# Practical Witness-Key-Agreement for Blockchain-based Dark Pools Financial Trading

Chan Nam Ngo[1],[⋆], Fabio Massacci[1],[2], Florian Kerschbaum[3], and Julian Williams[4]

[1] University of Trento, IT, {channam.ngo,fabio.massacci}@unitn.it
[2] Vrije Universiteit Amsterdam, NL, fabio.massacci@ieee.org
[3] University of Waterloo, CA, florian.kerschbaum@uwaterloo.ca
[4] Durham Business School, UK, julian.williams@durham.ac.uk

**Abstract.** We introduce a new cryptographic scheme, Witness Key Agreement (WKA), that allows a party to securely agree on a secret key with a counter party holding publicly committed information only if the counter party also owns a secret witness in a desired (arithmetic) relation with the committed information.

Our motivating applications are over-the-counter (OTC) markets and dark pools, popular trading mechanisms. In such pools investors wish to communicate only to trading partners whose transaction conditions and asset holdings satisfy some constraints. The investor must establish a secure, authenticated channel with eligible traders where the latter committed information matches a desired relation. At the same time traders should be able to show eligibility while keeping their financial information secret.

We construct a WKA scheme for languages of statements proven in the designated-verifier Succinct Zero-Knowledge Non-Interactive Argument of Knowledge Proof System (zk-SNARK). We illustrate the practical feasibility of our construction with some arithmetic circuits of practical interest by using data from US Dollar denominated corporate securities traded on Bloomberg Tradebook.

**Keywords:** Blockchain-based dark pool; witness-key-agreement; zk-SNARK; quadratic arithmetic program; designated-verifier.

## 1 Introduction

**Existing Blockchain-based Financial Systems** Financial intermediation is traditionally based on trusted third party solutions, such as exchanges (e.g. NASDAQ or CME) or clearing mechanisms (e.g. EU's TARGET2-Securities and US's Depository Trust & Clearing Corporation).

New technologies have been recently proposed to replace these intermediaries with distributed protocols on blockchain. See for example ZeroCash [44], a cryptocurrency, or FuturesMEX [39], a crypto-based distributed futures exchange, or

[⋆] This research was conducted during the author's visit to the University of Waterloo.

the dark pool exchange with three parties [14]. In those systems, the users commit financial information (e.g. accounts, bids and quotes) to a blockchain and use zero-knowledge proofs to show that their committed information satisfy a certain relation to preserve the integrity of the market and the solvency of the users. Noticeably, anonymity in those systems is as critical as confidentiality, e.g. the linkage of one's transactions can lead to strategic attacks against them [38].

**New Dark Pools Requirements** Private markets, i.e. dark pools, further reduce public information to protect large investors. The investor in a dark pool, who wants to sell at least $v$ shares at price $p$, wants to disclose $v$ and $p$ only to traders who committed to have cash $c \geq pv$. Alternatively she might be willing to buy from somebody who has at least $v'$ shares (an iceberg quote) or accept a price pegged within an interval, etc. For the very same reasons, the trader might not want to make his information fully public, but just to reassure the investor that he meets the constraints.

To make distributed dark pools possible, we propose *Witness Key Agreement* (WKA). In presence of a public blockchain holding parties' publicly committed information, WKA allows a party (the Verifier) to post a problem relation (e.g. a desired arithmetic or boolean combination of secret information) and securely agree on a secret key with another party (the Prover) holding a secret witness that *both* corresponds to the publicly committed information *and* satisfies the desired relation (i.e. the implicitly defined problem instance of the relation between the commits and the secret witness).

**Witness Key Agreement** Given $n$ parties each having committed their private information $\omega$ and published the respective commitments $\phi$ anonymously on a public bulletin board, we consider the problem that a party wants to securely and anonymously agree on a secret key $\mathsf{k}$ with each counterparty based on their committed information $\omega$. The initiating party wants to make sure that (and the key agreement is only successful if) the counterparty's committed information $\omega$ satisfies a public relation $R$ (given by the initiating party), i.e. $R(\phi,\omega) = 1$, while each counterparty does not want to disclose their $\omega$.

With our problem we push further the envelope of Non-Interactive Zero Knowledge (NIZK) [25]. In both cases, given an instance and an NP-relation $R$, a party (the Prover) can convince another party (the Verifier) that there exists a witness $\omega$ of the instance $\phi$ such that $R(\phi, \omega) = 1$, without leaking information about it. The successful outcome of NIZK is the binary verification result 1 while our desired outcome is a shared secret key.

**Anonymity-Preserving Communication Model** In our problem, we consider the anonymity of each party as critical as other WKA security properties. Therefore, our communication model assumes an anonymous network to hide the parties' identities (e.g., IP address) and all WKA communication must utilize the public bulletin board (e.g. a blockchain), i.e. to publish a message, a party sends it through the anonymous network to the public bulletin board which is readable by all parties.[5]

[5] WKA does not intend to hide whether the Prover/Verifier established communication as they are completely anonymous.

**Table 1.** Dark Pool Example Relations

In each relation we denote $[\![x]\!] = \mathsf{SHA256}(x; r_x)$ the public SHA256 commitment of the secret business variable $x$ using randomness $r_x$. For a dark pool transaction we denote by $c$ the cash capacity of a trader, $c'$ the threshold given by the investor. For a bid we denote $(p, v)$ as the bid price and the bid volume.

| | |
|---|---|
| **Sufficient Capacity (SC)** | |
| Public $\phi = ([\![c]\!], c')$ | Secret $\omega = (c, r_c)$ |
| Conditions: $[\![c]\!] = \mathsf{SHA256}(c; r_c) \wedge c \geq c'$ | |
| **Price Range (PR)** | |
| Public $\phi = ([\![p]\!],\ p'_+,\ p'_-)$ | Secret $\omega = (p,\ r_p)$ |
| Conditions: $[\![p]\!] = \mathsf{SHA256}(p; r_p) \wedge p'_- \leq p \leq p'_+$ | |
| **Matchable Bid (MB)** | |
| Public $\phi = ([\![p]\!],\ [\![v]\!],\ p'_+,\ p'_-,\ c')$ | Secret $\omega = (p,\ v,\ r_p,\ r_v)$ |
| Conditions: $[\![p]\!] = \mathsf{SHA256}(p; r_p),\ p'_- \leq p \leq p'_+ \wedge [\![v]\!] = \mathsf{SHA256}(v; r_v),\ c' \geq pv$ | |

**Practical WKA Construction** We base our WKA construction on the concrete efficient construction of zk-SNARK from Non-Interactive Linear Proof (NILP) [27] for Quadratic Arithmetic Programs (QAP) [22] given by Groth [27] and we utilize Linear-Only Encryption (LE) [8] to compile such NILP to a WKA scheme. We provide the *first practical Witness Key Agreement under designated-verifier zk-SNARK proof for QAP.*In our WKA scheme construction a designated verifier can first broadcast a common reference string as a challenge for the relation $R$ of interest. A prover can then publish a partial zk-SNARK proof as a response for the committed instance that satisfies $R$. Using the partial proof, the verifier can derive a shared secret key with the prover.

**Non-goals** The focus of our protocol design is to protect against *digital* attacks on integrity, anonymity and confidentiality. *Physical, economic and social* attacks are, and always will be, possible similarly to centralized systems (e.g. insider trading, cartels manipulating the underlying assets or the availability glitches such as the NASDAQ ones [46]) and they are typically dealt with by ex-post law enforcement [37].

## 2 Dark Pools as A Motivating Application For WKA

From a security perspective the constraints from the investor are easily captured by an NP-relation $R$ as in Table 1 where the instance $\phi$ is the public information (i.e. the trader's commitment and the investor's constraints) and the witness $\omega$ is the private information (the trader's committed information). An investor may look for traders with enough capacity and use the Sufficient Capacity (SC) relation in Table 1. A trader may ask the investor to show interest in some price ranges, e.g. from $p'_-$ to $p'_+$ using the Price Range (PR) relation and in addition check the consistency of the challenged threshold using Matchable Bid (MB), if the investor has previously committed to desired bid price $p$ and volume $v$ ,where $c' \geq pv$. Thus, the investor can simply post the relation $R$ and use WKA

**Table 2.** Comparison of Solutions

$n$ is the number of parties. The comparison criteria include: (i) **A**: is anonymous communication supported? (ii) **PB**: does the solution satisfies proportional burdern, i.e. only the involved parties perform the computation? (iii) **DL**: does the solution considers the information bound on a distributed ledger? (iv) **AC**: are arithmetic circuits supported? (v) **BR**: blockchain-round complexity, i.e. the number of rounds happen on the blockchain; (vi) **BC**: blockchain-communication complexity, i.e. the size of the data communicated through the blockchain; and (vii) **C**: computational complexity.

| **Solution** | **A** | **PB** | **DL** | **AC** | **BR** | **BC** | **C** |
|---|---|---|---|---|---|---|---|
| Full MPC [30] | y |  | y | y | 13 | $O(n^2)$ | $O(n^2)$ |
| 2-3 Servers MPC [14] |  | y |  | y | N/A | N/A | $O(1)$ |
| Paired 2PC [30] | y |  | y | y | 2 | $O(n)$ | $O(n)$ |
| Practical WE [21] | y | y | y |  | 1 | $O(1)$ | $O(1)$ |
| Practical AKE [29] | y | y | y |  | 2 | $O(1)$ | $O(1)$ |
| WKA | y | y | y | y | 2 | $O(1)$ | $O(1)$ |

to securely agree on a secret key with each interested and eligible trader holding a secret witness $\omega$ (to their committed instance $\phi$) that satisfies the desired relation, i.e. $R(\phi, \omega) = 1$. Each agreed key can then be used for the negotiation (usually a conversation, not just a single message) of the offer between the investor and each eligible trader.

Our WKA construction also aims for succinct communication which is important when using a distributed ledger. The committed information (the instance) is also frequently updated, while the relation $R$ of interest may be persistent. WKA is advantageous in this case as it works efficiently with different instances of the same relation. Additionally, WKA allows the trader to send a message encrypted using the key along with the public response (that will be used by the verifier to reconstruct the key and decrypt the message). This may save one round and is key when executing over a blockchain.[6]

## 3 Related Work and Alternative Candidate Schemes

We summarize a comparison of WKA in terms of usability and efficiency against applicable alternative candidate schemes in Table. 2 (see §A for more details).

[6] One can argue that there could be DDOS attacks where an attacker can post either malformed offers, or correctly formed ones but they have no intention of filling, to the blockchain. In the first case, as the Verifier only needs to forge the last proof element F (1) while the Prover has to compute the full proof (4(m-l+3n)) as shown in Table 3, such an attack will require tremendous effort from the Prover but not so much from the Verifier. In the second case, unfortunately we cannot solve this as it exists even in the centralized system. A trader/investor can post an offer, and cancel it before it is filled or immediately in the next round. However, at the point the offer was posted, the exchange cannot know whether the offer will be canceled or not.

A *trivial (but wrong)* solution is to ask each prover to couple a public key $\mathsf{pk}$ with a zk-SNARK proof $\boldsymbol{\pi}$ for the satisfaction of the arithmetic relation $R$. The verifier can then encrypt the private offer with $\mathsf{pk}$ after verifying the proof $\boldsymbol{\pi}$. Only the prover with the corresponding private key $\mathsf{sk}$ can decrypt. Since the decryption condition above says nothing about the validity of $\boldsymbol{\pi}$, one cannot guarantee that $\mathsf{pk}$ is actually from the prover that produced $\boldsymbol{\pi}$. Signature of Knowledge [28] (SoK), can be used to sign the public key $\mathsf{pk}$. However, SoK delivers only $\mathsf{pk}$ of the prover thus allows only a one-way communication from the verifier to the prover. Further, the prover cannot make sure that the upcoming message encrypted with $\mathsf{pk}$ is from the verifier: as $\mathsf{pk}$ is public, anyone can see it and send a message to the prover using $\mathsf{pk}$. Other similar generic constructions are generally based on the modification of $R$ to include a transformation of $\mathsf{k}_r$. Our WKA scheme uses directly $R$ which yields a lower bound of circuit complexity. Besides, those approaches usually require full proof verification (that involves pairings, e.g. 5 as in [27]), which is more costly than our construction, where the Verifier directly forges the last proof element (only computation in the field F) and it even stops 1 step early.

Secure Multiparty Computation (MPC) [15] can be a general solution but is with either usability and efficiency issues. Firstly, setting up an MPC using existing distributed ledgers is not trivial as every party must be known in advanced or a PKI must be available in the setup phase for securing the communication over the ledger, e.g. as in [11]. Additionally, general Full MPC (where $n$ parties join the computation, e.g. [30]) yields an unacceptable 13 rounds of blockchain communication; while the 2-3 Servers MPC (where $n$ parties secret share their private inputs to the servers and let them perform the computation, e.g. [14]) and Paired 2PC (where the verifier contacts and perform a 2PC with each other party, e.g. [30]) fail to guarantee anonymity which can be critical [38].

