Fair Distributed Exchange via Threshold Adaptor Signatures*

Ruben Baecker[1], Paul Gerhart[2], Jonathan Katz[†3], and Dominique Schröder[1,2]

[1] *Friedrich-Alexander-Universität Erlangen-Nürnberg*
[2] *TU Wien*
[3] *Google*

Abstract

Adaptor signatures enable scriptless protocols on blockchains such as Bitcoin by allowing a buyer to lock a coin based on an NP statement and release payment to any party that reveals the corresponding witness. They underlie a broad class of layer-2 protocols, including payment channels, cross-chain swaps, and coin-mixing services, which today secure billions of dollars in locked value. Fairness for existing adaptor-signature constructions, however, is inherently two-party, so does not apply to deployments in which the buyer or witness holder is operated by a committee.

In this work, we formalize and solve the fair exchange problem in that setting. We consider a buyer group holding shares of a signing key controlling a coin, and a seller group holding shares of a secret witness. The goal is to guarantee fairness both between groups (ensuring that each group obtains its asset if and only if the other does), as well as within groups (ensuring that all honest buyers learn the witness and all honest sellers receive payment), even in the presence of malicious insiders and/or partial information leakage.

We present efficient protocols achieving these guarantees under standard cryptographic assumptions on scriptless blockchains with timelocks. Our construction introduces two new primitives of independent interest, *threshold adaptor signatures* and *certified witness encryption*, which we define, analyze, and instantiate efficiently.

*This is the full version of the paper appearing at Asiacrypt 2026. A previous version of this work was first published as a preprint under the title *Fair Exchange for Decentralized Autonomous Organizations via Threshold Adaptor Signatures* [BGKS25]. This full version is a major revision of that preprint and supersedes it.

†Portions of this work were done while at the University of Maryland.

Contents

1 Introduction **1**
- 1.1 Our Contributions . . . 2
- 1.2 Related Work . . . 2

2 Technical Overview **4**
- 2.1 The Buyer Group . . . 5
- 2.2 The Seller Group . . . 6
- 2.3 Summary . . . 9

3 Preliminaries **9**

4 Fair Distributed Exchange **11**
- 4.1 Security Properties . . . 12

5 Threshold Adaptor Signatures **14**
- 5.1 Security of Threshold Adaptor Signatures . . . 16
- 5.2 Schnorr Threshold Adaptor Construction . . . 18

6 Certified Witness Encryption **23**

7 A Fair Distributed Exchange Protocol **27**

8 Impossibility Results **47**

A Additional Preliminaries **57**
- A.1 Hard Relations . . . 57
- A.2 Adaptor Signatures . . . 58
- A.3 Threshold Signatures . . . 58
- A.4 Non-interactive Arguments . . . 59
- A.5 (Verifiable) Public Key Encryption Schemes . . . 61
- A.6 Secret Sharing . . . 63

B A Framework for CWE **63**
- B.1 Certificate Extractable Signature Schemes . . . 67

1 Introduction

Payment-channel networks are the *de facto* scaling solution for Bitcoin, today securing billions of dollars in locked value [1ML26]. Their privacy-preserving variant, PTLC-based channels [MMS+19], achieves both scalability and payment unlinkability through a single cryptographic primitive, adaptor signatures [GSST24, Poe17]. At their core, adaptor signatures implement a fair coin-for-witness exchange. A buyer holding a signing key produces a pre-signature tied to a discrete-logarithm (DLog) statement $s = g^w$. Any party knowing the witness w can transform the pre-signature into a valid transaction signature. Once that signature appears on-chain, the witness can be efficiently extracted from it, giving the buyer its half of the exchange. This mechanism requires no smart-contract support and is directly deployable on widely used blockchains such as Bitcoin [Nak08] and Litecoin [The11]. Beyond payment channels, adaptor signatures underlie a broad class of layer-2 protocols including cross-chain swaps [TMM22] and coin mixing services [GMM+22].

All existing adaptor-signature constructions inherently enforce fairness only in the two-party setting, as both the buyer/signer and the witness holder are assumed to be individuals. In practice, however, the endpoints of these protocols are increasingly controlled by groups rather than individuals. Systems such as ARK [Ark26] distribute control across multiple operators, cross-chain bridges [TMM22] are governed by signing committees, and DAO treasuries are administered under joint custody [AJR22]. A single signing entity at each endpoint creates a central point of failure and concentrates trust in ways that contradict the decentralization goals of the underlying blockchain. The natural remedy is to distribute each endpoint according to a threshold policy, yet no existing adaptor-based protocol supports this. As a consequence, privacy-preserving threshold-controlled payment channels, bridges, and joint-custody trade protocols remain out of reach on minimal blockchains.

From two parties to two groups. At its core, the threshold-endpoint problem is one of fair exchange between two groups. We refer to the group of signers as the *buyer group* and the group of witness holders as the *seller group*, each governed by its own threshold policy (t_{B}-out-of-n_{B} for the buyers and t_{S}-out-of-n_{S} for the sellers). This abstraction captures threshold PTLC endpoints, cross-chain bridge committees, DAO treasuries, and consortium-controlled custody alike. Since the fair exchange of a DLog witness serves as a compiler for arbitrary digital assets [TSZ+24, TGT26], our results extend naturally to trading coins for other assets as well.

Realizing this multi-party generalization, however, is far from straightforward. In the two-party case, adaptor signatures provide fairness essentially for free, since the witness-extraction property atomically binds a timelocked payment to delivery [GSST24]. Distributing both roles across groups introduces new attack vectors by malicious insiders. On the buyer side, a malicious buyer can collect partial pre-signatures from honest co-signers, abort without contributing her own share, and privately forward the completed pre-signature to the seller group, leaving honest buyers unable to extract the witness despite having committed their funds. On the seller side, a corrupted seller that learns the witness first can race to claim payment alone, leaving honest sellers unpaid. It is therefore unclear whether scriptless blockchains can support fair coin-for-witness exchange at all in this setting. We ask:

> *Can we lift the scriptless coin-for-witness exchange framework to a multi-party threshold setting, while achieving fairness for all participants?*

In this work, we answer this question affirmatively and identify three security requirements that any such protocol must satisfy:

Buyer fairness: An adversary controlling all sellers and strictly fewer than t_B buyers cannot obtain honest buyers' funds unless all of them learn the witness.

Seller fairness: An adversary controlling all buyers and strictly fewer than t_S sellers cannot learn anything about the witness unless all honest sellers receive their expected funds.

Witness privacy: An adversary corrupting strictly fewer than t_S sellers learns nothing about the witness w, even if sellers receive payment. (Since every buyer is supposed to learn w upon successful completion of the protocol, here we must assume no buyers are corrupted.)

1.1 Our Contributions

At a high level, our contributions are as follows (see Section 2 for details).

Formalization. We introduce the first formal framework for multi-party fair exchange in the scriptless setting, extending prior notions of two-party fair exchange to the threshold case.

Feasibility. We present efficient protocols on scriptless blockchains with timelocks that achieve the above properties under standard cryptographic assumptions. Our protocols achieve buyer fairness for any threshold $t_{\mathsf{B}} \leq n_{\mathsf{B}}$. They achieve seller fairness provided that either the sellers put up deposits or $2 \cdot t_{\mathsf{S}} - 1$ sellers participate, i.e., a majority of sellers are honest.

Impossibility of fair exchange. We prove that, without timelocks, seller fairness is unachievable on scriptless blockchains, and that, even with timelocks, seller fairness requires either deposits from the sellers or $2 \cdot t_{\mathsf{S}} - 1$ participating sellers. This demonstrates the optimality of our protocols.

Building blocks. Along the way, we introduce two new cryptographic tools.

- *Threshold adaptor signatures* extend adaptor signatures to the threshold setting while remaining compatible with single-party adaptor signatures.
- *Certified witness encryption* (CWE) allows encryption of a payload relative to a verification-key/message pair (vk, m) such that decryption requires a valid signature on m under vk. This is useful for fairness as it can ensure that when a transaction signature appears on-chain, an encryption of a witness can be decrypted. We instantiate this primitive for a broad class of DLog-based signature schemes including Schnorr signatures [Sch91, BBS04, CL03, BSW06, KW03].

1.2 Related Work

Most related to our considered fair exchange problem between two groups are protocols based on adaptor signatures [Poe17, AEE+21, EFH+21, DOY22, GSST24, GRS26, GTT26, RST26]. Adaptor signatures provide a mechanism for the fair exchange of a witness and a payment between two parties in a blockchain setting. Our work can be viewed as extending that work to the case of two *groups* of parties with a threshold requirement. While it is natural to replace the adaptor signature with

a threshold adaptor signature to achieve fairness *between* sellers and buyers, we stress that adaptor signatures provide no fairness guarantees *amongst* buyers or *amongst* sellers.

(Threshold) adaptor signatures allow witness extraction given both a pre-signature and its corresponding adapted signature. While the latter is publicly observable on the blockchain once a payment happens, ensuring that all buyers receive the initial pre-signature requires a fair multi-party computation. Independent work by Ji et al. [JXGZ24] and Kajita et al. [KOT24] also explore n-party threshold adaptor signatures. Their formulations, however, rely on prior definitions that have been shown to be unsuitable in practice [GSST24], and the latter work considers only the n-out-of-n case.

Fair MPC. Fair multi-party computation is achievable under various assumptions. One line of work for fair MPC [CGJ+17, KGM19] relies on the blockchain to provide unpredictable proofs of publication that remain unknown to the protocol participants until the moment of inclusion. Under this assumption, the parties first run an MPC that may be unfair, returning encrypted outputs. Then, the parties encrypt their decryption shares using witness encryption, such that once the ciphertext is posted on the blockchain, and the blockchain publishes the proof of publication, each party can decrypt the other's decryption keys simultaneously. In practice, only blockchains with smart contracts can provide a proof of publication that cannot be anticipated by any party executing the protocol.

Trusted hardware. In a related direction, [CGJ+17, GKRS23] achieve fair MPC using blockchains and trusted execution environments such as Intel SGX. While these results allow general fair MPC, we show that computing adaptor pre-signatures is feasible under standard cryptographic assumptions without trusted hardware.

Rational adversaries. Other lines of work aim to achieve fair MPC by weakening the adversarial model. For example, fair MPC with penalties [ADMM14a, BK14, KMB15, BDD20, ADMM14b] ensures that parties who attempt to violate fairness are penalized by losing funds. At the end of a fair MPC protocol with penalties, each party either receives the function's output or is compensated financially. Enforcing penalties typically requires either an honest majority of participants or specialized transaction logic, such as smart contracts. Approaches combining penalties and smart contracts appear in [KZZ16, EFS20, DEF18]. A major drawback of penalty-based approaches is that the required collateral must scale with the value of the witness. If the witness is expensive, each seller must also lock up a correspondingly large amount of collateral, which significantly limits the protocol's practicality. In addition, we aim for a protocol that is secure against malicious adversaries without limiting the adversarial corruption capabilities on the buyer side. Our first settle protocol is closely related to the ladder of deposits of [BK14], but it differs from that line in three ways. First, we do not use deposits as a penalty mechanism but only to guarantee each seller that it can eventually sell its witness, so our protocol safeguards against fully malicious adversaries and is not limited to rational behavior. Second, since we use adaptor signatures, our protocol does not require scripts from the blockchain, and redeeming a deposit already publishes the value that opens the next one. Third, the sellers re-share the witness additively before the ladder starts, as the ladder works only in an n_{S}-out-of-n_{S} setting.

Witness encryption. We propose certified witness encryption to resolve the intra-buyer fairness problem. Certified witness encryption is a subclass of general witness encryption [GGSW13, GHK+25] that can be more efficient because it does not need to support arbitrary NP statements.

Closely related to certified witness encryption is signature-based witness encryption (SWE) [DHMW23]. In SWE a ciphertext can be encrypted w.r.t. a message and set of verification keys, and a threshold access structure on these keys. Such a ciphertext can be decrypted using an aggregated signature that is valid on the message m and the aggregation satisfies the access structure. CWE is a more minimal formalization compared to SWE. Instead of relying on an aggregatable signature scheme and coding a threshold access structure inside the encryption and decryption, CWE relies on ordinary signature schemes without access structures. In addition, SWE was built specifically with BLS in mind (for which adaptor signatures cannot exist [EFH+21]), while our focus for CWE was blockchains that support ordinary signature schemes. Our main contribution for CWE is not the formalization but the construction based on Schnorr signatures.

Another notable notion are witness signatures [GJK15]. The allow a party holding a witness for an NP statement to produce a signature for that statement. In our setting, however, sellers know the witness (while buyers do not), and buyers cannot trust sellers to behave honestly, so witness signatures are not applicable in our setting.

Timelocks. We also want to achieve fairness among the sellers. Approaches based on fair MPC are insufficient in this setting due to the presence of frontrunning adversaries (cf. Section 8). To avoid this kind of attack, we assume a scriptless blockchain with timelocks. Adding timelocks to avoid frontrunning is also done in [HLTW24, TMM21, GMM+22].

Sparkle. Our construction builds on the Sparkle threshold signature scheme [CKM23], whose initial analysis contained a gap identified by [BLT+24]. Acharya et al. [AFOS26] closed this gap by giving the first proof of static security for the original scheme, which suffices for our results, as we only require unforgeability against static corruptions.

2 Technical Overview

As discussed in the introduction, existing adaptor-signature constructions [Poe17, GSST24] enable fair exchange when each group consists of a single party. A buyer holding a signing key generates a pre-signature $\widetilde{\sigma}_{\mathsf{coin}}$ tied to an NP statement s with respect to a relation $\mathcal{R}$. Only someone who knows a witness w with $(s, w) \in \mathcal{R}$ can extend it to a valid signature. The buyer sends this pre-signature to the seller, who uses their knowledge of w to extend the pre-signature to a full signature σ on the transaction, and posts the resulting signature to the blockchain, thereby receiving payment. A crucial property of adaptor signatures is that, given a pre-signature $\widetilde{\sigma}$, a corresponding full signature σ, and the statement s, one can efficiently extract the witness w. Once σ is posted on-chain, the buyer therefore recovers w. Our goal is to extend this idea from two parties to a seller group and a buyer group, each with its own threshold requirement for executing the exchange. We discuss first how to handle a group of buyers, then a group of sellers, and finally discuss the remaining technical challenges.

2.1 The Buyer Group

To handle a group of buyers, we propose using *threshold* adaptor signatures. Achieving full fairness in our setting, however, also requires additional cryptographic tools as we will discuss.

Threshold Adaptor Signatures. We introduce a threshold variant of adaptor signatures that allows an authorized set of signers to jointly compute a pre-signature. To maintain compatibility with existing adaptor-signature-based protocols, the resulting pre-signature must be indistinguishable from a single-user pre-signature. Our thresholding scheme preserves the core functionality of classical adaptor signatures: (i) given a pre-signature $\widetilde{\sigma}$ and a witness w, one can derive a full signature σ (adaptation), and (ii) given a pre-signature $\widetilde{\sigma}$, a signature σ, and a statement s, one can extract the witness (extraction).

We base our construction on Sparkle [CKM23], a Schnorr threshold signature scheme with the following signing procedure: (i) Commitment Round. Each signer samples a random value $r_i \leftarrow\$ \mathbb{Z}_p$, computes the partial nonce $R_i = g^{r_i}$, commits to it using a hash function H_{cm}, and shares the commitment with the other signers. (ii) Opening Round. After receiving commitments from all participants, each signer reveals R_i. Assuming the revealed nonces are consistent with their respective commitments, parties then compute the aggregated nonce as $R = \prod R_i$. (iii) Partial Signature Computation. Each signer computes a partial signature $z_i = r_i + h \cdot \lambda_i \cdot \mathsf{sk}_i$, where $h = \mathsf{H}_{\mathsf{Sign}}(\mathsf{pk}, m, R)$ is the Schnorr challenge and λ_i the Lagrange coefficient of signer i. (iv) Aggregation. The partial values aggregate to $\widetilde{\sigma} = (R, z)$ with $z = \sum_i z_i = r + h \cdot \mathsf{sk}$ for the combined randomness r and the aggregated key sk.

To convert Sparkle into a threshold adaptor signature scheme for the DLog relation $\mathcal{R} = \{(s, w) \mid s = g^w\}$, we add a pre-sign protocol that is related to the sign protocol. The main difference is that in the pre-signing protocol, we modify the nonce to be $R \cdot s = g^{r+w}$ instead of $R = g^r$. This also changes the Schnorr challenge to $h = \mathsf{H}_{\mathsf{Sign}}(\mathsf{pk}, m, R \cdot s)$ (compared to $h = \mathsf{H}_{\mathsf{Sign}}(\mathsf{pk}, m, R)$ in the signing protocol of Sparkle). Since single-party Schnorr adaptor signatures also compute the challenge on the shifted nonce, the pre-signature has the same form as theirs, i.e., $\widetilde{\sigma} = (R \cdot s, z)$ with $z = r + h \cdot \mathsf{sk}$. Hence, one can use the same adapt and extract algorithms [AEE+21].

Certified Witness Encryption (CWE). Threshold adaptor signatures ensure fairness between a set of honest buyers and a single seller, but they do not inherently guarantee fairness *among* the buyers themselves. In particular, a malicious buyer could abort the pre-signing protocol after collecting the necessary partial pre-signatures from other buyers without revealing her own pre-signature, which would allow the adversary to compute a pre-signature $\widetilde{\sigma}$ while the remaining buyers are unable to do so. The malicious buyer can forward $\widetilde{\sigma}$ to the (possibly malicious) seller, who can convert it into a valid signature σ_{coin}. Once it posts σ_{coin} on the blockchain and receives payment, the malicious buyer can extract the witness using $\widetilde{\sigma}$. All honest buyers have now paid but cannot extract the witness w from σ_{coin} because they do not know $\widetilde{\sigma}$.

We address this "intra-buyer" fairness problem by introducing a primitive called *certified witness encryption*. A CWE scheme encrypts relative to a verification key vk and a designated message m, such that only parties knowing a valid signature on m under vk can decrypt. Using a CWE scheme, we achieve fairness for all honest buyers. Before disclosing its partial pre-signatures, each buyer encrypts them using the CWE scheme and forwards the resulting ciphertext, along with a proof of well-formedness, to all other buyers. Such a ciphertext can be decrypted by anyone who knows σ_{coin}, which eventually appears on-chain. After receiving and verifying enough CWE ciphertexts,

each honest buyer forwards its partial pre-signature to the other buyers. Once the seller posts σ_{coin} to receive its payment, every buyer decrypts the CWE ciphertexts it received, reconstructs the pre-signature from the decrypted partial pre-signatures, and extracts the witness using σ_{coin}.

We construct a CWE scheme for Schnorr signatures to ensure compatibility with our Schnorr threshold adaptor signature scheme and to stay compatible with blockchains like Bitcoin or Litecoin. We build our CWE scheme from the Schnorr verification equation. Given a Schnorr signature $\sigma = (R, z) = (R, \mathsf{sk} \cdot h + r)$, where $h = \mathsf{H}_{\mathsf{Sign}}(\mathsf{pk}, R, m)$, verification is performed by checking the equation $g^z = \mathsf{pk}^h \cdot R$. The RHS of the equation can be computed publicly from the nonce R, the public key pk, and the message m. Computing the LHS of the equation requires the discrete logarithm $z = \mathsf{sk} \cdot h + r$, which is feasible only for someone who either knows the secret key sk and the discrete logarithm r of R (typically, the signer), or learns the signature (R, z). We use this discrete logarithm relation to build encryption and decryption keys for our CWE scheme. To encrypt a message m_{Enc} that can be decrypted knowing a Schnorr signature valid for (pk, R, m), a party computes the encryption key $\mathsf{ek} = \mathsf{pk}^h \cdot R$ and constructs the ciphertext $ct = (g^y, \mathsf{ek}^y \cdot m_{\mathsf{Enc}})$, where $y \in \mathbb{Z}_p$ is a uniformly sampled random value. This corresponds to an ElGamal ciphertext under the public key ek. Given a valid signature (R, z) and the ciphertext ct, decryption follows standard ElGamal decryption using the decryption key $\mathsf{dk} = z$.[1] Following this blueprint, we extend CWE to a broad class of signature schemes, including Camenisch-Lysyanskaya (CL) [CL03], Waters+ [BSW06], and Katz-Wang [KW03] (cf. Appendix B).

2.2 The Seller Group

At first glance, handling multiple sellers seems symmetric to the case of multiple buyers. Consequently, one might expect that similar techniques could be used to solve seller fairness.[2] However, it turns out that the problem is fundamentally different. To illustrate this issue, consider a scenario where multiple sellers each possess a share of the witness w, and assume that the buyers have already provided a valid pre-signature $\widetilde{\sigma}$. A natural approach would be to execute a protocol among the sellers to reconstruct the witness w. Given the properties of threshold adaptor signatures, once the sellers obtain w and have a pre-signature $\widetilde{\sigma}$, they can efficiently compute the signature σ_{coin} and post it on the blockchain, which authorizes the payment to the sellers. We show next why this approach fails. In fact, we prove a broad impossibility result showing that seller fairness is impossible to achieve without making additional assumptions.

Impossibility of seller fairness with scriptless blockchains. Seller fairness requires that an adversary statically corrupting all buyers and up to $t_{\mathsf{S}} - 1$ sellers learns nothing about w unless all honest sellers are paid.

Now consider an adversary that corrupts all buyers and no seller at all. In any fair exchange protocol, an honest seller is the first to issue a message that allows the buyers to compute the witness. Initially, the buyers cannot compute w, but buyer fairness requires that they can do so once their signature is posted on-chain. They must therefore receive a message from a seller that lets them compute w before or when their payment takes effect. The adversary has the buyers follow the protocol until they receive this message and then computes w from their joint state. Even if the

[1] Since the partial pre-signatures that we encrypt are elements of $\mathbb{Z}_p$ rather than group elements, our protocol uses hashed ElGamal instead, i.e., $ct = (g^y, \mathsf{H}_{\mathsf{Enc}}(\mathsf{ek}^y) + m_{\mathsf{Enc}})$ for a hash function $\mathsf{H}_{\mathsf{Enc}}$ into $\mathbb{Z}_p$ (cf. Section 6).

[2] For $t_{\mathsf{S}} = 1$ the adversary in the seller fairness experiment corrupts no seller, so the exchange reduces to the standard adaptor-signature fair exchange. Thus, we consider $t_{\mathsf{S}} > 1$ for our constructions.

message is a transaction request, the adversary can process it after its own requests to spend the buyers' UTXOs, since it controls the order of requests within each round and no timelocks prevent these spends. With this strategy, even if the sellers obtain a valid signature from the buyers, the coin that the signature authorizes has already been spent. This frontrunning attack shows that seller fairness cannot be achieved in a blockchain setting without scripting support. In fact, it is not limited to the case $t_{\mathsf{S}} > 1$, but holds also in the single-party adaptor-signature exchange. Our first impossibility result in Section 8 formalizes this limitation.

Achieving seller fairness with timelocks. Since frontrunning rules out scriptless blockchains, we turn to blockchains with timelocks. A timelock allows buyers to reserve funds for a specific transaction, which can either be completed with a valid signature or be refunded after a timeout. The funds stay locked to the intended payment, so an adversarial buyer cannot divert them before the honest sellers finish the exchange.

As a starting point, consider two sellers holding shares w_1 and w_2 of the witness, which they sell for a price of one coin. The sellers can achieve a fair protocol as follows. First, the buyers pre-sign a transaction that pays the coin to seller 2 only upon revealing the full witness w, and they secure this payment with a timelock. If this timelock has sufficient duration, seller 2 knows that it is paid whenever it can produce a valid witness. Seller 2 therefore attempts to buy the witness share from seller 1 before the timelock of the buyers expires. To do this, it runs a standard single-party adaptor-signature exchange paying seller 1 half a coin if and only if seller 1 reveals its witness share to seller 2. Once seller 2 possesses both witness shares, it adapts the buyers' pre-signature and is paid the remaining coin. This protocol is fair for seller 1, as it receives half a coin if and only if it reveals its witness share. It is also fair for seller 2, as it pays seller 1 half a coin and receives a coin from the buyers, resulting in a win of also half a coin. If the exchange with seller 1 fails, seller 2 invalidates the pre-signed transaction, and no exchange happens.

We can extend the same idea to more than two sellers and to any threshold $t_{\mathsf{S}} \leq n_{\mathsf{S}}$. The sellers first convert their threshold shares into an additive sharing $b_1, \ldots, b_{n_{\mathsf{S}}}$ of the witness, i.e., $w = \sum_k b_k$, where seller k holds b_k. Seller 1 first sells b_1 to seller 2. Seller 2 adds its own value and sells the sum $b_1 + b_2$ to seller 3, and so on. Eventually, the last seller in line holds the whole witness and can redeem the transaction of the buyers. To ensure fairness, the timelocks increase along the line, such that seller i can exchange its learned witnesses with seller $i+1$ after obtaining the share from seller $i-1$. In addition, the pre-signatures (that guarantee seller i a payment from seller $i+1$, once it learned the shares of seller $i-1$) travel in the opposite direction, meaning that seller i forwards its pre-signature to seller $i-1$ only after receiving one from seller $i+1$. Similar to the two-seller setting, under this protocol, each seller is guaranteed that, after purchasing the witnesses from the previous seller, it can sell its shares to the next seller. We formalize this protocol in Figure 14.

While this protocol achieves fairness for sellers, it has the limitation that the sellers must post $2 \cdot (n_{\mathsf{S}} - 1)$ on-chain transactions, incurring transaction fees. In addition, the sellers have to own and lock funds in a linear amount relative to the value of the exchanged witness, which may be unrealistic depending on the use case. Finally, a single malicious seller can stall the protocol by refusing to sell its learned share. We therefore seek a protocol that does not require deposits.

Achieving seller fairness without deposits. While seeking such a protocol, we obtain a second impossibility result. As we formally show in Section 8, without deposits, a fair distributed exchange

protocol exists only if an honest majority of sellers participates. The intuitive argument is as follows.

If the sellers make no deposits, the only coins in the exchange are the buyers' coins. In addition, to achieve buyer fairness, the buyers lock a single payment that is released only against the full witness. The payment *cannot* be split across individual shares because the buyers do not trust the sellers, and a set of malicious sellers could otherwise collect the payments for $t_{\mathsf{S}} - 1$ witness shares and stop. The buyers would then have paid for shares that do not suffice to reconstruct the witness, which violates buyer fairness. An honest seller can therefore receive its payment only once the buyers' signature takes effect and irrevocably decreases their balance. Now let us assume that the adversary corrupts the buyers together with $t_{\mathsf{S}} - 1$ sellers, where, as before, any t_{S} sellers can reconstruct the witness. If fewer than $2 \cdot t_{\mathsf{S}} - 1$ sellers participate, there are at most $t_{\mathsf{S}} - 1$ sellers outside the malicious coalition. Hence, we can split the sellers into two sets of at most $t_{\mathsf{S}} - 1$ sellers, and each set forms, together with all buyers, a coalition that the adversary can corrupt.

For each of these two coalitions, there is a point after which it can reconstruct the witness from the values it has received, since the coalition contains all buyers, who eventually learn the witness. We let the adversary guess a bit indicating which coalition reaches its point first, and corrupt this coalition. The guess is correct with probability 1/2. After that point, the adversary reconstructs the witness and stops interacting with honest parties. Additionally, no honest seller can make the payment of the buyers take effect after this point. Otherwise, the other coalition, i.e., the honest sellers and all buyers, could compute this payment from its values before that point and extract the witness from it by buyer fairness. This coalition would thus have reached its point even earlier, which contradicts our choice. Consequently, the adversary obtained the witness, and no honest seller received their payment, violating seller fairness. We formalize this argument in Section 8.

For $n_{\mathsf{S}} \geq 2 \cdot t_{\mathsf{S}} - 1$, this argument no longer applies. There are then at least t_{S} sellers outside the coalition, so the honest sellers and the buyers no longer form a coalition that the adversary could corrupt, and the contradiction disappears. Our second protocol exploits precisely this gap and achieves seller fairness without deposits. As before, the protocol begins with the buyers locking their coins on the blockchain and generating a pre-signature. This time, however, the pre-signed transaction pays all the sellers. The buyers send this pre-signature to the sellers. Each seller then sends its witness share to all other sellers over private channels. With at least $2 \cdot t_{\mathsf{S}} - 1$ sellers participating, at least one honest seller will collect enough shares to reconstruct the witness before the lock expires. This seller adapts the pre-signature with the witness into a valid signature σ_{coin} and publishes it to the blockchain to complete the payment. We formalize this protocol in Figure 15.

Witness privacy. The seller-side protocols above require the sellers to reconstruct the witness in order to adapt the pre-signature. Since this would reveal the witness to the sellers, we instead let them work with a blinded witness. The buyers jointly choose a random blinding value, blind the statement with it, and send encrypted shares of the blinding value to the sellers, who use them to blind their witness shares. The sellers can then run the settle protocol on the blinded witness without learning the original witness. Once the exchange is complete, the buyers extract the blinded witness from the blockchain and remove the blinding locally. Thus, the sellers only learn the blinded witness, while the buyers recover the original witness. This provides witness privacy as long as fewer than t_{S} sellers are corrupted and the buyers remain honest.

2.3 Summary

In summary, our positive results show that (i) buyer fairness can be achieved for up to $t_{\mathsf{B}} - 1$ malicious buyers and only malicious sellers in the presence of a scriptless blockchain. (ii) Seller fairness can be achieved on blockchains that support timelocks, either for each t_{S} with deposits of the sellers, or without deposits when $2 \cdot t_{\mathsf{S}} - 1$ sellers participate in the exchange. (iii) Witness privacy is achieved as long as at most $t_{\mathsf{S}} - 1$ sellers are corrupted and all buyers are honest. Our impossibility results complement this by proving that our assumptions (timelocks and either deposits or an honest majority of participating sellers) are optimal.

3 Preliminaries

We review blockchains and defer our notation and preliminaries on adaptor signatures, threshold signatures, non-interactive arguments, hard relations, secret sharing, and verifiable public-key encryption schemes to Appendix A.

Blockchains. We model scriptless blockchains as a public bulletin board of signed Unspent Transaction Output (UTXO) transactions, following [AEE+21]. This captures cryptocurrencies such as Bitcoin, Dash, and Litecoin, among others.

Definition 1 (Transaction)**.** *A transaction* tx *is a tuple*

$$(\mathsf{txid}, \mathsf{In}, \mathsf{Out})$$

consisting of an identifier txid*, a set* $\mathsf{In} = \{\mathsf{utxoid}_1, \ldots, \mathsf{utxoid}_m\}$ *of input references, and a set*

$$\mathsf{Out} = \{(\mathsf{pk}'_1, x_1), \ldots, (\mathsf{pk}'_\ell, x_\ell)\}$$

of outputs, where $\mathsf{utxoid} \in \{0,1\}^\lambda$*,* (pk'_i, x_i) *denotes a receiver public key and an amount* $x_i \geq 0$*.*

Definition 2 (UTXO set)**.** *An unspent transaction output set* UTXO *is a set of triples*

$$(\mathsf{utxoid}, x, \mathsf{pk}),$$

where utxoid *is a unique identifier,* $x > 0$ *is an amount, and* pk *is the public key that can spend this UTXO.*

Definition 3 (Valid Transaction)**.** *Let* UTXO *be an unspent transaction output set,* Σ *be a signature scheme, and* Tx *be a set of transactions. We say that a transaction* $\mathsf{tx} = (\mathsf{txid}, \mathsf{In}, \mathsf{Out})$ *together with a signature* σ *is valid if*

- txid *is fresh, i.e.,* $\forall \mathsf{tx}' \in \mathsf{Tx} : \ \mathsf{tx}'.\mathsf{txid} \neq \mathsf{txid}$*,*
- *every input is present in* UTXO*, i.e.,* $\forall\ \mathsf{utxoid} \in \mathsf{In}\ \exists(\ \mathsf{utxoid}, \cdot, \cdot\ ) \in \mathsf{UTXO}$*,*
- *all output amounts are positive, i.e.,* $\forall\ (\mathsf{pk}', x) \in \mathsf{Out} : \ x > 0$*,*
- *the input and output amounts are equal, i.e.,*

$$\sum_{(\mathsf{utxoid}, x, \cdot) \in \mathsf{UTXO},\ \mathsf{utxoid} \in \mathsf{In}} x = \sum_{(\cdot, x') \in \mathsf{Out}} x',$$

- σ *is valid under the public keys of all inputs, i.e.,* $\Sigma.\mathsf{Vrfy}(\mathsf{pk}, \mathsf{tx}, \sigma) = 1$ *for all* $(\mathsf{utxoid}, \cdot, \mathsf{pk}) \in \mathsf{UTXO}$ *with* $\mathsf{utxoid} \in \mathsf{In}$.

Definition 4 (Blockchain)**.** *Let* Σ *be a signature scheme. A UTXO blockchain* $\mathbb{B} = (\mathsf{Tx}, \mathsf{UTXO})$ *w.r.t.* Σ *consists of an ordered list of signed transactions* Tx *and an unspent transaction output set* UTXO*. It provides the interfaces* Setup *and* Post*, which are defined as follows:*
$\underline{\mathbb{B} \leftarrow \mathsf{Setup}(\{(\mathsf{pk}_i, x_i)\})}$*: The setup interface reads a set of public key-amount pairs* $\{(\mathsf{pk}_i, x_i)\}$*, and outputs a blockchain* $\mathbb{B}$ *consisting of an empty transaction list* $\mathsf{Tx} = \emptyset$ *and an initial UTXO set*

$$\mathsf{UTXO} = \{(i, x_i, \mathsf{pk}_i)\}.$$

$\underline{\mathbb{B}' \leftarrow \mathbb{B}.\mathsf{Post}(\mathsf{tx}, \sigma)}$*: The post interface reads a transaction* tx *and a signature* σ*. It outputs an updated blockchain* $\mathbb{B}'$ *if the tuple* (tx, σ) *is a valid transaction w.r.t.* $(\mathsf{Tx}, \mathsf{UTXO}, \Sigma)$*. In this case, we append* (tx, σ) *to* Tx*, remove all UTXOs referenced in* In *from* UTXO*, and add each output to* UTXO*.*

Definition 5 (Balance)**.** *A public key* pk *has balance* v_{pk} *on a blockchain* $\mathbb{B}$ *if* v_{pk} *equals the total amount of all UTXOs in* $\mathbb{B}.\mathsf{UTXO}$ *spendable by* pk*, i.e.,*

$$v_{\mathsf{pk}} = \sum_{(\cdot, x, \mathsf{pk}) \in \mathbb{B}.\mathsf{UTXO}} x.$$

By convention, a UTXO of amount x with k holders adds x/k to each holder's balance.

