Crypto-Asset Collateralised Loans in Open Finance via Robust Fully Homomorphic Encryption

Lorenzo Martinico[1][0000-0002-4968-674X],
Raffaella Calabrese[1][0000-0002-0078-3151],
Tzameret Rubin[2][0000-0001-5390-8600], and
Michele Ciampi*[1][0000-0001-5062-0388]

[1] University of Edinburgh, Old College, South Bridge, Edinburgh EH8 9YL, UK
llorenzo@martinico.me {michele.ciampi,raffaella.calabrese}@ed.ac.uk
[2] Oxford Brookes University, Headington Campus, Oxford OX3 0BP UK
t.rubin@brookes.ac.uk

Abstract. Rapid advances in Financial Technology (FinTech) have transformed the landscape of financial services, with Open Banking (OB) and its extension, Open Finance (OF), enabling safe data sharing to third parties, promoting competition and accessibility. In parallel, Decentralised Finance (DeFi) has emerged as a blockchain-based ecosystem for permissionless financial services, offering innovative services, but facing volatility and regulatory uncertainty. Despite the fundamental differences between OF and DeFi, recent developments indicate growing interest in bridging the two, as traditional institutions increasingly incorporate digital assets into their services. This paper introduces a cryptographic protocol that enables secure interaction between OF and DeFi in the setting of collateralised lending. Our protocol allows lenders to assess borrowers' risk scores by evaluating a privacy-preserving function on their net worth, and to use blockchain assets as collateral, unlockable only through a proof of repayment or default in the form of OF authenticated data. We further propose a sub-protocol that directly integrates blockchain assets into risk estimation, ensuring both trustworthiness and privacy.

Our protocol is based on a novel notion of fully homomorphic encryption (FHE) that provides stronger security guarantees against a potentially corrupted decryptor. We formalise our notion, *Robust FHE* and show how to instantiate it, starting from any standard FHE scheme. We also argue that existing notions of FHE fail to achieve our security notion.

Finally, we prove the security of our OF protocol in the Universal Composable setting.

Keywords: FHE · Fully-Homomorphic Encryption · Open-Finance · Open-Banking.

* Corresponding author.

1 Introduction

The past decade has seen rapid technological change in the financial sector, with Financial Technology (*FinTech*) reshaping how companies and individuals manage money. A central development has been the rise of *Open Banking* (OB), a policy and technological framework that enables consumers to securely share their financial data with third-party providers through standardised Application Programming Interfaces (APIs). OB was introduced to foster competition between incumbent banks and FinTech firms, reduce the risks of unsafe data-sharing practices, and give consumers greater control over their financial information [19]. The current landscape of OB implementations across the world is not uniform, with some jurisdictions mandating participation and specific technologies, and others promoting a voluntary and market-driven approach. Nonetheless, the adoption of OB has led to new or improved services to consumers compared to those offered by traditional financial institutions [52]; in particular, in relation to lending products [33].

Building on this success, the novel concept of *Open Finance* (OF) is increasing in popularity. The aim of OF is to extend the principles of OB beyond banking to a wider range of financial institutions and services [5]. OF seeks to enable new business models, promote competition, and make financial products more accessible, particularly to groups traditionally underserved by the financial system. However, OF remains in development and raises significant questions about privacy, security, and the fair use of sensitive consumer data—especially for financially vulnerable individuals [34]. As such, ensuring effective regulation and risk mitigation will be crucial to engender consumer trust, and fully realise the potential benefits of OF [3].

In the same timeframe, a parallel movement to OB and OF has emerged in the form of Decentralised Finance (DeFi), a blockchain-based ecosystem that offers permissionless and transparent financial services outwith the traditional regulatory perimeter.

While OF and DeFi share the promise of greater access and innovation, they differ in their models and face distinct challenges. OF depends on regulated, auditable entities, has not converged to a well-defined standard, and there are significant open questions about its privacy and data governance guarantees. DeFi, by contrast, operates in a pseudonymous environment and is prone to volatility, economic exploits such as maximal extractable value (MEV), and persistent regulatory uncertainty [54]. These contrasting approaches highlight the diverse paths through which technology is reshaping finance — and the importance of balancing innovation with safeguards for consumers and markets alike.

Recent developments suggest increasing interest in bridging these two worlds [43]. Blockchains have been proposed to address a number of problems in traditional financial institution, like dispute resolution schemes, frontrunning prevention [25,26,24], certifying of financial documents [51]; and with weakening regulatory scrutiny [42], traditional financial institutions have shown curiosity in making digital (cryptocurrency) assets part of their portfolio of services. Large financial institutions, including J.P. Morgan, have begun to integrate digital assets

into their services, for example, by accepting Bitcoin ETFs as collateral [13] and incorporating cryptocurrency holdings into net-worth assessments [49]. These shifts highlight the need for cryptographic protocols that can securely combine traditional financial data with blockchain-based assets.

1.1 The Open Finance Context

Contrary to Open Banking, which has been implemented in multiple markets, Open Finance does not yet have a finalised regulatory framework clarifying how different participants are required or allowed to interact with each other. With that in mind, we anticipate that the following types of Open Banking entities will broadly carry over into Open Finance:

- *User Party* (P): the end-user of a financial application. This can be a thought of as an individual or business, whose financial activity might be distributed across different services and products.
- *Data Provider* (DP): an entity that creates, receives, and maintains data through financial transactions with different users. DPs are entities that typically have established businesses providing financial services, and due to the nature of their business, have collected valuable data. These entities are heavily regulated, and their correct behaviour is monitored and enforced by law. While in the OB ecosystem Data Providers were limited to banks and credit unions, the OF *scope expansion* will broaden these to other types of financial service providers.
- *Data User* (DU): a FinTech company that wishes to access data from Data Providers through Open Finance APIs in order to provide new services. In some jurisdiction, OB and OF regulations legally mandate Data Providers to share their data with a Data User when instructed by end-users.
- *Technical Service Provider*(TSP): a party contracted by Data Providers and/or Data Users to provide technical infrastructure and services.

We expect the behaviour of the above parties in most OF regimes to remain consistent to that of OB participants, as the following example illustrates. A new FinTech seeks to introduce an innovative credit product. To optimise its lending portfolio, i.e., maximise expected loan volume subject to credit risk constraints, the firm aims to access extensive information on prospective borrowers to improve screening, pricing, and risk assessment. Let P be an end-user who wants to access the new service provided by a FinTech, DU. The first thing that P does is to inform a Data Provider party DP (or multiple) that they wish to share some financial data with DU. For example, DP can be a bank where P has an account (as in OB), or any other entity that offers financial services, such as an insurance company. Once it has received consent, DU can then directly access user data through DP's API, to personalise its offer and provide a tailored service based on P's financial activity (including non-banking information obtained from a different type of DP). A Technical Service Provider TSP (or a

number of them) might additionally be contracted as data processor(s) to facilitate this, e.g. if the service requires running heavy computational workloads or aggregating data from multiple Data Providers.

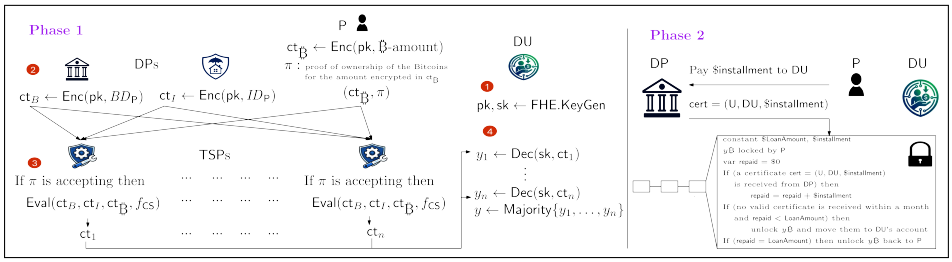

Fig. 1. In this example, the data providers (DPs) are the bank and the insurance company where P has an account. In Phase 1, these parties encrypt P's related data and send the FHE ciphertexts to the TSP*s*. The TSP*s* will perform the FHE evaluation of the lender's (DU) credit score function (denoted with f_{CS}), and return the obtained ciphertext to the lender. Assuming the lender agrees to lend fiat money to P only conditioned on P locking a collateral. This collateral is given back to P once the all amount has been repaid, or it is given to the lender if P misses a payment. The way the contract acknowledges that P has issued a payment is by receiving a signed certificate from the bank for $\mathsf{cert} = (\mathsf{U}, \mathsf{DU}, \mathsf{installment})$, where installment is the agreed amount that each month P needs to deposit (in fiat currency) into DU's bank account.

We stress that the above closely resembles the existing OB protocols, with the addition of allowing data sharing from non-bank Data Providers. While this approach can create clear advantages for emerging FinTechs and end-users (who might get a better product or service because of personalisation from the data sharing) there are serious privacy risks to the end-users.

1.2 Our Contribution

The contribution of this paper is two-fold. We provide a protocol to advance security and trust in Open Finance. To realise this, we also formulate a novel security notion for fully-homomorphic encryption, whose formalisation and construction are of independent interest.

Open Finance Contribution. We propose a *privacy-preserving* protocol that enables FinTechs to provide a personalised service to end-users, based on their financial profile in both the OF and DeFi economy. We believe this is the first work that studies how to build to connect the traditional financial system with DeFi through the OF ecosystem in a "trustless" manner.

Our protocol lets FinTechs evaluate a user P's financial risk based on P's assets held by traditional financial institutions, without directly revealing the

details of their value. The FinTech DU can then offer a loan product collateralised by cryptocurrency, without relying on a trusted intermediary.

We also propose a natural expansion of the Open Finance infrastructure by introducing a new type of Data Provider, whose honest behaviour is not enforced by law, but through cryptography. Our design allows P to establish a "self-custodial" data source to independently provide data to DUs through the blockchain. Although blockchain data is publicly accessible without authorisation, some care is needed to make sure that accounts or transactions associated with P can be identified correctly and privately. Indeed, as blockchain transactions are typically pseudonymous, it is in general not possible to associate a set of transactions with the identity of the parties involved, unless they issue a proof they control the relevant accounts.

In the next phase, our protocol provides an enhanced lending mechanism that allows end-users to borrow Fiat money using cryptocurrencies as a form of collateral. In particular, we enable the following *agreement* between parties P and DU:

- P owns an amount c of cryptocurrency (the *collateral*), and wishes to borrow an amount p of Fiat currency (the *principal*) from DU.
- Having computed a risk score for P using the privacy-preserving protocol outlined above, DU decides on the contractual terms for the loan.
- DU and P agree to the loan terms by stipulating he following smart-contract:
  - P locks a crypto-asset it controls, c, as collateral, to the smart contract
  - A partial repayment in Fiat currency needs to be issued every Δ days from P to the DU.
  - If a repayment is not issued within the agreed timeline, or if the value of the collateral drops beyond a minimum undercollateralisation threshold, then P has *defaulted*, and the crypto-assets c is transferred to DU
  - If after a sufficient amount of time P has repaid the loan and any additional interest accrued as set out by the terms of the contract, P will be able to *unlock* c and regain control of this asset.

In conjunction with the above, DU and the borrower P would also stipulate a more standard contract, that will be enforced by the standard legal mechanisms.

Clearly, one could just sell the crypto-asset and use the resulting liquidity to reduce the amount of borrowed money, but in many cases, it is preferable not to sell such assets, with the expectation that they will increase in value. Indeed, from P's perspective, there are opportunity costs, and interest rates play critical factors in the liquidity decision, but there could be some scenarios where it could be much better to maintain the crypto-assets, and use it to borrow Fiat money. In the event where P has repaid the loan, on receiving the collateral back it is likely to have further increased in value in the meantime. From the perspective of the DU, using our protocol ensures that, besides the standard legal guarantees that any lender currently receives through the loan contract, they will also receive an amount of crypto-assets in the event of P's default.

The specifics of which type and amount of crypto-assets should be considered for such a contract given a certain principal and a user's risk assessment are very

much dependent on each FinTech's business model and risk appetite. As such, we design our protocol in a modular way, so as to allow the same security guarantees for a number of lender strategies.

We show that our protocol implements an ideal collateralised lending functionality in the Universal Composability (UC) framework [15]. Since our goal is to build a protocol that is agnostic of any specific cryptocurrency or Open Finance standards, we use generic abstractions over the core technological components to avoid commitment to a specific implementation that might not be suitable for applying it to future deployments. In particular, our definitions rely on new UC formalisations for both private and public ledger functionalities. This allows us to capture the different Data Providers we consider (centralised and custodial vs decentralised and self-custodial), which we combine with a definition for secure channels with transmittable authentication as a common denominator to conduct interactions between parties and functionalities. We believe providing protocols that are inherently composable (via the UC framework) is a must for applications of this kind, making them suitable as building blocks for larger systems.