Authenticated key exchanges (AKE) [7,12] only support relations on credentials. Here we have other relations among values not related to credentials as they can change dynamically. Language-AKE [29] is more flexible but it does not support non-algebraic relations such as SHA-256 employed by ZeroCash [44]. One can also use Witness Encryption [21] (WE) with the desired arithmetic relation $R$, and only the provers who possess the witness $\omega$ for that instance $\phi$ such that $R(\phi, \omega) = 1$ could decrypt. However, general WE constructions [19,23,20,3] are impractical while practical WE under a GS proof [16] cannot support arithmetic relation of depth greater than 1, e.g. SHA-256 as employed by ZeroCash [44]).

## 4 Witness Key Agreement

**Notations** A multivariate polynomial $t : \mathbb{F}^m \to \mathbb{F}$ over a finite field $\mathbb{F}$ has a degree $d$ if the degree of each monomial in $t$ is at most $d$ and a monomial has degree d. A multivalued multivariate polynomial $\mathbf{t} : \mathbb{F}^m \to \mathbb{F}^\mu$ is a vector of polynomials $(t_1, \dots, t_\mu)$ where each $t_i : \mathbb{F}^m \to \mathbb{F}$ is a multivariate polynomial. We denote a scalar by $x$ and a vector by $\mathbf{x}$. We write $x \leftarrow \mathbb{X}$ when picking an element $x$ uniformly from a finite set $\mathbb{X}$. We write $y \leftarrow \mathsf{A}(x)$ when picking the randomness $r$ and returning $y = \mathsf{A}(x; r)$. $\Pr[\epsilon|\Omega]$ denotes the probability of an event $\epsilon$ over the probability space $\Omega$. We denote the security parameter by $1^\lambda$ in the unary form and the negligible function as $\mathsf{negl}(\cdot)$. Given two probability functions $f, g : \mathbb{N} \to [0, 1]$ we write $f(\lambda) \approx g(\lambda)$ when $|f(\lambda) - g(\lambda)| = O(\lambda^{-c})$ for every constant $c > 0$. We say that $f$ is *negligible* when $f(\lambda) \approx 0$.

*Remark 1 (Generation of the relation $R$).* We follow the notation of Groth [27] so that a relation generator $\mathcal{R}$ receives a security parameter $1^\lambda$ and returns a polynomial-time decidable binary relation $R$, i.e. $R \leftarrow \mathcal{R}(1^\lambda)$. Hence for notational simplicity we can assume $1^\lambda$ can be deduced from $R$.

**Definition 1 (Witness Key Agreement).** *Let $L$ be an NP-language with the witness relation $R(\phi, \omega)$. We call $\phi$ an instance of $L$ and $\omega$ a witness for $\phi$. A Witness Key Agreement (WKA) scheme $\Omega$ for $L$ is a tuple of polynomial-time algorithms (*$\mathsf{KChallenge}$*,* $\mathsf{KResponse}$*,* $\mathsf{KDerive}$*):*

$(\mathsf{p}_c, \mathsf{s}_c) \leftarrow \mathsf{KChallenge}(R)$ *is run by the verifier and takes as input the relation $R$ (from which the security parameter $1^\lambda$ can be deduced), outputs a public and a secret challenge parameter $(\mathsf{p}_c, \mathsf{s}_c)$.*

$(\mathsf{p}_r, \mathsf{k}_r) \leftarrow \mathsf{KResponse}(R, \mathsf{p}_c, \phi, \omega)$ *is run by the prover with inputs the relation $R$, the public challenge parameter $\mathsf{p}_c$, the instance $\phi$, and the corresponding witness $\omega$, outputs a public response parameter $\mathsf{p}_r$ and a secret key $\mathsf{k}_r$.*

$\{\mathsf{k}_c, \perp\} \leftarrow \mathsf{KDerive}(R, \mathsf{s}_c, \phi, \mathsf{p}_r)$ *is run by the verifier and takes as input the relation $R$, the secret challenge parameter $\mathsf{s}_c$, the instance $\phi$and the public response parameter $\mathsf{p}_r$, outputs a key $\mathsf{k}_c$ or $\perp$.*

**Security Properties** WKA is closely related to Non-Interactive Zero-Knowledge (NIZK) Proof System. The key difference is the outcome of NIZK is only a binary verification result while WKA's outcome is a key upon success. Hence the security properties of WKA are also very similar to those of NIZK. Furthermore, we require WKA to be secure against MITM attack. (See §B for a trivial WKA generic construction that is insecure under MITM attack.)

**WKA Construction Roadmap** We base our WKA construction on the efficient construction of zk-SNARK from Non-Interactive Linear Proof (NILP) [27] for Quadratic Arithmetic Programs (QAP) [22] given by Groth [27] and we utilize Linear-Only Encryption (LE) [8] to compile such NILP to a WKA scheme.

*Linear Interactive Proofs* (LIP) [8] is an extension of interactive proofs [26] in which each prover's message is an *affine combination* of the previous messages sent by the verifier. Groth renamed the input-oblivious two-message LIPs into NILP [27] to clarify the connection between LIP and NIZK. NILP considers only *adversaries using affine prover strategies*, i.e. a strategy which can be described by a tuple $(\boldsymbol{\Pi}, \boldsymbol{\pi}_0)$ where $\boldsymbol{\Pi} \in \mathbb{F}^{k \times y}$ represents a linear function and $\boldsymbol{\pi}_0 \in \mathbb{F}^k$ represents an affine shift. Then, on input a query vector $\boldsymbol{\sigma} \in \mathbb{F}^y$, the response vector $\boldsymbol{\pi} \in \mathbb{F}^k$ is constructed by evaluating the affine relation $\boldsymbol{\pi} = \boldsymbol{\Pi}\boldsymbol{\sigma} + \boldsymbol{\pi}_0$.

*Key Observation.* The proof $\boldsymbol{\pi}$ obtained with NILP consists of $k$ elements (by evaluating $k$ linear functions [7] corresponding to the proof matrix $\boldsymbol{\Pi}$), in which

[7] In the concrete construction by Groth [27] (see also Fig. 3), $k = 3$ and the proof matrix $\boldsymbol{\Pi}$ is represented as the coefficients of the linear functions.

**Perfect Correctness** Given a true instance, the key agreement is successful, i.e.

$$\Pr\left[\mathsf{k}_c = \mathsf{k}_r \,\middle|\, \begin{array}{l} R \leftarrow \mathcal{R}(1^\lambda) \\ R(\phi,\omega) = 1 \end{array}, \begin{array}{rl} (\mathsf{p}_c, \mathsf{s}_c) & \leftarrow \mathsf{KChallenge}(R) \\ (\mathsf{p}_r, \mathsf{k}_r) & \leftarrow \mathsf{KResponse}(R, \mathsf{p}_c, \phi, \omega) \\ \mathsf{k}_c & \leftarrow \mathsf{KDerive}(R, \mathsf{s}_c, \phi, \mathsf{p}_r) \end{array}\right] = 1 \tag{1}$$

**Computational Adaptive Knowledge Soundness** The key agreement is successful only with negligible probability if the prover knows no witness for the instance, i.e. for any PPT $\hat{\mathcal{A}}$, there exists a poly-time extractor $\epsilon_{\hat{\mathcal{A}}}$

$$\Pr\left[\begin{array}{l} R(\phi,\omega) \neq 1 \\ \mathsf{k}_c = \mathsf{k}_r \end{array} \,\middle|\, \begin{array}{rl} R & \leftarrow \mathcal{R}(1^\lambda) \\ (\mathsf{p}_c, \mathsf{s}_c) & \leftarrow \mathsf{KChallenge}(R) \\ (\phi, \mathsf{p}_r, \mathsf{k}_r) & \leftarrow \hat{\mathcal{A}}(R, \mathsf{p}_c) \\ \mathsf{k}_c & \leftarrow \mathsf{KDerive}(R, \mathsf{s}_c, \phi, \mathsf{p}_r) \\ \omega & \leftarrow \epsilon_{\hat{\mathcal{A}}}(R, \phi, \mathsf{p}_r, \mathsf{k}_c) \end{array}\right] < \mathsf{negl}(\lambda) \tag{2}$$

**Perfect Honest Verifier Zero-knowledge** The response leaks nothing about the witness in the honest setup, i.e. there is a simulator $\mathcal{S}_{ZK}$ that outputs a simulated response $(\mathsf{p}_r, \mathsf{k}_r)$ and key $\mathsf{k}_c$. Formally, for all $\lambda \in \mathbb{N}$, $R \leftarrow \mathcal{R}(1^\lambda)$, $R(\phi,\omega) = 1$ and any PPT $\hat{\mathcal{A}}$:

$$\begin{aligned} &\Pr\left[\hat{\mathcal{A}}(R, \mathsf{p}_c, \mathsf{s}_c, \phi, \mathsf{p}_r, \mathsf{k}_c) = 1 \,\middle|\, \begin{array}{rl} (\mathsf{p}_c, \mathsf{s}_c) & \leftarrow \mathsf{KChallenge}(R) \\ (\mathsf{p}_r, \mathsf{k}_r) & \leftarrow \mathsf{KResponse}(R, \mathsf{p}_c, \phi, \omega) \\ \mathsf{k}_c & \leftarrow \mathsf{KDerive}(R, \mathsf{s}_c, \phi, \mathsf{p}_r) \end{array}\right] \\ &= \Pr\left[\hat{\mathcal{A}}(R, \mathsf{p}_c, \mathsf{s}_c, \phi, \mathsf{p}_r, \mathsf{k}_c) = 1 \,\middle|\, \begin{array}{rl} (\mathsf{p}_c, \mathsf{s}_c) & \leftarrow \mathsf{KChallenge}(R) \\ (\mathsf{p}_r, \mathsf{k}_r, \mathsf{k}_c) & \leftarrow \mathcal{S}_{ZK}(R, \mathsf{p}_c, \mathsf{s}_c, \phi) \end{array}\right] \end{aligned} \tag{3}$$

**Perfect Response and Key Indistinguishability** The public response and the agreed key can be simulated without knowledge of a witness, i.e. for all $\lambda \in \mathbb{N}$, $R \leftarrow \mathcal{R}(1^\lambda)$, $R(\phi,\omega) = 1$ there is a simulator $\mathcal{S}_{RKI}$ s.t. for any PPT $\hat{\mathcal{A}}$:

$$\begin{aligned} &\Pr\left[\hat{\mathcal{A}}(R, \mathsf{p}_c, \phi, \mathsf{p}_r, \mathsf{k}_r) = 1 \,\middle|\, \begin{array}{rl} (\mathsf{p}_c, \mathsf{s}_c) & \leftarrow \mathsf{KChallenge}(R) \\ (\mathsf{p}_r, \mathsf{k}_r) & \leftarrow \mathsf{KResponse}(R, \mathsf{p}_c, \phi, \omega) \end{array}\right] \\ &= \Pr\left[\hat{\mathcal{A}}(R, \mathsf{p}_c, \phi, \mathsf{p}_r, \mathsf{k}_r) = 1 \,\middle|\, \begin{array}{rl} (\mathsf{p}_c, \mathsf{s}_c) & \leftarrow \mathsf{KChallenge}(R) \\ (\mathsf{p}_r, \mathsf{k}_r) & \leftarrow \mathcal{S}_{RKI}(R, \mathsf{p}_c, \phi) \end{array}\right] \end{aligned} \tag{4}$$

**Security against Man-In-The-Middle Attack** The key agreement is successful only with negligible probability under Man-In-The-Middle Attack, i.e. for any PPT $\hat{\mathcal{A}}$:

$$\Pr\left[\mathsf{k}_c = \mathsf{k}'_r \,\middle|\, \begin{array}{l} R \leftarrow \mathcal{R}(1^\lambda) \\ R(\phi,\omega) = 1 \end{array}, \begin{array}{rl} (\mathsf{p}_c, \mathsf{s}_c) & \leftarrow \mathsf{KChallenge}(R) \\ (\mathsf{p}'_c, \mathsf{s}'_c) & \leftarrow \mathsf{KChallenge}(R) \\ (\mathsf{p}_r, \mathsf{k}_r) & \leftarrow \mathsf{KResponse}(R, \mathsf{p}'_c, \phi, \omega) \\ (\mathsf{p}'_r, \mathsf{k}'_r) & \leftarrow \hat{\mathcal{A}}(R, \mathsf{p}_c, \mathsf{p}'_c, \mathsf{s}'_c, \mathsf{p}_r, \phi) \\ \mathsf{k}_c & \leftarrow \mathsf{KDerive}(R, \mathsf{s}_c, \phi, \mathsf{p}'_r) \end{array}\right] < \mathsf{negl}(\lambda) \tag{5}$$

**Fig. 1.** Security of Witness Key Agreement Scheme

the $k$-th element can be obtained in two ways given the first $k-1$ elements [27]: (1) On the prover's side, if $\boldsymbol{\pi}$ is valid then the first $k-1$ elements fully determine the last one; (2) On the verifier's side, the first $k-1$ elements can be used in a proof forging formula to obtain the last one. By the prover computing $\boldsymbol{\pi}$ and publishing the first $k-1$ elements of $\boldsymbol{\pi}$, both parties can agree on the last element to use as a shared secret key for secure communication.[8] With this observation we construct WKA from a new NILP notion: *split designated verifier NILP*. (§5).

