Paras - A Private NFT Protocol

Vanishree Rao*

Abstract

Non-fungible tokens (NFTs) are a blockchain application that has recently witnessed significant success. However, NFT marketplaces are majorly built on popular blockchain platforms that do not provide privacy tools. As a result, NFTs are easily visible to everyone. This has naturally given rise to various issues, including stolen/duplicate NFTs and attacks like shill trading. Furthermore, this architecture fails to reflect the real-life privacy notion as it digitizes unique physical goods.

In this project, we build *Paras* – a blockchain-agnostic protocol that offers privacy to NFTs. Specifically, one may hide the real NFTs and only display a reference to them on marketplaces, hide seller and bidder identities, hide bid values and user wallet balances.

Paras is based on cryptographic primitives, such as, threshold encryption and robust secret sharing. It does not rely on any trusted execution environments for security, unlike some existing protocols in this direction.

1 Introduction

Satoshi Nakamoto explained her vision for private transactions in [1]. Privacy is crucial not only in financial transactions (i.e., privacy of wallet balances, amounts transferred in transactions and user anonymity) but also in more complex financial DApps, as they find wider usage and involve large-value assets. One such application is of non-fungible tokens (NFTs).

1.1 The Need for Privacy in NFTs

NFTs are unique virtual or physical assets that are stored on a blockchain network, which provides information regarding their ownership. It is a fast-growing market with sales amounting to tens of billions of dollars per year [2].

*hrao.vanishree@gmail.com

In order to capture the audience excitement, NFT marketplaces are mostly built on popular blockchains, such as Ethereum and Solana. However, these blockchain platforms are not privacy enabling; specifically, user identities, bids, transaction details, etc. are completely visible to the general public. As a result, successful NFT categories are limited to those that do not need privacy, such as, simple images. This leaves out a plethora of meaningful and useful categories that require privacy.

1.1.1 More Meaningful NFT Categories Require Privacy

Below are examples of NFT categories that are more complex and useful and that need privacy.

Exclusive viewership. Art ownership in the non-digital world features rarity and exclusivity. Contrary to popular belief, not all art masterworks are held in museums and public galleries. Many of these are being kept in private art collections where only a selected few are able to observe them. To replicate exclusivity in the digital world, the privacy feature for NFTs is necessary.

Viewership with access control. NFTs where exclusivity is maintained through some access control, such as, previous interest/engagement, age limit and paywall, require privacy to enforce the access control policies.

Secret NFTs with sneak peaks. NFT artists can publish only sneak peaks to their protected content which will be revealed only to the buyers. A popular example is of Quentin Tarantino, the creator of the Pulp Fiction, a cult movie, who wanted to sell never-seen-before scenes of the movie in a protected manner by only revealing sneak peaks of the video to the general public and the exclusive content to only the buyer [3].

Sealed bid auctions. Sealed bid auctions are a popular form of auctions where no bidder knows how much bid the other auction participants have made. This method helps prevent malicious behaviors such as front-running attacks. The usefulness of this type of auction holds true for NFT auctions as well. Privacy is inherently necessary to enable sealed bid auctions.

Seller's asset value protection. Keeping the bids private helps ensure that if all bids are too low for any of them to be accepted by the seller,

the property will not become stigmatized by having a perceived low value in the marketplace.

Real estate and luxury goods. As NFTs replicate the real-world ownerships of real-world assets, especially those that involve personal privacy-sensitive data, their success depends on the availability of privacy options that exist in their real-world counterparts.

Event ticketing. NFTs are being used as tickets to events like concerts [4, 5]. User anonymity is necessary to preserve privacy of information such as users' physical whereabouts.

1.1.2 Generic Attacks in Marketplaces without Privacy

The fast growth of the NFT industry without the right technological advancements has opened the door to rampant piracy and fraud. While the above categories of use cases require privacy by design, privacy is arguably also crucial to avert many attacks.

Below are some important attack categories that can be effectively thwarted with privacy tools.

Shill bidding/wash trading. Shill bidding refers to a malicious behavior by an NFT seller who gets an accomplice to place high bids, in an attempt to artificially inflate others' bids. (If the shill wins the bid, she could later pay back the funds to the owner under the table.) Shill bidding is believed to occur pervasively across marketplaces even for high-profile NFT projects [6, 7]. Protecting privacy of bidding values can help fight against such malicious activities.

A similar attack is of wash trading, wherein a malicious seller sells NFTs to another wallet funded by herself, in an attempt to artificially inflate perception of the asset's value. Such malicious activities are rampant, as per some recent analysis [8, 9]. In a marketplace where privacy-powered trades are the default, it is conceivable that wash trading which do not employ privacy options are easier to tell apart.

Stealing. In traditional NFT marketplaces, it's not just the artist or the original seller who might have a copy of the underlying digital asset. Most such assets are displayed to the public on NFT marketplaces, so anyone can take a screenshot or possibly make a copy. That is, ownership of an NFT does not prevent a third party from making a copy of a digital asset

that is linked to the NFT *if they have access to it*. Stolen NFTs are indeed rife in NFT marketplaces [10, 11]. Privatizing NFTs and displaying only a reference, such as a thumbnail, can help effectively curb this menace.

1.2 Our Contribution

We design *Paras*, a blockchain-agnostic protocol that offers privacy to NFTs. Specifically, one may hide the real NFTs and only display a reference to them on marketplaces, hide seller and bidder identities, hide bid values and user wallet balances.

1.3 Related Work

A blockchain platform that is trying to solve the issue of privatizing NFTs is called Secret Network [12]. The main thesis of this protocol is to share all secrets with validator nodes but the secret will be stored inside trusted execution environments (TEEs). Specifically, the consensus seed is stored inside the TEE of each validator node, allowing for encrypted inputs to be fully decrypted and computed upon.

The security of the entire protocol mainly relies on the assumption that even when secret values are computed upon inside TEEs, the TEEs are able to keep the secrets completely secure even from the party that is running the TEEs. However, it is a well-known fact from deep and wide research that TEEs are subject to relatively easy attacks [13]. Given the relatively weak level of security of TEEs (as opposed to security from cryptographic protocols), it is not prudent to transact extremely high-value assets by relying on easily breakable security mechanisms.

2 Technical Introduction

2.1 The Main Technical Challenge

In pursuit of privacy for NFTs, let's for now focus on just the aspect of hiding NFTs from everyone except the seller and the winning bidder. In essence, the seller needs to encrypt her NFT and include the secret key in some hidden way in the blockchain state, so that, upon completion of the bidding phase, the secret key is revealed to just the winning bidder.