FHE Contribution. To evaluate P's financial risk (i.e., P's credit score) in a secure way, our protocol relies on fully homomorphic encryption. Our setting is quite typical: we have n parties that use the same FHE public-key to encrypt some plaintexts $(m_1, \dots m_n)$, thus obtaining the ciphertexts $(\mathsf{ct}_1, \dots, \mathsf{ct}_n)$ (these parties will be the DPs in our case). One or more parties perform the homomorphic evaluation of $(\mathsf{ct}_1, \dots, \mathsf{ct}_n)$ over a certain function f (these parties will then be the $\mathsf{TSP}s$), and the obtained final ciphertext(s) ct will be sent to the decryptor (the DU), who has the secret key and obtains the final result. This folklore approach should guarantee that, as long as there is no collusion (and the parties are semi-honest), the decryptor will only learn $f(m_1, \dots, m_n)$.

We show that this is simply not true as a malicious (even semi-honest) decryptor may be able to infer more than $f(m_1, \dots, m_n)$. We present a counterexample by constructing a pathological FHE scheme that is secure under the standard definition, but leaks information beyond $f(m_1, \dots, m_n)$ to the decryptor. Interestingly, none of the recent notions of FHE enhanced with a decryption oracle [44,10] are sufficient to prevent such leakage from a decryptor that has the secret key. Hence, we provide a novel definition that captures this type of security, *Robust FHE*, and propose a generic transform that takes as input a standard FHE protocol and turns it into a new scheme satisfying our security notion. We currently do not know whether there are FHE schemes that natively satisfy our definition, and we leave this as an interesting open question.

1.3 Related work

Lending is an important application in DeFi [38], with many peer-to-peer lending protocols providing liquidity for other applications, using algorithmic techniques based on demand and supply to set prices. These loans are generally collateralised

by another crypto-assets of greater value than the principal, stored in a smart contract [57]. *Flash loans*, a type of non-collateralised loan, are another lending product that is unique to the DeFi ecosystem [55].

Advanced cryptographic primitives for computation on confidential data have recently been proposed for a number of banking related-applications [36,37,46]. Being able to jointly compute programs with competitors could allow banks to satisfy regulations that require data sharing e.g. for anti-money laundering purposes [31]. Various techniques have been considered to compute lending risk without having access to consumer data [2,58], including Homomorphic Encryption [35]. In comparison, Open Banking and Open Finance have seen little interest from the cryptographic research community, beyond some blockchain-based protocols to improve the accountability of consent management in OB [56,50,29]. A recent exception lies in the work presented at [11], which applies advanced cryptography to OF, in the setting of the Account Aggregation ecosystem in India. An Account Aggregator in the OF ecosystem is a semi-trusted centralised entity that supports the consent flow between users, Data Providers, and Data Users, and forwards encrypted data between Data Providers and Data Users, who are able to decrypt it. The proposed protocol replaces the combination of Account Aggregator and Data User with an MPC protocol that maintains compatibility with the encryption scheme used by existing commercial providers. In a similar way to our risk estimation phase protocol, the Data User is able to compute a function over Data Providers' stored user data, without learning its inputs. While their techniques are generally applicable, the construction in the whitepaper [41] is very specific to the existing Indian commercial ecosystem.

FHE in the presence of corrupted decryptor. A number of different security properties for FHE have been proposed to capture different combinations of adversarial capabilities. It is well known [9] that homomorphic encryption is not CCA2-secure, given the malleability of ciphertexts through the evaluation procedure. Some homomorphic schemes have been proven to be CCA1-secure [4], but practical attacks to such schemes exist when the adversary has access to a ciphertext well-formedness oracle [45]. [47] proposes the notion of *verified Chosen Ciphertext Attack (vCCA)* for exact homomorphic encryption schemes, a stronger notion than CCA1 which gives the adversary a decryption oracle after the encryption of the challenge ciphertexts in the game, but forbids decryption of any ciphertext that resulted from evaluating a function over the challenge. [14] extends vCCA to approximate schemes. [44] introduces the new notion of IND-CPA$^{\mathrm{D}}$ security to capture real-world key-recovery attacks against (approximate) homomorphic encryption schemes that satisfy IND-CPA security. The IND-CPA$^{\mathrm{D}}$ adversary might now have information about the input messages and function being evaluated; therefore the game equips the adversary with encryption, evaluation and restricted decryption oracles, and permits multiple challenge queries. [10] provides a stronger notion of IND-CPA$^{\mathrm{D}}$ that allows the adversary to select the randomness used during encryption. However, none of the above provide any guarantees once the adversary knows the secret key and tries to infer what message originated the ciphertexts that decrypt to $f(\overline{m})$.

2 Technical Overview

We now give a high-level overview of our collateralised lending protocol. A *User Party* (P) wants to obtain a loan from lender DU. In order to decide whether to grant the loan to P, DU computes a risk function using P's financial data, currently held by multiple financial institutions (the data providers DPs). Without loss of generality, in this section we assume that the data collected is limited to bank transaction data and insurance data. The risk (or credit scoring) function, f_{CS}, will also consider the amount of cryptocurrency that P may have (for a pictorial representation of the protocol, we refer to Fig. 1). Because our goal is not to give away this sensitive data to the DU, we let parties engage in a simple multi-party computation protocol that works as follows. The DU generates an FHE public-secret-key pair $(\mathsf{pk}, \mathsf{sk})$ and broadcasts pk to all the parties. The two DPs (a bank and an insurance provider in our example) encrypt their data associated with P, thus obtaining ct_B and ct_I (respectively). These ciphertexts are then sent to a number of Technical Service Providers TSPs, whose role is to perform the homomorphic computation. At the same time, P encrypts the amount of cryptocurrency he owns, obtaining $\mathsf{ct}_{\text{₿}}$ and broadcasts it to the TSPs, along with a proof π, proving ownership of the amount of coins specified in $\mathsf{ct}_{\text{₿}}$.

Each TSP, upon receiving the ciphertexts, checks if the proof π verifies and if that is the case, performs a homomorphic evaluation for the credit score function f_{CS} on input $(\mathsf{ct}_B, \mathsf{ct}_I, \mathsf{ct}_{\text{₿}})$, and sends the resulting ciphertext to DU. DU uses the secret key to decrypt the ciphertexts received from the TSPs, and if the majority of the ciphertexts decrypt to the same message, takes this value to be the credit score of DU.

With this protocol, we aim to guarantee the following high-level properties:

1. As long as none of the TSP colludes with DU, the data of P are protected and DU only learns the credit score of P.
2. As long as the majority of the TSPs are honest, then DU is guaranteed to get the correct output.

Property 2 can simply be based on the correctness of FHE. Property 1 should come from the CPA security and the correctness of FHE. We note that while the correctness of FHE guarantees that the decryption would yield the correct result, the final ciphertext may still carry information about the plaintexts of $\mathsf{ct}_B, \mathsf{ct}_I$ and $\mathsf{ct}_{\text{₿}}$. Indeed, the standard definition of FHE allows a party that knows the secret key to potentially retrieve the plaintexts (or part of them) for $\mathsf{ct}_B, \mathsf{ct}_I, \mathsf{ct}_{\text{₿}}$, having only access to the result of the ciphertext evaluation by TSP (i.e., $\mathsf{ct}_i \leftarrow \mathsf{Eval}(\mathsf{ct}_B, \mathsf{ct}_I, \mathsf{ct}_{\text{₿}}, f_{\mathsf{CS}})$ in Fig. 1).

To solve this issue, we first formally define a notion of FHE that guarantees that a decryptor (who knows the secret key) can learn nothing more than the result of evaluating the function. We call this notion *Robust FHE*. This notion is formalised by requiring the existence of a simulator Sim, that on input (y, pk) (where y represents the credit score of P in our example) returns a ciphertext

indistinguishable form $\mathsf{Eval}(f_{\mathsf{CS}}, (\mathsf{ct}_B, \mathsf{ct}_I, \mathsf{ct}_{\mathrm{B}}))$ even in the eyes of a distinguisher that knows the FHE secret key. This captures the fact that once an evaluation is performed over a set of ciphertexts, a decryptor will learn nothing about the original ciphertexts. Our formal definition accounts for the stronger adversarial setting where the adversary can generate some of the ciphertexts involved in the homomorphic computation (i.e., only ct_{B} is generated by a non-corrupted party). We refer to Section 4 for more details.

Before describing how we realize such a notion, we argue that the standard FHE security notion of CPA does not trivially imply our new definition. Consider a CPA-secure encryption scheme $\mathsf{FHE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$. We now construct a new scheme $\mathsf{FHE}^\star = (\mathsf{KeyGen}^\star, \mathsf{Enc}^\star, \mathsf{Dec}^\star, \mathsf{Eval}^\star)$ that works as follows:

- $\mathsf{KeyGen}^\star$ works exactly as KeyGen;
- $\mathsf{Enc}^\star$, on input pk and a message m, returns $(\mathsf{FHE.Enc}(\mathsf{pk}, m), \mathsf{FHE.Enc}(\mathsf{pk}, 0))$;
- $\mathsf{Eval}^\star$, on input pk, ciphertext $\mathsf{ct} = (\mathsf{ct}_0, \mathsf{ct}_1)$ and function f, returns $(\mathsf{FHE.Eval}(\mathsf{pk}, f, \mathsf{ct}_0), \mathsf{ct}_0)$;
- $\mathsf{Dec}^\star$ in input $(\mathsf{sk}, (\mathsf{ct}_0, \mathsf{ct}_1))$ returns $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}_0)$.

$\mathsf{FHE}^\star$ is CPA-secure, as the security of FHE is preserved for every message encryption. But a party that knows the secret key, upon receiving the output of an evaluation phase, will be able to retrieve the original message by decrypting the second component of the ciphertexts that we denoted with ct_1. A similar counterexample can be used to argue that even the notion of IND-CPA$^{\mathrm{D}}$ [44], (or stronger variants such as [10]) which equips the adversary with a decryption oracle, does not satisfy Robust FHE. The reason is that a decryption oracle would simply ignore the second component of the ciphertext, and never give it to the adversary, whereas in our case, the adversary knows the secret key and can decrypt the ciphertext without going through the oracle.

Our FHE construction. We now show a generic construction that can turn a large class of CPA-secure FHE schemes into Robust FHE schemes. In particular, our construction subsumes the existence of a *Circuit-Private FHE scheme* (CP-FHE) scheme, which, as shown in [30,53] can be obtained from a large class of standard CPA secure FHE schemes.

A Circuit Private FHE scheme hides from any malicious party (even the party holding the secret key) the function that was used in the evaluation phase to generate a certain ciphertext[3]. Let $\mathsf{FHE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ be a CP-FHE scheme. At a high level, our scheme works as follows.

- The key-generation phase corresponds KeyGen.
- To encrypt a message m, we sample one-time pad key k and return $(\mathsf{Enc}(\mathsf{pk}, m \oplus k), k)$ as our ciphertext.

[3] This can be seen as the dual of our definition, where the function instead of the plaintext, needs to be protected

– To homomorphically evaluate a ciphertext $\mathsf{ct} = (\mathsf{ct}_0, k)$ over a function f we sample a fresh one time-pad key k' and output $(\mathsf{Eval}(R^{f,k,k'}, \mathsf{ct}_0), k')$ where the function $R^{f,k,k'}$, on input ciphertext c, returns $f(c \oplus k) \oplus k'$.

The intuitive reason why our protocol is secure is that the encrypted message is additionally protected by a one-time pad key, which is refreshed at every evaluation. The decryptor will have access only to the last one-time pad key used, and all prior keys are guaranteed to be protected due to circuit privacy. The fact that all the prior keys are protected guarantees that even if the FHE ciphertext does preserve information about the prior FHE plaintexts/ciphertexts during the evaluation (like in our counterexample), these FHE ciphertexts carry no useful information, as their content is encrypted using a one-time pad key that the adversary will never get to see.

Back to our lending protocol. Now that we have a secure protocol for computing the credit score, we are ready to show how the second phase of our protocol works (see Fig. 1, Phase 2 for a pictorial description). We want to give the option to P to use part of his crypto-assets as collateral, so that in the case P misses a payment, the crypto-asset (or part of it) will be given to the lender. To do this, we require the lender, P, and the bank where P has an account (DP), to create a smart contract where P locks collateral. Every month, the bank is supposed to issue a certificate proving that P has performed a repayment toward DU. When the loan has been fully repaid (the total amount is also a parameter of the smart-contract), then P can prove to the smart contract (through DP's certificates) that it has fulfilled its obligation, and the collateral is returned. If a payment is missing, DU can in its turn prove this through a certificate from DP, and receives part (or all) of the collateral of P from the smart contract in compensation. It is clear that a corrupted bank may receive the payment from P and not issue the certificate to the smart contract, or forge a prove of non-payment for DU's benefit. While our ideal functionality capture such cases, we do not believe these behaviours are problematic due to the pragmatic real-word assumption most of us already make. In our case, a user has to trust a bank (or other financial institution) sufficiently to register an account. There might be complex sociological reasons behind this, or a belief that as a heavily regulated industry the chances of misbehaviour are low. While this might not satisfy the most ardent cryptoanarchists, they might not be willing to use our protocol in the first place, and using banks as a source of truth has worked well enough for a majority of people over the last few centuries.