*Succinct zero-knowledge non-interactive argument of knowledge* (zk-SNARK) follows the relaxation from Perfect Soundness to Computational Soundness [24]. Bitansky *et al.* [8] also showed that NILP can be compiled into both publicly verifiable (verifier degree 2, using bilinear maps) and designated-verifier (using linear-only encryption scheme) zk-SNARK. Intuitively the prover computes the proof $\boldsymbol{\pi}$ as linear combinations of the CRS $\boldsymbol{\sigma}$ and the verifier checks the argument by checking the quadratic equations corresponding to the relation $R$.

*Linear-Only Encryption* (LE) scheme $\Sigma$ (Bitansky *et al.* [8]), e.g. a two-ciphertexts variant of Paillier [41], is a tuple of polynomial-time algorithms ($\mathsf{KeyGen}$, $\mathsf{Enc}$, $\mathsf{ImgVer}$, $\mathsf{Dec}$, $\mathsf{Add}$) where the $\mathsf{ImgVer}$ (image verification) prevents oblivious ciphertext samplings in the image of $\mathsf{Enc}$ using $\mathsf{pk}$, i.e. this property prevents the adversary from encrypting plaintexts from scratch (see §E for further details), and $\mathsf{Add}$ is for evaluating linear combinations of valid ciphertexts. An LE scheme satisfies *correctness*, *additive homomorphism*, *indistinguishability under chosen plaintext attack* (IND-CPA) and in addition *linear-only homomorphism* which essentially says that it is infeasible to generate a new valid ciphertext except by evaluating an affine combination of valid ciphertexts (via $\mathsf{Add}$)[9]. Such LE scheme can be instantiated using existing encryption schemes. The security of an LE scheme relies on the assumptions of q-power Diffie-Hellman, q-power Knowledge of Exponent and q-power Knowledge of Equality [8].

For relation functionality and efficiency in WKA we leverage on *Quadratic Arithmetic Programs* (QAP) by Gennaro et. al. [22]: an arithmetic circuit can be transformed into a system of equations that check the consistency of a set of instance variables $\phi$ and witness variables $\omega$ in a relation $R$. The consistency checker is compiled into zk-SNARK. Thus zk-SNARK for QAP covers applications that employ arithmetic relations of multiplicative depth larger than one such as SHA256. In our WKA construction the partial proof size is also succinct, as it has at most 3 elements regardless of $R$. Response computation and key derivation are efficient, i.e. only linear in QAP size.

**Limitations of our WKA construction** Our WKA scheme, as any scheme, inherits the limitations of its components:, i.e. the designated-verifier zk-SNARK

[8] The concrete example of this observation can be seen in Fig. 3 in section §6. The first two elements $A$ and $B$ (Eq. (7) and (8)) uniquely define $C$ (Eq. (9)) and they can be fed into the proof forging formula (Eq. (11)) to get the 3rd element $C$ which should be the same for either party.

[9] This property formally guarantees that given a valid ciphertext $\boldsymbol{\pi}$ by an adversary, it is possible to *efficiently* extract the corresponding affine function $(\boldsymbol{\Pi}, \boldsymbol{\pi}_0)$ that explains $\boldsymbol{\pi}$. Such property is important for Knowledge Soundness of WKA.

that is compiled from an NILP for QAP by Groth [27]. Firstly zk-SNARKs are not known to satisfy composability and therefore cannot be run out of the box in parallel in the design of larger protocols [34].[10] In a basic dark pool scenario we only consider sequential composition where each execution of WKA concludes before the next execution begins [13]. For extended scenario one might need to use other instruments to identify parallel runs as described in Principle 10 of security protocol design by Abadi and Needham [1]. However, note that we still consider security against MITM attack, which is important for key agreement protocols. Secondly our WKA scheme makes use of QAP [22] hence it is only as efficient as the circuit expressing the constraints. Finally, we opted for simplicity rather than making the WKA scheme subversion-resistant as this which would require the zero-knowledge property be maintained even when the CRS is maliciously generated (see Bellare *et al.* [5]). Abdolmaleki *et al.* [2] and Fuchsbauer [18] constructed subversion-resistant NIZK based on Groth's zk-SNARK construction [27]. However, both works consider only the publicly verifable zk-SNARK construction based on bilinear groups. Our WKA construction requires designated-verifier zk-SNARK, and therefore those constructions are not applicable to our scheme. Hence, we consider only honest setups.[11]

## 5 WKA From NILP

We first define our *split designated verifier NILP* based on Groth's definition [27]. The CRS is first split into two parts $(\boldsymbol{\sigma}_P, \boldsymbol{\sigma}_V)$ where $\boldsymbol{\sigma}_V$ is only available to the verifier. Subsequently, in proof computation we split the proof matrix $\boldsymbol{\Pi} \in \mathbb{F}^{k \times y}$ into two parts: $\boldsymbol{\Pi}_1 \in \mathbb{F}^{k-1 \times y}$ and $\boldsymbol{\Pi}_2 \in \mathbb{F}^{1 \times y}$. The proof $\boldsymbol{\pi}$ is also split into $\boldsymbol{\pi}_1 = \boldsymbol{\Pi}_1 \boldsymbol{\sigma}_P$ that consists of $k-1$ elements and $\boldsymbol{\pi}_2 = \boldsymbol{\Pi}_2 \boldsymbol{\sigma}_P$ consists of the last element. This split of $\boldsymbol{\Pi}$ and $\boldsymbol{\pi}$ is not necessary in a zk-SNARK proof system but it is essential in our WKA scheme as we need to split the proof into two parts (See our key observation in §4).

**Definition 2 (Split designated-verifier NILP).** *Let $L$ be an NP-language with the witness relation $R(\phi, \omega)$. We call $\phi$ an instance of $L$ and $\omega$ a witness for $\phi$. A split designated-verifier (split DV) NILP for $L$ consists of the tuple of polynomial-time algorithms (*Setup*,* Prove*,* Verify*,* Simulate*):*

$(\boldsymbol{\sigma}_P, \boldsymbol{\sigma}_V) \leftarrow \mathsf{Setup}(R)$**:** *output* $\boldsymbol{\sigma}_P \in \mathbb{F}^y$ *and* $\boldsymbol{\sigma}_V \in \mathbb{F}^x$ .

[10] Users are advised to run the shared secret through a hash function modelled as a random oracle before using it as a key for any other cryptosystem.

[11] Such an assumption can be relaxed by asking a TTP to generate the CRS (such as Bloomberg itself). Using a TTP for bootstrapping security protocols have been considered in literature, see for example HAWK [33]. This is a much weaker trust assumption than managing orders themselves because the generation of the CRS requires only the relation R and the public key for the encryption. Therefore such a TTP is only trusted to do the computation correctly. Without the private key, the TTP cannot learn additional information.

```
Following Groth [27] we assume $1^\lambda$ can be deduced from $R$.

$(\mathsf{p}_c, \mathsf{s}_c) \leftarrow \mathsf{KChallenge}(R)$ runs as follows.
1. Fix a linear-only encryption scheme $\boldsymbol{\Sigma}$;
2. Run $(\mathsf{pk}, \mathsf{sk}) \leftarrow \boldsymbol{\Sigma}.\mathsf{KeyGen}(1^\lambda)$ where $1^\lambda$ is the security parameter deduced from
   $R$ (see Remark 1); and $(\boldsymbol{\sigma}_P, \boldsymbol{\sigma}_V) \leftarrow \mathsf{Setup}(R)$;
3. Encrypt $[\sigma_{P,i}, r_{P,i}] \leftarrow \boldsymbol{\Sigma}.\mathsf{Enc}(\mathsf{pk}, \sigma_{P,i})$ for each $\sigma_{P,i} \in \boldsymbol{\sigma}_P$;
4. Encrypt $[r_{P,i}] \leftarrow \boldsymbol{\Sigma}.\mathsf{Enc}(\mathsf{pk}, r_{P,i})$ for each $r_{P,i}$ above;
5. Return $\mathsf{p}_c = (\mathsf{pk}, \{[\sigma_{P,i}, r_{P,i}]\}_{i=1}^{y}, \{[r_{P,i}]\}_{i=1}^{y})$ and $\mathsf{s}_c = (\mathsf{sk}, \boldsymbol{\sigma}_V)$.
$(\mathsf{p}_r, \mathsf{k}_r) \leftarrow \mathsf{KResponse}(R, \mathsf{p}_c, \phi, \omega)$: Upon receiving the challenge $\mathsf{p}_c$,
1. Run $(\boldsymbol{\Pi}_1, \boldsymbol{\Pi}_2) \leftarrow \mathsf{ProofMatrix}(\phi, \omega | R)$;
2. Compute $\{[\pi_{1,j}, r_{1,j}]\}_{j=1}^{k-1} = \boldsymbol{\Pi}_1(\{[\sigma_{P,i}, r_{P,i}]\}_{i=1}^{y})$ (with $\boldsymbol{\Sigma}.\mathsf{Add}$);
3. Compute $[\pi_2, r_2] = \boldsymbol{\Pi}_2(\{[\sigma_{P,i}, r_{P,i}]\}_{i=1}^{y})$ (with $\boldsymbol{\Sigma}.\mathsf{Add}$);
4. Compute $[r_2] = \boldsymbol{\Pi}_2(\{[r_{P,i}]\}_{i=1}^{y})$ (with $\boldsymbol{\Sigma}.\mathsf{Add}$);
5. Return $\mathsf{p}_r = (\{[\pi_{1,j}, r_{1,j}]\}_{j=1}^{k}, [r_2])$ and $\mathsf{k}_r = [\pi_2, r_2]$.
$\{\mathsf{k}_c, \perp\} \leftarrow \mathsf{KDerive}(R, \mathsf{s}_c, \phi, \mathsf{p}_r)$ Output $\perp$ if any verification fails:
1. Verify $\mathsf{ImgVer}(\mathsf{sk}, [\pi_{1,j}, r_{1,j}]) = 1$ for $1 \le j \le k-1$; and $\mathsf{ImgVer}(\mathsf{sk}, [r_2]) = 1$;
2. Decrypt $\pi_{1,j} = \boldsymbol{\Sigma}.\mathsf{Dec}(\mathsf{sk}, [\pi_{1,j}, r_{1,j}])$ for $1 \le j \le k-1$;
3. Obtain $\mathbf{t} \leftarrow \mathsf{Test}(R, \phi)$; use $\{\pi_{1,j}\}_{j=1}^{k-1}$ to solve $\mathbf{t}(\boldsymbol{\sigma}_V, \{\pi_{1,j}\}_{j=1}^{k}, \pi_2) = 0$ for $\pi_2$;
4. Decrypt $r_2 = \boldsymbol{\Sigma}.\mathsf{Dec}(\mathsf{sk}, [r_2])$
5. Return $\mathsf{k}_r = \boldsymbol{\Sigma}.\mathsf{Enc}(\mathsf{pk}, \pi_2, r_2)$ ($r_2$ as randomness).
```

**Fig. 2.** Construction of Witness Key Agreement

$(\boldsymbol{\pi}_1, \boldsymbol{\pi}_2) \leftarrow \mathsf{Prove}(R, \boldsymbol{\sigma}, \phi, \omega)$**:** *obtain* $(\boldsymbol{\Pi}_1, \boldsymbol{\Pi}_2) \leftarrow \mathsf{ProofMatrix}(R, \phi, \omega)$ *where* $\boldsymbol{\Pi}_1 \in \mathbb{F}^{k-1 \times y}$ *and* $\boldsymbol{\Pi}_2 \in \mathbb{F}^{1 \times y}$ *and output* $\boldsymbol{\pi}_1 = \boldsymbol{\Pi}_1 \boldsymbol{\sigma}_P$ *and* $\boldsymbol{\pi}_2 = \boldsymbol{\Pi}_2 \boldsymbol{\sigma}_P$

$\{0,1\} \leftarrow \mathsf{Verify}(R, \boldsymbol{\sigma}_V, \phi, \boldsymbol{\pi}_1, \boldsymbol{\pi}_2)$**:** *obtain* $\mathbf{t} \leftarrow \mathsf{Test}(R, \phi)$ *where* $\mathbf{t} : \mathbb{F}^{y+k} \to \mathbb{F}^{\eta}$ *is an arithmetic circuit corresponding to the evaluation of multivariate polynomials such that* $\mathbf{t}(\boldsymbol{\sigma}_V, \boldsymbol{\pi}_1, \boldsymbol{\pi}_2) = 0$ *if* $\boldsymbol{\pi}$ *is valid..*

$(\boldsymbol{\pi}_1, \boldsymbol{\pi}_2) \leftarrow \mathsf{Simulate}(R, \boldsymbol{\sigma}_V, \phi)$**:** *obtain* $\mathbf{t} \leftarrow \mathsf{Test}(R, \phi)$ *and solve* $\mathbf{t}(\boldsymbol{\sigma}_V, \boldsymbol{\pi}_1, \boldsymbol{\pi}_2) = 0$ *for the output* $(\boldsymbol{\pi}_1, \boldsymbol{\pi}_2)$*.*

*where* $y, x, k, \eta$ *and* $d$ *are constants or polynomials in* $1^\lambda$ *(deduced from* $R$ *[27]).*

A tuple of PPT algorithms (Setup, Prove, Verify, Simulate) is a split DV NILP if it has perfect completeness, perfect zero-knowledge and statistical soundness against affine prover strategies.

