SFT (Secured Financial Transaction) Protocol

Alex Firmani* and Benjamin Hauser†
(HyperLink Capital)
(Dated: November 28, 2018)

The SFT Protocol expands upon the ERC-20 standard to allow for the issuance, governance, and trading of tokenized securities on the Ethereum blockchain. The protocol is open-source and non-rent seeking, aiming to foster a diverse ecosystem for market participants to interact. It is modular, robust, and can be configured to represent both equity and debt based securities throughout their entire life cycle from issuance to redemption. Through automated on-chain compliance the protocol seeks to reduce costs for securities issuers, allow for rapid settlement of trades, and simplify reporting requirements and regulatory supervision. By introducing a common standard for the claim to ownership and transfer of a diverse range of assets we can reduce market segmentation and open previously illiquid markets to a global audience of investors.

I. INTRODUCTION

The SFT Protocol is a set of compliance-oriented smart contracts, written in Solidity for the Ethereum blockchain, that allow for the tokenization of debt and equity based securities. It provides a robust, flexible framework allowing issuers and investors to retain regulatory compliance throughout primary issuance and multi-jurisdictional secondary trading.

By placing as much regulatory logic as possible on-chain, we can significantly reduce compliance costs for issuers and streamline settlement time for secondary trading, either exchange facilitated or OTC.

Some fundamental features of SFT include:

- Shared KYC registries that allow for investors to trade multiple security tokens after only one identity verification
- Cohesive, robust system for tracking investor counts and enforcing limits based on country and accreditation status
- Automated on-chain compliance allowing for secondary trading via OTC, centralized or decentralized markets
- Modular design that allows for the introduction of a wide range of custom logic to meet the needs of any type of security in any jurisdiction, even as the requirements change over time
- MultiSig, MultiOwner functionality for increased security ease of contract management in large organizations.

SFT is based upon ERC-20, the de facto standard for tokens on the Ethereum blockchain. By expanding upon this widely accepted standard, security tokens created with the protocol are compatible with all major Ethereum wallet software and will seamlessly integrate with upcoming security token exchanges.

In this document we will start by outlining our broad approach in designing the protocol. We will then define the types of users and smart contracts that exist within the protocol. Finally we will give a detailed breakdown of each core component and the modular structure that allows the core components to be upgraded or modified.

II. APPROACH

The SFT protocol (Figure 1) is built with two central concepts in mind: identification and permission. Each investor has their identity verified by a registrar and a unique ID hash is associated to their wallet addresses. Based on this identity information, issuers and custodians apply a series of rules to determine how the investor may interact with them.

Our approach to these concepts is to centralize the identification records, and to move permissioning on-chain wherever possible.

A. Identification: Centralized Registries

SFT token contracts are connected through common KYC registry contracts. We envision several of these registries existing, each maintained by a federation of trusted authorities across multiple jurisdictions. Issuers decide which registries they will trust, attach the registries to their contracts, and then any investors that are approved by one of these registries may now trade that issuer's tokens.

By centralizing the KYC registry we can significantly reduce friction for investors seeking to trade in multiple security tokens. Once an investor has passed the KYC/AML checks for a registry, they are now free to trade in any tokens that obtain their identification

* alex.firmani@hyperlink.capital
† ben.hauser@hyperlink.capital

data from it. Streamlining the barriers for entry is an important step towards mass adoption.

Centralized registries also make it easier to impose broad restrictions upon bad actors or in situations of a security breach. If an investor is sanctioned, freezing their assets can be accomplished on the blockchain via a single call to the registry. Similarly if an investors private key is compromised they can request that the registering authority freeze all of their tokens.

If KYC whitelists are only unique to each token it will require many contract calls made by various issuers in order to freeze the assets of a single entity.

Decentralization is recognized as one of the core components of blockchain, but when dealing with securities we must consider not only the technical component but also the legal one. We should initially seek to architect a framework that can fit over top of the already existing system. Once accepted we will be better positioned to engage in conversations about modernizing regulation to take advantage of this superior technology.

B. Permission: On-Chain Compliance

Every action performed by an Ethereum smart contract requires that the caller pay a gas fee to the miner to complete it. This is necessary to prevent bad actors from crippling the network with computationally heavy, meaningless code. It also introduces many challenges for our particular protocol.

