

# KEYSTAMP

---

## An open-source Proof-of-Compliance standard on the blockchain

---

by Francis Pouliot, with the collaboration of Shayan Eskandari

Written during RegHackTO hackathon, November 27th 2016

Special thanks to:

- Jean-Philippe Beaudet
- Arthur Gerbelot
- Patrick Poirier
- Philippe Chevy

Presented under the team Existence

## Abstract

---

We propose an new open-source standard, Keystamp, for integrating applied cryptography and blockchain technologies in existing corporate processes and commercial relationships, such as compliance policy implementation and audit. Specifically, we design an innovative set of policies and frameworks for using those technologies which allows financial services participants to generate and verify irrefutable proofs that specific individuals or organizations possessed information, performed actions or signed messages at a certain point in time. Keystamp is designed to be deployed within organizations, and is not decentralized.

It is a protocol for participants to agree on how the authenticity, validity and integrity of factual data is determined. Participants will agree on the Keystamp protocol because it is easily implemented and can be audited independently by participants or third-party observers which are provided with the relevant data.

**Proof-of-Compliance** is the output of compliance-related data being processed by using the Keystamp standard. Compliance related data can consist of anything the participants believe is relevant, such as KYC forms, SMS verification logs, video recordings, emails, affidavits, contracts, etc.

Proof-of-compliance is a digital trail of unforgeable cryptographic signatures, immutable timestamps and encrypted contextual data. At each step of the compliance process, data is gathered, digitally signed, timestamped and encrypted. The decryption of the data, its the validation integrity, the authentication of tis signer and the timestamp's existence can be instantly performed for auditing purposes. Pseudonymous cryptographic keys are associated to legal identities in a reference index, maintained in a centralized or distributed database.

## Purpose and impact

Keystamp is a Proof-of-Knowledge institution for the digital economy, since this open-source system can be used by anyone, for free, to prove that he possessed any data at any time. The concept is designed to have a sophisticated aggregation of various blockchain-related technologies with an easy to use, graphical interface and a general-purpose open API. These incredibly powerful tools can be used by the public not only to take control of their identity but also to achieve complete trust through the audit-able and self-validating nature of the technologies used. As the process of education the public, starting from penetration at the top level of the institutions which are the ultimate beneficiaries within their firm of a Keystamp implementation.

It can be used for mediation in civil disputes (contracts) by the general public since proofs are easy to verify. The purpose is to foster trust through disruptive technologies. We aim to increase social cohesion in the digital economy, providing an entirely digital institutional framework of trust for the digital economy.

Keystamp has the potential to enormously decrease the costs of compliance and audit for financial institutions, technology providers, governments, and any institution that needs trust and hierarchical authorizations. It increases transparency and saves money to the taxpayers. Because Keystamp operates independently from any other system and rather connects via API or using the a web application, it is easy to implement in an existing production workflow without changing the system used, as long as the system used can communicate data to the Keystamp app or API.

## Blockchain and crypto technologies

- Public key encryption using Bitcoin's Elliptic Curve Digital Signature Algorithm
- The Hierarchical Deterministic public key infrastructure of Bitcoin (BIP32)
- Cryptographic hash functions
- Timestamps using OP\_RETURN Bitcoin transactions (Blockchain)
- AES encryption
- (In development) Open-timestamp infrastructure
- We introduce BIP32 derivation standard for linking keys to identities

## Implementation

The process is to transform data part of a process, such as internal policy compliance in a commercial interaction, and transform it into proof that can be validated and audited by both parties or third parties. As such, the implementation is a set of applications and real-world contracts where participants agree on a protocol to follow, whose output, the Proof-of-compliance, is mutually agreed to be a valid system to provide and validate evidence.

We specifically implement the Proof-of-Compliance use-case of Keystamp in the context of "know-your-customer" policies, or any other context where establishing that information was obtained by certain parties at a certain time is required. Proof-of-compliance provides a digital trail of unforgeable signatures and timestamps which consist of irrefutable evidence that someone had certain information, took action or participated in specific events at specific points in time. It is a protocol which turns data into tamper-proof legal evidence. Keystamp was designed, as an application, to be used by organizations - specifically financial institutions, government institutions and large corporations. Heads of the hierarchy in the organization issue certificates to subordinates (and so on) which grant them the ability to sign and encrypt data.