Communication model. We work in the synchronous communication model, where protocol execution proceeds in rounds and parties are connected by pairwise private and authenticated channels. The sellers additionally have access to a broadcast channel, which delivers the same message to every seller in the round in which it is sent. We consider a rushing adversary, meaning that in each round, corrupted parties first receive all messages sent by honest parties before sending their own messages. During each round, parties read the current state of the blockchain, exchange off-chain messages, and issue transaction requests. The round concludes as the adversary arbitrarily sequences these requests, defining the next state of the blockchain. The adversary orders the requests but cannot drop them, so every request of a round takes effect unless another request of the same round invalidates it.

Blockchains with timelocks. Intuitively, a blockchain with timelocks allows parties to lock the inputs of a transaction for $\mathtt{t}$ computation rounds. During this period, the locked coins cannot be used in other transactions, but the transaction can be claimed by providing an appropriate signature. If a locked transaction is not claimed within the specified time, the locked inputs become spendable again. This abstraction can be realized, e.g., via BIP-65 for Bitcoin [Tod14].

Definition 6 (Locked Transaction)**.** *A locked transaction is a tuple* $(\mathsf{txid}, \mathsf{InL}, \mathsf{Out}, \mathtt{t})$ *where* $\mathsf{InL} = \{(\mathsf{utxoid}, x, \mathsf{pk})\}$ *contains the UTXO triples that were locked.*

When we refer to a transaction $\mathsf{tx} = (\mathsf{txid}, \mathsf{In}, \mathsf{Out})$ as "locked", we mean the locked transaction tx' obtained from tx by replacing each input reference $\mathsf{utxoid} \in \mathsf{In}$ with the corresponding UTXO triple $(\mathsf{utxoid}, x, \mathsf{pk})$ from UTXO and storing these triples in InL, while keeping the same identifier txid and output set Out. We say a signature is valid w.r.t. some (possibly locked) transaction tx, if a single public key pk is part of the transaction's input UTXOs, and $\Sigma.\mathsf{Vrfy}(\mathsf{pk}, \mathsf{tx}, \sigma) = 1$.

Definition 7 (Blockchain with Timelocks)**.** *We say a UTXO blockchain* $\mathbb{B} = (\mathsf{Tx}, \mathsf{UTXO})$ *has timelocks if it additionally consists of the set of locked transactions* LTX *that is initialized as* $\emptyset$ *and provides the additional interfaces* Lock *and* Claim*.*
$\mathbb{B}' \leftarrow \mathbb{B}.\mathsf{Lock}(\mathsf{tx}, \sigma, \mathtt{t})$*: The lock interface reads a transaction* tx*, a signature* σ*, and a time parameter* $\mathtt{t}$*. Let* tx' *denote the locked transaction obtained from* tx *and* $\mathtt{t}$ *as described above. The interface outputs an updated blockchain* $\mathbb{B}'$ *if* tx *satisfies the validity conditions w.r.t.* $(\mathsf{UTXO}, \mathsf{Tx}, \Sigma)$*, where the signature condition is evaluated on the locked transaction, i.e.,* $\Sigma.\mathsf{Vrfy}(\mathsf{pk}, \mathsf{tx}', \sigma) = 1$ *for the public key* pk *of every input UTXO. In this case, the tuple* $((\mathsf{tx}, \mathtt{t}), \sigma)$ *is added to* Tx*, the input UTXOs of* tx *are removed from* UTXO*, and* tx' *is added as locked transaction to* LTX*.*
$\mathbb{B}' \leftarrow \mathbb{B}.\mathsf{Claim}(\mathsf{tx}, \sigma)$*: The claim interface reads a transaction* tx *and a signature* σ*. It outputs an updated blockchain* $\mathbb{B}'$ *if the transaction is stored as locked in* LTX*, and the signature is valid w.r.t. the plain transaction* tx*. In this case, the tuple* (tx, σ) *is appended to* Tx*, the outputs of* tx *are added to* UTXO *accordingly, and the locked transaction is removed from* LTX*.*

If a locked transaction is not claimed in $\mathtt{t}$ *computation rounds after a tuple* $((\mathsf{tx}, \mathtt{t}), \sigma)$ *was posted to* Tx*, the locked inputs are re-added to* UTXO*, and the locked transaction is removed from* LTX*.*

Locking and claiming authorize distinct messages. The lock signature signs the locked transaction tx', whereas claiming requires a signature on tx. Locking removes the affected UTXOs from UTXO and temporarily lowers the owner's balance until the lock is claimed or expires. Thus, our fairness statements consider balances after all timelocks have expired, when only irrevocably spent funds remain deducted. We sometimes write $\mathbb{B}.\mathsf{Lock}(\mathsf{tx}, \mathsf{sk}_i, \mathtt{t})$ which refers to a group of parties jointly computing a signature σ on the locked transaction tx' derived from tx and $\mathtt{t}$, and later calling $\mathbb{B}.\mathsf{Lock}(\mathsf{tx}, \sigma, \mathtt{t})$. We also write $\mathtt{t}' \leftarrow \mathbb{B}.\mathsf{VrfyLock}(\mathsf{tx})$ which outputs the number of rounds $\mathtt{t}'$, for which tx will be locked.

4 Fair Distributed Exchange

We formalize fair distributed exchange between a group of n_S sellers and a group of n_B buyers. We make the simplifying assumption that the exchange parameters, such as the transaction tx and the statement s, are agreed upon prior to running the protocol. Fixing the transaction means that the price is agreed upon in advance, and we refer to this amount as "a coin". The two groups are aware of each other, and each seller has verified that a signature on tx valid under the buyer group's public key yields the desired payment. We also assume that the decision to execute the exchange has already been made through the group's governance, ensuring that all honest parties participate in the protocol. In this setting, our goal is to ensure that the exchange proceeds in a *fair* manner, a property we capture formally below and enforce in our constructions.

Definition 8 (Distributed Exchange)**.** *Let* $t_\mathsf{S} \leq n_\mathsf{S} \in \mathbb{N}$ *and* $t_\mathsf{B} \leq n_\mathsf{B} \in \mathbb{N}$*. Let* $\mathcal{R}$ *be a hard relation, and let* $\mathbb{B}$ *be a blockchain w.r.t. a signature scheme* Σ*. A* fair distributed exchange *protocol* $\mathsf{FDE}_{\mathcal{R},\mathbb{B}}$ *is executed between a set* $\mathcal{B}$ *of buyers and a set* $\mathcal{S}$ *of sellers. The protocol consists of the following algorithms.*
$(\mathsf{pk}_\mathsf{B}, \{(\mathsf{pk}_{\mathsf{B}_i}, \mathsf{sk}_{\mathsf{B}_i})\}_{i \in \mathcal{B}}) \leftarrow \mathsf{Setup}_\mathsf{B}(t_\mathsf{B}, n_\mathsf{B})$*: $\mathcal{B}$'s setup algorithm reads the threshold t_B and the number of buyers n_B. It outputs a joint public key pk_B and key pairs for each buyer.*
$\{(\mathsf{pk}_{\mathsf{S}_j}, \mathsf{sk}_{\mathsf{S}_j}, s, w_j)\}_{j \in \mathcal{S}} \leftarrow \mathsf{Setup}_\mathsf{S}(t_\mathsf{S}, n_\mathsf{S}, s, w)$*: The seller group's setup algorithm reads the threshold t_S, the number of sellers n_S, a statement s, and a witness w. It shares the witness into $\{w_j\}_{j \in \mathcal{S}}$ and outputs a key pair, the statement, and a witness share to each seller.*

$(w', \bot) \leftarrow \langle \{\mathsf{B}_i^{\mathbb{B}}(\mathsf{sk}_{\mathsf{B}_i}, s, \mathsf{tx})_i\}_{i\in\mathcal{B}}, \{\mathsf{S}_j^{\mathbb{B}}(\mathsf{sk}_{\mathsf{S}_j}, \mathsf{tx}, s, w_j)_j\}_{j\in\mathcal{S}} \rangle_{\mathsf{FDE}}$: *The protocol* FDE *is executed between the buyers and the sellers. Each buyer* B_i *reads its secret key* $\mathsf{sk}_{\mathsf{B}_i}$*, the statement* s*, and the transaction* tx*. Each seller* S_j *reads its secret key* $\mathsf{sk}_{\mathsf{S}_j}$*, the transaction* tx*, the statement* s*, and a witness share* w_j*. All parties have access to the blockchain* $\mathbb{B}$ *and know all public keys. At the end of the protocol, the buyers output a witness* w'*, and the sellers output* $\bot$*.*

We say that a distributed exchange protocol $\mathsf{FDE}_{\mathcal{R},\mathbb{B}}$ is *correct* if for every security parameter λ and $t_{\mathsf{B}} \leq n_{\mathsf{B}}, t_{\mathsf{S}} \leq n_{\mathsf{S}} \in \mathbb{N}$, every statement–witness pair $(s, w) \in \mathcal{R}$, all honestly generated keys, all witness shares, and every transaction tx that spends coins of pk_{B}, every witness w' output by the buyers satisfies $\mathcal{R}(s, w') = 1$, and the blockchain $\mathbb{B}$ contains a signature σ_{coin} that verifies under the joint buyer public key pk_{B} on message tx.

4.1 Security Properties

We define the security properties of an FDE protocol in terms of *buyer fairness*, *seller fairness*, and *witness privacy*. We say a distributed exchange protocol is *fair* if it satisfies these three notions. In what follows, we consider a static adversary, meaning that corruptions occur before the adversary interacts with the protocol. For small group sizes, our results for a static adversary can be extended to an adaptive adversary via a guessing argument.

Buyer Fairness. Buyer fairness protects the honest buyers. It guarantees that even if an adversary corrupts all sellers and up to $t_{\mathsf{B}} - 1$ buyers, the buyer group's balance decreases only if all honest buyers learn a valid witness.

Definition 9 (Buyer Fairness)**.** *Let* $\mathcal{R}$ *be a hard relation and* $\mathbb{B}$ *a blockchain. A distributed exchange protocol* $(\mathsf{FDE}_{\mathcal{R},\mathbb{B}})$ *achieves* t_{B}-buyer fairness*, if for every* PPT *adversary* $\mathcal{A}$ *and any* $\mathsf{corr}_{\mathsf{B}} \subset \mathcal{B}$ *with* $|\mathsf{corr}_{\mathsf{B}}| < t_{\mathsf{B}} \leq n_{\mathsf{B}}$*, there exists a negligible function* negl *such that, for* $(s, w) \leftarrow \mathsf{genR}(\lambda)$*,*

$$\Pr\left[\mathsf{BF}^{\mathcal{A},\mathbb{B}}_{t_{\mathsf{B}},n_{\mathsf{B}},n_{\mathsf{S}},\mathsf{corr}_{\mathsf{B}}}(s, w) = 1\right] \leq \mathsf{negl}(\lambda).$$

The experiment BF *is described in Figure 1 and the probability is taken over the random choices of all probabilistic algorithms and the random choice of* genR*.*

Seller Fairness. Seller fairness protects the honest sellers. It guarantees that if an adversary controls all buyers and up to $t_{\mathsf{S}} - 1$ sellers, no honest seller loses any funds, and either every honest seller receives its fair share of the payment, or the adversary learns nothing about the witness beyond its prior knowledge.

To formalize seller fairness, we follow a semantic-security style definition [KL14a]. We compare the winning probability of a PPT adversary in two experiments, one in which it interacts with the honest sellers, and one in which it interacts with a simulator. The simulator emulates the honest sellers without their witness share. We model "learns any information" via a leakage function f, which can be any PPT function, and require the adversary to output $f(w)$. For trivial leakage functions, such as a constant function, the adversary always computes $f(w)$, which inflates both probabilities equally and gives it no advantage [KL14a].

$\underline{\mathsf{BF}^{\mathcal{A},\mathbb{B}}_{t_\mathsf{B},n_\mathsf{B},n_\mathsf{S},\mathsf{corr}_\mathsf{B}}(s,w)\text{:}}$

1. The challenger runs setup to generate keys and to share the witness w.
2. It forwards all public keys, the secret keys of all sellers, the secret keys of each buyer in corr_B, and the pair (s, w) to the adversary $\mathcal{A}$.
3. $\mathcal{A}$ can interact with the challenger, which simulates all honest buyers.
4. Eventually, $\mathcal{A}$ outputs an index i and wins the buyer fairness experiment, if buyer i is not corrupted ($i \notin \mathsf{corr}_\mathsf{B}$), and
   - the initial balance of pk_B decreased, and
   - buyer i does not output a valid witness w' for statement s.

Figure 1: Security experiment for buyer fairness.

Definition 10 (Seller Fairness)**.** *Let $\mathcal{R}$ be a hard relation and $\mathbb{B}$ a blockchain. A distributed exchange protocol ($\mathsf{FDE}_{\mathcal{R},\mathbb{B}}$) achieves t_S-*seller fairness*, if for every PPT adversary $\mathcal{A}$, every $\mathsf{corr}_\mathsf{S} \subset \mathcal{S}$ with $|\mathsf{corr}_\mathsf{S}| < t_\mathsf{S} \leq n_\mathsf{S}$, and every polynomial-time computable function f, there exist a negligible function negl and a PPT simulator Sim such that, for $(s, w) \leftarrow \mathsf{genR}(\lambda)$,*

$$\Pr[\mathsf{L}] \leq \mathsf{negl}(\lambda) \quad \textit{and} \quad \Pr\left[\mathsf{SF}^0 = 1\right] \leq \Pr\left[\mathsf{SF}^1 = 1\right] + \mathsf{negl}(\lambda).$$

The experiment SF^b and the event L are defined in Figure 2, and the probabilities are taken over the random choices of all probabilistic algorithms and the random choice of genR.

Witness Privacy. Witness privacy ensures that the witness w is revealed only to the buyers, and any coalition of $t_\mathsf{S} - 1$ malicious sellers learns nothing about it. We assume that all buyers are honest in this definition. Otherwise, witness privacy would be impossible to guarantee, since corrupted buyers eventually learn w and could trivially leak it to the adversary. We formalize the "learn nothing" notion in a semantic-security style, akin to seller fairness.

Definition 11 (Witness Privacy)**.** *Let $\mathcal{R}$ be a hard relation and $\mathbb{B}$ be a blockchain. A distributed exchange protocol ($\mathsf{FDE}_{\mathcal{R},\mathbb{B}}$) achieves* witness privacy*, if for every PPT adversary $\mathcal{A}$, any $\mathsf{corr}_\mathsf{S} \subset \mathcal{S}$ with $|\mathsf{corr}_\mathsf{S}| < t_\mathsf{S} \leq n_\mathsf{S}$, and any polynomial-time computable function f, there exist a negligible function negl and a PPT simulator Sim such that, for $(s, w) \leftarrow \mathsf{genR}(\lambda)$,*

$$\left|\Pr\left[\mathsf{WitPriv}^0 = 1\right] - \Pr\left[\mathsf{WitPriv}^1 = 1\right]\right| \leq \mathsf{negl}(\lambda).$$

The experiment $\mathsf{WitPriv}$ is described in Figure 3 and the probability is taken over the random choices of all probabilistic algorithms and the random choice of genR.

$\underline{\mathsf{SF}^{\mathcal{A},\mathbb{B},f,\mathsf{Sim},b}_{n_\mathsf{B},t_\mathsf{S},n_\mathsf{S},\mathsf{corr}_\mathsf{S}}(s,w)}$:

1. The challenger runs setup to generate keys and to share the witness w.
2. It forwards all public keys, the secret keys of all buyers, the secret keys and witness shares of each seller in corr_S, and the statement s to the adversary $\mathcal{A}$.
3. If $b = 0$, $\mathcal{A}$ can interact with the challenger, which simulates all honest sellers.
4. If $b = 1$, $\mathcal{A}$ can interact with a simulator Sim, that reads all public information and the secret keys of the honest sellers.
5. Eventually, $\mathcal{A}$ outputs an index j and a value y. The experiment outputs 1 if seller j is not corrupted ($j \notin \mathsf{corr}_\mathsf{S}$), $y = f(w)$, and the initial balance of seller j increased by less than $v_\mathsf{tx}/n_\mathsf{S}$, where v_tx denotes the total value that the payment transaction tx pays to the sellers.

Figure 2: Security experiment for seller fairness. We abbreviate this experiment by SF^b, and L denotes the event that the balance of an honest seller decreases in SF^0.

Design choices. Our buyer fairness definition enforces that every buyer learns the same witness. This definition implies the cases in which each buyer derives different information from the same witness, e.g., by using public-key encryption, as also done in [CD00]. Our definition of seller fairness enforces an equal payout amongst all sellers. It can be adapted to the cases in which each seller receives a different predetermined share of the payment by replacing $v_\mathsf{tx}/n_\mathsf{S}$ with the corresponding amount in Figure 2.

5 Threshold Adaptor Signatures

We introduce *threshold adaptor signatures* (TAS), a generalization of classical adaptor signatures that allows a subset of signers to jointly compute a pre-signature. Unlike standard adaptor signatures, where the pre-signature is derived via a single algorithm, our threshold construction requires an interactive protocol between at least t-out-of-n signers to produce a pre-signature $\widetilde{\sigma}$. To ensure compatibility with existing adaptor-based protocols (and thus with the underlying blockchain after adaptation), the resulting pre-signature must be indistinguishable from a single-user pre-signature. Furthermore, our scheme preserves the fundamental properties of classical adaptor signatures, amongst them:

- **Adaptation:** Given a pre-signature $\widetilde{\sigma}$ and a witness w, one can compute a full signature σ.
- **Extraction:** Given a pre-signature $\widetilde{\sigma}$, the adapted pre-signature σ, and a statement s, one can extract the corresponding witness w.

We now define threshold adaptor signatures.

Definition 12 (Threshold Adaptor Signature)**.** *A* threshold adaptor signature scheme *w.r.t. a hard relation* $\mathcal{R}$ *and a threshold signature scheme* $\mathsf{TS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *consists of a tuple of four algorithms and protocols* $\mathsf{TAS}_{\mathcal{R},\mathsf{TS}} = (\mathsf{pSign}, \mathsf{Adapt}, \mathsf{pVrfy}, \mathsf{Extract})$ *defined as:*

$\underline{\mathsf{WitPriv}^{\mathcal{A},\mathbb{B},f,\mathsf{Sim},b}_{n_\mathsf{B},t_\mathsf{S},n_\mathsf{S},\mathsf{corr}_\mathsf{S}}(s,w)\text{:}}$

1. The challenger runs setup to generate keys and to share the witness w.
2. It forwards all public keys, the secret keys and witness shares of each seller in corr_S, and the statement s to the adversary $\mathcal{A}$.
3. If $b = 0$, $\mathcal{A}$ can interact with the challenger, which simulates all honest parties.
4. If $b = 1$, $\mathcal{A}$ can interact with a simulator Sim, that reads all keys, the statement s, and the witness shares of all corrupted sellers and simulates all honest parties.
5. Eventually, $\mathcal{A}$ outputs a value y and wins the experiment, if $y = f(w)$.

Figure 3: Security experiment for witness privacy ($\mathsf{WitPriv}^b$).

$\underline{\widetilde{\sigma} \leftarrow \langle \mathsf{pSign}(\mathsf{sk}_i, \mathsf{pk}, m, s)\rangle}$. *The* pre-signing protocol *is an interactive algorithm of which an instance is run by each signer* $\mathcal{P}_1, \ldots, \mathcal{P}_t$ *concurrently. Concretely, signer* $\mathcal{P}_i$ *runs* pSign_i*, which takes as input a secret key share* sk_i*, the joined public key* pk*, a message* m*, and an NP-statement* s*. At the end of the protocol,* $\mathcal{P}_i$ *obtains a pre-signature* $\widetilde{\sigma}$ *as output.*
$\underline{b \leftarrow \mathsf{pVrfy}(\mathsf{pk}, m, s, \widetilde{\sigma})}$. *The* pre-verification algorithm *reads a public key* pk*, message* m*, statement* s*, and pre-signature* $\widetilde{\sigma}$*, outputs a bit* b*.*
$\underline{\sigma \leftarrow \mathsf{Adapt}(\mathsf{pk}, \widetilde{\sigma}, w)}$. *The* adapting algorithm*reads a pre-signature* $\widetilde{\sigma}$ *and witness* w *for the statement* s*, outputs an adapted signature* σ*.*
$\underline{w \leftarrow \mathsf{Extract}(\mathsf{pk}, \widetilde{\sigma}, \sigma, s)}$. *The* extracting algorithm *reads a pre-signature* $\widetilde{\sigma}$*, signature* σ*, and statement* s*, outputs a witness* w *such that* $(s, w) \in \mathcal{R}$*, or* $\perp$*.*

A threshold adaptor signature is correct under the following intuitive conditions. First, for honestly generated keys and an honestly computed instance of a hard relation, adapting an honestly generated pre-signature $\widetilde{\sigma}$ with a corresponding witness w yields a valid signature σ. Furthermore, given both a valid pre-signature and a valid signature together with the statement s, it is computationally feasible to deduce or extract the corresponding witness w efficiently. We refer to this property as *pre-signature correctness*. We deduce the security notions for threshold adaptor signatures from the revised security notions of adaptor signatures by Gerhart et al. [GSST24]. These are extractability, unique extractability, unlinkability, pre-signature adaptability, and pre-verify soundness. For an outline of the construction, we refer to Section 2.1.

Definition 13 (Pre-signature correctness)**.** *A threshold adaptor signature scheme* TAS *satisfies* pre-signature correctness*, if for all* $\lambda, t < n \in \mathbb{N}$ *and* $m \in \{0,1\}^{\ell_m}$

$$\Pr\left[\begin{array}{c|l} \mathsf{pVrfy}(\mathsf{pk}, m, s, \widetilde{\sigma}) = 1 \wedge & par \leftarrow \mathsf{Setup}(n,t) \\ \mathsf{Vrfy}(\mathsf{pk}, m, \sigma) = 1 \wedge & (\mathsf{pk}, \mathsf{sk}_i) \leftarrow \langle \mathsf{KeyGen}(i)\rangle \\ (s, w') \in \mathcal{R} & (s,w) \leftarrow \mathsf{genR}(\lambda), \\ & \widetilde{\sigma} \leftarrow \langle \mathsf{pSign}(\mathsf{sk}_i, \mathsf{pk}, m, s)\rangle \\ & \sigma := \mathsf{Adapt}(\mathsf{pk}, \widetilde{\sigma}, w), \\ & w' := \mathsf{Extract}(\mathsf{pk}, \widetilde{\sigma}, \sigma, s) \end{array}\right] = 1.$$

5.1 Security of Threshold Adaptor Signatures

We adapt the security definitions of Gerhart et al.[GSST24] to the threshold setting. A threshold adaptor signature scheme must satisfy extractability, unique extractability, unlinkability, pre-signature adaptability, and pre-verification soundness. These properties are formally defined below.

Extractability. Extractability ensures that a malicious party cannot use a pre-signature $\widetilde{\sigma}$ w.r.t. a statement s to forge a valid signature σ without revealing a corresponding witness w, thereby protecting the signer.

Definition 14 (Extractability)**.** *A threshold adaptor signature scheme* TAS *is* extractable *if for every* PPT *adversary* $\mathcal{A}$*, there is a negligible function* negl *such that for every* $\lambda, t \leq n \in \mathbb{N}$

$$\Pr\left[\mathsf{Ext}^{\mathcal{A}}_{\mathsf{TAS},t,n}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda),$$

where the experiment $\mathsf{Ext}^{\mathcal{A}}_{\mathsf{TAS},t,n}(\lambda)$ *is defined in Figure 4 and the randomness is taken over all random coins used by the probabilistic algorithms.*

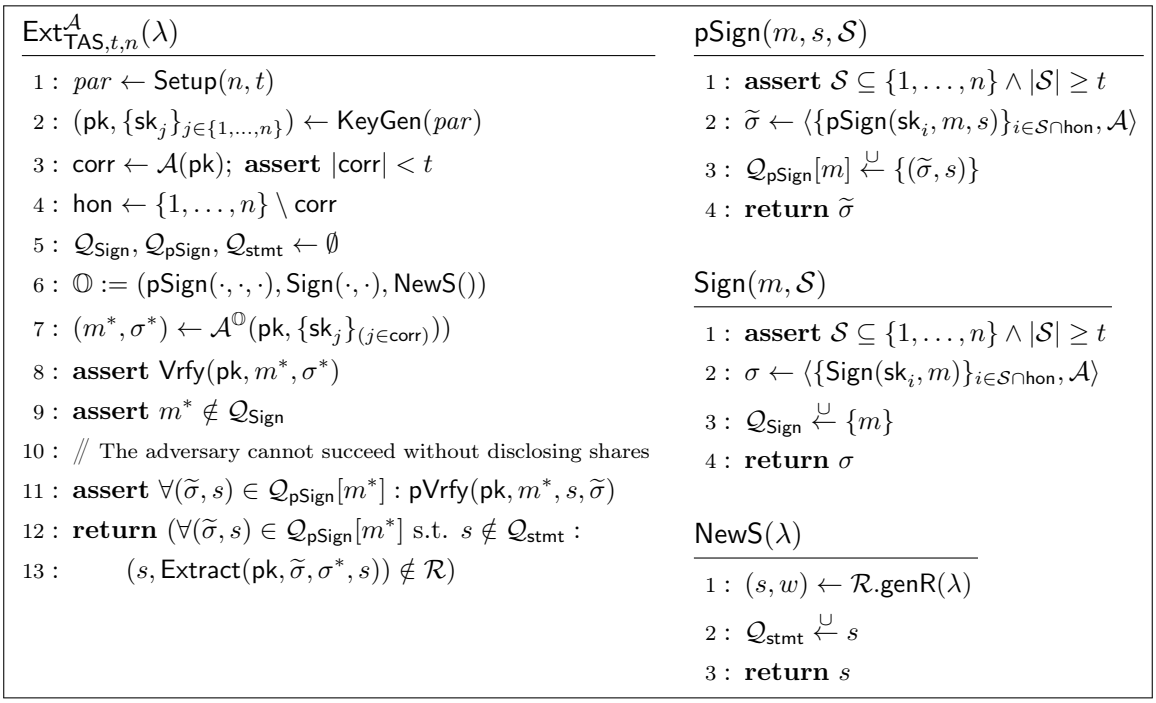


Figure 4: The security experiment $\mathsf{Ext}^{\mathcal{A}}_{\mathsf{TAS},t,n}(\lambda)$.

Unique Extractability. Unique extractability guarantees that a verified pre-signature commits to a single valid signature and a single witness. No efficient adversary can generate a pre-signature $\widetilde{\sigma}$ for a message m and a statement s such that two distinct signatures on m can be derived from $\widetilde{\sigma}$, each revealing a valid witness. More formally:

Definition 15 (Unique Extractability)**.** *A threshold adaptor signature scheme* TAS *is unique extractable, if for every* PPT *adversary* $\mathcal{A}$ *there exists a negligible function* negl *such that for every* $\lambda, t \leq n \in \mathbb{N}$

$$\Pr\left[\mathsf{UniqueExtractability}^{\mathcal{A}}_{\mathsf{TAS},t,n}(\lambda) = 1\right] \leq \mathsf{negl}(\lambda) \ ,$$

where experiment $\mathsf{UniqueExtractability}^{\mathcal{A}}_{\mathsf{TAS},t,n}(\lambda)$ *is described in Figure 5, and the probability is taken over the random choices of all probabilistic algorithms.*

$\mathsf{UniqueExtractability}^{\mathcal{A}}_{\mathsf{TAS},t,n}(\lambda)$

$par \leftarrow \mathsf{Setup}(n,t)$
$(\mathsf{pk}, \{\mathsf{sk}_j\}_{j\in\{1,\ldots,n\}}) \leftarrow \mathsf{KeyGen}(par)$
$\mathsf{corr} \leftarrow \mathcal{A}(\mathsf{pk})$; **assert** $|\mathsf{corr}| < t$
$\mathsf{hon} \leftarrow \{1,\ldots,n\} \setminus \mathsf{corr}$
$\mathcal{Q}_{\mathsf{Sign}}, \mathcal{Q}_{\mathsf{pSign}}, \mathcal{Q}_{\mathsf{stmt}} \leftarrow \emptyset$
$\mathbb{O} := (\mathsf{pSign}(\cdot,\cdot,\cdot), \mathsf{Sign}(\cdot,\cdot))$
$(m, s, \widetilde{\sigma}, \sigma, \sigma') \leftarrow \mathcal{A}^{\mathbb{O}}(\mathsf{pk}, \{\mathsf{sk}_j\}_{(j\in\mathsf{corr})})$
assert $(\sigma \neq \sigma')$
assert $\mathsf{Vrfy}(\mathsf{pk}, m, \sigma) \wedge \mathsf{Vrfy}(\mathsf{pk}, m, \sigma')$
assert $\mathsf{pVrfy}(\mathsf{pk}, m, s, \widetilde{\sigma})$
$y \leftarrow \mathsf{Extract}(\mathsf{pk}, \widetilde{\sigma}, \sigma, s); y' \leftarrow \mathsf{Extract}(\mathsf{pk}, \widetilde{\sigma}, \sigma', s)$
return $(s,y) \in \mathcal{R} \wedge (s,y') \in \mathcal{R}$

$\mathsf{pSign}(m, s, \mathcal{S})$

assert $\mathcal{S} \subseteq \{1,\ldots,n\} \wedge |\mathcal{S}| \geq t$
$\widetilde{\sigma} \leftarrow \langle \{\mathsf{pSign}(\mathsf{sk}_i, m, s)\}_{i\in\mathcal{S}\cap\mathsf{hon}}, \mathcal{A}\rangle$
return $\widetilde{\sigma}$

$\mathsf{Sign}(m, \mathcal{S})$

assert $\mathcal{S} \subseteq \{1,\ldots,n\} \wedge |\mathcal{S}| \geq t$
$\sigma \leftarrow \langle \{\mathsf{Sign}(\mathsf{sk}_i, m)\}_{i\in\mathcal{S}\cap\mathsf{hon}}, \mathcal{A}\rangle$
return σ

Figure 5: The security experiment $\mathsf{UniqueExtractability}^{\mathcal{A}}_{\mathsf{TAS},t,n}(\lambda)$.

Unlinkability. Unlinkability ensures that an adversary cannot distinguish between standard signatures and those derived from adapted pre-signatures, even if the adaptations use adversary-provided witnesses. The adversary chooses the message, the statement-witness pair, and the signer set of every challenge query, corrupts signers of its choice, and interacts with the honest signers through the pre-signing and signing oracles.

In contrast to the single-signer unlinkability notion, any party that participates in a challenge session or merely controls message delivery within it can trivially determine how the resulting signature was produced. Therefore, the challenge oracle executes both challenge sessions entirely internally, running all signers in $\mathcal{S}$ honestly on the key shares generated during key generation, and hands the adversary only the resulting signature σ_b.

Definition 16 (Unlinkability)**.** *A threshold adaptor signature scheme* TAS *is unlinkable if for every* PPT *adversary* $\mathcal{A}$ *there exists a negligible function* negl *such that for every* $\lambda, t \leq n \in \mathbb{N}$

$$\left|\Pr\left[\mathsf{Unlinkability}^{\mathcal{A}}_{\mathsf{TAS},t,n}(\lambda) = 1\right] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda) \ ,$$

where experiment $\mathsf{Unlinkability}^{\mathcal{A}}_{\mathsf{TAS},t,n}(\lambda)$ *is described in Figure 6, and the probability is taken over the random choices of all probabilistic algorithms.*

$\mathsf{Unlinkability}^{\mathcal{A}}_{\mathsf{TAS},t,n}(\lambda)$

1 : $par \leftarrow \mathsf{Setup}(n,t)$
2 : $(\mathsf{pk}, \{\mathsf{sk}_j\}_{j\in\{1,\ldots,n\}}) \leftarrow \mathsf{KeyGen}(par)$
3 : $\mathsf{corr} \leftarrow \mathcal{A}(\mathsf{pk})$; **assert** $|\mathsf{corr}| < t$
4 : $\mathsf{hon} \leftarrow \{1,\ldots,n\} \setminus \mathsf{corr}$
5 : $b \leftarrow\$ \{0,1\}$
6 : $\mathbb{O} := (\mathsf{pSign}(\cdot,\cdot,\cdot), \mathsf{Sign}(\cdot,\cdot), \mathsf{Chall}(b,\cdot,\cdot,\cdot))$
7 : $b' \leftarrow \mathcal{A}^{\mathbb{O}}(\mathsf{pk}, \{\mathsf{sk}_j\}_{(j\in\mathsf{corr})})$
8 : **return** $b' == b$

$\mathsf{Chall}(b, m, (s,w), \mathcal{S})$

1 : **assert** $\mathcal{S} \subseteq \{1,\ldots,n\} \wedge |\mathcal{S}| \geq t$
2 : **assert** $(s,w) \in \mathcal{R}$
3 : // Both sessions run internally; all signers in $\mathcal{S}$
4 : // are executed honestly, without $\mathcal{A}$
5 : $\widetilde{\sigma} \leftarrow \langle\{\mathsf{pSign}(\mathsf{sk}_i, m, s)\}_{i\in\mathcal{S}}\rangle$
6 : $\sigma_0 \leftarrow \mathsf{Adapt}(\mathsf{pk}, \widetilde{\sigma}, w)$
7 : $\sigma_1 \leftarrow \langle\{\mathsf{Sign}(\mathsf{sk}_i, m)\}_{i\in\mathcal{S}}\rangle$
8 : **return** σ_b

$\mathsf{pSign}(m, s, \mathcal{S})$

1 : **assert** $\mathcal{S} \subseteq \{1,\ldots,n\} \wedge |\mathcal{S}| \geq t$
2 : $\widetilde{\sigma} \leftarrow \langle\{\mathsf{pSign}(\mathsf{sk}_i, m, s)\}_{i\in\mathcal{S}\cap\mathsf{hon}}, \mathcal{A}\rangle$
3 : **return** $\widetilde{\sigma}$

$\mathsf{Sign}(m, \mathcal{S})$

1 : **assert** $\mathcal{S} \subseteq \{1,\ldots,n\} \wedge |\mathcal{S}| \geq t$
2 : $\sigma \leftarrow \langle\{\mathsf{Sign}(\mathsf{sk}_i, m)\}_{i\in\mathcal{S}\cap\mathsf{hon}}, \mathcal{A}\rangle$
3 : **return** σ

Figure 6: The security experiment $\mathsf{Unlinkability}^{\mathcal{A}}_{\mathsf{TAS},t,n}(\lambda)$.

