Privacy-preserving KYC on Ethereum

Alex Biryukov
University of Luxembourg
alex.biryukov@uni.lu

Dmitry Khovratovich
University of Luxembourg
(now with ABDK Consulting and Evernym Inc.)
khovratovich@gmail.com

Sergei Tikhomirov
University of Luxembourg
sergey.s.tikhomirov@gmail.com

ABSTRACT

Identity is a fundamental concept for the financial industry. In order to comply with regulation, financial institutions must verify the identity of their customers. Identities are currently handled in a centralized way, which diminishes users' control over their personal information and threats their privacy. Blockchain systems, especially those with support for smart contracts (e.g., Ethereum), are expected to serve as a basis of more decentralized systems for digital identity management.

We propose a design of a privacy-preserving KYC scheme on top of Ethereum. It would let providers of financial services leverage the potential of blockchain technology to increase efficiency of customer onboarding while complying with regulation and protecting users' privacy.

Author Keywords

blockchain, smart contracts, Ethereum, know your customer, KYC

INTRODUCTION

Digital identity is information used by a computer system to represent a user. It serves two purposes:

- Authentication: to prove that the user is who they claim to be;
- Authorization: to ensure that the user has the right to perform the action they are trying to perform.

Modern financial system adheres to the centralized identity model and depends on government-issued identities. Regulation in most jurisdictions demand that banks obtain proof of identity from customers before doing business with them ("know your customer", or KYC). "Anti money laundering" (AML) and "counter terrorist financing" (CTF) are related regulations that require banks to stop and report suspicious transactions.

Modern KYC is not only cumbersome but also privacy violating. Users' sensitive information is stored in banks' databases, where it is difficult to update and can be stolen by corrupt employees or external hackers. Banks implement KYC/AML procedures independently, which leads to high compliance cost for the industry as a whole, as well as multiplies the risk of identity theft and privacy violations.

Open blockchains, the first one being Bitcoin, take a different approach to identity: users join the network without any identification. This technology enabled the creation of more sophisticated decentralized networks with rich programming capabilities, e.g., Ethereum. Banks and other financial services companies see the potential of blockchain technology and are collaborating on its applications in consortia such as Enterprise Ethereum Alliance [17], Hyperledger [1], and R3 [2]. Though to comply with regulation, they have to handle government-issued identities in a blockchain setting, which is a non-trivial task. Taking into account the users' demand for better privacy protection, this becomes even harder. The upcoming European privacy regulation (GDPR [18]) coming into force in May 2018 poses even more challenges for organizations that handle users' personal data.

We first explore the centralized and decentralized approaches to identity. We then propose KYCE – a privacy preserving Ethereum-based KYC implementation for smart contract based financial services. KYCE allows banks to implement KYC checks via an external smart contract – a KYC provider. Our scheme uses zero-knowledge proofs to check users' eligibility without disclosing their private information to anyone except the KYC provider. The whitelist is stored in the KYC smart contract in the form of a cryptographic accumulator. This construction allows users to be efficiently added to, removed from, and checked against a list without storing any plaintext data on the blockchain. We then discuss possible use cases, implementation challenges, and outline the direction for future work.

Centralized identity

We can re-formulate the notion of identity in terms of asymmetric cryptography. Identity I of user U is a public-private key pair $(pub_U, priv_U)$. The public key pub_U authenticates the user (or, equivalently, links the current action to some past actions). Public identifiers like username or address are derived from pub_U. The private key $priv_U$ allows U to sign messages on behalf of I. From the point of view of the system, U is whoever possesses $priv_U$.