Since issuance and revocation of contracts are enforced cryptographically and are publicly audit-able, easy to authenticate and verify using graphical interfaces over complex cryptographic libraries, anybody can verify who they are talking to and where they derive their authority from in the context of a business relationship. It can be implemented by public bodies such as government and enforcement agencies, so as to limit fraud and foster public trust.

A functional API can integrate easily in any existing traditional workflow, with graphical easy-to-use tools to foster implementation. Since Keystamp aims to be a standard, adoption needs to be implemented voluntarily by a large number of firms in the economy to be successful. Users can also implement Keystamp in their workflow manually using a modern web application.

## Context and use-case

Keystamp was designed during the RegHackTO hackathon organized by the Ontario Securities Commission. The task of Keystamp was to respond to the following challenges:

- Ensure that know-your-customer policies have been implemented
- Ensure that investors have the proper knowledge they require
- Help investigators validate proper forms were filed, disclosures signed, etc.
- Remove ambiguity and deniability

Our use-case revolves around three types of financial relationships that are under the purview Ontario Securities Commission:

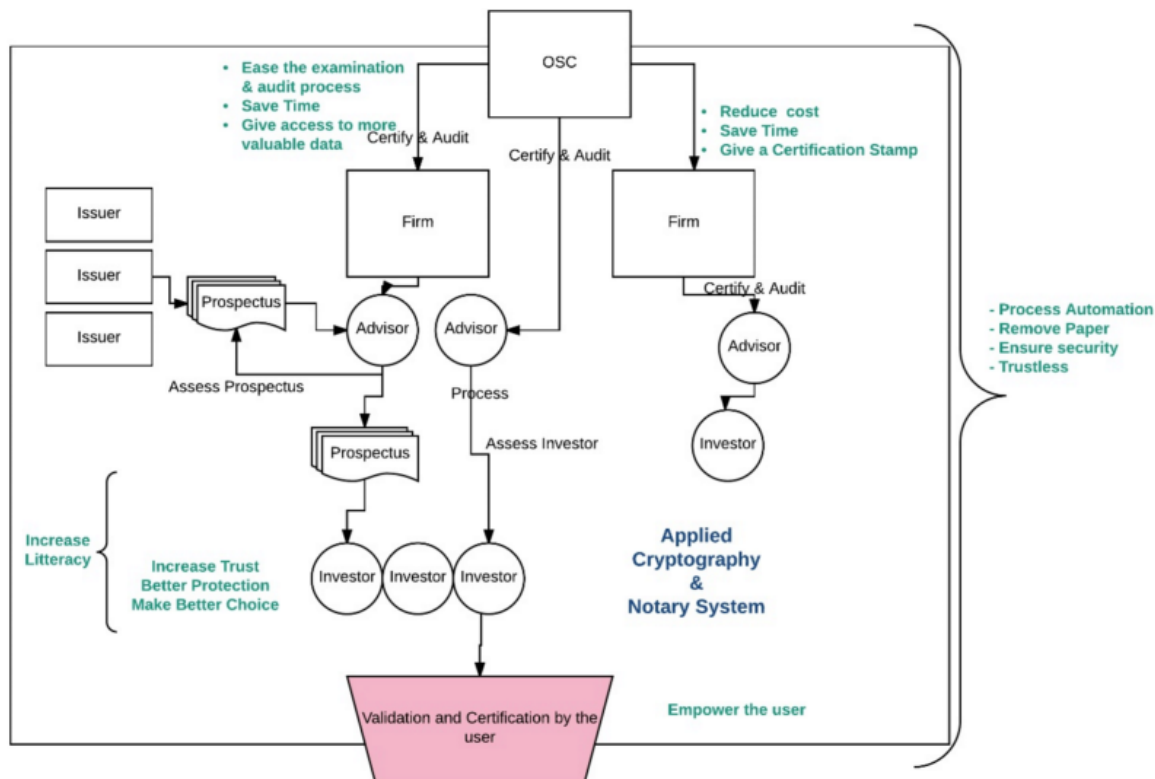
- The “know-your-customer” process initiated by financial advisors for assessing the risk profile of their investing clients.
- The delivery of data by the producers or sellers of investment products to financial advisors.
- The assessment of the risk profile of the investment products by financial advisors at time at which advice is given to investing clients.

