Abstract. Private permissioned blockchains are becoming gradually more sought-after. Such systems are reachable by authorized users, and tend to be completely transparent to whoever interacts with the blockchain. In this paper, we mitigate the latter. Authorized users can now stay unlinked to the transaction they propose in the blockchain while being authenticated before being allowed to interact. As a first contribution, we developed a consensus algorithm for private permissioned blockchains based on Hyperledger Fabric and the Practical Byzantine Fault Tolerance consensus. Building on this blockchain, five additional variations achieving various client-wise privacy preserving levels are proposed. These different protocols allow for different use cases and levels of privacy control and sometimes its revocation by an authority. All our protocols guarantee the unlinkability of transactions to their issuers achieving anonymity or pseudonymity. Miners can also inherit some of the above privacy preserving setting. Naturally, we maintain liveness and safety of the system and its data.

Keywords: Privacy, Byzantine Fault, Blockchain, Signature.

1   Introduction

Blockchains are replicated synchronized databases shared across a trustless network. They started with Bitcoin's Nakamoto consensus style [31], meaning having miners compete to create the next block with a longest-chain-win rule[1]. Anonymity in Bitcoin can be quite contentious [35], which led to more anonymous open blockchains such as Zcash [25] and Monero (based on Cryptonote [40]). Bitcoin was, and to some extent still is, open to all users and all miners. However, blockchains are undergoing an evolutionary shift towards private and permissioned architecture, and implementing BFT consensus instead of Nakamoto consensus[2]. This shift mainly aims to regulate access to sensitive information and to gain in scalability.

Distributed ledger privacy. Danezis and Gürses [16] divided privacy into confidentiality and control (over personal data). With distributed ledgers, control level depends on the openness and configuration of the ledger. Since control includes the right to erasure, and in a distributed setting there is no way to make sure that some data has indeed been deleted, then control is somewhat out of our reach. Confidentiality refers to the difficulty of extracting knowledge from data. With blockchains, the data is an operation, and it must have a specified format and must meet certain requirements. These requirements usually include a signature by its issuer. The two relevant ends of the identity confidentiality spectrum are pseudonymity and anonymity. Pseudonymity is having entities identified by pseudonyms, but not necessarily being able to link the pseudonyms to the identities behind them. A good example of pseudonymity is Bitcoin [31], where users are identified by their public keys, and it is generally infeasible to trace the public key to a real world identity. Note that in bitcoin a user can create as many pseudonyms as they wish. Anonymity, on another hand, is when it is infeasible to link anything. One such example is Monero's usage of ring signatures [36] in order to anonymize the sender's identity that is blended with a set of other identities.

Private (respectively permissioned) distributed ledgers must naturally restrict their usage (respectively their mining) to legitimate entities only. These restrictions seem, at first, incompatible with privacy, giving rise to a dilemma: how to restrict private (respectively permissioned) distributed ledgers usage (respectively mining) while ensuring user (respectively miner) privacy? Note that this apparent dilemma is applicable to virtually any type of private or permissioned distributed ledger, whether it be relying on Proof-of-Work, Proof-of-Stake, Byzantine Fault Tolerance (BFT) protocols or any other type of consensus mechanism.

Contributions. In this article, we propose a new BFT consensus blockchain, SignCons, modelled after Hyperledger Fabric [1,18], with several constructions to gain in privacy. The blockchain itself makes a distinction between two categories of miners: endorsers who verify transactions, and orderers who order valid transactions in blocks. This grants us an *execute-order-validate* architecture. Orderers reach consensus using a

---

[1] Length here does not necessarily mean number of blocks (as it did it at Bitcoin's conception). Length can for example denote difficulty in the current Bitcoin protocol, or weight in Ethereum's GHOST algorithm implementation [38].

[2] Note that some intermediate options exist, such as Proof-of-Stake blockchain, which is in the Nakamoto consensus style as well as being public and permissioned.

modified version of the Practical Fault Tolerance [13], yielding safety and liveness under very moderate assumptions. As for our main contributions, firstly we give three similar yet distinct ways to acquire user anonymity, then we give another two ways to acquire user pseudonymity, and finally we propose two constructions to gain endorser and/or orderer pseudonymity. Constructions aimed at users, endorsers and orderers are composable. All constructions (anonymous and pseudonymous) share a birthplace, however they differ in their applications and use cases. The first of the user-related anonymous constructions is BlindCons. It relies on having an authority, and uses blind signatures. The user authenticates themself to the authority, and if the user is part of the corresponding authorized set, then the authority blind signs their operation. And the data must be deemed valid by the miners if it is signed by the authority. The second construction is GroupCons, and it also relies on an authority but uses group signatures. The authority plays the role of the group manager, with the set of authorized users forming the signing group. The operation is group-signed by its issuer. Group-signed data must be deemed valid by miners. In this scenario, the authority can revoke the anonymity of the issuer. The third construction, RingCons does not rely on an authority, and uses ring signatures. The operation is ring-signed by its issuer, forming a ring with as many of the authorized users as the issuer wants. The ring must not include any user not member of the authorized set. Ring-signed operations in this manner must be deemed valid by miners. As for the two pseudonymous constructions for users, LinkGroupCons and LinkRingCons, they require respectively linkable group signatures and linkable ring signatures. Similarly, endorsers benefit of EndGroupCons and EndRingCons, while orderers have OrdGroupCons and OrdRingCons. Furthermore, the underlying blockchain, thanks to the PBFT structure, guarantees safety and liveness with whatever overlaid privacy construction. The different privacy constructions have different efficiencies, although all can be implemented without much added overhead computation (compared to the underlying blockchain). For a recap of the blockchain's properties refer to Tables 1 and 2. Note that these constructions could very easily be applied on top of any blockchain without inducing much latency into the system, though we do it here for a Hyperledger-like blockchain for which we prove the privacy properties

Related Work. Most known privacy works on blockchains aim at anonymizing cryptocurrencies. Monero (built on Cryptonote [40]) relies on ring signatures [5] and ring confidential transactions [33,39]. Zerocoin [30] introduced zero knowledge proofs (ZKPs) of set membership (which are quite onerous). Zcash [25] (resembling Zerocoin) relies on zk-SNARKS [3] which are much more effficient.

Blockchain privacy has been addressed quite extensively, but rarely has the identity confidentiality of private or permissioned distributed ledgers been addressed. One way to solve the issue is by using identity providers [23]: one permissions issuer and one or more permissions verifiers. The permissions issuer is in charge of verifying a user's identity and issuing them keying material; while the permissions verifier is in charge of verifying that an entity has valid keying material and issuing them a special key allowing them to transact. For the latter verification, the verifier permissions need not know the user's identity, but only verify the keying material.

Another way to ensure privacy on a private network uses puzzle-solving mechanism [15] and is dedicated to IoT networks. They propose to have two blockchains: a public one and a private one. To be allowed to transact on the private chain, users must solve a puzzle on the public chain inside a specific time-lapse. Legitimate users should be able to solve the puzzle within the time-lapse while non-legitimate users should not be able to do it. Other papers consider blockchain privacy from a network level perspective [24]. Some propose privacy-preserving private cryptocurrencies [26]. Even a dedicated survey of privacy-preserving solutions for blockchains [6] barely mentions privacy on private or permissioned blockchains. It mentions a blockchain architecture relying on blind signature that achieves trustless privacy-preserving reputation system [37].

Outline. In Section 2, we start by going over Fabric, PBFT and the cryptographic tools. In Section 3 we showcase our blockchain SignCons, before outlining the many privacy preserving constructions in Section 4. Subsequently we discuss the security and privacy properties in Section 5, the complexities in Section 6 and we conclude in Section 7.

	Revoke User	Authority No	Inactive	Privacy Level
BlindCons	✓			Ano.
GroupCons	✓		✓	Ano.
RingCons		✓	✓	Ano.
LinkGroupCons	✓		✓	Pseu.
LinkRingCons		✓	✓	Pseu.

Table 1: Users' Privacy Preserving Protocols.
Ano. : Anonymous; Pseu. : Pseudonymous.

	Revoke Endorser	Authority Not Needed	Privacy Level
(Ord) EndGroupCons	✓		Pseu.
(Ord) EndRingCons		✓	Pseu.

Table 2: Orderers' and Endorsers' Protocols.

2 Background

SignCons uses an instantiation of Fabric and a PBFT consensus.

- *Practical Byzantine Fault Tolerance (PBFT)* [13] which is a Replicated State Machine protocol. For f faulty (byzantine) nodes, it permits a network of $3f + 1$ nodes to reach consensus.
- *Hyperledger Fabric* [1,18] is a blockchain framework. Is is a highly tweakable permissioned blockchain framework.

2.1 Practical Byzantine Fault Tolerance (PBFT)

Practical Byzantine Fault Tolerance (PBFT) [13] is designed for systems of at least $3f + 1$ nodes where f is the number of faulty (byzantine) nodes. The protocol allows for a very powerful adversary that can coordinate faulty nodes, delay communication, or delay correct nodes. However, it is assumed that the adversary cannot delay correct nodes indefinitely. The adversary is also assumed to be a Probabilistic Polynomial Time (PPT) algorithm. The nodes are called *replicas* and form the set $\mathcal{R}$. Each replica is identified using an integer in $\{0, 1, \ldots, |\mathcal{R}| - 1\}$. One replica is the *primary*, and the rest are *backups*. A *view* is a configuration of replicas that determines the primary. For a view number v, the primary is $p = v \mod |\mathcal{R}|$. The algorithm comprises three stages: *pre-prepare*, *prepare* and *commit*. When a client c wishes to do an operation on the State Machine Replication [27], they send $m = (REQUEST, o, t, c)_{\sigma_c}$ to who the client c believes is the primary. In that message m, o is the operation and t is the timestamp. If the client does not receive replies soon enough, it broadcasts the request to all replicas: this takes care of the client's possible erroneous view and the primary's possible faultiness. Upon receiving a request, the primary p multicast a pre-prepare message to all replicas. That pre-prepare message is $((PRE-PREPARE, v, n, d)_{\sigma_p}, m)$ where v is the current view, n is a sequence number assigned by p and d is m's digest. When a backup receives the pre-prepare message, it checks if the signatures in the request and the pre-prepare message are correct and if d is m's digest; it checks if it is in v; it checks if it has not accepted a pre-prepare message for view v with the same sequence number n and a different digest d; and it checks if the sequence number $h < n < H$ with h, H low and high watermarks. If the pre-prepare message passes all thoses tests, the backup i accepts it and multicast $(PREPARE, v, n, d, i)_{\sigma_i}$. Otherwise, it does nothing. Upon receiving a prepare message, a replica accepts it and appends it to its log if the signatures check out and if d corresponds to its view and if $h < n < H$. When replica i has accepted $2f$ nonconflicting prepare messages, it then multicast a commit message $(COMMIT, v, n, D(m), i)_{\sigma_i}$ where $D(m)$ is the digest of m. A replica accepts and appends a commit message to its log if the signature, the view number check out and if $h < n < H$. As soon as a replica i accepted $f + 1$ nonconflicting commit messages, it executes the operation o on its state machine replication and sends a reply message to the client $(REPLY, v, t, c, i, r)_{\sigma_i}$ where r is the result of executing o.

View Change Each replica has a timer that counts down to zero. It runs when the replica received a valid request and has not executed it, and pauses otherwise. When the countdown reaches zero, the replica stops accepting new messages and initiates a view change. This is done by multicasting a view change message. When the primary of view $v+1$ has received more than $2f$, it multicasts a new view message, and undertakes its role as primary.

Properties Some details have been purposely omitted. We only want to encapsulate the gist of PBFT, since we use a modified version of it for consensus. PBFT relies on the assumption that for $|\mathcal{R}|$ replicas, at most $\left\lfloor \frac{|\mathcal{R}|-1}{3} \right\rfloor$ are faulty and the rather weak syncrony assumption that basically the delay time of communication of honest nodes is less than a given upper bound. With this, it is shown in [12] that PBFT achieves safety and liveness.

2.2 Hyperledger Fabric

Hyperledger Fabric [1,18] is a permissioned blockchain framework. It divides miners into two sets: on the one hand they have *endorsers* who check transactions, and on the other they have *orderers* who order transactions into blocks. When a client wishes to transact on the blockchain, the client sends the transaction to a set of endorsers who are accredited to endorse it. If deemed valid, the endorsers sign it as a sign of endorsement and send it back to the client. When the client collects enough endorsements, the client sends them to the orderers who then work on incorporating said transaction in a block. The structure of Fabric is modulable, in the sense that it is possible to define different endorsement policies for different types of transactions, and it is possible to choose whatever consensus mechanism for the orderers. For example, for a given type of transaction, there must be unanimous endorsement of a given transaction for it to be considered valid; and the orderers compete doing a Proof-of-Work for creating the next block. Fabric does not follow the regular *order-execute*, but rather the *execute-order-validate*. Order-execute blockchains have many limitations, such as sequential execution on all peers, non-deterministic code, and confidentiality of execution. Fabric does provide some privacy, for instance through the use of different channels. Indeed, users only have access to their subset of channels. Another facet of privacy is through the execute-order-validate architecture and the endorsement policies. Only a subset of endorsers have to execute the transaction, and only the state after execution is ultimately written into the blockchain, so those who have access to the channel can see the resulting state but cannot necessarily know what the operation was. However, endorsers have full knowledge of users and their operations.