3 Notation and Standard Notation

We assume familiarity with the Universal Composable model, notions of negligible functions, computational indistinguishability, and simulation-extractable non-interactive zero-knowledge, and refer to the full version for additional definitions. if S is a set, $|S|$ represents the number of elements in S. When x is chosen randomly in S, we write $x \leftarrow S$, to assign y to x we write $x \leftarrow y$. An NP

relation R is a relation for which membership of (x, w) w.r.t. R can be decided in time polynomial in $|x|$. If $(x, w) \in \mathsf{R}$ then we say that w is a *witness* for *instance* x.

3.1 Fully Homomorphic Encryption (FHE)

Our work uses a public-key Fully Homomorphic Encryption scheme, comprising of algorithms $\mathsf{FHE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$, such that:

- $\mathsf{KeyGen}(1^\lambda)$ takes in the security parameter (a string of length λ) and generates a pair of keys $(\mathsf{pk}, \mathsf{sk})$
- $\mathsf{Enc}(\mathsf{pk}, x)$ takes a public key and a message, and generates ciphertext ct
- $\mathsf{Eval}(\mathsf{pk}, f, \overline{\mathsf{ct}})$ takes a public key, a circuit representing a function f in the set of all computable functions, and a vector of ℓ ciphertexts $\overline{\mathsf{ct}} = (\mathsf{ct}_1, \ldots, \mathsf{ct}_\ell)$, and produce a new ciphertext ct'
- $\mathsf{Dec}(\mathsf{sk}, \mathsf{ct})$ takes a secret key and a ciphertext, and produce a new plaintext value

A large number of Fully Homomorphic Encryption schemes have been proposed in the last 15 years, based on different hardness problems and providing a variety of guarantees [48]. We give short definitions for some of the properties required by a homomorphic encryption scheme as preliminaries to our construction.

Definition 1 (Correctness (homomorphic evaluation)). *A FHE scheme as defined above is correct if, for any polynomial-size circuit f, any $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{FHE.KeyGen}(1^\lambda)$, and any plaintexts $(m_1, \ldots, m_\ell)$, if we set $\mathsf{ct}_i \leftarrow \mathsf{FHE.Enc}(\mathsf{pk}, m_i)$ for all $i \in [\ell]$ and $\mathsf{ct}^\star \leftarrow \mathsf{FHE.Eval}(\mathsf{pk}, f, \overline{\mathsf{ct}})$, then $\mathsf{FHE.Dec}(\mathsf{sk}, \mathsf{ct}^\star) = f(m_1, \ldots, m_\ell)$*

Definition 2 (Compactness). *A FHE scheme is compact if the bit-length of any evaluated ciphertext output by $\mathsf{FHE.Eval}(\mathsf{pk}, f, \mathsf{ct}_1, \ldots, \mathsf{ct}_\ell)$ is upper-bounded by a polynomial in λ (and the maximum input ciphertext length), and is independent of the size of f (beyond the supported parameterised class).*

Definition 3 (Chosen-plaintext attack (CPA) security). *A fully homomorphic encryption scheme FHE is $\mathsf{IND\text{-}CPA}$-secure if it is correct (i.e., it satisfies Def. 1) and if for every probabilistic polynomial-time adversary $\mathcal{A}$ there exists a negligible function $\mathsf{negl}(\cdot)$ such that*

$$\left| \Pr \left[\begin{array}{ll} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{FHE.KeyGen}(1^\lambda) & \\ b \leftarrow \{0,1\} & \\ (m_0, m_1) \leftarrow \mathcal{A}(\mathsf{pk}) & : b' = b \\ \mathsf{ct}^\star \leftarrow \mathsf{FHE.Enc}(\mathsf{pk}, m_b) & \\ b' \leftarrow \mathcal{A}(\mathsf{ct}^\star) & \end{array} \right] - \frac{1}{2} \right| \leq \mathsf{negl}(\lambda),$$

where we require that $|m_0| = |m_1|$.

Some FHE schemes satisfy an additional property of Circuit Privacy. Informally, a scheme with this property allows a party to run the Eval procedure without learning which function f is being computed.

Definition 4 (Circuit Privacy). *We say that a Fully Homomorphic Encryption scheme $\mathcal{E}$ is (maliciously) circuit-private (CP-FHE) [53] if, for all public keys pk and vectors of ciphertexts $\overline{c} = (c_1, \ldots, c_n)$, there exists a vector of equivalent messages $\overline{x} = (x_1, \ldots, x_n)$ and algorithms* Sim *and* Ext *such that*

$$\overline{x} \leftarrow \mathsf{Ext}(1^k, pk, \overline{c}), \mathsf{Sim}(1^k, pk, \overline{c}, f(\overline{x})) \approx \mathsf{Eval}(1^k, pk, f, \overline{c})$$

4 *Robust* Fully Homomorphic Encryption

In this section, we define our new security notion of FHE and provide a construction for it. At a high level, our definition requires the existence of a simulator that, on input $f(x_1, \ldots, x_1)$, returns a ciphertext indistinguishable from what the FHE evaluation algorithm would return on input FHE ciphertexts encrypting the plaintexts $x_1, \ldots, x_n$ and the description of the function f. The existence of such a simulator captures the fact that the output of the evaluation process does not leak anything about $(x_1, \ldots, x_n)$. The definition we have just sketched, however, does not take into account the fact that some of the ciphertexts involved in the evaluation may be adversarially chosen (i.e., the adversary knows the randomness and the plaintexts of some of the ciphertexts). To capture this scenario, we allow the adversary to choose a subset of FHE ciphertexts along with input-randomness pairs that explain these ciphertexts, and require indistinguishability even when the evaluation procedure considers such adversarial ciphertexts. To help the simulator, we provide the adversarial input-output pair as an additional input. In this sense, our notion is very similar to the formalization of semi-malicious multi-party computation, where the adversary provides their input and randomness, and the simulator must simulate a transcript having the final function evaluation and these adversarial inputs. Below, we provide the formal definition.

Definition 5 (Robust FHE). *A Fully Homomorphic Encryption scheme $\mathcal{E}$ is* Robust *if it is CPA secure according to definition 3 and the following holds : For any vector of messages $\overline{x}^\star = (x_1 \ldots, x_n)$ and every partition of set $\{1, \ldots, n\}$ into the two subsets* I, H *(potentially adversarially chosen), there exists an algorithm* Sim *such that for any pk, sk in $\mathcal{E}$'s key space, and any $\{x_i, r_i\}_{i \in I}$ the following holds,*

$\mathsf{Sim}(1^k, pk, f, f(\overline{x}^\star), \{x_i, r_i\}_{i \in \mathrm{I}}) \approx \mathsf{Eval}(1^k, pk, f, \overline{c}^\star)$ *where $\overline{c}^\star = \{c_i\}_{i \in (1,\ldots,n)}$, and $\forall i \in \mathrm{H} : c_i \xleftarrow{\$} \mathcal{E}.\mathsf{Enc}(1^k, pk, x_i)$ and $\forall i \in \mathrm{I} : c_i = \mathcal{E}.\mathsf{Enc}(1^k, pk, x_i; r_i)$.*

We now show how to construct an FHE scheme that satisfies this property from a circuit-private FHE scheme.

Definition 6. *Given a circuit-private* FHE *scheme $\mathcal{E}$, and the one-time pad encryption scheme $\epsilon = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$, $\mathcal{E}'$ is the set of algorithms* $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ *defined as follows:*

- $\mathcal{E}'.\mathsf{KeyGen}(1^\lambda) = \mathcal{E}.\mathsf{KeyGen}(1^\lambda)$

- $\mathcal{E}'.\mathsf{Enc}(pk, m) = (\mathcal{E}.\mathsf{Enc}(pk, \epsilon.\mathsf{Enc}(k, m)), k)$, *where* $k \leftarrow \epsilon.\mathsf{KeyGen}(1^k)$
- $\mathcal{E}'.\mathsf{Eval}(pk, f, \overline{c})$
  - *Parse the i-th element of* $\overline{c}$ *as the pair* (c_i, k_i) *for each* $i \in [n]$.
  - *Set* $\mathsf{k} = (k_1, \ldots, k_n)$
  - $k' \leftarrow \epsilon.\mathsf{KeyGen}(1^\lambda)$
  - *Define the function* $R_{\mathsf{k}}^{f,k'}(\overline{x}) = \epsilon.\mathsf{Enc}(k', f([\epsilon.\mathsf{Dec}(k_i, x_i) | \forall x_i \in \overline{x}]))$
  - *return* $(\mathcal{E}.\mathsf{Eval}(pk, R_{\mathsf{k}}^{f,k'}, (c_1, \ldots, c_n), k')$
- $\mathcal{E}'.\mathsf{Dec}(sk, c) = \epsilon.\mathsf{Dec}(k', \mathcal{E}.\mathsf{Dec}(sk, c'))$ *where* $c = (c', k')$

Theorem 1. *The FHE scheme of Definition 6 is a Robust FHE scheme (as in Definition 5).*

Proof. The CPA security of the FHE scheme is immediately implied by the CPA security of the circuit-private FHE scheme $\mathcal{E}$ (note that CPA security does not rely at all on the security of one-time pad as the one-time pad keys are part of the ciphertexts). To complete the proof, we need to show the existence of a simulator $\mathcal{E}'.\mathsf{Sim}$, such that the following holds for any any vector of messages $\overline{x}^\star = (x_1 \ldots, x_n)$ and every partition of set $\{1, \ldots, n\}$ into the two subsets I, H, for any pk, sk in $\mathcal{E}'$'s key space, and any $\{x_i, r_i\}_{i \in I}$ the following holds,

$\mathcal{E}'.\mathsf{Sim}(1^k, pk, f, f(\overline{x}^\star), \{x_i, r_i\}_{i \in \mathrm{I}}) \approx \mathcal{E}'.\mathsf{Eval}(1^k, pk, f, \overline{c}^\star)$ where $\overline{c}^\star = \{c_i\}_{i \in (1,\ldots,n)}$, and $\forall i \in \mathrm{H} : c_i \xleftarrow{\$} \mathcal{E}'.\mathsf{Enc}(1^k, pk, x_i)$ and $\forall i \in \mathrm{I} : c_i = \mathcal{E}'.\mathsf{Enc}(1^k, pk, x_i; r_i)$.

We start by defining how $\mathcal{E}'.\mathsf{Sim}$ works. Let $\mathcal{E}.\mathsf{Sim}$ the circuit-private simulator for the scheme $\mathcal{E}$. $\mathcal{E}'.\mathsf{Sim}$ on input $(\mathsf{pk}, f, y, \{x_i, r_i\}_{i \in \{I\}})$ does the following.

- For each $i \in I$, parse r_i as $r_i = (r_i^1, r_i^2, r^3)$ and run $\mathcal{E}.\mathsf{Enc}(\mathsf{pk}, m; r_i)$ thus obtaining c_i, k_i, where $c_i \leftarrow (\mathcal{E}.\mathsf{Enc}(pk, \epsilon.\mathsf{Enc}(k_i, m; r_i^1); r_i^2))$ where $k_i \leftarrow \epsilon.\mathsf{KeyGen}(1^\lambda; r_i^3)$ and where ϵ is the one-time-pad encryption scheme[4].
- For each $i \in H$ run $\mathcal{E}.\mathsf{Enc}(\mathsf{pk}, 0)$ thus obtaining c_i, k_i, where $c_i \leftarrow (\mathcal{E}.\mathsf{Enc}(pk, \epsilon.\mathsf{Enc}(k_i, 0)))$ and $k_i \leftarrow \epsilon.\mathsf{KeyGen}(1^\lambda)$,
- Set $\overline{c} = (c_1, \ldots, c_n)$, compute $k \leftarrow \epsilon.\mathsf{KeyGen}(1^\lambda)$ and provide as the final output $(\mathcal{E}.\mathsf{Sim}(1^\lambda, pk, \overline{c}, \epsilon.\mathsf{Enc}(k, y)), k)$ (recall $\mathcal{E}.\mathsf{Sim}$ is the circuit private simulator of $\mathcal{E}$, and that $y \leftarrow f(\overline{x}^\star)$).

