Implementing Offline Digital Currency on IPv8

Robbert Koning
dept. name of organization (of Aff.)
Delft University of Technology
Delft, The Netherlands
R.M.Koning@student.tudelft.nl

***Abstract*—This document is a model and instructions for LaTeX. This and the IEEEtran.cls file define the components of your paper [title, text, heads, etc.]. *CRITICAL: Do Not Use Symbols, Special Characters, Footnotes, or Math in Paper Title or Abstract.**

***Index Terms*—component, formatting, style, styling, insert**

I. Introduction

In recent years, the European Central Bank (ECB) has been exploring the possibility of realizing its own Central Bank Digital Currency (CDBC), the 'digital Euro'. The ECB has published various reports and resources that outline the need for such a project (i.e. [1], [2]). Calls for expression of interest are being published and the ECB aims to complete its investigation phase by October 2023 [3] [4]. The main reason for this development is the rise of digital payments and corresponding decline of cash usage. According to reports published by De Nederlandsche Bank (DNB), the national bank of the Netherlands, the share of cash payments dropped from 56% in 2010 to 21% in 2020 [5] [6]. The Swedish Riksbank mentions similar trends for Sweden [7].

Cash is the only publicly accessible form of sovereign money in the European Union (EU) [2]. Digital payments are not 'sovereign' like cash; they are made using services provided by private and/or non-European actors. The money involved in these transactions is a liability of the respective actor and not a claim on the ECB. Thus, a dependency on digital payments in their current form implies a dependency on these actors. They cannot safeguard a reliability comparable to that of ECB-backed cash. Nevertheless, there is demand for this reliability, especially in times of crisis [8].

The urgency is exacerbated by influential parties expressing increased interest in developing new payment services and CBDCs, such as the United States government and the People's Bank of China [9] [10]. A report published by the ECB discusses potential 'currency substitution'; a scenario where a new form of money that is unregulated by ECB becomes a viable medium of exchange and store of value within the EU. Currency substitution could reduce the effectiveness of the ECB's monetary policy, harm market competition, and finally even threaten the European Union's strategic independence, according to the report [1]. The private and/or non-European actors that are largely responsible for the fear of currency substitution are large corporations, big tech, and foreign central banks [2] [11].

The ECB has expressed interest for its CBDC to be usable in an offline environment. This is crucial in case of network/system failure or in areas without a reliable internet connection. A prominent example of currency that is spent offline is cash. The concept of (offline) digital money is not new; it is widely agreed upon that the idea was first proposed by Chaum in 1983 (see Section III-B) [12]. Since then, offline usable digital currencies have also been explored extensively . The literature makes a distinction between 'transferable' and 'non-transferable' money; transferable indicating that currency received offline can be re-spent offline to another party without requiring intermediate contact with a central authority [13].

This thesis concerns itself with implementing transferable digital currency on the IPv8[1] protocol stack. We limit ourselves to an implementation where currency is represented by digital units of fixed and indivisible value ('tokens'). This research contributes 1) a software-implemented simple token-based transaction system and 2) a performance analysis of various bottlenecks in this system.

II. Problem Description

Cash is passed around physically, which is crucial to its offline functioning. In a cash transaction, a sender passes their currency to a receiver, thereby ensuring that they are unable to spend it again. Trivial as it may seem, for digital systems this is difficult to realize. In the digital domain, parties can withhold information and lie. The traditional solution for this problem is to have trusted authorities verify the funds involved in all transactions; a transaction is considered complete when it is approved by an authority. This solution does not work in an offline setting because the trusted authorities are not available immediately. In offline transaction systems it is non-trivial to verify whether parties still own the funds they want to spend and have not spent them before. There are no central trusted parties that can be queried directly for this information. Comparable to cash, offline transactions involve only 2 parties: a sender and a receiver. Senders may withhold information and are therefore not trustworthy.

For an illustrative example, we envision a malicious party A who transacts their digital currency to an honest party B. After this transaction, A can 'undo' their last transaction, for instance by keeping copies of the funds they transacted to B and restoring them. Now A can transact the same funds

[1]For Kotlin-IPv8, see https://github.com/Tribler/kotlin-ipv8.

they already spent to an honest party *C*. Without an additional (in)direct connection from *B* to *C*, the honest parties cannot determine whether *A* committed fraud. This is called the 'double spending problem' [14].