2.3 Cryptographic Tools

Signatures schemes. Here we present digital signatures which are at the foundation of our constructions. A signature scheme guaranties integrity, authentication and non-repudiation of digital transmissions. A Signature schemes S is a tuple of algorithms composed of a key generation algorithm $\mathsf{KeyGen}(1^{\mathfrak{K}})\colon$] returning a key pair $(\mathsf{pk}, \mathsf{sk})$. The latest is used to sign through algorithm $\mathsf{Sign}(\mathsf{sk}, m)\colon$ producing a signature σ on a message m. This signature being verified by $\mathsf{Verif}(\mathsf{pk}, m, \sigma)\colon$ returning a bit b. It must achieve *EUF-CMA* (Existential Unforgeability under Chosen Message Attack) [34].

Our transformations are based on three extensions of digital signatures: *blind signatures* [34] $\mathsf{BS} = (\mathsf{KeyGen}_{\mathsf{blind}}, \mathsf{BlindSign}\langle\cdot,\cdot\rangle, \mathsf{Verify}_{\mathsf{BS}})$, producing signatures to someone else on messages unknown by the signer. *Group signatures* [14] $\mathsf{GS} = (\mathsf{KeyGen}_{\mathsf{group}}, \mathsf{GroupSign}, \mathsf{Verify}_{\mathsf{group}}, \mathsf{OpenGroupSign})$ where users can output signatures, where the said signature is not linkable to them but to the group. This works under supervision of a group authority. *Ring signatures* [36] $\mathsf{RS} = (\mathsf{KeyGen}_{\mathsf{ring}}, \mathsf{RingSign}, \mathsf{Verify}_{\mathsf{ring}})$ work similarly, but this time on a decentralized base. Refinements of group and ring signatures called *linkable group signatures* and *linkable ring signatures* allow achieving pseudonymity instead of anonymity. The formers rely on an additional Link algorithm to provide a link between the signatures. Here give their security properties:

Blind Signature. EUF-CMA and blindness
Ring Signature. EUF-CMA and anonymity
Group Signature. EUF-CMA, anonymity and traceability.

Algorithm 1 TxProp(o_{bc}, Payload, sk$_{c_{bc}}$)

---

1 (verDep, stateUpdate) $\leftarrow EXEC(o_{bc}, \mathsf{Payload})$
2 trans$_{\mathsf{prop}}$ $\leftarrow$ pk$_{c_{bc}}$||Payload||verDep||stateUpdate
3 $\sigma \leftarrow$ Sign(trans$_{\mathsf{prop}}$, sk$_{c_{bc}}$)
4 **return** (trans$_{\mathsf{prop}}, \sigma$)

Authentication Channel. These processes mainly rely on digital signature schemes. It can also be achieved through other means among which we can cite message authentication codes. Authenticated channel allows authenticated communications with a third party. Public key infrastructure or certificates are among the most used solutions. They do not directly address confidentiality of the transmitted information, but most protocols guaranteeing authentications also address this issue through a key exchange mechanism at the beginning of the communication. This is not a hypothesis we need to make here in order to guarantee the security of our blockchains. In our protocols, we assume that the administrator of the blockchain (when it exists) has knowledge of all the entities' identities that are allowed to transact on or mine the blockchain. And when a key is used to interact through our protocol, it should have been registered before by the authority after potential verifications. The authority's public key pk$_{auth}$ is considered as a global parameter and is not stated into the inputs of the algorithms.

While using blind signatures any peer has to authenticate themselves before the authority blind signs their message. In the upcoming algorithms we put together both the authentication process and the blind signature protocol and call it BlindSign. This aggregation is denoted by BlindSign$\langle \mathcal{U}(M, \mathsf{pk}_{\mathcal{S}}, \mathsf{sk}_{\mathcal{U}}), \mathcal{S}(\mathsf{sk}_{\mathcal{S}}) \rangle$ for two polynomial time algorithms $\mathcal{U}$ and $\mathcal{S}$.

3 Our Blockchain **SignCons**

Our generic SignCons blockchain is highly inspired by Hyperledger Fabric's modular blockchain framework [1,18]. It comprises: (1) a finite set Clients of authorized entities, (2) a finite set Miners of authorized miners divided into two categories: (2a) a set Endor of entities who endorse transactions of cardinal TotEnd and (2b) a set Order of entities who achieve consensus of cardinality TotOrd. The process is outlined in Figure 1 and goes as follows:
1. **Transaction Proposal.** The user signs their operation and sends it to the endorsers.
2. **Transaction Endorsement.** Each endorser peer verifies the transaction, and if valid, signs it (as a sign of endorsement) and sends it back to the user.
3. **Broadcasting to Consensus.** The client collects the endorsements and when enough are received, sends them to the orderers.
4. **Block Proposal.** The orderer leader, upon receiving enough endorsed transactions, creates a block and proposes it to the other orderers.
5. **Block Preparation .** The other orderers, upon receiving the block from the leader, check it, and validate it by signing and broadcasting it to the other orderers.
6. **Appending Block to Blockchain.** Finally, when each orderer receives more than $\frac{2\mathsf{TotOrd}}{3}$ block validations, then each orderer appends the new block to their local version of the blockchain, and broadcasts the new block to users.

Each step is detailed into a corresponding algorithm. Steps 1, 2 and 3 are about gathering enough endorsements, and steps 3, 4, 5 and 6 are about incorporating the transaction in a block and reaching consensus: this is done in a very similar way to how PBFT works. We go thoroughly through each step of our protocol:

Transaction Proposal. User client c_{bc} of public key pk$_{bc} \in$ Clients and secret key sk$_{bc}$ applies Algorithm 1 where o_{bc} denotes the operation with payload Payload. EXEC takes in the operation and its payload, simulates the execution, outputs verDep and stateUpdate that refer respectively to the set of variables invoked by the operation, and to the result of the simulation (these are later used to prevent respectively double spending and non-deterministic execution). At the end of Algorithm 1, c_{bc} finds themself with trans$_{\mathsf{prop}}$ and its signature σ, which they send to the endorsement peers.
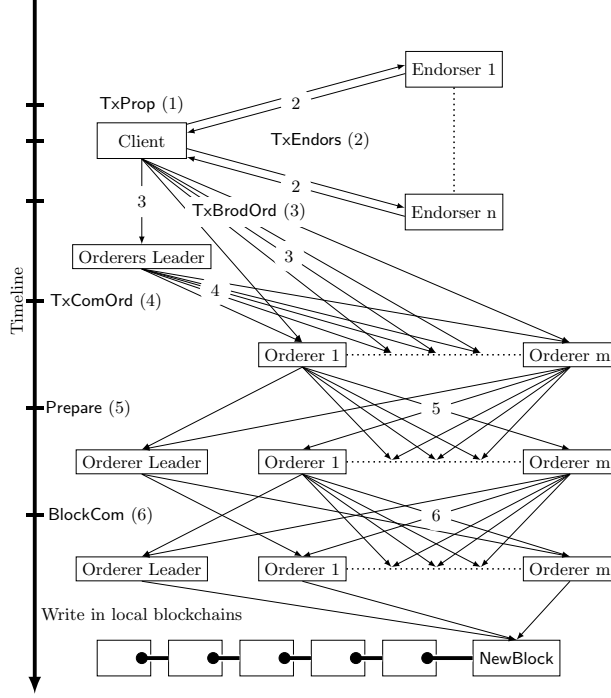
Fig. 1: Transaction Flow of our BFT Consensus.

Algorithm 2 $\mathsf{TxEndors}(\mathsf{trans_{prop}}, \sigma, \mathsf{SecPolicies}, \mathsf{pk}_{c_{bc}}, \mathsf{sk}_{ep})$

---

1 **if** $\mathsf{Verify}_{\mathsf{pk}_{c_{bc}}}(\mathsf{trans_{prop}}, \sigma) = 0$ **or** $\mathsf{pk}_{c_{bc}} \notin \mathsf{Clients}$:
2   **return** $\bot$
3 $(\mathsf{verDep}_{ver}, \mathsf{stateUpdate}_{ver})$
         $\leftarrow EXEC(\mathsf{trans_{prop}}.o_{bc}, \mathsf{trans_{prop}}.\mathsf{Payload})$
4 **if** $\mathsf{trans_{prop}}.\mathsf{verDep} \neq \mathsf{verDep}_{ver}$ **or** $\mathsf{SecPolicies} = invalid$ **or** $\mathsf{trans_{prop}}.\mathsf{stateUpdate} \neq \mathsf{stateUpdate}_{ver}$:
5   **return** $\bot$
6 $\sigma_{ver} \leftarrow \mathsf{Sign}_{\mathsf{sk}_{ep}}(\mathsf{trans_{prop}})$
7 **return** σ_{ver}

Transaction Endorsement. Each endorsement peer ep_{bc} of public key $\mathsf{pk}_{ep} \in \mathsf{Endor}$ and of secret key sk_{ep}, upon receiving the transaction proposal $\mathsf{trans_{prop}}$ and its signature σ, initiates Algorithm 2. The endorsement peer simulates the execution of the operation. If it yields different outputs as the one sent over (by checking verDep and $\mathsf{stateUpdate}$), the algorithm outputs $\bot$. $\mathsf{SecPolicies}$ is an algorithm returning *valid* or *invalid* based on the blockchain endorsement policy and the current state of knowledge of the entity. If all checks out, the algorithm outputs a signature of the operation. Known optimization are possible using aggregation [10] or threshold [17] signatures to fasten the endorsement.

Broadcasting to Consensus. As user client c_{bc} collects a new endorsement of their operation, it applies Algorithm 3 until it has enough endorsements to actually send them to the orderers as a message $\mathsf{blob} = (\mathsf{trans_{prop}}, \{\sigma_{ver,i}\}_{1 \leq i \leq l})$.

Block Proposal. The orderer leader, using Algorithm 4, verifies the endorsements and checks $\mathsf{SecPolicies}$: here this algorithm behaves such that it verifies that the verDep of $blob.\mathsf{trans_{prop}}$ does not collide with the verDep of a transaction already added to the current block (this prevents any potential conflict among transactions). If all the verifications check out, then either the leader creates a block or waits for the next endorsed transaction.

Preparing Block. Upon receiving a block proposal, the orderer verifies the validity of the block proposed by the leader, see Algorithm 5. Sanity checks are conducted: verifying the authenticity of the newly proposed

Algorithm 3 $\mathsf{TxBrodOrd}(\mathsf{trans_{prop}}, \{\sigma_{ver,i}, \mathsf{pk}_i\}_{1 \leq i \leq l})$

1   $endorsements \leftarrow \perp$
2   **for all** $1 \leq i \leq k$ **do**
3   **if** $\mathsf{Verify}_{\mathsf{pk}_i}(\mathsf{trans_{prop}}, \sigma_{ver,i}) = 1$ **and** $\mathsf{pk}_\sigma \in \mathsf{Endor}$:
4      $endorsements \leftarrow endorsements \cup \{\sigma_{ver,i}\}$
5   **if** $|endorsements| \geq \lfloor \frac{\mathsf{TotEnd}}{2} \rfloor + 1$:
6       $blob \leftarrow (\mathsf{trans_{prop}}, endorsements)$
7   **return** blob
8   **return** $\perp$

Algorithm 4 $\mathsf{TxComOrd}(\mathsf{NewBlock}, blob, \mathsf{SecPolicies}, \{\mathsf{pk}_{ep,i}\}_{1 \leq i \leq \mathsf{TotEnd}}, \mathsf{sk}_{ord_{lead}})$

1   $\sigma \leftarrow blob.\mathsf{trans_{prop}}.\sigma$
2   $\mathsf{counter} = 0$
3   **for all** $\sigma \in blob.endorsements$ and $\mathsf{pk}_{ep,i}$ associated and not already used **do**
4   **if** $\mathsf{Verify}_{\mathsf{pk}_\sigma}(\mathsf{trans_{prop}}, \sigma) = 1$ **and** $\mathsf{pk}_{ep,i} \in \mathsf{Endor}$:
5      $\mathsf{counter} = \mathsf{counter} + 1$
6   **if** $\mathsf{counter} < \lfloor \frac{\mathsf{TotEnd}}{2} \rfloor + 1$ **or** $\mathsf{SecPolicies} = invalid$:
7   **return** $\perp$
8   **else if** $\mathsf{NewBlock}.length = \mathsf{Block_{size}}$:
9     $\sigma_{ord_{lead}} \leftarrow \mathsf{Sign}_{\mathsf{sk}_{ord_{lead}}}(\mathsf{NewBlock}||\mathsf{SecPolicies})$
10   **return** $\mathsf{NewBlock}||\sigma_{ord_{lead}}$
11   **else**
12   **return** $\mathsf{NewBlock} \leftarrow \mathsf{NewBlock}||blob$

---

block, checking the validity of the transactions' approvals (*i.e.,* the signatures of the endorsers) and if the absolute majority of endorsement is reached. If none of the blockchain policies have been violated, the block is approved by broadcasting $(\mathsf{NewBlock}, \sigma_{ord})$.

Add Block to Blockchain. When enough valid prepare messages $(\mathsf{NewBlock}, \sigma_{ord})$ have been received by an orderer from its peers, it commits the changes to its local version of the blockchain. It sends a message approval to the group as detailed in Algorithm 6.

The orderers' consensus is highly inspired from PBFT [13], to which we have added endorsers and modified the messages' contents. The orderers mimic the protocol described in [13] for orderer leader's update in order to ensure safety and liveness. Note that what we call leader, they call *primary*. To quickly summarize the view change: if the leader is inactive or misbehaving, then another orderer can initiate a view change; it stops confirming new blocks, and proposes to the set of orderers to change leaders (as per a predetermined schedule). When enough of orderers reply positively, the view change happens.