Note that the simulator does not use any information related to the plaintextes or the ciphertexts of the honest parties (these with indexes in H). To complete the proof, we need to prove that $\mathcal{E}'.\mathsf{Eval}(pk, f, \overline{c}) \approx (\mathcal{E}.\mathsf{Sim}(1^\lambda, pk, \overline{c}, \epsilon.\mathsf{Enc}(k, y)), k)$

Let $\overline{c}$ be as defined above by the simulator $\mathcal{E}.\mathsf{Sim}$, and let $\tilde{c} = (\tilde{c}_1, \ldots, \tilde{c}_n)$ be such that for each $i \in H$ $\tilde{c}_i \leftarrow \mathcal{E}.\mathsf{Enc}(\epsilon.\mathsf{Enc}(k_i, x_i))$, and $\tilde{c}_i = c_i$ for all $i \in I$ (note that $\{x_i\}_{i \in H}$ correspond to the honest parties' inputs). From the semantic security of the one-time-pad scheme ϵ we have that the following holds

$$(\mathcal{E}.\mathsf{Sim}(1^\lambda, pk, \overline{c}, \epsilon.\mathsf{Enc}(k, y)), k) \approx (\mathcal{E}.\mathsf{Sim}(1^\lambda, pk, \tilde{c}, \epsilon.\mathsf{Enc}(k, y)), k)$$

We recall that in the evaluation phase of our scheme $\mathcal{E}$, we run the evaluation procedure of the circuit private FHE scheme $\mathcal{E}$ for the function $R_{\mathsf{k}}^{f,k}(\overline{x}) =$

[4] Without loss of generality, we assume that the randomness input of $\mathcal{E}.\mathsf{Enc}$ is split in three parts and used as we just described.

$\epsilon.\mathsf{Enc}(k, f([\epsilon.\mathsf{Dec}(k_i, x_i)|\forall x_i \in \overline{x}]))$, where k contains the one-time-pad keys which the honest and corrupted parties used to generate the ciphertexts $\tilde{c}$ defined above. From the circuit privacy of $\mathcal{E}$, we can then claim the following. $(\mathcal{E}.\mathsf{Sim}(1^\lambda, pk, \tilde{c}, \epsilon.\mathsf{Enc}(k, y)), k) \approx (\mathcal{E}.\mathsf{Eval}(1^\lambda, pk, R_{\mathsf{k}}^{f,k}, \tilde{c}), k)$

From how we have defined the evaluation algorithm of $\mathcal{E}$ to work (definition 6) we can claim the following

$(\mathcal{E}.\mathsf{Eval}(1^\lambda, pk, R_{\mathsf{k}}^{f,k}, \tilde{c}), k) = \mathcal{E}'.\mathsf{Eval}(1^k, pk, f, \tilde{c}^\star)$

where the i-th element of $\tilde{c}^\star$ corresponds to $(\tilde{c}_i, k_i)$, where $\tilde{c}_i$ is the i-th component of $\tilde{c}$, and k_i is the one-time-pad key used to compute the inner-ciphertext of $\tilde{c}_i$ for each $i \in [n]$. This concludes our proof.

5 Multiparty Computation in the *single* TSP Setting

Before diving into the details of our lending protocol, we show how it is possible to design a secure multi-party computation protocol using our notion of robust FHE in combination with NIZK. In particular, **in this section, we consider the simplified setting where there is only one, honest TSP. The goal is to describe feasibility of generic MPC from robust FHE, i.e., in the ideal world, let** DU **learning the evaluation of a function** f **over inputs** $(x_1, \ldots, x_n)$**, where** x_i **is the input provided by the** i**-th data provider** DP_i. We stress that this section is unrelated to the lending protocol, but it should help to give an ideal/real-world meaning of robust FHE notion.

To construct our protocol, we rely on the following tools:

- A Robust FHE scheme $\mathcal{E}$ (satisfing definition 5).
- A NIZK scheme for the NP-Relation $\mathsf{R}_{\mathsf{kg}} := \{(\mathsf{pk}, r) : (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathcal{E}.\mathsf{KeyGen}(1^\lambda)\}$, and another NIZK scheme for the NP-relation $\mathsf{R}_{\mathsf{Enc}} := \{((\mathsf{ct}, \mathsf{pk}), (x, r)) : \mathsf{ct} \leftarrow \mathcal{E}.\mathsf{Enc}(\mathsf{pk}, x; r)\}$.[5]
- Secure channels: parties can communicate with each other via a secure, authenticated channel. The adversary does not see nor can tamper with messages exchanged between honest parties, but he is notified about the fact that a message has been exchanged.

We call our protocol Π_{f}, and we formally describe it below.

Setup phase. The protocol has a setup phase where the CRSs of the NIZK schemes are generated: $\mathsf{crs}_{\mathsf{kg}} \leftarrow \text{NIZK}.\mathsf{Setup}(1^\lambda, \mathsf{R}_{\mathsf{kg}})$, $\mathsf{crs}_{\mathsf{enc}} \leftarrow \text{NIZK}.\mathsf{Setup}(1^\lambda, \mathsf{R}_{\mathsf{Enc}})$. Then $\mathsf{crs}_{\mathsf{kg}}, \mathsf{crs}_{\mathsf{enc}}$ are distributed among the DU and all the DPs and the TSP. This terminates the setup phase.

Phase 1. DU upon receiving the command INIT from the environment, it samples $r \leftarrow \{0,1\}^\lambda$ and run $\mathcal{E}.\mathsf{KeyGen}(1^\lambda; r)$ thus obtaining $(\mathsf{pk}, \mathsf{sk})$. Then DU computes $\pi \leftarrow \text{NIZK}[\mathsf{R}_{\mathsf{kg}}].\mathsf{Prove}_{\mathsf{crs}_{\mathsf{kg}}}(\mathsf{pk}, r)$ and broadcast (pk, π) to the $\mathsf{DP}s$.

[5] We use a notion of NIZK defined in the CRS model, but we could rely on any other form of setup, as long as the NIZK is simulation extractable. In particular, we could rely on the scheme proposed in [6,27] if one wants to rely only on the existence of a global random oracle.

Phase 2. DP_i (with $i \in [n]$), upon receiving the command (INPUT, x) from the environment does the following.

1. If no message (pk, π) has been received from DU or $0 = \text{NIZK}[\mathsf{R}_{\mathsf{kg}}].\mathsf{Verify}_{\mathsf{crs}}(\mathsf{pk}, \pi)$ then return $\perp$, else continue as follows.
2. Sample $r \leftarrow \{0,1\}^\lambda$, compute $\mathsf{ct}_i \leftarrow \mathcal{E}.\mathsf{Enc}(\mathsf{pk}, x; r)$ and $\pi_i \leftarrow \text{NIZK}[\mathsf{R}_{\mathsf{Enc}}].\mathsf{Prove}_{\mathsf{crs}_{\mathsf{enc}}}((\mathsf{ct}_i, \mathsf{pk}), (x, r_i))$.
3. Send (ct_i, π_i) to TSP.

Phase 3 TSP upon receiving the command COMPUTE from the environment, collects all the ciphertext-proof pairs received from the DPs $((\mathsf{ct}_i, \pi_1), \ldots, (\pi_n, \mathsf{ct}_n))$, and does the following

1. Initialize an empty list V. For each $i \in [n]$ if $1 = \text{NIZK}[\mathsf{R}_{\mathsf{Enc}}].\mathsf{Verify}_{\mathsf{crs}_{\mathsf{enc}}}((\mathsf{ct}_i, \mathsf{pk}), \pi)$ then add ct_i to V.
2. Compute $\mathsf{ct}^\star \leftarrow \mathcal{E}.\mathsf{Eval}(\mathsf{pk}, f, V)$ and send $\mathsf{ct}^\star$ to DU.

Phase 4 The DU upon receiving the command (OUTPUT) from the environment, if $\mathsf{ct}^\star$ is received from TSP then return $\mathcal{E}.(Dec)(\mathsf{sk}, \mathsf{ct}^\star)$, else return $\perp$.

In Fig. 2 we propose a formalization of the UC-functionality we want to realize and below state our formal theorem.

Fig. 2. Functionality $\mathcal{F}_{\mathsf{f}}$

Upon receiving Init from $(\mathsf{pid}, \mathsf{sid})$:

Initialise list $V \leftarrow [\,]$
Record registered parties DU and $(\mathsf{DP}_1, \ldots, \mathsf{DP}_n)$

Upon receiving (INPUT, x) from party DP_i:

if DP_i has not sent input before **then**
  if DU and TSP are both corrupted **then**
    Send (INP-REC, DP_i, x) to $\mathcal{A}$
  Append x to V
  Send (INP-REC, DP_i) to $\mathcal{A}$

Upon receiving COMPUTE from DU:

$y \leftarrow f(V)$
if TSP is corrupted **then**
  Send (ALLOW-COMPUTATION?) to $\mathcal{A}$
  Upon receiving the answer (b, y') do the following
  if b=0 and DU is honest **then**
    Return $\perp$ and stop accepting further commands
  if b=1 and DU is honest **then**
    Return y' to DU and stop accepting further commands
if DU is corrupted **then**
  Send (SET-OUT, y) to $\mathcal{A}$ and stop accepting further commands
else
  Return y and stop accepting further commands

Theorem 2. *Assuming that $\mathcal{E}$ is a Robust FHE scheme (as in Definition 5) and that* NIZK*is a secure simulation-extractable NIZK), then the protocol Π_{f} UC-realize $\mathcal{F}_{\mathsf{f}}$ (Fig. 2) (under static corruption).*

Before proving our theorem (we refer to section 5.2 for the reader interested in it), below we provide a high-level discussion on the type of security our protocol offers and on nuances about the needs for Robust FHE.

5.1 Discussion on Different Corruption Patterns and Security

The ideal functionality and the protocol proposed above capture the setting where there is a single honest TSP, but DU can collude with any of the DPs. In the next section, we will propose a functionality and a protocol that captures all the possible corruption scenarios, but in this section, we show at a high level how the protocol Π_{f} handles alternative corruption settings, starting from the one captured by our ideal functionaltiy (Fig. 2) in which the TSP is honest, but any combination of the remaining parties may arbitrarily collude.

In addition, in Tab. 1 we summarize the security that Π_f offers, and compare it with the setting where Π_f would instead be constructed using a generic FHE scheme instead of a Robust one. In a nutshell, the table shows that in all other corruption settings, standard notions of FHE provide the same level of security as Robust FHE. We will use this section to also discuss in detail how the table's entries are populated.

Table 1. Whether Π_f realizes $\mathcal{F}_{\mathsf{f}}$ under different corruption settings when instantiated using standard FHE versus Robust FHE.

Corrupted DU	**Corrupted TSP**	**Standard FHE**	**Robust FHE**
No	No	✓	✓
Yes	No	×	✓
No	Yes	✓	✓
Yes	Yes	✓	✓

1) Honest TSP, honest DU and corrupted DP_i with $i \in I$ and $I \subseteq [n]$. This corresponds to the first row in Tab. 1 .If DU is honest, then the adversary may corrupt any subset of data providers DP_i. In this case:

- The adversary learns the inputs of corrupted data providers.
- The adversary learns the output $y = f(x_1, \ldots, x_n)$ (formally, the output y is given to the environment, hence it is part of the input of the distinguisher).
- The adversary learns the identities of the honest parties who provided inputs (through leakage explicitly captured in the functionality).

However, the adversary learns nothing else about the inputs of honest data providers beyond what can be inferred from $f(V)$ and its own inputs.

2) Honest TSP, corrupted DU and corrupted DP_i with $i \in I$ and $I \subseteq [n]$. This corresponds to the first row in Tab. 1 If DU is corrupted, then the adversary receives $y = f(x_1, \ldots, x_n)$ from the functionality. In this case:

- The adversary learns the output $f(x_1, \ldots, x_n)$.
- The adversary learns the inputs of corrupted data providers.
- The adversary may arbitrarily choose the output delivered to the environment.

Importantly, also in this case, the adversary does not learn anything beyond what is implied by $f(x_1, \ldots, x_n)$ and the inputs it already controls, but inference attacks are not prevented. The functionality $\mathcal{F}_{\mathsf{f}}$ does not prevent inference attacks that arise from the value of $f(x_1, \ldots, x_n)$ itself. Below, we provide a few simple examples.

Simple Leakage Example. Consider two data providers DP_1 and DP_2, where DP_1 is honest and DP_2 is colluding with DU. Let the function be $f(x_1, x_2) = \frac{x_1+x_2}{2}$. If DU is corrupted, then the adversary learns $y = \frac{x_1+x_2}{2}$. Since it already knows x_2, it can compute $x_1 = 2y - x_2$. Thus, the adversary fully recovers the honest input x_1. This is *not* a violation of the theorem, as this leakage is inherent in the function f itself.

Example (Weighted credit scoring). Each DP_i holds a private financial feature x_i of a client (e.g., income, debt level, repayment history), and a credit score is computed as a weighted sum: $f(x_1, \ldots, x_n) = \sum_{i=1}^{n} w_i x_i$, where the weights w_i are public and reflect the importance of each feature. Suppose that DP_j is honest, while all other data providers are corrupted. The adversary learns $y = \sum_{i=1}^{n} w_i x_i$, and since it knows all inputs except x_j, it can recover the honest feature as $x_j = \frac{1}{w_j}\left(y - \sum_{i \neq j} w_i x_i\right)$. More generally, even if multiple parties are honest, the adversary may still reduce the uncertainty on their inputs. For example, if only two honest features remain, the adversary learns a linear relation between them, significantly narrowing the set of possible values. We recall, however, that the leakage arises from the linearity of the function and is inherent to f, hence it is allowed by the security guarantees of $\mathcal{F}_{\mathsf{f}}$.

