeps network observers from being able to identify the actual senders and recipients. Each rewrapping is analogous to the layers of an onion and is modeled after the onion routing process used by the TOR network. Figure 6 depicts how the sender wraps a message multiple times—once for each mediator that will forward the message on to the recipient. Each layer complies with the DIDComm message format and is encrypted using the DID of the mediator that will decrypt it. Once a mediator decrypts the incoming message, it will be able to forward it to the next specified mediator. Given this approach, the sender envelops the message in the reverse order of the mediators that will receive it, so that the receiving mediator can always decrypt and remove the outermost layer. There is no actual limit to how many wrapping layers a message can have, although there is certainly a practical limit.

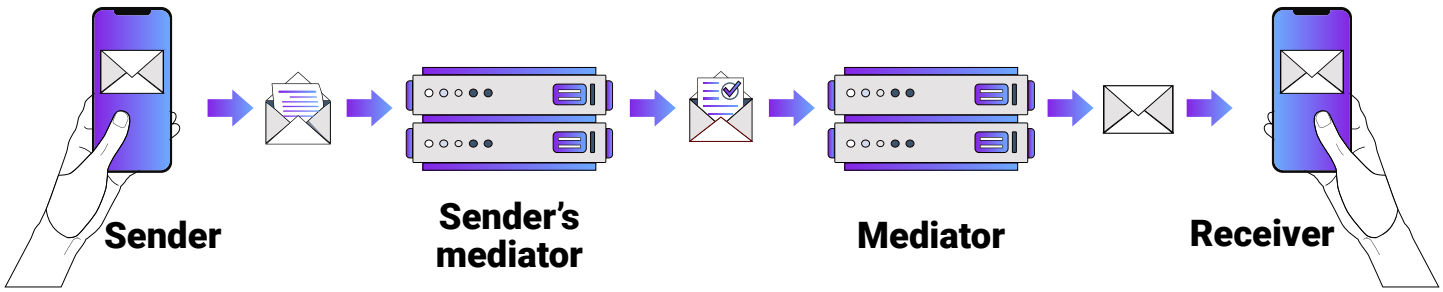
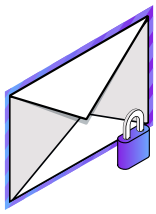



**Figure 6: Messages rewrapped for onion routing**

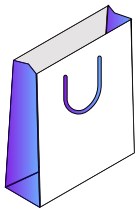
# Applications of DIDComm are virtually limitless

DIDComm provides a standards-based secure messaging channel that builds upon decentralized infrastructures. It employs very strong cryptographic methodologies that enhance the privacy and security of the end-to-end encrypted message participants. So, what can it be used for?

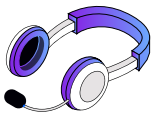


**Secure messaging**: The most obvious application is to create secure end-to-end (E2E) messaging apps. Many E2E messaging apps already exist, but each of those creates its own "walled garden" that doesn't allow communication with other E2E messaging apps. DIDComm bridges this divide by enabling E2E communications between any DIDComm-compatible provider!

**Doctor–patient communications**: Today, patients can receive digital communications (e.g., test results, x-rays) from their doctor, but they must login (with username and password again) to the website, navigate to the "test results section" and then search for the right message. Imagine an architecture where an E2E encrypted message (containing the specific test results) could be sent directly to a patient's DIDComm-enabled message reader that could automatically display the actual message instead of making patients search for it.



**Purchasing**: Current purchasing platforms need users to add their credit card information (e.g., card number, name, expiration, CVV, billing address, etc.) to the vendor's site when making a purchase. An easier solution is for a payment credential to be asserted over a DIDComm connection to a vendor who could process it. Purchasing becomes one question ("would you like to purchase?") followed by one answer ("yes"), with customers being spared the complicated details of the cryptographic payment processes.