There have been numerous attempts at solving or mitigating the double spending problem. Many cryptocurrencies (e.g. Bitcoin) mitigate the problem by utilizing 'global consensus', which removes the need for a central trusted authority but which does require near-immediate connectivity to parts of the network [15]. Global consensus disallows offline transfers and is therefore not a well-suited solution to make offline spending possible.

III. Related Work

A. Eurotoken

We consider the main prior work for this thesis to be Eurotoken [16], another implementation of a currency implemented on IPv8[1]. Based upon Eurotoken, and in line with many proposed digital cash schemes, we opted for a limited number of trusted authorities to verify transactions. This makes the system distributed but not decentralized. The advantage of this approach in the context of CBDCs is that it enables the respective central bank to enforce potential future policies. Moreover, it provides a non-deterministic near-immediate transaction finality.

Despite its name, Eurotoken is a balance-based system where individual units of currency are unnamed. A crucial lesson observed from this work as well as other digital cash schemes, is that balance-based systems appear to greatly complicate robustness measures [17]. Eurotoken's default mode of operation is non-transferable; cash can be spent offline once and is thereafter not accepted by recipients until validated online. We believe that a token-based architecture lends itself better for transferability. A token-based system requires the generation of tokens—analogous to minting coins—and a different transaction protocol. The token minting process and transaction protocol of our implementation are described in Section IV.

B. Blind signatures for untraceable payments

In 1983, Chaum introduced blind signatures in what is Widely accredited as the first paper to describe digital currency [12]. The paper describes a novel cryptographic primitive, the 'blind signature', which allows parties to sign messages without knowing their contents, thereby being unable to relate their signature to the original message. This primitive was used in the paper to describe the literature's first digital cash scheme.

IV. Design and Architecture

This research implements a centralized CBDC that allows offline transactions with fixed-value tokens and guarantees retroactive fraud detection.

The proposed system requires a number of trusted parties that are in charge of token exchange and transaction verification. We refer to these parties as 'authorities' and identify

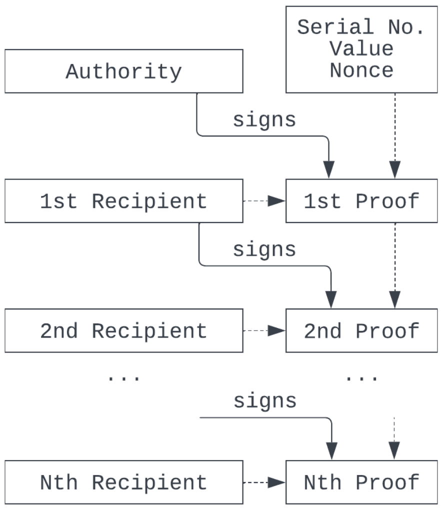

Fig. 1. Graphical representation of a token. Tokens represent monetary units of fixed value that store all their previous recipients until they are verified by an authority.

them by their public key. The limited number of trusted authorities makes verification a distributed yet centralized operation. The motivation for this design choice is elaborated upon in Section III-A. The process of exchanging currency for tokens is beyond the scope of this thesis and is briefly discussed in Section VI.

Clients are all system participants that are not authorities. They, too, are identified by their public key. It is assumed that clients know the public keys of the authorities in the network. It is also assumed that authorities know the real identities of clients. While this is not necessary for the proposed system to function, implicating a public key with fraud loses its severity if the instigator can remain anonymous. This is discussed further in Section VI-A.

Clients can transact tokens to each other and consult authorities to verify the validity of their tokens. If clients cannot connect to an authority, for instance during a power outage, they can continue transacting but defer verification until they can connect.

To realize retroactive fraud detection, the implemented system requires authorities to be able to unambiguously reconstruct the sequence of owners of a token. This is done by providing each token with a linked list of all previous owners until its last verification. Details of this procedure are explained further in this section.

A. Token Format

The token protocol is based upon transacting tokens. A diagram of a token is given in Figure 1. Each token contains[2]:

1) *Serial number*. An 8-byte unique token identifier.

[2]The bit-lengths of the signatures and public keys were adapted from those used in Kotlin-IPv8[1], upon which the implementation was built, and are not integral to the protocol's functioning.

