iPact
IP shared securely

Contents

1. introduction

1.1 Background and motivation

People and organizations are increasingly relying on digital platforms, cloud storage solutions and online communication channels to share sensitive information. However, this increasing digitalization not only brings opportunities and process efficiency, but also new challenges.

A key aspect, in addition to the security of data exchange, is the documentation of the fact that sensitive data has been exchanged. For example, when universities work on research projects with other research institutions or companies, confidentiality agreements usually have to be signed. Today, information covered by these agreements is mainly exchanged electronically. This raises the question of how to document that the exchange actually took place so that it can later be proven that a particular file was transferred and remained unchanged. Current methods such as email, cloud storage or messenger services often lack a reliable basis for providing evidence. Even in the case of an email data exchange, the sender may have a transmission log, but it remains unclear whether the recipient actually received the message. Furthermore, it is not automatically possible to clearly prove that the file received is the same file that was originally sent; these issues relate to key aspects of provability and process reliability

In addition, mechanisms are required to ensure that the exchange itself has taken place in a secure manner and that a file has not been modified without authorization after transmission. All in all, these challenges have not been solved in a satisfactory, cost-effective and user-friendly manner.

As part of a sub-project of the InduKo project (Innovation through Collaboration) funded by Stiftung für Innovation in der Hochschullehre, iPact was developed as a solution to the challenges described above. iPact makes it possible to document the exchange of files securely, easily and without a third party that needs to be trusted. Each transaction is recorded in an immutable ledger, making the integrity of the file and the entire transfer process verifiable. iPact uses Distributed Ledger Technology (DLT) to ensure documentation and evidence without a central infrastructure

1.2 Objectives and significance

iPact was developed to solve the central problems of (manipulation-) secure and traceable data exchange described above, which are a particular challenge for research organizations and individuals, in the most user-friendly way possible. The main objectives of the project include

1. **Development of a secure, decentralized framework for file exchange:**
   iPact eliminates the need for a centralized control instance by using DLT. Users retain full control over their data.

2. **Creating digital trust through immutability:**
   Utilizing the immutable nature of the DLT ledger ensures that all data exchanges are transparently recorded and cannot be tampered with.

3. **Promoting user-friendliness:**
   The platform is also being developed specifically for non-technical users in order to provide an intuitive and easily accessible solution without technical hurdles.

4. **Realization of a purely DLT-based solution:**
   iPact does not require any additional infrastructure such as a backend. The entire functionality is based entirely on IOTA technology.

5. **Subsequent verifiability even without iPact:**
   Subsequent proof of data exchange and the immutability of files exchanged with the help of iPact is possible purely with the standard tools of IOTA, even if iPact should later no longer be available as an application.

We only became particularly aware of the importance of the last two points 4 and 5 in the course of the development process and this challenged us during implementation. Fulfilling these two points means implementing iPact as a modern, completely decentralized application and documenting the results in a form that enables later queries without iPact. These goals were very important to us because, in our opinion, their fulfillment creates trust in the application among future users

In addition to the concrete application, iPact is also intended to show how decentralized technologies can create forward-looking solutions. In contrast to conventional centralized approaches, iPact eliminates vulnerabilities such as the "single point of failure" and minimizes risks through central data storage. The distribution of data via DLT significantly increases security and resilience, ensuring a high level of protection even in unexpected scenarios such as system failures or cyberattacks. As a tamper-proof solution, iPact also serves as an example for the development of other applications that focus on transparency, security and decentralization. The potential uses of iPact in an academic context are very interesting, but go beyond this. In addition to the exchange of confidential data in the context of non-disclosure agreements, the platform is also suitable for other scenarios in where the documentation of the exchange and the immutability of files are of central importance - both within and outside the academic world.

2. distributed ledger technology (DLT)

In order to understand how iPact works, it is necessary to have at least a very basic understanding of the underlying technical concepts. Distributed ledger technology (DLT) is based on a decentralized network architecture and differs fundamentally from traditional, centralized systems. In centralized systems, a central authority, such as a bank, is responsible for storing, managing and validating data. DLT, on the other hand, distributes these tasks to a network of independent participants or nodes.

DLT is characterized by decentralization, transparency and immutability. Decentralization ensures that no single node or central instance has control over the entire network. Transparency is ensured by the fact that all nodes have access to the same version of the ledger. This allows all participants in the network to check transactions independently. The immutability of the ledger is guaranteed by cryptographic mechanisms that ensure that data cannot be changed or deleted once stored; this is particularly important to create trust between participants and ensure the integrity of the data. DLT is ideal for applications in which security, reliability and transparency are paramount and in which the participants want to achieve these goals independently of a central authority.

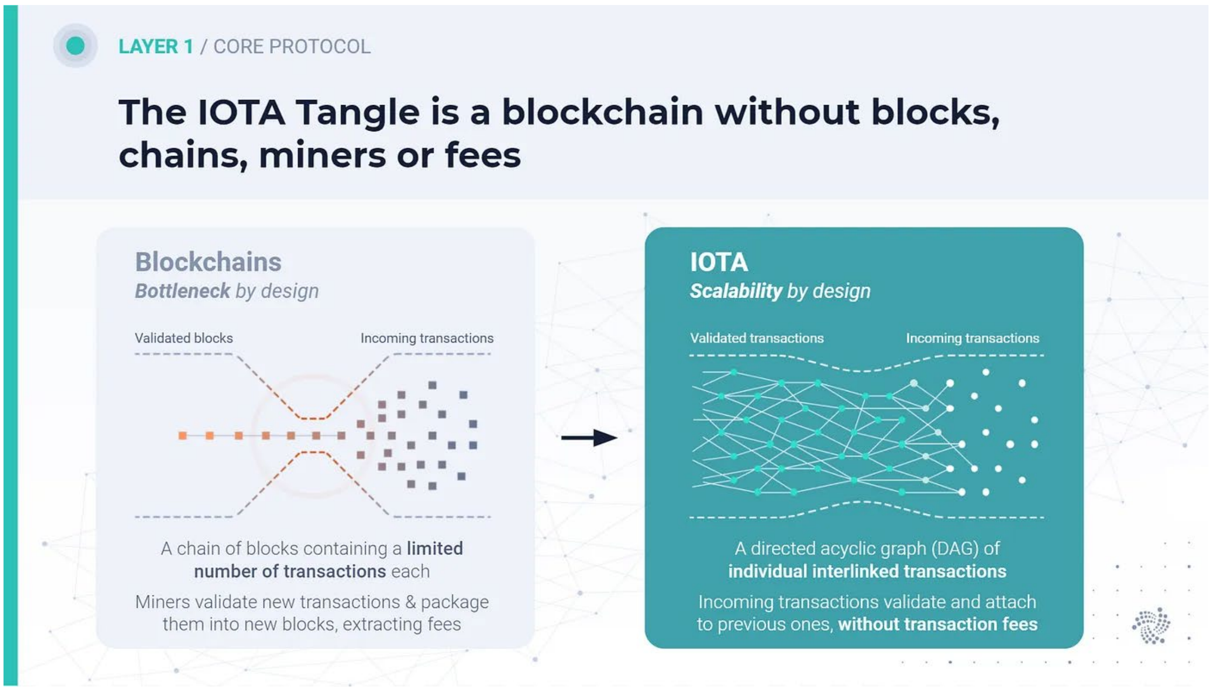
There are various implementation approaches within DLT, with blockchain and directed acyclic graph (DAG) currently being very relevant forms of implementation. Blockchain, the traditional form of DLT that is best known for Bitcoin, consists of a linear chain of blocks that are linked chronologically and cryptographically. Each new block contains transactions and a hash of the previous block, creating an unchangeable chain. Consensus mechanisms such as Proof of Work (PoW) or Proof of Stake (PoS) ensure that all transactions are validated before they are added to the blockchain. These mechanisms offer a high level of security, but are sometimes very resource-intensive and their scalability is limited. The linear structure of the blockchain means that transactions are processed sequentially, which limits the number of transactions per second (TPS).

In contrast, the Directed Acyclic Graph (DAG) does not use a linear structure, but a network-like architecture. DAG-based systems are particularly suitable for applications where high transaction volumes, low latency times and minimal transaction costs as well as very low resource consumption are required.

2.1 Reasons for IOTA: The Tangle compared to the blockchain

The choice of IOTA as the base technology for our project is based in particular on the specific characteristics of the IOTA Tangle, which make it more advantageous than traditional blockchain systems. While blockchain is based on a linear sequence of blocks validated by Proof-of-Work (PoW) or Proof-of-Stake (PoS), the IOTA Tangle uses a Directed Acyclic Graph (DAG) structure. The IOTA Tangle was the first implementation of DAG technology and was developed with the aim of meeting the requirements of modern applications in the Internet of Things (IoT).

The figure below illustrates the fundamental structural differences between traditional blockchains and the IOTA Tangle.


LAYER 1 / CORE PROTOCOL
The IOTA Tangle is a blockchain without blocks, chains, miners or fees
Blockchains
Bottleneck by design
Validated blocks
Incoming transactions
A chain of blocks containing a limited number of transactions each
Miners validate new transactions & package them into new blocks, extracting fees
IOTA
Scalability by design
Validated transactions
Incoming transactions
A directed acyclic graph (DAG) of individual interlinked transactions
Incoming transactions validate and attach to previous ones, without transaction fees

Source: https://medium.com/@parecejacob/iota-bytes-assembly-of-data-the-new-eco-5f413953660

The concept of a blockchain is shown on the left-hand side of the image. Each block contains a limited number of transactions and new transactions must be validated by miners and integrated into these blocks. This process leads to an inherent bottleneck, as the block size and the number of miners limit the processing capacity. In addition, many blockchain systems require fees for transaction processing, as miners must be incentivized for their work, which is not the case with IOTA, which is why there are no transaction fees.

The illustration on the right shows how the IOTA Tangle works. Instead of a linear chain, the Tangle uses a directed acyclic graph structure (DAG), a parallelized structure. Transactions do not have to be collected and then connected linearly, but can be added in parallel at the different ends of the DAG.

The proof-of-work mechanism that ensures the security of the network in Bitcoin, for example, consumes considerable amounts of energy, which not only causes high operating costs but also has significant environmental consequences. IOTA, on the other hand, requires no miners and is considered one of the most resource-efficient DLTs of all.

Apart from the key advantages of scalability, energy consumption and transaction costs, the IOTA Foundation works closely with international organizations, regulatory authorities and non-profit initiatives. The IOTA Foundation, with offices in Germany, Switzerland and the United Arab Emirates, has established itself as a trusted partner, especially in Europe, where it is involved in projects such as the European Blockchain Services Infrastructure (EBSI). The European Blockchain Services Infrastructure is an initiative of the European Commission, among others, with the aim of establishing a Europe-wide blockchain infrastructure that supports cross-border public services. IOTA therefore not only meets the requirements of the European legal framework, but is also actively involved in shaping DLT standards.

These characteristics make IOTA a technically, ecologically and economically sustainable choice for applications that rely on a secure, efficient and resource-saving DLT infrastructure.

2.2 IOTA - some important technical basics

The IOTA ecosystem comprises several networks that fulfill different requirements. The mainnet serves as a production environment for real-world transactions, while the testnet is used as a sandbox for developers to test new features. In addition, there is Shimmer, a staging network that allows developers to test protocol changes before they are introduced on the mainnet. These networks provide a flexible infrastructure that makes it easier for developers to develop and test secure and scalable applications such as iPact.

Each transaction in the IOTA Tangle consists of several essential components. The inputs of a transaction represent unused outputs from previous transactions and define which resources are available for a new transaction. Unlock blocks contain digital signatures that confirm that the inputs are authorized. These signatures ensure that only the owner of the inputs can initiate transactions, which prevents unauthorized access.

Outputs save the results of the transaction and remain in the Tangle until they are used again. Mechanisms such as the Expiration Unlock Condition define time limits after which unused outputs are transferred to a return address. The Timelock Unlock Condition delays access to outputs until a certain point in time. The metadata function makes it possible to store any data within transaction outputs. This data remains permanently available in the network and can be accessed on all nodes. For iPact, the metadata function is essential for anchoring information such as file descriptions, author IDs and notes directly in the transaction. The tag function allows up to 64 bytes of indexable data to be appended to a transaction output. This feature is particularly useful for iPact as it allows transactions to be categorized and tagged, for example with labels such as "research collaboration". This makes it much easier to search for and manage data sets. The tags, like the metadata, are stored as binary data and are compatible with all nodes in the network.