Algorithm 5 $\mathsf{Prepare}(\mathsf{Block_{size}}, \mathsf{NewBlock}, \sigma_{ord_{lead}}, \mathsf{SecPolicies}, \{\mathsf{pk}_{ep,i}\}_{1 \leq i \leq \mathsf{TotEnd}}, \mathsf{sk}_{ord})$

1   **if** $\mathsf{Verify}_{\mathsf{pk}_{leader}}(\mathsf{NewBlock}||\mathsf{SecPolicies}, \sigma_{ord_{lead}}) \neq 1$:
2   **return** $\perp$
3   Set $\mathsf{counter} = 0$ and parse $\mathsf{NewBlock} = \{blob_i\}_{1 \leq i \leq \mathsf{Block_{size}}}$
4   **for all** $1 \leq i \leq \mathsf{Block_{size}}$ **do**
5   **for all** $\sigma \in blob_i.endorsements$ and $\mathsf{pk}_{ep,i}$ associated and not already used **do**
6   **if** $\mathsf{Verify}_{\mathsf{pk}_\sigma}(blob_i.\mathsf{trans_{prop}}, \sigma) = 1$ **and** $\mathsf{pk}_{ep,i} \in \mathsf{Endor}$:
7       $\mathsf{counter} = \mathsf{counter} + 1$
8   **if** $(\mathsf{SecPolicies} = invalid)$:
9   **return** $\perp$
10   **if** $\mathsf{counter} < \lfloor \frac{\mathsf{TotEnd}}{2} \rfloor + 1$:
11   **return** $\perp$
12   $\sigma_{ord} \leftarrow \mathsf{Sign}_{\mathsf{sk}_{ord}}(\mathsf{NewBlock}||\mathsf{SecPolicies})$
13   **return** $(\mathsf{NewBlock}, \sigma_{ord})$

Algorithm 6 BlockCom(NewBlock, $\{\mathsf{pk}_{ord_i}, \sigma_{ord_i}\}_{1 \leq i \leq l}$, SecPolicies, sk_{ord})

---

1 **if** NewBlock.$length \neq$ Block$_{\mathsf{size}}$ or $l \leq \lfloor \frac{\mathsf{TotOrd}}{3} \rfloor$:
2   **return** $\perp$
3 **for all** $1 \leq i \leq l$ **do**
4   **if** Verify$_{\mathsf{pk}_{ord_i}}$(NewBlock$||$SecPolicies, σ_{ord_i}) $\neq 1$:
5   **return** $\perp$
6 $\sigma_{BlockCom} \leftarrow \mathsf{Sign}_{\mathsf{sk}_{ord}}(NewBlock)$
7 **return** $NewBlock||\sigma_{BlockCom}$

Algorithm 7 TxProp(o_{bc}, Payload, $\mathsf{sk}_{c_{bc}}$)

---

1 (verDep, stateUpdate) $\leftarrow EXEC(o_{bc}, \mathsf{Payload})$
2 crand$_{bc} \xleftarrow{\$} \mathbb{N}$
3 trans$_{\mathsf{prop}} \leftarrow$ crand$_{bc}||o_{bc}||$Payload$||$verDep$||$stateUpdate
4 $\sigma \leftarrow \mathsf{BlindSign}\langle \mathcal{U}(\mathsf{trans}_{\mathsf{prop}}, \mathsf{pk}_{auth}, \mathsf{sk}_{c_{bc}}), \mathbf{A}(\mathsf{sk}_{auth}) \rangle$
5 **return** (trans$_{\mathsf{prop}}, \sigma$)

Network Model We assume an asynchronous distributed system where nodes are connected by a network who may fail to deliver messages, delay them, duplicate them or deliver them out of order. We allow for the adversary to coordinate faulty nodes, delay communication, or delay correct nodes but not indefinitely.

4   Privacy Preserving Blockchains

In Section 3 we introduced a blockchain based on PBFT and requiring a signature scheme. Our construction can be extended in order to allow multiples privacy preserving settings. We can bring anonymity for the issuers and pseudonymity for issuers, endorsers and/or orderers through the use of privacy preserving signatures. In particular, we use blind signatures, group signatures and ring signatures. All these privacy preserving settings can be achieved independently for any of the defined roles. In Section 5, we show that composing any of these settings gives a secure blockchain.

4.1   User's Anonymity

Based on Blind Signatures Considering the permissioned BFT-based consensus protocol introduced in Section 3, users sign their transactions with own key. This causes a strong linkability issue between the users and the transactions, affecting the privacy level of the blockchain network. In order to overcome this issue, we propose to hide the users' identities by using blind signatures with a trusted entity. Thus, the protocol follows the following steps:

1. The client c_{bc} authenticates themselves with their registered keys ($\mathsf{pk}_{c_{bc}}, \mathsf{sk}_{c_{bc}}$) to one of the membership authorities $\mathbf{A}$,
2. Once the authentication succeeds (*i.e.*, $\mathsf{pk}_{c_{bc}} \in$ Clients), the client initiates a blind signature process with $\mathbf{A}$,
3. The client derives a signature σ for its transaction request,
4. We apply the consensus protocol introduced in Section 2.1 on the client's blindly signed transaction. The blind signature authorize the client to transact on the blockchain.

Aiming to keep the same structure as the original construction, we replace the user's ID with a random value crand. Now, to address the issue related to the digital signature, linking the client to a transaction, we replace it with a blind signature scheme. TxProp defined in Algorithm 7 is the modified version of the original TxProp of Algorithm 1. To maintain consistency and liveness, we keep the rest of the transactional flow unchanged. However, some steps are modified to accept the blind signature scheme to authenticate the clients and the peers. This variant of SignCons is called BlindCons.

Based on Group Signatures In the previously presented transaction mechanism, every transaction must first go through the authority to be blind signed before anything else can be done with it. Using a group signature there is a way to obtain a decentralized transaction proposal mechanism. Let $\mathsf{GS} = (\mathsf{KeyGen_{group}}, \mathsf{GroupSign}, \mathsf{Verify_{group}}, \mathsf{OpenGroupSign})$ be a group signature scheme with its usual security requirements (outlined in Section 2.3), this next variant of our scheme enables any registered user to sign hiding amongst the group of authorized users. We assume that authorized users are registered with the authority and that a public record of all of them is available. Hence, all keys are generated through a protocol with the authority and registered in the Clients record.

We call GroupCons the variant of SignCons adopting group signatures. This version is instantiated by replacing the signature in Algorithm 1 (as well as the signature verifications) with a group signature where the group consists of all the people having rights to write in the blockchain. By using group signatures instead of blind signatures, we can make the protocol less relying on the authority while simultaneously giving the authority power to reveal a message's signer if need be. This revealing can be done using the OpenGroupSign algorithm. Compared to the blind signature construction, group signatures also limit the computational load on the authority as it no longer needs to execute its part of the blind signature protocol for each new transaction.

Based on Ring Signatures This version is instantiated by replacing the signatures in Algorithm 1 (as well as the signature's verifications in Algorithm 2) with a ring signature where the ring consists of (potentially all the) authorized users. We call this variant RingCons. We thus come up with a private blockchain architecture with no authority where transactions are unlinkable to their issuers. Let $U = u_1, \ldots, u_n$ be the set of authorized users. Suppose u_1 wants to make a transaction. In the regular SignCons protocol, u_1 would sign its transaction using a regular signature protocol (Algorithm 1), and then send it for endorsement (Algorithm 2). To anonymize the user's identity we change the type of signature: u_1 signs the transaction using a ring signature in the name of U (or in the name of a subset of U if U is too big). That way the endorsers verifies the ring signature in Algorithm 2, and can thus know that it was indeed someone of S that produced the signature without being able to know which member it was (since it is a ring signature). The rest of the protocol remains unchanged.

4.2   User's Pseudonymity

Let GS_{link} be a *linkable group signature* and let RS_{link} be a *linkable ring signature*. We keep on relying on the same idea and operate these two types of signatures. In this modified version of our blockchain, the signature in the anonymized version of TxProp (Algorithm 1) is replaced by one of these linkable signatures, again the verification in Algorithm 2 is modified adequately. We call the version using linkable group signature LinkRingCons and the other version using linkable ring signature LinkGroupCons. These two primitives retain the properties of being unlinkable to the signature issuer, however they allow for linking the transaction to other transactions signed by the same secret key using the Link algorithm (see Section 2.3). As such, signers remain unknown, but one can track all the transactions created by the same entity. We emphasize in Section 5 why this achieves pseudonymity, it is straight forward to see that it does not achieve anonymity due to the linkability of the signatures. LinkGroupCons being based on a group signature requires an authority to be implemented, and gives it the power to revoke the privacy of the signer. On the other hand, LinkRingCons requires no authority and naturally is not revocable.

4.3   Endorser and Orderer Pseudonymity

In the current scheme, the endorsers and orderers are respectively linked to the transactions they endorse and to the blocks they commit. Enabling endorsers and orderers to use group or ring signature would not be fruitful, since it enables each entity to produce an unlimited number of different signatures without being detected, which is problematic since we need to count the number of approvals. Using linkable group signatures and linkable ring signatures does help us keep the endorsers' and the orderers' privacy, while at the same time restricting just enough of the excessive anonymity that is brought by group and ring signatures.

Thus, taking the same perspective on this as in Section 4.2, we can use linkable group signatures and linkable ring signatures in Algorithm 2 for the endorsers and call it respectively EndGroupCons and EndRingCons. The same modification is possible in Algorithms 5 and 6 for orderers, we call these protocols OrdGroupCons and OrdRingCons. Note that an extra step needs to be added after verifying the signature (with $\mathsf{Verify_{group}}$ and $\mathsf{Verify_{ring}}$): the verifier must also check if this signature can be linked to another signature of the same content (transaction or block) before taking it into account. For endorsers' pseudonymity, this extra verification is done in Algorithms 3 (executed by the orderer leader) and 4 (executed by the orderers). As for orderers, this is done in Algorithm 6. Note also that the orderer leader cannot use a pseudonym with these constructions, since the other orderer peers must be able to check that the leader's status and that this allows block proposal.

We claim, and later prove (in Section 5), that it is impossible to link the endorsers (respectively orderers) to their transaction endorsement (respectively block generation), while concurrently not allowing them to produce multiple acceptable signatures for the same transaction endorsement (respectively block generation).

All the presented layers of constructions maintain a Byzantine Fault Tolerant blockchain. They all allow some privacy for entities of various roles. In fact, it is possible to combine any construction for users (from Sections 4.1 and 4.2) with any of the constructions for endorsers and orderers from the current Section.

5   Protocol Properties

Under two hypotheses, our constructions satisfy *Safety*, *Liveness*, *Unforgeability* of a block and some privacy preserving settings, namely, *Anonymity* or *Pseudonymity*. The latter can be obtained for any of the three existing roles: issuers, endorsers and orderers and in any possible settings. For example, it is proven that composing anonymity of the clients with pseudonymity for the endorsers is still secure. Safety and Liveness are both inherited from PBFT, leading to the first hypothesis being that the adversary cannot delay correct nodes indefinitely. We model and prove the other security properties using game based formalism and reductions. In general, we consider a security experiment where a PPT challenger $\mathcal{C}$ interacts with a PPT adversary $\mathcal{A}$. The adversary simulates the behaviour of a malicious entity, while the challenger runs the rest of the system honestly. Based on these, we show that the full security of our protocols also depends on the secure primitives used for instantiating it. Any secure signature could instantiate ours blockchains.

Safety A protocol is said to be *consistent* if it ensures that a transaction generated by a valid user stays immutable in the blockchain. Our blockchains are based on PBFT which leads a consensus for each of the deciding steps of our blockchain.

Definition 1 (Safety). *A protocol $\mathcal{BC}$ is T-safe if a transaction tx generated by an honest client c_{cb} to execute a valid operation o_{bc}, is confirmed and stays immutable in the blockchain after T-round of new blocks.*

Theorem 1. *Our new protocols based on PBFT are 1-safe if at most $\lfloor \frac{n-1}{3} \rfloor$ out of total n orderer peers are malicious.*

Proof (Proof sketch). The described protocols are BFT based consensus. Safety is achieved by agreeing with the validity of the transaction through a byzantine agreement process. Hence, for a transaction tx that has reached a majority of valid endorsement for an operation o_{bc}, the probability of not settling it in a new block and having forks in the chain is neglected if we have at most $\lfloor \frac{n-1}{3} \rfloor$ malicious orderers, out of total n orderers as it has been shown in [11,13]. It is 1-consistent because we do not have any fork; hence only one block is needed to wait to have a transaction validated.

Liveness The liveness property means that a consensus protocol ensures that if an honest client submits a valid transaction, a new block is later appended to the chain with the transaction in it. Hence, the protocol must ensure that the blockchain grows if valid clients generate valid transactions.

Definition 2 (Liveness). *A consensus protocol $\mathcal{BC}$ ensures liveness for a blockchain C if $\mathcal{BC}$ ensures that after a period of time t, the new version of the blockchain C' verifies $C' > C$, if a valid client $c_{i_{bc}}$ has broadcasted a valid transaction tx_i during the time t.*

Theorem 2. *Our protocols satisfy liveness when at most* $\lfloor \frac{\mathsf{TotEnd}-1}{2} \rfloor$ *out of a total of* TotEnd *endorsers and* $\lfloor \frac{\mathsf{TotOrd}-1}{3} \rfloor$ *out of total* TotOrd *orderers are malicious.*

Proof (Proof sketch). Our protocols are BFT-based consensus protocol. Thus, liveness is achieved if after the transaction endorsement process, the ordering services propose a new block $\mathsf{NewBlock}$ with the transactions broadcast by the clients during a period of time t. Hence, for valid transactions tx_i (*i.e.,* accepted by the endorsers), where $i \in \mathbb{N}$, issued by valid a client during a period of time t, the probability that $C' = C$ is neglected if we have at most $\lfloor \frac{n-1}{3} \rfloor$ out of total n malicious orderers. A detailed proof for the type of consensus we are using can be found in [12].