We want to achieve the following goals:

- Issue license, permits, certifications or delegate authority in a hierarchy
- Provide proof that the KYC process was performed by financial advisors (e.g. risk assessment of clients)
- Provide proof of the risk assessment of a security, and the context of its recommendation to the client
- Provide proof that investors understand their risk assessment and the security they are purchasing
- Verify the integrity of data, detecting any tampering or forging of signatures
- Associate data to individuals to provide "Proof-of-Knowledge"
- Encrypt data for reporting purposes

Compliance data consist of any data used to prove that certain compliance policies were followed. This includes KYC, due diligence, internal audits, release form, consent forms, contracts, etc. In many cases, procedures were followed in person or over the phone. We envision organizational processes where one's perception of events, as well as factual data, is presented to the relevant parties which can "sign-off" on the statement and evidence of the other party.

### Internal / external compliance, monitoring and reporting with Keybstamp



We identify 5 main actors in our specific use-case for compliance process:

1. The regulator: the entity which assigns keys to firms and which is responsible for monitoring the compliance processes of firms.
2. Firms: they employ financial advisors and report to regulators.
3. Issuers: they sell securities to financial advisors (can be brokers) and report to regulators.

4. Advisors: they recommend securities or purchase securities on behalf of their clients and report to both the firms and the regulators.
5. Investors: they employ financial advisors and ultimately purchase securities and can complain to firms and regulators.

In this specific use-case, we see that there is a hierarchical relationship between the OSC and the advisory firms (because of the compliance requirements). The firm enjoys additional rights than its employees or subordinates (e.g. financial advisors, affiliated brokers), for which it generates additional sub certificates.

These sub-certificates are associated to the firm's identity in a public registry, which is common amongst financial regulatory regimes. For examples, numerous jurisdictions have Money Service Businesses and licensed Financial Advisor public registries.

These firms can then derive child keys for individual, subordinate members of the organization, and so on, according to parameters that are defined by the issuer and/or according to parameters they set themselves, depending on the program design. These child keys are then used for signing data and encrypting data.

Our two main use cases are:

#### **Financial advisors**

They will provide all the evidence of a KYC process and respect of certain policies to the investors that they are advising. For example, this can be a risk assessment form and the the conclusions drawn by the financial advisor. For instance, one can imagine a statement by the advisor declaring that he believes his client to be in a high risk profile, meaning that the securities he will recommend will be high risk, and that there is significant chances of financial loss. The advisor will include whatever process he used to come to this conclusion (such as an audio recording or form), and the investor will acknowledge that this accurately reflects reality using a cryptographic signature. To remove some complexity, the acknowledgement can be done using traditional remote KYC methods. Proof of these KYC methods, such as the reference ID and application log of a SMS verification event by a trusted service provider, is included in the data bundle that later becomes the Proof-of-Compliance.

The data is then encrypted by the parties using the private key of the signature as encryption password. The encrypted data bundle is hashed and published on the Bitcoin blockchain.

#### **Financial product issuers and sellers**

They will digitally sign the information about the security they are issuing / selling to the buyer, in our scenario a broker or financial advisor working in a larger firm, which we will call the "Agent". The information may include a prospectus, disclosure, risk assessment, statement of intentions, disclaimer, etc. The agents will then sign this data along with a statement of acknowledgement that the information was received and understood. The data is then encrypted by the parties using the private key of the signature as encryption password. The encrypted data bundle is hashed and published on the Bitcoin blockchain. It is then easy to prove that certain information was shared and acknowledged by specific parties at a specific time, and the evidence is tamper-evident (meaning that any tampering of the data or signatures is detected instantly).

The information may include a prospectus, disclosure, risk assessment, statement of intentions, disclaimer, etc. The agents will then sign this data along with a statement of acknowledgement that the information was received and understood. The data is then encrypted by the parties using the private key of the signature as encryption password. The encrypted data bundle is hashed and published on the Bitcoin blockchain.

