Understanding Smart Contracts: Hype or Hope? **Online Appendix**

Elizaveta Zinovyeva Raphael C. G. Reule
Blockchain Research Center, Humboldt-Universität zu Berlin, Germany.
International Research Training Group 1792, Humboldt-Universität zu Berlin, Germany.
elizaveta.zinovyeva[at]wiwi.hu-berlin.de irtg1792.wiwi[at]wiwi.hu-berlin.de

Wolfgang Karl Härdle
Blockchain Research Center, Humboldt-Universität zu Berlin, Germany.
Wang Yanan Institute for Studies in Economics, Xiamen University, China.
Sim Kee Boon Institute for Financial Economics, Singapore Management University, Singapore.
Faculty of Mathematics and Physics, Charles University, Czech Republic.
National Chiao Tung University, Taiwan.
haerdle[at]wiwi.hu-berlin.de

June 29, 2021

1 List of cryptocurrencies used in this research

Abbrev.	CC	Website
BTC (XBT)	Bitcoin	`bitcoin.com`, `bitcoin.org`
ETC	Ethereum Classic	`ethereumclassic.github.io`
ETH	Ethereum	`ethereum.org`
LEO	UNUS SED LEO	`bitfinex.com` (iFinex ecosystem)
USDC	USD Coin	`centre.io/usdc`

2 List of abbreviations

Terminus	Abbrev.
Blockchain	BC
Cryptocurrency	CC
Smart Contract (general terminus)	SC
Verfified Smart Contract (Source Code public)	VSC
decentralized Apps	DApps
State of the DApps	SDA
Externally Owned Account	EOA
Contract Account	CA
Transaction (BC recorded)	TX
Call/Message ("internal TX")	MSG

3 Literature Research

3.1 Literature Meta Analysis on the Keyword Level

We present the most frequent keywords as a wordcloud in Figure 1. The keywords *blockchain*, *smart contract*, and *ethereum* were filtered out since they were used to limit the query and therefore do not deliver any additional semantic information to differentiate between the outlets. Technical keywords such as *security*, *data*, *iot*, *solidity*, *distributed ledger*, and *decentralized* are the most frequently used. Unfortunately, these do not provide us with sufficient additional information to identify clear tendencies in the research. Each of these keywords could belong to any given paper in the field of SC research. Hence, bibliometric research on SC outlets needs to be conducted on a more granular level.

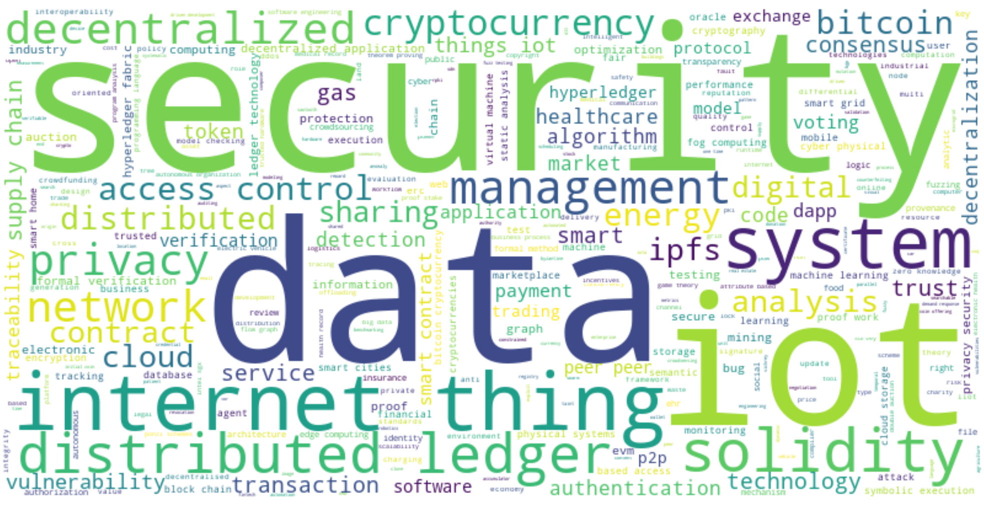


Figure 1: Wordcloud of existing research keywords

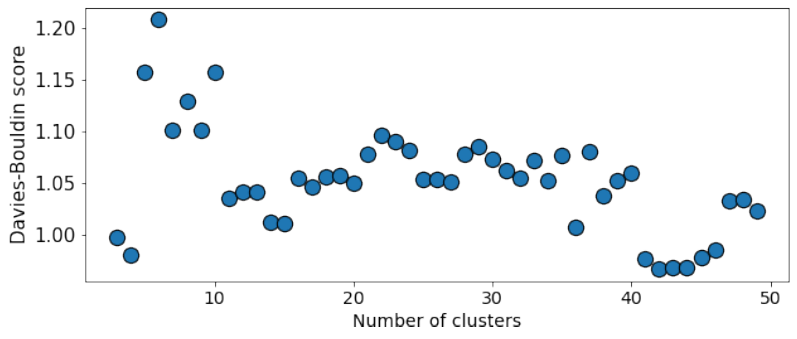
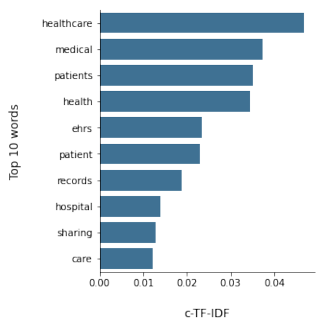


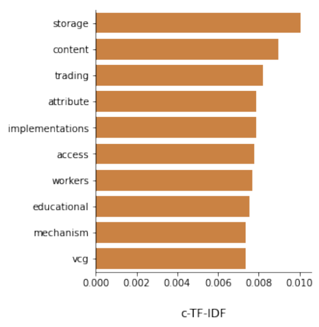
Figure 2: Identification of optimal number of groups using elbow-curve method with Davies-Bouldin score

3.2 Literature Meta Analysis on the Abstract Level

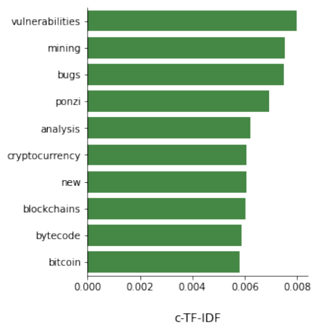
To identify the optimal amount of clusters in our literature research, we utilized the elbow criterion (Thorndike, 1953). This method measures the identified performance metric for different numbers of groups, plots them against each other, and identifies points that are knee or elbow of a curve. The possible metrics for clustering could be an intra- or intercluster variance, the Davies-Bouldin, and Silhouette scores. Since these metrics might produce contradictory results, we decided to utilize only one of them – The Davies-Bouldin score. This score (Davies and Bouldin, 1979) shows the average similarity measure of each cluster with its most similar cluster. Figure 2 displays the Davies-Bouldin score, where similarity is the ratio of within-cluster distances to between-cluster distances. It evaluates intra-cluster similarity and inter-cluster differences. Applying the elbow criterion to these diagnostic plots leads us to a value around 14 – one of the first *elbows* of the plot, as we prefer rather lower values of the Davies-Bouldin score and the lower amounts of topics an-ease the interpretability. In Figures 3, 4, you can find the 10 most important words for each one of these 14 topics.