Unforgeability An adversary against the *unforgeability* of a protocol tries to overstep the validation process of a transaction in order to engrave a transaction in the blockchain without obtaining the full transaction acceptance from the endorsers and the orderers. Our blockchain composes a validation procedure conducted by the Issuer with the endorser and a consensus agreement made by the orderers. This property ensures that these compositions retains security *i.e.,* no adversary could possibly overstep the validation procedure nor the consensus to engrave a block in the blockchain. The security model is defined in Appendix B: unforgeability is defined in Definition 6 through Experiment 2 and security is stated and proven in Appendix C in Theorem 4

Theorem 3 (informal). *Consider a blockchain* $\mathcal{BC}$ *defined by one of the above settings,* i.e.,*instantiated with some of the above described signatures. For any security parameters* $\mathfrak{K}$, $\mathcal{BC}$ *is unforgeable.*

The proof of this property, relies on the unforgeability of the signatures used in $\mathcal{BC}$. While the versions using linkable group (*resp.* ring) signature for the endorsers also depend on the traceability (*resp.* linkability) of the signatures. Otherwise, it would be possible for a single node to output multiple signatures. Hence, it is infeasible to append an invalid transaction to the existing blockchain without being given the endorsers and orderers agreement.

Pseudonymity A entity $\mathcal{E}$ and a witness w are said to be *linked* in a group G's perspective, if it is possible for G to infer that $\mathcal{E}$ produced w based on the available information to G. *Pseudonymity* of an entity holds when $\mathcal{E}$ cannot be linked to the witnesses $w_1, \ldots, w_k$ it has produced, but this property does not prevent from linking the witnesses one to each other. When linking w_i and w_j is hard for all $1 \leq i < j \leq k$, it is considered as a stronger privacy preserving property called *anonymity*. We split the actors of our blockchain into two groups, on one side the users and on the other side the endorsers and orderers to consider their pseudonymity.

    Users. Let $\mathcal{A}$ be an attacker against an user *pseudonymity*. Its goal is to link it to a transaction tx it has produced. Nevertheless, we assume that it is only possible to link tx to the client using information from the blockchain. As a consequence, we assume that it would be hard to identify the provenance of a transaction due to the redundancy of the sent communications as upon receiving a message each entity broadcasts it to all its peers (gossip). This is a classical assumption in blockchain. In order fulfil pseudonymity, o_{bc} and $\mathsf{Payload}$ should not leak information on the transaction requester. A public transaction always reveals a certain amount of information as this information is publicly enclosed, here we show that no additional information is revealed throughout the protocol.

Theorem 6 (informal). *Consider secure* BS *a blind signature,* GS *a (linkable) group signature and* RS *a (linkable) ring signature. Assume that an adversary* $\mathcal{A}$ *is unable to identify a user at the origin of a transaction based on* o_{bc} *and* $\mathsf{Payload}$. *Then the consensus presented in Section 4.2, instantiated with these signatures is pseudonymous.*

    Users sign their transaction with a signature mechanism and sends it to the nodes. Once this is executed, they are no longer involved in the process. Hence, what they output should be unlinkable to them. This result relies directly on the anonymity properties of the signatures considered in the article.

    Endorsers and Orderers. Endorsers and orderers can be unlinked from transactions and blocks they validated through the use of *linkable ring or group signatures*. The linking algorithm Link allows this feature, hence, enabling detection of nodes producing multiple validations. The signature anonymity requirement prevents from recovering the identity of the executant.

Theorem 7. *Base on a secure linkable group (resp. ring) signature the* **EndGroupCons,** *(resp.***EndRingCons***) protocol is pseudonymous for the endorsers. Under the same conditions, the protocols* **OrdGroupCons** *and* **OrdRingCons** *are pseudonymous for the orderers (excluding the orderer leader).*

Arguments supporting these properties are the same the ones evoked before for the users. Once again, these properties are defined in Appendix B: pseudonymity is defined in Definitions 7 and 8. The associated security is proven in Appendix C in Theorem 5 and 7

Anonymity of the users As stated before, anonymity is defined by two requirements: (i) there should be no link between the clients and the transactions they produced, (ii) transaction of the same user should not be linkable. We yet know that BlindCons, RingCons, GroupCons securely realize user's pseudonymity, statement (i). In fact, it is also possible to show that (ii) holds, as it appears that the output transaction are in these cases unlinkable one to each other.

Theorem 8. *Given that the client proceeds to a secure blind, ring or group signature to authenticate its transaction as defined in Section 4, anonymity of the client holds.*

This theorem relies on the unkinkability of the signatures produced by a user. A proof of this theorem is provided in Appendix C.4.

6 Complexity

Our approach is generic, hence allows to instantiate the protocol with the most efficient signature schemes in the literature. In order to provide a theoretical evaluation that can benefits from further works on these primitives, we evaluate the number of executions of each algorithm for the various entities in the blockchain. Let S_C be the signature used by the clients, S_E the signature used by the endorsers and S_O used by the orderers. These signatures can refer to any of the signature schemes used in our blockchains as specified through Section 3 and 4 due to modularity of our proposed constructions. In order to obtain a validated transaction, a client needs to execute S_C once and verify $\lfloor \mathsf{TotEnd}/2 \rfloor + 1$ signatures S_E. In the meantime, an endorser needs to verify a signature from S_C once and produce one signature S_E. Now, in order to validate a block, the orderer leader needs to verify $\mathsf{Block_{size}}$ signatures S_C, $\mathsf{Block_{size}} \cdot (\lfloor \mathsf{TotEnd}/2 \rfloor + 1)$ signatures S_E and later $2(\lfloor 2\mathsf{TotOrd}/3 \rfloor + 1)$ signatures S_O. It also needs to produce two regular signatures. The orderer verifies these signatures and $\mathsf{Block_{size}} \cdot (\lfloor \mathsf{TotEnd}/2 \rfloor + 1)$ signatures from the endorsers and $2(\lfloor 2\mathsf{TotOrd}/3 \rfloor + 1)$ signatures from other orderers. An orderer also produces two signatures S_O.

State-of-the-art blind, group or ring signature are known to be less efficient than regular signature schemes. All still achieve constant execution time [20,29,9]. Also their longer computation time must be put into perspective with the time needed for the numerous communications require by a blockchain. As the order of magnitude of a signature execution does not generally exceed the order of magnitude of a RTT (Round-Trip Time), the overhead brought by bringing anonymity to our blockchain seems acceptable. Blind signature usually requires 2 (at best) or 3 additional communication yielding as much aditional communications. Again this does not increase much the number of communications of the blockchain, hence has low impact on the performance. On another hand group and ring signatures requires to obtain the keys of the members of a ring. In the case of group signatures, we can assume that they are all provided by the registration authority on demand. This considered, for an equivalent level of security, our protocol is expected to be less efficient than a blockchain without any anonymity, as a counterpart it brings more security for its peers as their identity is not publicly disclosed.

7 Conclusion

In this paper we bring forth a blockchain solution to the apparent dilemma of combining private permissioned blockchains with privacy. We divide the miners into endorsers and orderers. And we propose different constructions for different entities privacy (user, endorser or orderer). The different constructions use different

building blocks and have different use cases. Some yield anonymity and others pseudonymity, some rely on an authority, some propose privacy revocation rights, some require less computation than others. We showed using game-based proofs the security of our constructions. As future works, we envision extending our results in the UC model, by considering the ideal functionality presented in [22].

References

1. Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, et al. Hyperledger fabric: A distributed operating system for permissioned blockchains. *arXiv preprint arXiv:1801.10228*, 2018.
2. Man Ho Au, Sherman S. M. Chow, Willy Susilo, and Patrick P. Tsang. Short linkable ring signatures revisited. In Andrea S. Atzeni and Antonio Lioy, editors, *Public Key Infrastructure*. Springer Berlin Heidelberg, 2006.
3. Aritra Banerjee, Michael Clear, and Hitesh Tewari. Demystifying the role of zk-snarks in zcash. In *2020 IEEE conference on application, information and network security (AINS)*. IEEE, 2020.
4. Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In *International conference on the theory and applications of cryptographic techniques*. Springer, 2003.
5. Adam Bender, Jonathan Katz, and Ruggero Morselli. Ring signatures: Stronger definitions, and constructions without random oracles. *Journal of Cryptology*, 2009.
6. Jorge Bernal Bernabe, Jose Luis Canovas, Jose L Hernandez-Ramos, Rafael Torres Moreno, and Antonio Skarmeta. Privacy-preserving solutions for blockchain: Review and challenges. *IEEE Access*, 2019.
7. David Bernhard, Georg Fuchsbauer, Essam Ghadafi, Nigel P Smart, and Bogdan Warinschi. Anonymous attestation with user-controlled linkability. *International Journal of Information Security*, 2013.
8. Olivier Blazy., Brouilhet Laura., Céline Chevalier., and Neals Fournaise. Round-optimal constant-size blind signatures. In *Proceedings of the 17th International Joint Conference on e-Business and Telecommunications SECRYPT,*. SciTePress, 2020.
9. Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matt Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.
10. Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *International conference on the theory and applications of cryptographic techniques*. Springer, 2003.
11. Gabriel Bracha and Sam Toueg. Asynchronous consensus and broadcast protocols. *Journal of the ACM (JACM)*, 1985.
12. Miguel Castro, Barbara Liskov, et al. A correctness proof for a practical byzantine-fault-tolerant replication algorithm. Technical report, Technical Memo MIT/LCS/TM-590, MIT Laboratory for Computer Science, 1999.
13. Miguel Castro, Barbara Liskov, et al. Practical byzantine fault tolerance. In *OSDI*, 1999.
14. David Chaum and Eugène van Heyst. Group signatures. In *Workshop on the Theory and Application of Cryptographic Techniques*. Springer, 1991.
15. Jollen Chen. Hybrid blockchain and pseudonymous authentication for secure and trusted iot networks. *ACM SIGBED Review*, 2018.
16. George Danezis and Seda Gürses. A critical review of 10 years of privacy technology. *Proceedings of surveillance cultures: a global surveillance society*, 2010.
17. Yvo Desmedt and Yair Frankel. Threshold cryptosystems. In *Conference on the Theory and Application of Cryptology*. Springer, 1989.
18. Linux Foundation. Hyperledger. `https://www.hyperledger.org/`, 2019.
19. Ashley Fraser, Lydia Garms, and Anja Lehmann. Selectively linkable group signatures - stronger security and preserved verifiability. Cryptology ePrint Archive, Paper 2021/1312, 2021.
20. Georg Fuchsbauer, Christian Hanser, and Daniel Slamanig. Practical round-optimal blind signatures in the standard model. In *CRYPTO*. Springer, 2015.
21. Georg Fuchsbauer, Antoine Plouviez, and Yannick Seurin. Blind schnorr signatures and signed elgamal encryption in the algebraic group model. In *Advances in Cryptology — EUROCRYPT 2020*. Springer, 2020.
22. Mike Graf, Daniel Rausch, Viktoria Ronge, Christoph Egger, Ralf Küsters, and Dominique Schröder. A security framework for distributed ledgers. In *ACM SIGSAC*, 2021.
23. Thomas Hardjono, Ned Smith, and Alex Sandy Pentland. Anonymous identities for permissioned blockchains, 2014.

24. Ryan Henry, Amir Herzberg, and Aniket Kate. Blockchain access privacy: Challenges and directions. *IEEE Security & Privacy*, 2018.
25. Daira Hopwood, Sean Bowe, Taylor Hornby, and Nathan Wilcox. Zcash protocol specification. *GitHub: San Francisco, CA, USA*, 2016.
26. Aram Jivanyan. Lelantus: Towards confidentiality and anonymity of blockchain transactions from standard assumptions. *IACR Cryptol. ePrint Arch.*, 2019.
27. Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. In *Concurrency: the Works of Leslie Lamport.* 2019.
28. Joseph K Liu and Duncan S Wong. Linkable ring signatures: Security models and new schemes. In *International Conference on Computational Science and Its Applications.* Springer, 2005.
29. Giulio Malavolta and Dominique Schröder. Efficient ring signatures in the standard model. In *International Conference on the Theory and Application of Cryptology and Information Security.* Springer, 2017.
30. Ian Miers, Christina Garman, Matthew Green, and Aviel D Rubin. Zerocoin: Anonymous distributed e-cash from bitcoin. In *2013 IEEE Symposium on Security and Privacy.* IEEE, 2013.
31. Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *self-published paper*, 2008.
32. Toru Nakanishi, Toru Fujiwara, and Hajime Watanabe. A linkable group signature and its application to secret voting. *Trans. of Information Processing Society of Japan*, 1999.
33. Shen Noether, Adam Mackenzie, et al. Ring confidential transactions. *Ledger*, 2016.
34. David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *J. Cryptol.*, 2000.
35. Fergal Reid and Martin Harrigan. An analysis of anonymity in the bitcoin system. In *Security and privacy in social networks.* Springer, 2013.
36. Ronald L Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. In *International conference on the theory and application of cryptology and information security.* Springer, 2001.
37. Alexander Schaub, Rémi Bazin, Omar Hasan, and Lionel Brunie. A trustless privacy-preserving reputation system. In *IFIP International Conference on ICT Systems Security and Privacy Protection.* Springer, 2016.
38. Yonatan Sompolinsky and Aviv Zohar. Secure high-rate transaction processing in bitcoin. In *International Conference on Financial Cryptography and Data Security.* Springer, 2015.
39. Shi-Feng Sun, Man Ho Au, Joseph K Liu, and Tsz Hon Yuen. Ringct 2.0: A compact accumulator-based (linkable ring signature) protocol for blockchain cryptocurrency monero. In *European Symposium on Research in Computer Security.* Springer, 2017.
40. Nicolas Van Saberhagen. Cryptonote v 2.0. 2013.

