How To Make Delegated Payments on Bitcoin: A Question for the AI Agentic Future

Jay Taylor[1], Paul Gerhart[2], and Sri AravindaKrishnan Thyagarajan[1]

[1]University of Sydney
[2]TU Wien

Abstract

AI agents and custodial services are increasingly being entrusted as intermediaries to conduct transactions on behalf of institutions. The stakes are high: The digital asset market is projected to exceed $16 trillion by 2030, where exchanges often involve proprietary, time-sensitive goods. Although industry efforts like Google's Agent-to-Payments (AP2) protocol standardize how agents authorize payments, they leave open the core challenge of fair exchange: ensuring that a buyer obtains the asset if and only if the seller is compensated without exposing sensitive information.

We introduce proxy adaptor signatures (PAS), a new cryptographic primitive that enables fair exchange through delegation without sacrificing atomicity or privacy. A stateless buyer issues a single request and does not need to manage long-term cryptographic secrets while proxies complete the exchange with a seller. The seller is guaranteed payment if the buyer can later reconstruct the purchased witness; meanwhile, the proxies remain oblivious to the witness throughout the protocol. We formalize PAS under a threshold model that tolerates the collusion of up to $t - 1$ proxies. We also present an efficient construction from standard primitives that is compatible with Bitcoin, Cardano, and Ethereum. Finally, we evaluate a Rust implementation that supports up to 30 proxies. Our prototype is concretely efficient: buyer and seller computations take place in microseconds, proxy operations in milliseconds, and on-chain costs are equivalent to those of a standard transaction without fair exchange.

Contents

1 Introduction **3**
1.1 Our Contributions 4
1.2 Related Work 5

2 Background **6**
2.1 Adaptor Signatures 6
2.2 Threshold Adaptor Signatures 7

3 Proxy Adaptor Signatures **7**
3.1 Generic Construction 11
3.2 Security Analysis 12

4 Deployment **13**

5 Conclusion **15**

A Additional Preliminaries **20**
A.1 Hard Relations 20
A.2 Digital Signatures 21
A.3 Public Key Encryption 21
A.4 Secret Sharing 22
A.5 Distributed Key Generation 23
A.6 Non-Interactive Zero-Knowledge Arguments 24
A.7 Threshold Signature Schemes 25
A.8 Correctness of Proxy Adaptor Signatures 26

B Proofs for Generic Proxy Adaptor Signature Construction **26**
B.1 Adaptability 26
B.2 Extractability 27
B.3 Witness Privacy 28

C Fairness Model **30**

1 Introduction

The market for digital assets is projected to reach $16.1 trillion by 2030 [GS25], with data marketplaces driving much of this growth through the exchange of datasets, trained AI models, and streaming telemetry [Gra25, AL22]. These transactions raise challenges distinct from traditional finance: digital assets are often proprietary, perishable, and must be transferred fairly without exposing sensitive information. Consider a vendor selling short-lived trading signals to a hedge fund. The fund must verify the signals meet advertised accuracy before paying, while the vendor needs guaranteed compensation if they deliver valid signals. Because the signals are highly time-sensitive and reveal valuable intellectual property, neither side can risk releasing their part first. This setting exemplifies the classic fair-exchange problem: both sides require guarantees that they receive their outcome if and only if the counterparty does too.

Current approaches to fair exchange of a digital good provide varying undesirable tradeoffs. Centralized exchanges, trusted platforms that facilitate trading between users while holding custody of assets, require users to forfeit custody of their assets to a single third party. Decentralized blockchain finance (DeFi) allows for atomicity and trustless, immutable settlement, but can incur substantial fees (eg. gas costs), as most DeFi protocols leverage smart contracts, involving complex business logic executed by a large network of nodes. Additionally, most smart contract-based DeFi solutions are not universally deployable: sellers and proxies may operate across heterogeneous systems, many of which (e.g., Bitcoin, private ledgers, or off-chain environments) lack expressive contract functionality. Recent efforts [Poe17, AEE+21, GSST24] describe protocols that circumvent the need for smart contracts via new cryptographic primitives, but demand continuous involvement from the parties involved in the exchange, often with many rounds of interaction.

This need for continuous direct participation of the principals creates barriers for institutional adoption. In practice, portfolio managers cannot be expected to personally execute cryptographic protocols to acquire market data, nor should AI firm executives need to manage private keys to purchase training datasets. The operational burden includes: (1) maintaining online presence during potentially lengthy negotiation and exchange phases; (2) securely storing and managing cryptographic keys; (3) discovering and vetting potential counter-parties; (4) executing complex cryptographic protocols correctly; and (5) handling dispute resolution when transactions fail. Additionally, recent developments in multi-party compute (MPC) wallets have led to widely-adopted "wallet-as-a-service" offerings, where key custody of a cryptographic wallet is distributed to specialized service providers [blo23]. This introduces added complexity to the exchange, as well as involvement from third parties, who should not necessarily be privy to the exchanged data. Recent rapid developments in agentic AI payments, where agents are able to transact on behalf of users, further demonstrate a tangible demand from institutions to delegate the purchasing of critical business assets to third parties [PS25].

Considering these emerging concerns, we find a critical gap: the absence of a formal framework for *proxy exchanges*. While existing cryptographic techniques enable delegation of certain operations, no existing system permits a buyer to delegate an end-to-end fair exchange to a group of *proxies* while retaining atomicity and privacy. To address institutional needs while maintaining security, we require a fair exchange protocol that satisfies several key properties:

- **Atomicity**: malicious parties should be unable to cheat by obtaining their desired outcome while preventing the counter-party from receiving theirs. This ensures fair exchange even in a trustless environment.
- **Witness privacy**: only the buyer should learn the witness they're purchasing; proxies should not learn the witness simply by participating in the protocol. This is crucial for delegating the purchase of sensitive assets like private keys or proprietary data.

- **Scalability**: the seller's workload should be independent of the number of proxies and ideally agnostic to their existence. If this were not the case, the seller would have to incur costs for a protocol which adds no benefit to them (as the proxy exchange model only benefits the buyer).
- **No buyer-seller interaction**: the buyer and seller do not interact; the seller interacts only with the proxies, as does the buyer. This allows the buyer to remain undiscoverable to the seller and reduces the buyer's online requirements. Clearly, this property requires some form of intermediary, whether it be a proxy or smart contract.
- **Lightweight buyer**: expensive cryptographic operations are offloaded to the proxies and the buyer maintains minimal secure state, performing only lightweight computations. This enables the use of low-power devices or "cold" wallets for the buyer.

Technical Challenges. A natural starting point would be to have the buyer hand over their signing authority (possibly for a temporary wallet used only for the exchange) to a set of proxies using the aforementioned existing solutions. The proxies could then participate in standard cryptographic fair-exchange protocols on behalf of the buyer. However, this naive approach immediately fails as proxies must extract secret information during the exchange process, information that should only be accessible to the buyer. One might attempt to resolve this by having the seller encrypt the sensitive data under the buyer's public key before the exchange. However, this approach is not fair, as the proxies may simply pass on the ciphertext to the buyer before payment is made. It is therefore imperative that we ensure atomicity in the exchange: the seller receives the payment if and only if the proxies learn the ciphertext. Moreover, as a design choice, we aim to minimize heavy cryptographic operations at the buyer's end, particularly to support low-end devices.

Our approach to solving these challenges is to let the proxies securely generate a *blinding factor* to obtain a blinded version of the desired secret witness, rather than the witness itself. No proxy knows the blinding factor, but the buyer, and only the buyer, is able to jointly reconstruct it to "unblind" the blinded witness and derive what they purchased. This retains fairness, witness privacy, and keeps the buyer lightweight. To capture this functionality, we introduce a new cryptographic primitive called *proxy adaptor signatures*. We summarize our contributions below.

1.1 Our Contributions

Formal model. We formalise *proxy adaptor signatures* (PAS), a new primitive that enables a buyer to delegate the purchase of a digital good (or witness wit for an NP statement stmt) to a set of proxies. PAS extends adaptor signatures [GSST24] in the style of proxy-based primitives such as proxy signatures [MUO96] and proxy re-encryption [AFGH06], and is tailored for blockchain-based fair exchange. The construction operates in a threshold setting with n proxies, tolerating corruption of up to $t-1$, and captures exchange-related guarantees in rigorous game-based security definitions that also account for collusion between principals (buyer or seller) and subsets of up to $t-1$ proxies. A defining feature is witness privacy: even though proxies carry out all operations, only the buyer can reconstruct the purchased witness.

At a high level, the buyer, or the proxies on their behalf, can post a request for a witness to a specific statement on a public bulletin board or marketplace, e.g., a job board or a dedicated data marketplace, which serves as an asynchronous communication channel. We make no assumptions on this bulletin board, but assume that potential sellers eventually notice the request. A seller, upon discovering this, publishes an "advertisement", which must include a cryptographic proof of knowledge of a corresponding witness for the statement. The buyer goes offline after issuing a request req to the proxies, who run an interactive $\mathsf{ProxGen}$ protocol followed by a local $\mathsf{Combine}$

to produce a proxy output τ. The seller, when given τ, can then apply Adapt to transform the proxy output into a valid signature σ on a payment transaction m using the witness wit. Then, the seller can publish the signature on-chain, and receive payment. Using τ and σ, any proxy can later run $\mathsf{ProxExt}$ to derive a blinded witness wit_τ and deliver it to the buyer. Crucially, the blinded witness leaks no information about the actual witness to the proxies. Finally, the buyer runs ReqExt locally to unblind and recover the full witness wit, completing the exchange. This design enables proxies to handle the exchange on the buyer's behalf, ensuring atomicity while keeping the buyer's involvement minimal.

Efficient construction. We design an efficient PAS construction that uses standard primitives in a black-box manner, in particular adaptor signatures compatible with Schnorr and EdDSA. This ensures immediate deployability on major blockchain platforms such as Bitcoin, Cardano, and Ethereum, requiring only transaction-signature verification scripts or lightweight contracts. PAS thereby enables delegated exchanges without reliance on complex smart contracts, improving scalability, reducing on-chain costs (e.g., gas fees), and improving fungibility. Our PAS construction lets the buyer be stateless aside from storing a secret to authenticate with the proxies and require negligible computation, and even be offline (needing no interaction with the seller) during the exchange. We provide a rigorous security analysis under standard cryptographic assumptions, demonstrating that the construction satisfies all the required properties of a proxy-based exchange, even in the presence of adversaries corrupting the buyer, seller, or a minority of proxies.

Performance evaluation. We provide an open-source implementation [pas] of our proxy exchange construction in Rust with an end-to-end demonstration on Bitcoin's testnet. We utilize the Secp256k1 curve to demonstrate feasibility with major blockchains, and we provide performance benchmarks for various parameter configurations involving up to 30 proxies participating in the threshold structure. Our results demonstrate that our construction is practical for a variety of real-world use cases: buyer and seller computations are completed in microseconds, proxy computation is performed in milliseconds, and the on-chain footprint is equivalent to that of a standard transaction. We share more details of our implementation in Section 4.

1.2 Related Work

There is extensive literature on fair exchange in blockchain settings. Adaptor signatures, originating from Poelstra's "Scriptless Scripts" [Poe17], enable the atomic fair exchange of witnesses for signatures, offering an attractive, lightweight, and portable alternative to the expensive and complex smart contract-based solutions. The formalization by Aumayr et al. [AEE$^+$21] established security definitions, with subsequent work strengthening these definitions [DOY22,GSST24,GRS25] and demonstrating applications beyond basic payments, including mixing protocols [GMM$^+$22] and oracle-based transfers [MTV$^+$23]. Functional adaptor signatures [VST24] extend this work to enable exchange of function outputs rather than complete witnesses, addressing privacy concerns in data markets.

Several works have extended these cryptographic primitives to multi-party settings. Threshold and multi-adaptor signatures [JXGZ24] allow groups of signers to participate in fair exchange, while consecutive adaptor signatures [KOT24] enable N-party chains of exchanges. Most closely related to our work, Baecker et al. [BGKS25] study fair exchange between two groups (buyer and seller DAOs) with threshold structures within each group. They formalize intra- and inter-group fairness and construct a protocol using time-based mechanisms and certified witness encryption. While one may initially propose a trivial implementation of these schemes for our use case, our

setting differs fundamentally: rather than multiple buyers collaborating to purchase from multiple sellers, we have a single buyer who delegates the exchange process to proxies. In particular, there is some third-party "buyer" that will not actively participate in the protocol, yet should learn the full witness, while the proxies do not.

The idea of delegating signing authority to proxies has been studied extensively [DSP06]. In proxy signature schemes, the original signer derives proxy keys that enable designated parties to sign on their behalf under specified constraints. Such signatures are publicly verifiable, distinguishable from ordinary signatures, and delegation links can be proved. Threshold variants [KPW97, LTH03] distribute authority among several proxies, requiring a quorum for signature generation. However, these schemes assume a persistent original signer who must hold the master key, making them unsuitable for our setting. While some work combines proxy signatures with fair exchange [GAS14], it relies on trusted proxies and is not well-suited for blockchain environments.

To summarise: adaptor signatures (AS) give two-party fairness; threshold adaptor signatures (TAS) distribute signing but the principals hold the shares; functional adaptor signatures (FAS) change the exchanged object (function outputs) but still require buyer-seller interaction; MPC wallets distribute custody but offer no witness privacy or fair exchange; proxy signatures delegate signing but require trust and don't facilitate fair exchange immediately. PAS uniquely combines delegation, threshold control, and witness privacy, allowing proxies to execute the exchange while the buyer remains offline and still learns the witness.

2 Background

We recall the main necessary cryptographic tools that will be used throughout this work and discuss additional preliminaries in Section A.

2.1 Adaptor Signatures

Adaptor signatures [GSST24, AEE+21, Poe17] extend conventional digital signatures to facilitate fair exchange of a signature for a witness to a hard relation. Given a signature scheme $\Sigma = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vf})$ and a hard relation $\mathcal{R}$, an adaptor signature scheme provides four algorithms enabling pre-signature generation, verification, adaptation, and witness extraction.

Definition 2.1. (Adaptor Signature Scheme) An adaptor signature scheme $\mathsf{AS}_{\Sigma,\mathcal{R}} = (\mathsf{PreSign}, \mathsf{PreVerify}, \mathsf{Adapt}, \mathsf{Ext})$ is defined with respect to a signature scheme $\Sigma = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vf})$ and a hard relation $\mathcal{R}$ as follows:

$\tilde{\sigma} \leftarrow \mathsf{PreSign}(\mathsf{sk}, m, \mathsf{stmt})$. *Takes secret key* sk*, message* $m \in \{0,1\}^*$*, and statement* $\mathsf{stmt} \in L_{\mathcal{R}}$*, outputs a pre-signature* $\tilde{\sigma}$*.*
$0/1 := \mathsf{PreVerify}(\mathsf{vk}, m, \mathsf{stmt}, \tilde{\sigma})$. *Takes public key* vk*, message* m*, statement* stmt*, and pre-signature* $\tilde{\sigma}$*, outputs a bit* b*.*
$\sigma \leftarrow \mathsf{Adapt}(\mathsf{vk}, \tilde{\sigma}, \mathsf{wit})$. *Takes public key* vk*, pre-signature* $\tilde{\sigma}$*, and witness* wit *such that* $(\mathsf{stmt}, \mathsf{wit}) \in \mathcal{R}$*, outputs an adapted signature* σ*.*
$\mathsf{wit} := \mathsf{Ext}(\mathsf{vk}, \tilde{\sigma}, \sigma, \mathsf{stmt})$. *Takes public key* vk*, pre-signature* $\tilde{\sigma}$*, signature* σ*, and statement* $\mathsf{stmt} \in L_{\mathcal{R}}$*, outputs witness* wit *or* $\perp$*.*

Definition 2.2. (Pre-signature Correctness) An adaptor signature scheme AS satisfies pre-signature correctness if for all λ, all key pairs $(\mathsf{sk}, \mathsf{vk}) \leftarrow \Sigma.\mathsf{KGen}(\lambda)$, all statements/witnesses $(\mathsf{stmt}, \mathsf{wit}) \in \mathcal{R}$,

and all messages m, the following holds:

$$\Pr\big[\mathsf{PreVerify}(\mathsf{vk}, m, \mathsf{stmt}, \tilde{\sigma}) = 1 \wedge \Sigma.\mathsf{Vf}(\mathsf{vk}, m, \sigma) = 1 \wedge (\mathsf{stmt}, \mathsf{wit}') \in \mathcal{R}\big] = 1,$$

where $\tilde{\sigma} \leftarrow \mathsf{PreSign}(\mathsf{sk}, m, \mathsf{stmt})$, $\sigma \leftarrow \mathsf{Adapt}(\mathsf{vk}, \tilde{\sigma}, \mathsf{wit})$, and $\mathsf{wit}' \leftarrow \mathsf{Ext}(\mathsf{vk}, \tilde{\sigma}, \sigma, \mathsf{stmt})$.