Where would our proof fail with alternative FHE notions? In Tab. 1 we claim that our protocol realizes $\mathcal{F}_{\mathsf{f}}$ only when instantiated with a Robust FHE scheme. Indeed, a key step in proving theorem 2 is showing that the protocol securely realizes $\mathcal{F}_{\mathsf{f}}$, i.e., that the DU learns nothing beyond the output $y = f(x_1, \ldots, x_n)$. While this might appear to follow from the CPA security and correctness of FHE, this is in fact not the case. Standard security notions for FHE (e.g., CPA security or its variants) guarantee that ciphertexts do not reveal information about the underlying plaintexts to an adversary that *does not* possess the secret key. However, in our protocol, the DU is precisely the party that holds the secret key and performs decryption.
Prior notions provide no guarantee about what a decryptor may learn from the output of a homomorphic evaluation. In particular, they do not rule out that the evaluated ciphertext $\mathsf{ct}^\star$ may encode additional information about the input ciphertexts beyond the value $f(x_1, \ldots, x_n)$. Recall that $\mathcal{F}_{\mathsf{f}}$ guarantees that the DU learns only $f(x_1, \ldots, x_n)$ and nothing else about the inputs of honest parties. Indeed, as shown in section 2, it is possible to construct such an

FHE CPA secure scheme that always leaks information about the inputs used during the evaluation phase. Stronger notions such as IND-CPA$^{\mathrm{D}}$ [44] or its variants consider adversaries with access to decryption oracles. However, these models still assume that the adversary *does not have direct access to the secret key* and that decryption is mediated through an oracle with restrictions. In our setting, the DU is the legitimate holder of the secret key and can freely decrypt any part of the evaluation output. This is the main reason why our notion is very suitable for designing multi-party protocols that follow a blueprint similar to the one considered in this paper.

3) Corrupted TSP and corrupted DP$_i$ with $i \in I$ and $I \subseteq [n]$. This corresponds to the third row in Tab. 1 . In this setting the DU is honest, hence, because the FHE secret key is in the hands of an honest party, the CPA security of the FHE scheme guarantees that the corrupted TSP, even if he is colluding some corrupted DPs, will learn nothing form the FHE ciphertexts generated from the the honest DPs. It should be noted that in this case, while there are no privacy issues, there are correctness issues as the TSP may perform an FHE evaluation for a function different from f. In the final protocol, we solve this problem by either requiring the TSP to attach a zero-knowledge proof, proving that the evaluation has been performed correctly, or relying on the existence of multiple TSPs as highlighted in section 2.

4)Corrupted TSP and corrupted DU. This corresponds to the fourth row in Tab. 1 . In such a scenario, the inputs of the honest parties are fully leaked to the adversary as the corrupted TSP can simply use the FHE secret key of the DU to decrypt everything. We note that this is captuerd in the description of our ideal functionality.

5.2 Proof of Theorem 2

Proof. We start our proof by showing how the simulator works in Fig. 3.

Fig. 3. Simulator $\mathcal{S}$

Upon receiving Init from (pid, sid):

Initialize and empty set V.
Record session identifier sid as $\mathsf{sid}_{\mathrm{main}}$
Initialise set $\mathcal{I}$ with the indices of the corrupted DPs.
Run $\mathcal{A}$ (the real-world adversary) internally and do the following.
$\mathsf{crs}_{\mathsf{enc}} \leftarrow \mathsf{Sim.SSetup}(1^\lambda, \mathsf{R}_{\mathsf{enc}})$
$\mathsf{crs}_{\mathsf{kg}} \leftarrow \mathsf{Sim.SSetup}(1^\lambda, \mathsf{R}_{\mathsf{kg}})$
Give $(\mathsf{crs}_{\mathsf{enc}}, \mathsf{crs}_{\mathsf{kg}})$ to $\mathcal{A}$
/* **Key generation phase** */
if DU is honest **then**
    $(pk, sk) \leftarrow \mathcal{E}.\mathsf{KeyGen}(1^\lambda)$
    $\pi_{\mathsf{kg}} \leftarrow \mathsf{SProve}(\mathsf{crs}_{\mathsf{kg}}, pk)$
    Send (pk, π_{kg}) to all the parties controlled by $\mathcal{A}$
else
    Upon receiving (pk, π_{kg}) from $\mathcal{A}$:
    if $\mathrm{NIZK}[\mathsf{R}_{\mathsf{kg}}].\mathsf{Verify}(pk, \pi_{\mathsf{kg}}) = 0$ **then**
        Stop the simulation and return whatever $\mathcal{A}$ returns

17: **else**
18: Record pk

19: **Upon receiving** (c_i, π_i) on the behalf of a corrupted DP_i do the following:

20: **if** NIZK[$\mathsf{R}_{\mathsf{enc}}$].$\mathsf{Verify}((\mathsf{pk}, x), \pi_i) = 1$ **then**
21: $(x_i, r_i) \leftarrow \mathsf{Ext}^{\mathcal{A}^{\mathsf{SProve}}}(\mathsf{crs}_{\mathsf{enc}})$
22: Add c_i to V and send (INPUT, x_i) to $\mathcal{F}_{\mathsf{f}}$ on the behalf of DP_i

23: **Upon receiving** (INP-REC, DP_i) from $\mathcal{F}_{\mathsf{f}}$:

24: **if** TSP is honest **then**
25: Send an emtpy message using the private channel between DP_i and TSP
26: **/*Note that in this case the TSP is honest, hence, fully controlled by the simulator who, in this case, emulates an honest party in the real world protocol without knowing their inputs. As such, the interaction between the TSP and the honest DP_i is fully simulated. The only thing the real adversary would be notified that a message has been sent over the private channel that connects DP_i) and TSP) */**
27: **if** TSP is corrupted and DU is honest **then**
28: $\mathsf{ct}_i \leftarrow \mathcal{E}.\mathsf{Enc}(\mathsf{pk}, 0)$
29: $\pi_{\mathsf{enc}} \leftarrow \mathsf{SProve}(\mathsf{crs}_{\mathsf{enc}}, \mathsf{ct}_i, \mathsf{pk})$
30: Add ct_i to V and Send (π, ct_i) to the corrupted TSP.
31: **if** TSP and DU are both corrupted **then**
32: Receive (INP-REC, DP_i, x_i) from $\mathcal{F}_{\mathsf{f}}$.
33: $\mathsf{ct}_i \leftarrow \mathcal{E}.\mathsf{Enc}(\mathsf{pk}, x_i)$
34: $\pi_{\mathsf{enc}} \leftarrow \mathsf{SProve}(\mathsf{crs}_{\mathsf{enc}}, \mathsf{ct}_i, \mathsf{pk})$
35: Send (π, ct_i) to the corrupted TSP.

36: **Upon receiving** COMPUTE from $\mathcal{F}_{\mathsf{f}}$:

37: **if** DU is corrupted and TSP is honest **then**
38: Receive (SET-OUT, y) from $\mathcal{F}_{\mathsf{f}}$
39: $c^\star \leftarrow \mathsf{Sim}(1^\lambda, pk, f, y, \{x_i, r_i\}_{i \in \mathcal{I}})$
40: Send $c^\star$ to $\mathcal{A}$
41: **if** DU is honest and TSP is corrupted **then**
42: **if** TSP has not send a ciphertext $\mathsf{ct}^\star$ to the real-world DU **then**
43: upon receiving (ALLOW-COMPUTATION?) from $\mathcal{F}_{\mathsf{f}}$ return 0 to $\mathcal{F}_{\mathsf{f}}$ and stop
44: **if** TSP has not send a ciphertext $\mathsf{ct}^\star$ to the real-world DU, with $\mathsf{ct}^\star \neq \mathcal{E}.\mathsf{Eval}(\mathsf{pk}, f, V)$ **then**
45: $y' \leftarrow \mathcal{E}.\mathsf{Dec}(\mathsf{sk}, \mathsf{ct}^\star)$
46: Send (SET-OUT, y') to $\mathcal{F}_{\mathsf{f}}$ and stop
47: Whenever $\mathcal{A}$ stops, returns whatever $\mathcal{A}$ returns.

We argue that the output of the simulator is indistinguishable from the output of the real-world adversary $\mathcal{A}$. We proceeded via standard hybrid arguments.

We prove that for every PPT environment $\mathcal{Z}$, the real execution of Π_{f} with adversary $\mathcal{A}$ is computationally indistinguishable from the ideal execution with $\mathcal{F}_{\mathsf{f}}$ and simulator $\mathcal{S}$ in Fig. 3. The proof proceeds via a sequence of hybrids, all defined in the real world.

Hybrid H_0 (Real execution). This is the real-world execution of Π_{f} with adversary $\mathcal{A}$ and environment $\mathcal{Z}$.

Hybrid H_1 (Simulated CRS and proofs). We modify the setup phase and proof generation as follows:

- The CRS are generated independently as $\mathsf{crs}_{\mathsf{kg}} \leftarrow \mathsf{Sim.SSetup}(1^\lambda, \mathsf{R}_{\mathsf{kg}})$ and. $\mathsf{crs}_{\mathsf{enc}} \leftarrow \mathsf{Sim.SSetup}(1^\lambda, \mathsf{R}_{\mathsf{Enc}})$
- All proofs generated by honest parties are replaced with simulated proofs using SProve.

We claim that $H_0 \approx H_1$. This follows from the simulation-extractability of the NIZK, which guarantees that the joint distribution of simulated CRS and simulated proofs is computationally indistinguishable from the real one, even in the presence of an adversary interacting with the prover.

Hybrid H_2 (Extraction of corrupted inputs). We modify our handling of corrupted data providers. Upon receiving (ct_i, π_i) from a corrupted DP_i, if $\text{NIZK}[\mathsf{R}_{\mathsf{Enc}}].\mathsf{Verify}((\mathsf{ct}_i, \mathsf{pk}), \pi_i) = 1$ then we extract a witness (x_i, r_i) using the extractor guaranteed by simulation-extractability, and internally record x_i as the input of DP_i. If the extraction fail (despite $\text{NIZK}[\mathsf{R}_{\mathsf{Enc}}].\mathsf{Verify}((\mathsf{ct}_i, \mathsf{pk}), \pi_i) = 1$) the hybrid stops and returns 0.

We argue that $H_1 \approx H_2$. Indeed, by simulation-extractability, for every accepting fresh proof, the extractor outputs a valid witness except with negligible probability.

Hybrid H_3. In the event that TSP is honest we modify the computation performed on behalf of TSP. Instead of computing $\mathsf{ct}^\star = \mathcal{E}.\mathsf{Eval}(\mathsf{pk}, f, V)$, (where V represnets all the ciphertexts generated from honest and corrupted DPs), we compute $\mathsf{ct}^\star \leftarrow \mathsf{Sim}(1^\lambda, \mathsf{pk}, f, y, \{x_i, r_i\}_{i \in \mathcal{I}})$, where $y = f(V)$ is computed using the honest inputs (which the hybrid knows, as we are not yet considering the case where we have the ideal functionality available) and the inputs extracted from the corrupted parties, denoted by $\{x_i\}_{i \in I}$.

We claim that $H_2 \approx H_3$. This follows from the robustness property of $\mathcal{E}$ (Definition 5), which guarantees that the output of Sim is computationally indistinguishable from a real homomorphic evaluation on honestly and adversarially generated ciphertexts, and from the assumption that the honest DPs are connected to the TSP via secure channels (hence, the potentially corrupted DU does not have access to the FHE ciphertexts of the honest DPs). In more detail, consider the case where the DU is corrupted (the case where DU is honest follows trivially).

In this case, the adversary provides (pk, π_{kg}). The execution proceeds only if the proof verifies. By soundness of the NIZK, any accepting proof implies that pk is well-formed (i.e., it belongs to the domain of valid keys). Moreover, in this hybrid, we are extracting the inputs and the randomness $(\{x_i, r_i\}_{i \in \mathcal{I}})$ that the corrupted DUs have used to compute their FHE ciphertexts, which they have sent to the TSP (controlled by the simulator). Hence, to run the simulator for the Robust FHE scheme, we are only missing y, which represents the final output of the computation. We can compute y by evaluating f using the input of the honest parties and the extracted inputs $\{x_i, r_i\}_{i \in \mathcal{I}}$, and then we can finally compute $\mathsf{ct}^\star \leftarrow \mathsf{Sim}(1^\lambda, \mathsf{pk}, f, y, \{x_i, r_i\}_{i \in \mathcal{I}})$ and send $\mathsf{ct}^\star$ to the adversary. The indistinguishability between the previous and this hybrid, in this case, comes from the Robustness of the FHE scheme.

In the event where instead the TSP is corrupted but the DU is honest, in this hybrid we replace the FHE ciphertexts generated by the honest $\mathsf{DP}s$ with FHE ciphertext of 0 (i.e., instead of encrypting the real input, the $\mathsf{DP}s$ encrypt 0).

