Why CBDCs will likely not support full smart contracts

Siebenbrunner, Christoph; Taudes, Alfred

Published: 01/01/2024

Christoph Siebenbrunner
Alfred Taudes

WU WIRTSCHAFTS UNIVERSITÄT WIEN VIENNA UNIVERSITY OF ECONOMICS AND BUSINESS

Why CBDCs will likely not support full smart contracts

Working Paper Series
01 / 2024

WU VIENNA | INTERDISCIPLINARY RESEARCH INSTITUTE FOR CRYPTOECONOMICS

Why CBDCs will likely not support full smart contracts

Christoph Siebenbrunner* Alfred Taudes,†

Abstract. Virtually all proposals for central bank digital currencies (CBDCs) include a provision for account limits. They are meant to protect the regular banking system from bank runs into the CBDC, and for this reason, they are likely to remain a permanent feature. We argue that account limits are at least practically not compatible with smart contracts like those offered by systems like Ethereum. This means that CBDCs will likely not be able to offer the same extent of programmability features as blockchains.

KEY WORDS

1. Smart Contracts. 2. Central Bank Digital Currencies. 3. Programmable Money. 4. Programmable Payments.

1 Introduction

The concept of smart contracts, initially introduced by Nick Szabo (1994,[1] 1996[2]), has undergone a significant evolution since its inception. Originally described as a "*protocol that executes the terms of a contract*", the term has been applied to a broad set of mechanisms ranging from vending machines to more complex, technology-driven contractual agreements. The advent of Ethereum[3] marked a pivotal moment, repurposing the term within the context of blockchains and distributed ledger technology (DLT). Smart contracts have since become synonymous with the often immutable code that resides and is executed on these decentralized platforms.

Arguably the most innovative aspect of the modern incarnation of smart contracts is their potential to act as credible commitment devices. This characteristic is particularly salient from a game-theoretical perspective, where smart contracts can guarantee consistent execution of code for all parties involved, akin to an impartial enforcer of rules. Hall et al. (2021)[4] delve into this by exploring the equilibria in games with the possibility of multiple, interacting smart contracts, examining their computational properties and strategic implications. Similarly, Chitra et al. (2023)[5] demonstrate how the credible commitment property of smart contracts can be harnessed to implement theoretically optimal auctions. While smart contracts provide a technical framework for credible commitments, their status as contracts in the legal sense is not as straightforward. Goldenfein and Leiter (2018)[6] and Vigliotti (2021)[7] discuss the nuances of smart contracts within the legal domain, suggesting that, while they may not constitute legal contracts in isolation, they

* Christoph Siebenbrunner (christoph.siebenbrunner@wu.ac.at) Research Institute for Cryptoeconomics, WU Vienna. All views expressed herein are those of the authors.

† Alfred Taudes (alfred.taudes@wu.ac.at) Research Institute for Cryptoeconomics, WU Vienna

can form integral parts of legally binding agreements. Clack et al. (2016)[8] provide concrete suggestions for bridging the gap between the technical execution of smart contracts and their legal enforceability by providing smart contract templates that can be used as part of legally binding contracts. From an economic standpoint, the implications of smart contracts as credible commitments are profound. Gans (2018)[9] shows that smart contracts expand the realm of feasible contracts, offering the potential for welfare enhancements e.g. through reduced frictions in international trade. Yermack (2017)[10] suggests that smart contracts could improve corporate governance through enhanced transparency. Additionally, Goldstein et al. (2019)[11] provide a practical example of how smart contracts could be designed to curtail rent-seeking behaviors in platform economies, highlighting their potential to foster competition and economic efficiency.

Related to smart contracts is the more recently coined term of *programmable money*. In this context, Sandner et al. (2021)[12] highlight the need to distinguish between programmable payments and programmable money. Programmable payments enhance existing payment infrastructures by allowing to attach conditionality to payments, akin to an advanced form of standing orders. While such conditions may also refer to stipulations from legally binding contracts, programmable payments differ from smart contracts in that they do not provide credible commitments: they serve as an automation tool for the sender, but there are no guarantees that the payment will indeed be executed when the condition is met (e.g. because the order has been cancelled after the contract has been signed, or the sender does not have a sufficient balance when the condition is met). Other uses of the terms programmable or conditional payments[13,14] also refer to the restriction of the uses of funds imposed by the sender or a central operator (e.g. certain funds may not be used to purchase certain goods). Programmable money, on the other hand, has an inherent logic that may again enable credible commitments. Lee (2021)[15] and George et al. (2023)[16] discuss the properties and possible definitions of programmable money. George et al. (2023) provide a definition of programmable currency as being comprised of four elements, (1) storage of both data and value, (2) instructions, (3) a coherence guarantee that should convince participants or third parties that the program will execute as specified, and (4) a set of rules governing who is allowed to create, call and verify programs. We will follow this definition and refer to smart contracts as programs with a coherence guarantee in the sense of George et al. (2023),[16] with one specification: George et al. (2023)[16] allow for restrictions on the creation of smart contracts – we say that a system supports *full* smart contracts if regular users have unrestricted ability to create smart contracts, as is the case for virtually all smart-contract-enabled DLT systems such as Ethereum.