2.2 Threshold Adaptor Signatures

Threshold adaptor signatures generalize adaptor signatures to the threshold setting, enabling any t-out-of-n signers to collaboratively generate pre-signatures.

Definition 2.3. (Threshold Adaptor Signature). A threshold adaptor signature scheme w.r.t. a hard relation $\mathcal{R}$ for some language $L_{\mathcal{R}}$ and a threshold signature scheme $\mathsf{TS} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Sign}, \mathsf{Vf})$ consists of a tuple of four algorithms and protocols $\mathsf{TAS}_{\mathcal{R},\mathsf{TS}} = (\mathsf{PreSign}, \mathsf{Adapt}, \mathsf{PreVerify}, \mathsf{Ext})$ defined as:

$\underline{\tilde{\sigma} \leftarrow \langle \mathsf{PreSign}(\mathsf{sk}_i, \mathsf{vk}, m, \mathsf{stmt}) \rangle}$. *The pre-signing protocol is an interactive algorithm of which an instance is run by each signer $\mathcal{S}_1, \ldots, \mathcal{S}_t$ concurrently, taking as input a signing key share sk_i, the joined public key vk, a message m, and a NP-statement stmt. At the end of the protocol, $\mathcal{S}_i$ obtains a pre-signature $\tilde{\sigma}$ as output.*
$\underline{0/1 \coloneqq \mathsf{PreVerify}(\mathsf{vk}, m, \mathsf{stmt}, \tilde{\sigma})}$. *The pre-verification algorithm is a DPT algorithm that on input a public key vk, message $m \in \{0,1\}^{\ell_m}$, statement $\mathsf{stmt} \in L_{\mathcal{R}}$, and pre-signature $\tilde{\sigma}$, outputs a bit $0/1$.*
$\underline{\sigma \coloneqq \mathsf{Adapt}(\mathsf{vk}, \tilde{\sigma}, \mathsf{wit})}$. *The adapting algorithm is a DPT algorithm that on input a pre-signature $\tilde{\sigma}$ and witness wit for the statement $\mathsf{stmt} \in L_{\mathcal{R}}$ outputs an adapted signature σ, or $\perp$.*
$\underline{\mathsf{wit} \coloneqq \mathsf{Ext}(\mathsf{vk}, \tilde{\sigma}, \sigma, \mathsf{stmt})}$. *The extracting algorithm is a DPT algorithm that on input a public key vk, pre-signature $\tilde{\sigma}$, signature σ, and statement $\mathsf{stmt} \in L_{\mathcal{R}}$, outputs a witness wit such that $(\mathsf{stmt}, \mathsf{wit}) \in \mathcal{R}$, or $\perp$.*

A threshold adaptor signature scheme satisfies *Pre-signature adaptability* if for all $\lambda \in \mathbb{N}$, messages $m \in \{0,1\}^*$, pre-signatures $\tilde{\sigma}$, statement/witness pairs $(\mathsf{stmt}, \mathsf{wit}) \in \mathcal{R}$, and public keys vk,

$$\mathsf{PreVerify}(\mathsf{vk}, m, \mathsf{stmt}, \tilde{\sigma}) = 1 \Rightarrow \mathsf{Vf}(\mathsf{vk}, m, \mathsf{Adapt}(\mathsf{vk}, \tilde{\sigma}, \mathsf{wit})) = 1.$$

A threshold adaptor signature scheme satisfies *Extractability* if for any PPT adversary $\mathcal{A}$, there exists a negligible function negl such that for all $\lambda, t < n \in \mathbb{N}$,

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

where the experiment $\mathsf{Ext}^{\mathcal{A}}_{\mathsf{TAS},t,n}(1^\lambda)$ is defined in Figure 1 and the randomness is taken over all random coins used by the probabilistic algorithms.

3 Proxy Adaptor Signatures

A proxy-enabled exchange involves a buyer, a seller, and n proxies who jointly control a (t, n) threshold wallet. The proxies buy a witness from the seller for the buyer. We assume that compensation for the proxies is handled externally, i.e., the buyer pays for the threshold service off-chain or through some other means.

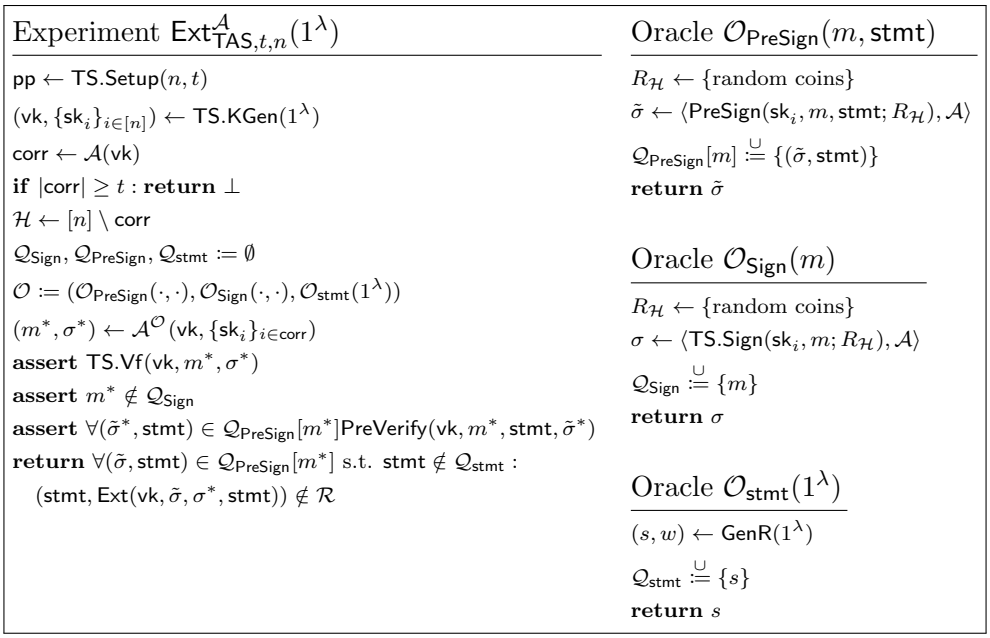

Figure 1: Extractability game for threshold adaptor signatures

Definition 3.1. (Proxy adaptor signatures). A proxy adaptor signature scheme $\mathsf{PAS} := (\mathsf{Setup}, \mathsf{ReqGen}, \mathsf{ReqVerify}, \mathsf{AdGen}, \mathsf{AdVerify}, \mathsf{ProxGen}, \mathsf{Combine}, \mathsf{Adapt}, \mathsf{ProxExt}, \mathsf{ReqExt})$ is defined with respect to a threshold signature scheme $\mathsf{TS} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Sign}, \mathsf{Vf})$ and a hard relation $\mathcal{R}_{\mathsf{DLog}}$. The interfaces are as follows:

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$. *Takes security parameter* 1^λ *and outputs public parameters* pp. *From now on,* pp *is an implicit input to all subsequent algorithms.*
$\mathsf{req} \leftarrow \mathsf{ReqGen}(\mathsf{stmt})$. *Takes statement* $\mathsf{stmt} \in L_{\mathcal{R}_{\mathsf{DLog}}}$ *and outputs a public request object.*
$0/1 := \mathsf{ReqVerify}(\mathsf{stmt}, \mathsf{req})$. *Takes statement* stmt *and request object* req *and outputs* $0/1$*, signifying whether* req *was generated correctly.*
$(\mathsf{st}_{\mathsf{advt}}, \mathsf{advt}) \leftarrow \mathsf{AdGen}(\mathsf{stmt}, \mathsf{wit})$. *Takes statement* stmt*, witness* wit *such that* $(\mathsf{stmt}, \mathsf{wit}) \in \mathcal{R}_{\mathsf{DLog}}$*, and outputs a secret state* $\mathsf{st}_{\mathsf{advt}}$ *and an advertisement* advt*.*
$0/1 := \mathsf{AdVerify}(\mathsf{stmt}, \mathsf{advt})$. *Takes statement* stmt*, advertisement* advt*, and outputs* $0/1$*, signifying whether* advt *was generated correctly.*
$(\{\mathsf{st}^i_\tau\}_{i\in\mathcal{S}}, \tilde{\tau}, \mathcal{S}) \leftarrow \langle \mathsf{ProxGen}(\mathsf{sk}_i, \mathsf{vk}, \mathsf{req}, \mathsf{advt}, \mathsf{stmt}, m)\rangle$. *The proxy generation protocol is an interactive algorithm of which an instance* $\mathsf{ProxGen}_i$ *is run by each proxy signer* $\mathcal{S}_1, \ldots, \mathcal{S}_t$ *concurrently.* $\mathsf{ProxGen}_i$ *takes a secret key share* sk_i*, corresponding shared verification key* vk *(previously generated by* TS.KGen*), request object* req*, advertisement* advt*, statement* stmt*, and message* m*. At the end of the protocol, signer* i *obtains public output* $\tilde{\tau}$ *and secret output* st^i_τ*. Additionally the index set of signers* $\mathcal{S}$ *is outputted.*
$\tau/\perp := \mathsf{Combine}(\mathsf{vk}, m, \mathsf{advt}, \mathsf{stmt}, \tilde{\tau}, \mathcal{S}, \mathsf{req})$. *Takes verification key* vk*, message* m*, advertisement* advt*, statement* stmt*, partial proxy output* $\tilde{\tau}$*, request* req*, and outputs* τ*, the full proxy output, or* $\perp$*.*
$\sigma/\perp := \mathsf{Adapt}(\mathsf{st}_{\mathsf{advt}}, \mathsf{wit}, \tau, \mathsf{vk})$. *Takes secret state* $\mathsf{st}_{\mathsf{advt}}$*, witness* wit*, combined proxy output* τ*, verification key* vk*, and outputs a signature for the message* m *and verification key* vk*, or* $\perp$*.*
$\mathsf{wit}_\tau := \mathsf{ProxExt}(\mathsf{vk}, \tau, \sigma, \mathsf{stmt})$. *Takes verification key* vk*, proxy output* τ*, signature* σ*, statement* stmt *and outputs a blinded witness* wit_τ*.*

$\underline{\mathsf{wit} \coloneqq \mathsf{ReqExt}(\{(i, \mathsf{st}_\tau^i)\}_{i\in\mathcal{S}}, \mathsf{wit}_\tau, \mathsf{vk})}$. *Takes a set* $\{(i, \mathsf{st}_\tau^i)\}_{i\in\mathcal{S}}$ *of secret states, a blinded witness* wit_τ*, a verification key* vk*, and outputs a witness* wit *for the statement* stmt*.*

After setup of public parameters (via Setup) and threshold wallet key shares (via $\mathsf{TS.KGen}$), a typical exchange proceeds as follows. The buyer runs ReqGen on a statement stmt and forwards the resulting public request req to the proxies; the proxies first validate req using $\mathsf{ReqVerify}$ before participating (e.g., to ensure the request is well-formed for the intended statement). A seller, who has a valid witness wit for the statement stmt, responds by publishing an advertisement advt (and keeping secret state $\mathsf{st}_{\mathsf{advt}}$), produced by AdGen; the proxies validate the advertisement via $\mathsf{AdVerify}$, which proves the seller's knowledge of a valid witness.

Next, a threshold of proxies runs the interactive protocol $\mathsf{ProxGen}$, using their signing key shares together with req, advt, stmt, and the target signing message m (typically a transaction to pay the seller). This produces a public transcript $\tilde{\tau}$ and per-proxy secret outputs st_τ^i (which are later delivered to the buyer). Anyone can then run $\mathsf{Combine}$ on $\tilde{\tau}$ to verify the request/advertisement and any NIZKs included by the proxies, and to obtain a constant-size proxy output τ (or $\perp$). The seller uses Adapt to turn τ into a full signature σ on m (e.g., posting a payment transaction on-chain). After σ is available, any proxy can run $\mathsf{ProxExt}$ on (τ, σ) to derive a blinded witness wit_τ for the buyer. Finally, once the buyer collects at least t secret outputs $\{(i, \mathsf{st}_\tau^i)\}$ from the proxies, they run ReqExt to recover the witness wit.

We now formally define various security properties of a proxy adaptor signature scheme. For readability, we assume that whenever the adversary aborts an oracle interaction, the oracle simulating the protocol party aborts the ongoing protocol execution and returns $\perp$. We defer correctness to Section A.8

Adaptability. Similar to the pre-signature adaptability of a threshold adaptor scheme, a proxy adaptor signature scheme satisfies *adaptability* if, given partial outputs $\tilde{\tau}$, a successful generation of a τ using $\mathsf{Combine}$ implies that knowing a valid witness wit, a valid signature can be adapted from τ using Adapt.

Definition 3.2. (Adaptability). A proxy adaptor signature scheme PAS satisfies *adaptability* if for all $\lambda \in \mathbb{N}$, messages $m \in \{0,1\}^*$, $(\mathsf{stmt}, \mathsf{wit}) \in \mathcal{R}_{\mathsf{DLog}}$, public keys vk, requests req, such that $\mathsf{ReqVerify}(\mathsf{stmt}, \mathsf{req}) = 1$, $(\mathsf{st}_{\mathsf{advt}}, \mathsf{advt}) \leftarrow \mathsf{AdGen}(\mathsf{stmt}, \mathsf{wit})$, partial proxy outputs $\tilde{\tau}$ generated with signing set $\mathcal{S}$, and $\tau \leftarrow \mathsf{Combine}(\mathsf{vk}, m, \mathsf{advt}, \mathsf{stmt}, \tilde{\tau}, \mathcal{S}, \mathsf{req})$:

$$\tau \neq \perp \Rightarrow \mathsf{TS.Vf}(\mathsf{vk}, m, \mathsf{Adapt}(\mathsf{st}_{\mathsf{advt}}, \mathsf{wit}, \tau, \mathsf{vk})) = 1.$$

Extractability. We adopt a similar definition of extractability from threshold adaptors (Theorem 2.3), letting the adversary corrupt the seller and $n-t-1$ proxies ($t \geq n/2$). We limit the set of corrupted proxies as otherwise the adversary can trivially win by not sending back their secret outputs from $\mathsf{ProxGen}$, preventing the buyer from obtaining the necessary t shares required in ReqExt. Intuitively, our definition of extractability captures that a malicious seller and $n-t-1$ proxies cannot prevent the buyer from obtaining a valid witness from an adapted signature. Similarly to the new notion of extractability introduced by [GSST24], we also capture unforgeability–the adversary cannot generate a valid signature without access to the corresponding witness for a statement.

Definition 3.3. (Extractability). A proxy adaptor signature scheme PAS satisfies *extractability* if for any PPT adversary $\mathcal{A}$, there exists a negligible function negl such that for all $\lambda, t < n \in \mathbb{N}$, $\Pr[\mathsf{paExt}^{\mathcal{A}}_{\mathsf{PE},t,n}(1^\lambda) = 1] \leq \mathsf{negl}(\lambda)$, where the experiment $\mathsf{paExt}^{\mathcal{A}}_{\mathsf{PE},t,n}(1^\lambda)$ is defined in Figure 2 and the randomness is taken over all random coins used by the probabilistic algorithms.