**Construction of Witness Key Agreement** We construct WKA from Split DV NILP as shown in Fig. 2. Below we describe the construction at a high level.

We first modify the LE scheme's encryption algorithm interface for explicit used randomness. We omit the randomness $r$ and write only $[m] \leftarrow \mathsf{Enc}(\mathsf{pk}, m)$ in case $r$ is not necessary in subsequent computation. We write $[m] = \mathsf{Enc}(\mathsf{pk}, m, r)$ to incorporate the randomness directly into the encryption algorithm. Secondly we require that the additive homomorphism of LE applies to both the message and the randomness used, i.e. $\mathsf{Add}(\mathsf{pk}, \langle [m_i, r_i] \rangle, \langle \alpha_i \rangle)$ evaluates $[\sum \alpha_i m_i, \sum \alpha_i r_i]$.

**The challenge phase.** In KChallenge, the verifier generates a CRS ($\boldsymbol{\sigma}_P$, $\boldsymbol{\sigma}_V$) from $R$ (using a split DV NILP). The verifier then encrypts each elements

$\{\sigma_{P,i}\}_{i=1}^{y}$ of the $\boldsymbol{\sigma}_P$ with an LE scheme (with key pair $\mathsf{pk}$, $\mathsf{sk}$). Additionally, we require the verifier to encrypt the randomnesses $\{r_{P,i}\}_{i=1}^{y}$ that are used for the encryption of the CRS $\{\sigma_{P,i}\}_{i=1}^{y}$ in $\mathsf{KChallenge}$ into $\{[r_{P,i}]\}_{i=1}^{y}$. Finally s/he publishes a challenge that consists of $\mathsf{pk}$ and the encrypted elements. The verifier keeps private $\mathsf{sk}$ of the LE scheme and the plain CRS $\boldsymbol{\sigma}_V$.

**The response phase.** Upon seeing the challenge, in $\mathsf{KResponse}$, the prover computes a response by generating a valid proof $\boldsymbol{\pi}$ for the desired tuple $(\phi,\ \omega)$ (using the proof matrix of the split DV NILP and the additive homomorphic operation $\mathsf{Add}$ of the LE scheme). When the prover evaluates the last encrypted element $[\pi_2, r_2]$ using the proof matrix $\boldsymbol{\Pi}_2$ and the encrypted CRS $\{[\sigma_{P,i}]\}_{i=1}^{y}$, by the additively homomorphic property of the LE scheme, s/he can also evaluate the ciphertext $[r_2]$ of the randomness $r_2$ of the encrypted $[\pi_2, r_2]$ using the same $\boldsymbol{\Pi}_2$ and $\{[r_{P,i}]\}$. The prover publishes the first encrypted $k-1$ elements $\{[\pi_{1,j}, r_{1,j}]\}_{j=1}^{k-1}$ and the encrypted randomness $[r_2]$ as a public response and keeps secret the last encrypted element $[\pi_2, r_2]$.

**The key derive phase.** When the verifier sees the instance $\phi$ and the corresponding response, in $\mathsf{KDerive}$, s/he can decrypt the encrypted elements using $\mathsf{sk}$ to get $\{\pi_{1,j}\}_{j=1}^{k-1}$ and forge the last element $\pi_2$ using the plain CRS $\boldsymbol{\sigma}_V$. The verifier then uses the evaluated $[r_2]$ to reconstruct the correct ciphertext $[\pi_2, r_2]$ of the last element, i.e. the verifier decrypts $[r_2]$ to get $r_2$ to use as the randomness in the final encryption of $\pi_2$ to get $[\pi_2]$. After that, both parties agree on the same $[\pi_2, r_2]$.

We refer the reader to §C for the proof sketch of our main theorem as follows.

**Theorem 1 (Security of WKA).** *If $\Sigma$ satisfies correctness, additive homomorphism, IND-CPA and linear-only homomorphism, and the underlying split DV NILP satisfies perfect completeness, perfect zero-knowledge and statistical knowledge soundness against affine prover strategies, then $\Omega$ satisfies correctness, adaptive knowledge soundness, honest verifier zero-knowledge, response and key indistinguishability, and security against man-in-the-middle attack.*

## 6 WKA from NILP based on QAP

We recall the formal definition of Quadratic Arithmetic Programs (QAP) [22] and how to construct a NILP for QAP [27].

**Definition 3 (QAP).** *A quadratic arithmetic program $\mathbb{Q}$ over a field $\mathbb{F}$ for a relation $R(\phi,\ \omega)$ consists of three sets of polynomial $\{u_i(X), v_i(X), w_i(X)\}_{i=0}^{m}$ and a target polynomial $t(X) = \Pi_{q=1}^{n}(X - r_q)$ such that with $a_0 = 1$, $\phi = \{a_i\}_{i=1}^{l}$, and $\omega = \{a_i\}_{i=l+1}^{m}$, the following Eq. (12) holds.*

$$\sum_{i=0}^{m} a_i u_i(X) \sum_{i=0}^{m} a_i v_i(X) = \sum_{i=0}^{m} a_i w_i(X) + h(X)t(X) \tag{12}$$

*where $u_i(X), v_i(X), w_i(X)$ are of degee $n-1$ and $h(X)$ is of degree $n-2$.*

We consider the QAP that defines a binary relation $R$ as described in Remark 2. NILP for such QAP is defined as a tuple of polynomial-time algorithms (Setup, Prove, Verify, Simulate):

$(\boldsymbol{\sigma}, \boldsymbol{\tau}) \leftarrow \mathsf{Setup}(R)$**:** Pick $\alpha, \beta, \gamma, \delta, x \leftarrow \mathbb{F}^*$. Set $\boldsymbol{\tau} = (\alpha, \beta, \gamma, \delta, x)$ and $\boldsymbol{\sigma}$:

$$\begin{gathered}\boldsymbol{\sigma} = \alpha, \beta, \gamma, \delta, \{x^i\}_{i=0}^{n-1}, \left\{\frac{\beta u_i(x) + \alpha v_i(x) + w_i(x)}{\gamma}\right\}_{i=0}^{l}, \\ \left\{\frac{\beta u_i(x) + \alpha v_i(x) + w_i(x)}{\delta}\right\}_{i=l+1}^{m}, \left\{\frac{x^i t(x)}{\delta}\right\}_{i=0}^{n-2}\end{gathered} \tag{6}$$

$\boldsymbol{\pi} \leftarrow \mathsf{Prove}(R, \boldsymbol{\sigma}, a_1, \ldots, a_m)$**:** Pick $r, s \leftarrow \mathbb{F}$ and compute

$$A = \alpha + \sum_{i=0}^{m} a_i u_i(x) + r\delta \tag{7}$$

$$B = \beta + \sum_{i=0}^{m} a_i v_i(x) + s\delta \tag{8}$$

$$C = \sum_{i=l+1}^{m} a_i \frac{\beta u_i(x) + \alpha v_i(x) + w_i(x)}{\delta} + \frac{h(x)t(x)}{\delta} + sA + rB - rs\delta \tag{9}$$

In NILP [27], $\boldsymbol{\pi} = (A, B, C)$. In Split DV NILP, $\boldsymbol{\pi}_1 = (A, B)$ and $\boldsymbol{\pi}_2 = (C)$. $\{0,1\} \leftarrow \mathsf{Verify}(R, \boldsymbol{\sigma}, a_1, \ldots, a_l, \pi)$**:** Output 1 iff:

$$AB = \alpha\beta + \sum_{i=0}^{l} a_i \frac{\beta u_i(x) + \alpha v_i(x) + w_i(x)}{\gamma}\gamma + C\delta \tag{10}$$

$\boldsymbol{\pi} \leftarrow \mathsf{Simulate}(\boldsymbol{\tau}|R, a_1, \ldots, a_l)$**:** Pick $A, B \leftarrow \mathbb{F}$, and output $\boldsymbol{\pi} = (A, B, C)$ where:

$$C = \frac{AB}{\delta} - \frac{\alpha\beta}{\delta} - \frac{\sum_{i=0}^{l} a_i(\beta u_i(x) + \alpha v_i(x) + w_i(x))}{\delta} \tag{11}$$

**Fig. 3.** Split NILP for QAP based on Groth [27])

*Remark 2 (QAP description).* For convenience we follow the QAP description of Groth [27], we consider the QAP $R$, i.e.

$$(\mathbb{F}, aux, l, \{u_i(X), v_i(X), w_i(X)\}_{i=0}^{m}, t(X))$$

where $\mathbb{F}$ is a finite field; $aux$ is some auxiliary information; $1 \leq l \leq m$; $u_i(X)$,$v_i(X)$,$w_i(X)$, $t(X) \in \mathbb{F}[X]$, $u_i(X)$,$v_i(X)$,$w_i(X)$ are of at most degree $n-1$. Such QAP defines a binary relation

$$R = \left\{ (\phi, \omega) \left| \begin{array}{c} a_0 = 1, \phi = \{a_i\}_{i=1}^{l}, \omega = \{a_i\}_{i=l+1}^{m} \\ \sum_{i=0}^{m} a_i u_i(X) \sum_{i=0}^{m} a_i v_i(X) \\ = \sum_{i=0}^{m} a_i w_i(X) + h(X)t(X) \end{array} \right. \right\}$$

We assume $1^\lambda$ can be deduced from $R$. Comparing to the original NILP in Fig. 3, our NILP does not make use of $\gamma$. As we only need Eq (7), (8), (9) and (11) that do not contains $\gamma$ ($\gamma$ is only needed in the verification equation Eq. (10)).

$\underline{(\mathsf{p}_c, \mathsf{s}_c) \leftarrow \mathsf{KChallenge}(R)}$**:** Fix an LE scheme $\Sigma$ (with key pair $(\mathsf{pk}, \mathsf{sk}) \leftarrow \Sigma.\mathsf{KeyGen}(1^\lambda)$), run $(\boldsymbol{\sigma}_P, \boldsymbol{\sigma}_V) \leftarrow \mathsf{Setup}(R)$ to obtain $\boldsymbol{\sigma}_V = (\alpha, \beta, \delta, x)$ and generate $\{[\sigma_{P,i}, r_{P,i}] \leftarrow \Sigma.\mathsf{Enc}(\mathsf{pk}, \sigma_{P,i})\}$ and $[r_{P,i}] \leftarrow \Sigma.\mathsf{Enc}(\mathsf{pk}, r_{P,i})$ for each $\sigma_{P,i} \in \boldsymbol{\sigma}_P$ where

$$\boldsymbol{\sigma}_P = \alpha, \beta, \delta, \{x^i\}_{i=0}^{n-1}, \left\{\frac{\beta u_i(x) + \alpha v_i(x) + w_i(x)}{\delta}\right\}_{i=l+1}^{m}, \left\{\frac{x^i t(x)}{\delta}\right\}_{i=0}^{n-2}; \quad (13)$$

Return $\mathsf{p}_c = (\mathsf{pk}, \{[\sigma_{P,i}, r_{P,i}]_{i=1}^{y}\}, \{[r_{P,i}]\}_{i=1}^{y})$ and $\mathsf{s}_c = (\mathsf{sk}, \boldsymbol{\sigma}_V)$ .