To retain legal compliance there are many checks that must be performed before a security token may transferred, and each of these checks requires gas. It is a point of debate within the community as to whether it is better to move as much logic as possible on-chain, at the expense of increased gas, or to handle it off-chain such that an authority must approve each send; either by a second approval call or by providing a signed hash to the sender to include with their transfer call.

Our approach with SFT is to handle as much functionality as possible on-chain. One of the major advantages of blockchain is the transparency and auditability around automated compliance. Moving everything off-chain adds a layer of friction for regulators seeking to observe, which in turn slows settlement time and opens the door for non-compliant action due to error or maliciousness.

We recognize that there are major scaling issues present and that in its current state the Ethereum network would buckle under even a fraction of the transactional throughput present in todays financial markets. Other less computation-heavy approaches to digital securities may scale slightly further than our initial implementation, but it is unlikely any of these first generation approaches will still be used in several years. The SFT standard is not intended as a solution to stand the test of time but rather an implementation of everything that is possible today. We must explore the limits of what is possible within the existing technology in order to evolve the conversation and determine where we can go tomorrow.

III. STRUCTURE AND FLOW

Before we define the shape of the SFT protocol we must outline the types of entities that will exist within it. There are four classes of entities on the platform: issuers, investors, custodians, and registrars.

- **Issuers** are entities that create and sell security tokens to fund their business operations.
- **Investors** are individuals or institutions that have cleared KYC/AML checks and are authorized to purchase and trade security tokens. Each investor is considered to be a single beneficial owner of the security token(s) they possess.
- **Custodians** are entities that are approved to hold tokens for multiple investors. A custodian may be a broker/dealer, an exchange, or any other entity that is given control over an investors tokens without directly being the beneficial owner.
- **Registrars** are legal entities that verify KYC/AML data and accreditation status of an investor, and in doing so authorize them to hold security tokens.

These entities all interact with one another through a series of layered, interoperable smart contracts adhering to the SFT standard. There are four primary contracts which drive the platform: KYCRegistrar, IssuingEntity, SecurityToken, and Custodian.

- **KYCRegistrar** contracts are registries that establish the identity and permissions of investors on the platform. Multiple addresses may be associated to a single entity. These contracts provide the highest permissioning point in the protocol for investors.
- Each issuer on the platform has their own **IssuingEntity** contract. This contract exists to track information about the issuer and allow common permissioning across multiple security tokens.
- Issuers may create one or more **SecurityToken** contracts. These are ERC-20 compliant token contracts that represent a type of fungible security sold by the issuer. The contracts contain additional permissioning functionality to allow for automated regulatory compliance around transfer of ownership.
- Custodians each have their own **Custodian** contract. Through this contract they are able to represent multiple beneficial owners within a single security token.

IssuingEntity, SecurityToken and Custodian contracts implement a modular design that can introduce additional transfer restrictions or functionality. Modules have a deep level of programmatic control available and can be used for many purposes including (but not limited to) dividend payments, voting rights, country/time/volume restrictions, decentralized and centralized exchanges, tender offers, and redemption.

KYCRegistrar, IssuingEntity, and Custodian contracts also implement a multi-sig, multi-owner design that allows for increased security and compartmentalized permissioning of the contract's owner level methods.

IV. BREAKDOWN

A. KYCRegistrar

Each KYCRegistrar contract acts as a whitelist, holding identity information for many investors and permitting them to interact with other contracts within the ecosystem.

At the time of deployment, addresses and a hash associated with the highest contract authority ("the owner") are set. The owner may create and modify permissions of other sub-authorities, that are restricted in which countries they may add, edit, or restrict investors. The owner cannot be restricted.

Each authority has multiple addresses associated with it, and a threshold setting which defines how many addresses must call a function before it will execute (multi-signature functionality). Because a single registrar contract may provide KYC information for many issuers, it is very important to take all security precautions possible around authorities.

After verifying KYC/AML data and accreditation status off-chain, authorities add investors into the registrar. They may also attach additional addresses to each investor, restrict an address, or restrict an investor altogether.

Each investor is represented by a unique hash that is generated by the authority that registers them on the platform. The hash is a bytes32 generated via a keccak256 of the following:

- For natural persons, a string that is a concatenation of the persons full legal name, date of birth as DD/MM/YYYY, and their national identification number.
- For artificial persons, that entitys LEI number[1].

To ensure maximal interoperabilty it is necessary that all authorities agree upon and follow a common hash generation pattern, so that there is no chance for an entity to exist with multiple IDs on different registry contracts.