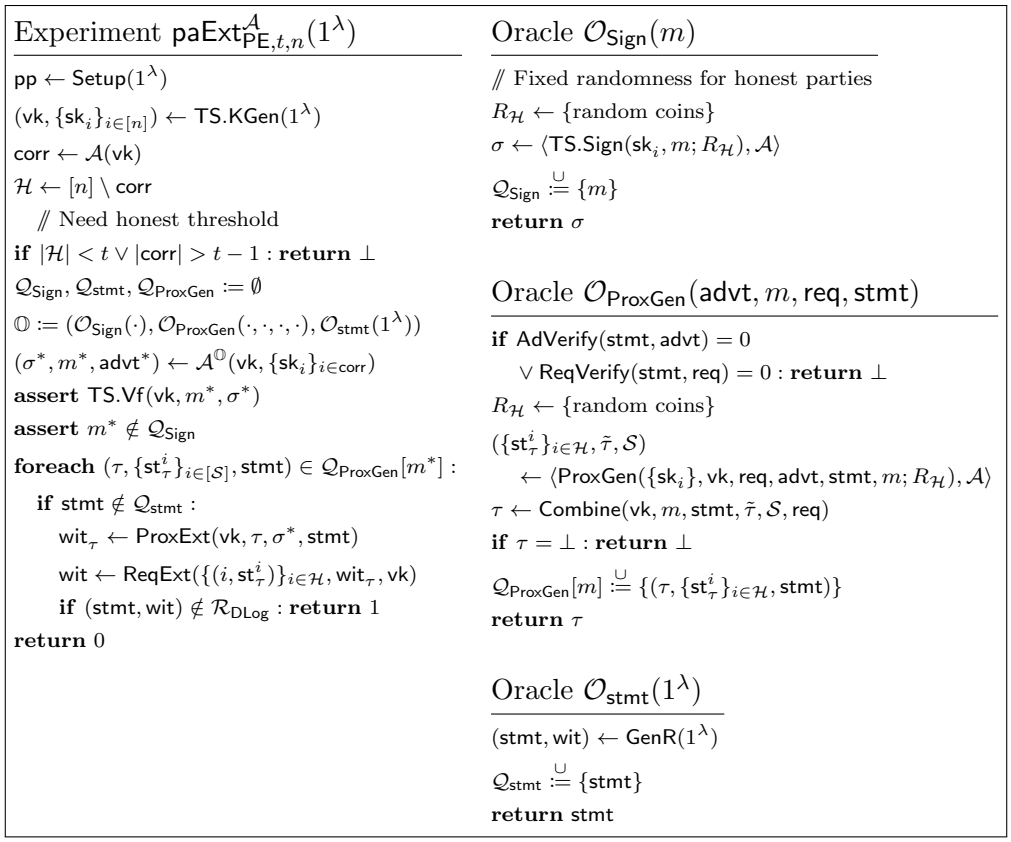

Figure 2: Extractability game for proxy adaptor signatures.

Witness Privacy. Witness privacy captures that the adversary cannot learn any information about the witness, even if $t-1$ proxies are corrupted. Intuitively, the size of the corrupted set is bounded by t, as otherwise the adversary will have access to t of the secret outputs from $\mathsf{ProxGen}$, allowing them the same power as the buyer in that they can run ReqExt themselves to obtain the full witness. We formally define our notion of witness privacy as zero knowledge, requiring that a simulator should be able to simulate the real experiment for the adversary without access to the witness.

Definition 3.4. (Witness Privacy). A proxy adaptor signature scheme PAS satisfies *witness privacy* if for every probabilistic polynomial-time (PPT) adversary $\mathcal{A}$, there exists a stateful PPT simulator $\mathsf{Sim} = (\mathsf{ReqGen}^*, \mathsf{AdGen}^*, \mathsf{ProxGen}^*, \mathsf{Adapt}^*)$ and a negligible function negl such that for all security parameters $\lambda \in \mathbb{N}$, for all $(\mathsf{stmt}, \mathsf{wit}) \in \mathcal{R}_{\mathsf{DLog}}$, and for all PPT distinguishers D

$$\left|\Pr[\mathsf{D}(\mathsf{paZKReal}^{\mathcal{A}}_{\mathsf{PAS},t,n}(1^\lambda, \mathsf{stmt}, \mathsf{wit})) = 1] - \Pr[\mathsf{D}(\mathsf{paZKIdeal}^{\mathcal{A}}_{\mathsf{PAS},t,n}(1^\lambda, \mathsf{stmt})) = 1]\right| \leq \mathsf{negl}(\lambda).$$

We define the experiments $\mathsf{paZKReal}^{\mathcal{A}}_{\mathsf{PAS},t,n}$ and $\mathsf{paZKIdeal}^{\mathcal{A}}_{\mathsf{PAS},t,n}$ in Figure 3.

Our definitions imply that proxy adaptor signatures achieve fair exchange between the buyer and the sellers, similar to fairness for ordinary adaptor signatures [GSST24]. Extractability ensures that if the proxies make a payment, the buyer can recover a requested witness; hence, we have fairness for the buyer. We ensure fairness for the seller, as the advertisement does not disclose any information about the witness (protected by witness privacy), and aside from the advertisement,

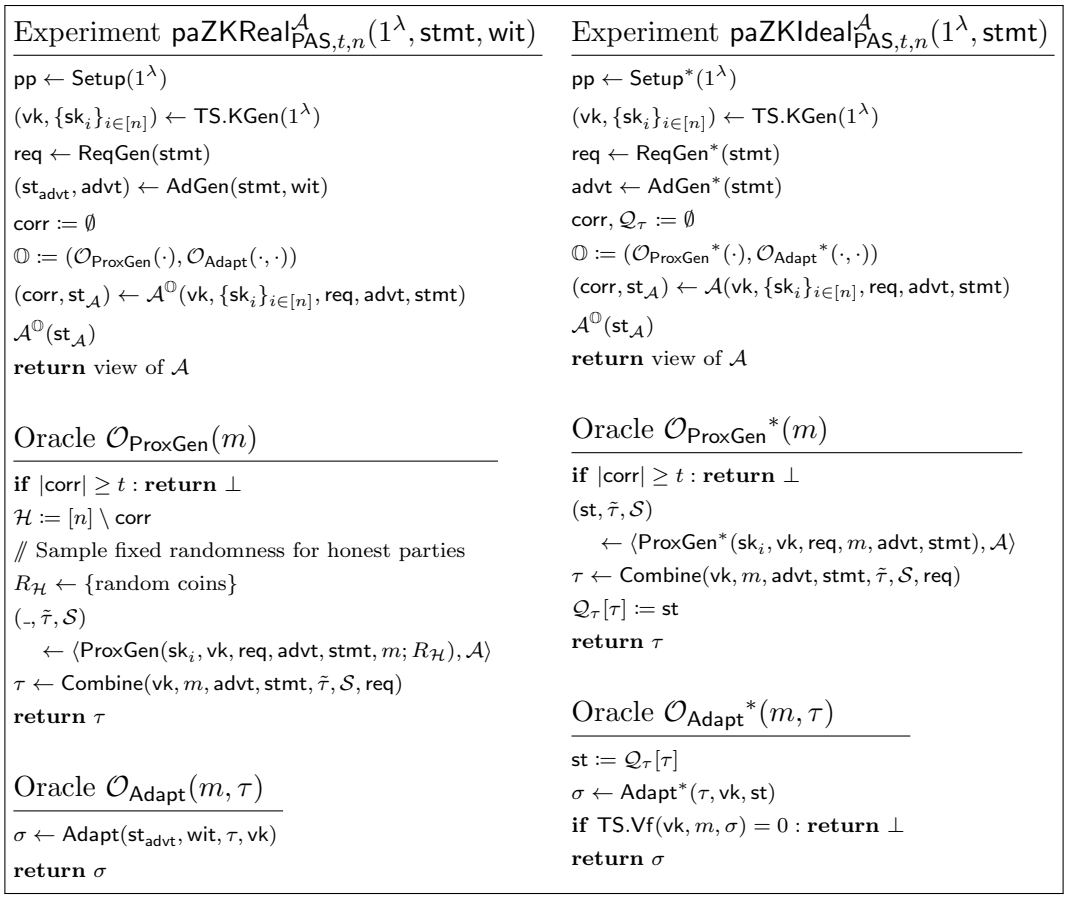

Figure 3: Witness privacy game for proxy adaptor signatures (real vs. simulated).

the seller only posts a valid signature on-chain. Therefore, whenever the seller leaks the witness (which happens only through posting the signature), the seller is already paid.

While our definitions achieve fairness in a cryptographical sense, i.e., the extracted witness matches the advertised one, our definition of proxy adaptor signatures is agnostic to the semantic quality of the witness. The semantic quality must be asserted by the higher-level protocol using proxy adaptor signatures to realize fair exchange, but is out of scope of the actual exchange.

As an example of how the semantic quality of a witness can differ, consider the case of "asset race conditions", where the witness is a wallet's private key. If the seller exchanges this key in a fair manner, but moves the funds before the buyer claims them, the exchange was fair, but not what the buyer desired. To mitigate such a race condition, a higher-level protocol would utilize a time-locked multisignature between the proxies and the seller to ensure that the seller cannot transfer the funds until after the exchange is complete. We defer a formal treatment of how proxy adaptor signatures achieve fairness to Section C.

3.1 Generic Construction

Let $\mathcal{R}_{\mathsf{DLog}}$ be a discrete-log NP relation with language $L_{\mathcal{R}_{\mathsf{DLog}}}$ such that $(\mathsf{stmt}, \mathsf{wit}) \in \mathcal{R}_{\mathsf{DLog}} \Leftrightarrow \mathsf{stmt} = g^{\mathsf{wit}}$, for some generator g of a prime order group. Let $\mathcal{R}_{\mathsf{PKE}}$ be an encryption correctness relation with language $L_{\mathcal{R}_{\mathsf{PKE}}}$ such that $((g^x, c, \mathsf{pk}), x) \in \mathcal{R}_{\mathsf{PKE}} \Leftrightarrow c = \mathsf{PKE.Enc}(\mathsf{pk}, x)$ for some secret value x and public key pk. This relation captures that a ciphertext c is the encryption of the

discrete-log of some public commitment g^x under the public key pk. Note that $\mathcal{R}_{\mathsf{DLog}}$ is practical for many use cases, as the exchanged witness could simply be a key to access sensitive data securely hosted elsewhere. This permits a protocol design that can feasibly support any type of data product using the compiler of [TSZ+24]. Our construction relies on the following cryptographic building blocks:

- NIZK argument systems $\mathsf{NIZK}_{\mathsf{DLog}}$ for discrete-log relation $\mathcal{R}_{\mathsf{DLog}}$ and $\mathsf{NIZK}_{\mathsf{PKE}}$ for encryption relation $\mathcal{R}_{\mathsf{PKE}}$, where $\mathsf{NIZK}_{\mathsf{PKE}}$ has an online extractor
- A (t, n) threshold adaptor signature scheme $\mathsf{TAS}_{\mathcal{R}_{\mathsf{DLog}},\mathsf{TS}}$ with guaranteed output delivery
- A linearly homomorphic encryption scheme PKE
- A (t, n) distributed key generation (DKG) scheme for $\mathcal{R}_{\mathsf{DLog}}$

To increase readability, we assume that $\mathsf{DKG.KGen}$ provides an interface $(Y, y_i) \leftarrow \mathsf{KGen}_i(1^\lambda)$ for each party $i \in [n]$, which returns the public and individual private output of the interactive DKG algorithm for party i.

In our generic construction, the buyer's request equals the statement. To create an advertisement, the seller computes a $\mathsf{NIZK}_{\mathsf{DLog}}$ proving knowledge of a witness which corresponds to the buyer's statement, as well as a public encryption key. This advertisement can be verified by verifying $\mathsf{NIZK}_{\mathsf{DLog}}$. To generate a proxy, the proxies collectively generate shares of a secret blinding factor r (and corresponding public output $R \coloneqq g^r$) through a DKG, where each proxy i obtains a share r_i, without any individual proxy learning the full blinding factor r. Proxy i then encrypts their share of the blinding factor r_i with the seller's encryption key, and proves the correctness of this encryption using $\mathsf{NIZK}_{\mathsf{PKE}}$. The proxies then interactively generate a pre-signature that is bound to a blinded version of the statement $\mathsf{stmt} \cdot R$. They also homomorphically combine all of the encrypted shares r_i into an encryption c of the blinding factor r.

To adapt, the seller derives a full signature by first decrypting the encryption of r and adapting the pre-signature with a blinded version of the witness $\mathsf{wit} + r$. To proxy extract, the proxies run the extract algorithm of the adaptor signature based on the previously computed pre-signature and the adapted signature (which the seller publishes on-chain to receive payment). To extract the witness, the buyer receives the shares of r and the blinded witness from the proxies and subtracts r from the blinded witness. We detail the construction in Figure 4.

3.2 Security Analysis

We now state the main security properties of our proxy adaptor signature construction. We defer the formal proofs to Section B.

Lemma 3.5. Suppose TAS is pre-signature adaptable, $\mathsf{NIZK}_{\mathsf{PKE}}$ is simulation extractable, and PKE is correct. Then, the proxy adaptor signature construction in Figure 4 satisfies adaptabillty (Theorem 3.2).

Lemma 3.6. Suppose TAS satisfies extractability (Figure 1) and DKG t-securely realizes $\mathcal{F}_{\mathsf{DKG}}$. Then, the proxy adaptor signature construction in Figure 4 satisfies extractability (Theorem 3.3).

Lemma 3.7. Suppose $\mathsf{NIZK}_{\mathsf{DLog}}$, $\mathsf{NIZK}_{\mathsf{PKE}}$ are simulation extractable (Theorem A.7), TAS satisfies pre-signature adaptability (Theorem 2.3), and DKG t-securely realizes $\mathcal{F}_{\mathsf{DKG}}$. Then, the proxy adaptor signature construction in Figure 4 satisfies witness privacy (Theorem 3.4).

