Nirvana: Instant and Anonymous Payment-Guarantees

Akash Madhusudan[1⋆], Mahdi Sedaghat[1⋆], Philipp Jovanovic[2], and Bart Preneel[1]

[1] imec-COSIC, KU Leuven, Leuven, Belgium
akash.madhusudan@esat.kuleuven.be
ssedagha@esat.kuleuven.be
bart.preneel@esat.kuleuven.be
[2] University College London, London, UK
p.jovanovic@ucl.ac.uk

Abstract. Given the high transaction confirmation latencies in public blockchains, cryptocurrencies such as Bitcoin, Ethereum, etc. are not yet suitable to support real-time services such as transactions on retail markets. There are several solutions to address this latency problem, with layer-2 solutions being the most promising ones. Existing layer-2 solutions, however, suffer from privacy and/or collateral issues when applied to retail environments where customer-merchant relationships are usually ephemeral.
In this paper, we propose Nirvana, that can be combined with existing cryptocurrencies to provide instant, anonymous and unlinkable payment guarantees. Nirvana does not require any trusted third party. It conceals the identities of honest participants, thus ensuring customer anonymity within the system while only relying on efficient Groth-Sahai proof systems. We introduce a novel randomness-reusable threshold encryption that mitigates double-spending by revealing the identities of malicious users. We formally prove how our scheme provides customer anonymity, unlinkability of transactions and payment guarantees to merchants. Our experiments demonstrate that Nirvana allows for fast (zero-confirmation) global payments in a retail setting with a delay of less than ~ 1.7 seconds.

Keywords: Blockchain, Instant Payments, Privacy-Preserving, Threshold Encryption, Non-Interactive Zero-Knowledge proofs, Structure-Preserving Threshold Signatures, Pseudo-Random Functions, Commitment.

1 Introduction

Public decentralized cryptocurrencies such as Bitcoin and Ethereum offer increased transparency and avoid trust in a central party. However, that comes at the cost of performance which precludes high throughput, real-time applications. First, the throughput is rather low; for Bitcoin the theoretical maximum limit has been shown to be 27 transactions per second (tps) [Geo19]. For the sake of comparison, Visa reached up to 47k tps [Tri13], while Alipay's peak was 459k tps in 2019.[3] Second, the latency of transaction confirmation is rather high, because transactions need to be verified by multiple parties rather than

⋆ The first and second authors are corresponding authors and contributed equally.
[3] tellimer.com

by a central operator. The latency of Ethereum for transaction confirmation is around 3 minutes [MWD+20] for transaction confirmation, whereas Visa only requires a few seconds [CDE+16].

Motivation. These weaknesses are a hindrance in the mass adoption of cryptocurrencies, especially in a retail setting that typically requires fast financial settlements. Various solutions to address these issues in popular public blockchains (e.g., Bitcoin) have been proposed. These are mainly based on alternative consensus schemes [BSAB+19], sharding [WSNH19] and layer-2 solutions [GMR+20]. Alternative consensus schemes and sharding mostly focus on increasing the transaction throughput. Layer-2 solutions emerged as a promising area of research to increase transaction throughput and solve the latency problem. Their approach allows a group of mistrusting parties to execute an application amongst themselves, hence bypassing the performance constraints of the underlying blockchain. The main requirement for these solutions to work is an unreusable monetary deposit (collateral) on the underlying blockchain, which is used to make final settlements. However, the existing proposals have several inherent limitations that make them suboptimal for retail payments. These limitations are discussed in the next section. Additionally, a new flavour of layer-2 solutions, referred to as rollups, is picking up traction in both industry and academia alike [But21,Eth21,Bje21,Ben21]. Similar to their predecessors (e.g., payment channels), rollups handle transactions away from the main blockchain, while taking advantage of the integrity properties offered by the underlying blockchain. However, these solutions are hybrid in the sense that, each transaction present in the rollup has some corresponding data on the main blockchain. Due to this property, rollups are not vulnerable to data availability attacks [GMR+20], which makes them applicable in a more general setting (unlike channels, channel hubs, or sidechains, that are application-specific [But21]). Despite having the advantage of increasing the transaction throughput of Ethereum up to 100 times compared to its current throughput [But21], rollups do not reduce the total confirmation time taken by each individual transaction.

This calls for a line of research that focuses on improving the *high transaction confirmation latency* of public blockchains while solving the shortcomings of the current state-of-the-art. Although such solutions already exist in literature [MWD+20], there is a significant room for improvement. Further motivation for our solution can be found in App. A

Our Solution. We propose Nirvana, an overlay network that provides instant, anonymous and unlinkable payment guarantees for payments made on existing public blockchains in a retail setting (customer-merchant interactions).

Nirvana is inspired by the abundant literature on e-cash schemes [Cha82,CFN90,Bra94,Sch95,FTY96,CHL05,CG08,CGT08,BCKL09,BCFK15] and hence driven by the notion of exposing identities belonging to malicious entities while maintaining the privacy of honest actors. It serves as an abstraction layer for payment

guarantees in a retail setting (in parallel to actual on-chain payments), thus can be used as an extension to any blockchain-based cryptocurrency.

The participants in Nirvana include customers willing to purchase products/services using their cryptocurrencies while expecting short confirmation times; an established consortium of merchants willing to accept cryptocurrency payments; witnesses who are selected from the merchant consortium and a group of authorities responsible for registration of users and verification of collaterals. The customers and authorities only need to be online when performing an action such as making a payment (customers) or registering users (authorities). Merchants and witnesses, however, need to be online to communicate and accept fast payments, especially when incentivized to do so. This incentivization helps in absolving possible concerns of witness availability. With Nirvana, we aim to hide the real identity of honest customers from all other participants, while ensuring that the transactions made by them are instant and unlinkable within the system. Most importantly, none of the participants have to trust each other.

In order to briefly explain our solution, we assume a customer, a merchant and a set of authorities. The customer wants to pay for their products with a cryptocurrency of their choice, for instance, Bitcoin. Due to the notorious transaction confirmation latency of Bitcoin, the merchant cannot be guaranteed a payment for ~ 60 minutes on average. As its first contribution, Nirvana solves this latency problem by providing the merchant with an instantaneous payment guarantee.

Registration: During registration, the customer deposits a one-time collateral in a smart contract controlled by Nirvana deployed on any smart contract supporting blockchain (e.g., Ethereum[4] or Celo[5]). The customer receives a Structure-Preserving Threshold Signature (SPTS) from the majority of authorities on a secret Pseudo-Random Function (PRF) key certifying that they own a collateral in Nirvana. Along with this certificate, customers are also issued a fixed list of witnesses responsible for that collateral. These witnesses are randomly selected by the authorities from the list of merchants.

Payment: Once a registered customer wants to issue a payment guarantee to the merchant, they make an on-chain payment to the merchant's blockchain address. Subsequently, they prove ownership of a collateral in Nirvana by using a combination of SPTS, Commitment and Non-Interactive Zero-Knowledge proofs (NIZK) (explained in detail in Sect. 3). The customer also provides the transaction hash of their on-chain payment to the merchant. Note that our construction relies on Groth-Sahai proof system that provides efficient NIZKs in the standard model. This allows Nirvana to make its second contribution, i.e., ensuring protocol-level customer anonymity and providing unlinkability of transactions. Note that we assume blockchain transaction privacy to be orthogonal to our work.

Verification: Once the merchants receive an on-chain transaction, they have two options: either wait for the on-chain payment to be confirmed or request an instant pay-

[4] `https://ethereum.org/en/`
[5] `https://celo.org/`

ment guarantee in Nirvana. Assuming that the merchant opts for verifying the Nirvana payment guarantee, they proceed by first verifying the NIZK proof and SPTS signature attached to the guarantee. Once verified, they forward it to the assigned witnesses in order to ensure that this guarantee is not being double-spent. Note that these guarantees only work for a fixed latency period called an epoch. This epoch can be set to easily encompass the average latency period of most existing public blockchains. The witnesses then check if they have received a similar payment guarantee in the current epoch. If they have not received a similar guarantee, they record this one as fresh and store it locally along with the current timestamp. Each witness deletes the timestamps from previous epochs for storage efficiency. In the rare case of a double-spend, each witness informs the receiving merchant, who then cancels the transaction. Subsequently, as our third contribution, we propose a novel randomness-reusable Threshold encryption in bilinear groups (rbTE) that enables the witnesses to combine the payment guarantees and reveal the identity of the perpetrator. Later, this identity is communicated to the authorities who blacklist this malicious customer as a penalty. Now assume another case, where the customer does not double-spend their payment guarantee, yet they double-spend their *actual payment* in the underlying blockchain. In this case, the collateral of the customer in Nirvana can be used to remunerate the victim merchant (explained in detail in Sect. 3). This means that an honest customer *never* needs to replenish their collateral unlike in most existing layer-2 solutions [PD16,Net19,DW15,MBB+19,HAB+17,KZF+18] and the merchant is *guaranteed* a payment.

Our Contributions and Results. This paper makes the following contributions:

1. Nirvana provides an instant payment guarantee to merchants in a retail setting, enabling customers to make payments with the cryptocurrency of their choice and avail services without needing to wait for on-chain payment confirmation or replenish their collaterals.
2. Nirvana ensures protocol-level customer anonymity and unlinkability of transactions between customers and merchants when used on top of any blockchain. We assume blockchain transaction privacy to be orthogonal to our work; however, by utilizing Nirvana overlay network, public blockchains can still benefit from its ability to provide instant transaction confirmation. Privacy-preserving ledgers such as Zcash and Monero, on the other hand, enjoy the performance enhancement offered by Nirvana without the risk of losing privacy.
3. We propose a novel randomness-reusable threshold version of the ElGamal encryption, (see App. B.5) which enables participants to reveal the identity of malicious customers when they double-spend without relying on trusted third parties; otherwise their identity is always preserved. This combination reveals the identity and collateral details of the perpetrator, ensuring remuneration to the victims and blacklisting of malicious entities.

4. We formally prove that NIRVANA preserves the anonymity of honest customers and unlinkability of transactions, yet at the same time guarantees payment to the merchants.
5. We implement NIRVANA and show that it allows for fast global retail payments. The average spending time taken by a customer using NIRVANA is ~ 0.71 seconds (s) with the combined verification times taken by merchants and witnesses being ~ 1.16 s.

Outline. This paper is organized as follows: Sect. 2 provides preliminaries and system model of NIRVANA and describes our assumptions. Sect. 3 describes NIRVANA's construction in detail along with a security analysis, followed by its instantiation in Sect. 4. In Sect. 5 we evaluate NIRVANA's performance and demonstrate its efficiency and scalability. Sect. 6 gives an overview of the related work. Sect. 7 concludes our work.

2 Preliminaries and System Architecture

Throughout this paper, we suppose the security parameter of the scheme to be λ with unary representation of 1^λ, and $\mathsf{negl}(\lambda)$ denotes a negligible function. We use $x \leftarrow\!\!\$\, X$ to denote that x is sampled uniformly from the set X. Also, $|X|$ denotes the cardinality of a set X. $[n]$ denotes the set of integers in the range of 1 through n. We assume a field of prime order $\mathbb{F}$ and denote $\mathbb{F}_{<d}[X]$ as a set of univariate polynomials with degree $< d$. We denote by $y \leftarrow \mathcal{A}(x; r)$ that for a given input of x, and a random string r, we get an output of y. The algorithms are randomized unless explicitly stated otherwise "PPT" refers to "Probabilistic Polynomial Time". Two computationally indistinguishable distributions A and B are denoted by $A \approx_\lambda B$.

Remark 2.1. For ease of reading, the secret values in NIRVANA's construction are represented with a *hat* operator (e.g., $\hat{\mathsf{sk}}$) and masked values are represented with the notation x' for the value x.

Definition 2.1 (Bilinear groups [BF01]). *A* Type-III bilinear group generator[6] $\mathcal{BG}(1^\lambda)$ *returns a tuple* $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathsf{p}, e, g, h)$*, consisting of cyclic Abelian groups* $\mathbb{G}_1$*,* $\mathbb{G}_2$*, and* $\mathbb{G}_T$ *with the same prime order* p*. For given generators of groups* $\mathbb{G}_1$ *and* $\mathbb{G}_2$ *namely* g *and* h*, an efficient non-degenerate Type-III bilinear pairing is denoted by* $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$*, such that,* $\forall\ a, b \in \mathbb{Z}_\mathsf{p} :\ e(g^a, h^b) = e(g^b, h^a) = e(g, h)^{ab}$ *and* $e(g, h) \neq 1_{\mathbb{G}_T}$*.*

Definition 2.2 (Indexed Diffie-Hellman Message Space [SSKP22]). *For a given hash function* H *that is modeled in the random oracle,* $\mathcal{M}^{\mathsf{H}}_{\mathsf{iDH}}$ *is called the indexed DH message space, if we have:*

1. $\forall\ (id, M_1, M_2) \in \mathcal{M}^{\mathsf{H}}_{\mathsf{iDH}}\ \exists\ m \in \mathbb{Z}_\mathsf{p} : \boldsymbol{g} = \mathsf{H}(id), M_1 = \boldsymbol{g}^m, M_2 = h^m$.

[6] No nontrivial homomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$ exists [GPS08].

2. *No two messages use the same index, i.e.,* $\forall\ (id, M_1, M_2) \in \mathcal{M}^{\mathsf{H}}_{\mathsf{iDH}}, (id', M'_1, M'_2) \in \mathcal{M}^{\mathsf{H}}_{\mathsf{iDH}}$, **if** $id = id' \Rightarrow (M_1, M_2) = (M'_1, M'_2) \in \mathcal{M}_{\mathsf{iDH}}$.

More formally, we recall the indexed DH message space generator in Fig. 1, such that $\mathsf{iDH}^{\mathsf{H}} : \mathbb{I} \times \mathbb{Z}_{\mathsf{p}} \to \mathbb{I} \times \mathbb{G}_1 \times \mathbb{G}_2$, where $\mathbb{I}$ is the index space.

$\mathsf{iDH}^{\mathsf{H}}(id, m)$	$\mathsf{H}(id)$
1 : $\boldsymbol{g} \leftarrow \mathsf{H}(id)$	1 : **If** $\mathcal{Q}_{\mathsf{H}}[id] = \perp$:
2 : $M_1 = \boldsymbol{g}^m$	2 : $r \leftarrow\$\, \mathbb{Z}_{\mathsf{p}}$
3 : $M_2 = h^m$	3 : $\mathcal{Q}_{\mathsf{H}}[id] \leftarrow g^r := \boldsymbol{g}$
4 : **return** $(id, M_1, M_2) \in \mathcal{M}^{\mathsf{H}}_{\mathsf{iDH}}$	4 : **return** $\mathcal{Q}_{\mathsf{H}}[id]$

Fig. 1: Indexed Diffie-Hellman Message Space in the ROM.

2.1 Nirvana participants

As shown in Fig. 2, in Nirvana the participants are divided into two groups: authorities $\mathcal{AU}$, and users, which include the customers $\mathcal{C}$, merchants $\mathcal{M}$, and witnesses $\mathcal{W}$.

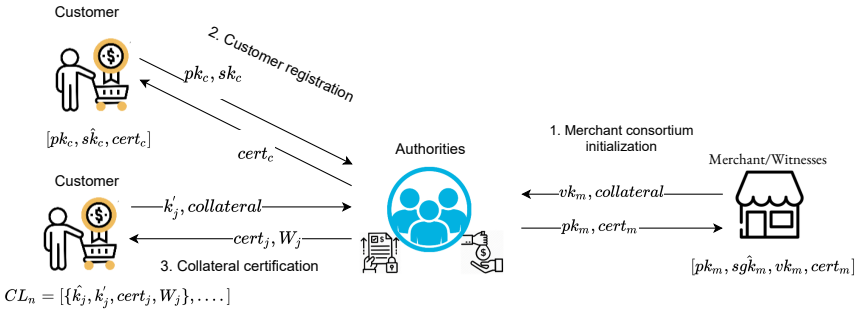

Fig. 2: Nirvana participants and their registration process.

(1) **Authorities:** $\mathcal{AU}$ are responsible for maintaining the system, along with the registration of users and certification of customers' collaterals. $\mathcal{AU}$ pregenerate keys for a fixed number of merchants during setup phase by assuming an upper bound ℓ (e.g., 1M merchants at once).

(2) **Merchant Consortium:** We assume that the merchants in Nirvana form a fixed consortium $\mathcal{M}$ that consists of both large retailers and small shops, where $|\mathcal{M}| \leq \ell$. Once this consortium is formed, new merchants can join until the upper bound set by $\mathcal{AU}$ is reached. After that, $\mathcal{M}$ does not change. Upon joining the consortium, a merchant $\mathsf{M}_m \in \mathcal{M}$ generates a pair of signing/verification keys $(\hat{\mathsf{sgk}}_{bm}, \mathsf{vk}_{bm})$ and sends its verification key vk_{bm} to the $\mathcal{AU}$ along with their collateral. In return, they receive a unique public key pk_{bm} along

with a registration certificate cert_{bm} from the majority of $\mathcal{AU}$. This consortium is assumed to be fixed before customers are allowed to register.

(3) **Customers:** A customer in NIRVANA can be any person who owns some cryptocurrency and is willing to use it to make payments in a retail context. Whenever a customer $\mathsf{C}_n \in \mathcal{C}$, where $|\mathcal{C}| \leq k$, wants to register in NIRVANA, they choose a pair of $(\mathsf{pk}_{cn}, \hat{\mathsf{sk}}_{cn})$, where its secret key $\hat{\mathsf{sk}}_{cn}$ corresponds to its secret identity. Customers register in NIRVANA by providing their public key and requesting a certificate of registration, cert_{cn}, from the majority of authorities.

(4) **Customer collaterals:** A customer C_n can request multiple fixed collaterals in NIRVANA. Each of these collaterals have to be certified by the majority of authorities. A list of collaterals for C_n is recorded in a set of $\mathcal{CL}_n$, and consists of a tuple $(\hat{k}_j, M_j, \mathsf{cert}_j)$ where $\hat{k}_j$ is a unique secret value to redeem the collateral and M_j is an indexed DH message format of the secret value $\hat{k}_j$. Each collateral that is uncertified by the authorities has $\mathsf{cert}_j = \perp$; however, once certified ($\mathsf{cert}_j \neq \perp$) and a list of witnesses ($\mathcal{W}_j$) is appended to it. For honest customers, these collaterals are reusable since they only act as *payment guarantees*.

(5) **Witness:** Each certified collateral is assigned a random list of witnesses ($\mathcal{W} \subseteq \mathcal{M}$) by the $\mathcal{AU}$. Witness $\mathsf{W}_i \in \mathcal{W}$ is responsible to ensure that a collateral is not double-spent by the customers. To do this, each witness locally stores a list of payment guarantees that they encounter within a given epoch.

(a) **Instantiation:** During the setup phase of NIRVANA, upon the establishment of the merchant consortium, each merchant is allocated a uniform interval ($\mathcal{S}_{\mathsf{W}_i}$) and added to the witness set ($\mathsf{W}_i \in \mathcal{W}$). For this we use a cryptographic hash function, $\mathsf{H} : \{0,1\}^* \rightarrow \{0,1\}^\mu$ modeled in the random oracle. Then we have:

$$\bigcap_{i=1}^{w} \mathcal{S}_{\mathsf{W}_i} = \emptyset, \forall\ \mathsf{W}_1 \neq \mathsf{W}_2 \in \mathcal{W} \text{ and } \bigcup_{i=1}^{w} \mathcal{S}_{\mathsf{W}_i} = [0, 2^\mu)\ ,$$

where μ is the bit length output of the hash function.

(b) **Witness selection:** Inspired by witness selection scheme of Osipkov et al. [OVHK07], we assume a reputation system based on proactive responsiveness. $\mathcal{AU}$ maintain a list of intervals for each witness, which changes after every epoch depending on the responsiveness of a witness. More formally, for a given cryptographic hash function $\mathsf{H} : \{0,1\}^* \rightarrow \{0,1\}^\mu$, we assign a witness range $\mathcal{S}_{\mathsf{W}_i} = [s_{\mathsf{W}_i,1}, s_{\mathsf{W}_i,2})$ for a witness $\mathsf{W}_i \in \mathcal{W} \subset [0, 2^\mu)$. This is done such that $\bigcap_{i=1}^{w} \mathcal{S}_{\mathsf{W}_i} > \log_{10}(w)$ and $\bigcup_{i=1}^{w} \mathcal{S}_{\mathsf{W}_i} = [0, 2^\mu)$, where w is the total number of witnesses. Based on their reputation, the witnesses will be assigned larger/smaller ranges that can be updated regularly. As mentioned before, $\mathcal{AU}$ provide a signature to the secret PRF key in the indexed DH message spaces (id, M_1, M_2) s.t. the index is a commitment to $\hat{k}_j$. A re-randomized signature can be provided by the customer as a collateral proof for the merchants. In order to provide real-time double-spending prevention, the authorities also allocate witnesses for each collateral based on a hash function modeled in random oracle.

ⓒ **Witness incentives:** As argued in [OVHK07], preventing double-spending helps the community as a whole and does not leave merchants (also playing the role of witnesses) with unpaid transactions. However, a more concrete motivation could be the financial incentives earned by witnesses for each honest signature they provide. Mavroudis et al. [MWD$^+$20] point out that if a receiving merchant agrees to pay a small percentage (0.5%) of their transaction to the witnesses, they could save on high card payment fees (1.5-5%) for each transaction.

