Investcoin: A System for Privacy-Preserving Investments [⋆]

Filipp Valovich
Email: filipp.valovich@rub.de

Horst Görtz Institute for IT Security
Faculty of Mathematics
Ruhr-Universität Bochum, Universitätsstraße 150, 44801 Bochum, Germany

Abstract. This work presents a new framework for Privacy-Preserving Investment systems in a distributed model. In this model, independent investors can transfer funds to independent projects, in the same way as it works on crowdfunding platforms. The framework protects the investors' single payments from being detected (by any other party), only the sums of each investor's payments are revealed (e.g to the system). Likewise, the projects' single incoming payments are concealed and only the final sums of the incoming payments for every project are revealed. In this way, no other party than the investor (not even the system administration) can detect how much she paid to any single project. Though it is still possible to confidentially exchange any part of an investment between any pair of investors, such that market liquidity is unaffected by the system. On top, our framework allows a privacy-preserving return of a multiple of all the held investments (e.g. interest payments or dividends) to the indivdual investors while still revealing nothing else than the sum of all returns for every investor. We provide reasonable security guarantees for this framework that are based on common notions from the Secure Multi-Party Computation (SMPC) literature. As an instantiation for this framework we present Investcoin. This is a proper combination of three cryptographic protocols, namely a Private Stream Aggregation scheme, a Commitment scheme and a Range test and it is usable in connection with any existing currency. The security of the three protocols is based on the DDH assumption. Thus, by a composition theorem from the SMPC literature, the security of the resulting Investcoin protocol is also based on the DDH assumption. Furthermore, we provide a simple decentralised key generation protocol for Investcoin that supports dynamic join, dynamic leave and fault-tolarance of investors and moreover achieves some security guarantees against malicious investors.

1 Introduction

The promise of performance benefit by using technologies like online-outsourcing and cloud-computing goes along with the loss of control over individual data. Therefore the public awareness of data protection increases. We use encryption and privacy technologies to protect our electronic messages, our consumer behaviour or patient records. In this work, we put the following question up for discussion: why is there only minor attention paid to the protection of sensitive *financial data* in the public? Indeed the requirement to trust in financial institutes may be an obstacle for the trade secrecy of companies. On the one hand, transactions on organised markets are registered by electronic systems, audited and eventually get under the control of the system administration (e.g. it can refuse a transaction). In some cases this is desired: e.g. it should be possible to detect a company financing criminal activities. On the

[⋆] The research was supported by the DFG Research Training Group GRK 1817/1

other hand, we would like to protect the trade secrecy of the companies. In this sense, there is a transparency/confidentiality trade-off in organised markets, such as exchange or to some extent also crowdfunding platforms.
In this work we address the problem of providing adequate privacy guarantees to investors. As observed by Nofer [18], although there is no observable significant effect concerning "the impact of privacy violations on the investment amount, (...) one has to remember that trust influences behavior (...) and privacy issues influences trust (...) and therefore an indirect influence still exists". Conversely, this means that individuals would participate more in investments if their privacy is protcted. As an effect, the reporting of investors concerning wins and losses (and therefore risks) becomes more reliable [14,9,3]. As further motivation of our work, the possibility to circumvent certain regulatories, such as financial sanctions by order of "repressive" countries, may be desired. Investors may look for a way to invest in sanctioned companies without being traced by their home country.
Consequently, the objective of this work is to solve privacy issues by concealing particular investment decisions but offering transparency of "aggregated" investment decisions. In this regard we introduce a Distributed Investment Encryption (DIE) scheme for the aggregation of the investments of a number of different investors funding different projects on an electronic platform. A DIE scheme maintains market liquidity, i.e. the scheme does not affect the possibility to trade assets among investors. Its cryptographic security is set through the definition of the Privacy-Preserving Investment (PPI) system. Informally, a PPI system conceals the single payments of investors from being traced but reveals to the system only the aggregates of the investors' payments. Similarly, the projects' single incoming payments are concealed and only the final sums of the incoming payments for every project are revealed. Moreover, a PPI system conceals the single returns (representing interest payments, coupons or dividends) from every single project to every single investor but reveals (to the system administration) the aggregated return of every single investor. Therefore, up to a certain extent, a PPI system simultaneously maintains transparency (e.g. taxes on the final return of every investor can be raised) and trade secrecy.
As a particular PPI system we present Investcoin, a combination of three cryptographic protocols: a Private Stream Aggregation (PSA) scheme, first introduced by Shi et al. [23], a (homomorphic) Commitment scheme and a Range test for commited values, all secure under the Decisional Diffie-Hellman assumption. Informally, the PSA scheme is used for the secure aggregation of funds for every particular project and the homomorphic Commitment scheme is used for the secure aggregation of all investments and returns of every particular investor. The Range test ensures that investments are not negative. We provide a simple secret sharing key generation protocol for Investcoin, that allows investors to dynamically join, leave or fail during the protocol execution and prevents investors from some malicious cheating.

Related work. The notion of Commitment schemes (first in [4,6]) is well-established in the literature. The notion of PSA was introduced in [23]. A PSA scheme is a cryptographic protocol which enables a number of users to individually and securely send encrypted time-series data to an untrusted aggregator requiring each user to send exactly one message per time-step. The aggregator is able to decrypt the aggregate of all data per time-step, but cannot retrieve any further information about the individual data. In [23] a security definition for PSA and a secure instantiation were provided. Joye and Libert [15] provided a scheme with a tighter security reduction and Benhamouda et al. [2] generalised the scheme in [15]. By lowering the security requirements established in [23], Valovich and Aldà [25] provided general conditions

for the existence of secure PSA schemes, based on key-homomorphic weak PRFs.
Investcoin is not a classical cryptocurrency. It can be thought of as a cryptographic layer on top of any currency used for investments, similar to what Zerocoin is intended to be for Bitcoin. Bitcoin is the first cryptocurrency, introduced by Satoshi Nakamoto [17]. Currently it is the cryptocurrency with the largest market capitalisation. Bitcoin is as a peer-to-peer payment system where transactions are executed directly between users without interaction of any intermediate party. The transactions are verified by the users of the network and publicly recorded in a blockchain, a distributed database. Zerocoin was proposed by Miers et al. [16] as an extension for Bitcoin (or any other cryptocurrency) providing cryptographic anonymity to recorded transactions in the blockchain (Bitcoin itself provides only pseudonymity). This is achieved by the use of a seperate mixing procedure based on Commitment schemes. Therefore particular transactions cannot be publicly traced back to particular Bitcoin adresses anymore. This is also the main principle of a PPI system: no investment in a particular project can be traced back to a particular investor. In this regard Investcoin has similarities with Zerocoin.
Methods for market regulation through aggregated privacy-preserving risk reporting were studied by Abbe et al. [1]. They constructed protocols allowing a number of users to securely compute aggregated risk measures based on summations and inner products. Flood et al. [12] considered balancing transparency and confidentiality for financial regulation by investigating cryptographic tools for statistical data privacy.

2 Preliminaries

In this section we provide the description of our investment model, the basic protocols underlying Investcoin and their corresponding security definitions.

2.1 Model

As initial situation we consider a network consisting of n investors and λ projects to be funded by the investors. As an analogy from the real world one can think of a crowdfunding platform or an exchange system where projects or companies try to collect funds from various individual investors. Each investor N_i, $i = 1, \ldots, n$, is willing to invest the amount $x_{i,j} \geqslant 0$ to the project P_j, $j = 1, \ldots, \lambda$, thus the total amount invested by N_i is $\sum_{j=1}^{\lambda} x_{i,j}$ and the total amount received by project P_j is $\sum_{i=1}^{n} x_{i,j}$. Moreover, there exists an administration (which may be the administration of the crowdfunding platform). The investors and the project managements *are not required to trust the administration.*
We consider a series of investment rounds. An investment round denotes the moment when the payments of all participating investors are registered by the administration of the system. From round to round the values n and λ may change, i.e. investors and projects may join or leave the network before any round.
After an investment round is over and the time comes to give a return to the investors (i.e. at maturity), the management of each project P_j publishes some value α_j defining the return for each investor (i.e. an indicator of economic growth, interest yield, dividend yield or similar). The untrusted system administration (or simply system) serves as a pool for the distribution of investments to the projects and of returns to the investors: first, for all $i = 1, \ldots, n$ it collects the total amount $\sum_{j=1}^{\lambda} x_{i,j}$ invested by investor N_i and rearranges the union of the total amounts into the aggregated investment $\sum_{i=1}^{n} x_{i,j}$ for project P_j for all $j = 1, \ldots, \lambda$; at maturity date, for all

$j = 1, \ldots, \lambda$ it collects the total returns $\alpha_j \sum_{i=1}^{n} x_{i,j}$ of the projects and rearranges the union of the total returns into the returns $\sum_{j=1}^{\lambda} \alpha_j x_{i,j}$ of the investors.
While the investors do not have to trust each other nor the system (i.e. an investor doesn't want the others to know her financial data), we consider a honest-but-curious model where the untrusted system administration tries to compromise investors to build a coalition. This coalition tries to infer additional information about uncompromised investors, but under the constraint of honestly following the investment protocol. On the other hand, we allow investors that are not part of the coalition, to execute some malicious behaviour, that we will concretise later. Thereby we have the following objectives in each investment round:

Security

- Hiding: For all $i = 1, \ldots, n$ the only financial data of investor N_i (if uncompromised) known to the system is $C_i = \sum_{j=1}^{\lambda} x_{i,j}$ and $E_i = \sum_{j=1}^{\lambda} \alpha_j x_{i,j}$. For all $j = 1, \ldots, \lambda$ the only financial data of project P_j known to the system is $X_j = \sum_{i=1}^{n} x_{i,j}$ and α_j. Particularly, the single investments of N_i to $P_1, \ldots, P_\lambda$ should remain concealed.
- Binding: Investors may not announce an incorrect investment, i.e. if N_i has send $x_{i,j}$ to P_j, then P_j should also receive $x_{i,j}$ from N_i.
- For all i, j it holds that $x_{i,j} \geqslant 0$, i.e. no investor can 'steal' money from a project.