Biryukov, Alex; Khovratovich, Dmitry; Tikhomirov, Sergei (2018): Privacy-preserving KYC on Ethereum. In: W. Prinz & P. Hoschka (Eds.), Proceedings of the 1st ERCIM Blockchain Workshop 2018, Reports of the European Society for Socially Embedded Technologies (ISSN 2510-2591), DOI: 10.18420/blockchain2018_09

In the centralized model of identity, which is prevalent on the internet today, users delegate managing their private keys to a trusted party and use a password to access them when necessary. This approach is sub-optimal in many regards. First of all, users do not control their identities. The trusted party always has the technical ability to sign messages without the user's consent or to prevent the user from signing the message they want. Moreover, users' personal data is stored by a centralized entity, which creates additional incentives for malicious actors to attack it. Finally, users have to create a new identity for each website they wish to register with. As a consequence, they adhere to a risky practice of reusing passwords. This problem is partially addressed with the "login with" feature, often implemented using protocols such as OAuth [25] and OpenID [28]. In this scheme, a third-party website queries the website that holds the user's existing identity (e.g., Google) and asks for permission to access a subset of the user's data (e.g., name and email). Upon approval, the access is granted. This approach alleviates the password management problem but increases the impact of a potential identity theft.

Even though users can revoke the access at any time, the "login with" scheme is still privacy violating. Imagine a user that reveals their date of birth to prove to a website that they are 18 years of age or older. Even if they later revoke the access, their date of birth will never change. Thus, they grant the third-party website an effectively unlimited access to a piece of private information.

Maintaining correspondence between "real world" identities and public keys has long been a challenge. Centralized solutions like PKI generally work, but suffer from risks associated with centralization: a fraudulent authority can issue rogue certificates [32].

Decentralized identity and open blockchains

A noteworthy approach to decentralized identity is the PGP "web of trust" [19]. It has not gained significant traction due in part to usability challenges [34] and concerns about the security of the long-term key model [42].

Bitcoin [24] is the first practical implementation of fully decentralized digital cash. It eliminates the problem of connecting public keys to identities in a radical manner: in Bitcoin, public keys *are* identities. Since its launch in 2009, hundreds of alternative open blockchains were developed, most of them adhering to this approach to identity management.

Ethereum [8] [45] is a decentralized blockchain-based smart contracts platform. Smart contracts were initially defined as "a set of promises, specified in digital form" [39]. In Ethereum, a smart contract is a piece of code in Ethereum virtual machine (EVM) bytecode, a Turing complete language. Programmers write contracts in high-level languages targeting EVM, most popular being Solidity, and deploy them onto the blockchain. Users interact with contracts by broadcasting transactions. Upon receiving a transaction, Ethereum nodes execute the corresponding function of the specified contract with given arguments. Nodes maintain a common view of the state using a proof-of-work consensus mechanism.

Contracts can call other contracts' functions and send them units of the Ethereum native cryptocurrency *ether*. Each EVM operation has a cost denominated in units of *gas* to prevent denial-of-service attacks. The user determines the maximum amount of resources their computation will consume and pays for it upfront when sending the transaction. If the computation executes normally, the user gets a refund for the remaining gas. In case of an exception, all allocated gas is consumed, but the transaction has no effect on the state of the blockchain[1].

Traditional financial institutions are becoming interested in blockchain technology, especially in networks enabling smart contracts [13]. However the way open blockchains handle identity may come at odds with financial regulation. We propose a design that will simultaneously leverage the power of blockchain-based smart contracts, enable banks to implement KYC to comply with the law, and preserve users' privacy.

KYCE: A DECENTRALIZED KYC-COMPLIANT EXCHANGE

Definitions and security properties

KYC requirements differ depending on jurisdiction [33] (see Appendix A for a brief overview of the regulatory landscape in the EU). A typical KYC procedure links users' real-world identities to their accounts and checks users against a whitelist or a blacklist. The details of the KYC procedure do not affect our design.

DEFINITION 1. *A **KYC procedure** is a process that determines if a given user is eligible for a given transaction.*

DEFINITION 2. *A **KYC provider** is an entity that performs a KYC procedure.*

DEFINITION 3. *A **financial service** is an information system that allows users to exchange units of value.*

DEFINITION 4. *A financial service is **KYC-compliant** w.r.t. the KYC procedure iff all users are eligible for all transactions they perform.*

DEFINITION 5. *A KYC-compliant financial service is **privacy-preserving** iff only the KYC provider has access to the users' private data.*

Tokens and exchanges

Our KYC solution can be applied for any type of service. For concreteness, consider a token exchange as an example of a financial service.