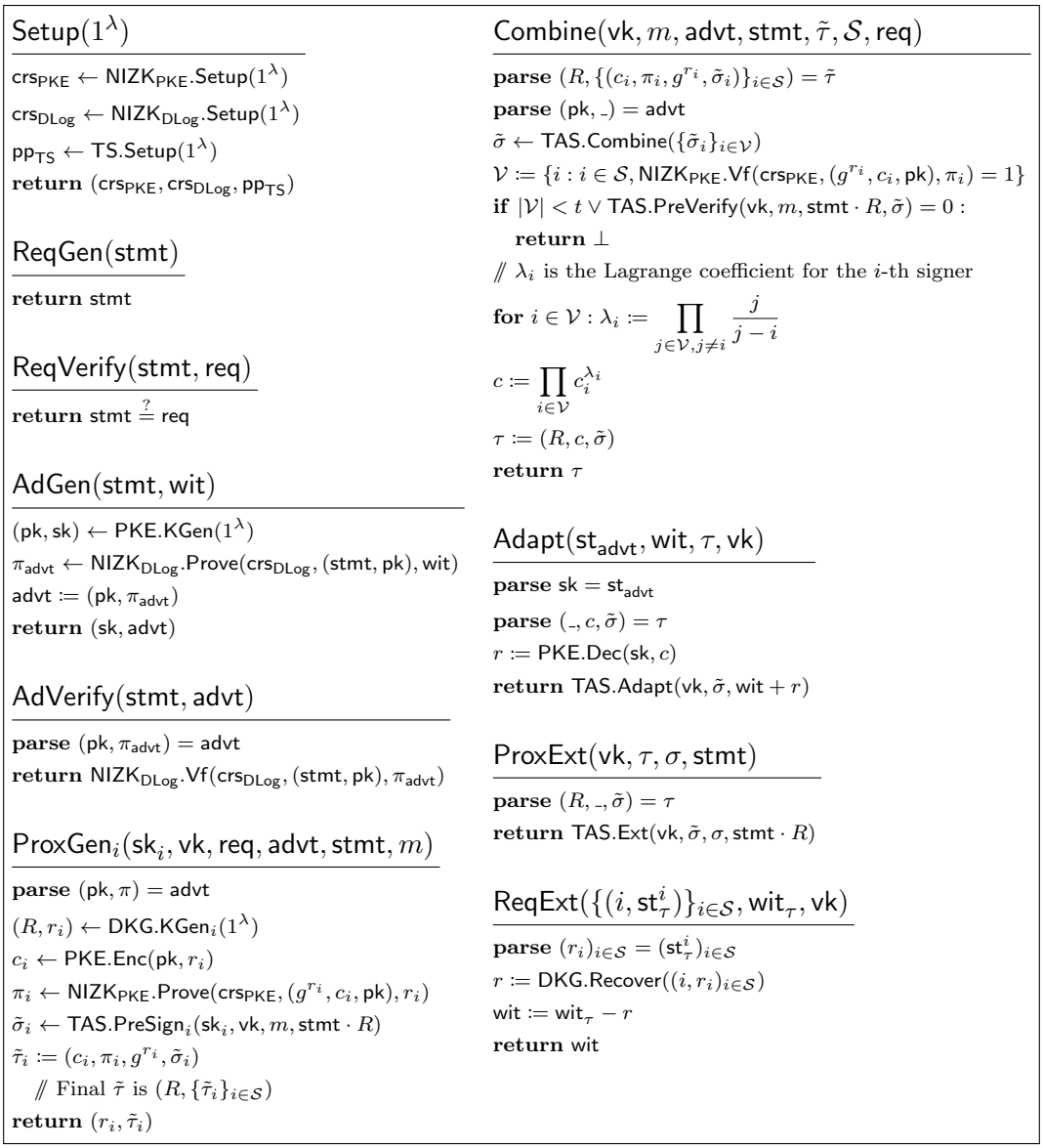

Figure 4: Generic construction for proxy adaptor signatures

4 Deployment

Deployment begins with a one-time proxy setup: n proxies run a DKG to obtain shares of a threshold signing key and publish the joint public key, which serves as the on-chain payment address. A buyer joins by funding this proxy address once, enabling delegated exchanges. Importantly, after funding the proxy, the buyer does not need to hold long-term secrets or state, as long as it can authenticate to the proxy. To initiate a purchase, the buyer issues a lightweight buy assignment; proxies use their threshold shares to produce an adaptor pre-signature tied to a blinded version of the witness. A seller participates by advertising a statement and, upon receiving the pre-signature, adapting it into a valid transaction signature using the blinded witness, which simultaneously transfers payment and reveals the blinded witness. When the buyer comes online again and successfully authenticates, the proxies interact with him to reveal the unblinded witness.

Table 1: Computational Efficiency of Proxy Adaptor Signature Construction

Params		**Running Times (seconds)**						
t	n	AdGen	AdVerify	ProxGen	ProxGen^c	Combine	Adapt	ReqExt
2	3	$6.76 \cdot 10^{-5}$	$1.05 \cdot 10^{-4}$	0.87	$3.4 \cdot 10^{-2}$	$6.7 \cdot 10^{-2}$	$1 \cdot 10^{-2}$	$3.11 \cdot 10^{-5}$
3	5	$4.96 \cdot 10^{-5}$	$7.69 \cdot 10^{-5}$	0.97	$3.5 \cdot 10^{-2}$	$8.7 \cdot 10^{-2}$	$1 \cdot 10^{-2}$	$4.16 \cdot 10^{-5}$
7	10	$4.99 \cdot 10^{-5}$	$8.74 \cdot 10^{-5}$	1.28	$3.5 \cdot 10^{-2}$	0.16	$1 \cdot 10^{-2}$	$8.61 \cdot 10^{-5}$
14	20	$4.83 \cdot 10^{-5}$	$7.25 \cdot 10^{-5}$	1.69	$3.5 \cdot 10^{-2}$	0.29	$1 \cdot 10^{-2}$	$1.69 \cdot 10^{-4}$
16	30	$4.9 \cdot 10^{-5}$	$8.02 \cdot 10^{-5}$	1.79	$3.4 \cdot 10^{-2}$	0.32	$1 \cdot 10^{-2}$	$1.9 \cdot 10^{-4}$
20	30	$5.19 \cdot 10^{-5}$	$7.72 \cdot 10^{-5}$	2.01	$3.5 \cdot 10^{-2}$	0.39	$1 \cdot 10^{-2}$	$2.43 \cdot 10^{-4}$
21	40	$4.73 \cdot 10^{-5}$	$7.25 \cdot 10^{-5}$	2.03	$3.5 \cdot 10^{-2}$	0.41	$1 \cdot 10^{-2}$	$2.53 \cdot 10^{-4}$
27	40	$4.77 \cdot 10^{-5}$	$7.25 \cdot 10^{-5}$	2.38	$3.5 \cdot 10^{-2}$	0.52	$1 \cdot 10^{-2}$	$3.36 \cdot 10^{-4}$
30	40	$4.77 \cdot 10^{-5}$	$7.31 \cdot 10^{-5}$	2.54	$3.5 \cdot 10^{-2}$	0.58	$1 \cdot 10^{-2}$	$5.24 \cdot 10^{-4}$

Prototype. We provide an open-source implementation of our proxy adaptor signature construction in Rust [pas]. We measured the costs on an Apple MacBook Pro with the M3 Max chip, featuring 16 cores (performance cores at 4.05 GHz and efficiency cores at 2.75 GHz), and 128GB of memory. For compatibility with Bitcoin and many other blockchains, we use the Secp256k1 elliptic curve, provided via the `k256` crate [Rus25]. We utilize the Paillier cryptosystem for public-key encryption, implemented efficiently in the `paillier-zk` crate [Var25], which includes a zero-knowledge proof system for discrete-log encryptions, as required. AdGen employs a standard Schnorr-based discrete-log proof of knowledge. Our implementation is a fork of an existing implementation of the Sparkle threshold signature scheme [tra25], which we extend to implement the multi-party fair exchange protocol described in [BGKS25]. We make the following implementation optimizations:

- In ProxGen, we assume the DKG for the blinding factor r is performed beforehand. This is practical because proxies can generate multiple blinding factors ahead of time for different sales in batched protocol runs. Indeed, they would prefer to perform a single DKG run ahead of time for multiple sales, rather than one DKG per sale ad hoc.
- Similarly, we do not include key generation time for the seller's Paillier keys in AdGen, as sellers would typically generate and reuse these keys across multiple sales.

To demonstrate the feasibility of our solution, we additionally provide an end-to-end demonstration of our construction on Bitcoin's testnet [btc]. We establish a (2, 2) multisig between a (2, 3) threshold proxy wallet and a seller's wallet, and execute the proxy adaptor signature protocol to generate a signature from the proxies' side for a transaction that pays the seller 0.00005 BTC in exchange for a secret witness. Our artifact includes the keys used for the example transaction, which can be verified via a CLI command. The total on-chain size for the operation was 564 bytes (246 for the transaction to fund the multisig, and 318 for the proxy adaptor signature exchange).

Computational efficiency. We share the performance results in Table 1. We test for a variety of threshold configurations, ranging from $(t, n) = (2, 3)$ to $(t, n) = (30, 40)$. In each case, t proxies participate in the protocol. We show ProxGen runtimes inclusive and exclusive of local networking times (the latter indicated with ProxGen^c). Considering compute only, Combine is the most expensive operation, scaling linearly with t. Crucially, ReqExt, the buyer's main step, is sub-millisecond, even with larger t, fulfilling our goal of a lightweight buyer. Figure 5 visualizes these trends.

Deployment cost. To contextualize these numbers, we estimate hardware costs on commodity cloud infrastructure. For the smallest threshold setting $(t, n) = (2, 3)$, a proxy requires about 0.93 seconds of compute per exchange, amounting to roughly 259 vCPU-hours for one million exchanges.

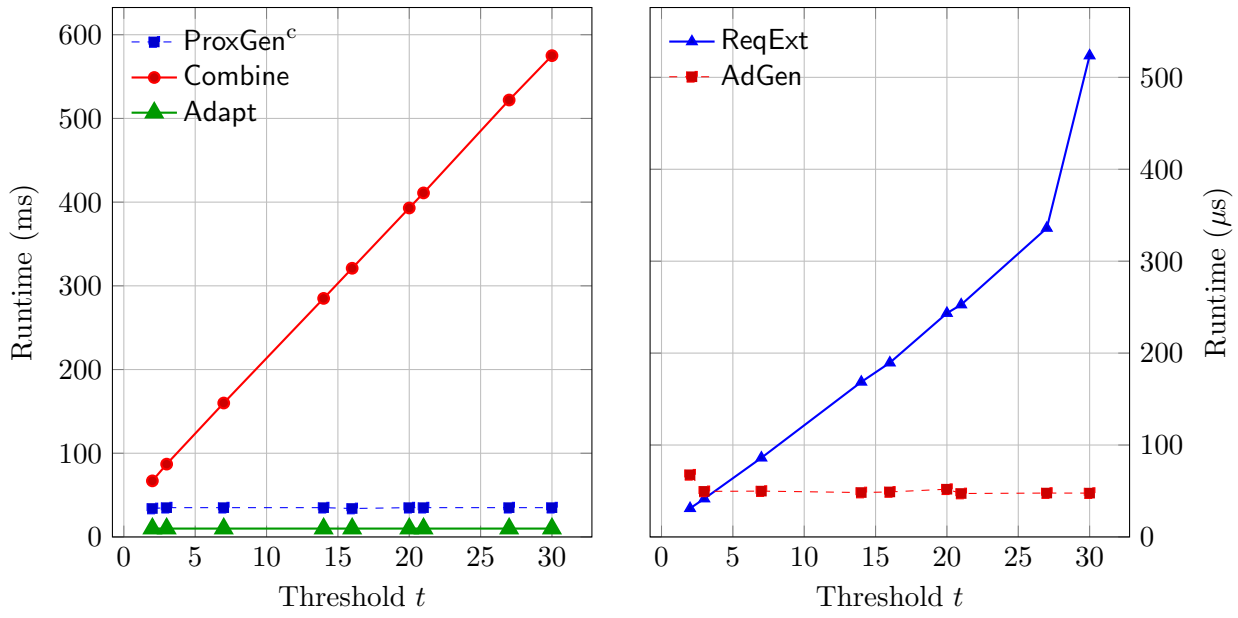

Figure 5: Workload scaling with threshold t. Left: Adaptor steps (ProxGen^c, Combine, Adapt) in milliseconds. Right: buyer/seller auxiliary steps (ReqExt, AdGen) in microseconds.

On AWS, this workload translates to only \$9–12 in compute cost on-demand using `c7i.large` (x86) or `c7g.large` (Arm) instances of comparable performance, with Arm instances being cheaper [aws]. At client scale, the cost per exchange falls below 10^{-5} USD–effectively negligible. On-chain, each exchange corresponds to exactly one standard blockchain transaction (approximately \$1.80 on Bitcoin at the time of writing [YCh25]). Overall, the cost of proxy adaptor signatures is dominated by transaction fees rather than computation, with both proxy and client workloads remaining lightweight and economically viable even at million-scale deployments.

5 Conclusion

We introduced proxy adaptor signatures, formalizing delegated fair exchange among a buyer, seller, and threshold of proxies, and gave a generic construction from standard components (threshold adaptor signatures, NIZKs, DKG, and homomorphic encryption). Our analysis demonstrates adaptability, extractability, and witness privacy, enabling fair exchange while keeping the buyer effectively stateless (up to storing authentication information) and offline, except for a lightweight request/retrieval step.

Future Work. The main limitation of our work is the honest-majority assumption among proxies. One could mitigate this by considering rational adversaries and a slightly stronger blockchain model that is capable of penalizing misbehavior. Another interesting direction is replacing our PKE-based blinding with witness encryption, which may remove the need for seller key generation and further streamline the exchange.

Acknowledgements

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. This work is further partially supported by the Vienna Science and Technology Fund (WWTF) under grant ICT25-081. The research of Paul Gerhart has been supported by the Google PhD Fellowship in Privacy, Safety, and Security.

References

[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, *Advances in Cryptology – ASIACRYPT 2021, Part II*, volume 13091 of *Lecture Notes in Computer Science*, pages 635–664, Singapore, December 6–10, 2021. Springer, Cham, Switzerland.

[AFGH06] Giuseppe Ateniese, Kevin Fu, Matthew Green, and Susan Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. *ACM Transactions on Information and System Security (TISSEC)*, 9(1):1–30, 2006.

[AL22] Santiago Andrés Azcoitia and Nikolaos Laoutaris. A survey of data marketplaces and their business models. *SIGMOD Rec.*, 51(3):18–29, November 2022.

[aws] Amazon ec2 on-demand pricing. `https://aws.amazon.com/ec2/pricing/`. Accessed: 2025-09-07.

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

[BK25] Renas Bacho and Alireza Kavousi. SoK: Dlog-based distributed key generation. Cryptology ePrint Archive, Paper 2025/819, 2025.

[blo23] Builder Vault Overview — builder-vault-tsm.docs.blockdaemon.com. `https://builder-vault-tsm.docs.blockdaemon.com/docs/what-is-builder-vault-tsm`, 2023.

[btc] Example proxy adaptor signature transaction on bitcoin. `https://blockexplorer.one/bitcoin/testnet/tx/befd13840f3f47642f6dc57c11422fe1eb1776a8c6ab5367cc8e6b20ea255065`.

[CGS97] Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. A secure and optimally efficient multi-authority election scheme. In Walter Fumy, editor, *Advances in Cryptology – EUROCRYPT'97*, volume 1233 of *Lecture Notes in Computer Science*, pages 103–118, Konstanz, Germany, May 11–15, 1997. Springer, Berlin, Heidelberg, Germany.

[CHK07] Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. *Journal of Cryptology*, 20(3):265–294, July 2007.

[CKM23] Elizabeth Crites, Chelsea Komlo, and Mary Maller. Fully adaptive schnorr threshold signatures. Cryptology ePrint Archive, Report 2023/445, 2023.

[CL15] Guilhem Castagnos and Fabien Laguillaumie. Linearly homomorphic encryption from DDH. In Kaisa Nyberg, editor, *Topics in Cryptology – CT-RSA 2015*, volume 9048 of *Lecture Notes in Computer Science*, pages 487–505, San Francisco, CA, USA, April 20–24, 2015. Springer, Cham, Switzerland.

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[DMP88] Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. Non-interactive zero-knowledge proof systems. In Carl Pomerance, editor, *Advances in Cryptology – CRYPTO'87*, volume 293 of *Lecture Notes in Computer Science*, pages 52–72, Santa Barbara, CA, USA, August 16–20, 1988. Springer, Berlin, Heidelberg, Germany.

[DOTT22] Ivan Damgård, Claudio Orlandi, Akira Takahashi, and Mehdi Tibouchi. Two-round n-out-of-n and multi-signatures and trapdoor commitment from lattices. *Journal of Cryptology*, 35(2):14, April 2022.

[DOY22] Wei Dai, Tatsuaki Okamoto, and Go Yamamoto. Stronger security and generic constructions for adaptor signatures. In Takanori Isobe and Santanu Sarkar, editors, *Progress in Cryptology - INDOCRYPT 2022: 23rd International Conference in Cryptology in India*, volume 13774 of *Lecture Notes in Computer Science*, pages 52–77, Kolkata, India, December 11–14, 2022. Springer, Cham, Switzerland.

[DR23] Sourav Das and Ling Ren. Adaptively secure BLS threshold signatures from DDH and co-CDH. Cryptology ePrint Archive, Paper 2023/1553, 2023.

[DSP06] Manik Lal Das, Ashutosh Saxena, and Deepak B. Phatak. Algorithms and approaches of proxy signature: A survey. *CoRR*, abs/cs/0612098, 2006.

[ElG85] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.

[Fis05] Marc Fischlin. Communication-efficient non-interactive proofs of knowledge with online extractors. In Victor Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 152–168, Santa Barbara, CA, USA, August 14–18, 2005. Springer, Berlin, Heidelberg, Germany.