The ownership of security tokens by various hashes is available publicly on the blockchain, however the personal information of the entity who controls the hash can only be determined if it is revealed by the hash owner or the registering authority. In some jurisdictions, issuers have a legal requirement to know the identity of all investors at all times; in this case the authority will be obliged to provide the investors identity to the issuer upon request. The legal framework of this is beyond the scope of the technical document, we only wish to outline that an investors personally identifiable information is not publicly visible within the blockchain.

Investors additionally have data stored relating to their country, region, and accreditation status. Because some countries impose time limits on accredited verification, each investors approval to buy and sell security tokens is also dependent on a time set by the authority. Once this time has past the investor will be restricted until they re-submit their information to the authority and are approved again.

Once an address has been associated with an entity this association may never be fully removed, it can only be restricted such that the entity is no longer able to interact with any contracts using that address. This is because removing the association would mean that any tokens held at that address are now orphaned, and if the address is later attached to a different entity it could result in an unlawful / non-compliant transfer of ownership.

Authorities have the ability to restrict investors. The restriction may be applied to a single address, or to all addresses controlled by the investor. When an investor is restricted they are unable to transfer any tokens, however the tokens may still be transferred by the issuer that created them.

The registrar contract also contains a variety of getter functions that allow anyone to check an entitys status, rating, country, etc. These functions are used by IssuingEntity and SecurityToken contracts when checking permissions before a token transfer.

B. IssuingEntity

Each issuer has a single IssuingEntity contract. It is through this contract that issuers deploy new security tokens, and so it can be considered to be a digital representation of the sum total of the issuers securities. It exists as single point to monitor and restrict transfers of all securities sold by the issuer.

[1] https://www.gleif.org/en/about-lei

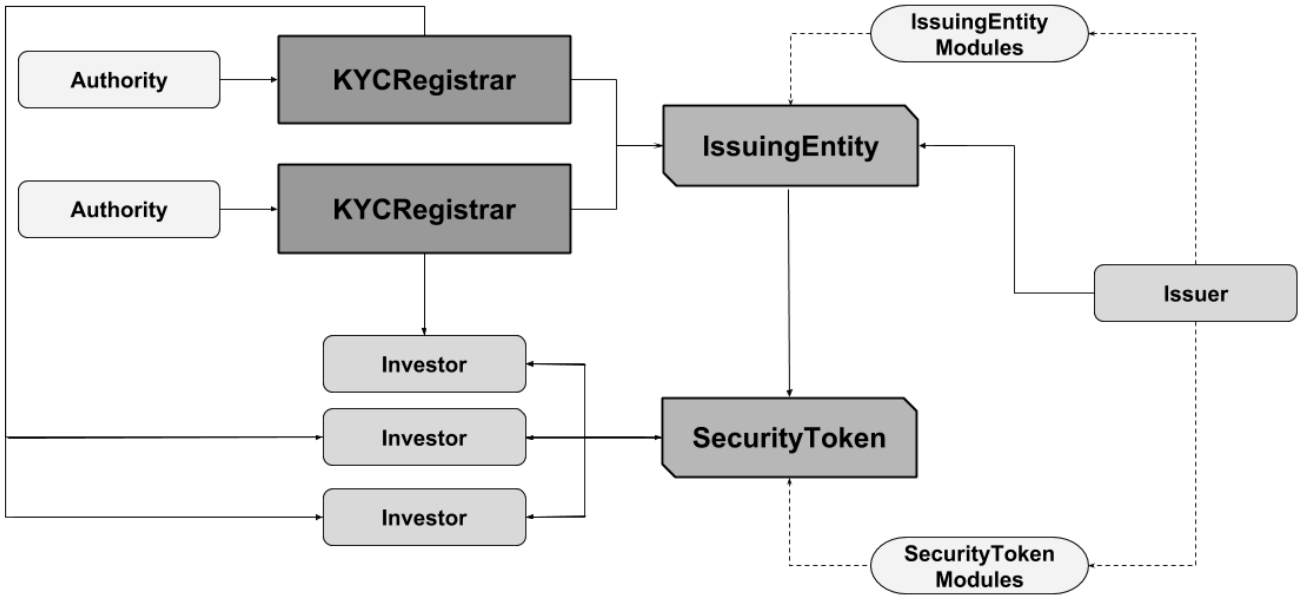

FIG. 1. SFT (Secured Financial Transaction) Protocol

Issuers must approve KYCRegistrar contracts so that investor data may be obtained from them. If an issuer determines that a registry has been corrupted they may remove it and optionally replace it with another. Once an address has been associated to an ID via a registry, this association may never be changed. In this way, there is no possibility for the loss of tokens if an address is registered to different owners in different registries.