DEFINITION 6. *A **token** is a transferable fungible unit of value maintained by a smart contract.*

[1] After the Byzantium update in October 2017, certain types of exceptions no longer consume all gas.

ERC20 [44] is the de-facto standard API for implementing token contracts in Ethereum. A token contract keeps track of users' token balances and enables them to transfer tokens using the following functions:

- `transfer` sends a given amount of tokens to a given address.
- `approve` allows a given user to withdraw up to a given amount of tokens from the account of the user calling the function.
- `transferFrom` sends a given amount of tokens from one given address to another (the amount has to be `approve`d beforehand).

DEFINITION 7. *An **exchange** is a service that enables users to exchange tokens.*

The most prevalent type of exchanges is centralized ones, implemented as a regular web service. In this work, we are mostly interested in decentralized, or on-chain exchanges, implemented as smart contracts.

An exchange without KYC support may be used as follows.

1. Alice creates an order to sell X A-tokens for Y B-tokens.
2. Bob creates an order to sell Y B-tokens for X A-tokens.
3. The exchange matches the two orders and transfers (by calling `transferFrom`) X A-tokens from Alice to Bob and Y B-tokens from Bob to Alice.

The transaction succeeds if Alice and Bob `approve`d the exchange with sufficient amount of A- and B-tokens respectively before `transferFrom` is called. Users withdraw tokens from the exchange by calling `approve(exchangeAddress,0)`.

Privacy-preserving KYC

We propose KYCE – a privacy-preserving KYC design for Ethereum-based financial services.

A KYC contract provides an API to other contracts so that external services can determine if a given user is KYC-approved for using a given token. A KYC provider (a governmental entity or company in charge of customer onboarding) performs the necessary checks for a new customer and adds their address to the whitelist.

A naive approach to implementing KYC check with a separate contract would be the following. The KYC contract stores the whitelist of approved addresses. On every `transfer`, token contracts check if the address which is being used belongs to the whitelist. This design has a fundamental drawback from the privacy-preserving standpoint: all whitelisted addresses are stored on the blockchain in plaintext. Moreover, users must use the same addresses they registered with the KYC provider, which violates privacy: an adversary can link the user's transactions in the public blockchain.

Our approach

We use cryptographic techniques to design a privacy preserving KYC solution. In KYCE, the KYC contract stores a **cryptographic accumulator** of the whitelisted addresses.

A cryptographic accumulator A absorbs certain algebraic objects and provides an interface to generate and verify zero-knowledge proofs that a certain value was accumulated. In our construction, to generate a proof for value $x \in A$ one needs a *witness*, which depends on A and x and is provided by the accumulator owner to the user who submitted x. We suggest an accumulator based on bilinear maps due to Camenisch et al. [9].

Briefly, the KYC setup and workflow is as follows. The KYC provider creates and publishes a smart contract, which is initialized with an empty accumulator. The User interacts with the KYC provider physically or online and provides credentials needed to pass the KYC procedure. He also generates his own master secret m and during the authenticated session gives the provider a Pedersen commitment $g_1^m \cdot g_2^r$ to it, where g_1, g_2 are certain group generators[2] and r is random. If the checks are passed, the provider updates the accumulator with user-dependent data and provides the User with a witness, needed to prove the KYC property in the future. In every Ethereum transaction to KYCE, the User provides a proof that he has been registered in the accumulator, that his right has not been revoked, and that the proof owner and the transaction sender are the same person. The latter statement is verified by KYCE, whereas the rest is submitted to the KYC contract for verification against the current accumulator value. If the checks pass, the command is executed in KYCE.

Details on the accumulator construction

We follow the approach by Camenisch et al. [9], who construct an accumulator based on a pairing function $e(\cdot,\cdot)$ in some pairing setting [3]. The accumulator contains just serial numbers, possibly consecutive integers[4]. The accumulator is constructed as follows. We assume a bilinear pairing $e : G \times G \to G_T$ where G, G_T are groups of order q. The KYC provider selects generator g and the secret value $\gamma \xleftarrow{\$} \mathbb{Z}_q$. It also selects L as an upper bound of users enabled for KYC and computes $z = e(g,g)^{\gamma^{L+1}}$. The accumulator value A is initialized by 1.