Pre-Signature Adaptability. Pre-signature adaptability ensures that a pre-signature related to a statement s can be transformed into a valid signature with a witness w, even in cases where the signer acts maliciously.

Definition 17 (Pre-signature adaptability)**.** *A threshold adaptor signature scheme* TAS *satisfies* pre-signature adaptability*, if for all* $\lambda \in \mathbb{N}$*, messages and pre-signatures* $\widetilde{\sigma}, m \in \{0,1\}^*$*, statement/witness pairs* $(s,w) \in \mathcal{R}$*, public keys* pk *we have,*

$$\textit{if } \mathsf{pVrfy}(\mathsf{pk}, m, s, \widetilde{\sigma}) = 1, \textit{ then } \mathsf{Vrfy}(\mathsf{pk}, m, \mathsf{Adapt}(\mathsf{pk}, \widetilde{\sigma}, w)) = 1.$$

Pre-Verify Soundness. Pre-verify soundness ensures that the pre-verification algorithm satisfies *computational* soundness w.r.t. the relation $\mathcal{R}$. In particular, pVrfy should reject pre-signatures computed using statements $s \notin \mathcal{R}$. Intuitively, pre-verify soundness ensures that every valid pre-signature can be adapted to a full signature, and one can extract a witness from it. This strengthens the property of pre-signature adaptability (Definition 17), which is restricted to honestly generated pre-signatures on statements in the relation.

Definition 18 (Computational Pre-Verify Soundness)**.** *An adaptor signature scheme* TAS *satisfies* computational pre-verify soundness *if for every* PPT *adversary* $\mathcal{A}$ *there exists a negligible function* negl *such that for every* $\lambda \in \mathbb{N}$ *and polynomially-bounded* $s \notin \mathcal{L}_{\mathcal{R}}$*,*

$$\mathsf{Pr}\left[(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}(\lambda), (m, \widetilde{\sigma}) \leftarrow \mathcal{A}(\mathsf{sk}) : \mathsf{pVrfy}(\mathsf{pk}, m, s, \widetilde{\sigma}) = 1\right] \leq \nu(\lambda).$$

5.2 Schnorr Threshold Adaptor Construction

Construction 1 (Schnorr Threshold Adaptor Signature)**.** *Let* $\lambda \in \mathbb{N}$ *be the security parameter,* $\mathcal{R}$ *be a relation for the language* $\mathcal{L}_{\mathcal{R}} := \{(s,w) | s = g^w\}$*. Let* Sparkle *be a threshold signature as*

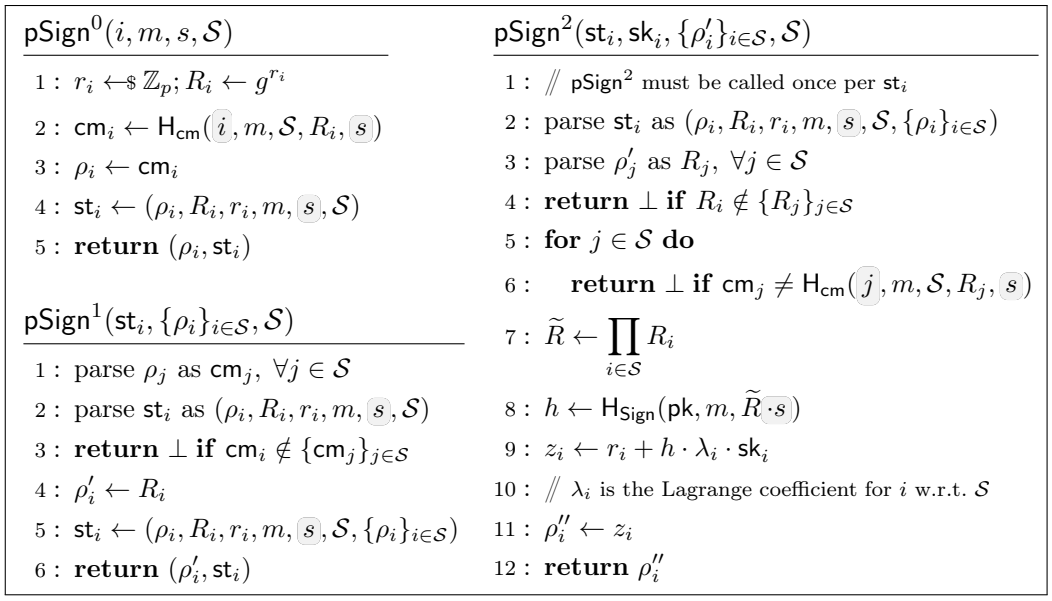


Figure 7: Schnorr threshold adaptor signature construction based on Sparkle [CKM23]. We state the algorithms Adapt, Extract, pVrfy, and Combine in Figure 8. We mark the main changes compared to Sparkle in gray.

defined in Construction 4. We construct the threshold adaptor signature $\mathsf{TAS}_{\mathsf{Sparkle},\mathcal{R}}$ *by defining the algorithms* $\mathsf{pSign}^0, \mathsf{pSign}^1, \mathsf{pSign}^2$ *in Figure 7 and* Combine, Adapt, Extract, pVrfy *in Figure 8 which lead to the following pre-signing protocol.*
$\widetilde{\sigma} \leftarrow \langle \mathsf{pSign}(\mathsf{sk}_i, \mathsf{pk}, m, s) \rangle$ *Each signer runs the three pre-signing rounds* $\mathsf{pSign}^0, \mathsf{pSign}^1, \mathsf{pSign}^2$*, forwards the local outputs* $\rho_i, \rho'_i, \rho''_i$ *to all other signers, and combines the collected outputs with* Combine*.*

Theorem 1 (Security of Construction 1)**.** *If* Sparkle *is existentially unforgeable and* $\mathcal{R}$ *is a hard relation, then* $\mathsf{TAS}_{\mathsf{Sparkle},\mathcal{R}}$ *of Construction 1 achieves extractability, unique extractability, unlinkability, pre-signature adaptability, and computational pre-verify soundness in the random oracle model.*

Lemma 1 (Extractability)**.** *Let* $\mathsf{TAS}_{\mathsf{Sparkle},\mathcal{R}}$ *be defined as in Construction 1. If* Sparkle *is existentially unforgeable and* $\mathcal{R}$ *is a hard relation, then* $\mathsf{TAS}_{\mathsf{Sparkle},\mathcal{R}}$ *achieves extractability in the random oracle model.*

Proof of Lemma 1. We assume towards contradiction that there exists an adversary $\mathcal{A}$ that breaks the full extractability of TAS and build a reduction $\mathcal{B}$ that breaks the existential unforgeability of Sparkle or the hardness of the relation $\mathcal{R}$. This reduction $\mathcal{B}$ simulates the experiment Ext perfectly to the adversary, which we show using a sequence of experiments with negligible transitions.

Experiment $\mathcal{G}_0$: The initial experiment $\mathcal{G}_0$ is the original Ext experiment. Since $\mathcal{G}_0$ simulates Ext perfectly, it holds that $\Pr[\mathsf{Ext}(\lambda) = 1] = \Pr[\mathcal{G}_0(\lambda) = 1]$.

Experiment $\mathcal{G}_1$: The first experiment $\mathcal{G}_1$ differs from $\mathcal{G}_0$ when the event $\mathsf{Break}_{\mathcal{R}}$ happens. $\mathsf{Break}_{\mathcal{R}}$ occurs if the adversary outputs a forgery, which allows breaking a challenge instance of the hard

$\mathsf{Adapt}(\mathsf{pk}, \widetilde{\sigma}, w)$

1 : parse $\widetilde{\sigma}$ as $(\widetilde{\sigma}_1, \widetilde{\sigma}_2)$
2 : **return** $(\widetilde{\sigma}_1, \widetilde{\sigma}_2 + w)$

$\mathsf{Extract}(\mathsf{pk}, \widetilde{\sigma}, \sigma, s)$

1 : parse $\widetilde{\sigma}$ as $(\widetilde{\sigma}_1, \widetilde{\sigma}_2)$
2 : parse σ as (σ_1, σ_2)
3 : **return** $(\sigma_2 - \widetilde{\sigma}_2)$

$\mathsf{pVrfy}(\mathsf{pk}, m, s, \widetilde{\sigma})$

1 : parse $\widetilde{\sigma}$ as $(\widetilde{R} \cdot s, z)$
2 : $h \leftarrow \mathsf{H}_{\mathsf{Sign}}(\mathsf{pk}, m, \widetilde{R} \cdot s)$
3 : **return** $\widetilde{R} \cdot \mathsf{pk}^h = g^z$

$\mathsf{Combine}(\{\rho'_i, \rho''_i\}_{i \in \mathcal{S}})$

1 : parse ρ'_i as R_i; $z_i \leftarrow \rho''_i \; \forall i \in \mathcal{S}$
2 : $\widetilde{R} \leftarrow \prod_{i \in \mathcal{S}} R_i; z \leftarrow \sum_{i \in \mathcal{S}} z_i$
3 : $\widetilde{\sigma} \leftarrow (\widetilde{R} \cdot s, z)$
4 : **return** $\widetilde{\sigma}$

Figure 8: The algorithms $\mathsf{Adapt}, \mathsf{Extract}, \mathsf{pVrfy}$, and $\mathsf{Combine}$

relation (i.e., a statement that was output by the oracle NewS). If $\mathsf{Break}_{\mathcal{R}}$ happens, the experiment $\mathcal{G}_1$ aborts. This hop does not differ from Ext if the adversary returns an adapted pre-signature on a non-challenge statement since the winning condition covers this already.

Claim 1. *Let* $\mathsf{Break}_{\mathcal{R}}$ *be the event, whereby* $\mathcal{G}_1$ *aborts. Then, the adversary* $\mathcal{A}$ *found a signature forgery* σ^* *that allows extracting a challenge witness using a previously output pre-signature. If* $\mathsf{Break}_{\mathcal{R}}$ *happens, then we can build a reduction* $\mathcal{B}'$ *that can break the hardness of the hard relation* $\mathcal{R}$*. Thus,* $\mathsf{Break}_{\mathcal{R}}$ *occurs only with negligible probability* $\mathsf{negl}_1(\lambda)$*.*

Proof of the claim. To show this claim, we build a reduction $\mathcal{B}$ against the hardness of the hard relation $\mathcal{R}$. As input, $\mathcal{B}$ gets a challenge statement s'. The reduction $\mathcal{B}$ guesses which of the polynomially many requested challenge statements will be broken by $\mathcal{A}$ and replaces this statement with s'. When the experiment $\mathcal{G}_1$ aborts as $\mathsf{Break}_{\mathcal{R}}$ happened, $\mathcal{B}$ extracts a witness for s' using the extract algorithm, the signature forgery σ^* and the corresponding pre-signature. This extracting yields a valid witness since $\mathsf{Break}_{\mathcal{R}}$ happened. $\mathcal{B}$ then outputs this valid witness for s'. The probability that $\mathcal{B}$ is successful is polynomially bounded by the ability of the adversary to cause the event $\mathsf{Break}_{\mathcal{R}}$ to happen. As we assume that the relation $\mathcal{R}$ is a hard relation, such a $\mathcal{B}$ cannot exist, and hence, the gap between the experiments $\mathcal{G}_0$ and $\mathcal{G}_1$ is negligible. □

Experiment $\mathcal{G}_2$: In this experiment, we simulate the pre-signing and signing oracles using the signing oracles provided by the existential unforgeability experiment of $\mathsf{Sparkle}$. As input, $\mathcal{B}$ receives the public key pk and forwards pk to $\mathcal{A}$. Eventually, $\mathcal{A}$ outputs the corrupted set corr, and $\mathcal{B}$ also outputs corr. Eventually, $\mathcal{B}$ receives the corrupted signing keys and forwards these to $\mathcal{A}$. When $\mathcal{A}$ queries the signing oracle on $(m, \mathcal{S})$, $\mathcal{B}$ queries its own signing oracle on $(m, \mathcal{S})$ and relays the interaction between $\mathcal{A}$ and the honest signers in $\mathcal{S} \cap \mathsf{hon}$. When $\mathcal{A}$ queries the pre-signing oracle on $(m, s, \mathcal{S})$, $\mathcal{B}$ samples a message m' uniformly at random and queries its own signing oracle on $(m', \mathcal{S})$; this oracle executes all honest signers in $\mathcal{S} \cap \mathsf{hon}$ via the rounds $\mathsf{Sign}^0, \mathsf{Sign}^1, \mathsf{Sign}^2$ of $\mathsf{Sparkle}$. In the commitment round, $\mathcal{B}$ forwards the commitments of the honest signers, which it obtains from its Sign^0 oracle, to $\mathcal{A}$. Eventually, $\mathcal{A}$ provides its commitments. Once the reduction received all commitments of the adversary, it queries its Sign^1 oracle. By doing so, $\mathcal{B}$ receives the nonces $\{R_i\}_{i \in \mathcal{S} \cap \mathsf{hon}}$ of the honest signers. It recovers the nonces behind the delivered commitments from the random oracle H_{cm} and associates each delivered commitment with its nonce, since a commitment that opens correctly in pSign^2 at position j is the answer to a query of the form $\mathsf{H}_{\mathsf{cm}}(j, m, \mathcal{S}, R_j, s)$. If a delivered commitment matches no such query, the opening check of the receiving signer fails except with negligible probability, and $\mathcal{B}$ lets this signer abort as in the real protocol. The lookup

also allows $\mathcal{B}$ to translate the delivered commitments into commitments on the message m' for its own signing session.

Since $\mathcal{A}$ may deliver inconsistent commitments to the honest signers, $\mathcal{B}$ performs the following step once for every honest signer $i \in \mathcal{S} \cap \mathsf{hon}$. It computes the combined nonce $\widetilde{R}_i \leftarrow \prod_{j \in \mathcal{S}} R_j^{(i)}$ from the nonces $R_j^{(i)}$ underlying the commitments delivered to signer i, and it programs the random oracles such that $\mathsf{H}_{\mathsf{cm}}(i, m, \mathcal{S}, R_i, s) = \mathsf{cm}_i$ and $\mathsf{H}_{\mathsf{Sign}}(\mathsf{pk}, m, \widetilde{R}_i \cdot s) = \mathsf{H}_{\mathsf{Sign}}(\mathsf{pk}, m', \widetilde{R}'_i)$, where cm_i and $\widetilde{R}'_i$ are taken from its own signing session. Only after programming these points does $\mathcal{B}$ reveal the honest nonces to $\mathcal{A}$, so every programmed point is fixed before $\mathcal{A}$ learns the last honest nonce that enters it. The partial signatures that its Sign^2 oracle outputs for m' are then valid partial pre-signatures for (m, s) w.r.t. the programmed oracles, and $\mathcal{B}$ forwards them to $\mathcal{A}$. The challenger for the unforgeability of $\mathsf{Sparkle}$ simulates the keys and oracles perfectly, but $\mathcal{A}$ can notice a gap if it queried one of the programmed points before $\mathcal{B}$ programs it.

Claim 2. *Let* coll *denote the event that the simulation of* $\mathcal{G}_2$ *deviates from* $\mathcal{G}_1$*. Then* coll *occurs only with negligible probability* $\mathsf{negl}_2(\lambda)$*.*

Proof of the claim. The simulation of $\mathcal{G}_2$ deviates from $\mathcal{G}_1$ in exactly three situations. First, the adversary $\mathcal{A}$ queried the random oracle H_{cm} on a point of the form $(i, m, \mathcal{S}, R_i, s)$, or the random oracle $\mathsf{H}_{\mathsf{Sign}}$ on a point of the form $(\mathsf{pk}, m, \widetilde{R}_i \cdot s)$, for some honest signer $i \in \mathcal{S} \cap \mathsf{hon}$ of some pre-signing query, *before* $\mathcal{B}$ programs this point, so that $\mathcal{B}$ cannot program it anymore. Second, $\mathcal{A}$ delivered a commitment together with a nonce that $\mathcal{B}$ does not associate with it, because the commitment answers no query of $\mathcal{A}$ to H_{cm} at the position and with the parameters of its session, and the receiving honest signer nevertheless accepts the opening in pSign^2, so that $\mathcal{B}$ aborts this signer while the real protocol continues. Third, two honest nonces of pre-signing queries on the same $(m, \mathcal{S}, s)$ coincide, so that $\mathcal{B}$ would have to program H_{cm} twice at the same point. Let q_{H} denote the number of random-oracle queries of $\mathcal{A}$ and q_{pSign} the number of pre-signing queries.

For the first situation, let early denote the event that $\mathcal{A}$ queries H_{cm} on the input $(i, m, \mathcal{S}, R_i, s)$ of an honest signer i before this signer reveals R_i. Until the reveal, the only value that $\mathcal{A}$ obtains from signer i is cm_i, which is the value of H_{cm} on an input that $\mathcal{A}$ has not queried and therefore uniform and independent of R_i. A single query matches a fixed honest opening with probability at most $1/p$, and a union bound over the q_{H} queries and the at most $n \cdot q_{\mathsf{pSign}}$ honest commitments gives $\Pr[\mathsf{early}] \leq n \cdot q_{\mathsf{pSign}} \cdot q_{\mathsf{H}}/p$. Conditioned on early not occurring, the same argument applies to the second programmed input $(\mathsf{pk}, m, \widetilde{R}_i \cdot s)$. The reduction programs it before it reveals R_i, so $\widetilde{R}_i$ still contains a nonce that is hidden from $\mathcal{A}$ and uniformly distributed. This bounds the first situation by $2 \cdot n \cdot q_{\mathsf{pSign}} \cdot q_{\mathsf{H}}/p$. Note that $\mathcal{B}$ programs one pair of points per honest signer, since $\mathcal{A}$ controls the message delivery within a pre-signing query and the honest signers need not agree on one combined nonce.

For the second situation, an accepted opening never causes a deviation, because the checked point then carries exactly the delivered commitment and the real protocol accepts as well. A deviation therefore requires that the real protocol accepts a pair that $\mathcal{B}$ rejects. If the delivered nonce equals an honest nonce that is not revealed yet, then $\mathcal{A}$ guessed it, which happens with probability at most $1/p$. If the checked point carries the answer to a query of $\mathcal{A}$, the check passes only on a collision, which the third situation covers. Otherwise the point is undetermined, its value is uniform and independent of the delivered commitment, and the check passes with probability at most $1/p$. With at most n^2 deliveries per pre-signing query, this bounds the second situation by $n^2 \cdot q_{\mathsf{pSign}}/p$.

For the third situation, the honest nonces are sampled uniformly and independently, so two of the at most $n \cdot q_{\mathsf{pSign}}$ of them coincide with probability at most $(n \cdot q_{\mathsf{pSign}})^2/p$. The q_{H} answers of H_{cm} to queries of $\mathcal{A}$ are uniform and independent at pairwise distinct points, so two of them collide with probability at most q_{H}^2/p. Summing the four bounds yields

$$\Pr[\mathsf{coll}] \leq \frac{2 \cdot n \cdot q_{\mathsf{pSign}} \cdot q_{\mathsf{H}}}{p} + \frac{n^2 \cdot q_{\mathsf{pSign}}}{p} + \frac{(n \cdot q_{\mathsf{pSign}})^2}{p} + \frac{q_{\mathsf{H}}^2}{p} =: \mathsf{negl}_2(\lambda),$$

which is negligible in λ since q_{H}, q_{pSign}, and n are polynomially bounded. □

When winning the Ext experiment, the adversary's output (m^*, σ^*) is a valid forgery that breaks the existential unforgeability of $\mathsf{Sparkle}$ with overwhelming probability. We note that existential unforgeability suffices for our reduction, since $\mathcal{B}$ queries its signing oracle only on the messages m' that it samples freshly and uniformly at random, so the forgery message m^* is new in the unforgeability game with overwhelming probability. The pair (m^*, σ^*) is only not a valid forgery against the existential unforgeability of Sparkle if m^* was previously chosen as a random message when our reduction queried its signature oracle. Since the messages were chosen uniformly at random by the reduction, this is only the case with a probability of at most $q/2^\lambda$, where q is the number of queried pre-signatures. This probability is negligible in λ since q is polynomially bounded. □

Lemma 2 (Unique Extractability)**.** *Let* $\mathsf{TAS}_{\mathsf{Sparkle},\mathcal{R}}$ *be defined as in Construction 1. Then it achieves unique extractability in the random oracle model, up to the probability that the adversary finds two random oracle queries that collide in the sense of Equation (1).*

Proof of Lemma 2. To prove the unique extractability of Construction 1, we show that it is impossible to construct two distinct signatures, which both extract with the same pre-signature into a valid witness. The first observation of this impossibility is that sparkle is defined w.r.t. a bijective hard relation. I.e., each statement s corresponds to exactly one single witness w. Therefore, if there exist two distinct signatures $\sigma = (\sigma_1, \sigma_2) \neq \sigma' = (\sigma_1', \sigma_2')$ that both extract with a pre-signature $\widetilde{\sigma} = (\widetilde{\sigma_1}, \widetilde{\sigma_2})$, then it holds, that :

$$\sigma_2 - \widetilde{\sigma_2} = w = \sigma_2' - \widetilde{\sigma_2},$$

which implies that $\sigma_2 = \sigma_2'$. Both signatures verify under the same key on the same message, so $\sigma_1 \mathsf{pk}^{\mathsf{H}(\mathsf{pk},m,\sigma_1)} = g^{\sigma_2} = \sigma_1' \mathsf{pk}^{\mathsf{H}(\mathsf{pk},m,\sigma_1')}$. For $\sigma_1 \neq \sigma_1'$ this means

$$\mathsf{H}(\mathsf{pk}, m, \sigma_1) - \mathsf{H}(\mathsf{pk}, m, \sigma_1') = \log_{\mathsf{pk}}(\sigma_1'/\sigma_1), \tag{1}$$

which ties the outputs of two random oracle queries to a value fixed before the second query is answered. An adversary making q_{H} queries meets this condition with probability at most q_{H}^2/p, so $\sigma_1 = \sigma_1'$ except with negligible probability. This implies that the winning condition of unique extractability cannot be met for Construction 1. □

Lemma 3 (Unlinkability)**.** *Let* $\mathsf{TAS}_{\mathsf{Sparkle},\mathcal{R}}$ *be defined as in Construction 1. Then* $\mathsf{TAS}_{\mathsf{Sparkle},\mathcal{R}}$ *achieves unlinkability.*

Proof. We show that the output σ_b of the challenge oracle is identically distributed for $b = 0$ and $b = 1$. Fix a challenge query $(m, (s, w), \mathcal{S})$ and recall that the challenge oracle executes both sessions internally, running every signer in $\mathcal{S}$ honestly. In the signing session, every signer samples its nonce share uniformly at random, so the ordinary signature has the form $\sigma_1 = (\widetilde{R}', r' + \mathsf{sk} \cdot h')$, where

$\widetilde{R}' = g^{r'}$ is uniformly distributed in $\mathbb{G}$ and $h' = \mathsf{H}_{\mathsf{Sign}}(\mathsf{pk}, m, \widetilde{R}')$. In the pre-signing session, the combined pre-signature has the form $\widetilde{\sigma} = (\widetilde{R} \cdot s, r + \mathsf{sk} \cdot h)$ with $\widetilde{R} = g^r$ uniformly distributed in $\mathbb{G}$ and $h = \mathsf{H}_{\mathsf{Sign}}(\mathsf{pk}, m, \widetilde{R} \cdot s)$, and adapting with w yields $\sigma_0 = (\widetilde{R} \cdot s, r + w + \mathsf{sk} \cdot h)$. The first component $\widetilde{R} \cdot s$ is uniformly distributed in $\mathbb{G}$ because $\widetilde{R}$ is, and since $s = g^w$, its discrete logarithm is $r + w$. Both σ_0 and σ_1 therefore consist of a uniformly random group element A together with the second component $\log_g(A) + \mathsf{sk} \cdot \mathsf{H}_{\mathsf{Sign}}(\mathsf{pk}, m, A)$. Hence, the two distributions are identical, the challenge output carries no information about b, and Construction 1 achieves unlinkability. □

Lemma 4 (Pre-Signature Adaptability)**.** *Let* $\mathsf{TAS}_{\mathsf{Sparkle},\mathcal{R}}$ *be defined as in Construction 1. Then* $\mathsf{TAS}_{\mathsf{Sparkle},\mathcal{R}}$ *achieves pre-signature adaptability.*

Proof. A Schnorr signature $\sigma = (R, z) = (g^r, \mathsf{sk} \cdot h + r)$ with $h = \mathsf{H}(\mathsf{pk}, R, m)$ verifies, if $\mathsf{pk}^h \cdot R = g^z$. A pre-signature $\widetilde{\sigma} = (R \cdot s, z) = (g^r, \mathsf{sk} \cdot h + r)$ with $h = \mathsf{H}(\mathsf{pk}, R \cdot s, m)$ of Construction 1 pre-verifies, if $\mathsf{pk}^h \cdot R \cdot s = g^z \cdot s$. Therefore, each adapted pre-signature (i.e., $(R \cdot s, z + w)$) verifies if and only if it pre-verifies as pre-signature since $g^z \cdot s = g^{z+w}$. □

Lemma 5 (Computational Pre-Verify Soundness)**.** *Let* $\mathsf{TAS}_{\mathsf{Sparkle},\mathcal{R}}$ *be defined as in Construction 1. Then* $\mathsf{TAS}_{\mathsf{Sparkle},\mathcal{R}}$ *achieves pre-verify soundness.*

Proof. For the DLog relation over $\mathbb{G}$, every element of $\mathbb{G}$ has a witness. Hence, the language of the relation is $\mathbb{G}$ itself. Thus, every statement considered by the construction is in the language, which directly gives computational pre-verify soundness. □

6 Certified Witness Encryption

We introduce a new cryptographic primitive, which we call *certified witness encryption* (CWE). At a high level, CWE is a form of witness encryption where a valid signature serves as the witness required for decryption. More precisely, in a CWE scheme, a party provides a certificate $(\mathsf{vk}, m) = s$ and receives an encryption key ek. Any ciphertext encrypted under ek can only be decrypted by someone who possesses a valid signature σ on m with respect to vk. This naturally defines a witness encryption scheme for the language

$$\mathcal{L}_{\mathcal{R}} := \{((\mathsf{vk}, m), \sigma) = (s, w) \mid \mathsf{Vrfy}(\mathsf{vk}, \sigma, m) = 1\}.$$

The security of CWE aligns with the standard notion of Chosen Plaintext Attack (CPA) security for encryption schemes, with one difference, the adversary is allowed to choose the message m defining an instance of the language $\mathcal{L}_{\mathcal{R}}$.

We formalize CWE, define its corresponding security notion, and present a construction based on Schnorr signatures and ElGamal encryption. Additionally, we provide a general framework for constructing CPA-secure CWE encryption from a broad class of signature schemes in Appendix B. Our framework applies to signatures based on the DDH and CDH assumptions, including the Camenisch-Lysyanskaya (CL) [CL03], Waters+ [BSW06], and Katz-Wang [KW03] schemes.

Definition of Certified Witness Encryption. A CWE scheme is defined w.r.t. some signature scheme $\Pi_\Sigma = (\mathsf{KGen}_\Sigma, \mathsf{Sign}, \mathsf{Vrfy})$ for the relation $\mathcal{L}_{\mathcal{R}}$ as defined above. The key generation algorithm $\mathsf{KG}_{\mathsf{CWE}}$ takes as input the statement s and outputs an encryption key ek only. The encryption algorithm $\mathsf{Enc}_{\mathsf{CWE}}$ encrypts messages under ek, and the resulting ciphertexts can be decrypted using

a witness w. We emphasize that, while statements for $\mathcal{L}_{\mathcal{R}}$ can be efficiently sampled, generating valid statement-witness pairs (s, w) efficiently requires knowledge of a signing key corresponding to the verification key vk.

Definition 19 (Certified Witness Encryption)**.** *A* certified witness encryption *scheme* $\Pi_{\mathsf{CWE}} = (\mathsf{KG}_{\mathsf{CWE}}, \mathsf{Enc}_{\mathsf{CWE}}, \mathsf{Dec}_{\mathsf{CWE}})$ *w.r.t. a signature scheme* Π_{Sign} *and the hard relation* $\mathcal{R}$ *corresponding to the language* $\mathcal{L}_{\mathcal{R}}$ *as defined above consists of the following efficient algorithms:*
$\underline{\mathsf{ek} \leftarrow \mathsf{KG}_{\mathsf{CWE}}(s)}$*: The* key generation *algorithm reads a statement* $s \in \mathcal{L}_{\mathcal{R}}$ *as input and outputs an encryption key* ek*.*
$\underline{ct \leftarrow \mathsf{Enc}_{\mathsf{CWE}}(\mathsf{ek}, m)}$*: The* encryption *algorithm reads an encryption key* ek *and a message* m *as input and outputs a ciphertext* c*.*
$\underline{m \leftarrow \mathsf{Dec}_{\mathsf{CWE}}(w, ct)}$*: The* decryption *algorithm reads a witness* w *and a ciphertext* c *as input and it outputs a message* m*.*

Let $\Pi_{\Sigma} = (\mathsf{KGen}_{\Sigma}, \mathsf{Sign}, \mathsf{Vrfy})$ be a signature scheme for the relation $\mathcal{R}$ as defined above. A certified witness-encryption scheme $\Pi_{\mathsf{CWE}} = (\mathsf{KG}_{\mathsf{CWE}}, \mathsf{Enc}_{\mathsf{CWE}}, \mathsf{Dec}_{\mathsf{CWE}})$ is *correct*, if for all security parameter λ, all pairs $(s, w) \leftarrow \mathsf{genR}(\lambda)$, all $\mathsf{ek} \leftarrow \mathsf{KG}_{\mathsf{CWE}}(s)$, all messages m, we have $m = \mathsf{Dec}_{\mathsf{CWE}}(w, \mathsf{Enc}_{\mathsf{CWE}}(\mathsf{ek}, m))$.

In cases where we need verifiability of our encryption, we extend the scheme with two additional algorithms $(\mathsf{VEnc}_{\mathsf{CWE}}, \mathsf{Vrfy}_{\mathsf{CWE}})$, yielding a *verifiable* certified witness encryption scheme, defined as follows.

Definition 20 (Verifiable Certified Witness Encryption)**.** *A* verifiable certified witness encryption *scheme extends a certified witness encryption scheme with two additional algorithms:*
$\underline{ct \leftarrow \mathsf{VEnc}_{\mathsf{CWE}}(\mathsf{ek}, m, M)}$*: The* verifiable encryption *algorithm reads an encryption key* ek*, a message* m*, and a commitment* M *to* m*, and outputs a ciphertext* ct*.*
$\underline{b \leftarrow \mathsf{Vrfy}_{\mathsf{CWE}}(\mathsf{ek}, M, ct)}$*: The* verification *algorithm reads an encryption key* ek*, a commitment* M*, and a verifiable ciphertext* ct*. It outputs a bit* $b \in \{0, 1\}$ *indicating whether* ct *is a well-formed encryption of a valid opening of* M*.*

We can lift CWE schemes to the verifiable setting by using simulation-extractable NIZKs with transparent setup. We provide references for concrete instantiations and describe the corresponding relations in Section A.5.

Security of Certified Witness Encryption. More precisely, we consider the hard relation $\mathcal{R}(\lambda, m)$ with auxiliary information m for the language

$$\mathcal{L}_{\mathcal{R}} := \{((\mathsf{vk}, m), \sigma) = (s, w) \text{ s.t. } \mathsf{Vrfy}(\mathsf{vk}, \sigma, m) = 1\}.$$

Since every CWE is defined with respect to a signature scheme Π_{Σ}, we give the attacker $\mathcal{A}$ the public key vk of the signature scheme as input. The attacker outputs two equal-length messages m_0, m_1 and one message m for the key derivation. It receives the encryption c_b of the message m_b under ek as well as the statement s as input and outputs its choice bit b. Our definition is sufficient for our purposes, and we leave stronger notions of security, e.g., ones where the adversary has access to the signing oracle, open for future research.

Definition 21 (CPA Security of CWE)**.** *Let* $\mathcal{R}$ *be a hard relation for the language* $\mathcal{L}_{\mathcal{R}}$*, and* $\Pi_{\Sigma} = (\mathsf{KGen}_{\Sigma}, \mathsf{Sign}, \mathsf{Vrfy})$ *be a signature scheme. A* certified witness encryption *scheme* $\Pi_{\mathsf{CWE}} = (\mathsf{KG}_{\mathsf{CWE}},$

$\mathsf{CWE} - \mathsf{CPA}^{b}_{\mathcal{A},\Pi_{\mathsf{CWE}},\Pi_{\Sigma},\mathcal{R}}(\lambda)$

1 : $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KGen}_{\Sigma}(\lambda)$
2 : $(m_0, m_1, m) \leftarrow \mathcal{A}(\mathsf{vk})$ // *s.t.* $|m_0| = |m_1|$
3 : $(s, w) \leftarrow \mathsf{genR}(\lambda, \mathsf{vk}, m)$; $\mathsf{ek} \leftarrow \mathsf{KG}_{\mathsf{CWE}}(s)$; $c_b \leftarrow \mathsf{Enc}_{\mathsf{CWE}}(\mathsf{ek}, m_b)$
4 : $b' \leftarrow \mathcal{A}(c_b, s)$
5 : **return** b'

Figure 9: The security experiment $\mathsf{CWE} - \mathsf{CPA}^{b}_{\mathcal{A},\Pi_{\mathsf{CWE}}}(\lambda)$.