It appears that we are looking to solve the following notorious problem:

"How to usefully hide a secret value in a smart contract?"

Specifying usefulness of the secret value is important, because, otherwise, one can simply encrypt/commit to that secret and place it in the smart contract whereby the secret key turns useless.

Given the complexity of this question, despite some active research [14, 15], the solutions are far from ideal.

2.2 Why zk-SNARKs Don't Suffice?

Recall that the power of zk-SNARKs [16] is to prove correctness of some computation while hiding some selected parts of it. On the other hand, as discussed in Section 2.1, the need of the hour is completely different; we need to be able to hide a secret, without knowing ahead of time to whom it will be revealed. Hence, zk-SNARKs won't fit the puzzle. As a result, zk-SNARK-powered blockchains, such as Mina [17], Aleo [18] and Zcash [19], are not equipped enough to solve the problem.

2.3 Why Other Primitives such as Witness Encryption Don't Suffice?

On the same lines, one needs to check if other seemingly relevant primitives, such as, witness encryption [20] or fully homomorphic encryption [21], would be useful for our purposes. We will consider these two primitives in some detail.

Witness encryption. Recall that witness encryption allows one to encrypt a secret to a statement (instead of a public key), so that, any party who has a witness to that statement can decrypt the ciphertext. Witness encryption is a natural candidate for the following reason. Consider witness encrypting an NFT so that the witness to open it is the randomness knowledge of the bidding transaction with the highest bidding value. However, challenges arise because of the following reasons.

- The number of bids is not fixed ahead of time. So it is not clear how to prevent an adversarial bidder from holding back all bids that are higher than her own, while performing witness decryption.
- Even if we somehow restrict the number of bids, an adversarial bidder can simply make up a few transactions with lower bids and hold back the transactions with higher bids.
- Furthermore, one might argue for the witness encryption approach that even if someone adversarially learns the secret key that decrypts

the NFT, they may still not own the NFT per the blockchain state. However, this approach does not take us further in the goal of privatizing NFTs.

Fully Homomorphic Encryption (FHE). Recall that the power of fully homomorphic encryption lies in performing computations "under the hood"; i.e., in order to compute on encrypted data, one doesn't need to decrypt the encrypted data before performing computation on it. However, the output of the computation remains encrypted. On the other hand, we are looking for special key management techniques whereby the output is decrypted only to some party, namely the winning bidder, whose identity is unknown at the time of encryption. Clearly, this gadget too doesn't fit the puzzle.

3 Preliminaries

3.1 Basic Notations

We write PPT for probabilistic polynomial-time. Any algorithm discussed here will implicitly take the security parameter 1^λ as an input unless mentioned otherwise.

In the contexts of secret sharing, we use Σ to denote the alphabet. Readers can simply regard Σ as a prime-ordered finite field $\mathbb{F}$. For any sequence $s = (s_1, s_2, \ldots, s_n) \in \Sigma$ and sequence of indices $W = (\mathsf{w}_1, \ldots, \mathsf{w}_t) \in [n]^t$ with $t \leq n$, let s_W be the sub-sequence $(s_{\mathsf{w}_1}, s_{\mathsf{w}_2}, ..., s_{\mathsf{w}_t})$.

Unless specified otherwise, we will use superscripts to specify the type of a variable and subscripts to specify the entity the variable belongs to. For example, $\mathtt{pk}_i^{\mathsf{kp}}$ denotes that this public key variable is of 'key-private' type (a notion that will be discussed later) and belongs to some i-th party.

We will denote encryption of a plaintext v with public key $\mathtt{pk}$ as $[\![v]\!]_{\mathtt{pk}}$. Note that we will use different types of encryption such as key private encryption and threshold encryption; to clarify the type of encryption, we will write the type in the superscript as in $[\![v]\!]^{\mathsf{KP}}_{\mathtt{pk}}$ and $[\![v]\!]^{\mathsf{THR}}_{\mathtt{pk}}$, respectively.

To denote a variable that is not specified, we will use '*'.

3.2 Public-key Encryption

A public key encryption scheme PKE with message space $\mathcal{M}$ consists of three PPT algorithms $\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}$ defined as follows.

- $\mathsf{KeyGen}(1^\lambda)$: The key generation algorithm takes as input the security parameter and outputs a public key $\mathtt{pk}$ and a secret key $\mathtt{sk}$.
- $\mathsf{Enc}(\mathtt{pk}, m; r)$: The encryption algorithm takes a public key $\mathtt{pk}$, a message m and randomness r, and outputs a ciphertext $\mathtt{c}$.
- $\mathsf{Dec}(\mathtt{sk}, \mathtt{c})$: The decryption algorithm takes a secret key $\mathtt{sk}$ and a ciphertext $\mathtt{c}$ and outputs a message m.

3.3 Robust Secret Sharing

A secret sharing scheme allows a dealer to randomly split a secret between n parties so that qualified subsets (i.e., subsets of size $\geq$ threshold k) of parties can reconstruct the secret from their shares while unqualified subsets learn nothing about the secret. We consider a robust variant where the secret should be correctly reconstructed even if at most $k-1$ shares are corrupted by an adversary. We formalize this below.

Definition 1 (Robust secret sharing) *An (n, k) secret sharing scheme, with alphabet Σ and message length m, is a pair of functions $(\mathsf{Share}, \mathsf{Combine})$, where $\mathsf{Share} : \Sigma^m \to \Sigma^n$ is probabilistic and $\mathsf{Combine} : \Sigma^n \to \Sigma^m$ is deterministic, which satisfy the following properties.*

Privacy. *For a privacy threshold k, the adversary can choose a sequence $W = (\mathsf{w}_1, ..., \mathsf{w}_{k-1}) \in [n]^{k-1}$ of share indices to observe. We say that the scheme is ϵ-private if for every such strategy, there is a share distribution $\mathcal{D}$ over Σ^{k-1} such that for every secret message $x \in \Sigma^m$, $\mathsf{Share}(x)_W$ is ϵ-close (in statistical distance) to $\mathcal{D}$. We refer to ϵ as the privacy error and say that the scheme has perfect privacy if $\epsilon = 0$.*

Reconstruction. *We say that the scheme has reconstruction error η if for every secret message $x \in \Sigma^m$, $\Pr[\mathsf{Combine}(\mathsf{Share}(x)) = x] \geq 1 - \eta$. We say the scheme has perfect reconstruction if $\eta = 0$.*

We are also interested in robust secret sharing, where an adversary is allowed to modify at most $k-1$ shares.

