Challenges and Solutions for the MLETR

The TraaaX by KeeeX white paper

May 2024

Abstract

This document analyses the technical difficulties involved in implementing **UNICITRAL's MLETR**, or "Model Law for Electronic Transferable Records", that aims to digitize the entire Trade Finance sector by providing an acceptable digital substitute to centuries old paper-based processes. The MLETR is currently being implemented by a number of countries (Singapore, UK, France). We identify **five sticking points**, of which cyber security, user experience and functional coverage, which are broken down in **ten challenges**: document fraud resistance, exclusive control, unicity, resilience, crypto and key friendliness, interoperability, coverage, versioning and transfer approval. All these issues are matched with the proper technical solution as offered by KeeeX alone or the **TraaaX by KeeeX MLETR Control Tower**.

Author

Laurent Henocque and the KeeeX Team

Note1: this document may involve trade secrets and work covered with patents either granted or under reviewing process.

Note: this document cannot but name highly technical notions (up to cryptographic hashes, Bitcoin anchors, EVM blockchains, ERC721 smart contracts) and references advanced KeeeX technology and tools. It however aims at being readable by any person informed about the MLTER or curious about its consequences. Contact us for more if needed.

Introduction and Context

Huge amounts of paper clutter and slow the entire maritime trade and associated trade finance sector. The UNCITRAL has produced the MLETR (ie. Model Law on Electronic Transferable

Records) to help Trade Finance turn digital by removing legal uncertainty in the hope that viable and secure implementations can exist.

The whole problem amounts to replacing paper with digital. But not any kind of paper. What is dealt with here is paper producing the possibility for its owner to exercise a right. For instance, to claim a good (the Bill of Lading will be used to pick a surrendered container upon arrival) or to receive a payment (Promissory note). It is easily understood why digitizing these is difficult. This is just the same as digitizing a bank note, while ensuring that it cannot be used twice.

Preventing the double use of a digital resource sounds contradictory since everything digital can be almost infinitely replicated. It may well be one on the most important challenges that the software industry ever has had to solve since its inception. Doing so in the scope of international trade is even more complex since it operates at a global scale.

The first global system invented to prevent the double usage of a digital asset under acceptable security conditions is Bitcoin. Bitcoin invented how to prevent the double spending, and it happens that the key ideas involved (or the blockchain itself) can be leveraged to prevent the double usage of any digital asset.

Bitcoin at the same time introduced the self-sovereign identity of participants who pseudonymously exist via their capacity to digitally sign a monetary transfer or transaction and who retain the exclusive control over their actions as soon as they have never leaked their private key.

This cryptography comes at a cost, and the MLETR is very demanding in terms of preventing fraud or misuses when replacing digital by paper. In the long run, digital may prove better than paper since even the best watermarked paper can be tampered with (think about banknotes) and not a digitally signed file (under proper conditions).

Legal implementations of the MLETR have coined the idea that the concept of "exclusive control" was a fair replacement of physically holding a document. Although the intent seems clear the notion remains vague in its consequences, and software implementations must put this to work. Indeed, blockchain occurs as one of the most promising technologies to use in that scope, since Bitcoin invented a "digital signature based exclusive control" scheme that succeeded to stand since late 2008 in a context of extremely high hacking intensity. Blockchain however now comes in many flavors.

The 5 Sticking Points

The complexity of digitizing the Trade Documentation and Trade Finance can now be carefully analyzed. Our experience leads us to know that an absolute barrier to acceptance in entering a digital interaction is the fear that someone might be able to cheat. This is one among five other sticking points:

1. **Cyber Security**: must help every participant trust that nobody can cheat
2. **User eXperience**: no complexity should add to what people are used to
3. **Function coverage**: must implement all that the paper currently does

4. **Legal acceptance**: this is what the MLETR is about
5. **Market makers acceptance**: in the Trade Finance industry this relates with the P&I and banking sectors

The rest of this paper will focus on points 1 to 3. Considering 4, governments are in the ongoing process of passing laws that implement the MLETR. Considering 5, the MLETR provisions the requirement that a participant cannot reject a document on the sole basis of its digital nature. Furthermore, the implementation decrees will describe the requirements for a "reliable method". Matching these requirements will foster a solution's acceptance.