Let us denote $g_i = g^{\gamma^i}$. The provider publishes $\mathrm{A}, \{g_i\}_{1 \le i \le L,\, L+2 \le i \le 2L}$, the set of registered KYC indices $V = \emptyset$, and the parameters g, z needed to perform a verification.

[2]Here and in the further text all multiplications take place in the pre-selected group of prime order q, typically an elliptic-curve group.

[3]The original paper [9] uses type-1 pairings, but type-3 pairings can be adopted as well.

[4]It is possible to store public keys but it would be less efficient.

Every User who passes the KYC check is issued a new serial number i, the witness $w_i = \prod_{j \in V, j \neq i} g_{L+1-j+i}$, where V is the set of all issued serial numbers, and a signature σ_i of $g_i||i$ on the provider's private signature key. The witness is used to generate a proof of accumulating[5]. The accumulator is updated by the KYC provider with i by

$$\mathrm{A}_{\mathrm{V} \cup \{\mathrm{i}\}} \leftarrow \mathrm{A}_{\mathrm{V}} \cdot g_{L+1-i}$$

multiplying it by $g_{L+1-i} = g^{\gamma^{L+1-i}}$, and i is published as a new valid serial number. To prove that i has been committed to A and has not been revoked without disclosing it, the holder of w_i must update it[6] so that the following equation holds:

$$\frac{e(g_i, A)}{e(g, w_i)} = z.$$

Note that revocation is also efficient: the KYC contract owner simply multiplies the accumulator value by the inverse of g_{L+1-i}. The witness value can not be updated anymore.

Presentation

When issuing a transaction to use the exchange (e.g., create an order), the user submits a **zero-knowledge proof** of the following statement:

- I know the private key of the current user address (`msg.sender`), and
- I know a signature σ_i and a witness w_i for some number i that has been accumulated in the accumulator A in the KYC contract.

It is crucial that this compound statement is *atomic*, i.e. the sub-statements can not be extracted as separate valid proofs, as this would make the transaction malleable.

The atomicity (and thus non-malleability) are ensured as follows. Let us denote the proof of knowledge for the witness and signature by PK_w, which is given in [9], Section 4.2. Then Prover submits

$$P = \{PK_w \wedge PK_s\},$$

where PK_s is the proof of knowledge of the private key of the `msg.sender`'s ECDSA public key, which can be taken from [11]. The technique to make a composite proof of knowledge is straightforward as both PoKs are non-interactive and is standard in complex PoK protocols:

1. Prover collects a set $\mathcal{C}$ of commitments asserted in subproofs PK_w and PK_s.
2. Prover makes necessary randomization of $\mathcal{C}$ to create t-values $\mathcal{T}$.
3. Prover computes $c \leftarrow H(\mathcal{C}, \mathcal{T})$.
4. Prover computes s-values $\mathcal{S}$ using $\mathcal{C}$, $\mathcal{T}$, and c.
5. The proof P is $(\mathcal{C}, \mathcal{S}, c)$. To verify it one computes asserted t-values $\widehat{\mathcal{T}}$ and verifies
$$c \stackrel{?}{=} H(\mathcal{C}, \widehat{\mathcal{T}}).$$

The resulting proof P is submitted as an Ethereum transaction argument. KYCE retrieves the most recent accumulator value and verifies P against it and the public key of the message sender, which is available in the transaction metadata. If the proof is correct the order is executed.

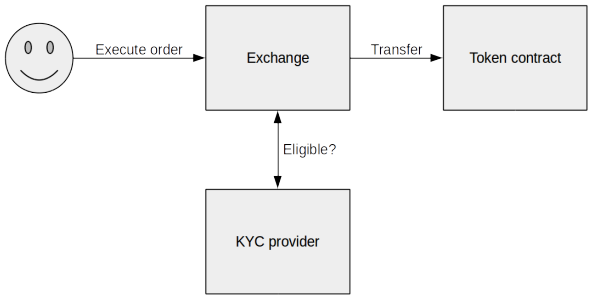
Use cases

Either the exchange contract or the token contract must be KYC-compliant – i.e., check eligibility of transacting parties using the implementation of the cryptographic scheme described above in the KYC contract.