2) *Value*. A 1-byte representation of the token's worth in Euros. Like cash, tokens have a limited number of fixed denominations and the byte values are mapped to those. For example, a token worth 1 Euro has a byte value of 7, though this mapping is arbitrary.
3) *Authority public key*. A 74-byte public key of the authority that is in charge of the token (the 'authority').
4) *Nonce*. A 64-byte pseudo-random nonce used by the authority to differentiate between differing occasions where the same token is sent to the same recipient.
5) *Recipients*. A list of recipient-proof pairs in chronological order. This list must contain at least a first pair:
    a) *First recipient public key*. A 74-byte public key of the token's first recipient after creation or validation.
    b) *First proof*. A 64-byte signature ('proof') given by the authority signing *Serial number*, *Value*, *Nonce*, and *First recipient public key*.

    All pairs in the list are of the same format and bit-length. The second pair (if present) contains *Second recipient public key* and a signature given by *First recipient public key* signing *First proof* and *Second recipient public key*. Likewise, all subsequent pairs follow the same pattern; they contain a signature by the previous public key in the list, signing the previous proof together with the next public key. This signature chain corresponds to the token changing ownership during transactions.

B. Token Minting

When a token is created, its *Serial number*, *Value*, *Nonce*, *Authority public key*, and *Recipients* list are set as specified in Section IV-A. The authority stores a copy of the entire token and sends it to the intended client.

C. Client Verification

When a client obtains a token, it verifies it in a 3-step process. First, the client verifies that the token's last recipient (that is, the last public key in the *Recipients* list) refers to them. Second, the client verifies that it knows the token's *Authority public key* and that this key created the token's *First proof*. Third, the client verifies the remaining chain of proofs in the *Recipients* list. The purpose of the client's verification process is merely to ensure that they have received an unambiguous proof of transfer from their transaction's counterparty. This proof can later be used by the relevant authority to proof potential fraud. A client deciding that a token is valid does not imply that an authority will decide the same. The client's verification does however guarantee that clients victimized by fraud can proof so eventually.

D. Client Transaction

A token's initial recipient may choose to send it to another client. If it does, it must append a new pair to the token's *Recipients* list that contains the desired recipient's public key and a signature of the token's last proof together with the desired recipient's public key. This is depicted in Figure 1.

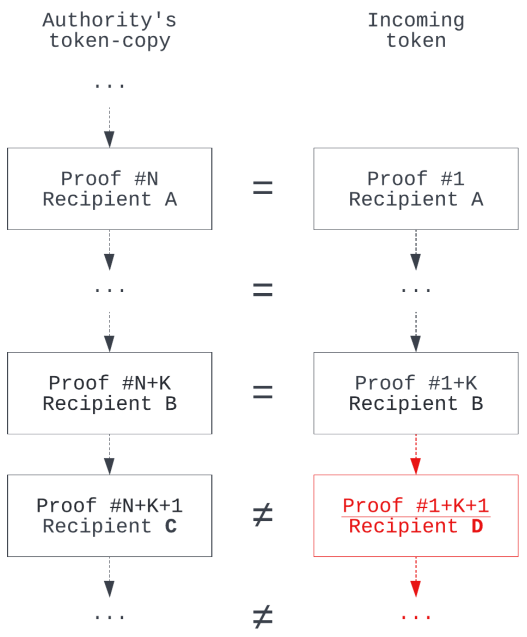

Fig. 2. The authority's double spending detection mechanism. In the figure, recipient B double spent a token, which was detected because proof $N+K+1$ of the authority was not equal to proof $1 + K + 1$ of the incoming token.

E. Authority Verification

The authority's verification process is started when a client sends them a token to verify. The verification process contains 6 steps:

1) The authority ensures that the received token has more than 1 recipient in its *Recipients* list. If not, the token is either invalid or ineligible for verification.
2) The authority ensures that the token's last recipient is the client that sent the token in for verification.
3) The authority queries if the token is still valid. The knowledge that the authority once signed the received token, which can be derived from the token's *First proof*, says little about the token's current state. The authority compares its public key against the token's *Authority public key* and queries the token's *Serial number* to ensure that itself is the authority that manages the token. Then it verifies that the token is still in circulation.
4) The authority will, like an honest client, verify the chain of proofs in the *Recipients* list.
5) The authority will attempt to detect double spending by comparing the proof of the last pair ('last proof') of its token-copy to *First proof* in the received token. If these are identical, double spending cannot be proven (see Section IV-F) and the authority will finalize verification. Finalizing verification requires the authority to update its copy of the token by appending all new recipient-proof pairs of the received token to its *Recipients* list. It will also append a new pair containing the desired recipient—the one who sent the token for verification—and a corresponding proof.