Correctness

- For all i, if C_i is the real aggregate of N_i's investments, then $C_i = \sum_{j=1}^{\lambda} x_{i,j}$, the system knows C_i and can charge the bank account of N_i with amount C_i.
- For all j, if X_j is the real aggregate of P_j's funds, then $X_j = \sum_{i=1}^{n} x_{i,j}$, the system knows X_j and transfers the amount X_j to the bank account of P_j.
- For all i, if E_i is the real aggregate of N_i's returns, then $E_i = \sum_{j=1}^{\lambda} \alpha_j x_{i,j}$, the system knows E_i and transfers the amount E_i to the bank account of N_i.
- If one of these conditions is violated (e.g. on the purpose of stealing money), then the injured party should be able to detect this fact and to prove it to the network latest after the end of the corresponding investment round.

Now we provide the building blocks for a scheme satisfying these objectives.

2.2 Private Stream Aggregation

In this section, we define Private Stream Aggregation (PSA) and provide a security definition. The notion of PSA was introduced by Shi et al. [23].

The definition of Private Stream Aggregation. A PSA scheme is a protocol for safe distributed time-series data transfer which enables the receiver (here: the system administrator) to learn nothing else than the sums $\sum_{i=1}^{n} x_{i,j}$ for $j = 1, 2, \ldots$, where $x_{i,j}$ is the value of the ith participant in (time-)step j and n is the number of participants (here: investors). Such a scheme needs a key exchange protocol for all n investors together with the administrator as a precomputation, and requires each investor to send exactly one message (namely the amount to spend for a particular project) in each step $j = 1, 2, \ldots$.

Definition 1 (Private Stream Aggregation [23]) *Let κ be a security parameter, $\mathcal{D}$ a set and $n, \lambda \in \mathbb{N}$ with $n = poly(\kappa)$ and $\lambda = poly(\kappa)$. A* Private Stream Aggregation *(PSA) scheme $\Sigma = (\mathsf{Setup}, \mathsf{PSAEnc}, \mathsf{PSADec})$ is defined by three ppt algorithms:*

Setup: $(\mathsf{pp}, T, s_0, s_1, \ldots, s_n) \leftarrow \mathsf{Setup}(1^\kappa)$ *with public parameters* pp*,* $T = \{t_1, \ldots, t_\lambda\}$ *and secret keys* s_i *for all* $i = 1, \ldots, n$*.*

PSAEnc: *For* $t_j \in T$ *and all* $i = 1, \ldots, n$*:* $c_{i,j} \leftarrow \mathsf{PSAEnc}_{s_i}(t_j, x_{i,j})$ *for* $x_{i,j} \in \mathcal{D}$*.*

PSADec: *Compute* $\sum_{i=1}^n x'_{i,j} = \mathsf{PSADec}_{s_0}(t_j, c_{1,j}, \ldots, c_{n,j})$ *for* $t_j \in T$ *and ciphers* $c_{1,j}, \ldots, c_{n,j}$*. For all* $t_j \in T$ *and* $x_{1,j}, \ldots, x_{n,j} \in \mathcal{D}$ *the following holds:*

$$\mathsf{PSADec}_{s_0}(t_j, \mathsf{PSAEnc}_{s_1}(t_j, x_{1,j}), \ldots, \mathsf{PSAEnc}_{s_n}(t_j, x_{n,j})) = \sum_{i=1}^{n} x_{i,j}.$$

The system parameters pp are public and constant for all t_j with the implicit understanding that they are used in Σ. Every investor encrypts her amounts $x_{i,j}$ with her own secret key s_i and sends the ciphertext to the administrator. If the administrator receives the ciphertexts of *all* investors for some t_j, it can compute the aggregate of the investors' data using the decryption key s_0.
While in [23], the $t_j \in T$ were considered to be time-steps within a time-series (e.g. for analysing time-series data of a smart meter), in our work the $t_j \in T$ are associated with projects P_j, $j = 1, \ldots, \lambda$, to be funded in a particular investment round.

Security of Private Stream Aggregation. Our model allows an attacker to compromise investors. It can obtain auxiliary information about the values of investors or their secret keys. Even then a secure PSA scheme should release no more information than the aggregates of the uncompromised investors' values.

Definition 2 (Aggregator Obliviousness [23]) *Let* κ *be a security parameter. Let* $\mathcal{T}$ *be a ppt adversary for a PSA scheme* $\Sigma = (\mathsf{Setup}, \mathsf{PSAEnc}, \mathsf{PSADec})$ *and let* $\mathcal{D}$ *be a set. We define a security game between a challenger and the adversary* $\mathcal{T}$*.*

Setup. *The challenger runs the* Setup *algorithm on input security parameter* κ *and returns public parameters* pp*, public encryption parameters* T *with* $|T| = \lambda = poly(\kappa)$ *and secret keys* $s_0, s_1, \ldots, s_n$*. It sends* $\kappa, \mathsf{pp}, T, s_0$ *to* $\mathcal{T}$*.*

Queries. $\mathcal{T}$ *is allowed to query* $(i, t_j, x_{i,j})$ *with* $i \in \{1, \ldots, n\}, t_j \in T, x_{i,j} \in \mathcal{D}$ *and the challenger returns* $c_{i,j} \leftarrow \mathsf{PSAEnc}_{s_i}(t_j, x_{i,j})$*. Moreover,* $\mathcal{T}$ *is allowed to make compromise queries* $i \in \{1, \ldots, n\}$ *and the challenger returns* s_i*.*

Challenge. $\mathcal{T}$ *chooses* $U \subseteq \{1, \ldots, n\}$ *such that no compromise query for* $i \in U$ *was made and sends* U *to the challenger.* $\mathcal{T}$ *chooses* $t_{j*} \in T$ *such that no encryption query with* t_{j*} *was made. (If there is no such* t_{j*} *then the challenger simply aborts.)* $\mathcal{T}$ *queries two different tuples* $(x_{i,j*}^{[0]})_{i \in U}, (x_{i,j*}^{[1]})_{i \in U}$ *with*

$$\sum_{i \in U} x_{i,j*}^{[0]} = \sum_{i \in U} x_{i,j*}^{[1]}.$$

The challenger flips a random bit $b \leftarrow_R \{0, 1\}$*. For all* $i \in U$ *the challenger returns* $c_{i,j*} \leftarrow \mathsf{PSAEnc}_{s_i}(t_{j*}, x_{i,j*}^{[b]})$*.*

Queries. $\mathcal{T}$ *is allowed to make the same type of queries as before restricted to encryption queries with* $t_j \neq t_{j*}$ *and compromise queries for* $i \notin U$*.*

Guess. $\mathcal{T}$ *outputs a guess about* b*.*

The adversary wins the game if it correctly guesses b*. A PSA scheme achieves* Aggregator Obliviousness *(*AO*) or is* secure *if no ppt adversary* $\mathcal{T}$ *has more than negligible advantage (with respect to the parameter* κ*) in winning the above game.*

Encryption queries are made only for $i \in U$, since knowing the secret key for all $i \notin U$ the adversary can encrypt a value autonomously. If encryption queries in time-step t_{j*} were allowed, then no deterministic scheme would be secure. The adversary $\mathcal{T}$ can determine the original data of all $i \notin U$, since it knows $(s_i)_{i \notin U}$. Then $\mathcal{T}$ can compute the sum $\sum_{i \in U} x_{i,j} = \mathsf{PSADec}_{s_0}(t_j, c_{1,j}, \ldots, c_{n,j}) - \sum_{i \notin U} x_{i,j}$ of the uncompromised investors' values. If there is an investor's cipher which $\mathcal{T}$ does not receive, then it cannot compute the sum for the corresponding t_j.
It is also possible to define AO in the non-adaptive model as in [25,24]: here an adversary may not compromise investors adaptively, but has to specify the coalition U of compromised investors *before* making any query.

Feasibility of AO. In the random oracle model we can achieve AO for some constructions [23,2,25]. Because of its simplicity and efficient decryption, we use the PSA scheme proposed in [25] and present it in Figure 1. It achieves AO based on the DDH assumption.

PSA scheme

Setup: The public parameters are a primes $q > m \cdot n, p = 2q + 1$ and a hash function $H : T \to \mathcal{QR}_{p^2}$ modelled as a random oracle. The secret keys are $s_0, \ldots, s_n \leftarrow_R \mathbb{Z}_{pq}$ with $\sum_{i=0}^{n} s_i = 0 \bmod pq$.
PSAEnc: For $t_j \in T$ and $i = 1, \ldots, n$, encrypt $x_{i,j} \in [-m, m]$ by $c_{i,j} \leftarrow (1 + p \cdot x_{i,j}) \cdot H(t_j)^{s_i} \bmod p^2$.
PSADec: For $t_j \in T$ and ciphers $c_{1,j}, \ldots, c_{n,j}$ compute $V_j \in \{1 - p \cdot mn, \ldots, 1 + p \cdot mn\}$ with

$$V_j \equiv H(t_j)^{s_0} \cdot \prod_{i=1}^{n} c_{i,j} \equiv \prod_{i=1}^{n} (1 + p \cdot x_{i,j}) \equiv 1 + p \cdot \sum_{i=1}^{n} x_{i,j} \bmod p^2$$

and compute $\sum_{i=1}^{n} x_{i,j} = (V_j - 1)/p$ over the integers (this holds if the $c_{i,j}$ are encryptions of the $x_{i,j}$).