$\mathsf{Enc}_{\mathsf{CWE}}, \mathsf{Dec}_{\mathsf{CWE}}$) *has* indistinguishable encryptions under a chosen-plaintext attack *if for every* PPT *adversary* $\mathcal{A}$*, there exists a negligible function* negl *such that*

$$\left| \Pr\left[\mathsf{CWE} - \mathsf{CPA}^{1}_{\mathcal{A},\Pi_{\mathsf{CWE}}}(\lambda) = 1\right] - \Pr\left[\mathsf{CWE} - \mathsf{CPA}^{0}_{\mathcal{A},\Pi_{\mathsf{CWE}}}(\lambda) = 1\right] \right| \leq \mathsf{negl}(\lambda),$$

where the experiment $\mathsf{CWE} - \mathsf{CPA}^{b}_{\mathcal{A},\Pi_{\mathsf{CWE}}}$ *is described in Figure 9, and the probabilities are taken over the random choices of all probabilistic algorithms.*

We propose a certified witness encryption scheme for Schnorr signatures. This scheme adapts the ElGamal public-key encryption scheme but uniquely incorporates Schnorr signatures [Sch91] to serve as decryption witnesses.

CWE for Schnorr Signatures. Schnorr signatures are defined within a group $\mathbb{G}$ of prime order p with a generator g and a hash function H. For a given verification key $\mathsf{vk} = g^{\mathsf{sk}}$ and message m, a Schnorr signature has the form $\sigma = (R, z) = (g^r, \mathsf{sk} \cdot h + r)$, where h is obtained from $\mathsf{H}(\mathsf{vk}, R, m)$ and r is a randomly chosen element of $\mathbb{Z}_p$. Verifying a Schnorr signature involves checking whether $\mathsf{vk}^h \cdot R$ equals g^z.

Given the intrinsic nature of Schnorr signatures, where each message-key pair can generate an exponential number of witnesses (i.e., different signatures corresponding to each element of randomness in $\mathbb{Z}_p$), we design our construction to avoid the need to decrypt for every potential witness. I.e., we formulate a specific hard relation for Schnorr signatures that possesses unique witnesses. The statement of the relation includes the verification key vk, the message m, and a commitment to the randomness R. The witness for such a statement is a Schnorr signature (R, z), incorporating the random commitment R. More formally, we define the hard relation w.r.t. the language

$$\mathcal{L}_{\mathcal{R}} := \{((\mathsf{vk}, m, R), (R, z)) = (s, w) \text{ s.t. } \mathsf{Vrfy}(\mathsf{vk}, (R, z), m) = 1\}.$$

This language refines the one above by adding the nonce R to the statement, and the definitions above apply to it verbatim. This relation is as hard as computing a valid Schnorr signature on the verification key vk, the message m, and the random commitment R. Next, we build a CWE scheme for this relation.

To compute encryption keys, we observe the structure of the verification algorithm of Schnorr signatures. Given the values (vk, m, R) (which can all be made public without allowing for signature forgeries), one can locally compute the value $\mathsf{vk}^h \cdot R$, which equals g^z. This means, the pair $(\mathsf{ek}, z) = (\mathsf{vk}^h \cdot R, z)$ is a standard ElGamal key pair. While everyone can compute the public key ek, computing

$\mathsf{KG}_{\mathsf{CWE}}(\mathsf{vk}, m, R)$	$\mathsf{Enc}_{\mathsf{CWE}}(\mathsf{ek}, m_{\mathsf{Enc}})$	$\mathsf{Dec}_{\mathsf{CWE}}(\sigma, c)$
1 : $h \leftarrow \mathsf{H}(\mathsf{vk}\|\|R\|\|m)$	1 : $y \leftarrow\!\!\$\ \mathbb{Z}_p$	1 : parse c as (c_1, c_2)
2 : $\mathsf{ek} := \mathsf{vk}^h \cdot R$	2 : **return** $(g^y, \mathsf{ek}^y \cdot m_{\mathsf{Enc}})$	2 : parse σ as (R, z)
3 : **return** ek		3 : **return** $c_2 / (c_1^z)$

Figure 10: Schnorr-based CWE encryption from DDH. In Appendix B, we extend this construction into a "signed" setting by leveraging the random oracle (RO) to mask encrypted messages. The use of the RO enables a security proof based on the Computational Diffie-Hellman (CDH) assumption.

its discrete logarithm z from (vk, m, R) amounts to forging a Schnorr signature (R, z) on m. Hence, only the signer or a party that has seen the signature knows the decryption key z. The encryption of a message m_{Enc} using the encryption key ek follows a process similar to ElGamal encryption. We sample a random scalar $y \leftarrow\!\!\$\ \mathbb{Z}_p$ and output the ciphertext $(g^y, \mathsf{ek}^y \cdot m_{\mathsf{Enc}})$. To decrypt a ciphertext (c_1, c_2) using a signature (R, z), we compute $m'_{\mathsf{Enc}} = c_2/(c_1^z)$. More formally, we construct CWE for Schnorr signatures in Construction 2.

Construction 2 (DDH-Based CWE encryption scheme)**.** *Let $\mathbb{G}$ be a group of prime order p with a generator g. We define the* CWE *scheme $\Pi_{\mathsf{CWE}} = (\mathsf{KG}_{\mathsf{CWE}}, \mathsf{Enc}_{\mathsf{CWE}}, \mathsf{Dec}_{\mathsf{CWE}})$ for Schnorr signatures w.r.t. $(\mathbb{G}, p, g)$ in Figure 10.*

This construction is correct and CPA secure. We show correctness in Lemma 6 and CPA security in Theorem 2. Our exchange protocol encrypts partial pre-signatures, which are elements of $\mathbb{Z}_p$. For that plaintext space we encrypt as $(g^y, \mathsf{H}_{\mathsf{Enc}}(\mathsf{ek}^y) + m_{\mathsf{Enc}})$ with a hash function $\mathsf{H}_{\mathsf{Enc}}$ mapping into $\mathbb{Z}_p$, and decryption subtracts $\mathsf{H}_{\mathsf{Enc}}(c_1^z)$. Modeling $\mathsf{H}_{\mathsf{Enc}}$ as a random oracle, CPA security follows from the CDH assumption, as shown in Theorem 7.

Theorem 2 (Security of Schnorr CWE)**.** *If the decisional Diffie-Hellman (DDH) assumption holds, and* H *is a compressing, non-zero hash function, then Construction 2 is a CPA secure* CWE *scheme.*

Lemma 6. *The Schnorr* CWE *scheme in Construction 2 is correct.*

Proof. To verify correctness of Construction 2, we have to check, that for all messages m and statement-witness pairs $((\mathsf{vk}, m, R), (R, z)) \in \mathcal{R}$, with $\mathsf{vk} = g^{\mathsf{sk}}$ and $R = g^r$ it holds that $m = \mathsf{Dec}_{\mathsf{CWE}}(w, \mathsf{Enc}_{\mathsf{CWE}}(\mathsf{ek}, m))$. If $\sigma = (R, z)$ is a valid signature w.r.t. (vk, m) it holds that $\mathsf{vk}^h \cdot R = g^z$ for $h = \mathsf{H}(\mathsf{vk}||R||m)$. Therefore, we have that for all (c_1, c_2) output by $\mathsf{Enc}_{\mathsf{CWE}}$, that

$$\frac{c_2}{c_1^z} = \frac{\mathsf{ek}^y \cdot m}{g^{y \cdot z}} = \frac{g^{(\mathsf{sk} \cdot h + r) \cdot y} \cdot m}{g^{y \cdot (\mathsf{sk} \cdot h + r)}} = m.$$

Henceforth, Construction 2 is correct. □

Proof of Theorem 2. By contradiction, we assume that there exists an adversary $\mathcal{A}$ that breaks the CPA security of Π_{CWE} with non-negligible probability. We use this adversary to build a reduction from the CPA security of Π_{CWE} to the hardness of DDH. The reduction $\mathcal{R}$ gets as input the values $([\alpha], [\beta], [\gamma])$, such that γ is either a random value or $\alpha \cdot \beta$. The reduction forwards $[\alpha]$ as verification key to $\mathcal{A}$, which eventually outputs three messages (m_0, m_1, m). $\mathcal{R}$ samples a fresh randomness,

commitment pair $(R, r) \leftarrow \mathsf{genR}(\lambda)$ and computes the challenge ciphertext on the challenge message m_b via

$$c_b \leftarrow ([\beta], [\beta]^r \cdot [\gamma]^h \cdot m_b).$$

The reduction forwards c_b to $\mathcal{A}$, which eventually outputs a bit b' and $\mathcal{R}$ returns $b' == b$.

Claim 3. *If $\gamma \neq \alpha \cdot \beta$, then $\mathcal{A}$ wins with probability at most $1/2$.*

Proof. If $\gamma \neq \alpha \cdot \beta$, then γ is a random element in $\mathbb{Z}_p$. Thus, the challenge ciphertext c_b is also a random element in $\mathbb{Z}_p$ (as H is a non-zero function) and is independent of the challenge message m_b. So, the claim follows. □

Claim 4. *If $\gamma = \alpha \cdot \beta$, then $\mathcal{A}$ wins with non-negligible advantage ε.*

Proof. If $\gamma = \alpha \cdot \beta$, then the ciphertext is exactly constructed as the adversary expects it to be:

$$\begin{aligned} c := ([\beta], [\beta]^r \cdot [\gamma]^h \cdot m_b) &= ([\beta], [\beta \cdot r + \alpha \cdot \beta \cdot h] \cdot m_b) \\ &= ([\beta], [\beta \cdot (r + \alpha \cdot h)] \cdot m_b). \end{aligned}$$

As we assume by contradiction that $\mathcal{A}$ breaks the CPA security of the encryption scheme with non-negligible probability, this probability carries over if $\gamma = \alpha \cdot \beta$. □

The reduction samples the challenge bit b itself and outputs 1 whenever $\mathcal{A}$ returns b. A CPA advantage of ε makes $\mathcal{A}$ return b with probability $\frac{1}{2} + \frac{\varepsilon}{2}$ if $\gamma = \alpha \cdot \beta$, and with probability $\frac{1}{2}$ otherwise, so

$$\left|\Pr\left[\mathsf{DDH}^0_{\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathsf{DDH}^1_{\mathcal{A}}(\lambda) = 1\right]\right| \geq \left|\frac{1}{2} + \frac{\varepsilon}{2} - \frac{1}{2}\right| \geq \frac{\varepsilon}{2}.$$

This contradicts the hardness of DDH, so no efficient adversary against the CPA security of Construction 2 can exist.

□

7 A Fair Distributed Exchange Protocol

We present a protocol that fairly exchanges a coin for a DLog witness on a blockchain that verifies Schnorr signatures. As building blocks, we use threshold Schnorr adaptor signatures, verifiable certified witness encryption for Schnorr signatures, and an additional verifiable public-key encryption scheme for $\mathbb{Z}_p$, which encrypts blinding values to the sellers. Our protocol proceeds in three phases: (i) Commit phase. All buyers jointly compute a pre-signature and forward it to the sellers. (ii) Settle phase. The sellers jointly adapt the pre-signature, and post the result to the blockchain. (iii) Reveal phase. Each buyer reads the blockchain and locally extracts the witness w.

As discussed in Section 2.2, we provide two protocol versions that differ in the settle phase. The first protocol uses deposits amongst the sellers (we therefore refer to it as $\mathsf{SettleDeposit}$), and the second protocol (Settle) assumes an honest majority of participating sellers.

$\mathsf{Setup}_{\mathsf{B}}(t_{\mathsf{B}}, n_{\mathsf{B}})$

1 : $k \leftarrow\$ \{0,1\}^{\lambda}$; $\mathcal{B} = \{1, \ldots, n_{\mathsf{B}}\}$
2 : $par_{\mathcal{B}} \leftarrow \mathsf{TS.Setup}(n_{\mathsf{B}}, t_{\mathsf{B}})$
3 : $(\mathsf{pk}_{\mathsf{B}}, \{\mathsf{sk}_{\mathsf{TS},i}\}_{i\in\mathcal{B}}) \leftarrow \mathsf{TS.KGen}(par_{\mathcal{B}})$
4 : **for** $i \in \mathcal{B}$: $\mathsf{sk}_{\mathsf{B}_i} \leftarrow (\mathsf{sk}_{\mathsf{TS},i}, k)$
5 : **return** $(\mathsf{pk}_{\mathsf{B}}, \{(g^{\mathsf{sk}_{\mathsf{TS},i}}, \mathsf{sk}_{\mathsf{B}_i})\}_{i\in\mathcal{B}})$

$\mathsf{Setup}_{\mathsf{S}}(t_{\mathsf{S}}, n_{\mathsf{S}}, s, w)$

1 : $\mathcal{S} = \{1, \ldots, n_{\mathsf{S}}\}$
2 : $\{w_j\}_{j\in\mathcal{S}} \leftarrow \mathsf{SShare}(w, t_{\mathsf{S}}, n_{\mathsf{S}})$; **for** $j : s_j \leftarrow g^{w_j}$
3 : **for** $j \in \mathcal{S}$:
4 : $\quad(\mathsf{pk}_{\mathsf{Enc},j}, \mathsf{sk}_{\mathsf{Enc},j}) \leftarrow \Pi_{\mathsf{Enc}}.\mathsf{KGen}(\lambda)$
5 : $\quad(\mathsf{pk}_{\Sigma,j}, \mathsf{sk}_{\Sigma,j}) \leftarrow \Sigma.\mathsf{KGen}(\lambda)$
6 : $\quad\mathsf{sk}_{\mathsf{S}_j} \leftarrow (\mathsf{sk}_{\Sigma,j}, \mathsf{sk}_{\mathsf{Enc},j})$
7 : $\quad\mathsf{pk}_{\mathsf{S}_j} \leftarrow (\mathsf{pk}_{\Sigma,j}, \mathsf{pk}_{\mathsf{Enc},j})$
8 : **return** $\{(\mathsf{pk}_{\mathsf{S}_j}, \mathsf{sk}_{\mathsf{S}_j}, s, s_j, w_j)\}_{j\in\mathcal{S}}$

Figure 11: The setup algorithms.

Construction 3 (Schnorr-Based FDE)**.** *Let $\mathcal{R}$ be the DLog relation, and* TAS *be the threshold adaptor signature based on* Sparkle*. Let* CWE *be the verifiable certified witness encryption scheme for Schnorr signatures with plaintext space $\mathbb{Z}_p$ of Construction 2, and let Π_{Enc} be a verifiable public-key encryption scheme for plaintext space $\mathbb{Z}_p$. Let $\mathbb{B}$ be a scriptless blockchain with timelocks that verifies Schnorr signatures. We define the algorithms and protocols* Setup, Commit, SettleDeposit, Settle*, and* Reveal *in Figures 11 to 15.*

Setup. The setup algorithms generate the private and public keys for all participants in the protocol and share the witness amongst the sellers using Shamir secret sharing. For the buyer group setup, we first run key generation of the threshold adaptor signature with the respective thresholds $(t_{\mathsf{B}}, n_{\mathsf{B}})$, which creates a joint key pk_{B} and threshold signing key-verification key shares $(\mathsf{sk}_{\mathsf{TS},i}, \mathsf{pk}_{\mathsf{TS},i})$ for each buyer B_i. In addition, each buyer receives a random value $k \leftarrow\$ \{0,1\}^{\lambda}$, identical for all buyers. Together they form the buyer's secret key $\mathsf{sk}_{\mathsf{B}_i} = (\mathsf{sk}_{\mathsf{TS},i}, k)$.

The seller group setup receives a statement-witness pair (s, w),and threshold parameters $(t_{\mathsf{S}}, n_{\mathsf{S}})$. It generates encryption keys $(\mathsf{pk}_{\mathsf{Enc},j}, \mathsf{sk}_{\mathsf{Enc},j})$ and signing keys $(\mathsf{pk}_{\Sigma,j}, \mathsf{sk}_{\Sigma,j})$ for each seller. These key-pairs are combined into the seller's secret key $\mathsf{sk}_{\mathsf{S}_j} = (\mathsf{sk}_{\Sigma,j}, \mathsf{sk}_{\mathsf{Enc},j})$, and the public key $\mathsf{pk}_{\mathsf{S}_j} = (\mathsf{pk}_{\mathsf{Enc},j}, \mathsf{pk}_{\Sigma,j})$. Additionally, the seller group setup shares w using Shamir secret sharing via $\{w_j\}_{j\in\mathcal{S}} \leftarrow \mathsf{SShare}(w, t_{\mathsf{S}}, n_{\mathsf{S}})$. It outputs the respective secret key and witness share to each seller, and all public keys and the statement to all sellers. From this point forward, we assume that all public keys and the statement are known to all parties.

Commit. The commit protocol is executed by the buyers. Its goal is to jointly compute a pre-signature that can be sent to the sellers. To ensure buyer fairness, the Commit protocol is designed so that when any buyer obtains a pre-signature, and it is eventually adapted and posted, every other buyer has a local state that allows them to recover the same pre-signature.

To ensure witness privacy, the Commit protocol computes a pre-signature on a blinded statement $s' = s \cdot g^e$ that has the witness $w' = w + e$, where the blinding e is a pseudorandom value derived from the buyers' shared key k. We use a blinded statement instead of s, since given a pre-signature–signature pair, each party can locally extract the used witness. This would also allow the sellers to learn from the witness during an exchange. Our blinding avoids this, since with it every party can

$\mathsf{B}_i.\mathsf{Commit}^{\mathbb{B}}(\mathsf{sk}_{\mathsf{B}_i}, s, \mathsf{tx}, \mathtt{t})$

1 : // Compute rerandomization factors.
2 : $(e, \{e_j\}_{j\in\mathcal{S}}) \leftarrow \mathsf{H}(k, \mathsf{tx}, s);\ E \leftarrow g^e$

3 : // Agree on a pre-signature randomness $\widetilde{R}$
4 : $(\rho_i, \mathsf{st}_i) \leftarrow \mathsf{pSign}^0(i, \mathsf{tx}, s \cdot E, \mathcal{B})$
5 : send ρ_i to all B_j; upon receiving all other ρ_i
6 : $(\rho'_i, \mathsf{st}_i) \leftarrow \mathsf{pSign}^1(\mathsf{st}_i, \{\rho_i\}_{i\in\mathcal{B}}, \mathcal{B})$
7 : send ρ'_i to all B_j; upon receiving all other ρ'_i
8 : $\widetilde{R} \leftarrow \prod_{j\in\mathcal{B}} \rho'_j \cdot E \cdot s$
9 : $\mathbb{B}.\mathsf{Lock}(\mathsf{tx}, \mathsf{sk}_{\mathsf{B}_i}, 3 + \mathtt{t})$ // Lock tx for the 3 remaining rounds plus t

10 : // Encrypt the pre-signature share using the CWE scheme
11 : $\mathsf{ek} \leftarrow \mathsf{KG}_{\mathsf{CWE}}(\mathsf{pk}, \mathsf{tx}, \widetilde{R})$
12 : $\rho''_i \leftarrow \mathsf{pSign}^2(\mathsf{st}_i, \mathsf{sk}_{\mathsf{B}_i}, \{\rho'_i\}_{i\in\mathcal{B}}, \mathcal{B})$
13 : $com_i \leftarrow \mathsf{VEnc}_{\mathsf{CWE}}(\mathsf{ek}, \rho''_i)$
14 : send com_i to all B_j

15 : // Reveal the pre-signature share after verifying all commitments
16 : upon receiving all other com_j :
17 : **assert** $\mathsf{Vrfy}_{\mathsf{CWE}}(com_j)$
18 : send ρ''_i to all B_j; upon receiving all other ρ''_j :
19 : $\widetilde{\sigma} \leftarrow \mathsf{Combine}(\{\rho'_i, \rho''_i\}_{i\in\mathcal{B}});\ M \leftarrow (E, g^{e_1}, \ldots, g^{e_{n_\mathsf{S}}})$
20 : $\mathsf{state}_i \leftarrow (\{com_i\}_{i\in\mathcal{B}}, e); \rho \leftarrow (\widetilde{\sigma}, \mathsf{tx}, E, M, \{\Pi_{\mathsf{Enc}}.\mathsf{VEnc}(\mathsf{pk}_{\mathsf{S}_j}, e_j)\}_{1\le j\le n_\mathsf{S}})$
21 : **return** (state_i, ρ)

Figure 12: Construction of the Commit protocol.

extract the re-randomized witness w', but only the buyers can remove the blinding e to recover the true witness w.

The Commit protocol begins by computing the blinding factor e. Each buyer computes $(e, e_1, \ldots, e_{n_\mathsf{S}}) \leftarrow \mathsf{H}(k, \mathsf{tx}, s)$, where k is the key shared amongst the buyers, tx is the transaction and s is the statement (Lines 1-2, Figure 12). Here, the buyers interpret the output of H as the coefficients of a polynomial f of degree $t_\mathsf{S} - 1$ and set $e = \mathsf{f}(0)$ and the share of seller j to $e_j = \mathsf{f}(j)$. Looking slightly ahead, in addition to a pre-signature, the buyers will later forward ciphertexts $\{\Pi_{\mathsf{Enc}}.\mathsf{VEnc}(\mathsf{pk}_{\mathsf{S}_j}, e_j)\}_{j\in\mathcal{S}}$ to the sellers (Line 20, Figure 12). From these ciphertexts, each seller can decrypt its share e_j. The encryption is verifiable w.r.t. the commitments $M = (g^e, g^{e_1}, \ldots, g^{e_{n_\mathsf{S}}})$, which the buyers also forward, so each seller can check that its share is well-formed before it uses it (see VerifyInputs below). In addition, the value e can be reconstructed by any buyer (as buyers know k), and the blinded witness $w' = w + e$ by a quorum of at least t_S sellers.

Once $E = g^e$ is fixed, buyers jointly generate pre-signature randomness $\widetilde{R}$ using the threshold adaptor signature protocols pSign^0 and pSign^1, and lock the transaction tx using their signing keys (cf. lines 3-9, Figure 12). The lock lasts $3 + \mathtt{t}$ rounds, i.e., for the three remaining rounds of Commit plus $\mathtt{t}$ rounds for the settle phase, where $\mathtt{t}$ is a protocol parameter. This locking ensures liveness, since either the sellers complete the protocol by adapting the pre-signature and spending the transaction, or the buyers automatically regain access to the locked funds once the timeout expires. Each buyer then computes a partial pre-signature using pSign^2, verifiably encrypts it with the CWE scheme, and sends the resulting encrypted partial pre-signature to the other buyers. This encryption using the CWE scheme can already be computed, since the nonce $\widetilde{R}$ of the adapted signature is known at this point. After receiving and verifying sufficient ciphertexts from other buyers, each buyer is convinced that once the adapted pre-signature is posted on-chain, they can decrypt the pre-signature shares of enough other buyers to recover the pre-signature. This holds since ek is derived from $(\mathsf{pk}_\mathsf{B}, \mathsf{tx}, \widetilde{R})$, so the adapted signature on tx with nonce $\widetilde{R}$ decrypts the ciphertexts, and their proofs guarantee valid pre-signature shares. Thus, buyers continue and reveal their pre-signature shares such that each buyer learns the pre-signature $\widetilde{\sigma}$ (Lines 15-18, Figure 12).

At the end of the protocol, each buyer stores the verified pre-signature ciphertexts in its local state state, while the sellers receive the pre-signature, the transaction message, the blinding commitments M, and the blinding shares, verifiably encrypted with Π_{Enc} (cf. Section A.5).

Depending on the settle protocol, the timing parameter $\mathtt{t}$ and the transaction tx of the commit phase differ. The protocol with an honest majority uses $\mathtt{t} = 3$ and a tx that pays each seller an equal share of the coin, while the protocol with deposits uses $\mathtt{t} = 2 \cdot n_\mathsf{S}$ and a tx that pays the whole coin to the last seller.

Settle. We now describe both our settle protocols that are run by the sellers. The goal of each settle protocol is either to jointly sell the sellers' witness shares or to provide no information about them at all.

In both our settle protocols, each seller parses and checks the state ρ_i that the buyers output at the end of Commit before it sends any secret material, using the subroutine VerifyInputs of Figure 13. The checks confirm that the state refers to the transaction tx, that the pre-signature pre-verifies under pk_B w.r.t. the blinded statement $s \cdot E$, that every verifiable encryption of a blinding share is valid, and that the coins stay locked for at least $\mathtt{t}$ more rounds. They also confirm that all ciphertexts carry the same commitment vector $(E_0, \ldots, E_{n_\mathsf{S}})$, and, by interpolation in the exponent, that this vector forms a single sharing of degree $t_\mathsf{S} - 1$ with constant term E (Line 5).

Settle with deposits. In the settle protocol with deposits, the sellers form a ladder of conditional payments. This is inspired by fair MPC with penalties [BK14, ADMM14b], where parties lock deposits such that a party that aborts after learning the output compensates the honest parties. Seller i receives a payment from seller $i + 1$ for passing on its part of the witness, while seller $i + 1$ in turn receives a larger payment from seller $i + 2$. Each seller is thus both a buyer and a seller in the ladder, except for seller 1, which only sells. The ladder starts with seller 1 and ends with seller n_S, who sells the complete witness to the buyers. Deposits ensure that each step of the ladder is funded, so that if seller i buys the witness of seller $i - 1$, seller i has the guarantee that it can sell it to seller $i + 1$.

In the first step, the sellers re-share their witness shares additively to make these payments conditional on the gradual reconstruction of the witness (Lines 5-8, Figure 14). Seller i splits its

$\mathsf{VerifyInputs}(\mathsf{pk_B}, s, \mathsf{tx}, \rho, \mathsf{t})$

1 : parse ρ as $(\widetilde{\sigma}, \mathsf{tx}', E, \{ct_j\}_{j\in\mathcal{S}})$
2 : **if** $\mathsf{tx}' \neq \mathsf{tx} \vee \mathbb{B}.\mathsf{VrfyLock}(\mathsf{tx}) < \mathsf{t} \vee \rho.M = \bot$: **return** $\bot$
3 : **if** $\mathsf{pVrfy}(\mathsf{pk_B}, \mathsf{tx}, s \cdot E, \widetilde{\sigma}) \neq 1$: **return** $\bot$
4 : **if** $\exists j \in \mathcal{S}$ s.t. $\Pi_{\mathsf{Enc}}.\mathsf{Vrfy}(ct_j) \neq 1$: **return** $\bot$
5 : // Blinding commitments must form one degree-$(t_\mathsf{S}-1)$ sharing
6 : $(E_0, E_1, \ldots, E_{n_\mathsf{S}}) \leftarrow \rho.M$
7 : **if** $E_0 \neq E \vee \exists x \in \{0, 1, \ldots, n_\mathsf{S}\} : E_x \neq \prod_{\ell=1}^{t_\mathsf{S}} E_\ell^{\lambda_\ell(x)}$:
8 : **return** $\bot$
9 : **return** $(\widetilde{\sigma}, E, \{ct_j\}_{j\in\mathcal{S}})$

Figure 13: The input check that each seller runs in both settle protocols. $\lambda_\ell(x)$ denotes the Lagrange coefficient of ℓ w.r.t. the set $\{1, \ldots, t_\mathsf{S}\}$ evaluated at x.

Lagrange-weighted share $\lambda_i \cdot w'_i$ into additive shares $a_{i,1}, \ldots, a_{i,n_\mathsf{S}}$, verifiably encrypts share $a_{i,k}$ for seller k, and broadcasts the ciphertexts together with the commitments $g^{a_{i,k}}$. The commitments allow every seller to verify that the shares of seller i add up to its original share before any coins are locked. Seller k decrypts the shares it receives and adds them up to a value b_k. The values b_k form an n_S-out-of-n_S sharing of w'.

The resulting sharing lets the ladder reveal the witness one step at a time. After seller k has revealed b_k, the first k values determine only the prefix $y_k = \sum_{\ell\leq k} b_\ell$. Only after all sellers have contributed does this prefix equal the complete witness w'. This ensures that malicious sellers cannot exclude honest sellers once they learn the share of a single honest seller. We therefore associate each step of the ladder with a partial statement whose witness is the corresponding prefix. From the commitments, every seller computes $B_k = \prod_j g^{a_{j,k}}$ and the partial statements $\mathsf{Y}_k = \prod_{\ell\leq k} B_\ell$ (Line 8). Each deposit of the ladder is then pre-signed under the corresponding partial statement.

In the second step, the deposits are locked in decreasing order (Lines 9-13). To do this, seller i deposits $(i-1) \cdot v_\mathsf{tx}/n_\mathsf{S}$ coins for seller $i-1$ under a pre-signature on Y_{i-1}. Before locking, seller i verifies that the deposit of seller $i+1$ is already locked, carries a valid pre-signature, and outlives its own deposit (Line 10).

In the third step, the deposits are redeemed in increasing order (Lines 14-17). Seller 1 reveals b_1 to claim the deposit of seller 2, seller 2 extracts that value from the resulting signature, adds b_2 and claims the next deposit, and so on until seller n_S holds w', adapts the buyers' pre-signature, and claims tx. Every seller that redeems gains $v_\mathsf{tx}/n_\mathsf{S}$, and the ladder locks $(n_\mathsf{S}-1) \cdot v_\mathsf{tx}/2$ coins in total.

Settle with honest majority. With an honest majority of sellers, sellers can exchange their witness shares directly rather than using a ladder of deposits. Since at least t_S of the participating sellers are honest, each honest seller obtains enough shares to reconstruct the blinded witness independently. The sellers first agree on their common input in a ready round (Lines 4-6, Figure 15). Every seller that passes $\mathsf{VerifyInputs}$ broadcasts the digest $\mathsf{H}(\rho_i)$ of its state, and it continues only if at least $2 \cdot t_\mathsf{S} - 1$ sellers broadcast that same digest. A seller that continues therefore knows that at least t_S honest sellers hold the same state and passed the same checks, since at most $t_\mathsf{S} - 1$ sellers

$\mathsf{S}_i.\mathsf{SettleDeposit}^{\mathbb{B}}(\mathsf{sk}_{\mathsf{S}_i}, w_i, s, \mathsf{tx}, \rho_i, \mathsf{pk}_{\mathsf{B}})$

1 : // Check the state and re-share the blinded witness
2 : $(\widetilde{\sigma}, E, \{ct_j\}_{j\in\mathcal{S}}) \leftarrow \mathsf{VerifyInputs}(\mathsf{pk}_{\mathsf{B}}, s, \mathsf{tx}, \rho_i, 2n_{\mathsf{S}})$
3 : **if** $\widetilde{\sigma} = \bot$: **return** $\bot$
4 : $e_i \leftarrow \Pi_{\mathsf{Enc}}.\mathsf{Dec}(\mathsf{sk}_{\mathsf{S}_i}, ct_i), w'_i = w_i + e_i$
5 : $a_{i,1}, \ldots, a_{i,n_{\mathsf{S}}} \leftarrow\!\!\$\ \mathbb{Z}_p$ with $\sum_k a_{i,k} = \lambda_i \cdot w'_i$
6 : **broadcast** $\{g^{a_{i,k}}, \Pi_{\mathsf{Enc}}.\mathsf{VEnc}(\mathsf{pk}_{\mathsf{S}_k}, a_{i,k})\}_{k\in\mathcal{S}}$
7 : **if** $\exists j \in \mathcal{S}$ s.t. $\prod_k g^{a_{j,k}} \neq (s_j \cdot E_j)^{\lambda_j}$: **return** $\bot$
8 : $b_i \leftarrow \sum_j \Pi_{\mathsf{Enc}}.\mathsf{Dec}(\mathsf{sk}_{\mathsf{S}_i}, ct_{j,i})$; **for** $k : B_k \leftarrow \prod_j g^{a_{j,k}}$, $\mathsf{Y}_k \leftarrow \prod_{\ell\le k} B_\ell$

9 : // Lock the deposit, in decreasing order of i
10 : **if** $\mathbb{B}.\mathsf{VrfyLock}(\mathsf{tx}_{i+1}) \le \mathsf{t}_i \vee \mathsf{pVrfy}(\mathsf{pk}_{\Sigma,i+1}, \mathsf{tx}_{i+1}, \mathsf{Y}_i, \widetilde{\sigma}_{i+1}) = 0$: **return** $\bot$
11 : **if** $i > 1$:
12 : $\quad \mathbb{B}.\mathsf{Lock}(\mathsf{tx}_i, \mathsf{sk}_{\mathsf{S}_i}, \mathsf{t}_i)$ // tx_i pays $(i-1)\cdot v_{\mathsf{tx}}/n_{\mathsf{S}}$ to $\mathsf{pk}_{\Sigma,i-1}$
13 : $\quad \widetilde{\sigma}_i \leftarrow \mathsf{pSign}(\mathsf{sk}_{\Sigma,i}, \mathsf{tx}_i, \mathsf{Y}_{i-1})$; send $\widetilde{\sigma}_i$ to all S_{i-1}

14 : // Redeem the deposit, in increasing order of i
15 : **if** $i > 1 : y_{i-1} \leftarrow \mathsf{Extract}(\mathsf{pk}_{\Sigma,i}, \widetilde{\sigma}_i, \sigma_i, \mathsf{Y}_{i-1})$
16 : $y_i \leftarrow y_{i-1} + b_i$
17 : **return** $\mathbb{B}.\mathsf{Claim}(\mathsf{tx}_{i+1}, \mathsf{Adapt}(\mathsf{pk}_{\Sigma,i+1}, \widetilde{\sigma}_{i+1}, y_i))$

Figure 14: Settling with deposits. Here, λ_ℓ is the Lagrange coefficient of ℓ w.r.t. $\mathcal{S}$ at 0, σ_i the signature that claims tx_i, and $ct_{j,i}$ the summand that S_j broadcasts for S_i. Seller i accepts tx_{i+1} only if it pays $i\cdot v_{\mathsf{tx}}/n_{\mathsf{S}}$ to $\mathsf{pk}_{\Sigma,i}$, and it chooses t_i such that it can still finalize after t_{i-1} has expired. We write $\mathsf{tx}_{n_{\mathsf{S}}+1} = \mathsf{tx}$, $\mathsf{pk}_{\Sigma,n_{\mathsf{S}}+1} = \mathsf{pk}_{\mathsf{B}}$, $\widetilde{\sigma}_{n_{\mathsf{S}}+1} = \widetilde{\sigma}$, $\mathsf{Y}_{n_{\mathsf{S}}} = s \cdot E$, $\mathsf{t}_1 = 0$, and $y_0 = 0$.

are corrupted. After the ready round succeeds, each seller sends its blinded witness share $w_i + e_i$ to all other sellers. Once a seller has collected at least t_{S} such shares, it combines them to the blinded witness $w + e$, adapts the pre-signature with it, and claims the transaction tx on the chain (Lines 9-11, Figure 15).