$\underline{(\mathsf{p}_r, \mathsf{k}_r) \leftarrow \mathsf{KResponse}(\phi = \{a_i\}_{i=0}^{l}, \omega = \{a_i\}_{i=l+1}^{m}, R, \mathsf{p}_c)}$**:** Upon receiving the challenge $\mathsf{p}_c$,

1. Pick $r, s \leftarrow \mathbb{F}$;
2. Compute $[A]$, $[B]$, and $[C]$ (as well as $[r_2]$) using the affine functions in Fig. 3 (Eq. (7), (8) and (9)) on $\{[\sigma_{P,i}, r_{P,i}]_{i=1}^{y}\}$ (and $\{[r_{P,i}]\}_{i=1}^{y}$) with $\Sigma.\mathsf{Add}$;
3. Set $[\pi_{1,1}] = [A]$, $[\pi_{1,2}] = [B]$ and $[\pi_2, r_2] = [C]$;
4. Return $\mathsf{p}_r = ([\pi_{1,1}], [\pi_{1,2}], [r_2])$ and $\mathsf{k}_r = [\pi_2, r_2]$.

$\underline{\{\mathsf{k}_c, \perp\} \leftarrow \mathsf{KDerive}(R, \mathsf{s}_c, \phi, \mathsf{p}_r)}$ outputs $\perp$ if any verification fails:

1. Verify $\mathsf{ImgVer}(\mathsf{pk}, [\pi_{1,j}]) = 1$ for $j = \{1, 2\}$;
2. Verify $\mathsf{ImgVer}(\mathsf{pk}, [r_2]) = 1$;
3. Decrypt $A = \Sigma.\mathsf{Dec}(\mathsf{sk}, [\pi_{1,1}])$; and $B = \Sigma.\mathsf{Dec}(\mathsf{sk}, [\pi_{1,2}])$;
4. Decrypt $r_2 = \Sigma.\mathsf{Dec}(\mathsf{sk}, [r_2])$;
5. Compute $C$ as in Eq. (11) with $A$ and $B$;
6. Return $\mathsf{k}_r = \Sigma.\mathsf{Enc}(\mathsf{pk}, C, r_2)$ (using $r_2$ as randomness).

**Fig. 4.** Witness key agreement for QAP

A split DV NILP for QAP can be directly reformulated as in Fig. 3 by modifying the $\mathsf{Prove}$ algorithm. We simply split the proof matrices into two matrices $\boldsymbol{\Pi}_1$ and $\boldsymbol{\Pi}_2$ where $\boldsymbol{\Pi}_1 \in \mathbb{F}^{2 \times y}$ corresponds to the matrix used in Eq. (7) and (8) while $\boldsymbol{\Pi}_2 \in \mathbb{F}^{1 \times y}$ corresponds to the matrix used in Eq. (9). Since the NILP in Fig. 3 is secure (see Groth's security proof [27, Theorem 1]), our split DV NILP is also secure (see §D). We show in Fig. 4 how to construct $\Omega$ using a split DV NILP obtained from the NILP in Fig. 3.

**Theorem 2.** *If the LE scheme $\Sigma$ satisfies correctness, additive homomorphism, IND-CPA and linear-only homomorphism, then the construction in Fig. 4 yields a WKA scheme $\Omega$ that satisfies correctness, adaptive knowledge soundness, honest verifier zero-knowledge, response and key indistinguishability, and security against man-in-the-middle attack.*

**Table 3.** Theoretical Performance Evaluation

| Alg. | #Enc | #Dec | #Mult |
|---|---|---|---|
| KChallenge | $4(m-l+2n)$ | - | - |
| KResponse | - | - | $4(m-l+3n)$ |
| KDerive | 1 | $k$ | - |

$m$ is the number of variables in a QAP, $l$ is the number of instance variables, and $n-1$ is the degree of polynomials in the QAP. The number of decryption $k$ is construction dependent. In our case we have $k=3$.

## 7 Instantiation And Performance Evaluation

**Instantiation** We choose to instantiate the linear-only encryption scheme $\Sigma$ with a variant of the Paillier cryptosystem [41] similarly to Gennaro *et al.* [22] and Bitansky *et al.* [8] (see Fig. 8 in §E).

**Theoretical WKA Performance Evaluation** We can then estimate the theoretical performance of our WKA scheme $\Omega$ based on the number of encryptions, decryptions, and scalar multiplications for computing $\boldsymbol{\Pi}_1(\{[\sigma_{P,i}]\})$ and $\boldsymbol{\Pi}_2(\{[\sigma_{P,i}]\})$ (Table. 3). Let $m$ b the number of variables of a QAP, $l$ be the number of instance variables, and $n-1$ be the degree of polynomials of the QAP. The KChallenge algorithm requires the generation of $\{[\sigma_{P,i}]\}$ hence $m-l+2n$ encryptions on the investor's side. The KResponse algorithm requires only the proof computation on the trader's side which yields $m-l+3n$ scalar multiplications. The above numbers are doubled to fix the malleability of the scheme (see Fig. 8). It is then doubled again for computing the ciphertexts of the randomnesses. Finally the KDerive algorithm only requires $k$ decryptions and one encryption on the investor's side. The proof size is also only 6 Paillier ciphertexts.

**Baseline Performance** Paillier [41] is the main ingredient in our construction and its performance is well-studied in literature. Several optimization techniques were already present in the original paper [41], and Jost *et al.* [31] took a step further to improve the performance by orders of magnitude faster compared to a naïve implementation. For the timing of the Paillier encryption scheme we use the data from Table 4 by Jost *et al.* [31] *as an upper bound*[12] for the encryption time. The numbers were obtained on an Intel i7-4600U CPU at 2.10GHz with 4 cores running Ubuntu 64-bit v14.04. In particular, the reported result shows that, at 2048-bit key length, the encryption rate for 32-bit messages can reach 56K/s at the cost of 5.7s pre-computation time.

**Circuit Evaluation** We implement the relations SC, PR,MB and a new relation PR' which is the same as PR but with added check, e.g. $(p_1 < p < p_2) \vee (p_3 < p < p_4)$, in Table 1 as arithmetic circuits with the libsnark library [45] and measure the number of required variables $m$ and the corresponding degree of the polynomials $(n-1)$. Finally the runtime of KChallenge and KResponse, the most costly for 138-bit security for guessing $r$ [31], 2048-bit key length, using the 32-bit messages and the encryption rate as in Scheme 3 from Jost *et al.* [31].

[12] Benchmarked in 2015. As such, it provides a lower bound to our WKA performance

**Table 4.** Specific circuit evaluation

| **Relation** $R$ | $m$ | $n-1$ | $\mathcal{T}_C$ (s) | $\mathcal{T}_R$ (s) |
|---|---|---|---|---|
| SC | 25821 | 28312 | 5.8 | 7.8 |
| PR | 26080 | 28572 | 5.8 | 7.9 |
| PR' | 26598 | 29094 | 6 | 8 |
| MB | 51382 | 56361 | 11.6 | 15.6 |

We support 2048-bit key length and provide 112-bit security. Recall $m$ is the number of variables and $n-1$ is the degree of polynomials of the QAP. SC and PR are used for our dark pool simulation.

The evaluation of the new PR' relation and the MB relation illustrates the scalability of WKA. PR' consists of 1 consistency check for 1 commitment (1 private variable) and 4 arithmetic conditions with public variables, while MB consists of 2 consistency checks for 2 commitments (2 private variables). MB is in fact a building block for more general relation: $c' > p_1 \cdot v_1 + p_2 \cdot v_2 + p_3 \cdot p_3 + \ldots p_h \cdot v_h$. This is usable for both Multi-bids Auction and Biometrics Sharing (Hamming distance between two extracted features). This will require $2h$ commitments as it scales linearly with the number of private variables.

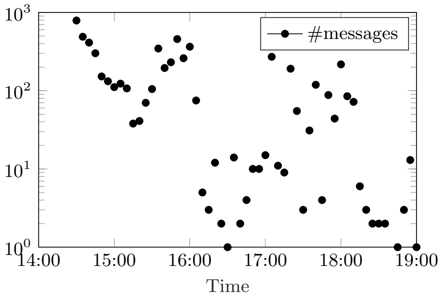
As shown in Table 4, the performances of SC, PR and PR' are close as their circuit complexity are similar to each other, as SC, PR and PR' require only one commitment consistency check while MB requires two of them. Hence, the runtime of MB is approximately double that of the others. KChallenge ($\mathcal{T}_C$) requires only 5.8s for the SC while PI takes only 5.9s. After the KChallenge, the key-agreement with KResponse ($\mathcal{T}_R$) takes only 7.8s for SC and 7.9s for PR. Even if we add 1s of one-way network latency into each message as we are employing an anonymous network (e.g. Tor) [17, Fig. 2]. The overhead of each WKA operation is lower than any known permission-less blockchain's block generation time (with Ethereum being the fastest at around 15s).[13] Hence each step can be fit within a single block generation time.[14]

**Dark Pools Simulation** For our simulation we make use of the Bloomberg Tradebook [9] for the period 13/03-1/5/2019 (35 trading days).

The Tradebook only contains the number of messages and the number of trades per day (see Fig. 5). Using WKA, an investor can setup a secure conversation including multiple messages which eventually lead to a trade. This means that the number of conversations (i.e. the truly necessary WKA executions) can be much smaller than the number of messages in Fig. 5. These conversations can also happen in parallel if they belong to different trades (or traders). From the available data we cannot know exactly *which messages belong to the same conversation*, or *how many conversations there are* and *the point of time at which*

[13] `https://ethstats.net/`.

[14] In our protocol, the blockchain is the actual bottleneck. Looking at Table 4, the runtime of each step (including setups) is less than the block time of the fastest permissionless blockchain (Ethereum roughly generates a block every 15s). Hence evaluating the interfaces of our scheme with the blockchain is equivalent to evaluating the blockchain itself. We should add that the current blockchain technologies is not adequate yet for high speed dark pools. Our major concern and main evaluation focus therefore is our scheme's crypto overhead.



For an actual trade it requires multiple messages. For a high day, the number of messages can reach 14103 (with 55 trades). For a low day, the number of messages is only 4514 (with 53 trades).

**Fig. 5.** Example of Tradebook messages and trades (May 1st, 2019)

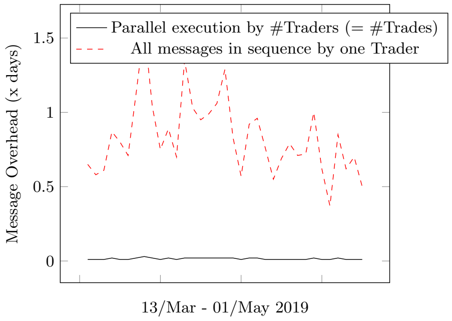
*they happened* as this is the whole point of a Dark Pool. We therefore considered the *worst possible case* where each message is a conversation by itself (almost always ending nowhere) and they are executed sequentially one after another by a single trader. We also considered a more plausible scenario *one trade-one trader* where each trade is done by a different trader and all messages of the day eventually belong to some trade.

We can combine the number of messages and trades from the extracted market data (examples shown in Fig. 5) and Table 4 to estimate the corresponding execution overhead throughout a day of trading. The final results are reported in Fig. 6. Performance is evaluated in terms of execution overhead to the expected processing time (1 day) as in a realistic setting using actual trading data is at least comparable on a day by day basis: if we were to run a day of trading messages, we would expect it to not take more than a day to actually exchange those messages. We combine the relations SC with PR and we consider the execution time of a message as the running time of SC's KChallenge (5.8s). For trades execution time we consider the sequential execution of KResponse from SC and the whole challenge and response time of PR (21.6s), adding the one-way delay of Tor (1s) per message. As shown in Fig. 6, even under worst possible assumption, only 7 days out of 35 days require more than 1 day of execution in our simulation. With a less extreme approach (solid line) the overhead is smaller than 10%.

## 8 Conclusion

We introduced the notion of witness-key-agreement. Specifically we defined split designated-verifier non-interactive linear proof following Groth's definition of NILP [27]. We then compiled the obtained split DV NILP into a Witness Key Agreement scheme using Linear-Only Encryption. Our obtained construction is efficient. After a one-time setup that yields a common challenge for a relation $R$ of interest, a party can agree on a secret key with another party given that the latter knows a witness of a committed instance.

Finally, our concrete WKA scheme for quadratic arithmetic programs yields both succinct communication complexity, i.e. the response to the common chal-



Assuming all computation done sequentially by 1 trader and messages are sent through Tor, only 7 out of 35 days of trading exhibits overheads greater than 1x in our simulation. With a more plausible scenario (each trade is done by a different trader) the overhead is at most 2%.

**Fig. 6.** WKA Evaluation on Bloomberg Tradebook

lenge consists of only 3 encrypted elements (6 Paillier ciphertexts), and efficient response computation and key derivation, i.e. only linear to the QAP size.