Fig. 1: PSA scheme secure in the random oracle model.

The scheme proposed in [23] is similar to the one in Figure 1, but is inefficient, if the range of possible decryption outcomes in is super-polynomially large in the security parameter. The scheme in Figure 1 also achieves the non-adaptive version of AO in the standard model.

2.3 Commitment schemes

A Commitment scheme allows a party to publicly commit to a value such that the value cannot be changed after it has been committed to (binding) and the value itself stays hidden to other parties until the owner reveals it (hiding). For the basic definitions we refer to [4] and [6]. Here we just recall the Pedersen Commitment introduced in [19] (Figure 2), which is computationally binding under the dlog assumption and perfectly hiding. In Section 3 we will combine the Pedersen Commitment with the PSA scheme from Figure 1 for the construction of Investcoin and thereby consider the input data $x = x_{i,j}$ to the Commitment scheme as investment amounts from investor N_i to project P_j. An essential property for the construction of Investcoin is that the Pedersen Commitment contains a homomorphic commitment algorithm, i.e. $\mathsf{Com}_{pk}(x, r) * \mathsf{Com}_{pk}(x', r') = \mathsf{Com}_{pk}(x + x', r + r')$.

Commitment scheme

GenCom: $(\mathsf{pp}, pk) \leftarrow \mathsf{GenCom}(1^\kappa)$, with public parameters pp describing a cyclic group G of order q and public key $pk = (h_1, h_2)$ for two generators h_1, h_2 of G. (where $\mathrm{dlog}_{h_1}(h_2)$ is not known to the commiting party).

Com: For $x \in \mathbb{Z}_q$ choose $r \leftarrow_R \mathbb{Z}_q$ and compute

$$com = \mathsf{Com}_{pk}(x, r) = h_1^x \cdot h_2^r \in G.$$

Unv: For $pk = (h_1, h_2)$, commitment com and opening (x, r) it holds that

$$\mathsf{Unv}_{pk}(com, x, r) = 1 \Leftrightarrow h_1^x \cdot h_2^r = com.$$

Fig. 2: The Pedersen Commitment.

2.4 Range test

To allow the honest verifier to verify in advance, that the (possibly malicious) prover commits to a an integer x in a certain range, a Range test must be applied. Range tests were studied in [5,7,20,21]. For Investcoin, an interactive procedure can be applied. It is a combination of the Pedersen Commitment to the binary representation of x and the extended Schnorr proof of knowledge [22] (Figure 3) applied to proving knowledge of one out of two secrets as described by Cramer [10]. Its basic idea was described by Boudot [5].
By the interactive procedure from Figure 4, the prover shows to the verifier, that the commited value x lies in the interval $[0, 2^l - 1]$ without revealing anything else about x. For the security of the construction in Figure 3 we refer to [10], where a more general protocol was considered (particularly the special honest verifier zero-knowledge property is needed). We use the Fiat-Shamir heuristic [11] to make the Range test non-interactive.

Extended Schnorr Proof of Knowledge

Let G be a cyclic group of order q and h a generator of G. The prover wants to show to the verifier that she knows the discrete logarithms of either $R = h^r$ or $S = h^s$ without revealing its value and position. W.l.o.g. the prover knows r.

Com: The prover chooses random $v_2, w_2, z_1 \leftarrow_R \mathbb{Z}_q^*$ and sends $a_1 = h^{z_1}, a_2 = h^{w_2} \cdot S^{-v_2}$ to the verifier.
Chg: The verifier sends a random $v \leftarrow_R \mathbb{Z}_q^*$ to the prover.
Opn: The prover sends $v_1 = v - v_2, w_1 = z_1 + v_1 r, v_2$ and w_2 to the verifier.
Chk: The verifier verifies that $v = v_1 + v_2$ and that $h^{w_1} = a_1 R^{v_1}, h^{w_2} = a_2 S^{v_2}$.

Fig. 3: Schnorr Proof of Knowledge of one out of two secrets.

2.5 Secure Computation

Our security definition for Investcoin will be twofold. In the first part, we consider a honest-but-curious coalition consisting of the untrusted system administration together with its compromised investors and the group of honest investors. Here we refer to notions from Secure Multi-Party Computation (SMPC). In the second part, we identify reasonable malicious behaviour that an investor could possibly execute and show, how the system can be secured against such malicious investors.
In this Section we focus on defining security against the honest-but-curious coalition.

Range test

Gen: $(\mathsf{pp}, pk) \leftarrow \mathsf{Gen}(1^\kappa)$, with public parameters pp describing the cyclic group G of order q, a test range $[0, 2^l - 1] \subset G$ and public key $pk = (g, h)$ for generators g, h of G (the prover does not know $dlog_g(h)$).
Com: For $x = \sum_{k=0}^{l-1} x^{(k)} \cdot 2^k$ with $x^{(k)} \in \{0, 1\}$ for all $k = 0, \ldots, l-1$, the prover chooses random $r^{(k)}, v_2^{(k)}, w_2^{(k)}, z_1^{(k)} \leftarrow_R \mathbb{Z}_q^*, k = 0 \ldots, l-1$, computes $r \equiv \sum_{k=0}^{l-1} r^{(k)} \cdot 2^k \bmod q$ and sends

$$com = g^x \cdot h^r \bmod q,\ com^{(k)} = g^{x^{(k)}} \cdot h^{r^{(k)}} \bmod q$$
$$a_1^{(k)} = h^{z_1^{(k)}}$$
$$a_2^{(k)} = h^{w_2^{(k)}} \cdot (h^{r^{(k)}} g^{2x^{(k)}-1})^{-v_2^{(k)}}$$

for $k = 0, \ldots, l-1$ to the verifier.
Chg: The verifier verifies that $com \equiv \prod_{k=0}^{l-1} (com^{(k)})^{2^k} \bmod q$. The verifier sends random $v^{(k)} \leftarrow_R \mathbb{Z}_q^*$ for all $k = 0 \ldots, l-1$ to the prover.
Opn: The prover sends $v_1^{(k)} = v^{(k)} - v_2^{(k)}, w_1^{(k)} = z_1^{(k)} + v_1^{(k)} r^{(k)}, v_2^{(k)}$ and $w_2^{(k)}$ for all $k = 0 \ldots, l-1$ to the verifier.
Chk: For all $k = 0 \ldots, l-1$, the verifier verifies that $v^{(k)} = v_1^{(k)} + v_2^{(k)}$ and that either
$(h^{w_1^{(k)}}, h^{w_2^{(k)}}) = (a_1^{(k)} (com^{(k)})^{v_1^{(k)}}, a_2^{(k)} (com^{(k)} \cdot g^{-1})^{v_2^{(k)}})$ or
$(h^{w_1^{(k)}}, h^{w_2^{(k)}}) = (a_1^{(k)} (com^{(k)} \cdot g^{-1})^{v_1^{(k)}}, a_2^{(k)} (com^{(k)})^{v_2^{(k)}})$.

Fig. 4: Range test for a commited value.

Definition 3 *Let κ be a security parameter and $n, \lambda \in \mathbb{N}$ with $n = poly(\kappa)$. Let ρ be a protocol executed by a group of size $u \leqslant n$ and a coalition of honest-but-curious adversaries of size $n - u + 1$ for computing the deterministic functionality f_ρ. The protocol ρ performs a* secure computation *(or* securely computes *f_ρ), if there exists a ppt algorithm $\mathcal{S}$, such that $\{\mathcal{S}(1^\kappa, y, f_\rho(x, y))\}_{x,y,\kappa} \approx_c \{\mathsf{view}^\rho(x, y, \kappa)\}_{x,y,\kappa}$, where $\mathsf{view}^\rho(x, y, \kappa) = (y, r, m)$ is the view of the coalition during the execution of the protocol ρ on input (x, y), x is the input of the group, y is the input of the coalition, r is its random tape and m is its vector of received messages.*

This definition follows standard notions from SMPC (as in [13]) and is adapted to our environment: first, we consider two-party protocols where each party consists of multiple individuals (each individual in a party has the same goals) and second, we do not consider security of the coalition against the group, since the system administration has no input and thus its security against honest-but-curious investors is trivial. Rather we will later consider its security against malicious investors.
Since Investcoin is the combination of various protocols, we will prove the security of these protocols separately and then use the composition theorem by Canetti [8].

Theorem 1 (Composition Theorem in the Honest-but-Curious Model [8]) *Let κ be a security parameter and let $m = poly(\kappa)$. Let π be a protocol that computes a functionality f_π by making calls to a trusted party computing the functionalities $f_1, \ldots, f_m$. Let $\rho_1, \ldots, \rho_m$ be protocols computing the functionalities $f_1, \ldots, f_m$ respectively. Denote by $\pi^{\rho_1, \ldots, \rho_m}$ the protocol π, where the calls to a trusted party are replaced by executions of $\rho_1, \ldots, \rho_m$. If $\pi, \rho_1, \ldots, \rho_m$ non-adaptively perform secure computations, then also $\pi^{\rho_1, \ldots, \rho_m}$ non-adaptively performs a secure computation.*

3 Investcoin

In this section, we introduce Investcoin. This protocol is build from a combination of the PSA scheme from Figure 1, the homomorphic Commitment scheme from Figure

2 and the Range test from Figure 4. Moreover, we provide a simple key-generation protocol for the Investcoin protocol that allows the dynamic join and leave of investors and is fault-tolerant towards investors.