Note that regarding the notion of a "reliable method", many decrees develop criteria for an actor to be "qualified", which will pose difficulties with the use of Bitcoin or public blockchains.

The 10 Challenges

We have come up with a list of 10 prominently standing challenges pertaining to sticking points 1 to 3. We understand that any viable solution must implement:

1. **[Cyber] Document fraud resistance**: forbid anyone to counterfeit or tamper with a document
2. **[Cyber] Exclusive Control**: forbid unauthorized use of a document
3. **([Cyber] Unicity**: forbid any double use of the same document.
4. **[UX] Resilience**: make sure no process can be stopped by a participant having lost their signing key or login password
5. **[UX] Crypto friendliness**: if blockchain is used allow the system to be used without the participants owning crypto currency themselves
6. **[UX] Key friendliness**: remove the complexity to manage signing keys (although using full-fledged blockchain)
7. **[Function] Interoperability**: allow for interaction with different solutions or portals
8. **[Function] Coverage**: allow for addressing the entire Trade document bundle and the relationships between these documents
9. **[Function] Versioning**: enable provably accurate access to a document's version status and possibly to its verifiable latest version
10. **[Function] Transfer approval**: a user may decline receipt of a document and the system must not stall in case of a missing response

No priority order is set in this list as we think that all challenges must be addressed by a solution or a combination.

The rest of the document will attempt to detail and solve these issues or challenges.

Sketching the Solution - TraaaX by KeeeX

We place ourselves in the context of leveraging the KeeeX technology (https://keeex.me) and more precisely its implementation within TraaaX (https://traaax.com). This summary section can be skipped by a non-tech reader in a first pass. A tech reader may decide to stop reading there.

Put briefly under those settings:

1. **Documents are made unforgeable and forever verifiable** for free by being keeexed (details follow)
2. **Documents can only be used by the owner of exclusive control**: the person or entity that can sign with the owner's signature
3. **Documents can never be used twice** on the same registry, since « using » them involves transferring them to a new party in control or sealing them. Documents cannot either be used on distinct registries since an unforgeable bidirectional link is set between the keeexed document and any registry involved in tracing the changes of some variable property.
4. **Users belong to groups**; the rights being attached to a group and not a single user. This prevents a user losing their credentials to compromise the process.
5. **Proxy smart contracts on a publicly auditable blockchain** own the crypto and pass through genuine user signed transactions after verification.
6. **« Invisible » encrypted wallets are deciphered at user login**. The wallets remain under exclusive user control and private keys should never be sent to a server.
7. **API access, back to paper** and even transfer from one registry to another achieve outstanding interoperability
8. **Embedded links to other documents and blockchain mapping** smart contracts enable document linking to any level.
9. **A versioning smart contract complements embedded linking to previous versions** to allow for bidirectional traversal of the version graph and instant knowledge of latest version
10. **ERC721 smart contracts provision for delegation**, a process that can be used natively to achieve transfer acceptance and resilience to missing action.

Document Fraud Resistance - Keeexing details (1)

KeeeX exploits CNRS patents EP2949070 and US10218714 that describe the possibility to implement self-verified files in ways independent from the original file format, plus several additional features. In our implementation the keeexing process involves several layers of proof:

- **Integrity** controlled by an embedded multi-decade cryptographic multihash called an IDX. Basically, the IDX is a hash computed over the document plus most of KeeeX metadata including the signer public keys and registry references and excluding the IDX itself and signatures. Tampering with any byte from 0 to end of file is detected.
- **Authenticity or provenance** controlled by one or several digital signatures that range from Bitcoin ECDSA signatures to level 3 qualified X509 eIDAS signatures. Since the signer or delegator public keys are covered by the IDX, a document cannot be signed by two distinct entities.
- **Time** certified by the combination of System time, a legal (RFC3161) timestamp, and when applicable by signed GPS timestamps. In situations that deserve it the RFC3161 timestamp can be issued by a eIDAS qualified time server, and can be injected in the file for standalone verifiability.