Our scheme is particularly suitable for private auctions in financial intermediation in which one party wants to privately communicate with another party about committed financial information which satisfies a relation $R$ of interest. It is also usable in other applications such as biometric-data sharing.

Our new notions, i.e. Witness-Key-Agreement and Split Designated Verifier NILP may be of independent research interest as well as interesting application of NILP.

## References

1. Abadi, M., Needham, R.: Prudent Engineering Practice for Cryptographic Protocols. IEEE Transactions on Software Engineering **22**(1), 6–15 (1996)
2. Abdolmaleki, B., Baghery, K., Lipmaa, H., Zajac, M.: A Subversion-Resistant SNARK. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 3–33. Springer (2017)
3. Abusalah, H., Fuchsbauer, G., Pietrzak, K.: Offline Witness Encryption. In: International Conference on Applied Cryptography and Network Security. pp. 285–303. Springer (2016)
4. Archer, D.W., Bogdanov, D., Pinkas, B., Pullonen, P.: Maturity and Performance of Programmable Secure Computation. IEEE security & privacy **14**(5), 48–56 (2016)
5. Bellare, M., Fuchsbauer, G., Scafuro, A.: NIZKs with an Untrusted CRS: Security in the face of Parameter Subversion. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 777–804. Springer (2016)
6. Bellare, M., Hoang, V.T.: Adaptive Witness Encryption and Asymmetric Password-Based Cryptography. In: IACR International Workshop on Public Key Cryptography. pp. 308–331. Springer (2015)
7. Bellovin, S.M., Merritt, M.: Encrypted Key Exchange: Password-based Protocols Secure Against Dictionary Attacks. In: 1992 IEEE Computer Society Symposium on Research in Security and Privacy. pp. 72–84. IEEE (1992)

8. Bitansky, N., Chiesa, A., Ishai, Y., Paneth, O., Ostrovsky, R.: Succinct Non-Interactive Arguments via Linear Interactive Proofs. In: Theory of Cryptography Conference. pp. 315–333. Springer (2013)
9. Bloomberg: Tradebook Bloomberg Professional Services. `https://www.bloomberg.com/professional/solution/tradebook/` (2019), accessed: 2019-05-01
10. Bogetoft, P., Christensen, D.L., Damgård, I., Geisler, M., Jakobsen, T., Krøigaard, M., Nielsen, J.D., Nielsen, J.B., Nielsen, K., Pagter, J., et al.: Secure Multiparty Computation Goes Live. In: International Conference on Financial Cryptography and Data Security. pp. 325–343. Springer (2009)
11. Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H.B., Patel, S., Ramage, D., Segal, A., Seth, K.: Practical secure aggregation for privacy-preserving machine learning. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. pp. 1175–1191. ACM (2017)
12. Camenisch, J., Casati, N., Groß, T., Shoup, V.: Credential Authenticated Identification and Key Exchange. In: International Cryptology Conference. pp. 255–276. Springer (2010)
13. Canetti, R.: Universally Composable Security: A New Paradigm for Cryptographic Protocols. In: 2001 IEEE International Conference on Cluster Computing. pp. 136–145. IEEE (2001)
14. Cartlidge, J., Smart, N.P., Alaoui, Y.T.: MPC Joins the Dark Side. Cryptology ePrint Archive, Report 2018/1045 (2018), `https://eprint.iacr.org/2018/1045`. Accessed: 2019-05-01
15. Chaum, D., Crépeau, C., Damgard, I.: Multiparty Unconditionally Secure Protocols. In: the twentieth ACM symposium on Theory of Computing. pp. 11–19. ACM (1988)
16. Derler, D., Slamanig, D.: Practical Witness Encryption for Algebraic Languages or How to Encrypt under Groth–Sahai Proofs. Designs, Codes and Cryptography **86**(11), 2525–2547 (2018)
17. Dhungel, P., Steiner, M., Rimac, I., Hilt, V., Ross, K.W.: Waiting for anonymity: Understanding delays in the tor overlay. In: 2010 IEEE Tenth International Conference on Peer-to-Peer Computing (P2P). pp. 1–4. IEEE (2010)
18. Fuchsbauer, G.: Subversion-Zero-Knowledge SNARKs. In: IACR International Workshop on Public Key Cryptography. pp. 315–347. Springer (2018)
19. Garg, S., Gentry, C., Halevi, S.: Candidate Multilinear Maps from Ideal Lattices. In: International Conference on the Theory and Applications of Cryptographic Techniques. pp. 1–17. Springer (2013)
20. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate Indistinguishability Obfuscation and Functional Encryption for all Circuits. SIAM Journal on Computing **45**(3), 882–929 (2016)
21. Garg, S., Gentry, C., Sahai, A., Waters, B.: Witness Encryption and Its Applications. In: 45th ACM symposium on Theory of Computing. pp. 467–476. ACM (2013)
22. Gennaro, R., Gentry, C., Parno, B., Raykova, M.: Quadratic Span Programs and Succinct NIZKs without PCPs. In: International Conference on the Theory and Applications of Cryptographic Techniques. pp. 626–645. Springer (2013)
23. Gentry, C., Lewko, A., Waters, B.: Witness Encryption from Instance Independent Assumptions. In: International Cryptology Conference. pp. 426–443. Springer (2014)

24. Gentry, C., Wichs, D.: Separating Succinct Non-Interactive Arguments from all Falsifiable Assumptions. In: 43rd ACM symposium on Theory of computing. pp. 99–108. ACM (2011)
25. Goldreich, O., Micali, S., Wigderson, A.: Proofs that Yield Nothing but Their Validity or All Languages In NP Have Zero-Knowledge Proof Systems. Journal of the ACM **38**(3), 690–728 (1991)
26. Goldwasser, S., Micali, S., Rackoff, C.: The Knowledge Complexity of Interactive Proof Systems. SIAM Journal on computing **18**(1), 186–208 (1989)
27. Groth, J.: On the size of Pairing-Based Non-Interactive Arguments. In: International Conference on the Theory and Applications of Cryptographic Techniques. pp. 305–326. Springer (2016)
28. Groth, J., Maller, M.: Snarky Signatures: Minimal Signatures of Knowledge from Simulation-Extractable SNARKs. In: International Cryptology Conference. pp. 581–612. Springer (2017)
29. Hamouda, F.B., Blazy, O., Chevalier, C., Pointcheval, D., Vergnaud, D.: Efficient UC-Secure Authenticated Key-Exchange for Algebraic Languages. In: International Workshop on Public Key Cryptography. pp. 272–291. Springer (2013)
30. Hazay, C., Scholl, P., Soria-Vazquez, E.: Low cost constant round mpc combining bmr and oblivious transfer. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 598–628. Springer (2017)
31. Jost, C., Lam, H., Maximov, A., Smeets, B.J.: Encryption Performance Improvements of the Paillier Cryptosystem. IACR Cryptology ePrint Archive **2015**, 864 (2015)
32. Kiayias, A., Tsiounis, Y., Yung, M.: Group Encryption. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 181–199. Springer (2007)
33. Kosba, A., Miller, A., Shi, E., Wen, Z., Papamanthou, C.: Hawk: The Blockchain Model of Cryptography and Privacy-Preserving Smart Contracts. In: 2016 IEEE symposium on security and privacy. pp. 839–858. IEEE (2016)
34. Kosba, A.E., Zhao, Z., Miller, A., Qian, Y., Chan, T.H.H., Papamanthou, C., Pass, R., Shelat, A., Shi, E.: How to Use SNARKs in Universally Composable Protocols. IACR Cryptology ePrint Archive **2015**, 1093 (2015)
35. Lindell, Y.: Secure Multiparty Computation for Privacy Preserving Data Mining. In: Encyclopedia of Data Warehousing and Mining, pp. 1005–1009. IGI Global (2005)
36. Malinova, K., Park, A., Riordan, R.: Do Retail Traders suffer from High Frequency Traders? (2013)
37. Markham, J.W.: Manipulation of Commodity Futures Prices-the Unprosecutable Crime. Yale Journal on Regulation **8**, 281 (1991)
38. Massacci, F., Ngo, C.N., Nie, J., Venturi, D., Williams, J.: The Seconomics (Security-Economics) Vulnerabilities of Decentralized Autonomous Organizations. In: Cambridge International Workshop on Security Protocols. pp. 171–179. Springer (2017)
39. Massacci, F., Ngo, C.N., Nie, J., Venturi, D., Williams, J.: FuturesMEX: Secure, Distributed Futures Market Exchange. In: 2018 IEEE Symposium on Security and Privacy. pp. 335–353. IEEE (2018)
40. Orlandi, C.: Is Multiparty Computation Any Good In Practice? In: 2011 IEEE International Conference on Acoustics, Speech and Signal Processing. pp. 5848–5851. IEEE (2011)

41. Paillier, P.: Public-Key Cryptosystems based on Composite Degree Residuosity Classes. In: International Conference on the Theory and Applications of Cryptographic Techniques. pp. 223–238. Springer (1999)
42. Rabin, T., Ben-Or, M.: Verifiable Secret Sharing and Multiparty Protocols with Honest Majority. In: the twenty-first ACM symposium on Theory of Computing. pp. 73–85. ACM (1989)
43. Sahai, A., Waters, B.: Fuzzy Identity-based Encryption. In: International Conference on the Theory and Applications of Cryptographic Techniques. pp. 457–473. Springer (2005)
44. Sasson, E.B., Chiesa, A., Garman, C., Green, M., Miers, I., Tromer, E., Virza, M.: ZeroCash: Decentralized Anonymous Payments from Bitcoin. In: 2014 IEEE Symposium on Security and Privacy. pp. 459–474. IEEE (2014)
45. SCIPR Lab: libsnark: a C++ library for zkSNARK proofs. https://github.com/scipr-lab/libsnark (2019), accessed: 2019-05-01
46. TheVerge: Data glitch sets tech company stock prices at USD 123.47. https://www.theverge.com/2017/7/3/15917950/nasdaq-nyse-stock-market-data-error, accessed: 2019-05-01

# A Reviews of Alternate Candidate Schemes

## A.1 Secure Multiparty Computation (MPC)

Any functionality can be securely realized by a distributed protocol assuming honest minority (in the computational setting) [15] and honest majority (in the information-theoretic setting) [42]. Recent advances in the implementations of generic MPC protocols [4] allow efficient MPC applications, e.g. to privacy-preserving data mining [35], and exchanges [39,14]. See also Orlandi [40] for an overview.

Secure Multiparty Computation (MPC) [15] could be a general solution. In particular there are three possible setups as follows.

**Full MPC** In the simplest setup, all parties in the systems run MPCs for all transactions. Firstly, setting up an MPC using existing distributed ledgers is not trivial as every party must be known in advanced or a PKI must be available in the setup phase for securing the communication over the ledger, e.g. as in [11]. Secondly, general MPC, even constant round protocols [30], often yield high round complexity which is unacceptable for protocols involving blockchain communication.[15]

Most importantly, this approach is not viable in practice as it requires all parties to participate into the processing of all transactions but it is unlikely that all of them are interested or have the capacity[16]. As an example, this solution would place an unacceptable burden on retail and institutional investors [39] (in most markets 90% of quotes come from 10% of the traders and investors [36]). Finally it is unclear whether general MPC will scale to the actual number of traders and investors ( thus it fails Proportional Burden, see Fig. 7 in [39]). A small OTC market might have tens or even a hundred investors.

**2-3 Servers MPC** An alternative setup is to replace the single trusted server by two or three servers running an MPC to intermediate all transactions. Parties only provide their secret inputs to the server. Cartlidge *et al.* [14] is an example of this approach. This setup clearly does not provide anonymity (which can be critical [38]) as, in the authors own words, it only focuses on "securing what and how much is being traded" rather than "on enabling anonymity of who is executing a trade". Secondly, to leverage on existing distributed financial systems we want to support communication over a distributed ledger where trader information is bound, e.g. its cash margin. The presence of a distributed ledger was not considered by Cartlidge *et al.* [14]. Further, the very idea of using 2-3 trusted servers is economically questionable: servers must be paid to offer such

[15] The state of the art MPC by Hazay *et al.* [30] yields 13 rounds.

[16] For example OTC and dark pools typically focus on large and specialized trades that few people can do but that may affect the market if known.

services and, once you need to pay and trust a server, you might as well use the existing ones and avoid crypto altogether.[17]

**Paired 2PC** As a last possible setup each pair of verifier and eligible-prover run a 2PC for each conversation (that may eventually lead to a transaction). This is more open than the previous two. However, to be anonymous until the deal is closed, the verifier may be unwilling to reveal herself by directly contacting all other parties, either randomly or round robin (thus it fails Proportional Burden). This implies that the setup and the execution of each 2PC must happen through the distributed ledger and this may significantly amplify the communication complexity of this approach as the verifier have to communicate a different garbled circuit for each other trader.