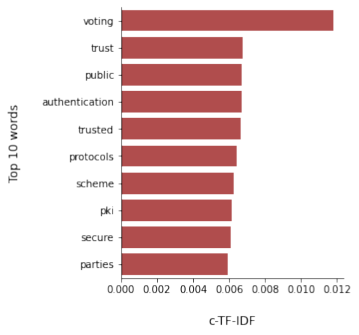
(a) Healthcare/medicine



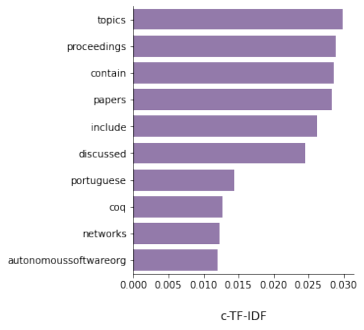
(b) Implementation/Storage



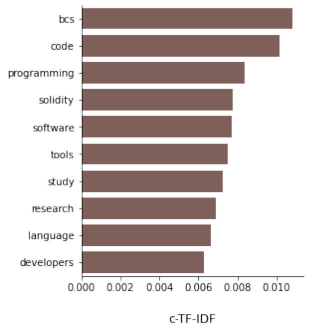
(c) Vulnerabilities/Ponzi



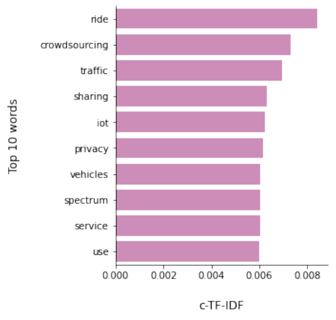
(d) Voting/Trust



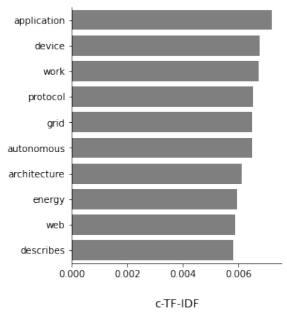
(e) Research/Categories of Research



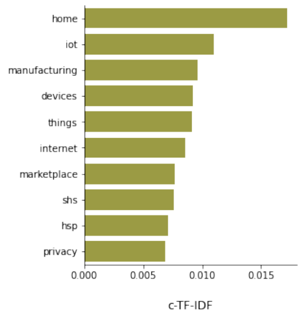
(f) Programming language/Solidity/Software Development



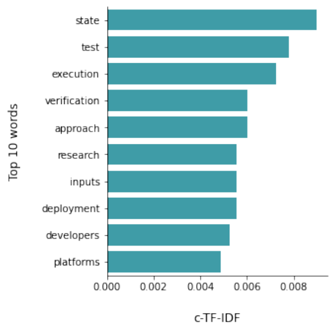
(g) Crowdsoursing/Sharing



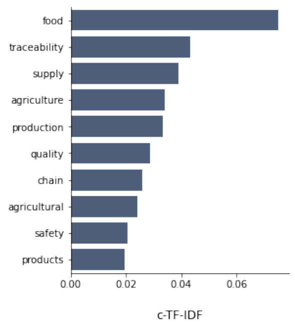
(h) Applications/Development



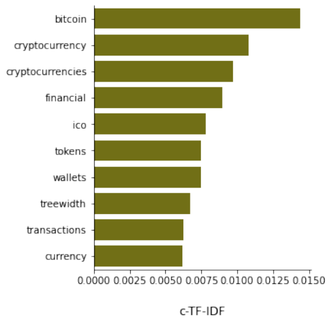
(i) IOT/Smart devices



(j) Verification/Deployment

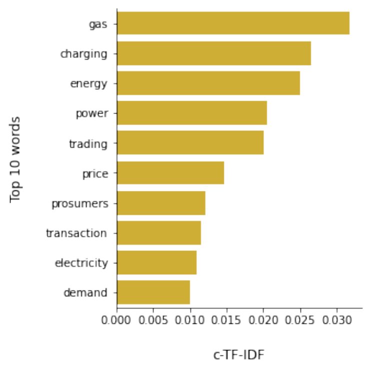


(k) Food/Agriculture

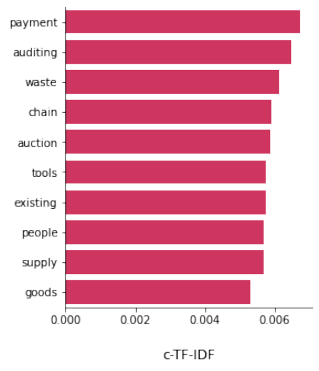


(l) BTC/Finance/ICO

Figure 3: Top 10 the most important words per topic identified in the existing SC research (Part 1)



(a) Energy/Electricity/Gas



(b) Payments/Audit

Figure 4: Top 10 the most important words per topic identified in the existing SC research (Part 2)