## **Keystamp Process**

---

In short, the process includes:

- 1) Generating keys
- 2) Assigning keys to subordinates
- 3) Signing data
- 4) Encrypting data
- 5) Arranging hashes and signatures into "data proof" bundles
- 6) Encrypting data
- 7) Timestamping data

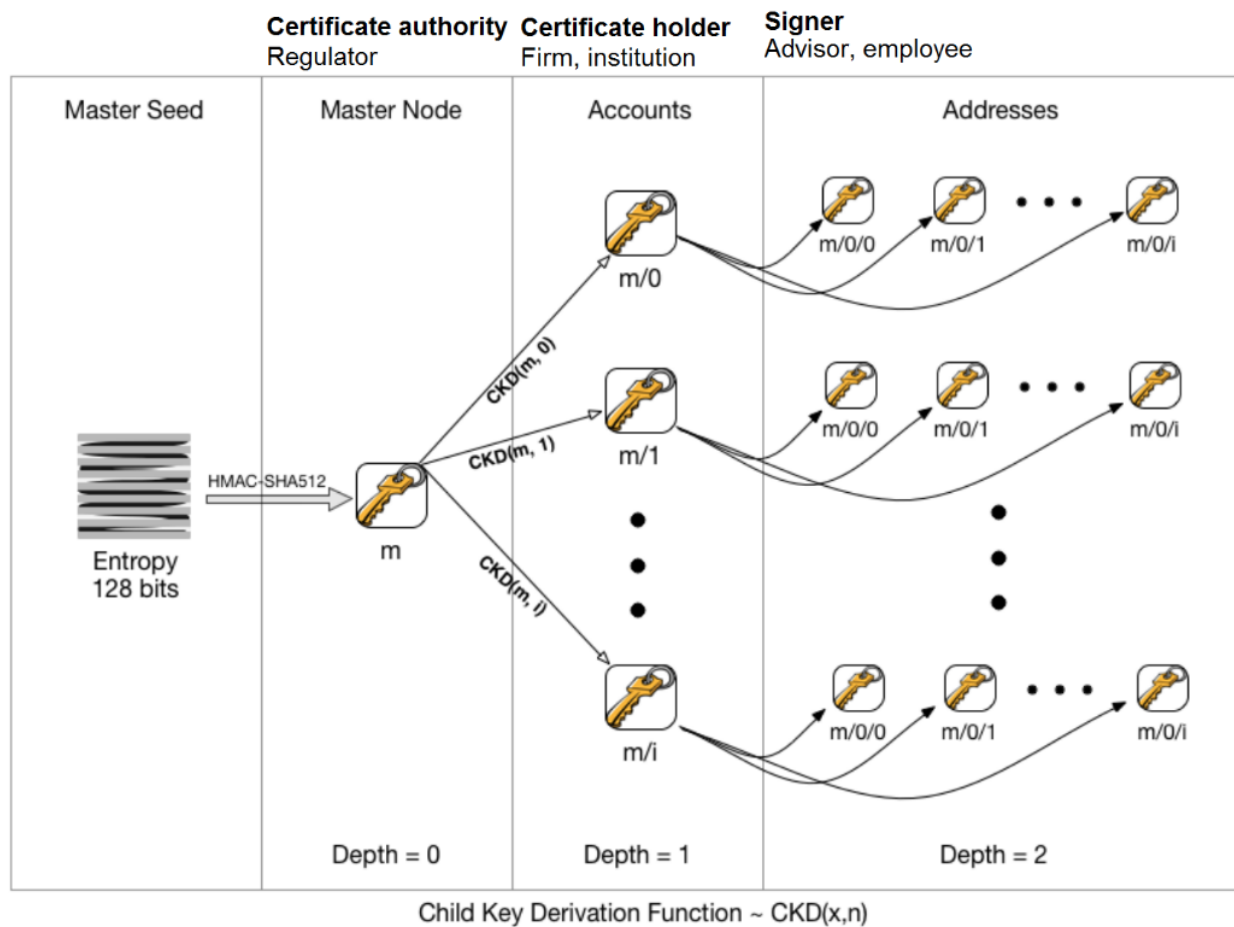
Below are the detailed steps with notes.

## Generation and issuance of cryptographic keys

1) The issuing authority generates a master seed and derives its master key using the Bitcoin BIP32 standard. It is from this key that all subsequent keys in the infrastructure are derived. The issuer of a master has a complete overview of all the keys that are subsequently derived. The issuer can re-create any of the keys derived by its subordinates, and as such has access to all the private keys in the key hierarchy

2) The issuing authority generates (derives) child keys for its master key, which are assigned by the issuer to each of the "subordinates" in the hierarchy. These keys can be also conceived as "certificates".

We recommend to use "hardened" derivation for all the child keys. This way, individual keys or certificates can be exported validated without necessarily exposing other keys.