In this paper, we will argue that allowing users to create smart contracts introduces complications that make this practically incompatible with the concept of account limits, an issue that has so far been overlooked in the ongoing debate to the best of our knowledge. In chapter 2 we will explain why this is the case, and in chapter 3 we will discuss the important practical and policy implications of this finding, which are particularly salient for central bank digital currencies (CBDCs). chapter 4 concludes.

2 Smart contracts and account limits

Before we can discuss the difficulties from combining smart contracts with account limits, we need to define what we mean by account limits. There are many design options, but at a minimum,

a monetary system that wants to implement account limits will need to define entities to which the limit applies and be able to determine the current balance of any account at any point in time, in order to enforce the limit. The limit itself may come in various forms, it may apply to all or some of the system participants, homogeneously or heterogeneously, it may enforce a cap that must not be exceed at any time or at certain points in time, or averaged across some time periods. Design choices are numerous and most of the discussion in this chapter applies irrespective of the concrete design choice. To be clear, we will assume a very straightforward account limit design with a hard cap that must be complied with by all participants at all points in time, with a dedicated 'overflow' mechanism where excess funds can be withdrawn to in real time. This is similar to the design choices made by CBDCs such as the Digital Euro.* While our arguments regarding account limits and smart contracts apply irrespective of the application, they are most relevant for CBDCs, as we will explain in the next chapter.

The reason why smart contracts are difficult to combine with account limits is that they allow parties to put funds into a *latent state*, in which ownership may be impossible to attribute. This can be exemplified through a simple example of a smart contract acting as an escrow: consider a smart contract that can take possession of a specified sum of money and a specified asset (which needs a digital representation that can be transferred by a smart contract, for example an NFT[19] or another tokenized asset). After either of the items has been transferred to the smart contract, it cannot be reclaimed up until a specified date. If at any time before that date the second item has been transferred to the smart contract as well, it will automatically send the specified amount of money to one specified party and the digital asset to another party. If the second condition has not been met before the specified date, any items sent to the smart contract will be returned to the original sender. Such a simple program could act as a practical automated escrow, and it highlights the problem with attribution of latent funds: suppose that at a given time before the date specified in the escrow account, the agreed-upon sum of money has already been sent to the smart contract, but the digital asset has not yet been transferred. It is at this point impossible for anyone but the owner of the digital asset to tell who will ultimately receive the money, as it depends on their choices. For a different smart contract, the ultimate recipient may be impossible to predict for anyone, e.g. if payout condition depends on exogoneous events (e.g. a weather insurance) or verifiable randomness (e.g. for a lottery smart contract).

We have seen that in the presence of smart contracts, it may be impossible to determine the owner of some funds at any given point in time. Any system that wants to combine a smart contract offering with account limits will thus have to take a stance on how to attribute ownership of such latent funds. We numerate all possible design options for this attribution:

1. Latent funds are not counted towards the balance of any end user.
2. Latent funds are counted towards the balance of the ultimate end user recipient.
3. Latent funds are attributed to the original sender until they end up in another end user's account, and they have no more control over the funds when they are in the latent state.
4. Latent funds are attributed to the original sender until they end up in another end user's account, but the sender is free to dispose of the funds otherwise up until the transfer has taken place.

We will now discuss these options in turn: the first option, not counting latent funds at all,

* Current proposals for the Digital Euro foresee an account limit for individuals,[17] rumoured to be around €3.000.[18]