A Digital Signatures

In this section we formally define the signatures used in the article. First their description is given, then the associated security requirements.

Blind Signatures are schemes allowing one to obtain someone else's signature on an undisclosed message. It finds utility when one wants to be unlikable to some signed data they produce. For example, during an enrollment process, the authority in charge could affix its signature onto the user's message without the latter ever revealing it. Hence, implying no direct linkability between the user and the signed message it obtains. For example, digital cash frequently use blind signature schemes to protect the anonymity of senders and receivers while still making sure the transaction is valid. Let $m \in \mathcal{M}$ be a message in an arbitrary message space. Take $\mathcal{U}$ a user holding this message and let $\mathcal{S}$ be the signing entity.

Definition 3 (Blind Signature [34]). *A* blind signature *scheme* BS *consists of the following algorithms/protocols:*
KeyGen($1^{\mathfrak{K}}$) *returns a key pair* (pk, sk) *for a security parameter* $\mathfrak{K}$.
BlindSign$\langle \mathcal{U}(m, \mathsf{pk}), \mathcal{S}(\mathsf{sk}) \rangle$ *is a protocol run between a user* $\mathcal{U}$ *inputting a message* m *and a public key* pk, *while a signer* $\mathcal{S}$ *uses their private key* sk. *Outputs are either the signature* σ *for the user, or* $\perp$ *if the protocol ended unsuccessfully.*
Verify$_{\mathsf{BS}_{\mathsf{pk}}}(m, \sigma)$ *is a deterministic algorithm taking a public key* pk, *a message* m *and a signature* σ *as input, and returning 1 if* σ *is a valid signature on* m *under* pk *and 0 otherwise.*
A blind signature should achieve the properties of Correctness, *EUF-CMA and* Blindness *described below.*

Three security properties need to be achieved by a blind signature.
- *Correctness.* A blind signature scheme has *correctness* if for a message space $\mathcal{M}$, it verifies:

$$\forall m \in \mathcal{M}, \forall (\mathsf{sk}, \mathsf{pk}) \in [\mathsf{KeyGen}()],$$
$$\forall \sigma \in [\mathsf{BlindSign}\langle \mathcal{U}(\mathsf{pk}, m), \mathcal{A}(\mathsf{sk})\rangle], \mathsf{Verify}_{\mathsf{BS}_{\mathsf{pk}}}(m, \sigma) = 1.$$

- *EUF-CMA.* An adversary $\mathcal{U}^*$, attacking *unforgeability*, tries to generate $q_s + 1$ valid signatures after at most q_s complete interactions with the honest signer $\mathcal{S}$. A blind signature scheme BS is *unforgeable* if for an honest signer $\mathcal{S}$ and for any polynomial time adversary $\mathcal{U}^*$, the probability $\mathsf{Adv}^{uf}_{\mathsf{BS}, \mathcal{U}^*}(\mathfrak{K}) = \Pr[\mathsf{Exp}^{uf}_{\mathsf{BS}, \mathcal{U}^*}(\mathfrak{K}) = 1]$ is negligible, where $\mathsf{Exp}^{uf}_{\mathsf{BS}, \mathcal{U}^*}(\mathfrak{K})$ is defined in Experiment 1a, adversary $\mathcal{U}^*$ has access to q_s queries to a blind signature oracle $\mathcal{O}^{blind}_{\mathsf{sk}}(m) \to (\sigma, T)$ returning the signature (if produced) and in any case the transcript of the protocol.
- *Blindness.* The adversary is a malicious signer $\mathcal{S}^*$ with polynomial capability and two honest user instances are simulated by a challenger $\mathcal{C}$ within Experiment 1b. A blind signature scheme BS achieves *blindness* if for some security parameter $\mathfrak{K} \in \mathbb{N}$, for all PPT adversary $\mathcal{S}^*$, there is a negligible function $\epsilon(\cdot)$ such that $\mathsf{Adv}^{bl}_{\mathsf{BS}, \mathcal{S}^*}(\mathfrak{K}) = |1/2 - \Pr[\mathsf{Exp}^{bl}_{\mathsf{BS}, \mathcal{S}^*}(\mathfrak{K}) = 1]| < \epsilon(\mathfrak{K})$.

$\mathsf{Exp}^{uf}_{\mathsf{BS}, \mathcal{A}}(\mathfrak{K})$:
1. params$\leftarrow$ Setup($1^{\mathfrak{K}}$)
2. $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{params})$
4. $(m_i, \sigma_i)_{1 \le i \le q_s+1} \leftarrow \mathcal{A}^{\mathcal{O}^{blind}_{sk}(m)}(\mathsf{pk})$
5. If $\exists i \ne j \in [1, q_s+1], m_i = m_j$ or
$\quad \exists i \in [1, q_s+1], \mathsf{Verify}_{\mathsf{BS}_{\mathsf{pk}}}(m_i, \sigma_i) = 0$:
$\quad\quad$ Return 0
6. Else Return 1

(a) Unforgeability of a Blind Signature.

$\mathsf{Exp}^{bl}_{\mathcal{BS}, \mathcal{A}}(\mathfrak{K})$:
1. params$\leftarrow$ Setup($1^{\mathfrak{K}}$)
2. $(\mathsf{pk}, m_0, m_1) \leftarrow \mathcal{A}(\mathsf{params})$
3. $b \xleftarrow{\$} \{0, 1\}$
4. $\sigma_b \leftarrow \mathsf{BlindSign}\langle \mathcal{U}(\mathsf{pk}, m_b), \mathcal{A}\rangle$
5. $\sigma_{1-b} \leftarrow \mathsf{BlindSign}\langle \mathcal{U}(\mathsf{pk}, m_{1-b}), \mathcal{A}\rangle$
6. $b^* \leftarrow \mathcal{A}((m_0, \sigma_0), (m_1, \sigma_1))$
7. Return $b^* = b$

(b) Blindness of a Blind Signature.

Experiment 1: Blind Signatures' Security Experiements

Many recent constructions for blind signature schemes satisfy these properties, such as [8,21,34] among others.

Group Signatures It allows members of a group to sign messages anonymously. Group signatures include a group manager, which is an entity that can: (i) add members to the group, and (ii) reveal the original signer (in the event of disputes for example). Later this role is played by the authority of the private blockchain.

Definition 4 ((Linkable) Group Signature [14]). *A* group signature *scheme consists of four algorithms organised as follows:*

$\mathsf{KeyGen}_{\mathsf{group}}(1^{\mathfrak{K}}, N)$ *takes as input a security parameter $\mathfrak{K}$ and the group size N, it returns a vector (*gpk, gmsk, $\mathsf{gsk}_1, \ldots, \mathsf{gsk}_N$*) composed of a group public key* gpk*, a group manager secret key* gmsk*, and group secret keys* gsk_i *for the members.*

$\mathsf{GroupSign}_{\mathsf{gsk}}(m, \mathsf{gpk})$ *outputs a group signature σ, on the message m.*

$\mathsf{Verify}_{\mathsf{group}_{\mathsf{gpk}}}(m, \sigma, \mathsf{gpk})$ *outputs a bit b representing the validity of the signature.*

$\mathsf{OpenGroupSign}_{\mathsf{gmsk}}(m, \sigma)$ *returns the identity of the member who produced σ.*

A Linkable Group Signature *has one additional algorithm:*

$\mathsf{Link}(L_1, \sigma_1, \sigma_2, m_1, m_2)$ *checks if two pairs (m_1, σ_1), (m_2, σ_2) come from the same signing key, in this case it returns 1, otherwise 0.*

A group signature has Correctness, Unforgeability*,* Anonymity *and* Traceability*. In the case of linkable group signatures an additional requirement of* Linkability*.*

An informal description of the security requirements for this type of signature is given below. These definitions are formalised for properties used in our security proofs. Please refer to [4,14,32] for a full formalisation of these requirements.

- *Correctness.* Execution of the algorithm must lead to a valid signature.
- *Unforgeability.* This ensures that no entity outside the group should be able to produce valid group signatures.
- *Anonymity.* A given message-signature pair does not leak any information about the member of the group that made it.

  An adversary $\mathcal{A}$ plays the experiment $\mathsf{Exp}_{\mathsf{S},\mathcal{B}}^{f-anon}(\mathfrak{K})$ for a security parameter $\mathfrak{K}$ in two stages: choose and guess. During the first phase, $\mathcal{A}$ receives a group member's secret key gsk for a chosen index along with the group members' public key $\{\mathsf{gpk}\}_{i=1,\ldots,N}$. During this phase it can query the opening oracle $\mathsf{OpenGroupSign}(\mathsf{gmsk},\cdot)$ on group signatures. $\mathcal{A}$ is required to output two identities $1 \leq \mathsf{id}_0, \mathsf{id}_1 \leq N$ and a message m. The adversary is allowed to keep some state information for the second stage. This ends the first phase. Starting the choose phase, $\mathcal{A}$ receives its status information and a signature on m produced with the secret key of id_0 or id_1 chosen at random. The adversary can still query the opening oracle on other signatures and is meant to return a bit b^*. The experiment outputs 1 if this value is the index of the identity chosen to sign the message. Then, $\mathsf{Adv}_{\mathsf{S},\mathcal{B}}^{f-anon}(\mathfrak{K}) = |1/2 - \mathsf{Pr}[\mathsf{Exp}_{\mathsf{GS},\mathcal{A}}^{f-anon}(\mathfrak{K}) = 1]|$ is negligible.
- *Traceability.* The group manager should always be able to identify the user behind a signature provided knowledge of the message-group signature pair. This property means that there exists an extractor algorithm Extract that on inputs a group of user, the group manager secret key and a signature returns the identity of the group member which produced it.
- *Linkability (only for linkable group signatures).* Informally, it captures the ability of the adversary to outputs two signature of the same entity returning 0 for the Link algorithm.

An extended line of research avoids the trust in an entity for opening and supports pseudonymous group signatures where each member of the group can choose to sign either with a fresh and unlinkable pseudonym or re-use one that was established. Making all signatures under the same pseudonym linkable for all [7,19].

Ring Signatures , mostly similar to group signatures, avoid the need to rely on a trusted entity. Any user $\mathcal{U}$ can sign a message in the name of any set of users to without setup. In a ring the anonymity cannot be revoked, still it can be crucial in some settings to be protected against multiple signatures of the same message coming from the same entity. Just like in the previous scheme, *linkable ring signatures* have been introduced to solve this issue. Anyone can identify signatures coming from the same entity but still is not able to recover its identity.

Definition 5 ((Linkable) Ring Signature [28]). *A* ring signature *scheme consists of three algorithms organised as follows:*

$\mathsf{KeyGen}_{\mathsf{ring}}(1^{\mathfrak{K}})$ *generating a pair (*pk, sk*) for an entity.*

$\mathsf{RingSign}_{\mathsf{sk}_0}(m, \mathsf{pk}_1, \ldots, \mathsf{pk}_N)$ *outputs a ring signature σ for the set of the entities L.*

$\mathsf{Verify}_{\mathsf{ring}\{\mathsf{pk}_i\}_{i \in L}}(m, \sigma)$ *returns 1 for a valid σ, 0 for an invalid.*

  A linkable ring signature *has an additional algorithm:*

$\mathsf{Link}(L_1, L_2, \sigma_1, \sigma_2, m_1, m_2)$ *checks if pairs (m_1, σ_1), (m_2, σ_2) were made from the same signing key. If it is the case it returns 1, otherwise 0.*

Security of ring signatures is based on Correctness, Unforgeability, Anonymity. *Additionally, linkable ring signatures have* linkability *and sometimes its stronger version* Linkability w.r.t. adversarially-chosen keys.

A ring signature has several properties to reach in order to be consider secure. These properties are described in []. Please refer to [2,5] for a formalisation of the security requirements. Here, we give some insights about these properties and formalise the needed ones for our latter proofs.

- *Correctness.* It checks that the verification algorithm is consistent with its inputs. A valid message-signature pair should be accepted if it has been generated rightfully, otherwise the verification algorithm should reject the signature.
- *Unforgeability.* It states that no polynomial time adversary should be able to output a valid signature after only polynomial many calls to a ring signature oracle outputting valid signature for the messages provided by the adversary.

– *Anonymity.* An adversary should not be able to identify the actual user outputting a signature within a group of n entities with probability greater than $1/n$. If the adversary is part of the group and controls t entities, the probability increase to $1/(n-t)$. The formalisation of this property is similar to the one in the context of group signatures. Still the adversary does not have access to an open algorithm as none does exist for ring signatures.