3.1 Construction of Investcoin

The DIESet algorithm in Figure 5 executes the Setup algorithms of the underlying schemes. Additionally, DIESet generates a verification parameter β_j for each project P_j (and an additional β_0 - this will be used for the security against malicious investors) which is only known to the system administration. In Section 3.2 we provide a simple protocol for generating the secrets. The encryption algorithm DIEEnc executes the encryption algorithm of Σ and encrypts the amounts invested by N_i into P_j. In order to prove that $C_i = \sum_{j=1}^{\lambda} x_{i,j}$, the N_i execute the commitment algorithm DIECom commiting to the amounts $x_{i,j}$ invested using the randomness $r_{i,j}$ by executing the commitment algorithm of Γ and encrypting the $r_{i,j}$ with Σ. The Range test algorithm DIETes ensures that the investments are larger or equal 0. The payment verification algorithm DIEUnvPay first verifies that the combination of the commited amounts *in the correct order* is valid for the same combination of amounts encrypted *in the correct order* by executing the verification algorithm of Γ. If the investor has not cheated, this verification will output 1 by the homomorphy of Γ and the fact that $\prod_{j=0}^{\lambda} H(t_j)^{\beta_j} = \prod_{j=1}^{\lambda} H(\tilde{t}_j)^{\beta_j} = 1$. The DIEUnvPay algorithm verifies that the combination of commitments is valid for the aggregate C_i of the investments of N_i. The decryption algorithm DIEDec then decrypts the aggregated amounts for every project by executing the decryption algorithm of Σ. After the projects are realised, each investor N_i should receive back a multiple $\alpha_j x_{i,j}$ of her amount invested in each project P_j (e.g. a ROI). The factor α_j is publicly released by the management of project P_j and denotes a rate of return, interest or similar. This value is equal for every investor, since only the investor's stake should determine how much her profit from that project is. If the first check in the DIEUnvPay algorithm has output 1, the return verification algorithm DIEUnvRet verifies that the combination of commitments and return factors is valid for the claimed aggregate E_i of the returns to receive by N_i.
We emphasize the low communication effort needed after the DIESet algorithm: every investor sends the messages for DIEEnc, DIECom, DIETes and DIEUnvPay in one shot to the system, later only the messages for DIEUnvRet have to be sent. Thus, there are only two rounds of communication between the investors and the system.

Theorem 2 (Correctness of Investcoin) *Let Ω be the protocol in Figure 5. Then the following properties hold.*

1. *For all $j = 1, \ldots, \lambda$ and $x_{1,j}, \ldots, x_{n,j} \in [0, m]$:*

$$\mathsf{DIEDec}_{sk_0}(t_j, \mathsf{DIEEnc}_{sk_1}(t_j, x_{1,j}), \ldots, \mathsf{DIEEnc}_{sk_n}(t_j, x_{n,j})) = \sum_{i=1}^{n} x_{i,j}.$$

2. *For all $i = 1, \ldots, n$ and $x_{i,1}, \ldots, x_{i,\lambda} \in [0, m]$:*

$$\mathsf{DIEUnvPay}_{pk}\left(\prod_{j=1}^{\lambda} com_{i,j}, \sum_{j=1}^{\lambda} x_{i,j}, \sum_{j=1}^{\lambda} r_{i,j}\right) = 1$$
$$\Leftrightarrow \exists\, (\tilde{c}_{i,1}, \ldots, \tilde{c}_{i,\lambda}) \, : \, (com_{i,j}, \tilde{c}_{i,j}) \leftarrow \mathsf{DIECom}_{pk,sk_i}(x_{i,j}, r_{i,j}) \,\forall\, j = 1, \ldots, \lambda.$$

Investcoin

Let κ be a security parameter and $n, \lambda \in \mathbb{N}$ with $n = \mathrm{poly}(\kappa)$ and $\lambda = \mathrm{poly}(\kappa)$. Let $\Sigma = (\mathsf{Setup}, \mathsf{PSAEnc}, \mathsf{PSADec})$ be the PSA scheme from Figure 1 and let $\Gamma = (\mathsf{GenCom}, \mathsf{Com}, \mathsf{Unv})$ be the Commitment scheme from Figure 2. We define Investcoin $\Omega = (\mathsf{DIESet}, \mathsf{DIEEnc}, \mathsf{DIECom}, \mathsf{DIETes}, \mathsf{DIEUnvPay}, \mathsf{DIEDec}, \mathsf{DIEUnvRet})$ as follows.

DIESet:
- The system generates public parameters $\mathsf{pp} = \{q, p, H\}$ with primes $q > m \cdot n, p = 2q + 1$ (where $m = 2^l - 1$ is the maximum possible amount to invest into a single project by a single investor), parameters $T = \{t_0, t_1, \tilde{t}_1, \dots, t_\lambda, \tilde{t}_\lambda\}$ and a hash function $H : T \to \mathcal{QR}_{p^2}$.
- The system and the investors together generate secret keys s_0 and $sk_i = (s_i, \tilde{s}_i) \leftarrow_R \mathbb{Z}_{pq} \times \mathbb{Z}_{pq}$ for all $i = 1, \dots, n$ with $s_0 = -\sum_{i=1}^{n} s_i \pmod{pq}$.
- The system generates secret parameters $\beta_0, \dots, \beta_\lambda \leftarrow_R [-q', q'], q' < q/(m\lambda)$, such that $\prod_{j=0}^{\lambda} H(t_j)^{\beta_j} = \prod_{j=1}^{\lambda} H(\tilde{t}_j)^{\beta_j} = 1$ (see Section 3.2 for the details). It sets $sk_0 = (s_0, \beta_0, \dots, \beta_\lambda)$.
- The system generates a public key $pk = (h_1, h_2) \in \mathbb{Z}^*_{p^2} \times \mathbb{Z}^*_{p^2}$ with $\mathrm{ord}(h_1) = \mathrm{ord}(h_2) = pq$.

DIEEnc: For all $j = 1, \dots, \lambda$, each investor N_i chooses $x_{i,j} \in [0, m]$ and $x_{i,0} = 0$ and for all $j = 0, \dots, \lambda$ N_i sends the following ciphers to the system:

$$c_{i,j} \leftarrow \mathsf{DIEEnc}_{sk_i}(t_j, x_{i,j}) = \mathsf{PSAEnc}_{s_i}(t_j, x_{i,j}).$$

DIECom: For all $j = 1, \dots, \lambda$, each N_i chooses $r_{i,j} \leftarrow_R [0, m]$ and sends the following to the system:

$$(com_{i,j}, \tilde{c}_{i,j}) \leftarrow \mathsf{DIECom}_{pk, sk_i}(x_{i,j}, r_{i,j}) = (\mathsf{Com}_{pk}(x_{i,j}, r_{i,j}), \mathsf{PSAEnc}_{\tilde{s}_i}(\tilde{t}_j, r_{i,j})).$$

DIETes: For all $j = 1, \dots, \lambda$ the algorithm from Figure 4 on $(x_{i,j}, r_{i,j})$ is executed between the system (verifier) and each investor N_i (prover) using pk. Note that there always exists a random representation $r_{i,j}^{(0)}, \dots, r_{i,j}^{(l-1)} \leftarrow_R [0, m]$, such that $r_{i,j} \equiv \sum_{k=0}^{l-1} r_{i,j}^{(k)} \cdot 2^k \bmod p^2$.

DIEUnvPay: The system verifies for each investor N_i with commitment values $(com_{i,1}, \tilde{c}_{i,1}), \dots, (com_{i,\lambda}, \tilde{c}_{i,\lambda})$ and ciphers $c_{i,1}, \dots, c_{i,\lambda}$ that

$$\mathsf{Unv}_{pk}\left(\prod_{j=1}^{\lambda} com_{i,j}^{\beta_j}, A_i, B_i\right) = 1,$$

where $A_i = \left(\left(\prod_{j=0}^{\lambda} c_{i,j}^{\beta_j} \bmod p^2\right) - 1\right)/p$ and $B_i = \left(\left(\prod_{j=1}^{\lambda} \tilde{c}_{i,j}^{\beta_j} \bmod p^2\right) - 1\right)/p$. Then each investor N_i sends $C_i = \sum_{j=0}^{\lambda} x_{i,j}$ and $D_i = \sum_{j=1}^{\lambda} r_{i,j}$ to the system which computes

$$b_{P,i} = \mathsf{DIEUnvPay}_{pk}\left(\prod_{j=1}^{\lambda} com_{i,j}, C_i, D_i\right) = \mathsf{Unv}_{pk}\left(\prod_{j=1}^{\lambda} com_{i,j}, C_i, D_i\right), b_{P,i} \in \{0, 1\}.$$

and charges the bank account of N_i with the amount C_i, if $b_{P,i} = 1$.

DIEDec: For all $j = 0, \dots, \lambda$ and ciphertexts $c_{1,j}, \dots, c_{n,j}$, the system computes

$$X_j = \mathsf{DIEDec}_{sk_0}(t_j, c_{1,j}, \dots, c_{n,j}) = \mathsf{PSADec}_{s_0}(t_j, c_{1,j}, \dots, c_{n,j})$$

and verifies that $X_0 = 0$.