Finally, the Storage Deposit Return Unlock Condition ensures that users must make a deposit for storage space, which is refunded after the transaction. This "deposit" ensures that the data is retained and not deleted from the ledger. In this respect, although the transaction itself does not cost any fees, its use is not entirely free, because the necessary deposit ties up capital without interest. In this respect, this lost interest on the tied-up capital does lead to certain (opportunity) costs, even if they are absolutely negligible for our use case.

In the IOTA protocol, so-called mnemonic phrases are used to generate and manage cryptographic seeds in a user-friendly way. A mnemonic phrase is a sequence of easily understandable and pronounceable words that allows users to easily store and retrieve complex cryptographic keys when needed. This method significantly improves both security and accessibility by minimizing human error in the management of cryptographic data.

A mnemonic phrase can be compared to a password in this respect, but fulfills a more specific cryptographic function. It usually consists of 12, 18 or 24 words that are selected from a fixed dictionary, such as the BIP39 standard. Each of these words represents a part of a cryptographic key. Together they form the so-called seed, which is the basis for the generation of private and public keys or addresses. These words are deliberately designed to be easy to remember and linguistically unique. A phrase such as "apple window cat moon tree etc." conceals a complex cryptographic key that enables access to an account and the execution of transactions.

A seed is a random, cryptographically generated string of characters that is used in the IOTA network for all important operations. A seed acts as a "master key" that allows access to all data and operations associated with the account. If a seed is derived from a mnemonic phrase, it can be reproduced at any time as long as the phrase has been stored securely. This provides a high level of security and recoverability. iPact utilizes this concept to make cryptographic key management as simple and secure as possible. In the IOTA protocol, public addresses are then derived from a seed using cryptographic algorithms. These public addresses serve as identifiers, similar to account numbers in a banking system.

The IOTA Explorer is an online tool that makes it possible to transparently view and check all transactions and data stored in the IOTA Tangle. The explorer supports both the mainnet and the testnet and is therefore versatile - from developing new applications to checking live data. For applications such as iPact, this means that all processes documented in the Tangle can be checked at any time without additional software, even if this is not convenient.

2.3 Cryptography

Cryptography is a technology used to protect digital information from unauthorized access, manipulation and misuse. It provides the basis for secure communication, data integrity and the authentication of users and systems. Two central approaches to encryption are symmetric and asymmetric encryption, which are each based on specific principles . In addition, hashing plays a crucial role in verifying the integrity of data.

Symmetric encryption is based on the fact that the same key is used to both encrypt and decrypt data. The main disadvantage here is usually the distribution of the key: both parties must securely exchange the same key, which is a challenge in many scenarios.

In contrast, asymmetric encryption uses a key pair consisting of a public key and a private key. The public key is used to encrypt the data and can be securely shared with others, while the private key is only known to the owner and is used for decryption. This enables secure communication without the direct exchange of a shared key.

Another key aspect of cryptography is hashing. This is a process in which a unique, fixed length of characters (known as a hash) is generated from any data. Hashing is primarily used to check the integrity of data, as any change to the input data, no matter how small, completely changes the resulting hash. Well-known algorithms such as SHA-256 are used, for example, in blockchain networks to secure transactions or in digital signatures to ensure the authenticity of documents. Hashing is deterministic, meaning that the same input always produces the same hash. In addition, it is practically impossible, at least to date, to recover the original data from a hash, which makes hashing an important tool for data security.

3. basic architecture of the solution

The development of the basic architecture and the final solution of our project was characterized by challenges that were shaped both by our own, in our opinion very important and sensible specifications, as well as by technological changes.

During the project period, IOTA released a major protocol upgrade (version 1.5, also known as "Chrysalis"), which significantly changed the technological basis for our development. Among other things, this update introduced the Shimmer testnet and new features in terms of content, which opened up new opportunities for us, but also created new challenges. The technical development of the underlying technology with its far-reaching changes has not only taken away conceivable solutions, it has also created new possibilities; these new functions also included the introduction of smart contracts, for example. We then also included the use of smart contracts in our considerations and did not rule them out for later development stages, but we endeavored to develop a solution that, if possible, could be implemented completely without additional infrastructure on layer 1, i.e. natively within the IOTA core protocol, so to speak, with all the advantages in terms of costs, security and technical and economic sustainability.

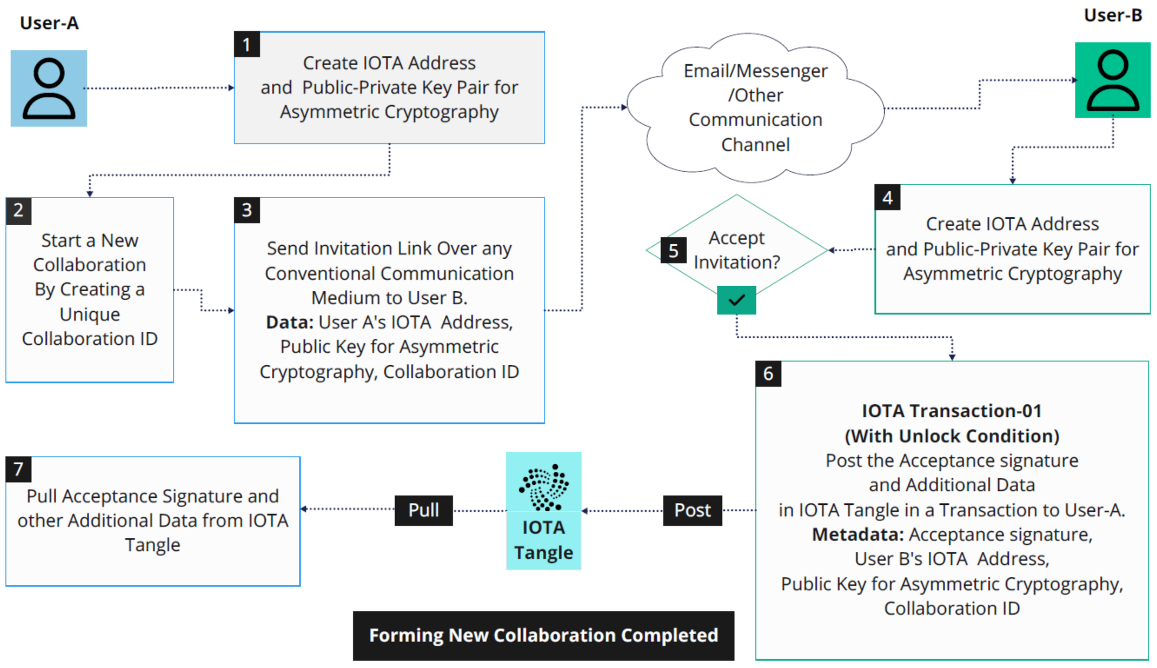
The basic architecture is presented below. First, the overall solution concept based entirely on IOTA's Layer 1 is described. Then an exemplary technical implementation is presented, which was developed during the project in order to validate the architecture at . The final step describes the implementation as a user-friendly smartphone application based on the cross-platform development environment Flutter.

3.1 Step 1: Establishing a new "Collaboration" between User-A and User-B

In order to understand the process of setting up a "collaboration" within the iPact system, it is important to first clarify what is meant by "collaboration" in this technical context. In this case, collaboration involves the structured establishment of a connection between two parties, for example between institutes, professors, research groups and companies that wish to exchange sensitive data with each other. This exchange often takes place in the context of research projects or other collaborations in which a high degree of confidentiality must be guaranteed, for example through a non-disclosure agreement (NDA).

This technical mapping of a collaboration fulfills several essential functions: It technically documents the intention and mutual agreement of the parties involved, ensures that the technological basis for the subsequent secure exchange of data is created and thus provides a comprehensible, structured basis. The creation of this structure is therefore the first phase of secure collaboration and lays the foundation for all subsequent steps.

1 Forming New Collaboration Between User-A and User-B


User-A
User-B
1 Create IOTA Address and Public-Private Key Pair for Asymmetric Cryptography
Email/Messenger /Other Communication Channel
2 Start a New Collaboration By Creating a Unique Collaboration ID
3 Send Invitation Link Over any Conventional Communication Medium to User B. Data: User A's IOTA Address, Public Key for Asymmetric Cryptography, Collaboration ID
5 Accept Invitation?
4 Create IOTA Address and Public-Private Key Pair for Asymmetric Cryptography
6 IOTA Transaction-01 (With Unlock Condition) Post the Acceptance signature and Additional Data in IOTA Tangle in a Transaction to User-A. Metadata: Acceptance signature, User B's IOTA Address, Public Key for Asymmetric Cryptography, Collaboration ID
7 Pull Acceptance Signature and other Additional Data from IOTA Tangle
Pull
IOTA Tangle
Post
Forming New Collaboration Completed

The diagram shows the process of creating a collaboration; the individual steps are explained below:

1. **Creation of the IOTA address and the key pair for User-A**

   User-A begins the process by creating an IOTA address, which serves as a unique identifier and enables transactions to be carried out on the Tangle. At the same time, User-A generates an asymmetric key pair consisting of a public and a private key. These keys are required for subsequent encryption and decryption as part of the data exchange.

2. **Create a unique Collaboration ID**

   In the next step, User-A generates a unique collaboration ID. This ID serves as a unique identifier for the collaboration and is required to uniquely identify and track the communication and interaction between the participants. You can visualize this unique "collaboration" as a dedicated folder that the parties will use in the future to exchange information.

3. **Sending the invitation to User-B**

   User-A sends the invitation to User-B conveniently via a conventional communication channel, for example by email or messenger. The invitation contains User-A's IOTA address, the public key for asymmetric encryption and the collaboration ID. This data is essential so that User-B can check and confirm the invitation.

4. **Setup for User-B**

   After User-B has received the invitation, he carries out a similar setup to User-A. He creates his own IOTA address and generates an asymmetric key pair. These steps ensure that User-B also has the necessary resources for encrypted communication and authentication.

5. **Accepting the invitation**: User-B decides whether to accept the invitation to collaborate. If he rejects it, the process ends at this point. However, if User-B accepts the invitation, step no. 6 follows.

6. **Posting a transaction on the IOTA Tangle**

   After User-B has accepted the collaboration, he creates a transaction on the IOTA Tangle. This transaction contains metadata that documents the collaboration. The metadata includes:

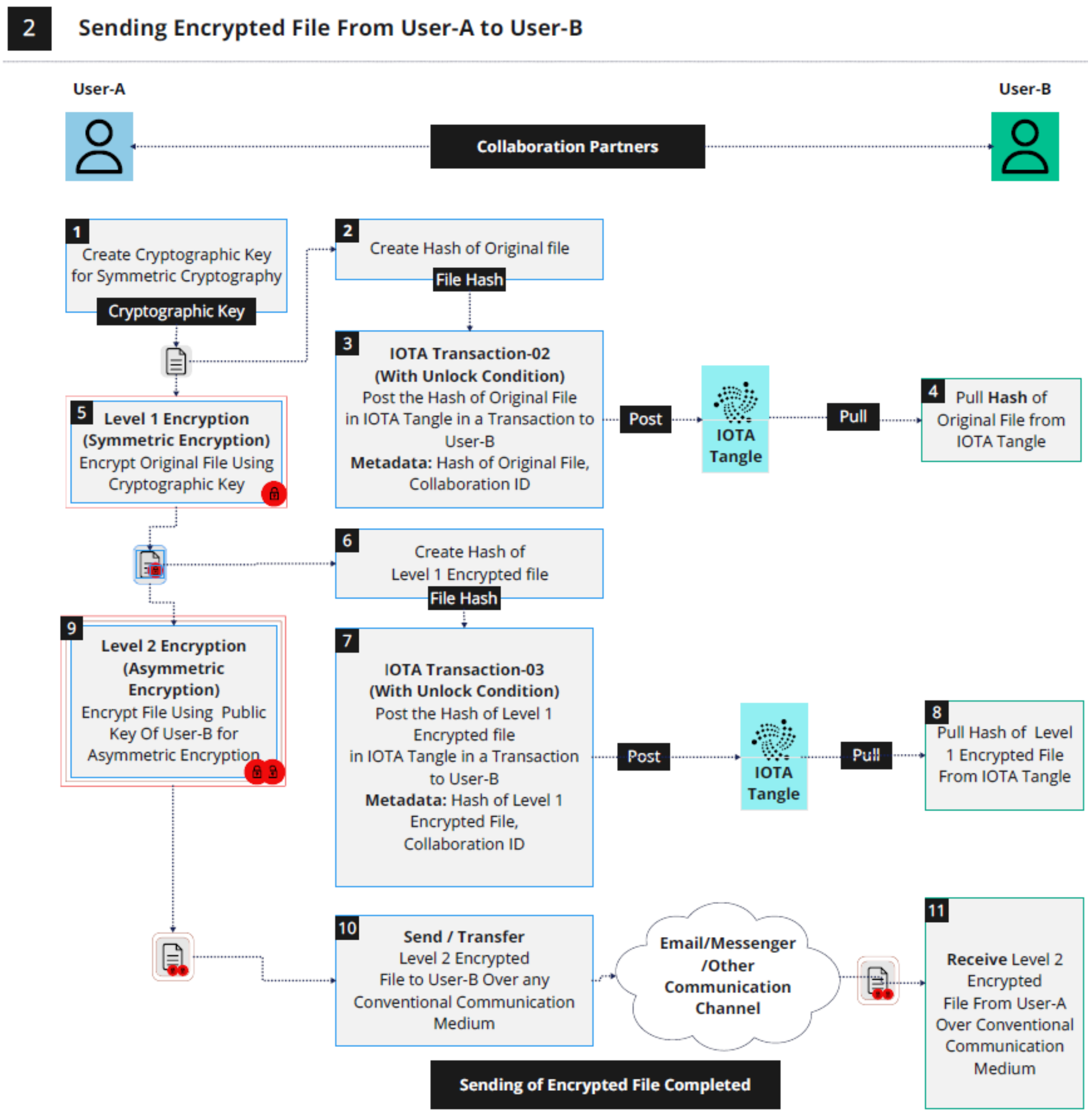
   - The acceptance signature confirming User-B's approval.
   - The IOTA address and the public key of User-B.
   - The collaboration ID to uniquely assign the transaction.
   - In addition, User-B deposits a minimal amount of IOTA tokens as security to protect the data from future deletion (see section 2.2).
   - These tokens, and thus also the metadata associated with this transaction, remain locked or protected against deletion for a long-term period of, for example, 50 years (controllable in the code).

7. **Confirmation of**
   User-A retrieves User-B's transaction data from the IOTA Tangle. This data includes the acceptance signature, the IOTA address, the public key and the collaboration ID of User-B. This step officially concludes the collaboration and both participants can interact with each other securely.

Once this process was completed, a secure and verifiable collaboration between User-A and User-B was created and documented on the DLT ("blockchain").

3.2 Step 2: Transmission of an encrypted file from User-A to User-B

The transfer of a file from user A to user B goes far beyond mere encryption and transmission. A key challenge was to develop a tamper-proof method that documents beyond doubt that the recipient has not only received the file, but can actually access it. It also had to be ensured that the file received was exactly the same as the one sent. The solution to this "handshake mechanism" was ultimately made possible by using functions available on IOTA Layer 1 in combination with proven cryptographic procedures and a corresponding process design. The steps are explained in detail below.


2
Sending Encrypted File From User-A to User-B
User-A
User-B
Collaboration Partners
1 Create Cryptographic Key for Symmetric Cryptography
Cryptographic Key
2 Create Hash of Original file
File Hash
3 IOTA Transaction-02 (With Unlock Condition) Post the Hash of Original File in IOTA Tangle in a Transaction to User-B Metadata: Hash of Original File, Collaboration ID
Post
IOTA Tangle
Pull
4 Pull Hash of Original File from IOTA Tangle
5 Level 1 Encryption (Symmetric Encryption) Encrypt Original File Using Cryptographic Key
6 Create Hash of Level 1 Encrypted file
File Hash
7 IOTA Transaction-03 (With Unlock Condition) Post the Hash of Level 1 Encrypted file in IOTA Tangle in a Transaction to User-B Metadata: Hash of Level 1 Encrypted File, Collaboration ID
Post
IOTA Tangle
Pull
8 Pull Hash of Level 1 Encrypted File From IOTA Tangle
9 Level 2 Encryption (Asymmetric Encryption) Encrypt File Using Public Key Of User-B for Asymmetric Encryption
10 Send / Transfer Level 2 Encrypted File to User-B Over any Conventional Communication Medium
Email/Messenger /Other Communication Channel
11 Receive Level 2 Encrypted File From User-A Over Conventional Communication Medium
Sending of Encrypted File Completed

1. creation of a symmetric key by User-A

User-A generates a cryptographic key for symmetric encryption. This key forms the basis for the first encryption of the file, i.e. for the "inner shell" of the encryption.

2. generation of a hash value of the original file

User-A creates a unique hash value of the original file using a hashing algorithm (e.g. SHA-256). This hash serves as an unchangeable fingerprint of the file and is later used to ensure that the file has not been manipulated during the process.

3. posting the hash value of the original file in the IOTA Tangle

The hash value of the original file is published together with the collaboration ID as a transaction on the IOTA Tangle. This transaction is provided with a time lock (lock condition) so that the token amount of the transaction remains secure for a specified period (e.g. 50 years). This lock protects the transaction information from unauthorized access and ensures its long-term availability.

4. retrieval of the hash value by User-B

User-B downloads the hash of the original file stored in the Tangle. This hash enables User-B to check the integrity of the received file later.

5. first encryption level (symmetric)

User-A encrypts the original file with the previously generated symmetric key. This encryption level also protects the file from unauthorized access during transmission, but above all, this encryption simultaneously implements an essential prerequisite for the handshake mechanism: The subsequent decryption of this layer by user B proves that the file has been received and that access by user B is possible.

6. generation of a hash value of the level 1 encrypted file

User-A generates a new hash value from the level 1 encrypted file. This hash is published again in the Tangle to ensure that the encrypted file remains unchanged after the initial encryption.

7. posting the hash value of the level 1 encrypted file in the IOTA Tangle

The hash value of the level 1 encrypted file is stored together with the collaboration ID as a second transaction on the Tangle. This transaction is also protected by a time lock, which ensures that it cannot be changed.

8. retrieval of the hash value of the level 1 encrypted file by User-B

User-B downloads the hash value of the level 1 encrypted file. This step provides an additional level of verification to ensure that the encrypted file has not been tampered with during transmission.

9. second encryption level (asymmetric)

User-A encrypts the already symmetrically encrypted file (level 1 encrypted file) with the public key of User-B. This second encryption level ensures that only User-B, who has the corresponding private key, can decrypt the file.

10. transfer of the level 2 encrypted file

The double-encrypted file is sent to User-B via a conventional communication channel (e.g. email, messenger). This method is secure, even if the communication channel is compromised, because the file can only be accessed with the two required keys.

11. reception by User-B

User-B receives the double-encrypted file, the file is ready for decryption. This process and the associated documentation are described below.

3.3 Step 3: Decryption of the received encrypted file by User-B

After receiving the double-encrypted file, User-B begins the decryption process to make the original file content accessible. This phase, as shown in Figure 3, describes the individual steps for secure decryption, verification and confirmation of data integrity. The aim is to ensure that the file remains unchanged and only accessible to User-B.

1. first decryption (asymmetric decryption)

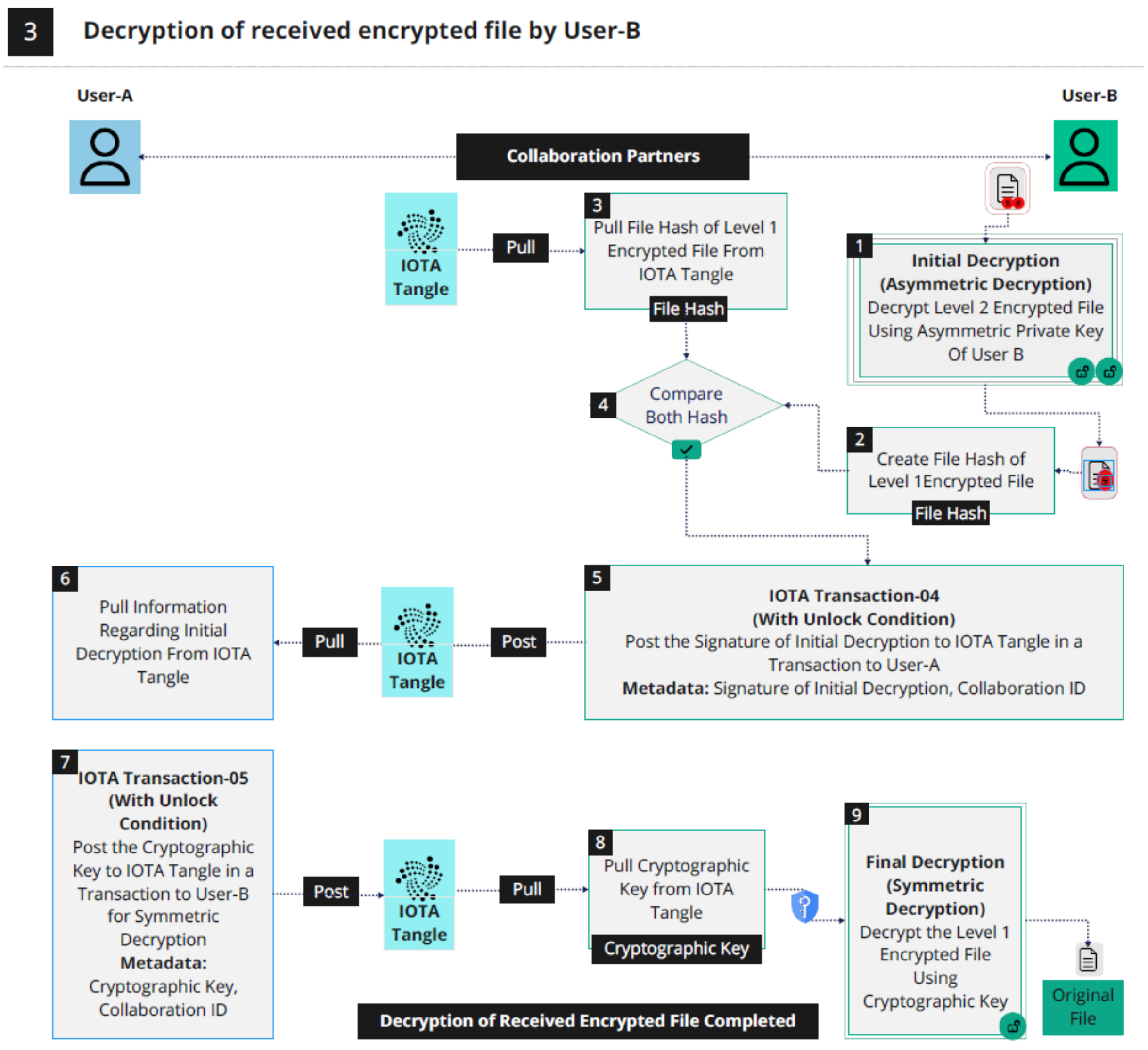
User-B decrypts the double-encrypted file using his private key. This layer of decryption removes the outer "shell" that User-A previously encrypted with User-B's public key. This exposes the level 1 encrypted file.

2. creation of the hash of the level 1 file

After the level-1 encrypted file has been decrypted, User-B generates a hash of this file using the standardized hashing algorithm that was also used by User-A. This hash acts as a fingerprint of the level-1 file. This hash, which acts as a fingerprint of the level-1 file, can later be used to verify the integrity of the data.

3. retrieve the level 1 file hash from the IOTA Tangle

User-B accesses the hash of the level-1 encrypted file previously published by User-A, which was stored as an immutable reference point on the IOTA Tangle. This hash was uploaded by User-A during the file transfer phase to guarantee the integrity of the file.