KYC-compliant exchange

If the exchange is KYC-compliant, the tokens do not need to be aware of the KYC.

Figure 1. KYC-compliant exchange



Consider an established exchange that trades dozens of tokens. It applies for official approval in a jurisdiction that requires all customers to pass the KYC procedure. The governmental body acts as a KYC provider, deploys a KYC contract, and publishes its address. The exchange adds KYC checks to its codebase and continues operation. Users who do not want to apply for KYC can simply withdraw their tokens from the exchange and use them elsewhere.

KYC-compliant token

If the token is KYC-compliant, the exchange does not need to be aware of the KYC.

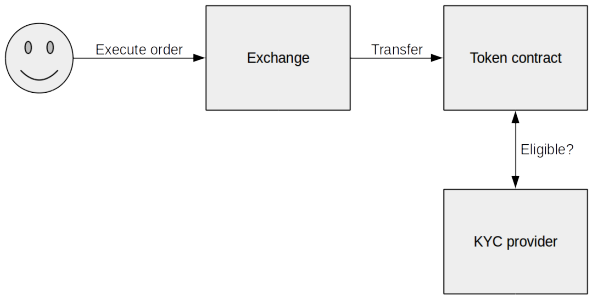
Consider a government that issues its own tokens[7]. Government tokens could be used by KYC-approved users for tax payments, fees, fines, etc. Such solution leverages the flexibility and auditability of smart contracts while limiting the userbase of the token to the approved entities only. The KYC-enabled government token can be also traded on exchanges. This allows citizens to hold their wealth in currency portfolios of their choice and

[5] We refer an interested reader to [9] for the details.

[6] We omit the details, but the update can be performed just before the presentation, not necessarily after every accumulator update.

[7] Bank of England [12] and the Monetary Authority of Singapore [4] already did research in this direction.

Figure 2. KYC-compliant token



only purchase government tokens to transact with the state.

Transaction-dependent checks

Many jurisdictions impose additional restrictions that depend on the value of the transaction. E.g., the EU regulation [30] states that "the obligation to check whether information on the payer or the payee is accurate should [...] be imposed only in respect of individual transfers of funds that exceed €1000". EU member states impose further restrictions for transactions of higher value, e.g., exceeding €10000 in Belgium, €15000 in Germany and in the Netherlands [33]. Either the exchange contract or the token contract can perform such checks by storing the following mappings:

- address => accumulated transaction volume in the current period (day, month, year);
- address => timestamp of the latest transaction.

IMPLEMENTATION DETAILS

We created a proof-of-concept implementation of the proposed design. Our project consists of two smart contracts written in Solidity: KycProvider and KyceToken.

Initial (not privacy-preserving) implementation

In the initial (not privacy-preserving) implementation, KycProvider maintains a 2-dimensional boolean array that stores the eligibility status across users and tokens. On initialization, the address that deploys the contract to the blockchain is made the *owner*, allowing it to add and remove users from the array. The ownership may be transferred (using the functionality inherited from the standard Ownable contract).

The KycProvider exposes the following API:

- `add(address _user, address _token)` – makes the user eligible for using the token (callable only by the owner)
- `remove(address _user, address _token)` – makes the user not eligible for using the token (callable only by the owner)
- `isEligible(address _user, address _token)` – checks if the user is eligible for using the token

KyceToken adheres to the de-facto standard token API in Ethereum – ERC20. To minimize the risk of security issues due to implementation subtleties, we inherit a widely used and tested ERC20 implementation by OpenZeppelin. We override the functions `approve`, `transfer`, and `transferFrom` to check if the given user (`msg.sender`) is eligible for using this token. Namely, the function `isEligible` is called. If the returned value is `false`, the execution stops; is it is `true`, the corresponding function of the super class is invoked.

The implementation of the proposed scheme requires cryptographic primitives partially already available in Ethereum as pre-compiled contracts (namely, elliptic curve addition and scalar multiplication, as well as pairing checks). For the proposed scheme to be fully implemented, pairing evaluation is also required. We are looking into the possibilities to add this functionality.