ⓓ **Reputation update:** After every pre-determined epoch, $\mathcal{S}_{\mathsf{W}_i}$ for $i \in \{1, \ldots, \mathcal{W}\}$ is updated by $\mathcal{AU}$ such that it reflects the responsiveness of each W_i in the previous epoch. For a more responsive W_i, the interval is increased, and similarly for a less responsive W_i, the interval is decreased. This is done while satisfying $\bigcap_{i=1}^{w} \mathcal{S}_{\mathsf{W}_i} = \emptyset$, $\forall\ \mathsf{W}_1 \neq \mathsf{W}_2 \in \mathcal{W}$ and $\bigcup_{i=1}^{w} \mathcal{S}_{\mathsf{W}_i} = [0, 2^{\mu})$.

Remark 2.2 (Merchant/Witness collateral). As merchants are required to also take on the role of witnesses, $\mathcal{AU}$ require them to deposit collaterals upon registration in the consortium. Like Snappy [MWD$^+$20], each merchant is required to deposit a collateral equal to the total payments that all merchants in the consortium expect to receive within a given confirmation period, also known as epoch. This deposit (admittedly high for smaller merchants) is mandatory to maintain the security of system.

Remark 2.3. It is important to note that due to the random selection of witnesses from the set of merchants, it is possible that a merchant is chosen to be their own witness. However, this does not cause any issues to our security, as an incorrect signature from a witness (in this case the receiving merchant itself), leads to slashing of their deposit. We also assume that the merchant has no financial incentive to cheat in this scenario.

Remark 2.4. In Nirvana, the majority of the authorities randomly assign a list that consists of a logarithmic number of witnesses to each collateral in order to proactively detect double-spending of a payment guarantee during an epoch. This list consists of the identities of witnesses along with a signature from authorities. This signature acts as a *proof of responsibility* for the assigned witnesses. The list of assigned witnesses needs to be public since the payee merchant is required to forward the received payment guarantee to each witness in the list. However, due to this public witness list, two payment guarantees become identical to an observer; making Nirvana transactions done by the same customer linkable. To overcome this issue, a customer appends a list of dummy witnesses of size poly-logarithmic in the order of the size of Nirvana's merchant consortium to the assigned witnesses. This is done such that the proof of responsibility is still verifiable by each assigned witness; while all unassigned witnesses discard the guarantee. The list of dummy witnesses can be altered in each transaction, which makes it hard to trace and link two transactions done by the same customer.

2.2 Threat model and assumptions

In Nirvana, we assume active adversaries, meaning the adversary can take malicious actions during the security game to disrupt the protocol. We assume that the customers and merchants are fully malicious and actively try to perform double-spending and generate fake proofs of double-spending respectively. During a transaction, the adversary can control an arbitrary number of customers and merchants except the target merchant accepting the payment guarantee. We assume that at least 50% of $\mathcal{AU}$ are honest. This set of authorities consists of Nirvana itself along with a group of merchants. The policy for selection of this group of merchants is beyond the scope of this paper; however, a possible policy could be to select the group of merchants with the most transactions in the network and if required, randomly change this set of merchants for fairness [CGMV18]. Real life examples such as Facebook's Diem [Fac21] show that implementing such a consortium is possible. We assume side-channel attack [BP15,TBP20] resistance to be important yet orthogonal to our work. Finally, we also assume that the cryptocurrency of the customers choice is reliable; however, we do not explicitly assume that the chosen cryptocurrency supports private transactions. As a workaround, customers can utilise mixers like CoinJoin [Max13] that provide a layer of privacy to public cryptocurrencies such as Bitcoin.

Our system is designed with retail markets as its primary use-case; however, it can be used for any use-case where the customer userbase is larger than the merchant userbase. Nirvana assumes a fixed network of known merchants in the system who form a consortium. Similar to [MWD+20], the merchants in the Nirvana consortium do not accept any risk, and only provide services when they are *guaranteed* a payment. The consortium merchants have permanent and reliable network connections, and only accept payment guarantees when they are online. Customers are not expected to be online, unless transacting with a merchant. We also assume that the merchants in the Nirvana consortium who also play the part of witnesses for customer collaterals are fairly responsive and stay online when incentivized to do so.

Epoch: In Nirvana, time is divided into certain intervals of predetermined length that start from the time a customer performs a payment, called epochs. The choice of this epoch duration is a design choice, which can be defined with the transaction confirmation latency of public cryptocurrencies in the worst case. Each witness deletes payment guarantees older than the present epoch from their local storage for storage efficiency, since customer collaterals are reusable.

Collateral assumptions: Nirvana assumes a fixed, reusable collateral, the amount of which depends on how much money the customer plans to spend during an epoch. For instance, a Bitcoin transaction takes $\sim$ 60 minutes to be finalized, hence the amount of Nirvana's fixed collateral is decided to be sufficient for this latency period. For all honest customers, these collaterals can be used as payment guarantees again in the next epoch. Revocation of a collateral can be easily handled by putting the publicly available blinded certificate in a revoked list or a blacklist depending on whether an honest customer churns

or a malicious customer is banned from Nirvana. This list can be checked by merchants when they accept these blinded certificates as guarantees of payments. Additionally, Peeters and Pashalidis [PP13] show how the checking of these blacklists can also be done in a privacy-preserving way.

Communication channel: For Nirvana we assume perfect synchronicity, so there is a strict time limit between rounds. Also we assume the existence of *secure channels* for all the communications within the protocol, thus an adversary cannot overhear or tamper with the transferred messages [BCFK15].

3 Nirvana: Generic Construction

In this section, we give an overview of Nirvana's protocol, and later formalize its generic construction. Finally, we prove the security properties of Nirvana.

3.1 Protocol overview

In order to explain Nirvana protocol, we assume that the set of authorities is already established and the merchants in the network have registered and formed a consortium. As illustrated in Fig. 3, the protocol between a customer and a merchant proceeds as follows:

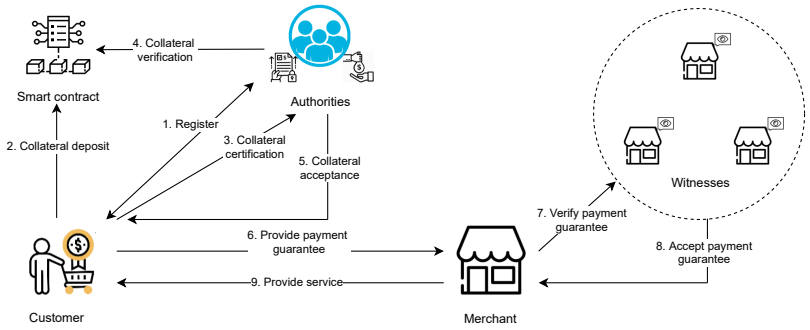

Fig. 3: Nirvana protocol overview.

(1) In order to avail the services of Nirvana, the new customer begins by registering themselves with the majority of authorities. (2) After successfully registering in the network, the customer deposits a collateral in Nirvana's smart contract. (3) Once the deposit is confirmed on the blockchain where Nirvana's smart contract is deployed, the customer asks for a collateral certification, which is necessary to prove that the customer owns some collateral in Nirvana. This proof is used to generate payment guarantees for merchants. (4) Upon receiving the certification request of the customer, the authorities check the smart contract to confirm if the customer deposited a collateral. (5) Once confirmed, the authorities provide the customer with a signed certificate of their collateral's existence in the network. Along with this confirmation, the customer is also assigned a

signed list of witnesses in charge of tracking the collateral usage. (6) During a transaction in the retail market, the customer makes a payment to the target merchant using the cryptocurrency of their choice. Once the payment is in *pending* state on the blockchain, the customer generates a *payment guarantee* by using the certified collateral and sends this to the target merchant along with the assigned list of witnesses and the transaction hash of their cryptocurrency payment. (7) The target merchant forwards this *payment guarantee* to the assigned witnesses who individually confirm that they have not seen a similar guarantee in the current epoch. (8) Upon confirmation, each witness returns a signed payment guarantee to the merchant. (9) On receiving signed payment guarantees from a majority of the witnesses, the merchant aggregates these signatures and accepts the payment guarantee and provides the customer with necessary services/products.

3.2 Formal construction

Nirvana relies on cryptography to achieve its goals. More precisely, it builds on Pseudo-Random Function (PRF), Non-Interactive Zero-Knowledge (NIZK) arguments, Signatures (S), Structure-Preserving Threshold Signatures (SPTS), Commitments (CO), and a novel randomness-reusable Threshold Encryption in the bilinear pairing groups (rbTE). We list the formal definition and the underlying security properties for the listed primitives in App. B and outline the scheme in Algo. 1 for security parameter 1^λ and a family of collision-resistant hash functions $\mathcal{H}$. The list of master public keys is considered as an implicit input for all algorithms except the parameter generation algorithm. We now formalise the functions in the Algo. 1 as follows:

- $\mathsf{PGen}(1^\lambda, \mathcal{H}, \mathbf{R_L})$ is a PPT algorithm that takes security parameter 1^λ, a family of cryptographic hash functions $\mathcal{H} : \{0,1\}^* \to \{0,1\}^\mu$ and a relation $\mathbf{R_L}$ as inputs. It returns the master public key mpk as output. All the following functions implicitly take these data as inputs.
- $\mathsf{AuKeyGen}(\mathcal{AU})$ is a distributed PPT algorithm that is executed by the group of authorities $\mathcal{AU}$ that returns the pair of signing/verification keys $(\hat{\mathsf{sgk}}_i, \mathsf{vk}_i)$ for $1 \le i \le n$ along with a global verification key vk_a. $\mathcal{AU}[\hat{\mathsf{sgk}}_{ai}, \mathsf{vk}_{ai}, \mathsf{vk}_a]_{i=1}^n$ represents the list of credentials for $\mathcal{AU}$.
- $\mathsf{MKeyGen}(\mathsf{M}_m)$ is a PPT algorithm that is executed by merchant $\mathsf{M}_m \in \mathcal{M}$ in order to join the network. It initially generates a pair of signing/verification keys $(\hat{\mathsf{sgk}}_{bm}, \mathsf{vk}_{bm})$ and returns a tuple $(\hat{\mathsf{sgk}}_{bm}, \mathsf{vk}_{bm}, \mathsf{pk}_{bm} = \perp, \hat{\mathsf{cert}}_{bm} = \perp)$. The list of keys belonging to the group of merchants is recorded in $\mathcal{M}[\hat{\mathsf{sgk}}_{bi}, \mathsf{vk}_{bi}]_{i=1}^\ell$.
- $\mathsf{MRegister}(\mathcal{AU}[\hat{\mathsf{sgk}}_{ai}]_{i=1}^t, \mathcal{M}[\mathsf{vk}_{bi}]_{i=1}^\ell)$ is a distributed PPT algorithm that is run by any subset of $\mathcal{AU}$ of size at least t to register the merchants who deposit a collateral to join Nirvana's merchant consortium and assign them a public key pk_{bm}. For each merchant $\mathsf{M}_m \in \mathcal{M}$, it takes the secret signing key of the authorities $\mathcal{AU}[\hat{\mathsf{sgk}}_{ai}]$ for $1 \le i \le t$, and returns a pair $(\mathsf{pk}_{bm}, \hat{\mathsf{cert}}_{bm})$ along with an interval $\mathcal{S}_m$ that is used for witness

selection. After this phase, the list of parameters for the m^{th} merchant can be updated as $\mathcal{M}[\hat{\mathsf{sgk}}_{bm}, \mathsf{vk}_{bm}, \mathsf{pk}_{bm}, \hat{\mathsf{cert}}_{bm}, \mathcal{S}_m]$.

- $\mathsf{CuKeyGen}(\mathsf{C}_n)$ is a PPT algorithm run by the customers, that for each customer $\mathsf{C}_n \in \mathcal{C}$, generates an initial secret key $\hat{\mathsf{sk}}_{cn}$ and its corresponding public key pk_{cn}. It returns a tuple of $(\mathsf{pk}_{cn}, \hat{\mathsf{sk}}_{cn}, \hat{\mathsf{cert}}_{cn} = \perp)$ along with a zero-knowledge proof π_{cn} to prove that pk_{cn} is well-formed. The list of customers' keys are kept in $\mathcal{C}[\hat{\mathsf{sk}}_{ci}, \mathsf{pk}_{ci}, \hat{\mathsf{cert}}_{ci} = \perp]_{i=1}^{k}$.
- $\mathsf{CuRegister}(\mathcal{AU}[\hat{\mathsf{sgk}}_{ai}]_{i=1}^{t}, \mathcal{C}[\mathsf{pk}_{cn}], \pi_{cn})$ is a distributed PPT algorithm executed by any subset of $\mathcal{AU}$ of size at least t to certify the public key pk_{cn} of a customer C_n corresponds to some secret value $\hat{\mathsf{sk}}_{cn}$ under a proof π_{cn}. Once the customer C_n is registered by the authorities, it receives a certificate $\hat{\mathsf{cert}}_{cn}$ and it updates $\mathcal{C}[\mathsf{pk}_{cn}, \hat{\mathsf{sk}}_{cn}, \hat{\mathsf{cert}}_{cn}]$.
- $\mathsf{CuCreate}(\mathcal{C}[\hat{\mathsf{cert}}_{cn}])$ is a PPT algorithm executed by the customer to request for certification of their collateral. A successfully registered customer C_n, with certificate $\hat{\mathsf{cert}}_{cn} \neq \perp$, can deposit collaterals in Nirvana. For each deposit j in Nirvana's smart contract, the customer samples a random value k_j from a uniform distribution $\mathcal{K}$ in a way that the deposit is not directly linkable to the customer. Then it returns a tuple $\mathcal{CL}[\hat{k}_j, M_j]$ as an uncertified collateral along with a proof of knowledge of $\hat{k}_j$ and the fact that it belongs to $\mathcal{K}$ and $M_j \leftarrow \mathsf{iDH}^{\mathsf{H}}(k'_j, \hat{k}_j)$, where k'_j is the commitment on $\hat{k}_j$.
- $\mathsf{AuCreate}(\mathcal{AU}[\hat{\mathsf{sgk}}_{ai}]_{i=1}^{t}, \mathcal{C}[\mathsf{cert}'_{cn}], \pi_{\mathsf{cert}}, \mathcal{CL}[M_j], \pi_j)$ is a distributed PPT algorithm executed by a group of authorities $\mathcal{AU}$ of size at least t. It takes the authorities' secret signing keys $(\hat{\mathsf{sgk}}_{ai})$, an indexed DH message space of PRF key and a re-randomized certificate (cert'_n) as inputs. To create a certified collateral, it checks the validity of the proof π_j and whether this collateral exists in the smart contract, and returns the certificate $\hat{\mathsf{cert}}_j$ along with the signed witness list $(\mathsf{W}_j, \sigma_{\mathsf{W}_j})$. The list of parameters for each collateral is kept as $\mathcal{CL}[\hat{k}_j, \hat{k}'_j, \hat{\mathsf{cert}}_j, \mathsf{W}_j, \sigma_{\mathsf{W}_j}]$. Note that no $\mathcal{SPTS}.\mathsf{Recon}(.)$ algorithm is operated upon, and a user may execute this algorithm if it receives at least t partial signatures from the authorities.
- $\mathsf{Spend}(\mathcal{C}[\hat{\mathsf{sk}}_{cn}, \hat{\mathsf{cert}}_{cn}], \mathcal{CL}[\hat{k}_j, \hat{\mathsf{cert}}_j, \mathsf{W}_j, \sigma_{\mathsf{W}_j}], \mathcal{M}[\mathsf{pk}_{bm}], t)$ is a PPT algorithm that is executed by a customer $\mathsf{C}_n \in \mathcal{C}$ who performs a payment to the merchant $M_m \in \mathcal{M}$ using the cryptocurrency of their choice at time t. The registered customer uses a certified collateral $\mathcal{CL}[\hat{k}_j, \hat{\mathsf{cert}}_j, \mathsf{W}_j, \sigma_{\mathsf{W}_j}]$ to provide a payment guarantee to the merchant M_m. It returns the transaction details as a list of parameters $\mathcal{T}[\mathsf{x}_m, \pi_m, R_t]$, which contains a pair of instance (x_m) and proof (π_m), along with a set of auxiliary data R_t. A customer can use different collaterals that they own in Nirvana and combine them if needed by using the aggregation algorithm in our SPTS construction.
- $\mathsf{Vf}(\mathcal{M}[\mathsf{pk}_{bm}], \mathcal{T}[\pi_m, \mathsf{x}_m, R_t], t)$ is a deterministic algorithm executed by the merchant $\mathsf{M}_m \in \mathcal{M}$ to check the validity of a received payment guarantee. If the proof is verified successfully and the majority of witnesses confirm that they have not seen a similar payment guarantee in the current epoch (by providing their signatures), the merchant verifies their individual signatures and aggregates them. Once the aggregation is complete, the merchant provides the items/services to the customer without waiting for the transaction confirmation of the customer's

original payment on the blockchain. If the proof verification fails, or the majority of witnesses do not confirm the guarantee, the merchant rejects the payment guarantee.

Algorithm 1: NIRVANA: Generic Construction.

Function $\texttt{PGen}(1^\lambda, \mathcal{H}, \mathbf{R_L})$:
- $\mathsf{H} \leftarrow\$ \mathcal{H}$
- $(\vec{\mathsf{crs}}, \hat{\mathsf{ts}}, \hat{\mathsf{te}}) \leftarrow \mathcal{ZK}.\mathsf{K}_{\vec{\mathsf{crs}}}(1^\lambda, \mathbf{R_L})$
- $(\mathsf{pp}) \leftarrow \mathcal{SPTS}.\mathsf{Setup}(1^\lambda)$
- **return** $\mathsf{mpk} := (\mathsf{pp}, \mathsf{H}, \vec{\mathsf{crs}})$

Function $\texttt{AuKeyGen}(\mathcal{AU})$:
- $(\vec{\mathsf{sgk}}_a, \vec{\mathsf{vk}}_a, \mathsf{vk}_a) \leftarrow \mathcal{SPTS}.\mathsf{KGen}(\mathsf{mpk}, n, t)$
- **return** $(\mathcal{AU}[\hat{\mathsf{sgk}}_{ai}, \mathsf{vk}_{ai}, \mathsf{vk}_a]_{i=1}^n)$

Function $\texttt{MKeyGen}(\mathsf{M}_m)$:
- $(\hat{\mathsf{sgk}}_{bm}, \mathsf{vk}_{bm}) \leftarrow \mathcal{DS}.\mathsf{KGen}(\mathsf{pp})$
- **return** $(\mathcal{M}[\hat{\mathsf{sgk}}_{bm}, \mathsf{vk}_{bm}])$

Function $\texttt{MRegister}(\mathcal{AU}[\hat{\mathsf{sgk}}_{ai}]_{i=1}^t, \mathcal{M}[\mathsf{vk}_{bi}]_{i=1}^\ell)$:
- **for** $j \in \mathrm{range}(\ell)$ **do**
  - $(\mathsf{pk}_{bj}, \mathsf{pk}_b) \leftarrow rb\mathcal{TE}.\mathsf{KGen}(\mathsf{mpk}, \ell, t, 2)$
  - $S_j \leftarrow \mathsf{H}(\mathcal{AU}[\hat{\mathsf{sgk}}_{ai}]_{i=1}^t, \mathsf{vk}_{bj})$
- **return** $(\mathcal{M}[\mathsf{pk}_{bi}]_{i=1}^\ell, [\mathcal{S}_i]_{i=1}^\ell, \mathsf{pk}_b)$

Function $\texttt{CuKeyGen}(\mathsf{C}_n)$:
- $\hat{\mathsf{sk}}_{cn} \leftarrow\$ \mathbb{Z}_\mathsf{p}^*$
- $\mathsf{pk}_{cn} = g^{\hat{\mathsf{sk}}_{cn}}$
- $\pi_{cn} \leftarrow \mathrm{PoK}\{(\hat{\mathsf{sk}}_{cn}) : \mathsf{pk}_{cn} := g^{\hat{\mathsf{sk}}_{cn}}\}$
- **return** $(\mathcal{C}[\mathsf{pk}_{cn}, \hat{\mathsf{sk}}_{cn}], \pi_{cn})$

Function $\texttt{CuRegister}(\mathcal{AU}[\hat{\mathsf{sgk}}_{ai}]_{i=1}^t, \mathcal{C}[\mathsf{pk}_{cn}], \pi_{cn})$:
- **if** $\mathcal{ZK}.\mathsf{Vf}(\vec{\mathsf{crs}}, \pi_{cn}, \mathsf{pk}_{cn}) == 1$ **then**
  - $(\hat{\mathsf{cert}}_{cn}) \leftarrow \mathcal{SPTS}.\mathsf{Sign}(\mathcal{AU}[\hat{\mathsf{sgk}}_{ai}]_{i=1}^t, \mathsf{pk}_{cn})$
- **return** $(\mathcal{C}[\hat{\mathsf{cert}}_{cn}])$

Function $\texttt{CuCreate}(\mathcal{C}[\hat{\mathsf{sk}}_{cn}, \hat{\mathsf{cert}}_{cn}])$:
- $\hat{k}_j \leftarrow \mathcal{PRF}.\mathsf{KGen}(\mathsf{pp})$
- $k'_j \leftarrow \mathcal{CO}.\mathsf{Com}(\mathsf{pp}, \hat{k}_j)$
- $\pi_j \leftarrow \mathrm{PoK}\{(\hat{k}_i) : k'_j := \mathsf{Com}(\hat{k}_j)\}$
- $M_j := (k'_j, M_1, M_2) \leftarrow \mathsf{iDH}^\mathsf{H}(k'_j, k_j)$
- $\mu \leftarrow\$ \mathbb{Z}_\mathsf{p}^*$
- $(\mathsf{cert}'_{cn}) \leftarrow \mathcal{SPTS}.\mathsf{Randz}(\hat{\mathsf{cert}}_{cn}; \mu)$
- $\pi_{\mathsf{cert}} \leftarrow \mathrm{PoK}\{(\mu, \mathsf{pk}_{cn}) : \mathcal{SPTS}.\mathsf{Vf}(\mathsf{cert}'_{cn}) = 1\}$
- **return** $(\mathcal{CL}[\hat{k}_j, M_j], \pi_j, \mathcal{C}[\mathsf{cert}'_{cn}], \pi_{\mathsf{cert}})$