6) The authority sends the verified token to the desired recipient.

F. Double Spending Detection

In Section IV-E it is mentioned that the authority updates its token-copy's *Recipients* list upon a valid verification. This means that its last proof is updated as well. To detect double spending, an authority compares the last proof of its token-copy to *First proof* in the received token. A diagram of this scenario is depicted in Figure 2.

If a token is double spent, then multiple versions of the token will eventually reach their authority. The first time, double spending cannot be detected and the token-copy is updated. Subsequent times, the authority's token-copy already has an updated *Recipients* list and therefore its last proof does not correspond to the double spent token's *First proof* anymore. Thus, if the proofs differ double spending has occurred. If the proofs are equal, double spending might have occurred.

When double spending is detected, the authority will search for the instigator. It will find the received token's *First proof* in the *Recipients* list of its token-copy. It will then compare the recipient-proof pairs of the token-copy with those of the received token starting from the pairs that contain *First proof* in both lists, respectively. Eventually, it must find two differing pairs, after which all pairs will be different because proofs are chained to each other. The first differing pairs are the start of the token's split history and proof that double spending was performed by the client that signed them.

G. Replay Attack Prevention

The detection mechanism of Section IV-F allows for a replay attack in an offline environment. If a malicious sender A were to replay sending the same token to the same receiver B as before, said receiver would not flag this as malicious behavior. If B in turn were to spend this token, upon verification of the token, B would be flagged as a double spender. When an authority compares the transaction history of the token, it cannot distinguish A's first transaction to B from its second. Thus B spending the token is the first occurrence that differs from the authority's history. As described in Section IV-F, B is therefore marked as a fraudster.

There exist various solutions for preventing such an attack. One such solution is to initiate a transaction with the receiver sending a short handshake that includes a pseudo-random nonce. The sender must include this nonce in its transaction to proof with overwhelming probability that they did not replay the transaction. Another solution is to have receivers maintain a list of the last proofs of all tokens they have ever received.

V. Performance Analysis

We analyzed the system's performance to expose its shortcomings. For a proper frame of reference, we also performed a brief performance analysis of low-level functionality such as data transfer throughput and cryptographic operations.

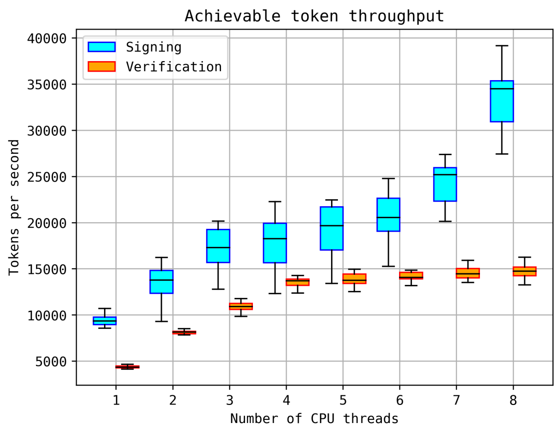

Fig. 3. Throughput of *only* cryptographic verification and signing of tokens.

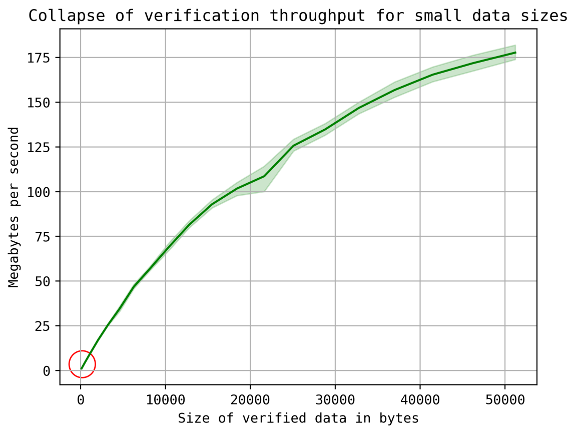

Fig. 4. The throughput of cryptographic verification collapses for small data sizes. Tokens are 201 bytes, marked by the red circle. Measurements were performed on a single CPU thread.