3
Decryption of received encrypted file by User-B
User-A
User-B
Collaboration Partners
IOTA Tangle
Pull
3
Pull File Hash of Level 1 Encrypted File From IOTA Tangle
File Hash
1
Initial Decryption (Asymmetric Decryption)
Decrypt Level 2 Encrypted File Using Asymmetric Private Key Of User B
4
Compare Both Hash
2
Create File Hash of Level 1Encrypted File
File Hash
6
Pull Information Regarding Initial Decryption From IOTA Tangle
Pull
IOTA Tangle
Post
5
IOTA Transaction-04 (With Unlock Condition)
Post the Signature of Initial Decryption to IOTA Tangle in a Transaction to User-A
Metadata: Signature of Initial Decryption, Collaboration ID
7
IOTA Transaction-05 (With Unlock Condition)
Post the Cryptographic Key to IOTA Tangle in a Transaction to User-B for Symmetric Decryption
Metadata: Cryptographic Key, Collaboration ID
Post
IOTA Tangle
Pull
8
Pull Cryptographic Key from IOTA Tangle
Cryptographic Key
9
Final Decryption (Symmetric Decryption)
Decrypt the Level 1 Encrypted File Using Cryptographic Key
Original File
Decryption of Received Encrypted File Completed

4. verification of file integrity

User-B compares the locally generated hash of the level 1 file with the hash that was retrieved from the IOTA Tangle. If both hashes match, this confirms that the level 1 file was neither changed nor manipulated during the transfer.

5. confirmation of the decryption

After successful verification of the file integrity, User-B publishes a transaction on the IOTA Tangle. This transaction contains a signature that documents the successful decryption of the file up to the first level. The transaction also contains the collaboration ID and relevant metadata. This informs User-A that User-B has successfully decrypted and verified the file - User-B therefore owns the file and has access to it, the file is not corrupted.

6. verification of the decryption confirmation

User-A retrieves the confirmation data from the corresponding transaction on the IOTA Tangle. This serves as proof that User-B has received the file correctly, decrypted it and verified its integrity. This is a crucial step for User-A to ensure that the process has run as intended and that all the requirements for completing the process have been met.

7. publication of the symmetric key

After verification by User-A, the symmetric key used for level 1 encryption is stored by User-A on the IOTA Tangle. The key is published as a time-locked transaction, which means that User-B can access it within a defined period of time.

8. retrieving the symmetric key

User-B retrieves the symmetric key from the corresponding transaction on the IOTA Tangle. This key is required to remove the last encryption layer and gain access to the original file.

9. final decryption

User-B decrypts the level 1 encrypted file with the symmetric key. This final step releases the original file and marks the end of the process.

The presented process implements a tamper-proof handshake, which is ensured by the combination of cryptographic methods and the immutable documentation in the IOTA Tangle. Important points of the security mechanism are

1. **Unchangeable documentation**
   Every step - from creating the hash to publishing the key - is documented on the Tangle. This provides transparent and tamper-proof traceability.
2. **Confirmation of access**
   The decryption of the level 2 encryption by User-B and the subsequent documentation in the Tangle prove that User-B was not only able to obtain the file, but also successfully access it.
3. **Final handover of keys**
   The symmetric key is only released by User-A after confirmation by User-B in the Tangle, which guarantees the security and integrity of the entire process.

4. **Long-term security**
   The temporary blocking of transactions ensures that the stored data remains accessible in the long term without the possibility of manipulation or deletion.

The innovative combination of asymmetric and symmetric encryption with documentation via the IOTA Tangle ensures that the entire exchange process remains fully traceable at every stage. Not only is it documented that User-B has received the file, but it is also proven beyond doubt that he actually had access to it. Only after this confirmation is the final symmetric key released, which guarantees a tamper-free transfer. All steps are designed in such a way that no trust in third parties is required, as security and transparency are fully guaranteed by cryptographic processes and the decentralized nature of the IOTA Tangle.

This solution shows that even under the selected requirements - such as the exclusive use of layer 1 functionalities without external infrastructure - maximum security, transparency and long-term traceability can be achieved.

3.4 Step 4: Checking the immutability of the shared file

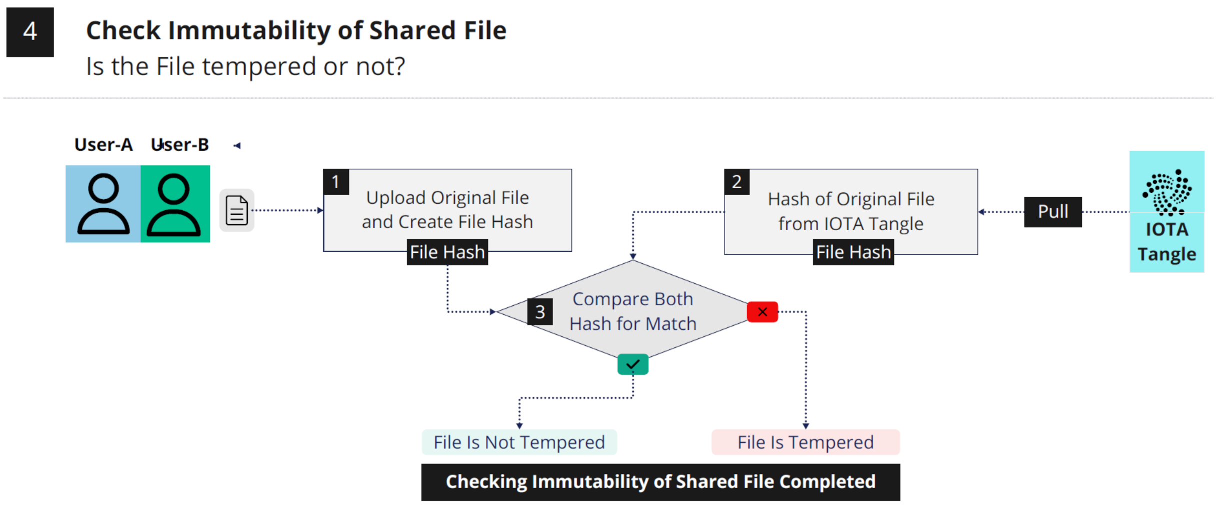
Once the data exchange and decryption phases have been completed, both User-A and User-B can independently verify the immutability of the shared file. This verification ensures that the file has remained unchanged over time, so even years later it is still possible to prove which file it was, even though the file itself cannot be stored in the IOTA Tangle. The process, as shown in Figure 4, uses cryptographic hashing and the immutable records on the IOTA Tangle to ensure that no modifications have been made to the file since it was shared.

1. generation of a current file hash

The respective user (either User-A or User-B) uploads the file in question and generates its current hash value using the same cryptographic hash algorithm that was used during the original file transfer process (e.g. SHA-256). This hash value serves as the current "fingerprint" of the file and represents its current state.

2. retrieval of the stored file hash from the IOTA Tangle

The user retrieves the original file hash from the IOTA Tangle that was saved during the initial file transfer. This stored hash value serves as a verifiable reference for the original state of the file and forms the basis for the integrity check.


4
Check Immutability of Shared File
Is the File tempered or not?
User-A
User-B
1
Upload Original File and Create File Hash
File Hash
2
Hash of Original File from IOTA Tangle
File Hash
Pull
IOTA Tangle
3
Compare Both Hash for Match
File Is Not Tempered
File Is Tempered
Checking Immutability of Shared File Completed

3. comparison of the hashes for the integrity check

The current, newly generated hash value of the file is compared with the original hash stored on the IOTA Tangle. If the two hash values match, this confirms that the file has not been modified or manipulated since it was first transferred and its integrity remains intact. However, if the hash values do not match, this indicates that the file has been modified, which means that the immutability check has not been passed.

4. completion of the immutability check

After comparing the hash values, the user receives clear confirmation as to whether the file is still in its original, unaltered state. This verification process can be repeated at any time, allowing the collaboration partners to independently verify the authenticity of the file, especially at a later point in time.

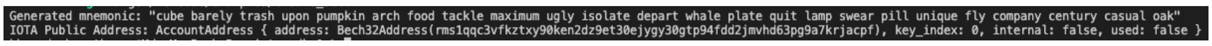
The process described in the previous chapters describes a coordinated interplay of various mechanisms designed to ensure the tamper-proof and transparent exchange of sensitive files - an innovative solution, but one that also entails a corresponding level of complexity. The next challenge therefore lies not only in the technical validation of the concept in order to confirm its functionality as planned, but also in the development of a user-friendly smartphone app. This should enable users to organize the exchange efficiently - especially when multiple files are involved, which is often the case in practice. In the following chapter, the technical validation of the concept is briefly described before the developed smartphone app is presented in detail.

4. experimental implementation and validation

To validate the proposed framework architecture, a series of experiments were conducted to evaluate the feasibility of the framework. In this chapter, the main experimental implementations related to the integration of IOTA and cryptographic functions are described. For IOTA-related operations, the IOTA SDK and the IOTA Stronghold library were used in a Rust environment. The RSA and Fernet libraries in Python were used for the cryptographic functions.

4.1 Creating an IOTA account

The following illustration shows the process of creating an IOTA account.



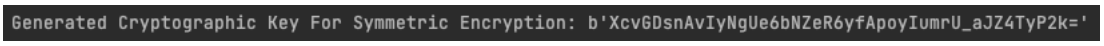
```
Generated mnemonic: "cube barely trash upon pumpkin arch food tackle maximum ugly isolate depart whale plate quit lamp swear pill unique fly company century casual oak"
IOTA Public Address: AccountAddress { address: Bech32Address(rms1qqc3vfkztxy90ken2dz9et30ejygy30gtp94fdd2jmvhd63pg9a7krjacpf), key_index: 0, internal: false, used: false }
```

Creation of an IOTA account

1. **Generation of the mnemonic phrase**
   A random mnemonic phrase is generated that serves as the basis for deriving a unique seed for the IOTA account. This seed is required to securely create or restore an IOTA account.
2. **Creation of the account**
   The mnemonic phrase is used to create a wallet and a stronghold file. The stronghold file acts as a local "vault database" that enables secure synchronization with the IOTA Tangle and allows the account to be restored if necessary.
   During account creation, an Ed25519 address is generated, which serves as the IOTA wallet address and is required for transactions on the IOTA Tangle.
3. **Verification**
   Unique IOTA accounts and addresses were created for User-A and User-B (IOTA Shimmer Testnet Explorer).

4.2 Creating a cryptographic key for symmetric encryption

A cryptographic key created using the Python Fernet library is required for level 1 encryption:



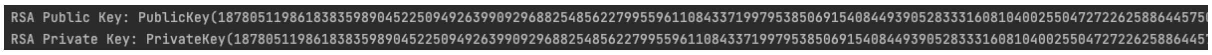
```
Generated Cryptographic Key For Symmetric Encryption: b'XcvGDsnAvIyNgUe6bNZeR6yfApoyIumrU_aJZ4TyP2k='
```

Creation of a cryptographic key for symmetric encryption

1. **Key generation**
   A symmetric encryption key was generated using the Fernet library.
2. **Security aspects**
   The generated key was stored securely and only passed on in accordance with the framework's access and verification protocols.

4.3 Creating a public and private key pair for asymmetric cryptography

The RSA algorithm, a widely used method, was used for asymmetric cryptography. The following figure shows the steps and results of generating a public and private key pair with RSA.



```
RSA Public Key: PublicKey(18780511986183835989045225094926399092968825485622799559611084337199795385069154084493905283331608104002550472722625886445750
RSA Private Key: PrivateKey(1878051198618383598904522509492639909296882548562279955961108433719979538506915408449390528333160810400255047272262588644457
```

Creation of a public and private key pair for asymmetric cryptography

1. **RSA key generation**
   A public and private key pair was created using the RSA algorithm to enable secure data exchange between User-A and User-B.

2. **Key distribution:**
   The public key was shared with the collaboration partners to ensure secure encrypted communication. The private key was kept securely by each user in order to decrypt files.

4.4 Creating files with level 1 and level 2 encryption

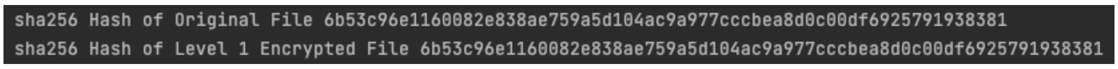
The file encryption process consisted of two stages to maximize data security:

1. **Level 1 encryption:**
   The original file was encrypted using the cryptographic key generated by symmetric encryption (Fernet).

2. **Level 2 encryption:**
   The level-1 encrypted file was additionally encrypted with the public key of user-B, which was generated during the RSA key pair creation.