Function $\texttt{AuCreate}(\mathcal{AU}[\hat{\mathsf{sgk}}_{ai}, \mathsf{vk}_a]_{i=1}^t, \mathcal{C}[\mathsf{cert}'_{cn}], \pi_{\mathsf{cert}}, \mathcal{CL}[M_j], \pi_j)$:
- **if** $\mathcal{ZK}.\mathsf{Vf}(\vec{\mathsf{crs}}, \pi_{\mathsf{cert}}, \mathsf{cert}'_{cn}) == 1 \ \wedge \ \mathcal{ZK}.\mathsf{Vf}(\vec{\mathsf{crs}}, \pi_j, k'_j) == 1$ **then**
  - $(\hat{\mathsf{cert}}_j) \leftarrow \mathcal{SPTS}.\mathsf{Par\text{-}Sign}(\mathcal{AU}[\hat{\mathsf{sgk}}_{ai}]_{i=1}^t, M_j)$
  - $\mathsf{W}_j \leftarrow \mathsf{H}(\hat{\mathsf{cert}}_j)$
  - $(\sigma_{\mathsf{W}_j}) \leftarrow \mathcal{SPTS}.\mathsf{Par\text{-}Sign}(\mathcal{AU}[\hat{\mathsf{sgk}}_{ai}]_{i=1}^t, \mathsf{W}_j)$
- **return** $(\mathcal{CL}[\hat{\mathsf{cert}}_j, \mathsf{W}_j, \sigma_{\mathsf{W}_j}])$

Function $\texttt{Spend}(\mathcal{C}[\hat{\mathsf{cert}}_{cn}, \hat{\mathsf{sk}}_{cn}], \mathcal{CL}[\hat{k}_j, \hat{\mathsf{cert}}_j, \mathsf{W}_j, \sigma_{\mathsf{W}_j}], \mathcal{M}[\mathsf{pk}_{bm}], t)$:
- $\mu \leftarrow\$ \mathbb{Z}_\mathsf{p}^*$
- $(\mathsf{cert}'_j) \leftarrow \mathcal{SPTS}.\mathsf{Randz}(\hat{\mathsf{cert}}_j; \mu)$
- $(\sigma'_j) \leftarrow \mathcal{SPTS}.\mathsf{Randz}(\sigma_j; \mu)$
- $(r_t) \leftarrow \mathsf{PRF}_{\hat{k}_j}(t)$
- $R_t := e(r_t, h)$
- $\mathtt{Ct}_m \leftarrow rb\mathcal{TE}.\mathsf{Enc}(\mathsf{pk}_{bm}, \hat{\mathsf{sk}}_{cn}; r_t)$
- $\hat{\mathsf{w}}_m = (\hat{\mathsf{cert}}_j, \mu, \hat{k}_j, \mathsf{W}_j, \sigma_{\mathsf{W}_j}, r_t, \hat{\mathsf{sk}}_{cn})$
- $\mathsf{x}_m = (\mathsf{cert}'_j, \mathtt{Ct}_m, \mathsf{W}'_j, \sigma'_{\mathsf{W}_j})$
- $\pi_m \leftarrow \mathcal{ZK}.\mathsf{P}(\mathbf{R_L}, \vec{\mathsf{crs}}, \mathsf{x}_m, \hat{\mathsf{w}}_m)$
- **return** $(\mathcal{T}[\pi_m, \mathsf{x}_m, R_t])$

Function $\texttt{Vf}(\mathcal{M}[\mathsf{pk}_{bm}], \mathcal{T}[\pi_m, \mathsf{x}_m, R_t], t)$:
- **if** $\mathcal{ZK}.\mathsf{Vf}(\mathbf{R_L}, \vec{\mathsf{crs}}, \mathsf{x}_m, \pi_m) == 1$ **then**
  - **for** $i \in \mathsf{W}_i$ **do**
    - **if** $\mathcal{SPTS}.\mathsf{Vf}(\mathsf{vk}_a, \sigma'_{\mathsf{W}_i}, R_t) == 1 \ \wedge \ R_t \notin \mathcal{L}_i$ **then**
      - $(\sigma_{R_t}) \leftarrow \mathcal{DS}.\mathsf{Sign}(\hat{\mathsf{sgk}}_{bi}, R_t)$
  - **if** $\mathcal{DS}.\mathsf{Vf}(\mathsf{vk}_{\mathsf{W}_i}, \sigma_{R_t}) == 1 \ \wedge \ |\sigma_{R_i}| \geq \log(w)/2$ **then**
    - **return** 1

Function $\texttt{RevealID}(\mathtt{Ct}_m, \mathtt{Ct}_{m'}, v)$:
- $(\hat{\mathsf{sk}}_n) \leftarrow rb\mathcal{TE}.\mathsf{Dec}(\mathtt{Ct}_m, \mathtt{Ct}_{m'}, v)$
- **return** (sk_n)

– $\mathsf{RevealID}(\mathtt{Ct}_m, \mathtt{Ct}_{m'}, v)$ is a deterministic algorithm that takes two ciphertexts $\mathtt{Ct}_m$ and $\mathtt{Ct}_{m'}$ generated under the public key of two distinct merchants m and m' and returns the plaintext, i.e., the ID of the customer and the secret to their collateral $\hat{k}_j$. This ID can be used by $\mathcal{AU}$ to blacklist the cheating customer and the secret $\hat{k}_j$ can be used to remunerate the victim merchant.

3.3 Security Analysis

Next we formally define the two main security requirements for NIRVANA, namely (1) Anonymity of honest customers and Unlinkability of payment guarantees, and (2) Payment certainty. Later, we formally prove that the proposed generic construction described in Algo. 1 satisfies these requirements. In the following definitions, it is implicitly assumed that there exists a PPT adversary $\mathcal{A}$ who has access to the following oracles provided by the challenger $\mathcal{B}$:

– **Oracle** $\mathcal{O}_{\mathsf{AuCorrupt}}(.)$: Adversary $\mathcal{A}$, by having access to this oracle, can corrupt at most $t-1$ authorities and receive their internal states. The set of corrupted authorities is denoted by $\mathcal{AU}'$.
– **Oracle** $\mathcal{O}_{\mathsf{CuCorrupt}}(.)$: Adversary $\mathcal{A}$ can corrupt any customer $\mathsf{C}_n \in \mathcal{C}$ by querying this oracle, and receive its uncertified secret key $\hat{\mathsf{sk}}_{cn}$. The list of corrupted customers is denoted by $\mathcal{C}'$.
– **Oracle** $\mathcal{O}_{\mathsf{ColCorrupt}}(.)$: $\mathcal{A}$ can corrupt at most q_D collaterals $CL_j \in \mathcal{CL}$ to receive their uncertified secret value $\hat{k}_j$. The list of corrupted collaterals is represented by $\mathcal{CL}'$.
– **Oracle** $\mathcal{O}_{\mathsf{MCorrupt}}(.)$: Adversary $\mathcal{A}$ can corrupt any merchant $\mathsf{M}_m \in \mathcal{M}$ and receive its uncertified public key pk_{bm}. The list of corrupted merchants is denoted by $\mathcal{M}'$.
– **Oracle** $\mathcal{O}_{\mathsf{WCorrupt}}(.)$: Adversary $\mathcal{A}$ can corrupt at most $\log_{10}(w)/2 - 1$ witnesses and receive their internal states. The list of corrupted witnesses is denoted by $\mathcal{W}'$.
– **Oracle** $\mathcal{O}_{\mathsf{Revoke}}(.)$: Adversary $\mathcal{A}$ can revoke at most q_R certified collaterals $CL_j \in \mathcal{CL}$ and redeem the deposited money. The list of revoked collaterals is denoted by $\mathcal{CL}_R$.
– **Oracle** $\mathcal{O}_{\mathsf{Spend}}(.)$: Adversary $\mathcal{A}$ can make at most q_S payment guarantees created by any arbitrary non-corrupted customer to any non-corrupted merchant.
– **Oracle** $\mathcal{O}_{\mathsf{Vf}}(.)$: Adversary $\mathcal{A}$ has access to this oracle to check the validity of a payment guarantee.

Definition 3.1 (Payment Unlinkability and Anonymity). NIRVANA *preserves the anonymity of honest customers and provides unlinkability of payment guarantees, if no PPT adversary* $\mathcal{A}$ *by getting access to* $\mathcal{O}_{\mathsf{AuCorrupt}}, \mathcal{O}_{\mathsf{CuCorrupt}}, \mathcal{O}_{\mathsf{ColCorrupt}}, \mathcal{O}_{\mathsf{MCorrupt}}, \mathcal{O}_{\mathsf{Revoke}}, \mathcal{O}_{\mathsf{Spend}}, \mathcal{O}_{\mathsf{Vf}}$ *oracles,* $\mathcal{O}_{\mathsf{ANON}}$ *in short, and with advantage of* $Adv_{\mathcal{A}}^{\mathrm{ANON}}(\lambda, b) = 2\left((\mathrm{Exp}_{\mathcal{A}}^{\mathrm{ANON}}(1^{\lambda}, b) = 1) - 1/2\right)$*, has a non-negligible chance of winning the following experiment, i.e.,*

$$\left| Adv_{\mathcal{A}}^{\mathrm{ANON}}(\lambda, b = 0) - Adv_{\mathcal{A}}^{\mathrm{ANON}}(\lambda, b = 1) \right| \leq \mathsf{negl}(\lambda) \ .$$

The only restriction here is that the adversary cannot query $\mathcal{O}_{\mathsf{Spend}}(.)$ oracle on the challenge collaterals and secret identities.

$\textsc{Exp}^{\textsc{anon}}_{\mathcal{A}}(1^\lambda, b)$

$\left(\mathsf{mpk}, \mathsf{H}, \mathcal{AU}[\hat{\mathsf{sgk}}_{ai}, \mathsf{vk}_{ai}, \mathsf{vk}_a]_{i=1}^n, \mathcal{M}[\hat{\mathsf{sgk}}_{bi}, \mathsf{vk}_{bi}, \mathsf{pk}_{bi}]_{i=1}^\ell\right) \leftarrow \mathsf{AllGen}(1^\lambda, \mathcal{H}, \mathbf{R_L})$
$\left(\mathsf{M}_m, \hat{\mathsf{sk}}_0^*, \hat{\mathsf{sk}}_1^*, \hat{k}_0^*, \hat{k}_1^*\right) \leftarrow \mathcal{A}^{\mathcal{O}_{\textsc{anon}}}(\mathsf{mpk})$
$b \leftarrow\$ \{0,1\}$
$(\pi_b, \mathsf{x}_b, R_{t,b}) \leftarrow\$ \mathsf{Spend}\left(\mathcal{C}[\hat{\mathsf{cert}}_b^*, \hat{\mathsf{sk}}_b^*], \mathcal{CL}[\hat{k}_b^*, \hat{\mathsf{cert}}_b^*], \mathcal{M}[\mathsf{pk}_{bm}], t\right)$
$b' \leftarrow\$ \mathcal{A}^{\mathcal{O}_{\textsc{anon}}}(\mathcal{T}[\pi_b, \mathsf{x}_b, R_{t,b}])$
if $b' == b$:
  return 1
else return 0

More precisely, this property enforces that no PPT adversary can expose any information about the transaction such as the identity of honest customers or be able to link it to any other transaction made by the customer. In reality, the adversary could choose any two challenge collaterals with different lists of witnesses, allowing him to distinguish between them by checking this list. Therefore, in order to ensure that such trivial attacks are excluded, we quantified unlinkability and privacy over all valid adversaries such that both challenge collaterals have the same list of witnesses. As we discussed earlier in Remark 2.4, a secondary set of witnesses is added to the main list in order to increase the probability of this event occurring in the actual game.

Definition 3.2 (Payment Certainty). Nirvana *provides payment certainty (PC) if no transaction τ is approved with a non-negligible advantage such that $q_S + q_R + \tau > q_D$. No PPT adversary $\mathcal{A}$ with access to $\mathcal{O}_{\mathsf{AuCorrupt}}, \mathcal{O}_{\mathsf{CuCorrupt}}, \mathcal{O}_{\mathsf{ColCorrupt}}, \mathcal{O}_{\mathsf{MCorrupt}}, \mathcal{O}_{\mathsf{Revoke}}, \mathcal{O}_{\mathsf{Spend}}, \mathcal{O}_{\mathsf{Vf}}$ oracles, $\mathcal{O}_{\mathsf{PC}}$ in short, can win the following experiment with a non-negligible advantage in λ and we can write,*

$$Adv^{\textsc{pc}}_{\mathcal{A}}(\lambda) := \Pr[\textsc{Exp}^{\textsc{pc}}_{\mathcal{A}}(1^\lambda) = 1] \leq \mathsf{negl}(\lambda) \ .$$

$\textsc{Exp}^{\textsc{pc}}_{\mathcal{A}}(1^\lambda)$

$\left(\mathsf{mpk}, \mathsf{H}, \mathcal{AU}[\hat{\mathsf{sgk}}_{ai}, \mathsf{vk}_{ai}, \mathsf{vk}_a]_{i=1}^n, \mathcal{M}[\hat{\mathsf{sgk}}_{bi}, \mathsf{vk}_{bi}, \mathsf{pk}_{bi}]_{i=1}^\ell\right) \leftarrow \mathsf{AllGen}(1^\lambda, \mathcal{H}, \mathbf{R_L})$
$(\pi^*, \mathsf{x}^*, R_t^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\textsc{pc}}}(\mathsf{mpk})$
if $\mathcal{ZK}.\mathsf{Vf}(\mathbf{R_L}, \vec{\mathsf{crs}}, \pi^*, \mathsf{x}^*) == 1 \ \wedge \ q_D < q_S + q_R$:
  return 1
else return 0

More precisely, NIRVANA provides payment certainty if no entity, not even after colluding with a group of participants, can transfer and/or revoked more money than the amount deposited.

Remark 3.1 (Auditing and Double-Spend Detection). In a secure and anonymity preserving payment guarantee system with the above mentioned properties, if a malicious customer attempts to use one collateral to pay multiple merchants at the same time, NIRVANA's construction ensures that the cheating entity is caught and their identity is revealed to the network.

Theorem 3.1. *If the Pseudo-Random Function is correct and weakly-robust, the signature and Structure-Preserving Threshold signature schemes are existentially unforgeable, the commitment scheme is computationally hiding and binding, the randomness-reusable threshold encryption is statistically semantic secure, the NIZK proof is zero-knowledge and knowledge sound, and the hash function is built in the random oracle model then the proposed generic construction in Algo. 1 describes an unlinkable, anonymity-preserving, and certain payment guarantee method as defined in Section 3.*

Proof. The proof can be found in App. C. □

4 Nirvana: An Efficient Instantiation

Until now, we presented the generic construction of an overlay network that provides instant, anonymous and unlinkable payment guarantees. The implementation of our construction completely relies on the use-case; however, in this section we aim to demonstrate the efficiency of NIRVANA's generic construction in providing payment guarantees to merchants by selecting specific cryptographic primitives along with the reasons for our choice as follows:

Pseudo-Random Function: In order to make customers' collaterals reusable, we utilise a weakly-robust PRF proposed by Dodis and Yampolskiy [DY05]. Let a cyclic group $\mathbb{G}$ of prime order p with a generator g. The PRF on input $x \in \mathbb{Z}_{\mathsf{p}}$ under key space $\mathbb{Z}_{\mathsf{p}}$ is defined by $\mathsf{PRF}_k(x) = g^{1/(k+x)}$, where $k \leftarrow\!\$\, \mathbb{Z}_{\mathsf{p}}$ is a secret key.

In NIRVANA, the input x is the time of payment occurrence (which is verified by the target merchants). This makes the proof of knowledge of the secret key Schnorr-friendly. However, our generic construction requires $\mathcal{AU}$ to sign an indexed DH message of this key as a masked PRF key for privacy, as a malicious customer can generate multiple fake proofs and the signature provides non-repudiation of the collateral's origin.

Structure-Preserving Threshold Signature (SPTS): As mentioned above, in order to verify the validity of customers' collaterals, our work adopts SPS [AFG+10] (see App. B.3).

Since NIRVANA is designed with the main idea of preserving the anonymity of honest customers and providing unlinkability of transactions in its network, we require unlinkable signatures, such as blind signatures [Cha82]. However, we observe that NIRVANA differs from existing e-cash schemes [CFN90,FY93,CHL05], since it does not provide actual payments but only a guarantee of payments happening in parallel (on the underlying blockchain). Hence, we require the signatures to be reusable in our use-case, and blind signatures do not achieve this property [CDHK15]. Another motivation for our choice of SPS signatures is their compatibility with Groth-Sahai proof systems [GS08], which enables efficient verification of collaterals in zero knowledge.[7] Although the SPS relies on a single authority and does not meet our requirements, recently, Sedaghat et al. [SSKP22] defined the notion of (non-interactive) SPTS and proposed an efficient instantiation over indexed Diffie-Hellman message spaces, stated in Def. 2.2. In a (n,t)-SPTS scheme (see App. B.3), the secret signing key is distributed among n authorities and the generation of any signature requires the cooperation of a subset of authorities of size at least t. Moreover, any adversary that learns $t-1$ or fewer shares of the secret key cannot forge a valid signature. The key generation phase can be executed either by a trusted dealer or a distributed key generation protocol. Since in NIRVANA no trusted third party exists then we recall their proposed non-interactive SPTS based on the Pedersens's DKG [Ped91].

- $(\mathsf{pp}) \leftarrow \mathcal{SPTS}.\mathsf{Setup}(1^\lambda)$: It takes the security parameter 1^λ as input and executes the bilinear pairing group generator $\mathcal{BG}(1^\lambda)$ and returns the global public parameters $\mathsf{pp} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, h, \mathsf{p})$ as output.
- $(\hat{\vec{\mathsf{sk}}}, \vec{\mathsf{vk}}, \mathsf{vk}) \leftarrow \mathcal{SPTS}.\mathsf{KGen}(\mathsf{pp}, t, n)$: For a given group of authorities $\{Au_1, \ldots, Au_n\}$, this probabilistic algorithm takes pp and integers $t, n \in \mathsf{poly}(\lambda)$ s.t. $1 \leq t \leq n$ as inputs and acts as follows:
  1. Each authority Au_i, $1 \leq i \leq n$, samples two initial random integers $(x_{i0}, y_{i0}) \leftarrow\!\$\, \mathbb{Z}_\mathsf{p}^*$ and does the following:
     a) It samples t random pairs $\{x_{ij}, y_{ij}\}_{j=1}^t$ and forms two univariate polynomials $F_i[X] = x_{i0} + x_{i1}X + \ldots + x_{it}X^t \in \mathbb{Z}_\mathsf{p}[X]$ and $G_i[X] = y_{i0} + y_{i1}X + \ldots + y_{it}X^t \in \mathbb{Z}_\mathsf{p}[X]$ of degree t and commits the coefficients by publishing, $V_{ij} = (V_{1ij}, V_{2ij}) = (h^{x_{ij}}, h^{y_{ij}})\ \forall j \in \{0, \ldots, t\}$.
     b) It sends the pair of $(F_i(\ell), G_i(\ell))$ to the ℓ^{th} authority, Au_ℓ, s.t. $\ell \in \{1, \ldots, n\} \setminus \{i\}$ and keeps $(F_i(i), G_i(i))$ secret.
  2. Authority Au_i checks the consistency of the received shares, $(F_\ell(i), G_\ell(i))$, from authority Au_ℓ by computing the equations $h^{F_\ell(i)} = \prod_{j=0}^t V_{1\ell j}^{ij}$ and $h^{G_\ell(i)} = \prod_{j=0}^t V_{2\ell j}^{ij}$. If these equations hold, authority Au_i accepts the shares, otherwise it will reject and report the faulty authority Au_ℓ.
  3. Any faulty authority that receives at least t complaints is labelled as disqualified. At the end of this phase t parties from the set of qualified authorities, $\mathcal{Q} \subset \mathcal{AU}$, do next steps.

[7] `https://crypto.ethereum.org/blog/groth-sahai-blogpost`

4. The global verification key is determined as $\mathsf{vk} := (\mathsf{vk}_1, \mathsf{vk}_2) := (\prod_{i\in\mathcal{Q}} V_{1i0}, \prod_{i\in\mathcal{Q}} V_{2i0}) = (h^{\sum_{i\in\mathcal{Q}} x_{i0}}, h^{\sum_{i\in\mathcal{Q}} y_{i0}})$.
5. Each qualified authority Au_i defines its private key share $\hat{\mathsf{sk}}_i$ as a pair of $\hat{\mathsf{sk}}_i = (\mathsf{sk}_{i,1}, \mathsf{sk}_{i,2}) = \big(\sum_{\ell\in\mathcal{Q}} F_\ell(i), \sum_{\ell\in\mathcal{Q}} G_\ell(i)\big)$.
6. The corresponding verification key vk_i is obtained by computing, $\mathsf{vk}_i = \big(h^{F(i)}, h^{G(i)}\big) = \Big(\prod_{\ell\in\mathcal{Q}} \prod_{j=0}^{t} (V_{1\ell j})^{i^j}, \prod_{\ell\in\mathcal{Q}} \prod_{j=0}^{t} (V_{2\ell j})^{i^j}\Big)$, where $F[X] = \sum_{\ell\in\mathcal{Q}} F_\ell[X]$ and $G[X] = \sum_{\ell\in\mathcal{Q}} G_\ell[X]$.
7. For the disqualified authorities Au_j for $j \in \{1, \ldots, n\} \setminus \mathcal{Q}$ we define $\hat{\mathsf{sk}}_j = (0, 0)$ and corresponding verification key $\mathsf{vk}_j = (1_{\mathbb{G}_2}, 1_{\mathbb{G}_2})$.