4 Cryptocurrency Dynamics

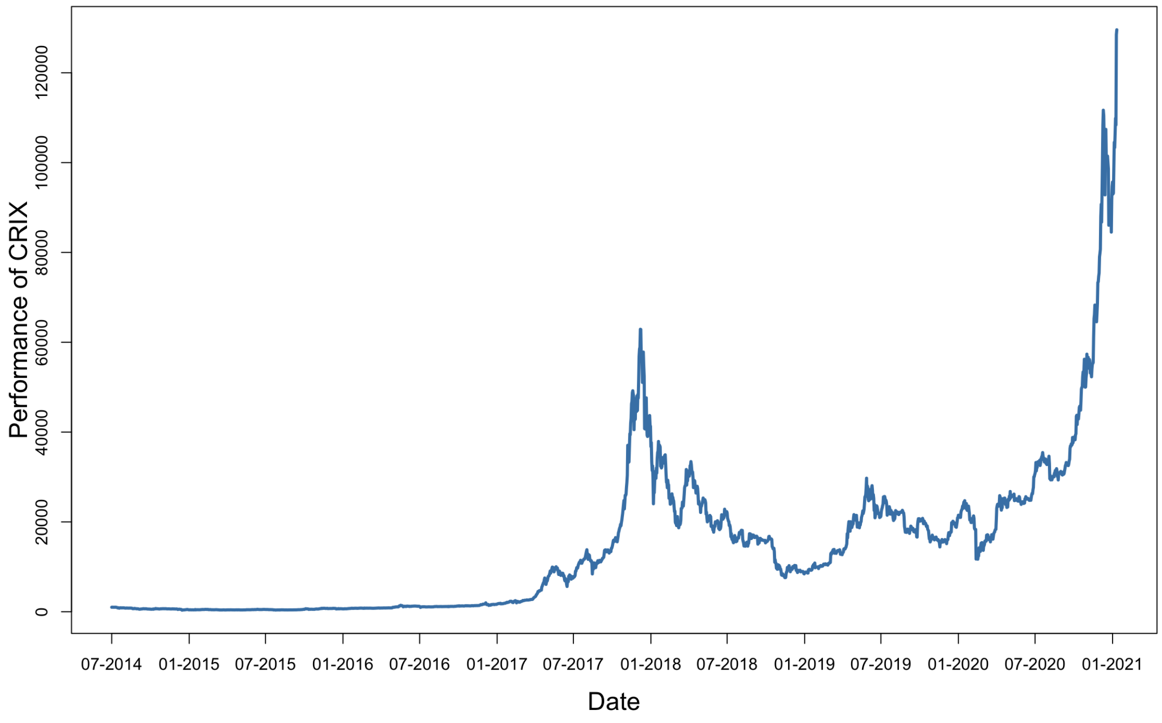


Figure 5: CRyptocurrency IndeX CRIX, 20140731-20210211

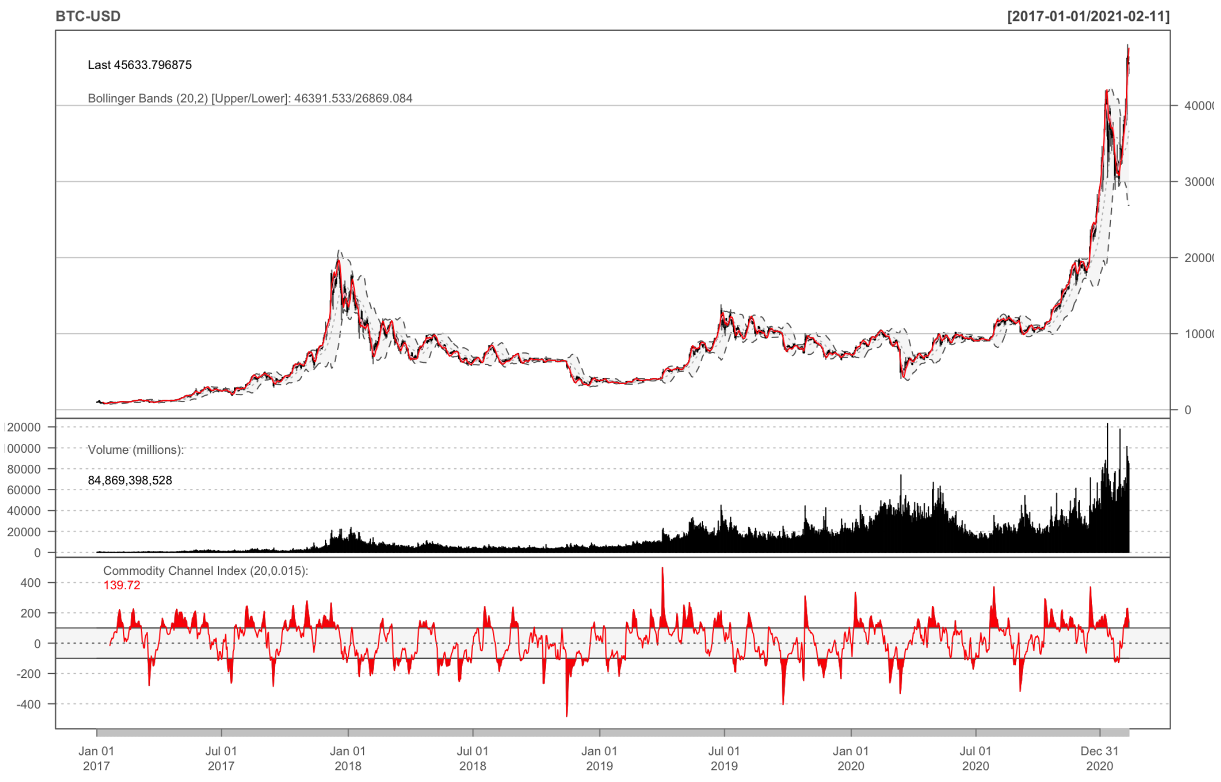


Figure 6: BTC-USD, 20170101 - 20210211

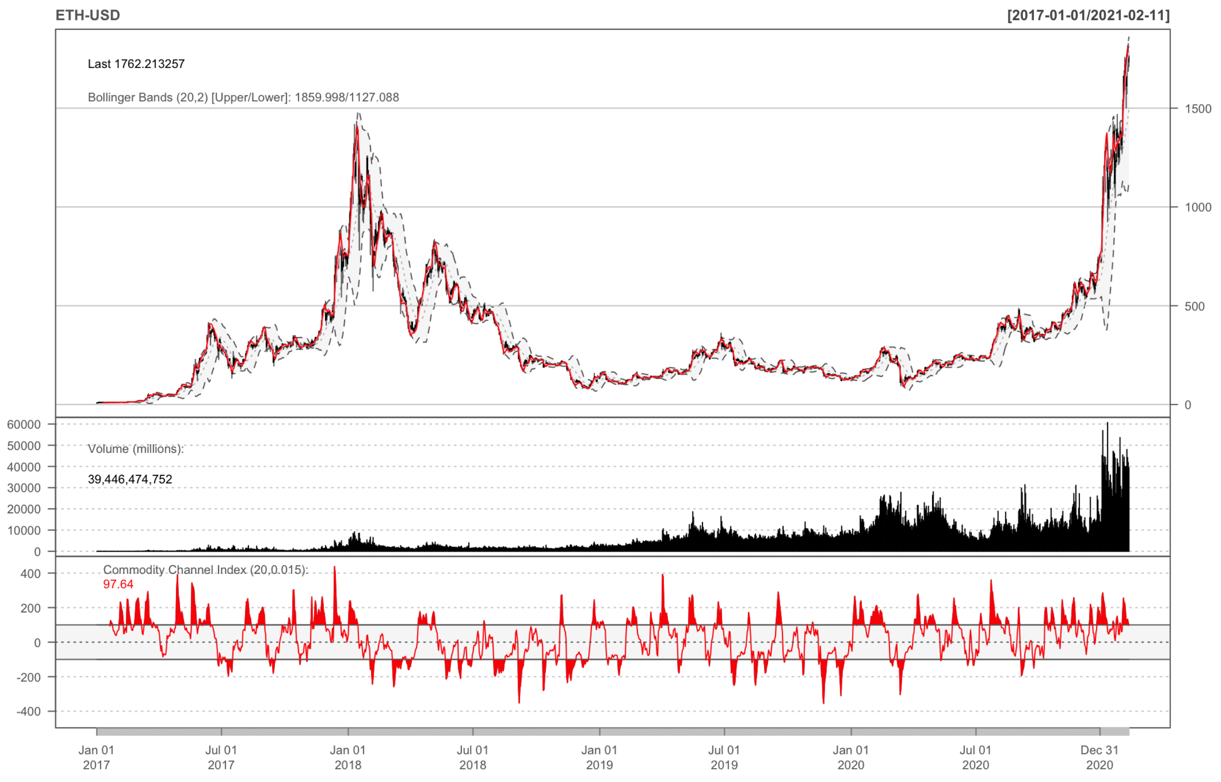


Figure 7: ETH-USD, 20170101 - 20210211

5 ETH value denominations

Further Units and Globally Available Variables used in Solidity can be accessed through Ethereum - Read the Docs.

Unit	**Wei Value**	**Wei**
Wei	1 wei	1
Kwei (babbage)	1e3 wei	1,000
Mwei (lovelace)	1e6 wei	1,000,000
Gwei (shannon)	1e9 wei	1,000,000,000
Microether (szabo)	1e12 wei	1,000,000,000,000
Milliether (finney)	1e15 wei	1,000,000,000,000,000
Ether	1e18 wei	1,000,000,000,000,000,000

6 Elliptic Curve Digital Signature Algorithm

Asymmetric cryptography is used to create accounts in ETH in three steps: Firstly, the private key associated with an EOA is randomly generated as SHA256 output and could look like this: b032ac4de581a6f65c41889f2c90b3a629dc80667bf7167611f8b5575744f818 - a random 256 bit/32 bytes big and 64 hex character long output. Secondly, the public key is derived from the private key via the Elliptic Curve Digital Signature Algorithm (secp256k1 ECDSA) and could then look like this: fc2921c35715210aeb0f2fffeaad94 d906aaf2feda8a71e52c5d0a0da8ada4a44099e8ea65e3ec214b6686189255bba2373bd2ee6b05 520dfd4dc571b682ccc3 - a 512 bits/64 bytes big and 128 hex character long output. The public key is therefore subsequently calculated from the private key, but as we are dealing with a *trapdoor function*, this is not possible vice versa being an irreversible calculation. A private key is therefore kept non-public, whereas the public key is used to derive, in a third step - via Keccak-256 hashing of that public key, which results in a bytestring of length 32, from which the first 12 bytes are removed and result in a bytestring of length 20 - the individual ETH *address*, that could look like 0x9255bba2373BD2Ee6B05520DfD4dC571B682b349 - a 160 bits/20 bytes big and 40 hex character long output (leaving out the 0x prefix). A public key can hence be used to determine, that the given signature of the information at hand is genuine, that means created with the respective public key and address of the interactor, without requiring the private key to be divulged (see further appendix 7 and Buchanan, 2020).