Reveal. The reveal algorithm allows each buyer to extract the sellers' witness w from the adapted signature σ using their internal state. This state contains verifiable certified witness-encryption (CWE) ciphertexts of the pre-signature shares contributed by the other buyers. Once the adapted signature σ appears on the chain, each buyer decrypts the CWE ciphertexts to recover enough pre-signature shares to reconstruct the full pre-signature. Our security analysis will show that whenever sellers succeed in posting a valid signature, every buyer's state contains a sufficient number of shares for this reconstruction. With the pre-signature recovered, each buyer applies the extraction algorithm of the threshold adaptor signature scheme, obtaining the blinded witness $w + e$. Since all buyers know the blinding factor e, they can remove it and efficiently recover the witness w. We provide a formal specification of the reveal algorithm in Figure 15.

$\mathsf{S}_i.\mathsf{Settle}^{\mathbb{B}}(\mathsf{sk}_{\mathsf{S}_i}, w_i, s, \mathsf{tx}, \rho_i, \mathsf{pk}_{\mathsf{B}})$

1 : // Parse and check the state received from the buyers
2 : $(\widetilde{\sigma}, E, \{ct_j\}_{j\in\mathcal{S}}) \leftarrow \mathsf{VerifyInputs}(\mathsf{pk}_{\mathsf{B}}, s, \mathsf{tx}, \rho_i, 3)$
3 : **if** $\widetilde{\sigma} = \perp$: **return** $\perp$

4 : // Ready round: agree on the common input
5 : $h_i \leftarrow \mathsf{H}(\rho_i)$; **broadcast** h_i; receive $\{h_j\}_{j\in\mathcal{S}}$
6 : **if** $|\{j \in \mathcal{S} : h_j = h_i\}| < 2t_{\mathsf{S}} - 1$: **return** $\perp$

7 : // Jointly adapt the pre-signature
8 : $e_i \leftarrow \Pi_{\mathsf{Enc}}.\mathsf{Dec}(\mathsf{sk}_{\mathsf{S}_i}, ct_i), w'_i = w_i + e_i$
9 : send w'_i to all S_j; receive back $\{w'_j\}_{j\in\mathcal{S}}$
10 : $w' \leftarrow \mathsf{SComb}(s \cdot E, \{w'_j : g^{w'_j} = s_j \cdot E_j\})$
11 : **return** $\mathbb{B}.\mathsf{Claim}(\mathsf{tx}, \mathsf{Adapt}(\mathsf{pk}_{\mathsf{B}}, \widetilde{\sigma}, w'))$

$\mathsf{Reveal}.\mathsf{B}_i(\mathsf{pk}_{\mathsf{B}}, \mathsf{state}_i, \sigma)$

13 : parse state_i as $(\{com_j\}_{j\in\mathcal{B}}, e)$
14 : $\mathsf{dk} \leftarrow \sigma$
15 : **foreach** $j \in \mathcal{B}$ **do** :
16 : $\quad \rho''_j \leftarrow \mathsf{Dec}_{\mathsf{CWE}}(\mathsf{dk}, com_j)$
17 : $\widetilde{\sigma} \leftarrow \mathsf{Combine}(\mathsf{pk}_{\mathsf{B}}, \{\rho''_j\}_{j\in\mathcal{B}})$
18 : $w' \leftarrow \mathsf{Extract}(\mathsf{pk}_{\mathsf{B}}, \widetilde{\sigma}, \sigma, s \cdot E)$
19 : **return** $w' - e$

Figure 15: Settle protocol with an honest majority and the reveal algorithm.

Security. We now establish the security of our FDE construction. For clarity of exposition, we speak of t_{S}-seller fairness and t_{B}-buyer fairness, implicitly assuming that in these settings at most $t_{\mathsf{S}} - 1$ sellers or $t_{\mathsf{B}} - 1$ buyers may be corrupted.

Theorem 3. *Let $\mathcal{R}$ be a hard relation for the DLog language in the group $\mathbb{G}$, and let* TAS *be the secure Schnorr threshold adaptor signature scheme defined in Construction 1. Let* CWE *be a verifiable certified witness encryption scheme for Schnorr signatures using a simulation-extractable* NIZK*. Let $\mathbb{B}$ be a blockchain with timelocks that verifies Schnorr signatures. Let Π_{Enc} be a CPA secure public key encryption scheme whose proof of well-formedness is a simulation-extractable* NIZK*. If the CDH assumption holds in $\mathbb{G}$, then Construction 3 guarantees the following in the random oracle model:*

- *t_{B}-buyer fairness for any $t_{\mathsf{B}} \leq n_{\mathsf{B}}$ in the presence of a scriptless blockchain,*
- *t_{S}-seller fairness in the presence of a blockchain with timelocks, either for $n_{\mathsf{S}} \geq 2 \cdot t_{\mathsf{S}} - 1$ with Figure 15, or for any n_{S} with the deposits of Figure 14,*
- *witness privacy for any $t_{\mathsf{S}} \leq n_{\mathsf{S}}$.*

We show each property in a separate lemma.

Lemma 7 (Buyer Fairness)**.** *Let $\mathcal{R}$ be a hard relation for the DLog language in the group $\mathbb{G}$, and let* TAS *be the secure Schnorr threshold adaptor signature scheme defined in Construction 1. Let* CWE *be a verifiable certified witness encryption scheme for Schnorr signatures, as defined in Construction 5, using a simulation-extractable* (NIZK)*. If the CDH assumption holds in $\mathbb{G}$, then Construction 3 achieves t_{B}-buyer fairness for any $t_{\mathsf{B}} \leq n_{\mathsf{B}}$.*

Correctness. We first record that Construction 3 is correct. In an honest execution the buyers agree on the blinding e and jointly produce a pre-signature on tx w.r.t. the blinded statement $s \cdot E$.

Both settle protocols hand the blinded witness w' to a seller who adapts that pre-signature, and the resulting signature verifies under pk_B on tx, so $\mathbb{B}$ accepts the claim. Every buyer then extracts w' from the posted signature and removes the blinding with the shared key k, which yields the witness w.

Initially, our goal was to analyze the security of the FDE scheme in relation to the CPA security of the CWE scheme. However, in our construction, a partial pre-signature is encrypted using the CWE scheme, and the adapted and combined pre-signature effectively functions as the decryption key. As a result, a direct reduction to the CPA security of CWE is not possible. Instead, we adopt the framework of transparent reductions [GSST24] and prove the security of our FDE scheme based on the hardness of the CDH assumption in the underlying group. Additionally, we ensure that the random oracle is programmed in a way that prevents circular security.

Proof of Lemma 7. To prove Lemma 7, we construct a reduction from the buyer fairness of our construction to one of the following security properties:

- The *simulation extractability* and zero-knowledge property of the NIZK.
- The *hardness of the Computational Diffie-Hellman (CDH)* problem in $\mathbb{G}$.
- The *extractability* of the threshold adaptor signature scheme.

We assume, for contradiction, that there exists an efficient adversary $\mathcal{A}$ that breaks the buyer fairness of Construction 3. To succeed, the adversary must post a valid signature signed by the buyers on tx while ensuring that at least one buyer cannot reconstruct the corresponding challenge witness.[3] In our reduction, the interaction with honest buyers is simulated by providing the adversary with buyer oracles. We define two events based on the type of forgery the adversary produces:

- **Fresh forgery** (fresh): The forgery is fresh if it corresponds to a random commitment $\widetilde{R}$ and the transaction tx that was never used in a signing session when interacting with the oracles.
- **Non-fresh forgery** ($\neg\mathsf{fresh}$): The forgery is non-fresh if the pair $(\widetilde{R}, \mathsf{tx})$ was previously used in a signing session.

The adversary wins if either of these cases holds, and it holds that

$$\Pr[\mathsf{BF}(\lambda) = 1] = \Pr[\mathsf{BF}(\lambda) = 1 \wedge \mathsf{fresh}] + \Pr[\mathsf{BF}(\lambda) = 1 \wedge \neg\mathsf{fresh}].$$

We analyze both cases independently and begin by analyzing the **non-fresh** case: In the non-fresh case, we distinguish two subcases. The first case occurs when at least one honest buyer never receives the plaintext pre-signature shares of all other buyers (Line 18 in Figure 12). A corrupted buyer can cause this for a single honest buyer alone, by sending a valid ciphertext to some honest buyers and an invalid one to others. We define the event where the adversary does not wait for the shares as EA. Looking ahead, we will reduce the case EA to the CDH problem. The second case occurs when $\mathcal{A}$ completes the interaction and receives the pre-signature shares before forging a signature. We will reduce this case to the simulation extractability of the NIZK. It holds that

$$\Pr[\mathsf{BF}(\lambda) = 1 \wedge \neg\mathsf{fresh}] = \Pr[\mathsf{BF}(\lambda) = 1 \wedge \neg\mathsf{fresh} \wedge \mathsf{EA}] + \Pr[\mathsf{BF}(\lambda) = 1 \wedge \neg\mathsf{fresh} \wedge \neg\mathsf{EA}].$$

[3] Recall from the blockchain model that locking the funds during the commit protocol produces a threshold signature on the locked transaction tx', while claiming the locked funds requires a signature on the distinct message tx. The lock signature published on the blockchain is therefore not a valid claim signature, and the adversary must produce a signature on tx, a message on which the honest buyers only ever engage in pre-signing sessions.

Since the event EA is a part of the event $\neg\mathsf{fresh}$, we omit $\neg\mathsf{fresh}$ for better readability. To show that non-fresh forgeries happen only with negligible probability, we analyze these two subcases separately and show that both probabilities are negligible. In the first case, EA, the adversary outputs a valid signature before receiving its commitments to the honest buyer's pre-signature shares. We now construct a reduction that uses this adversary to break the CDH assumption in $\mathbb{G}$. To achieve this, we perform a sequence of experiments, gradually transforming the adversary's attack into a CDH solver. We begin with the initial experiment $\mathcal{G}_0$, which is simply the BF security experiment conditioned on the event EA. By definition, the adversary's success probability in this game equals

$$\Pr[\mathsf{BF}(\lambda) = 1 \wedge \mathsf{EA}] = \Pr[\mathcal{G}_0(\lambda) = 1].$$

In the first hop, we move to experiment $\mathcal{G}_1$, where we modify the protocol by replacing the NIZK proofs used in the verifiable CWE encryption with simulated proofs from the simulator of the simulation-extractable NIZK scheme. Since the zero-knowledge property of NIZK ensures that this change does not affect the adversary's ability to distinguish the real and simulated proofs, this transition introduces only a negligible difference in the adversary's success probability:

$$\Pr[\mathcal{G}_0(\lambda) = 1] \leq \Pr[\mathcal{G}_1(\lambda) = 1] + \mathsf{negl}_1(\lambda).$$

Next, in experiment $\mathcal{G}_2$, we set up a reduction to the hardness of the CDH problem. The reduction reads two group elements $[\alpha]$ and $[\beta]$ and must compute the CDH solution $[\gamma] = [\alpha \cdot \beta]$. To do so, the reduction simulates the protocol execution while embedding the CDH challenge into the adversary's environment. The reduction begins by generating random signing key shares sk_i for all parties and forwards the signing keys of the corrupted parties to the adversary. It then guesses the signing session in which the adversary will attempt to break buyer fairness (the guess is efficient, as there are q_{sess} signing sessions which are polynomially bounded). Since we are in the non-fresh case, such a session exists. To embed the CDH challenge, the reduction programs the signature randomness $\widetilde{R}$ in the guessed session to be $[\alpha]$. This is achieved by setting the randomness of a single honest party to

$$R_i = \frac{[\alpha]}{E \cdot s \cdot R'},$$

where R' is the product of the random nonces of all other parties, extracted from the commitments of the first signing round by using the random oracle. Note that this causes $\widetilde{R} = [\alpha]$, so the CWE encryption key $\mathsf{ek} = \mathsf{KG}_{\mathsf{CWE}}(\mathsf{pk}, \mathsf{tx}, \widetilde{R})$ is determined by $[\alpha]$; it is through ek (not directly through R_i) that the CDH challenge is embedded. The reduction guesses the honest buyer that is left without the shares, which costs a factor n_{B}. To further embed the challenge, the reduction sets the ciphertext com_i for the guessed honest party to

$$com_i = ([\beta], x), \quad \text{where } x \leftarrow\!\!\$\ \mathbb{Z}_p.$$

For all other signing sessions, the reduction answers queries honestly. This is feasible as it knows the signing key shares of all honest parties. The final step in the reduction is extracting the CDH solution. The reduction guesses if and when the adversary queries the random oracle on the CDH solution. This introduces a loss factor of $1/2$ and $1/q_{\mathsf{ro}}$, where q_{ro} is the number of random oracle queries (also polynomially bounded). If the guess is correct and the adversary indeed queries the RO on a solution for CDH, the reduction can compute

$$[\gamma] = [\beta]^{-h \cdot \mathsf{sk}} \cdot x,$$

where x is the adversary's random oracle query and $h = \mathsf{H}(\mathsf{pk}, \widetilde{R}, m)$. If the guess is correct, the reduction obtains $[\gamma]$, thereby solving the CDH problem. The reduction terminates, and the adversary cannot distinguish the simulated com_i from an honest one, since it never receives the RO evaluation of its input. If the adversary does not query the random oracle on a CDH solution, it must still produce a non-fresh forgery. Since the reduction controls the signing key, it can extract $\alpha = \widetilde{r}$ from the forgery $(\widetilde{R}, \mathsf{sk} \cdot \mathsf{H}(\mathsf{pk}, \widetilde{R}, m) + \widetilde{r})$ and output $[\beta]^\alpha$, thereby still solving CDH. When guessing the reduction, obtaining the CDH solution incurs a polynomial loss in the reduction. If all guesses are correct, the reduction $\mathcal{B}$ solves the CDH instance. This imposes

$$\Pr[\mathcal{B}([\alpha],[\beta]) = [\alpha \cdot \beta]] = \Pr[\mathcal{G}_2(\lambda,[\alpha],[\beta]) = 1] = \frac{q_{\mathsf{ro}} + 1}{2 \cdot n_{\mathsf{B}} \cdot q_{\mathsf{sess}} \cdot q_{\mathsf{ro}}} \cdot \Pr[\mathcal{G}_1(\lambda) = 1].$$

Since we assume the probability of solving CDH to be negligible, and we have a negligible transition from experiment $\mathcal{G}_1$ to $\mathsf{BF} \wedge \mathsf{EA}$, this concludes the proof for the EA case. Specifically, we showed that the adversary's success probability in the case EA is negligible.

We now analyze the second case, where the adversary does not abort early ($\neg\mathsf{EA}$). In this scenario, the adversary successfully completes the interaction, receives the plaintext pre-signature shares, and subsequently produces a non-fresh forgery. This implies that the challenger has verified the NIZK corresponding to the VPKE of the pre-signature shares provided by the adversary, yet at least one honest buyer is unable to reconstruct a valid pre-signature after the adversary outputs the forgery. By the correctness of the CWE scheme and the bijectivity of the Schnorr signature scheme, the challenger is able to decrypt all well-formed commitments com_i that the adversary submitted before the challenger outputs the pre-signature forgery. Since the adversary violates buyer fairness, at least one honest buyer must be unable to reconstruct a valid pre-signature using the decrypted commitments. This violates the commitments' well-formedness and reduces the simulation extractability of the NIZK system. To construct this reduction, the challenger replaces all proofs by simulated ones and extracts the encrypted partial pre-signatures of the adversary. To do so, the challenger runs the setup algorithm for the FDE scheme, ensuring that it can answer all adversary queries. When the adversary produces a ciphertext that verifies correctly but fails to decrypt into a valid pre-signature share, the challenger outputs the pre-signature encryption along with the corresponding NIZK proof. This contradicts the simulation extractability of the NIZK, since the extracted witness cannot be a valid pre-signature share, which is required by the proof statement accompanying the verifiable encryption (c.f. Equation (2)). This reduction shows that the adversary's success probability in this case is negligible. Having established that the adversary's success probability is negligible in both cases—whether it aborts early (EA) or completes the interaction ($\neg\mathsf{EA}$)—we conclude that no efficient adversary can break buyer fairness by outputting a non-fresh signature in Construction 3.

In the case where the forgery is fresh (fresh), the adversary's forgery was not obtained by adapting any pre-signature that had been partially output by an honest buyer. This directly contradicts the extractability of the threshold adaptor signature scheme. This can be shown by a trivial reduction to the extractability: The reduction forwards corrupted keys and answers each oracle query using the provided oracles. In particular, the pre-signing interactions on tx are answered with the pSign oracle of the extractability experiment, and the lock-signing interaction of the commit protocol is answered with its Sign oracle on the lock message tx'. Since locking signs the locked transaction tx' and not tx, the signing queue $\mathcal{Q}_{\mathsf{Sign}}$ of the extractability experiment contains tx' but never tx, so the forgery on $m^* = \mathsf{tx}$ satisfies the freshness condition $m^* \notin \mathcal{Q}_{\mathsf{Sign}}$. Since the forgery is fresh and valid, its nonce differs from every nonce of a recorded pre-signature, and a recorded pre-signature

yields a witness only if the random oracle maps the fresh nonce to a value that depends on it, which happens with negligible probability. Thus, the reduction breaks extractability whenever the adversary breaks buyer fairness in the fresh case. Thus, the probability of winning with a fresh forgery is negligible.

Summarizing our results, we have divided the adversary's winning probability into three distinct cases: the event where the adversary aborts early and the forgery is non-fresh ($\mathsf{EA} \wedge \neg \mathsf{fresh}$), the event where the adversary completes the interaction but still produces a non-fresh forgery ($\neg \mathsf{EA} \wedge \neg \mathsf{fresh}$), and the event where the forgery is fresh (fresh). We have shown that in each of these cases, the adversary's success probability is negligible. Furthermore, we need to guess which case applies. This guessing introduces, at most, a polynomial loss. Therefore, we conclude that no efficient adversary can successfully violate the buyer fairness of Construction 3. □

Lemma 8 (Seller Fairness)**.** *Let $\mathcal{R}$ be the DLog hard relation in the group $\mathbb{G}$, and let TAS be the secure Schnorr threshold adaptor signature scheme defined in Construction 1, and let Π_{Enc} be a verifiable public key encryption scheme based on a simulation-extractable NIZK. Let $\mathbb{B}$ be a scriptless blockchain with timelocks that verifies existentially unforgeable Schnorr signatures. Then, Construction 3 with the settle protocol of Figure 15 achieves t_{S}-seller fairness if $n_{\mathsf{S}} \geq 2 \cdot t_{\mathsf{S}} - 1$ sellers participate in the protocol in the random oracle model.*

Proof of Lemma 8. To prove seller fairness, we must show two properties. First, the balance of an honest seller decreases with negligible probability. Second, the probability that the adversary outputs $y = f(w)$ while some honest seller is paid less than $v_{\mathsf{tx}}/n_{\mathsf{S}}$ exceeds the probability of the same event against a simulator by at most a negligible amount. We construct an efficient simulator Sim that emulates the behavior of all honest sellers in the SF experiment with $b = 1$. The simulator works as follows: It receives all public information and the secret keys of the honest sellers.

Eventually, $\mathcal{A}$ forwards a state ρ_i to each honest seller $i \in \mathsf{hon}_{\mathsf{S}}$. For each simulated seller i, the simulator runs $\mathsf{VerifyInputs}(\mathsf{pk}_{\mathsf{B}}, s, \mathsf{tx}, \rho_i)$ of Figure 15, and it broadcasts the digest $\mathsf{H}(\rho_i)$ whenever this subroutine does not return $\bot$. The subroutine checks that the state refers to tx, that the lock duration for the transaction has sufficient length, that the pre-signature pre-verifies w.r.t. the blinded statement $s \cdot E$, that the proof of each ciphertext verifies, and that the blinding commitments form a single sharing of degree $t_{\mathsf{S}} - 1$ (Line 5). All of this runs on public values only, so the simulator performs it without any secrets of the honest sellers. If the subroutine returns $\bot$, or if fewer than $2t_{\mathsf{S}} - 1$ sellers broadcast the digest of seller i, the simulator outputs $\bot$ as this seller's protocol output, exactly as an honest seller would. Otherwise, the simulator halts without sending any witness shares.

We now prove both properties in a sequence of experiments, starting from a challenger honestly simulating the experiment SF with $b = 0$ to the adversary. The first hop establishes that no honest seller loses balance. The remaining hops establish that whenever the honest sellers send their witness shares in the settle protocol, the payment follows; runs in which an honest seller is underpaid are therefore exactly the runs that our simulator reproduces.

Experiment 0. In the initial experiment, the challenger honestly simulates the experiment SF with $b = 0$ (Figure 2) instantiated with Construction 3 (Figures 12 and 15) to the adversary. This simulation is perfect, since the challenger runs the setup and holds the statement s, the witness w, and all secrets of the honest sellers.

Experiment 1. In Experiment 1, the challenger aborts if there exists an honest seller i w.r.t. whose public key a valid signature is posted to the blockchain during the experiment.

Claim 5. *Since Schnorr signatures are existentially unforgeable under chosen message attacks, the abort in Experiment 1 happens only with negligible probability.*

Proof. The challenger only forwards verification keys to the adversary and simulates the settle protocol, whose only output is an adapted pre-signature. No computation involving any seller's signing key is performed during this run. We build the following reduction that breaks the existential unforgeability of Schnorr if the abort occurs.

The reduction receives a Schnorr verification key vk. Instead of simulating the key generation honestly, the reduction guesses which seller will incur the loss and embeds vk as that seller's verification key. The reduction simulates the whole experiment as in Experiment 0. This is possible, since the reduction knows all keys, and never needs to sign w.r.t. any honest sellers key. When the adversary triggers the abort, it observes the blockchain. Since the abort occurs, there must exist a signature signed by S_i that spends from this seller's account. As the reduction never signed any transaction for S_i, this is a forgery against existential unforgeability. The reduction outputs the corresponding transaction and the signature. Our reduction is efficient, as it only runs an efficient adversary and forwards polynomial messages. □

Since decreasing the balance of an honest seller requires a valid signature under this seller's key on the blockchain, Experiment 1 already establishes the first property of seller fairness: the balance of an honest seller decreases only with negligible probability.

Experiment 2. In Experiment 2, the challenger decrypts the blinding value of each honest seller i by computing $e_i \leftarrow \Pi_{\mathsf{Enc}}.\mathsf{Dec}(\mathsf{sk}_{\mathsf{Enc},i}, ct_i)$ and aborts if $g^{e_i} \neq E_i$ for some honest seller i, where E_i is the i-th entry of the commitment vector $(E_0, E_1, \ldots, E_{n_\mathsf{S}})$ checked in the settle protocol (Line 5, Figure 15).

Claim 6. *If the NIZK underlying the verifiable public key encryption scheme Π_{Enc} is simulation extractable, the gap between Experiment 1 and Experiment 2 is negligible in λ.*

Proof. All buyers are corrupted in the seller fairness experiment, so no honest party produces proofs for relation Equation (3); every verifying statement-proof pair processed in the settle protocol was created by the adversary and is fresh in the sense of Definition 31. Assume the abort of Experiment 2 occurs for some honest seller i. The settle checks ensure that the pair $(\mathbb{x}_i, \pi_i)$ with $\mathbb{x}_i = (ct_i, M, \mathsf{pk}_{\mathsf{S}_i})$ verifies, where M is the commitment vector. Our reduction against the simulation extractability of the NIZK simulates Experiment 1 honestly, which is possible since it holds all keys, the witness, and the witness shares, and outputs the pair $(\mathbb{x}_i, \pi_i)$. By Definition 31, the extractor returns a witness e_i^* with $E_i = g^{e_i^*}$ and $ct_i = \Pi_{\mathsf{Enc}}(\mathsf{pk}_{\mathsf{S}_i}, e_i^*)$, except with negligible probability. By the correctness of Π_{Enc}, decrypting ct_i returns exactly this value e_i^*, so $g^{e_i} = E_i$, contradicting the abort. A union bound over the at most n_S honest sellers bounds the gap between the experiments by n_S times the extraction failure probability, which is negligible in λ. □

We record a consequence of Experiment 2 for the following hops. The consistency check of the settle protocol ensures $E_0 = E$ and $E_x = \prod_{\ell=1}^{t_\mathsf{S}} E_\ell^{\lambda_\ell(x)}$ for all $x \in \{0, 1, \ldots, n_\mathsf{S}\}$, so the discrete logarithms of $E_1, \ldots, E_{n_\mathsf{S}}$ lie on a single polynomial of degree $t_\mathsf{S} - 1$ whose constant term is the discrete logarithm of E. Since Experiment 2 guarantees $g^{e_i} = E_i$ for every honest seller i, the decrypted

blinding values of the honest sellers are points on this polynomial, and Lagrange interpolation of the values of any t_{S} honest sellers yields a value e' with $g^{e'} = E$.

Experiment 3. In Experiment 3, the challenger computes w' using all honest witness shares and the value e' as obtained in Experiment 2 (As in Line 10,Figure 15). It aborts if the resulting adapted signature (Line 11,Figure 15) is invalid.

Claim 7. *If the threshold adaptor signature scheme is pre-signature adaptable (Definition 17), the probability that Experiment 3 aborts while Experiment 2 would not is negligible in λ.*

Proof. Experiment 3 aborts additionally, if the adapted pre-signature (Line 11, Figure 15) is invalid w.r.t. pk_{B} and tx. We show that this violates the pre-signature adaptability requirement. Pre-signature adaptability shows that for each pre-signature $\widetilde{\sigma}$, each message m, each statement/witness pair, all public keys pk, if the pre-signature pre-verifies on the message-statement pair under pk, then the adapted signature verifies under pk on the same message.

To show that an additional abort violates pre-signature adaptability, we have to show that the computed witness w' is indeed a valid witness for $s \cdot E$. However, this is easy to see, since in Experiment 2, we have established, that $g^{e'} = E$. By the homomorphism of DLog statement-witness pairs, it follows that $w + e'$ is a valid witness for $s \cdot E$. The witness shares of the honest sellers lie on the degree-$(t_{\mathsf{S}} - 1)$ polynomial produced by SShare during setup, and by Experiment 2 the decrypted blinding values of the honest sellers lie on a single polynomial of degree $t_{\mathsf{S}} - 1$ as well. The blinded shares $w_i + e_i$ of the honest sellers are therefore points on the sum of these two polynomials, and since the challenger controls at least t_{S} honest sellers, interpolating their shares recovers w' identical to $w + e'$. Hence, the claim holds. □

Experiment 4. In Experiment 4, the challenger aborts, if the blockchain $\mathbb{B}$ would reject claiming the transaction tx under σ (Line 11, Figure 15).

Claim 8. *If $\mathbb{B}$ is a blockchain with timelocks, the additional abort in Experiment 4 never occurs.*

Proof. The challenger verifies, before the honest sellers broadcast their digests, that the transaction remains locked for at least three rounds. Since that check, the sellers have run the ready round and have exchanged their blinded shares, so at most two rounds have passed and the transaction is still locked. Therefore, when the first seller claims tx by sending $\mathsf{Claim}(\mathsf{tx}, \sigma)$ to $\mathbb{B}$ (Line 11, Figure 15), and the signature is valid under pk_{B} on tx (which it is since Experiment 3), and the transaction is a valid locked transaction (which it is since we checked in Experiment 3), then, the blockchain will always accept claiming the transaction. □

The ready round makes the honest sellers act consistently. If an honest seller i passes it, then at least $2t_{\mathsf{S}} - 1$ sellers broadcast $\mathsf{H}(\rho_i)$, and at most $t_{\mathsf{S}} - 1$ of them are corrupted, so at least t_{S} honest sellers broadcast the same digest. Each of these sellers passed $\mathsf{VerifyInputs}$ on a state whose digest equals $\mathsf{H}(\rho_i)$, and this state is ρ_i itself unless $\mathcal{A}$ found a collision of H. All sellers see the same broadcast values, so each seller counts the same $2t_{\mathsf{S}} - 1$ digests and sends its blinded witness share in the same round. Every honest seller therefore receives at least t_{S} blinded shares in Line 9 (Figure 15).

At this point of the proof, we have established that if the honest sellers send their witness shares in Line 9 (Figure 15), then the transaction tx will be posted on the blockchain, and since no other

payment authorized by honest sellers happened (ensured in Experiment 1), each honest seller is compensated with at least $v_{\mathsf{tx}}/n_{\mathsf{S}}$ coins.

In the real experiment, the payment thus follows whenever the honest sellers send their shares, up to the negligible abort probabilities collected in the previous hops. A run in which some honest seller j is paid less than $v_{\mathsf{tx}}/n_{\mathsf{S}}$ is therefore, except with negligible probability, a run in which the honest sellers never send their shares, either because the adversary never initiated the settle protocol or because some check failed. In such a run, the messages of the honest sellers consist of the broadcast digests and the protocol output $\perp$, both computed from public information exactly as our simulator computes them, so the adversary's view is identical to its view in the experiment with $b = 1$. The blockchain interaction of $\mathcal{A}$ is a function of this view, and the honest sellers authorize no payment in either experiment, so the balance of every seller is the same in both runs, and seller j is underpaid against our simulator as well. Consequently, the probability that $\mathcal{A}$ outputs an index $j \notin \mathsf{corr}_{\mathsf{S}}$ and the value $y = f(w)$ while seller j is underpaid is, up to the negligible losses above, at most the probability of the same event against our simulator. This is exactly the second property of seller fairness, and our lemma holds. This completes our proof. □

Lemma 9 (Seller Fairness with Deposits)**.** *Let $\mathcal{R}$ be the DLog hard relation in the group $\mathbb{G}$, and let* TAS *be the secure Schnorr threshold adaptor signature scheme defined in Construction 1 instantiated with a single signer. Let Π_{Enc} be a CPA secure verifiable public key encryption scheme based on a simulation-extractable NIZK. Let $\mathbb{B}$ be a scriptless blockchain with timelocks that verifies Schnorr signatures. Then, Construction 3 with the settle protocol of Figure 14 achieves t_{S}-seller fairness for every $t_{\mathsf{S}} \leq n_{\mathsf{S}}$ in the random oracle model.*

Proof of Lemma 9. As for Lemma 8, we must show two properties. First, the balance of an honest seller decreases with negligible probability, where we read the balance once all timelocks of the protocol have expired. Second, the probability that the adversary outputs $y = f(w)$ while some honest seller gains less than $v_{\mathsf{tx}}/n_{\mathsf{S}}$ exceeds the probability of the same event against a simulator by at most a negligible amount.

Let k^* denote the honest seller with the largest index, which exists since the adversary corrupts at most $t_{\mathsf{S}} - 1 < n_{\mathsf{S}}$ sellers. We construct an efficient simulator Sim that emulates the honest sellers in the SF experiment with $b = 1$. It receives all public information and the secret keys of the honest sellers. For each honest seller i it runs $\mathsf{VerifyInputs}(\mathsf{pk}_{\mathsf{B}}, s, \mathsf{tx}, \rho_i, 2n_{\mathsf{S}})$ on the state that $\mathcal{A}$ forwards, which reads public values only, and it outputs $\perp$ as this seller's protocol output whenever the subroutine does. For the re-sharing of seller i, the simulator samples $a_{i,k} \leftarrow\!\!\$\ \mathbb{Z}_p$ for every $k \neq k^*$, sets $g^{a_{i,k^*}} := (s_i \cdot E_i)^{\lambda_i} \cdot \prod_{k \neq k^*} g^{-a_{i,k}}$, broadcasts the resulting commitments, and sends $\Pi_{\mathsf{Enc}}.\mathsf{VEnc}(\mathsf{pk}_{\mathsf{S}_{k^*}}, 0)$ with a simulated proof to seller k^*. It locks the deposits and produces the pre-signatures with the signing keys it holds. For every honest seller $i \neq k^*$ it computes b_i by adding the summands it sampled to the summands it decrypts with $\mathsf{sk}_{\mathsf{S}_i}$ and emulates the claim of seller i, and it never posts a claim for seller k^*.

We now prove both properties in a sequence of experiments, starting from a challenger honestly simulating the experiment SF with $b = 0$ to the adversary. The first six hops establish that a seller whose deposit is spent always redeems the deposit above it, which gives the first property. The remaining two steps show that a run in which some honest seller is underpaid is a run that our simulator reproduces.

Experiment 0. In the initial experiment, the challenger honestly simulates the experiment SF with $b = 0$ (Figure 2) instantiated with Construction 3 (Figures 12 and 14) to the adversary. This simulation is perfect, since the challenger runs the setup and holds the statement s, the witness w, and all secrets of the honest sellers.

Experiment 1. In Experiment 1, the challenger aborts if the blockchain carries a valid signature under the signing key $\mathsf{pk}_{\Sigma,i}$ of an honest seller i on a message other than the deposit tx_i and the transaction tx'_i that locks it.

Claim 9. *If* TAS *is extractable (Definition 14), the abort in Experiment 1 happens with probability at most* $n_{\mathsf{S}} \cdot \mathsf{negl}_1(\lambda)$.