### A.2 Witness Encryption

(WE) was introduced by Garg *et al.* [21] and refined by Bellare and Hoang [6]. In a WE scheme defined for a NP language $L$ with witness relation $R(\phi, \omega)$, i.e. $L = \{\phi \mid \exists\ \omega : R(\phi, \omega) = 1\}$, the encryption algorithm takes as input a message $M$, an instance $\phi$ and produces a ciphertext $C$. Only a party with a witness $\omega$ such that $R(\phi, \omega) = 1$ can decrypt $C$ (correctness) and if $\phi \notin L$, the message $M$ is computationally hidden (soundness). Existing WE for arbitrary NP languages are currently considered impractical as they require multilinear maps [19,23] or Indistinguishability Obfuscation [20]. The improvement proposed by Abusalah *et al.* [3] moved the the computational hard part to an *offline* setup phase so that *online* encryption and decryption can be efficiently done but still relies on Indistinguishability Obfuscation. For some *particular* NP languages, WE is efficient[18]. For example Derler *et al.* [16] proposed an offline WE construction under a Groth-Sahai (GS) proof for algebraic languages defined over bilinear groups which can be employed for group encryption [32] and language-authenticated key exchange [29].

A verifier can use WE with the desired constraints on the committed information represented as a relation $R$, and only the provers who possess the witness $\omega$ for that instance $\phi$ such that $R(\phi, \omega) = 1$ can decrypt. Unfortunately, such constraints are usually in the form of an arithmetic relation, e.g. between the secret information of the provers and the public values from the verifier. This means we cannot use WE because general WE constructions [19,23,20,3] are impractical while practical WE under a Groth-Sahai (GS) proof [16] cannot support arithmetic relation of depth greater than 1 therefore even our very simple Matchable Bid constraint ($c \geq pv$, Table 1) would not be supported.

[17] Anonymity is compromised even with a single corrupted server. As the traders must communicate their (encrypted) inputs to each server, the servers always learn the sender of the messages (See [10]).

[18] An example is public key encryption: $\phi$ is the public key $\mathsf{pk}$ and $\omega$ is the private key $\mathsf{sk}$. Similarly for identity- and attribute-based encryption [43].

The Public Key Cryptosystem supports a tuple of standard key generation, encryption, and decryption PPT algorithms ($\mathsf{KeyGen}$, $\mathsf{Enc}$, $\mathsf{Dec}$) while the Zero-Knowledge Proof System supports a tuple of standard setup, proving, and verification PPT algorithms ($\mathsf{Setup}$, $\mathsf{Prove}$, $\mathsf{Verify}$).

$(\mathsf{p}_c, \mathsf{s}_c) \leftarrow \mathsf{KChallenge}(R)$ The verifier:

1. Generates the CRS for the Zero-Knowledge Proof for the relation $R$ of interest, i,e. $\boldsymbol{\sigma} \leftarrow \mathsf{Setup}(R)$;
2. Generates a public/private key pair, i.e. $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$;
3. Broadcasts $\mathsf{p}_c = (\boldsymbol{\sigma}, \mathsf{pk})$ and keeps secret $\mathsf{s}_c = (\mathsf{sk})$.

$(\mathsf{p}_r, \mathsf{k}_r) \leftarrow \mathsf{KResponse}(R, \mathsf{p}_c, \phi, \omega)$ The prover parses $\mathsf{p}_c = (\boldsymbol{\sigma}, \mathsf{pk})$ and:

1. Generates the proof, i.e. $\pi \leftarrow \mathsf{Prove}(R, \boldsymbol{\sigma}, \phi, \omega)$;
2. Samples a random key, i.e. $\mathsf{k}_r \leftarrow \{0,1\}^\lambda$;
3. Encrypts the proof and the random key together, i.e. $\mathsf{p}_r = \mathsf{Enc}(\mathsf{pk}, (\pi, \mathsf{k}_r))$;
4. Broadcasts $\mathsf{p}_r$ and keep secret $\mathsf{k}_r$.

$\{\mathsf{k}_c, \perp\} \leftarrow \mathsf{KDerive}(R, \mathsf{s}_c, \phi, \mathsf{p}_r)$ The verifier:

1. Decrypts and parses $(\pi, \mathsf{k}_r) = \mathsf{Dec}(\mathsf{sk}, \mathsf{p}_r)$;
2. Verifies $\{0,1\} = \mathsf{Verify}(R, \boldsymbol{\sigma}, \phi, \pi)$;
3. Set $\mathsf{k}_c = \mathsf{k}_r$ if the verification is successful;
4. Otherwise outputs $\perp$.

**Fig. 7.** Generic (but insecure) WKA Construction

### A.3 Authenticated Key Exchange (AKE)

AKE allows two parties to share a secret key over an insecure network using various authentication means. For example *Password-Authenticated Key Exchange* (PAKE) [7] allows two parties to agree on strong keys (in different sessions) if they both know a weak shared password. *Credential-Authenticated Key Exchange* (CAKE) [12] allows two parties to generate a common secret key if a specific relation is satisfied between credentials held by the two players. CAKE indeed can also be used to instantiate PAKE. However concrete instantiation of CAKE only supports limited relations such as vectored unions of product relations, equality testing or product relations [12, Section 6, 7 and 8]. *Language-Authenticated Key Exchange* (LAKE) is closely related to CAKE. It allows two parties to share a secret key if they hold credentials that belong to a specific algebraic language [29]. However our relations are *not* between credentials but between other objects unrelated to credentials (volumes, prices, etc.).

## B Generic (but Insecure) WKA Construction

One can think to have a trivial (and generic) construction of WKA utilizing Public Key Cryptosystem [41] and Zero-Knowledge Proof [27] as in Fig. 7.

The above construction satisfies all the security properties of WKA: Correctness follows the construction; Knowledge Soundness and Zero-KNowledgeness all follow the security properties of the used Zero-Knowledge Proof System; while Response Indistinguishability follows the indistinguishability under chosen plaintext attack (IND-CPA) of the Public Key Cryptosystem.

*Proof (Knowledge Soundness).* In the Soundness game, the assumption $\mathsf{k}_c = \mathsf{k}_r$ implies that $1 = \mathsf{Verify}(R, \boldsymbol{\sigma}, \phi, \pi)$; which implies $\pi$ is a valid proof of $\phi$; which implies that there must exists an extractor for $\omega$ such that $R(\phi, \omega) = 1$; which gives a contradiction to the assumption $R(\phi, \omega) \neq 1$.

*Proof (Zero-Knowledge).* We construct $\mathcal{S}_{ZK}$ as follows:

1. Generate $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$;
2. Generate $\boldsymbol{\sigma} \leftarrow \mathsf{Setup}(R)$;
3. Set $\mathsf{p}_c = (\boldsymbol{\sigma}, \mathsf{pk})$ and $\mathsf{s}_c = (\mathsf{sk})$;
4. Samples a random key, i.e. $\mathsf{k}_r \leftarrow \{0,1\}^\lambda$;
5. Set $\mathsf{k}_r = \mathsf{k}_c$;
6. Samples a valid proof using the Zero-Knowledge Simulator of the Zero-Knowledge Proof System, i.e. $\pi \leftarrow \mathcal{S}(R, \boldsymbol{\sigma}, \phi)$;
7. Encrypts $\mathsf{p}_r = \mathsf{Enc}(\mathsf{pk}, (\pi, \mathsf{k}_r))$;

*Proof (Response and Key Indistinguishability).* We construct $\mathcal{S}_{RI}$ as follows:

1. Generate $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)$;
2. Generate $\boldsymbol{\sigma} \leftarrow \mathsf{Setup}(R)$;
3. Set $\mathsf{p}_c = (\boldsymbol{\sigma}, \mathsf{pk})$ and $\mathsf{s}_c = (\mathsf{sk})$;
4. Samples a random key, i.e. $\mathsf{k}_r \leftarrow \{0,1\}^\lambda$;
5. Set $\mathsf{k}_r = \mathsf{k}_c$;
6. Samples a random proof, i.e. $\pi \leftarrow \{0,1\}^\lambda$;
7. Encrypts $\mathsf{p}_r = \mathsf{Enc}(\mathsf{pk}, (\pi, \mathsf{k}_r))$;

Yet, such a generic construction is susceptible to the Man In The Middle Attack (MITM). The MITM can simply intercept the $\mathsf{p}_c$ from the verifier, replace $\mathsf{pk}$ with the MITM's own $\mathsf{pk}'$ (with his own corresponding $\mathsf{sk}'$), then send $\mathsf{p}'_c = (\boldsymbol{\sigma}, \mathsf{pk}')$ to the prover[19] The prover will run $\mathsf{KResponse}$ but with $\mathsf{p}'_c$ instead of $\mathsf{p}_c$ which results in $\mathsf{p}'_r = \mathsf{Enc}(\mathsf{pk}', (\pi, \mathsf{k}'_r))$. Therefore the MITM, upon receiving $\mathsf{p}'_r$ can decrypt and see $(\pi, \mathsf{k}'_r) = \mathsf{Dec}(\mathsf{sk}', \mathsf{p}'_r)$. The MITM then can successfully relay the valid proof $\pi$ and agree on $\mathsf{k}_r$ by sending $\mathsf{p}_r = \mathsf{Enc}(\mathsf{pk}, (\pi, \mathsf{k}_r))$ to the verifier who will then set $\mathsf{k}_c = \mathsf{k}_r$ upon a successful verification.[20]

[19] Signing $\mathsf{p}_c$ will not help, since the MITM can always re-sign the new public challenge: no one knows who is the verifier due to the anonymity requirement.One can also argue that the prover can observe the blockchain to detect that there are two $\mathsf{p}_c$ and $\mathsf{p}'_c$ and abort the protocol. Yet, such a solution leads to a 100% successful DoS.

[20] It is possible that $\mathsf{k}_r = \mathsf{k}'_r$ but this is not important as the MITM has access to both.

## C Proof Sketch of Theorem 1

*Correctness* follows the algorithms' description. We focus on *adaptive knowledge soundness*, *honest verifier zero-knowledge*, *response and key indistinguishability*, and *security against MITM attack.*

For simplicity we only sketch the proofs as follows.

*Proof (Adaptive Knowledge Soundness).* In the soundness game adversary $\hat{\mathcal{A}}$ comes up with some proof $(\{[\pi_{1,j}]\}_{j=1}^{k-1}, [\pi_2]) = (\mathsf{p}_r, \mathsf{k}_r)$ for the instance $\phi$. Given $\mathsf{k}_c \leftarrow \mathsf{KDerive}(R, \mathsf{s}_c, \phi, \mathsf{p}_r)$, the assumption $\mathsf{k}_c = \mathsf{k}_r$ implies that the sampled proof $(\{[\pi_{1,j}]\}_{j=1}^{k-1}, [\pi_2])$ have passed the image verifications. If $(\{[\pi_{1,j}]\}_{j=1}^{k-1}, [\pi_2])$ are not affine combinations of $\{[\sigma_{P,i}]\}_{i=1}^{y}$, it means that $\hat{\mathcal{A}}$ has broken IND-CPA or linear-only homomorphism of $\Sigma$. Otherwise if $(\{[\pi_{1,j}]\}_{j=1}^{k-1}, [\pi_2])$ are affine combinations of $\{[\sigma_{P,i}]\}_{i=1}^{y}$, there must exists an extractor for some matrices $\boldsymbol{\mathit{\Pi}}_1$ and $\boldsymbol{\mathit{\Pi}}_2$ from $(\{[\pi_{1,j}]\}_{j=1}^{k-1}, [\pi_2])$ and $\{[\sigma_{P,i}]\}_{i=1}^{y}$ such that $\{[\pi_{1,j}, r_{1,j}]\}_{j=1}^{k-1} = \boldsymbol{\mathit{\Pi}}_1(\{[\sigma_{P,i}, r_{P,i}]\}_{i=1}^{y})$ and $[\pi_2, r_2] = \boldsymbol{\mathit{\Pi}}_2(\{[\sigma_{P,i}, r_{P,i}]\}_{i=1}^{y})$ unless $\hat{\mathcal{A}}$ has broken IND-CPA or linear-only homomorphism of $\Sigma$. Consequently, from the extractable matrices $\boldsymbol{\mathit{\Pi}}_1$ and $\boldsymbol{\mathit{\Pi}}_2$, as $\mathsf{k}_c = \mathsf{k}_r$ implies that $(\{\pi_{1,j}\}_{j=1}^{k-1}, \pi_2)$ is a valid proof for $R$ (as $(\{\pi_{1,j}\}_{j=1}^{k-1}, \pi_2)$ has passed the test $\mathbf{t}(\boldsymbol{\sigma}_V, \{\pi_{1,j}\}_{j=1}^{k}, \pi_2) = 0$, see Fig. **??**), there must exists an extractor for the witness $\omega$ s.t. $R(\phi, \omega) = 1$, otherwise $\hat{\mathcal{A}}$ has broken the statistical soundness property of the underlying split DV NILP (see Eq. **??**). Thus we conclude that $\Omega$ is adaptively knowledge sound.