It then returns the vectors $\vec{\hat{\mathsf{sk}}} = (\hat{\mathsf{sk}}_1, \ldots, \hat{\mathsf{sk}}_n)$ and $\vec{\mathsf{vk}} = (\mathsf{vk}_1, \ldots, \mathsf{vk}_n)$ for each party Au_i for $i \in [n]$ along with a common verification key vk.

- $(\sigma_i) \leftarrow \mathcal{SPTS}.\mathsf{Par\text{-}Sign}(\mathsf{pp}, \hat{\mathsf{sk}}_i, M)$: An authority Au_i who owns the secret signing key $\hat{\mathsf{sk}}_i$ takes an indexed DH message $M := (id, M_1, M_2) \in \mathcal{M}^{\mathsf{H}}_{\mathsf{iDH}}$ as input and it then runs the hash function $\mathsf{H}(id)$ to get the random basis $\boldsymbol{g}$. If $e(\boldsymbol{g}, M_2) = e(M_1, h)$, it computes the partial signature $\sigma_i = (\boldsymbol{g}, s_i) = (\boldsymbol{g}, \boldsymbol{g}^{\mathsf{sk}_{i1}} M_1^{\mathsf{sk}_{i2}})$ and returns σ_i as output; otherwise it responds by $\perp$.
- $(0, 1) \leftarrow \mathcal{SPTS}.\mathsf{Par\text{-}Vf}(\mathsf{pp}, \mathsf{vk}_i, \tilde{M}, \sigma_i)$: The partial verification algorithm takes the i^{th} verification keys vk_i, a partial signature σ_i and message $\tilde{M} := (M_1, M_2) \in \mathcal{M}_{\mathsf{iDH}}$ as inputs. If all conditions: $M_1, s_i \in \mathbb{G}_1$, $\boldsymbol{g} \neq 1_{\mathbb{G}_1}$ and $M_2 \neq 1_{\mathbb{G}_2}$, $e(\boldsymbol{g}, M_2) = e(M_1, h)$ and $e(\boldsymbol{g}, \mathsf{vk}_{i1}) e(M_1, \mathsf{vk}_{i2}) = e(s_i, h)$ hold, then it returns 1 (accept); otherwise it returns 0 (reject).
- $(\sigma, \perp) \leftarrow \mathcal{SPTS}.\mathsf{Recon}(\mathsf{pp}, \{i, \sigma_i\}_{i\in\mathcal{T}})$: The reconstruction algorithm takes a set of verified partial signatures $\{\sigma_i\}_{i\in\mathcal{T}}$. It returns a reconstructed signature by computing $\sigma := (\boldsymbol{g}, s) := \Big(\boldsymbol{g}, \prod_{i\in\mathcal{T}} s_i^{L_i^{\mathcal{T}}(0)}\Big)$, where $L_i^{\mathcal{T}}(0)$ is the Lagrange coefficient for the i^{th} index corresponding to set $\mathcal{T}$ (see App. B.2) and returns σ as output if and only if $|\mathcal{T}| \geq t$, otherwise it returns $\perp$.
- $(0, 1) \leftarrow \mathcal{SPTS}.\mathsf{Vf}(\mathsf{pp}, \mathsf{vk}, \tilde{M}, \sigma)$: To verify a reconstructed signature σ, this algorithm takes the verification key vk and message $\tilde{M} \in \mathcal{M}_{\mathsf{iDH}}$ as inputs. If all conditions: $M_1, s \in \mathbb{G}_1$, $\boldsymbol{g} \neq 1_{\mathbb{G}_1}$ and $M_2 \neq 1_{\mathbb{G}_2}$, $e(\boldsymbol{g}, M_2) = e(M_1, h)$ and $e(\boldsymbol{g}, \mathsf{vk}_1) e(M_1, \mathsf{vk}_2) = e(s, h)$ hold, then it returns 1 (accept); otherwise it returns 0 (reject).

The correctness and Threshold EUD-CiMA-security with adaptive adversary is formally proved in [SSKP22]. Despite not being discussed in [SSKP22], the signature components and underlying message are efficiently re-randomizable. In general, one can re-randomize a signature $\sigma = (h, s)$ by sampling a random integer $r \leftarrow\$ \mathbb{Z}_{\mathsf{p}}^*$ so that $\sigma' = (h', s') = (h^r, s^r)$ can be verified under the re-randomized message $\tilde{M}' = (M_1', M_2') = (M_1^r, M_2^r)$.

In addition to the authorities who use the SPTS scheme to compute signatures in order to ensure that no customer can generate fake collateral proofs, witnesses in Nirvana also need to sign payment guarantees that they have not seen in the current epoch. This is done to ensure a merchant that this payment guarantee is not being double-spent.

However, unlike the authorities, witnesses do not need a threshold and re-randomizable signature scheme that is compatible with Groth-Sahai proof systems [GS08]; however, it needs to be aggregatable [MWD+20]. Hence, the witnesses in our instantiation rely on BLS signatures [BLS04].

BLS Signatures: Based on the formal definition of a signature scheme in App. B.3, we outline BLS signatures [BLS04] as an efficient and aggregatable signature scheme. For a given security parameter 1^λ and cyclic group $(\mathbb{G}, \mathsf{p})$, the BLS signature consists of the following PPT algorithms:

- $(\mathsf{pp}) \leftarrow \mathsf{Setup}(1^\lambda)$: Given security parameter 1^λ, it samples $h \leftarrow\$ \mathbb{G}$ and a hash-to-curve function $\mathsf{H} : \{0,1\}^* \rightarrow \mathbb{G}$. It returns $\mathsf{pp} = (\mathsf{H}, h)$ as output.
- $(\mathsf{vk}, \hat{\mathsf{sgk}}) \leftarrow \mathcal{DS}.\mathsf{KGen}(\mathsf{pp})$: This algorithm takes pp and samples a random integer $s \leftarrow\$ \mathbb{Z}_\mathsf{p}^*$ and returns the pair of signing/verification keys $(\hat{\mathsf{sgk}}, \mathsf{vk}) = (s, h^s)$.
- $(\sigma) \leftarrow \mathcal{DS}.\mathsf{Sign}(\hat{\mathsf{sgk}}, m)$: This deterministic algorithm takes as inputs the signing key $\hat{\mathsf{sgk}}$ and message $m \in \mathcal{M}$ and computes $\sigma = \mathsf{H}(m)^s$ and returns the signature σ.
- $(0,1) \leftarrow \mathcal{DS}.\mathsf{Vf}(\mathsf{vk}, \sigma, m)$: The verification algorithm takes vk, σ and message m as inputs. It return 1 (accept), if $\sigma \in \mathbb{G}_1$ and the equation $e(\sigma, h) = e(\mathsf{H}(m), \mathsf{vk})$ holds, otherwise it returns 0 (reject).

Commitment: As we already discussed the given SPTS construction is defined over the indexed DH message spaces and each secret PRF key needs to get an index. In this case we utilize a commitment to the secret scalar message as an index (see App. B.4). The hiding property of such cryptographical primitives masks the secret PRF keys used in our construction. In addition, the binding property of a commitment scheme ensures the unforgeability of these secret PRF keys. Over a cyclic group $(\mathbb{G}, \mathsf{p})$, we recall the Pedersen commitment scheme [Ped92] as an efficient and structure-preserving construction. (to avoid complex proof systems in the proof of knowledge of committed value.)

- $(\mathsf{pp}) \leftarrow \mathcal{CO}.\mathsf{Setup}(1^\lambda)$: It takes the security parameter 1^λ as input, picks two generators $g \leftarrow\$ \mathbb{G}$ and $h \leftarrow\$ \mathbb{G}$ uniformly at random and returns the public parameters $\mathsf{pp} = (\mathbb{G}, \mathsf{p}, g, h)$ as output.
- $(\mathsf{com}) \leftarrow \mathcal{CO}.\mathsf{Com}(\mathsf{pp}, m)$: It takes the public parameters pp and a message $m \in \mathbb{Z}_\mathsf{p}$ as inputs, picks random opening $\tau \leftarrow\$ \mathbb{Z}_\mathsf{p}^*$ and computes and outputs the commitment $\mathsf{com} = g^\tau h^m$.
- $(0,1) \leftarrow \mathcal{CO}.\mathsf{Vf}(\mathsf{pp}, \mathsf{com}, m', \tau')$: It computes $\mathsf{com}' = g^{\tau'} h^{m'}$, if $\mathsf{com} = \mathsf{com}'$ it accepts and returns 1: else it responds with 0 and rejects the commitment.

The preservation of customer's anonymity and the unlinkability of transactions while providing payment guarantees is the main motivation of Nirvana; however, this can lead

to some issues. In the case of a collusion between a malicious witness and a customer, the identity of the witness is publicly known, but the customer still remains anonymous. To tackle this issue, we introduce a novel randomness-reusable threshold encryption, which enables participants to reveal the identity of a customer when they double-spend without relying on a trusted third party. Using this identity, the malicious customer can be blacklisted and could also undergo possible legal procedures that are orthogonal to our work.

Collaborative Key Generation is a variation of DKG that uses threshold cryptography to achieve distributed key generation. The (n, t)-DKG generates n pairs of secret and public keys such that at least t parties among n are required to execute a key oriented operation, whereas any subset of size smaller than t is not able to execute it. Likewise in a (n, t, k)-CKG, a global public key pk corresponding to a secret key sk is shared among n parties such that any subset of larger than k can rebuild pk and any subset of larger than $k < t \leq n$ can reconstruct the sk. The main difference here is that any operation that needs the secret key sk requires the cooperation of at least t collaborators, whereas the public key itself can be reconstructed by at least $k < t$ parties.

Randomness-Reusable Threshold Encryption: By referring to formal definition of randomness-reusable (n, t)-threshold encryptions (see App. B.5), we introduce a randomness-reusable variant of the ElGamal threshold encryption [DF90]. For given public parameters $\mathsf{pp} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g, h, \mathsf{p}, e)$, the construction is defined as follows:

- $(\vec{\mathsf{pk}}, \mathsf{pk}) \leftarrow rb\mathcal{TE}.\mathsf{KGen}(\mathsf{pp}, \ell, t, 2)$: The key generation is a collaborative algorithm that is executed by the group of authorities $\mathcal{AU}$ of size n to generate public keys for ℓ merchants, defined in the Algo. 1 with a fixed threshold $k = 2$, while their corresponding secret keys remain hidden as long as the majority of the authorities are honest ($t \geq n/2 + 1$).
  1. Each authority $\mathcal{AU}_i$ samples an initial random value $x_{i0} \leftarrow\!\!\$\, \mathbb{Z}_{\mathsf{p}}^*$ and does the following:
     a) It samples a random integer $\{x_{i1}\}$, forms a polynomial $F_i[X] = x_{i0} + x_{i1}X \in \mathbb{Z}_{\mathsf{p}}[X]$ and commits the coefficients by publishing, $V_{ij} = h^{x_{ij}}\ \forall\ j \in \{0, 1\}$.
     b) It broadcasts $F_i(j)$ to the rest of authorities as a share corresponding to the j^{th} merchants.
  2. Each authority checks the consistency of the received shares, $F_i(j)$, from $\mathcal{AU}_i$ by computing the equations $g^{F_i(j)} = V_{i0}V_{i1}^j$ for all merchants' label $j \in [\ell]$. If this equation holds, the shares generated by $\mathcal{AU}_i$ will be accepted, otherwise it will reject and then report the faulty authority $\mathcal{AU}_i$.
  3. Any faulty authority that receives at least $t \geq n/2 + 1$ complaints is labelled as disqualified. At the end of this phase t parties from the set of qualified authorities, $\mathcal{Q} \subset \mathcal{AU}$ perform the next steps.

4. The global public key is determined as $\mathsf{pk} := \prod_{i \in \mathcal{Q}} V_{i0} = h^{\sum_{i \in \mathcal{Q}} x_{i0}}$.
5. Each merchant $\mathcal{M}_j$ is assigned by a public key pk_j that is obtained by computing, $\mathsf{pk}_j = h^{F(j)} = \prod_{i \in \mathcal{Q}} \left(V_{i0}(V_{i1})^j\right)$, where $F[X] = \sum_{i \in \mathcal{Q}} F_i[X]$.

These steps complete the collaboration key generation phase and return the set of public keys $\{\mathsf{pk}_j\}_{1 \le j \le \ell}$ along with the global public key pk.

- $(\mathtt{Ct}_j) \leftarrow rb\mathcal{TE}.\mathsf{Enc}(\mathsf{pp}, \mathsf{pk}, m, \mathsf{pk}_j)$: The encryption algorithm takes public parameters pp, global public key pk, the message m and public key pk_j as inputs. It samples $r \leftarrow\!\$\, \mathbb{G}_1$ uniformly at random and generates the ciphertext underlying each recipient by computing $(\mathtt{Ct}_j, v) := \big(e(r, \mathsf{pk}_j), m \cdot e(r, \mathsf{pk})\big)$.
- $(m, \perp) \leftarrow rb\mathcal{TE}.\mathsf{Dec}(\mathsf{pp}, \mathtt{Ct}_j, \mathtt{Ct}_{j'}, v)$: The decryption algorithm takes twin ciphertexts $(\mathtt{Ct}_j, v)$ and $(\mathtt{Ct}_{j'}, v)$ along with public parameters pp as inputs. Let $\mathcal{J} = \{j, j'\}$, it computes $g_T^{rs} : z = \left(\mathtt{Ct}_j^{L_j^{\mathcal{J}}(0)} \cdot \mathtt{Ct}_{j'}^{L_{j'}^{\mathcal{J}}(0)}\right)$ and then returns $m := v/z$, otherwise, it responds with $\perp$.

Non-Interactive Zero-Knowledge proofs: Until now, we have seen how reusable collaterals can be verified without the risk of a double-spending attack. However, in order to preserve the anonymity of customers in Nirvana, the merchant should be able to verify a payment guarantee without needing to access all information. Hence, we require NIZKs to convince the merchant that a payment guarantee provided by a customer is correct without requiring the customer to reveal their secret information. To do this, we need two properties of NIZKs, i.e., zero-knowledge and knowledge soundness defined in App. B.6. Based on the chosen building blocks, the prover should generate four NIZK proofs as follows:

$$\pi_1 = \left\{ \left(\mathsf{w}_1 := \hat{k}_j, \mathsf{x}_1 = (\mathsf{pp}, \mathsf{com}_j) \right) : y_1 := \mathcal{CO}.\mathsf{Com}(\mathsf{pp}, \hat{k}_j) \right\} \ ,$$

$$\pi_2 = \left\{ \left(\mathsf{w}_2 := \hat{k}_j, \mathsf{x}_2 = (\mathsf{pp}, R_t) \right) : y_2 := R_t^{\hat{k}_j} \right\} \ ,$$

$$\pi_3 = \left\{ \left(\mathsf{w}_3 := (r_t, \hat{k}_j), \mathsf{x}_3 = (\mathsf{pp}, \mathsf{pk}_{bm}, \mathtt{Ct}_m) \right) : y_3 := \mathtt{Ct}_m^{\hat{k}_j} \right\} \ ,$$

$$\pi_4 = \left\{ \left(\mathsf{w}_4 := (-t \cdot \hat{\mathsf{sk}}_{cn}, \hat{k}_j), \mathsf{x}_2 = (\mathsf{pp}, \mathtt{Ct}) \right) : \ \ y_4 := (v/\mathsf{pk}_{cn})^{\hat{k}_j} \cdot e(g,h)^{-t \cdot \hat{\mathsf{sk}}_{cn}} \right\} \ .$$

Respectively, the verifier can check the validity of the proofs by running the NIZK's verify algorithm along with checking whether the equations $e(g,h) \cdot R_t^{-t} = y_2$, $e(g, \mathsf{mpk}) \cdot (v)^{-t} = y_4$ and $e(g, \mathsf{pk}_{bm}) \cdot (\mathtt{Ct}_m)^{-t} = y_3$ hold or not.

Despite the fact that the described relation can be proved by the standard Schnorr proofs [Sch90], in this paper, we use the Groth-Sahai proof systems [GS08] that are secure in the standard model and support a straight-line extraction of the witnesses, i.e., avoids rewinding as required for Fiat-Shamir (FS) heuristic [FS87]. More precisely, non-interactive versions of Schnorr proofs obtained via the FS heuristic and its security is guaranteed only

within the random oracle model [BR93]. This leads to protocols only providing heuristic security guarantees [CGH98]: this is problematic if they are within composable frameworks such as the universal composability (UC) framework [Can01] that we leave it to the future works.

Groth-Sahai (GS) proof system [GS08]. Groth and Sahai proposed efficient and practical Non-Interactive Zero-Knowledge (NIZK) proofs and Non-Interactive Witness Indistinguishable (NIWI) proofs in the standard model for algebraic statements that are bilinear group-dependent languages. In this paper, we take one of the instantiation proposed in [GS08] that relies on the symmetric external Diffie-Hellman (SXDH) assumption over prime order groups. Over an asymmetric bilinear group $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathsf{p}, e)$, the relation over variables $\mathcal{X}_1, \ldots, \mathcal{X}_m \in \mathbb{G}_1$, $\mathcal{Y}_1, \ldots, \mathcal{Y}_n \in \mathbb{G}_2$ and a constant $T \in \mathbb{G}_T$ can be any product-pairing equation (PPE) on the form:

$$\prod_{i=1}^{n} e(A_i, \mathcal{Y}_i) \prod_{i=1}^{m} e(\mathcal{X}_i, B_i) \prod_{j=1}^{m} \prod_{i=1}^{n} e(\mathcal{X}_j, \mathcal{Y}_i)^{\mu_{i,j}} = T \ ,$$

where $(A_1, \ldots, A_n) \in \mathbb{G}_1$, $(B_1, \ldots, B_m) \in \mathbb{G}_2$ and $\{\mu_{i,j}\}_{1 \leq i \leq m \ \& \ 1 \leq j \leq n} \in \mathbb{Z}_{\mathsf{p}}^{m \cdot n}$ are constant.

The high-level idea is that by taking the setup parameters and the common reference string (CRS), the prover can commit to the hidden variables (witness components) and then provide the proofs to convince the verifier about the validity of above relation underlying the statement to be proven. Note that the GS-proof is Witness-Indistinguishable (WI) when $T \in \mathbb{G}_T$ while this can be transformed into a Zero-Knowledge proof system if $T = 1_{\mathbb{G}_T}$.

To be more concrete, we outline a simple case of $m = n = 1$ for the above mentioned PPE and the CRS contains group elements $f_1 = (g_x, f_1) \in \mathbb{G}_1$ and $f_2 = (g_y, f_2) \in \mathbb{G}_1$. The prover computes the commitments $C_{\mathcal{X}} = (1_{\mathbb{G}_1}, \mathcal{X}) \cdot {f_1}^{r_1} \cdot {f_2}^{s_1}$ and $C_{\mathcal{Y}} = (1_{\mathbb{G}_1}, \mathcal{Y}) \cdot {f_1}^{r_2} \cdot {f_2}^{s_2}$, where $r_1, r_2, s_1, s_2 \leftarrow\!\!\$\, \mathbb{Z}_{\mathsf{p}}^*$ are sampled uniformly at random by the prover. In addition, the prover sends the proof $\pi = (\pi_1, \pi_2) = (g_x^{-r_1} g_y^{-r_2}, g_x^{-s_1} g_y^{-s_2})$ and the tuple $(C_{\mathcal{X}}, C_{\mathcal{Y}}, \pi)$ to the verifier. The verifier checks whether the equation $T = e(C_{\mathcal{X}}, g_x) e(C_{\mathcal{Y}}, g_y) e(f_1, \pi_1) e(f_2, \pi_2)$ holds or not.

5 Performance Analysis

In this section, we demonstrate the performance of Nirvana. Based on the application, the costs incurred in each phase are divided into two parts, termed "offline phase" and "online phase". The former includes the parameter generation, key generation and registration functions. The latter is solely responsible for spending and verification and is the main focus of this evaluation.

We implemented Nirvana by using the Charm-Crypto framework [AGM+13], a Python library for Pairing-based Cryptography and obtained the benchmarks on four AWS EC2 instances.

Table 1: The list of scenarios and the location of entities.

Scenarios	Nirvana Location	Customer Location	Merchant Location	Witness Location	Average Spending Time	Average Verification Time
1	Singapore	California	Frankfurt	London	0.93	0.99
2	Singapore	Frankfurt	London	California	0.35	1.27
3	Frankfurt	London	California	Singapore	0.85	1.24

As can be seen in Tab. 1, we ran three sets of experiments, with the following location configurations: (1) Nirvana located in Singapore, customer in California, merchant in Frankfurt and witnesses in London. (2) Nirvana located in Singapore, customer in Frankfurt, merchant in London and witnesses in California. (3) Nirvana located in Frankfurt, customer in London, merchant in California and witnesses in Singapore.