- **Existence** proved by anchoring on a blockchain ledger, with merkle proofs made available for embedding in thus files themselves, thus guaranteeing tier less and standalone verifiability. KeeeX defaults to Bitcoin anchors for the durability of proofs and their limited impact thanks to large daily pooling.
- **Traceability** by referring inside the document to the blockchain ledgers involved in tracing the values of variable properties (for instance in the MLETR context a document's completion status, latest version etc.)

Altogether these elements enforce that a keeexed file is unforgeable and

- Can be verified from byte 0 to end and no addition is possible
- Can be verified using extremely generic code also available from the web archive and documented
- Can thus be verified forever without business model, user capture or trusted tier
- Proves its provenance according to extremely flexible schemes that include delegation
- Cannot be re-signed
- Cannot be traced on several distinct registries
- Allows for bi directional mapping between registries and the file itself to account for tamperless variable properties.

Exclusive control (2)

Exclusive control replicates the physical ownership of a paper document. This feature is required to make sure that the rights attached to a transferable document may only be exercised once and may be transferred along the owner chain. Technically, the requirement is the same as was originally addressed by the inventors of the Bitcoin blockchain: prevent double spending of a digital asset.

Even though the laws and associated decrees may choose to enable the use by implementors of "non-blockchain" registries as a "reliable method" we shall focus here on blockchain solutions and smart contracts since they best describe the challenges and expected properties as they were invented to address the very same issues.

Exclusive control requires:

- a "publicly" **auditable blockchain** so as to prevent any party to deny evidence. Here being public may be restricted to access from a consortium, although end users may favor trustless solutions.
- a **blockchain smart contract** that uses document IDX and or hashes as tokens so as to implement by directional linking to and from the document
- a **"transferOwnership"** function in the smart contract that transfers the token to a public key and makes the owner of the matching private key the sole person capable of signing a subsequent operation. Such a function is implemented in ERC721 EVM smart contracts for instance and used for transferring NFT ownership.

It can be recalled that the "exclusive control" challenge contradicts any intuition regarding digital assets, since digital is made for being replicated. Replication is not an issue (and may turn into an advantage) for non-transferable items but yields complexity when a digital document grants the bearer (owner) to exercise a right (of being paid, being delivered a good etc.).

TraaaX by KeeeX addresses this by a number of smart contracts on the KeeeX Chain publicly auditable EVM blockchain (https://blk.keeex.me).

Private or consortium based non-blockchain registries might be used for this, but "exclusive control" cannot be separated from the possibility by a participant to perform cryptographic operations **under their sole control**, however complex this may be (using a software or hardware wallet, NFC card, Fido2 key…).

An issue remains: what if there is a choice of several distinct registries/smart contracts?

Unicity (3)

Unicity covers two notions:

1. a document must not be traced in several distinct registries or distinct smart contracts within the same registry. In KeeeX's case this is made impossible by the fact that the document's IDX is computed by considering the registry (blockchain) where ownership is traced,
2. even within a single registry, the document cannot be used twice: this is a native property of blockchains, provided the blockchain offers sufficient protection against rollback. Any non-blockchain implementation must achieve this in an auditable way.

According to eIdas 2.0 some actors may probably accept that a registry be "non-blockchain" as soon as it implements "a tamper proof electronic record of data, providing authenticity and integrity of the data it contains, accuracy of their date and time, and of their chronological ordering". The requirement for some degree of decentralization is beyond scope in this formulation and is beyond scope as well in the general Trade or even Industrial context, where fighting against censorship is not an issue. Blockchain solutions have proven superior to more traditional approaches (e.g. database based) at guaranteeing the expected properties.

Resilience (4)

By enabling transfer of ownership to a group and not a single user the system is natively able to resist loss of access to signing keys by a participant. When so desired, the provision of one-person groups remains, meaning that standard behavior can be implemented (where a user losing their keys also loses their asset).

In the current TraaaX by KeeeX system, a group smart contract is responsible for letting an individual user or system make an action. When ownership is involved, it is transferred to a group smart contract instead of a user address. The group later accepts or declines transactions based upon user membership.