Robustness. *For any secret $x \in \Sigma^m$, let $Y = \mathsf{Share}(x)$. Consider an arbitrary adversary who observes $k-1$ shares and can then arbitrarily change these $k-1$ shares, transforming Y to Y'.*

The scheme is robust if for every such adversary,

$$\Pr[\mathsf{Combine}(Y') = x] \geq 1 - \eta.$$

3.4 Key-private Encryption

A key-private public-key encryption scheme is a special kind of encryption scheme where ciphertexts cannot be linked to the public key used to encrypt them. The definition is borrowed from [22].

Definition 2 (Key-private Encryption) *A key-private public-key encryption scheme* $\mathcal{KP} = (\mathsf{Setup}^{\mathsf{KP}}, \mathsf{KeyGen}^{\mathsf{KP}}, \mathsf{Enc}^{\mathsf{KP}}, \mathsf{Dec}^{\mathsf{KP}})$ *is a public-key encryption scheme that satisfies the below security properties.*

1. *ciphertext indistinguishability under chosen-ciphertext attack (IND-CCA security); and*
2. *key indistinguishability under chosen-ciphertext attack (IK-CCA security). While the first property is standard, the second is less known; informally, IK-CCA requires that ciphertexts cannot be linked to the public key used to encrypt them, or to other ciphertexts encrypted with the same public key. For definitions, we refer the reader to [23].*

3.5 Threshold Encryption

A threshold encryption scheme allows sharing a secret key across multiple servers which can compute decryption shares of a given ciphertext and the shares can be combined to obtain the plaintext. The formal description below is borrowed from [24].

Definition 3 (Threshold Encryption) *A threshold encryption scheme consists of the following algorithms.*

- $\mathsf{KeyGen}^{\mathsf{THR}}(1^\lambda, 1^n, 1^k) \rightarrow (\mathtt{pk}^{\mathsf{THR}}, \mathtt{vk}^{\mathsf{THR}}, \overrightarrow{\mathtt{sk}}^{\mathsf{THR}})$**.** *The probabilistic key generation algorithm* $\mathsf{KeyGen}^{\mathsf{THR}}$ *takes as input a security parameter* λ*, the number of decryption servers* $n \geq 1$*, and the threshold parameter* k $(1 \leq k \leq n)$ *(all in their binary forms); it outputs*

$$(\mathtt{pk}^{\mathsf{THR}}, \mathtt{vk}^{\mathsf{THR}}, \overrightarrow{\mathtt{sk}}^{\mathsf{THR}}) \leftarrow \mathsf{KeyGen}^{\mathsf{THR}}(1^\lambda, 1^n, 1^k)$$

where $\mathtt{pk}^{\mathsf{THR}}$ *is the public encryption key,* $\mathtt{vk}^{\mathsf{THR}}$ *is the public verification key, and* $\overrightarrow{\mathtt{sk}}^{\mathsf{THR}} = (\mathtt{sk}_1^{\mathsf{THR}}, \mathtt{sk}_2^{\mathsf{THR}}, \ldots, \mathtt{sk}_n^{\mathsf{THR}})$ *is the list of private key shares. The servers can execute an interactive protocol to compute the* $\mathsf{KeyGen}^{\mathsf{THR}}$ *functionality in such a way that* $\mathtt{pk}^{\mathsf{THR}}, \mathtt{vk}^{\mathsf{THR}}$ *will be known by all parties and* $\mathtt{sk}_i^{\mathsf{THR}}$ *will be known only by the* i*-th decryption server.*

- $\mathsf{Enc}^{\mathsf{THR}}(\mathtt{pk}^{\mathsf{THR}}, m) \to \mathtt{c}$. *A probabilistic encryption algorithm* $\mathsf{Enc}^{\mathsf{THR}}$ *that takes as input a public key* $\mathtt{pk}^{\mathsf{THR}}$ *and a plaintext* m*, and outputs a ciphertext* $\mathtt{c} = \mathsf{Enc}^{\mathsf{THR}}(\mathtt{pk}^{\mathsf{THR}}, m)$.

- $\delta_i \leftarrow \mathsf{DecShare}(\mathtt{sk}_i^{\mathsf{THR}}, \mathtt{c})$. *A probabilistic decryption share generation algorithm* DecShare *that takes as input a private key share* $\mathtt{sk}_i^{\mathsf{THR}}$ *and a ciphertext* $\mathtt{c}$*, and outputs a decryption share* $\delta_i \leftarrow \mathsf{DecShare}(\mathtt{sk}_i^{\mathsf{THR}}, \mathtt{c})$.

- $\mathsf{DecShareVfy}(\mathtt{vk}^{\mathsf{THR}}, \mathtt{c}, \delta_i) \in \{\mathtt{valid}, \mathtt{invalid}\}$. *A share verification algorithm* DecShareVfy *that takes as input the public verification key* $\mathtt{vk}^{\mathsf{THR}}$*, a ciphertext* $\mathtt{c}$*, and decryption share* δ_i*, and outputs either* `valid` *or* `invalid`.

- $m \leftarrow \mathsf{DecCombine}(\mathtt{vk}^{\mathsf{THR}}, \mathtt{c}, \boldsymbol{\Delta})$. *A combining algorithm* DecCombine *that takes as input the public verification key* $\mathtt{vk}^{\mathsf{THR}}$*, a ciphertext* $\mathtt{c}$*, and a set of decryption shares* $\boldsymbol{\Delta}$*, and outputs a plaintext* $m \leftarrow \mathsf{DecCombine}(\mathtt{vk}^{\mathsf{THR}}, \mathtt{c}, \boldsymbol{\Delta})$. *The combining algorithm is also allowed to output a special '?' symbol that is distinct from all possible plaintext messages.*

We shall say a set $\boldsymbol{\Delta}$ of decryption shares is full if it contains k shares, no two of which belong to the same server.