4.5 Creating hashes of the original and encrypted files

To verify and ensure immutability, a hash was generated for both the original file and the level 1 encrypted file. The figure shows the steps and results of hash creation:



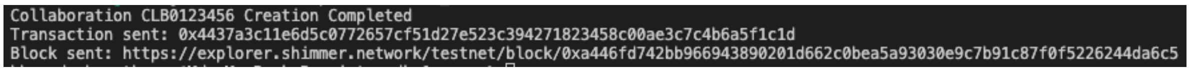
```
sha256 Hash of Original File 6b53c96e1160082e838ae759a5d104ac9a977cccbea8d0c00df6925791938381
sha256 Hash of Level 1 Encrypted File 6b53c96e1160082e838ae759a5d104ac9a977cccbea8d0c00df6925791938381
```

Creation of hashes of the original and encrypted files

Using the FileHash library and the SHA-256 hashing algorithm, unique hashes were created for the original file and the level 1 encrypted file.

4.6 IOTA transactions with metadata and verification of authenticity

In the following, five IOTA transactions were carried out, each containing metadata in order to map the process as described above. During the initial phase of collaboration establishment, User-B posts an IOTA transaction (Transaction-01) to User-A. This transaction contains the IOTA address of User-B, the collaboration acceptance signature, the public key and the collaboration ID. The transaction has been locked for 50 years to ensure the integrity and long-term availability of the transaction. The following image shows the details of Transaction-01 in the IOTA Shimmer Testnet Explorer:



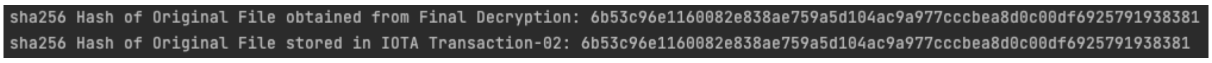
```
Collaboration CLB0123456 Creation Completed
Transaction sent: 0x4437a3c11e6d5c0772657cf51d27e523c394271823458c00ae3c7c4b6a5f1c1d
Block sent: https://explorer.shimmer.network/testnet/block/0xa446fd742bb966943890201d662c0bea5a93030e9c7b91c87f0f5226244da6c5
```

Collaboration acceptance signature transaction

To ensure that the exchanged files remain verifiable at all times and that their integrity is maintained, User-A saves the hashes of the original file and the encrypted files one after the other on the IOTA Tangle: User-A integrates the hash of the original file into transaction-02 and sends it to User-B. The hash of the level-1 encrypted file is split in transaction-03; both transactions were recorded on the IOTA Tangle and provided with time blocking conditions.

After "receiving" the level-2 encrypted file, User-B uses his private key to decrypt the level-2 encrypted file. User-B then posts transaction-04 to confirm the successful first decryption.
User-A then posts transaction-05, which contains the cryptographic key required for the final decryption by User-B. With this key, User-B decrypts the now level-1 encrypted file and receives the original file.

The test framework enables both users to check the authenticity of the file by comparing the current file hash with the hash stored in transaction-02:



```
sha256 Hash of Original File obtained from Final Decryption: 6b53c96e1160082e838ae759a5d104ac9a977cccbea8d0c00df6925791938381
sha256 Hash of Original File stored in IOTA Transaction-02: 6b53c96e1160082e838ae759a5d104ac9a977cccbea8d0c00df6925791938381
```

4.7 Discussion of the experimental implementation

The experiments to validate the proposed framework have shown that the concept can be successfully implemented. The results confirm that the combination of dual encryption, cryptographic procedures and the use of the decentralized structure of the IOTA tangle creates a secure and tamper-resistant basis for the exchange of sensitive files.

As intended, the decentralized architecture of the framework makes it possible to operate without a central server and still ensure a high level of trust between the collaboration partners. The use of proven cryptographic techniques ensures that files are not only securely encrypted, but can also be shared in a traceable manner. In addition, the immutability of the tangle provides permanent and verifiable documentation of the key steps in the exchange process.

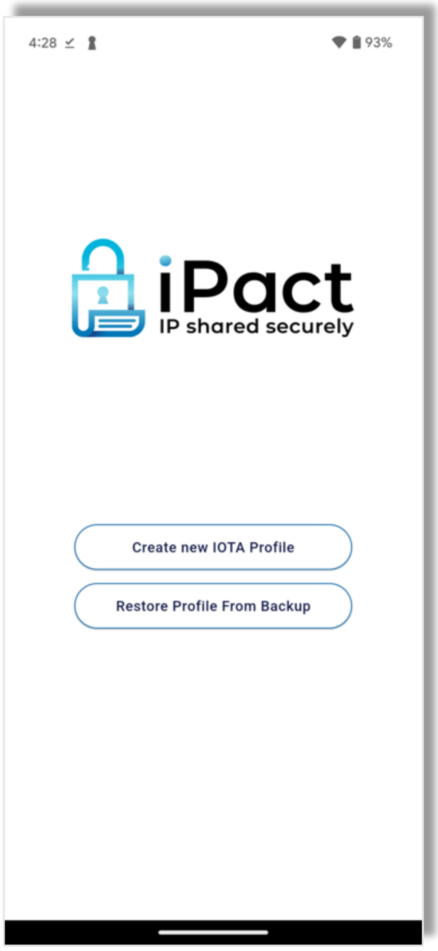
The experiments not only proved the technical feasibility of the concept, but also created a basis for the development of an app. This application is intended to transfer the functionalities of the framework into a practicable solution for end users.

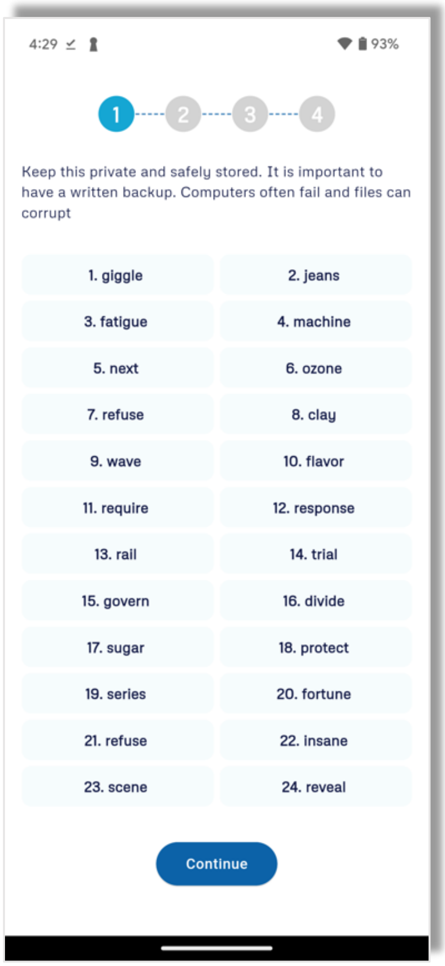
5. practical application results of the iPact smartphone app

The central functions of the iPact smartphone app include downloading the app from the Google Play Store (it has not been made publicly available, only for test users), creating a secure profile including the generation of an IOTA-based wallet, creating collaborations using invitations, secure encryption, transfer and decryption of files, and checking the immutability of shared files over time.

The installation and setup of the application begins with the download of the app via the Google Play Store. During installation, permissions are requested for file storage (for uploading and downloading files) and internet access to enable communication with the IOTA network. The creation of a user profile combines the setup of an account-specific access with the creation of a secure IOTA wallet.

This includes the generation of a new IOTA profile, which serves both as a user account and as a crypto wallet for the IOTA Shimmer testnet. The following screen shown on the left allows users to choose between creating a new profile and restoring a previously saved profile, e.g. after a device change.



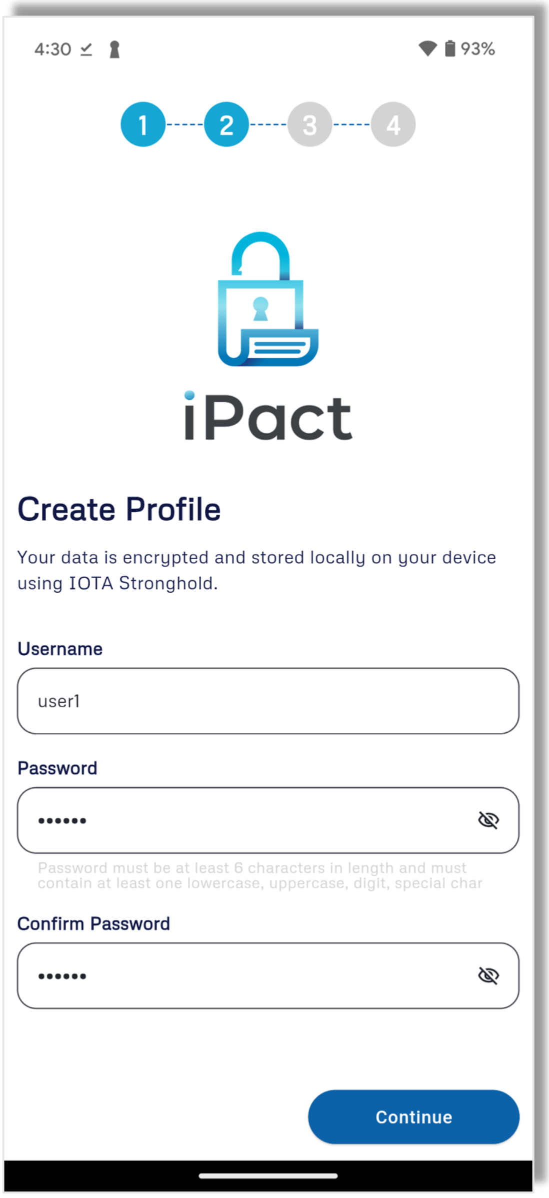


Keep this private and safely stored. It is important to have a written backup. Computers often fail and files can corrupt

1. giggle	2. jeans
3. fatigue	4. machine
5. next	6. ozone
7. refuse	8. clay
9. wave	10. flavor
11. require	12. response
13. rail	14. trial
15. govern	16. divide
17. sugar	18. protect
19. series	20. fortune
21. refuse	22. insane
23. scene	24. reveal

Choice: existing or new profile **24-word mnemonic seed generation**

The screen on the right shows how a unique 24-word mnemonic seed is generated during profile creation. Anyone who has ever used a crypto wallet to store cryptocurrencies such as Bitcoin, Ethereum or similar will already be familiar with this process. The seed is stored securely within the app's Stronghold ("local IOTA vault"). The IOTA Stronghold technology protects this seed from unauthorized access and encrypts it with a password set by the user. A public address is also generated as part of the profile creation process, which can be used for transactions and can be verified via the IOTA Shimmer Testnet Explorer.



Create Profile

Your data is encrypted and stored locally on your device using IOTA Stronghold.

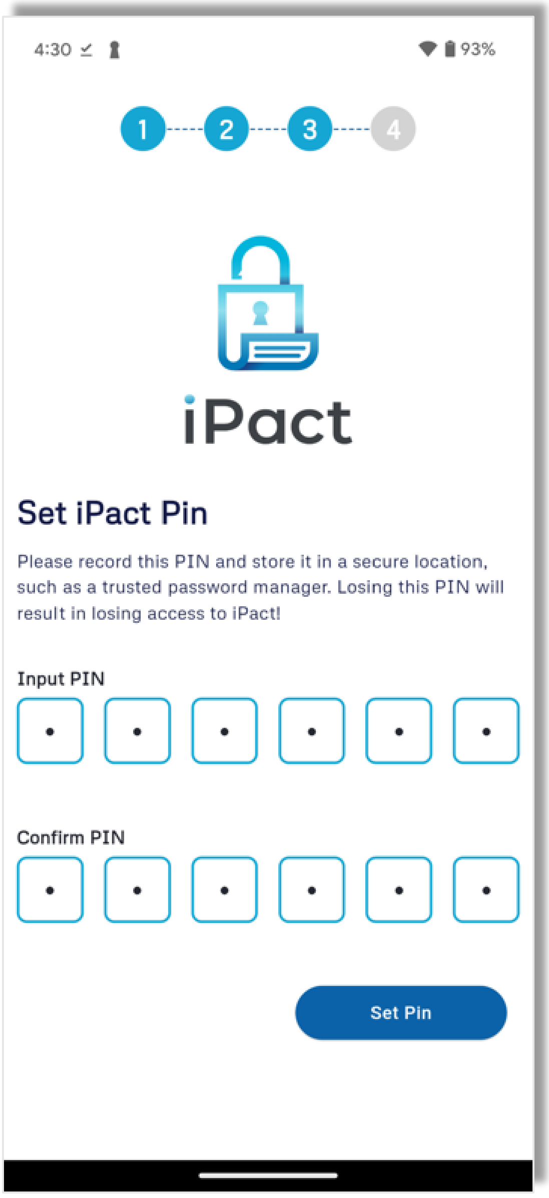
Username: user1

Password: ••••••

Password must be at least 6 characters in length and must contain at least one lowercase, uppercase, digit, special char

Confirm Password: ••••••

Profile creation/ Stronghold Password



Set iPact Pin

Please record this PIN and store it in a secure location, such as a trusted password manager. Losing this PIN will result in losing access to iPact!