RELATED WORK

Parra-Moyano and Ross use distributed ledger technology to improve the KYC process [31]. Their proposal can be summarized as follows:

- the regulator maintains a database with all users' private data;
- the first bank a user signs a contract with (the "home bank") stores hashes of the user's documents in a smart contract in a permissioned blockchain;
- all subsequent banks the user wants to work with obtain the user's documents from the database and look the hash up to ensure that the user had been KYC-approved (without knowing which home bank had done it);
- a cost-sharing mechanism for banks allows to proportionally share the cost of the initial KYC approval among all banks that use it.

In this design, all banks store users' private data – contrary to our solution, where it is stored only with the KYC provider. A more decentralized design is also proposed, but the authors claim it to be of a lesser practical relevance.

Sullivan and Burger investigate possible implications of further development of the Estonian e-residency program using blockchain technology [38]. E-residency of Estonia is a governmental program that provides applicants with a digital identity, which can then be used, e.g., to register a company and open a bank account. Estonian e-residency disconnects a digital identity from citizenship or physical residence. Within the e-residency program, Estonia collaborates with a blockchain project Bitnation [6] [14]. Oraclize, a company that provides trusted external data to Ethereum smart contracts, implemented a connector that lets Ethereum contracts handle e-residency identities [29].

An existing project [27] implements a KYC scheme in an Ethereum smart contract, but stores the KYC status on the blockchain in plaintext.

There are multiple projects aimed at easing customer onboarding (creating an identity for a new user and ensuring KYC compliance) for banks. Some of the projects are: Cambridge Blockchain [7], Cetas [10], Fundchain [20][8], KYC-chain [22], KYCStart [15], SnapSwap [36], Tradle [40]. Blockchain consortium R3 developed a proof-of-concept implementation of a shared KYC between ten banks based on its blockchain platform Corda [3].

CONCLUSION AND FUTURE WORK

We proposed a modular design of an Ethereum-based financial service with an external KYC check, which brings benefits to all participants:

- **Users** obtain a unified identity which they can use to utilize multiple financial services. Users' personal data is stored only with the KYC provider and can be easily updated. Personal data is neither stored on the blockchain nor transmitted to third parties.

- **Financial services** greatly simplify the KYC process: it boils down to a single API call. Our design lets them cut KYC costs while at the same time diminishing risks of handling sensitive data.

- **Governments** get an opportunity to stimulate innovation in the financial sector by providing a unified and simple KYC API. This is especially important in the context of rapidly growing fintech and blockchain industries.

Our design is agnostic to the nature of the entity behind the KYC contract: it does not have to be a government body. The proposed solution can be used in any setting where a smart contract based service wants to limit the set of its users according to some criteria. For instance, many jurisdictions (e.g., the US [35]) only allow certain type of investment to be offered to "accredited investors" – typically, high-net-worth individuals and financial institutions. This logic can be replicated in a blockchain setting. Consider a blockchain-based financial service that only wants to deal with experienced cryptocurrency users (e.g., those who possess more than $10000 in ether and did their first transaction earlier than 2016). The "accrediting" functionality is delegated to a third party KYC provider. Proving net worth and previous activity on the blockchain is straightforward; additional checks can also be added. Once accredited, a blockchain investor uses multiple "restricted" services without revealing any personal details to their developers. Privacy-preserving KYC might be a good use case for Ethereum-based identity projects [23], e.g., Sovrin [37] and uPort [41].

[8] A blockchain-based asset management solution including KYC implementation.

ACKNOWLEDGEMENTS

A proof-of-concept implementation of the design described above was created in May 2017 during the Luxblock hackathon in Luxembourg by the CryptoLUX team, and was awarded a joint first prize. The team included Daniel Feher, Dmitry Khovratovich, Sergei Tikhomirov, Aleksei Udovenko, and Maciej Żurad.