[FLS90] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple non-interactive zero knowledge proofs based on a single random string (extended abstract). In *31st Annual Symposium on Foundations of Computer Science*, pages 308–317, St. Louis, MO, USA, October 22–24, 1990. IEEE Computer Society Press.

[GAS14] Kosar Ghorbani, Maryam Rajabzadeh Asaar, and Mahmoud Salmasizadeh. An optimistic fair exchange protocol for proxy signatures. In *7'th International Symposium on Telecommunications (IST'2014)*, pages 913–918, 2014.

[GJKR07] Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Secure distributed key generation for discrete-log based cryptosystems. *Journal of Cryptology*, 20(1):51–83, January 2007.

[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: 29th Conference on Computer and Communications Security*, pages 1259–1273, Los Angeles, CA, USA, November 7–11, 2022. ACM Press.

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, April 1988.

[GOS06] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Non-interactive zaps and new techniques for NIZK. In Cynthia Dwork, editor, *Advances in Cryptology – CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 97–111, Santa Barbara, CA, USA, August 20–24, 2006. Springer, Berlin, Heidelberg, Germany.

[Gra25] Grand View Research. Data marketplace platform market size, share & trends analysis report. Grand View Research, 2025. Market projected to reach USD 5.73 billion by 2030, growing at CAGR of 25.2% from 2025 to 2030.

[GRS25] Paul Gerhart, Daniel Rausch, and Dominique Schröder. Universally composable adaptor signatures. Cryptology ePrint Archive, Paper 2025/1363, 2025.

[GS25] Boston Consulting Group and ADDX Singapore. Digital assets now on a legitimate footing. Vietnam Investment Review, July 2025. Accessed: July 30, 2025.

[GSST24] Paul Gerhart, Dominique Schröder, Pratik Soni, and Sri Aravinda Krishnan Thyagarajan. Foundations of adaptor signatures. In Marc Joye and Gregor Leander, editors, *Advances in Cryptology – EUROCRYPT 2024, Part II*, volume 14652 of *Lecture Notes in Computer Science*, pages 161–189, Zurich, Switzerland, May 26–30, 2024. Springer, Cham, Switzerland.

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

[Kat24] Jonathan Katz. Round-optimal, fully secure distributed key generation. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology – CRYPTO 2024, Part VII*, volume 14926 of *Lecture Notes in Computer Science*, pages 285–316, Santa Barbara, CA, USA, August 18–22, 2024. Springer, Cham, Switzerland.

[KG20] Chelsea Komlo and Ian Goldberg. FROST: Flexible round-optimized Schnorr threshold signatures. In Orr Dunkelman, Michael J. Jacobson Jr., and Colin O'Flynn, editors, *SAC 2020: 27th Annual International Workshop on Selected Areas in Cryptography*, volume 12804 of *Lecture Notes in Computer Science*, pages 34–65, Halifax, NS, Canada (Virtual Event), October 21-23, 2020. Springer, Cham, Switzerland.

[KOT24] Kaisei Kajita, Go Ohtake, and Tsuyoshi Takagi. Consecutive adaptor signature scheme: From two-party to n-party settings. Cryptology ePrint Archive, Paper 2024/241, 2024.

[KPW97] Seungjoo Kim, Sangjoon Park, and Dongho Won. Proxy signatures, revisited. In Yongfei Han, Tatsuaki Okamoto, and Sihan Qing, editors, *ICICS 97: 1st International Conference on Information and Communication Security*, volume 1334 of *Lecture Notes in Computer Science*, pages 223–232, Beijing, China, November 11–14, 1997. Springer, Berlin, Heidelberg, Germany.

[Lin17] Yehuda Lindell. Fast secure two-party ECDSA signing. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part II*, volume 10402 of *Lecture Notes in Computer Science*, pages 613–644, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Cham, Switzerland.

[LTH03] Li-Hua Li, Shiang-Feng Tzeng, and Min-Shiang Hwang. Generalization of proxy signature-based on discrete logarithms11this research was partially supported by the national science council, taiwan, r.o.c., under contract no.: Nsc91-2213-e-324-003. *Computers & Security*, 22(3):245–255, 2003.

[MTV+23] Varun Madathil, Sri Aravinda Krishnan Thyagarajan, Dimitrios Vasilopoulos, Lloyd Fournier, Giulio Malavolta, and Pedro Moreno-Sanchez. Cryptographic oracle-based conditional payments. In *ISOC Network and Distributed System Security Symposium – NDSS 2023*, San Diego, CA, USA, February 2023. The Internet Society.

[MUO96] Masahiro Mambo, Keisuke Usuda, and Eiji Okamoto. Proxy signatures for delegating signing operation. In *Proceedings of the 3rd ACM Conference on Computer and Communications Security*, pages 48–57, 1996.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *Advances in Cryptology – EUROCRYPT'99*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238, Prague, Czech Republic, May 2–6, 1999. Springer, Berlin, Heidelberg, Germany.

[pas] Proxy adaptor signatures implementation. `https://github.com/proxy-adaptor-signatures/pas-rust`.

[PKM+24] Rafaël Del Pino, Shuichi Katsumata, Mary Maller, Fabrice Mouhartem, Thomas Prest, and Markku-Juhani O. Saarinen. Threshold raccoon: Practical threshold signatures from standard lattice assumptions. In Marc Joye and Gregor Leander, editors, *Advances in Cryptology – EUROCRYPT 2024, Part II*, volume 14652 of *Lecture Notes in Computer Science*, pages 219–248, Zurich, Switzerland, May 26–30, 2024. Springer, Cham, Switzerland.

[Poe17] Andrew Poelstra. Scriptless scripts. Presentation Slides, 2017.

[PS25] Stavan Parikh and Rao Surapaneni. Announcing Agent Payments Protocol (AP2), 9 2025.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the Association for Computing Machinery*, 21(2):120–126, February 1978.

[Rus25] RustCrypto. k256. `https://crates.io/crates/k256`, 2025. Accessed: September 13, 2025.

[Sha79] Adi Shamir. How to share a secret. *Communications of the Association for Computing Machinery*, 22(11):612–613, November 1979.

[TPCZ23] Guofeng Tang, Bo Pang, Long Chen, and Zhenfeng Zhang. Efficient lattice-based threshold signatures with functional interchangeability. *Trans. Info. For. Sec.*, 18:4173–4187, January 2023.

[tra25] traffictse. tss-schnorr-sparkle. `https://github.com/traffictse/tss-schnorr-sparkle`, 2025. MIT License; Accessed: September 13, 2025.

[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 *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, CCS '24, page 3227–3241, New York, NY, USA, 2024. Association for Computing Machinery.

[Var25] Varlakov, Denis and maurges. paillier-zk. `https://docs.rs/paillier-zk/latest/paillier_zk/`, 2025. Version 0.4.2; MIT OR Apache-2.0; Accessed: September 13, 2025.

[VST24] Nikhil Vanjani, Pratik Soni, and Sri AravindaKrishnan Thyagarajan. Functional adaptor signatures: Beyond all-or-nothing blockchain-based payments. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, CCS '24, page 1493–1507, New York, NY, USA, 2024. Association for Computing Machinery.

[YCh25] YCharts. Bitcoin average transaction fee (usd per transaction). Website, September 2025. Accessed September 17, 2025; value $1.801/tx.

A Additional Preliminaries

Notations. We fix a security parameter λ and assume all probabilistic algorithms run in time polynomial in λ. A function $\nu : \mathbb{N} \to \mathbb{R}$ is called *negligible* if for every integer $k > 0$ there exists n_0 such that for all $n \geq n_0$, $|\nu(n)| \leq n^{-k}$. We write $x \leftarrow\!\!\$\ X$ to denote that x is sampled uniformly at random from a finite set X. For a probabilistic polynomial-time (PPT) algorithm $\mathcal{A}$, we denote $y \leftarrow \mathcal{A}(x; r)$ when $\mathcal{A}$ on input x and randomness r returns y. If $\mathcal{A}$ is deterministic polynomial-time (DPT), we write $y := \mathcal{A}(x)$. In our games, we use the function call **assert** $\mathsf{condition}$ to enforce invariants: the call aborts the current experiment if $\mathsf{condition}$ evaluates to **false**.

A.1 Hard Relations

We recall the notion of a hard relation $\mathcal{R}$ with statement/witness pairs $(\mathsf{stmt}, \mathsf{wit})$. We denote the associated language $L_{\mathcal{R}} = \{\mathsf{stmt} \mid \exists \mathsf{wit}, (\mathsf{stmt}, \mathsf{wit}) \in \mathcal{R}\}$. We say $\mathcal{R}$ is hard if it satisfies the following properties:

Definition A.1. (Hardness of a Relation) Let $\mathcal{R}$ be a relation with language $L_{\mathcal{R}}$. We require:

- *Sampleability:* There exists a PPT algorithm $(\mathsf{stmt}, \mathsf{wit}) \leftarrow \mathsf{GenR}(1^\lambda)$ that outputs a statement/witness pair $(\mathsf{stmt}, \mathsf{wit}) \in \mathcal{R}$.
- *Decidability:* There exists a DPT algorithm $0/1 := \mathsf{Decide}_{\mathcal{R}}(\mathsf{stmt}, \mathsf{wit})$ that returns 1 if $(\mathsf{stmt}, \mathsf{wit}) \in \mathcal{R}$, and 0 otherwise.
- *Hardness:* For every non-uniform PPT adversary $\mathcal{A}$, there exists a negligible function negl such that
$$\Pr[\mathsf{G}_{\mathcal{R},\mathcal{A}}(1^\lambda) = 1] \leq \mathsf{negl}(\lambda)$$
where the game $\mathsf{G}_{\mathcal{R},\mathcal{A}}(1^\lambda)$ is defined in Figure 6.

$$\boxed{\begin{array}{l}\text{Game } \mathsf{G}_{\mathcal{R},\mathcal{A}}(1^\lambda) \\ \hline (\mathsf{stmt},\mathsf{wit}) \leftarrow \mathsf{GenR}(1^\lambda) \\ \mathsf{wit}' \leftarrow \mathcal{A}(1^\lambda,\mathsf{stmt}) \\ \textbf{return } \mathsf{Decide}_{\mathcal{R}}(\mathsf{stmt},\mathsf{wit}')\end{array}}$$

Figure 6: Hardness game for relation R

A common relation is the discrete logarithm (or DLog) relation, which can be formalised as

$$(\mathsf{stmt},\mathsf{wit}) \in \mathcal{R} \iff \mathsf{stmt} = g^{\mathsf{wit}}$$

for all $\mathsf{stmt},\mathsf{wit} \in \mathbb{G}$ for some group $\mathbb{G}$ with generator g. It's known that for certain groups (e.g. $\mathbb{Z}_p^*$ for prime p) the DLog relation is hard.

A.2 Digital Signatures

In a digital signature scheme [DH76, GMR88], a signer runs the signing algorithm to generate a signature for a message using a private signing key. The validity of the signature can be publicly verified with a verification algorithm that takes as input the signer's public key, the message, and the signature.

Definition A.2. (Digital Signature Scheme) A digital signature scheme $\Sigma = (\mathsf{KGen},\mathsf{Sign},\mathsf{Vf})$ consists of the following efficient algorithms:

- $\underline{(\mathsf{sk},\mathsf{vk}) \leftarrow \mathsf{KGen}(1^\lambda)}$. *Takes a security parameter* 1^λ *and outputs a private/public key pair* $(\mathsf{sk},\mathsf{vk})$.
- $\underline{\sigma \leftarrow \mathsf{Sign}(\mathsf{sk},m)}$. *Takes a private key* sk *and message* m *and outputs a signature* σ.
- $\underline{0/1 := \mathsf{Vf}(\mathsf{vk},m,\sigma)}$. *Takes a public key* vk*, message* m*, and signature* σ*, and outputs a bit indicating acceptance or rejection.*

Definition A.3. (Correctness) A digital signature scheme Σ is correct if, for all security parameters 1^λ, all messages $m \in \{0,1\}^*$, and all key pairs $(\mathsf{sk},\mathsf{vk}) \leftarrow \mathsf{KGen}(1^\lambda)$, it holds that:

$$\Pr\big[\mathsf{Vf}(\mathsf{vk},m,\mathsf{Sign}(\mathsf{sk},m)) = 1\big] = 1,$$

where the probability is taken over the randomness of KGen and Sign.

We require the digital signature scheme to satisfy strong existential unforgeability [GMR88].

A.3 Public Key Encryption

A public key encryption (PKE) scheme provides confidentiality in the asymmetric-key model: a sender uses the recipient's public key to compute a ciphertext, and the recipient recovers the plaintext by applying the corresponding secret key [DH76, RSA78, CL15].

Definition A.4. (Public Key Encryption). A public key encryption scheme $\mathsf{PKE} = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ for message space $\mathcal{M}$ consists of the following algorithms defined as:

$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\lambda)$. *Takes a security parameter* 1^λ *and outputs a public/secret key pair* $(\mathsf{pk}, \mathsf{sk})$.

$c \leftarrow \mathsf{Enc}(\mathsf{pk}, m)$. *Takes a public key* pk *and a message* $m \in \mathcal{M}$ *and outputs a ciphertext* c.

$m := \mathsf{Dec}(\mathsf{sk}, c)$. *Takes a secret key* sk *and a ciphertext* c *and outputs the decrypted message* m *or* $\perp$.

A **linearly homomorphic** PKE scheme additionally has the property that for any two ciphertexts c_1, c_2 decrypting to m_1, m_2, it is the case that $c_1 c_2$ decrypts to $m_1 + m_2$, and for a scaler α, it is the case that c_1^α decrypts to αm_1. These properties are captured by the following evaluation algorithms:

$c^* := \mathsf{EvalAdd}(c_1, c_2)$. *Takes as input two ciphertexts* c_1 *and* c_2*, and outputs a new ciphertext* c^* *which decrypts to* $m_1 + m_2$ *where each* $c_i, i \in \{1, 2\}$ *decrypts to* m_i.

$c^* := \mathsf{EvalScal}(c, \alpha)$. *Takes as input a ciphertext* c *which decrypts to* m*, and a scalar* α*, and outputs a new ciphertext* c^* *which decrypts to* $\alpha \cdot m$.

A PKE scheme is **IND-CPA secure** (or semantically secure under chosen plaintext attack) if for any PPT adversary $\mathcal{A}$, there exists a negligible function negl such that for all $\lambda \in \mathbb{N}$,

$$\Pr[\text{IND-CPA}_{\mathsf{PKE},\mathcal{A}}(1^\lambda) = 1] \leq \frac{1}{2} + \mathsf{negl}(\lambda),$$

where $\text{IND-CPA}_{\mathsf{PKE},\mathcal{A}}$ is defined in Figure 7.

$\text{IND-CPA}_{\mathsf{PKE},\mathcal{A}}(1^\lambda)$

$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\lambda)$
$(m_0, m_1) \leftarrow \mathcal{A}(\mathsf{pk})$
$b \leftarrow \{0, 1\}$
$c^* \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b)$
$b' \leftarrow \mathcal{A}(c^*)$
return $(b' = b)$

Figure 7: IND-CPA security game IND-CPA for public-key encryption

A notable efficient linearly-homomorphic PKE scheme is the Paillier cryptosystem [Pai99], derived from the decisional composite residuosity assumption. Certain schemes offer limited homomorphism, such as the ElGamal cryptosystem [ElG85], which provides the homomorphic property that $c_1 \cdot c_2$ decrypts to $m_1 \cdot m_2$. Variants of ElGamal, notably "Exponential" and "Hashed" ElGamal, offer limited additive homomorphism, useful in certain applications like secure voting schemes [CGS97].

A.4 Secret Sharing

Secret sharing [Sha79] involves the distribution of a secret into n shares such that any t subset of the shares can recover the original secret, but no subset of size less than t can recover any information

about the original secret.