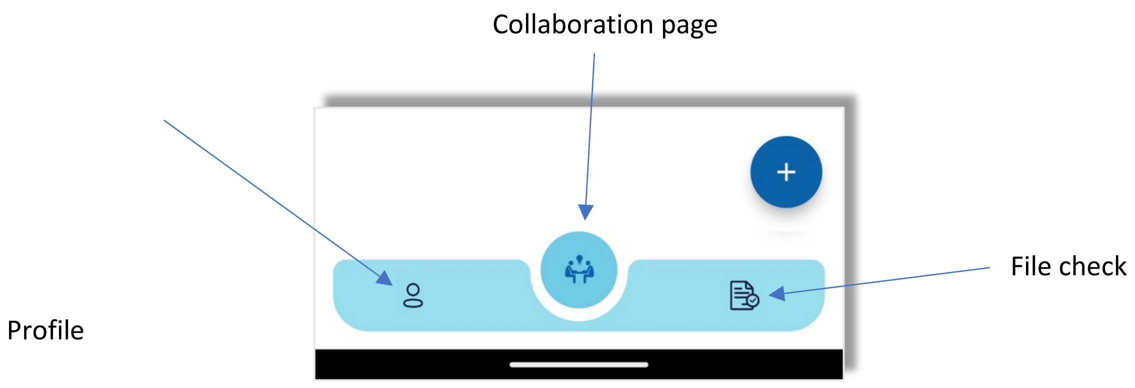
Input PIN: • • • • • •

Confirm PIN: • • • • • •

Setting the iPact pin

Users also set up two credentials: a PIN for logging into the iPact application and a Stronghold password, which is required for actions related to the IOTA network, such as accepting collaborations or encrypting and decrypting files.

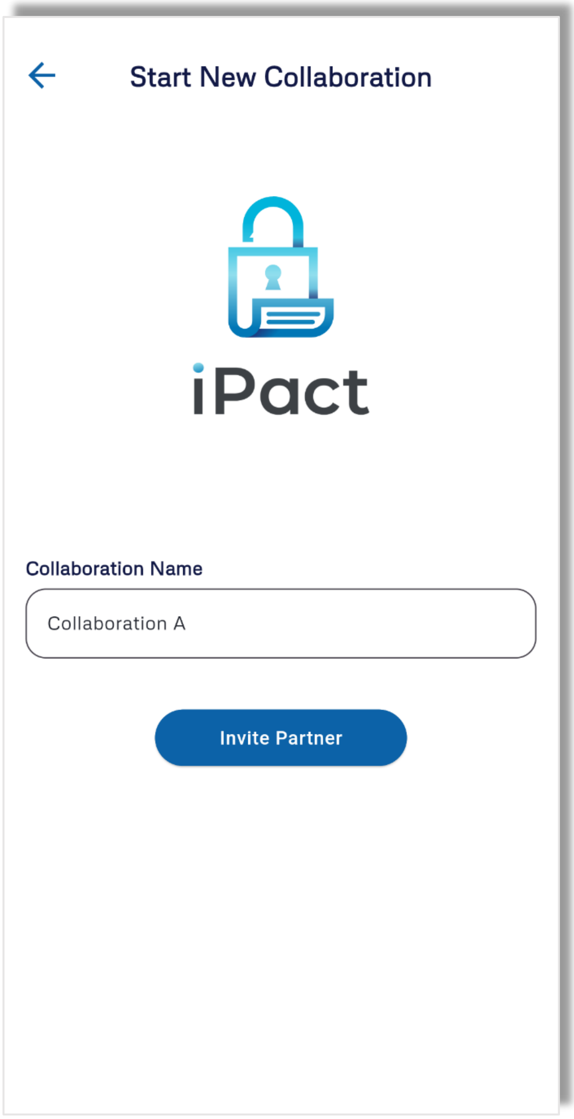
After logging in, users are taken to the main page of the iPact application, which consists of three central areas: profile page, collaboration page, file authenticity check page. The following screenshot shows these menu items located at the bottom of the screen. In this example, the tab showing the collaborations is activated.

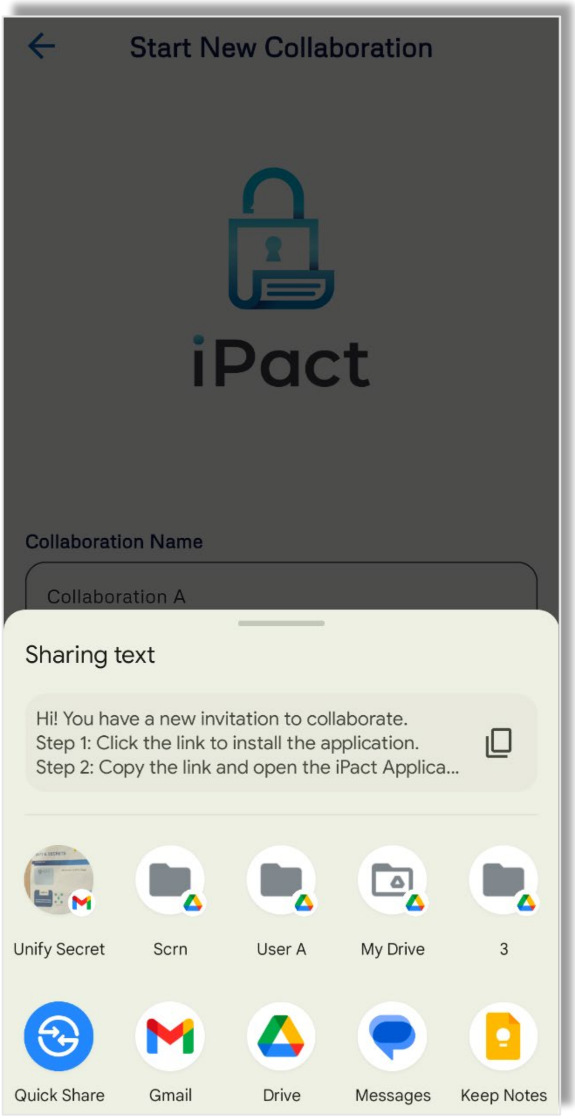


Access to the various menu items

Collaboration page

Collaboration management is at the heart of the iPact application, allowing users to securely share and encrypt files.

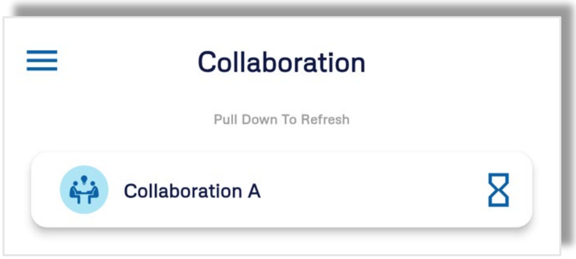


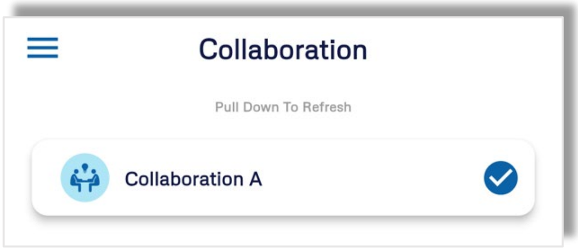


Hi! You have a new invitation to collaborate.
Step 1: Click the link to install the application.
Step 2: Copy the link and open the iPact Applica...

Creating a new collaboration **Sending the invitation link**

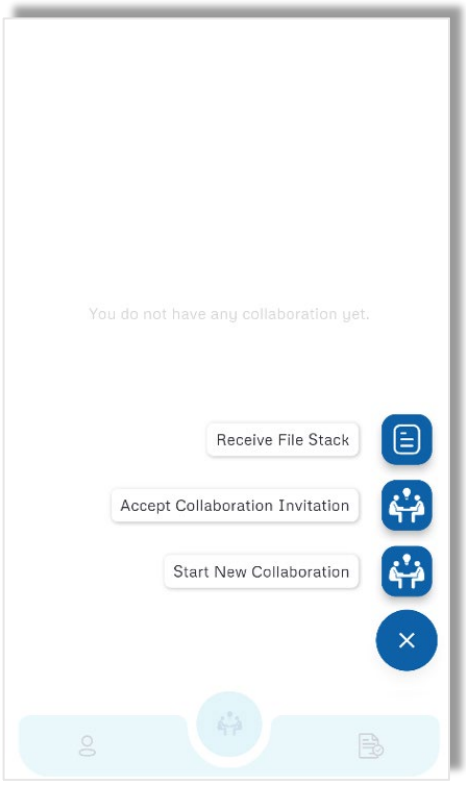
Before files can be exchanged, a collaboration must first be created between the users involved. This process begins with a user initiating a new collaboration, giving it a name and generating a unique invitation link (unique and unmistakable), which can then be shared with the desired partner. The invitation link can be conveniently sent via various communication channels such as email or messaging services. As long as the invitation has not been accepted, the status of the collaboration is displayed as "Pending". As soon as the recipient accepts the invitation, the status changes to "Completed" and both users can use the collaboration for the secure exchange of files.



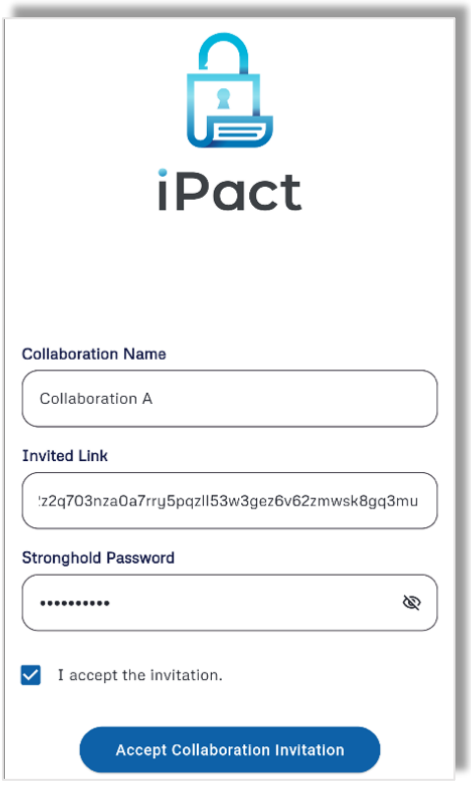


Acceptance is pending **Acceptance is completed**

After receiving the invitation, the invited user opens the iPact application, accepts the collaboration and gives the collaboration a name that does not have to be the same as the name chosen by the sender; there is a deliberate option here for each party to choose a name that makes sense for them.


You do not have any collaboration yet.
Receive File Stack
Accept Collaboration Invitation
Start New Collaboration

Collaboration" submenu



Collaboration Name

Collaboration A

Invited Link

:z2q703nza0a7rry5pqzll53w3gez6v62zmwsk8gq3mu

Stronghold Password

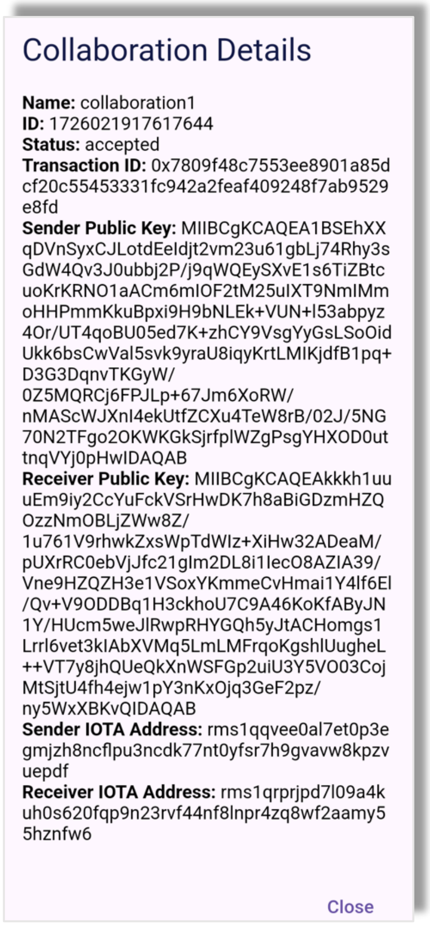
••••••••••

☑ I accept the invitation.