– *Linkability (only for linkable ring signatures).* Captures the impossibility of an adversary controlling a group member trying to (i) generate two signatures that cannot be linked using the Link algorithm. This property also captures abuses of an adversary trying to (ii) generate a signature such that algorithm Link links the outputted Signature to another entity. The second layer of security is sometimes called *slanderability.*

Linkability is described through two experiments. In any case it starts by generating keys for all authorized members and providing the adversary $\mathcal{A}$ with a set of public keys, the cardinal of the set of authorized entities and the security parameter. (i) $\mathcal{A}$ is also provided with a signature oracle $\mathcal{SO}(\cdot, \cdot, R)$, where R is a subring of the existing entities and a corruption oracle $\mathcal{CO}$. A record of the corrupted users is kept in C. It must answer by sending a corrupted user index and $((L_i, n_i, m_i, \sigma_i)_{1 \le i \le 2})$ such that L_i is a subring composed by n_i entities, both message signature pair are valid, were not query to the oracle and cannot be linked: $\mathsf{Link}(L_1, L_2, \sigma_1, \sigma_2, m_1, m_2) = 0$. Moreover, there must only be one entity in the set $|(L_1 \cup L_2) \cap C| \le 1$. In such a case the experiment return 1, otherwise 0.

The second experiment starts just the same way, then (ii) $\mathcal{A}$ again being provided with a signature oracle has to return (L, n, π_1, m_1) such that $\pi \in L$, $|L| = n$ and L a subring of the entities. The challenger produces a signature σ_1 on m_1 using π_1's key. The answers of the adversary should be a player index $\pi_2 \in L \setminus \{pi_1\}$ and after receiving its secret key a valid message signature pair (m_2, σ_2) linking to the first one.

B Security Model

In conducting their attack, the adversary has access to a registration oracle $\mathcal{O}^{register}(\mathsf{pk})$.

B.1 Unforgeability

Definition 6 (Unforgeability). *Let $\mathcal{BC}$ be an instantiation of our blockchain protocols and $k \in \mathbb{N}$ be a security parameter. The adversary $\mathcal{A}$ controls a subset of the participants of the blockchain: a coalition of n entities among which, one is a user of the $\mathcal{BC}$ scheme, n_{ep} is the number of corrupt endorsers and n_{or} is the number of corrupt orderers. We consider* unforgeability *to hold if for any PPT adversary algorithm $\mathcal{A}$ the advantage $\mathsf{Adv}^{unf}_{\mathcal{BC},\mathcal{A}}(\mathfrak{K}, \mathsf{SecPolicies}, \mathsf{TotEnd}, \mathsf{TotOrd}) = \Pr[\mathsf{Exp}^{unf}_{\mathcal{BC},\mathcal{A}}(\cdot) = 1]$ is negligible for any given parameters. $\mathcal{A}$ can register new users through a registration oracle $\mathcal{O}^{register}(\mathsf{pk})$ in order to be able to create new users.*

B.2 Pseudonymity

Pseudonymity of a user

Definition 7 (Clients' Pseudonymity). *Consider the experiment Exp^{ps} defined in Experiment 3. A consensus $\mathcal{BC}$ is* pseudonymous for the clients *if for any* $\mathsf{params} = (\mathfrak{K}, \{o_{bc,i}, \mathsf{Payload}_i\}_{i \in \{0,1\}})$, *a PPT adversary $\mathcal{A}$ playing the pseudonymous experiment has a negligible advantage $\mathsf{Adv}^{ps}_{\mathcal{BC},\mathcal{A}}(\mathsf{params}, \mathfrak{K}) = |\mathsf{Exp}^{ps}_{\mathcal{BC},\mathcal{A}}(\mathsf{params}) - 1/2|$.*

Pseudonymity of the Endorsers and the Orderers

Definition 8 (Endorsers' and Orderers' Pseudonymity). *A protocol $\mathcal{BC}$ is said to be* pseudonymous for the endorsers *(resp. orderers) if for all parameters* $\mathsf{params} = (\mathfrak{K}, o_{bc}, \mathsf{Payload}, \mathsf{SecPolicies})$, *for all PPT adversary $\mathcal{A}$, the following advantage is negligible $\mathsf{Adv}^{ps}_{\mathcal{BC},\mathcal{A}}(\mathsf{params}, \mathfrak{K}) = |\mathsf{Exp}^{(\cdot)}_{\mathcal{BC},\mathcal{A}}(\mathsf{params}) - 1/2|$, for $\mathsf{Exp}^{ps_{Endor}}$ based on Experiment 4 (resp. $\mathsf{Exp}^{ps_{Ord}}$ from Experiment 5).*

$\mathsf{Exp}_{\mathcal{BC},\mathcal{A}}^{unf}(\mathfrak{K}, \mathsf{SecPolicies}, \mathsf{TotEnd}, \mathsf{TotOrd})$:
1. $n_{ep}, n_{ord} \leftarrow \mathcal{A}(\mathfrak{K}, \mathsf{SecPolicies}, \mathsf{TotEnd}, \mathsf{TotOrd})$
2. **if** $\mathsf{TotEnd} \leq 2n_{ep} - 1$ or $\mathsf{TotOrd} \leq 3n_{ord} - 1$: Return 0
3. $(\mathsf{pk}, \mathsf{sk}), (\mathsf{pk}_{ep}, \mathsf{sk}_{ep}), \{\mathsf{pk}_{ord,i}, \mathsf{sk}_{ord,i}\}_{n_{ord} \leq i \leq \mathsf{TotOrd}}$
   $\leftarrow \mathsf{KeyGen}(\mathfrak{K})$
4. $\mathsf{blob}, \{\mathsf{pk}_{ep,i}\}_{1 \leq i \leq n_{ep}}, \{\mathsf{pk}_{ord,i}\}_{1 \leq i \leq n_{ord}}$
   $\leftarrow \mathcal{A}(\mathsf{pk}, \mathsf{pk}_{ep}\{\mathsf{pk}_{ord,i}\}_{n_{ord} \leq i \leq \mathsf{TotOrd}})$
5. **For all** $1 \leq i \neq j \leq n_{ep}$: **if** $\mathsf{pk}_{ep,i} = \mathsf{pk}_{ep,j}$: , Return 0
6. **if** $\mathsf{TxEndors}(\mathsf{blob.trans_{prop}}, \mathsf{SecPolicies}, \mathsf{pk}, \mathsf{sk}_{ep}) \neq \perp$:
Return 0
7. $(\mathsf{NewBlock}||\sigma_{ord_{lead}}) \leftarrow \mathcal{A}()$,
8. **if** $\mathsf{blob} \notin \mathsf{NewBlock}$: Return 0
9. **for** $(n_{ord} < i \leq \mathsf{TotOrd})$:
10. $\mathsf{message}_{\mathsf{Prepare},i} \leftarrow \mathsf{Prepare}(\mathsf{Block}_{\mathsf{size}}, \mathsf{NewBlock},$
    $\sigma_{ord_{lead}}, \mathsf{SecPolicies}, \{\mathsf{pk}_{ord,i}\}_{1 \leq i \leq \mathsf{TotEnd}}, \mathsf{sk}_{ord,i})$
11. $\{\mathsf{message}_{\mathsf{Prepare},i}\}_{1 \leq i \leq n_{ord}}$
    $\leftarrow \mathcal{A}(\{\mathsf{message}_{\mathsf{Prepare},i}\}_{n_{ord} < i \leq \mathsf{TotOrd}})$
12. **if** $|\{\mathsf{message}_{\mathsf{Prepare},i} : 1 \leq i \leq \mathsf{TotOrd}, \mathsf{message}_{\mathsf{Prepare},i} \neq \perp\}|$
         $< \lfloor 2 \cdot \mathsf{TotEnd}/3 \rfloor + 1$: Return 0
13. **while** $(|\mathsf{NewBlock}| < \mathsf{Block}_{\mathsf{size}})$:
14.   *Fill the block with valid random transactions*
15. **if** $\mathsf{BlockCom}(\mathsf{NewBlock}, \mathsf{SecPolicies}, \mathsf{sk}_{ord}) \neq \perp$:
16.   Return 1
17. Return 0

Experiment 2: Unforgeability Experiment

$\mathsf{Exp}_{\mathcal{BC},\mathcal{A}}^{ps}(\mathfrak{K}, \{o_{bc,i}, \mathsf{Payload}_i\}_{i \in \{0,1\}})$:
1. $(\mathsf{pk}_{c_0}, \mathsf{sk}_{c_0}), (\mathsf{pk}_{c_1}, \mathsf{sk}_{c_1}) \leftarrow \mathsf{KeyGen}(\mathfrak{K})$
2. $\mathsf{TotEnd}, \mathsf{TotOrd} \leftarrow \mathcal{A}(\mathfrak{K})$
3. $\{\mathsf{pk}_{ep,i}\}_{1 \leq i \leq \mathsf{TotEnd}}, \{\mathsf{pk}_{ord,i}\}_{1 \leq i \leq \mathsf{TotOrd}} \leftarrow \mathcal{A}(\mathsf{pk}_{c_0}, \mathsf{pk}_{c_1})$
4. $b \xleftarrow{\$} \{0, 1\}$
5. $(\mathsf{trans}_{\mathsf{prop,b}}, \sigma_b) \leftarrow \mathsf{TxProp}(o_{bc,b}, \mathsf{Payload}_b, \mathsf{sk}_{c_b})$
6. $\{\mathsf{trans}_{\mathsf{prop,b}}||\sigma_{ep,i}\}_{1 \leq i \leq l} \leftarrow \mathcal{A}(\mathsf{trans}_{\mathsf{prop,b}}, \sigma_b)$
7. **if** $l < \lfloor 2 \cdot \mathsf{TotEnd}/3 \rfloor + 1$: $b^* \leftarrow \mathcal{A}()$
8.   Return $b = b^*$
9. $\mathsf{blob} \leftarrow \mathsf{TxBrodOrd}(\mathsf{trans}_{\mathsf{prop,b}}, \{\sigma_{ep}, \mathsf{pk}_{ep,i}\}_{1 \leq i \leq l})$
10. $b^* \leftarrow \mathcal{A}(\mathsf{blob})$
11. Return $b = b^*$

Experiment 3: Experiment for the Pseudonymity of the Client

$\mathsf{Exp}_{\mathcal{BC},\mathcal{A}}^{psEndor}(\mathfrak{K}, o_{bc}, \mathsf{Payload}, \mathsf{SecPolicies})$:
1. $(\mathsf{pk}_0, \mathsf{sk}_0), (\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{KeyGen}(\mathfrak{K})$
2. $(\mathsf{trans}_{\mathsf{prop}}, \sigma), \mathsf{pk}_{c_{bc}} \leftarrow \mathcal{A}(\mathfrak{K}, \mathsf{pk}_0, \mathsf{pk}_1, \mathsf{SecPolicies})$
3. $b \xleftarrow{\$} \{0, 1\}$
4. $\mathsf{message}_{\mathsf{verif}} \leftarrow \mathsf{TxEndors}(\mathsf{trans}_{\mathsf{prop}}, \sigma,$
              $\mathsf{SecPolicies}, \mathsf{pk}_{c_{bc}}, \mathsf{sk}_b)$
5. **if** $\mathsf{message}_{\mathsf{verif}} = \perp$: Return 0
6. $b^* \leftarrow \mathcal{A}(\mathsf{message}_{\mathsf{verif}})$
7. Return $b = b^*$

Experiment 4: Experiment for the Pseudonymity of the Endorsers

```
Exp_{BC,A}^{psOrd}(ℜ, o_{bc}, Payload, SecPolicies):
1. TotEnd, TotOrd ← A(ℜ)
2. (pk_0, sk_0), (pk_1, sk_1) ← KeyGen(ℜ)
3. pk_{client}, {pk_{ep,i}, sk_{ep,i}}_{1≤i≤TotEnd}, {pk_{or,i}, sk_{or,i}}_{2≤i≤TotOrd}
        ← A(pk_0, pk_1)
4. NewBlock, σ_{ord_{lead}} ← A(o_{bc}, Payload, SecPolicies)
5. b ←$ {0, 1}
6. message_{Prep} ← Prepare(Block_{size}, NewBlock,
        σ_{ord_{lead}}, SecPolicies, {pk_{ep,i}}_{1≤i≤TotEnd}, sk_b)
7. if message_{Prep} =⊥:  Return b
8. {NewBlock, σ_{or,i}}_{1≤i≤l} ← A(message_{Prep})
9. if l < ⌊2 · TotEnd/3⌋ + 1:  Return b
10. Block ← BlockCom(NewBlock, {pk_{ord_{bc},i}, σ_{ord_{bc},i}}_{1≤i≤l},
                        SecPolicies, sk_b)
11. if Block =⊥:  Return b
12. b* ← A(Block)
13. Return b = b*
```

Experiment 5: Experiment for the Pseudonymity of an Orderer.
n: index of the orderer leader.

```
Exp_{BC,A}^{Anony}(ℜ, {o_{bc,i}, Payload_i}_{i∈{0,1}}, o_{bc}', Payload'):
1. SecPolicies, TotEnd, TotOrd ← A(ℜ)
2. if TotEnd, TotOrd ≤ 0:
3.      Return b* ←$ {0, 1}
4. (pk_0, sk_0), (pk_1, sk_1) ← KeyGen(ℜ)
5. {pk_{ep,i}}_{1≤i≤TotEnd}, {pk_{ord,i}}_{1≤i≤TotOrd} ← A(pk_0, pk_1)
6. (trans_{prop,0}, σ_0) ← TxProp(o_{bc,0}, Payload_0, sk_0)
7. (trans_{prop,1}, σ_0) ← TxProp(o_{bc,1}, Payload_1, sk_1)
8. b ←$ {0, 1}
9. (trans_{prop}', σ') ← TxProp(o_{bc}', Payload', sk_b)
10. b* ← A((trans_{prop,0}, σ_0), (trans_{prop,1}, σ_1), (trans_{prop}', σ'))
11. Return b = b*
```

Experiment 6: Anonymity of the transactions in our Protocols.