Completeness. Let $(\mathtt{pk}^{\mathsf{THR}}, \mathtt{vk}^{\mathsf{THR}}, \overrightarrow{\mathtt{sk}}^{\mathsf{THR}})$ be an output of $\mathsf{KeyGen}^{\mathsf{THR}}(1^\lambda, 1^n, 1^k)$. We require the following two consistency properties:

1. For any ciphertext $\mathtt{c}$, if $\delta_i \leftarrow \mathsf{DecShare}(\mathtt{sk}_i^{\mathsf{THR}}, \mathtt{c})$ where $\mathtt{sk}_i^{\mathsf{THR}}$ is the i-th private key share in $\overrightarrow{\mathtt{sk}}^{\mathsf{THR}}$, then $\mathsf{DecShareVfy}(\mathtt{vk}^{\mathsf{THR}}, \mathtt{c}, \delta_i) \to \mathtt{valid}$.

2. If $\mathtt{c}$ is the output of $\mathsf{Enc}^{\mathsf{THR}}(\mathtt{pk}^{\mathsf{THR}}, m)$ and $\boldsymbol{\Delta} = \{\delta_i\}_i$ is a set of decryption shares $\delta_i \leftarrow \mathsf{DecShare}(\mathtt{sk}_i^{\mathsf{THR}}, \mathtt{c})$ for k distinct private keys in $\overrightarrow{\mathtt{sk}}^{\mathsf{THR}}$, then we require that $\mathsf{DecCombine}(\mathtt{vk}^{\mathsf{THR}}, \mathtt{c}, \boldsymbol{\Delta}) = m$.

Security. Security of a threshold PKE is defined using two properties:

1. security against chosen ciphertext attacks (CCA), and

2. decryption consistency.

Intuitively, security against CCA means the following: Even when an adversary corrupts $k-1$ of n servers (and thereby gets their corresponding private keys shares and gets to view corresponding decryption shares of maliciously chosen ciphertexts), she cannot distinguish encryptions of two different plaintexts. Decryption consistency means that an adversary cannot produce two sets of full decryption shares for a ciphertext so that each decryption share verifies correctly by the verification key but the results of combining the two decryption shares are different. For formal definitions, we refer the readers to [25].

4 A Private NFT Protocol

Definition 4 (A private NFT (pNFT) protocol) *A private NFT protocol is a protocol between a seller $\mathcal{S}$, validators $\mathcal{V}_1, \ldots, \mathcal{V}_{n_v}$, and bidders $B_1, \ldots, B_{n_b}$, consists of the following phases.*

- ***The Setup Phase.*** *This phase generates secret parameters for every party and public parameters.*
- ***The Putting-up-an-auction Phase.*** *In this phase, a seller $\mathcal{S}$ puts up an NFT for auction.*
- ***The Bidding Phase.*** *Bidders bid on the NFT.*
- ***The Bid Resolution Phase.*** *The validators sort the bids and execute the transaction from the highest bidder to the seller.*

Privacy of a pNFT protocol. A pNFT protocol offers privacy in the following dimensions.

1. **Sender and bidder identities.** It must be computationally hard to link different transactions (bidding, buying or selling) performed using the same key.
2. **Seller wallet balance.** It must be computationally hard to know a seller's wallet balance. This is in the interest of the seller's financial privacy.
3. **Bidder wallet balance.** It must be computationally hard to know any bidder's wallet balance. This is in the interest of the bidder's financial privacy.

4. **Closed bidding.** For any asset being sold, one can have no knowledge of other bids until the sale completes the bidding phase. This privacy aspect holds true even from the point of view of validators.

5. **The winning bid.** Once the bidding phase is complete, only the validators will learn the bid values. All other parties (except the seller and the corresponding bidder) do not learn anything about the bids, including the winning bid.

4.1 Zero Knowledge Proofs alone are not sufficent for pNFT.

4.1.1 Sketch of Zerocash

Zerocash works in a UTXO model. That is, money is represented via coins. The commitment of a coin is published on the ledger when the coin is created, and its serial number is published when the coin is consumed. Each transaction on the ledger attests that some "old" coins were consumed in order to create some "new" coins: it contains the serial numbers of the consumed coins $\mathtt{sn}$, commitments of the created coins $\mathtt{comm}$, and a zero knowledge proof π attesting that the serial numbers belong to coins created in the past (without identifying which ones), and that the commitments contain new coins of the same total value. The commitments and the serial numbers are designed in such a way that they do not reveal the public key of the sender nor the coin.

In a transaction, in order for the sender to send the new coins to some recipient's public key $\mathtt{pk}$, the sender also includes an encryption under the recipient's public key of the commitment randomnesses $\mathtt{r_{comm}}$ and the coin value v: $\mathsf{Enc}(\mathtt{pk}, (\mathtt{r_{comm}}, v, \rho))$, where, ρ is the seed for the serial number corresponding to the coin. In order for the ciphertext to not reveal the recipient public key, a special encryption scheme called a key-private encryption scheme is used.

5 Paras - The pNFT Protocol

Let $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a public-key encryption scheme. Let $(\mathsf{Setup}^{\mathsf{KP}}, \mathsf{KeyGen}^{\mathsf{KP}}, \mathsf{Enc}^{\mathsf{KP}}, \mathsf{Dec}^{\mathsf{KP}})$ be a key-private public-key encryption scheme. Let $(\mathsf{Share}, \mathsf{Combine})$ be a robust secret sharing scheme. Let $(\mathsf{KeyGen}^{\mathsf{THR}}, \mathsf{Enc}^{\mathsf{THR}}, \mathsf{DecShare}, \mathsf{DecShareVfy}, \mathsf{DecCombine})$ be a threshold encryption scheme. Let PRF be a pseudorandom function. Paras works as follows.

5.1 The Setup Phase

The setup phase consists of the following components.

- Threshold key generation
- Common key generation
- Individual key generation

5.1.1 Threshold Key Generation

All the validators participate in this interactive protocol of threshold key generation based on the threshold encryption scheme. Each validator inputs their randomness share to the key generation $(\mathtt{pk}_V^{\mathsf{THR}}, \mathtt{vk}_V^{\mathsf{THR}}, \overrightarrow{\mathtt{sk}}^{\mathsf{THR}}) \leftarrow \mathsf{KeyGen}^{\mathsf{THR}}(1^\lambda, 1^n, 1^k)$. The result of the protocol is a threshold public key $\mathtt{pk}_V^{\mathsf{THR}}$ and a decryption share verification key $\mathtt{vk}_V^{\mathsf{THR}}$ which all validators receive, and a private key share $\mathtt{sk}_i^{\mathsf{THR}}$ that $\mathcal{V}_i$ receives.