Accept Collaboration Invitation

Naming and acceptance by recipient

The invited user then inserts the invitation link received into the field provided and confirms acceptance by entering their Stronghold password. Once these steps have been completed, the collaboration is marked as complete and the user can add files to the collaboration. At the same time, the sender updates the collaboration overview to ensure that the collaboration has been successfully completed.



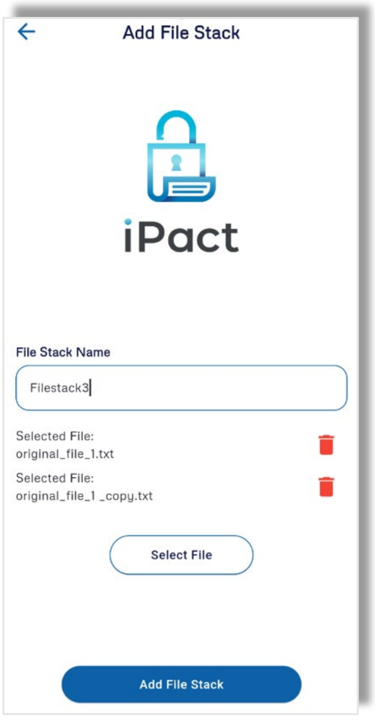
Collaboration Details

Name: collaboration1
ID: 1726021917617644
Status: accepted
Transaction ID: 0x7809f48c7553ee8901a85dcf20c55453331fc942a2feaf409248f7ab9529e8fd
Sender Public Key: MIIBCgKCAQEA1BSEhXXqDVnSyxCJLotdEeIdjt2vm23u61gbLj74Rhy3sGdW4Qv3J0ubbj2P/j9qWQEySXvE1s6TiZBtcuoKrKRNO1aACm6mIOF2tM25uIXT9NmIMmoHHPmmKkuBpxi9H9bNLEk+VUN+l53abpyz4Or/UT4qoBU05ed7K+zhCY9VsgYyGsLSoOidUkk6bsCwVal5svk9yraU8iqyKrtLMIKjdfB1pq+D3G3DqnvTKGyW/0Z5MQRCj6FPJLp+67Jm6XoRW/nMAScWJXnI4ekUtfZCXu4TeW8rB/02J/5NG70N2TFgo2OKWKGkSjrfplWZgPsgYHXOD0uttnqVYj0pHwIDAQAB
Receiver Public Key: MIIBCgKCAQEAkkkh1uuuEm9iy2CcYuFckVSrHwDK7h8aBiGDzmHZQOzzNmOBLjZWw8Z/1u761V9rhwkZxsWpTdWIz+XiHw32ADeaM/pUXrRC0ebVjJfc21gIm2DL8i1IecO8AZIA39/Vne9HZQZH3e1VSoxYKmmeCvHmai1Y4lf6El/Qv+V9ODDBq1H3ckhoU7C9A46KoKfAByJN1Y/HUcm5weJlRwpRHYGQh5yJtACHomgs1Lrrl6vet3kIAbXVMq5LmLMFrqoKgshlUugheL++VT7y8jhQUeQkXnWSFGp2uiU3Y5VO03CojMtSjtU4fh4ejw1pY3nKxOjq3GeF2pz/ny5WxXBKvQIDAQAB
Sender IOTA Address: rms1qqvee0al7et0p3egmjzh8ncflpu3ncdk77nt0yfsr7h9gvavw8kpzvuepdf
Receiver IOTA Address: rms1qrprjpd7l09a4kuh0s620fqp9n23rvf44nf8lnpr4zq8wf2aamy55hznfw6

Transparent information on technical details possible

Users can view the technical details of the collaboration if required. With the IOTA addresses, it is possible to find all transactions linked to these addresses in the Tangle Explorer. These addresses serve as unique identifiers that are assigned to transactions in the Tangle. The explorer can be used to view publicly stored information such as metadata, tags or hashes, but not sensitive or encrypted content, as it is not possible to control or view protected data without the private key. This transparent design of IOTA enables the traceability of transactions, even after many years, even if iPact as the application with which this data was written to the Tangle is no longer available. This essential information, such as the public IOTA address and the asymmetric public key, is also displayed on the profile page.

As soon as collaboration is active, files can be added and exchanged securely. The user creates a so-called "file stack" to which they upload one or more files. Each file is first symmetrically encrypted with AES as described above. Two transactions are noted on the IOTA Tangle by iPact: one for the hash of the original file and one for the hash of the symmetrically encrypted file.


Add File Stack

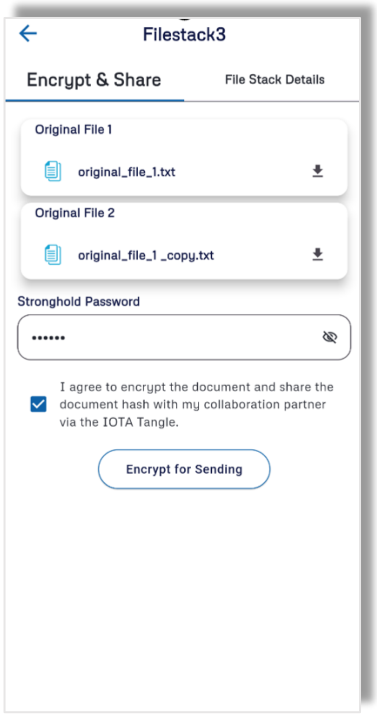
iPact

File Stack Name

Filestack3

Selected File:
original_file_1.txt

Selected File:
original_file_1 _copy.txt


Filestack3

Encrypt & Share | File Stack Details

Original File 1
original_file_1.txt

Original File 2
original_file_1 _copy.txt

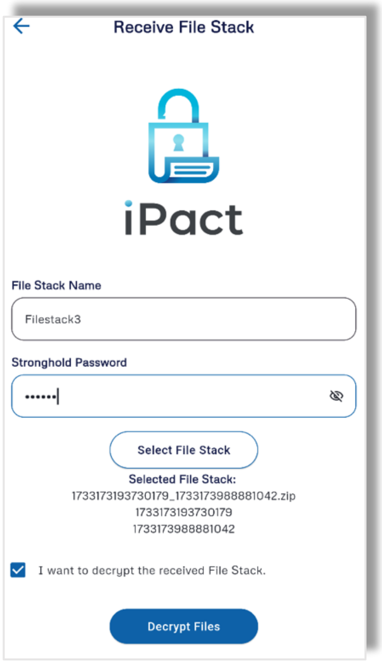
Stronghold Password
••••••

I agree to encrypt the document and share the document hash with my collaboration partner via the IOTA Tangle.

Sender: Create a file stack **Sender: Encryption before sending**

Once encryption is complete, the files protected from access are bundled and sent securely to the recipient as a convenient ZIP file via a conventional communication channel, similar to the sending of the invitation link described above.

On the recipient side, the decryption process begins as soon as the encrypted ZIP file has been imported into the iPact app.


Receive File Stack

iPact

File Stack Name

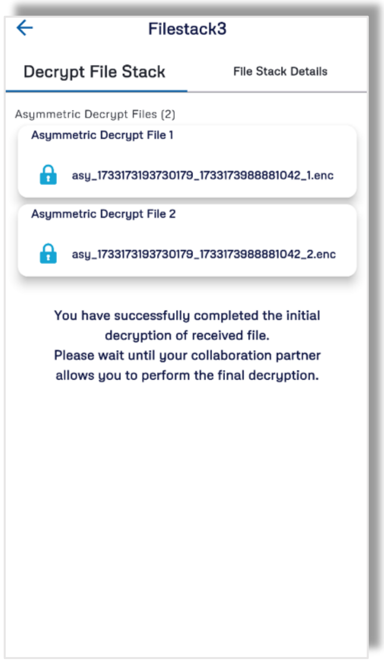
Filestack3

Stronghold Password

••••••

Selected File Stack:
1733173193730179_1733173988881042.zip
1733173193730179
1733173988881042

I want to decrypt the received File Stack.


Filestack3

Decrypt File Stack | File Stack Details

Asymmetric Decrypt Files (2)

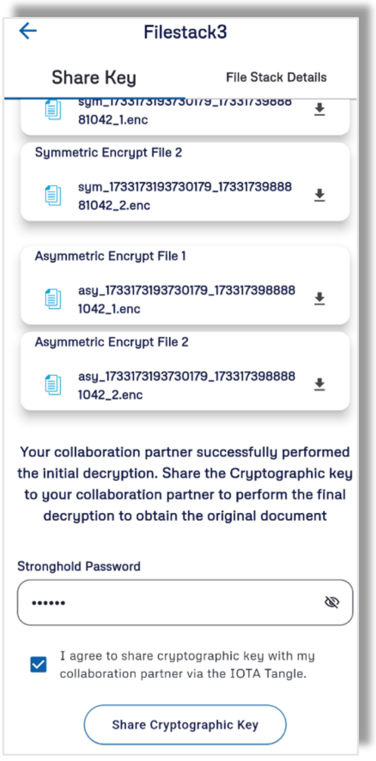
Asymmetric Decrypt File 1
asy_1733173193730179_1733173988881042_1.enc

Asymmetric Decrypt File 2
asy_1733173193730179_1733173988881042_2.enc

You have successfully completed the initial decryption of received file.
Please wait until your collaboration partner allows you to perform the final decryption.

Receiver: Decryption 1 envelope **Receiver: Decryption 1 envelope successful**

The recipient first removes the outer encryption layer with his private key and stores the files locally. A transaction on the IOTA Tangle documents this step and informs the sender of the successful first decryption step, as a prerequisite for the subsequent publication of the symmetric key by the sender on the IOTA Tangle


Filestack3

Share Key | File Stack Details

sym_1733173193730179_17331739888 81042_1.enc

Symmetric Encrypt File 2
sym_1733173193730179_17331739888 81042_2.enc

Asymmetric Encrypt File 1
asy_1733173193730179_173317398888 1042_1.enc

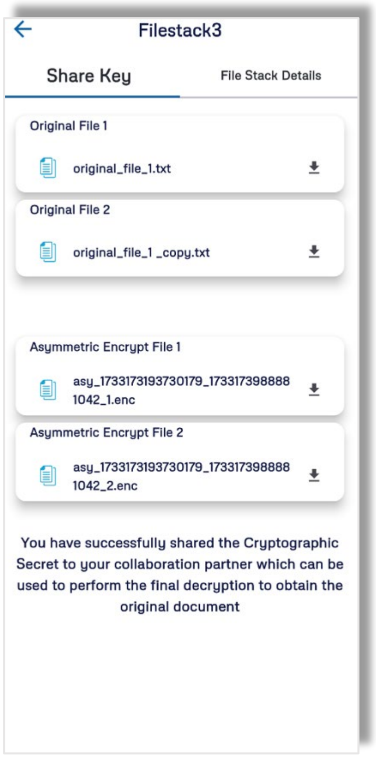
Asymmetric Encrypt File 2
asy_1733173193730179_173317398888 1042_2.enc

Your collaboration partner successfully performed the initial decryption. Share the Cryptographic key to your collaboration partner to perform the final decryption to obtain the original document

Stronghold Password
••••••

☑ I agree to share cryptographic key with my collaboration partner via the IOTA Tangle.