Proof. An honest seller i uses its signing key twice, once to lock its deposit tx_i and once to pre-sign tx_i w.r.t. the partial statement Y_{i-1}. Our reduction guesses the honest seller that triggers the abort and embeds the key of the extractability experiment, instantiated with a single signer, as $\mathsf{pk}_{\Sigma,i}$. It answers the lock with the Sign oracle and the pre-signature with the pSign oracle on the pair $(\mathsf{tx}_i, \mathsf{Y}_{i-1})$, and it simulates the rest of the experiment as in Experiment 0, which is possible since it holds all remaining keys, the witness, and the witness shares. Locking signs the locking transaction tx'_i and not tx_i, so the signing queue holds $\mathcal{Q}_{\mathsf{Sign}} = \{\mathsf{tx}'_i\}$ and the pre-signing queue holds $\mathcal{Q}_{\mathsf{pSign}}[\mathsf{tx}_i] = \{(\widetilde{\sigma}_i, \mathsf{Y}_{i-1})\}$. When the abort occurs, the blockchain carries a valid signature σ^* under $\mathsf{pk}_{\Sigma,i}$ on a message m^* outside these two queues, so $\mathcal{Q}_{\mathsf{pSign}}[m^*]$ is empty and the reduction wins Figure 4 with the pair (m^*, σ^*). A union bound over the at most n_{S} honest sellers gives the bound. □

Experiment 2. In Experiment 2, the challenger replaces the proofs that the honest sellers attach to their re-sharing ciphertexts with proofs of the zero-knowledge simulator of the NIZK.

Claim 10. *If the* NIZK *underlying* Π_{Enc} *is zero-knowledge, the gap between Experiment 1 and Experiment 2 is at most* ${n_{\mathsf{S}}}^2 \cdot \mathsf{negl}_2(\lambda)$.

Proof. Each of the at most n_{S} honest sellers produces at most n_{S} proofs. Replacing them one at a time gives ${n_{\mathsf{S}}}^2$ hybrids, and a distinguisher between two neighboring hybrids breaks the zero-knowledge property of the NIZK. □

Experiment 3. In Experiment 3, the challenger decrypts the blinding value of each honest seller i by computing $e_i \leftarrow \Pi_{\mathsf{Enc}}.\mathsf{Dec}(\mathsf{sk}_{\mathsf{S}_i}, ct_i)$ and aborts if $g^{e_i} \neq E_i$, where E_i is the i-th entry of the commitment vector checked by $\mathsf{VerifyInputs}$.

Claim 11. *If the* NIZK *underlying* Π_{Enc} *is simulation extractable, the gap between Experiment 2 and Experiment 3 is at most* $n_{\mathsf{S}} \cdot \mathsf{negl}_3(\lambda)$.

Proof. All buyers are corrupted in the seller fairness experiment, so every statement-proof pair that $\mathsf{VerifyInputs}$ accepts was created by the adversary. It also differs from the proofs that Experiment 2 simulates, which are proofs of honest sellers for their own re-sharing ciphertexts. Each such pair is therefore fresh in the sense of Definition 31, and the argument of Experiment 2 applies once the reduction takes its simulated proofs from Experiment 2. A union bound over the at most n_{S} honest sellers gives the bound. □

Experiment 4. In Experiment 4, the challenger decrypts every re-sharing ciphertext $ct_{j,i}$ that a corrupted seller j broadcasts, and it aborts if the decrypted summand $a_{j,i}$ does not match the commitment $g^{a_{j,i}}$ that S_j broadcast alongside it.

Claim 12. *If the* NIZK *underlying* Π_{Enc} *is simulation extractable, the gap between Experiment 3 and Experiment 4 is at most* ${n_\mathsf{S}}^2 \cdot \mathsf{negl}_4(\lambda)$.

Proof. The proof attached to $ct_{j,i}$ was created by the adversary and was never produced by the simulator of Experiment 2, so it is fresh in the sense of Definition 31. Its statement contains the ciphertext, the broadcast commitment, and the key $\mathsf{pk}_{\mathsf{S}_i}$, so the extractor returns a value a^* with $g^{a^*} = g^{a_{j,i}}$ and $ct_{j,i} = \Pi_{\mathsf{Enc}}(\mathsf{pk}_{\mathsf{S}_i}, a^*)$, except with negligible probability. By the correctness of Π_{Enc}, decrypting $ct_{j,i}$ returns exactly a^*, which contradicts the abort. A union bound over the at most ${n_\mathsf{S}}^2$ ciphertexts gives the bound. □

We record a consequence of Experiment 3 and Experiment 4 for the following hops. The consistency check of $\mathsf{VerifyInputs}$ places the discrete logarithms of $E_1, \ldots, E_{n_\mathsf{S}}$ on a single polynomial of degree $t_\mathsf{S} - 1$. Its constant term is the discrete logarithm of E, and the witness shares lie on the sharing polynomial of w. The re-sharing check of Figure 14 reads $\sum_k a_{j,k} = \lambda_j \cdot (w_j + e_j)$ in the exponent for every $j \in \mathcal{S}$, so summing over $\mathcal{S}$ and interpolating yields $\sum_k b_k = w + e' = w'$ with $g^{w'} = s \cdot E$. Every honest seller k therefore holds a value b_k with $g^{b_k} = B_k$, and the partial statements satisfy $\mathsf{Y}_k = g^{y_k}$ for $y_k = \sum_{\ell \le k} b_\ell$.

Experiment 5. In Experiment 5, the challenger aborts if a signature σ_i that spends the deposit tx_i of an honest seller i appears on the blockchain and the extracted value $\mathsf{Extract}(\mathsf{pk}_{\Sigma,i}, \widetilde{\sigma}_i, \sigma_i, \mathsf{Y}_{i-1})$ is not the discrete logarithm of Y_{i-1}.

Claim 13. *If* TAS *is extractable (Definition 14), the abort in Experiment 5 happens with probability at most* $n_\mathsf{S} \cdot \mathsf{negl}_5(\lambda)$.

Proof. Our reduction guesses the honest seller i and embeds the key of the extractability experiment as $\mathsf{pk}_{\Sigma,i}$, exactly as in Experiment 1. It obtains $\widetilde{\sigma}_i$ from the pSign oracle on the message tx_i and the statement Y_{i-1}, which records the pair $(\widetilde{\sigma}_i, \mathsf{Y}_{i-1})$ in $\mathcal{Q}_{\mathsf{pSign}}[\mathsf{tx}_i]$. This pre-signature pre-verifies, and it is the only one recorded for tx_i. The reduction never queries the Sign oracle on tx_i, since locking signs tx'_i instead, and it never queries NewS, so the partial statement satisfies $\mathsf{Y}_{i-1} \notin \mathcal{Q}_{\mathsf{stmt}}$. When the abort occurs, the reduction outputs $(\mathsf{tx}_i, \sigma_i)$, where σ_i verifies under $\mathsf{pk}_{\Sigma,i}$ and $\mathsf{Extract}(\mathsf{pk}_{\Sigma,i}, \widetilde{\sigma}_i, \sigma_i, \mathsf{Y}_{i-1})$ is no witness of Y_{i-1}, which are the winning conditions of Figure 4. A union bound over the at most n_S honest sellers gives the bound. □

Experiment 6. In Experiment 6, the challenger aborts if the blockchain rejects a claim that an honest seller posts in Line 14 of Figure 14 while the claimed transaction is still unspent.

Claim 14. *If* TAS *is pre-signature adaptable (Definition 17) and* $\mathbb{B}$ *is a blockchain with timelocks, the additional abort in Experiment 6 never occurs.*

Proof. An honest seller i posts a claim of tx_{i+1} only after Line 10 confirmed that $\widetilde{\sigma}_{i+1}$ pre-verifies under $\mathsf{pk}_{\Sigma,i+1}$ on tx_{i+1} w.r.t. Y_i. The consequence recorded above makes y_i a witness for Y_i, so pre-signature adaptability makes the adapted signature valid, and seller n_S claims tx under pk_B by the same argument. It remains to show that the claimed transaction is still locked. Seller 1

posts its claim in the round after the ladder is locked, which is the round in which tx_2 becomes spendable. Seller $i > 1$ posts its claim in the round after tx_i was spent, and Line 10 confirmed before it locked tx_i that tx_{i+1} stays locked for more than t_i further rounds, so tx_{i+1} is still open when the claim arrives. The re-sharing takes one round and the ladder is locked within n_S rounds. Seller n_S therefore posts its claim of tx no later than $2n_\mathsf{S}$ rounds after $\mathsf{VerifyInputs}$ confirmed that tx stays locked for that long. □

We now establish the first property of seller fairness. By Experiment 1, coins leave an honest seller only through its own deposit tx_i, and seller 1 locks no deposit at all. If tx_i is never spent, its coins return to seller i when the timelock expires. If tx_i is spent, then Experiment 5 gives seller i the value y_{i-1}, and Experiment 6 makes its claim succeed. Write $\mathsf{tx}_i^{\uparrow}$ for the transaction that pays seller i, which is tx_{i+1} for $i < n_\mathsf{S}$ and tx for seller n_S. Seller i then receives $i \cdot v_\mathsf{tx}/n_\mathsf{S}$ out of $\mathsf{tx}_i^{\uparrow}$ against the $(i-1) \cdot v_\mathsf{tx}/n_\mathsf{S}$ it had locked, so its balance grows by $v_\mathsf{tx}/n_\mathsf{S}$. The balance of an honest seller thus never decreases, up to the negligible abort probabilities collected above, and $\Pr[\mathsf{L}] \leq \mathsf{negl}(\lambda)$ follows. Reading the same computation in the other direction, an honest seller gains $v_\mathsf{tx}/n_\mathsf{S}$ whenever $\mathsf{tx}_i^{\uparrow}$ is spent, and it gains nothing otherwise, since tx_i unspent leaves its balance untouched. An honest seller is therefore underpaid precisely when $\mathsf{tx}_i^{\uparrow}$ is never spent.

Claim 15. *In Experiment 6, the probability that seller k^* posts a claim while some honest seller gains less than $v_\mathsf{tx}/n_\mathsf{S}$ is at most $n_\mathsf{S} \cdot (\mathsf{negl}_6(\lambda) + \mathsf{negl}_7(\lambda))$.*

Proof. Assume the event occurs and let m be an honest seller that gains less than $v_\mathsf{tx}/n_\mathsf{S}$. Seller k^* posts a claim, so $\mathsf{tx}_{k^*}^{\uparrow}$ is spent and seller k^* gains $v_\mathsf{tx}/n_\mathsf{S}$, which gives $m \neq k^*$ and hence $m < k^*$ because k^* is the largest honest index. For $k^* = 1$ no other honest seller exists and the claim holds vacuously, so we assume $k^* > 1$. Seller k^* reads y_{k^*-1} only off the spending of its own deposit, hence tx_{k^*} is spent as well. The transaction $\mathsf{tx}_m^{\uparrow}$ is never spent, so seller m never posts a claim.

Our reduction guesses the seller m among the honest sellers and modifies Experiment 6 in two steps. It first replaces the two re-sharing ciphertexts that seller m addresses to itself and to seller k^* by encryptions of 0 with simulated proofs, which Experiment 2 already supplies. Both recipients are honest, so the adversary never decrypts these ciphertexts, and the probability of the event changes by at most $\mathsf{negl}_6(\lambda)$ under the CPA security of Π_Enc. The reduction against Π_Enc does not hold the keys of these two recipients, and it recovers the summands addressed to them with the extractor of Definition 31 rather than by decrypting, which Experiment 2 makes possible. The two plaintexts are read nowhere else, since the only step of the challenger that uses them is a claim posted by seller m or by seller k^*.

In the modified experiment the reduction reads a discrete logarithm challenge $[\alpha]$ and sets $g^{a_{m,m}} := [\alpha]$. It samples $a_{m,k}$ for every $k \notin \{m, k^*\}$ and defines $g^{a_{m,k^*}} := (s_m \cdot E_m)^{\lambda_m} \cdot [\alpha]^{-1} \cdot \prod_{k \notin \{m,k^*\}} g^{-a_{m,k}}$. The re-sharing check of Figure 14 still passes, and the commitments of seller m keep the distribution they have in Experiment 6, since sampling all summands under the constraint of that check and deriving one of them from the others agree. Every other value the challenger produces is either a group element or a summand that the reduction sampled or decrypted, so the reduction runs all honest sellers except for the claims of m and k^*. It aborts if seller m posts a claim, which never happens in a run in which seller m gains less than $v_\mathsf{tx}/n_\mathsf{S}$, and it stops as soon as tx_{k^*} is spent, so it never reaches the claim of seller k^*.

At that point the reduction extracts y_{k^*-1} from σ_{k^*}, which is the discrete logarithm of $\mathsf{Y}_{k^*-1} =$

$\prod_{\ell<k^*} B_\ell$ by Experiment 5, and it outputs

$$y_{k^*-1} - \sum_{\ell<k^*,\ell\neq m} b_\ell - \sum_{i\neq m} a_{i,m} \ .$$

The index m lies below k^*, so b_m is one of the summands of $y_{k^*-1} = \sum_{\ell<k^*} b_\ell$, while the column k^* does not occur in that sum. The re-sharing ciphertexts are broadcast and the reduction holds every key $\mathsf{sk}_{\mathsf{S}_k}$, so Experiment 4 gives it the summand behind every commitment of a corrupted dealer, and it sampled the summands of the honest dealers itself. Every remaining value on the right is therefore known to it, and $b_m = a_{m,m} + \sum_{i\neq m} a_{i,m}$ makes the output equal to $a_{m,m}$, the discrete logarithm of $[\alpha]$. The reduction is efficient, since it runs the adversary once and performs a linear number of group operations. Guessing m costs a factor n_S and gives the claimed bound, where negl_7 is the advantage against the discrete logarithm problem. □

Experiment 7. In Experiment 7, the challenger produces the re-sharing of every honest seller i as our simulator does, sampling $a_{i,k}$ for every $k \neq k^*$, defining $g^{a_{i,k^*}}$ as the quotient above, and sending an encryption of 0 to seller k^*. It emulates the claim of every honest seller other than k^* with the summands it sampled and decrypted, and it posts no claim for seller k^*.

Claim 16. *The probability that $\mathcal{A}$ outputs an index $j \notin \mathsf{corr}_\mathsf{S}$ and the value $y = f(w)$ while seller j gains less than $v_{\mathsf{tx}}/n_\mathsf{S}$ differs between Experiment 6 and Experiment 7 by at most $n_\mathsf{S} \cdot \mathsf{negl}_8(\lambda)$ plus the bound of Claim 15.*

Proof. The commitments of the honest sellers are distributed identically in both experiments. Sampling all summands of a dealer under the constraint of the re-sharing check yields the same distribution as sampling all but one of them and deriving the last. The two experiments therefore differ in the plaintexts that the honest sellers send to seller k^* and in the claim of seller k^*. Seller k^* is honest, so the adversary never decrypts these ciphertexts, and a distinguisher between the two plaintext distributions breaks the CPA security of Π_{Enc}, which contributes $n_\mathsf{S} \cdot \mathsf{negl}_8(\lambda)$. The remaining difference shows only in runs in which seller k^* posts a claim, and Claim 15 bounds the probability that such a run underpays an honest seller. □

The challenger of Experiment 7 reads no witness share and no secret of a corrupted seller, and it behaves exactly as our simulator Sim, so Experiment 7 is the experiment SF with $b = 1$. Collecting the bounds of all hops gives a negligible function negl with $\Pr[\mathsf{SF}^0 = 1] \leq \Pr[\mathsf{SF}^1 = 1] + {n_\mathsf{S}}^2 \cdot \mathsf{negl}(\lambda)$, which is the second property of seller fairness. This completes our proof. □

Lemma 10 (Witness Privacy)**.** *Let $\mathcal{R}$ be the DLog hard relation. Let Π_{Enc} be a CPA secure verifiable public key encryption scheme based on a NIZK. Then, Construction 3 with either settle protocol of Figures 14 and 15 achieves witness privacy for any $t_\mathsf{S} \leq n_\mathsf{S}$ in the random oracle model.*

Proof of Lemma 10. To show witness privacy, we create the following simulator.

When the adversary starts the FDE interaction on input the statement s and some transaction, the seller simulates the Commit protocol. First, the simulator samples $\alpha \leftarrow \mathbb{Z}_p$, and computes $\{\alpha_i\}_{i\in\mathcal{S}} \leftarrow \mathsf{SShare}(\alpha, t_\mathsf{S}, n_\mathsf{S})$. Then, it computes $E = g^\alpha / s$. Then, for each $j \in \mathsf{corr}_\mathsf{S}$, it computes $e_j = \alpha_j - w_j$, where w_j is an adversarial witness share, and $E_j = g^{e_j}$. It also computes E_i from g^{α_i} and s_i (the partial statement of honest sellers).

Then, it computes $\widetilde{\sigma} \leftarrow \mathsf{pSign}(\mathsf{sk}_\mathsf{B}, \mathsf{tx}, g^\alpha)$. Then, the simulator encrypts the values e_j under $\mathsf{pk}_{\mathsf{S}_j}$ for all corrupted servers and encrypts 0 for all honest servers. Finally, the simulator uses the zero-knowledge simulator to make the ciphertext verifiable. The simulator outputs ρ consisting of the pre-signature $\widetilde{\sigma}$, the blinding commitment E, and the ciphertexts. Eventually, $\mathcal{A}$ outputs its view, which the simulator also outputs.

We show that the adversary's view when interacting with this simulator is indistinguishable from its view when running $\mathsf{WitPriv}$ with bit $b = 0$ by performing a sequence of game hops. Our starting experiment is the $\mathsf{WitPriv}$ experiment with $b = 0$, simulated using s and w. During our transitions, we show how this honest interaction is indistinguishable from our simulator. Then, witness privacy holds, as the adversary's probability in outputting the correct $y = f(w)$ is independent of the set bit b.

Experiment 0. In the initial experiment, the simulator receives as input s, w and has black-box access to $\mathcal{A}$. It runs the experiment $\mathsf{WitPriv}$ as described in Figure 3 instantiated with Construction 3 (Figures 12, 14 and 15) and $b = 0$ to the adversary. This simulation is perfect, since the simulator in this experiment receives the statement s and the witness w.

Experiment 1. In Experiment 1, the simulator replaces honestly computed proofs of well-formed encryption with zero-knowledge simulations.

Claim 17. *Let Π_{Enc} be a verifiable public key encryption scheme based on a NIZK, then the difference between Experiment 0 and Experiment 1 is negligible in λ.*

Proof. This claim follows directly from the definition of zero-knowledge, which ensures that simulated proofs are computationally indistinguishable from honestly computed proofs. □

Experiment 2. In Experiment 2, the simulator replaces encryptions of values e_i for honest $i \in \mathsf{hon}_\mathsf{S}$ with encryptions of 0. To run the settle protocol, it uses the locally stored values e_i instead of decrypting the respective ciphertexts.

Claim 18. *If Π_{Enc} is a CPA secure verifiable public key encryption scheme, then the difference between Experiment 1 and Experiment 2 is negligible.*

Proof. We build a reduction that uses any gap of the adversary's view in Experiment 2 or Experiment 1 to break the CPA security of Π_{Enc}. Assume via contradiction that there exists an efficient distinguisher $\mathcal{D}$ that can distinguish Experiment 2 from Experiment 1 with non-negligible advantage. We proceed via a standard hybrid argument, in which we either replace all encryptions of honest parties (this is one extreme hybrid), or only have honest encryptions. For the intermediate hybrid ℓ, we encrypt the first $\ell - 1$ ciphertexts honestly, and the remaining ones with encryptions of 0. By the standard hybrid argument, it remains to show that two neighboring hybrids are indistinguishable. Since there are at most n_S hybrids, the claim follows. For the intermediate hybrid, we build a reduction to the CPA security of Π_{Enc}. The reduction receives pk. We set $\mathsf{pk}_{\mathsf{S}_\ell} = \mathsf{pk}$ (note that to ease notation, we refer to the seller ℓ as the ℓ-st honest one). Then, instead of encrypting either e_ℓ, or 0, our reduction forwards $(e_\ell, 0)$ to it's challenger and receives ct_b. It uses ct_b, and the other ciphertexts, the pre-signature, and E as in *Experiment 1*, and the $\ell - 1$ hybrid to build ρ.

Eventually, the adversary outputs its view, which our reduction forwards to $\mathcal{D}$. The reduction outputs whatever $\mathcal{D}$ outputs. First, we have to discuss why our simulation of the first extreme hybrid perfectly simulates Experiment 1. This is the case, since in this experiment, the simulator encrypts the values e_i honestly, and uses them from storage instead of decrypting. Since each seller has consistent inputs (all buyers are honest) and the encryption scheme is correct, using blinding values from storage is equivalent to decrypting. If $\mathcal{D}$ has a non-negligible advantage, then this advantage carries over to our reduction-breaking CPA of Π_{Enc} in intermediate hybrid ℓ. Our reduction is efficient, since it just forwards polynomial-size messages, runs an efficient adversary, and compares two views. Our claim holds using the standard hybrid argument, since we have at most n_{S} intermediate hybrids, which is polynomial in λ. □

At this stage of the proof, the adversary only receives a pre-signature and can only deduce information from the ciphertexts that dishonest sellers can decrypt. We run our last hop in which we replace the way we form E, e, and w. In this last experiment, the simulator does not need w as input and is exactly the one we described from the beginning. Hence, witness privacy holds.

Experiment 3. In the last experiment, the simulator computes e_j, for each $j \in \mathsf{corr}_{\mathsf{S}}$ and E_j for each $i \in \mathsf{hon}_{\mathsf{S}}$ as described at the beginning of this proof.

Claim 19. *The adversary's view in Experiment 2 and Experiment 3 is computationally close.*

Proof. On input a statement s and the adversarial witness shares w_j, the simulator samples $\alpha \leftarrow \mathbb{Z}_p$, and computes $\{\alpha_i\}_{i\in\mathcal{S}} \leftarrow \mathsf{SShare}(\alpha, t_{\mathsf{S}}, n_{\mathsf{S}})$.

Then, for each $j \in \mathsf{corr}_{\mathsf{S}}$, it computes $e_j = \alpha_j - w_j$, and $E_j = g^{e_j}$. For honest sellers it computes $E_i = g^{\alpha_i}/s_i$. We note, that the value E and all values E_i are equally distributed in Experiment 3 and Experiment 2, since the adversary does not know k, so it cannot query $\mathsf{H}(k, \mathsf{tx}, s)$. Finally g^{α} is equally distributed as $E \cdot s$. In addition, instead of sending w'_i, each honest seller sends α_i (Line 9, Figure 15) which have the same distribution. The adversary sends $w'_j = \alpha_j - e_j$, which are drawn from the same distribution as expected. Finally, it holds that

$$\begin{aligned}
&\mathsf{SComb}(s \cdot E, \{\alpha_i\}_{i\in\mathsf{hon}_{\mathsf{S}}}, \{w'_j\}_{j\in\mathsf{corr}_{\mathsf{S}}}) = \\
&\mathsf{SComb}(g^{\alpha}, \{\alpha_i\}_{i\in\mathsf{hon}_{\mathsf{S}}}, \{w_j + (\alpha_j - w_j)\}_{j\in\mathsf{corr}_{\mathsf{S}}}) = \\
&\mathsf{SComb}(g^{\alpha}, \{\alpha_i\}_{i\in[n_{\mathsf{S}}]}) = \alpha.
\end{aligned}$$

Therefore, the recovered blinded witness $w' = \alpha$ is a valid witness for $s \cdot E$, and our proof holds. □

Claim 20. *The view of the adversary in the settle protocol with deposits (Figure 14) is distributed as in Experiment 3 as well.*

Proof. Every value that an honest seller i sends in Figure 14 is a function of its blinded share w'_i and of public data. The simulator of Experiment 3 supplies α_i in place of w'_i and computes from it the summands $a_{i,k}$ with $\sum_k a_{i,k} = \lambda_i \cdot \alpha_i$, their commitments, the deposits, and the pre-signatures. The re-sharing check forces the summands of a corrupted seller j to add up to $\lambda_j \cdot (w_j + e_j) = \lambda_j \cdot \alpha_j$, so the column commitments satisfy $\prod_k B_k = g^{\alpha}$. Redeeming tx_{k+1} publishes the partial sum y_k, and the claim of tx publishes $y_{n_{\mathsf{S}}} = \alpha$. These are the values that the sellers of Experiment 2 recover as well, since there the settle protocol reconstructs $w' = \alpha$. The deposits and the pre-signatures use the signing keys of the sellers, which the experiment generates independently of w, and α is uniform and independent of w given the corrupted witness shares. □

This completes our proof, since the adversary's view is identically distributed in *Experiment* 3 and the WitPriv experiment with $b = 1$. Since we only had negligible transitions, the view of the adversary in both versions of the WitPriv experiment is negligibly close, and the probability of outputting the correct value $y = f(w)$ is independent of the bit b (up to a negligible probability). □

8 Impossibility Results

We prove the impossibility of fairness in a distributed exchange protocol on a scriptless blockchain. Our proof follows a two-step approach. First, we show that seller fairness cannot be guaranteed under any threshold assumptions when relying solely on a scriptless blockchain. Second, we consider a scriptless blockchain with timelocks. There, fairness is possible only if either the sellers use deposits or an honest majority of sellers participates in the exchange. An FDE protocol for exchanging a coin for a witness using a scriptless blockchain operates in this setting:

- The witness w is secret-shared among the sellers as their initial state, so any t_{S} sellers can reconstruct it, and any $t_{\mathsf{S}} - 1$ sellers learn nothing about it.
- The buyers collectively hold a secret-shared signing key as their initial state.
- The blockchain is an append-only, ordered bulletin board for UTXO transactions.
- The total number of coins in our model are the coins initially owned by all buyers and all sellers. There are no third parties owning coins.

As in our construction, we consider the discrete logarithm relation, whose statements have a unique witness, so every valid witness for s equals w.

Adversarial Model. Throughout this section, we refine the rushing adversary of Section 3, so that it also controls the order in which parties are activated to perform computations. To constrain the adversary, we impose a round-robin-like ordering constraint, namely that no party can compute twice until all other parties have computed at least once in the meantime. While this constraint limits the adversary's power, it ultimately strengthens our impossibility result, since we show that no protocol can exist even under this weakened adversarial model.

Protocol Execution. When the adversary activates a party, the party receives its pending messages and the blockchain's current state. The party then computes in polynomial time and hands its outgoing messages and transaction requests to the adversary. Therefore, the adversary can read every request before it takes effect on the blockchain.

Impossibility. We now show that no FDE protocol that achieves t_{B}-buyer fairness can also achieve t_{S}-seller fairness in the presence of a scriptless blockchain.

Definition 22 (Joint State and Computability)**.** *Fix a protocol together with the fixed schedule of our adversaries, and fix a set X of parties. The* joint state *of X after an activation consists of the internal states and random tapes of the parties in X, together with all messages sent to X so far, and the current state of the blockchain. If X contains all corrupted parties, the joint state also includes every request to the blockchain that has been issued so far. We say that X* can compute *a*

value after an activation if there exists a PPT algorithm that outputs this value from the joint state of X after this activation with non-negligible probability.

We remark that at some point of the FDE protocol such an algorithm must exist, as the buyers learn a witness during the exchange.

Lemma 11 (Final Message)**.** *Let Π_{FDE} be a correct fair distributed exchange protocol of a coin for a witness w that achieves buyer fairness. Let the adversary corrupt a set X that consists of all buyers and of at most $t_{\mathsf{S}} - 1$ sellers, and let every corrupted party follow the protocol. Then there is a first activation after which X can compute w, and this activation belongs to a seller outside X. We call the messages and requests that this seller issues in this activation the* final message m^* *of X.*

Proof. We first show that some activation lets X compute w, so that a first such activation exists. Since all parties follow the protocol, correctness guarantees that the chain contains a signature σ_{B} on tx at the end of the execution, except with negligible probability. This signature posts or claims tx, so it irrevocably decreases the balance of the buyers. By buyer fairness, every buyer outputs w at the end, except with negligible probability, and this output lies in the joint state of X after the last activation. Before the first activation, the joint state of X consists of at most $t_{\mathsf{S}} - 1$ witness shares and the keys and random tapes of the parties in X. It further contains the public keys, the statement, the transaction, and the initial chain. These values reveal no information about w beyond the statement, because the sharing has threshold t_{S} and the keys are independent of w given the statement. Since the relation is hard, X cannot compute w before the first activation, and a first activation with the claimed property exists.

It remains to show that this activation does not belong to a party in X. When a party in X is activated, it computes its new state and its outgoing messages from its previous state, its random tape, the chain, and its incoming messages. All of these values lie in the joint state of X before the activation. A PPT algorithm on the joint state before the activation can therefore simulate the activation, so X can compute w after the activation only if it can do so before. The same holds for the blockchain update at the end of a round, since the new state is a function of the requests in the joint state of X. Consequently, the first activation after which X can compute w belongs to a seller outside X. □

Theorem 4 (Impossibility of Fairness for Scriptless Blockchains)**.** *No correct fair distributed exchange protocol Π_{FDE} that achieves t_{B}-buyer fairness can also achieve t_{S}-seller fairness in the presence of a scriptless blockchain.*

Proof of Theorem 4. We describe an adversary $\mathcal{A}$ that corrupts all buyers and no seller, lets the buyers follow the protocol, and breaks t_{S}-seller fairness. The adversary corrupts no seller, so the attack applies to every threshold t_{S}, including the standard adaptor-signature fair exchange case [AEE+21]. Let X denote the set of buyers, and let τ be the activation of the final message of X from Lemma 11, which belongs to a seller S^*. The adversary follows the protocol with all buyers until S^* has finished this activation. Then, $\mathcal{A}$ takes three steps. First, in a buyer activation later in this round, it issues one request per coin that the buyers can spend. Each request pays its coin to a fresh key pk^* that the adversary generates on its own. The buyers sign these requests with their secret keys, and the adversary sequences them first among all requests of the current round. The blockchain has no timelocks, so every coin of the buyers is spendable and these requests are valid. Every other request of this round that spends a coin of the buyers becomes invalid instead. Second, it runs the algorithm

of Definition 22 on the joint state of X and obtains w with non-negligible probability. Third, the adversary sends no further message and issues no further request. At the end of the execution it samples an index j uniformly among the n_{S} sellers and outputs w together with j.

We now show that the balance of some seller does not increase, which the sampled index hits with probability $1/n_{\mathsf{S}}$. Any coin leaves pk_{B} only through a signature of the buyers that takes effect and irrevocably decreases their balance. We claim that no such signature lies on the chain or in a request before activation τ, except with negligible probability. Otherwise a PPT algorithm on the joint state of X after $\tau - 1$, which contains the chain and all requests, would output w with non-negligible probability. Then X could compute w before τ, contradicting the choice of τ. Hence pk_{B} holds at least its initial value at activation τ, except with negligible probability. By the first step of the adversary, no coin of the buyers ever reaches a seller during the execution. Initially only sellers hold the coins outside pk_{B}, so their balances sum to at least the same value at the start. Consequently, the balance of at least one seller did not increase.

The sampled index j names such a seller with probability at least $1/n_{\mathsf{S}}$, independently of the rest of the execution. No seller is corrupted, and the payment value $v_{\mathsf{tx}}/n_{\mathsf{S}}$ is positive, so a seller whose balance did not increase satisfies the winning condition. The adversary therefore wins SF^0 with non-negligible probability for the identity as leakage function. In SF^1, the simulator receives only the statement and the keys of the honest sellers, which reveal nothing about w beyond the statement. Since the relation is hard, the adversary outputs w in SF^1 only with negligible probability, whichever simulator is used. The two probabilities therefore differ by a non-negligible amount, which contradicts t_{S}-seller fairness. □

Impossibility with Timelocks and Dishonest Majority. A timelock binds the coin of the buyers to tx until the lock expires, which removes the front-running attack behind Theorem 4. On such a blockchain, the settle protocol with deposits of Figure 14 achieves t_{S}-seller fairness. For $n_{\mathsf{S}} \geq 2 \cdot t_{\mathsf{S}} - 1$, the protocol of Figure 15 shows that the deposits can be dropped. We now show that no protocol without deposits or an honest majority of participating sellers achieves fairness.

To this end, we update our setting so that we have a scriptless blockchain with timelocks. In addition, in our adversarial model, we assume that the adversary can delay messages at most $t_{\mathcal{A}}$ rounds. We also assume that the time of a timelock is a multiple of the maximal computation time of a round. Therefore, we can assume that each party can run computations while coins are locked, and the adversary cannot circumvent the locking by delaying messages. In this setting, we show our impossibility. We start with an auxiliary lemma.

Lemma 12. *Let Π_{FDE} be a correct fair distributed exchange protocol that achieves buyer fairness, and let $n_{\mathsf{S}} < 2 \cdot t_{\mathsf{S}} - 1$. For a set T of $t_{\mathsf{S}} - 1$ sellers, let X_T consist of all buyers and the sellers in T, and let τ_T be the activation of the final message of X_T. Let T^* be a set of $t_{\mathsf{S}} - 1$ sellers with the smallest τ_{T^*}, and let the buyers and the sellers in T^* send no message and issue no request after τ_{T^*}. Then, except with negligible probability, no seller outside T^* issues a request at or after τ_{T^*} that takes effect and carries a signature of the buyers decreasing their balance irrevocably.*

Proof. Let $H = \mathcal{S} \setminus T^*$ denote the set of honest sellers, and let $\tau^* = \tau_{T^*}$ denote the activation of the final message of X_{T^*}. Every party follows the protocol up to the final message, so up to this point, the execution is honest. The activations τ_T of all sets T therefore refer to the same distribution, and the execution with T corrupted is well defined for every T. Assume, towards a contradiction, that with non-negligible probability some seller in H issues such a request with a signature σ at or

after τ^*. Since $n_\mathsf{S} < 2 \cdot t_\mathsf{S} - 1$, the set H has at most $t_\mathsf{S} - 1$ members, so we can choose a set T' of $t_\mathsf{S} - 1$ sellers with $H \subseteq T'$. We consider the execution in which the adversary corrupts $X_{T'}$ and show that $X_{T'}$ can compute w after activation $\tau^* - 1$. By Lemma 11, activation τ^* belongs to a seller h outside T^*, so $h \in H \subseteq T'$. The buyers and the sellers in T^* issue no request and send no message from τ^* on, so only the sellers in H still act. Their continuation follows from their states, their random tapes, the chain, the requests issued so far, and their incoming messages. All of these values lie in the joint state of $X_{T'}$ after activation $\tau^* - 1$, since $H \subseteq T'$ and this joint state contains all requests. A PPT algorithm on this joint state therefore reproduces the whole continuation and obtains the request that carries σ.