7 Zero-Knowledge Proofs

The standard notion of a mathematical proof can be related to the definition of an non-deterministic polynomial (NP) time complexity class of decision problems. That is, to prove that a statement is true one provides a sequence of symbols on a piece of paper, and a verifier checks that they represent a valid proof. Usually, though, some additional knowledge, other than the sole fact that a statement is true, is gained as a byproduct of the proof. Zero-knowledge proofs were introduced as a way to circumvent that, i.e., to convey no additional knowledge beyond proving the validity of an assertion. Goldwasser, Micali, and Rackoff (1989) first described zero-knowledge proofs as interactive proof systems.

As the term itself suggests, interactive zero-knowledge proofs require some interaction between a prover and a verifier. Intuitively, a proof system is considered zero-knowledge if whatever the verifier can compute, while interacting with the prover, it can compute by itself without going through the protocol (Goldreich and Oden, 1994). Formally they can be defined as follows, as per Boaz and Sanjeev (2009).

Given an interactive proof system, or an interactive protocol (P, V), where P and V can be seen as interactive probabilistic polynomial-time (PPT) Turing machines symbolizing a Prover and a Verifier, for a formal NP-language $L \subset \{0,1\}^*$ and an input x, the *output* of V on x at the end of interaction between P and V can be written as

$$out_V[P(x), V(x)].$$

(P, V) is called a zero-knowledge protocol for L if the following three conditions hold:

Completeness:

$$\forall x \in L, u \in \{0,1\}^*, \qquad Pr[out_V[P(x,u), V(x)]] \geq 2/3,$$

where u is a certificate for the fact that $x \in L$. In other words the prover can convince the verifier of $x \in L$ if both follow the protocol properly.

Soundness: If $x \notin L$, then

$$\forall P^*, u \in \{0,1\}^*, \qquad Pr[out_V[P^*(x,u), V(x)]] \leq 1/3.$$

I.e., the prover cannot fool the verifier, except with small probability.

Perfect Zero-Knowledge: For every strategy V^* there exists an expected PPT simulator S^* such that

$$\forall x \in L, u \in \{0,1\}^*, \qquad out_{V^*}[P(x,u), V*(x)] \equiv S^*(x).$$

The last condition prevents the verifier from learning anything new from the interaction, even if she does not follow the protocol but rather uses some other strategy V^*. Otherwise she could have learned the same thing by just running the simulator S^* on the publicly known input x. S^* is called the simulator for V^*, as it simulates the outcome of V^*'s interaction with the prover without any access to such an interaction.

8 Understanding Smart Contracts' Network

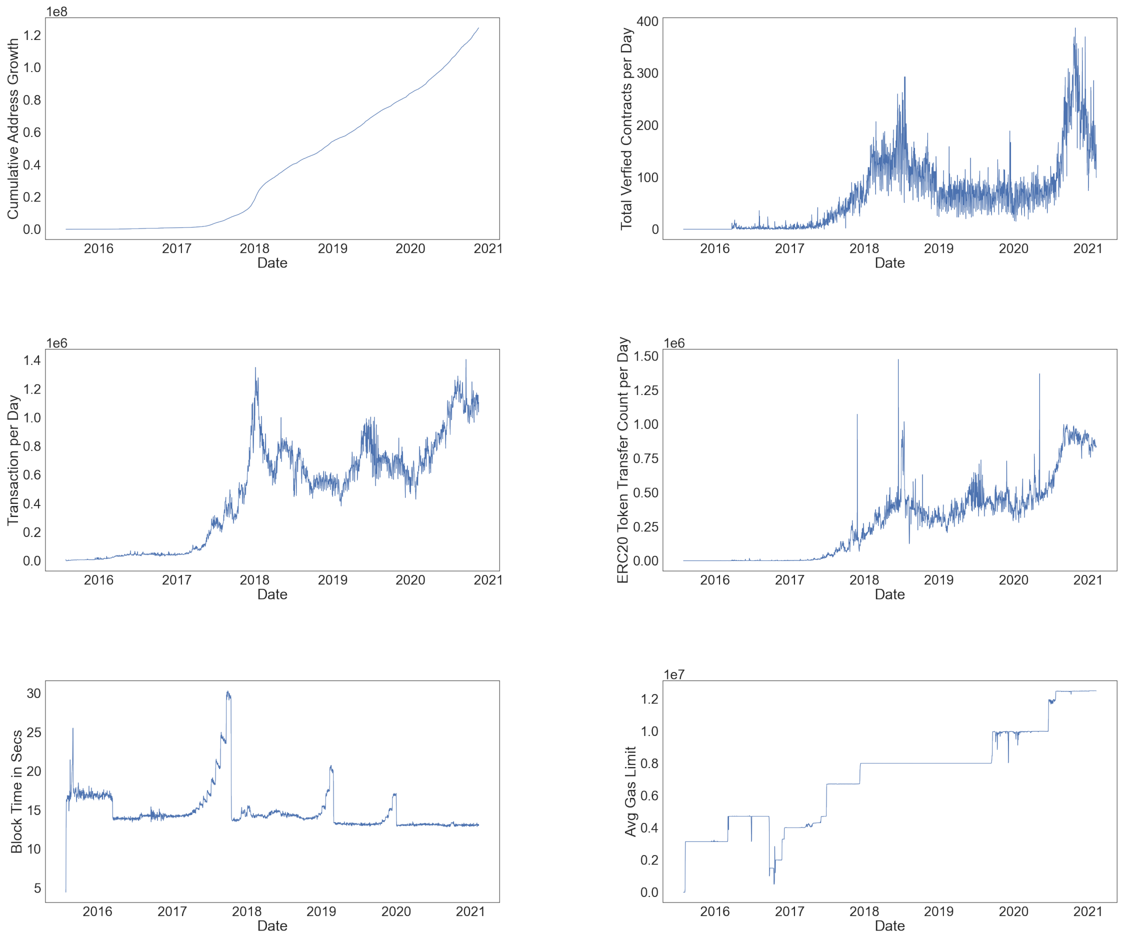


Figure 8: Network Parameter time series, 20150630-20210211

9 t-SNE (t-Distributed Stochastic Neighbor Embedding)

t-SNE is a non-linear technique for dimension reduction and data visualisation. it allows to preserve the local structure and is proposed by v. d. Maaten and Hinton (2008). It aims to design an embedding of high-dimensional input to low-dimensional map while preserving much of a significant structure. On the algorithm 1 below, the reader can find the pseudocode of the t-SNE computation.

$$X = \{x_1, x_2, \dots, x_n\} \rightarrow Y = \{y_1, y_2, \dots, y_n\}$$

x_i is the i^{th} object in high-dimensional space.
y_i is the i^{th} object in high-dimensional space.