would make it trivial to circumvent any holding limits by creating smart contracts to hold any excess funds, making it thus unpractical. The second option, counting funds towards the balance of the recipient, is not always possible, as evidenced by the escrow example above: in many smart contract applications, the ultimate recipient of funds depends on conditions that will only materialize in the future and cannot be determined beforehand. The third option of locking up latent funds for the sender while still counting them towards their balance may seem like the most practical solution, a closer inspection reveals a security vulnerability, however: the duration for which funds remain locked in a smart contract may be indeterminate, meaning that the account of the sender may remain partially or entirely locked for long or indeterminate periods. This may happen by accident, e.g. through a bug in the code, a malicious attacker may also create traps, enticing other users to send their funds to smart contracts that will never release them. From the perspective of the sender, such an attack would be worse than a simple theft, because not only are the funds lost, but their account's holding limit would be permanently reduced by that amount, possibly wiping it out entirely. Such an attack could undermine the liveness of the entire system. One possible solution to this problem could be to limit the duration for which smart contracts can keep funds in a latent state. This, however, would require removing the general computing capabilities typically associated with modern smart contracts, and it would also undermine many current use cases of smart contracts on DLT platforms like Ethereum, such as decentralized exchanges. The same applies if a central operator were given the ability to remove funds trapped inside smart contracts. The latter also comes with liability or reputation risks, as users may fraudulently claim that their funds are trapped and ask for them to be moved out of a smart contract. Smart contract creation may also be limited only to certain types or only allowed for certain parties, but this would not offer the same full smart contract functionality as DLT platforms, as defined in the introdcution. The fourth option, only debiting the sender's account when the conditions for the transfer to the recipient have been met, may seem like another possible solution in the face of these problems. Here we point to the distinction by Sandner et al. (2021)[12] again: such a mechanism would correspond to programmable payments, but it lacks the credible commitment property that we associate with programmable money or smart contracts.

In summary, the only options that could provide smart contracts with account limits would either have to come with significant security or liability risks or require limiting smart contract creation abilities to the point that the system can no longer offer full smart contracts such as those offered by DLT systems like Ethereum. We argue that none of the solutions discussed above is truly practical, especially in the context of central bank digital currencies, where the central operators would be central banks, who should generally aim to protect their credibility and independence.

3 Policy implications for CBDCs

Account limits are a rarity in the context of cryptocurrencies, not only because of the problems mentioned above, but also because pseudonymous wallets, standard for virtually all major blockchains, make the idea of account limits rather pointless. There are, however, currency products for which account limits are a standard design element, namely CBDCs. We will briefly explain the reason for this.

CBDCs differ from other digital money products such as online bank accounts in that they

are issued directly by the central bank. Bank deposits represent an unsecured claim against a commercial bank on which the bank can default. In the absence of other protection, it can be rational to participate in a bank run when the solvency of a bank is in question (Diamond and Dybvig, 1983).[20] Deposit insurance is the traditional solution to this problem, however, insured amounts are capped in many jurisdictions. Central bank money is different in that regard in that, while it is a liability of the central bank, generally nothing can be claimed for it (in the absence of a gold standard or similar agreements like the Bretton-Woods system, effectively abandoned in 1973). Conversely, this means that central bank money cannot be unbacked. It is the purest form of money and usually comprises the most narrow monetary aggregate, M0. This makes central bank money more attractive than bank deposits, especially during times of systemic crises. The only form in which central bank money is currently available to non-banks is physical cash, which comes with sufficient inconvenience to hold in large quantities that this is usually enough to prevent mass flight from commercial bank deposits into central bank money. This is about to change with the introduction of CBDCs, however. CBDCs, being central bank liabilities, by definition have the same monetary quality as cash, while also promising comparable convenience to online bank accounts. This means that CBDCs without account limits would create the risk of a *digital bank run*[21] where bank depositors may flee en masse from bank deposits into CBDC holdings during a crisis. Account limits on CBDCs are hence a practical measure to effectively mitigate this risk.

We have just seen that account limits on CBDCs are in place to mitigate a fundamental financial stability risk, and for this reason they are likely to be in place permanently rather than a temporary safety guard for a new product. But we have also seen in the previous chapter that account limits are not practically compatible with smart contracts as defined in the introduction. The important corollary from this discussion is that CBDCs will not be able to support full smart contracts, at least such smart contracts that can act as credible commitment devices and that constitute much of the innovation of DLT-based smart contracts. This insight is relevant because the debate about whether CBDCs should support smart contracts is ongoing,[12] and some countries like China already promote smart contract features of their CBDCs.[22] Regarding the latter, it should be noted that those programmability features of the Digital Yuan[23] would fall under programmable payments and not count as full smart contracts as defined in the introduction. We argue that CBDCs will likely permanently lack behind in programmability features compared to blockchains and other systems, leaving a market opportunity for private sector solutions such as stablecoins and tokenized bank deposits.

4 Conclusion

Follwing George et al. (2023),[16] we define smart contracts as programs that provide a coherence guarantee regarding their execution. We argue that this coherence guarantee is the main innovation of smart contracts, as it allows users to enter credible commitments, making smart contracts a possible replacements for services like escrows and promises enabling innovative approaches to corporate governance and reducing trade frictions. We argue that full smart contract features are not compatible with the concept of account limits, where the maximum holdings per user are capped. Solutions that try to combine account limits with smart contracts would either have to soften the account limit, introduce security vulnerabilities or liabilitiy risks or require

substantially limiting programmability features to the point that they can no longer be considered full smart contracts, such as those offered by DLT systems like Ethereum. The relevance of this finding lies particularly in the context of CBDCs. CBDCs generally feature account limits in order to prevent the risk of digital bank runs. Account limits are hence likely to be a permanent feature of CBDCs. Taken together, these insights imply that CBDCs will likely not offer full smart contract functionality. They may include various aspects of programmability, but the credible commitment property that comes with it is a separate concern that is not practically compatible with account limits.