For the sake of convenience, during our implementation, we assumed that all witnesses are located in the same location. However, placing the witnesses in different locations would only add a small delay based on their distance from the merchant; in order to reflect this delay, we do sequential witness verification instead of parallel verification.

All our EC2 instances had the same computational configuration, i.e., an Ubuntu Server 20.04 LTS (HVM) with an Intel (R) Xeon(R) CPU @ 2.50 GHz and 16 GB of memory. We apply the Barreto-Naehrig (BN254) curve (also known as type F groups), $y^2 = x^3 + b$ with embedding curve degree 12 [BN06]. In this pairing group, the base field order is 256 bits. Based on our Python code[8], the overhead of the spending and verification algorithm is summarized in Fig. 4.

Latency: Fig. 4 shows the almost constant relationship between the total number of witnesses for each collateral and their spending time. With spending time, we capture the *Spend* functionality in Algo. 1, where the customer generates the payment ciphertext and NIZK proofs to provide a payment guarantee to the merchant. It also includes the time required to send the payment guarantee to a merchant located in either of the locations in Tab. 1. As can be seen, the time required to generate a payment guarantee in Nirvana for merchants in Scenario 1 and 3 is almost identical because the distance between the customer and merchant in both scenarios is roughly the same (i.e., California to Frankfurt and London to California). The effect of distance is also reflected in Scenario 2 when the customer is in Frankfurt and the merchant is in London, with each payment guarantee only taking 350 ms on average to be spent as compared to $\sim$ 900 ms in the other two cases.

Fig. 4 shows the linear relationship between the total number of witnesses for each collateral and the verification time required by a merchant and witnesses. With verification time, we capture the *Vf* functionality in Algo. 1, where the merchant verifies the NIZK

[8] `https://github.com/NirvanaPayments/Nirvana`

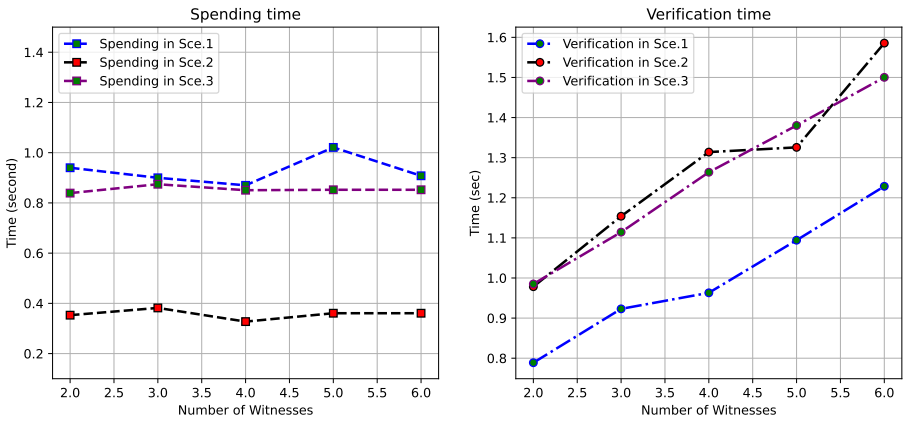

Fig. 4: Analysis of time required for spending and verification of payment guarantees in Nirvana.We have considered three different scenarios for the location of participants as described in Tab. 1.

proofs provided by the customer as a payment guarantee and waits for a majority of the witnesses to approve that this guarantee is unique. It also includes the time required to receive these proofs from the customer located in either location in Tab. 1. As can be seen, the time required to verify a payment guarantee in Nirvana for merchants located in all locations grows linearly with a total number of witnesses, with verification only requiring 990 ms on average in Scenario 1 with two witnesses and 1.3 s on average with 6 witnesses. This is due to the requirement of each individual witness to check their local storage for these guarantees, sign the guarantee if it is unique and finally send the signed guarantee to the merchant. Similar to spending, the distance between countries also effects the time taken to verify each payment guarantee. In scenario 1 of Tab. 1, the merchant and witnesses are located nearby, hence the verification only takes 990 ms when each collateral is assigned to two witnesses as compared to ~ 1.25 s for both scenario 2 and 3. This is because of the latency incurred due to the geographical distance between, e.g., California and London.

Smart contract cost: Tab. 2 provides USD equivalents of the cost of executing functions (such as registration, etc.) on Nirvana's smart contract. Since our smart contract can be deployed on any Ethereum Virtual Machine (EVM) supporting blockchain, we calculated the costs on both Ethereum (high Gas fees) and Celo (low Gas fees) using the current conversion rates and a Gas price (Gwei) of 27 for Ethereum and 2.42 for Celo.[9] As can be seen in Tab. 2, one-time registration of a customer costs them 107 400 Gas or \$7.37 on Ethereum and \$0.67 on Celo. This Gas cost is a bit high due to the requirement for customers to store a secret to enable victim merchants to redeem their collateral on the smart contract. However, this cost goes down for merchant registration since merchants

[9] `https://ethgasstation.info/`, `https://explorer.bitquery.io/celo_rc1/gas`

do not need to store a secret to redeem their collaterals, hence only costing 54 317 Gas or $3.73 on Ethereum and $0.34 on Celo. In case a victim merchant wants to claim a malicious customer's collateral, it costs them 34 972 Gas or $2.40 on Ethereum and $0.22 on Celo. Finally, if a merchant or customer want to withdraw their money from Nirvana's smart contract, it costs them 22 525 Gas or $1.55 on Ethereum and $0.14 on Celo.

Table 2: Costs of transactions in Nirvana's smart contract deployed on Ethereum and Celo.

Function	Gas	USD (Ethereum)	USD (Celo)
Customer reg.	107 400	7.37	0.67
Merchant reg.	54 317	3.73	0.34
Claim collateral	34 972	2.40	0.22
Withdraw collateral	22 525	1.55	0.14

We acknowledge that Nirvana is not cost efficient when used on Ethereum; however, it is important to note that with the current ETH 2.0 roadmap[10], efforts are being made to lower the Gas fees on Ethereum and eventually make it more usable for Nirvana. More information on the functionality of Nirvana's smart contract can be found in App. D

6 Related Work

Threshold-Issuance Anonymous credentials (TIAC). Anonymous Credentials (AC) are an important privacy-preserving authentication technique, which allows users to prove the possession of attributes while preserving their anonymity from verifiers. A popular approach to construct ACs is to rely on specific signature schemes like Camenisch-Lysyanskaya [CL03,CL04] or Pointcheval-Sanders signatures [PS16], which can sign attribute vectors and can be re-randomized to support unlinkable verifications. The use of zero-knowledge proofs in verification, in addition, allows to prove ownership of undisclosed attributes.

Basically, a credential system is a certificate that is generated via a cryptographic process executed by the credential issuer authority. Before generating such a certificate, the authority verifies the information that this certificate certifies. However, AC constructions are prone to compromise since they rely on a single credential issuer authority; hence, a single point of failure. In order to overcome this issue, Sonnino et al. proposed Coconut [SAB+19], a so-called Threshold-Issuance AC (TIAC) system, that enables a subset of credential issuers to jointly issue credentials and already found practical applica-

[10] `https://ethereum.org/en/eth2/`

tions [BSKD22,TBA+22]. More precisely, they rely on a threshold variant of PS signatures [PS16] and distribute the signing among n credential issuers where a cooperation of t issuers is required to generate a valid credential. This improves the availability and also solves the single point of failure problem of centralized AC schemes. Meanwhile no subset of credential issuers less than t can generate a fake credential and also the credentials are re-randomizable, which protects the anonymity of the parties even in the case of collusion between authorities and a verifier. In blockchains, threshold issuance setting is more desirable, because decentralized blockchains guarantee integrity when the number of dishonest authorities is below a threshold (Byzantine fault tolerance). By retaining the structure of the scheme and avoiding structure-destroying primitives like hash functions, our suggested system can improve the efficiency of the Coconut. While Coconut is practical from a performance point of view and already found practical applications[11], it unfortunately lacks a rigorous security analysis. Recently, Rial and Piotrowska [RP22] conducted a security analysis of Coconut modeled via an attribute-based access control with threshold issuance functionality in the UC model. However, this analysis required some changes to the original Coconut scheme. As an independent line of research, we believe the collateral issuance function described in Algo. 1 can be considered as a TIAC, while it relies on a structure-preserving threshold signature that can simplify the UC model. We leave it as an interesting future work.

E-cash. The idea of balancing privacy dates back to the introduction of untraceable electronic (or "digital") payment schemes involving a customer, bank and merchant by Chaum [Cha82]. His idea enabled a customer to get a "coin", signed blindly by an issuing bank. This coin could later be used for payments to a merchant in exchange for a service. Finally, the merchant would then deposit this coin back to the issuing bank for remuneration. Since these coins were digital, double-spending them was conceptually trivial because digital information is easy to copy. As a solution, in "on-line" e-cash [Cha82] the issuing bank was asked to verify each transaction individually before it was marked successful. Chaum, Fiat and Naor later extended this idea to support "off-line" payments, i.e, a customer could make untraceable payments to the merchant without involving a bank for every transaction [CFN90] . Due to this "off-line" nature of payments, the solution for double-spending combined prevention with tamper-resistant hardware with detection through successful tracking, i.e., the issuing bank would check the list of all coins spent, and once double-spending was spotted the identity of the perpetrator would be revealed. This approach of realizing offline payments while detecting double-spending, known as the Chaum-Fiat-Naor approach (CFN paradigm) was adopted and improved by several following e-cash systems. Franklin and Yung [FY93] provided the first provably secure e-cash scheme, D'Amiano and Crescenzo [DD95] proposed a storage efficient transferrable e-cash scheme that made forwarding received e-cash to other merchants possible and Okamoto and Ohta [OO92]

[11] `https://github.com/nymtech/coconut`

proposed the first untraceable divisible e-cash scheme. Another e-cash scheme proposed by Osipkov et al. [OVHK07], inspired by Brands's work involving a tamper-proof device with observers [Bra94], relies on a co-operative peer-to-peer (p2p) network for preventing double-spends in real time. There is a plethora of literature with several improvements to Chaum's e-cash [Bra94,Sch95,FTY96,CHL05,CG08,CGT08,CG10,BCKL09,BCFK15]. All these schemes, however, work with centrally issued currency and mostly rely on a custodian bank to catch double-spending, except [OVHK07] where a co-operative p2p network is utilised. Nirvana is similar to [OVHK07] in the sense that it also uses an underlying currency (the cryptocurrency of user's choice) and has parties responsible for spotting double-spending attempts (merchants/witnesses instead of banks).

Payment channels/networks/hubs. Layer-2 solutions such as payment channels [Hea13], or its extensions such as payment channel networks (e.g., Lightning network [PD16], Raiden [Net19], Bolt [GM17]) and payment hubs (TumbleBit [HAB+17]) provide instant transaction confirmation, at the cost of some drawbacks. Payment channels require depositing and constantly replenishing a collateral for each merchant a customer wants to transact with. This requirement makes the utilization of payment channels expensive. Moreover, they do not support unilateral payments (customer-to-merchant payments); payment channel networks enable transferring money over established channels between parties (routes), but suffer from route availability issues in case any involved party is unresponsive. Moreover, due to the unilateral nature of payments in retail markets, the funds locked in a channel deplete quickly, and hence their plausibility to act as intermediaries decreases [MWD+20]. Although fund re-balancing techniques [KG17] exist to tackle this issue, they require a user to have multiple channels; more centralized layer-2 schemes [HAB+17,KZF+18] that do not require multiple collaterals and solve route availability issues, either require a substantial collateral to be deposited by the intermediary and/or they do not guarantee privacy from such intermediaries. These hubs also become a single point of failure, and increase data availability risks. Despite the risks that central hubs entail, Avarikioti et al. [AHWW20] suggested that payment channel networks are more stable and efficient when centralized structures are present. Moreover, Zabka et al. [ZFSD22] show the rising centrality in lightning network, an instantiation of payment channel networks on Bitcoin blockchain. Hence it becomes increasingly important to make such semi-decentralized architectures more trustless, robust and secure. With Nirvana, we propose a completely trustless, robust, anonymity-preserving and unlinkable semi-decentralized solution that does not require a customer to make multiple collaterals, or constantly replenish them. Additionally, Nirvana also offers customer anonymity and unlinkability of payment guarantees to its participants.

Snappy. Recently, Mavroudis et al. proposed Snappy [MWD+20], which solves the latency problem of existing public blockchains such as Ethereum and enables fast (zero-

confirmation) and secure payments. This is achieved by designating a set of untrusted validators (state-keepers). These state-keepers are tasked to track all payments of a customer, s.t. a merchant never receives an invalid payment. In the scenario of an invalid payment, the merchants can redeem their payments from the customer's deposit on an Ethereum smart contract, also known as the arbiter. Due to this functionality, customers never have to replenish their deposits as long as they behave honestly, since this deposit is only used in case of misbehaviour. The merchants are also required to deposit collaterals to prevent collusion with malicious customers. This guarantee of remuneration enables a merchant to instantly accept an on-chain payment, as they are ensured a payment even in the worst case, such as a customer performing a double-spending attack. Snappy, by design, does not offer any protocol-level anonymity from the arbiter and state-keepers. This is due to the fact that all transactions are in the clear, and the set of state-keepers is the same. Hence, the transaction details of all customers is available to them. Nirvana ensures protocol-level customers' anonymity and unlinkability of payment guarantees. Like Snappy, deposits in Nirvana are reusable unless used to compensate victims for the actions of misbehaving parties.

LDSP: LDSP [NCWW21] is a concurrent work to Nirvana that aims to solve privacy issues of Snappy while providing fast payments. However, LDSP does not support reusable collaterals.

Table 3: Efficiency and Functionalities Comparison.

Scheme	Customer Anonymity	Transaction unlinkability	Privacy-Balancing	Efficient Transaction Approval	Re-usability of Collaterals
Blockchain Transaction	✓[a]	✓[a]	✗	✗	-
Snappy [MWD+20]	✗	✗	✗	✓	✓
Nirvana	✓	✓	✓	✓	✓

[a] Depends on the underlying blockchain.

Comparison of Nirvana with related work: In Tab. 3, we compare Nirvana with Snappy on several characteristics, such as: (1) **Customer Anonymity preservation (CA) and Transaction Unlinkability (TU):** A payment system should preserve the anonymity of honest entities and provide unlinkability of payments. As discussed before, Snappy [MWD+20] is not privacy-preserving by design; however, Nirvana fulfill this requirement. (2) **Privacy-Balancing (PB):** Much like traditional e-cash schemes, a payment system should balance the privacy of its participants by revealing the identities of all malicious entities who try to perform double-spending. This requirement is not fulfilled by Snappy. Thanks to our novel rbTE scheme (see Sect. 4), Nirvana fulfills this requirement. (3) **Efficient Transaction Approval (ETA):** A payment system should enable

instant and secure payments. Both Snappy and Nirvana provide ETA. Nirvana only requires a small subset ($\log_{10}(n)$) of the merchant consortium of size n to approve a payment guarantee and regardless of our enhanced design choices, NIRVANA's evaluation shows its efficiency in doing so. (4) **Re-usability of collaterals (RE):** A payment system should only use collaterals as payment guarantees and not actual means of payments. Snappy and NIRVANA support reusable collaterals as long as the customers are honest. However, in NIRVANA, this is achieved in a privacy-preserving way with only the dishonest customers losing their collaterals and privacy. In terms of a normal blockchain transaction, it can be privacy-preserving in some cases (e.g., Monero [Noe15]), but it does not fulfill any other requirement.

7 Conclusion and Future Work

In this paper, we proposed NIRVANA. It provides anonymity-preserving and unlinkable payment guarantees for instant confirmation of on-chain payments. NIRVANA complements recent solutions for increasing the transaction throughput of permissionless blockchains by solving the latency problem and enabling a high system-level transaction throughput. It allows merchants in a retail system to safely accept zero-confirmation payments without risk of double-spending. NIRVANA uses a collaborative p2p network to prevent double-spending in real time. Additionally, we designed a novel randomness-reusable threshold scheme, that enables participants to audit the payments in the network and reveal the identity of malicious customer who perform double-spending. We formally proved that NIRVANA is secure w.r.t. two main security features namely *customers' anonymity and unlinkability of transactions* and *payment certainty for merchants*. Our evaluation showed the capability of NIRVANA in allowing for fast global payments with a delay of less than ~ 1.7 seconds. As future work, NIRVANA can be modified to handle merchant churn, as the witness allocation and collateral requirements prevent merchants from instantly dropping out and assume a fixed consortium. The extension of our privacy-balancing payment guarantee mechanism to capture other possible types of illicit activities using cryptocurrencies, such as terrorist funding, money laundering, also paves a promising path for future research.

Acknowledgments. We would like to thanks Aysajan Abidin, Markulf Kohlweiss, Svetla Nikova and Roozbeh Sarenche for their helpful discussions and valuable comments. Akash Madhusudan, Mahdi Sedaghat and Bart Preneel were supported in part by the Flemish Government through the FWO SBO project SNIPPET S007619, the Research Council KU Leuven C1 on Security and Privacy for Cyber-Physical Systems and the Internet of Things with contract number C16/15/058 and by CyberSecurity Research Flanders with reference number VR20192203.

References

AFG+10. Masayuki Abe, Georg Fuchsbauer, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. Structure-preserving signatures and commitments to group elements. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 209–236. Springer, Heidelberg, August 2010.

AGM+13. Joseph A. Akinyele, Christina Garman, Ian Miers, Matthew W. Pagano, Michael Rushanan, Matthew Green, and Aviel D. Rubin. Charm: a framework for rapidly prototyping cryptosystems. *Journal of Cryptographic Engineering*, 3(2):111–128, June 2013.

AHWW20. Zeta Avarikioti, Lioba Heimbach, Yuyi Wang, and Roger Wattenhofer. Ride the lightning: The game theory of payment channels. In *FC 2020*, LNCS, pages 264–283. Springer, Heidelberg, 2020.

BCC88. Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *J. Comput. Syst. Sci.*, 37(2):156–189, 1988.

BCF+11. Olivier Blazy, Sébastien Canard, Georg Fuchsbauer, Aline Gouget, Hervé Sibert, and Jacques Traoré. Achieving optimal anonymity in transferable e-cash with a judge. In Abderrahmane Nitaj and David Pointcheval, editors, *AFRICACRYPT 11*, volume 6737 of *LNCS*, pages 206–223. Springer, Heidelberg, July 2011.

BCFK15. Foteini Baldimtsi, Melissa Chase, Georg Fuchsbauer, and Markulf Kohlweiss. Anonymous transferable E-cash. In Jonathan Katz, editor, *PKC 2015*, volume 9020 of *LNCS*, pages 101–124. Springer, Heidelberg, March / April 2015.

BCKL09. Mira Belenkiy, Melissa Chase, Markulf Kohlweiss, and Anna Lysyanskaya. Compact e-cash and simulatable VRFs revisited. In Hovav Shacham and Brent Waters, editors, *PAIRING 2009*, volume 5671 of *LNCS*, pages 114–131. Springer, Heidelberg, August 2009.

Ben21. Yaacov Benmeleh. Blockchain firm starkware valued at $2 billion in funding round. `https://www.bloomberg.com`, 2021.

BF01. Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 213–229. Springer, Heidelberg, August 2001.

BFM88. Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 103–112. ACM, 1988.

Bje21. Mihailo Bjelic. Polygon takes a major lead in zk rollups; welcomes mir, a groundbreaking zk startup in a $400m deal. https://blog.polygon.technology/polygon-takes-a-major-lead-in-zk-rollups-welcomes-mir-a-groundbreaking-zk-startup-in-a-400m-deal/, 2021.

BLS04. Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. *Journal of Cryptology*, 17(4):297–319, September 2004.

BN06. Paulo S. L. M. Barreto and Michael Naehrig. Pairing-friendly elliptic curves of prime order. In Bart Preneel and Stafford Tavares, editors, *SAC 2005*, volume 3897 of *LNCS*, pages 319–331. Springer, Heidelberg, August 2006.

BP15. Alex Biryukov and Ivan Pustogarov. Bitcoin over Tor isn't a good idea. In *2015 IEEE Symposium on Security and Privacy*, pages 122–134. IEEE Computer Society Press, May 2015.

BR93. Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *ACM CCS 93*, pages 62–73. ACM Press, November 1993.

Bra94. Stefan Brands. Untraceable off-line cash in wallets with observers (extended abstract). In Douglas R. Stinson, editor, *CRYPTO'93*, volume 773 of *LNCS*, pages 302–318. Springer, Heidelberg, August 1994.

BSAB+19. Shehar Bano, Alberto Sonnino, Mustafa Al-Bassam, Sarah Azouvi, Patrick McCorry, Sarah Meiklejohn, and George Danezis. SoK: Consensus in the age of blockchains. In *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*, pages 183–198, 2019.

BSKD22. Mathieu Baudet, Alberto Sonnino, Mahimna Kelkar, and George Danezis. Zef: Low-latency, scalable, private payments. Cryptology ePrint Archive, Report 2022/083, 2022. https://eprint.iacr.org/2022/083.

But21. Vitalik Buterin. An incomplete guide to rollups. https://vitalik.ca/general/2021/01/05/rollup.html, 2021.

Can01. Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, October 2001.