Definition A.5. (Secret Sharing). A (n, t) secret sharing scheme $\mathsf{SS} = (\mathsf{Share}, \mathsf{Recover})$ consists of the following algorithms defined as:

$(d_1, \ldots, d_n) \leftarrow \mathsf{Share}(s)$. *Takes a secret s, and outputs an ordered n-tuple of shares $(d_1, \ldots, d_n)$.*

$s \coloneqq \mathsf{Recover}((a_1, d_1), ..., (a_k, d_k))$. *Takes a set of $k \geq t$ ordered pairs (a_i, d_i), where $a_i \in [n]$, and outputs the recovered secret s.*

In Shamir's original scheme, the secret sharing involves the sampling of a $t-1$ degree polynomial and the evaluation of the polynomial at n points. The shares are the evaluations of the polynomial at these points. Using the Lagrange interpolation formula, any t shares can be used to recover the secret. Specifically, for each $i \in [n]$, the Lagrange coefficient λ_i is defined as:

$$\lambda_i = \prod_{1 \leq j \leq n, j \neq i} \frac{a_j}{a_j - a_i}$$

The secret is then recovered as:

$$s = \sum_{i=1}^{n} \lambda_i s_i$$

For our construction and games, we assume that shared secrets can be recovered in this manner.

A.5 Distributed Key Generation

A distributed key generation (DKG) scheme is a protocol for a group of parties to jointly generate a shared secret key. It amounts to sharing a random, uniformly distributed secret where no one party ever learns the original jointly computed secret. We follow definitions from [GJKR07,Kat24,BK25]. For the purpose of our construction, we assume a synchronous communication model between the participants.

Definition A.6. (DLog Based Verifiable Distributed Key Generation). A (n, t) DLog-based verifiable distributed key generation scheme $\mathsf{DKG} = (\mathsf{KGen}, \mathsf{Recover})$ defined with respect to a group $\mathbb{G}$ with generator g consists of the following algorithms:

$(Y, (g^{y_i})_{i \in [n]}, (y_i)_{i \in [n]}) \leftarrow \langle \mathsf{KGen}(1^\lambda) \rangle$. *Executed interactively by n parties, outputs a public value Y, public commitments $(g^{y_i})_{i \in [n]}$, and secret shares $(y_i)_{i \in [n]}$, where each party i receives share y_i.*

$y \coloneqq \mathsf{Recover}((i_1, y_{i_1}), \ldots, (i_k, y_{i_k}))$. *Takes a set of $k \geq t$ ordered pairs (i_j, y_{i_j}), where $i_j \in [n]$, and outputs the secret key y corresponding to Y.*

Security. A DKG scheme $t-$securely realizes $\mathcal{F}_{\mathsf{DKG}}$ if for any PPT adversary $\mathcal{A}$ corrupting up to $t-1$ parties, there exists a PPT simulator $\mathcal{S}$ such that no distinguisher can distinguish $\mathcal{A}$ interacting in the real protocol and $\mathcal{S}$ interacting with $\mathcal{F}_{\mathsf{DKG}}$. We provide the functionality $\mathcal{F}_{\mathsf{DKG}}$ in Figure 8. We note, that this strictly implies correctness (honest parties output valid shares consistent with Y) and unbiasability (the adversary cannot influence the distribution of Y), assuming the protocol does not abort. This notion of security implies the notions of secrecy and robustness [BK25], which we recite here.

$\mathcal{F}_{\mathsf{DKG}}$

1. Sample $y \leftarrow\!\!\$\ \mathbb{Z}_p$ uniformly at random, and then compute $Y = g^y \in \mathbb{G}$.
2. Generate (t, n)-threshold Shamir shares $\{y_1, \ldots, y_n\}$ of y. Set $Y_i = g^{y_i}$ for $i \in [n]$, and $\vec{Y} \leftarrow (Y_1, \ldots, Y_n)$.
3. Send $(Y, \vec{Y})$ to the adversary.
4. The adversary responds with $\mathsf{continue}$ or abort. If abort, send $(\mathsf{abort}, \perp)$ to each honest party, otherwise send y_i to party i.

Figure 8: The ideal DKG functionality with abort.

Secrecy. A DKG protocol is $t-$secret, if for any PPT adversary $\mathcal{A}$ corrupting up to $t-1$ parties, there exists a PPT simulator $\mathcal{S}$ that on input a uniformly random element $Y \in \mathbb{G}$, produces a view which is indistinguishable from $\mathcal{A}$'s view in a run of Π_{DKG} that terminates with output Y as the public key.

Robustness. A DKG-protocol is t-robust if in the presence of up to $t-1$ corrupt parties, if the adversary does not abort the protocol, every honest party i outputs the same tuple $(Y, \vec{Y})$, and its secret key share y_i satisfies $Y_i = g^{y_i}$. Furhter for any two sets, each having at least t honest shares, a unique secret key y can be reconstructed, where $g^y = Y$.

A.6 Non-Interactive Zero-Knowledge Arguments

Definition A.7. (Secure NIZK Argument System). A NIZK [DMP88, Fis05] for language $L_{\mathcal{R}}$ in the common reference string (CRS) model consists of the following possibly randomized algorithms $\mathsf{NIZK}_{\mathsf{DLog}} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Vf})$ defined as:

$\underline{\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)}$. *It takes in a security parameter* 1^λ *and outputs the common reference string* crs *which is publicly known to everyone.*

$\underline{\pi \leftarrow \mathsf{Prove}(\mathsf{crs}, \mathsf{stmt}, \mathsf{wit})}$. *It takes in the common reference string* crs*, a statement* $\mathsf{stmt} \in L_{\mathcal{R}}$*, and a witness* wit *such that* $(\mathsf{stmt}, \mathsf{wit}) \in \mathcal{R}$*, and outputs a proof* π*.*

$\underline{0/1 := \mathsf{Vf}(\mathsf{crs}, \mathsf{stmt}, \pi)}$. *It takes in the common reference string* crs*, a statement* $\mathsf{stmt} \in L_{\mathcal{R}}$*, and a proof* π*, and either accepts (with output* 1*) the proof, or rejects (with output* 0*) it.*

A secure NIZK argument system must satisfy the following properties:

- **Completeness:** For all $\mathsf{stmt}, \mathsf{wit}$ where $\mathcal{R}(\mathsf{stmt}, \mathsf{wit}) = 1$, if $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$ and $\pi \leftarrow \mathsf{Prove}(\mathsf{crs}, \mathsf{stmt}, \mathsf{wit})$, then,
$$\Pr[\mathsf{Vf}(\mathsf{crs}, \mathsf{stmt}, \pi) = 1] = 1.$$
- **Adaptive Soundness:** For all non-uniform PPT provers P^*, there exists a negligible function negl such that for all $\lambda \in \mathbb{N}$ and $(\mathsf{stmt}, \pi) \leftarrow \mathsf{P}^*(\mathsf{crs})$, then,
$$\Pr[\mathsf{stmt} \notin L_{\mathcal{R}} \wedge \mathsf{Vf}(\mathsf{crs}, \mathsf{stmt}, \pi) = 1] \leq \mathsf{negl}(\lambda).$$

- **Zero-Knowledge:** There exists a PPT simulator $\mathsf{Sim} = (\mathsf{Setup}^*, \mathsf{Prove}^*)$ and there exists a negligible function negl such that for all PPT distinguishers D, for all $\lambda \in \mathbb{N}$, for all $(\mathsf{stmt}, \mathsf{wit}) \in \mathcal{R}$,
$$|\Pr[\mathsf{D}(\mathsf{crs}, \pi) = 1] - \Pr[\mathsf{D}(\mathsf{crs}^*, \pi^*) = 1]| \leq \mathsf{negl}(\lambda).$$
where $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$, $\pi \leftarrow \mathsf{Prove}(\mathsf{crs}, \mathsf{stmt}, \mathsf{wit})$, $(\mathsf{crs}^*, \tau) \leftarrow \mathsf{Setup}^*(1^\lambda)$, $\pi^* \leftarrow \mathsf{Prove}^*(\mathsf{crs}^*, \tau, \mathsf{stmt})$ and τ is a trapdoor used by Sim to come up with proof π^* for a statement stmt without a witness.

A NIZK argument system has an **online extractor** [Fis05] if for any probabilistic polynomial-time (PPT) adversary $\mathcal{A}$, there exists a PPT algorithm Ext (the extractor) such that for all security parameters $1^\lambda \in \mathbb{N}$,
$$\Pr\left[(\mathsf{stmt}, \mathsf{wit}) \notin \mathcal{R} \middle| \begin{matrix}(\mathsf{stmt}, \pi) \leftarrow \mathcal{A}^{\mathsf{RO}}, \\ \mathsf{wit} \leftarrow \mathsf{Ext}(\mathsf{stmt}, \pi, \mathcal{Q}_{\mathsf{RO}}(\mathcal{A}))\end{matrix}\right] \leq \mathsf{negl}(\lambda),$$
where RO is a random oracle and $\mathcal{Q}_{\mathsf{RO}}(\mathcal{A})$ is the sequence of queries of $\mathcal{A}$ to RO and RO's answers.

A NIZK argument system $\mathsf{NIZK} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Vf})$ for relation $\mathcal{R}$ is *simulation-extractable* with respect to a zero-knowledge simulator $\mathsf{Sim} = (\mathsf{Setup}^*, \mathsf{Prove}^*)$ if for any PPT adversary $\mathcal{A}$ there exists a PPT extractor Ext and a negligible function negl such that for all security parameters $1^\lambda \in \mathbb{N}$,
$$\Pr\left[\begin{matrix}\mathsf{Vf}(\mathsf{crs}, \mathsf{stmt}, \pi) = 1 \wedge \\ (\mathsf{stmt}, \pi) \notin Q \wedge \\ (\mathsf{stmt}, \mathsf{wit}) \notin \mathcal{R}\end{matrix} \middle| \begin{matrix}(\mathsf{crs}, \tau) \leftarrow \mathsf{Setup}^*(1^\lambda) \\ (\mathsf{stmt}, \pi) \leftarrow \mathcal{A}^{\mathsf{SimProve}(\cdot)}(\mathsf{crs}) \\ \mathsf{wit} \leftarrow \mathsf{Ext}(\tau, \mathsf{stmt}, \pi)\end{matrix}\right] \leq \mathsf{negl}(\lambda),$$
where $\mathsf{SimProve}(\mathsf{stmt})$ returns a simulated proof $\pi^* \leftarrow \mathsf{Prove}^*(\mathsf{crs}, \tau, \mathsf{stmt})$ and appends (stmt, π^*) to the set Q of simulator-generated statement–proof pairs.

Standard NIZK argument systems are known from the factoring assumption [FLS90] and standard assumptions on bilinear pairings [CHK07, GOS06], and constructions with online extractors are known [Fis05].

A.7 Threshold Signature Schemes

Threshold signatures [Lin17, DOTT22] are a family of 't-out-of-n' signatures: a set of n parties run an interactive key generation protocol to produce a shared public key, and thereafter any subset of at least t parties can collaboratively sign messages under the shared key. Unforgeability guarantees that no coalition of up to $t-1$ malicious signers can produce a valid signature without the participation of honest parties:

Definition A.8. (Threshold Signature Scheme). A threshold signature scheme $\mathsf{TS} \coloneqq (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Sign}, \mathsf{Vf})$ consists of algorithms and protocols as follows:

$\underline{\mathsf{pp} \coloneqq \mathsf{Setup}(n, t)}$. *Takes number of signers n and signing threshold t, and outputs public parameters* pp. *From now on,* pp *is an implicit input to all subsequent algorithms.*

$\underline{(\mathsf{vk}, \{\mathsf{sk}_i\}_{i\in[n]}) \leftarrow \mathsf{KGen}(1^\lambda)}$. *Takes security parameter 1^λ and outputs a combined verification key* vk, *and a signing key share* sk_i *for each signer i.*

$\underline{\sigma \leftarrow \langle \mathsf{Sign}(\mathsf{sk}_i, m)\rangle}$. *The signing protocol is an interactive algorithm run by each signer concurrently. Each signer i runs* Sign_i *with their secret key share* sk_i *and message m. At the end of the*

protocol, a signature σ is output.

$0/1 := \mathsf{Vf}(\mathsf{vk}, m, \sigma)$. *Takes verification key* vk*, message* m*, and signature* σ *and outputs* $0/1$*, signifying whether* σ *was generated correctly.*

Definition A.9. (Correctness) A threshold signature scheme TS is correct if, for all security parameters λ, all parameters n, t with $t \leq n$, all messages $m \in \{0,1\}^*$, and any set $\mathcal{H}$ of at least t honest signers, the signing protocol followed by these t parties always produces a signature that verifies under the shared public key. Formally, if

$$(\mathsf{vk}, \{\mathsf{sk}_i\}_{i\in[n]}) \leftarrow \mathsf{TS.KGen}(\lambda), \quad \sigma \leftarrow \langle \mathsf{TS.Sign}(\{\mathsf{sk}_i\}_{i\in\mathcal{H}}, m)\rangle, \quad b \leftarrow \mathsf{TS.Vf}(\mathsf{vk}, m, \sigma),$$

then

$$\Pr[b = 1] = 1,$$

where the probability is taken over the randomness of KGen and the signing protocol.

Threshold cryptosystems are known from standard assumptions and signature schemes, including Schnorr [KG20, CKM23] and BLS [DR23]. Recent work has also demonstrated efficient lattice-based constructions [TPCZ23, PKM+24]

A.8 Correctness of Proxy Adaptor Signatures

Definition A.10. (Proxy Adaptor Signature Correctness). A proxy adaptor signature scheme PAS satisfies correctness, if for all $\lambda, t < n \in \mathbb{N}$, $m \in \{0,1\}^*$, and $(\mathsf{stmt}, \mathsf{wit}) \in \mathcal{R}_{\mathsf{DLog}}$:

$$\Pr\left[\begin{array}{l|l} \mathsf{ReqVerify}(\mathsf{stmt}, \mathsf{req}) = 1 \wedge & \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda) \\ \mathsf{AdVerify}(\mathsf{stmt}, \mathsf{advt}) = 1 \wedge & (\mathsf{vk}, \{\mathsf{sk}_i\}_{i\in[n]}) \leftarrow \mathsf{TS.KGen}(1^\lambda) \\ \mathsf{TS.Vf}(\mathsf{vk}, m, \sigma) = 1 \wedge & \mathsf{req} \leftarrow \mathsf{ReqGen}(\mathsf{stmt}) \\ (\mathsf{stmt}, \mathsf{wit}') \in \mathcal{R}_{\mathsf{DLog}} & (\mathsf{st}_{\mathsf{advt}}, \mathsf{advt}) \leftarrow \mathsf{AdGen}(\mathsf{stmt}, \mathsf{wit}) \\ & (\{\mathsf{st}^i_\tau\}_{i\in\mathcal{S}}, \tilde{\tau}, \mathcal{S}) \leftarrow \langle \mathsf{ProxGen}(\{\mathsf{sk}_i\}_{i\in\mathcal{S}}, \mathsf{vk}, \mathsf{req}, \mathsf{advt}, \mathsf{stmt})\rangle \\ & \tau \leftarrow \mathsf{Combine}(\mathsf{vk}, m, \mathsf{advt}, \mathsf{stmt}, \tilde{\tau}, \mathsf{req}) \\ & \sigma \leftarrow \mathsf{Adapt}(\mathsf{st}_{\mathsf{advt}}, \mathsf{wit}, \tau, \mathsf{vk}) \\ & \mathsf{wit}'_\tau \leftarrow \mathsf{ProxExt}(\mathsf{vk}, \tau, \sigma, \mathsf{stmt}) \\ & \mathsf{wit}' \leftarrow \mathsf{ReqExt}(\{(i, \mathsf{st}^i_\tau)\}_{i\in\mathcal{S}}, \mathsf{wit}'_\tau, \mathsf{vk}) \end{array}\right] = 1$$

B Proofs for Generic Proxy Adaptor Signature Construction

This section contains the formal proofs for the security properties of our proxy adaptor signature construction stated in Section 3.2.