*Proof (Honest Verifier Zero-knowledge).* To prove the honest verifier zero-knowledge property of $\Omega$, we show how to construct $\mathcal{S}_{ZK}$ from the underlying split DV NILP (Setup,Prove,Verify,Simulate):

1. Fix an LE scheme $\Sigma$;
2. Run $(\mathsf{pk}, \mathsf{sk}) \leftarrow \Sigma.\mathsf{KeyGen}(1^\lambda)$; and $(\boldsymbol{\sigma}_P, \boldsymbol{\sigma}_V) \leftarrow \mathsf{Setup}(R)$;
3. Encrypt $[\sigma_{P,i}, r_{P,i}] = \Sigma.\mathsf{Enc}(\mathsf{pk}, \sigma_{P,i})$ for each $\sigma_{P,i} \in \boldsymbol{\sigma}_P$;
4. Encrypt $[r_{P,i}] = \Sigma.\mathsf{Enc}(\mathsf{pk}, r_{P,i})$ for each $r_{P,i}$ used above;
5. Set $\mathsf{p}_c = (\mathsf{pk}, \{[\sigma_{P,i}, r_{P,i}]\}_{i=1}^{y}, \{[r_{P,i}]\}_{i=1}^{y})$ and $\mathsf{s}_c = (\mathsf{sk}, \boldsymbol{\sigma}_V)$.
6. Run $(\boldsymbol{\pi}_1, \boldsymbol{\pi}_2) \leftarrow \mathsf{Simulate}(R, \boldsymbol{\sigma}_V, \phi)$ where $\boldsymbol{\pi}_1 = \{\pi_{1,j}\}_{j=1}^{k-1}$ and $\boldsymbol{\pi}_2 = \{\pi_2\}$;
7. Encrypt $[\pi_{1,j}] \leftarrow \Sigma.\mathsf{Enc}(\mathsf{pk}, \pi_1^j)$ for each $\pi_{1,j} \in \boldsymbol{\pi}_1$;
8. Encrypt $[\pi_2, r_2] \leftarrow \Sigma.\mathsf{Enc}(\mathsf{pk}, \pi_2)$ and $[r_2] \leftarrow \Sigma.\mathsf{Enc}(\mathsf{pk}, r_2)$;
9. Set $\mathsf{p}_r = (\{[\pi_{1,j}]\}_{j=1}^{k}, [r_2])$ and $\mathsf{k}_r = [\pi_2]$.
10. Return $(\mathsf{s}_c, \mathsf{p}_c, \mathsf{p}_r, \mathsf{k}_r = [\pi_2], \mathsf{k}_c = [\pi_2])$.

The simulation and the real protocol only differs in Step 6 where the simulated proof $(\boldsymbol{\pi}_1, \boldsymbol{\pi}_2)$ is obtained. Due to the zero-knowledge property of the underlying split DV NILP, the simulated and the real proof are statistically indistinguishable. Hence the views of the adversary $\hat{\mathcal{A}}$ in the simulation and the real protocol are statistically indistinguishable. Hence $\Omega$ is honest verifier zero-knowledge.

*Proof (Response and Key Indistinguishability).* To prove response and key indistinguishability of $\Omega$, we show how to construct $\mathcal{S}_{RKI}$:

1. Randomly pick $(\boldsymbol{\Pi}_1, \boldsymbol{\Pi}_2) \leftarrow (\mathbb{F}^{k-1\times y}, \mathbb{F}^{1\times y})$;
2. Compute $\{[\pi_{1,j}, r_{1,j}]\}_{j=1}^{k-1} = \boldsymbol{\Pi}_1(\{[\sigma_{P,i}, r_{P,i}]\}_{i=1}^{y})$ (with $\Sigma.\mathsf{Add}$);
3. Compute $[\pi_2, r_2] = \boldsymbol{\Pi}_2(\{[\sigma_{P,i}, r_{P,i}]\}_{i=1}^{y})$ (with $\Sigma.\mathsf{Add}$);
4. Compute $[r_2] = \boldsymbol{\Pi}_2(\{[r_{P,i}]\}_{i=1}^{y})$ (with $\Sigma.\mathsf{Add}$);
5. Return $\mathsf{p}_r = (\{[\pi_{1,j}, r_{1,j}]\}_{j=1}^{k}, [r_2])$ and $\mathsf{k}_r = ([\pi_2, r_2])$.

The simulation and the real protocol is only different in Step 1 where in the simulation, instead of the valid proof matrices (as in the real protocol), $\mathcal{S}$ obtains the completely random matrices $(\boldsymbol{\Pi}_1, \boldsymbol{\Pi}_2)$. Since the adversary $\hat{\mathcal{A}}$ can only see the IND-CPA secure ciphertexts $(\mathsf{p}_c)$, the views of the adversary $\hat{\mathcal{A}}$ (without $\mathsf{s}_c$) in the simulation and the real protocol are computationally indistiguishable unless $\hat{\mathcal{A}}$ has broken the IND-CPA property of $\Sigma$. Thus we conclude that $\Omega$ satisfies Response Indistinguishability.

*Proof (Security against MITM).* In the MITM game adversary $\hat{\mathcal{A}}$ generates some proof $(\{[\pi_{1,j}]\}_{j=1}^{k-1}, [\pi_2]) = (\mathsf{p}_r, \mathsf{k}_r)$ for the instance $\phi$. Let us assume that $\mathsf{k}_c = \mathsf{k}'_r$.

We distinguish 2 cases: $(\mathsf{p}_r, \mathsf{k}_r) \neq (\mathsf{p}'_r, \mathsf{k}'_r)$ and $(\mathsf{p}_r, \mathsf{k}_r) = (\mathsf{p}'_r, \mathsf{k}'_r)$.

In the first case, $(\mathsf{p}'_r, \mathsf{k}'_r) \neq (\mathsf{p}_r, \mathsf{k}_r)$, following the same strategy of the Knowledge Soundness proof above, from $(\mathsf{p}'_r, \mathsf{k}'_r)$ there must exist an extractor for the witness $\omega'$ s.t. $R(\phi, \omega') = 1$. If $\omega' \neq \omega$, it contradicts with the assumption that $\omega$ is the witness of $\phi$ $(R(\phi, \omega) = 1)$. Otherwise if $\omega' = \omega$, $\hat{\mathcal{A}}$ has broken the zero-knowledge property of the underlying split DV NILP (as $\hat{\mathcal{A}}$ is able to distinguish the simulated and the real proof).

In the second case, $(\mathsf{p}'_r, \mathsf{k}'_r) = (\mathsf{p}_r, \mathsf{k}_r)$. First note that $(\mathsf{p}_r, \mathsf{k}_r) \leftarrow \mathsf{KResponse}(R, \mathsf{p}'_c, \phi, \omega)$ is run honestly, which implies $(\mathsf{p}_r, \mathsf{k}_r)$ are affine combinations of $\mathsf{p}'_c$. Given $\mathsf{k}_c \leftarrow \mathsf{KDerive}(R, \mathsf{s}_c, \phi, \mathsf{p}'_r = \mathsf{p}_r)$, the assumption $\mathsf{k}_c = \mathsf{k}'_r = \mathsf{k}_r$ implies that $(\mathsf{p}_r, \mathsf{k}_r)$ have passed the image verifications; which implies $(\mathsf{p}_r, \mathsf{k}_r)$ are affine combinations of $\{[\sigma_{P,i}]\}_{i=1}^{y}$; otherwise it means that $\hat{\mathcal{A}}$ has broken IND-CPA or linear-only homomorphism of $\Sigma$. Thus $(\mathsf{p}_r, \mathsf{k}_r)$ are affine combinations of $\mathsf{p}'_c$ implies that $(\mathsf{p}'_c, \mathsf{s}'_c) = (\mathsf{p}_c, \mathsf{s}_c)$; which implies that $\hat{\mathcal{A}}$ has broken IND-CPA of $\Sigma$.

Thus we conclude that $\Omega$ is secure against MITM.

## D Proof Sketch of Theorem 2

*Proof.* Since our NILP is reformulated from Groth's NILP in Fig. 3, it satisfies perfect completeness (straight forward to verify from the construction), perfect zero-knowledge (real proof computed in $\mathsf{Prove}$ and simulated proof computed in $\mathsf{Simulate}$ have uniformly random field elements $(A, B, C)$) and statistical knowledge soundness against affine prover strategies (for any affine prover strategy we can extract a witness with non-negligible probability). We refer the reader to Groth [27, Theorem 1] for additional details.

The correctness, additive homomorphism, IND-CPA and linear-only homomorphism of $\Sigma$ *and* the perfect completeness, perfect zero-knowledge and statistical knowledge soundness against affine prover strategies of the underlying split DV NILP implies that $\Omega$ satisfies correctness, adaptive knowledge soundness, honest verifier zero-knowledge, response and key indistinguishability, and security against man-in-the-middle attack. (Theorem 1).

## E The Two-Ciphertext Variant of Paillier

We summarize in Fig. 8 the two-ciphertext variant of Paillier similarly to Gennaro *et al.* [22] and Bitansky *et al.* [8].

The original *Scheme 3* of Paillier requires a multiplicative group $\mathbb{Z}^*_{N^2}$, for $N = pq$ where $p$ and $q$ are two prime numbers:

$\underline{(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\lambda)}$**:** runs as follows.

1. Select random primes $p$ and $q$ ($|p|, |q| \leq \frac{\lambda}{2}$),
2. Compute $N = pq$ and $\gamma = \mathsf{lcm}(p-1, q-1)$,
3. Randomly select $g$ where $g \in \mathbb{Z}^*_{N^2}$ and the order of $g$ is $\gamma N$.

Output public $\mathsf{pk} = (N, g)$ and keep secret $\mathsf{sk} = (p, q, \gamma)$.

$\underline{c \leftarrow \mathsf{Enc}(\mathsf{pk}, m)}$**:** Sample $r \in \mathbb{Z}_N$; output $c = g^{m+rN} \mod N^2$.

*Note 1:* Additional ciphertexts are required to adapt Paillier into linear-only encryption: the encryption of a message $m$ will output a pair of ciphertexts $c = \mathsf{Enc}(\mathsf{pk}, m)$ and $c' = \mathsf{Enc}(\mathsf{pk}, \theta m)$ (for some pre-defined secret parameter $\theta$).

*Note 2:* As the order of $g$ is $\gamma N$, there could be bias in the output distribution of $\mathsf{Enc}$ if $r \in \mathbb{Z}_N$ (this bias was present in the original Paillier's paper). To avoid this bias one could pick $r \in \mathbb{Z}_\gamma$. However this bias should be negligible as the attacker cannot distinguish between sampling in $N$ or $\phi(N)$. Furthermore, even though $\gamma$, the secret key, is not usually available to the party that runs $\mathsf{Enc}$, in our case, the investor knows $\gamma$ and is the only party supposed to run $\mathsf{Enc}$ she can pick $r \in \mathbb{Z}_\gamma$ and avoid the bias.

$\underline{\{0,1\} \leftarrow \mathsf{ImgVer}(\mathsf{sk}, \mathsf{pk}, c)}$**:** Output 1 iff $c \in \mathbb{Z}^*_{N^2} \wedge \gcd(c, N) = 1$. *Note:* $\mathsf{ImgVer}$ must also check the additional linear relation ($\mathsf{ImgVer}$ needs to decrypt $c$, $c'$ and check that they are consistent with regarding to $\theta$, i.e. $c' = \theta c$).

$\underline{m = \mathsf{Dec}(\mathsf{sk}, c)}$**:** Output $m = \frac{\mathsf{L}(c^\gamma \mod N^2)}{\mathsf{L}(g^\gamma \mod N^2)} \mod N$.

*Note:* The decryption of a ciphertext $[m]$ also means decrypting two ciphertexts $c = [m]$ and $c' = [\theta m]$.

$\underline{\hat{c} = \mathsf{Add}(\langle \alpha_i \rangle_{i=0}^n | \mathsf{pk}, \langle c_i \rangle_{i=0}^n)}$**:** Output $\hat{c} = \prod_{i=0}^n c_i^{\alpha_i} \mod N^2$.

The additive homomorphism is straight forward to verify as:

$$\prod_{i=0}^{n} (c_n^i)^{\alpha_i} = g^{\sum_{i=0}^{n} \alpha_i m_i + (\sum_{i=0}^{n} \alpha_i r_i)N} \mod N^2$$

**Fig. 8.** Scheme 3 of Paillier [41] and notes on its Two-Ciphertexts Variant [22,8]