Eurotoken

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 increased its efforts in exploring the possibility of realising its own Central Bank Digital Currency (CDBC), the 'digital Euro'. The ECB has published various reports and resources that outline the desirability, necessity even, of such a project (i.e. [1], [2]). Most resources mention the decline of cash usage and corresponding rise of digital payments as a prominent reason for a digital Euro. 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 [3] [4]. Cash is currently the only publicly accessible form of sovereign money [2]. Digital payments are made using services provided by private and/or foreign (non-European) actors. The money involved in these transactions is a liability of the respective actor and not a claim on a European central bank.

Perhaps more importantly, a report published by ECB discusses a potential 'currency substitution'; a scenario where a new form of money that is entirely unregulated by ECB becomes a viable medium of exchange and store of value. Currency substitution could reduce the effectiveness of ECB's monetary policy, harm market competition, and finally even threaten the European Union's strategic independence [1]. The private and/or foreign actors that are largely responsible for the fear of currency substitution are large corporations, big tech, and foreign central banks [2] [5]. In order to compete with these parties, the ECB has enumerated many requirements and wishes for its CBDC. First and foremost it is necessary for the value of digital Euros to be anchored to physical Euros. Moreover, the ECB wishes for its CBDC to enjoy beneficial cash-like features, such as being protective of citizens' privacy, being spendable in an offline setting, and being able to be remunerated at varying interest rates. For a full specification of the ECB's requirements and wishes for its CBDC, we refer the reader to the report [1].

Some of the demands and wishes mentioned by the ECB are difficult to realise individually and perhaps not even unifiable. The aforementioned report outlines multiple scenarios and analyses [1]. This research focuses on a scenario that attempts to closely resemble cash usage; physical Euros are mimicked by digital units of fixed, indivisible value ('tokens') and emphasis is placed on researching their spendability in an offline setting. Due to technical limitations however, some design choices were made that do not fall in line with the anonymity and decentralisation of cash usage. In particular the choice for a centralised validation process instead of peer-to-peer stands out. Please refer to Section ? for further elaboration. This research contributes 1) a token-based transaction system 2) a performance analysis of various bottlenecks in the system to highlight its weaknesses and empirical upper performance bounds and 3) a fully functional and software-tested reference implementation.

II. Problem Description

III. Related Work

A. Eurotoken

A related work that was important for this thesis is Eurotoken [6]. It faced similar difficulties and shares much of its design decisions with this research. An important difference is that Eurotoken is balance-based and this research is token-based. A token-based system requires generation of tokens and a different transaction protocol. The token generation process is described in Section ? and the transaction protocol in Section ?. A major implication of a token-based system is that multiple tokens need to be sent per payment, comparable to how cash payments often require multiple notes and coins. From measurements it became apparent that Eurotoken's transaction throughput was not high enough to facilitate the needs of the Eurozone. Transactions were measured to be around ?. VISA is capable of processing 24000 transactions per second (self-proclaimed, [7]) and Alipay 544000 [8]. It is worth noting that the scale of these systems is massively larger than measured in the Eurotoken paper, which results in skewed measurements [6].

IV. Design and Architecture

Initially, tokens are created and distributed in exchange for Euros. The process of exchange is beyond the scope of this paper. We refer to authorities in charge of token exchange and verification as 'verifiers' and identify them by their public key. Clients are all parties that are not verifiers. They, too, are

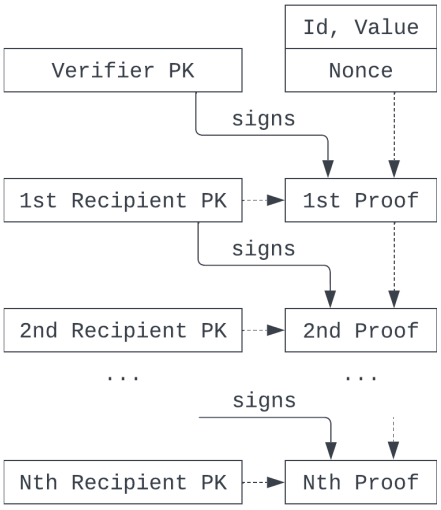

Fig. 1. Graphical representation of a token.

identified by their public key. It is assumed that clients know the public keys of the verifiers in the network.

A. Token Format

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

1) *Id*. An 8-byte unique token identifier.
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) *Verifier public key*. A 74-byte public key of the authority that is in charge of the token (the 'verifier').
4) *Nonce*. A 64-byte pseudo-random nonce used by the verifier 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 verifier signing *Id*, *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.