Algorithm 1 t-SNE Pseudocode

```
Data: data set $\chi = \{x_1, x_2, \dots, x_n\}$
cost function parameters: perplexity $Perp$,
optimization parameters: number of iterations $T$, learning rate $\eta$, momentum $\alpha(t)$
Result: low-dimensional data representation $Y^{(T)} = \{y_1, y_2, \cdots y_n\}$
begin
    compute pairwise affinities $p_{j|i}$ with perplexity $Perp$ (using Equation 1)  set $p_{ij} = \frac{p_{j|i}+p_{i|j}}{2n}$  sample initial solution $Y^{(0)} = \{y_1, y_2, \dots, y_n\}$ from $N(0, 10^{-4}I)$
    for $t = 1$ to $T$ do
        computer low-dimensional affinities $q_{ij}$ (using Equation 2))
        compute gradient $\frac{\delta C}{\delta Y}$ (using Equation 3))
        set $Y^{(t)} = Y^{(t-1)} + \eta\frac{\delta C}{\delta Y} + \alpha(t)(Y^{(t-1)} - Y^{(t-2)})$
    end
end
```

$$p_{j|i} = \frac{\exp(-\|x_i - x_j\|^2/2\sigma_i^2)}{\sum_{k\neq i}\exp(-\|x_i - x_k\|^2/2\sigma_i^2)} \tag{1}$$

$$q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_{k\neq l}(1 + \|y_i - y_l\|^2)^{-1}} \tag{2}$$

$$\frac{\delta C}{\delta y_i} = 4\sum_j (p_{ij} - q_{ij})(y_i - y_j)(1 + \|y_i - y_j\|^2)^{-1} \tag{3}$$

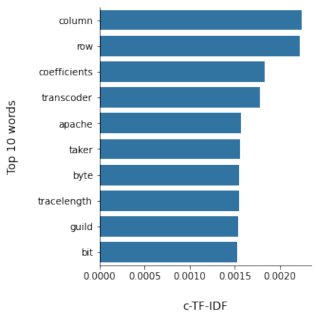
10 UMAP vs. *t*-SNE

The dimensionality reduction algorithm that we are using here is the Uniform Manifold Approximation and Projection for Dimension Reduction (UMAP) (McInnes, Healy, and Melville, 2018) and is like t-SNE – a neighbor graph algorithm. The mathematical foundations of the UMAP rely on Laplacian Eigenmaps and are very extensive. The important differences between t-SNE and the UMAP are the following.The UMAP aims to better preserve more of the global structure while requiring less computational time. As compared to the t-SNE, the UMAP relies not only on the the Kullback-Leibler divergence measure, but on the cross-entropy.

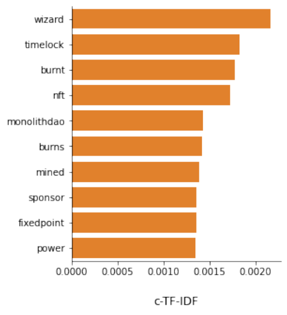
$$C_{UMAP} = \sum_{i \neq j} \left\{ v_{ij} \log \left(\frac{v_{ij}}{w_{ij}} \right) + (1 - v_{ij}) \log \left(\frac{1 - v_{ij}}{1 - w_{ij}} \right) \right\} \tag{4}$$

where v_{ij} are the pair-wise similarities in the high dimensional space and w_{ij} - in the low-dimensional. The optimization problem used in the UMAP is the stochastic gradient descent instead of gradient descent used in the t-SNE, which speeds up the computations and decreases the required memory resources. Moreover, UMAP does not require the distance Euclidean.

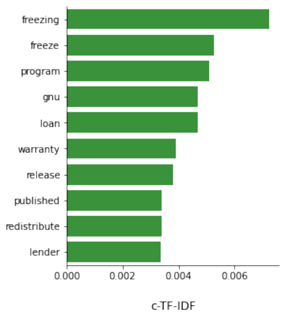
11 Topics in the Unlabelled Smart Contracts



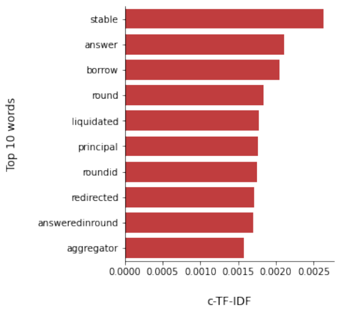
(a) Topic 1



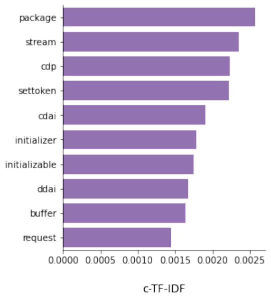
(b) Topic 2



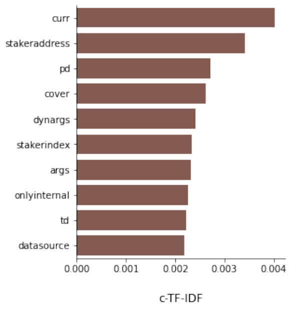
(c) Topic 3



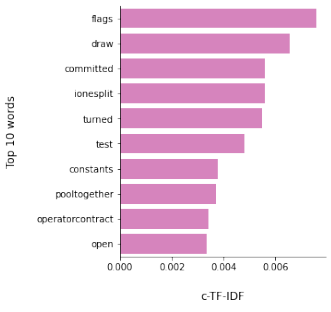
(d) Topic 4



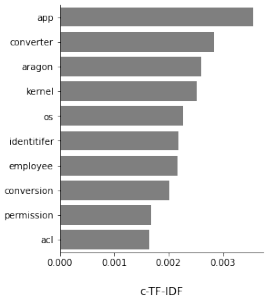
(e) Topic 5



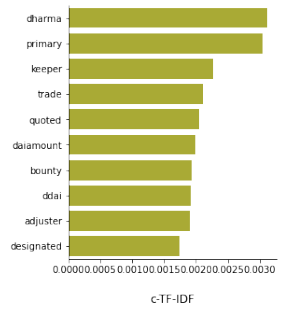
(f) Topic 6



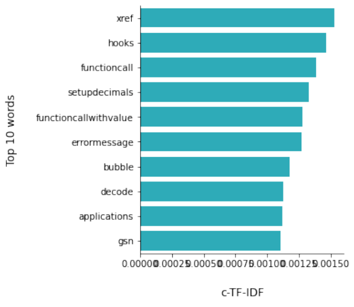
(g) Topic 7



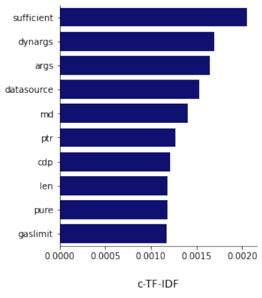
(h) Topic 8



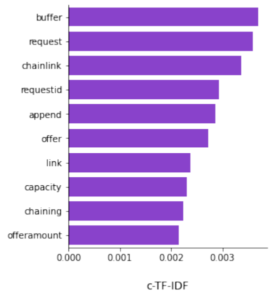
(i) Topic 9



(j) Topic 10



(k) Topic 11



(l) Topic 12

Figure 9: Top 10 the most important words per topic identified in the unlabelled SC dataset with 12 topics

12 Manual Classification of Solidity contracts

In this section, we present our first part of the classification analysis of SCs. We randomly picked the source codes of 800 verified SCs provided by Etherscan (2019), using the functions of random sampling in Python, and tried to group the source codes into categories by looking at the code, reading the comments, searching for titles of the SC. The 9 categories that we came up with are:

- **Token creation**, **Token sale** or **Token creation/Token sale**: SCs sole purpose is to define a token, or its crowdsale, or both.
- **Finance** or **Token creation/Finance**: whenever an SC has an implementation of a financial product or a derivative, a bidding mechanism, a lottery, a multisignature wallet, or in general any profit-creating mechanism. A token can be implemented in a separate SC and be evoked, or defined in a Finance SC directly.
- **Airdrop** or **Token creation/Airdrop**: an implementation of a token airdrop, sometimes simultaneously with a token definition. An airdrop is a free distribution of a token, unlike a crowdsale.
- **DApp**: an implementation of an Ethereum DApp.
- **Scam**: pyramid schemes, or financial SCs that offer e.g. 5 percent daily interest.
- **Utility**: diverse helper functions, pieces of code that are necessary for running actual SCs, extensions, libraries, etc.