DIEUnvRet: The management of each project P_j generates a public return factor $\alpha_j \in [-q', q']$. The system charges the bank account of the management of each project P_j with the amount $\alpha_j X_j$. Each investor sends $E_i = \sum_{j=1}^{\lambda} \alpha_j x_{i,j}$ and $F_i = \sum_{j=1}^{\lambda} \alpha_j r_{i,j}$ to the system. If the verification in the $\mathsf{DIEUnvPay}$ algorithm has output 1, the system computes

$$b_{R,i} = \mathsf{DIEUnvRet}_{pk}\left(\prod_{j=1}^{\lambda} com_{i,j}^{\alpha_j}, E_i, F_i\right) = \mathsf{Unv}_{pk}\left(\prod_{j=1}^{\lambda} com_{i,j}^{\alpha_j}, E_i, F_i\right), b_{R,i} \in \{0, 1\}$$

and transfers the amount E_i to the bank account of investor N_i, if $b_{R,i} = 1$.

Fig. 5: The Investcoin protocol.

3. For all $i = 1, \dots, n$, public return factors $\alpha_1, \dots, \alpha_\lambda$ and $x_{i,1}, \dots, x_{i,\lambda} \in [0, m]$:

$$\begin{aligned}&\mathsf{DIEUnvRet}_{pk}\left(\prod_{j=1}^{\lambda} com_{i,j}^{\alpha_j}, \sum_{j=1}^{\lambda} \alpha_j x_{i,j}, \sum_{j=1}^{\lambda} \alpha_j r_{i,j}\right) = 1 \\ \Leftrightarrow &\exists\, (\tilde{c}_{i,1}, \dots, \tilde{c}_{i,\lambda}) \,:\, (com_{i,j}, \tilde{c}_{i,j}) \leftarrow \mathsf{DIECom}_{pk,sk_i}(x_{i,j}, r_{i,j}) \,\forall\, j = 1, \dots, \lambda.\end{aligned}$$

Proof. The first correctness property is given by the correctness of the PSA scheme from Figure 1. The second and third correctness properties are given by the correctness and the homomorphy of the Commitment scheme from Figure 2. □

By the first property, the decryption of all ciphers results in the sum of the amounts they encrypt. So the projects receive the correct investments. By the second property, the total investment amount of each investor is accepted by the system if the investor has commited to it. Thus, the investor's account will be charged with the correct amount. By the third property, the total return to each investor is accepted by the system if the investor has commited to the corresponding investment amount before. Thus, the investor will receive the correct return on investment (ROI).

3.2 Generation of public parameters and secret key generation protocol

In this section, we show how the system sets the random oracle $H : T \to \mathcal{QR}_{p^2}$ and we provide a decentralised key generation protocol for Investcoin. It supports dynamic join, dynamic leave and fault-tolarance of investors using one round of communication between the investors. In the Section 4.2, we will show how to use the public parameters and the secret key generation protocol for the security of Investcoin.

Setting the random oracle. Recall that we need to generate public parameters, a random oracle $H : T \to \mathcal{QR}_{p^2}$ and secret parameters $\beta_0, \dots, \beta_\lambda \leftarrow_R [-q', q']$, $q' < q/(m\lambda)$, such that for $t_0, \dots, t_\lambda, \tilde{t}_1, \dots, \tilde{t}_\lambda \in T$ the following equation holds.

$$\prod_{j=0}^{\lambda} H(t_j)^{\beta_j} = \prod_{j=1}^{\lambda} H(\tilde{t}_j)^{\beta_j} = 1. \tag{1}$$

First, for $j = 0, \dots, \lambda - 2$, on input t_j let $H(t_j)$ be random in $\mathcal{QR}_{p^2}$ and for $j = 1, \dots, \lambda - 2$, on input $\tilde{t}_j$ let $H(\tilde{t}_j)$ be random in $\mathcal{QR}_{p^2}$. The system chooses $\beta_0, \dots, \beta_{\lambda-2} \leftarrow_R [-q', q'], \beta_{\lambda-1} \leftarrow_R [-q', q' - 1]$ as part of its secret key (note that choosing these values according to a different distribution gives no advantage to the system). Then it computes

$$\begin{aligned}(H(t_{\lambda-1}), H(\tilde{t}_{\lambda-1})) &= \left(\prod_{j=0}^{\lambda-2} H(t_j)^{\beta_j}, \prod_{j=1}^{\lambda-2} H(\tilde{t}_j)^{\beta_j}\right), \\ (H(t_\lambda), H(\tilde{t}_\lambda), \beta_\lambda) &= (H(t_{\lambda-1}), H(\tilde{t}_{\lambda-1}), -1 - \beta_{\lambda-1}),\end{aligned}$$

instructs each investor N_i to set $x_{i,\lambda-1} = x_{i,\lambda} = 0$ and sets $\alpha_{\lambda-1} = \alpha_\lambda = 1$. In this way Equation (1) is satisfied. The projects $P_{\lambda-1}, P_\lambda$ deteriorate to 'dummy-projects' (e.g. if any investor decides to set $x_{i,\lambda} > 0$, then the system simply collects $X_\lambda > 0$).

Key generation for PSA within Investcoin. The building block for a key generation protocol is a $n-1$ out of n secret sharing scheme between the investors and the system. It is executed before the first investment round in the following way.
For all $i = 1, \ldots, n$, investor N_i generates uniformly random values $s_{i,1}, \ldots, s_{i,n}$ from the key space and sends $s_{i,i'}$ to $N_{i'}$ for all $i' = 1, \ldots, n$ via secure channel. Accordingly, each investor $N_{i'}$ obtains the shares $s_{1,i'}, \ldots, s_{n,i'}$. Then each investor N_i sets the own secret key $s_i = \sum_{i'=1}^{n} s_{i,i'}$ and each investor $N_{i'}$ sends $\sum_{i=1}^{n} s_{i,i'}$ to the system. The system then computes

$$s_0 = -\sum_{i'=1}^{n} \left(\sum_{i=1}^{n} s_{i,i'} \right) = -\sum_{i=1}^{n} \left(\sum_{i'=1}^{n} s_{i,i'} \right) = -\sum_{i=1}^{n} s_i.$$

By the secret sharing property this is a secure key generation protocol in the sense that only N_i knows s_i for all $i = 1, \ldots, n$ and only the system knows s_0.
For key generation, each investor has to send one message to every other investor and one message to the system which makes n^2 messages for the total network.
As a drawback, note that the key of each single investor is controlled by the other investors together with the system: for example, if $N_1, \ldots, N_{n-1}$ (maliciously) send the shares $s_{1,n}, \ldots, s_{n-1,n}$ and $s_{n,1}, \ldots, s_{n,n-1}$ to the system, it can compute the entire key s_n of N_n.
Assume that before the start of an arbitrary investment round, new investors want to join the network or some investors want to leave the network or some investors fail to send the required ciphers. In order to be able to carry out the protocol execution, the network can make a key update that requires $O(n)$ messages (rather than $O(n^2)$ messages for a new key setup) using the established secret sharing scheme. Due to space limitations, we omit the (simple) details.

4 Security of Investcoin

4.1 Definition of security

The administration is honest-but-curious and may compromise investors to build a coalition for learning the values of uncompromised investors. Investors who are not in the coalition may try to execute the following (reasonable) malicious behaviour:

1. Use different values for $x_{i,j}$ in DIEEnc and DIECom in order to have a larger profit than allowed.
2. Invest negative amounts $x_{i,j} < 0$ in order to 'steal' funds from the projects.
3. Use different parameters than generated in the Setup-phase (i.e. send inconsistent or simply random messages) in order to distort the whole computation.

Definition 4 (Privacy-Preserving Investment system) *Let κ be a security parameter and $n, \lambda \in \mathbb{N}$ with $n = poly(\kappa)$ and $\lambda = poly(\kappa)$. A protocol $\Omega =$ (DIESet, DIEEnc, DIECom, DIETes, DIEUnvPay, DIEDec, DIEUnvRet) consisting of ppt algorithms and executed between a group of honest parties of size $u \leqslant n$ and a coalition of honest-but-curious adversaries of size $n - u + 1$ is a* Privacy-Preserving Investment *(*PPI*) system if the following properties hold.*

1. *Let f_Ω be the deterministic functonality computed by Ω. Then Ω performs a secure computation (according to Definition 3).*

2. *Ω provides* computational linkage, *i.e. for all $i = 1, \ldots, n$, $sk_i, pk, T = \{t_1, \ldots, t_\lambda\}$, $(x_{i,j})_{j=1,\ldots,\lambda}$, $(\alpha_j)_{j=1,\ldots,\lambda}$ and all ppt adversaries $\mathcal{T}$ the following holds:*

$$\Pr\left[c_{i,j} \leftarrow \mathsf{DIEEnc}_{sk_i}(t_j, x_{i,j})\,\forall\, j = 1, \ldots, \lambda \;\wedge\; \mathsf{DIEUnvPay}_{pk}\left(\prod_{j=1}^{\lambda} com_{i,j}, C_i, D_i\right) = 1 \right.$$
$$\left. \wedge \mathsf{DIEUnvRet}_{pk}\left(\prod_{j=1}^{\lambda} com_{i,j}^{\alpha_j}, E_i, F_i\right) = 1 \;\wedge\; \left(C_i \neq \sum_{j=1}^{\lambda} x_{i,j} \vee E_i \neq \sum_{j=1}^{\lambda} \alpha_j x_{i,j}\right)\right]$$
$$\leqslant \mathsf{neg}(\kappa)$$

The probability is over the choice of $(c_{i,j})_{j=1,\ldots,\lambda}$, $(com_{i,j})_{j=1,\ldots,\lambda}$, C_i, D_i, E_i, F_i.