## Key derivation paths

3) We use a specific key derivation path which allows us to identify specific keys to IDs in our database.

`m / issuer / firmID / agentID / signatureID`

Issuer: the Issuer generates a master seed and derives its master key FirmID: first level of derivation by the issuer




AgentID: lower level of derivation initiated by the firm

Using this derivation path, we can easily associate certain keys to their legal ids.

In the example below, we can see that:

m = the master key generated by the issuer 623 = the ID of a firm 231 = the ID of an agent 423 = the number of the key used by the agent in a specific operation

## MASTER KEYS

 Root HD Private Key	xprv9s21ZrQH143K4EhS1ashEYM34cgYrz7eEtvKmnz095yq9Btqtq3e958pgYae5VK	
 Root HD Public Key	xpub661MyMwAqRbcGimu7cQhbgHmceX3GSqVc7qvaBCcMUcxhwX3PS9JBwPcfyLQzTSxUNT	

## DERIVED KEYS

Path

**Path: m/623'/231'/423**

[^ LESS INFO](#)

 <b>HD PRIVATE</b>	xprv9ysoy84rMpyaXksHSUQD5tVB2zRfE52EQaFzNpox2FSpnaqUXw7TYJ6hGsKLM9h3CesmopXCt
 <b>PRIVATE KEY</b>	0b1d58228e0fa6050b92a3cf01bc85f5c5ed0289cd8827a3e6a654e28ee6fd51
 <b>HD PUBLIC KEY</b>	xpub6CsANdbkCCXskEwkYVwDT2Rub2G9dXk5moBbBDDZaayofPAd5URi66RB8BsmYhKXG2cNRh/
 <b>PUBLIC KEY</b>	03dbd29a58a911990fc89f5d263722eb6e994e8613bd8b656a10cb4c46fef9da77
<b>ADDRESS</b>	1MzTfDAKT9HSPVJ2MahfAXqTf8RyvRufTR

**Path: m/623'/231'**

[v MORE INFO](#)

**Path: m/623'**

[v MORE INFO](#)

**Path: m**

[v MORE INFO](#)

Customization: the issuing authority could decide not to derive hardened keys, which means that all the certificate holders would be aware of each other and thus track each other's transactions on the blockchain and be able to verify each other's signatures.

**Note: the derivation tree can have additional levels for more complex arrangements**

The Keystamp standard is not unlike the BIP44 standard used by Bitcoin wallets, which standardizes the derivation path as:

```
m / purpose' / coin_type' / account' / change / address_index
```

This goes to show that the specific organizational structures can be encoded as BIP32 standards such as BIP44.

4) Each institution (or person) such as an advisory firm derives extended public keys, again hardened or not, for each other subordinate, such as individual employees and advisors of an advisory firm. These are referred to as "agents", meaning that they do not have the capacity to issue keys but only to sign.

5) Registries of public keys are kept, either internally or public, depending on the settings. This allows signature to be publicly associated to certain keys, meaning that anybody can independently verify a signature. If a firm or issuer is

associated to a public key in a public registry, members of the public can independently verify that a certain firm has been accredited by the regulatory authority, as long as the firm is using the Keystamp protocol.

## Steps to verify a signature

- When you receive a signed message, the signature appears as an extra bit of text
- You input this signature, along with the registered key of the signatory and the original text in the verification API
- You can verify the legal identity of the signatory and the authenticity of the signature and message

6) Each agent uses a new key, derived from his extended key, to cryptographically sign documents and compliance data. Private keys are used each for only one "transaction" or event in the recorded database.

## Compliance data

7) Compliance data is collected by the participants in the scheme. It consists of:

- contextual data
- proof that KYC process was followed
- Due diligence, disclosures, release forms, consent forms, fee tables, risk assessment forms
- Recorded video and audio
- Email and chat exchanges
- Application logs
- Statement of intentions, contracts, affidavits
- Any potential evidence The end user provides his consent that he agrees that the information presented by the advisor accurately reflects the KYC and disclosure process that was completed by the advisor by either digitally signing, validating with phone sms verification or any other traditional remote KYC methods.

## Hashing process

8) A data bundle including the compliance data as well as the end-user's consent proof (signature or SMS validation log) is hashed using SHA-256. The data and the hash is "signed" by the end-user, either using a private key or using traditional KYC methods, that can include:

- SMS 2FA verification logs
- DocuSign references
- Recorded interaction
- Any remote KYC or identity method leaving evidence of process

9) The hash is signed by the private key of participants that wish to prove that he has knowledge of the data.