5.1.2 Common key generation

The outcome of this step is a common key-pair $(\mathtt{sk}_V, \mathtt{pk}_V)$ – common to all the validators – per the standard public key encryption scheme (i.e., $(\mathtt{sk}_V, \mathtt{pk}_V) \leftarrow \mathsf{KeyGen}(1^\lambda)$); no other party learns about $\mathtt{sk}_V$. See Section 5.4.3 for implementation details.

5.1.3 Individual key generation

Each validator $\mathcal{V}_i$ generates its own key pair $(\mathtt{sk}_i, \mathtt{pk}_i)$ per the standard encryption scheme (i.e., $(\mathtt{sk}_i, \mathtt{pk}_i) \leftarrow \mathsf{KeyGen}(1^\lambda)$).

5.2 The Putting-up-an-auction Phase

Towards minting an NFT, a seller performs the following steps.

1. Sample a fresh key pair $(\mathtt{sk}_{\mathsf{nft}}, \mathtt{pk}_{\mathsf{nft}})$ per the standard encryption scheme (i.e., $(\mathtt{sk}_{\mathsf{nft}}, \mathtt{pk}_{\mathsf{nft}}) \leftarrow \mathsf{KeyGen}(1^\lambda)$).
2. Encrypt the asset with $\mathtt{pk}_{\mathsf{nft}}$ before storing it in a distributed storage such as IPFS.
3. Secret-share $\mathtt{sk}_{\mathsf{nft}}$ and distribute the shares privately to corresponding validators: $(\mathtt{sk}_{\mathsf{nft}}[1], \mathtt{sk}_{\mathsf{nft}}[2], \ldots, \mathtt{sk}_{\mathsf{nft}}[n_v]) \leftarrow \mathsf{Share}(\mathtt{sk}_{\mathsf{nft}})$. Send $\mathtt{sk}_{\mathsf{nft}}[i]$ to $\mathcal{V}_i$ over a private channel (i.e., by encrypting under $\mathtt{pk}_i$ of $\mathcal{V}_i$).

This completes the phase of putting up an auction.1

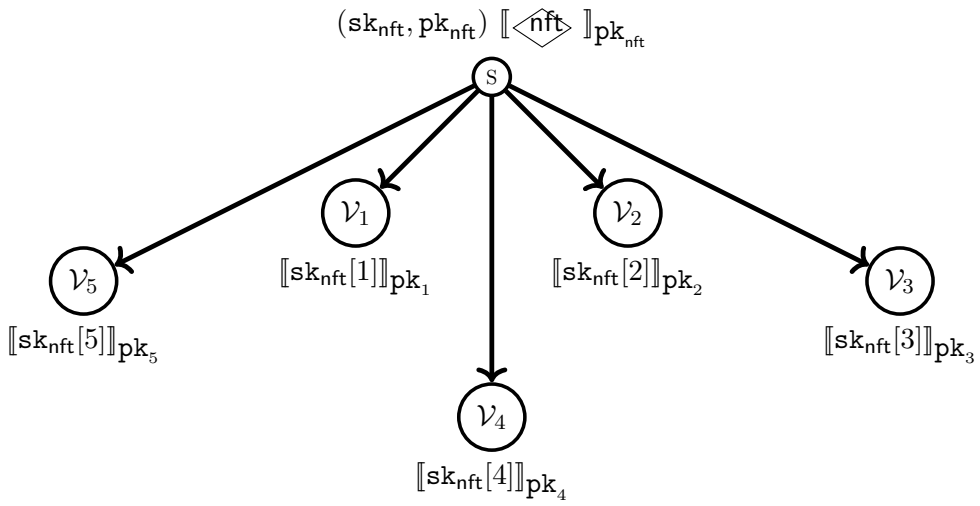

Figure 1: The Putting-up-an-auction Phase

5.3 The Bidding Phase

Let B be a bidder with a secret-key public-key pair $(\mathtt{msk}_B, \mathtt{mpk}_B)$ per the standard encryption scheme. These keys will be used to derive non-linkable key pairs as described below and hence the notations convey that the key pair is a master key pair in that respect.

Let the number of bids performed by B so far be $n_b - 1$. To place the n_b-th bid, B proceeds as follows:

1. Derive a secret key for to the current bid:

$$\mathtt{sk}_B \leftarrow \mathsf{PRF}(\mathtt{msk}_B, n_b)$$

   where, PRF is a pseudorandom function.

2. Derive $\mathtt{pk}_B$ corresponding to $\mathtt{sk}_B$. For example, $\mathtt{pk}_B \leftarrow \mathsf{KeyGen}(1^\lambda; \mathtt{sk}_B)$.

3. Let v be the bid that B wants to place for an NFT. Let $\mathtt{pk}_V^{\mathsf{THR}}$ be the threshold public key of the validators. Compute the threshold ciphertext:

$$\mathsf{c}^{\mathsf{THR}} \leftarrow [\![v]\!]_{\mathtt{pk}_V^{\mathsf{THR}}}$$

4. Compute $(\mathtt{comm}, \overrightarrow{\mathtt{sn}})$, where $\overrightarrow{\mathtt{sn}} = (\mathtt{sn}_1, \mathtt{sn}_2)$, like in zerocash (as explained in Section 4.1.1). Roughly, $\mathtt{comm}$ is a commitment to the bidding value v and $\mathtt{sn}_1, \mathtt{sn}_2$ are serial numbers corresponding to the old coins being spent to make this transaction.

5. Let $\mathtt{pk}_{\mathcal{S}}^{\mathsf{kp}}$ be the public key of the seller $\mathcal{S}$ where the encryption scheme is key-hiding[1]. Compute $\mathsf{c}^{\mathsf{kp}} \leftarrow \mathsf{Enc}^{\mathsf{KP}}(\mathtt{pk}_{\mathcal{S}}^{\mathsf{kp}}, (\mathtt{r}_{\mathtt{comm}}, v, \rho))$, where, $\mathtt{r}_{\mathtt{comm}}$ is the commitment randomnesses and ρ is the seed for the serial number corresponding to the coin being potentially sent to $\mathcal{S}$. Compute encryption of c^{kp} under $\mathtt{pk}_V$ as $[\![\mathtt{pk}_V]\!]_{\mathsf{c}^{\mathsf{kp}}}$.