3. *For all $i = 1, \ldots, n$ and $j = 1, \ldots, \lambda$: $\mathsf{DIETes}_{pk}(x_{i,j}) = 1$ iff $x_{i,j} \geqslant 0$.*
4. *For all $i = 1, \ldots, n$ there is a ppt distinguisher $\mathcal{D}_i$, s.t. for $c_{i,j} \leftarrow \mathsf{DIEEnc}_{sk_i}(t_j, x_{i,j})$, $(com_{i,j}, \tilde{c}_{i,j}) \leftarrow \mathsf{DIECom}_{pk,sk_i}(x_{i,j}, r_{i,j})$ for a $r_{i,j} \leftarrow_R [0, m]$, where $j = 1, \ldots, \lambda$, for $c^*_{i,j*} \leftarrow \mathsf{DIEEnc}_{sk_i^*}(t^*_{j*}, x_{i,j*})$ and $(com^*_{i,j*}, \tilde{c}^*_{i,j*}) \leftarrow \mathsf{DIECom}_{pk,sk_i^*}(x_{i,j*}, r_{i,j*})$, where $j^* \in \{1, \ldots, \lambda\}$ and $(sk_i^*, t^*_{j*}) \neq (sk_i, t_{j*})$, it holds that*

$$\Big|\Pr\left[\mathcal{D}_i\left(1^\kappa, c_{i,1}, com_{i,1}, \tilde{c}_{i,1}, \ldots, c_{i,j*}, com_{i,j*}, \tilde{c}_{i,j*}, \ldots, c_{i,\lambda}, com_{i,\lambda}, \tilde{c}_{i,\lambda}\right) = 1\right]$$
$$- \Pr\left[\mathcal{D}_i\left(1^\kappa, c_{i,1}, com_{i,1}, \tilde{c}_{i,1}, \ldots, c^*_{i,j*}, com^*_{i,j*}, \tilde{c}^*_{i,j*}, \ldots, c_{i,\lambda}, com_{i,\lambda}, \tilde{c}_{i,\lambda}\right) = 1\right]\Big|$$
$$\geqslant 1 - \mathsf{neg}(\kappa).$$

The definition is twofold: on the one hand it covers the security of honest investors against a honest-but-curious coalition consisting of the untrusted system administration and compromised investors (Property 1) and on the other hand it covers the security of the system against maliciously behaving investors (Properties $2, 3, 4$). Note that we have to distinguish between these two requirements, since we assume different behaviours for the two groups of participants, i.e. we cannot simply give a real-world-ideal-world security proof as in the SMPC literature in the malicious model. Instead, we follow the notions of the SMPC literature [13] for the security of honest investors (only) in the honest-but-curious model and additionally provide security notions against some reasonably assumable behaviour of malicious investors. For the security against a honest-but-curious coalition, the first property ensures that from the outcome of the decryption no other information than X_j can be detected for all $j = 1, \ldots, \lambda$ and that the single amounts comitted to by the investors for payment and return are hidden. For the security of the system against maliciously behaving investors, imagine the situation where an investor N_i claims to having payed amount $x_{i,j}$ to project P_j (in the DIECom algorithm) but in fact has only payed $\tilde{x}_{i,j} < x_{i,j}$ (in the DIEEnc algorithm). If the return factor α_j is larger than 1, then N_i would unjustly profit more from her investment than she actually should and the system would have a financial damage.[1] Therefore the second property says that for all $i = 1, \ldots, n$ with overwhelming probability, whenever $x_{i,1}, \ldots, x_{i,\lambda}$ were send by N_i using the DIEEnc algorithm and DIEUnvPay, DIEUnvRet accept C_i, E_i respectively, then C_i and E_i must be the legitimate amounts that N_i respectively has invested in total and has to get back as return in total. The third property ensures that no investor is able to perform a negative investment. The fourth property ensures that all investors use the correct parameters as generated by the DIESet algorithm.[2]

[1] Usually the investor cannot know if $\alpha_j > 1$ at the time of cheating, since it becomes public in the future. However, in the scenario where a priori information about α_j is known to some investors or where investors simply act maliciously, we need to protect the system from beeing cheated.

[2] More precisely, it ensures that a cheating investor will be identified by the system.

4.2 Proof of security

First, we concentrate on the security against the honest-but-curious coalition (Property 1) and then show security against malicious investors (Propertis $2, 3, 4$).

Theorem 3 *By the DDH assumption in the group $\mathcal{QR}_{p^2}$ of quadratic residues modulo p^2 for a safe prime p, Investcoin is a* PPI *system in the random oracle model.*

Proof. The proof follows from Lemma 7, Lemma 8, Lemma 9 and Lemma 10 below. □

Security against honest-but-curious coalition. Investcoin is the combination of the protocols described in Figures $1, 2, 3$.[3] We first show the security of these protocols seperately and then use Theorem 1 in order to show composition. In this section, assume w.l.o.g. that the indices $i = 1, \ldots, u$ belong to the group of honest investors and the indices $i = u+1, \ldots, n$ belong to the compromised investors. Before showing security, we briefly explain the functionalities used in the proofs. The functionality f_Σ is computed by the protocol in Figure 1. It takes as input n values $x_1, \ldots, x_n$ and outputs their sum $f_\Sigma(x_1, \ldots, x_n) \equiv \sum_{i=1}^n x_i$. The functionality f_Γ is computed by the protocol in Figure 2. It takes as input three values com, x, r and outputs 1 iff (x, r) is a valid opening for the commitment com. The functionality f_Υ is computed by the protocol in Figure 3. It takes as input two values r, R and outputs 1 iff r is a discrete logarithm of either R or $S = R \cdot g^{-1}$ with base h.

Lemma 4 *Let Σ be the protocol in Figure* 1. *By the DDH assumption, Σ securely computes f_Σ in the random oracle model.*

Proof. Assume the coalition is corrupted. Let $\{x_{i,j} : i = 1, \ldots, n, j = 1, \ldots, \lambda\}$ be the input messages of all investors. We construct a simulator $\mathcal{S}$ for the

$$\mathsf{view}^\Sigma((x_{i,j})_{i=1,\ldots,n,j=1,\ldots,\lambda}, \kappa) = ((x_{i,j})_{i=u+1,\ldots,n,j=1,\ldots,\lambda}, \perp, (c_{i,j})_{i=1,\ldots,u,j=1,\ldots,\lambda})$$

of the coaltion, where $c_{i,j} = \mathsf{PSAEnc}_{s_i}(t_j, x_{i,j})$ for all i, j and $\mathsf{PSAEnc}_{s_0}(t_j, 0) \cdot \prod_i c_{i,j} = 1 + p \cdot f_\Sigma((x_{i,j})_{i=1,\ldots,n}) \bmod p^2$ for all j, as follows.[4]

$\mathcal{S}$ takes as input 1^κ, messages $(x_{i,j})_{i=u+1,\ldots,n,j=1,\ldots,\lambda}$ and the protocol outputs $(X_j)_{j=1,\ldots,\lambda} = (f_\Sigma((x_{i,j})_{i=1,\ldots,n}))_{j=1,\ldots,\lambda}$.

1. Choose messages $(x'_{i,j})_{i=1,\ldots,u,j=1,\ldots,\lambda}$ with each $x'_{i,j} \in [0, 2^l - 1]$, such that $\sum_{i=1}^u x'_{i,j} = X_j - \sum_{i=u+1}^n x_{i,j}$ for all $j = 1, \ldots, \lambda$.
2. Compute $c'_{i,j} = \mathsf{PSAEnc}_{s_i}(t_j, x'_{i,j})$ for $i = 1, \ldots, u$ and $c_{i,j} = \mathsf{PSAEnc}_{s_i}(t_j, x_{i,j})$ for $i = u+1, \ldots, n$, $j = 1, \ldots, \lambda$ for the keys $s_1, \ldots, s_n \leftarrow_R \mathbb{Z}_{pq}$ with $s_0 = -\sum_{i=1}^n s_i \bmod pq$.

$\mathcal{S}$ outputs $(x_{i,j})_{i=u+1,\ldots,n,j=1,\ldots,\lambda}, \perp, (c'_{i,j})_{i=1,\ldots,u,j=1,\ldots,\lambda}$.

The computational indistinguishability of the output distribution of $\mathcal{S}$ from $\left\{\mathsf{view}^\Sigma((x_{i,j})_{i=1,\ldots,n,j=1,\ldots,\lambda}, \kappa)\right\}_{(x_{i,j})_{i=1,\ldots,n,j=1,\ldots,\lambda},\kappa}$ immediately follows from the AO security of Σ, which holds by the DDH assumption. □

Lemma 5 *The protocol Γ in Figure* 2 *securely computes f_Γ.*

[3] Note that the Range test in Figure 4 is the combination of the protocols in Figure 2 and 3. Therefore it suffices to show the security of these protocols.

[4] Note that there is no internal randomness of the coalition. Instead the common public randomness from the random oracle H is used.

Proof. Assume the honest verifier is corrupted. The simulator $\mathcal{S}$ for the $\mathsf{view}^{\Gamma}(x, \perp, \kappa) = (\perp, com)$ of the verifier can be constructed by choosing a message x', randomness r' and outputting the corresponding commitment com'. By the perfect hiding property of Γ, the output distribution is indistinguishable from $\{\mathsf{view}^{\Gamma}(x, \perp, \kappa)\}_{x,\kappa}$. □

Lemma 6 *The protocol Υ in Figure 3 securely computes f_{Υ} by the dlog assumption.*

Proof. Assume the honest verifier is corrupted and the prover knows the discrete logarithm of either R or $S = R \cdot g^{-1}$ base h. We construct a simulator $\mathcal{S}$ for the

$$\mathsf{view}^{\Upsilon}(t, R, S, \kappa) = (R, S, v, a_1, a_2, (v_1, w_1), (v_2, w_2))$$

of the verifier, such that $v = v_1 + v_2$ and $h^{w_1} = a_1 R^{v_1}, h^{w_2} = a_2 S^{v_2}$, as follows.