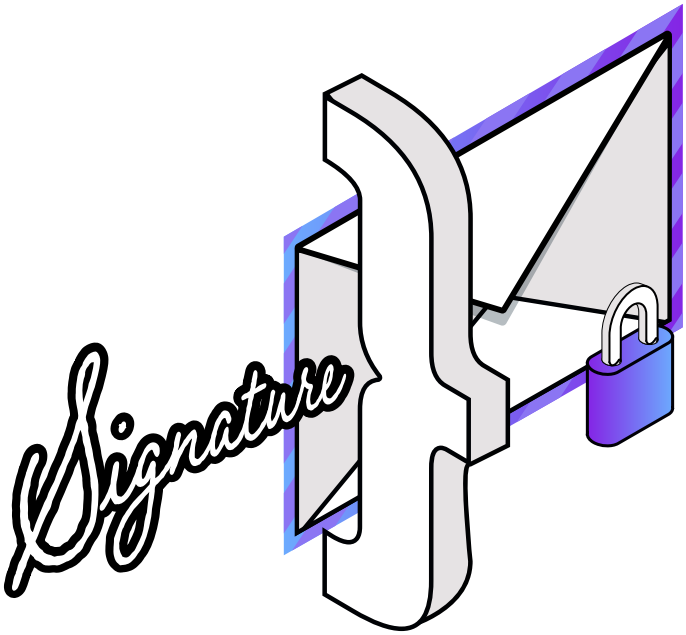
**Customer support**: Today, customers must connect with the support site and verify a series of identity details (e.g., name, birth date, home address) every time they connect to or get transferred to a new customer service rep. Since DIDComm is built on DIDs and unique cryptographic relationships, the fact that a customer is talking to a customer service rep over a DIDComm connection means that all of that verbal identity verification is obsolete! The new process is "I need some help with …" which is met with "I'm happy to assist."

**Login anywhere**: As we said, today's login processes require users to remember a myriad of username/password combinations and use centrally managed federated logins or other complicated processes. The purpose of DIDComm is to establish unique cryptographic relationships based on DIDs. Once a DIDComm connection is established, the parties' identity has already been verified and embodied in the very unique persistent connection. The result? Establishing the DIDComm connection IS the login and nothing else is necessary!

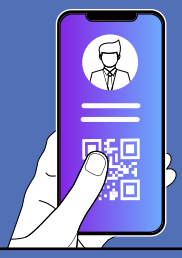
The applications of DIDComm are virtually limitless. By combining easy-to-use complex cryptography, decentralized identities, and persistent person-to-person connections, DIDComm and the related DID architectures dramatically change and simplify any application or infrastructure requiring identity verification and secure communications.

# ANONYOME LABS supports DIDComm Standards activities

The DIDComm Messaging v2 standard is created and maintained by the DIDComm Working Group, which is a part of the Decentralized Identity Foundation. DIDComm Messaging v2 has recently attained DIF Ratified Standard status and is available for immediate inclusion in any software application. Anonyome Labs supports this ongoing effort as a co-chair of the DIDComm Working Group. For software implementers of the DIDComm standard, the DIDComm Users Group meets regularly to assist developers. The DIDComm community has created didcomm.org which contains documentation about the DIDComm standard and helps implementers with creating and publishing application-level protocols layered on top of the core DIDComm standard.




## DIDComm gives users complete control over their identity. **Get on board**



The main goal of DIDComm is to create a standards-based encrypted messaging methodology that is built upon DIDs and facilitates connections between participants using software applications built by different manufacturers. This cross-vendor interoperability is critical to the mission of the larger DI movement in that it diffuses control of the communication network such that no one vendor controls its operation and definition. DIDComm's enveloping and routing processes further mask the identity of the sender and recipient and keep them hidden from network-based observers. By using DI elements (e.g., DIDs), DIDComm users first create their cryptographic identities and secondly use them to create peer relationships with other users, services, or platform providers. This gives end users complete control over their identity without being beholden to the changing policies of any particular platform provider. What's more, the DID Exchange process we've covered in this whitepaper simplifies cryptographic key distribution without requiring single points of failure, such as key repositories or certificate authorities.

**To learn more about DIDComm or to get involved in its development and enhancement, please visit didcomm.org.**