6. Compute a proof π of the following statement: There exist

   - a transaction: $\mathsf{txn}_1 = (\mathtt{comm}_1, \overrightarrow{\mathtt{sn}_1}, \pi_1, [\![v_1]\!]_{\mathtt{pk}_V^{\mathsf{THR}}}, \mathtt{pk}_{B_1}, \mathsf{c}_1^{\mathsf{kp}})$
   - a transaction: $\mathsf{txn}_2 = (\mathtt{comm}_2, \overrightarrow{\mathtt{sn}_2}, \pi_2, [\![v_2]\!]_{\mathtt{pk}_V^{\mathsf{THR}}}, \mathtt{pk}_{B_2}, \mathsf{c}_2^{\mathsf{kp}})$
   - a proof of consensus: ϕ_{txn_1}
   - a proof of consensus: ϕ_{txn_2}
   - a bid value v, randomness r
   - a secret key: $\mathtt{msk}_B$ (per the standard encryption scheme) and a positive integer n_b (to represent the n_b-th bid by the bidder)

   Such that conjunction of the following sub-statements holds:

   - Statement 0: $\mathtt{comm}_1, \mathtt{comm}_2$ commit to v_1, v_2 respectively such that $v = v_1 + v_2$; B is authorized to spend the coins in txn_1 and txn_2; the coins in txn_1 and txn_2 are unspent; $\mathtt{sn}_1, \mathtt{sn}_2$ correspond to coins in $\mathsf{txn}_1, \mathsf{txn}_2$, respectively; π_1, π_2 are valid proofs[2].
   - Statement 1: There exist v' and randomness r such that $\mathsf{c}^{\mathsf{THR}} = \mathsf{Enc}^{\mathsf{THR}}(\mathtt{pk}_V^{\mathsf{THR}}, v; r)$
   - Statement 2: Consensus proofs ϕ_{txn_1} and ϕ_{txn_2} are valid (in other words, $\mathsf{ConsensusVfy}(\mathsf{txn}_1, \phi_{\mathsf{txn}_1}) = \mathtt{valid}$ and $\mathsf{ConsensusVfy}(\mathsf{txn}_2, \phi_{\mathsf{txn}_2}) = \mathtt{valid}$).
   - Statement 3: Let $\mathtt{sk}_B = \mathsf{PRF}(\mathtt{msk}_B, n_b)$; then $(*, \mathtt{pk}_B) = \mathsf{KeyGen}(1^\lambda; \mathtt{sk}_B)$.

7. Broadcast $(\mathtt{comm}, \overrightarrow{\mathtt{sn}}, \pi, [\![v]\!]_{\mathtt{pk}_V^{\mathsf{THR}}}, \mathtt{pk}_B, [\![\mathsf{c}^{\mathsf{kp}}]\!]_{\mathtt{pk}_V})$ to all validators.

[1] This can be derived in a way similar to $\mathtt{pk}_B$ of the bidder.
[2] This can be generalized to arbitrary number of "old" coins

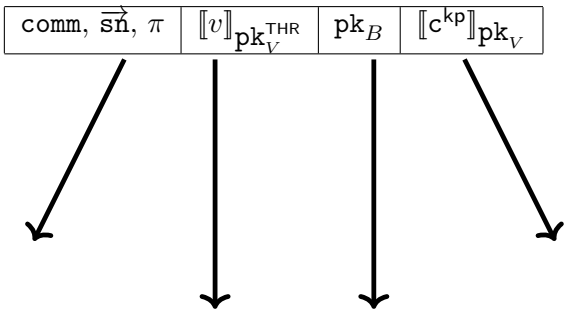

Figure 2: The Bidding Phase

5.4 The Bid Resolution Phase

In this phase, the validators decipher the bids, sort them and execute the transaction corresponding to the highest bid. The validators proceed as follows.

1. After the bidding phase is complete, the validators perform threshold decryption of every bid value. Specifically, consider a bidding transaction $(\mathtt{comm}, \overrightarrow{\mathtt{sn}}, \pi, [\![v]\!]_{\mathtt{pk}_V^{\mathsf{THR}}}, \mathtt{pk}_B)$. Every validator $\mathcal{V}_j$ computes its decryption share δ_j of the ciphertext $[\![v]\!]_{\mathtt{pk}_V^{\mathsf{THR}}}$ and encrypts it under the common public key $\mathtt{pk}_V$. That is, $\mathcal{V}_j$, who has threshold secret key $\mathtt{sk}_j^{\mathsf{THR}}$, computes $\delta_j \leftarrow \mathsf{DecShare}(\mathtt{sk}_j^{\mathsf{THR}}, \mathsf{c})$ where $\mathsf{c} = [\![v]\!]_{\mathtt{pk}_V^{\mathsf{THR}}}$; $\mathcal{V}_j$ then computes encryption under validators' common public key of δ_j as $\mathsf{c}_j = [\![\delta_j]\!]_{\mathtt{pk}_V}$.

2. For every $i \neq j$, validator $\mathcal{V}_j$ decrypts c_i as $\delta_i \leftarrow \mathsf{Dec}(\mathtt{sk}_V, \mathsf{c}_i)$, where $\mathtt{sk}_V$ is the common secret key of validators. Then, for all i, $\mathcal{V}_j$ checks whether $\mathsf{DecShareVfy}(\mathtt{vk}_V^{\mathsf{THR}}, \mathsf{c}, \delta_i) = \mathtt{valid}$. Let $\boldsymbol{\Delta}$ denote the set of all valid decryption shares. Then, $\mathcal{V}_j$ combines the shares to obtain the bidding value $v \leftarrow \mathsf{DecCombine}(\mathtt{vk}_V^{\mathsf{THR}}, \mathsf{c}, \boldsymbol{\Delta})$.

3. Every validator sorts the bids and selects the highest bidder B for the following steps. Let $(\mathtt{comm}, \overrightarrow{\mathtt{sn}}, \pi, [\![v]\!]_{\mathtt{pk}_V^{\mathsf{THR}}}, \mathtt{pk}_B)$ be the bidding

transaction by B.

4. Recall that validator $\mathcal{V}_i$ had received a secret key share $\mathtt{sk}_{\mathsf{nft}}[i]$ of the secret key that encrypts the NFT in question, in the putting-up-an-auction phase. $\mathcal{V}_i$ encrypts and sends $\mathtt{sk}_{\mathsf{nft}}[i]$ to the winning bidder: $[\![\mathtt{sk}_{\mathsf{nft}}[i]]\!]_{\mathtt{pk}_B}$

5. B decrypts $[\![\mathtt{sk}_{\mathsf{nft}}[i]]\!]_{\mathtt{pk}_B}$ to obtain $\mathtt{sk}_{\mathsf{nft}}[i]$, for all i. Then B recovers $\mathtt{sk}_{\mathsf{nft}}$ by computing $\mathtt{sk}_{\mathsf{nft}} \leftarrow \mathsf{Combine}(\mathtt{sk}_{\mathsf{nft}}[1], \ldots, \mathtt{sk}_{\mathsf{nft}}[n_v])$.