Experiments were performed on standard consumer electronics; a Lenovo Thinkpad L13 with an Intel i5 CPU operating at 2.11 GHz and 8 GB of DDR4 RAM. All experiments were performed 10 times.

A. Cryptographic Verification

We measured the throughput of various cryptographic operations to ascertain the upper performance bounds of the protocol and IPv8[1]. The core idea is that by stripping the implementation of all other factors, the influence of cryptographic operations on an authority's throughput can be determined. All operations were performed with Ed25519 using a Kotlin port of Libsodium[3] that is also used by IPv8[1] [18]. The chosen parameters were identical to those used in IPv8[1].

Figure 3 shows the throughput of the cryptographic operations required to verify tokens in an online scenario. As described in Section IV, the authority's signature needs to be verified as well as the first recipient's. Figure 3 shows that

[3]For Lazysodium, see https://github.com/terl/lazysodium-java.

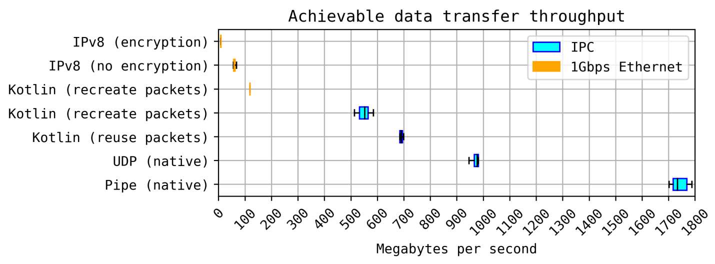

Fig. 5. Throughput of various data transfer methods.

throughput increases monotonically although not linearly with the number of CPUs, even though verification and signing processes can be executed independently from each other. We suspect the diminishing increase to be due to resource sharing, although the exact reasons are unknown. Interestingly, the highest verification measurement of 17483 tokens per second, at 201 bytes to verify per token, corresponds to a signature verification throughput of only 3.51 megabytes per second. To verify this was not an erroneous result, we measured signature verification for different data sizes.

Figure 4 shows the throughput of cryptographic verification for varying data sizes on a single thread. It is apparent that larger file sizes are tremendously faster to verify than smaller. We expect this to hold true for signing operations well.

B. *Data Transfer*

IPv8[1] uses its own acknowledgement protocol, EVA[4].

TODO: EVA STUFF

EVA's implementer suspected EVA's observed low throughput to be due to a limitation of the underlying IPv8[1] framework [19]. To verify this claim, we performed additional measurements that we show in Figure 5. Figure 5 shows that the overhead of using Kotlin as opposed to a natively compiled UDP sender is significant but not problematic. Kotlin maximally utilizes the available bandwidth when constrained by a 1Gbps connection, measuring a throughput of almost 125 megabytes per second. The overhead of IPv8[1] is however problematic, as throughput drops to an average of 60.2 megabytes per second without encryption and 8.3 with. When encryption is enabled, each individual IPv8 packet is encrypted. Based upon the results of Figure 4, we expect encryption to also be a bottleneck for packet throughput. Nevertheless, encrypted IPv8 traffic was massively faster than EVA's throughput for all measured configurations.

VI. Discussion & Future Work

Realizing offline functionality with a simple digital currency protocol came at the cost of sacrificing other desirable properties, which are described in this section.

[4]For the EVA protocol, see https://github.com/Tribler/kotlin-ipv8/tree/master/ipv8/src/main/java/nl/tudelft/ipv8/messaging/eva

A. *Anonymity*

For offline usage, the implemented system requires aggregating a linked list of previous owners of a token, up until the last verification by an authority. Unless the means of identifying past owners—currently by means of a public key—can be decoupled from the owners' real identities, privacy is harmed. Specifically, recipients of a token can see all previous recipients of that token until its last verification. There are digital cash schemes that provide stronger notions of client anonymity. Some schemes protect the identities of previous recipients and provide 'unlinkability' such that it is also impossible to relate different payments from the same client [20]. Some schemes provide an even stronger notion of anonymity where an adversary cannot recognize a token spent between other clients, even if he has already owned the token [21]. It has however been proven that an adversary can always recognize his previously-owned tokens if they are paid back to him [21].