Obviously, these categories are slightly different then the ones coming in the SDA data. Only the category **Finance** came up in the manual selection as well in the dataset from SDA. Moreover, just by looking at the code ourselves we were not always able to identify the category of the code or what DApp's category it belongs to. Interestingly, some of the categories were identified through meaningful comments or naming the product. Sometimes however, it required additional search to specify what the code is about. Thus, it is not always possible to accurately identify the category of the SC without using additional check up by a human-being. It requires a lot of engineering and Solidity domain expertise, which makes manual labeling more difficult or at least its scaling and outsourcing. On the other side though, it might happen that automatic classifiers would pick up the features that we do not see as a human.

13 Legal Documents and their Signing

The United Nations have worked on respective measures to replicate traditional frameworks for digital societies since at last 1996, resulting in a Model Law on Electronic Signatures that is of importance due to the beforehand mentioned algorithmic uniqueness of private keys/signatures (UNICITRAL, 2001). The United States of America's

Uniform Electronic Transactions Act (UETA) from 1999, and subsequently the United States Electronic Signatures in Global and National Commerce (ESIGN) Act from the year 2000, present a national legal solution to the issues raised by a digital society. They grant legal recognition to electronic signatures and records, given that the contractual parties choose to use electronic means for their agreements. Besides special requirements, such as having the intent to sign such an agreement and common consent to do business electronically, the requirements of associating the signature with continuously updated records that can be fully accessed, plays into the hand of BC-based constructions such as SCs. Under this legislation, in order to be accepted as a prima facie effective agreement, it is mandatory to have an associated record reflecting how the signature was generated or an attached proof that this agreement was reached by using electronic signatures. Moreover, the electronic signature records need to be capable of retention and accurate reproduction – once again, basic functions provided by BC-based systems. We can conclude that certain legal frameworks have been established that can be used directly or in an analogue way for SCs. BC users can: 1) be uniquely identified and linked to a signature, 2) have sole control over the genesis of their signatures, 3) identify accompanying signed data stored on the BC, 4) easily prove the event that the accompanying data has been changed and revoke their consent.

This leads to the possibility to also accept digital signatures of uniquely identifiable BC participants as legally valid signatures for contracts – in that sense SCs. Therefore, while certain steps towards the digitization of TCs have been accomplized, SCs can only be defined as technological means to facilitate TCs and not as contracts *sui generis*, or *authenticated agreements*.

14 Legal Conflicts and SC *smartness* versus the BC

A failsafe "bugout" solution in the hands of a trustful third party would be a solution, which however would make it a circle argument for some governmental backing of such systems to induce trust into the *intermediary-less BC trust machine*. Plenty of legal literature and decisions are available on non-/wrong-fulfillment of any kind of agreements, ranging from, for example, an automobile being delivered in the (partially) wrong color, to not having the expected quality of lacquer et cetera. These fine nuances are not representable by SCs unless the contractual parties agree *post id factum* to alter the SC – which is unfortunately only safely possible by deactivating the initial SC and by redeploying an altered one as a different instance. The fundamental problem of *legal effectiveness* of a *particular* contract also has to be taken into consideration, like for example, the legal age or a mental illness of a party, or if digital signatures can be accepted au lieu de physical signatures in a given case. However, these discussions revolve more about the regulation of oracles and, therefore, data service providers – a field that is not linked to SCs, but to classic service contract law for which a plethora of literature exists. It is *not* necessary to create yet another legal framework, as SCs are trying to be technological portrayals of their legal counterparts – a rental contract will still be a rental contract, just in a different presentation.

This serves to underline the redundancy of an elaborated exegesis of fulfillment problems in the context of SCs, as there are more legal problems that may hinder SCs to be classified as actual and effective contracts in their legal meaning. While an aficionado may certainly be able to write a technically effective smart "contract", it does not mean that this "contract" is legally effective, leave alone legally enforceable. Depending on respective legal constructions, a contract may also only be partially effective, i.e., the obligation to fulfill the contract may be ineffective, yet the transfer of ownership/property part of *the* contract may be effective. Also, the contract needs to follow a certain phenotype, like being in a certain language – where it can be up to discussion, if a programming language can be chosen given the envisioned type of agreement. In the end, this is defined by the already existing respective national limits of contractual freedom. Once again we recurse on the fact that a third party – possibly governmentally backed system is needed to solve problems if an agreement should *ex tunc* be deemed as *ineffective* as BC-based actions are considered to be eternal. This is a question primarily referring to technical solutions to make such constructions post factum editable (hence against the initial idea of BC), as having, for example, illegal content on a BC could potentially invalidate it completely for further usage.

The solution to fork the BC before this event and to abandon the "illegal BC" is highly problematic given governmental power and activity structures on such a system. The idea of ETH's SCs to grant different *states* (stages of contractual fulfillment so to say), proposes a solution if a *fallback function* to a given previous state or *reverse TX* can be coded. Courts often use the "doctrine of reasonable expectations" as a fallback function and justification for invalidating parts or all of a given agreement: the weaker party will not be held to adhere to contract terms that are beyond what the weaker party would have reasonably expected from the contract, even if what the party reasonably expected was outside the strict letter of agreement. The necessary ambiguity in the TC and SC situation alike arises when there are plausible and competing interpretations of a certain term. Given that the contractual parties can choose the language of the agreement in certain legal frameworks – may it be French, Chinese, or Solidity – it is important to note that this exemplary doctrine is not a rule granting substantive rights to any party if there is no doubt about the meaning of the used terms. Respectively, *clickwrap/-through agreements* can pose considerable sources for problems, especially when thinking of *adhesion clickwrap*. In a traditional context, these can be invalidated, which is not as easy to do in SC/BC software code cases. As the parties agreed to use a particular language, both parties are seen as being able to understand what is written in the agreement. The limit of freedom of contract, however, is obviously reached when the counterparty does not understand the given agreement at all and is respectively exploited. Unlike natural language on physical paper, it is harder to hide nefarious proceedings in standardized code forms, as both parties can de facto read the source code in any font they want and can check for all functions present to not be of disadvantage for them. Unless the disadvantageous party can prove to have been in a state of mental distress or similar while agreeing to this code, the only way out would be through *bona fides* rulings – which in itself are not helping the idea of legal security and may infringe on the freedom of contract.

As long as this level of *smartness* is not present with SCs, the solution to most of these problems is not to employ an armada of legal scholars and practitioners to forcibly create completely new legal frameworks but to employ the already existing legal knowledge analogously rewritten as these have proven their efficiency to eventually serve as *ius consuetudinis* with specialized international courts. Given the scope of applicability and the inherent nature of these agreements, we can obviously observe that unfair and surprising outcomes, lack of notice or understanding of clauses can cause problems especially, when considering SCs as replacements for TCs. There exists a potential for unconscionability if the TC and SC do not represent equal information. However, such an event of discrepancy and ambiguity will be resolved contra proferentem against the party drafting the contract and its mean of execution – again, there is no need to specialised codices. The European Union most notably works around these classic issues, for example, in the Rome I and Rome II regulations (see further discussion e.g., Rühl, 2021).