The indistinguishability between the previous and this hybrid, in this case, comes from the CPA security of the FHE scheme.

From Hybrid H_3 to the ideal execution. The only difference between H_3 and the ideal execution is that in the ideal execution, every time a pair (x_i, r_i) is extracted from a corrupted DP_i, a message (INPUT, x_i) is sent on the behalf of the corrupted party DP_i to the ideal functionality $\mathcal{F}_\mathsf{f}$. Moreover, the final output y (which we need to run the Robust FHE simulator) will be received from $\mathcal{F}_\mathsf{f}$. Hence, no knowledge of the honest party's inputs is used. In summary:

- The inputs of corrupted DP_i are extracted exactly as in $\mathcal{S}$ and provided to $\mathcal{F}_\mathsf{f}$.
- Honest inputs are never revealed and are not needed for generating $\mathsf{ct}^\star$.
- The value $\mathsf{ct}^\star$ is generated using Sim based only on $y = f(V)$ and the extracted inputs, exactly as in $\mathcal{S}$.
- The interaction between honest DP_i and TSP is fully simulated via dummy communication, which leaks only the occurrence of a message, as in the real protocol.

Therefore, the joint distribution of the environment's view in Hybrid H_3 is identical to that in the ideal execution with $\mathcal{S}$. Then, by transitivity of indistinguishability, $H_0 \approx H_1 \approx H_2 \approx H_3$ and Hybrid H_3 is identical to the ideal execution. This concludes our proof.

6 Our Loan Functionality

We now design an ideal functionality that enables a lender to provide borrowers with loans collateralised by digital assets.

The process of loan application proceeds through three main phases. The first phase is risk estimation, where the lender obtains the result of a risk function over the assets owned by the borrower. This determines the terms under which the loan will be offered. Next, the lender and borrower agree to lock a collateral asset for the duration of the loan agreement, after which the funds are transferred between the lender and the borrower. In the repayment phase, the borrower will conduct partial repayments to the lender. During this phase, the loan might end either when the debt has been repaid, or if the lender can provide evidence of a default, i.e. the borrower having broken the contractual terms.

The first phase of the protocol (starting with message COMPUTERISK from a borrower party P) is used to compute, based on DU's risk estimation algorithms and on the total assets controlled by P across different financial accounts, the probability of repayment of a loan for amount p (the "principal" of the loan). DU's risk estimation function, ρ, may contain sensitive proprietary information. Therefore, it only reveals to P some partial information $\mathcal{P}(\rho)$, based on which P (as instructed by the environment with the response to RISKPRED) decides whether to go ahead with sharing its data. More concretely, this decision will be based on what information ρ reveals about the user data, with a natural choice for the partial information revealed by $\mathcal{P}()$ being the domain and range of ρ.

In the next phase (message COLLATERALISEDLOAN from the borrower), assuming lender DU is happy with risk factors from the previous phase, a loan for principal p is agreed, along with the collateral (stored in an account with address

BC), and contractual terms for the loan, encoded as functions $\mathsf{f}_{\mathsf{default}}$,$\mathsf{f}_{\mathsf{unlock}}$. The functions take the current clock time and the state of the ledger, and specify when the collateral can be released: $\mathsf{f}_{\mathsf{unlock}}$ specifies the conditions under which the loan can be considered to be fully repaid by the borrower; $\mathsf{f}_{\mathsf{default}}$ determines the conditions under which DU can claim the collateral, e.g. P fails to provide one of the expected instalments after some predetermined amount of time Δ, or a drop of the crypto-asset's value makes the loan under-collateralised. These terms are agreed between P and DU, out of bound from the ideal protocol, and the functionality confirms with DU they still hold (ConfirmTerms). If so, the collateral c is locked in a special new account, tied with the defaulting and unlocking functions, and p is transferred from DU's account to P's (held by financial institution $\mathsf{DP}^\star$). After that, when P proceeds with partial loan repayments, it adds them to a special repayment account. If DU is not satisfied with the repayment rate, it can attempt to default the account using the previously agreed-upon function $\mathsf{f}_{\mathsf{default}}$, which checks the repayment rate based on the current clock time. If the default attempt fails, P is notified. Finally, once P has finished repaying the loan, it can unlock the collateral, if it can prove that function $\mathsf{f}_{\mathsf{unlock}}$ is satisfied. DU can not stop the collateral from being unlocked if the loan has been repaid, but is notified if P tries to unlock early.

Beyond the above description of the desired functionality behaviour, we also capture how corrupted parties can diverge from this. Corrupted parties can break some elements of confidentiality e.g. a corrupted TSP can leak the risk estimation function, or collusion between one TSP and DU will cause all user data to be revealed. Corrupted DP parties can also affect correctness, by tampering the data they hold about P users. Similarly, if the majority of participating TSP parties collude, they could replace the risk estimation function; or a corrupted DP could collude with DU (or P respectively) to cause an incorrect loan default (resp. unlock). The possible corruption scenarios are outlined in Table 2.

We give a formal specification of the ideal functionality, along with invariants for correctness and confidentiality below.

The functionality captures which assets are owned by each party through an internal table **Assets**, which includes "virtual" accounts such as the money committed as collateral. **Assets** is a simple key-value store that records the current asset balance for a certain account. The keys for each account are concatenations that can include the identity of the account holder, the institution where the money would be stored, or special keywords to denote that the asset is reserved for a specific use by the functionality. Our functionality guarantees that if the adversary aborts during setup, no party will lose any of its assets before the loan is initialised, and after the loan is agreed, the only way to unlock the collateral is by providing evidence of the contract being fulfilled or breached, according to the relevant functions. The only exception to the latter is that the DP can collude with the DU or the user P to force a loan defaulting or a collateral unlock respectively, as they would be able to jointly tamper with the source of transactions to trigger the unlocking.

Below the formal description. Note that we consider the functionality to be parameterised by a set of Technical Service Providers $\overline{\mathsf{TSP}}$. While these parties do not send any messages to the ideal functionality and are only used in the protocol realisation, their corruption in the ideal has implication to what the adversary learns. We refer to the fact that any member of this set is corrupted by $\exists\mathsf{TSP}$, or that the majority of them are corrupted as $\frac{1}{2}\mathsf{TSP}$.

Note that party DP does not have a top-level interface as it is not able to initiate any phase of the protocol, but can only respond to messages initiated from other parties. The TSP parties do not contribute any meaningful messages to the ideal protocol; however, due to the UC emulation statement, they also exist in the ideal world as a dummy party, whose corrupted status has security implications as described.

Fig. 4. Functionality $\mathcal{F}_{Loan}$

Data structures:
Assets[ ]: table of user assets

Upon receiving $\left(\textsc{ComputeRisk}, \overline{\mathsf{DP}}, \mathsf{DU}, \mathsf{p}\right)$ from P:

Send $\left(\textsc{RiskFunctionReq}, \mathsf{P}, \mathsf{p}, \overline{\mathsf{DP}}\right)$ to DU ($\mathbb{A}$ if corrupted) and receive ρ
if $\exists\mathsf{TSP}$ is corrupted **then**
  Send $(\textsc{RiskPred}, \rho)$ to $\mathbb{A}$ and wait for OK ▷ $\mathbb{A}$ *learns risk estimation function*
else
  Send $(\textsc{RiskPred}, \mathcal{P}(\rho))$ to $\mathbb{A}$ and wait for OK ▷ $\mathbb{A}$ *learns partial information*
for Any corrupted party DP in $\overline{\mathsf{DP}}$ **do**
  send $(\textsc{RiskFuncReq}, \mathsf{DP}, \mathsf{P}, \textbf{Assets}[\mathsf{DP}|\mathsf{P}])$ to $\mathbb{A}$ and wait for x ▷ $\mathbb{A}$ *can change user data*
  if $x \neq$ OK **then** **Assets**$[\mathsf{DP}|\mathsf{P}] \leftarrow$ x
$\pi \leftarrow$ *list of* **Assets**$[\mathsf{DP}\|\mathsf{P}]$ *for each* DP IN $\overline{\mathsf{DP}}$
if both $\exists\mathsf{TSP}$ and DU are corrupted **then**
  send $(\textsc{IllegalDecryption}, \pi)$ to $\mathbb{A}$ ▷ $\mathbb{A}$ *can learn* P*'s data*
if $\frac{1}{2}\mathsf{TSP}$ are corrupted **then**
  Send $(\textsc{NewRiskFunction}, \rho)$ to $\mathbb{A}$ and wait for r
else $r \leftarrow \rho(\mathsf{p}, \pi)$
send $(\textsc{Risk}, \mathsf{P}, \mathsf{p}, r)$ to DU ($\mathbb{A}$ if corrupted)

Upon receiving $(\textsc{CollateralisedLoan}, \mathsf{DU}, \mathsf{DP}^\star, \mathrm{BC}, \mathsf{p}, \mathsf{f}_{\mathsf{default}}, \mathsf{f}_{\mathsf{unlock}})$ from P:

Let $\mathsf{c} \leftarrow$ **Assets**[BC] ▷ *BC is an address on the blockchain*
Send $(\textsc{ConfirmTerms}, \mathsf{p}, \mathsf{c}, \mathsf{f}_{\mathsf{default}}, \mathsf{f}_{\mathsf{unlock}})$ to DU ($\mathbb{A}$ if corrupted) and wait for OK
if **Assets**$[\mathsf{DU}] < \mathsf{p}$ **then** abort
Assets$[\mathit{Lock}\|\mathsf{DU}\|\mathsf{P}] \leftarrow \mathsf{c}$
Assets$[\mathrm{BC}] \leftarrow 0$
Assets$[\mathsf{DP}^\star\|\mathsf{P}] \leftarrow \mathsf{p}$
Assets$[\mathsf{DU}] \mathrel{-}= \mathsf{p}$
Record $(\mathit{Loan}, \mathsf{DU}, \mathsf{DP}^\star, \mathsf{P}, \mathsf{p}, \mathsf{c}, \mathsf{f}_{\mathsf{default}}, \mathsf{f}_{\mathsf{unlock}})$
send $(\textsc{LoanAgreed}, \mathsf{P}, \mathsf{DU}, \mathsf{DP}^\star)$ to $\mathbb{A}$

Upon receiving $(\textsc{Repayment}, \mathsf{n}, \mathsf{DU})$ from P:

if **Assets**$[\mathsf{P}] < \mathsf{n}$ **then** return $(\textsc{InsufficientFunds}, \mathsf{n}, \mathsf{P})$
Assets$[\mathsf{P}] \mathrel{-}= \mathsf{n}$
Assets$[\mathit{Repayment}\|\mathsf{P}\|\mathsf{DU}] \mathrel{+}= \mathsf{n}$
if any of $\mathsf{P}, \mathsf{DU}, \mathsf{DP}^\star$ is corrupted **then**
  send GetTime to $\mathcal{F}_{clock}$ and receive t
  send $(\textsc{Payment}, \mathsf{P}, \mathsf{DU}, \mathsf{n}, t)$ to $\mathbb{A}$

Upon receiving $(\textsc{DefaultLoan}, P)$ from DU:

35: Find $(Loan, \mathsf{DU}, \mathsf{DP}^\star, \mathsf{P}, \mathsf{p}, \mathsf{c}, \mathsf{f}_{\mathsf{default}}, \mathsf{f}_{\mathsf{unlock}})$
36: **if** DU and $\mathsf{DP}^\star$ are corrupted **then**
37: send $(\textsc{DefaultClaim}, \mathsf{P}, \mathsf{DU})$ to $\mathbb{A}$ and wait for f ▷ $\mathbb{A}$ *can force illegitimate default*
38: send $\textsc{GetTime}$ to $\mathcal{F}_{clock}$ and receive t
39: **if** $\mathsf{f}_{\mathsf{default}}(t, \mathbf{Assets}[Repayment \| \mathsf{P} \| \mathsf{DU}])) = 1 \vee f = 1$ **then**
40: $\mathbf{Assets}[\mathsf{DU} \| blockchain] \mathrel{+}= \mathbf{Assets}[Lock \| \mathsf{DU} \| \mathsf{P}]$
41: $\mathbf{Assets}[Lock \| \mathsf{DU} \| \mathsf{P}] \leftarrow 0$
42: return $(\textsc{DefaultSuccess}, \mathsf{P}, \mathsf{DU})$ (send to $\mathbb{A}$ if any of $\mathsf{P}, \mathsf{DU}, \mathsf{DP}^\star$ is corrupted)
43: **else** return $\textsc{DefaultClaim}$ to P ($\mathbb{A}$ if corrupted) ▷ *notification that claim has failed*

44: **Upon receiving** $(\textsc{UnlockCollateral}, \mathsf{DU}, \mathrm{BC})$ from P:

45: Find $(Loan, \mathsf{DU}, \mathsf{DP}^\star, \mathsf{P}, \mathsf{p}, \mathsf{c}, \mathsf{f}_{\mathsf{default}}, \mathsf{f}_{\mathsf{unlock}})$
46: **if** P and $\mathsf{DP}^\star$ are corrupted **then**
47: send $(\textsc{UnlockClaim}, \mathsf{P}, \mathsf{DU})$ to $\mathbb{A}$ and wait for f ▷ $\mathbb{A}$ *can force illegitimate unlock*
48: send $\textsc{GetTime}$ to $\mathcal{F}_{clock}$ and receive t
49: **if** $\mathsf{f}_{\mathsf{unlock}}(t, \mathbf{Assets}[Repayment \| \mathsf{P} \| \mathsf{DU}])) = 1 \vee f = 1$ **then**
50: $\mathbf{Assets}[\mathrm{BC}] \mathrel{+}= \mathbf{Assets}[Lock \| \mathsf{DU} \| \mathsf{P}]$
51: $\mathbf{Assets}[Lock \| \mathsf{DU} \| \mathsf{P}] \leftarrow 0$
52: return $(\textsc{UnlockSuccess}, \mathsf{P}, \mathsf{DU}, \mathrm{BC})$ (send to $\mathbb{A}$ if any of $\mathsf{P}, \mathsf{DU}, \mathsf{DP}^\star$ is corrupted)
53: **else** return $\textsc{UnlockClaim}$ to DU ($\mathbb{A}$ if corrupted) ▷ *notification that claim has failed*

54: **Upon receiving** $(\textsc{ViewBalance}, \mathrm{A})$ from $(\mathsf{pid}, \mathsf{sid})$:

55: **if** $(\mathsf{pid}, \mathsf{sid}) \in \mathrm{A}$ **then** return $\mathbf{Assets}[\mathrm{A}]$

56: **Upon receiving** $(\textsc{Tx}, \mathsf{DP}, n)$ from $(\mathsf{pid}, \mathsf{sid})$:

57: $\mathbf{Assets}[\mathsf{DP} | (\mathsf{pid}, \mathsf{sid})] \leftarrow n$
58: **if** $(\mathsf{pid}, \mathsf{sid})$ orDP are corrupted **then** send $(\textsc{Tx}, \mathsf{DP}, n, (\mathsf{pid}, \mathsf{sid}))$ to $\mathbb{A}$

We now formulate the invariants required to ensure the correctness of the protocol.

Definition 7 (Correctness). $\mathcal{F}_{Loan}$ *preserves* correctness *if*

- *when executing* ComputeRisk*, a corrupted majority of* TSP*s does not overwrite the risk estimation function, and no corrupted* DP *tampers with user data (i.e the branches in line 13 and line 9 are not executed, respectively)*
- *when executing* DefaultLoan*, colluding* $\mathsf{DP}^\star, \mathsf{DU}$ *do not forcibly cause an early default (i.e. the value of* f *in the check on line 39 is 0)*
- *when executing* UnlockCollateral*, colluding* $\mathsf{DP}^\star, \mathsf{P}$ *do not forcibly cause an early unlock (i.e. the value of* f *in the check on line 49 is 0)*

7 Loan Protocol and More

Due to space constraints, we present the protocol that realizes the loan functionality, along with the security proof in the full version. We also show how this result can be extended to the setting where the function evaluated takes input from a ledger and includes a benchmark section where we empirically evaluate how FHE performs when computing a simple function for credit scoring. For the implementation, we relied on the TFHE-rs library [59] (version 1.5). We refer to the full version for more detail.

P	DU	DP	TSP	Outcome
·	·	·	1	ρ is leaked
·	·	·	maj	ρ can be replaced
·	·	any	·	P data is altered
·	Y	·	1	P data is leaked
·	Y	$\mathsf{DP}^\star$	·	Force loan default
Y	·	$\mathsf{DP}^\star$	·	Force collateral unlock

Table 2. Possible outcome for corruption scenarios in $\mathcal{F}_{loan}$. P, DU corruptions are marked with Y. DP corruption are marked as "any" for $\overline{\mathsf{DP}}$ parties during Phase 1, and $\mathsf{DP}^\star$ for the supporting party chosen during Phase 2. The TSP column distinguishes whether at least 1 party or the majority ("maj") of $\overline{\mathsf{TSP}}$ parties are corrupted.

Acknowledgments. This research is supported by the Input Output Research Hub (IORH) of the University of Edinburgh. Moreover, Raffaella Calabrese is grateful for financial support from the Economic and Social Research Council [grant number ES/W010259/1]. We thank the anonymous reviewers of ACNS 2026, and in particular the shepherd whose advice led to the version of the paper that has been accepted.

Disclosure of Interests. Tzameret Rubin is on the FCA PRISM multi-stakeholder committee (Prioritisation and Real-world Insights Selection Matrix), advising about Open Finance use cases with the greatest societal impact. This research is supported by the Input Output Research Hub (IORH) of the University of Edinburgh. Moreover, Raffaella Calabrese is grateful for financial support from the Economic and Social Research Council [grant number ES/W010259/1].

References

1. An, J.H., Dodis, Y., Rabin, T.: On the security of joint signature and encryption. In: Knudsen, L.R. (ed.) Advances in Cryptology – EUROCRYPT 2002. Lecture Notes in Computer Science, vol. 2332, pp. 83–107. Springer Berlin Heidelberg, Germany, Amsterdam, The Netherlands (Apr 28 – May 2, 2002). https://doi.org/10.1007/3-540-46035-7_6
2. Andolfo, L., Coppolino, L., D'Antonio, S., Mazzeo, G., Romano, L., Ficke, M., Hollum, A., Vaydia, D.: Privacy-Preserving Credit Scoring via Functional Encryption, p. 31–43. Springer International Publishing (2021). https://doi.org/10.1007/978-3-030-87010-2_3, http://dx.doi.org/10.1007/978-3-030-87010-2_3
3. Briones de Araluze, G., Cassinello Plaza, N.: The relevance of initial trust and social influence in the intention to use open banking-based services: An empirical study. Sage Open **13**(3) (Jul 2023). https://doi.org/10.1177/21582440231187607, http://dx.doi.org/10.1177/21582440231187607
4. Armknecht, F., Katzenbeisser, S., Peter, A.: Group homomorphic encryption: characterizations, impossibility results, and applications. Designs, Codes and Cryptography **67**(2), 209–232 (2013). https://doi.org/10.1007/s10623-011-9601-2
5. Arner, D.W., Buckley, R.P., Zetzsche, D.A.: Open Banking, Open Data, and Open Finance: Lessons from the European Union, p. 147–172. Oxford University Press-New York (Mar 2022). https://doi.org/10.1093/oso/9780197582879.003.0009, http://dx.doi.org/10.1093/oso/9780197582879.003.0009

6. Badertscher, C., Campanelli, M., Ciampi, M., Russo, L., Siniscalchi, L.: Universally composable SNARKs with transparent setup without programmable random oracle. In: Kalai, Y.T., Kamara, S.F. (eds.) Advances in Cryptology – CRYPTO 2025, Part VII. Lecture Notes in Computer Science, vol. 16006, pp. 225–258. Springer, Cham, Switzerland, Santa Barbara, CA, USA (Aug 17–21, 2025). https://doi.org/10.1007/978-3-032-01907-3_8
7. Badertscher, C., Maurer, U., Tschudi, D., Zikas, V.: Bitcoin as a transaction ledger: A composable treatment. Journal of Cryptology **37**(2), 18 (Apr 2024). https://doi.org/10.1007/s00145-024-09493-7
8. Ballandras, M., de Bellabre, M., Bergerat, L., Bonte, C., Bootland, C., Curtis, B.R., Khatib, J., Klemsa, J., Meyre, A., Montaigu, T., Orfila, J.B., Sarlin, N., Tap, S., Testé, D.: TFHE-rs: A (practical) handbook. Github (2025), https://github.com/zama-ai/tfhe-rs-handbook
9. Bellare, M., Desai, A., Pointcheval, D., Rogaway, P.: Relations among notions of security for public-key encryption schemes. In: Krawczyk, H. (ed.) Advances in Cryptology – CRYPTO'98. Lecture Notes in Computer Science, vol. 1462, pp. 26–45. Springer Berlin Heidelberg, Germany, Santa Barbara, CA, USA (Aug 23–27, 1998). https://doi.org/10.1007/BFb0055718
10. Bernard, O., Joye, M., Smart, N.P., Walter, M.: Drifting towards better error probabilities in fully homomorphic encryption schemes. In: Fehr, S., Fouque, P.A. (eds.) Advances in Cryptology – EUROCRYPT 2025, Part VIII. Lecture Notes in Computer Science, vol. 15608, pp. 181–211. Springer, Cham, Switzerland, Madrid, Spain (May 4–8, 2025). https://doi.org/10.1007/978-3-031-91101-9_7
11. Bhimrajka, N., Kondi, Y., Noble, D., Varadarajan, S.: Deploying mpc in open finance: Challenges and opportunities. Talk given at RWC 2025 (2025), video at https://youtu.be/xpLhpmpyF2g
12. Bobolz, J., Farshim, P., Kohlweiss, M., Takahashi, A.: The brave new world of global generic groups and UC-secure zero-overhead SNARKs. In: Boyle, E., Mahmoody, M. (eds.) TCC 2024: 22nd Theory of Cryptography Conference, Part I. Lecture Notes in Computer Science, vol. 15364, pp. 90–124. Springer, Cham, Switzerland, Milan, Italy (Dec 2–6, 2024). https://doi.org/10.1007/978-3-031-78011-0_4
13. Braun, H.: Jpmorgan (jpm) to accept bitcoin (btc) etfs, ibit, as loan collateral in expansion of crypto access: Bloomberg (2025), https://www.coindesk.com/business/2025/06/04/jpmorgan-to-accept-bitcoin-etfs-as-loan-collateral-in-expansion-of-crypto-access-bloomberg, accessed: 2025-09-11
14. Brzuska, C., Canard, S., Fontaine, C., Phan, D.H., Pointcheval, D., Renard, M., Sirdey, R.: Relations among new CCA security notions for approximate FHE. IACR Communications in Cryptology (CiC) **2**(1), 20 (2025). https://doi.org/10.62056/aee0iv7sf
15. Canetti, R.: Universally composable security. Journal of the ACM **67**(5), 1–94 (Sep 2020). https://doi.org/10.1145/3402457, http://dx.doi.org/10.1145/3402457
16. Canetti, R., Hogan, K., Malhotra, A., Varia, M.: A universally composable treatment of network time. In: Köpf, B., Chong, S. (eds.) CSF 2017: IEEE 30th Computer Security Foundations Symposium. pp. 360–375. IEEE Computer Society Press, Santa Barbara, CA, USA (Aug 21–25, 2017). https://doi.org/10.1109/CSF.2017.38
17. Canetti, R., Krawczyk, H., Nielsen, J.B.: Relaxing chosen-ciphertext security. In: Boneh, D. (ed.) Advances in Cryptology – CRYPTO 2003. Lecture Notes in Computer Science, vol. 2729, pp. 565–582. Springer Berlin Heidelberg, Germany, Santa