One of the main purposes of the IssuingEntity is the capitalization table: tracking and enforcing limits around the number of investors. The issuer can set a maximum total number of investors, and also impose limits based on country and/or investor rating (accredited, qualified, etc). In this way a company can issue multiple tranches or types of securities while still easily maintaining compliance with private placement limitations. Getter functions exist that allow anyone to query the current number and limit for investors for a given country / rating.

Issuers may authorize any number of custodians. Custodians interact with an issuers cap table differently from regular investors. When an investor transfers a balance into a custodian it does not increase the overall count, instead the investor is now included in the list of beneficial owners represented by the custodian. Even if the investor now has a balance of 0, they will be still be included in the issuers investor count.

The issuer can apply restrictions on any investor, rendering that investor unable to transfer the issuers security tokens. In this way an issuer may blacklist investors that are permitted by the main KYC registry, but excluded from the specific token due to a conflict of interest. Issuers may also lock or unlock any of their issued security tokens. When locked, only the issuer may perform a token transfer.

The issuer can record hashes of documents within the IssuingEntity contract, such that investors can easily verify the authenticity of a digitally distributed document relating to the security. Upon registering the document hash, an event is triggered on the blockchain which alerts investors that new information has been made available.

The issuer may also attach one or more modules to the IssuingEntity contract, that hook in at different defined locations to add restrictions and/or functionality. Modules attached to the IssuingEntity are permitted to perform actions on any token issued by that issuer. Modules are covered in detail later in this document.

C. SecurityToken

An issuer may deploy multiple SecurityToken contracts which each represent a single, fungible class of securities. These contracts conform to the ERC-20 standard, with additional functions provided to verify a transfer will succeed without actually attempting it.

At the time of creation the total supply of tokens are assigned to the issuer. Depending on the nature of the security and the issuers preference, they may either be manually distributed or sold via an attached module similar to an ICO crowdsale.

Before tokens may be transferred, a series of checks are applied to ensure the transfer is compliant:

- Sender and receiver addresses must be validated by an approved KYC registrar or registered as custodians.
- Issuer limits on investor counts: global, country specific, and accreditation rating specific.
- Optional restrictions added via modules applied to SecurityToken and IssuingEntity.

Transfers that move tokens between different wallets owned by the same entity are not as heavily restricted because there is no change of ownership. Any address belonging to a single entity can call SecurityToken.transferFrom() to move tokens from any of their wallets. The issuer can use the same function to move any tokens between any addresses, even if the investor has been restricted.

Similar to the IssuingEntity contract, the issuer may also attach one or more modules to the SecurityToken contract to add restrictions and/or functionality. SecurityToken modules may only ever be attached to one token, and only have permission to perform actions upon that token.

The operational flow in a token transfer between to investors is:

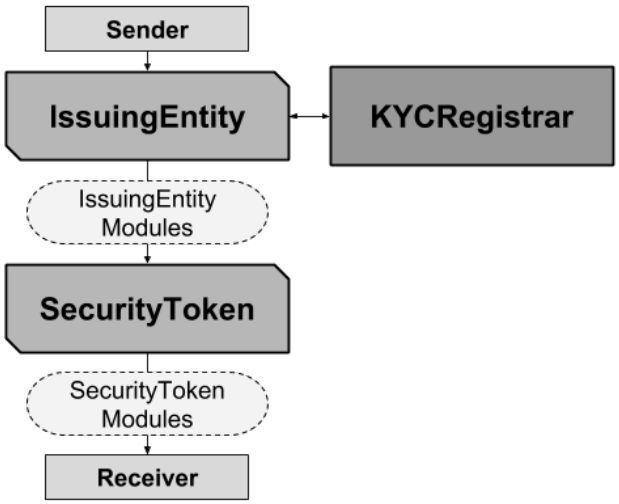

FIG. 2. Permission flow of transfer between investors

D. Custodian

Custodian contracts allow approved entities to hold tokens on behalf of multiple investors. Common examples of custodians include broker/dealers and secondary markets.