10) At the end of the process, all the hashes are bundled together and hashed into a "final hash" which is the one timestamped in the Bitcoin blockchain. If any of the input data resulting in one of those hashed has been tampered with, the final hash will be changed. At this point, we can test every individual hash to find out which one was tampered with.

```
Final hash = SHA256(data hash, signature1, signature2)
```

## Encryption of data

11) Since we are using ECDSA Bitcoin keys that cannot by themselves encrypt documents, we rather use a derived private key as the encryption password for AES encryption. The data is encrypted with that same private key, whose derivation path displays the source of the signing authority and to which internal document this key is attached to.

```
XPRIV + DERIVATION PATH OF CHILD KEY = PRIVATE KEY = AES DECRYPTION PASSWORD
```

12) The encrypted data can be stored in distributed storage networks such as IPFS, Mailsafe, or secure cloud storage. For privacy, the index or registry of legal identities to derivation paths (cryptographic identities)

## Timestamping

A blockchain timestamp is the act of including certain data as metadata in a Bitcoin transaction. We can prove that whatever data is included at that time existed at the time that this transaction was mined into a block in the Bitcoin blockchain. The data we add in a transaction is publicly available. However, because hashes and timestamp are only used to validate the integrity of data, they are meaningless unless someone has the original data (and thus are not by default breach of privacy, although privacy is an issue).

13) The signed hash is stored in the Bitcoin blockchain using OP\_RETURN.

We first create a Notarization string using our own header KEYSTAMP:

**KEYSTAMP:HASH**

1. The string is included in a Bitcoin transaction using the OP\_RETURN opcode. The address from which the transaction is sent is a Notarizing address. Ideally, the notarizing is a multisignature address where signing keys are divided amongst the hierarchies. This function can be used to create certain access control logic for timestamping functions within the organization.

Let's imagine that an organization has 4 levels of derivation in a large corporate, such as:

**m / Firm board or CEO / Executives / Managers / Agents**

The Notarization address for Agents could be a 2-of-2 scheme with the other signature being that of the Manager. So for a timestamp to be valid, it must be signed by both the manager and the agent. Any scheme can be implemented, which allows for greater flexibility in design cryptographic access controls to the timestamping functions.

As long as the parameters are agreed upon in a Keybase implementation between the prover and the authority which is meant to be convinced of the proof (e.g. superiors our outside observers such as regulators).

One address is used to better keep track of the OP\_RETURN timestamps on the Bitcoin blockchain. It is from this address that the OP\_RETURN transactions are broadcast. This address is associated to a participant in the Keystamp implementation in a private or public registry.

## Keeping a timestamped record

In the Keystamp process, it is important that certain keys (and thus certain derivation paths) are associated to internal identities. Recall that the derivation path is:

**m / issuer / firmID / agentID / signatureID**

This allows anybody with access to the signatures and derivation paths to prove that certain data was signed by a certain key at a certain time. However, they are only associated to legal identities because the derivation paths correspond to coordinates in the Keystamp process user's internal database.

As such, it is crucial that the database be hashed, signed and timestamp whenever there is a new entry. The issuer can thus modify the registry, revoking keys for certain legal identities or rights. Keys that are compromised can be removed from the registry.

The hash of the registry is timestamped and saved along with an archive cache of the registry at that time, so that it can be proven that a certain signature by performed by a legal id by looking at the official registry of the organization at that time.