$\mathcal{S}$ takes as input 1^{κ}, the verifier's input (R, S) and the protocol output $b_T \in \{0, 1\}$.
1. Choose random values $v'_1, v'_2, w'_1, w'_2 \leftarrow_R \mathbb{Z}_q^*$.
2. Set $v' = v'_1 + v'_2$.
3. Set $a'_1 = h^{w'_1} R^{-v'_1}$ and $a'_2 = h^{w'_2} S^{-v'_2}$.

$\mathcal{S}$ outputs $R, S, v', a'_1, a'_2, (v'_1, w'_1), (v'_2, w'_2)$.

The value v' is random, since v'_1, v'_2 are random. Then the distribution of the output of $\mathcal{S}$ is computationally indistinguishable from $\{\mathsf{view}^{\Upsilon}(t, R, S, \kappa)\}_{t,R,S,\kappa}$ by the special honest verifier zero-knowledge property of Υ, which holds by the dlog assumption. □

Lemma 7 *Investcoin satisfies Property 1 of Definition 4 in the random oracle model by the DDH assumption.*

Proof. Let $\Omega^{f_\Sigma, f_\Gamma, f_\Upsilon}$ be the protocol Ω, where executions of Σ, Γ, Υ are replaced by calls to a trusted party computing the according functionalities $f_\Sigma, f_\Gamma, f_\Upsilon$. We show that $\Omega^{f_\Sigma, f_\Gamma, f_\Upsilon}$ performs a secure computation. Assume the coalition is corrupted. Let $\{x_{i,j} : i = 1, \ldots, n, j = 1, \ldots, \lambda\}$ be the input messages of all investors. We construct a simulator $\mathcal{S}$ for the

$$\mathsf{view}^{\Omega^{f_\Sigma, f_\Gamma, f_\Upsilon}}((x_{i,j})_{i=1,\ldots,n,j=1,\ldots,\lambda}, \kappa)$$
$$=((x_{i,j})_{i=u+1,\ldots,n,j=1,\ldots,\lambda}, \perp, (com_{i,j})_{i=1,\ldots,u,j=1,\ldots,\lambda}, (com_{i,j}^{(k)})_{i=1,\ldots,u,j=1,\ldots,\lambda,k=0,\ldots,l-1})$$

of the coaltion, where $com_{i,j}$ is a commitment to $x_{i,j}$ and $com_{i,j} = \prod_{k=0}^{l-1} (com_{i,j}^{(k)})^{2^k}$ for all i, j, as follows.[5]

$\mathcal{S}$ takes as input 1^{κ}, messages $x_{i,j}$ for $i = u+1, \ldots, n, j = 1, \ldots, \lambda$ and the protocol outputs $X_j = f_\Sigma((x_{i,j})_{i=1,\ldots,n})$ for $j = 1, \ldots, \lambda$, $C_i = \sum_{j=1}^{\lambda} x_{i,j}$, $E_i = \sum_{j=1}^{\lambda} \alpha_j x_{i,j}$ for $i = 1, \ldots, n$.

1. For $i = 1, \ldots, u, j = 1, \ldots, \lambda$ choose messages $x'_{i,j}$ with $x'_{i,j} \in [0, 2^l - 1]$, such that $\sum_{i=1}^{u} x'_{i,j} = X_j - \sum_{i=u+1}^{n} x_{i,j}$ for all $j = 1, \ldots, \lambda$, $\sum_{j=1}^{\lambda} x'_{i,j} = C_i$ and $\sum_{j=1}^{\lambda} \alpha_j x'_{i,j} = E_i$ for all $i = 1, \ldots, u$.[6]

[5] Since we rely on a trusted party to compute $f_\Sigma, f_\Gamma, f_\Upsilon$ there is no communication between the group and the coalition. The system administration, which is part of the coalition, and the investors simply feed the trusted party with their according inputs, outputs and received messages to compute the respective functionalities.

[6] Since in the context of this lemma we do not care about the linkage property, this can be done efficiently (there are $u + \lambda$ equations and $u \cdot \lambda$ free variables).

2. Choose random values $r_{i,j} \leftarrow_R [0,m]$ and their random representation $(r_{i,j}^{(0)},\dots,r_{i,j}^{(l-1)})$ with $r_{i,j}=\sum_{k=0}^{l-1} r_{i,j}^{(k)}\cdot 2^k$ for all $i=1,\dots,u, j=1,\dots,\lambda$.
3. Compute $com'_{i,j}=\mathsf{Com}_{pk}(x'_{i,j},r_{i,j})$ for all $i=1,\dots,u, j=1,\dots,\lambda$.
4. Compute $com'^{(k)}_{i,j}=\mathsf{Com}_{pk}(x_{i,j}^{(k)},r_{i,j}^{(k)})$ for all $i=1,\dots,u, j=1,\dots,\lambda, k=0,\dots,l-1$, where $(x_{i,j}^{(0)},\dots,x_{i,j}^{(l-1)})$ is the binary representation of $x'_{i,j}$ for all i,j.

$\mathcal{S}$ outputs

$$(x_{i,j})_{i=u+1,\dots,n,j=1,\dots,\lambda},\bot,(com'_{i,j})_{i=1,\dots,u,j=1,\dots,\lambda},(com'^{(k)}_{i,j})_{i=1,\dots,u,j=1,\dots,\lambda,k=0,\dots,l-1}.$$

The output of $\mathcal{S}$ is indistinguishable from the view of the coalition by the perfect hiding property of Γ.[7] The statement of the lemma follows from Lemma 4, Lemma 5, Lemma 6 and Theorem 1. □

Security against malicious investors.

Lemma 8 *Investcoin satisfies Property* 2 *of Definition* 4.

Proof. Investcoin satisfies the linkage property of a PPI system by the binding-property of Γ, the choice of $H : T \to \mathcal{QR}_{p^2}$, $(\beta_0,\dots,\beta_\lambda)$ such that $\beta_0,\dots,\beta_\lambda \in [-q',q']$, $q' < q/(m\lambda)$, are pairwise different with $\prod_{j=0}^{\lambda} H(t_j)^{\beta_j}=1$ and $\prod_{j=1}^{\lambda} H(\tilde{t}_j)^{\beta_j}=1$ and the non-commutativity of the map $f_{\beta_0,\dots,\beta_\lambda}:[0,m]^\lambda \to [-q,q]$, $f_{\beta_0,\dots,\beta_\lambda}(x_0,\dots,x_\lambda)=\sum_{j=0}^{\lambda}\beta_j x_j$.
Note that $\beta_0,\dots,\beta_\lambda$ are not known to the investors. Therefore for all $i=1,\dots,n$ and $j=0,\dots,\lambda$, if any amount $x_{i,j}$ invested by N_i to P_j (P_0 can be seen as the system administration itself) deviates from the corresponding amount commited to, then with overwhelming probability this will be detected by the system during the execution of DIEUnvPay, i.e. it will be $\mathsf{Unv}_{pk}\left(\prod_{j=1}^{\lambda} com_{i,j}^{\beta_j},A_i,B_i\right)=0$. □

Note that even if N_i keeps $C_i=\sum_{j=0}^{\lambda}x_{i,j}$ but interchanges two amounts in the commitment phase, say x_{i,j_1},x_{i,j_2}, this will be still detectable, since

$$f_{\beta_0,\dots,\beta_\lambda}(x_{i,0},\dots,x_{i,j_1},\dots,x_{i,j_2},\dots,x_{i,\lambda}) \neq f_{\beta_0,\dots,\beta_\lambda}(x_{i,0},\dots,x_{i,j_2},\dots,x_{i,j_1},\dots,x_{i,\lambda})$$

by the non-commutativity of $f_{\beta_0,\dots,\beta_\lambda}$.

Lemma 9 *Investcoin satisfies Property* 3 *of Definition* 4.

Proof. The proof follows from the analysis in [10]. □

Imagine an investor who sends inconsistent messages in the DIEEnc algorithm (i.e. it uses different secret keys in the DIEEnc algorithm than generated in the DIESet algorithm) in order to maliciously distort the total amounts $X_1,\dots,X_\lambda$ invested in the single projects. We show how the described key generation protocol can be used for detecting such malicious investors within the Investcoin protocol.
First, in the DIESet algorithm, the network generates the additional public parameter t_0 and the secret $\beta_0 \in [-q',q']$ such that Equation (1) is satisfied. Then each investor $N_{i'}$ publishes $T_{i,i'}=\mathsf{PSAEnc}_{s_{i,i'}}(t_0,0)$ on a black board for all the key shares

[7] Note that only the execution of interaction and the verification algorithm Unv of Γ are replaced by the trusted party, not the commitment algorithm Com.

$s_{1,i'}, \ldots, s_{n,i'}$ received by $N_{i'}$ during the key generation. Then each N_i verifies that the other investors used the correct key shares $s_{i,1}, \ldots, s_{i,n}$ of her secret key s_i in the publications. Moreover, using the received values $\sum_{i=1}^{n} s_{i,i'}$ for all $i' = 1, \ldots, n$, the system verifies that

$$\mathsf{PSADec}_{\sum_{i=1}^{n} s_{i,i'}}(t_0, T_{1,i'}, \ldots, T_{n,i'}) = 0.$$

In this way, the system can be sure that only correct key shares were used to compute the published values $T_{i,i'}$.[8]
Now for all $i = 1, \ldots, n$, the system involves the encryption of $x_{i,0} = 0$ (i.e. the value encrypted using t_0) in order to compute the verification value A_i of N_i. Because of Equation (1) and the linkage property, N_i has to use the same secret key s_i (and the correct $H(t_j), H(\tilde{t}_j)$) for all $j = 0, 1, \ldots, \lambda$. On the other hand, as discussed above, the encryption of $x_{i,0} = 0$ is verifiably generated with the correct secret key shares from the DIESet algorithm. This means, s_i is the correct secret key of N_i and the system knows $\mathsf{PSAEnc}_{s_i}(t_0, 0)$. Therefore the use of a different key in the DIEEnc algorithm than generated in the DIESet algorithm is not possible without being detected by the system and thus malicious behaviour becomes prevented. Now we can prove the following lemma.