15 Legal Example

As a hands-on-example, let us look at a SC as a mean to execute a long term flat rental TC under German law: Landlord (L) and Tenant (T) sign a TC on renting an unrenovated ground floor road-view flat at BC-wonderland Berlin "X-Berg"'s Tourist and Club area, i.e., EUR 2.000,– gross warm rent for 45 square meters, on the 01. August 2021. The flat sports a well maintained and shiny smart-lock belonging and connected to a dedicated and extremely hyped Berlin-Startup (S) BC system that runs exclusively for such businesses – obviously L and T agree in the TC to have an SC to manage respective value flows, as this is portrayed as saving them money and time, instead of making a musty repeating standing order via online banking.

After they have forwarded all their contractual, professional, and private information, besides creditworthiness et cetera, to the service provider to set up the SC, the complete heating system collapses on the 01. November 2021 and is not fixed for the whole month. This is the first time that T realizes that the SC has no function coded for any kind of such rent reduction due to heating shortages – the smart meters on the heating devices are not linked to the SC via an Oracle to cut on respective costs, as it is expected that lodgers in this area will always be hot. T declares to L a rental reduction of 100% for the month of November, as he was not able to use the flat accordingly – a shortcoming in the sense of § 536 Par. 1 S. 1 Bürgerliches Gesetzbuch (BGB; German Civil Code). As the L does not react and the service provider S is not reachable, T addresses his credit institute – from where the SC automatically deducted the full rent as advised repeatedly – to cancel the permission to have any funds going off towards this address. Meanwhile, T tries to reach L and S, the SC is unable to automatically deduct the rent from the banking account of T on the 01. December and locks T out of the flat via a coded command to the smart-lock. After T – who suspected a typical Berlin-esque "Digital Advancement" technical error – had the door broken open by a locksmith and having the smart-lock replaced by a conventional lock, the S tries to call T why the data-stream from the device has broken off – just seconds before the L also tries to call T after having received an automatic alert message, that T has not paid the rent.

The days pass – S has gone bankrupt in the meantime due to an overflow of unpaid marketing bills – and L prompts T – who has now access to the flat via the installed traditional lock-key-system and transfers the rent via a classic standing order – to pay the missing rent for October 2021 on the firsts of December 2021, January and February 2022. Eventually, L files a lawsuit against T in March 2022 to pay the rent for October 2021 stating that the SC is an exhaustive and final provision regarding the rights and obligations of this rental contract. T eventually terminates the TC properly and moves back to his Uckermark home village in April 2022 after not becoming a successful Avant-garde fashion designer.

In the following, we provide a rough sketch on how this case of interaction between SC and TC would be handled legally in Germany and programmatically respectively in adjacent legal system constructions:

I. Admissibility of the lawsuit of L against T to pay the rent for November 2021

1. Juristiction

Given the TC and missing any contradicting information — the SC is completely irrelevant — the Amtsgericht Tempelhof—Kreuzberg is factually and locally responsible for this case, § 23 Nr. 2a Gerichtsverfassungsgesetz (GVG; Courts Constitution Act) in conjunction with

§ 29a Par. 1 Zivilprozessordnung (ZPO; German Code of Civil Procedure).

2. Defensible Interest

In case one should see an SC as a fully viable replacement for a TC, then one could argue, that there could be an easier way of justice than going to the courts and have a lengthy process. As we do not see SCs as equal to TC, but merely as a mean of execution, we do not need to discuss "self executing and enforcing justice".

II. Reasonableness of the lawsuit of L against T to pay the rent for November 2021

Conditioned that L's claim to pay the rent for October 2021 based on the rental TC holds.

1. Contractual Base

Again, we do not consider an SC to be a contract in the legal sense, hence only the TC is important here with no contradicting information regarding issues with the TC. The SC is just a mean of execution of the TC. Therefore the plead of L, that the SC is an exhaustive and final provision is meaningless.

Thinking of an SC of being just a mere mean on execution, problems regarding a potentially required written form of an agreement are of redundant nature. Importantly, it is commonly agreed, that eMails, Telefaxes, or Computerfaxes do not hold up to fulfill the required written form.

On the other hand, if one would think of an SC being a replacement of a TC, one would pay attention to respective regulations on the way the agreement is presented, i.e., the written form requirement of § 126 Par. 1 BGB for a contract in the sense of § 550 S. 1 BGB can be replaced according to § 126 Par. 3 BGB given that respective signatures in the sense of § 126a BGB are given (recall our multiple hints towards the importance of signatures in a technical (digital signature) and legal (electronic signature) sense – nevertheless a missing written form according to § 550 S. 1 BGB does not provoke a nullity according to § 125 S. 1 BGB, but just an indefinite rental period of time in this particular case).

The full legal reference of §§ 126a, 126 Par. 3, 127 Par. 1 and 3 BGB however hints towards the border of the freedom of contract in the sense of §§ 13, 125 BGB, as a given law may define this otherwise. Examples would be the §§ 484 Par. 1 S. 2, 492 Par. 1 S. 2, 623 Half–S. 2, 766 S. 2 BGB, or the § 780 S. 2, 781 S. 2 BGB.

2. Rights based on Irregularity In Contractual Performance

Rights resulting from an irregularity in contractual performance can be called given that a contractual performance is not or not as owed given.

In this case, the T ows the L to rent for November 2021. We could consider that T exercised his right to set–off this obligation according to § 389 BGB with the rent of October 2021 given that he has a right to reduction.

The SC itself does not have any other function besides checking for cash flows and smart–lock data stream – it is obvisouly not *that* smart as it will never be able to represent the status of the rental flat, unless that flat if full of sensors and these are connected flawlessly to a respectively coded SC. This, however, does not change anything in the rights a tenant has by law. Here, as T was unable to live in the flat for the month of November 2021, he has the right to reduce the rent of November 2021 by 100% according to § 536 Par. 1 BGB and to set–off this position according to § 389 BGB. L can not demand the rent for November 2021.

L's statement, that the SC is an exhaustive and final provision is on one hand irrelevant, as they also signed a TC for which the SC is only the mean of execution, and the rights given by law to T are not of a dispositive nature. In general, any contractual agreement needs to stand up to the questions of, for example, if they stand up against control towards their legality.

3. Cancellation of Contract

The right to exercise the termination of a contract is a right to influence a legal relationship just like a contestation, a revocation, or others besides § 134 BGB. As outlined beforehand, if an SC should be seen as a complete surrogate for TCs, there can be issies if a respective cancellation is not coded or not coded properly.

Given that an SC is not coded properly, the respective legal rights are not superseded by them missing in an "exhaustive and final provision" as outlined above regarding the reduction of the rent or the extraordinary termination of the agreement. In our example case, the T could have terminated the TC immediately in December when the smart–lock did not let T in the flat which represents an important issue according to § 543 Par. 1 & Par. 2 S. 1 BGB, as the respective requirements for locking a tenant out of a flat via a forced eviction or clearance order according to § 940a ZPO were missing.

III. Conclusion

The abovehand outlined case would be admissible to the courts, but it can not be seen as being reasonable.

Should one see an SC as a replacement for a TC, without a given legal foundation that identifies this equally, then one could argue, that in the case of problems arising the people utilising this framework willingly let go of their otherwise existing frameworks of rights — caveat emptor. One can compare that to illicit employment or moonlighting to save on taxes and issues resulting from a normally given warranty for defects or in the case of the customers insolvency. Using surrogate systems to circumvent the existing regulated system does not deserve to be protected.