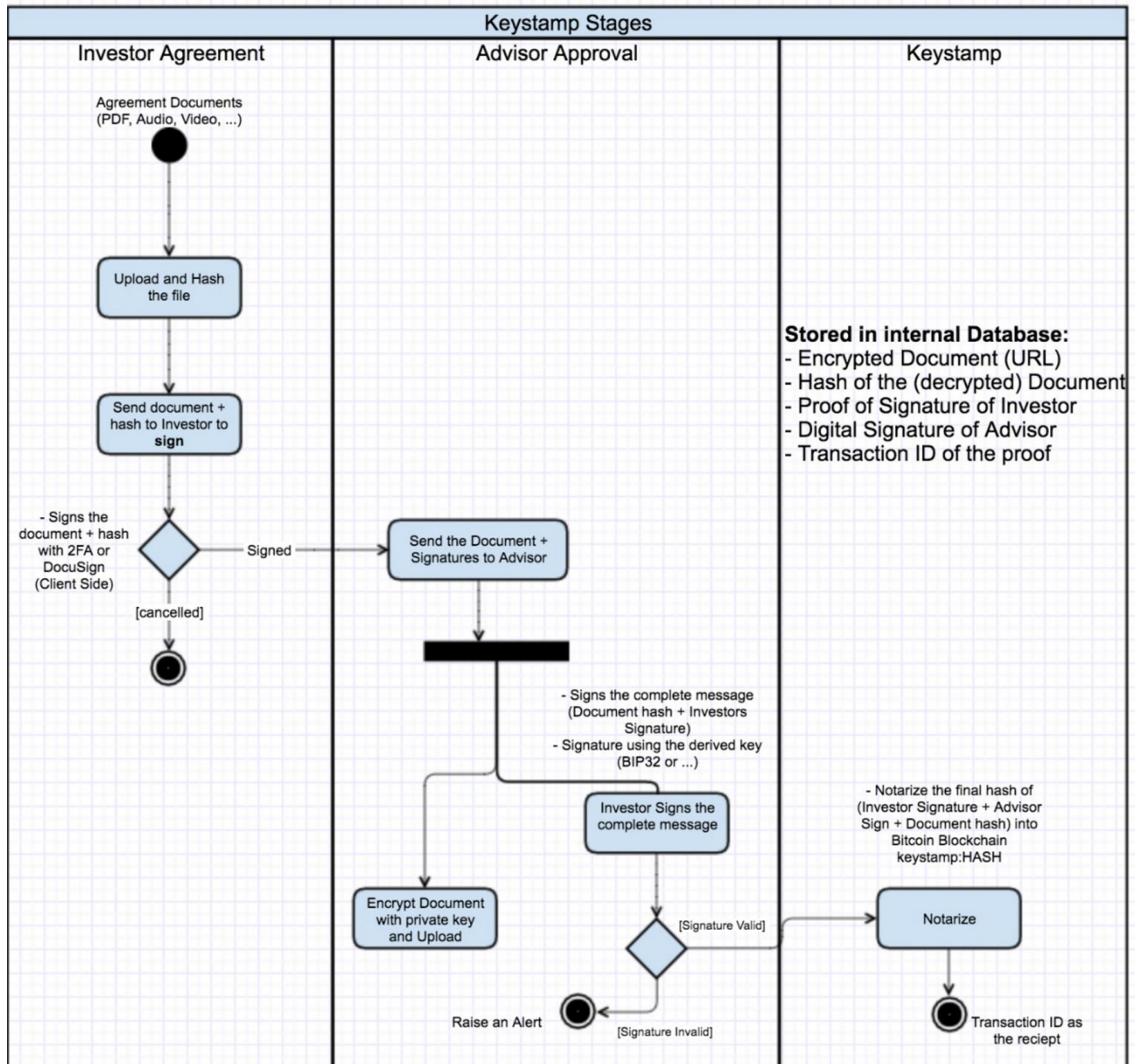
## Proof-of-compliance data

The Proof-of-Compliance is the output of the Keystamp process. It is a protocol for generating and auditing tamper-evident proofs. Amongst the data provided is:

- Hash of the decrypted data in the internal database
- Encrypted data
- Key derivation paths



- Key derivation paths = map to private key to obtain the encryption password
- Decrypted data
- Digital signatures
- Public keys of signatories
- TransactionID
- OP\_RETURN format (KEYSTAMP:HASH)
- Block height in Bitcoin blockchain
- Time according to the miner's timestamp server



## Audit and validation

### Auditing the timestamp

Requirement 1: we are trying to determine someone signed a piece of data at a certain time. To start, we need to possess the data the is trying to be proved.

Requirement 2: we also need the signatures which are being presented as evidence that the data was signed by them.

Requirement 3: finally we need the derivation paths of the keys used, which will give the auditor the public key corresponding to the private key that was used to sign and encrypt the data.