It remains to obtain w from this request. This is possible because of buyer fairness. If there would exist one buyer that cannot recover w from σ, an adversary against buyer fairness could reproduce this execution with all sellers corrupted and output a uniformly sampled buyer index. $\mathcal{A}$ wins with probability at least $1/n_\mathsf{B}$ times the probability that some buyer misses w, so this probability is negligible. The algorithm therefore reads w from the output of the buyers, and $X_{T'}$ can compute w before τ^*. This contradicts the choice of T^* as a set with the smallest final message. □

Theorem 5 (Impossibility of Fairness without Deposits if $n_\mathsf{S} < 2 \cdot t_\mathsf{S} - 1$)**.** *No correct fair distributed exchange protocol Π_{FDE} that achieves t_B-buyer fairness can also achieve t_S-seller fairness in the presence of a scriptless blockchain with timelocks if the sellers put up no deposits and $n_\mathsf{S} < 2 \cdot t_\mathsf{S} - 1$ holds.*

Proof of Theorem 5. Seller fairness must hold for every set of at most $t_\mathsf{S} - 1$ corrupted sellers, so it suffices to break it for the set T^* of Lemma 12, which applies since $n_\mathsf{S} < 2 \cdot t_\mathsf{S} - 1$. Let X^* consist of all buyers and the sellers in T^*, and let τ^* denote the activation of the final message of X^*. We describe an adversary $\mathcal{A}$ that corrupts X^* and lets every corrupted party follow the protocol until the seller activated at τ^* has finished. Then $\mathcal{A}$ runs the algorithm of Definition 22 on the joint state of X^* and obtains w with non-negligible probability. From this point on, $\mathcal{A}$ sends no message and issues no request, and at the end of the execution, it outputs w together with the index of an honest seller.

It remains to show that the balance of no honest seller increases, except with negligible probability. The requests issued up to the end of activation τ^* are distributed as in the execution without corrupted parties, in which every seller is honest. Since the sellers put up no deposits, none of these requests spends a coin of a seller, except with negligible probability. After τ^*, the corrupted parties issue no request, and no request of an honest seller spends a coin of an honest seller, again, since the sellers put up no deposits. Hence, no coin of a seller is ever spent, and the balance of an honest seller grows only through coins of the buyers.

A coin of the buyers reaches a seller only through a signature of the buyers that takes effect and irrevocably decreases their balance. Such a signature takes effect through a request, which is issued before τ^*, at or after τ^* by an honest seller, or after τ^* by a corrupted party. The corrupted parties issue no request after τ^*, and Lemma 12 rules out the honest sellers, except with negligible probability. A request issued before τ^* lies in the joint state of X^* after $\tau^* - 1$. A PPT algorithm on that joint state would then let X^* compute w before τ^*, contradicting the choice of τ^*. Hence, no coin of the buyers reaches a seller, and the balance of every honest seller stays unchanged, except with negligible probability.

Honest sellers exist because T^* omits at least one seller, and the payment value $v_{\mathsf{tx}}/n_\mathsf{S}$ is positive. The adversary therefore wins SF^0 with non-negligible probability for the identity as leakage function.

As in the proof of Theorem 4, every simulator in SF^1 lets it output w only with negligible probability. This gap contradicts t_{S}-seller fairness, which rules out a protocol without deposits below the bound. □

Acknowledgments

This work was partially supported by Deutsche Forschungsgemeinschaft as part of the Research and Training Group 2475 "Cybercrime and Forensic Computing" (grant number 393541319/GRK2475/1-2019), grant 442893093, by the state of Bavaria at the Nuremberg Campus of Technology (NCT) which is a research cooperation between the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) and the Technische Hochschule Nürnberg Georg Simon Ohm (THN), and by the Smart Networks and Services Joint Undertaking (SNS JU) under the European Union's Horizon Europe research and innovation program in the scope of the CONFIDENTIAL6G project under Grant Agreement 101096435. The contents of this publication are the sole responsibility of the authors and do not in any way reflect the views of the EU. This work is also partially supported by SBA Research (SBA-K1 NGC), which is a COMET Center within the COMET – Competence Centers for Excellent Technologies Programme and funded by BMIMI, BMWET, and the federal state of Vienna. The COMET Programme is managed by FFG.

Paul Gerhart's research was also supported by a Google PhD Fellowship in Privacy, Safety, and Security.

References

[1ML26] 1ML. Lightning Network statistics, 2026. Accessed: 2026-05-21.

[ADMM14a] Marcin Andrychowicz, Stefan Dziembowski, Daniel Malinowski, and Lukasz Mazurek. Fair two-party computations via Bitcoin deposits. In Rainer Böhme, Michael Brenner, Tyler Moore, and Matthew Smith, editors, *FC 2014 Workshops*, volume 8438 of *LNCS*, pages 105–121. Springer, Berlin, Heidelberg, March 2014.

[ADMM14b] Marcin Andrychowicz, Stefan Dziembowski, Daniel Malinowski, and Lukasz Mazurek. Secure multiparty computations on Bitcoin. In *2014 IEEE Symposium on Security and Privacy*, pages 443–458. IEEE Computer Society Press, May 2014.

[AEE+21] Lukas Aumayr, Oguzhan Ersoy, Andreas Erwig, Sebastian Faust, Kristina Hostáková, Matteo Maffei, Pedro Moreno-Sanchez, and Siavash Riahi. Generalized channels from limited blockchain scripts and adaptor signatures. In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part II*, volume 13091 of *LNCS*, pages 635–664. Springer, Cham, December 2021.

[AFOS26] Ojaswi Acharya, Georg Fuchsbauer, Adam O'Neill, and Marek Sefranek. Revisiting the security of Sparkle. Cryptology ePrint Archive, Paper 2026/431, 2026.

[AGR+16] Martin R. Albrecht, Lorenzo Grassi, Christian Rechberger, Arnab Roy, and Tyge Tiessen. MiMC: Efficient encryption and cryptographic hashing with minimal multiplicative complexity. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASI-*

ACRYPT 2016, Part I, volume 10031 of *LNCS*, pages 191–219. Springer, Berlin, Heidelberg, December 2016.

[AJR22] Henrik Axelsen, Johannes Jensen, and Omri Ross. When is a DAO decentralized? *Complex Systems Informatics and Modeling Quarterly*, (31):51–75, July 2022.

[Ark26] Ark Protocol Contributors. Ark Protocol. https://ark-protocol.org/, 2026. Accessed: 2026-05-20.

[BBB+18] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy*, pages 315–334. IEEE Computer Society Press, May 2018.

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 41–55. Springer, Berlin, Heidelberg, August 2004.

[BCC+25] Christian Badertscher, Matteo Campanelli, Michele Ciampi, Luigi Russo, and Luisa Siniscalchi. Universally composable SNARKs with transparent setup without programmable random oracle. In Yael Tauman Kalai and Seny F. Kamara, editors, *CRYPTO 2025, Part VII*, volume 16006 of *LNCS*, pages 225–258. Springer, Cham, August 2025.

[BDD20] Carsten Baum, Bernardo David, and Rafael Dowsley. Insured MPC: Efficient secure computation with financial penalties. In Joseph Bonneau and Nadia Heninger, editors, *FC 2020*, volume 12059 of *LNCS*, pages 404–420. Springer, Cham, February 2020.

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *20th ACM STOC*, pages 103–112. ACM Press, May 1988.

[BGKS25] Ruben Baecker, Paul Gerhart, Jonathan Katz, and Dominique Schröder. Fair exchange for decentralized autonomous organizations via threshold adaptor signatures. Cryptology ePrint Archive, Report 2025/388, 2025.

[BK14] Iddo Bentov and Ranjit Kumaresan. How to use Bitcoin to design fair protocols. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part II*, volume 8617 of *LNCS*, pages 421–439. Springer, Berlin, Heidelberg, August 2014.

[BLT+24] Renas Bacho, Julian Loss, Stefano Tessaro, Benedikt Wagner, and Chenzhi Zhu. Twinkle: Threshold signatures from DDH with full adaptive security. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part I*, volume 14651 of *LNCS*, pages 429–459. Springer, Cham, May 2024.

[BSW06] Dan Boneh, Emily Shen, and Brent Waters. Strongly unforgeable signatures based on computational Diffie-Hellman. In Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin, editors, *PKC 2006*, volume 3958 of *LNCS*, pages 229–240. Springer, Berlin, Heidelberg, April 2006.

[CD00] Jan Camenisch and Ivan Damgård. Verifiable encryption, group encryption, and their applications to separable group signatures and signature sharing schemes. In Tatsuaki Okamoto, editor, *ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 331–345. Springer, Berlin, Heidelberg, December 2000.

[CGJ+17] Arka Rai Choudhuri, Matthew Green, Abhishek Jain, Gabriel Kaptchuk, and Ian Miers. Fairness in an unfair world: Fair multiparty computation from public bulletin boards. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 719–728. ACM Press, October / November 2017.

[CKM23] Elizabeth C. Crites, Chelsea Komlo, and Mary Maller. Fully adaptive Schnorr threshold signatures. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part I*, volume 14081 of *LNCS*, pages 678–709. Springer, Cham, August 2023.

[CL03] Jan Camenisch and Anna Lysyanskaya. A signature scheme with efficient protocols. In Stelvio Cimato, Clemente Galdi, and Giuseppe Persiano, editors, *SCN 02*, volume 2576 of *LNCS*, pages 268–289. Springer, Berlin, Heidelberg, September 2003.

[DDO+01] Alfredo De Santis, Giovanni Di Crescenzo, Rafail Ostrovsky, Giuseppe Persiano, and Amit Sahai. Robust non-interactive zero knowledge. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 566–598. Springer, Berlin, Heidelberg, August 2001.

[DEF18] Stefan Dziembowski, Lisa Eckey, and Sebastian Faust. FairSwap: How to fairly exchange digital goods. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 967–984. ACM Press, October 2018.

[DHMW23] Nico Döttling, Lucjan Hanzlik, Bernardo Magri, and Stella Wohnig. McFly: Verifiable encryption to the future made practical. In Foteini Baldimtsi and Christian Cachin, editors, *FC 2023, Part I*, volume 13950 of *LNCS*, pages 252–269. Springer, Cham, May 2023.

[DOTT21] Ivan Damgård, Claudio Orlandi, Akira Takahashi, and Mehdi Tibouchi. Two-round n-out-of-n and multi-signatures and trapdoor commitment from lattices. In Juan Garay, editor, *PKC 2021, Part I*, volume 12710 of *LNCS*, pages 99–130. Springer, Cham, May 2021.

[DOY22] Wei Dai, Tatsuaki Okamoto, and Go Yamamoto. Stronger security and generic constructions for adaptor signatures. In Takanori Isobe and Santanu Sarkar, editors, *INDOCRYPT 2022*, volume 13774 of *LNCS*, pages 52–77. Springer, Cham, December 2022.

[EFH+21] Andreas Erwig, Sebastian Faust, Kristina Hostáková, Monosij Maitra, and Siavash Riahi. Two-party adaptor signatures from identification schemes. In Juan Garay, editor, *PKC 2021, Part I*, volume 12710 of *LNCS*, pages 451–480. Springer, Cham, May 2021.

[EFS20] Lisa Eckey, Sebastian Faust, and Benjamin Schlosser. OptiSwap: Fast optimistic fair exchange. In Hung-Min Sun, Shiuh-Pyng Shieh, Guofei Gu, and Giuseppe Ateniese, editors, *ASIACCS 20*, pages 543–557. ACM Press, October 2020.

[FKMV12] Sebastian Faust, Markulf Kohlweiss, Giorgia Azzurra Marson, and Daniele Venturi. On the non-malleability of the Fiat-Shamir transform. In Steven D. Galbraith and Mridul Nandi, editors, *INDOCRYPT 2012*, volume 7668 of *LNCS*, pages 60–79. Springer, Berlin, Heidelberg, December 2012.

[GCRS26] Paul Gerhart, Davide Li Calsi, Luigi Russo, and Dominique Schröder. Fully-adaptive two-round threshold schnorr signatures from DDH. In Joan Daemen and Emmanuel Thomé, editors, *EUROCRYPT 2026, Part II*, volume 16542 of *LNCS*, pages 251–283. Springer, Cham, May 2026.

[GGSW13] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 467–476. ACM Press, June 2013.

[GHK+25] Sanjam Garg, Mohammad Hajiabadi, Dimitris Kolonelos, Abhiram Kothapalli, and Guru-Vamsi Policharla. A framework for witness encryption from linearly verifiable SNARKs and applications. In Yael Tauman Kalai and Seny F. Kamara, editors, *CRYPTO 2025, Part III*, volume 16002 of *LNCS*, pages 504–539. Springer, Cham, August 2025.

[GJK15] Vipul Goyal, Aayush Jain, and Dakshita Khurana. Witness signatures and non-malleable multi-prover zero-knowledge proofs. Cryptology ePrint Archive, Report 2015/1095, 2015.

[GKR+21] Lorenzo Grassi, Dmitry Khovratovich, Christian Rechberger, Arnab Roy, and Markus Schofnegger. Poseidon: A new hash function for zero-knowledge proof systems. In Michael Bailey and Rachel Greenstadt, editors, *USENIX Security 2021*, pages 519–535. USENIX Association, August 2021.

[GKRS23] Sivanarayana Gaddam, Ranjit Kumaresan, Srinivasan Raghuraman, and Rohit Sinha. How to design fair protocols in the multi-blockchain setting. Cryptology ePrint Archive, Report 2023/762, 2023.

[GM17] Jens Groth and Mary Maller. Snarky signatures: Minimal signatures of knowledge from simulation-extractable SNARKs. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part II*, volume 10402 of *LNCS*, pages 581–612. Springer, Cham, August 2017.

[GMM+22] Noemi Glaeser, Matteo Maffei, Giulio Malavolta, Pedro Moreno-Sanchez, Erkan Tairi, and Sri Aravinda Krishnan Thyagarajan. Foundations of coin mixing services. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022*, pages 1259–1273. ACM Press, November 2022.

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In *17th ACM STOC*, pages 291–304. ACM Press, May 1985.

[GRS26] Paul Gerhart, Daniel Rausch, and Dominique Schröder. Universally composable adaptor signatures. In *Proceedings of the 2026 ACM SIGSAC Conference on Computer*

and Communications Security (CCS '26), The Hague, Netherlands, November 2026. Association for Computing Machinery.

[GSST24] Paul Gerhart, Dominique Schröder, Pratik Soni, and Sri Aravinda Krishnan Thyagarajan. Foundations of adaptor signatures. In Marc Joye and Gregor Leander, editors, *EUROCRYPT 2024, Part II*, volume 14652 of *LNCS*, pages 161–189. Springer, Cham, May 2024.

[GTT26] Paul Gerhart, Jay Taylor, and Sri Aravinda Krishnan Thyagarajan. Probabilistic atomic swaps for Bitcoin and friends, 2026.

[HLTW24] Lucjan Hanzlik, Julian Loss, Sri Aravinda Krishnan Thyagarajan, and Benedikt Wagner. Sweep-UC: Swapping coins privately. In *2024 IEEE Symposium on Security and Privacy*, pages 3822–3839. IEEE Computer Society Press, May 2024.

[JXGZ24] Yunfeng Ji, Yuting Xiao, Birou Gao, and Rui Zhang. Threshold/multi adaptor signature and their applications in blockchains. *Electronics*, 13(1), 2024.

[KGM19] Gabriel Kaptchuk, Matthew Green, and Ian Miers. Giving state to the stateless: Augmenting trustworthy computation with ledgers. In *NDSS 2019*. The Internet Society, February 2019.

[KL14a] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography*. Chapman and Hall, CRC Press, third edition, 2014.

[KL14b] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography, Second Edition*. Chapman & Hall/CRC, 2nd edition, 2014.

[KMB15] Ranjit Kumaresan, Tal Moran, and Iddo Bentov. How to use Bitcoin to play decentralized poker. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *ACM CCS 2015*, pages 195–206. ACM Press, October 2015.

[KOT24] Kaisei Kajita, Go Ohtake, and Tsuyoshi Takagi. Consecutive adaptor signature scheme: From two-party to N-party settings. In Joseph K. Liu, Liqun Chen, Shi-Feng Sun, and Xiaoning Liu, editors, *ProvSec 2023, Part I*, volume 14903 of *LNCS*, pages 23–42. Springer, Singapore, September 2024.

[KW03] Jonathan Katz and Nan Wang. Efficiency improvements for signature schemes with tight security reductions. In Sushil Jajodia, Vijayalakshmi Atluri, and Trent Jaeger, editors, *ACM CCS 2003*, pages 155–164. ACM Press, October 2003.

[KZZ16] Aggelos Kiayias, Hong-Sheng Zhou, and Vassilis Zikas. Fair and robust multi-party computation using a global transaction ledger. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 705–734. Springer, Berlin, Heidelberg, May 2016.

[Lin17] Yehuda Lindell. Fast secure two-party ECDSA signing. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part II*, volume 10402 of *LNCS*, pages 613–644. Springer, Cham, August 2017.

[MMS+19] Giulio Malavolta, Pedro Moreno-Sanchez, Clara Schneidewind, Aniket Kate, and Matteo Maffei. Anonymous multi-hop locks for blockchain scalability and interoperability. In *NDSS 2019*. The Internet Society, February 2019.

[Nak08] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. https://bitcoin.org/bitcoin.pdf, 2008. Accessed: 2026-09-03.

[Poe17] Andrew Poelstra. Scriptless scripts. https://download.wpsoftware.net/bitcoin/wizardry/mw-slides/2017-05-milan-meetup/slides.pdf, 2017. Presentation slides, Milan Bitcoin meetup, accessed 2026-09-09.

[RST26] Paul Rösler, Katharina Schambach, and Julian Thomas. Atomic swaps with relaxed assumptions. Cryptology ePrint Archive, Paper 2026/746, 2026.

[Sah99] Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *40th FOCS*, pages 543–553. IEEE Computer Society Press, October 1999.

[Sch91] Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, January 1991.

[TGT26] Jay Taylor, Paul Gerhart, and Sri Aravinda Krishnan Thyagarajan. How to make delegated payments on Bitcoin: A question for the AI agentic future. *Financial Cryptography and Data Security*, 2026.

[The11] The Litecoin Project. Litecoin. https://litecoin.org/, 2011. Accessed: 2026-09-03.

[TMM21] Erkan Tairi, Pedro Moreno-Sanchez, and Matteo Maffei. A^2L: Anonymous atomic locks for scalability in payment channel hubs. In *2021 IEEE Symposium on Security and Privacy*, pages 1834–1851. IEEE Computer Society Press, May 2021.

[TMM22] Sri Aravinda Krishnan Thyagarajan, Giulio Malavolta, and Pedro Moreno-Sanchez. Universal atomic swaps: Secure exchange of coins across all blockchains. In *2022 IEEE Symposium on Security and Privacy*, pages 1299–1316. IEEE Computer Society Press, May 2022.

[Tod14] Peter Todd. BIP 65: OP_CHECKLOCKTIMEVERIFY. https://bips.dev/65/, 2014. Bitcoin Improvement Proposal.

[TSZ+24] Ertem Nusret Tas, István András Seres, Yinuo Zhang, Márk Melczer, Mahimna Kelkar, Joseph Bonneau, and Valeria Nikolaenko. Atomic and fair data exchange via blockchain. In Bo Luo, Xiaojing Liao, Jun Xu, Engin Kirda, and David Lie, editors, *ACM CCS 2024*, pages 3227–3241. ACM Press, October 2024.

[Wat05] Brent R. Waters. Efficient identity-based encryption without random oracles. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 114–127. Springer, Berlin, Heidelberg, May 2005.

[WTs+18] Riad S. Wahby, Ioanna Tzialla, abhi shelat, Justin Thaler, and Michael Walfish. Doubly-efficient zkSNARKs without trusted setup. In *2018 IEEE Symposium on Security and Privacy*, pages 926–943. IEEE Computer Society Press, May 2018.

A Additional Preliminaries

Notation. We denote uniform sampling of x from the set X by $x \leftarrow\!\$\ X$, and let λ represent the security parameter. For a probabilistic polynomial-time (PPT) algorithm A that outputs a value y given input x and randomness r, we write $y \leftarrow \mathsf{A}(x)$. If A is a deterministic polynomial-time (DPT) algorithm, we use the notation $y := \mathsf{A}(x)$. A function $\mathsf{negl}(\lambda) : \mathbb{N} \to \mathbb{R}$ is called negligible in λ if, for every $k \in \mathbb{N}$, there exists $\lambda_0 \in \mathbb{N}$ such that for all $\lambda \geq \lambda_0$, it holds that $|\mathsf{negl}(\lambda)| \leq \lambda^{-k}$. We use the function **assert** to enforce conditions. Specifically, the function call **assert** condition terminates the execution in which it is invoked if condition is false. A vector of elements $x_1, \ldots, x_n$ is denoted by $\vec{x}$. We write $[\alpha]$ to denote g^α, for a given group $(\mathbb{G}, g, p)$. To denote the execution of a protocol among parties $\mathcal{P}_1, \ldots, \mathcal{P}_n$, we write $\langle \mathcal{P}_1(k_1, m), \ldots, \mathcal{P}_n(k_n, m) \rangle$. If the inputs of the parties are clear from the context, we omit them. When an adversary controls multiple parties, we abuse notation to collapse all adversarial parties into a single entity, denoted by $\langle \cdot, \mathcal{A} \rangle$.

A.1 Hard Relations

Definition 23 (Hard Relation [GSST24])**.** *Let $\mathcal{R}$ with statement/witness pairs (s, w) be a relation. We say that $\mathcal{R}$ is a hard relation if:*
Sampling $(s, w) \leftarrow \mathsf{genR}(\lambda)$. There exists a PPT sampling *algorithm $\mathsf{genR}(\lambda)$ that on input the security parameter λ outputs a pair $(s, w) \in \mathcal{R}$.*
Decidability $\mathcal{R}(s, w)$. The relation $\mathcal{R}(s, w)$ is poly-time decidable.
Hardness. An efficiently samplable and decidable relation $\mathcal{R}$ is hard*, if for all* PPT *adversaries $\mathcal{A}$, the following probability is negligible:*

$$\Pr[\mathcal{R}(s, w^*) = 1 \mid (s, w) \leftarrow \mathsf{genR}(\lambda); w^* \leftarrow \mathcal{A}(s)] \leq \mathsf{negl},$$

where the probability is taken over the random choice of genR *and $\mathcal{A}$.*

To construct CWE encryption schemes, we require hard relations where sampling is not performed uniformly at random but instead depends on auxiliary information. We model this using a sampling algorithm genR' that takes as input the security parameter λ and an auxiliary input x. In the absence of auxiliary information, genR' reduces to the standard sampling algorithm for hard relations. We impose the usual requirements of decidability and hardness, ensuring that these properties hold even when an adversary provides auxiliary information. More formally:

Definition 24 (Hard Relation with Auxiliary Information)**.** *Let $\mathcal{R}$ be a relation. We say that $\mathcal{R}$ is a hard relation with auxiliary input if:*
Sampling $(s, w) \leftarrow \mathsf{genR}'(\lambda, x)$. There exists a PPT *sampling algorithm $\mathsf{genR}'(\lambda, x)$ that on input the security parameter λ and auxiliary input $x \in \{0,1\}^\lambda$ outputs a pair $(s, w) \in \mathcal{R}$.*
Decidability $\mathcal{R}(s, w)$. The relation $\mathcal{R}(s, w)$ is poly-time decidable.
Hardness. $\mathcal{R}$ is hard*, if for all* PPT *adversaries $\mathcal{A}$, the following probability is negligible:*

$$\Pr[\mathcal{R}(s, w^*) = 1 \mid x^* \leftarrow \mathcal{A}(\lambda); (s, w) \leftarrow \mathsf{genR}(\lambda, x^*); w^* \leftarrow \mathcal{A}(s, x^*)] \leq \mathsf{negl}(\lambda),$$

where the probability is taken over the random choice of genR *and $\mathcal{A}$.*

A.2 Adaptor Signatures

An adaptor signature scheme [EFH+21, AEE+21], defined for a hard relation $\mathcal{R}$ and signature scheme Σ, consists of four algorithms. The pre-signing algorithm links a statement s from $\mathcal{R}$ and a message msg to a publicly verifiable pre-signature $\widetilde{\sigma}$. It ensures that anyone with the correct witness w can convert $\widetilde{\sigma}$ to a full signature σ using the Adapt method. In addition, possession of σ and $\widetilde{\sigma}$ allows efficient extraction of the witness w using the $\mathsf{Extract}$ algorithm.

Definition 25 (Adaptor Signature)**.** *An* adaptor signature scheme *w.r.t. a hard relation* $\mathcal{R}$ *and a signature scheme* $\Sigma = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vrfy})$ *consists of a tuple of four algorithms* $\mathsf{AS}_{\mathcal{R},\Sigma} = (\mathsf{pSign}, \mathsf{Adapt}, \mathsf{pVrfy}, \mathsf{Extract})$ *defined as:*
$\widetilde{\sigma} \leftarrow \mathsf{pSign}(\mathsf{sk}, m, s)$*. The* pre-signing *algorithm is a* PPT *algorithm that on input a secret key* sk*, message* $m \in \{0,1\}^{l_m}$ *and statement* $s \in L_{\mathcal{R}}$*, outputs a pre-signature* $\widetilde{\sigma}$*.*
$b \leftarrow \mathsf{pVrfy}(\mathsf{pk}, m, s, \widetilde{\sigma})$*. The* pre-verification *algorithm is a* DPT *algorithm that on input a public key* pk*, message* $m \in \{0,1\}^{l_m}$*, statement* $s \in L_{\mathcal{R}}$ *and pre-signature* $\widetilde{\sigma}$*, outputs a bit* b*.*
$\sigma \leftarrow \mathsf{Adapt}(\mathsf{pk}, \widetilde{\sigma}, w)$*. The* adapting *algorithm is a* PPT *algorithm that on input a pre-signature* $\widetilde{\sigma}$ *and witness* w *for the statement* $s \in L_{\mathcal{R}}$ *outputs an adapted signature* σ*.*
$w \leftarrow \mathsf{Extract}(\mathsf{pk}, \widetilde{\sigma}, \sigma, s)$*. The* extracting *algorithm is a* DPT *algorithm that on input a pre-signature* $\widetilde{\sigma}$*, signature* σ *and statement* $s \in L_{\mathcal{R}}$*, outputs a witness* w *such that* $(s, w) \in \mathcal{R}$*, or* $\perp$*.*

Definition 26 (Pre-signature correctness)**.** *An adaptor signature* $\mathsf{AS}_{R,\Sigma}$ *satisfies* pre-signature correctness*, if for all* $\lambda \in \mathbb{N}$ *and* $m \in \{0,1\}^{l_m}$*:*

$$\Pr\left[\begin{array}{c|l} \mathsf{pVrfy}(\mathsf{pk}, m, s, \widetilde{\sigma}) = 1 \wedge & (\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}(\lambda), \\ \mathsf{Vrfy}(\mathsf{pk}, m, \sigma) = 1 \wedge & (s, w) \leftarrow \mathsf{genR}(\lambda), \\ (s, w') \in \mathcal{R} & \widetilde{\sigma} \leftarrow \mathsf{pSign}(\mathsf{sk}, m, s) \\ & \sigma := \mathsf{Adapt}(\mathsf{pk}, \widetilde{\sigma}, w), \\ & w' := \mathsf{Extract}(\mathsf{pk}, \widetilde{\sigma}, \sigma, s) \end{array}\right] = 1.$$

For the security properties of adaptor signatures, we refer to [GSST24].

A.3 Threshold Signatures

Threshold signatures are a type of 't-out-of-n' signature: They require a key generation process that involves n participants, and after this, any subset of t members (where n and t are predetermined parameters set during the initial key generation) can collaboratively generate a valid signature under a collective public key. This public key represents the entire group of n participants. Furthermore, unforgeability guarantees that even a group of up to $t-1$ malicious signers cannot produce a forged signature.

Definition 27 (Threshold Signature Scheme [DOTT21, Lin17])**.** *A* threshold signature scheme $\mathsf{TS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *consists of algorithms and protocols as follows:*
$par \leftarrow \mathsf{Setup}(n, t)$*: The* setup algorithm Setup *takes as input the number* n *of signers and the signing threshold* t*, and outputs public parameters* par*. From now on,* par *is an implicit input to all subsequent algorithms.*
$(\mathsf{pk}, \{\mathsf{sk}_1, \ldots, \mathsf{sk}_n\}) \leftarrow \mathsf{KeyGen}(par)$*: The* key generation algorithm KeyGen *takes as input the public parameters* par *and outputs a combined public key* pk*, and a signing key share* sk_i *for each signer* $\mathcal{P}_i$*.*

$\sigma \leftarrow \langle \mathsf{Sign}(\mathsf{sk}_i, m)\rangle$: *The* signing protocol Sign *is an interactive algorithm of which an instance is run by each signer* $\mathcal{P}_1, \ldots, \mathcal{P}_n$ *concurrently. Concretely, signer,* $\mathcal{P}_i$ *runs* Sign_i*, which takes as input a secret key share* sk_i*, and a message* m*. At the end of the protocol,* $\mathcal{P}_i$ *obtains a signature* σ *as output.*
$b \leftarrow \mathsf{Verify}(\mathsf{pk}, m, \sigma)$: *The verification algorithm takes as input a public key* pk*, a message* m*, and a signature* σ*. It outputs a boolean* b*, where* $b = \textbf{true}$ *means that the signature is valid and* **false** *that it is invalid.*

Correctness. A threshold signature scheme is correct if for all $\lambda, t \leq nin\mathbb{N}$, all messages $m \in \{0,1\}^*$, for all keys $(\mathsf{pk}, \{\mathsf{sk}_1, \ldots, \mathsf{sk}_n\}) \leftarrow \mathsf{KeyGen}(par)$, each $\mathsf{hon} \subset \{1, \ldots, n\}$ with $|\mathsf{hon}| \geq t$, each output of the signing protocol also verifies, i.e.,

$$\mathsf{Vrfy}(\mathsf{pk}, m, \langle \{\mathsf{Sign}(\mathsf{sk}_i, m)\}_{i \in \mathsf{hon}} \rangle) = 1.$$

Construction 4 (Sparkle Schnorr Threshold Signature [CKM23])**.** *We construct the sparkle scheme using the algorithms* $\mathsf{Setup}, \mathsf{KGen}, \mathsf{Sign}^0, \mathsf{Sign}^1, \mathsf{Sign}^2, \mathsf{Combine}, \mathsf{Vrfy}$ *given in Figure 16.*

A.4 Non-interactive Arguments

We follow the notion of [BCC+25, GCRS26]. A *non-interactive argument system* (NARG) for relation $\mathcal{R}$ in the random oracle model, denoted by Π, consists of a tuple of algorithms $(\mathsf{Setup}, \mathcal{P}, \mathcal{V})$ having black-box access to a random oracle $\mathsf{H} : \{0,1\}^* \to \{0,1\}^\lambda$, with the following syntax:
$\mathsf{pp} \leftarrow \mathsf{Setup}(\lambda)$. The *setup* algorithm is a PPT algorithm. It reads a security parameter λ as input and outputs the public parameters pp.
$\pi \leftarrow \mathcal{P}^{\mathsf{H}}(\mathsf{pp}, \mathbb{x}, \mathbb{w})$. The *prover* algorithm is a PPT algorithm. It reads parameters pp, a statement $\mathbb{x}$ and a witness $\mathbb{w}$ as input and outputs a proof π if $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$.
$b \leftarrow \mathcal{V}^{\mathsf{H}}(\mathsf{pp}, \mathbb{x}, \pi)$. The *verifier* algorithm is a DPT algorithm. It reads parameters pp, a statement $\mathbb{x}$, and a proof π as input and it accepts $(b = 1)$ or rejects $(b = 0)$.

In this work, we consider NARGs with *transparent* setup, i.e., pp can be generated with a call to the random oracle. For this reason, we may omit pp in the description of the prover and verifier algorithms as it is implicit.

Definition 28 (Completeness)**.** *A NARG* Π *satisfies* completeness *if for every* $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$*, it holds that*

$$\Pr\left[b = 1 : \mathsf{pp} \leftarrow \mathsf{Setup}(\lambda); \pi \leftarrow \mathcal{P}^{\mathsf{H}}(\mathsf{pp}, \mathbb{x}, \mathbb{w}); b \leftarrow \mathcal{V}^{\mathsf{H}}(\mathsf{pp}, \mathbb{x}, \pi)\right] = 1.$$

Definition 29 (Soundness)**.** *A NARG* Π *satisfies* soundness *if for every* PPT *adversary* $\mathcal{A}$*, it holds that*

$$\Pr\left[b = 1 \wedge \mathbb{x} \notin \mathcal{L}_{\mathcal{R}} : \mathsf{pp} \leftarrow \mathsf{Setup}(\lambda); (\mathbb{x}, \pi) \leftarrow \mathcal{A}^{\mathsf{H}}(\mathsf{pp}); b \leftarrow \mathcal{V}^{\mathsf{H}}(\mathsf{pp}, \mathbb{x}, \pi)\right] = 0,$$

where $\mathcal{L}_{\mathcal{R}} := \{\mathbb{x} \mid \exists \mathbb{w} : (\mathbb{x}, \mathbb{w}) \in \mathcal{R}\}$ *is the set of true-statements.*

If soundness holds with respect to a computationally unbounded adversary, we say that the argument has *statistical* (or *information-theoretic*) soundness, and we call the argument a *proof* system.