Furthermore, it is assumed that authorities know the identities of their clients. It is expected that fraudsters cannot always be penalized within the confines of the transaction system. For example, dealing a corrective fine requires a convict to own enough tokens to pay. If a fine cannot be paid, corrective actions need to be taken in another way that does not involve tokens. Finding a fair way to correct fraud and penalize fraudsters was intentionally left out of scope.

B. *Exact Payments*

Tokens used in this thesis are indivisible in value. The disadvantage is that often multiple tokens need to be sent to make an exact payment, like with cash. Balance-based systems do not suffer from this problem but are much less suited for double spending detection in offline settings [17].

It has been shown that digital cash can be made 'divisible' such that all currency denominations up to and smaller than the value of the owned digital coin can be spent incrementally [22].

C. *Distributed Authorities*

?

D. *Decentralization*

The system depends on a number of trusted authorities to verify transactions. If the system is deployed as a true substitute for cash, then decentralizing the trusted parties is desirable. Decentralizing the system would likely have disadvantages that might be unacceptable, such as delayed or probabilistic transaction finality, limited scalability, or less effective monetary policy. In line with most of the literature on digital cash and our main prior work, we opted for a centralized approach [16].

E. *Price Stability*

It is fundamental for a European CBDC to be tethered in value to the Euro. A high price volatility like Bitcoin's is undesirable for a medium of exchange [23]. There are various ways in which the value of an asset can be kept stable and this topic has gained renewed interest with the rise of ‘stablecoins’—cryptocurrencies that aim to be non-volatile with regards to a major non-cryptocurrency or physical asset. There is an inverse relationship between the potential stability of stablecoins and how much they are decentralized [24]. The strongest stabilization mechanism is collateralization by currency or off-chain assets such as gold. By allowing free trade between a stablecoin and its collateral at a fixed price, arbitrage prevents the stablecoin’s price from fluctuating greatly. However, off-chain assets are not traded in a decentralized way and as such there is a trade-off between decentralization and stability. To the best of our knowledge, no decentralized and highly stable stablecoins exist.

References

[1] ECB, “Report on a digital euro,” ECB, Tech. Rep., 2020. [Online]. Available: https://www.ecb.europa.eu/pub/pdf/other/Report_on_a_digital_euro~4d7268b458.en.pdf

[2] F. Panetta, “Public money for the digital era: towards a digital euro,” 2022, keynote speech by Fabio Panetta, Member of the Executive Board of the ECB. [Online]. Available: https://www.ecb.europa.eu/press/key/date/2022/html/ecb.sp220516~454821f0e3.en.html

[3] ECB, “Call for expression of interest for digital euro front-end prototyping,” 2022, a call for expression of interest by ECB regarding a front-end prototype for a digital Euro. [Online]. Available: https://www.ecb.europa.eu/paym/digital_euro/investigation/profuse/shared/files/dedocs/ecb.dedocs220428.en.pdf

[4] ——, “Faqs on the digital euro,” 2022, an FAQ by ECB for the digital Euro project. [Online]. Available: https://www.ecb.europa.eu/paym/digital_euro/faqs/html/ecb.faq_digital_euro.en.html

[5] N. Jonker, L. Hernandez, R. de Vree, and P. Zwaan, “From cash to cards: how debit card payments overtook cash in the netherlands,” DNB, Tech. Rep., 2018. [Online]. Available: https://www.dnb.nl/media/kx1akmnb/201802_nr_1_-2018-_from_cash_to_cards_how_debit_card_payments_overtook_cash_in_the_netherlands.pdf

[6] N. Jonker and P. Zwaan, “Betalen aan de kassa 2020,” DNB, Tech. Rep., 2020. [Online]. Available: https://www.dnb.nl/media/e34bo5zu/betalen_kassa_2020.pdf

[7] S. Riksbank, “Pandemic hastening development towards digital payments,” 2021, an article published by Sveriges Riksbank about the declining usage of cash in Sweden and other countries. [Online]. Available: https://www.riksbank.se/en-gb/payments--cash/payments-in-sweden/payments-report-2021/1.-trends-on-the-payment-market/pandemic-hastening-development-towards-digital-payments/cash-used-less-and-less-frequently-in-sweden-and-abroad/