CDE+16. Kyle Croman, Christian Decker, Ittay Eyal, Adem Efe Gencer, Ari Juels, Ahmed E. Kosba, Andrew Miller, Prateek Saxena, Elaine Shi, Emin Gün Sirer, Dawn Song, and Roger Wattenhofer. On scaling decentralized blockchains - (A position paper). In Jeremy Clark, Sarah Meiklejohn, Peter Y. A. Ryan, Dan S. Wallach, Michael Brenner, and Kurt Rohloff, editors, *FC 2016 Workshops*, volume 9604 of *LNCS*, pages 106–125. Springer, Heidelberg, February 2016.

CDHK15. Jan Camenisch, Maria Dubovitskaya, Kristiyan Haralambiev, and Markulf Kohlweiss. Composable and modular anonymous credentials: Definitions and practical constructions. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part II*, volume 9453 of *LNCS*, pages 262–288. Springer, Heidelberg, November / December 2015.

CFN90. David Chaum, Amos Fiat, and Moni Naor. Untraceable electronic cash. In Shafi Goldwasser, editor, *CRYPTO'88*, volume 403 of *LNCS*, pages 319–327. Springer, Heidelberg, August 1990.

CG08. Sébastien Canard and Aline Gouget. Anonymity in transferable e-cash. In Steven M. Bellovin, Rosario Gennaro, Angelos D. Keromytis, and Moti Yung, editors, *ACNS 08*, volume 5037 of *LNCS*, pages 207–223. Springer, Heidelberg, June 2008.

CG10. Sébastien Canard and Aline Gouget. Multiple denominations in e-cash with compact transaction data. In Radu Sion, editor, *FC 2010*, volume 6052 of *LNCS*, pages 82–97. Springer, Heidelberg, January 2010.

CGH98. Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited (preliminary version). In *30th ACM STOC*, pages 209–218. ACM Press, May 1998.

CGMV18. Jing Chen, Sergey Gorbunov, Silvio Micali, and Georgios Vlachos. ALGORAND AGREEMENT: Super fast and partition resilient byzantine agreement. Cryptology ePrint Archive, Report 2018/377, 2018. https://eprint.iacr.org/2018/377.

CGT08. Sébastien Canard, Aline Gouget, and Jacques Traoré. Improvement of efficiency in (unconditional) anonymous transferable e-cash. In Gene Tsudik, editor, *FC 2008*, volume 5143 of *LNCS*, pages 202–214. Springer, Heidelberg, January 2008.

Cha82. David Chaum. Blind signatures for untraceable payments. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *CRYPTO'82*, pages 199–203. Plenum Press, New York, USA, 1982.

CHL05. Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. Compact e-cash. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 302–321. Springer, Heidelberg, May 2005.

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

CL04. Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 56–72. Springer, Heidelberg, August 2004.

DD95. Stefano D'Amiano and Giovanni Di Crescenzo. Methodology for digital money based on general cryptographic tools. In Alfredo De Santis, editor, *EUROCRYPT'94*, volume 950 of *LNCS*, pages 156–170. Springer, Heidelberg, May 1995.

DF90. Yvo Desmedt and Yair Frankel. Threshold cryptosystems. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 307–315. Springer, Heidelberg, August 1990.

DGK+21. Ivan Damgård, Chaya Ganesh, Hamidreza Khoshakhlagh, Claudio Orlandi, and Luisa Siniscalchi. Balancing privacy and accountability in blockchain identity management. In Kenneth G.

Paterson, editor, *Topics in Cryptology - CT-RSA 2021 - Cryptographers' Track at the RSA Conference 2021, Virtual Event, May 17-20, 2021, Proceedings*, volume 12704 of *Lecture Notes in Computer Science*, pages 552–576. Springer, 2021.

DW15. Christian Decker and Roger Wattenhofer. A fast and scalable payment network with Bitcoin duplex micropayment channels. In *Symposium on Self-Stabilizing Systems*, pages 3–18. Springer, 2015.

DY05. Yevgeniy Dodis and Aleksandr Yampolskiy. A verifiable random function with short proofs and keys. In Serge Vaudenay, editor, *PKC 2005*, volume 3386 of *LNCS*, pages 416–431. Springer, Heidelberg, January 2005.

Eth21. Ethereum.org. Layer 2 rollups. Available in this Link, 2021.

Fac21. Facebook. To build a trusted and innovative financial network that empowers people and businesses around the world. https://www.diem.com/en-us/, 2021.

FS87. Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Heidelberg, August 1987.

FTY96. Yair Frankel, Yiannis Tsiounis, and Moti Yung. "indirect discourse proof": Achieving efficient fair off-line E-cash. In Kwangjo Kim and Tsutomu Matsumoto, editors, *ASIACRYPT'96*, volume 1163 of *LNCS*, pages 286–300. Springer, Heidelberg, November 1996.

Fuc11. Georg Fuchsbauer. Commuting signatures and verifiable encryption. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 224–245. Springer, Heidelberg, May 2011.

FY93. Matthew Franklin and Moti Yung. Secure and efficient off-line digital money. In *International Colloquium on Automata, Languages, and Programming*, pages 265–276. Springer, 1993.

Geo19. Evangelos Georgiadis. How many transactions per second can bitcoin really handle? Theoretically. Cryptology ePrint Archive, Report 2019/416, 2019. `https://eprint.iacr.org/2019/416`.

GGM84. Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions (extended abstract). In *25th FOCS*, pages 464–479. IEEE Computer Society Press, October 1984.

GM17. Matthew Green and Ian Miers. Bolt: Anonymous payment channels for decentralized currencies. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 473–489. ACM Press, October / November 2017.

GMR89. Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on computing*, 18(1):186–208, 1989.

GMR+20. Lewis Gudgeon, Pedro Moreno-Sanchez, Stefanie Roos, Patrick McCorry, and Arthur Gervais. SoK: Layer-two blockchain protocols. In *FC 2020*, LNCS, pages 201–226. Springer, Heidelberg, 2020.

GPS08. Steven D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156(16):3113–3121, 2008. Applications of Algebra to Cryptography.

GS08. Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 415–432. Springer, Heidelberg, April 2008.

HAB+17. Ethan Heilman, Leen Alshenibr, Foteini Baldimtsi, Alessandra Scafuro, and Sharon Goldberg. TumbleBit: An untrusted bitcoin-compatible anonymous payment hub. In *NDSS 2017*. The Internet Society, February / March 2017.

Hea13. Mike Hearn. Micro-payment channels implementation now in bitcoinj. `Bitcointalk.org`, 2013.

KG17. Rami Khalil and Arthur Gervais. Revive: Rebalancing off-blockchain payment networks. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 439–453. ACM Press, October / November 2017.

KZF+18. Rami Khalil, A Zamyatin, G Felley, P Moreno-Sanchez, and A Gervais. Commit-chains: Secure, scalable off-chain payments. Technical report, Cryptology ePrint Archive, Report 2018/642, 2018.

Max13. Gregory Maxwell. Coinjoin: Bitcoin privacy for the real world. In *Post on Bitcoin forum*, 2013.

MBB+19. Andrew Miller, Iddo Bentov, Surya Bakshi, Ranjit Kumaresan, and Patrick McCorry. Sprites and state channels: Payment networks that go faster than lightning. In Ian Goldberg and Tyler Moore, editors, *FC 2019*, volume 11598 of *LNCS*, pages 508–526. Springer, Heidelberg, February 2019.

MWD+20. Vasilios Mavroudis, Karl Wüst, Aritra Dhar, Kari Kostiainen, and Srdjan Capkun. Snappy: Fast on-chain payments with practical collaterals. In *NDSS 2020*. The Internet Society, 2020.

NCWW21. Lucien K. L. Ng, Sherman S. M. Chow, Donald P. H. Wong, and Anna P. Y. Woo. Ldsp: Shopping with cryptocurrency privately and quickly under leadership. In *2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*, pages 261–271, 2021.

Net19. Raiden Network. What is the raiden network. `https://raiden.network/101.html`, 2019.

Noe15. Shen Noether. Ring signature confidential transactions for monero. Cryptology ePrint Archive, Report 2015/1098, 2015. `https://eprint.iacr.org/2015/1098`.

OO92. Tatsuaki Okamoto and Kazuo Ohta. Universal electronic cash. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 324–337. Springer, Heidelberg, August 1992.

OVHK07. Ivan Osipkov, Eugene Y Vasserman, Nicholas Hopper, and Yongdae Kim. Combating double-spending using cooperative p2p systems. In *27th International Conference on Distributed Computing Systems (ICDCS'07)*, pages 41–41. IEEE, 2007.

PD16. Joseph Poon and Thaddeus Dryja. The Bitcoin lightning network: Scalable off-chain instant payments. `https://lightning.network/lightning-network-paper.pdf`, 2016.

Ped91. Torben P. Pedersen. A threshold cryptosystem without a trusted party (extended abstract) (rump session). In Donald W. Davies, editor, *EUROCRYPT'91*, volume 547 of *LNCS*, pages 522–526. Springer, Heidelberg, April 1991.

Ped92. Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 129–140. Springer, Heidelberg, August 1992.

PP13. Roel Peeters and Andreas Pashalidis. Privacy-friendly checking of remote token blacklists. In Simone Fischer-Hübner, Elisabeth de Leeuw, and Chris Mitchell, editors, *Policies and Research in Identity Management*, pages 18–33, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

PS16. David Pointcheval and Olivier Sanders. Short randomizable signatures. In Kazue Sako, editor, *CT-RSA 2016*, volume 9610 of *LNCS*, pages 111–126. Springer, Heidelberg, February / March 2016.

RP22. Alfredo Rial and Ania M. Piotrowska. Security analysis of coconut, an attribute-based credential scheme with threshold issuance. Cryptology ePrint Archive, Report 2022/011, 2022. `https://eprint.iacr.org/2022/011`.

RSY21. Leonid Reyzin, Adam D. Smith, and Sophia Yakoubov. Turning HATE into LOVE: compact homomorphic ad hoc threshold encryption for scalable MPC. In Shlomi Dolev, Oded Margalit, Benny Pinkas, and Alexander A. Schwarzmann, editors, *Cyber Security Cryptography and Machine Learning - 5th International Symposium, CSCML 2021, Be'er Sheva, Israel, July 8-9, 2021, Proceedings*, volume 12716 of *Lecture Notes in Computer Science*, pages 361–378. Springer, 2021.

SAB+19. Alberto Sonnino, Mustafa Al-Bassam, Shehar Bano, Sarah Meiklejohn, and George Danezis. Coconut: Threshold issuance selective disclosure credentials with applications to distributed ledgers. In *NDSS 2019*. The Internet Society, February 2019.

Sch90. Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 239–252. Springer, Heidelberg, August 1990.

Sch95. Berry Schoenmakers. An efficient electronic payment system withstanding parallel attacks. *Information & Computation*, 1995.

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

SP21. Mahdi Sedaghat and Bart Preneel. Cross-Domain Attribute-Based Access Control Encryption. In Mauro Conti, Marc Stevens, and Stephan Krenn, editors, *Cryptology and Network Security*, pages 3–23. Springer International Publishing, 2021.

SSKP22. Mahdi Sedaghat, Daniel Slamanig, Markulf Kohlweiss, and Bart Preneel. Structure-Preserving Threshold Signatures. Cryptology ePrint Archive, Paper 2022/839, 2022. `https://eprint.iacr.org/2022/839`.

TBA+22. Alin Tomescu, Adithya Bhat, Benny Applebaum, Ittai Abraham, Guy Gueta, Benny Pinkas, and Avishay Yanai. Utt: Decentralized ecash with accountable privacy. Cryptology ePrint Archive, Paper 2022/452, 2022. `https://eprint.iacr.org/2022/452`.

TBP20. Florian Tramèr, Dan Boneh, and Kenny Paterson. Remote side-channel attacks on anonymous transactions. In *USENIX Security 2020*, pages 2739–2756. USENIX Association, 2020.

Tri13. Manny Trillo. Stress test prepares Visanet for the most wonderful time of the year. *Available in Link*, 2013.

WSNH19. Gang Wang, Zhijie Jerry Shi, Mark Nixon, and Song Han. SoK: Sharding on blockchain. Cryptology ePrint Archive, Report 2019/1178, 2019. `https://eprint.iacr.org/2019/1178`.

ZFSD22. Philipp Zabka, Klaus-Tycho Foerster, Stefan Schmid, and Christian Decker. A centrality analysis of the lightning network, 2022.

A Motivation for Nirvana

As already discussed in Sect. 6, despite providing numerous improvements to the performance and scalability of public blockchains, existing layer-2 proposals [PD16,Net19,DW15,MBB+19,Hea13,HAB+17,KZF+18] have a fundamental limitation, i.e., collateral depletion. While this issue is somewhat addressed with collateral re-balancing techniques [KG17], this technique imposes additional requirements such as having multiple channels. Moreover, the state of these channels has to be favourable in order to allow funds to be transferred between them. Hence, *collateral reusability* becomes an interesting property for solutions that provide instant finality to payments made with cryptocurrencies. Mavroudis et al. [MWD+20] proposed Snappy, that provides instant transaction finality with a concept called *payment guarantee*. Payment guarantees assure a payee (merchant) that they will receive a payment once the transaction on the underlying blockchain has reached its eventual finality, even if the payer (customer) is malicious. Their design is simple, efficient and most importantly, the collateral provided by a payer is small and *reusable*. This reusablity comes from the abstraction of payment guarantees, where the customer makes a payment on the underlying blockchain and assures the payee about this payment by using irrefutable evidence that this payment will be successful. The design of Snappy is tuned for efficiency and simplicity; hence, it completely overlooks any kind of privacy-preservation. The identities of the customers are known to every participant, along with the transaction amounts and the identity of payees. Snappy's design relies on the availability of data (public transaction details) in order to ensure double-spends cannot happen, hence making their scheme private would not be possible with its current building blocks. Of course, the identity of merchants can be replaced with cryptographic commitments; however, upon opening this commitment on the arbiter smart contract in order to

reclaim their funds, a merchant would link the payment to themselves. For more details, interested readers are referred to [MWD+20].

With Nirvana we improve upon Snappy's idea of payment guarantees and offer collateral reusability along with *privacy-preservation*, namely, customer anonymity, transaction amount obfuscation and transaction unlinkability. The choice of our building blocks is directly in line with our aim to provide the aforementioned privacy properties in an efficient manner. By utilizing the combination of PRF and SPTS, we enable a payer to provide a proof of solvency by utilizing efficient Groth-Sahai NIZK proofs. This is because of the structure-preserving nature of these schemes. By using any other kind of threshold signatures that employs hash functions, we would have to utilize zk-SNARKs to prove the knowledge of pre-image of the hash function. This becomes a problem especially because the time required to generate a proof using zk-SNARKs is linear in the order of circuit size. Hence, the customer would require more time to spend their payment guarantee. We enable proactive double-spending detection efficiently, by assigning a unique set of fixed witnesses to each collateral. Unlike Snappy, transactions in Nirvana do not utilize an Ethereum address as a pseudonym since it is not required. This is because in Snappy, if a witness gets the same transaction from a customer, they can identify it and catch double-spending attempts by just looking at the address. Nirvana borrows concepts from e-cash, and due to our novel randomness-reusable ElGamal encryption we enable both proactive and offline double-spend detection. For proactive double-spending detection, the witnesses constantly merge every existing payment guarantee in their storage with the freshly received one. If any of these guarantees is being double-spent, the guarantees merge together to reveal the identity of the malicious customer. Similar technique can be applied by an offline witness to track cheaters. Please note that the witnesses can remove these guarantees from their storage after a fixed period of time called epoch. To the best of our knowledge we are the first ones to utilize these building blocks and design an instant finality layer-2 scheme that utilizes *reusable* collaterals, while offering several privacy-preserving properties. Admittedly, our system design is somewhat centralized due to the presence of authorities. In order to offer these properties, we need to rely on such a design. However, despite the risks that central hubs entail, Avarikioti et al. [AHWW20] suggested that payment channel networks are more stable and efficient when centralized structures are present. Moreover, Zabka et al. [ZFSD22] show the rising centrality in lightning network, an instantiation of payment channel networks on Bitcoin blockchain. Hence it becomes increasingly important to make such semi-decentralized architectures more trustless, robust and secure. With Nirvana, we propose a trustless, robust, anonymity-preserving and unlinkable semi-decentralized solution that does not require a customer to make multiple channels, or constantly replenish them.

B Cryptographic Building-Blocks

Nirvana heavily relies on cryptography to achieve its goals. In this section, we begin with reviewing the required building blocks, which include Pseudo-Random Functions (PRFs), Shamir Secret Sharing, Digital Signature (DS), Structure Preserving Threshold Signatures (SPTS), Commitments (CO) and Non-Interactive Zero-Knowledge (NIZK) proofs. Finally, we also introduce a novel randomness-reusable variant of the threshold ElGamal encryption to balance the privacy of the users. Next the utilized cryptographical primitives and their security requirements are discussed.

B.1 Pseudo-Random Function (PRF)

Definition B.1 (Pseudo-random Functions [GGM84]). *Let $\mathcal{K}$ and $\mathcal{I}$ be the key and input spaces, respectively. We say a family of functions like $f : \mathcal{K} \times \mathcal{I} \to \mathcal{F}$ is a pseudo-random function (PRF) family if it is efficiently computable and for all PPT distinguishers $\mathcal{D}$ we have,*

$$\left| \Pr_{k \leftarrow\$ \mathcal{K}}[\mathcal{D}^{\mathsf{PRF}_k} \text{ accepts}] - \Pr_{g \leftarrow\$ \mathcal{G}}[\mathcal{D}^{g} \text{ accepts}] \right| \leq \mathsf{negl}(\lambda) \ ,$$

where $\mathcal{D}^{O}$ denotes the output of distinguisher $\mathcal{D}$ when given access to oracle O. It is assumed that the distinguisher can adaptively choose the inputs and $\mathcal{G} : \mathcal{I} \to \mathcal{F}$ is a set of uniformly random functions.

We recall the weak notion of robustness for a PRF function, defined by Damgård et al. [DGK+21], such that no PPT adversary can find a key that produces collisions with a PRF generated by an honest key.

Definition B.2 (Weakly-Robust PRF [DGK+21]). *A PRF scheme, Ψ_{PRF}, under query set $\mathcal{Q} = (x, y) \in \mathcal{I} \times \mathcal{F}$ is weakly-robust if for all PPT adversaries $\mathcal{A}$ we have:*

$$\Pr\left[\begin{matrix} k \leftarrow\$ \mathsf{KGen}(\lambda), (x^*, k^*) \leftarrow\$ \mathcal{A}^{\mathsf{PRF}_k(.)}(1^\lambda) : \\ \exists\ (x, y) \in \mathcal{Q}, \mathsf{PRF}_{k^*}(x^*) = y = \mathsf{PRF}_k(x) \end{matrix}\right] \leq \mathsf{negl}(\lambda) \ .$$

B.2 Shamir Secret Sharing

A (n, t)-Shamir Secret Sharing (SSS) [Sha79] divides a secret s among n shareholders such that each subset of t shareholders can reconstruct secret s and any smaller subset of them learn nothing about the secret. For this purpose, the dealer who knows the secret s forms a polynomial $f(x)$ of degree $t+1$ with a randomly chosen coefficients such that $f(0) = s$. Then the dealer securely provides each shareholder with $s_i = f(i), i \in \{1, \ldots, n\}$. Particularly, each subset of $\mathcal{T} \subset \{1, \ldots, n\}$ with size at least t by pooling their shares can reconstruct the secret s using the Lagrange polynomial interpolation as $s = f(0) = \sum_{i \in \mathcal{T}} s_i L_i^{\mathcal{T}}(0)$, where $L_i^{\mathcal{T}}(x) = \prod_{j \in \mathcal{T}, j \neq i} \frac{x-j}{i-j}$.

B.3 Digital Signatures and SPTS

Digital signatures are an electronic analogue of written signatures that ensure data authentication, and the non-repudiation of the sender. Next we formally define digital signature and list their security requirements.

Definition B.3 (Digital Signature). *For a given security parameter 1^λ, a digital signature consists of the following PPT algorithms defined as follows:*

- $(\mathsf{pp}) \leftarrow \mathcal{DS}.\mathsf{Setup}(1^\lambda)$*: It takes the security parameter 1^λ as input and outputs the set of public parameters* pp*.*
- $(\mathsf{vk}, \hat{\mathsf{sgk}}) \leftarrow \mathcal{DS}.\mathsf{KGen}(\mathsf{pp})$*: This algorithm takes the global public parameters* pp *and returns the pair of signing/verification keys* $(\hat{\mathsf{sgk}}, \mathsf{vk})$ *associated with a message space* $\mathcal{M}$*.*
- $(\sigma) \leftarrow \mathcal{DS}.\mathsf{Sign}(\mathsf{pp}, \hat{\mathsf{sgk}}, m)$*: On input of the signing key* $\hat{\mathsf{sgk}}$ *and a message* $m \in \mathcal{M}$ *This algorithm outputs a signature* σ*.*
- $(0,1) \leftarrow \mathcal{DS}.\mathsf{Vf}(\mathsf{pp}, \mathsf{vk}, \sigma, m)$*: This deterministic algorithm takes as inputs the verification key* vk*, a signature* σ *and* m *and outputs either* 1 *(accept) or* 0 *(reject).*

The primary security requirements for a signature scheme are *correctness* and *unforgeability against chosen message attack*, which are defined as follows:

Definition B.4 (Correctness). *A digital signature scheme, $\Psi_{\mathcal{DS}}$, is called correct, if we have,*

$$\Pr\begin{bmatrix}\forall\ (\hat{\mathsf{sgk}}, \mathsf{vk}) \leftarrow \mathsf{KGen}(\mathsf{pp}), m \in \mathcal{M}: \\ \mathsf{Vf}\left(\mathsf{pp}, \mathsf{vk}, m, \mathsf{Sign}(\mathsf{pp}, \hat{\mathsf{sgk}}, m)\right) = 1\end{bmatrix} \geq 1 - \mathsf{negl}(\lambda)\ .$$

Definition B.5 (EUF-CMA**).** *A digital signature, $\Psi_{\mathcal{DS}}$, is called* EUF-CMA*-secure if for all PPT adversaries $\mathcal{A}$ with an access to the signing oracle $\mathcal{O}_{\mathsf{Sign}}$ we have:*

$$Adv^{\text{EUF-CMA}}_{DS,\mathcal{A}}(\lambda) = \Pr\begin{bmatrix}\forall\ (\mathsf{pp}) \leftarrow \mathsf{Setup}(1^\lambda), (\hat{\mathsf{sgk}}, \mathsf{vk}) \leftarrow \mathsf{KGen}(\mathsf{pp}), \\ (\sigma^*, m^*) \leftarrow_{\$} \mathcal{A}^{\mathcal{O}_{\mathsf{Sign}}}(\mathsf{pp}, \mathsf{vk}): \\ m^* \notin \mathcal{Q}_{msg} \wedge \mathsf{Vf}(\mathsf{vk}, \sigma^*, m^*) = 1\end{bmatrix} \leq \mathsf{negl}(\lambda)\ ,$$

where the signing oracle $\mathcal{O}_{\mathsf{Sign}}$ takes a message $m \in \mathcal{M}$, runs $\mathsf{Sign}(\mathsf{pp}, \hat{\mathsf{sgk}}, m)$ and adds the message to a query set $\mathcal{Q}_{msg}$.