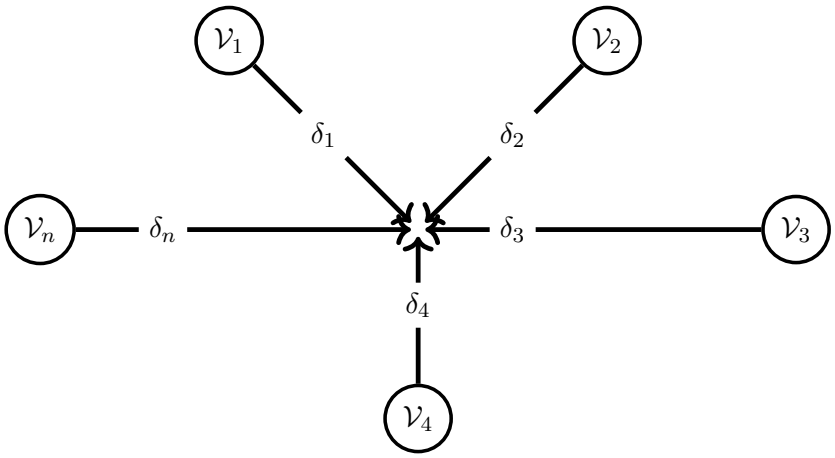

Figure 3: The Bid Resolution Phase

5.4.1 Key-private Encryption Scheme

To instantiate the key-private encryption scheme, we use the key-private Elliptic-Curve Integrated Encryption Scheme (ECIES) [26, 27]; it is one of the few standardized key-private encryption schemes with available implementations.

5.4.2 Threshold Key Generation.

Our protocol works with Delegated Proof-of-Stake model, where the community elects a set of validators to run the consensus. Note that standard threshold schemes treat every player identically and have no notion of "weight" in the consensus. Hence, the network must adapt them to take

validators' weight into account. A simple approach is to assign multiple threshold shares to larger validators.

Threshold key generation algorithms are usually interactive protocols between multiple participants. A special transaction on the Paras-powered network instructs the validators to commence execution of this stateful protocol. Each validator runs a threshold daemon process that is responsible for the secure keeping of the secret state. For each phase of the protocol:

1. A validator keeps the state of the protocol in its local memory.
2. It calls the secret daemon to generate the messages as per the protocol description for other validators.
3. It propagates the messages either via the broadcast or via the private channels to other validators (see 4).
4. Each validator executes state transition functions to update its state, proceed to the next phase of the protocol, and repeat the above steps.

At the end of the protocol, a threshold public key is generated on the Paras-powered network, and it can be displayed back to the user (e.g., for bidding/selling) or to the application that generated the initial request.

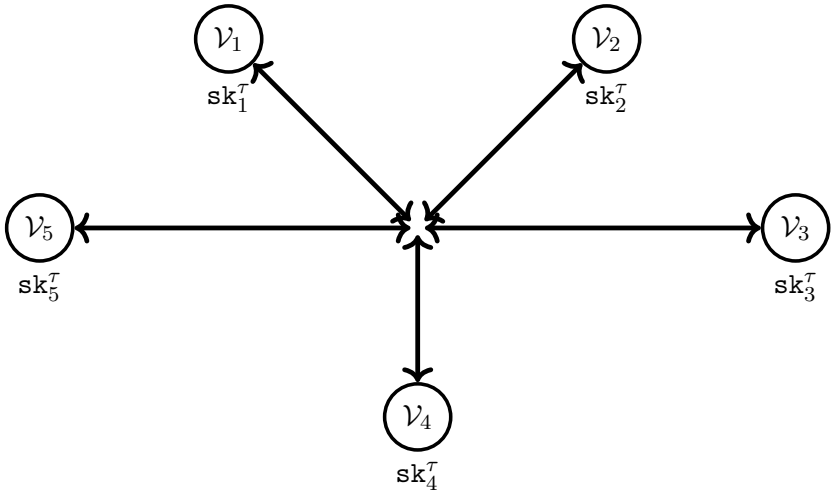

Figure 4: Distributed Threshold-key Generation

Threshold Decryption. Decryption requests on the Paras-powered network are processed similarly to the key-generation requests. These are invoked when a bidding phase is complete for a given seller. These are interactive protocols, and state transition between the rounds is triggered as a function of the messages propagated via the Paras-powered blockchain view and every validator's local memory.

Handling Validator Membership Changes. The validator set needs to be rotated periodically to allow for new stakeholders to join the set. Upon a validator set update, we need to update the threshold key to be shared across the new set. Thus if we allowed anyone to join at any time, we would have to update the threshold key very frequently. To prevent this, we rotate validators every T blocks. Within intervals of T rounds, the set V^R and the threshold key are fixed. At every round that is an integral multiple of the parameter T, we update the validator set as follows:

1. At any round R, the Paras state keeps track of the current validator set V^R. $V^{R+1} = V^R$ unless $R+1$ is a multiple of T.

2. During rounds $((i-1)T, iT]$, users post bonding/unbonding messages.

3. At the end of round iT, these messages are applied to V^{iT-1} to get V^{iT}.

Threshold Key Generation and Threshold decryption in the Presence of Rotating Validators. Paras-powered blockchain may issue a request for a new key or a threshold decryption at round R. The decryption process takes longer than one round; instead of slowing down consensus, we request that the decryption is produced before round $R+x$ starts, where x is related to the number of rounds it takes to produce a decryption. In particular, validators start round $R+x$ only after seeing a certificate for round $R+(x-1)$ and a decryption for each decryption request issued at round R. The outcome of all round R requests must be included in block $R+x$. In other words, a round R block proposal that does not contain the outcomes from a round $R-x-1$ is considered invalid, and validators don't vote on it. To ensure that all threshold messages are decrypted before a validator set update, Paras does not issue any threshold requests during a round equal to $-1, -2, \ldots, -x-1 \mod T$.