B.1 Adaptability

Proof of Theorem 3.5. By our construction, if $\mathsf{ReqVerify}(\mathsf{stmt}, \mathsf{req}) = 1$, we know that $\mathsf{stmt} = \mathsf{req}$. Each NIZK proof π_i attests that c_i is a valid encryption of r_i under pk w.r.t. R_i. Furthermore, it can be publicly checked that the values R_i interpolate R using Shamir's secret sharing, if one knows at least t values r_i. We can show by the simulation-extractability of $\mathsf{NIZK}_{\mathsf{PKE}}$, that if $\mathcal{V} := \{i : i \in \mathcal{S}, \mathsf{NIZK}_{\mathsf{PKE}}.\mathsf{Vf}(\mathsf{crs}_{\mathsf{PKE}}, (g^{r_i}, c_i, \mathsf{pk}), \pi_i) = 1\}$ has size of at least t, then knowing pk, r can be recovered.

Hence, if $\mathsf{Combine}$ does not abort, we have that the decrypted r satisfies $g^r = R$, and so $(\mathsf{stmt} \cdot R, \mathsf{wit} + r) \in \mathcal{R}_{\mathsf{DLog}}$, and since

$$\tau \neq \bot \Rightarrow \mathsf{TAS.PreVerify}(\mathsf{vk}, m, \mathsf{stmt} \cdot R, \tilde{\sigma}) = 1.$$

Finally, we use the pre-signature adaptability of TAS, which implies that $\tilde{\sigma}$ can be adapted into a valid signature under vk, m using $\mathsf{wit} + r$. Therefore, we have by construction of Adapt that

$$\mathsf{TS.Vf}(\mathsf{vk}, m, \mathsf{Adapt}(\mathsf{st}_{\mathsf{advt}}, \mathsf{wit}, \tau, \mathsf{vk})) = 1.$$

□

B.2 Extractability

Proof of Theorem 3.6. We need to show that for every probabilistic polynomial-time (PPT) adversary $\mathcal{A}$, there exists a negligible function negl such that for all security parameters $\lambda \in \mathbb{N}$, for all $t < n \in \mathbb{N}$:

$$\Pr[\mathsf{paExt}^{\mathcal{A}}_{\mathsf{PE},t,n}(1^\lambda) = 1] \leq \mathsf{negl}(\lambda).$$

We proceed via a reduction to the extractability of the threshold adpator signature scheme.

Assume towards a contradiction that there exists a PPT adversary $\mathcal{A}$ such that $\Pr[\mathsf{paExt}^{\mathcal{A}}_{\mathsf{PE},t,n}(1^\lambda) = 1] > \epsilon$ for non-negligible ϵ. Then, $\mathcal{B}$ is formally defined in Figure 9. The reduction interacts with the challenger of the TAS extractability game $\mathsf{Ext}^{\mathcal{A}}_{\mathsf{TAS},t,n}$ by forwarding the corrupted index set chosen by the internal adversary to the challenger, and returning back to the challenger whatever message and signature the internal adversary outputs. Notice that $\mathcal{B}$ can run the extractability game perfectly for the adversary $\mathcal{A}$, except that it doesn't obtain all secret key shares of the threshold scheme. In cases where the game requires the secret key shares, $\mathcal{B}$ instead forwards internal oracle calls to the oracles provided by the challenger, which provide an indistinguishable view to the adversary, and hence the witness extractability game is fully simulated by $\mathcal{B}$.

When our construction runs a DKG instance, we observe that in our proof, we replace this interaction by $\mathcal{F}_{\mathsf{DKG}}$ (i.e., we perform the proof in the $\mathcal{F}_{\mathsf{DKG}}$ model [Kat24]). This means that instead of running the DKG protocol with the adversary, the challenger samples values $r_i \leftarrow\!\!\$\, \mathbb{Z}_p$ for all $i \in [n]$, computes $R_i = g^{r_i}$, and Shamir interpolates R from R_i. Then, the challenger forwards $(R, \vec{R})$ to the adversary that either continues or aborts. If the adversary aborts, the challenger quits simulating the oracle in which the DKG was used. Otherwise, the challenger forwards the r_i values of the malicious parties to the adversary. We note that if the adversary does not abort, the blinding R is uniformly distributed (unbiasability), and the shares r_i of the honest parties suffice to reconstruct a valid witness r for R (robustness). Since we only need to be able to reconstruct in this security game, and do not need to blind the witness from the adversary, simulating $\mathcal{F}_{\mathsf{DKG}}$ like this (without embedding a challenge) suffices.

Eventually, the adversary outputs its forgery. Due to the way we construct wit_τ using $\mathcal{F}_{\mathsf{DKG}}$, if σ^* is a valid forgery against extractability of the proxy adaptor signature, σ^* is also a valid forgery for breaking the extractability of the threshold adaptor signature scheme. This is true since, $\mathsf{wit}_\tau = \mathsf{wit} + r$, and r is well-formed, and known to the reduction. Therefore, we have

$$\mathsf{paExt}^{\mathcal{A}}_{\mathsf{PE},t,n}(1^\lambda) = 1 \Rightarrow \mathsf{Ext}^{\mathcal{B}}_{\mathsf{TAS},t,n}(1^\lambda) = 1.$$

Since we assume $\mathsf{paExt}^{\mathcal{A}}_{\mathsf{PE},t,n}$ is won with non-negligible probability ϵ, we have that $\mathsf{Ext}^{\mathcal{B}}_{\mathsf{TAS},t,n}$ is won with non-negligible probability ϵ, which breaks the extractability of TAS.

□



Figure 9: Extractability game for proxy adaptor signatures with reduction $\mathcal{B}$ to extractability of TAS

B.3 Witness Privacy

Proof of Theorem 3.7. We need to show that for every probabilistic polynomial-time (PPT) adversary $\mathcal{A}$, there exists a stateful PPT simulator $\mathsf{Sim} = (\mathsf{ReqGen}^*, \mathsf{AdGen}^*, \mathsf{ProxGen}^*, \mathsf{Adapt}^*)$ and a negligible function negl such that for all security parameters $\lambda \in \mathbb{N}$, for all $(\mathsf{stmt}, \mathsf{wit}) \in \mathcal{R}_{\mathsf{DLog}}$, for all wit_τ such that there exists $\mathsf{st}_{\mathsf{req}}, \mathsf{vk}$ with $\mathsf{wit} = \mathsf{ReqExt}(\mathsf{st}_{\mathsf{req}}, \mathsf{wit}_\tau, \mathsf{vk})$, and for all PPT distinguishers D:

$$\left|\Pr[\mathsf{D}(\mathsf{paZKReal}^{\mathcal{A}}_{\mathsf{PAS},t,n}(1^\lambda, \mathsf{stmt}, \mathsf{wit})) = 1] - \Pr[\mathsf{D}(\mathsf{paZKIdeal}^{\mathcal{A}}_{\mathsf{PAS},t,n}(1^\lambda, \mathsf{stmt})) = 1]\right| \leq \mathsf{negl}(\lambda).$$

To prove witness privacy, we introduce a series of hybrid experiments that incrementally convert from the real-world experiment to a simulated ideal experiment. Again we do this in the $\mathcal{F}_{\mathsf{DKG}}$ model [Kat24], which is feasible since we assume that the DKG t-securely realizes $\mathcal{F}_{\mathsf{DKG}}$.

Hybrid $\mathcal{H}_0$: This is the same as the experiment in Figure 10.

Hybrid $\mathcal{H}_1$: This is the same as $\mathcal{H}_0$ except that $\mathsf{NIZK}_{\mathsf{DLog}}$ and $\mathsf{NIZK}_{\mathsf{PKE}}$ are replaced with their respective zero-knowledge simulators.

Hybrid $\mathcal{H}_2$: This is the same as $\mathcal{H}_1$ except that the honest parties encrypt 0 instead of shares of r, and use local state for correctness.

Hybrid $\mathcal{H}_3$: This is the same as $\mathcal{H}_2$ except that the simulator extracts the encrypted value r'_i from all malicious parties and aborts, if this value does not equal the value r_i output by $\mathcal{F}_{\mathsf{DKG}}$.

Hybrid $\mathcal{H}_4$: This is the same as $\mathcal{H}_3$, except that the challenger provides the adversary access to $\mathcal{F}_{\mathsf{DKG}}$, and it simulates its behavior based on a uniformly random group element R.

Hybrid $\mathcal{H}_5$: This is the same as $\mathcal{H}_4$ except regarding the generation of R. Instead of invoking $\mathcal{F}_{\mathsf{DKG}}$ to generate a random R, the challenger samples $\mathsf{wit}_\tau \leftarrow\!\$\, \mathbb{Z}_q$ and computes $R \coloneqq g^{\mathsf{wit}_\tau} \cdot \mathsf{stmt}^{-1} = g^{\mathsf{wit}+r-\mathsf{wit}} = g^r$ (recall from Section 3.1 that stmt is in a discrete-log NP relation, hence stmt is a group element and so we can compute stmt^{-1}). In $\mathcal{O}_{\mathsf{Adapt}}$, if $\mathsf{TS.Vf}(\mathsf{vk}, m, \sigma) = 0$, then the oracle aborts. With this change, $\mathcal{H}_4$ can be simulated without knowing the witness.

We proceed to argue the indistinguishability of each hybrid experiment from its predecessor.

$\mathcal{H}_0 \overset{\mathrm{c}}{\approx} \mathcal{H}_1$: The difference between $\mathcal{H}_0$ and $\mathcal{H}_1$ is just the way honest proofs are computed. We replace all honestly computed proofs one-by-one (first the advertisement proof, then each honest proxy's encryption-correctness proof produced in each call to $\mathcal{O}_{\mathsf{ProxGen}}$). Each replacement changes the adversary's view by at most a negligible amount by the zero-knowledge of the corresponding NIZK system (Theorem A.7), and the number of replaced proofs is polynomial in 1^λ. Therefore, using the standard hybrid argument, $\mathcal{H}_0 \overset{\mathrm{c}}{\approx} \mathcal{H}_1$.

$\mathcal{H}_1 \overset{\mathrm{c}}{\approx} \mathcal{H}_2$: Suppose towards a contradiction that there exists a distinguisher D and a non-negligible ϵ such that

$$|\Pr[\mathsf{D}(\mathsf{view}_{\mathcal{A}}(\mathcal{H}_1)) = 1] - \Pr[\mathsf{D}(\mathsf{view}_{\mathcal{A}}(\mathcal{H}_2)) = 1]| = \epsilon.$$

Note that the view of $\mathcal{A}$ differs only in that it receives $\mathsf{PKE.Enc}(\mathsf{pk}, 0)$ instead of $\mathsf{PKE.Enc}(\mathsf{pk}, r_i)$ from each honest proxy i. We show that this gap is bounded by a standard hybrid argument over the CPA security of PKE. We just consider the gap between two hybrids: We build a reduction $\mathcal{B}$ which breaks the CPA security of PKE. The reduction works by running the experiment as normal, except that instead of honest proxies sending $\mathsf{PKE.Enc}(\mathsf{pk}, r_i)$. In this case, the reduction sends $\mathsf{PKE.Enc}(\mathsf{pk}, r_i)$ for the first $\ell - 1$ proxies, and $\mathsf{PKE.Enc}(\mathsf{pk}, 0)$ for all other proxies except for proxy ℓ. For the honest proxy ℓ, it sends $(r_\ell, 0)$ to the challenger as the choice of plaintexts. After receiving a challenge ciphertext c^*, the reduction continues as normal, returning $\mathsf{D}(\mathsf{view}_{\mathcal{A}})$ to the challenger. Note that if r_ℓ is chosen for the ciphertext, the view of $\mathcal{A}$ is the same as in the previous hybrid. Similarly, if 0 is chosen, the view of $\mathcal{A}$ is the same as in the latter hybrid. The two extreme hybrids of our hybrid argument equal the experiments $\mathcal{H}_4$ and $\mathcal{H}_5$. Hence, if D has non-negligible advantage ϵ in distinguishing between $\mathcal{H}_4$ and $\mathcal{H}_5$, then $\mathcal{B}$ has non-negligible advantage against the challenger for the IND-CPA game. Hence, $\mathcal{B}$ breaks the CPA security of PKE.

$\underline{\mathcal{H}_2 \overset{c}{\approx} \mathcal{H}_3}$: The only difference between $\mathcal{H}_2$ and $\mathcal{H}_3$ appears if the extraction of the value r_i' does not equal the value r_i simulated for malicious user i. We bound the probability that this happens via a reduction to the simulation-extractability of $\mathsf{NIZK}_{\mathsf{PKE}}$. As input, the reduction receives the public parameters of NIZK and simulates all oracles as in $\mathcal{H}_2$. When an abort happens in $\mathcal{H}_3$, the reduction returns the ciphertext, R_i, and the proof. This reduction breaks simulation extractability, since $r_i' \neq r_i$ is no valid witness for R_i. Additionally, the proof was generated by the adversary. Again, this generalizes using the standard hybrid argument to all proofs output by the adversary.

$\underline{\mathcal{H}_3 \overset{c}{\approx} \mathcal{H}_4}$: The expected output of $\mathcal{F}_{\mathsf{DKG}}$ is outputting a uniformly random R, partial commitments R_i and shares r_i. To simulate this, our reduction does the following: First, it also samples $R \leftarrow\!\$\, \mathbb{G}$, and $t-1$ values r_i, such that the index i corresponds to at least all malicious parties. (If fewer than $t-1$ parties are corrupted, the reduction uses the honest parties' indices as well, until $t-1$ values are assigned). Then, the reduction computes $R_i = g^{r_i}$ for these values, and interpolates all remaining R_j values using the set of R_i and R. It sends r_i to the adversary for all corrupted i. The distribution of this simulation is identical to the distribution of $\mathcal{F}_{\mathsf{DKG}}$. In addition, the challenger can simulate all oracles, since in the situation of $\mathcal{H}_3$, the value r_j for honest parties does not need to be known to the challenger in order to simulate. The hybrids $\mathcal{H}_3$ and $\mathcal{H}_4$ are statistically close, so the claim holds.

$\underline{\mathcal{H}_4 \overset{c}{\approx} \mathcal{H}_5}$: In $\mathcal{H}_4$, $R = g^r$ where $r \leftarrow\!\$\, \mathbb{Z}_q$ is chosen uniformly at random. In $\mathcal{H}_5$, $R = g^{\mathsf{wit}_\tau \cdot \mathsf{stmt}^{-1}} = g^{\mathsf{wit}_\tau - \mathsf{wit}}$. Since wit_τ is sampled uniformly from $\mathbb{Z}_q$, the value $(\mathsf{wit}_\tau - \mathsf{wit})$ is also uniformly distributed in $\mathbb{Z}_q$. Thus, R is uniformly distributed in the group $\mathbb{G}$ in both experiments, and this change cannot be detected. Aside from this, the hybrids only differ if the challenger aborts, as the adapted signature does not verify. Assume towards a contradiction that

$$|\Pr[\mathcal{H}_4 \text{ aborts}] - \Pr[\mathcal{H}_5 \text{ aborts}]| = \epsilon$$

for some non-negligible ϵ. Observe that the only functional difference causing an abort in $\mathcal{H}_5$ is the check if $\mathsf{TS.Vf}(\mathsf{vk}, m, \sigma) = 0$. We hence have

$$\Pr[\mathsf{TS.Vf}(\mathsf{vk}, m, \sigma) = 0 \mid \mathsf{TAS.PreVerify}(\mathsf{vk}, m, \mathsf{stmt} \cdot R, \tilde{\sigma}) = 1] = \epsilon$$

Since it is given that $\mathsf{stmt} \in L_{\mathcal{R}_{\mathsf{DLog}}}$, this implies that a valid pre-signature failed to adapt into a valid signature, which breaks the pre-signature adaptability of TAS.

$\mathcal{H}_0$ and $\mathcal{H}_5$ are formally defined in Figure 10.

□

C Fairness Model

This section formalizes the system model, the involved parties, and the notion of fairness for proxy adaptor signatures.