Barbara, CA, USA (Aug 17–21, 2003). https://doi.org/10.1007/978-3-540-45146-4_33

18. Canetti, R., Shahaf, D., Vald, M.: Universally composable authentication and key-exchange with global PKI. In: Cheng, C.M., Chung, K.M., Persiano, G., Yang, B.Y. (eds.) PKC 2016: 19th International Conference on Theory and Practice of Public Key Cryptography, Part II. Lecture Notes in Computer Science, vol. 9615, pp. 265–296. Springer Berlin Heidelberg, Germany, Taipei, Taiwan (Mar 6–9, 2016). https://doi.org/10.1007/978-3-662-49387-8_11
19. Casolaro, A.M.B., Rauber, G.N., de Lima, U.S.M.: Open banking: a systematic literature review. Journal of Banking Regulation **26**(3), 340–355 (Dec 2024). https://doi.org/10.1057/s41261-024-00262-x, http://dx.doi.org/10.1057/s41261-024-00262-x
20. Chatzigiannis, P., Baldimtsi, F., Chalkias, K.: SoK: Blockchain light clients. In: Eyal, I., Garay, J.A. (eds.) FC 2022: 26th International Conference on Financial Cryptography and Data Security. Lecture Notes in Computer Science, vol. 13411, pp. 615–641. Springer, Cham, Switzerland, Grenada (May 2–6, 2022). https://doi.org/10.1007/978-3-031-18283-9_31
21. Chen, M., Dey, P., Ganesh, C., Mukherjee, P., Sarkar, P., Sasmal, S.: Universally composable non-interactive zero-knowledge from sigma protocols via a new straight-line compiler. In: Jager, T., Pan, J. (eds.) PKC 2025: 28th International Conference on Theory and Practice of Public Key Cryptography, Part I. Lecture Notes in Computer Science, vol. 15674, pp. 381–417. Springer, Cham, Switzerland, Røros, Norway (May 12–15, 2025). https://doi.org/10.1007/978-3-031-91820-9_13
22. Chiesa, A., Fenzi, G.: zkSNARKs in the ROM with unconditional UC-security. In: Boyle, E., Mahmoody, M. (eds.) TCC 2024: 22nd Theory of Cryptography Conference, Part I. Lecture Notes in Computer Science, vol. 15364, pp. 67–89. Springer, Cham, Switzerland, Milan, Italy (Dec 2–6, 2024). https://doi.org/10.1007/978-3-031-78011-0_3
23. Chillotti, I., Gama, N., Georgieva, M., Izabachène, M.: TFHE: Fast fully homomorphic encryption over the torus. Journal of Cryptology **33**(1), 34–91 (Jan 2020). https://doi.org/10.1007/s00145-019-09319-x
24. Ciampi, M., Ishaq, M., Magdon-Ismail, M., Ostrovsky, R., Zikas, V.: Fairmm: A fast and frontrunning-resistant crypto market-maker. In: Dolev, S., Katz, J., Meisels, A. (eds.) Cyber Security, Cryptology, and Machine Learning - 6th International Symposium, CSCML 2022, Be'er Sheva, Israel, June 30 - July 1, 2022, Proceedings. pp. 428–446. Lecture Notes in Computer Science, Springer (2022). https://doi.org/10.1007/978-3-031-07689-3_31, https://doi.org/10.1007/978-3-031-07689-3_31
25. Ciampi, M., Kiayias, A., Shen, Y.: Universal composable transaction serialization with order fairness. In: Reyzin, L., Stebila, D. (eds.) Advances in Cryptology – CRYPTO 2024, Part II. Lecture Notes in Computer Science, vol. 14921, pp. 147–180. Springer, Cham, Switzerland, Santa Barbara, CA, USA (Aug 18–22, 2024). https://doi.org/10.1007/978-3-031-68379-4_5
26. Ciampi, M., Kiayias, A., Shen, Y.: Universally composable transaction order fairness: Refined definitions and adaptive security. In: Hanaoka, G., Yang, B.Y. (eds.) Advances in Cryptology – ASIACRYPT 2025, Part II. Lecture Notes in Computer Science, vol. 16246, pp. 305–337. Springer, Singapore, Singapore, Melbourne, VIC, Australia (Dec 8–12, 2025). https://doi.org/10.1007/978-981-95-5096-8_10

27. Ciampi, M., Visconti, I.: Efficient NIZK arguments with straight-line simulation and extraction. In: Beresford, A.R., Patra, A., Bellini, E. (eds.) CANS 22: 21st International Conference on Cryptology and Network Security. Lecture Notes in Computer Science, vol. 13641, pp. 3–22. Springer, Cham, Switzerland, Dubai, United Arab Emirates (Nov 13–15, 2022). https://doi.org/10.1007/978-3-031-20974-1_1
28. Dagher, G.G., Bünz, B., Bonneau, J., Clark, J., Boneh, D.: Provisions: Privacy-preserving proofs of solvency for Bitcoin exchanges. In: Ray, I., Li, N., Kruegel, C. (eds.) ACM CCS 2015: 22nd Conference on Computer and Communications Security. pp. 720–731. ACM Press, Denver, CO, USA (Oct 12–16, 2015). https://doi.org/10.1145/2810103.2813674
29. Dong, C., Wang, Z., Chen, S., Xiang, Y.: BBM: A Blockchain-Based Model for Open Banking via Self-sovereign Identity, p. 61–75. Springer International Publishing (2020). https://doi.org/10.1007/978-3-030-59638-5_5, http://dx.doi.org/10.1007/978-3-030-59638-5_5
30. Döttling, N., Dujmovic, J.: Maliciously circuit-private FHE from information-theoretic principles. In: Dachman-Soled, D. (ed.) ITC 2022: 3rd Conference on Information-Theoretic Cryptography. Leibniz International Proceedings in Informatics (LIPIcs), vol. 230, pp. 4:1–4:21. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Cambridge, MA, USA (Jul 5–7, 2022). https://doi.org/10.4230/LIPIcs.ITC.2022.4
31. van Egmond, M.B., Dunning, V., van den Berg, S., Rooijakkers, T., Sangers, A., Poppe, T., Veldsink, J.: Privacy-preserving anti-money laundering using secure multi-party computation. In: Clark, J., Shi, E. (eds.) FC 2024: 28th International Conference on Financial Cryptography and Data Security, Part II. Lecture Notes in Computer Science, vol. 14745, pp. 331–349. Springer, Cham, Switzerland, Willemstad, Curaçao (Mar 4–8, 2024). https://doi.org/10.1007/978-3-031-78679-2_18
32. Ernstberger, J., Lauinger, J., Wu, Y., Gervais, A., Steinhorst, S.: ORIGO: Proving provenance of sensitive data with constant communication. Cryptology ePrint Archive, Report 2024/447 (2024), https://eprint.iacr.org/2024/447
33. Fang, J., Zhu, J.: The impact of open banking on traditional lending in the brics. Finance Research Letters **58**, 104300 (Dec 2023). https://doi.org/10.1016/j.frl.2023.104300, http://dx.doi.org/10.1016/j.frl.2023.104300
34. Grassi, L., Figini, N., Fedeli, L.: How does a data strategy enable customer value? the case of fintechs and traditional banks under the open finance framework. Financial Innovation **8**(1) (Aug 2022). https://doi.org/10.1186/s40854-022-00378-x, http://dx.doi.org/10.1186/s40854-022-00378-x
35. He, H., Wang, Z., Jain, H., Jiang, C., Yang, S.: A privacy-preserving decentralized credit scoring method based on multi-party information. Decision Support Systems **166**, 113910 (Mar 2023). https://doi.org/10.1016/j.dss.2022.113910, http://dx.doi.org/10.1016/j.dss.2022.113910
36. HSBC, ATI: Review of regulator-informed application of homomorphic encryption in finance and adjacent sectors.https://www.turing.ac.uk/research/research-projects/review-regulator-informed-application-homomorphic-encryption-finance-and (2022), online; accessed 9 September 2024
37. IBM: Intesa Sanpaolo and IBM secure digital transactions with fully homomorphic encryption. https://www.ibm.com/blog/intesa-sanpaolo-ibm-secure-digital-transactions-fhe/ (2024), online; accessed 9 September 2024

38. Jiang, E., Qin, B., Wang, Q., Wang, Z., Wu, Q., Weng, J., Li, X., Wang, C., Ding, Y., Zhang, Y.: Decentralized finance (DeFi): A survey. Cryptology ePrint Archive, Report 2023/1210 (2023), https://eprint.iacr.org/2023/1210
39. Joye, M.: TFHE public-key encryption revisited. Cryptology ePrint Archive, Report 2023/603 (2023), https://eprint.iacr.org/2023/603
40. Kluczniak, K.: Circuit privacy for FHEW/TFHE-style fully homomorphic encryption in practice. Cryptology ePrint Archive, Report 2022/1459 (2022), https://eprint.iacr.org/2022/1459
41. Kondi, Y., Kanwar, K., Shetty, S., Arjun, A., Prakash, J.: Open Finance Revisited: Strengthening Data Governance with Cryptographic Privacy and Auditability. https://md.silencelaboratories.com/s/CQW5fu-fP (2024)
42. Krause, D.: Rewriting crypto regulation: Trump's actions and the future of digital financial technology. Available at SSRN 5119438 (2025)
43. Lee, J., Khan, V.M.: Developing cybersecurity in open banking and open finance: the application of blockchain technology and decentralised autonomous organisations, p. 113–138. Edward Elgar Publishing (Jan 2025). https://doi.org/10.4337/9781803929996.00013, http://dx.doi.org/10.4337/9781803929996.00013
44. Li, B., Micciancio, D.: On the security of homomorphic encryption on approximate numbers. In: Canteaut, A., Standaert, F.X. (eds.) Advances in Cryptology – EUROCRYPT 2021, Part I. Lecture Notes in Computer Science, vol. 12696, pp. 648–677. Springer, Cham, Switzerland, Zagreb, Croatia (Oct 17–21, 2021). https://doi.org/10.1007/978-3-030-77870-5_23
45. Loftus, J., May, A., Smart, N.P., Vercauteren, F.: On CCA-secure somewhat homomorphic encryption. In: Miri, A., Vaudenay, S. (eds.) SAC 2011: 18th Annual International Workshop on Selected Areas in Cryptography. Lecture Notes in Computer Science, vol. 7118, pp. 55–72. Springer Berlin Heidelberg, Germany, Toronto, Ontario, Canada (Aug 11–12, 2012). https://doi.org/10.1007/978-3-642-28496-0_4
46. Malinka, K., Hujňák, O., Hanáček, P., Hellebrandt, L.: E-banking security study—10 years later. IEEE Access **10**, 16681–16699 (2022)
47. Manulis, M., Nguyen, J.: Fully homomorphic encryption beyond IND-CCA1 security: Integrity through verifiability. Cryptology ePrint Archive, Report 2024/202 (2024), https://eprint.iacr.org/2024/202
48. Marcolla, C., Sucasas, V., Manzano, M., Bassoli, R., Fitzek, F.H.P., Aaraj, N.: Survey on fully homomorphic encryption, theory, and applications. Cryptology ePrint Archive, Report 2022/1602 (2022), https://eprint.iacr.org/2022/1602
49. Morgan, J.: Blockchain asset tokenization with kinexys.https://www.jpmorgan.com/insights/payments/wallets/blockchain-kinexys-asset-tokenization (2022), accessed: 2025-09-11
50. Mukhopadhyay, I., Ghosh, A.: Blockchain-Based Framework for Managing Customer Consent in Open Banking, p. 77–90. Springer Singapore (Nov 2020). https://doi.org/10.1007/978-981-15-9317-8_3, http://dx.doi.org/10.1007/978-981-15-9317-8_3
51. Nzomiwu, A.C., Nwobodo, M.N.: Beyond the hype: Evaluating the real-world impact of ai and distributed ledger technologies in financial risk management. Available at SSRN 5195554 (2025). https://doi.org/10.2139/ssrn.5195554, http://dx.doi.org/10.2139/ssrn.5195554
52. OBL: The Open Banking Impact Report 7. https://openbanking.foleon.com/live-publications/the-obl-impactreport-7/ (2025), online; accessed 5 June 2025

53. Ostrovsky, R., Paskin-Cherniavsky, A., Paskin-Cherniavsky, B.: Maliciously circuit-private FHE. In: Garay, J.A., Gennaro, R. (eds.) Advances in Cryptology – CRYPTO 2014, Part I. Lecture Notes in Computer Science, vol. 8616, pp. 536–553. Springer Berlin Heidelberg, Germany, Santa Barbara, CA, USA (Aug 17–21, 2014). https://doi.org/10.1007/978-3-662-44371-2_30
54. Qin, K., Zhou, L., Afonin, Y., Lazzaretti, L., Gervais, A.: CeFi vs. DeFi – Comparing Centralized to Decentralized Finance (2021)
55. Wang, D., Wu, S., Lin, Z., Wu, L., Yuan, X., Zhou, Y., Wang, H., Ren, K.: Towards a first step to understand flash loan and its applications in defi ecosystem. In: Proceedings of the Ninth International Workshop on Security in Blockchain and Cloud Computing. p. 23–28. ASIA CCS '21, ACM (May 2021). https://doi.org/10.1145/3457977.3460301, http://dx.doi.org/10.1145/3457977.3460301
56. Wang, H., Ma, S., Dai, H.N., Imran, M., Wang, T.: Blockchain-based data privacy management with nudge theory in open banking. Future Generation Computer Systems **110**, 812–823 (Sep 2020). https://doi.org/10.1016/j.future.2019.09.010, http://dx.doi.org/10.1016/j.future.2019.09.010
57. Xu, J., Vadgama, N.: From Banks to DeFi: the Evolution of the Lending Market, p. 53–66. Springer International Publishing (2022). https://doi.org/10.1007/978-3-030-78184-2_6, http://dx.doi.org/10.1007/978-3-030-78184-2_6
58. Yang, F., Abedin, M.Z., Hajek, P.: An explainable federated learning and blockchain-based secure credit modeling method. European Journal of Operational Research **317**(2), 449–467 (Sep 2024). https://doi.org/10.1016/j.ejor.2023.08.040, http://dx.doi.org/10.1016/j.ejor.2023.08.040
59. Zama: TFHE-rs: A Pure Rust Implementation of the TFHE Scheme for Boolean and Integer Arithmetics Over Encrypted Data (2022), https://github.com/zama-ai/tfhe-rs
60. Zhang, F., Maram, D., Malvai, H., Goldfeder, S., Juels, A.: DECO: Liberating web data using decentralized oracles for TLS. In: Ligatti, J., Ou, X., Katz, J., Vigna, G. (eds.) ACM CCS 2020: 27th Conference on Computer and Communications Security. pp. 1919–1938. ACM Press, Virtual Event, USA (Nov 9–13, 2020). https://doi.org/10.1145/3372297.3417239