5.4.3 Common Key Generation.

We will implement the step of common key generation by having one of the validators (which is arbitrarily chosen – for example, it can be the first one in a lexicographic ordering by their identities) sample the key pair and send the key pair to every validator over a private channel.

In order to disallow a cheating validator from sharing malicious key pairs, we require the following property from the encryption scheme. For every public key, there exists a unique private key and the correctness of a key pair can be verified efficiently given the key pair. We note that commonly used encryption schemes satisfy this property.

5.4.4 Consensus Proofs

Recall that the protocol relies on an additional proof (i.e., in addition to the Zerocash approach of checking for knowledge of commitment randomnesses and secrets) that a bidding transaction went through consensus for a bidder to spend the coins in the said transaction. In the bidding phase, these proofs are denoted as ϕ_{txn} to denote a proof of consensus on a transaction $\mathtt{txn}$.

Towards implementing such a consensus proof, we can either rely on the traditional method of verifying the state proof of the blockchain or rely on a threshold signature from the validators.

References

[1] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2009. https://bitcoin.org/bitcoin.pdf.

[2] Nft investors have sent 37 billion to marketplaces in 2022, on track to surpass 2021. https://nairametrics.com/2022/05/06/nft-investors-have-sent-37-billion-to-marketplaces-in-2022-on-track-to-surpass-2021/.

[3] Quentin tarantino announces dates for nft auction of scenes from original, handwritten pulp fiction screenplay on secret network. https://aithority.com/technology/blockchain/nft/quentin-tarantino-announces-dates-for-nft-auction-of-scenes-from-original-handwritten-pulp-fiction-screenplay-on-secret-network/.

[4] The chainsmokers will headline first-ever nft-ticketed music festival. https://www.edmtunes.com/2022/02/the-chainsmokers-will-headline-first-ever-nft-ticketed-music-festival/.

[5] Could nfts be a potential game-changer for the live events industry? https://www.rollingstone.com/culture-council/articles/nfts-game-changer-industry-1151676/.

[6] Are nft sales susceptible to shill bidding nft skeptics think its possible. https://news.bitcoin.com/are-nft-sales-susceptible-to-shill-bidding-nft-skeptics-think-its-possible/.

[7] The specter of shill bidding around nfts. https://nft.substack.com/p/the-specter-of-shill-bidding-around?s=r.

[8] Crime and nfts: Chainalysis detects significant wash trading and some nft money laundering in this emerging asset class. https://blog.chainalysis.com/reports/2022-crypto-crime-report-preview-nft-wash-trading-money-laundering/.

[9] Over 33 percent of nft volume is wash trading. https://cryptopotato.com/over-33-of-nft-volume-is-wash-trading-bitscrunch-ceo-interview/.

[10] The flip side of nft art world: Theft and plagiarism. https://www.hindustantimes.com/lifestyle/art-culture/the-flip-side-of-nft-art-world-theft-and-plagiarism-101646458700163.html.

[11] Artists say plagiarized nfts are plaguing their community. https://hyperallergic.com/702309/artists-say-plagiarized-nfts-are-plaguing-their-community/.

[12] Secret network website. https://scrt.network/.

[13] Cristofaro Mune and Niek Timmers. Teepwn: Breaking trusted execution environments - extended edition. https://www.offensivecon.org/trainings/2022/teepwn-breaking-trusted-execution-environments-extended-edition.html.

[14] Jianting Ning, Hung Dang, Ruomu Hou, and Ee-Chien Chang. Keeping time-release secrets through smart contracts. *IACR Cryptol. ePrint Arch.*, page 1166, 2018. https://eprint.iacr.org/2018/1166.

[15] Enrico Bacis, Dario Facchinetti, Marco Guarnieri, Marco Rosa, Matthew Rossi, and Stefano Paraboschi. I told you tomorrow: Practical time-locked secrets using smart contracts. In Delphine Reinhardt and Tilo Müller, editors, *ARES 2021: The 16th International Conference*

on Availability, Reliability and Security, Vienna, Austria, August 17-20, 2021, pages 17:1–17:10. ACM, 2021. https://aisberg.unibg.it/handle/10446/202638.

[16] Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. Succinct non-interactive zero knowledge for a von neumann architecture. In Kevin Fu and Jaeyeon Jung, editors, *Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, August 20-22, 2014*, pages 781–796. USENIX Association, 2014.

[17] Mina protocol. https://minaprotocol.com/.

[18] Aleo website. https://www.aleo.org/.

[19] Zcash website. https://z.cash/.

[20] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 467–476. ACM, 2013.

[21] Vinod Vaikuntanathan. Computing blindfolded: New developments in fully homomorphic encryption. In Rafail Ostrovsky, editor, *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 5–16. IEEE Computer Society, 2011.

[22] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy, SP 2014, Berkeley, CA, USA, May 18-21, 2014*, pages 459–474. IEEE Computer Society, 2014.

[23] Mihir Bellare, Alexandra Boldyreva, Anand Desai, and David Pointcheval. Key-privacy in public-key encryption. In Colin Boyd, editor, *Advances in Cryptology - ASIACRYPT 2001, 7th International Conference on the Theory and Application of Cryptology and Information Security, Gold Coast, Australia, December 9-13, 2001, Proceedings*, volume 2248 of *Lecture Notes in Computer Science*, pages 566–582. Springer, 2001.

[24] Dan Boneh, Xavier Boyen, and Shai Halevi. Chosen ciphertext secure public key threshold encryption without random oracles. In David Pointcheval, editor, *Topics in Cryptology - CT-RSA 2006, The Cryptographers' Track at the RSA Conference 2006, San Jose, CA, USA, February 13-17, 2006, Proceedings*, volume 3860 of *Lecture Notes in Computer Science*, pages 226–243. Springer, 2006.

[25] Dan Boneh, Xavier Boyen, and Shai Halevi. Chosen ciphertext secure public key threshold encryption without random oracles. In David Pointcheval, editor, *Topics in Cryptology - CT-RSA 2006, The Cryptographers' Track at the RSA Conference 2006, San Jose, CA, USA, February 13-17, 2006, Proceedings*, volume 3860 of *Lecture Notes in Computer Science*, pages 226–243. Springer, 2006.

[26] Victor Shoup. A proposal for an iso standard for public key encryption (version 2.1), 2001. `https://www.shoup.net/papers/iso-2_1.pdf`.

[27] Certicom Research. Standards for efficient cryptography, SEC 1: Elliptic curve cryptography (version 2.0), 2009. `https://www.secg.org/sec1-v2.pdf`.