At least the German legal system is not in need of a specialised BC—law or similar lex cryptographia creatures, as it is an effective tool given its abstractness to adapt to a given case, as well as given its neutrality towards any kind of technological advances, may they have been electronic signatures in the past or BC—systems at the given time of this writing.

With this rudimentary example – we are not going deeper into any claims of T against L (e.g. § 231 BGB due to being locked out – beyond §§ 858 ff. BGB), or of L against S (e.g. consequential damage in the sense of § 280 Par. 1 BGB due to a primitively coded SC), or of S against L and/or T (e.g. due to the willingly damaged smart-lock in the sense of the civil and penal law), et cetera – we can quite easily present the frictions between the "vision" and the reality of SC in other realms than being a financial vehicle like peer-to-peer lending or data transmitter in areas such as say renewable energy Prosumer situations.

We want to underline, that law enforcement and legal security do not work in the SC-framework we have at hand at the time of writing this – especially when thinking about, for example, §§ 273, 320, 229, 230, 539 Par. 2, 997 BGB, or §§ 765a, 811 ZPO. Hence (nearly) every thinkable contractual construction will run into problems when being pushed in an SC application. Gravely, the exclusive right of courts to rule on what is eventually right or wrong is circumvented and can lead to unjust overenforcement, like locking the tenant out of the flat in the above shown example. Legal systems all over the world, and to our knowledge especially such coming from a Franco-Germanic genesis, are clear enough on what individual rights are and on what these individual rights borders are – to this point, a *embedded legal knowledge* can not be represented by SCs.

16 Example Code

Adding to the source code review and possible use cases, we are presenting a simple *Hello World*-esque SC as provided by the Ethereum Github with explanatory adaptions to outline some of the technical complexity that has a grave impact on every adjacent structure. We have observed many non-technical outlets – especially Blogs and the such – discussing a theme, that they have apparently never seen as code itself – consequently, we will also keep this to a very brief overview to introduce the structures in a very simple example. Each comment starts with /** and ends with /*.

```
    /** Version control for the compiler, see above section 6, as each Solidity version may have
        ↪ different commands that can be coded — higher versions will obviously improve
        ↪ efficiency of the coding and lead to better code controlling. /*
pragma solidity >=0.4.22 <0.6.0;

    /** "Mortal" is the name of this SC. /*
contract Mortal {
        /** Defines the variable "owner" of the type "address". /*
    address owner;

        /** The "constructor" is executed at initialization and sets the owner of the SC, i.e., it is
            ↪  executed once when the CA/SC is first deployed. Similar to other class−based
            ↪ programming languages, it initializes state variables to specified values. "msg.
            ↪ sender" refers to the address where the CA is being created from, i.e., here in the
            ↪  constructor setting the "owner" to the address of the SC creator. SCs depend on
            ↪  external TX/MSG to trigger its functions, whereas "msg" is a global variable
            ↪ that includes relevant data on the given interaction, such as the address of the
            ↪ sender and the value included in the interaction. This is assured by the "public"
            ↪ function, which can be called from within the CA/SC or externally via MSG's,
            ↪ like here getting the address of the interactor. "private" functions are not callable
            ↪  and can only reached by the SC itself — a particular source for grave errors, as
            ↪ you can not change the SC once deployed. /*
    constructor() public { owner = msg.sender; }

        /** Another important function, and source for grave errors if missing, follows and
            ↪ represents a mean to recover funds stored on the CA. Alternatively, calling "
            ↪ selfdestruct(address)" sends all of the SCs current balance to address specified.
            ↪ Remember, that once deployed the SC can not be changed unlike non−BC−
            ↪ based software. The only way to modify an SC is to deploy a corrected one —
            ↪ best after deactivating and recovering all funds in the problematic one.
            ↪ Interestingly, "selfdestruct" consumed "negative Gas", as it frees up BC/EVM
            ↪ space by clearing all of the CA/SCs's data./*
    function kill() public { if (msg.sender == owner) selfdestruct(msg.sender); }
}



    /** After "Mortal", "Greeter" is another SC presented to visualize, that CA/SCs can "inherit
        ↪ " characteristics of CA/SCs enabling SCs to be written shorter and clearer. By
        ↪ declaring that "Greeter is Mortal", "Greeter" inherits all characteristics of "Mortal"
        ↪ and keeps the "Greeter" code herewith crisp and clear to to point, where is has
        ↪ individual functions to be executed. In this example, the inherited characteristic of "
        ↪ Mortal" gives, as defined beforehand in "Mortal", that "Greeter" can be deactivated
        ↪ with all locked funds being recovered. /*
contract Greeter is Mortal {

        /** Defines the variable "greeting" of the type "string", i.e., a sequence of characters. /*
    string greeting;

        /** This is defined as beforehand in "Mortal", whereas in this case the underscore in "\
            ↪ _greeting" is a style used to differentiate between function arguments and global
            ↪ variables. There is no semantic difference between "greeting" and "_greeting",
            ↪ whereas the latter one is defined as such not to shadow the first one. Here, the
            ↪ underscore differentiates between the global variable "greeting" and the
            ↪ corresponding function parameter. Strings can be stored in both "storage" and "
            ↪ memory" depending on the type of variable and usage. "memory" lifetime is
            ↪ limited to a function MSG and is meant to be used to temporarily store variables
            ↪ and respective values. Values stored in "memory" do not persist on the network (
            ↪ EVM & BC) after the interaction has been completed. /*

    constructor(string memory _greeting) public {
        greeting = _greeting;
    }

        /** Main function of the SC that returns the greeting once "greet" function is MSG'ed
            ↪ /*
    function greet() public view returns (string memory) {
        return greeting;
    }
}
```

References

Boaz, B. and A. Sanjeev. 2009. *Computational Complexity: A Modern Approach*. Princeton University: Cambridge University Press, 1st edition ed.

Buchanan, W. 2020. "Ethereum Address Generation, Asecuritysite." URL `https://asecuritysite.com/encryption/ethadd`. Online; accessed 10 January 2021.

Davies, D. and D. Bouldin. 1979. "A cluster separation measure." *IEEE transactions on pattern analysis and machine intelligence* (2):224–227.

Etherscan. 2019. "Etherscan verified contracts." URL `https://etherscan.io/contractsVerified`.

Goldreich, O. and Y. Oden. 1994. "Definitions and properties of zero-knowledge proof systems." *Journal of Cryptology* 7:1–32.

Goldwasser, S., S. Micali, and C. Rackoff. 1989. "The Knowledge Complexity of Interactive Proof Systems." *SIAM Journal on Computing* 18:168–298.

McInnes, L., J. Healy, and J. Melville. 2018. "Umap: Uniform manifold approximation and projection for dimension reduction." *arXiv preprint arXiv:1802.03426* .

Rühl, G. 2021. "Smart (Legal) Contracts, or: Which (Contract) Law for Smart Contracts?" In *Blockchain, Law and Governance.*, edited by B. Cappiello and G. Carullo. Springer International Publishing, 159–180. ISBN 978–3–030–52722–8.

Thorndike, R. 1953. "Who belongs in the family?" *Psychometrika* 18 (4):267–276.

UNICITRAL. 2001. "UNCITRAL Model Law on Electronic Signatures with Guide to Enactment 2001." URL `http://www.uncitral.org/pdf/english/texts/electcom/ml-elecsig-e.pdf`. Online; accessed 10 January 2021.

v. d. Maaten, L. and G. Hinton. 2008. "Visualizing data using t-SNE." *Journal of machine learning research* 9 (Nov):2579–2605.