To audit:

Step 1: Extracting the hash of proof on the blockchain by looking up the transaction ID in which was included the timestamp  
Step 2: Compare with the hash of the evidence that is presented to audit

```
Final hash = SHA256(proof hash, SHA256(signature1), SHA256(signature2))
```

Step 3: If the hash in the blockchain is the same hash as the final hash of the data and signatures presented, we know that the integrity of the data and signatures is preserved. If not, then we know that the data has been tampered with. If a single byte of data is changed in the compliance data or the user consent proof, the hash will be modified unpredictably. By comparing the compliance hash of the original document on the blockchain with the compliance hash that is presented, we can immediately prove that they are the same (or not), and so that it existed at the time of the block in which it was included.

Step 4: If the data has been tampered with, we separate the hashes and check them individually. We will be able to figure whether it is the data or signatures that have been tampered with.

### Validating the signatures

To validate the signatures, one only needs to have the public key of the signatory, the signature and the data. We can independently check that the signature would have been performed by the private key from which is derived the public key.

### Cross-checking against the registry timestamp

Once it has been established that a certain private key signed a specific set of data at a specific time, we need to associate the public key with an identity. We can see from the transaction ID which included the OP\_RETURN which keys were used to sign off of the timestamp, so we can suspect that the superior signer knew that the subordinate signer was who he claimed he was.

However, in order to make sure, we can look at the history of changes of the registry and since each change is hashed we know that the registry data at time of transaction broadcast has not been tampered with.

## Innovation and significance

### Extending BIP32 for encryption

One of the novelties of Keystamp is the innovative use of the BIP32 key architecture to encrypt data. Indeed, Bitcoin keys can only be used for signing. However, private keys can be used as encryption password for other algorithms such as AES. By using keys as passwords, we allow superiors in the Keystamp hierarchy to have access to data encrypted by its subordinates or agents. It's a one way function. As such, only the issuing authority has access to all the data, and permissions are hierarchical, perfect for institutions such as financial regulators.

This creates a system for cryptographically-enforced access permissions. The derivation path of the key which used to sign the data and encrypt the data.

XPRIV + CHILD DERIVATION PATH = DECRYPTION KEY

We believe that this function of BIP32 is unique and adds value to Keystamp, and is thus superior to other schemes such as PGP.

### Extending BIP32 for cryptographic certificates

We test the implementation of Bitcoin BIP32 in a public key infrastructure. By associated the derivation path of keys to public registries, we can allow selected participants to independently verify where certain keys are derived from, i.e. where they get their signing authority and who can decrypt their data.

### Multi-signature notarization signatures

We implement a system by which notarizations are valid only if the transaction is broadcast from a multi-signature address. This allows participants of the Keystamp process to specify schemes where the agents lower in the hierarchy request their superiors to co-sign. In order to revoke the right of the agent to notarize certain information and be recognized as valid evidence, for instance if it is believed that the agent's key is compromised, the superior just refuses to co-sign and the agent cannot broadcast its valid proof.

### **Future development: blockchain assets (meta-protocols like Colu)**

The same private keys can be used to issue or receive any blockchain asset, from securities to obligations, gift cards, vouchers, etc. When those innovations become mainstream, the validation of transactions will be trustless perfectly synced with cryptographic identities. The master private keys can be exported from a Keystamp implementation and imported in a Colu implementation. We can prove beyond doubt that a certain asset was generated or received by the signatory of in a Keystamp process. This proof is irrefutable.

### **Implementation infrastructure components**

- [An API for all the crypto functions](#)
- Abstraction layer for complex cryptographic libraries (ease of integration)
- Gateway connection to blockchain P2P network through full nodes
- Public key infrastructure application
- Policies for generating and issuing keys
- Registry (centralized or decentralized) for associating keys to legal identity
- Traditional KYC methods for assessing user identity online
- A graphical interface for internal and regulatory compliance showing the map of all key trees

Data kept in internal database:

- Encrypted data
- Decrypted data hash
- Signatures
- Transaction ids
- Registry of identities

---

### **More Information and resources:**

Keystamp Client: <https://github.com/existencelabs/keystamp-client>

Keystamp API : <https://github.com/existencelabs/keystamp-api>

Fully Functional Backend API for Notarization and Validation of Keystamps

Keystamp-Crypto: <https://github.com/shayanb/keystamp-crypto>

Whitepaper: <https://github.com/existencelabs/keystamp-whitepaper>

Contact [francis@existencelabs.com](mailto:francis@existencelabs.com) [phil@existencelabs.com](mailto:phil@existencelabs.com)