B.3 Anonymity of the users

Definition 9 (Anonymity). *A consensus protocol is* Anonymous *if for any* $\mathsf{params} = (\mathfrak{K}, \{o_{bc,i}, \mathsf{Payload}_i\}_{i\in\{0,1\}}, o_{bc}', \mathsf{Payload}')$, *all PPT algorithm* $\mathcal{A}$ *have a negligible advantage* $\mathsf{Adv}_{\mathcal{BC},\mathcal{A}}^{anony}(\mathsf{params}, \mathfrak{K}) = |Exp_{\mathcal{BC},\mathcal{A}}^{anony}(\mathsf{params}) - 1/2|$. $Exp_{\mathcal{BC},\mathcal{A}}^{anony}$ *is defined in Experiment 6.*

C Security Proofs

C.1 Security Proof of Unforgeability

We recall Theorem 4 and prove it through this section.

Theorem 4. *Let* $\mathcal{BC}$ *be one of the above defined protocols. For any* $(\mathfrak{K}, \mathsf{SecPolicies}, (\mathsf{sk}, \mathsf{pk}), \mathsf{TotEnd}, \mathsf{TotOrd})$ *inputted in the unforgeability experiment, given a version of our blockchain* $\mathcal{BC}$ *instantiated with an EUF-CMA signature (signature, blind/(linkable) group/(linkable) ring signature) then* $\mathcal{BC}$ *is unforgeable. The versions using linkable group signature (resp. linkable ring signature) for the endorsers respectively need traceability (resp. linkability) to guarantee the security of the schemes.*

Proof. We prove the unforgeability theorem by using a disjunction of cases.

Unforgeablility of SignCons

Let $\mathcal{BC}$ be the SignCons blockchain and S a secure signature scheme. The proof of unforgeability is divided into two main steps. At first, we introduce a failure event failing if one of the endorser's signature is not seen in the transaction validation. A second step consists of a reduction on the unforgeability of the signature scheme. After this step it is impossible for an attacker to bypass the verifications of the endorsers and to engrave an invalid transaction in the blockchain without being authorized by a substantial amount of corrupt endorsers or being able to forge valid signatures.

Game 0. Let Game 0 be the original experiment $\mathsf{Exp}_{\mathcal{BC},\mathcal{A}}^{unf}$.

Game 1. A pool of TotEnd endorsers is set before the experiment, among them $n_{ep} < \lfloor \mathsf{TotEnd}/2 \rfloor + 1$ are corrupt by $\mathcal{A}$. We randomly pick one endorser at the beginning of the game, if this one get corrupted by $\mathcal{A}$ abort the game. The picked honest endorser is called $\mathcal{E}_{\mathcal{BC}}$. Game 1 is based on Game 0, with the described added condition. We obtain $\mathsf{Adv}_{\mathcal{BC},\mathcal{A}}^{\mathsf{G}_0} \leq (\mathsf{TotEnd} - n_{ep})/\mathsf{TotEnd}\mathsf{Adv}_{\mathcal{BC},\mathcal{A}}^{\mathsf{G}_1}$.

Game 2. If the signature of the picked endorser does not appear in the transaction approval message blob, the experiment fails and return 0. This additional condition defines Game 2. As the adversary needs at least $\lfloor \mathsf{TotEnd}/2 \rfloor + 1$ approval for the transaction to be accepted. And that $\lfloor \mathsf{TotEnd}/2 \rfloor + 1 > n_{ep}$ we are assured that at least some honest endorser has their signature engraved in the transaction acceptance vector. Hence, the picked endorser has at least probability $(\lfloor \mathsf{TotEnd}/2 \rfloor + 1 - n_{ep})/\mathsf{TotEnd}$ to be involved in the transaction. Then,

$$\mathsf{Adv}_{\mathcal{BC},\mathcal{A}}^{\mathsf{G}_1} \leq (\lfloor \mathsf{TotEnd}/2 \rfloor + 1 - n_{ep})/\mathsf{TotEnd} \cdot \mathsf{Adv}_{\mathcal{BC},\mathcal{A}}^{\mathsf{G}_2}.$$

From the previous failure cases, we are ensuring that a signature associated to $\mathcal{E}_{\mathcal{BC}}$'s public key has validated the transaction and was later engraved in the current block. We are going to show that in this case we have an adversary $\mathcal{A}$ breaking the current game, then we can construct from it an adversary $\mathcal{D}$ breaking the unforgeability of the signature scheme.

Let $\mathcal{Out}_{\mathcal{E}_{\mathcal{BC},i+1}}$ be the set of all message-signature pairs (m, σ) associated to $\mathsf{pk}_{\mathcal{E}_{\mathcal{BC}}}$. If $\mathcal{A}$ was able to engrave a valid signature for $\mathcal{E}_{\mathcal{BC}}$ in the new block, it has produced a forgery of $\mathcal{E}_{\mathcal{BC}}$'s signature, as its entity does not appear in the experiment. As we have assumed the unforgeability of this scheme we can directly conclude that the SignCons scheme is unforgeable for any number of corrupted user n_{ed} and n_{cs}. We are now providing a reduction in order to show that this property is rightfully given. Let us consider an adversary $\mathcal{A}$ capable of outputting a forgery for the $\mathcal{BC}$ scheme. Using a successful $\mathcal{A}$, we are going to construct a polynomial time algorithm $\mathcal{D}$ able to break the unforgeability property of the signature scheme S with non-negligible probability 1.

$\mathcal{D}$ starts by receiving the public key for the EUF-CMA's game for the signature scheme and is given access to a signature oracle for this key. Then $\mathcal{D}$ simulates the SignCons protocol playing the roles usually played by the challenger. First $\mathcal{D}$ generates the other necessary keys and has to register a key given by $\mathcal{A}$. The challenger's key pk is used as $\mathcal{E}_{\mathcal{BC}}$'s public key. The simulation starts according to the experiment of Experiment 2 by sending the elements $\mathfrak{K}$, SecPolicies, pk these elements are taken according to the signature scheme security parameter. Being given the previous element, $\mathcal{A}$ returns TotEnd and TotOrd, the lower bound on these values is checked by $\mathcal{D}$. After this step we are receiving the keys from $\mathcal{A}$. These keys are registered by the authority simulated by $\mathcal{D}$. Then $\mathcal{A}$ has to produce a consensus array blob that is checked with the TxComOrd algorithm by the challenger. The first clause of verification of TxComOrd is to verify all the signatures that are provided in the array blob. If it does not contain enough signatures, the experiment stops and $\mathcal{A}$ was not successful. Otherwise, there exist at least $\lfloor \mathsf{TotEnd}/2 \rfloor + 1$ signatures with the associated public keys of one new user each time.

As we assume that $\mathcal{A}$ has non-negligible chances of success, there exist some cases where the verification and the preparation of the block have been successfully executed. If the procedure went through, $\mathcal{D}$ extracts the entire set of signatures and their associated messages from the block $NewBlock$. Assuming that $\mathcal{A}$ succeeded in its forgery of a block, there must exist at least one signature for one of the honest users. We know from the failure event of the beginning that $\mathcal{E}_{\mathcal{BC}}$ is involved in the block. Hence, a message-signature pair has been returned for its public key $\mathsf{pk}_{\mathcal{E}_{\mathcal{BC}}}$. As no signature has ever been queried to $\mathcal{C}$ we can directly return the obtained signature to $\mathcal{C}$ resulting in a valid forgery as the signature has been verified in the Prepare algorithm.

The following upper bound is obtained on the adversary's advantage: $\mathsf{Adv}^{unf}_{\mathcal{BC},\mathcal{A}} \leq (\mathsf{TotEnd} - n_{ep}) \cdot (\lfloor \mathsf{TotEnd}/2 \rfloor + 1 - n_{ep})/\mathsf{TotEnd}^2 \cdot \mathsf{Adv}^{EUF-CMA}_{S,\mathcal{D}}$.

Unforgeablility of BlindCons, GroupCons, RingCons, LinkGroupCons and LinkRingCons

The way the client commits to a transaction does not influence the steps of the previous reduction. Hence, the proof also apply if the transaction is proposed with a blind, group or ring signature ensuring its authenticity, this cases are already covered. Hence, our constructions BlindCons, GroupCons, RingCons, LinkGroupCons and LinkRingCons are unforgeable.

Unforgeablility of EndGroupCons

Let $\mathcal{LS}_{group}$ be the SignCons protocol with the additional layer of pseudonymity for the endorsers and for the orderers as they are signing their validation messages with a linkable group signature GS. Additional steps need to be carried out in order to prove the security while opting for linkable signatures for the endorsers and/or the orderers validations. This time our proof relies on the unforgeability and on the linkability of the linkable group signature scheme.

Game 0 is the original experiment $\mathsf{Exp}^{unf}_{\mathcal{LS}_{group},\mathcal{A}}$.

Game 1. At some point in $\mathsf{Exp}^{unf}_{\mathcal{LS}_{group},\mathcal{A}}$, $\mathcal{A}$ has to output a new block proposal NewBlock with the transaction blob embedded in it. blob must contain $l \geq \lfloor \mathsf{TotEnd}/2 \rfloor + 1 > n_{ep}$ linkable signatures'. Game 1 is a modification of Game 0 where for each of these signatures, the challenger uses knowledge of the group manager to extract the identity of each of the signer from the signature through the extraction algorithm Extract. If any of the identities appear more than once, the game is aborted. This leads to the following advantage:

$$\mathsf{Adv}^{\mathsf{G}_0}_{\mathcal{LS}_{group},\mathcal{A}} \leq (\lfloor \mathsf{TotEnd}/2 \rfloor + 1) \cdot \mathsf{Adv}^{trace}_{\mathsf{GS},\mathcal{A}} + \mathsf{Adv}^{\mathsf{G}_1}_{\mathcal{LS}_{group},\mathcal{A}}.$$

These steps remove all possible double approbation by a corrupted endorser. Hence, $\mathcal{A}$ has outputted l linkable group signatures in blob while it is only allowed to corrupt $n_{ep} < l$ endorsers (*i.e.,* group members). Assume that $\mathcal{A}$ wins with non-negligible probability. We can directly build a simulator $\mathcal{D}$ by forwarding all of $\mathcal{A}$'s requests during the setup of the $\mathcal{LS}_{group}$ key generation to the challenger of the EUF-CMA security property of the linkable group signature. This concludes our proof as it is in contradiction with the assumed security of the linkable group signature. We obtain the upper bound:

$$\mathsf{Adv}^{unf}_{\mathcal{LS}_{group},\mathcal{A}} \leq (\lfloor \mathsf{TotEnd}/2 \rfloor + 1) \cdot \mathsf{Adv}^{trace}_{\mathsf{GS},\mathcal{A}} + \mathsf{Adv}^{EUF-CMA}_{\mathsf{GS},\mathcal{A}}.$$

Unforgeablility of EndRingCons

Let $\mathcal{LS}_{ring}$ be the SignCons protocol with the additional layer of pseudonymity for the endorsers and for the orderers as they are signing their validation messages with a linkable ring signature RS. A stronger requirement is needed here: RS must be linkable w.r.t. adversarially-chosen keys as we let $\mathcal{A}$ produce his own keys for the endorsers and the orderers.

Game 0 is the original experiment $\mathsf{Exp}_{\mathcal{LS}_{ring},\mathcal{A}}^{unf}$.

Game 1 is a modified version of Game 0, the challenger is now extracting the identities behind each of the signature provided by $\mathcal{A}$ in blob using the extraction algorithm Extract. If it appears that two signatures come from the same entity, the game fails. As we are already checking the redundancy of the signature using Link in the Prepare algorithm. If the game fails it would mean that $\mathcal{A}$ was able to output two signatures from the same entity such that Link returns 0, this would lead to an attack again the linkability w.r.t. adversarially-chosen keys of RS. But this was one of our hypothesis, thus: $\mathsf{Adv}_{\mathcal{LS}_{ring},\mathcal{A}}^{\mathsf{G_0}} \leq 1/2 \cdot ((\lfloor \mathsf{TotEnd} \rfloor + 1)\mathsf{TotEnd})$ $\mathsf{Adv}_{\mathsf{RS},\mathcal{A}}^{link-adv} + \mathsf{Adv}_{\mathcal{LS}_{ring},\mathcal{A}}^{\mathsf{G_1}}$.

Now, we are assured that $\mathcal{A}$ has outputted l linkable ring signatures in blob while it is only allowed to corrupt $n_{ep} < l$ endorsers. Assume that $\mathcal{A}$ wins with non-negligible probability. We can directly build a simulator $\mathcal{D}$ by forwarding all of $\mathcal{A}$'s requests during the setup of the $\mathcal{LS}_{ring}$ key generation to the challenger of the EUF-CMA security property of the linkable ring signature. This concludes our proof as it is in contradiction with the assumed security of the linkable group signature. We obtain the upper bound: $\mathsf{Adv}_{\mathcal{LS}_{ring},\mathcal{A}}^{unf} \leq 1/2 \cdot ((\lfloor \mathsf{TotEnd} \rfloor + 1)\mathsf{TotEnd})\mathsf{Adv}_{\mathsf{RS},\mathcal{A}}^{link-adv} + \mathsf{Adv}_{\mathsf{RS},\mathcal{A}}^{EUF-CMA}$.