Notes and References

[1] Szabo, N. "Smart contracts." (1994) (accessed 7 November 2023) https://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart.contracts.html.

[2] Szabo, N. "Smart Contracts: Building Blocks for Digital Markets." (1996) (accessed 7 November 2023) https://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart_contracts_2.html.

[3] Buterin, V. "Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform." (2014) (accessed 8 November 2023) https://ethereum.org/669c9e2e2027310b6b3cdce6e1c52962/Ethereum_Whitepaper_-_Buterin_2014.pdf.

[4] Hall-Andersen, M., Schwartzbach, N. I. "Game theory on the blockchain: a model for games with smart contracts." In *International Symposium on Algorithmic Game Theory* Springer 156–170 (2021) .

[5] Chitra, T., Ferreira, M. V., Kulkarni, K. "Credible, optimal auctions via blockchains." *arXiv preprint arXiv:2301.12532* .

[6] Goldenfein, J., Leiter, A. "Legal engineering on the blockchain:'Smart contracts' as legal conduct." *Law and Critique* **29** 141–149 (2018).

[7] Vigliotti, M. G. "What do we mean by smart contracts? open challenges in smart contracts." *Frontiers in Blockchain* **3** 553671 (2021).

[8] Clack, C. D., Bakshi, V. A., Braine, L. "Smart contract templates: foundations, design landscape and research directions." *arXiv preprint arXiv:1608.00771* .

[9] Gans, J. S. *The fine print in smart contracts*. *Technical report* National Bureau of Economic Research (2019).

[10] Yermack, D. "Corporate governance and blockchains." *Review of finance* **21.1** 7–31 (2017).

[11] Goldstein, I., Gupta, D., Sverchkov, R. "Initial coin offerings as a commitment to competition." *Available at SSRN* .

[12] Sandner, P. G., Groß, J., Chung, J.-C. "The Programmable Euro: Review and Outlook." *Available at SSRN* .

[13] Panetta, F. "The digital euro: our money wherever, whenever we need it." (2023) (accessed 8 November 2023) https://www.ecb.europa.eu/press/key/date/2023/html/ecb.sp230123~2f8271ed76.en.html.

[14] Ledgerinsights "Digital pound CBDC won't be programmable to avoid government control perceptions." (2023) (accessed 8 November 2023) https://www.ledgerinsights.com/digital-pound-cbdc-not-programmable-government-control/.

[15] Lee, A. *What is Programmable Money? Technical report* Board of Governors of the Federal Reserve System (US) (2021).

[16] George, N., Dryja, T., Narula, N. "A Framework for Programmability in Digital Currency." *MIT Digital Currency Initiative* (accessed 7 November 2023) https://static1.squarespace.com/static/59aae5e9a803bb10bedeb03e/t/64c91329bbcec415ae9b06e9/1690899242356/Public+Copy+-+A+Framework+for+Programmability+in+Digital+Currency+-+August+1st%2C+2023+-+MIT+DCI.pdf.

[17] "Digital Euro FAQs." (2023) European Commission. (accessed 22 December 2023) https://finance.ec.europa.eu/digital-finance/digital-euro/frequently-asked-questions-digital-euro-and-legal-tender-cash_en.

[18] TechXplore "EU moves closer to launching digital euro." (2023) (accessed 22 December 2023) https://techxplore.com/news/2023-06-eu-giant-digital-euro.html.

[19] ERC721 "EIP-721: Non-Fungible Token Standard." *https://eips.ethereum.org/EIPS/eip-721, accessed 2022/11/28* .

[20] Diamond, D. W., Dybvig, P. H. "Bank runs, deposit insurance, and liquidity." *Journal of political economy* **91.3** 401–419 (1983).

[21] Klein, M., Gross, J., Sandner, P. "The digital euro and the role of DLT for central bank digital currencies." *Frankfurt School of Finance & Management GmbH, FSBC Working Paper* .

[22] Lanxu, Z. "Digital yuan solutions enjoy major upgrade." (2023) (accessed 8 November 2023) https://www.chinadaily.com.cn/a/202209/09/WS631a7c32a310fd2b29e76bfd.html.

[23] "Progress of Research and Development of e-CNY in China." (2021) People's Bank of China. (accessed 13 November 2023) http://www.pbc.gov.cn/en/3688110/3688172/4157443/4293696/2021071614584691871.pdf.