The approach is safe and flexible: one user may lose their credentials, and access right can be dynamically revoked or granted by admins.

Crypto Friendliness (5)

No viable approach to the MLETR must expect their users to own a crypto currency required for paying the gas fees induced when emitting transactions. (Nor must it require the user to master the use of a crypto wallet of any kind, either software or hardware as will be seen below).

The TraaaX by KeeeX system uses a proxy smart contract on the KeeeX Chain blockchain responsible for verifying then forwarding transactions duly signed by the sender to the operational smart contracts.

The proxy owns the Ether needed to pay for the gas fees. So, **any user may authenticate with genuine signing capabilities without owning Ether.**

Key Management friendliness (6)

Beyond owning a crypto currency, most industrial users are totally unaware of and unwilling to control private signing keys. Henceforth use any kind of wallet system. Even managing their own password is generally in itself a challenge.

Along with other KeeeX tools, the TraaaX by KeeeX solution encrypts private keys with (a derivation of) the user's password then stores it on the KeeeX Vault service. When a user logs in, the portal fetches the encrypted private key from KeeeX Vault, deciphers it in the browser and uses it there for signing and/or authenticating. The key never leaves the browser, nor does it reach KeeeX servers.

This is done at no compromise: users can display and store the private key recovery seed in the form of a multiword string at creation time. This can be stored separately from the login password. Losing the password or losing the seed means losing exclusive control.

Depending upon acceptance or enforcement, hardware signing devices can be used as well, therefore ensuring that the participant's private key never leaves their premises.

Interoperability (7)

Interoperability forms with Unicity the master keywords of the entire MLETR context. In fact, these concepts are not fully disjoint since interoperability must not introduce the possibility that an asset be used twice, or used by the wrong person, notably regarding the eBL at surrender time.

Interoperability can be understood at several distinct levels:

- A/ different systems or portals may share calls to a common underlying proof system assessing exclusive control
- B/ different systems or portals may be used along the course of a single trade finance operation, yet implement their own ways for exclusive control as long as a document is hosted by them.

TraaaX by KeeeX provides an API to address case A and support the full range of MLETR requirements (change of exclusive control with or without approval, change of properties, finalization, back to paper etc.). The portal in fact simply demoes the API use.

Note that the API does not require that the data is shared. Only undecipherable unique identifiers are shared (and made public) to the registry.

Level B requires a publicly auditable control tower independent from actors. The tool must make publicly verifiable that a process involving some transferable title has left the premises of actor one to enter the premises of actor two. This is required so that no actor is in conflict of interest, and no risk exists that the same asset starts following a multiplicity of distinct paths.

As was previously stated, all keeexed files bear the information of which registry is used to trace their exclusive control and variable properties. The following is operational for levels A and B:

- Document creator keeexes the file and declares TraaaX as responsible for tracing control.
- Level A/ Document creator transfers exclusive control to next party in the process, and every participant proceeds either using API of the portal (the portal gives permissioned access to a document provided control was granted)
- Level B/ Document creator transfers document and grants exclusive control to platform one, responsible for internally handling part of the exclusive control workflow. If the customer wishes so, exclusive control can be transferred to a different platform two by using a publicly auditable API call (or portal action) to TraaaX by KeeeX.

TraaaX by KeeeX henceforth acts as a global control tower monitoring the status of every document whether it is transferred among individual companies acting for their own good, or among larger portals handling part or all of the complexity involved in transferring exclusive control and setting variable properties.

As will be seen below, TraaaX goes even further regarding interoperability than what UNCITRAL's MLETR envisioned by allowing to make the decision to leave TraaaX for some other control tower publicly auditable. In greater technical detail, the interoperability support by TraaaX comes in three flavors: API based, Ethereum based and full transfer.