REFERENCES

1. 2018. Hyperledger Business Blockchain Technologies. (2018). https://www.hyperledger.org/.
2. 2018. R3. (2018). https://www.r3.com/.
3. Ian Allison. 2016. R3 develops proof-of-concept for shared KYC service with 10 global banks. (2016). http://www.ibtimes.co.uk/r3-develops-proof-concept-shared-kyc-service-10-global-banks-1590908.
4. Monetary authority of Singapore. 2017. The future is here. Project Ubin: SGD on Distributed Ledger. (2017). http://www.mas.gov.sg/Singapore-Financial-Centre/Smart-Financial-Centre/Project-Ubin.aspx.
5. Matthias Berberich and Malgorzata Steiner. 2016. Blockchain Technology and the GDPR – How to Reconcile Privacy and Distributed Ledgers? *European Data Protection Law Review* 2 (2016), 422 – 426. Issue 3. http://edpl.lexxion.eu/article/EDPL/2016/3/21.
6. Bitnation. 2015. Estonia e-residency program & Bitnation DAO public notary partnership. (2015). https://bitnation.co/blog/pressrelease-estonia-bitnation-public-notary-partnership/.
7. Cambridge Blockchain. 2017. (2017). http://cambridge-blockchain.com/.
8. Vitalik Buterin. 2014. A Next-Generation Smart Contract and Decentralized Application Platform. (2014). https://github.com/ethereum/wiki/wiki/White-Paper.
9. Jan Camenisch, Markulf Kohlweiss, and Claudio Soriente. 2009. An Accumulator Based on Bilinear Maps and Efficient Revocation for Anonymous Credentials. In *Public Key Cryptography (Lecture Notes in Computer Science)*, Vol. 5443. Springer, 481–500.
10. Cetas. 2017. (2017). https://cetas.systems/.
11. Melissa Chase, Chaya Ganesh, and Payman Mohassel. 2016. Efficient Zero-Knowledge Proof of Algebraic and Non-Algebraic Statements with Applications to Privacy Preserving Credentials. In *CRYPTO (3) (Lecture Notes in Computer Science)*, Vol. 9816. Springer, 499–530.

12. George Danezis and Sarah Meiklejohn. 2015. Centrally Banked Cryptocurrencies. *CoRR* abs/1505.06895 (2015). http://arxiv.org/abs/1505.06895.

13. Michael del Castillo. 2017. Enterprise Ethereum Alliance Adds 86 Members to Blockchain Consortium. (2017). http://www.coindesk.com/enterprise-ethereum-alliance-new-members-blockchain/.

14. e-Estonia. 2015. New Possibilities for e-residents. (2015). https://e-estonia.com/new-possibilities-for-e-residents/.

15. EconoTimes. 2017. Deloitte Luxembourg develops blockchain PoC 'KYCStart' to perform customer onboarding. (2017). http://www.econotimes.com/Deloitte-Luxembourg-develops-blockchain-PoC-KYCStart-to-perform-customer-onboarding-691965.

16. Meghan Elison. 2016. Christopher Kong: PSD2 Means Opportunity. (2016). https://ripple.com/insights/christopher-kong-psd2/.

17. Enterprise Ethereum Alliance. 2017. (2017). https://entethalliance.org/.

18. EUGDPR. 2016. EU General Data Protection Regulation. (2016). http://www.eugdpr.org/.

19. Patrick Feisthammel. 2017. Explanation of the web of trust of PGP. (2017). https://www.rubin.ch/pgp/weboftrust.en.html.

20. Fundchain. 2017. (2017). http://fundchain.lu/.

21. Viola Hellström. 2017. PSD2 – the directive that will change banking as we know it. (2017). https://www.evry.com/en/news/articles/psd2-the-directive-that-will-change-banking-as-we-know-it/.

22. KYC-Chain. 2017. (2017). http://kyc-chain.com/.

23. Elena Mesropyan. 2017. 21 Companies Leveraging Blockchain for Identity Management and Authentication. (2017). https://letstalkpayments.com/22-companies-leveraging-blockchain-for-identity-management-and-authentication/.

24. Satoshi Nakamoto. 2008. Bitcoin: A Peer-to-Peer Electronic Cash System. (2008). https://bitcoin.org/bitcoin.pdf.

25. OAuth. 2017. (2017). https://oauth.net/.

26. Council of the EU. 2016. Money laundering and terrorist financing: Council agrees its negotiating stance. (2016). http://www.consilium.europa.eu/en/press/press-releases/2016/12/20-money-laundering-and-terrorist-financing/.