[8] ECB, “The paradox of banknotes: understanding the demand for cash beyond transactional use,” *ECB Economic Bulletin, Issue 2/2021*, 2021. [Online]. Available: https://www.ecb.europa.eu/pub/economic-bulletin/articles/2021/html/ecb.ebart202102_03~58cc4e1b97.en.html

[9] O. of Science and T. Policy, “Technical evaluation for a u.s. central bank digital currency system,” Office of Science and Technology Policy, Tech. Rep., 2022. [Online]. Available: https://www.whitehouse.gov/wp-content/uploads/2022/09/09-2022-Technical-Evaluation-US-CBDC-System.pdf

[10] W. G. on E-CNY Research and D. of the People’s Bank of China, “Progress of research & development of e-cny in china,” Working Group on E-CNY Research and Development of the People’s Bank of China, Tech. Rep., 2021. [Online]. Available: http://www.pbc.gov.cn/en/3688110/3688172/4157443/4293696/2021071614584691871.pdf

[11] BIS, “G7 working group on stablecoins,” BIS, Tech. Rep., 2019. [Online]. Available: https://www.bis.org/cpmi/publ/d187.pdf

[12] D. Chaum, “Blind signatures for untraceable payments,” in *Advances in cryptology*. Springer, 1983, pp. 199–203.

[13] D. Chaum and T. P. Pedersen, “Transferred cash grows in size,” in *Workshop on the Theory and Application of Cryptographic Techniques*. Springer, 1992, pp. 390–407.

[14] U. Chohan, “The double spending problem and cryptocurrencies,” *SSRN*, 2021.

[15] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” *Decentralized Business Review*, p. 21260, 2008.

[16] R. Blokzijl, “Eurotoken,” Master’s thesis, Delft University of Technology, 2021. [Online]. Available: http://resolver.tudelft.nl/uuid:132faae8-6883-454f-a8ce-94735340dce9

[17] S. Canard, D. Pointcheval, O. Sanders, and J. Traoré, “Divisible e-cash made practical,” in *IACR International Workshop on Public Key Cryptography*. Springer, 2015, pp. 77–100.

[18] D. Bernstein, N. Duif, T. Lange, P. Schwabe, and B. Yang, “High-speed high-security signatures,” *Journal of cryptographic engineering*, vol. 2, no. 2, pp. 77–89, 2012.

[19] J. Bambacht and J. Pouwelse, “Web3: A decentralized societal infrastructure for identity, trust, money, and data,” Master’s thesis, Delft University of Technology, 2022.

[20] S. Canard, A. Gouget, and J. Traoré, “Improvement of efficiency in (unconditional) anonymous transferable e-cash,” in *International Conference on Financial Cryptography and Data Security*. Springer, 2008, pp. 202–214.

[21] S. Canard and A. Gouget, “Anonymity in transferable e-cash,” in *International Conference on Applied Cryptography and Network Security*. Springer, 2008, pp. 207–223.

[22] T. Okamoto and O. Kazuo, “Universal electronic cash,” in *Annual international cryptology conference*. Springer, 1991, pp. 324–337.

[23] V. Hajric and K.Greifeld, “Bitcoin went mainstream in 2021. it’s just as volatile as ever,” Bloomberg, 2021. [Online]. Available: https://www.bloomberg.com/graphics/2021-bitcoin-volitility/

[24] D. Bullmann, J. Klemm, and A. Pinna, “In search for stability in crypto-assets: are stablecoins the solution?” ECB, Tech. Rep., 2019. [Online]. Available: https://www.ecb.europa.eu/pub/pdf/scpops/ecb.op230%7Ed57946be3b.en.pdf

Acknowledgment

The preferred spelling of the word “acknowledgment” in America is without an “e” after the “g”. Avoid the stilted expression “one of us (R. B. G.) thanks . . .”. Instead, try “R. B. G. thanks. . .”. Put sponsor acknowledgments in the unnumbered footnote on the first page.

IEEE conference templates contain guidance text for composing and formatting conference papers. Please ensure that all template text is removed from your conference paper prior to submission to the conference. Failure to remove the template text from your paper may result in your paper not being published.