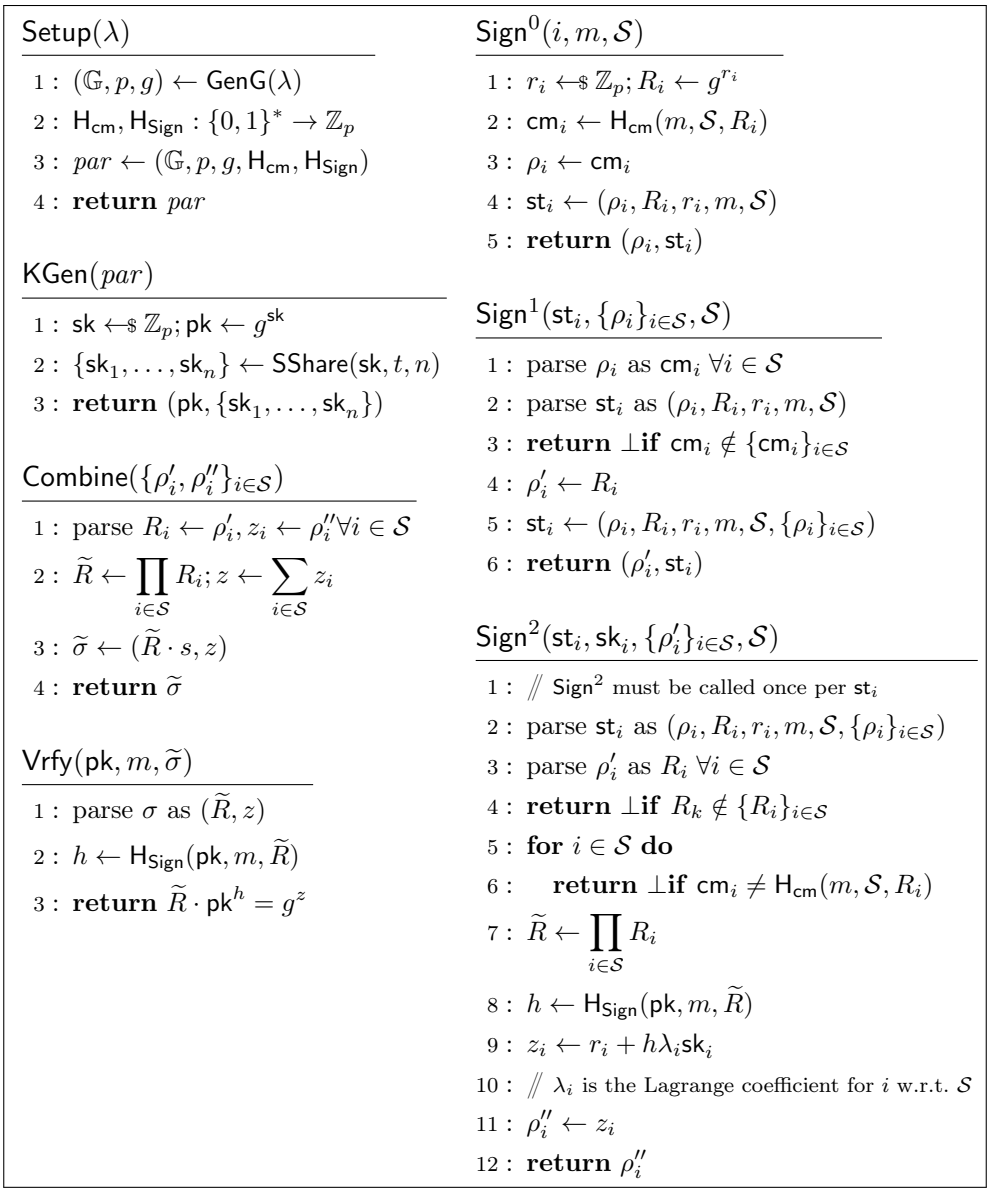


Figure 16: Algorithms of Sparkle [CKM23].

Zero-knowledge. Informally, an argument is zero-knowledge if a proof reveals no information about the witness [GMR85]. We formalize this property by following the syntax of [FKMV12]. A zero-knowledge simulator $\mathcal{S}$ is defined as a stateful algorithm, whose initial state $\mathsf{st} = \mathsf{pp}$, that operates in two modes. The first mode, $(y, \mathsf{st}') \leftarrow \mathcal{S}(1, \mathsf{st}, a)$ takes care of handling calls to the oracle H on input a query a. The second mode, $(\pi, \mathsf{st}') \leftarrow \mathcal{S}(2, \mathsf{st}, \mathbb{x})$ simulates a proof for the input statement $\mathbb{x}$. We define the following wrapper oracles, that are stateful and share their internal state:

- $\mathcal{S}_1(a)$ returns the first output of $\mathcal{S}(1, \mathsf{st}, a)$;
- $\mathcal{S}_2(\mathbb{x}, \mathbb{w})$ returns the first output of $\mathcal{S}(2, \mathsf{st}, \mathbb{x})$ if $(\mathbb{x}, \mathbb{w}) \in \mathcal{R}$ and $\perp$ otherwise;

- $\mathcal{S}'_2(\mathbb{x})$ returns the first output of $\mathcal{S}(2, \mathsf{st}, \mathbb{x})$.

Definition 30 (Zero-knowledge)**.** *A NARG Π is zero-knowledge if there exists a simulator $\mathcal{S}$, with wrapper oracles $\mathcal{S}_1, \mathcal{S}_2$, such that for all* PPT *adversaries $\mathcal{A}$ it holds that:*

$$\Pr\left[\begin{array}{l}\mathsf{pp} \leftarrow \mathsf{Setup}(\lambda) \\ \mathcal{A}^{\mathcal{P},\mathsf{H}}(\mathsf{pp}) = 1\end{array}\right] \approx \Pr\left[\begin{array}{l}\mathsf{pp} \leftarrow \mathsf{Setup}(\lambda) \\ \mathcal{A}^{\mathcal{S}_1,\mathcal{S}_2}(\mathsf{pp}) = 1\end{array}\right]$$

Notice that zero-knowledge is a security property that is only guaranteed for valid statements in the language, hence the above definition uses $\mathcal{S}_2$ as a proof simulation oracle. Also, a zero-knowledge NARG with statistical soundness is simply referred to as a non-interactive zero-knowledge proof (NIZK) [BFM88].

Simulation extractability. A strong security property of NARGs is *simulation extractability* [DDO+01, GM17]. Informally, from a prover producing a valid proof after seeing polynomially many proofs provided by a simulator (even for *false* statements), it is possible to extract a valid witness. This property can be formalized in many different ways, depending on the model and the constraints of the adversary and the extractor. We refer to [BCC+25] for a recent work that summarizes the main differences.

Definition 31 (Simulation extractability)**.** *A NARG Π is simulation-extractable with respect to a zero-knowledge simulator $\mathcal{S}$, with wrapper oracles $\mathcal{S}_1, \mathcal{S}'_2$, if for all* PPT *adversaries $\mathcal{A}$ there exists a* PPT *extractor $\mathcal{E}$ such that:*

$$\Pr\left[\begin{array}{l|l}\mathcal{V}^{\mathcal{S}_1}(\mathbb{x}, \pi) = 1 & \mathsf{pp} \leftarrow \mathsf{Setup}(\lambda) \\ \wedge\ (\mathbb{x}, \pi) \notin \mathcal{Q} & (\mathbb{x}, \pi) \leftarrow \mathcal{A}^{\mathcal{S}_1,\mathcal{S}'_2}(\mathsf{pp}) \\ \wedge\ (\mathbb{x}, \mathbb{w}) \notin \mathcal{R} & \mathbb{w} \leftarrow \mathcal{E}(\mathsf{st}, \mathbb{x}, \pi)\end{array}\right] \in \mathsf{negl}(\lambda)$$

where st *is the final internal state of $\mathcal{S}$, and $\mathcal{Q}$ is the set of statement-proof pairs obtained by $\mathcal{A}$ when interacting with $\mathcal{S}'_2$.*

The previous definition implies the weaker notion of *simulation soundness* [Sah99]. Informally, a NARG is simulation-sound if it is infeasible to generate a valid proof π for a false statement $\mathbb{x}$, even after seeing polynomially many simulated proofs. The implication holds because Definition 31 imposes a negligible probability of extraction failure, and extraction is by definition impossible for a false statement.

A.5 (Verifiable) Public Key Encryption Schemes

We recite the definition of a public key encryption scheme.

Definition 32 (Public-Key Encryption Scheme [KL14b])**.** *A* public-key encryption scheme $\Pi = (\mathsf{KGen}, \mathsf{Enc}, dec)$ *consists of three* PPT *algorithms*
$(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}(\lambda)$*: The* key generation *algorithm outputs a pair (sk, pk).*
$ct \leftarrow \mathsf{Enc}(\mathsf{pk}, m)$*: The* encryption *algorithm takes as input a public-key pk and a plaintext message m and outputs a ciphertext c.*
$m \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$*: The* decryption *algorithm takes as input a private-key* sk *and a ciphertext c and outputs some plaintext m.*

We say a public key encryption scheme Π verifiably [CD00] encrypts a message $m \in \mathbb{Z}_p$ with respect to a group $\mathbb{G}$ of order p (where p is prime), if the final ciphertext consists of the ciphertext c of the encryption scheme, a commitment M to m, and a proof of well formedness π showing that c is an encryption of the value m committed in M. We define the following interfaces.
$(ct, M, \pi) \leftarrow \mathsf{VPKE.Enc}(\mathsf{pk}, m)$: The encryption interface takes as input a public key pk and a message $m \in \mathbb{Z}_p$. It outputs a ciphertext ct, a commitment M, and a proof of well-formedness π.
$b \leftarrow \mathsf{VPKE.Vrfy}(\mathsf{pk}, (ct, M, \pi))$: The verification interface takes as input a public key pk and a ciphertext ct. It runs $\mathcal{V}(ct, M, \pi)$ and outputs 1, iff. this proof verification succeeds.

To simplify formalism, we omit the proof π and the commitment M, when they are clear from context. In this case, we simply write $ct \leftarrow \mathsf{VPKE.Enc}(\mathsf{pk}, m)$ and $b \leftarrow \mathsf{VPKE.Vrfy}(\mathsf{pk}, ct)$.

In our construction Construction 3, we use two types of verifiable public-key encryption schemes. For both schemes, we provide the corresponding commitments M and the respective proofs.

Verifiable encryption of partial pre-signatures. The first verifiable encryption encrypts partial pre-signatures, written as $com_i \leftarrow \mathsf{VEnc}_{\mathsf{CWE}}(\mathsf{ek}, \rho_i'')$ in Line 13, Figure 12. For our threshold adaptor signature scheme based on $\mathsf{Sparkle}$, the value ρ_i'' has the form

$$\rho_i'' = r_i + h \cdot \lambda_i \cdot \mathsf{sk}_i, \text{ where } h \leftarrow \mathsf{H}_{\mathsf{Sign}}(\mathsf{pk}, m, \widetilde{R}).$$

The values g^{r_i}, h, λ_i, and $\mathsf{pk}_i = g^{\mathsf{sk}_i}$ are publicly known. Therefore, we set

$$M = g^{r_i} \cdot \left(g^{\mathsf{sk}_i}\right)^{h \cdot \lambda_i} = g^{r_i} \cdot \mathsf{pk}_i^{h \cdot \lambda_i} = g^{\rho_i''}.$$

This is a perfectly binding commitment to ρ_i'' which can be verified publicly. To prove well-formedness, we use the language

$$\mathcal{R}_{\mathsf{CWE}} := \left\{ \begin{array}{ll} \mathbb{x} = (ct, M, \mathsf{ek}) & : \quad M = g^{\rho_i''} \\ \mathbb{w} = (\rho_i'', \mathsf{sk}_i) & \quad\; ct \leftarrow \mathsf{CWE.Enc}(\mathsf{ek}, \rho_i'') \end{array} \right\}.$$

Alternatively, we can define the equivalent relation w.r.t. $g^{r_i}, \mathsf{pk}_i, h, \lambda_i$ via

$$\mathcal{R}_{\mathsf{CWE}} := \left\{ \begin{array}{ll} \mathbb{x} = (ct, \mathsf{ek}, \mathsf{pk}_i, g^{r_i}, h, \lambda_i) & : \quad \mathsf{pk}_i = g^{\mathsf{sk}_i} \\ \mathbb{w} = (\rho_i'', r_i, \mathsf{sk}_i) & \quad\; \rho_i'' = r_i + h \cdot \lambda_i \cdot \mathsf{sk}_i \\ & \quad\; M = g^{\rho_i''} \\ & \quad\; ct \leftarrow \mathsf{CWE.Enc}(\mathsf{ek}, \rho_i'') \end{array} \right\}. \tag{2}$$

Verifiable encryption of blinding values. Other values we verifiably encrypt are the blinding values e_i, for which we also publish commitments E_i, as done in $\{\Pi_{\mathsf{Enc}}.\mathsf{VEnc}(\mathsf{pk}_{\mathsf{S}_j}, e_j)\}_{1 \le j \le n_{\mathsf{S}}}$, Line 20, Figure 12. A suitable relation would be

$$\mathcal{R}_E := \left\{ \begin{array}{ll} \mathbb{x} = (ct, M, \mathsf{pk}_{\mathsf{S}_j}) & : \quad M = (E, \vec{E}) \\ \mathbb{w} = (e_j) & \quad\; E_j = g^{e_j} \\ & \quad\; ct \leftarrow \Pi_{\mathsf{Enc}}(\mathsf{pk}_{\mathsf{S}_j}, e_j) \end{array} \right\}. \tag{3}$$

Suitable instantiations. For both these relations, we can add verifiable encryptions using Bulletproofs [BBB+18] and Hyrax [WTs+18]. This is practically efficient, even if we use hashed ElGamal as the underlying encryption scheme, provided we use an optimized hash function such as MiMC [AGR+16] or Poseidon [GKR+21]. Then, proving and verification time is in the milliseconds [GKR+21]. In addition, verifiable encryption comes a no additional setup, since Bulletproofs and Hyrax are trapdoorless, hence we do not need to use a CRS. Finally, Bulletproofs are proven secure from the DL assumption in the random oracle, hence adding these proofs does not change our underlying assumptions.

A.6 Secret Sharing

We assume a witness w is initially distributed among n_{S} sellers using a threshold secret-sharing scheme, so that any subset of at least t parties can reconstruct it, but fewer than that cannot. To capture this functionality, we use the following algorithms.
$\{w_i\}_{i\in[n]} \leftarrow \mathsf{SShare}(w, t, n)$: The sharing interface takes as input a witness w, a threshold t, and the number of parties n. It outputs a set of n witness shares, one for each party.
$w/\perp \leftarrow \mathsf{SComb}(s, \{w_i\})$: The reconstruction interface takes as input a statement s and a collection of witness shares. It outputs a witness w such that $\mathcal{R}(s, w) = 1$, or $\perp$ if reconstruction fails or the reconstructed value is not a valid witness.

To simplify the formalism, we assume that whenever at least t valid witness shares are provided to SComb, it always outputs a valid witness. In practice, this is enforced using a verifiable secret-sharing scheme: each share is accompanied by a commitment to the sharing polynomial (e.g., as in Feldman's VSS), allowing any party to publicly verify that a received share is consistent with the committed secret and the public statement s.

B A Framework for CWE

A detailed examination of the DDH-based construction described in Sec. 6 identifies certain properties that are essential for developing a highly efficient CWE. In this section, we will abstract these properties with the goal of constructing CPA-secure CWE for a large class of signatures. We refer to this class of signatures as "certificate-extractable" signatures and provide a construction that leverages the properties of certificate-extractable signatures into a CWE scheme. Our generalization is general enough to encompass constructions that are secure under the CDH assumption, which is of particular interest in the context of bilinear-pairing friendly groups. As for concrete instances, we show that our framework is compatible with the Schnorr [Sch91], the BBS+ [BBS04] the Camenisch-Lysyanskaya (CL) [CL03], (Waters+) [BSW06], and Katz-Wang [KW03] signature schemes.

In light of our warmup example, three principal properties inherent to the signature scheme are necessary for the construction of a secure CWE scheme. First, it is necessary to establish a methodology for the computation of an encryption key derived from the tuple (vk, m, and R) and a decryption key derived from the corresponding signature (σ). This property is referred to as *certifiability*. Secondly, it is necessary that there are not exponentially many signatures that can serve as a decryption key; instead, only the signature generated with the correct randomness can be used. Therefore, it is essential that we can extract the commitment, denoted as *commitment extractability*, from a signature. More formally, we define certificate-extractable signatures as follows.

Definition 33 (Certificate-Extractable Signature)**.** *Let* $\Sigma = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vrfy})$ *be a randomized signature scheme with randomness space* $\mathbb{Z}_p$*. We denote* R *as the commitment of a signature's randomness,* vk *as a verification key from the signature schemes keyspace, and* m *a message from the message space. We say* Σ *is* certificate extractable *if the following conditions hold.*
Certifiability: There exist deterministic functions PrivC *and* PubC*. The function* PrivC *extracts a certificate from a signature (i.e.,* $c \leftarrow \mathsf{PrivC}(\sigma)$*). The function* PubC *computes a commitment to a signatures certificate using public inputs (i.e.,* $C \leftarrow \mathsf{PubC}(\mathsf{vk}, m, R)$*).*
One-Wayness: It holds, that $\mathsf{PubC}(\mathsf{vk}, m, R) = [\mathsf{PrivC}(\sigma)]$*.*
Commitment Extractability: There exists an efficient algorithm ExtR *that on input a signature* σ *and a verification key* vk *outputs a commitment* R *to the signature's randomness.*

CWE for Certificate-Extractable Signatures. The use of certificate-extractable signatures necessitates the definition of a language for an underlying hard relation that ensures that not every valid signature on a message can be decrypted, but only those signatures that are based on the correct randomness. To construct such a language, we generalize the language from the Schnorr setting and utilize the commitment extractability of the certificate-extractable signature scheme. More formally, we define a statement of $\mathcal{L}_{\mathcal{R}}$ as a tuple of the verification key vk, message m, and commitment to a randomness R and a witness a valid signature w.r.t. (vk, m) which extracts the commitment R leading to the language

$$\mathcal{L}_{\mathcal{R}} := \{((\mathsf{vk}, m, R), \sigma) \text{ s.t. } \mathsf{Vrfy}(\mathsf{vk}, \sigma, m) = 1 \wedge \mathsf{ExtR}(\sigma) = R\}.$$

Using this language, we now define the CWE scheme for certificate-extractable signatures.

Construction 5 (CWE for CE Signatures)**.** *Let* $\mathbb{G}$ *be a group of prime order* p*. Let* H *be a function mapping into* $\mathbb{Z}_p$*, which is also the plaintext space. We define the* CWE *scheme* $\Pi_{\mathsf{CWE}} = (\mathsf{KG}_{\mathsf{CWE}}, \mathsf{Enc}_{\mathsf{CWE}}, \mathsf{Dec}_{\mathsf{CWE}})$ *for certificate extractable signatures w.r.t.* $(\mathbb{G}, p)$ *in Figure 17.*

$\mathsf{KG}_{\mathsf{CWE}}(\mathsf{vk}, m, R)$	$\mathsf{Dec}_{\mathsf{CWE}}(\sigma, c)$	$\mathsf{Enc}_{\mathsf{CWE}}(\mathsf{ek}, m)$
1 : $\mathsf{ek} \leftarrow \mathsf{PubC}(\mathsf{vk}, m, R)$	1 : parse c as (c_1, c_2)	1 : $a \leftarrow\!\!\$\ \mathbb{Z}_p$
2 : **return** ek	2 : $\mathsf{dk} \leftarrow \mathsf{PrivC}(\sigma)$	2 : $\mathsf{h} \leftarrow \mathsf{H}(\mathsf{ek}^a) + m$
	3 : **return** $c_2 - \mathsf{H}(c_1^{\mathsf{dk}})$	3 : **return** $([a], h)$

Figure 17: CWE encryption for certificate-extractable signatures.

CWE from DH. In this section, we show the CPA security of Construction 5 based on the decisional Diffie-Hellman (DDH) and the computational Diffie-Hellman (CDH) assumptions.

Looking ahead to the proof of the CPA security of Construction 5 w.r.t. the DDH/CDH assumptions, we first formalize a property that the signature scheme needs to achieve for our proof to simulate the challenge ciphertexts properly. We refer to this property as DH simulatability. On a high level, DH simulatability guarantees, that given a DH tuple $[\alpha], [\beta]$, our reduction can come up with a verification key vk, a random commitment R, such that for any message m, our reduction can simulate the value $[\delta] = C^{\alpha}$ for $C \leftarrow \mathsf{PubC}(\mathsf{vk}, m, R)$ using the factors $[\alpha], [\beta]$. Looking ahead, this will allow us to reduce the security of our reduction to both the CDH and the DDH assumption.

Definition 34 (DH Simulatability)**.** *We say a certificate-extractable signature scheme* Σ *is* DH-simulatable*, if for any* $\beta \leftarrow\$ \mathbb{Z}_p$*, there exists an algorithm* SimDDH*, that on input* $[\beta]$ *outputs a set* $(\mathsf{vk}, R, x_0, x_1)$*, such that* vk *is in the domain of* Σ*'s verification keys,* $R = \mathsf{OWF}(r)$ *is a commitment to a signatures randomness* r*,* $x_0, x_1 \in \mathbb{Z}_p$*, and* $x_0 \neq 0$*. In addition, for all* m *in the message space, the value* $C \leftarrow \mathsf{PubC}(\mathsf{vk}, m, R)$ *can be expressed via* $C = x_0[\beta] + x_1$*.*

If $C = [c]$ can be represented via $x_0[\beta]+x_1$, we can build the value $[\alpha{\cdot}c]$ via $[\alpha{\cdot}c] = [\alpha(x_0\beta{+}x1)] = x_0[\alpha\beta]+x_1[\alpha]$. Thus, if we are able to find $[\alpha \cdot c]$, we can compute $[\alpha\beta]$ via $[\alpha \cdot \beta] = \frac{[\alpha\cdot c]-x_1[\alpha]}{x_0}$ solving the CDH instance. In addition, given a DDH triple $([\alpha], [\beta], [\gamma])$, we can express $[\alpha{\cdot}c]$ by non-trivially using $[\gamma]$, allowing us to reduce the CPA security of CWE to DDH. Using DH simulatability, we now state the CPA security of Construction 5 based on both DDH and CDH for DH-simulatable certificate-extractable signature schemes. We prove Theorem 6 below. In addition, we defer the proof of CWE from DDH, and instantiations of our framework to Appendix B.

Theorem 6 (CPA Secure CWE from DDH)**.** *Let* Σ *be a certificate-extractable signature scheme with DH-simulatability w.r.t. a one-way function* OWF*. If the decisional Diffie-Hellman (DDH) assumption is hard in* $\mathbb{G}$ *w.r.t.* OWF*, and* H *is the identity, so that the plaintext space is* $\mathbb{G}$*, then the* CWE *construction in Figure 17 is a CPA secure* CWE *scheme.*

Theorem 7 (CPA Secure CWE from CDH)**.** *Let* Σ *be a certificate-extractable signature scheme with DH-simulatability w.r.t. a one-way function* OWF*. If the CDH assumption is hard in* $\mathbb{G}$ *w.r.t.* OWF*, and* H *modeled as a random oracle, then the* CWE *construction in Figure 17 is a CPA secure* CWE *scheme.*

Lemma 13. *The* CWE *scheme in Construction 5 is correct.*

Proof. To verify correctness of Construction 5, we have to check, that for all messages m, m', verification keys vk, and certificate pairs $C \leftarrow \mathsf{PubC}(\mathsf{vk}, R, m)$, $c \leftarrow \mathsf{PrivC}(\sigma)$, where σ is a valid signature w.r.t. (m', R, vk) it holds that $m = \mathsf{Dec}_{\mathsf{CWE}}(c, \mathsf{Enc}_{\mathsf{CWE}}(C, m))$. By the certificate extractability of the signature scheme, we have that $C = c$; hence, we have

$$c_2 - \mathsf{H}(c_1^c) = \mathsf{H}([c \cdot y]) + m - \mathsf{H}([c \cdot y]) = m.$$

Henceforth, Construction 5 is correct. □

Proof of Theorem 6. This proof primarily follows the proof of Theorem 2, with the main difference being the need to incorporate DH simulatability and certificate-extractability as abstract concepts, both provided by Schnorr signatures. Assume, for contradiction, that there exists an adversary $\mathcal{A}$ that can break the CPA security of Π_{CWE} with a non-negligible probability. We will use this adversary to build a reduction from the CPA security of Π_{CWE} to the hardness of DDH. The reduction $\mathcal{R}$ receives the values $([\alpha], [\beta], [\gamma])$, where γ is either a random value or $\alpha \cdot \beta$. The reduction runs $\mathsf{SimDDH}([\beta])$ to obtain the values $(\mathsf{vk}, R, x_0, x_1)$, such that $C = x_0[\beta] + x_1$. The reduction then forwards the key vk to $\mathcal{A}$, which eventually outputs three messages (m_0, m_1, m). The reduction computes the challenge ciphertext for the challenge message m_b as follows:

$$c_b \leftarrow ([\alpha], \mathsf{H}(x_0 \cdot [\gamma] + x_1[\alpha]) \cdot m_b).$$

Note that in Theorem 6, the hash function is the identity, so this computation matches the construction. Furthermore, by the DH simulatability, the value $x_0 \cdot [\gamma] + x_1[\alpha]$ equals $[\alpha \cdot c]$. The reduction forwards c_b to $\mathcal{A}$, which eventually outputs a bit b', and $\mathcal{R}$ returns $b' == b$.

Claim 21. *If $\gamma \neq \alpha \cdot \beta$, then $\mathcal{A}$ wins with probability at most $1/2$.*

Proof. If $\gamma \neq \alpha \cdot \beta$, then γ is a random element in $\mathbb{Z}_p$. Thus, the challenge ciphertext c_b is also a random element in $\mathbb{Z}_p$, since $x_0 \neq 0$ and is independent of the challenge message m_b. So, the claim follows. □

Claim 22. *If $\gamma = \alpha \cdot \beta$, then $\mathcal{A}$ wins with non-negligible advantage ε.*

Proof. If $\gamma = \alpha \cdot \beta$, then the ciphertext is exactly constructed as the adversary expects it to be. As we assume by contradiction that $\mathcal{A}$ breaks the CPA security of the encryption scheme with non-negligible probability, this probability carries over if $\gamma = \alpha \cdot \beta$. □

With the standard argument for the CPA security of the original Elgamal encryption, it holds that

$$\begin{aligned} \left|\Pr\left[\mathsf{DDH}^0_{\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathsf{DDH}^1_{\mathcal{A}}(\lambda) = 1\right]\right| \geq \\ \left|\Pr\left[\mathsf{CPA}^1_{\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathsf{CPA}^0_{\mathcal{A}}(\lambda) = 1\right]\right| \geq \\ \left|\frac{1}{2} + \varepsilon - \frac{1}{2}\right| \geq \varepsilon. \end{aligned}$$

This contradicts the hardness of DDH; thus, no efficient adversary against the CPA security of Construction 5 can exist. □

Proof of Theorem 7. To prove Theorem 7, we assume via contradiction, that there exists an adversary against the CPA security of Construction 5 and build a reduction that leverages the capabilities of this adversary to compute a solution for a CDH instance. As input, the reduction gets a CDH-pair $[\alpha], [\beta]$. Following the proof of Theorem 6, the reduction uses the SimDDH algorithm to obtain values $(\mathsf{vk}, R, x_0, x_1)$, such that $C = x_0[\beta] + x_1$. The reduction then forwards the key vk to $\mathcal{A}$, which eventually outputs three messages (m_0, m_1, m). The reduction computes the challenge ciphertext for the challenge message m_b as follows by sampling a random value $h \leftarrow\!\!\$\ \mathbb{G}$ and outputting

$$c_b \leftarrow ([\alpha], h \cdot m_b).$$

Since the value h is randomly sampled and the RO models a truly random function, the adversary can only distinguish the message encrypted in the challenge ciphertext with a higher probability than guessing if it queries $[\alpha \cdot c] = [\alpha(x_0\beta + x1)] = x_0[\alpha\beta] + x_1[\alpha]$ at least once to the random oracle. To catch the respective query, the reduction guesses the occurrence of this query amongst all random oracle calls and aborts the simulation by outputting the element

$$\frac{h_1 - x_1[\alpha]}{x_0},$$

where h_1 is the respective query. If the guess is correct, the simulator successfully returns $[\alpha \cdot \beta]$, since the DH simulatability holds. Hence, the simulator outputs a valid solution for the CDH problem. The probability that the guess is correct is polynomially bounded since the adversary can only make polynomially many RO queries. Thus, there cannot be an efficient adversary against the CPA security of Construction 5. □

Instances. We now demonstrate that the BBS [BBS04] signature scheme complies with Theorem 7. Thus, we have CPA-secure CWE encryptions for BBS+. We defer the examples of the other signature schemes to Section B.1. We show in this section that the BBS+ signature is a certificate-extractable signature scheme. Since the BBS+ is defined w.r.t. a group capable of bilinear pairings, we show that the BBS+ signature satisfies the requirements of Theorem 7, i.e., BBS+ is DH simulatable and lives in a CDH-hard group.

Lemma 14 (BBS+ is CE)**.** *The BBS+ signature scheme is a certificate-extractable DH simulatable signature scheme.*

Proof. A BBS+ signature has the form (A, e, r), where $r \leftarrow\!\!\$\ \mathbb{Z}_p$. We define R as g_1^r where g_1 is part of the verification key. To show certifiability, we define $\mathsf{PubC}(\mathsf{vk}, m, R) := R$ and $\mathrm{PRIV}(A, e, r) := r$. Having available a signature (A, e, r) and a verification key vk, we define $\mathsf{ExtR}(\mathsf{vk}, (A, e, r)) := g_1^r$, which is well-defined, since g_1 is part of the verification key. Moreover, it holds that $R = [r]$, so one-wayness holds. So, BBS+ is a certificate-extractable signature scheme. To show DH simulatability, we define SimDDH via $\mathsf{SimDDH}([\beta]) := (\mathsf{vk} \leftarrow\!\!\$\ \mathbb{G}, [\beta], 1, 0)$.This way, for all m in the message space, $[\alpha \cdot c]$ can truly be expressed via $[\alpha \cdot c] = [\alpha \cdot r] = [\alpha \cdot \beta]$. □

Since the security of the BBS+ signature scheme is based on q-SDH [BBS04], whose hardness implies the hardness of CDH, all the assumptions of Theorem 7 are satisfied, and Construction 5 is a CPA secure CWE scheme for BBS+.

B.1 Certificate Extractable Signature Schemes

In this section, we show that Schnorr [Sch91], the Camenisch-Lysyanskaya (CL) [CL03], (Waters+) [BSW06], and Katz-Wang [KW03] signature schemes are certificate extractable signature schemes which are DH simulatable.

CL Signatures. CL signatures are defined w.r.t. the quadratic residues modulo n QR_n. Our framework is stated for a group $\mathbb{G}$ of prime order, so the instantiation below applies to a prime-order subgroup of QR_n in which the underlying assumption is believed to hold. A public key of a CL signature has the form $\mathsf{vk} = a, b, c, n$, and a signature of the CL signature scheme consists of the elements (v, e, r), where $v^e \equiv a^m \cdot b^r \cdot c \mod n$.

Lemma 15. *The CL signature scheme [CL03] is a certificate-extractable DH simulatable signature scheme.*

Proof. We define $R := b^r$, which is extractable from a signature (v, e, r). We define $\mathsf{PubC}(\mathsf{vk}, m, R) := R$ and $\mathsf{PrivC}(v, e, r) := r$, such that it holds $C = R = [r] = [c]$. To show DH-simulatability, we set $\mathsf{SimDDH}([\beta]) = (\mathsf{vk}, [\beta], 1, 0)$, where vk is a normally crafted encryption key, and b has the same basis as β. □

Waters+ Signatures. The Waters+ signature arises by the waters signature if one applies the BSW transformation [BSW06] to the waters signature [Wat05]. We defer a formal description of the BSW compiler to [BSW06], since we only rely on the fact, that BSW compiled signatures have the randomness in plain, and use a commitment to this randomness in the signature.

Lemma 16. *The Waters+ signature scheme [BSW06] is a certificate-extractable DH simulatable signature scheme.*

Proof. The structure of CL signatures and BSW compiled signatures is similar, i.e., the signatures consist of three elements (σ_1, σ_2, r), where a random commitment $[r]$ is used to compute σ_1. Hence, we follow the proof of CL and set $C = R = [r]$ and $c = r$, which shows all needed properties. □

Katz-Wang Signatures. A signature of the Katz-Wang signature scheme [KW03] consists of the three elements (A, B, z), for some $r \leftarrow\!\!\$\ \mathbb{Z}_p$, and random group elements $g, h \in \mathbb{G} \subset \mathsf{vk}$, $A = g^r, B = h^r$, and $z = \mathsf{H}(\mathsf{vk}, A, B, m) \cdot \mathsf{sk} + r$. The security of the Katz-Wang signature scheme is based on the hardness of DDH in $\mathbb{G}$.

Lemma 17. *The Katz-Wang signature scheme [KW03] is a certificate-extractable DH simulatable signature scheme.*

Proof. To show the certificate-extractability, we define $R := (A, B)$ and $\mathsf{PrivC}(A, B, z) := z$. The public certificate is $\mathsf{PubC}(\mathsf{vk}, m, R) := A \cdot \mathsf{vk}^{\mathsf{H}(\mathsf{vk}, A, B, m)}$. It holds, that $C = [z] = [c]$. Moreover, $R = (A, B)$ can be recovered from a signature σ. The Katz-Wang signature scheme is DH simulatable when defining $\mathsf{SimDDH}([\beta]) = ([\beta], [r], h, r)$ for a random $r \leftarrow\!\!\$\ \mathbb{Z}_p$, and a random h. For DDH simulatability to hold, we have to program the RO such that $\mathsf{H}(\mathsf{vk}, A, B, m) = h$. □

Schnorr Signatures. We have already introduced Schnorr signatures since a combined Sparkle signature verifies as a Schnorr signature. Moreover, we have implicitly shown in our warm-up section that Schnorr signatures are certificate-extractable and DH-simulatable. For a more formal treatment, we refer to the paragraph of the Katz-Wang signatures, which imply the DH simulatability and certificate-extractability of Schnorr.