Lemma 10 *Investcoin satisfies Property* 4 *of Definition* 4.

Proof. For all $i = 1, \ldots, n$ we construct $\mathcal{D}_i$ as follows. On input

$$1^\kappa, c_{i,0}, c_{i,1}, com_{i,1}, \tilde{c}_{i,1}, \ldots, c, com, \tilde{c}, \ldots, c_{i,\lambda}, com_{i,\lambda}, \tilde{c}_{i,\lambda}$$

it has to decide whether $(c, com, \tilde{c}) = (c_{i,j*}, com_{i,j*}, \tilde{c}_{i,j*})$ or $(c, com, \tilde{c}) = (c^*_{i,j*}, com^*_{i,j*}, \tilde{c}^*_{i,j*})$. It uses its input to compute

$$A_i = \left(\left(\prod_{j=0}^{\lambda} c_{i,j}^{\beta_j} \bmod p^2\right) - 1\right)/p \text{ and } B_i = \left(\left(\prod_{j=1}^{\lambda} \tilde{c}_{i,j}^{\beta_j} \bmod p^2\right) - 1\right)/p.$$

Then it computes and outputs $b = \mathsf{Unv}_{pk}\left(com^{\beta_{j*}} \cdot \prod_{j \neq j*} com_{i,j}^{\beta_j}, A_i, B_i\right)$. The case $b = 1$ is interpreted as $(c, com, \tilde{c}) = (c_{i,j*}, com_{i,j*}, \tilde{c}_{i,j*})$, the case $b = 0$ is interpreted as $(c, com, \tilde{c}) = (c^*_{i,j*}, com^*_{i,j*}, \tilde{c}^*_{i,j*})$. □

References

1. Emmanuel A. Abbe, Amir E. Khandani, and Andrew W. Lo. Privacy-preserving methods for sharing financial risk exposures. *Am. Economic Review*, 102(3):65–70, 2012.
2. Fabrice Benhamouda, Marc Joye, and Benoît Libert. A new framework for privacy-preserving aggregation of time-series data. *ACM Transactions on Information and System Security*, 18(3), 2016.
3. Avrim Blum, Jamie Morgenstern, Ankit Sharma, and Adam Smith. Privacy-preserving public information for sequential games. In *Proc. of ITCS '15*, pages 173–180, 2015.
4. Manuel Blum. Coin flipping by telephone. In *Proc. of CRYPTO '81*, pages 11–15, 1981.
5. Fabrice Boudot. Efficient proofs that a committed number lies in an interval. In *Proc. of EUROCRYPT '00*, pages 431–444, 2000.

[8] If a published value was generated using an incorrect share, the system is able to detect the (coalition of) cheating investors.

6. Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *J. Comput. Syst. Sci.*, 37(2):156–189, 1988.
7. Jan Camenisch, Rafik Chaabouni, and Abhi Shelat. Efficient protocols for set membership and range proofs. In *Proc. of ASIACRYPT '08*, pages 234–252, 2008.
8. Ran Canetti. Security and Composition of Multiparty Cryptographic Protocols. *Journal of Cryptology*, 13:143–202, 2000.
9. Financial Crisis Inquiry Comission. *The Financial Crisis Inquiry Report: Final Report of the National Commission on the Causes of the Financial and Economic Crisis in the United States*, 2011.
10. Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Proc. of CRYPTO '94*, pages 174–187, 1994.
11. Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Proc. of CRYPTO '86*, pages 186–194, 1987.
12. Mark Flood, Jonathan Katz, Stephen Ong, and Adam Smith. Cryptography and the economics of supervisory information: Balancing transparency and confidentiality. *Federal Reserve Bank of Cleveland, Working Paper no. 13-11*, 2013.
13. Oded Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications.* Cambridge University Press, 2004.
14. Nicola Jentzsch. *The Economics and Regulation of Financial Privacy - A Comparative Analysis of the United States and Europe.* 2001.
15. Marc Joye and Benoît Libert. A scalable scheme for privacy-preserving aggregation of time-series data. In *Proc. of FC '13*, pages 111–125. 2013.
16. Ian Miers, Christina Garman, Matthew Green, and Aviel D. Rubin. Zerocoin: Anonymous distributed e-cash from bitcoin. In *Proc. of SP '13*, pages 397–411, 2013.
17. Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system.
18. Michael Nofer. *The Value of Social Media for Predicting Stock Returns - Preconditions, Instruments and Performance Analysis.* PhD thesis, Techn. Univ. Darmstadt, 2014.
19. Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Proc. of CRYPTO '91*, pages 129–140, 1991.
20. Kun Peng, Colin Boyd, Ed Dawson, and Eiji Okamoto. A novel range test. pages 247–258, 2006.
21. Kun Peng and Ed Dawson. A range test secure in the active adversary model. In *Proc. of ACSW '07*, pages 159–162, 2007.
22. Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In *Proc. of CRYPTO '89*, pages 239–252, 1989.
23. Elaine Shi, T.-H. Hubert Chan, Eleanor G. Rieffel, Richard Chow, and Dawn Song. Privacy-preserving aggregation of time-series data. In *Proc. of NDSS '11*, 2011.
24. Filipp Valovich. On the hardness of the learning with errors problem with a discrete reproducible error distribution. *CoRR abs/1605.02051*, 2016.
25. Filipp Valovich and Francesco Aldà. Private stream aggregation revisited. *CoRR abs/1507.08071*, 2015.

A Preservation of Market liquidity

We show that it is still possible to privately exchange any part of an investment between any pair of investors within Investcoin, i.e. market liquidity is unaffected. Assume that an investment round is over but the returns are not yet executed, i.e. the system already received $c_{i,j}, (com_{i,j}, \tilde{c}_{i,j}), C_i, D_i$ for all $i = 1, \ldots, n$ and $j = 1, \ldots, \lambda$ but not E_i, F_i. Assume further that for some $i, i' \in \{1, \ldots, n\}$, investors N_i and $N_{i'}$ confidentially agree on a transfer of amount $x_{(i,i'),j}$ (i.e. a part of N_i's investment in project P_j) from investor N_i to investor $N_{i'}$. This fact needs to be confirmed by the protocol in order to guarantee the correct returns from project P_j to investors N_i and $N_{i'}$. Therefore the commitments to the invested amounts $x_{i,j}$ and $x_{i',j}$ respectively

need to be updated.
For the update, N_i and $N_{i'}$ agree on a value $r_{(i,i'),j} \leftarrow_R [0,m]$ via secure channel. This value should be known only to N_i and $N_{i'}$. Then N_i and $N_{i'}$ respectively compute

$$(com'_{i,j}, \tilde{c}'_{i,j}) \leftarrow \mathsf{DIECom}_{pk,sk_i}(x_{(i,i'),j}, r_{(i,i'),j}),$$
$$(com'_{i',j}, \tilde{c}'_{i',j}) \leftarrow \mathsf{DIECom}_{pk,sk_{i'}}(x_{(i,i'),j}, r_{(i,i'),j})$$

and send their commitments to the system which verifies that $com'_{i,j} = com'_{i',j}$. Then the system updates $(com_{i,j}, \tilde{c}_{i,j})$ by $(com_{i,j} \cdot (com'_{i,j})^{-1}, \tilde{c}_{i,j} \cdot (\tilde{c}'_{i,j})^{-1})$ (which is possible since DIECom is injective) and $(com_{i',j}, \tilde{c}_{i',j})$ by $(com_{i',j} \cdot com'_{i',j}, \tilde{c}_{i',j} \cdot \tilde{c}'_{i',j})$. As desired, the updated values commit to $x_{i,j} - x_{(i,i'),j}$ and to $x_{i',j} + x_{(i,i'),j}$ respectively. Moreover, N_i updates the return values (E_i, F_i) by $(E_i - \alpha_j \cdot x_{(i,i'),j}, F_i - \alpha_j \cdot r_{(i,i'),j})$ and $N_{i'}$ updates $(E_{i'}, F_{i'})$ by $(E_{i'} + \alpha_j \cdot x_{(i,i'),j}, F_{i'} + \alpha_j \cdot r_{(i,i'),j})$.
The correctness of the update is guaranteed by Property 2 and the confidentiality of the transferred amount $x_{(i,i'),j}$ (i.e. only N_i and $N_{i'}$ know $x_{(i,i'),j}$) is guaranteed by Property 1 of Definition 4.
Note that in general this procedure allows a short sale for N_i when $x_{(i,i'),j} > x_{i,j}$ or for $N_{i'}$ when $x_{(i,i'),j} < 0$ and $|x_{(i,i'),j}| > x_{i',j}$ (over the integers). If this behaviour is not desired, it may also be necessary to perform a Range test for the updated commitments $com_{i,j} \cdot (com'_{i,j})^{-1}$ (between the system and N_i) and $com_{i',j} \cdot com'_{i',j}$ (between the system and $N_{i'}$) to ensure that they still commit to amounts $\geqslant 0$.