1. The KeeeX backend API can be consumed by a dedicated SDK (Javascript /React / Node JS) for integration in every possible industrial web portal, ERP Engine or other. This is straightforward and uses the KeeeX Chain blockchain as a default.
2. At keeexing time, the files provably embed the reference to all their traceability blockchain systems and smart contracts. Although using the KeeeX Chain Ethereum blockchain is available as a default, the TraaaX solution is currently compatible with any

Ethereum GETH/Solidity blockchain. This allows any single or group of actors to join forces and operate a hybrid Ethereum network that will scale to their needs. Even if a non-blockchain or non EVM (Ethereum Virtual Machine) compatible blockchain is used, this can be implemented as well.

3. The smart contracts on KeeeX Chain provide the unique feature of allowing the transfer of an asset to another registry/blockchain+smart contract. When dropping a file on TraaaX's verifier for verification (the file is never uploaded and remains in the browser), the system will inform the user that traceability is now available on some other place. This allows for completely moving an item from a portal/blockchain combination to another. This also supports the back to paper situation and move to non-EVM blockchains.

Note that the MLETR authors have provisioned for "back to paper" processes but did not envision the change of registry in the generality of case 3 above.

Document Bundle (8)

A Trade Finance process involves many documents, some of which being transferrable with rights attached to owner (BL, Promissory note, CMR) and other not (a photo of a container).

Most often the Bill of Lading or the Export Booking may serve as a master or parent document, to which other (child) documents are attached. In general, every document may have children.

TraaaX by KeeeX provides for the possibility of mentioning at keeexing time within every document a pre-existing parent document. A smart contract complements this by enabling and proving the reverse traversal of this graph (from parents to children).

Versioning (9)

Traditional endorsement of paper documents used to be performed by adding signatures and stamps at the bottom of the text. This can be viewed as a new version of the document, so we can analyze whether a MLETR tool must implement this.

In the digital, it is often preferred to add to the document a detached endorsement in the form of a separate child document. This is possible here since a blockchain smart contract will also acknowledge the endorsement and possibly prevent the negative impact of a file loss (a signed proof of endorsement is present in the blockchain, which may suffice in many cases).

In the real life, transfer approval often has the semantics of an endorsement. This is so for instance when sending a promissory note to a bank. Endorsement will thus in some cases amount to a successful transfer of exclusive control / ownership.

However, endorsement is not the only reason why a new version of a file might be created. This happens too in case of an error or a mandatory edit. TraaaX by KeeeX enables for the traceability of versioning by means of a versioning smart contract providing bi directional traversal of the version chain, and also direct access to the most recent version.

Approval (10)

There are both functional and resilience reasons to implement an approval process when transferring ownership.

On the functional side, it seems desirable that someone cannot receive ownership (or exclusive control control) of some item without being granted the right or possibility to decline or ignore.

- There may be errors.
- Some situations may induce that participants change their minds.

Dealing with resilience, it would be a process hole if a user receives ownership and no action follows, leaving the process in a stalled status. Of course, we have seen at the resilience chapter that sharing the right to operate at a group level allows for unlocking situations. However, this possibility does not cover for the fact that the sender themselves may change their minds or that as long as the recipient has not acknowledged receipt they might retain the right to cancel the operation.

To those ends, we use a smart contract feature enabling the document's owner to grant the expected recipient the right to transfer ownership. They can use it to transfer to themselves, which acts as an "approval" or decide to transfer to someone else. Until such an action is performed the initial (an still current) owner retains the possibility to cancel. This supports the above requirements:

- if the recipient does not act, the sender retains ownership and may cancel and redirect the process
- if the recipient acts, the sender loses ownership in favor of the recipient as expected, or in favor of a participant that the recipient has willfully chosen.

Conclusion

We have presented the most salient challenges faced by any MLETR implementation. We have also described the foundational principles of TraaaX by KeeeX, and how they address the aforementioned challenges.

Technically today, TraaaX by KeeeX is a ready to deploy solution that matches the requirements of the MLETR. Contact us for testing.

Contact: contact@keeex.net

References

[1] KeeeX WebSite https://keeex.me

[2] TraaaX by KeeeX WebSite https://traaax.com

[3] [Linkedin] How KeeeX addresses all MLETR issues

[4] [Linkedin] Contribution to the french Decree for the ETR Law

[5] https://uncitral.un.org/en/texts/ecommerce/modellaw/electronic_transferable_records