Custodians exist to facilitate off-chain transfers of ownership. Once explicitly approved by an issuer, they may receive that issuer's tokens from investors and make calls to the IssuingEntity contract that affect the number of beneficial owners recorded in the cap table. In making these changes, no checks are made against country restrictions, investor limits, or minimum investor ratings. It is the responsibility of the custodian to understand and abide by the issuer's compliance requirements when performing off-chain transfers of ownership.

Custodians may apply one or more modules to their contract to introduce new restrictions or functionality. This can, for example, allow a custodian contract to function as a decentralized exchange.

The operational flow in a token transfer to a custodian is:

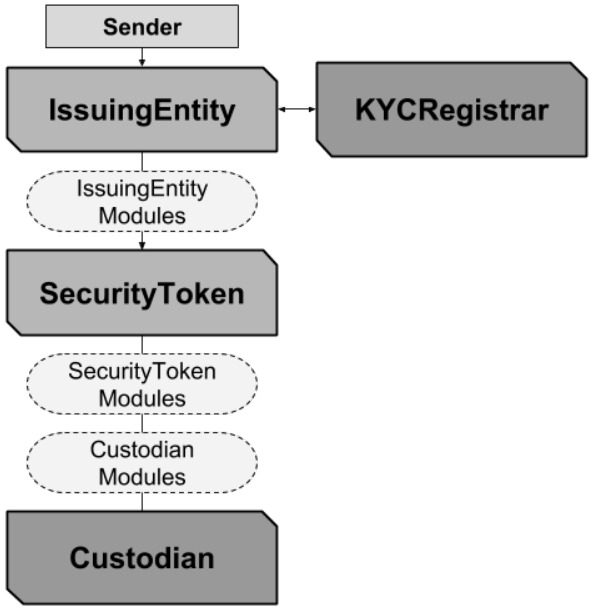

FIG. 3. Permission flow of transfer to a custodian

E. Modules

Modules are contracts that attach to IssuingEntity, SecurityToken or Custodian contracts. They may be used to introduce extra functionality, or add restrictions around token transfers. Because of the wide range of functionality that modules can hook into and their access to modify the parent contract state, many different applications are possible through them. Some examples include dividend

payments, voting rights, country/time based token locks, decentralized exchanges, and redemption.

The contract owner first deploys the module contract and then attaches it to the parent contract. Modules hook into specific parent contract functions. Once attached, any functions called from the parent will also include a call to the relevant module function.

Modules have a similar level of permission to that of the owner of the contract they are attached to. They may (where applicable) transfer tokens, modify balances or investor counts, or further restrict transfers. For this reason it is very important that all modules are audited and understood fully before they are added. Attaching an unknown module from a malicious or incompetent third party could have severe consequences.

In some situations (e.g. a crowdsale or dividend payment) a module is only needed for a portion of the lifecycle of the security. When the module ceases to be required it should be removed by the issuer. This is a good practice as it will reduce the gas costs for token transfers.

F. MultiSig / MultiOwner

IssuingEntity and Custodian contracts both implement a common multisig, multiowner functionality that allows the contract owner to designate other authorities the ability to call specific admin-level contract methods. KYCRegistrar contracts use a slightly modified implementation.

At the time of deployment, addresses and a hash associated with the highest contract authority ("the owner") are set. The owner may create and modify permissions of other sub-authorities, that are restricted in which contract methods they may call. The permission of an authority is also time based, and so can be given for a temporary period and then extended if needed. The owner cannot be restricted.

Each authority has multiple addresses associated with it, and a threshold setting which defines how many addresses must call a function before it will execute. It is also possible to apply this multi-sig functionality to external contracts such as modules.

By setting up multiple authorities that are only capable of calling a limited range of functionality, the contract owner can effectively compartmentalize control over the contract and greatly decrease the potential damage possible in the event of a compromised private key. This compartmentalized approach can also be used in large organizations, to provide access to individual employees or third party contractors only in the areas that are relevant to them.

V. CONCLUSION

We have presented the SFT protocol, an expansion of the ERC20 standard that allows for the issuance and permissioned transfer of tokenized securities on the Ethereum blockchain. SFT has been developed with a focus on interoperability, ease of adoption, and adherence to a wide variety of regulatory requirements. The modular design allows issuers to adapt based on their own needs and an ever changing regulatory landscape. Through standardized, on-chain compliance we lower the entry barriers for investors, issuers, and exchanges, and help to push towards a more global future.