The modification for orderers' pseudonymity falls within the proof of unforgeability for SignCons as they do not interact in the transaction validation. Hence, unforgeability of OrdGroupCons and OrdRingCons holds from the previous arguments. We now know that all the 9 protocols are unforgeable on their own and that OrdGroupCons and OrdRingCons are unforgeable being composed with any of the user's pseudonymity protocols. As the two last arguments do not consider which signature is used for the transaction proposal TxProp, the proofs also covers the EndGroupCons and the EndRingCons being composed with the 4 variations of user's pseudonymity. We have shown that all possible compositions achieve unforgeability.

C.2 Security Proof of the Pseudonymity of a User

We recall Theorem 5 and prove it in this section.

Theorem 5. *Let* BS *be a blind signature,* GS *be a (linkable) group signature with anonymity and* RS *a (linkable) ring signature with anonymity. Assume that an adversary $\mathcal{A}$ is unable to identify a user at the origin of a transaction based on the elements o_{bc} and* Payload. *Then the consensus presented in Section 4.2, instantiated with these signatures is pseudonymous.*

Proof. **Pseudonymity of BlindCons**

Game 0. Let Game 0 be the original Experiment 3.

Game 1. We start by a bridging step. We define Game 1 by asking $\mathcal{A}$ to send back a unique binary value b^* on its third round of answers and then the experiment returns the value of $b = b^*$ directly after that. The challenger's execution of TxBrodOrd involves no additional data that were not already known by $\mathcal{A}$, is a simple rearrangement of the data that $\mathcal{C}$ received. Hence, a bridging step involving no differences in the advantages.

Game 2. We want to show that no adversary is able to distinguish whether the messages $o_{bc,0}$ and $\mathsf{Payload}_0$ was used by the experiment to create the transcript T_1 or T_2. For that, we define Game 2 that is the same as Game 1, except that the elements $o_{bc,0}$ and $\mathsf{Payload}_0$ are permuted with the elements $o_{bc,1}$ and $\mathsf{Payload}_1$. During the experiment, $\mathcal{A}$ has two views of the protocols TxProp. We call these transcripts T_1 and T_2, representing respectively the first and the second execution. If BS is a blind signature, then no PPT

adversary can distinguish these two executions as it has blindness by hypothesis, we obtain $\mathsf{Adv}^{\mathsf{G_0}}_{\mathcal{BC},\mathcal{A}}(\mathfrak{K}) = \mathsf{Adv}^{\mathsf{G_2}}_{\mathcal{BS},\mathcal{S}^*}(\mathfrak{K}) + \mathsf{Adv}^{bl}_{\mathcal{BS},\mathcal{S}^*}$. We can see that Game 2 is the same as Game 1 except that the bit b is flipped in these two games.

Game 3. This done we can invert the modification introduced in Game 1 and instantiate it as Game 3. Then Game 3 is the same as Game 0 but with the bit b flipped in these two games. This concludes our proof.

Pseudonymity of GroupCons

We are considering the GroupCons version of our protocol instantiated with a group signature GS. Pseudonymity has been defined in Definition 9. We recall that in the current context the action $\sigma \leftarrow \mathsf{Sign}(\mathsf{trans}_{\mathsf{prop}}, \mathsf{sk}_{c_{bc}})$ of the TxProp algorithm has been replaced by $\sigma \leftarrow \mathsf{GroupSign}_{\mathsf{gsk}_{c_{bc}}}(M, \mathsf{gpk})$.

Game 0. Let Game 0 be the Experiment 3.

Game 1. This new game is the same bridging step as for the first part. We ask the adversary to send back a bit b^* after receiving $\mathsf{trans}_{\mathsf{propb}}$ and the challenger returns the value of $b = b^*$ directly after that. The same argument as in previous part of the proof can be applied, the forthcoming actions had no influence on the adversary's knowledge. They do not contain more information than already known by the adversary hence this change does not modify the probability of the game.

This step describes the reduction of the pseudonymity of our protocol to the anonymity of the group signature. Take $\mathcal{A}$ a successful adversary against the pseudonymity of our protocol, we show how to construct $\mathcal{D}$, an adversary against the group signature that can break the anonymity of the group signature.

$\mathcal{D}$ starts the anonymity's experiment of the group signature scheme and gain access to its oracles after generation of the group manager public key gpk with the group members public keys. Among them $\mathcal{D}$ draws two keys randomly and set this keys for pk_0 and pk_1. It sends gpk, pk_0, pk_1 and the security parameter to $\mathcal{A}$ who's answers is the set of public keys for the endorsers and the orderers. $\mathcal{D}$ produce the elements $\{o_{bc,b}, \mathsf{Payload}_b\}_{b \in \{0,1\}}$. It executes the TxProp algorithm for $(o_{bc,b}, \mathsf{Payload}_b)$ after drawing $b \xleftarrow{\$} \{0,1\}$ randomly. The result of the algorithm is $(\mathsf{trans}_{\mathsf{prop}}, \sigma)$ and sends it to the adversary. $\mathcal{A}$'s answers is a vector $\{\mathsf{trans}_{\mathsf{prop,b}} || \sigma_{ep,i}\}_{1 \leq i \leq l}$, $\mathcal{D}$ checks if $l < \lfloor 2 \cdot \mathsf{TotEnd}/3 \rfloor + 1$ if so it transfers $\mathcal{A}$ guess b^* to its challenger, otherwise $\mathcal{D}$ executes the TxBrodOrd algorithm, sends blob to $\mathcal{A}$ and forward the adversary's guess b^* to its challenger. Hence, $\mathcal{D}$ has the same probability of winning against the anonymity of the signature as does have $\mathcal{A}$ of winning against pseudonymity. Finally, if $\mathcal{A}$ has a non-negligible advantage so does $\mathcal{D}$, this concludes the proof with the following: $\mathsf{Adv}^{pseudo}_{\mathcal{BC},\mathcal{A}}(\mathfrak{K}) = \mathsf{Adv}^{f-anon}_{\mathsf{GS},\mathcal{D}}(\mathfrak{K})$.

Pseudonymity of RingCons, LinkGroupCons and LinkRingCons

The proof is similar to the case of group signatures. Here we rely on anonymity of the ring signature instead of the anonymity of the group signature. Which is fairly similar.

The above arguments show security of the protocols even if instead of group or ring signature their linkable versions are used. Hence, we have already covered the proofs for LinkGroupCons and LinkRingCons, and thus the five cases.

C.3 Security Proof of the Pseudonymity of the endorsers and the Orderers

In this section we prove Theorem 7.

Theorem 7. *Base on a secure linkable group (resp. ring) signature the EndGroupCons, (resp. EndRingCons) protocol is pseudonymous for the endorsers. Under the same conditions, the protocols OrdGroupCons and OrdRingCons are pseudonymous for the orderers (excluding the orderer leader).*

Proof. We devide this proof into two steps.

Pseudonymity of EndGroupCons and EndRingCons

We must prove that it is impossible for an adversary to win Experiment 4 with non-negligible probability for EndGroupCons instantiated with a secure linkable group signature. As the full-anonymity property is the same so is the reduction for EndRingCons with has a linkable ring signature instead.

Let Game 0 be the previously mentioned experiment. Assume there exists an adversary $\mathcal{A}$ able to establish with non-negligible probability an execution of Game 0 with $b = 0$ and $b = 1$. We are building a distinguisher $\mathcal{D}$ simulating Game 0 with the adversary $\mathcal{A}$ and thus solving the anonymity of the linkable group signature GS with comparable probability that $\mathcal{A}$ has to win this game. This would be in direct contradiction with assumed anonymity of our linkable group signature GS, hence showing that we achieve the claimed security.

Assuming that $\mathcal{A}$ has a non-negligible advantage against Game 0. $\mathcal{D}$ starts the anonymity experiment of the signature scheme and gain access to its oracles after generation of the group manager public key gpk. $\mathcal{D}$ generates the elements o_{bc}, Payload, SecPolicies. $\mathcal{D}$ pick two public keys pk_0 and pk_1 associated to two group members' index i_0 and i_1. While it sends this two index to its challenger, it also sends the two public keys alongside the SecPolicies to $\mathcal{A}$. The adversary answers with $(\mathsf{trans}_{\mathsf{prop}}, \sigma), \mathsf{pk}_{c_{bc}}$. $\mathcal{D}$ executes the TxEndors algorithm with these values and by asking its challenger to obtain the signatures it requires on $\mathsf{trans}_{\mathsf{prop}}$. It sends the result of the execution to $\mathcal{A}$ who answers by a bit b^*. $\mathcal{D}$ forward it to the fully-anonymity's challenger.

If $\mathcal{A}$ has non-negligible chances to distinguish Game 0 this allows distinguishing between the two players of the full-anonymity experiment of the linkable group signature. We conclude that, $\mathsf{Adv}_{\mathcal{BC},\mathcal{A}}^{pseudo_{Endor}}(\mathfrak{K}) \leq \mathsf{Adv}_{\mathsf{GS},\mathcal{D}}^{f-anon}(\mathfrak{K})$.

The reduction for the linkable ring signature is similar and leads to the same bound on the adversary's advantage.

Pseudonymity of **OrdGroupCons** and **OrdRingCons**

Game 0 is the Experiment 5. In this experiment after the generation of the keys, $\mathcal{A}$ should generate, approve and send a commitment message following the template given to the orderers leader. Then the challenger is simulating for one out of two users the preparation and validation of the block. And latter broadcast of the block commitment after receiving the validation of the block from a majority of the orderers. Finally, $\mathcal{A}$ must guess the orderer that was involved in the transaction (*i.e.,* its public key).

Game 1. The secret key $\mathsf{sk}_{(1-b)}$ is used to prepare the block (*i.e.,* execute the Prepare algorithm) instead of the prescribed key sk_b. The Prepare algorithm consist of four steps of verifications before producing a signature on the NewBlock if all of them passed. This signature in our case is a linkable group/ring signature $\mathcal{LS}$, and we have taken as granted its anonymity. It must be impossible to distinguish the signer who produced the signature. This step introduced only a negligible difference with the previous one: $\mathsf{Adv}_{\mathcal{BC},\mathcal{A}}^{\mathsf{G_0}}(\mathfrak{K}) \leq \mathsf{Adv}_{\mathcal{LS},\mathcal{D}}^{f-anon}(\mathfrak{K}) + \mathsf{Adv}_{\mathcal{BC},\mathcal{A}}^{\mathsf{G_1}}(\mathfrak{K})$, where $\mathcal{D}$ is the distinguisher in the reduction.

We were able to change the value of b used in the game to $(1-b)$ with only a negligible chance that $\mathcal{A}$ could notice it. Thus, unlinkability of the orderer holds.

C.4 Security Proof of the Anonymity of the users

We now prove Theorem 8.

Theorem 8. *Given that the client proceeds to a secure blind, ring or group signature to authenticate its transaction as defined in Section 4, anonymity of the client holds.*

Proof. The experiment $\mathsf{Exp}_{\mathcal{BC},\mathcal{A}}^{Anony}$ models the anonymity of a user against an adversary $\mathcal{A}$ describes a challenger simulating two transaction with different Payload and then a third one linked to one of the first transactions. If the adversary can infer a link between two transaction, it retrieves the value b. Otherwise, if $\mathcal{A}$ can produce a link between a transaction tx and the associated client identify by its public key pk, then it can also find out the value b. Thus, it covers the two requirements for anonymity. This experiment actually models the execution of a client, even if the challenger is not executing the TxBrodOrd algorithm. The latest only invokes public keys or data coming from the corrupted endorser controlled by $\mathcal{A}$. Hence, the adversary is capable to fully simulate these executions of TxBrodOrd that $\mathcal{C}$ could have made.

We have elaborated why our definition covers anonymity of our protocol, we can now show that for all probabilistic polynomial time adversary $\mathcal{A}$ can break this property without a negligible advantage. Consider the experiment $\mathsf{Exp}_{\mathcal{BC},\mathcal{A}}^{Anony}$ as Game 0.

For the following we need to assume that the elements o_{bc} and Payload does not leak information on the authors of the transactions. Then assuming that $\mathcal{A}$ has a non-negligible advantage against Game 0, we build upon it an adversary $\mathcal{B}$ breaking the anonymity of the signature. $\mathcal{B}$ starts the anonymity experiment of the signature scheme and gain access to its oracles after generation of the group manager public key gpk. $\mathcal{B}$ produces the elements $\{o'_{bc,b}, \mathsf{Payload}'_b\}_{b \in \{0,1\}}$. It executes the TxProp algorithm for each of them and generates two elements $\mathsf{trans}_{\mathsf{prop},0}$ and $\mathsf{trans}_{\mathsf{prop},1}$. For each of them it calls the signing oracle respective identities id_0 and id_1 picked randomly out of the group of participants $\mathcal{G}$. Once it has received the signatures it finishes the execution of the TxProp algorithm for both and sends the result to $\mathcal{A}$. Using TxProp, $\mathcal{B}$ produce a new $\mathsf{trans}_{\mathsf{prop}'}$ for freshly generated $o_{bc}', \mathsf{Payload}'$ and sends $(\mathsf{trans}_{\mathsf{prop}'}, \mathsf{id}_0, \mathsf{id}_1)$ to its challenger which returns a signature σ'. $\mathcal{B}$ finishes the generation of the new transaction with it and sends it to $\mathcal{A}$ which send back a bit b directly forwarded by $\mathcal{B}$ to the challenger. $\mathsf{Adv}_{\mathcal{BC},\mathcal{A}}^{anony}(\mathfrak{K}) = \mathsf{Adv}_{\mathsf{S},\mathcal{B}}^{f-anon}(\mathfrak{K})$.