Share Cryptographic Key


Filestack3

Share Key | File Stack Details

Original File 1
original_file_1.txt

Original File 2
original_file_1 _copy.txt

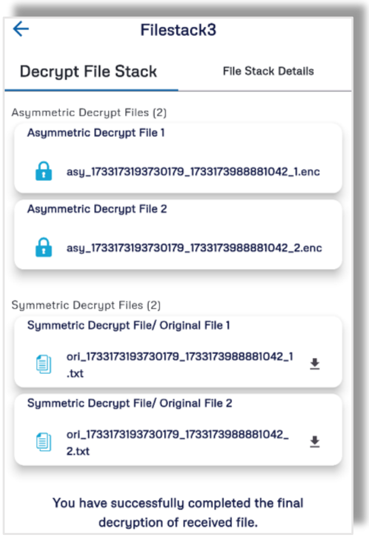
Asymmetric Encrypt File 1
asy_1733173193730179_173317398888 1042_1.enc

Asymmetric Encrypt File 2
asy_1733173193730179_173317398888 1042_2.enc

You have successfully shared the Cryptographic Secret to your collaboration partner which can be used to perform the final decryption to obtain the original document

Sender: shares symmetric key **Sender: share successful**

The recipient uses this key to remove the remaining inner encryption layer and make the original files fully accessible:


Filestack3

Decrypt File Stack | File Stack Details

Asymmetric Decrypt Files (2)

Asymmetric Decrypt File 1
asy_1733173193730179_1733173988881042_1.enc

Asymmetric Decrypt File 2
asy_1733173193730179_1733173988881042_2.enc

Symmetric Decrypt Files (2)

Symmetric Decrypt File/ Original File 1
ori_1733173193730179_1733173988881042_1 .txt

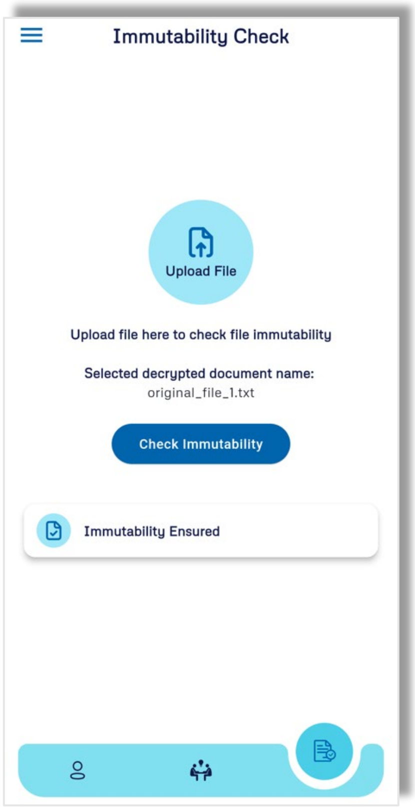
Symmetric Decrypt File/ Original File 2
ori_1733173193730179_1733173988881042_ 2.txt

You have successfully completed the final decryption of received file.

Recipient: final decryption successful, the exchange process is complete

Verification of file authenticity

As already described, the process for checking immutability is used check whether a shared file has been manipulated or not, possibly many years later. To do this, the user first opens the tab for checking immutability and uploads the file to be checked. The "Check immutability" function is then started, which compares the hash of the uploaded file with the hash stored on the IOTA Tangle.


Immutability Check

Upload File

Upload file here to check file immutability

Selected decrypted document name:
original_file_1.txt

Check Immutability

Immutability Ensured

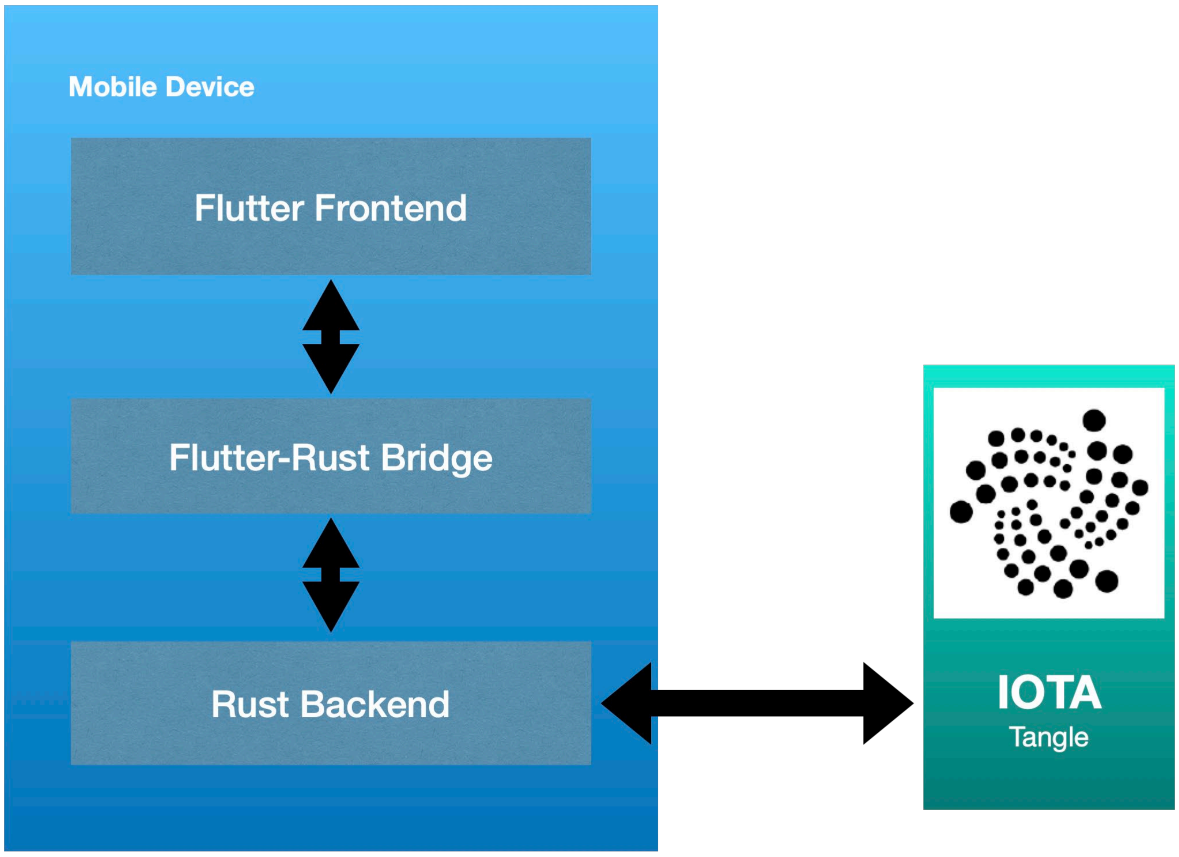
"Immutabillity Ensured" - this is the original file

If the file is confirmed as unchanged, a corresponding message indicates that the file has remained unchanged and its integrity is therefore guaranteed.

The following section briefly describes the technologies and software packages used in the development process.

6. overview of software technology used

The iPact application combines various front-end and back-end technologies, with the Flutter and Rust technologies playing a key role alongside IOTA. The technologies used, the architecture and the cryptographic functionality are briefly presented below.



Overview of the software architecture

Flutter, a framework developed by Google, was used for front-end development. It enables cross-platform app development for Android and iOS with a single code base.

The backend of iPact was developed in Rust, a programming language that was originally initiated by Mozilla and is particularly known for its combination of resource efficiency and security and is therefore well suited for security-critical applications, which is why the IOTA Foundation also uses Rust.

The Flutter-Rust Bridge connects these two technologies and enables seamless communication between the frontend and backend. This bridge allows complex cryptographic calculations and transaction operations to be transferred efficiently between the user interface and the backend. In this architecture, the IOTA Tangle acts as a distributed ledger that immutably stores data such as transactions, metadata, file hashes and keys.

Front-end implementation and cryptography library
The front end was developed entirely in Flutter using the Dart programming language. The frontend's main functions include registration, setting up secure profiles, establishing collaborations, encrypted file exchange and checking the immutability of files.

A central element of the front end is the in-house developed cryptography library EncryptionUtils, which provides functions for symmetric and asymmetric encryption. The library supports the generation of RSA and AES keys, the encryption and decryption of files and integrity checks by hashing with the SHA-256 algorithm.

The library offers comprehensive support for generating and managing keys. This includes functions such as generateRSAkeyPair, which generates RSA key pairs (public and private), and functions such as encodeKeyToPem and decodePemToKey, which can be used to securely store and restore keys in PEM format. These PEM formats are widely used and enable simple key management across different platforms. For symmetric encryption, the generateKeyFileForSymmetricCryptography function enables the generation of secure AES keys, which are then stored in an encrypted file.

The library supports both asymmetric and symmetric approaches for data encryption and decryption. With RSA-based functions such as rsaEncrypt and rsaDecrypt, sensitive data can be securely encrypted and only decrypted by authorized recipients. The symmetricEncryptFile and symmetricDecryptFile functions offer efficient methods for AES-based encryption, which is particularly suitable for larger files.

Another important aspect is file management and hashing. The createHashFromFile function generates SHA-256 hashes, which serve as unique digital fingerprints of files and ensure the integrity of the data. In addition, functions such as writeEncryptedFile and readFileAsBytes enable the secure storage and retrieval of files in encrypted format. These functions are specially designed to make the management of sensitive data secure and user-friendly.

The cryptography library is fully integrated into the user interface and is synchronized with the backend via the Flutter-Rust Bridge. This allows cryptographic operations to be carried out efficiently and securely at both the user and backend development level.

Backend API implementation in Rust

The backend of iPact was implemented entirely in Rust. It uses the IOTA SDK to manage transactions on the Tangle and consists of two main modules: api.rs and wallet_custom.rs. The api.rs module controls network interactions and enables the creation of advanced transactions with features such as time locks, metadata and storage depots. The wallet_custom.rs module provides advanced wallet functions, including the secure management of stronghold data and the creation of conditional transactions.

A particular challenge was the integration of new IOTA libraries such as the IOTA SDK, which replaced previous libraries such as IOTA Wallet and IOTA Stronghold. This transition required extensive testing and customization, but resulted in improved functionality and security. The Flutter-Rust Bridge enabled efficient communication between the Rust backend and the Flutter frontend, allowing complex operations such as transaction creation and processing to be performed securely.

7 Conclusion and outlook

As part of iPact's research and development, the Induko project has created an innovative, decentralized application that can be used to document the exchange of files between researchers and organizations in a tamper-proof manner - without a third party that would otherwise have to be trusted, such as a notary. By combining robust cryptographic techniques with the immutable nature of the IOTA Tangle ("blockchain"), this framework provides a novel, trustworthy digital support for an important aspect of collaborative work with confidential and sensitive data.

The use of dual-layer encryption, which uses both symmetric and asymmetric methods, not only ensures that files are only accessible to authorized users, but also enables robust and tamper-proof execution of the so-called handshake through the innovative combination with the possibilities of IOTA. The decentralized architecture promotes peer-to-peer data exchange and ensures that the exchange of sensitive information is documented in a tamper-proof manner.

Working on iPact was an innovation process with numerous challenges. In the end, we were very pleased not only to have developed a functioning concept, but also to be able to demonstrate that such a solution is even possible in a user-friendly way by implementing an application as a dApp entirely on IOTA Layer 1. The fact that the second key is only transferred to the recipient once the sender has received the corresponding confirmation from the recipient means that there are additional intermediate steps that would not be necessary when simply sending an encrypted file. On the other hand, the proofs that are crucial for verifiability and security are achieved without the need for a central authority and without a great deal of additional effort - especially when you consider that many files can be exchanged at once.

Although the goals set have basically been achieved, there are numerous approaches for further activities and developments:

- **Extensive user tests and elimination of vulnerabilities**
  A comprehensive test with a larger group outside the circle of developers was not possible within the project duration.
- **cryptography**
  In order to ward off future threats from quantum computers, the framework is to be expanded to include quantum-safe algorithms such as lattice or hash-based cryptography.
- **Extension of the supported data types**
  Currently only optimized for text files, the functionality was to be extended to images, videos and complex data sets in the future. This extension was postponed because it is more of an implementation task that is not central to the core function and could then no longer be implemented during the project period.
- **Automation with Flutter Deep Linking**
  Manual processes such as invitations or file transfers could be automated and made more user-friendly through deep linking.
- **Transition to the IOTA 2.0 network**
  The use of the IOTA Shimmer testnet is to be replaced by the production-ready IOTA 2.0 network, which is to offer greater scalability, more decentralization ("Coordicide") and security . After the end of the project, however, another extremely fundamental technological change to IOTA was surprisingly announced in November 2024. In future, the platform will no longer be based on the Tangle, but on MoveVM and technology from SUI. As

with the IOTA Tangle, this will continue to be a DAG and not a blockchain; MoveVM was developed by Meta (Facebook) and is considered to be very innovative and secure. SUI was operated by former Meta developers as a further development after Meta discontinued the development of its own cryptocurrency. For iPact, this change means that the backend development would have to be completely adapted to this new technology.

- **Multi-user and cross-platform support**
  The optimization of the framework for multiple users (not only 1:1, but also 1:n or n:n collaborations) and iOS devices.