[1]The bit-lengths of the signatures and public keys were adapted from those used in ipv8 [BRON], upon which the implementation was built, and are not integral to the protocol's functioning.

B. Token Creation

When a token is created, its *Id*, *Value*, *Nonce*, *Verifier public key*, and *Recipients* list are set as specified in Section IV-A. The verifier 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 *Verifier 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. A client deciding that a token is valid does not imply that a verifier 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.

E. Verifier Verification

The verifier's verification process is started when a client sends them a token to verify. The verification process contains 5 steps:

1) The verifier 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 verifier ensures that the token's last recipient is the client that sent the token in for verification.
3) The verifier queries if the token is still valid. The knowledge that the verifier once signed the received token, which can be derived from the token's *First proof*, says little about the token's current state. The verifier compares its public key against the token's *Verifier public key* and queries the token's *Id* to ensure that itself is the authority that manages the token. Then it verifies that the token is still in circulation.
4) The verifier 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 verifier will finalize verification. Finalizing verification requires the verifier to update its copy of the token by appending all new recipient-proof pairs of the received token to its *Recipients* list. It will

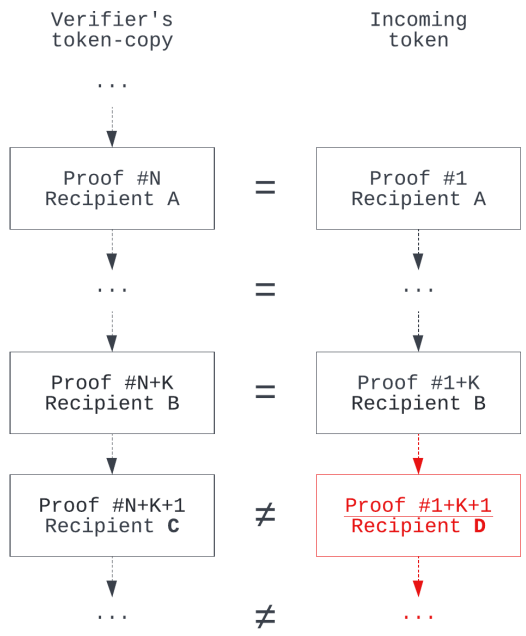

Fig. 2. The verifier's double spending detection mechanism. In the figure, recipient B doubly spent a token, which was detected because proof $N+K+1$ of the verifier was not equal to proof $1+K+1$ of the incoming token. This mechanism can only be applied if the verifier's token-copy's last proof is not equal to the incoming token's *First proof*.

also append a new pair containing the desired recipient - the one who sent the token for verification - and a corresponding proof.

5) The verifier sends the verified token to the desired recipient.

F. Double Spending Detection

In Section IV-E it is mentioned that the verifier 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, a verifier 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 doubly spent, then multiple versions of the token will eventually reach the verifier. The first time, double spending cannot be detected and the token-copy is updated. Subsequent times, the verifier's token-copy already has an updated *Recipients* list and therefore its last proof does not correspond to the doubly 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 verifier 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. Tokens vs. Balances

The choice to use tokens instead of balances was motivated by the protocol's emphasis on supporting offline transactions. Integral to the effectiveness of this protocol is another that, given a proof of fraud, provides conflict resolution and damage mitigation. It thereby deters fraud without the need to 'solve' the double spending problem in an offline setting. The protocol described in Section IV provides a way for authorities to eventually proof fraud has occurred, even in offline settings, on a per-token basis. This proof only requires knowledge that pertains to a single token. Generalizing such a proof to a balance, which is an aggregation of multiple transactions, requires a more complex proof. Although we cannot give conclusive evidence regarding the difficulty of such a proof, we consider it to be beyond the scope of this paper.

V. Performance Analysis

We did a performance analysis to identify the system's 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.

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] 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

[4] 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

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

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

[7] VISA, "Visa acceptance for retailers." [Online]. Available: https://usa.visa.com/run-your-business/small-business-tools/retail.html

[8] J. Zhang, "How alibaba powered billions of transactions on singles' day with 'zero downtime'," South China Morning Post, 2019. [Online]. Available: https://www.scmp.com/tech/e-commerce/article/3038539/how-alibaba-powered-billions-transactions-singles-day-zero-downtime?module=perpetual_scroll_0&pgtype=article&campaign=3038539

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.