Relation and assets. Let $\mathbb{G}$ be a cyclic group of prime order q generated by g. We define the discrete logarithm relation

$$\mathcal{R}_{\mathsf{DLog}} \coloneqq \{(X, x) \mid X \in \mathbb{G},\ x \in \mathbb{Z}_q,\ \text{and } X = g^x\}.$$

A statement stmt is a group element $X \in \mathbb{G}$. A witness wit is a scalar $x \in \mathbb{Z}_q$ such that $X = g^x$. The witness x is the secret asset owned by the seller. The payment asset is a valid threshold signature on a message m under the shared verification key vk generated by the threshold signature scheme. In practice, the message m encodes a blockchain transaction that authorizes a transfer of funds from the threshold-controlled account associated with verification key vk to the seller. The validity of the transaction is enforced by the ledger through verification of the threshold signature under vk.

Parties and delegation. The system consists of a seller S, a buyer B, and a set of n proxy signers $P = \{\mathcal{S}_1, \ldots, \mathcal{S}_n\}$. The seller S holds a witness $\mathsf{wit} \in \mathbb{Z}_q$ for a public statement $\mathsf{stmt} \in \mathbb{G}$ such that $(\mathsf{stmt}, \mathsf{wit}) \in \mathcal{R}_{\mathsf{DLog}}$. The proxies jointly hold secret key shares $\{\mathsf{sk}_i\}$ of a threshold signing key generated by $\mathsf{TS.KGen}$ with threshold t and corresponding verification key vk. The buyer B does not hold any signing key material, but is capable of authenticating to the proxies.

Delegation is modeled as follows. Prior to executing any exchange, the buyer B transfers the payment funds to an account controlled by the proxies via the threshold verification key vk. After this transfer, the proxies control the payment asset on behalf of the buyer. When the buyer wishes to purchase an asset, it issues a public request object $\mathsf{req} \leftarrow \mathsf{ReqGen}(\mathsf{stmt})$ specifying the statement stmt. The request authorizes the proxies to identify a matching seller S and to execute a proxy adaptor signature protocol with S. Any subset $\mathcal{S} \subseteq P$ with $|\mathcal{S}| \geq t$ may participate in the protocol. The buyer does not participate in the cryptographic execution and may remain offline until returning to obtain the witness via the extraction procedure. After transferring the funds to the proxies, the buyer's only required state is the authorization secret needed to authenticate to the proxies and to obtain the proxy secret states required for witness extraction.

Execution model. The seller publishes an advertisement advt generated from $(\mathsf{stmt}, \mathsf{wit})$. The proxies interactively execute $\mathsf{ProxGen}$ on input $(\mathsf{vk}, \mathsf{req}, \mathsf{advt}, \mathsf{stmt}, m)$ and produce a proxy output τ. The seller may adapt τ using wit to obtain a valid threshold signature σ on m. Given $(\mathsf{vk}, \tau, \sigma, \mathsf{stmt})$, the proxies derive a blinded witness $\mathsf{wit}_\tau \leftarrow \mathsf{ProxExt}(\mathsf{vk}, \tau, \sigma, \mathsf{stmt})$. When the buyer comes online again and successfully authenticates with the proxies, each proxy $\mathcal{S}_i$ sends its secret state st_τ^i and the blinded witness wit_τ to the buyer. Using the blinded witness and the secret proxy states from any subset of at least t proxies, the buyer locally recovers the witness $\mathsf{wit} \leftarrow \mathsf{ReqExt}(\{(i, \mathsf{st}_\tau^i)\}_{i \in \mathcal{S}}, \mathsf{wit}_\tau, \mathsf{vk})$.

Fair exchange. Fairness requires that payment and witness revelation are cryptographically linked and occur atomically. We follow the definitions of [GSST24].

Definition C.1. (Fair exchange). A proxy adaptor signature scheme satisfies fair exchange if the following properties hold for all probabilistic polynomial-time adversaries.

1. *Seller fairness.* For any adversary that may corrupt the buyer and all proxies, if the adversary obtains a witness wit such that $(\mathsf{stmt}, \mathsf{wit}) \in \mathcal{R}_{\mathsf{DLog}}$, then there exists a valid threshold signature σ on the message m under the verification key vk that can be produced by the seller.

2. *Buyer fairness.* For any adversary that may corrupt the seller and up to $t - 1$ proxies with $2 \cdot t \geq n$, a valid threshold signature σ on the message m under vk can be produced only if an honest buyer is able to recover a witness wit such that $(\mathsf{stmt}, \mathsf{wit}) \in \mathcal{R}_{\mathsf{DLog}}$.

3. *Witness privacy.* For any adversary that may corrupt up to $t-1$ proxies while the seller is honest, the adversary's view of the protocol execution is simulatable without knowledge of the witness wit. In particular, no proxy learns any information about wit beyond what is implied by the public statement stmt, and the witness is revealed only to the buyer.

Theorem C.2. (PAS achieves fair exchange). Let PAS be a proxy adaptor signature construction that satisfies *adaptability*, *extractability*, and *witness privacy* as defined in Section 3, then PAS satisfies fair exchange according to Theorem C.1.

We prove each item of Theorem C.1 via a dedicated lemma. Combining the three lemmas yields Theorem C.2.

Lemma C.3. (Seller fairness from adaptability and witness extractability). If PAS satisfies adaptability and witness privacy, then it satisfies seller fairness.

Proof. Fix any probabilistic polynomial-time adversary that corrupts the buyer and all proxies. We show that if the adversary outputs a witness wit such that $(\mathsf{stmt}, \mathsf{wit}) \in \mathcal{R}_{\mathsf{DLog}}$, then there exists a valid threshold signature σ on m under vk that can be produced by the seller. By the PAS interfaces, the seller's input witness wit is used only in $\mathsf{AdGen}(\mathsf{stmt}, \mathsf{wit})$ and in $\mathsf{Adapt}(\mathsf{st}_{\mathsf{advt}}, \mathsf{wit}, \tau, \mathsf{vk})$. All other algorithms executed by the adversary and the proxies take as input only public values and proxy secret states. By witness privacy, the advertisement advt output by AdGen can be simulated without knowledge of wit. Therefore, advt does not provide the adversary with information sufficient to compute a valid witness for stmt. In addition, by adaptability, the output of $\mathsf{Adapt}(\mathsf{st}_{\mathsf{advt}}, \mathsf{wit}, \tau, \mathsf{vk})$ yields a valid signature to the seller, and this establishes seller fairness. □

Lemma C.4. (Buyer fairness from extractability). If PAS satisfies extractability, then it satisfies buyer fairness.

Proof. Consider any adversary that corrupts the seller and up to $t-1$ proxies. Suppose a valid threshold signature σ on m under vk is produced. By extractability, such a valid signature is either signed by the proxies or can be used to extract a valid witness. Since extractability requires that $2 \cdot t \geq n$, even if up to $t-1$ proxies are corrupted, there exists a threshold set containing sufficiently many honest proxies to provide the required secret proxy states to the buyer. Therefore, a valid payment signature can be produced only if an honest buyer is able to recover a valid witness, thereby proving buyer fairness. □

Lemma C.5. (Witness privacy). If PAS satisfies witness privacy, then it satisfies the witness privacy condition of Theorem C.1.

Proof. This follows directly from the witness privacy definition of PAS by instantiating the adversary with any coalition of at most $t-1$ proxies and taking the simulator promised by the definition. Thus, the coalition learns no information about wit beyond what is implied by stmt, and the witness is revealed only to the buyer through ReqExt. □

Experiments $\mathsf{paZKReal}^{\mathcal{A}}_{\mathsf{PAS},t,n}(1^\lambda, \mathsf{stmt}, \mathsf{wit})$, $\boxed{\mathsf{paZKIdeal}^{\mathcal{A},\mathsf{NIZK}_{\mathsf{DLog}}.\mathsf{Sim},\mathsf{DKG}.\mathsf{Sim}}_{\mathsf{PAS},t,n}(1^\lambda, \mathsf{stmt})}$

$(\mathsf{vk}, \{\mathsf{sk}_i\}_{i\in[n]}) \leftarrow \mathsf{TS.KGen}(1^\lambda)$

$\mathsf{crs}_{\mathsf{PKE}} \leftarrow \mathsf{NIZK}_{\mathsf{PKE}}.\mathsf{Setup}(1^\lambda)$ $\boxed{(\mathsf{crs}_{\mathsf{PKE}}, \tau_{\mathsf{PKE}}) \leftarrow \mathsf{NIZK}_{\mathsf{PKE}}.\mathsf{Setup}^*(1^\lambda)}$

$\mathsf{crs}_{\mathsf{DLog}} \leftarrow \mathsf{NIZK}_{\mathsf{DLog}}.\mathsf{Setup}(1^\lambda)$ $\boxed{(\mathsf{crs}_{\mathsf{DLog}}, \tau) \leftarrow \mathsf{NIZK}_{\mathsf{DLog}}.\mathsf{Setup}^*(1^\lambda)}$

$\mathsf{req} \leftarrow \bot$

$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{PKE.KGen}(1^\lambda)$ $\boxed{(\mathsf{pk}, _) \leftarrow \mathsf{PKE.KGen}(1^\lambda)}$

$\pi_{\mathsf{advt}} \leftarrow \mathsf{NIZK}_{\mathsf{DLog}}.\mathsf{Prove}(\mathsf{crs}_{\mathsf{DLog}}, (\mathsf{stmt}, \mathsf{pk}), \mathsf{wit})$ $\boxed{\pi_{\mathsf{advt}} \leftarrow \mathsf{NIZK}_{\mathsf{DLog}}.\mathsf{Prove}^*(\mathsf{crs}_{\mathsf{DLog}}, \tau, (\mathsf{stmt}, \mathsf{pk}))}$

$\mathsf{advt} := (\mathsf{pk}, \pi_{\mathsf{advt}})$

$\mathsf{corr}, \mathcal{Q}_\tau := \emptyset$

$\mathbb{O} := (\mathcal{O}_{\mathsf{ProxGen}}(\cdot), \mathcal{O}_{\mathsf{Adapt}}(\cdot,\cdot))$

$\boxed{\mathbb{O} := (\mathcal{O}_{\mathsf{ProxGen}}{}^*(\cdot), \mathcal{O}_{\mathsf{Adapt}}{}^*(\cdot,\cdot,\cdot))}$

$\mathsf{corr} \leftarrow \mathcal{A}^{\mathbb{O}}(\mathsf{vk}, \{\mathsf{sk}_i\}_{i\in[n]}, \mathsf{req}, \mathsf{advt}, \mathsf{stmt})$

return view of $\mathcal{A}$

Oracle $\mathcal{O}_{\mathsf{ProxGen}}(m)$, $\boxed{\mathcal{O}_{\mathsf{ProxGen}}{}^*(m)}$

if $|\mathsf{corr}| \geq t$: **return** $\bot$

$\mathcal{H} := [n] \setminus \mathsf{corr}$

$R_{\mathcal{H}} \leftarrow \{\text{random coins}\}$

$(R, \{R_1, \dots R_n\}) \leftarrow \mathcal{F}_{\mathsf{DKG}}$

$\boxed{\mathsf{wit}_\tau \leftarrow\$ \mathbb{Z}_q, R := g^{\mathsf{wit}_\tau} \cdot \mathsf{stmt}^{-1}}$

$\boxed{\{R_1, \dots R_n\}, \{r_i\}_{i\in\mathsf{corr}} \leftarrow \mathcal{S}_{\mathsf{DKG}}(R)}$

$\mathsf{msg} \leftarrow \mathcal{A}(R, \{R_1, \dots R_n\})$

if $\mathsf{msg} == \mathsf{abort}$ **then return** $\bot$

$\{r_1, \dots r_n\} \leftarrow \mathcal{F}_{\mathsf{DKG}}(\mathsf{continue})$

// Interactively run all $\mathsf{ProxGen}_i$, $\mathcal{A}$ receives malicious r_i before.

$(\tilde{\tau}, \mathcal{S}) \leftarrow \langle \mathsf{ProxGen}(\mathsf{sk}_i, \mathsf{vk}, \mathsf{req}, \mathsf{advt}, \mathsf{stmt}, m; R_H), \mathcal{A}\rangle$

$\boxed{(\tilde{\tau}, \mathcal{S}) \leftarrow \langle \mathsf{ProxGen}^*(\mathsf{sk}_i, \mathsf{vk}, \mathsf{req}, \mathsf{advt}, \mathsf{stmt}, m, R, \vec{R}; R_H), \mathcal{A}\rangle}$

parse $(R, \{(c_i, \pi_i, g^{r_i}, \tilde{\sigma}_i)\}_{i\in\mathcal{S}}) = \tilde{\tau}$

$\mathcal{V} := \{i : i \in \mathcal{S}, \mathsf{NIZK}_{\mathsf{PKE}}.\mathsf{Vf}(\mathsf{crs}_{\mathsf{PKE}}, (g^{r_i}, c_i, \mathsf{pk}), \pi_i) = 1\}$

if $|\mathcal{V}| < t$: **return** $\bot$

for $i \in \mathcal{V} : \lambda_i := \prod_{j\in\mathcal{V}, j\neq i} \frac{j}{j-i}$

$c := \prod_{i\in\mathcal{V}} c_i^{\lambda_i}$

$\tau := (R, c, \mathsf{TAS.Combine}(\{\tilde{\sigma}_i\}_{i\in\mathcal{V}}))$

$\boxed{\mathcal{Q}[\tau] := \mathsf{wit}_\tau}$

return τ

Oracle $\mathcal{O}_{\mathsf{Adapt}}(m, \tau)$

parse $(R, c, \tilde{\sigma}) = \tau$

$r := \mathsf{PKE.Dec}(\mathsf{sk}, c)$

if $g^r \neq R \vee \mathsf{TAS.PreVerify}(\mathsf{vk}, m, \mathsf{stmt} \cdot R, \tilde{\sigma}) = 0$: **return** $\bot$

return $\mathsf{TAS.Adapt}(\mathsf{vk}, \tilde{\sigma}, \mathsf{wit} + r)$

$\mathsf{ProxGen}_i(\mathsf{sk}_i, \mathsf{vk}, \mathsf{req}, \mathsf{advt}, \mathsf{stmt}, m)$

$\boxed{\mathsf{ProxGen}^*_i(\mathsf{sk}_i, \mathsf{vk}, \mathsf{req}, \mathsf{advt}, \mathsf{stmt}, m, R, \vec{R})}$

parse $(\mathsf{pk}, _) = \mathsf{advt}$

$c_i \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, r_i)$ $\boxed{c_i \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, 0)}$

$\pi_i \leftarrow \mathsf{NIZK}_{\mathsf{PKE}}.\mathsf{Prove}(\mathsf{crs}_{\mathsf{PKE}}, (g^{r_i}, c_i, \mathsf{pk}), r_i)$

$\boxed{\pi_i \leftarrow \mathsf{NIZK}_{\mathsf{PKE}}.\mathsf{Prove}^*(\mathsf{crs}_{\mathsf{PKE}}, \tau_{\mathsf{PKE}}, (R_i, c_i, \mathsf{pk}))}$

$\tilde{\sigma}_i \leftarrow \mathsf{TAS.PreSign}_i(\mathsf{sk}_i, \mathsf{vk}, m, \mathsf{stmt} \cdot R)$

$\tilde{\tau}_i := (c_i, \pi_i, g^{r_i}, \tilde{\sigma}_i)$

return $(r_i, \tilde{\tau}_i)$

$\boxed{\mathcal{O}_{\mathsf{Adapt}}{}^*(m, \tau)}$

parse $(R, _, \tilde{\sigma}) = \tau$

$\mathsf{wit}_\tau := \mathcal{Q}_\tau[\tau]$

if $\mathsf{TAS.PreVerify}(\mathsf{vk}, m, \mathsf{stmt} \cdot R, \tilde{\sigma}) = 0$: **return** $\bot$

$\sigma := \mathsf{TAS.Adapt}(\mathsf{vk}, \tilde{\sigma}, \mathsf{wit}_\tau)$

if $\mathsf{TS.Vf}(\mathsf{vk}, m, \sigma) = 0$: **return** $\bot$

return σ

Figure 10: Zero knowledge security of proxy adaptor signature construction