A digital signature is called structure-preserving [AFG+10], when it preserves the group structure over bilinear group setting, if it satisfies the following requirements:

- The verification key consists of $\mathbb{G}_1$ and $\mathbb{G}_2$ group elements.
- The signature consists of group elements in $\mathbb{G}_1$ and $\mathbb{G}_2$.

– The messages are composed of $\mathbb{G}_1$ and $\mathbb{G}_2$ elements.
– Only $\mathbb{G}_1$ and $\mathbb{G}_2$ membership and pairing product equations of the form of $\prod_i \prod_j \hat{e}(G_i, H_j)^{c_{i,j}} = 1_T$ need to be considered in the verification algorithm, where $G_i \in \mathbb{G}_1$ and $H_j \in \mathbb{G}_2$ and $c_{i,j} \in \mathbb{Z}_{\mathsf{p}}$.

By avoiding structure-destroying operations such as hash functions, SPS are able to construct efficient schemes when combined with other primitives such as Zero-Knowledge proof systems. In a SPS, both signed messages and signatures are group elements that can be used to verify the validity of a signature by performing pairing-product equations. These unique properties make the SPS schemes attractive for a variety of privacy-preserving applications, like anonymous credentials [Fuc11,CDHK15], anonymous e-cash [BCF+11] or access control encryption [SP21]. Moreover, these signatures are efficiently re-randomizable under the knowledge of a secret randomness such that the re-randomized and original signatures are computationally indistinguishable. We utilise this property of re-randomization to ensure unlinkability of transactions in Nirvana.

Given the fact that the SPS relies on a single issuer then it does not meet Nirvana's desirable properties. In this aim, we recall the definition of Structure-Preserving Threshold Signatures from a recent work of Sedaghat et al. [SSKP22]: it preserves the SPS's properties while mitigating the needed trust to a single entity.

Definition B.6 (Structure-Preserving Threshold Signatures [SSKP22]). *For a given security parameter 1^λ and an asymmetric bilinear group, a (n,t)-SPTS over message space $\mathcal{M}$, consists of the following PPT algorithms:*

– $(\mathsf{pp}) \leftarrow \mathcal{SPTS}.\mathsf{Setup}(1^\lambda)$*: It takes the security parameter 1^λ as input and returns the set of global public parameters* pp *as the output.*
– $(\vec{\hat{\mathsf{sgk}}}, \vec{\mathsf{vk}}, \mathsf{vk}) \leftarrow \mathcal{SPTS}.\mathsf{KGen}(\mathsf{pp}, t, n)$*: The key generation is a distributed and interactive algorithm that takes the global public parameters* pp*, and integers $t, n \in \mathsf{poly}(1^\lambda)$ s.t. $1 \leq t \leq n$, as inputs. It then returns the vectors of secret signing keys $\vec{\hat{\mathsf{sgk}}} = (\hat{\mathsf{sgk}}_1, \ldots, \hat{\mathsf{sgk}}_n)$ and verification keys $\vec{\mathsf{vk}} = (\mathsf{vk}_1, \ldots, \mathsf{vk}_n)$ along with a common verification key* vk*.*
– $(\sigma_i) \leftarrow \mathcal{SPTS}.\mathsf{Par\text{-}Sign}(\mathsf{pp}, \hat{\mathsf{sgk}}_i, m)$*: It takes the public parameter* pp*, the i^{th} signing key $\hat{\mathsf{sgk}}_i$ and a message $m \in \mathcal{M}$ as inputs. The partial signing algorithm then returns the partial signature σ_i as output.*
– $(0,1) \leftarrow \mathcal{SPTS}.\mathsf{Par\text{-}Vf}(\mathsf{pp}, \mathsf{vk}_i, m, \sigma_i)$*: It takes the i^{th} verification key vk_i, message $m \in \mathcal{M}$ and partial signature σ_i as inputs. It then returns either 1 (accept) or 0 (reject).*
– $(\sigma, \perp) \leftarrow \mathcal{SPTS}.\mathsf{Recon}(\mathsf{pp}, \{i, \sigma_i\}_{i \in \mathcal{T}})$*: It takes public parameters* pp *and successfully verified partial signatures $\{i, \sigma_i\}$ over subset $\mathcal{T} \in \{1, \ldots, n\}$ as inputs. It returns a reconstructed signature σ if $|\mathcal{T}| > t$, otherwise it responds with $\perp$.*
– $(0,1) \leftarrow \mathcal{SPTS}.\mathsf{Vf}(\mathsf{pp}, \mathsf{vk}, m, \sigma)$*: It takes the public parameters* pp*, verification key* vk*, a message $m \in \mathcal{M}$ and a reconstructed signature σ as inputs. It outputs either 1 (accept) or 0 (reject).*

As it is discussed in [SSKP22], two main security requirements for a SPTS scheme over the indexed DH message spaces are correctness and threshold existential unforgeability under chosen indexed message attacks. We refer the readers to this paper for more details.

B.4 Commitment schemes

A commitment scheme [BCC88] is a strong cryptographic primitive, which allows a committer to commit to a secret value with two main security properties, i.e., *Perfect Hiding* and *Computational Binding*. Informally, perfect hiding guarantees that the commitment does not reveal any information about the hidden committed value. Computational hiding ensures that a committer cannot open a commitment under two distinct messages.

Definition B.7 (Commitment schemes [BCC88]). *A commitment scheme Ψ_{CO} over the message space of $\mathcal{M}$ and opening space of $\mathcal{T}$ consists of the following PPT algorithms:*

- $(\mathsf{pp}) \leftarrow \mathcal{CO}.\mathsf{Setup}(1^\lambda)$*: The setup algorithm takes the security parameter 1^λ as input and returns the public parameters* pp *as output.*
- $(\mathsf{com}) \leftarrow \mathcal{CO}.\mathsf{Com}(\mathsf{pp}, m)$*: The commitment algorithm takes the public parameters* pp *and a message $m \in \mathcal{M}$ as inputs, and outputs a commitment $\mathsf{com} \in \mathcal{C}$ computed under the random opening value $\tau \in \mathcal{T}$.*
- $(0,1) \leftarrow \mathcal{CO}.\mathsf{Vf}(\mathsf{pp}, \mathsf{com}, m', \tau')$*: The verification algorithm is a deterministic algorithm that given commitment $\mathsf{com} \in \mathcal{C}$, public parameters* pp*, a message $m' \in \mathcal{M}$ and an opening value $\tau' \in \mathcal{T}$ as inputs, returns a bit that indicates either accept (1) or reject (0).*

The primary security requirements for a commitment can be defined as follows:

Definition B.8 (Correctness). *A commitment scheme, Ψ_{CO}, satisfies* correctness*, if we have:*

$$\Pr\begin{bmatrix}\forall\ m \in \mathcal{M}\ \ \wedge\ \ (\mathsf{pp}) \leftarrow \mathsf{Setup}(1^\lambda) : \\ \mathsf{Vf}\left(\mathsf{pp}, m, \mathsf{Com}(\mathsf{pp}, m; \tau), \tau\right) = 1\end{bmatrix} \geq 1 - \mathsf{negl}(\lambda)\ .$$

Definition B.9 (Computationally Hiding). *A commitment, Ψ_{CO}, satisfies Computationally Hiding, if for all PPT adversaries $\mathcal{A}$ we have:*

$$\left|2\Pr\begin{bmatrix}(\mathsf{pp}) \leftarrow \mathsf{Setup}(1^\lambda), (m_0, m_1) \leftarrow\!\$\, \mathcal{A}^{\mathsf{Com}(.)}(\mathsf{pp}), b \leftarrow\!\$\, \{0,1\}, \\ (\mathsf{com}_b) \leftarrow \mathsf{Com}(\mathsf{pp}, m_b), b' \leftarrow \mathcal{A}(\mathsf{pp}, \mathsf{com}_b) : b == b'\end{bmatrix} - 1\right| \leq \mathsf{negl}(\lambda)\ .$$

The commitment scheme is called perfectly hiding if the above probability is equal to 0.

Definition B.10 (Computationally Binding). *A cryptographic commitment scheme, Ψ_{com}, meets computationally binding security, if for all PPT adversaries $\mathcal{A}$ we have,*

$$\Pr\left[\begin{matrix}(\mathsf{pp}) \leftarrow \mathsf{Setup}(1^\lambda), ((\mathsf{com}, m_0, \tau_0), (\mathsf{com}, m_1, \tau_1)) \leftarrow\!\$\, \mathcal{A}^{\mathsf{Com}(.)}(\mathsf{pp}) : \\ \mathsf{Vf}(\mathsf{pp}, \mathsf{com}, m_0, \tau_0) = \mathsf{Vf}(\mathsf{pp}, \mathsf{com}, m_1, \tau_1) = 1 \wedge m_0 \neq m_1\end{matrix}\right] \leq \mathsf{negl}(\lambda) \ .$$

The commitment scheme is called perfectly binding if the above probability is 0.

B.5 Threshold Encryption

Definition B.11 (Threshold Encryption). *A (n, t)-threshold encryption (TE) scheme, Ψ_{TE}, over the message space $\mathcal{M}$ and ciphertext space $\mathcal{C}$, consists of five PPT algorithms defined as follows:*

- $(\mathsf{pp}, \hat{\mathsf{sk}}) \leftarrow \mathcal{TE}.\mathsf{Setup}(1^\lambda, \mathcal{R}, t)$*: This probabilistic algorithm takes the security parameter 1^λ, the set of receivers $\mathcal{R}$ and an integer $t \in \mathsf{poly}(\lambda)$ as inputs. It returns the secret key $\hat{\mathsf{sk}}$ and the corresponding public parameters pp as outputs.*
- $(\hat{\mathsf{sk}}_i, \mathsf{pk}_i) \leftarrow \mathcal{TE}.\mathsf{KGen}(\hat{\mathsf{sk}}, i)$*: Key generation is a probabilistic algorithm that takes the secret key $\hat{\mathsf{sk}}$ along with an index $i \in \mathcal{R}$ as inputs and returns secret key $\hat{\mathsf{sk}}_i$ as output.*[12]
- $(\mathtt{Ct}) \leftarrow \mathcal{TE}.\mathsf{Enc}(\mathsf{pp}, m)$*: The encryption algorithm takes the public parameters pp, and a message $m \in \mathcal{M}$ as inputs. It returns ciphertext $\mathtt{Ct} \in \mathcal{C}$ as output. When we want to assign a specific value to the random integer r, we write $\mathsf{Enc}(\mathsf{pp}, m; r)$.*
- $(\mathsf{pd}_j) \leftarrow \mathcal{TE}.\mathsf{PDec}(\mathsf{pp}, \mathsf{sk}_j, \mathtt{Ct})$*: The partial decryption algorithm is run by receiver $j \in \mathcal{R}$ and takes the public parameters pp, the secret key $\hat{sk}_j$ of the receiver j and a ciphertext $\mathtt{Ct}$ as inputs. It returns a partially decrypted ciphertext pd_j as output.*
- $(\perp, m) \leftarrow \mathcal{TE}.\mathsf{Dec}(\mathsf{pp}, \mathtt{Ct}, \{\mathsf{pd}_j\}_{j\in\mathcal{K}})$*: The decryption algorithm takes the public parameters pp, a ciphertext $\mathtt{Ct}$ and the partially decrypted ciphertexts $\{\mathsf{pd}_j\}_{j\in\mathcal{K}}$ as inputs. If $|\mathcal{K}| \geq t$, it returns $m \in \mathcal{M}$, else it responds by $\perp$.*

The primary security requirements for a (n, t)-threshold encryption are correctness and *static semantic security* and *partial decryption simulatability* based on the static security definitions of Reyzen et al. [RSY21].

Definition B.12 (Correctness). *A (n, t)-threshold encryption, Ψ_{TE}, for all λ and messages $m \in \mathcal{M}$ and $\mathcal{R}$ is called correct if for any $|\mathcal{K}| \geq t$ we have:*

$$\Pr\left[\begin{matrix}(\mathsf{pp}, \hat{\mathsf{sk}}) \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{R}, t), (\hat{\mathsf{sk}}_i, \mathsf{pk}_i) \leftarrow \mathsf{KGen}(\hat{\mathsf{sk}}, i), \\ \mathsf{Dec}\left(\mathsf{pp}, \{\mathsf{PDec}(\hat{\mathsf{sk}}_j, \mathtt{Ct})\}_{j\in\mathcal{K}}, \mathsf{Enc}(\mathsf{pp}, m)\right) = m\end{matrix}\right] \geq 1 - \mathsf{negl}(\lambda) \ .$$

[12] Note that in the notion of a standard threshold encryption, this algorithm does not return public key while in the randomness-reusable threshold encryption no secret key is returned.

Definition B.13 (Static Semantic Security [RSY21]). *A (n,t)-threshold encryption, Ψ_{TE}, is said to be (n,t)-statically semantic secure (SSS) if for all PPT adversaries $\mathcal{A}$ in winning the following experiment we have* $\Pr[\text{EXP}_{\mathcal{A}}^{SSS}(1^\lambda,\mathcal{R},t)=1] \leq 1/2-\mathsf{negl}(\lambda)$. *Where adversary $\mathcal{A}$ has access to a partial decryption oracle and can obtain up to $t-1$ partially decrypted values of the given ciphertext.*

$\text{EXP}_{\mathcal{A}}^{SSS}(1^\lambda,\mathcal{R},t)$

$\mathcal{C} \leftarrow \mathcal{A}(1^\lambda,\mathcal{R},t)$
$(\hat{\mathsf{sk}}_i) \leftarrow \mathcal{TE}.\mathsf{KGen}(\mathsf{msk},i)$ For $i \in [n]$;
$(m_0,m_1) \leftarrow \mathcal{A}^{\mathcal{O}}(\{\hat{\mathsf{sk}}_i\}_{i\in\mathcal{C}})$;
$b \leftarrow\$ \{0,1\}$,
$(\mathtt{Ct}_b) \leftarrow \mathcal{TE}.\mathsf{Enc}(\mathsf{pp},m_b)$;
$b' \leftarrow\$ \mathcal{A}^{\mathcal{O}}(\mathtt{Ct}_b)$;
if $(b'=b \wedge |m_0| \neq |m_1| \wedge |\mathcal{R}\cap\mathcal{K}| < t)$:
  return 1

A threshold variant of the ElGamal encryption. An additively homomorphic threshold encryption was proposed by Desmedt and Frankel [DF90] as a variation of the ElGamal encryption over $(\mathbb{G},g,\mathsf{p})$ such that $\mathbb{G}$ is a cyclic group of prime order p with generator g. It consists of the following algorithms:

- $(\mathsf{pp},\hat{\mathsf{sk}}) \leftarrow \mathcal{TE}.\mathsf{Setup}(1^\lambda,\mathcal{R},t)$: This probabilistic algorithm picks an integer $s \leftarrow\$ \mathbb{Z}_{\mathsf{p}}$ and a random polynomial $f \leftarrow\$ \mathbb{F}[X]$ of degree $t-1$ s.t. $f(0)=s$ and sets $h := g^s$. It returns the public parameter $\mathsf{pp}=(h,g)$ and the secret key $\hat{\mathsf{sk}}=(f,s)$.
- $(\hat{\mathsf{sk}}_i) \leftarrow \mathcal{TE}.\mathsf{KGen}(\hat{\mathsf{sk}},i)$: The key generation algorithm takes $\mathsf{msk}=(f,s)$, and receiver index $i \in \mathcal{R}$ as inputs. It returns the secret key $\hat{\mathsf{sk}}_i = f(i)$ as output.
- $(\mathtt{Ct}) \leftarrow \mathcal{TE}.\mathsf{Enc}(\mathsf{pp},m)$: The encryption algorithm to encrypt a message $m \in \mathcal{M}$ samples a random integer $r \leftarrow\$ \mathbb{Z}_{\mathsf{p}}$ and returns the ciphertext $\mathtt{Ct} := (u,v) := (g^r, m\cdot h^r)$.
- $(\mathsf{pd}_j) \leftarrow \mathcal{TE}.\mathsf{PDec}(\mathsf{pp},\hat{\mathsf{sk}}_j,\mathtt{Ct})$: A receiver who knows the secret key $\hat{\mathsf{sk}}_j$, partially decrypts the ciphertext by computing $\mathsf{pd}_j = u^{\mathsf{sk}_j}$.
- $(m,\perp) \leftarrow \mathcal{TE}.\mathsf{Dec}(\mathsf{pp},\mathtt{Ct},\{\mathsf{pd}_j\}_{j\in\mathcal{K}})$: The decryption algorithm takes public parameters pp, ciphertext $\mathtt{Ct}=(u,v)$, the partially decrypted values $\{\mathsf{pd}_j\}_{j\in\mathcal{K}}$ as inputs. If $|\mathcal{K}| \geq t$, it computes $u^s : z = \prod_{j\in\mathcal{K}} \mathsf{pd}_j^{L_j^{\mathcal{K}}(0)}$, where $L_j^{\mathcal{K}}(0)$ is a Lagrange coefficient according to subset $\mathcal{K}$ and returns $m := v/z$, else it responds with $\perp$

Next we define a randomness-reusable variation of the ElGamal threshold encryption, where the encryptor can encrypt a single message multiple times under the same randomness.

- $(\mathsf{pp},\hat{\mathsf{sk}}) \leftarrow r\mathcal{TE}.\mathsf{Setup}(1^\lambda,\mathcal{R},t)$: It samples a random integer $s \leftarrow\$ \mathbb{Z}_{\mathsf{p}}$ and a random polynomial $f \leftarrow\$ \mathbb{F}[X]$ of degree $t-1$ s.t. $f(0)=s$ and sets $h=g^s$. It returns $\mathsf{pp}=(g,h)$ and $\hat{\mathsf{sk}}=(f,s)$ as outputs.

- $(\hat{\mathsf{sk}}_i, \mathsf{pk}_i) \leftarrow r\mathcal{TE}.\mathsf{KGen}(\hat{\mathsf{sk}}, i)$: It takes $\hat{\mathsf{sk}}$ and an index $i \in \mathcal{R}$ as inputs and computes the secret key $\hat{\mathsf{sk}}_i = f(i)$ and the corresponding public key $\mathsf{pk}_i = g^{\hat{\mathsf{sk}}_i}$. It then returns $(\hat{\mathsf{sk}}_i, \mathsf{pk}_i)$ as output.
- $(\mathtt{Ct}) \leftarrow r\mathcal{TE}.\mathsf{Enc}(\mathsf{pp}, \{\mathsf{pk}_i\}_{i\in\mathcal{K}}, m)$: It takes $m \in \mathcal{M}$, the set of public parameters of receivers intended in a chosen set $\mathcal{K} \subset \mathcal{R}$ as inputs. It samples a random integer $r \leftarrow\$ \mathbb{Z}_{\mathsf{p}}$ and computes $\mathtt{Ct} := (u, \{\mathtt{Ct}_j\}_{j\in\mathcal{K}}, v) := (g^r, \{u_j = \mathsf{pk}_j^r\}_{j\in\mathcal{K}}, mh^r)$.
- $(\mathsf{pd}_j) \leftarrow r\mathcal{TE}.\mathsf{PDec}(\mathsf{pp}, \hat{\mathsf{sk}}_j, \mathtt{Ct})$: It takes j^{th} receiver's secret key $\hat{\mathsf{sk}}_j$ s.t. $j \notin \mathcal{K}$ and then partially decrypts the ciphertext by computing $\mathsf{pd}_j = u^{\hat{\mathsf{sk}}_j}$ and returns pd_j as output.
- $(m, \perp) \leftarrow r\mathcal{TE}.\mathsf{Dec}(\mathsf{pp}, \mathtt{Ct}, \{\mathsf{pd}_j\}_{j\in\mathcal{K}'})$: The decryption algorithm takes $\mathtt{Ct}$, the partially decrypted values $\{\mathsf{pd}_j\}_{j\in\mathcal{K}'}$ such that $,\mathcal{K}, \mathcal{K}' \subset \mathcal{R}$. If $|\mathcal{K} \cup \mathcal{K}'| \geq t$, it computes $u^s : z = \prod_{j\in\mathcal{K}} u_j^{L_j^{\mathcal{K}}(0)} \prod_{j\in\mathcal{K}'} \mathsf{pd}_j^{L_j^{\mathcal{K}'}(0)}$ and returns $m := v/z$, otherwise it responds with $\perp$.