27. Mikko Ohtamaa. 2016. Know Your Customer partner integration. (2016). https://github.com/TokenMarketNet/ethereum-tokens/blob/master/KYC.rst.

28. OpenID. 2017. (2017). https://openid.net/.

29. Oraclize. 2017. Identity on the blockchain – chapter 3. (2017). https://blog.oraclize.it/identity-on-the-blockchain-chapter-3-585bc5c7e2c7.

30. European Parliament. 2015. Regulation (EU) 2015/847 of the European Parliament and of the Council of 20 May 2015 on information accompanying transfers of funds and repealing Regulation (EC) No 1781/2006 (Text with EEA relevance). (2015). http://eur-lex.europa.eu/legal-content/EN/ALL/?uri=celex:32015R0847.

31. José Parra-Moyano and Omri Ross. 2017. KYC Optimization Using Distributed Ledger Technology. (2017). https://ssrn.com/abstract=2897788.

32. Business Unit Prins, JR and Cybercrime. 2011. DigiNotar Certificate Authority breach "Operation Black Tulip". *Fox-IT, November* (2011).

33. PWC. 2015. Know your customer: quick reference guide. (2015). https://www.pwc.lu/en/anti-money-laundering/docs/pwc-aml-know-your-customer-2015.pdf.

34. Scott Ruoti, Jeff Andersen, Daniel Zappala, and Kent E. Seamons. 2015. Why Johnny Still, Still Can't Encrypt: Evaluating the Usability of a Modern PGP Client. *CoRR* abs/1510.08555 (2015). http://arxiv.org/abs/1510.08555.

35. US securities and exchange comission. 2014. Accredited Investors. (2014). https://www.sec.gov/fast-answers/answers-accredhtm.html.

36. SnapSwap. 2017. (2017). https://snapswap.eu/.

37. Sovrin. 2017. (2017). https://www.sovrin.org/.

38. Clare Sullivan and Eric Burger. 2017. E-residency and blockchain. (2017). https://doi.org/10.1016/j.clsr.2017.03.016.

39. Nick Szabo. 1996. Smart Contracts: Building Blocks for Digital Markets. (1996).

40. Tradle. 2017. (2017). https://tradle.io/.

41. Uport. 2017. (2017). https://www.uport.me/.

42. Filippo Valsorda. 2016. I'm giving up on PGP. (2016). https://blog.filippo.io/giving-up-on-long-term-pgp/.

43. Niels Vandezande. 2017. Virtual currencies under EU anti-money laundering law. (2017). https://doi.org/10.1016/j.clsr.2017.03.011.

44. Fabian Vogelsteller. 2017. ERC: Token standard. (2017). https://github.com/ethereum/EIPs/issues/20.

45. Gavin Wood. 2014. Ethereum: A secure decentralised generalised transaction ledger. (2014). http://gavwood.com/paper.pdf.

APPENDIX

FINANCIAL AND PRIVACY REGULATION IN THE EU

The current EU legislation "on information accompanying transfers of funds" came into effect in 2015 [30]. In the wake of the rapid growth of cryptocurrencies, the EU is tightening its **anti-money laundering regulations**, stating that "virtual currency exchange platforms and custodian wallet providers will have to apply customer due diligence controls, ending the anonymity associated with such exchanges" [26]. Vandezande analyzes virtual currencies under the EU anti-money laundering law [43].

2018 is set to be a "game-changing" year for European financial industry, as two important regulations come into force.

The **Revised Payment Service Directive** (PSD2) obligates banks to provide third-party providers access to their customers' accounts through open APIs [21]. This is meant to foster competition and give rise to third-party financial service providers. For instance, unified banking API will likely make connecting banks' infrastructure to open blockchains simpler [16].

The **General Data Protection Regulation** (GDPR), coming into force on 25 May 2018, harmonizes data privacy laws across the EU [18] and introduces stricter rules for handling data of EU residents even for companies from outside the EU. Berberich and Steiner describe possible implications of blockchain adoption from the point of view of the EU data protection regulation [5].