Similar to the original threshold ElGamal scheme, the ciphertext in this scheme leaks no information about the message as long as the threshold is not reached. The sender selects a subset $\mathcal{K} \subset \mathcal{R}$ and the subset $\mathcal{K}'$ can be considered as a complement set for $\mathcal{K}$ to reach the threshold t such that $|\mathcal{K} \cup \mathcal{K}'| \geq t$.

B.6 Non-Interactive Zero-Knowledge arguments

Zero-Knowledge proofs [GMR89] are two-party protocols, which are a fundamental and powerful cryptographic tool. They allow a prover to convince the verifier about the validity of a statement without revealing any other information. Non-Interactive Zero-Knowledge proofs remove the interaction between the parties in two possible settings, either the Random Oracle Model (ROM) [FS87] or the Common Reference String (CRS) model [BFM88]. We recall the definition of NIZK arguments[13] in the CRS model, in which the prover is computationally bounded to ensure the soundness. Hence, for security parameter 1^λ, let $\mathcal{R}$ be a relation generator, such that $\mathcal{R}(1^\lambda)$ returns an efficiently computable binary relation $\mathbf{R_L} = \{(\mathsf{x}, \mathsf{w})\}$, where x is the instance and w is the corresponding witness. Let $\mathbf{L} = \{\mathsf{x} : \exists\, \mathsf{w} \mid (\mathsf{x}, \hat{\mathsf{w}}) \in \mathbf{R_L}\}$ be the NP-language consisting of the statements in relation $\mathbf{R_L}$.

Definition B.14 (NIZK arguments). *Formally, a NIZK argument Ψ_{NIZK} for $\mathcal{R}$ consists of a tuple of PPT algorithms $\mathcal{ZK}.(\mathsf{K}_{\vec{\mathsf{crs}}}, \mathsf{P}, \mathsf{Vf}, \mathsf{Sim})$, defined as follows:*

- $(\vec{\mathsf{crs}}, \hat{\mathsf{ts}}, \hat{\mathsf{te}}) \leftarrow \mathcal{ZK}.\mathsf{K}_{\vec{\mathsf{crs}}}(1^\lambda, \mathbf{R_L})$*: The CRS generator as a probabilistic algorithm takes the security parameter 1^λ and relation $\mathbf{R_L}$ as inputs. It then generates common reference string $\vec{\mathsf{crs}}$ by sampling a simulation trapdoor $\hat{\mathsf{ts}}$ and an extraction trapdoor $\hat{\mathsf{te}}$. It keeps the trapdoors $(\hat{\mathsf{te}}, \hat{\mathsf{ts}})$ hidden while publishes $\vec{\mathsf{crs}}$.*

[13] The CRS does not depend on the language distribution or language parameters.

– $(\pi, \perp) \leftarrow \mathcal{ZK}.\mathsf{P}(\mathbf{R_L}, \vec{\mathsf{crs}}, \mathsf{x}, \hat{\mathsf{w}})$*: Prove as a probabilistic algorithm takes the CRS,* $\vec{\mathsf{crs}}$*, and a pair of statement and witness* $(\mathsf{x}, \hat{\mathsf{w}})$ *as inputs. If* $(\mathsf{x}, \hat{\mathsf{w}}) \in \mathbf{R_L}$ *it returns a proof* π*, otherwise it responds with* $\perp$*. This algorithm sometimes is denoted by* $\mathsf{PoK}\{\hat{\mathsf{w}} \mid (\mathsf{x}, \hat{\mathsf{w}}) \in \mathbf{R_L}\}$*.*
– $(0, 1) \leftarrow \mathcal{ZK}.\mathsf{Vf}(\mathbf{R_L}, \vec{\mathsf{crs}}, \mathsf{x}, \pi)$*: Verification as a deterministic algorithm takes CRS,* $\vec{\mathsf{crs}}$*, and a pair of statement and proof* (x, π) *as inputs. It either returns* 1 *(accept) or* 0 *(reject).*
– $(\pi') \leftarrow \mathcal{ZK}.\mathsf{Sim}(\mathbf{R_L}, \vec{\mathsf{crs}}, \hat{\mathsf{ts}}, \mathsf{x})$*: The Simulator algorithm takes the tuple* $(\mathbf{R_L}, \vec{\mathsf{crs}}, \mathsf{ts}, \mathsf{x})$ *as input and without knowing the corresponding secret witness, outputs a simulated proof* π' *s.t. it is computationally indistinguishable from* π*.*

Definition B.15 (Completeness). *A NIZK argument* Ψ_{NIZK} *is called* complete *for relation* $\mathbf{R_L} \in \mathcal{R}$*, if for all security parameters* 1^λ *and* $(\mathsf{x}, \hat{\mathsf{w}}) \in \mathbf{R_L}$*, we have:*

$$\Pr\left[\begin{matrix}(\vec{\mathsf{crs}}, \hat{\mathsf{ts}}, \hat{\mathsf{te}}) \leftarrow \mathsf{K}_{\vec{\mathsf{crs}}}(1^\lambda, \mathbf{R_L}) : \\ \mathsf{Vf}(\mathbf{R_L}, \vec{\mathsf{crs}}, \mathsf{x}, \mathsf{P}(\mathbf{R_L}, \vec{\mathsf{crs}}, \mathsf{x}, \hat{\mathsf{w}})) = 1\end{matrix}\right] \geq 1 - \mathsf{negl}(\lambda) \ .$$

Definition B.16 (Soundness). *A NIZK argument,* Ψ_{NIZK}*, is* Sound *for any relation* $\mathbf{R_L} \in \mathcal{R}$*, if for all PPT adversaries* $\mathcal{A}$*, we have:*

$$\Pr\left[\begin{matrix}(\vec{\mathsf{crs}}, \hat{\mathsf{ts}}, \hat{\mathsf{te}}) \leftarrow \mathsf{K}_{\vec{\mathsf{crs}}}(1^\lambda, \mathbf{R_L}), (\mathsf{x}, \pi) \leftarrow \mathcal{A}(\mathbf{R_L}, \vec{\mathsf{crs}}) : \\ \mathsf{Vf}(\mathbf{R_L}, \vec{\mathsf{crs}}, \mathsf{x}, \pi) = 1 \wedge \mathsf{x} \notin \mathbf{L_R}\end{matrix}\right] \leq \mathsf{negl}(\lambda) \ .$$

Definition B.17 (Statistically Zero-Knowledge). *A NIZK argument,* Ψ_{NIZK}*, is called* statistically Zero-Knowledge*, if for all security parameter* 1^λ*, and all PPT adversaries* $\mathcal{A}$ *we have,* $\varepsilon_0^{unb} \approx_\lambda \varepsilon_1^{unb}$*, where,*

$$\varepsilon_b = \Pr\left[(\vec{\mathsf{crs}}, \hat{\mathsf{ts}}, \hat{\mathsf{te}}) \leftarrow \mathsf{K}_{\vec{\mathsf{crs}}}(1^\lambda, \mathbf{R_L}) : \mathcal{A}^{\mathsf{O}_b(\cdot,\cdot)}(\mathbf{R_L}, \vec{\mathsf{crs}}) = 1\right] \cdot$$

Here, the oracle $\mathcal{O}_0(\mathsf{x}, \hat{\mathsf{w}})$ *returns* $\perp$ *(reject) if* $(\mathsf{x}, \hat{\mathsf{w}}) \notin \mathbf{R_L}$*, otherwise it returns* $\mathcal{ZK}.\mathsf{P}(\mathbf{R_L}, \vec{\mathsf{crs}}, \mathsf{x}, \hat{\mathsf{w}})$*. Similarly,* $\mathcal{O}_1(\mathsf{x}, \hat{\mathsf{w}})$ *returns* $\perp$ *(reject) if* $(\mathsf{x}, \hat{\mathsf{w}}) \notin \mathbf{R_L}$*, otherwise it returns* $\mathcal{ZK}.\mathsf{Sim}(\mathbf{R_L}, \vec{\mathsf{crs}}, \mathsf{x}, \hat{\mathsf{ts}})$*.*

Definition B.18 (Computational Knowledge-Soundness). *A NIZK argument,* Ψ_{NIZK}*, is called computationally (adaptively)* knowledge-sound for $\mathcal{R}$*, if for all PPT adversary* $\mathcal{A}$ *and* $(\mathbf{R_L}) \in \mathcal{R}$*, there exists an extractor* $\mathsf{Ext}_{\mathcal{A}}$*, s.t. for all* 1^λ *we have,*

$$\Pr\left[\begin{matrix}(\vec{\mathsf{crs}}, \hat{\mathsf{ts}}, \hat{\mathsf{te}}) \leftarrow \mathsf{K}_{\vec{\mathsf{crs}}}(1^\lambda, \mathbf{R_L}), (\mathsf{x}, \pi) \leftarrow \mathcal{A}(\mathbf{R_L}, \vec{\mathsf{crs}}), \\ (\hat{\mathsf{w}}) \leftarrow \mathsf{Ext}_{\mathcal{A}}(\mathbf{R_L}, \vec{\mathsf{crs}}, \hat{\mathsf{te}}, \pi) : (\mathsf{x}, \hat{\mathsf{w}}) \notin \mathbf{R_L} \ \wedge \ \mathsf{Vf}(\mathbf{R_L}, \vec{\mathsf{crs}}, \mathsf{x}, \pi) = 1\end{matrix}\right] \leq \mathsf{negl}(\lambda) \ .$$

C Omitted Proofs

C.1 Proof of Theorem 3.1

Proof. To prove this theorem we demonstrate that the proposed scheme is secure w.r.t two main security properties namely *Anonymity of honest customers and Unlinkability of payment guarantees*, and *payment certainty*.

Anonymity of honest customers and Unlinkability of payments. For each payment request, the customer should transfer a tuple $\mathcal{T}[\pi, \mathsf{x}, R_t]$ where R_t is the auxiliary data at time slot t to convince the merchant and the group of witnesses about the uniqueness of a collateral. To prove that NIRVANA preserves the anonymity of honest customers and provides unlinkability of payments we show that no PPT adversary $\mathcal{A}$, by providing two pair of challenge secret keys/collateral keys $(\hat{\mathsf{sk}}_0^*, \hat{k}_0^*)$ and $(\hat{\mathsf{sk}}_1^*, \hat{k}_1^*)$, can distinguish between $(\pi_0, \mathsf{x}_0, R_{t,0})$ and $(\pi_1, \mathsf{x}_1, R_{t,1})$ as the output of the spending algorithm. This property is guaranteed because of three main security properties for the given primitives: Zero-Knowledge of the NIZK proof, computationally hiding property of commitment scheme, static semantically secure property of randomness-reusable threshold encryption in bilinear groups and also the weakly robust for the given PRF.

Let the hybrid H^b be the case where the *Anonymity* experiment, $\text{EXP}_{\mathcal{A}}^{\text{ANON}}(\lambda, b)$ is run for $b = \{0, 1\}$. In this case, we form a sequence of hybrids and show that each of the successive hybrids are computationally indistinguishable from the preceding ones.

- **Hybrid H_1^b:** In this game, we modify H^b by creating the challenge NIZK proof π_b and running $\pi_b' \leftarrow \mathcal{ZK}.\mathsf{Sim}(\vec{\mathsf{crs}}, \hat{\mathsf{ts}}, \mathsf{x}_b)$.

The Zero-Knowledge property of NIZK arguments provided in Def. B.17 guarantees that this experiment is indistinguishable from the one for H^b.

- **Hybrid H_b^2:** In this game, we modify H_1^b by committing $\hat{\mathsf{sk}}_{1-b}^*$ instead of $\hat{\mathsf{sk}}_b^*$.

According to the hiding property of the given commitment scheme, this experiment is indistinguishable from H_b^1.

- **Hybrid H:** In this game, we modify H_b^2 by assuming the challenger runs the threshold encryption algorithm under the message m_{1-b} instead of m_b.

According to the Static Semantic Security property of the proposed randomness-reusable Threshold encryption, this experiment is indistinguishable from H_b^2. To be more concrete, $\mathcal{A}$ cannot distinguish between $\mathtt{Ct}_b$ and $\mathtt{Ct}_{1-b}$ as long as no twin ciphertext is generated even if the proofs are simulated. Thereby we have, $H_0 \approx_\lambda H_0^1 \approx_\lambda H_0^2 \approx_\lambda H \approx_\lambda H_1^1 \approx_\lambda H_1^2 \approx_\lambda H_1$.

To conclude this security property for the proposed construction, it is straightforward to show that the output of a PRF under two distinct keys is computationally indistinguishable and no PPT adversary can distinguish $R_{t,0}$ and $R_{t,1}$.

Payment Certainty. We prove this property by contradiction. Let there is a PPT adversary $\mathcal{A}$ that can break the payment certainty of the scheme and pass the verification phase without meeting the corresponding requirements. The proof relies on the existence of a weakly-robust PRF, a *Knowledge Sound* NIZK argument, an existentially unforgeable SPS construction, computationally binding of commitment scheme, and a Statically Semantic Secure randomness-reusable Threshold encryption. Having played a sequence of indistinguishable games between $\mathcal{B}_{\text{WR}}, \mathcal{B}_{\text{EUF-CMA}}, \mathcal{B}_{\text{KS}}, \mathcal{B}_{\text{CB}}, \mathcal{B}_{\text{SSS}}$ and a PPT adversary $\mathcal{A}$, we gradually turn the payment certainty security game into the security features of the underlying primitives.

- **Game G_0:** In the first security game, let $\mathcal{A}$ forms a challenge transaction τ^* such that $\sum \mathcal{L}_c + \tau^* > \mathsf{col}_{\mathcal{A}}$ return a valid pair (π^*, x^*) with a non-negligible advantage ϵ. By contradiction, we assume $\mathcal{A}$ can win this game with a non-negligible advantage ϵ and we can write, $Adv^{\mathsf{PC}}_{\text{NIRVANA}}(\mathcal{A}) = \Pr[\mathcal{A} \text{ Wins } G_0] \geq \epsilon$.
- **Game G_1:** In this game, we modify G_0 such that the existence of an extraction trapdoor is assumed. In this case, there exists an extractor that takes te and the received tuple (π^*, x^*) as inputs, and returns the corresponding witness $(\hat{\mathsf{w}}^*) \leftarrow \mathsf{Ext}(\hat{\mathsf{te}}, \mathsf{x}^*, \pi^*)$ such that $\mathsf{w}^* = (\mathsf{cert}^*, \mathtt{ID}^*, r_t^*, k^*)$. The indistinguishability of G_0 and G_1 can be proven via the *Knowledge Extraction* property of NIZK arguments, defined in Def. B.18. This property guarantees the existence of an efficient extractor under non-falsifiable assumptions and we can write, $Adv^{\mathsf{PC}}_{\text{NIRVANA}}(\mathcal{A}) = \Pr[\mathcal{A} \text{ Wins } G_0] \approx \Pr[\mathcal{A} \text{ Wins } G_1]$ and this advantage consequently depends on two possible cases,

  $$\Pr[\mathcal{A} \text{ Wins } G_1] = \Pr[\mathcal{A} \text{ Wins } G_1 : (\mathsf{w}^*, \mathsf{x}^*) \in \mathbf{R_L}] + \Pr[\mathcal{A} \text{ Wins } G_1 : (\mathsf{w}^*, \mathsf{x}^*) \notin \mathbf{R_L}] \ .$$

  The probability of an adversary in the latter case can be bounded by the advantage a NIZK's knowledge soundness adversary faces.

  $$Adv^{\mathsf{PC}}_{\text{NIRVANA}}(\mathcal{A}) \leq \Pr[\mathcal{A} \text{ Wins } G_1 : (\mathsf{w}^*, \mathsf{x}^*) \in \mathbf{R_L}] + Adv^{\text{KS}}_{\text{NIZK}}(\mathcal{B}_{\text{KS}}) \, .$$

  Hence, the adversary can win the game when the event of $(\mathsf{w}^*, \mathsf{x}^*) \in \mathbf{R_L}$ occurs.
- **Game G_2:** The challenger for the payment certainty security game can modify G_1 to an attacker against the weakly-robust PRF security game. The intended key k^* is either a valid key $k^* \in \mathcal{K}$ or it is generated under a random key $k^* \notin \mathcal{K}$. The latter case will be bounded by the advantage of $\mathcal{B}_{\text{WR}}$ attacker, then we can write,

  $$\Pr[\mathcal{A} \text{ Wins } G_2] = \Pr[\mathcal{A} \text{ Wins } G_2 : k^* \in \mathcal{K}] + \Pr[\mathcal{A} \text{ Wins } G_2 : k^* \notin \mathcal{K}] \leq$$
  $$\Pr[\mathcal{A} \text{ Wins } G_2 : k^* \in \mathcal{K}] + Adv^{\text{WR}}_{\text{PRF}}(\mathcal{B}_{\text{WR}}) \ .$$

- **Game G_3:** This is the game G_2, except for a valid pair of witness and statement in $\mathbf{R_L}$ and a valid PRF key in $\mathcal{K}$, one can reduce it to a forgery attack for the underlying SPS

scheme. More specifically, if $k^* \notin \mathcal{Q}_{msg}$ then $\mathcal{B}_{\text{SPS}}$ returns the pair (k^*, σ^*) as a forgery of the defined EUF-CMA security game in Def. B.5. We can write,

$$Adv^{\mathsf{PC}}_{\text{NIRVANA}}(\mathcal{A}) \leq Adv^{\text{KS}}_{\text{NIZK}}(\mathcal{B}_{\text{KS}}) + Adv^{\text{WR}}_{\text{PRF}}(\mathcal{B}_{\text{WR}}) + \Pr[\mathcal{A} \text{ Wins } G_3 : k^* \notin \mathcal{Q}_{msg}] +$$
$$\Pr[\mathcal{A} \text{ Wins } G_3 : k^* \in \mathcal{Q}_{msg}] \leq Adv^{\text{KS}}_{\text{NIZK}}(\mathcal{B}_{\text{KS}}) + Adv^{\text{WR}}_{\text{PRF}}(\mathcal{B}_{\text{WR}}) +$$
$$Adv^{\text{EUF-CMA}}_{\text{SPS}}(\mathcal{B}_{\text{EUF-CMA}}) + \Pr[\mathcal{A} \text{ Wins } G_3 : k^* \in \mathcal{Q}_{msg}] \quad .$$

Since it is assumed that the adversary $\mathcal{A}$ cannot fulfill one of the above requirements, the probability of $\Pr[\mathcal{A} \text{ Wins } G_3 : k^* \in \mathcal{Q}_{msg}]$ is equal to zero. Then we can write,

$$Adv^{\mathsf{PC}}_{\text{NIRVANA}}(\mathcal{A}) \leq Adv^{\text{KS}}_{\text{NIZK}}(\mathcal{B}_{\text{KS}}) + Adv^{\text{WR}}_{\text{PRF}}(\mathcal{B}_{\text{WR}}) + Adv^{\text{EUF-CMA}}_{\text{SPS}}(\mathcal{B}_{\text{EUF-CMA}}) \quad .$$

Since it is assumed $Adv^{\mathsf{PC}}_{\text{NIRVANA}}(\mathcal{A}) \geq \epsilon$, then at least for one of the aforementioned cases we have $Adv^{\text{KS}}_{\text{NIZK}}(\mathcal{B}_{\text{KS}}) \geq \epsilon/3$ or $Adv^{\text{WR}}_{\text{PRF}}(\mathcal{B}_{\text{WR}}) \geq \epsilon/3$ or $Adv^{\text{EUF-CMA}}_{\text{SPS}}(\mathcal{B}_{\text{EUF-CMA}}) \geq \epsilon/3$. This contradicts the defined security properties, and we can conclude the theorem.

□

D Smart contract functionality

The transition between states happens depending on the function calls on the smart contract. For simplicity, we describe here only the functionality of the smart contract focusing on one customer and multiple merchants. We refer to NIRVANA's smart contract as Nir_{SC}, an entity is referred to as e_x where $x = c$ or m for customer or merchant respectively. The underlying ledger is referred to as L_{SC} and an entity's account on that ledger is referred to as Acc^x_L where $x = c$ or m respectively. The private ledger of the merchants is referred to as $Bull_m$, since it behaves like a bulletin board. Nir_{SC} has seven states as follows:

init: Nir_{SC} is deployed. e_c can now deposit funds (col_c). If so, then change state to *ready*. Else do not change state.
ready: e_c successfully registers by depositing col_c in Nir_{SC}. If col_c is available in Nir_{SC}, change state to *pay*. Else do not change state.
pay: If e_c has made payment (pay_{m_i}), change state to $reclaim_m$. Else do not change state.
$reclaim_m$: Check $Bull_m$ for double-spends from e_c. If double-spend present, use secret to reclaim pay_{m_i} and change state to *withdraw*. If no double-spend found until actual payment received, change state to $reclaim_c$.
$reclaim_c$: If 1 day has passed since pay_{m_i}, reclaim pay_{m_i} and add it to col_c. Then, change state to *withdraw*. Else do not change state.
withdraw: If e_x wants to exit NIRVANA, send money from Nir_{SC} to Acc^x_L and change state to *exit*. Else change state to *ready*.
exit: Remove e_x from Nir_{SC} and change state to *init*.